Leveraging Generative Models for Music and Signal Processing

John Thickstun

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2021

Reading Committee:

Sham M. Kakade, Chair

Zaid Harchaoui, Chair

Sewoong Oh

Noah A. Smith

Program Authorized to Offer Degree:
Computer Science & Engineering

University of Washington

Abstract

Leveraging Generative Models for Music and Signal Processing

John Thickstun

Co-Chairs of the Supervisory Committee:
Associate Professor Sham M. Kakade
Paul G. Allen School of Computer Science & Engineering, Department of Statistics

Associate Professor Zaid Harchaoui
Department of Statistics

Generative models can serve as a powerful primitive for creative interaction with data. Generative models give us the ability to synthesize or re-synthesize multimedia; conditional generative modeling empowers us to control the outputs of these models. By steering a generative model with conditioning information, we can sketch the essential aspects of our creative vision, and the generative model will fill in the details. This dissertation explores the possibilities of generative modeling as a creative tool, with a focus on applications to music and audio.

The dissertation proceeds in three parts:

1. We develop algorithms and evaluation metrics for aligning musical scores to audio. Alignments provide us with a dense set of labels on musical audio, that we can use to supervise conditional generation tasks such as transcription: synthesis of a musical score conditioned on an audio performance. This work on alignments leads to the construction of MusicNet: a collection of 330 freely-licensed classical music recordings, together with over 1 million annotated labels indicating the precise time of each note in every recording, the instrument that plays each note. We use this dataset to train state-

of-the-art music transcription models for the MIREX Multiple Fundamental Frequency Estimation task.

2. We construct autoregressive generative models of musical scores, which exploit invariances in the structure of music. Whereas most recent work on music modeling has represented music as an ordered sequence of notes, we explore an alternative representation of music as a multi-dimensional tensor. We consider a variety of factorizations of the joint distribution over these tensors. We then turn to our attention to discriminative modeling of scores, using this tensor representation. We construct a classifier that can reliably identify the composer of a classical musical score. Our methods, which operate on the generic tensor score representation, outperform previously reported results using SVM and kNN classifiers with handcrafted features, specialized for the composer classification task.

3. We develop a sampling algorithm for likelihood-based models that allows us to steer an unconditional generative model using conditioning information. We work within a Bayesian posterior sampling framework, using a pre-trained unconditional generative model as a prior, to sample from the posterior distribution of a conditional likelihood. Samples are obtained using noise-annealed Langevin dynamics to construct a Markov chain for approximating samples from this posterior distribution. We develop these ideas for a variety of models and applications, including source separation, in both the visual and audio domains.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I thank my academic advisors, Sham M. Kakade and Zaid Harchaoui, who have influenced every aspect of my growth and development as a researcher. Thank you both for your mentorship, your patience, and your belief in me. I thank the other members of my committee: Sewoong Oh, Noah Smith, and Lalit Jain. You have been so generous with your time, your feedback on this document and your career advice. At the emotional low-point of my PhD, I spent a summer at Amazon Music working with Ted Sandler and Ben London, that revitalized my enthusiasm for research. Thank you for accepting me as an intern and for your mentorship that summer. And I thank our graduate school advisor Elise Dorough, who has helped me—and so many students—to navigate the experience and the institution of graduate school.

I thank my labmates for their friendship and support. Rahul Kidambi and Corinne Jones, who I have known since my first year of graduate school. And later, as Sham's and Zaid's labs grew: Krishna Pillutla, Alec Greaves-Tunnell, Aravind Rajeswaran, Ramya Vinayak, Weihao Kong, Vincent Roulet, Lang Liu, and Aditya Kusupati. Our graduate school environment is so strongly influenced by our advisors that often labmates are the only people who can fully understand and share our mutual experiences. I want to single out Krishna in particular, with whom I share both my advisors, and who has been a constant and unwavering source of support and encouragement.

I was fortunate to share office 418 with wonderful friends during my first several years of graduate school: Colin Lockard, Robbie Weber, Chris Xie, and Xin Yang. We've all graduated now, 6 years after that office group first formed, and I think the camaraderie of the office has contributed a lot to our successes. Later on, I shared a work space with

many more friends in the ML lab: Jennifer Brennan, Ian Covert, Sam Ainsworth, Andrew Wagenmaker, Rahul Nadkarni, Romain Camilleri, among others. It is a privilege to be able to work alongside such a sociable, intellectually stimulating, and thoughtful group of people.

I thank my undergraduate advisors: Eugene Charniak, Björn Sandstede, and Basilis Gidas. Eugene, for the generous time he spent mentoring me as an undergraduate researcher and fostering my interest in the intersection of music and machine learning. Björn, for encouraging my mathematical development and broad interests in mathematics and computing. And I thank each of them for encouraging me to pursue a PhD, when I was very uncertain about the decision: I recall Basilis in particular scolding me for even considering abandoning my work on music. I am so glad they pushed my to apply to graduate schools.

I thank all my co-authors (in chronological order): Sham M. Kakade, Zaid Harchaoui, Dean P. Foster, Harsh Verma, Vivek Jayaram, Bhargavi Paranjape, Mandar Joshi, Hannaneh Hajishirzi, Luke Zettlemoyer, Jennifer Brennan, Samuel Ainsworth, Kendall Lowrey, Siddhartha Srinivasa, Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, Yejin Choi, Tyler Benster, Darwin Babino, Matthew Hunt, Xiyang Liu, Sewoong Oh, and Russell N. Van Gelder. I have learned so much from working with you.

I thank my fiancée, Leah Perlmutter, who has been an unending source of emotional support and understanding throughout the all-consuming experience that is graduate school. I thank my parents, who have always supported me and given me the confidence to take risks as I work to build a life and career for myself. And finally I thank my childhood cello teacher, Suzanne Beevers. Beyond concrete lessons on musicianship, Sue taught me how to *practice*, to be simultaneously *self-critical* and *self-assured* of my work, to work toward a goal on the far horizon. Sue's lessons have deeply influenced this dissertation, from my choice of topic to the skills I relied on daily to do the work.

DEDICATION

To my cello teacher, Suzanne Beevers.

Chapter 1

INTRODUCTION

This dissertation explores probabilistic generative modeling as a paradigm for analysis and synthesis of music. This work is motivated by the potential of these models as tools for artists and creators. As generative modeling techniques mature, we are beginning to see applications of these models to facilitate the creative process in various mediums: visual editing [Xiong et al., 2019], creative writing [Clark et al., 2018] and music composition [Donahue et al., 2019b]. The foci of this dissertation are the music and audio domains. We consider applications of generative models to music and signal processing problems, and how to adapt generative modeling techniques to the specific structures of musical media. We focus in particular on classical western music. This choice of genre is in part pragmatic: classical music is relatively unencumbered by licensing restrictions, which facilitates open research in this genre.

Classical music is expressed through an unusually tightly-coupled constellation of modalities: the visual medium of musical scores, acoustic musical performances of these scores, and symbolic digital representations of music that can represent alternately (i) a digitization of a visual score (ii) a transcription of a musical performance with expressive timings or (iii) some artifact occupying a gray area between a performance and score. In addition to the generative modeling questions intrinsic to each modality, these modes present us with sequence-to-sequence modeling questions analogous to translation. This coupling of visual, symbolic, and acoustic media make classical music a potential Rosetta-stone for generative modeling of music.

A generative model is a tool that we can use to synthesize musical structures. Before we immerse ourselves in the technical details of generative modeling, we should acknowledge

that music is *art*, which is inherently tied to the human experience. Algorithms and models cannot generate art, which is inextricably connected to questions of provenance, interpretation, and meaning. An algorithmic creation lacks the context and purpose of an artistic creation. Nevertheless, generative models can serve as a tool to assist the artistic process. We use generative models as a tool for analysis in Chapter 3, constructing models that generate a symbolic description of the content of a musical audio recording. We use generative models as an unconstrained tool for synthesizing symbolic structures with musical form in Chapter 4. We demonstrate how unconstrained generative models can be steered towards desired outputs in Chapter 5. This ability to steer a generative model provides control and agency to the artist: we can imagine a generative model as a creative collaborator, with impeccable knowledge of its craft, that will diligently defer to our artistic vision.

In Chapter 2 we describe two fundamental digital encodings of music that we are interested in modeling: symbolic piano-rolls, and discretized audio waves. The remainder of this chapter is dedicated to a review of generative modeling, and applications of these techniques to the music domain. We focus in particular on *likelihood-based* generative models that explicitly parameterize a probability density over musical structures. Our interest in likelihood-based models is motivated by the methods developed in Chapter 5, which rely on (gradients of) a log-likelihood function to steer the conditional generation process.

In Chapter 3 we investigate music transcription: the translation an audio recording into a corresponding musical score. This translation is insightful because it provides us with a symbolic, dense semantic labeling of an audio recording. The utility of these labels has earned music transcription recognition as a "key enabling technology" for music signal processing [Benetos et al., 2013, 2019]; the combined importance and difficulty of this problem have earned it a reputation as a "holy grail" of the music signal processing field [Benetos et al., 2012, Kim, 2020]. We can frame music transcription as a conditional generative modeling task: generate a likely score (or the *most* likely score) associated with a given audio recording. We will discuss a prevalent *frame-based* simplification of the music transcription task, which motivates an analysis of alignment algorithms that we can use to construct labels for frame-

based transcription.

The work presented in Chapter 3 contributes to both the audio-to-score alignment literature and to the music transcription literature. We introduce a formal definition of a temporal audio-to-score alignment, and contrast this definition with the related concept of a note-based alignment. We introduce evaluation metrics for analyzing the performance of a temporal alignment algorithm, construct a dataset of ground truth alignments upon which these metrics can be computed, and evaluate several popular temporal alignment algorithms. We go on to use an alignment algorithm to construct the MusicNet dataset, a collection of freely-licensed chamber music recordings aligned to corresponding musical scores. We use MusicNet to train supervised transcription models that achieve state-of-the-art frame-based transcription results for the popular Su benchmark dataset [Su and Yang, 2015b]. Finally, we analyze the learned low-level features of end-to-end transcription models and draw analogies between these features and classical filterbank representations of audio.

In Chapter 4 we investigate symbolic generative and discriminative models of musical scores. This necessitates a more nuanced discussion of symbolic representations of music, beyond the simple piano-rolls considered in Chapters 2 and 3. We develop (unsupervised) generative models of musical scores, and (supervised) discriminative classification models for attributing composers to musical scores. The generative modeling problem for musical scores is "almost as old as that of computers" [Herremans et al., 2017], and is intertwined with questions of computational creativity and machine intelligence [Conklin, 2003]. The discriminative composer attribution problem is arguably less well-motivated: most scores are distributed with composer metadata. Where this metadata is absent, composer attribution is a musicological question and the value of a black-box classifier's prediction is debatable. Instead, our interest in these discriminative models is motivated by their potential use as *likelihoods* for steering generative models using the techniques developed in Chapter 5.

The work presented in Chapter 4 makes contributions to the modeling and evaluation of symbolic music scores. We define several autoregressive factorizations of a musical score, drawing connections between these factorizations, traditional piano-roll factorizations, and

more recently proposed sequential factorizations. We define a cross-entropy metric for generative models of musical scores, which is independent of any choice of factorization. We construct neural architectures tailored to the structure of a musical scores, and evaluate a variety architectural choices for generative modeling of scores under this cross-entropy metric. Finally, we develop state-of-the-art discriminative composer classification models trained on a corpus of just 2,500 scores, demonstrating the effectiveness of these end-to-end neural models in this small-data regime.

In Chapter 5 we investigate conditional sampling, using a likelihood function (such as a classifier) to steer the outputs of a generative model towards samples with specified criteria or constraints. We develop algorithms for sampling from the posterior distribution of our model (in the Bayesian sense) for a given likelihood. We are particularly interested in source separation, also known as the *cocktail party problem* [Cherry, 1953, Haykin and Chen, 2005], which involves decomposing an audio recording into an additive mixture of sources. Like transcription, we can frame source separation as a conditional generative modeling problem: generate a collection of audio recordings, subject to the constraint that these recordings sum to a given mixture. Music transcription and source separation have been juxtaposed as complementary foundational questions in music signal processing [Plumbley et al., 2002].

The work presented in Chapter 5 makes methodological contributions at the intersection of Markov-chain Monte Carlo methods and deep learning, as well as empirical contributions to the source separation literature. Methodologically, we introduce a procedure for smoothing discretized autoregressive models for use in conjunction with Markov-chain Monte Carlo techniques based on Langevin dynamics. Empirically, we extend the annealed Langevin sampling procedure introduced by Song and Ermon [2019] to posterior sampling problems, and in particular to the class of linear inverse problems which includes source separation. This unsupervised approach to source separation expands upon the Bayesian source separation literature [Benaroya et al., 2006], using deep generative models as Bayesian priors over sources. Our unsupervised method achieves state-of-the-art results for visual source separation, and competitive results for audio source separation compared to supervised approaches.

1.1 Publications and Authorship

The work in this dissertation is the result of several research collaborations; prior publications of work contained in this thesis are discussed below, organized by the chapter in which this work appears. All of the writing in this dissertation that is borrowed from these publications is my own.

Chapter 3. I am the primary author of all work presented in this chapter.

- Learning Features of Music from Scratch. John Thickstun, Zaid Harchaoui, and Sham M. Kakade. Published in International Conference on Learning Representations (ICLR) 2017.

- Frequency Domain Convolutions for Multiple F0 Estimation. John Thickstun, Zaid Harchaoui, Dean P. Foster, and Sham M. Kakade. Technical Report (MIREX) 2017.

- Invariances and Data Augmentation for Supervised Music Transcription. John Thickstun, Zaid Harchaoui, Dean P. Foster, and Sham M. Kakade. Published in International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2018.

- Rethinking Evaluation Methodology for Audio-to-Score Alignment. John Thickstun, Jennifer Brennan, and Harsh Verma. Preprint Report (ArXiv) 2020.

Chapter 4. I am the primary author of the work on coupled recurrent models. I had a supervisory role in the work on composer classification, in collaboration with Harsh Verma, who was an undergraduate student at that time at the University of Washington. The experimental work on composer classification was conducted by Harsh.

- Coupled Recurrent Models for Polyphonic Music Composition. John Thickstun, Zaid Harchaoui, Dean P. Foster, and Sham M. Kakade. Published in International Symposium on Music Information Retrieval (ISMIR) 2019.

- Convolutional Composer Classification. Harsh Verma and John Thickstun. Published in International Symposium on Music Information Retrieval (ISMIR) 2019.

Chapter 5. The work presented in this chapter is the result of a close collaboration with Vivek Jayaram. We are equal co-authors of both of these papers, with shared responsibility for the problem formulation, methodological contributions, and experimental work.

- Source Separation with Deep Generative Priors. Vivek Jayaram and John Thickstun. Published in International Conference on Machine Learning (ICML) 2020.

- Parallel and Flexible Sampling from Autoregressive Models via Langevin Dynamics. Vivek Jayaram and John Thickstun. Published in International Conference on Machine Learning (ICML) 2021.

Chapter 2

GENERATIVE MODELS OF MUSIC

This chapter provides background on generative modeling techniques. It is not intended as a complete survey of generative modeling, but rather as an introduction to the models used throughout this dissertation and—more broadly—applications of generative modeling techniques in music research. We begin in Section 2.1 by formalizing definitions of two of the basic objects of interest in the music domain. In Section 2.2 we introduce the premise of generative modeling and discuss some of the challenges in constructing and evaluating these models. In Section 2.3 we review autoregressive models, a ubiquitous approach to modeling sequential data that we use extensively in Chapters 4 and 5. In Section 2.4 we review variational auto-encoders, and in Section 2.5 we review generative adversarial networks. Unlike the rest of the models described in this chapter, variational autoencoders and generative adversarial networks are not put to use later in this dissertation, and we discuss the obstruction to combining these models with the sampling techniques developed in Chapter 5. In Section 2.6 we review flow-based networks, which feature prominently in Chapter 5. And finally, in Section 2.7 we review energy based models; the sampling techniques used by these models directly inform the algorithms introduced in Chapter 5.

2.1 Basic Definitions

One standard and generic way to represent symbolic music is a *piano-roll*, which we formally define in Definition 2.1. A piano-roll is a binary matrix that encodes the presence or absence of a musical event (e.g., a pitch class) at a given time index. For example, we can encode solo piano music using a vocabulary of $N = 88$ pitch classes, corresponding to the keys on a standard piano; the full Western semi-tone scale can be represented using $N = 128$, which

is the convention adopted by MIDI standard [International MIDI Association, 1983].

Definition 2.1. *A piano-roll of length T is a discrete-valued function* **pianoroll** $: [0, T) \to \{0, 1\}^N$.

If we discretize time T into sub-divisions of length dt, then the matrix $M \in \{0, 1\}^{N \times (T/dt)}$ defined by $M_{n,k} = \textbf{pianoroll}(kdt)_n$ is the standard, discrete-time version of a piano-roll.

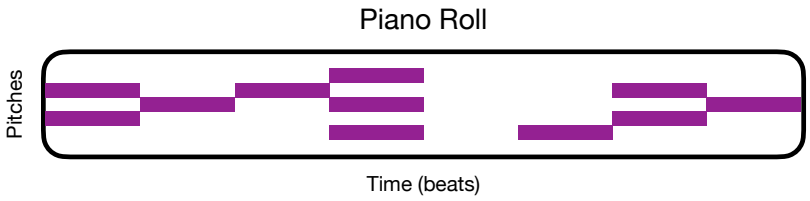


Figure 2.1: A piano-roll representation of symbolic music.

We can encode a musical score as a piano-roll, denoted by **score** $: [0, S) \to \{0, 1\}^N$, defined by the rule $\textbf{score}(s)_n = 1$ iff note n is indicated by the score at position $s \in [0, S]$. While this encoding can be convenient, it is important not to identify piano-rolls with scores. A piano-roll discards substantial information about the score, which can be difficult to reconstruct post-hoc (the encoding of scores as piano-rolls is not injective). More subtly, most piano-rolls do not correspond to any score (the encoding is not surjective). We refer the reader to the concepts of a **pscore** and a **symbp**, introduced in Section 3.1.2, for examples of piano-rolls that have very different structure than a score, but can also be encoded by piano-rolls. We discuss more sophisticated encodings of scores at length in Section 4.1.

Given a tempo marking, e.g., 120 beats per minute, we can convert between time in a score and time in an ideal performance of that score. A human performer will not play at the exact given tempo, nor will they hold a constant tempo throughout their performance. Constructing an accurate alignment of time in a score with time in an expressive human performance of that score is discussed in Section 3.1. As a convention, we measure time in a score with units of beats; a score defined on the interval $[0, S)$ is S beats long. In contrast, we measure time in a performance in units of seconds; a performance on the interval $[0, T)$ is T seconds long.

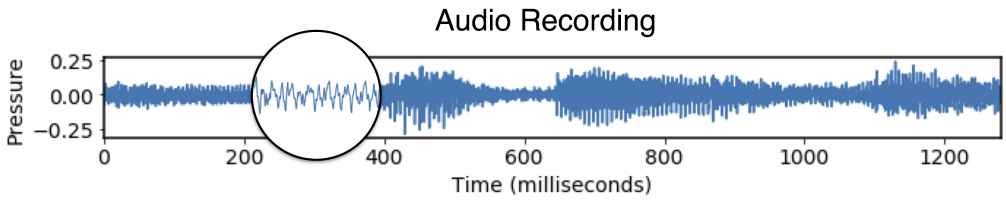


Figure 2.2: A digital audio recording of a musical performance.

An acoustic musical performance is transmitted to the human ear in the form a non-stationary audio wave. By convention, we normalize the global amplitude of this wave to the interval $(-1, 1)$.

Definition 2.2. *An audio wave of length of T is a real-valued function* **audio** $: [0, T) \to (-1, 1)$.

A performance be can recorded by a microphone, and represented digitally by a sequence of recorded real-valued air pressure readings sampled at discrete time steps. The Nyquist-Shannon sampling theorem guarantees that frequency content below f hertz is completely determined by samples taken at a constant frequency $2f$ hertz [Lüke, 1999]. The limit of human acoustic frequency perception is bounded above by approximately $f = 20$kHz [Rossing, 2007], so the perceptible content of an acoustic performance is fully captured using a sampling rate greater than 40kHz. Real pressure values can be be stored as floating point values, or quantized using integer values; a sequence of quantized pressure readings sampled at a constant rate is known as pulse-code modulation (PCM) [Black and Edson, 1947]. For example, Standard CD-quality audio is stored using the PCM format with linearly quantized 16-bit values sampled at 44.1kHz, resulting in a vector representation $X \in \mathbb{Z}^{44,100 \times T}$ for an audio performance of length T where $X_t = \lfloor 2^{15} \times \textbf{audio}(t/44,1000) \rfloor$. In contrast to scores and piano-rolls, acoustic performances and their discretized vector encodings are approximately isomorphic.

2.2 Generative Modeling

Generative modeling asks the following question: how can we (approximately) sample from an unknown probability distribution p over a space $\mathcal{X}$, given independent observations $\mathbf{x}_1, \ldots, \mathbf{x}_n \sim p$? An answer to this question consists of procedure that, when followed, generates samples $\mathbf{x} \sim \hat{p}$ where $\hat{p} \approx p$. Throughout this work, we will presume access to samples from simple distributions such as Uniform$(0, 1)$. We can construct more rich random variables given a primitive source of uniform randomness using a pushforward construction.

Definition 2.3. *Given a probability space $(\mathcal{Z}, q)$, a (measurable) function $g : \mathcal{Z} \to \mathcal{X}$ induces a **pushforward** distribution on $\mathcal{X}$ defined, for any (measurable) set $A \subset \mathcal{X}$ by*

$$\Pr(A) = \int_{g^{-1}(A)} q(\boldsymbol{z})\, d\boldsymbol{z}. \tag{2.1}$$

For example, suppose we want to sample from a biased coin, i.e., $x \sim$ Bernoulli(p). Defining $g(z) = \mathbf{1}_{z<p}$ with $z \sim$ Uniform$(0, 1)$ induces the desired pushforward distribution $g(z) \sim$ Bernoulli(p). A more interesting example is the Box-Muller transform [Box and Muller, 1958], which induces a $\mathcal{N}(0, 1)$ pushforward when applied to inputs $z \sim$ Uniform$(0, 1)$.

Generating samples $\mathbf{x} \sim p$ using a pushforward of a primitive source of randomness is sometimes called simulation of p in the statistics community. In the machine learning community, the function $g : \mathcal{Z} \to \mathcal{X}$ is often called a generator and constructing such a function is called generative modeling. Formally, a generative model is a function (a generator) $g : \mathcal{Z} \to \mathcal{X}$ that maps uniform random variables $\mathbf{z} \sim$ Uniform$(0, 1)^d$ to structured ouputs $\mathbf{x} = g(\mathbf{z}) \sim \hat{p}$ such that $\hat{p} \approx p$.

Definition 2.4. *Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ be independent, identically distributed samples from p. A **generative model** of observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is a function (a generator) $g : (0, 1)^d \to \mathcal{X}$ that induces a pushforward distribution $g(\mathbf{z}) \sim \hat{p}$ such that $\hat{p} \approx p$.*

A good generative model is one that minimizes error in the approximation $\hat{p} \approx p$. How do we measure this error? Evaluation of generative models is a difficult problem that we will

revisit throughout this work. To talk about approximations, we must put a topology on the space of probability measures. By far the most popular topology on probability distributions is the topology of KL divergence, and one natural measure of the approximation error of a generative model is $D(p \parallel \hat{p})$. This choice is also convenient because, while we cannot analytically compute this information divergence—we do not know p—we can compute a Monte Carlo estimate of the related cross-entropy from samples.

Definition 2.5. *The cross-entropy of a distribution $\hat{p}$ relative to p is given by*

$$H(p, \hat{p}) \equiv \mathop{\mathbb{E}}_{\mathbf{x} \sim p} - \log \hat{p}(\mathbf{x}). \tag{2.2}$$

If $H(p)$ denotes the (unknown) entropy of p then it is easy to show that $H(p, \hat{p}) = H(p) + D(p \parallel \hat{p})$ [Cover and Thomas, 2001]. The cross entropy is a useful information functional because it can be estimated from samples using a straightforward Monte Carlo estimator:

$$H(p, \hat{p}) = -\int_{\mathcal{X}} p(\mathbf{x}) \log \hat{p}(\mathbf{x}) \, d\mathbf{x} \approx -\frac{1}{n} \sum_{i=1}^{n} \log \hat{p}(\mathbf{x}_i), \text{ where } \mathbf{x}_i \sim p. \tag{2.3}$$

This approach to evaluation underlies the cross-entropy metric developed in Chapter 4. However, it requires the construction of an explicit density estimator $\hat{p}(\mathbf{x}_i)$ which, as we will see next, not all generative models provide. Because of this complication, we take a different approach to evaluating generative models in Chapter 5.

Some of the methods considered in this work do not fit the density estimation framework. Recall that our goal is to sample from $\mathbf{x} \sim \hat{p}$. A density estimator provides us with values $\hat{p}(\mathbf{x})$. It is usually a mechanical exercise to construct a generator to simulate samples from a given density $\hat{p}(\mathbf{x})$. But if all we want is a sample, construction of a density estimate $\hat{p}(\mathbf{x})$ may not be necessary. In Section 2.5 (generative adversarial networks) and Section 2.7 (energy-based models) we will see generative models that do not explicitly estimate $\hat{p}(\mathbf{x})$. These models define a generator of the pushforward distribution $\hat{p}$, but inferring $\hat{p}(\mathbf{x})$ for a

particular value of $\mathbf{x}$ may be quite difficult.

To see why this could be, suppose $\hat{p}$ is defined implicitly as the pushforward of a density q by an invertible generator $g : \mathcal{Z} \to \mathcal{X}$. Changing variables from $\mathbf{z}$ to $\mathbf{x}$, we see that

$$\Pr(A) = \Pr(g^{-1}(A)) = \int_{g^{-1}(A)} q(\mathbf{z})\, d\mathbf{z} = \int_A q(g^{-1}(\mathbf{x}))|\nabla_x g^{-1}(\mathbf{x})|\, d\mathbf{x}. \tag{2.4}$$

Therefore, the density $q(\mathbf{z})$ pushes forward to

$$\hat{p}(\mathbf{x}) = q(g^{-1}(\mathbf{x}))|\nabla_x g^{-1}(\mathbf{x})|. \tag{2.5}$$

If the inverse and Jacobian of g are easily computable, then we can use Equation (2.5) to convert a generator into a density estimator. But we are interested in learning rich distributions over highly structured data. A generator g that accurately models this distribution may not have an easily computable inverse or Jacobian. Nevertheless, the pushforward distribution $\hat{p}$ induced by g could be a very good estimate of p. And it is easy to generate samples $\mathbf{x} = g(\mathbf{z}) \sim \hat{p}$ given samples $\mathbf{z} \sim q$.

In the remainder of this chapter we survey of various answers to these modeling challenges. One option is construct an explicit density estimator, which usually involves factoring a high-dimensional joint distribution into smaller conditional distributions with manageable partition functions; this approach is taken by the autoregressive models [Larochelle and Murray, 2011] discussed in Section 2.3. Another approach is to construct an estimate of the objective, e.g., Equation (2.17), and optimize with respect to this proxy estimate; two very different approximation methods are the variational autoencoders [Kingma and Welling, 2014] and generative adversarial networks [Goodfellow et al., 2014] described in Sections 2.4 and 2.5 respectively. Alternatively, we can construct explicit, high-dimensional density estimators using restricted parameterized function families $\{g_\theta, \theta \in \Theta\}$ that support efficiently and exactly computable inverses and Jacobians. This approach is taken by the flow-based models [Dinh et al., 2017] that we discuss in Section 2.6. This approach trades of expressiv-

ity in the parameterization of the model for computational tractability. Another option is to try to conquer the challenge of computing inverses and Jacobians for more general function families; this approach is less well-developed, but is partially addressed by Hand and Voroninski [2020], Ma et al. [2018]. Finally, in Section 2.7 we consider some several variants of the energy-based modeling paradigm [Hinton, 2002, LeCun et al., 2006], which model un-normalized energy functions rather than probability densities and rely on Markov-chain Monte Carlo generation procedures to construct samples [Song and Ermon, 2019].

2.3 Autoregressive Sequence Models

In this section, we consider structured data of the form $\mathbf{x} \in \mathbb{R}^{T \times d}$, which we will interpret as a sequence of d-dimensional vectors x_t indexed by a discrete temporal value $0 \leq t < T$. Examples of sequential data include both discretized piano rolls (Definition 2.1) and sampled audio waves (Definition 2.2). Our goal is to construct a generative model of an unknown probability distribution p, based on observed sequences $\mathbf{x}^1, \ldots, \mathbf{x}^n \sim p$. For time series, we will index sequences with a superscript to distinguish this index from the temporal index. In this section, we consider autoregressive models as a tool for generative modeling. Density estimation using autoregressive models is a ubiquitous modern approach to generative modeling in the music community. These models largely follow the neural autoregressive distribution estimation paradigm (NADE) [Larochelle and Murray, 2011] discussed below. Autoregressive models in the audio domain include WaveNet [van den Oord et al., 2016a], SampleRNN [Mehri et al., 2017], and Jukebox [Dhariwal et al., 2020]. Autoregressive models of symbolic music include the BachBot [Liang et al., 2017] and Music Transformer [Huang et al., 2019], as well as the models constructed in Chapter 4 of this dissertation. Methodology for smoothing discretized autoregressive models is developed in Chapter 5, which allows these models to be combined with Langevin sampling techniques (Section 2.7).

An autoregressive model parameterizes the conditional distributions $p_\theta(x_t|\mathbf{x}_{<t})$ over values x_t given preceding values $\mathbf{x}_{<t} = \{x_0, \ldots, x_{t-1}\}$, with parameters θ. The chain rule for probabilities allows us to factor the joint distribution over sequences $\mathbf{x}$ into conditional

distributions over items in the sequence:

$$p_\theta(\mathbf{x}) = \prod_{t=1}^{T} p_{\theta,t}(x_t|\mathbf{x}_{<t}). \tag{2.6}$$

Given a collection of conditional estimates $p_{\theta,t}(x_t|\mathbf{x}_{<t})$, we can sample from the model $p_\theta(\mathbf{x})$ by iteratively sampling $\hat{x}_t \sim \hat{p}_{\theta,t}(\cdot|\hat{\mathbf{x}}_{<t})$; which we will refer to as the *ancestral sampler*. An autoregressive model is a density estimator, in the sense that assigns a normalized probability $p_\theta(\mathbf{x})$ to any given sequence $\mathbf{x}$. Because we can efficiently compute probability densities under the model, we can train an autoregressive models using maximum likelihood estimation and evaluate a trained model using empirical cross-entropy.

We can parameterize an autoregressive model over discrete values x_t using a softmax normalization $p_{\theta,t}(x_t|\mathbf{x}_{<t}) = \text{softmax}(f_{\theta,t}(\mathbf{x}_{<t}))$, where the function $f_{\theta,t} : \mathbb{R}^{(t-1)d} \to \mathbb{R}^d$ is defined by a neural network with weights θ. This possibility was observed by Bengio and Bengio [1999], applied to language modeling by Bengio et al. [2003], and further developed into its modern incarnation by Larochelle and Murray [2011]. Neural parameterization can be extended to model continuous time-series, using the outputs $f_{\theta,t}$ to parameterize a continuous distribution over the conditionals $p(\mathbf{x}_t|\mathbf{x}_{<t})$, instead of a softmax. This idea is developed under the name real-valued NADE (RNADE) [Uria et al., 2013]. Deep neural parameterizations of NADE are developed in detail by Uria et al. [2016]. Both Uria et al. [2016] and van den Oord et al. [2016b] develop the idea of a convolutional NADE, making use of the convolutional neural network architecture [LeCun et al., 1989]. The modern, efficient implementation of NADE using input masks was introduced by [Germain et al., 2015], and is often combined with a Transformer neural network parameterization [Vaswani et al., 2017].

2.4 Variational Autoencoders

In contrast to the autoregressive models discussed in Section 2.3, the variational autoencoder (VAE) [Kingma and Welling, 2014] does not construct an explicit density estimator $\hat{p}(\mathbf{x})$. Instead, the VAE constructs a generator function (Definition 2.4) together with a variational

approximation for the density of the pushforward distribution induced by this generator. These pairs of generators and variational approximators are are typically referred to as decoders and encoders respectively; together, a decoder/encoder pair can be interpreted as an autoencoder [Vincent et al., 2010]. The MusicVAE [Roberts et al., 2018] describes an application of a variational autoencoders as a generative model of symbolic music. Although the VAE does not directly model variable-length sequences, local VAE model of fixed windows of sequential data is often used in conjunction with a global autoregressive model over latent codes of the VAE [Chung et al., 2015, van den Oord et al., 2017]. These hybrid models are used to construct a generative model of audio by Dhariwal et al. [2020].

The VAE models a distribution $p(\mathbf{x})$ by introducing a latent variable $\mathbf{z} \sim r$ on an auxiliary space $\mathcal{Z}$ and a parameterized likelihood $p_\theta(\mathbf{x}|\mathbf{z})$. Together, $r(\mathbf{z})$ and $p_\theta(\mathbf{x}|\mathbf{z})$ define a joint distribution over $(\mathbf{x}, \mathbf{z}) \in \mathcal{X} \times \mathcal{Z}$. A density estimator for the data distribution is implicitly defined by

$$p_\theta(\mathbf{x}) = \int_{\mathcal{Z}} p_\theta(\mathbf{x}|\mathbf{z}) r(\mathbf{z}) \, d\mathbf{z}. \tag{2.7}$$

We can sample from $p_\theta(\mathbf{x})$ by first sampling a latent $\mathbf{z} \sim r$, and then sampling $\mathbf{x} \sim p_\theta(\cdot|\mathbf{z})$. The idea is to use a simple prior distribution $r(\mathbf{z})$, e.g., $\mathcal{N}(0, I)$, and parameterize the conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$ with an expressive model. In contrast to ancestral sampling from an autoregressive model (Section 2.3) which requires d serial evaluations of the model to generate a sample $x \in \mathbb{R}^d$, a latent variable model can produce a sample with just one evaluation of the model that parameterizes $p_\theta(\mathbf{x}|\mathbf{z})$.

Directly evaluating the density $p_\theta(\mathbf{x})$ requires computation of Equation (2.7), which makes direct maximum likelihood training and cross-entropy evaluation difficult. This is also a potential obstruction to using a VAE with the sampling techniques developed in Chapter 5. We can construct a variational lower bound on $\log p_\theta(\mathbf{x})$ using a proposal distribution $q(\mathbf{z}|\mathbf{x})$ as an importance sampler [Burda et al., 2016]:

$$\log p_\theta(\mathbf{x}) = \log \mathop{\mathbb{E}}_{\mathbf{z}_i \sim q(\cdot|\mathbf{x})} \left[\sum_{i=1}^{M} \frac{p_\theta(\mathbf{x}, \mathbf{z}_i)}{q(\mathbf{z}_i|\mathbf{x})} \right] \geq \mathop{\mathbb{E}}_{\mathbf{z}_i \sim q(\cdot|\mathbf{x})} \left[\log \sum_{i=1}^{M} \frac{p_\theta(\mathbf{x}, \mathbf{z}_i)}{q(\mathbf{z}_i|\mathbf{x})} \right]. \tag{2.8}$$

When $M = 1$ we recover the evidence lower-bound described by Kingma and Welling [2014]. The bound becomes tight when $q(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$, the posterior distribution over $\mathbf{z}$. Using a variational lower bound like Equation (2.8), we can rephrase empirical cross-entropy minimization (Equation (2.3)) as a variational optimization. This allows us to re-write the maximum likelihood objective as

$$\theta^* = \arg\max_\theta \sup_q \mathop{\mathbb{E}}_{\mathbf{x}\sim p} \log p_\theta(\mathbf{x}) = \arg\max_\theta \sup_q \mathop{\mathbb{E}}_{\mathbf{x}\sim p} \mathop{\mathbb{E}}_{\mathbf{z}_i\sim q(\cdot|\mathbf{x})} \left[\log \sum_{i=1}^{M} \frac{p_\theta(\mathbf{x}, \mathbf{z}_i)}{q(\mathbf{z}_i|\mathbf{x})}\right]. \tag{2.9}$$

The proposal family q is typically chosen to be another expressive, neural parameterization $q_\varphi(\mathbf{z}|\mathbf{x})$. This approach is sometimes called *amortized inference* [Shu et al., 2018] because we replace an expensive computation of the posterior at each value of $\mathbf{x}$ with a model that approximates the posterior across all values of $\mathbf{x}$.

One way to evaluate a VAE is to report a Monte Carlo estimate of Equation (2.8) on fresh samples $\mathbf{x}_1, \ldots, \mathbf{x}_m \sim p$. If q effectively approximates the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ then Equation (2.8) will be a relatively accurate estimate of the log-likelihood. This is conservative estimate, in the sense that it is a lower bound on the true log-likelihood, and will therefore be an overly pessimistic estimate of the cross entropy $H(p, \hat{p})$. We can sharpen the estimate by evaluating Equation 2.8 with $M > 1$ [Burda et al., 2016, Tomczak and Welling, 2018].

$$\log p_\theta(x) = \log \mathop{\mathbb{E}}_{z\sim q(\cdot|x)} \left[\frac{p_\theta(x, z)}{q(z|x)}\right] \approx \log \frac{1}{M} \sum_{i=1}^{M} \frac{p_\theta(x|z_i) r(z_i)}{q(z_i|x)}, \text{ where } z_i \sim q(z|x). \tag{2.10}$$

For any finite M (2.8) is an an upper bound on the log-likelihood, and as $M \to \infty$, the bound becomes tight. Because we cannot compute an exact log-likelihood, we will not consider the VAE when evaluating the conditional sampling methodology developed in Chapter 5. However, it would be possible to extend these results to VAE's using Equation (2.10), where M becomes an additional hyper-parameter that controls a tradeoff between sample quality and computational efficiency.

2.5 *Generative Adversarial Networks*

Like the VAE discussed in Section 2.4, a generative adversarial network (GAN) [Goodfellow et al., 2014] is a procedure for generating samples from a pushforward distribution (Definition 2.3) that avoids construction of an explicit density estimator $p_\theta(\mathbf{x})$. Unlike the VAE, there is no easy way to even approximate the density that a GAN assigns to data. For this reason, GAN's are sometimes referred to as implicit generative models [Mohamed and Lakshminarayanan, 2016]. Implicit generative modeling poses challenges for both training and evaluation, cannot be easily combined with the Bayesian sampling methods developed in Chapter 5. Nevertheless, GAN's have dominated the generative modeling field since their introduction in 2014; notable examples in the audio domain include WaveGAN [Donahue et al., 2019a], GanSynth [Engel et al., 2019], and MelGAN [Kumar et al., 2019]. Curiously, these models have had surprisingly little influence in generative modeling of symbolic data (including written language and musical scores).

A GAN is an optimization procedure that constructs a parameterized generator function $g_\theta : \mathbb{R}^d \to \mathcal{X}$ (Definition 2.4) by solving a saddle point problem. First we attempt to estimate a *lower bound* on a measure of divergence (e.g., the KL-divergence) between the true distribution p and the pushforward distribution p_θ induced by g_θ (Definition 2.3). (e.g., the KL-divergence). Given an estimate of this divergence, we attempt to minimize the lower bound. This procedure should be contrasted with the VAE (Section 2.4) where we instead estimate an *upper bound* on the divergence.

Definition 2.6. *Let $f : \mathbb{R} \to \mathbb{R}$ be a convex, lower-semicontinuous function, such that $f(1) = 0$. We define the **f-divergence** between two distributions with densities p and p_θ on $\mathcal{X}$ by*

$$D_f(p \parallel p_\theta) \equiv \int_{\mathcal{X}} p_\theta(\mathbf{x}) f\left(\frac{p(\mathbf{x})}{p_\theta(\mathbf{x})}\right) d\mathbf{x}. \tag{2.11}$$

An f-divergence generalizes the KL-divergence between two probability distributions. For example, if we take $f(t) = t \log t$ then $D_f(p \parallel p_\theta) = D(p \parallel p_\theta)$. We can construct a lower

bound on a f-divergence $D_f(p \parallel p_\theta)$ without explicit evaluation of the density $p_\theta(\mathbf{x})$. To do this, we introduce the convex conjugate of f, defined by $f^*(s) \equiv \sup_t\{st - f(t)\}$, which can be lifted into a variational representation of the f-divergence [Nguyen et al., 2010] with the following form:

$$D_f(p \parallel p_\theta) = \sup_{T:\mathcal{X}\to\mathbb{R}} \left[\mathop{\mathbb{E}}_{\mathbf{x}\sim p} T(\mathbf{x}) - \mathop{\mathbb{E}}_{\mathbf{x}\sim p_\theta} f^*(T(\mathbf{x})) \right]. \tag{2.12}$$

The f-GAN uses this variational form of the f-divergence to formulate a saddle point problem [Nowozin et al., 2016]. Using an expressive parameterized family of functions T_φ to approximate T, we can minimize an f-divergence between p and p_θ by optimizing

$$\theta^* = \arg\min_\theta \sup_\varphi \left[\mathop{\mathbb{E}}_{\mathbf{x}\sim p} T_\varphi(\mathbf{x}) - \mathop{\mathbb{E}}_{\mathbf{x}\sim p_\theta} f^*(T_\varphi(\mathbf{x})) \right]. \tag{2.13}$$

A second notable GAN formulation is the Wasserstein GAN [Arjovsky et al., 2017], which instead seeks to minimize an optimal transport distance

$$W(p, p_\theta) = \inf_{\pi\in\Pi(p,p_\theta)} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim\pi} \left[\|\mathbf{x} - \mathbf{y}\|_2\right]. \tag{2.14}$$

The notation $\Pi(p, q)$ denotes the collection of probability distributions $\pi(\mathbf{x}, \mathbf{y})$ on the product space $\mathcal{X} \times \mathcal{X}$ with marginals $p(\mathbf{x})$ and $p_\theta(\mathbf{y})$ respectively. We refer the reader to Peyré and Cuturi [2019] for motivation of the Wasserstein distance and further discussion of optimal transport. As written in Equation (2.15), the inner estimation of the Wasserstein distance $W(p, p_\theta)$ is intractable. But a duality argument allows us to reformulate the Wasserstein distance as the solution to a maximization over 1-Lipschitz functions, the Kantorovich-Rubinstein duality [Villani, 2009]:

$$W(p, q) = \inf_{\pi\in\Pi(p,p_\theta)} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim\pi} \left[\|\mathbf{x} - \mathbf{y}\|_2\right] = \sup_{\|h\|_L\le 1} \left[\mathop{\mathbb{E}}_{\mathbf{x}\sim p} \left[h(\mathbf{x})\right] - \mathop{\mathbb{E}}_{\mathbf{x}\sim p_\theta} \left[h(\mathbf{x})\right] \right]. \tag{2.15}$$

This turns the Wasserstein GAN optimization problem into a saddle-point problem with a remarkably similar form to the f-GAN: compare Equation (2.12) to Equation (2.15).

2.6 Generative Flow

Generative flow, or *flow-based* modeling describes a family of density estimators that parameterize a pushforward distribution (Definition 2.3) using discrete flow networks [Dinh et al., 2017], which permit efficient calculation of inverse and Jacobian operations. Flow-based models have been applied extensively to generative modeling in the audio domain. Notably, the Parallel WaveNet model uses a student-teacher paradigm to distill an autoregressive WaveNet model into a flow-based model that can be sampled efficiently [van den Oord et al., 2018]. Direct flow-based models of audio waves include WaveGlow [Prenger et al., 2019], FlowWavenet [Kim et al., 2019], and WaveFlow [Ping et al., 2020]. We will use a flow-based model [Kingma and Dhariwal, 2018] in Chapter 5 as a Bayesian prior for source separation.

Let p_θ be defined implicitly as the pushforward of a density q using a generator $g_\theta : \mathcal{Z} \to \mathcal{X}$ (Definition 2.3). In Equation (2.5), we showed that the density $q(\mathbf{z})$ pushes forward to

$$p_\theta(\mathbf{x}) = q(g_\theta^{-1}(\mathbf{x}))|\nabla_x g_\theta^{-1}(\mathbf{x})|. \tag{2.16}$$

In general this density is difficult to calculate, which lead to our discussions of the VAE in Section 2.4 and the GAN in Section 2.5, which construct optimization objectives that avoid explicit evaluation of Equation (2.16). But if the model class $\{g_\theta : \theta \in \Theta\}$ is chosen carefully so that inverses and Jacobians of g_θ are easily computed, then we can use Equation (2.16) to convert a generator into a density estimator. Furthermore, we can train this generator using maximum likelihood estimation: given i.i.d. samples $\mathbf{x}_1, \ldots, \mathbf{x}_n \sim p$,

$$\theta^* = \arg\max_\theta \mathop{\mathbb{E}}_{\mathbf{x}\sim p} \log p_\theta(\mathbf{x}) \approx \arg\min_\theta \sum_{i=1}^{n} -\log q(g_\theta^{-1}(\mathbf{x}_i)) - \operatorname{logdet} \nabla_x g_\theta^{-1}(\mathbf{x}_i). \tag{2.17}$$

An idea that adapts flow to parameterize a pushforward distribution is developed by Dinh et al. [2015]. The idea is to partition features of the data $\mathbf{x} \in \mathcal{X}$ into two sets. For example, if $\mathcal{X} = \mathbb{R}^d$ then we can partition $\mathcal{X}$ into the first $d/2$ and second $d/2$ features: $\mathbf{x} = (x_{1,\ldots,d/2}, x_{d/2+1,\ldots,d})$. We parameterize a function $h_\theta : \mathbb{R}^{d/2} \to \mathbb{R}^{d/2}$ using an expressive

neural network, and proceed to construct an *additive coupling* $g_\theta : \mathbb{R}^d \to \mathbb{R}^d$ defined by the rule $\mathbf{x} = g_\theta(\mathbf{z})$ where

$$\mathbf{x}_{1:d/2} = \mathbf{z}_{1:d/2}, \tag{2.18}$$

$$\mathbf{x}_{d/2+1:d} = \mathbf{z}_{d/2+1:d} + h_\theta(\mathbf{z}_{1:d/2}). \tag{2.19}$$

This coupling function is invertible: given $\mathbf{x} \in \mathbb{R}^d$, we can recover $\mathbf{z} = g_\theta^{-1}(\mathbf{x})$ by

$$\mathbf{z}_{1:d/2} = \mathbf{x}_{1:d/2}, \tag{2.20}$$

$$\mathbf{z}_{d/2+1:d} = \mathbf{x}_{d/2+1:d} - h_\theta(\mathbf{x}_{1:d/2}). \tag{2.21}$$

And the Jacobian of the inverse transformation is simply

$$\nabla_{\mathbf{x}} g_\theta^{-1}(\mathbf{x}) = \begin{bmatrix} I_{d/2} & 0 \\ -\frac{\partial h_\theta(\mathbf{x}_{1:d/2})}{\partial \mathbf{x}_{1:d/2}} & I_{d/2} \end{bmatrix}. \tag{2.22}$$

Therefore $\operatorname{logdet}(\nabla_{\mathbf{x}} g_\theta^{-1}(\mathbf{x})) = 0$: this is a volume-preserving transformation.

While an additive coupling is not very expressive (it is the identity on half of the features) the idea is to chain multiple affine couplings $g_\theta^{(k)}$ together (with different partitions at each step) to construct a deep network $g_\theta(\mathbf{z}) = g_\theta^{(L)} \circ \cdots \circ g_\theta^{(1)}(\mathbf{z})$. Curiously, these chains of additive couplings are the same construction as the Feistel network [Luby and Rackoff, 1988] used in the cryptography community to construct ciphers (e.g., DES and Blowfish). Even at significant depth, generative flow models seem to suffer empirically from this restrictive model structure, requiring substantially more parameters and training time than other modeling techniques (at least for popular vision tasks). Some preliminary work that attempts to remove the architectural restrictions using approximations to the inverse and its Jacobian are presented by Keller et al. [2020].

One simple extension of additive flow is the real non-volume preserving transformation (realNVP) [Dinh et al., 2017] which replaces the additive couplings (2.18) and (2.19) with

affine couplings. Given parameterized scale and translation functions $s_\theta, t_\theta : \mathbb{R}^{d/2} \to \mathbb{R}^{d/2}$, we define an *affine coupling* by $\mathbf{x} = g_\theta(\mathbf{z})$ by

$$\mathbf{x}_{1:d/2} = \mathbf{z}_{1:d/2}, \tag{2.23}$$

$$\mathbf{x}_{d/2+1:d} = \mathbf{z}_{d/2+1:d} \odot \exp(s_\theta(\mathbf{z}_{1:d/2})) + t_\theta(\mathbf{z}_{1:d/2}). \tag{2.24}$$

The multiplicative scaling $\odot$ should be interpreted elementwise. The Jacobian in this case is

$$\nabla_{\mathbf{x}} g_\theta^{-1}(\mathbf{x}) = \begin{bmatrix} I_{d/2} & 0 \\ \dots & \operatorname{diag}\left(\exp(-s_\theta(\mathbf{x}_{1:d/2}))\right) \end{bmatrix} \tag{2.25}$$

And the log determinant of the Jacobian is just

$$\operatorname{logdet}(\nabla_{\mathbf{x}} g_\theta^{-1}(\mathbf{x})) = -\sum_{i=1}^{d/2} s_\theta(\mathbf{x}_{1:d/2})_i. \tag{2.26}$$

This idea is explored at scale by Kingma and Dhariwal [2018] to create the Glow models that we use extensively in Chapter 5.

2.7 Energy-Based Models

The idea of an energy-based model (EBM) [Hinton, 2002, Ranzato et al., 2007] is, rather than explicitly learning a probabilistic model $p_\theta(\mathbf{x})$ over a space $\mathcal{X}$, to instead learn an energy functional $E_\theta : \mathcal{X} \to \mathbb{R}$. This energy functional can be used to implicitly define a probability distribution, for example a Gibbs distribution

$$p_\theta(\mathbf{x}) = \frac{1}{Z_\theta} e^{-E_\theta(\mathbf{x})}, \text{ where } Z_\theta = \int_{\mathcal{X}} e^{-E_\theta(\mathbf{y})}\, d\mathbf{y}. \tag{2.27}$$

The point is that, while it is easy to construct a function $E_\theta(\mathbf{x})$, it can be quite challenging to enforce the constraint $\int_{\mathcal{X}} p_\theta(\mathbf{x}) = 1$, or to compute the partition function Z_θ for a given energy function $E_\theta(\mathbf{x})$.

To use an EBM as a generative model, we need to solve two problems. First, we need a training procedure for optimizing the parameters of the energy function E_θ so that the implicit distribution $p_\theta(\mathbf{x})$ approximates the data generating distribution $p(\mathbf{x})$. Second, we need a sampling procedure for drawing samples $\mathbf{x} \sim p_\theta$. Solutions to both problems should avoid calculation of the intractable integral Z_θ. For early approaches to this problem based on the contrastive divergence, see Hinton [2002] and Hinton et al. [2006]. For a modern, empirical realization of these ideas see Du and Mordatch [2019]. These ideas have been applied to generative modeling in the audio domain in recent concurrent work by Kong et al. [2021] (DiffWave) and Chen et al. [2021] (WaveGrad). We will use an energy-based model [Song and Ermon, 2019] in Chapter 5 as a Bayesian prior for source separation.

Sampling from an EBM. Suppose we have a model E_θ and we want to sample from the implied distribution $\mathbf{x} \sim p_\theta$. While directly sampling from p_θ is difficult, we can approximate samples using a Markov chain with stationary distribution p_θ. One Markov chain on $\mathcal{X} = \mathbb{R}^d$ with the required stationary distribution is the Langevin diffusion; this is a continuous Markov process with dynamics given by the stochastic differential equation

$$\frac{\partial \mathbf{x}_t}{\partial t} = \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}_t)\, dt + \sqrt{2}\, dW_t, \tag{2.28}$$

where dW_t is a white noise process, given by the derivative of standard Brownian motion W_t. The Fokker-Planck equation establishes that a diffusion following these dynamics converges asymptotically to samples $\mathbf{x}_t \sim p_\theta$, in the sense that $D(\mathbf{x}_t \parallel p_\theta) \to 0$ as $t \to \infty$. This general Langevin-based Markov-chain Monte Carlo sampler was first proposed by Grenander [1983], Grenander and Miller [1994]. For an expository treatment of these ideas, see Pavliotis [2014].

We cannot exactly construct a diffusion $\mathbf{x}_t$ that follows the dynamics of Equation (5.1) using numerical methods. In practice, we will discretize the diffusion and follow a discrete Markov chain driven by i.i.d. Gaussian noise $\varepsilon_t \sim \mathcal{N}(0, I)$:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}_t) + \sqrt{2\eta}\varepsilon_t. \tag{2.29}$$

Equation (5.1) is referred to as the *unadjusted Langevin algorithm* [Roberts and Tweedie, 1996, Durmus and Moulines, 2017]. It can be viewed as the stochastic analog to an Euler discretization of a deterministic differential equation. As $\eta \to 0$, the approximation to the continuous dynamics of Equation (5.1) becomes more precise, but mixing is slow; an effective accelerated mixing algorithm based on simulated annealing [Kirkpatrick et al., 1983] is developed by Song and Ermon [2019]; we build upon these techniques in Chapter 5. We can use the unadjusted Langevin algorithm to sample from an energy based model, because

$$\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} E_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log Z_\theta = -\nabla_{\mathbf{x}} E_\theta(\mathbf{x}). \tag{2.30}$$

Training an EBM. A direct approach to modeling the *score function* $s : \mathbb{R}^d \to \mathbb{R}^d$ defined by $\mathbf{x} \mapsto \nabla_{\mathbf{x}} \log p(\mathbf{x})$ (or equivalently, $\mathbf{x} \mapsto \nabla_{\mathbf{x}} E_\theta(\mathbf{x})$) is described by Song and Ermon [2019]; we use models based on this approach extensively in Chapter 5. Want can parameterize the score function with a neural network $s_\theta : \mathbb{R}^d \to \mathbb{R}^d$, which implicitly defines an energy function $E_\theta : \mathbb{R}^d \to \mathbb{R}$ (by integration) and a density p_θ (by normalization). We can fit this score function by minimization of the *Fisher divergence*

$$D_{\text{Fisher}}(p \parallel p_\theta) \equiv \mathop{\mathbb{E}}_{\mathbf{x}\sim p}\left[\frac{1}{2}\left\|\nabla_x \log \frac{p(\mathbf{x})}{p_\theta(\mathbf{x})}\right\|_2^2\right] = \mathop{\mathbb{E}}_{\mathbf{x}\sim p}\left[\frac{1}{2}\|s_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{x})\|_2^2\right]. \tag{2.31}$$

There is a precise connection between Fisher divergence and the rate of change in KL-divergence over smoothed versions of p sand q [Lyu, 2009]. Define $\tilde{\mathbf{x}}_t = \mathbf{x} + \sqrt{t}\varepsilon_x$ and $\tilde{\mathbf{y}}_t = \mathbf{y} + \sqrt{t}\varepsilon_y$, where $\mathbf{x} \sim p$, $\mathbf{y} \sim q$, and $\varepsilon_x, \varepsilon_y \sim \mathcal{N}(0, I)$ (independent samples). Let $p_t(\tilde{\mathbf{x}}_t)$ and $q_t(\tilde{\mathbf{y}}_t)$ denote the densities of $\tilde{\mathbf{x}}_t$ and $\tilde{\mathbf{y}}_t$ respectively; adding Gaussian noise to $\mathbf{x}, \mathbf{y}$ corresponds to smoothing of their probability densities (Gaussian convolution). Under mild regularity conditions,

$$\frac{d}{dt} D(p_t \parallel q_t) = -D_{Fisher}(p_t \parallel q_t). \tag{2.32}$$

Because Fisher divergence is non-negative, integrating we see that $D(p_t \parallel q_t) \to 0$ as $t \to \infty$, and this convergence is monotonic.

We cannot directly compute the score matching objective, because it required evaluation of (gradients of) the unknown density $p(\mathbf{x})$. But a result by Hyvärinen [2005] shows how we can minimize it implicitly:

$$\arg\min_\theta \mathop{\mathbb{E}}_{\mathbf{x}\sim p}\left[\frac{1}{2}\|s_\theta(\mathbf{x}) - \nabla_x \log p(\mathbf{x})\|_2^2\right] = \arg\min_\theta \mathop{\mathbb{E}}_{\mathbf{x}\sim p}\left[\operatorname{tr}\left(\nabla_x s_\theta(\mathbf{x})\right) + \frac{1}{2}\|s_\theta(\mathbf{x})\|_2^2\right]. \quad (2.33)$$

The right-hand side of Equation (2.33) can be approximated by Monte Carlo estimation. But this is not yet a convenient objective, because $\operatorname{tr}(\nabla_x s_\theta(\mathbf{x}))$ is a second-order statistic: the trace of the Hessian of $\log p_\theta(\mathbf{x})$. Song et al. [2019] proposed a tractable variant by minimizing Equation (2.31) along random projections $v \sim r$, e.g., from a Gaussian $r = \mathcal{N}(0, I)$:

$$L(\theta, v) \equiv \mathop{\mathbb{E}}_{\mathbf{x}\sim p}\left[\frac{1}{2}\left(v^T s_\theta(\mathbf{x}) - v^T \nabla_{\mathbf{x}} \log p(\mathbf{x})\right)^2\right]. \quad (2.34)$$

Applying Proposition **??** yields an objective that can be estimated from samples:

$$\arg\min_\theta \mathop{\mathbb{E}}_{v\sim r} L(\theta, v) = \arg\min_\theta \mathop{\mathbb{E}}_{v\sim r} v^T \mathop{\mathbb{E}}_{\mathbf{x}\sim p}\left[\frac{1}{2}\|s_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{x})\|^2\right] v \quad (2.35)$$

$$= \arg\min_\theta \mathop{\mathbb{E}}_{v\sim r} v^T \mathop{\mathbb{E}}_{\mathbf{x}\sim p}\left[\operatorname{tr}\left(\nabla_{\mathbf{x}} s_\theta(\mathbf{x})\right) + \frac{1}{2}\|s_\theta(\mathbf{x})\|_2^2\right] v \quad (2.36)$$

$$= \arg\min_\theta \mathop{\mathbb{E}}_{\substack{v\sim r \\ \mathbf{x}\sim p}}\left[v^T \nabla_{\mathbf{x}} s_\theta(\mathbf{x}) v + \frac{1}{2}\left(v^T s_\theta(\mathbf{x})\right)^2\right]. \quad (2.37)$$

Equation (2.37) involves only Hessian-vector products, which can be computed with time complexity that is independent of the data dimension. So long as our random projections $v \sim r$ span $\mathbb{R}^d$, we can recover the data generating distribution $p(\mathbf{x})$ by minimizing the expected loss $L(\theta, v)$. To be precise, if $p(\mathbf{x}) = p_{\theta^*}(\mathbf{x})$ for some value of the parameters θ^* (the data-generating distribution is realizable) and if r is positive definite (i.e., $\mathbb{E}_{v\sim r}[vv^T] \succ 0$) then $\mathbb{E}_{v\sim r} L(\theta, v) = 0$ if and only if $\theta = \theta^*$ [Song et al., 2019].

Chapter 3

MUSIC ALIGNMENT AND TRANSCRIPTION

Music transcription can be framed as a structured prediction task, which requires us to predict a score (for example, a piano roll described by Definition 2.1) given an audio recording of that score (as described by Definition 2.2). We can view this task as a conditional generative modeling problem, in which we seek to estimate the distribution $p(\textbf{score}|\textbf{audio})$. Because both **score** and **audio** objects have a temporal axis, we can also view this as a sequence-to-sequence learning problem [Sutskever et al., 2014]. Learning the conditional distribution can be supervised by collections of $(\textbf{audio}, \textbf{score})$ pairs.

In this chapter we focus on a simplified transcription task: *frame-based* music transcription [Bay et al., 2009]. This task, which we formalize in Section 3.3.2, requires us to predict a *performance-aligned score*, **pscore**, aligned to a given audio recording **audio**. We can again view this task as a conditional generative modeling problem: estimate the distribution $p(\textbf{pscore}|\textbf{audio})$. Frame-based transcription can be a convenient simplification of the transcription task, because it eliminates the problem of constructing a global alignment between a score and a performance timings, focusing our attention on the relatively local problem of extracting the precise content of an acoustic signal at any given time in the recording. The frame-based simplification of transcription is so standard that it is often identified with the broader music transcription problem in the music information retrieval literature.

While the frame-based simplification of music transcription can be a convenient formulation of a learning problem, it has shortcomings. From a practitioner's perspective, the output of a frame-based transcription algorithm (a **pscore**) is not an object that a musician would recognize as a score. Indeed, a **pscore** cannot be typeset as a score without solving an additional structured prediction task: prediction of a **score** given a **pscore**. Little work has

been done on this latter problem, and without a robust algorithm for converting **pscore**'s to **score**'s, it is difficult to advocate for frame-based transcription as a complete solution to the music transcription problem.

From a researcher's perspective, frame-based transcription simplifies the learning problem at the expense of complicating the data collection process. The supervision required to learn a conditional model $p(\textbf{score}|\textbf{audio})$ is a simple collection of $(\textbf{audio}, \textbf{score})$ pairs; scores and performances are available in abundance, and pairing up audio files with the corresponding scores is generally a straightforward task using metadata. But frame-based transcription requires pairs $(\textbf{audio}, \textbf{pscore})$. Performance-aligned scores are not readily available, and must be constructed from a score for any given performance. Consequently, after gathering a collection of $(\textbf{audio}, \textbf{score})$ pairs, we must then somehow convert our **score**'s into **pscores**'s using some variant of a procedure known as *audio-to-score alignment*.

The objective of this chapter is to build towards a solution to the frame-based music transcription problem. Because alignment algorithms play a fundamental role in constructing datasets for this task, it is natural to ask: how do we evaluate an alignment algorithm? We pose this question in Section 3.1. Answering this question requires a precise definition of what constitutes an alignment; we propose a formal definition of an alignment in Section 3.1.1 and define the notion of a ground-truth alignment in Section 3.1.2. We review previous efforts to evaluate alignments in Section 3.1.3, and proceed in Section 3.1.4 to construct a dataset for evaluating alignments based on these principles introduced in Section 3.1.2. In Section 3.2 we put an alignment algorithm to use, constructing the MusicNet dataset: a collection of freely-licensed recordings of chamber music performances together with corresponding performance-aligned scores. Using the evaluation framework developed in Section 3.1, we quantify the behavior of the alignment algorithm used to construct MusicNet in Section 3.2.3. We proceed to consider the quality of the alignments constructed using this algorithm in Section 3.2.4 and Section 3.2.5. Finally, we construct and train models for the frame-based transcription task using the aligned MusicNet dataset in Section 3.3. We are particularly interested in answering the following question: are time-frequency representations of audio

(Section 3.3.3) an advantageous featurization of musical recordings, or should we prefer to learn to transcribe directly from raw audio (Section 3.3.4)? In Section 3.3.6, we study these questions using MusicNet.

3.1 Audio-to-Score Alignment

Audio-to-score alignment is a fundamental problem in music information retrieval [Sakoe and Chiba, 1978] with applications ranging from score following [Dixon, 2005] to music transcription [Turetsky and Ellis, 2003]. The concept of an alignment is typically given by an intuitive definition, for example:

1. "Music alignment is the association of events in a score with points in the time axis of an audio signal." [Orio and Schwarz, 2001].

2. "A procedure which, for a given position in one representation of a piece of music, determines the corresponding position within another representation." [Ewert et al., 2009].

3. "Match notes in a music performance signal (called an aligned signal) to those in a reference musical score or another performance signal (reference signal)." [Nakamura et al., 2017].

We identify two problems with deferring to intuition for such a foundational concept. First, as we will discuss in Section 3.1.1, these definitions are subtly inconsistent with each other. This is a potential source of confusion in how the community thinks and talks about alignments. Second, while the definition of an alignment is vague, the traditional metrics used to evaluate alignments are quite precise (see Section 3.1.2). In the absence of an explicit definition of an alignment, these metrics become an implicit substitute for a definition. This is a problem if these metrics fail to capture the right intuitive concept. Furthermore, without a precise definition, it is difficult to critically analyze how well these metrics measure the concept of a good alignment.

With the aim of developing a principled methodology for evaluating the quality of an audio-to-score alignment algorithm, we formalize a definition of an alignment in Section 3.1.1. In Section 3.1.2, we introduce the concept of a ground-truth alignment and methodology for evaluating an alignment algorithm: the idea is to measure an alignment algorithm by its ability to accurately approximating a ground-truth alignment. In Section 3.1.3 we consider previous methodologies for evaluating alignment algorithms, and the difficulties these methods face. In Section 3.1.4 we introduce an dataset of approximate ground-truth alignments, which can be used to evaluate alignment algorithms using the methodology introduced in Section 3.1.2. We will use these methods in Section 3.2 to evaluate the alignment algorithm used to construct the MusicNet dataset.

3.1.1 A Definition of an Alignment

Given a score of length S and an expressive performance of this score of duration T, we define an audio-to-score alignment to be a function of the following form.

Definition 3.1. *A (temporal) audio-to-score alignment is a monotonic real function* $\tau : [0, S) \to [0, T)$ *that assigns each position* $s \in [0, S)$ *in the score to a time* $t = \tau(s) \in [0, T)$ *in the audio performance.*

This definition is expansive, intended to capture all functions that could meaningfully be interpreted as alignments. We introduce the concept of the *best* alignment between a score and a performance in Section 3.1.2.

Definition 3.1 formalizes the concept of a *temporal alignment*, described clearly by Ewert et al. [2009] (quoted at the top of Section 3.1, Item 2). An alternate definition, that we will refer to as a *note-based alignment*, is eloquently stated by Nakamura et al. [2017] (quoted at the top of Section 3.1, Item 3). Whereas Ewert et. al. view alignments as functions (i.e., "procedures") that map *positions* in one sequence to another, Nakamura et. al. view alignments as functions (i.e., "matchings") that map *notes* in one sequence to another. Both perspectives on alignment are common in the literature, although the definition of an

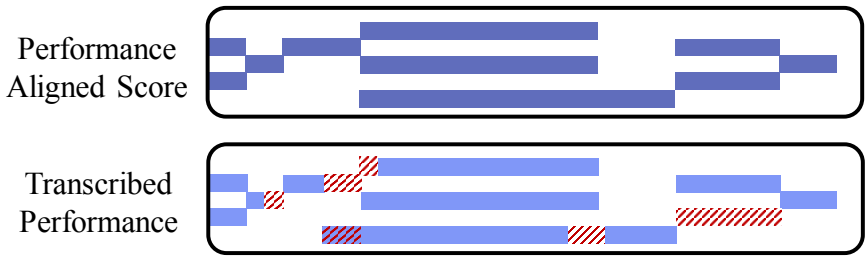


Figure 3.1: Two symbolic performance piano-rolls. Top: a score is aligned to an audio performance to create a performance-aligned score. Bottom: the same audio performance is been transcribed to create a performance transcript. Asynchronies and errors in the pianist's performance prevent a perfect alignment (red dashes).

alignment is often presented in more ambiguous terms (e.g., the definition quoted at the top of Section 3.1, Item 1).

Imagine a performer strikes three notes on the fourth beat of a score asynchronously, as in Figure 3.1. A temporal alignment necessarily maps the onsets of these three notes in the score to a single point of time in the performance. In contrast, a note-based alignment could map the onsets of these notes to distinct times in the performance. Note-based alignments have more flexibility to alter the structure of a score to match the content of an audio recording. But flexibility comes at a cost. If the note-based alignment maps three notes in the triad to different times in the performance, then it cannot answer the following question: at what time in the performance is the fourth beat?

An alignment function τ (Definition 3.1) implicitly assigns notes in a score to time intervals in a performance. If a note spans the interval $[s_1, s_2)$ in the score, then it aligns to the interval $[\tau(s_1), \tau(s_2))$ in the performance. Monotonicity ensures that $\tau(s_1) \leq \tau(s_2)$. Mapping the timings of all the notes in the score to timings in the performance via an alignment function yields a performance-aligned score. Given a piano-roll **score** and an alignment τ to a performance, we can construct a performance-aligned version of the score with expressive timings given by τ, which we formalize in Definition 3.2.

Definition 3.2. *Given a piano-roll* **score** *and an alignment τ, the associated performance-*

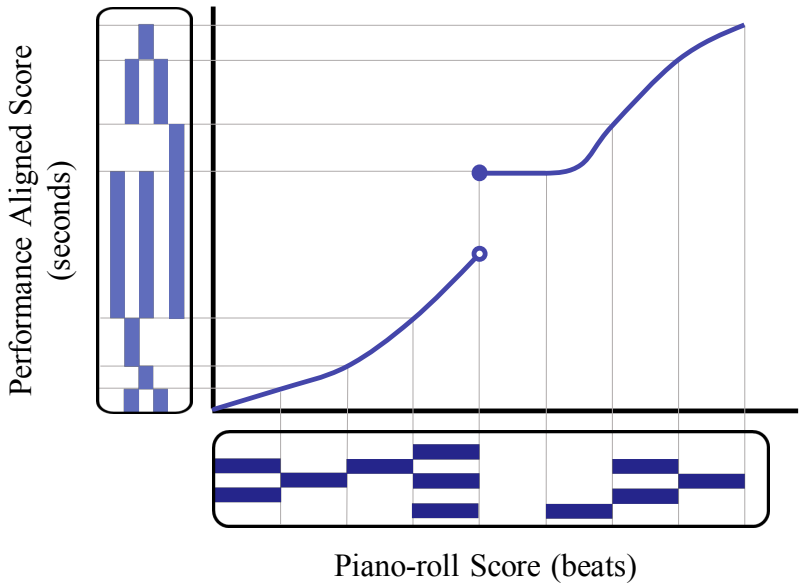


Figure 3.2: An alignment function τ, that maps time in a score (measured in beats) to time in a performance (measured in seconds). A piano-roll **score** is mapped by the alignment to a performance-aligned score $\mathbf{pscore}_\tau$. Time pauses in the score as the performer extends the triad, resulting in a jump discontinuity in the alignment. Because the alignment τ is not invertible, the set $\tau^{-1}(t)$ may contain 0, 1, or many elements.

aligned score is given by the piano-roll $\mathbf{pscore}_\tau : [0, T) \to \{0,1\}^N$ *such that*

$$\mathbf{pscore}_\tau(t) = \bigvee_{s \in \tau^{-1}(t)} \mathbf{score}(s).$$

The inverse image of t in τ, denoted by $\tau^{-1}(t)$, is the set of times in the score that τ aligns to time t in the performance. Alignments τ need not be invertible, so the set $\tau^{-1}(t)$ may contain 0, 1, or many elements, as seen in Figure 3.2. The notes in a **pscore** at time t are the union of notes denoted in the score at each time $s \in \tau^{-1}(t)$. We represent notes by indicators in a binary vector $\{0,1\}^N$, so this union can be represented as the bitwise logical disjunction of these vectors: $x_n \vee y_n = 1$ iff $x_n = 1$ or $y_n = 1$. The $\bigvee$ notation in Definition 3.2 indicates bitwise logical disjunction over a collection of values.

We can think of a **pscore** as a dense, fine-grained local labeling of an audio recording

analogous to a semantic segmentation map for visual data [Farabet et al., 2013, Long et al., 2015, Isola et al., 2017]. Another way to think about Definition 3.2 is that we have elevated an alignment $\tau : [0, S) \to [0, T)$ to a function $\texttt{align}_\tau$: Piano Roll $\to$ Piano Roll defined by the rule $\texttt{align}_\tau(\textbf{score}) = \textbf{pscore}_\tau$. It is easy to conflate the real function τ with the lifted function $\texttt{align}_\tau$, or with the image $\textbf{pscore}_\tau$ of the score in this lifted alignment function; this may have contributed to the confusion about definitions alluded to in Section 3.1.

While we have formalized and focused on the concept of a temporal alignment, we emphasize that note-based alignment is also an important problem. Audio-to-score alignment is commonly used as an intermediate tool to facilitate other MIR tasks: score-following, music transcription, beat tracking, and onset detection, among others. For tasks such as score following and and beat tracking, we want a temporal alignment that tell us what time a particular position in the score corresponds to in the performance. For tasks including frame-based music transcription and onset detection, we would actually prefer a note-based alignment that tells us where notes in the score occur in the performance. Nevertheless, we focus on temporal alignments in this work because this is the definition that captures the notion of an alignment produced by a dynamic time warping algorithm; this family of algorithms has been used to construct all of the current alignment-based music transcription datasets discussed in Section 3.2 (including MusicNet).

3.1.2 Alignment Algorithm Evaluation Metrics

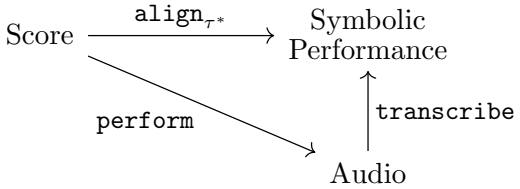
To evaluate an alignment algorithm, we want to evaluate the expected quality of the alignments it produces. To evaluate the quality of an alignment between a given score and a corresponding performance, we want to compare to the best, ideal alignment between that score and performance. Conceptually, this ideal alignment exists and is unique. To convince ourselves of this, imagine assigning the start of each beat in a score to a specific time in the performance. This assignment forms a crude, discretized alignment; by carefully interpolating this alignment to sub-beats and sub-sub-beats etc., we flesh out the full alignment function. Such thinking motivated the evaluation procedure described by Raphael [2004],

which approximates ground-truth alignments from annotations given by a person, tapping along to the beat of a performance.

To define the ideal ground-truth alignment, we introduce the concept of a *symbolic performance*. For example, a symbolic performance could consist of a sequence of notes recorded by a MIDI keyboard. We can represent a symbolic performance as a piano-roll, denoted by $\mathbf{symbp} : [0, T) \to \{0, 1\}^N$, defined by the rule $\mathbf{symbp}(t)_n = 1$ iff note n is being performed at time $t \in [0, T)$. In contrast to a score, a symbolic performance has expressive timings, measured in units of seconds (see Figure 3.2). A performance-aligned score is a special case of a symbolic performance. Another example of a symbolic performance is the output of an ideal, frame-based music transcription system `transcribe` : Audio $\to$ Piano Roll which, at any point in an audio performance, indicates the set of notes being performed at that point in time.

A performance-aligned score is a special case of a symbolic performance. Another example of a symbolic performance is the output of an ideal, frame-based music transcription system `transcribe` : Audio $\to$ Piano Roll which, at any point in an audio performance, indicates the set of notes being performed at that point in time. We consider the acquisition of symbolic performance transcripts in Section 3.1.4. For now, we assume that we have access to an accurate transcription of a given audio performance.

We can use symbolic performance transcripts to define ideal ground-truth alignments between scores and audio performances. Let `perform` : Score $\to$ Audio denote the action of a performer, who converts a score into audio through the act of performance. Borrowing the language of category theory [Mac Lane, 1971], we define an ideal ground-truth alignment to be an alignment function that makes the following diagram "commute:"



What we mean when we say that the diagram commutes is captured by the following defi-

nition.

Definition 3.3. *An ideal ground-truth alignment between a score* **score** *and an audio performance* $\texttt{perform}(\textbf{score})$ *is an alignment* τ^* *such that*

$$\texttt{transcribe}(\texttt{perform}(\textbf{score})) = \texttt{align}_{\tau^*}(\textbf{score}).$$

In words: the ideal ground-truth alignment of a score to a given audio performance yields the same symbolic performance as a perfect transcript of the audio.

Unfortunately, the ideal ground-truth alignments defined in Definition 3.3 do not exist for most score-performance pairs, because human performances do not faithfully reproduce a musical score. In some cases, this is due to performance error: the insertion or deletion of a note that is not reflected in the score. Even for highly-accurate, professional performances, the deliberate extensions, asynchronies, and staccato articulations that comprise an expressive performance prevent an alignment that exactly satisfies equality in Definition 3.3.

Mismatches between a performance-aligned score and a corresponding performance transcript are illustrated in Figure 3.1. Observe that, for the score and performance transcript illustrated in the figure, there is no alignment of the score that is equal to the given performance; among other problems, the missing note (marked completely in red) in the second-to-last beat of the performance cannot be avoided by any performance-aligned score constructed from a function given by Definition 3.1. We must therefore settle for an approximation to the equality in Definition 3.3; we discuss how to obtain approximate ground-truth alignments in Section 3.1.4.

We can measure the quality of a proposed alignment by how well it conforms to the commutative identity. We first observe that there is an inherent alignment ambiguity between changepoints in a score. During these intervals, time evolves even though the set of notes being played does not change. The path the alignment takes between these two score events is therefore underspecified. This is demonstrated in the left panel of Figure 3.3, where the ground-truth and candidate alignments satisfy $\tau(s_i) = \tau^*(s_i)$ at each changepoint s_i, but

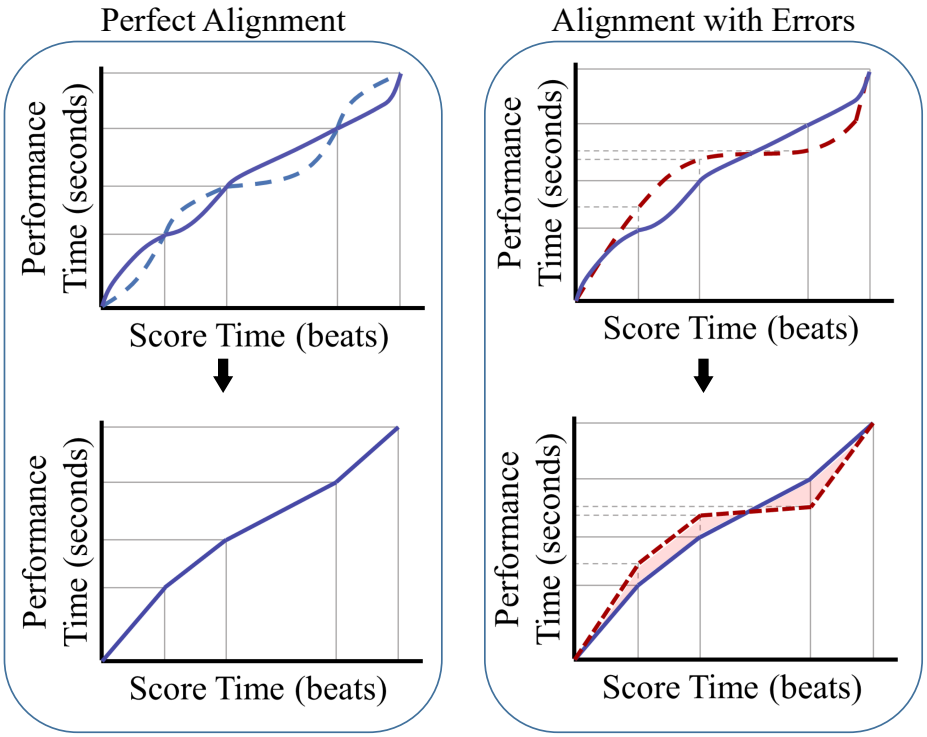


Figure 3.3: A possible ground-truth alignment (solid) compared to two candidate alignments (dashed), shown before (top) and after (bottom) linearization. The first candidate alignment (left) is perfect; its canonical linear representative exactly matches that of the ground-truth. The second candidate alignment (right) maps score changepoints erroneously; its error is given by the shaded region.

evolve differently between changepoints. We consider two alignments τ_1, τ_2 *equivalent* if $\tau_1(s_i) = \tau_2(s_i)$ for every changepoint s_i in the score. To compare two alignments, we will compare canonical representatives of their equivalence classes. As a canonical representative, we pick the alignment that linearly interpolates between changepoints; we write $\tilde{\tau}$ to denote this linearization of an alignment function τ. Linear interpolation [Kress, 1998] is a natural choice because it represents time evolving at a constant pace between changepoints.

Given a ground-truth alignment τ^*, we define the temporal error of an alignment τ by the L^1 distance between τ and τ^*.

Definition 3.4. *Temporal average error between alignments τ and τ^* is given by*

$$\text{TimeError}(\tau, \tau^*) \equiv \frac{1}{S} \int_0^S |\tilde{\tau}(s)) - \tilde{\tau}^*(s)| ds.$$

At any point s in the score, the quantity $|\tilde{\tau}(s)) - \tilde{\tau}^*(s)|$ measures how far the candidate alignment's position in the performance, $\tilde{\tau}(s)$, differs from the ground-truth time in the performance, $\tilde{\tau}^*(s)$. TimeError computes the average of these errors over score time.

The alignment paths produced by common alignment algorithms, including dynamic time warping, are not necessarily functions: the alignment path can have vertical jumps, with one time position in the score corresponding to an interval of time in the performance. In these cases, we choose to treat the alignments as right-continuous functions, as illustrated in Figure 3.2. A discontinuous alignment function τ arising from an alignment path with vertical jumps will map the content of the score at position s to the last performance time given by the provided alignment.

The TimeError metric alone does not distinguish between different distributions of the alignment error. For example, one alignment may lag by 10ms for the duration of the performance, while another may have one measure with large error in an otherwise perfect alignment. To distinguish between these types of errors, we also consider the standard deviation of errors over the entire score.

Definition 3.5. *Temporal standard deviation (*TimeDev*) between alignments τ and τ^* is given by*

$$\text{TimeDev}(\tau, \tau^*) \equiv \sqrt{\frac{1}{S} \int_0^S \left(\tilde{\tau}(s)) - \tilde{\tau}^*(s)\right)^2 ds}.$$

Because the alignments are linearized, both TimeError and TimeDev can be calculated in closed form. To avoid extraneous alignment errors involved in guessing the offset time of the final notes of a performance, we define the end of the score S to be the time of the last *onset* in the score.

The TimeError and TimeDev metrics measure temporal differences in the performance

with respect to *positions* in the score. In contrast, audio-to-score alignment are usually evaluated based on temporal differences in the performance with respect to *notes* in the score. These note-based metrics are clearly described by Cont et al. [2007], although similar metrics appeared in earlier audio-to-score alignment papers [Meron and Hirose, 2001, Orio and Schwarz, 2001, Shalev-Shwartz et al., 2004]. The most commonly reported metric is the mean absolute difference between aligned note onsets and ground-truth onsets [Meron and Hirose, 2001, Orio and Schwarz, 2001, Shalev-Shwartz et al., 2004, Keshet et al., 2007, Ewert and Müller, 2008, Ewert et al., 2009, Devaney and Ellis, 2009, Devaney, 2014, Niedermayer, 2009, Niedermayer and Widmer, 2010, Lajugie et al., 2016, Kwon et al., 2017, Arzt and Lattner, 2018]. This metric is a note-based analog to Definition 3.4; given a list of N onsets $\mathbf{s}$ in a score corresponding to onsets $\mathbf{p}$ in a performance, we can compute

$$\text{NoteError}(\mathbf{s}, \mathbf{p}) \equiv \frac{1}{N} \sum_{i=1}^{N} |\mathbf{s}_i - \mathbf{p}_i|. \tag{3.1}$$

Some works additionally report mean deviation of note-offsets, and standard deviations of onset and offset times. The standard deviation of onset times is a note-based analog to Definition 3.5:

$$\text{NoteDev}(\mathbf{s}, \mathbf{p}) \equiv \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\mathbf{s}_i - \mathbf{p}_i)^2}. \tag{3.2}$$

Another commonly reported metric is the fraction of onsets that exceed some threshold deviance from ground-truth, sometimes called the "onset recognition rate" [Cont et al., 2007, Joder et al., 2011, 2013, Joder and Schuller, 2013, Carabias-Orti et al., 2015].

In light of the discussion in Section 3.1.1, there is an evident mismatch between the concept of a temporal alignment and these note-based evaluation metrics. Indeed, we argue that these note-based metrics have pushed researchers to overly focus on the note-based alignment problem: the flexibility to adjust the onset times of individual notes allows note-based alignment algorithms to easily outperform temporal alignment algorithms when we measure them using note-based metrics. Even Ewert's work [Ewert and Müller, 2008, Ewert

et al., 2009], which clearly states its intent to construct temporal alignments, evaluates using the note-based metrics.

3.1.3 Related Work on Evaluating Alignments

A common way to evaluate alignments is to construct a synthetic dataset using a synthesizer [Meron and Hirose, 2001, Orio and Schwarz, 2001, Hu et al., 2003, Maezawa and Okuno, 2015, Lajugie et al., 2016]. This approach is convenient, because the deterministic nature of a synthesizer gives a direct ground-truth alignment between a score and its synthesized performances. But results on synthesized performances could mislead us if we want to understand how an algorithm will behave on human performances. Furthermore, a score cannot be directly synthesized in this approach: because synthesizers produce inexpressive, constant-tempo performances, and aligning to such performances is trivial. The usual solution is to perturb note onsets and offsets in the score before synthesizing [Ewert et al., 2012, Raffel and Ellis, 2016], but the artificial nature of these perturbations could also make the results misleading.

Another approach is to use performance transcripts captured during an acoustic piano performance using a sensor array that records key and pedal presses [Soulez et al., 2003, Shalev-Shwartz et al., 2004, Keshet et al., 2007, Niedermayer, 2009, Niedermayer and Widmer, 2010, Joder et al., 2010, 2011, 2013, Joder and Schuller, 2013, Kwon et al., 2017]. Collections of these performances and transcripts have been made widely available in the MAPS [Emiya et al., 2010] and MAESTRO [Hawthorne et al., 2019] datasets. This captured data consists of aligned symbolic performances (**symbp**'s in the language of Section 3.1.2) that, by construction, is aligned to the recorded acoustic performance. Therefore, as with synthesized datasets, common practice is to randomly adjust the timings of the performance transcripts to create a de-aligned "score" for input to the alignment algorithm. These adjusted transcripts are unlikely to look anything like actual scores, which casts doubt on resulting alignment evaluations.

Finally there is the labor-intensive approach of gathering scores, corresponding perfor-

mances, and manually annotating ground-truth alignments between these pairs. This approach was taken for the Bach10 dataset [Duan et al., 2010]. Other datasets have been constructed this way, and are used for the MIREX Real-time Audio to Score Alignment task [Miron et al., 2014, Arzt and Lattner, 2018], but these datasets are not public. This secrecy is understandable, because the effort required to create this data means the datasets are most valuable as privately-held test sets for use in competitions. The high cost of obtaining alignment labels through manual annotation greatly limits the scale of these datasets: the Bach10 dataset consists of 10 short recordings of Bach Chorales, totaling 5.5 minutes of music. Larger datasets are desirable, particularly if we anticipate the development of alignment algorithms based on machine learning that require training data.

3.1.4 A Dataset of Ground-Truth Alignments

To evaluate alignment algorithms using the metrics introduced in Section 3.1.2 requires a dataset consisting of:

1. Musical scores in a symbolic digital format.

2. Relatively faithful performances of these scores.

3. Ground-truth alignment annotations.

We will now introduce a dataset that satisfies these requirements, derived from the public KernScores [Sapp, 2005] and MAESTRO [Hawthorne et al., 2019] datasets. The chief technical difficulty in constructing this dataset involves constructing the ground-truth alignment annotations; the key insight is that, while constructing a temporal alignment between a score and an *acoustic* performance is difficult, aligning a score to a *symbolic* performance transcript is much easier.[1]

[1]For this reason, works that study score-to-MIDI alignment problem focus on note alignments, which present a challenge even in the symbolic setting [Nakamura et al., 2017].

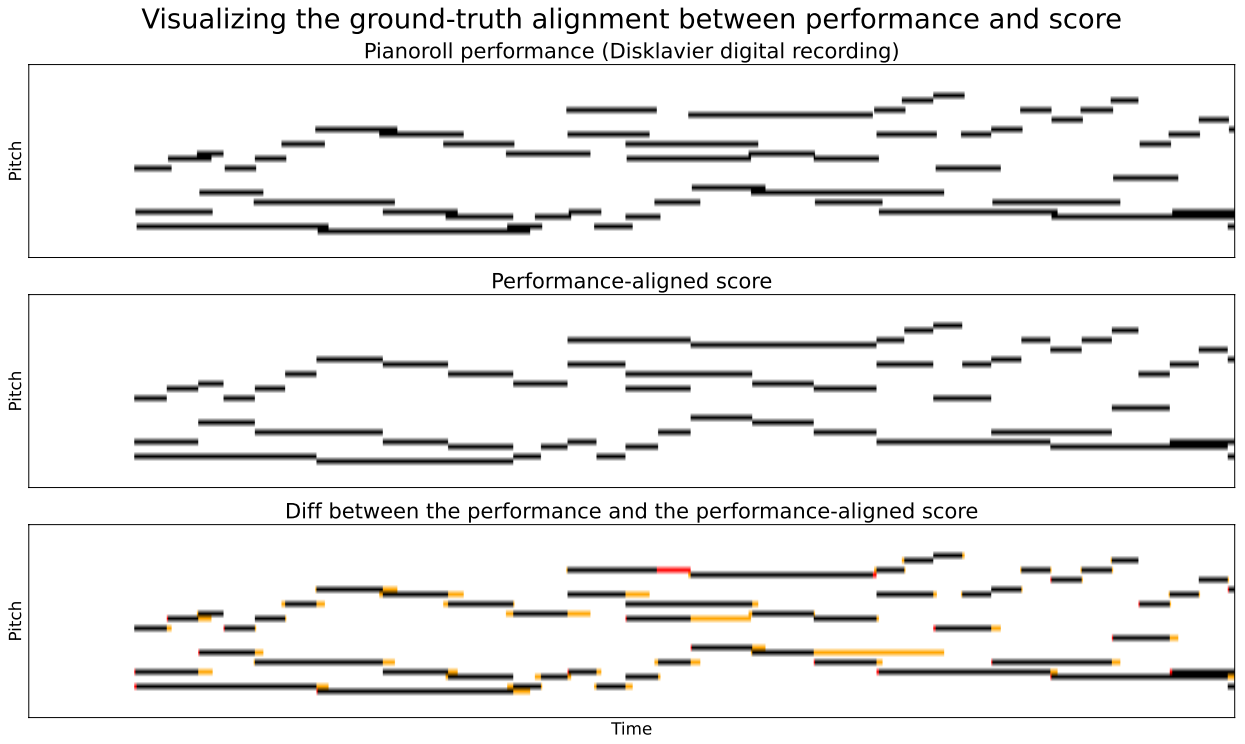


Figure 3.4: To understand the behavior of the ground-truth alignments, we can visually compare the piano-roll performance (top) captured by the Yamaha Disklavier to the performance-aligned score created by warping the score according to the ground-truth alignment (middle). In the comparison plot (bottom) we use red to identify notes that are indicated by the performance-aligned score but not performed and yellow to identify notes that are performed but not indicated by the performance-aligned score. This example visualizes the beginning of a performance of the Bach's Prelude and Fugue in G-sharp minor (BWV 863).

We construct a dataset of ground-truth alignments by cross-referencing a subset of the KernScores collection [Sapp, 2005] of musical scores with a subset of the MAESTRO v2.0.0 dataset [Hawthorne et al., 2019] of piano performances and transcripts. This subset consists of 193 performances, with a total of 444 minutes of audio. The chief technical difficulty in constructing a dataset involves constructing the ground-truth alignment annotations; the key insight to overcoming this is that, while constructing a temporal alignment between a score and an *acoustic* performance is challenging, aligning a score to a *symbolic* performance transcript is relatively straightforward [Nakamura et al., 2017].

In order to quantify an approximation of the equality in Definition 3.3, we must define a metric on the space of symbolic performances. We propose using the L^1 distance.

Definition 3.6. *Given two symbolic performances* $\boldsymbol{sp}_1$ *and* $\boldsymbol{sp}_2$ *of duration* T*, the* L^1 *distance between them is*

$$d_1(\boldsymbol{sp}_1, \boldsymbol{sp}_2) = \frac{1}{T}\int_0^T \|\boldsymbol{sp}_1(t) - \boldsymbol{sp}_2(t)\|_1 \, dt.$$

Intuitively, $\|\mathbf{sp}_1(t) - \mathbf{sp}_2(t)\|_1$ counts the number of differences between $\mathbf{sp}_1$ and $\mathbf{sp}_2$ at an instant in time t, and the integral computes the cumulative average difference. Finding τ that minimizes $d_1(\mathbf{pscore}_\tau, \mathbf{transcript})$, approximating the equality in Definition 3.3, can be achieved by classical dynamic time warping in $O(ST)$ time and space [Sakoe and Chiba, 1978], where S is the length of the score and T is the length of the performance.

We define the best approximation to the ideal alignment given by Definition 3.3 as the solution to the following regularized optimization problem:

$$\begin{aligned} \underset{\tau:[0,S]\to[0,T]}{\text{minimize}} \quad & d_1(\mathbf{pscore}_\tau, \mathbf{transcript}) + \lambda R(\tau), \\ \text{subject to} \quad & \tau(s_1) \le \tau(s_2) \text{ if } s_1 < s_2. \end{aligned} \tag{3.3}$$

The constraints force τ to satisfy the definition of an alignment (Definition 3.1). The distance term d_1 forces τ to approximate the equality in Definition 3.3, and can be interpreted as a measure of compatibility of the alignment τ with the performance. The term $\lambda R(\tau)$ regularizes the optimization towards a uniform tempo, and can be interpreted as a measure of compatibility of τ with the score.

Let $\rho(\tau)$ be the mean inverse-tempo of τ (measured in seconds per beat). We use the variance of the inverse-tempo, $R(\tau)$, to regularize τ towards its mean tempo:

$$R(\tau) \equiv \frac{1}{S}\int_0^S \left(\frac{d\tau(s)}{ds} - \rho(\tau)\right)^2 ds. \tag{3.4}$$

Algorithm 1: Tempo-Regularized Dynamic Time Warping (The Forward Pass)

Input: **score**, **pscore**, $ds, \lambda \in \mathbb{R}$

timings $\leftarrow$ `changepoints`(**score**) {**timings** $\in \mathbb{R}_+^n$}

$m \leftarrow$ `int`(`length`(**pscore**)$/ds$)

prior $\leftarrow$ `length`(**pscore**)/**timings**$[n-1]$

Initialize $\mathbf{cost}_{\text{local}} \leftarrow \mathbf{0}$ {$\mathbf{cost}_{\text{local}} \in \mathbb{R}^{n \times m}$}

for $j = 0$ **to** $n-1$ **do** {Precompute the local cost of aligning $\mathbf{score}_s$ with $\mathbf{audio}_t$.}

  for $k = 0$ **to** $m-1$ **do**

    $\mathbf{cost}_{\text{local}} \leftarrow \|\mathbf{score}(\mathbf{timings}[j]) - \mathbf{pscore}(k \times ds)\|_1$

  end for

end for

Initialize $\mathbf{cost}_{\text{global}} \leftarrow \infty, \mathbf{cost}_{\text{incr}} \leftarrow 0$ {$\mathbf{cost}_{\text{global}} \in \mathbb{R}^{n \times m}$, $\mathbf{cost}_{\text{incr}} \in \mathbb{R}$}

for $k = 0$ **to** $m-1$ **do** {The base case ($j = 0$)}

  tempo $\leftarrow (k \times ds)/\mathbf{timings}[0]$

  $R \leftarrow \lambda \times (\mathbf{tempo} - \mathbf{prior})^2$

  $\mathbf{cost}_{\text{incr}} \leftarrow \mathbf{cost}_{\text{incr}} + \mathbf{cost}_{local}[0, k]$

  $\mathbf{cost}_{\text{global}}[0, k] \leftarrow \mathbf{cost}_{\text{incr}} \times ds + R$

end for

Initialize $\mathbf{cost}_{\text{global}}[0, 0] \leftarrow 0$

for $j = 1$ **to** $n-1$ **do** {Calculate the local tempo-regularized cost matrix.}

  for $k = 1$ **to** $m-1$ **do**

    Initialize $\mathbf{cost}_{\text{incr}} \leftarrow 0$ {$\mathbf{cost}_{\text{incr}} \in \mathbb{R}$}

    for $m = k+1$ **to** 0 **do**

      tempo $\leftarrow ((k-m) \times ds)/\mathbf{timings}[j]$

      $R \leftarrow \lambda \times (\mathbf{tempo} - \mathbf{prior})^2$

      $\mathbf{cost}_{\text{global}}[j, k] \leftarrow \min\{\mathbf{cost}_{\text{global}}[j, k], \mathbf{cost}_{\text{global}}[j-1, m] + \mathbf{cost}_{\text{incr}} \times ds + R\}$

      $\mathbf{cost}_{\text{incr}} \leftarrow \mathbf{cost}_{\text{incr}} + \mathbf{cost}_{local}[j, m]$

    end for

  end for

end for

Return: $\mathbf{cost}_{\text{global}}$

We can write the mean inverse-tempo of an alignment as

$$\rho(\tau) \equiv \frac{1}{S}\int_0^S \frac{d\tau(s)}{ds}\,ds = \frac{\tau(S)-\tau(0)}{S} = \frac{T}{S}. \tag{3.5}$$

Crucially, the mean inverse-tempo is independent of the path τ, which enables us to write a dynamic program–analogous to dynamic time warping–to compute the minimization problem Equation 3.3 in $O(ST^2)$ time using $O(ST)$ space. An forward-backward algorithm that achieves this complexity is presented in Algorithms 1 and 2. The results of this algorithm are visualized for a specific paired score and performance in Figure 3.4.

Like the dynamic time warping algorithm, tempo-regularized dynamic time warping proceeds in two phases: the "forward" pass, in which we compute a cost matrx $\mathbf{cost}_{\text{global}} \in \mathbb{R}^{n\times m}$ of reaching each possible state (i, j) via alignment, and a "backward" pass in, in which we trace the minimum-cost path back through the cost matrix. The crucial distinction from classic dynamic time warping is the introduction of a tempo-regularization term R. Instead of considering only three possible advancements from state (i, j) to states $(i+1, j)$, $(i+1, j+1)$, or $(i, j+1)$, we consider all advancements from state (i, j) to state $(i+1, j+k)$ for each $k > 0$, penalizing the cost of moving to state $(i+1, j+k)$ by the instantaneous tempo implied by making the move from state (i, j) to state $(i+1, j+k)$. Considering advancement to each state $(i+1, j+k)$ results in an $O(ST^2)$ time complexity for tempo-regularized dynamic time-warping, in contrast to the $O(ST)$ time complexity of classic dynamic time warping. Both tempo-regularized dynamic time warping and classic dynamic time warping require $O(ST)$ space.

The dataset introduced in this chapter, along with code for evaluating alignments, is available on GitHub.[2] We also provide the code for generating the dataset, including an implementation of the tempo regularization algorithm. Tempo regularization can be seen as a principled alternative to gully or penalty methods [Raffel and Ellis, 2016] and could be of general interest as a method for regularizing alignments. Finally, while we constructed this

[2] https://github.com/jthickstun/alignment-eval

Algorithm 2: Tempo-Regularized Dynamic Time Warping (The Backward Pass)

Input: **score**, **pscore**, $\mathbf{cost}_{\text{global}} \in \mathbb{R}^{n \times m}$, $ds, \lambda \in \mathbb{R}$
timings $\leftarrow$ `changepoints`(**score**) {**timings** $\in \mathbb{R}_+^n$}
$m \leftarrow$ `int`(`length`(**pscore**)$/ds$)
prior $\leftarrow$ `length`(**pscore**)/**timings**$[n-1]$
$k \leftarrow m - 1$
Initialize **path** $\leftarrow$ `list`()
for $j = n - 1$ **to** 1 **do** {Calculate the min-cost path through the global cost matrix}
  $\mathbf{cost}_{\text{incr}} \leftarrow 0$
  for $m = k + 1$ **to** 0 **do**
    tempo $= ((k - m) \times ds)/$**timings**$[j]$
    $R \leftarrow \lambda \times (\mathbf{tempo} - \mathbf{prior})^2$
    if $\mathbf{cost}_{\text{global}}[j, k] = \mathbf{cost}_{\text{global}}[j - 1, m] + \mathbf{cost}_{\text{incr}} + R$ **then**
      Append (j, k) to **path**
      $k \leftarrow m$
      break {Found the match}
    end if
    $\mathbf{cost}_{\text{incr}} \leftarrow \mathbf{cost}_{\text{incr}} + \|\mathbf{score}(\mathbf{timings}[j]) - \mathbf{pscore}(m \times ds)\|_1$
  end for
end for
Return: **path**

dataset for the purpose of evaluating alignments, it could be repurposed for other tasks that require high-quality alignments between performances and scores.

In Section 3.2, we will immediately put this dataset to work to evaluate the alignment algorithm used to construct MusicNet. This algorithm is a variant of the same dynamic time warping techniques used in this section. But whereas discrete sequences can be robustly aligned using a simple local similarity metric (e.g., the L^1 distance proposed in Definition 3.6)

designing a cost function for comparing the local content of a score with corresponding local content in an audio recording is considerably more nuanced. As we will see in Section 3.2.3, the accuracy of an audio-to-score alignment algorithm can be quite sensitive to the chosen cost function. The dataset constructed here can help us to understand the effectiveness of different cost functions.

3.2 The MusicNet Dataset

The prominent success of deep learning has popularized the end-to-end learning paradigm for supervised classification tasks [Russakovsky et al., 2015]. This approach depends upon large quantities of labeled training data. Many researchers have proposed supervised methods for music transcription that use synthesized training data, including recent MIREX participants [Troxel, 2016, Marolt, 2004, Mita et al., 2017]. While synthesized recordings provide an effectively infinite supply of labeled data, we will see strong evidence in Section 3.3 that models trained on synthetic data do not generalize well to human recordings.

In contrast to, e.g., the vision domain, there has been a notable historical absence of large-scale datasets for music information retrieval tasks. This lack of data is partially attributable to lack of funding for music research programs, and also domain-specific licensing issues surrounding the copyright and distribution of datasets containing music. The situation has prompted researchers to issue the following call to action: "Deep architectures often require a large amount of labeled data for supervised training, a luxury music informatics has never really enjoyed. Given the proven success of supervised methods, MIR would likely benefit a good deal from a concentrated effort in the curation of sharable data in a sustainable manner" [Humphrey et al., 2012].

In this section, we introduce the MusicNet dataset: a freely-licensed, public, curated collection of classical music recordings and scores.[3] The dataset consists of 34 hours of human-verified aligned recordings, containing a total of $1,299,329$ individual labels on seg-

[3] https://zenodo.org/record/5120004.

MusicNet

Minutes	Labels	Recordings	Error Rate
2,045	1,089,540	330	4.0%

Ensemble	Minutes	Labels
Solo Piano	917	435,155
String Quartet	405	220,317
Accompanied Violin	148	97,640
Piano Quartet	73	60,362
Accompanied Cello	63	37,550
String Sextet	48	32,178
Piano Trio	46	28,872
Piano Quintet	25	27,545
Wind Octet	36	26,166
Wind Quintet	43	24,820
Horn Piano Trio	30	18,797
Clarinet-Cello-Piano Trio	25	13,447
Pairs Clarinet-Horn-Bassoon	24	11,972
Clarinet Quintet	26	11,161
Solo Cello	49	10,876
Accompanied Clarinet	20	10,049
Solo Violin	30	8,837
Violin and Harpsichord	16	7,469
Viola Quintet	15	4,113
Solo Flute	8	2,214

Composer	Minutes	Labels
Beethoven	1,085	566,159
Schubert	253	146,576
Brahms	192	131,899
Mozart	156	75,930
Bach	184	62,776
Dvorak	56	31,605
Cambini	43	24,820
Faure	33	22,349
Ravel	27	21,134
Haydn	15	6,292

Instrument	Minutes	Labels
Piano	1,346	633,598
Violin	874	200,467
Cello	800	91,109
Viola	621	89,288
Clarinet	173	24,150
Bassoon	102	14,747
Horn	132	11,327
Oboe	66	8,624
Flute	69	8,310
Harpsichord	16	4,914
String Bass	38	3,006

	Piano	Violin	Cello	Viola	Clarinet	Bassoon	Horn	Oboe	Flute	Bass	Harpsichord
Pitch Classes	83	50	45	45	41	36	40	28	37	43	51

Table 3.1: Summary statistics of the MusicNet dataset. See Section 3.2.2 for a description of the construction of MusicNet and the alignment algorithm used in the labelling process. Section 3.2.4 describes the methodology used to compute the error rate of the labelling process. The statistics reported in this table differ slightly from those reported in the original publication of MusicNet [Thickstun et al., 2017] due to a tabulation error in the earlier work.

ments of these recordings. Table 3.1 summarizes statistics of MusicNet. The existence of MusicNet is made possible by licensing initiatives of the European Archive, the Isabella

Stewart Gardner Museum, Musopen, and various individual artists.

In Section 3.2.1 we discuss other music transcription datasets. In Section 3.2.2, we introduce the alignment algorithm used to construct the MusicNet dataset. Because there are no ground-truth alignments for the MusicNet recordings, the labels constructed via alignment must somehow be validated. We address this question from three perspectives. First, in Section 3.2.3 we investigate the effectiveness of several alignment algorithms using the quantitative methodology and dataset introduced in Section 3.1. Second, in Section 3.2.4 we investigate the quality of the resulting alignments produced via application of an alignment algorithm to the MusicNet scores and recordings. In a sense, this second question is the only important one: so long as the MusicNet alignments are determined to be substantially correct, then the effectiveness of the algorithm used to produce them is mostly irrelevant. But because human judgement is imperfect, both perspectives may be helpful for establishing confidence in the quality of the MusicNet alignments. Finally there is third, operational perspective on the validity of the MusicNet alignments: are they useful as labels for supervising the frame-based music transcription task? We will see in Section 3.3 that training models using the MusicNet labels has proven remarkably effective.

3.2.1 Related Music Transcription Datasets

Aligned training data is typically obtained in one of three ways. First is the use of synthetic performances, derived from a dataset of digital music scores using a commercial music synthesizer, was pioneered by Poliner and Ellis [2007]. This approach is appealing because we can generate vast quantities of precisely-aligned training data, at the cost of a significant distributional shift between the training and testing data. The second approach to dataset construction is by performing music on instruments wired with sensors that record a MIDI transcription (a **pscore**) of the performance as the instrument is played. This approach is largely limited to piano music and specifically the Yamaha Disklavier piano, which is wired to record MIDI, leading to the creation of the MAPS [Emiya et al., 2010] and MAESTRO [Hawthorne et al., 2019] datasets. While these piano-roll transcripts are precisely time-

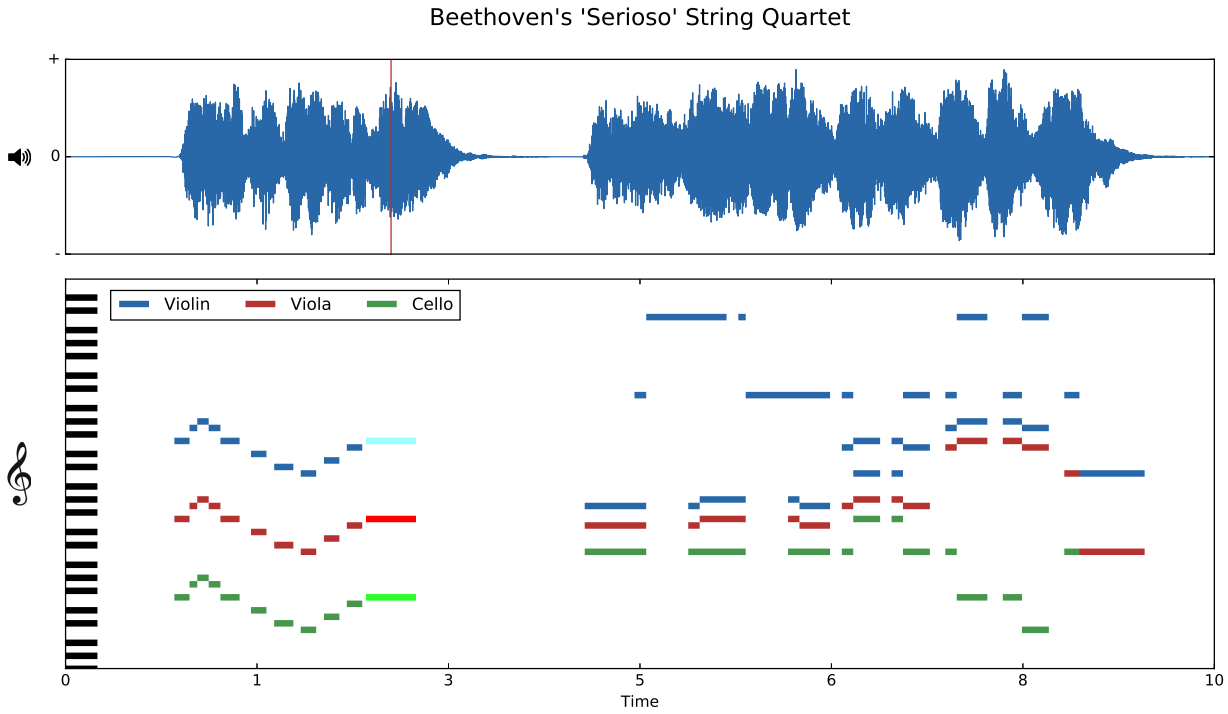


Figure 3.5: An example of the MusicNet labels. Top: an audio recording of Beethoven's String Quartet No. 11 in F minor (Opus 95, 'Serioso'). Bottom: a piano-roll aligned to the audio recording above, i.e., a performance-aligned score (Definition 3.2). The highlighted notes in the piano-roll indicate the active notes at the time indicated by the red tick-mark on the audio wave.

aligned to the performance, they are limited to a particular variety of piano. Furthermore, by directly constructing a pair (**pscore**, **audio**), this means of supervision is missing metrical and other information needed to construct a **score** object; the Disklavier datasets are effective for the frame-based music transcription task but they cannot be directly used to supervise a more general transcription task. The third approach to dataset construction makes use of an alignment algorithm, warping a musical score **score** into a **pscore** corresponding to a given recorded performance of that score. These alignments can be constructed using an alignment algorithm, which is the approach taken for to construct the SyncRWC [Goto et al., 2003] and LakhMIDI [Raffel, 2016] datasets. Alternatively, alignments can be constructed using information supplied by a human annotator, which is the approach taken with the Su

dataset used by MIREX evaluation [Su and Yang, 2015b].

3.2.2 Dataset Construction

The MusicNet labels are constructed from digital MIDI scores, created by enthusiasts and collected from various archives including the Classical Archives (`classicalarchives.com`) Suzuchan's Classic MIDI (`suzumidi.com`) and HarfeSoft (`harfesoft.de`). The methods in this section exploit side information provided by a commercial music synthesizer to produce an alignment between a digital score and a corresponding freely-licensed recording. A recording is labeled with events in the score, associated to times in the performance via the alignment.

Music-to-score alignment is a long-standing problem in the music research and signal processing communities [Raphael, 1999]. Dynamic time warping (DTW) is a classical approach to this problem. An early use of DTW for music alignment is discussed by Orio and Schwarz [2001], whereby a recording is aligned to a crude synthesis of its score, designed to capture some of the structure of an overtone series. The method described in this section aligns recordings to synthesized performances of scores, using side information from a commercial synthesizer. Commercial synthesis was first proposed to assist in construction of alignments by Turetsky and Ellis [2003]. The majority of previous work on alignment focuses on pop music. This is more challenging than aligning classical music because commercial synthesizers do a poor job reproducing the wide variety of vocal and instrumental timbers that appear in modern pop. Furthermore, pop features inharmonic instruments such as drums for which natural metrics on frequency representations–including ℓ^2–are not meaningful. For classical music to score alignment, a variant of the techniques described by Turetsky and Ellis [2003] works robustly. This method is described below; we discuss the evaluation of this procedure and its error rate on MusicNet in the appendix.

In order to align the performance with a score, we will introduce a local similarity metric $\mathbf{cost}_{s,t}$ that compares the local content at location s in the score with local content at time t in the performance. Like in Section 3.1, we say that alignment is *good* if it minimizes average

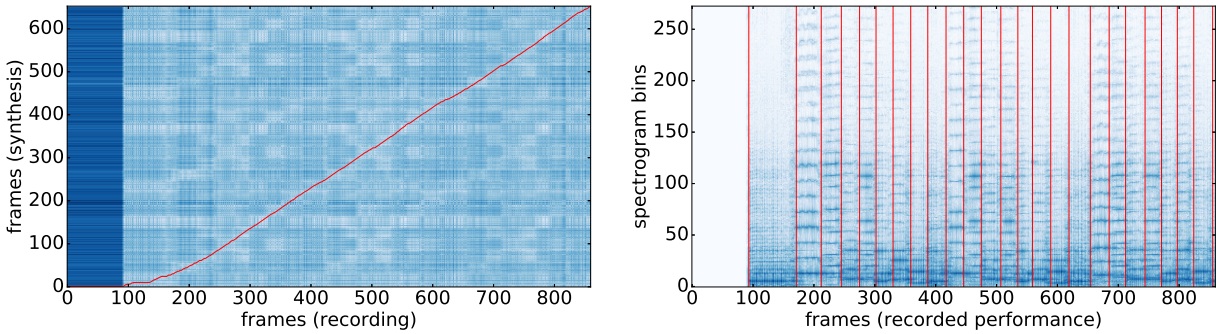


Figure 3.6: (Left) Heatmap visualization of local alignment costs between the synthesized and recorded spectrograms, with the optimal alignment path in red. The block from $x = 0$ to $x = 100$ frames corresponds to silence at the beginning of the recorded performance. The slope of the alignment can be interpreted as an instantaneous tempo ratio between the recorded and synthesized performances. The curvature in the alignment between $x = 100$ and $x = 175$ corresponds to an extension of the first notes by the performer. (Right) Annotation of note onsets on the spectrogram of the recorded performance, determined by the alignment shown on the left.

cost of aligning location in the score with times in the performance. This is comparable to the comparison metric we defined between **pscore**'s given by Definition 3.6, replacing the L^1 norm on bit-vectors with a more general pairwise cost $\mathbf{cost}_{s,t}$.

Definition 3.7. *Given a score* **score**, *a corresponding performance* **audio**, *the c-distance between a* **score** *and* **audio** *under an alignment* $\tau : [0, S) \to [0, T)$ *is given by*

$$d_c(\mathbf{score}, \mathbf{audio}) = \frac{1}{T} \int_0^T \mathbf{cost}_{\tau^{-1}(t),t}\,(\mathbf{score}, \mathbf{audio})\; dt.$$

Where the alignment is non-invertible, we can define $\tau^{-1}(t) \equiv \max\{s : \tau(s) \leq t\}$. Discretizing time in the score and performance into sequences of length n and m respectively, the minimal cost alignment is given by an optimal solution to the following integer program:

$$\begin{aligned} \underset{s \in \mathbb{Z}^n}{\text{minimize}} \quad & \sum_{t=1}^{n} \mathbf{cost}_{s_t,t}(\mathbf{score}, \mathbf{audio}) \\ \text{subject to} \quad & s_0 = 0, s_n = m, \text{ and } s_{t_1} \leq s_{t_2} \text{if } t_1 < t_2. \end{aligned} \tag{3.6}$$

Dynamic time warping gives an exact solution to the problem in $O(mn)$ time and space [Müller, 2007].

The success of dynamic time warping crucially depends on the signal provided by the local comparison metric **cost**. Unlike in Section 3.1, it is not clear how define a local comparison between **score** objects and **audio** objects. Previous work on audio-to-score alignment can be broadly categorized into three groups, each of which address this comparison question by injecting **score** and **audio** into a common normed space. We will denote these injections by maps Ψ and Φ, and write

$$\mathbf{cost}_{s,t}(\mathbf{score}, \mathbf{audio}) = \|\Psi_s(\mathbf{score}) - \Phi_t(\mathbf{audio})\|. \tag{3.7}$$

The most popular approach–and the one adopted to construct MusicNet–maps the score into the space of the performance [Orio and Schwarz, 2001, Turetsky and Ellis, 2003, Soulez et al., 2003]. An alternative approach maps both the score and performance into some third space, commonly a chromogram space that collapses a multi-octave frequency domain into a single 12-tone domain modulo octaves [Dannenberg and Hu, 2003, Izmirli and Dannenberg, 2010, Joder et al., 2013]. Finally, some recent methods consider alignment in score space, taking Φ to be the identity and learning Ψ [Lajugie et al., 2014, 2016].

With reference to the general cost (3.7), we must specify the feature maps Ψ_s, Φ_t, and the norm $\|\cdot\|$. We will use the log-spectrograms as features, with a window size of 2048 samples. We use a stride of 512 samples between features. Hence adjacent feature frames are computed with 75% overlap. For audio sampled at 44.1kHz, this results in a feature representation with $44,100/512 \approx 86$ frames per second. A discussion of these parameter choices can be found in the appendix. The map Φ is computed by a synthetizer: we used Plogue's Sforzando sampler together with Garritan's Personal Orchestra 4 sample library.

For a (pseudo)-metric on spectrograms, we take the L^2 norm $\|\cdot\|_2$ on the low 50 dimensions of the magnitude spectrum. We can roughly interpret the k'th coordinate of the spectrum as the energy associated with the frequency $k \times (22,050/1024) \approx k \times 22.5$Hz, where 22,050Hz

is the Nyquist frequency of a signal sampled at 44.1kHz. The 50 dimension cutoff is chosen empirically: we observe that the resulting alignments are more accurate using a small number of low-frequency bins rather than the full magnitude spectrum. One possible explanation for this observation is that synthesizers do not accurately reproduce the high-frequency features of a musical instrument; by ignoring the high frequencies, we align on a part of the spectrum where the synthesis is most accurate. Such cutoffs have been proposed before, but the cutoff proposed here is aggressive compared to usual settings: for instance, Turetsky and Ellis [2003] propose cutoffs in the 2.5kHz range. The fundamental frequencies of many notes in MusicNet are higher than the $50 \times 22.5\text{Hz} \approx 1\text{kHz}$ cutoff. Nevertheless, we find that all notes align well using only the low-frequency information.

3.2.3 Quantitative Evaluation of Alignment Algorithms

The alignment algorithm introduced in Section 3.2.2 is rather arbitrary, and a reasonable person would question the choices made in constructing the feature maps Φ and Ψ. In this section, we consider four variants of dynamic time-warping algorithm, using different featurizations of the audio:

1. Spectra: (log-)spectrograms.
2. Chroma: (log-)chromagrams [Hu et al., 2003].
3. CQT: constant-Q transforms [Raffel and Ellis, 2016].
4. Truncated: truncated (log-)spectrograms (Section 3.2.2).

We compute features from a synthesized performance of the score, created using PrettyMidi's FluidSynth interface [Raffel and Ellis, 2014], with a hop-size of 512 samples ($\approx$ 12ms). We use librosa's [McFee et al., 2015b] implementation of dynamic time warping to compute an alignment between featurizations. For the Constant-Q featurization, we use Raffel and Ellis's hyper-parameter settings [Raffel and Ellis, 2016], however we do not apply their gully and

	Spectra	Chroma	CQT	Truncated
TimeError (Definition 3.4)	37	35	33	35
NoteError (Equation 3.1)	25	26	23	23
TimeDev (Definition 3.5)	114	76	97	111
NoteDev (Equation 3.2)	92	64	66	87

Table 3.2: The average value of each metric across all performances in the dataset. Values are reported in milliseconds of performance time (lower is better). Spectrogram results exclude 13 outliers with TimeError $>$ 300ms. Truncated spectrogram results exclude 2 outliers with TimeError $>$ 300ms.

penalty constraints as we found that these degrade results for performances that take a very different tempo than the corresponding score.

Computing the note-based metrics discussed in Section 3.1.2 requires a correspondence between notes in the score and notes in the performance. We construct this correspondence from the ground-truth alignment using a heuristic. For each note in the score, we map its onset time to a time t in the performance via the ground-truth alignment. The closest note in the performance transcript with the same pitch is matched to the note in the score. If no note of the same pitch is found within $\pm$100ms of t, the note in the score is considered unmatched and excluded from note-based calculations; using the 100ms threshold, we achieve a 96.7% correspondence between notes on this dataset.

Quantitative results for each featurization of dynamic time warping are summarized in Table 3.2. We find that the chroma and constant-Q alignments have comparable performance. This is consistent with previously reported results on synthetic data [Raffel and Ellis, 2016]. Spectrogram alignments were prone to catastrophic failure; we removed 13 outlier spectrogram alignments from our evaluation, for which TimeError $>$ 300ms. Even on the remaining 182 performances, the spectrogram alignments are substantially worse than chroma or constant-Q. The truncated spectrogram features mitigate the extreme failures seen for for full spectrogram alignment (only truncated spectrogram alignments exhibited TimeError $>$ 300ms) but the truncated spectrogram still exhibits a higher variance than the

	Spectra	Chroma	CQT
TimeError vs. NoteError	.94	.93	.87
TimeDev vs. NoteDev	.85	.83	.85

Table 3.3: Correlation coefficients between temporal alignment metrics and the analogous note-based alignment metrics. There is substantial agreement between these two sets of metrics about whether a performance is well-aligned.

chroma or constant-Q featurizations. We conclude that the truncated features are a reasonable approach, but chroma or constant-Q would be a better choice for future alignment work. A visualization of the behavior of each alignment featurization is presented in Figure 3.7

We can also ask how the temporal metrics introduced in Section 3.1.2 compare to the standard note metrics. Table 3.3 shows that the correlation between the temporal and note-based metrics is high. This assuages concerns that note-based metrics could be overly sensitive to regions of a score with a high density of note onsets: rapid processions of short-duration notes, or regions of high polyphony. While they correlate well, we recommend the new time-based metrics for the temporal alignment task for two reasons. First, temporal metrics eliminate the need for the ad-hoc thresholding heuristic used to construct the correspondence between notes in the score and notes in the performance. Second, the temporal metrics narrowly target the temporal alignment task (see the distinction made between temporal and note-based alignment that we identified in Section 3.1). This removes the temptation to make misleading comparisons with results for the note alignment task, or to shift focus to the note alignment task in order to boost performance on the note-based alignment metrics.

Finally, we are able to gain some insight into the robustness of these results to variations in the underlying ground truth alignments using the recently introduced ASAP dataset [Foscarin et al., 2020]. Like the dataset introduced in Section 3.1.4, ASAP leverages the MAESTRO performance-aligned scores and a score-to-MIDI alignment algorithm to construct a dataset of ground-truth alignments. Whereas we used KernScores as corresponding scores for for MAESTRO performances, ASAP uses MusicXML scores. And whereas we

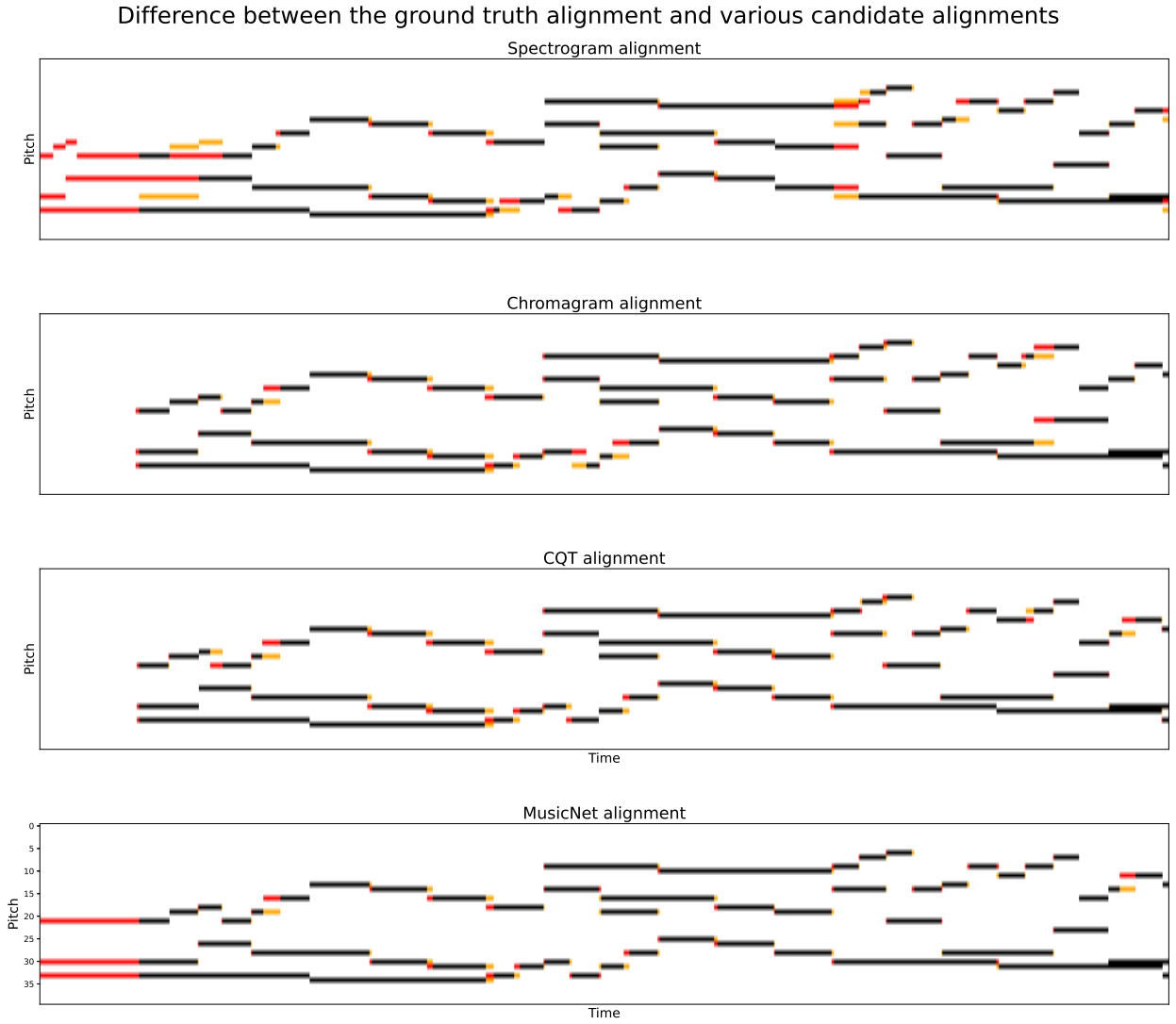


Figure 3.7: Comparing the results of various candidate alignment algorithms to the ground-truth alignment. In each case, red is used to identify notes that are indicated by the candidate alignment algorithm, but not by the ground-truth alignment, and yellow is used to identify notes that are indicated by the ground-truth alignment, but not by the candidate alignment. This example visualizes the beginning of a performance of the Bach's Prelude and Fugue in G-sharp minor (BWV 863).

used tempo-regularized dynamic time warping to construct ground truth alignments, ASAP uses a note-based alignment algorithm to align scores with performances followed by a hybrid system rules and human annotation to post-process these note-based alignments into

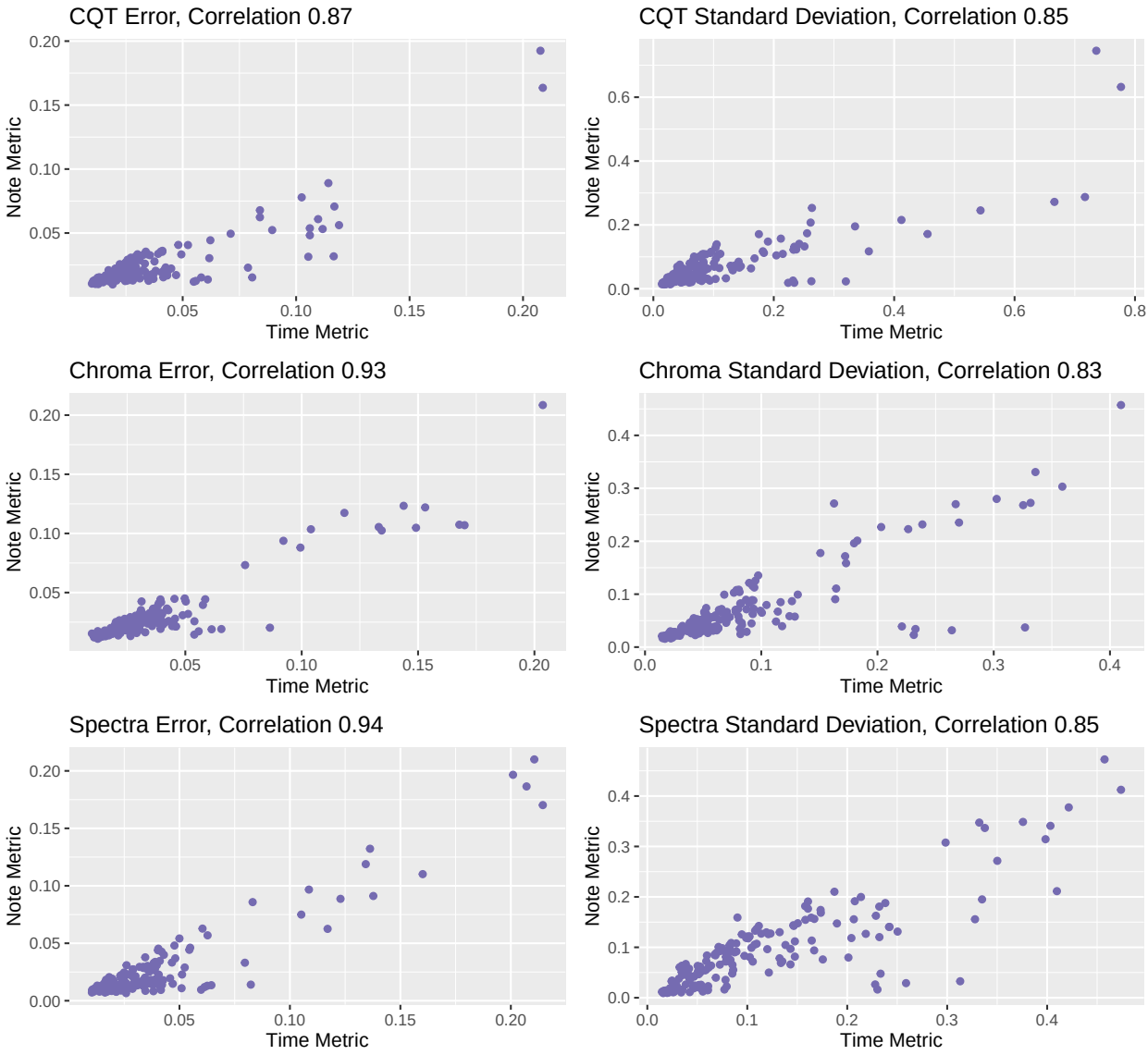


Figure 3.8: A visual illustration of correlation between the new temporal metrics TimeError and TimeDev introduced in Section 3.1.2 with the old note-based metrics NoteError and NoteDev. Each point is one of the 193 performances in the dataset described in Section 3.1.4. For spectrogram results, 13 outliers with TimeError $>$ 300ms are omitted.

temporal alignments. There is considerable overlap between the Bach Preludes and Fugues used to construct these two datasets, and so we are able to repeat our analysis of audio-to-score alignment algorithms using the Bach Preludes and Fugues from the ASAP dataset

	Spectra	Chroma	CQT
TimeError (Definition 3.4)	39	36	31
NoteError (Equation 3.1)	36	33	27
TimeDev (Definition 3.5)	114	69	68
NoteDev (Equation 3.2)	115	67	61

Table 3.4: The average value of each metric across 151 Bach Prelude and Fugue performances in the ASAP dataset of ground truth alignments [Foscarin et al., 2020]. Values are reported in milliseconds of performance time (lower is better). Chroma and Constant-Q results exclude 6 outliers with TimeError $>$ 300ms, and Spectrogram results exclude 15 outliers with TimeError $>$ 300ms.

	Spectra	Chroma	CQT
TimeError vs. NoteError	.97	.97	.93
TimeDev vs. NoteDev	.98	.96	.90

Table 3.5: Correlation coefficients between temporal alignment metrics and the analogous note-based alignment metrics using the ASAP dataset of ground truth alignments [Foscarin et al., 2020].

to better understand the the robustness of our findings to variations in the construction of ground-truth alignments.

Results for the ASAP dataset are presented in Tables 3.4 and 3.5 (compare to Tables 3.2 and 3.3). We find that results using the ASAP alignments as ground-truth are largely consistent with findings using the dataset constructed in Section 3.1.4: spectrogram alignments are worst, chroma and cqt alignments are comparable, and the new temporal metrics correlate strongly with older note-based metrics. Because the procedure for constructing the ASAP dataset alignments is substantially different than the approach taken in Section 3.1.4, we take this as evidence that our evaluation is indeed robust to minor variations in the alignments used as ground-truth.

3.2.4 Validating the MusicNet Labels

We validated the accuracy of the MusicNet alignments using a human listening test. We created synthesized audio renditions of each aligned score-performance pair by mixing a short sine wave into the performance at the onset of each note indicated by the alignment, with a frequency indicated by the score. We can listen to this mix and, if the alignment is correct, the sine tones will exactly overlay the onsets of notes in the original performance; if the alignment is incorrect, the mix will sound arhythmic and dissonant. Two of the authors of MusicNet (John Thickstun and Zaid Harchaoui) listened to sections of each recording in the aligned dataset: the beginning, several random samples of middle, and the end. Both evaluators are trained classical musicians. Recordings that exhibited substantial audible inaccuracies using this methodology were rejected from the final MusicNet dataset.

Among rejected alignments, the primary cause of failure was content differences between the score and the corresponding recording. The most common source of content differences between otherwise correctly paired scores and performances is musical repeats. Classical music often contains markings that indicate a section should be repeated a second time; in classical music performance culture it is often acceptable to ignore these directions, resulting in a content difference between the score and the recording. When the score does not indicate a repeat that occurs in the performance, the alignment typically warps over the entire repeated section, with correct alignments before and after. When the score indicates a repeat that was not taken in the performance, the alignment typically compresses it into very short segment, with correct alignments on either side. We rejected alignments exhibiting either of these failure modes from MusicNet. While there are alignment algorithms that can accommodate repeated structures [Fremerey et al., 2010], we opted to use the simpler alignment algorithm described in Section 3.2.2, at the cost of a rejecting some data that could otherwise have been incorporated into MusicNet.

From the aligned performances that we deemed sufficiently accurate to admit to the dataset, we randomly sampled 30 clips for more careful annotation and analysis. We weighted

the sample to cover a wide range of recordings with various instruments, ensemble sizes, and durations. For each sampled performance, we randomly selected a 30 second clip. Using software transforms, it is possible to slow a recording down to approximately 1/4 speed. Two of the clips were too richly structured and fast to precisely analyze (slowing the signal down any further introduces artifacts that make the signal difficult to interpret). Even in these two rejected samples, the alignments sounded substantially correct. For the other 28 clips, we carefully analyzed the aligned performance mix and annotated every alignment error. The two authors independently checked for errors and found our analyses were nearly identical. In the few cases of disagreement, we used the more pessimistic author's analysis. Across this sample set, we identified a 4.0% error rate.

We did not identify every type of error. Mistaken note onsets are more easily identified than mistaken offsets. Typically the release of one note coincides with the onset of a new note, which implicitly verifies the release. However, release times at the ends of phrases may be less accurate; these inaccuracies would not be covered by our error analysis. We were also likely to miss performance mistakes that maintain the meter of the performance, but for professional recordings such mistakes are rare. For stringed instruments, chords consisting of more than two notes are *rolled*; i.e., they are performed serially from the lowest to the highest note. The temporal alignment protocol described in Section 3.2.2 cannot separate notes that are notated simultaneously in the score (see the distinction between a temporal alignment and a note-based alignment in Section 3.1); a rolled chord is labeled with a single starting time, usually the beginning of the first note in the roll. Therefore, there is some time period at the beginning of a roll where the top notes of the chord are labeled but have not yet occurred in the performance. There are reasonable interpretations of labeling under which these labels would be judged incorrect. On the other hand, if the labels are used to supervise a transcription that notates these onsets concurrently, then ours may be the most desirable labeling.

We can also qualitatively characterize the types of errors we observed. The most common types of errors are anticipations and delays: a single, or small sequence of labels is aligned

to a slightly early or late location in the time series. Another common source of error is missing ornaments and trills: these are short flourishes in a performance are sometimes not annotated in our score data, which results in a missing annotation in the alignment. Finally, there are rare performance errors in the audio recordings and editing errors in the scores.

3.2.5 Alignment Parameter Robustness

The definitions of audio featurization and the alignment cost function were contingent on several parameter choices. These choices were optimized by systematic exploration of the parameter space. We investigated what happens as we vary each parameter and made the choices that gave the best results in our listening tests. Fine-tuning these parameters yields only marginal gains. We find that the quality of alignments improves uniformly with the quality of synthesis. The time-resolution of labels improves uniformly as the stride parameter decreases; minimization of stride is limited by system memory constraints. The other parameters are governed by a tradeoff curve; the optimal choice is determined by balancing desirable outcomes. The Fourier window size is a classic tradeoff between time and frequency resolution. The L^2 norm can be understood as a tradeoff between the extremes of L^1 and L^∞. The behavior of the L^1 norm seems too sensitive to a preponderance of errors due to synthesis quality; these errors add up and overwhelm the signal. On the other hand, the L^∞ norm ignores too much of the signal in the spectrogram. The spectrogram cutoff discussed in Section 3.2.2 is also a tradeoff between accounting for imperfection and variance in the synthesis, and making maximal use of this side-information.

3.3 Music Transcription

There is a large and growing collection of human recordings annotated with labels suitable for supervision of frame-based music transcription models: SyncRWC [Goto et al., 2003], MAPS [Emiya et al., 2010], LakhMIDI [Raffel, 2016], MAESTRO [Hawthorne et al., 2019], as well as the MusicNet dataset presented in Section 3.2. This motivates us to explore models that can make effective use of this data. While the amount of available data is substantial,

it is not infinite. Furthermore, frames of music sampled from an audio recording are more highly correlated than, for example, a pair of images from ImageNet. We therefore focus our attention on models and data augmentation techniques that incorporate prior knowledge of invariances in the problem domain to efficiently use the data.

Recent work shows that end-to-end architectures construct a first-layer feature representation that is qualitatively comparable to classical frequency filterbank transforms such as the STFT or CQT [Dieleman and Schrauwen, 2014]. This leads us to reconsider hand-crafted filterbanks as a low-level representation of musical audio. By replacing the first layer of an end-to-end network with a fixed filterbank transform, we dramatically reduce the number of model parameters and the corresponding risk of overfitting. A filterbank representation has a further advantage over end-to-end learning: its channels are ordered from low to high frequency. This order introduces a topology on the channel axis (the frequency domain) that motivates a convolutional architecture, analogous to how the Euclidean structure of $\mathbb{R}^2$ motivates the classic convolutional network used in computer vision. Constructing a convolutional network on the channels of a filterbank representation yields performance gains from parameter sharing that are not obviously replicable in an end-to-end architecture.

We will now consider a variety of neural architectures as models for frame-based music transcription. In Section 3.3.1 we discuss related work on music transcription. We proceed to formalize the frame-based music transcription task in Section 3.3.2 . In Sections 3.3.3 and 3.3.4 we discuss models for the transcription task that combine classical signal processing techniques with modern neural network architectures, culminating in a frequency-invariant architecture proposed in Section 3.3.5. We present empirical results for these models in Section 3.3.6, trained on the MusicNet dataset evaluated on the MusicNet and Su test sets. Code for all the experiments presented in this section is available on GitHub.[4]

[4] https://github.com/jthickstun/thickstun2018invariances/

3.3.1 Related Work on Transcription

Because access to large aligned datasets was historically limited, older work on music transcription is largely unsupervised; a good survey of older work can be found in Benetos et al. [2013]. To the best of our knowledge, music transcription was first considered as a statistical, supervised learning problem by Poliner and Ellis [2007] using labels obtained from MIDI files to train an SVM on the spectrograms of synthesized recordings of these MIDIs. Subsequent work on supervised transcription improves upon these results in two directions: use of more sophisticated models, and construction of datasets of recorded human performances (as opposed to synthesized data).

The development of models for music transcription conceptually factors into two subproblems: acoustic modeling and time series prediction. We focus on the acoustic modeling problem, as introduced by Poliner and Ellis [2007], making a strong assumption of conditional independence on the prediction of each note at each time step. Recent developments in this area model the acoustics using deep neural networks [Nam et al., 2011, Trabelsi et al., 2018] or convolutional neural networks [Kelz et al., 2016, Bittner et al., 2017, Pons and Serra, 2017]. Some recent work explores hybrid models that combine a deep or convolutional acoustic model with a recurrent time-series model to jointly estimate transcriptions [Sigtia et al., 2015, 2016].

Choosing an appropriate model for a supervised learning problem requires consideration of both the structure of the problem and the available data. A biased model can compensate for a smaller dataset at the risk of making overly powerful assumptions about the problem structure. The frequency-invariance ideas presented in Section 3.3.5 are anticipated by Sigtia et al. [2016], Bittner et al. [2017], and Pons and Serra [2017]. Our contribution is to demonstrate that this class of models represents a good bias-variance tradeoff for current datasets. Our dataset augmentation techniques are inspired by analogous transformations introduced by the vision community for image classification [Simard et al., 2003, Krizhevsky et al., 2012] and extensions of these ideas to audio [McFee et al., 2015a].

Subsequent to the work presented in this section, we observe several emerging trends. The release of the large-scale MAESTRO piano dataset [Hawthorne et al., 2019] has lead to strong empirical transcription results of piano music recorded in a controlled studio environment, supporting the argument that the primary bottleneck to general-purpose transcription is access to large quantities of labeled data. Ongoing work uses MusicNet to study the performance of transcription models for specific instrument types [Pedersoli et al., 2020], and explore uses of the dataset beyond frame-based transcription, including note-based transcription [Cheuk et al., 2020b] and instrument recognition tasks [Hung et al., 2019]. Finally, there is recent work toward eliminating independence assumptions between note predictions for piano-roll transcription by post-processing a predicted piano-roll using a language model [Ycart et al., 2019a] or GAN [Ycart et al., 2019b] as a prior.

3.3.2 A Frame-Based Transcription Task

Given an audio frame of length $2w$, $\mathbf{x} = \mathbf{audio}([t-w, t+w]) \in [-1,1]^{44{,}100 \times 2w} = \mathcal{X}$, we seek to predict the notes present at the midpoint of $\mathbf{audio}(t)$, given by an aligned piano-roll score, which we encode as a binary label vector $\mathbf{y} = \mathbf{pscore}_\tau(t) \in \{0,1\}^N$ (Definition 3.2). The temporal window $[t-w, t+w]$ should be thought of as a local context for making predictions about the instantaneous content of the audio signal at time t. We model this prediction task by learning a feature map $f_\theta : \mathcal{X} \to \mathcal{H}$ (with learned parameters θ) to a latent representation space $\mathcal{H}$, along with a multivariate linear regression to estimate $\hat{\mathbf{y}} = f_\theta(\mathbf{x})$ given the learned representation $f_\theta(\mathbf{x})$. We supervise learning of the parameters θ by minimizing the mean-squared error of our predictions $\hat{\mathbf{y}}$ over a training dataset $\mathcal{X}_{\text{train}}$:

$$\theta^* = \min_\theta \sum_{\mathbf{x} \in \mathcal{X}_{\text{train}}} \tfrac{1}{2} \|\mathbf{y} - f_\theta(\mathbf{x})\|^2. \tag{3.8}$$

We preprocess $\mathbf{x}$ by the normalization $\mathbf{x} \mapsto \mathbf{x}/\|\mathbf{x}\|_2$; this can be interpreted physically as normalizing the audible volume of each frame. The first layer of every network we consider is a strided convolution with a $4,096$-sample receptive field and a 512-sample stride. We use

a frame length of $w = 16,384$ samples, resulting in $25 = (16,384 - 4,096)/512$ regions per frame. The 16,384-sample frame size reflects a tradeoff between a shorter frame, which could miss important context for the classification task, and a longer frame, which has diminishing returns at computational cost. Very long frames grow the number of parameters in the model, creating additional risk of overfitting. The 512-sample stride is subject to a similar tradeoff.

We augment our training dataset by stretching or shrinking our input audio with linear interpolation. This corresponds to a pitch-shift in the frequency domain. For small shifts (± 5 semitones or less) the transformed audio sounds natural to the human ear. Randomly shifting each data points in a minibatch by an integral number of semitones in the range $[-5, 5]$ augments the dataset by an order of magnitude. The translational nature of this augmentation reinforces the architectural structure of the frequency-invariant networks. In addition to an integral semitone shift, we also apply a continuous shift to each data point in the range $[-.1, .1]$. This makes the models more robust to tuning variation between recordings.

3.3.3 Learning from Spectrograms

The simplest model we consider is a two layer neural network. For each region of the layer-one convolution we construct a filterbank representation of the input, creating a spectrogram representation $\mathcal{H}$ at layer two. We perform linear classification on $\log \mathcal{H}$: the pointwise logarithm of the spectrogram. We consider several variants on the choice of filterbank below, as well as an end-to-end architecture where the filters are learned from data.

When we parameterize a network, we must choose the width of the set of weights in the bottom layer. This width is called the receptive field in the vision community; in the music community it is called the window size. Traditional frequency analyses, including spectrograms, are highly sensitive to the window size. Windows must be long enough to capture relevant information, but not so long that they lose temporal resolution; this is an instance of the classical time-frequency tradeoff [Heisenberg, 1927]. Furthermore, windowed

frequency analysis is subject to boundary effects, known as spectral leakage. Classical signal processing attempts to dampen these effects with predefined window functions, which apply a mask that attenuates the signal at the boundaries [Rabiner and Schafer, 2007].

We consider several variants on the choice of filterbank below, as well as an end-to-end architecture where the filters are learned from data.

(Short-time Fourier transform) This is the classical filterbank consisting of Fourier coefficient magnitudes. We truncate the magnitude spectrum at 6kHz because we find that frequencies above this cutoff do not meaningfully improve classification accuracy.

(Log-spaced filterbank) This filterbank consists of 512 sine and cosine filters with log-spaced frequencies ranging from 50Hz to 6kHz. For each filter pair $\mathbf{w}_{\text{k,sin}}, \mathbf{w}_{\text{k,cos}}$, we compute inner products with the input region $\mathbf{x}_t \in [-1, 1]^{4,096}$ and sum the square of these values, analogous to the STFT:

$$\text{filter}_k = (\mathbf{w}_{k,\text{sin}}^T \mathbf{x}_t)^2 + (\mathbf{w}_{k,\text{cos}}^T \mathbf{x}_t)^2. \tag{3.9}$$

(Windowed filterbank) Here we apply the cosine window $1 - \cos(t)$ to each filter in our filterbank. This combats the spectral leakage phenomenon caused by boundary effects introduced by the finite-window frequency analysis [Rabiner and Schafer, 2007]. We will examine the effects of windowing on both the STFT and log-spaced filterbank.

(Learned filterbank) In this architecture, the filter coefficients $\mathbf{w}_k$ are learned as parameters in the classifier optimization (see Section 3.3.4).

3.3.4 Learning Features of Music from Scratch

Spectrogram representations are closely related to a two-layer ReLU network. If $\mathbf{x} = (x_1, \ldots, x_t)$ denotes a segment of an audio signal of length t then we can define

$$\text{Spec}_k(\mathbf{x}) \equiv \left| \sum_{s=0}^{t-1} e^{-2\pi iks/t} \mathbf{x}_s \right|^2 = \left(\sum_{s=0}^{t-1} \cos(2\pi ks/t) \mathbf{x}_s \right)^2 + \left(\sum_{s=0}^{t-1} \sin(2\pi ks/t) \mathbf{x}_s \right)^2. \tag{3.10}$$

These features are not precisely learnable by a two-layer ReLU network. But recall that

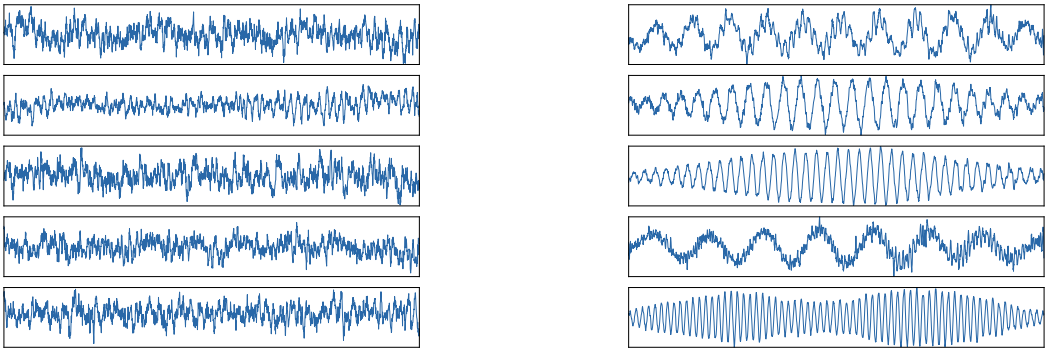


Figure 3.9: (Left) Features learned by a 2-layer ReLU network trained on small monophonic subset of MusicNet. (Right) Features learned by the same network, trained on the full MusicNet dataset.

$|x| = \max(0, x) + \max(0, -x)$ and if we take weight vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^T$ with $u_s = \cos(2\pi ks/t)$ and $v_s = \sin(2\pi ks/t)$ then the ReLU network can learn

$$f_{k,\cos}(\mathbf{x}) + f_{k,\sin}(\mathbf{x}) \equiv |\mathbf{u}^T\mathbf{x}| + |\mathbf{v}^T\mathbf{x}| = \left|\sum_{s=0}^{t-1} \cos(2\pi ks/t)\mathbf{x}_s\right| + \left|\sum_{s=0}^{t-1} \sin(2\pi ks/t)\mathbf{x}_s\right|. \quad (3.11)$$

We call this family of features a ReLUgram and observe that it has a similar form to the spectrogram; we merely replace the $x \mapsto x^2$ non-linearity of the spectrogram with $x \mapsto |x|$.

The ReLUgram features achieve similar performance to spectrograms on the transcription task (see Section 3.3.6) and the weights of the learned features visually approximate a filterbank of sinusoids: see Figure 3.9. Moreover, the ReLUgram model learns window functions: the magnitude of the learned weights attenuates at the boundaries. If we parameterize this model with a large window size, the model learns that distant information is irrelevant to local prediction. We therefore focus on two window sizes: 2048 samples, which captures the local content of the signal, and 16,384 samples, which is sufficient to capture almost all relevant context.

The size of MusicNet is essential to achieving the results in Figure 3.9. In Figure 3.9 (Left) we optimize a two-layer ReLU network on a small subset of MusicNet consisting of $65,000$ monophonic data points. While these features do exhibit dominant frequencies, the signal is

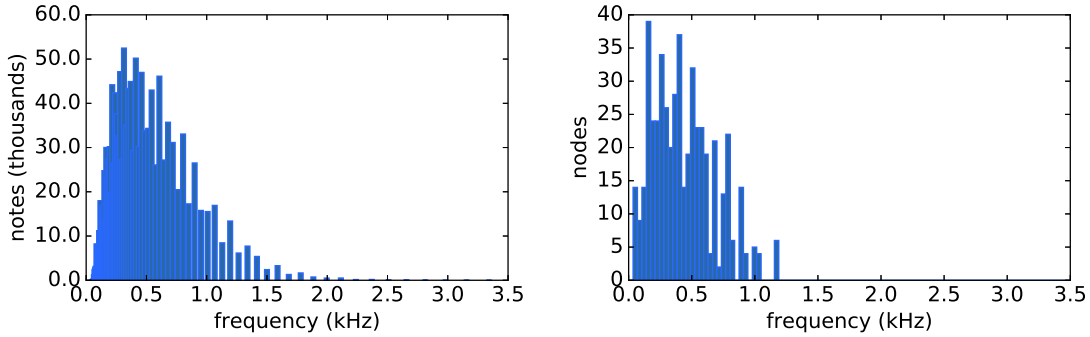


Figure 3.10: (Left) The frequency distribution of notes in MusicNet. (Right) The frequency distribution of learned nodes in a 500-node, two-layer ReLU network.

quite noisy. Comparable noisy frequency selective features were recovered by Dieleman and Schrauwen [2014]; see their Figure 3. We can recover clean features on a small dataset using regularization, e.g., a heavy L^2 penalty term, but this destroys classification performance; regularizing with dropout poses a similar tradeoff. By contrast, Figure 3.9 (Right) shows weights learned by an unregularized two-layer network trained on the full MusicNet dataset. The models described in this paper do not overfit to MusicNet and are trained without L^2 regularization.

A spectrogram of length n is computed from $2n$ samples, so a linear 1024-point spectrogram model is directly comparable to a multi-layer perceptron with 2048 raw samples. In Section 3.3.6, we will see that learned features modestly outperform spectrograms for comparable window sizes. The discussion of windowing partially explains this. Figure 3.10 suggests a second reason. Equation 3.10 shows that the spectrogram features can be interpreted as the magnitude of the signal's inner product with sine waves of linearly spaced frequencies. In contrast, the proposed networks learn weights with frequencies distributed similarly to the distribution of notes in MusicNet. This gives the network higher resolution in the most critical frequency regions.

A natural extension of the two layer networks discussed above is a three layer network with a fully connected layer interposed between the layer-one convolutions and the linear output

layer. If we interpret the output of layer one as a spectrogram, then this intermediate layer captures non-linear relationships between features of this spectrogram. A filter in layer two might be sensitive to a particular chord, for example, or to a certain progression of notes. In this vein, it is possible to build much deeper models on top of either the raw audio or filterbank representation. These ideas are explored by Trabelsi et al. [2018]. However, as we see in Section 3.3.6 (Table 3.6), simply building a deeper architecture does not directly improve performance for the note classification task.

3.3.5 Frequency-Invariant Networks

We propose a *frequency-invariant* architecture, which introduces an inductive bias towards translational symmetries in the frequency content of an audio signal. This architecture is motivated by analogous work that exploits translational symmetries in images for object recognition tasks [LeCun et al., 1998, Gens and Domingos, 2014, Benton et al., 2020]. The frequency-invariant network is built on top of a filterbank, with two learned representational layers. See Figure 3.11 for a visual schematic and description of this architecture. A hand-crafted layer-one filterbank is crucial to support frequency-invariant convolutions at layer two. Because the layer-one filters are frequency-ordered, the layer-two filters can learn patterns that are invariant to translations in frequency. Consider, for example, a major triad chord. This pattern is preserved under linear translations in log-frequency space. In the frequency-invariant architecture, a single filter consisting of only 128 parameters could be sensitive to major triads rooted at arbitrary frequencies. In contrast, a fully connected three layer network would require distinct filters to identify this chord pattern at each location in the log-frequency spectrum.

The preceding arguments about music-theoretic concepts like intervals and triads are contingent on the use of a log-frequency filterbank for layer one. If we used a linear filterbank (for example, the STFT) then a linear shift in the frequency domain would correspond to a non-linear shift in the musical relationships between notes (low notes would translate further than high notes) due to the human ear's logarithmic perception of frequency. On the other

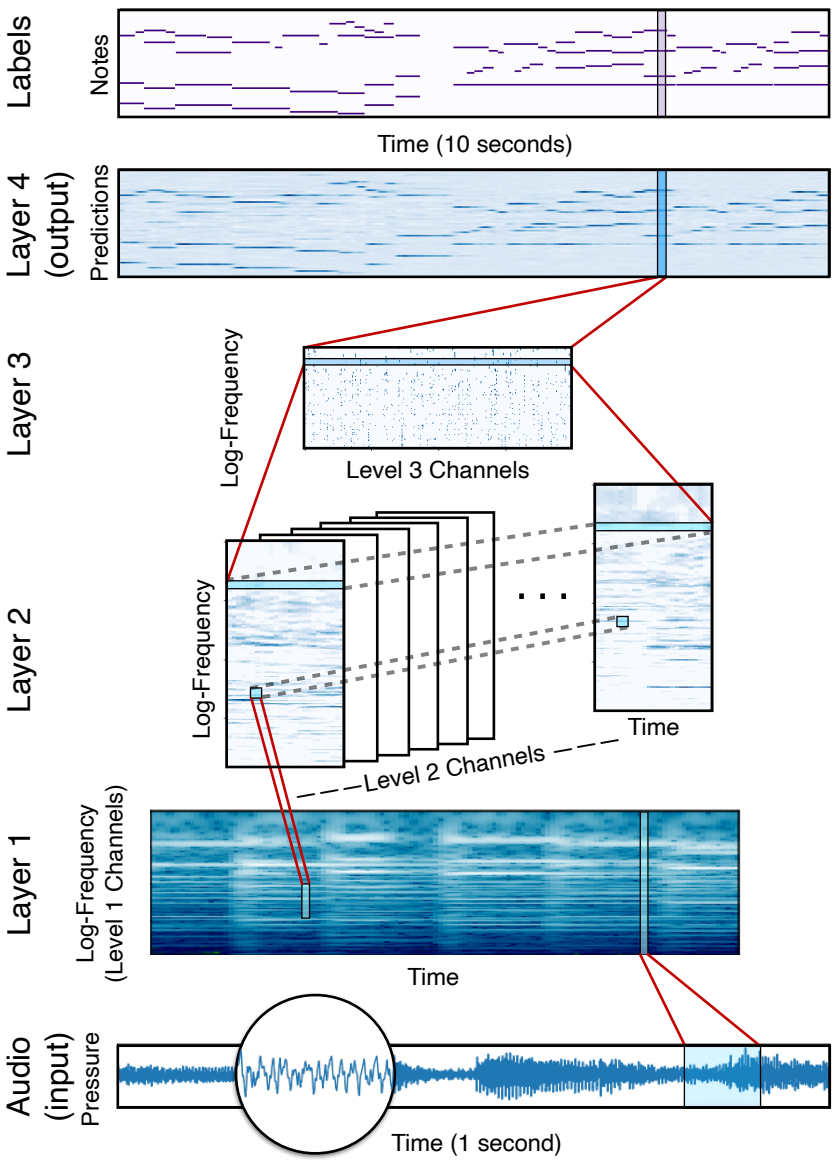


Figure 3.11: A frequency-invariant network for note classification: this is the TKH1 architecture evaluated in Tables 3.6, 3.7, and 3.8 . Audio input maps to Layer 1 according to the log-spaced, cosine-windowed filterbank. Layer 1 maps to Layer 2 by convolving a set of 128×1 learned filters along the log-frequency axis at each fixed time location. Layer 2 maps to Layer 3 by convolving again along the log-frequency axis, this time with a set of filters of height 1 that fully connect along the time and channel axes of Layer 2. Notes are predicted at Layer 4 by linear classification on the learned representation $\mathcal{H}$ given at Layer 3. Using a pre-defined filterbank at Layer 1 is essential; compare the "frequency invariant" network (filterbank) to "channel convolution" network (learned weights in Layer 1) in Table 3.6. Sensitivity of transcription results to the choice of filterbank is presented in Table 3.6, and this question is studied further by Cheuk et al. [2020a].

hand, the physics of audio (for example, overtones) exist on a linear scale. A model that uses a linear filterbank can exploit translation invariances in the physics. We find empirically that log-scale invariance yields greater performance gains for note classification than linear-scale invariance.

Finally we consider a network that has the same architecture as the frequency-invariant network, but instead treats the weights in the layer-one filterbank as optimization parameters (i.e. the filterbank is learned). Because the weights in layer one are learned from a random initialization, it is not clear that the learned filters will be ordered by frequency or that their output channels will exhibit any topological structure. However, because the layer-two convolutions in this architecture are designed to exploit local topological structure in the channels, we might hope that end-to-end parameter optimization would find a good topological structure for this space, a kind of self-organizing map [Kohonen, 1990].

3.3.6 Evaluating Transcription Models

For MusicNet, we evaluate results (Table 3.6) using *average precision*: the area under the precision-recall curve of a classifier, as we vary its decision threshold for prediction [Salton and McGill, 1984] (higher is better). This is a popular evaluation metric in the computer vision community [Everingham et al., 2010] that characterizes the performance of a classifier over the full spectrum prediction thresholds. We also report the Accuracy (higher is better) and Error (lower is better) proposed as metrics for music transcription by Poliner and Ellis [2007] for a specific classification threshold 0.4. The Su dataset [Su and Yang, 2015b] results presented in Tables 3.7 and 3.8 were calculated by the MIREX organization; our THK1 submission is the frequency-invariant model described in Section 3.3.5 using a classification threshold of 0.4.

Our best frequency-invariant network achieves 77.3% average precision on MusicNet, outperforming the previous state of the art reported by Trabelsi et al. [2018], and popular commercial software Celemony [Celemony]. Furthermore, we find that the frequency-invariant architecture with a handcrafted filterbank outperforms end-to-end models trained on this

Model	Average Precision	Accuracy	Error
filterbanks			
STFT (no compress)	40.4	15.9	.860
STFT	60.4	36.2	.681
log frequencies	62.7	39.8	.646
cosine windows	66.1	38.7	.637
log + windows	66.7	38.9	.633
three layer network	73.8	51.4	.541
frequency-invariant (THK1)	**77.3**	**55.3**	**.474**
end-to-end			
learned filterbank	67.8	48.9	.634
three layer network	70.8	48.8	.558
deep complex [Trabelsi et al., 2018]	72.9	-	-
channel convolution	73.3	50.4	.531
commercial software			
Melodyne [Celemony]	58.8	41.0	.760

Table 3.6: Average Precision, Accuracy, and Error for each of the models discussed in this section, evaluated using the test set proposed by Thickstun et al. [2017]. Average Precision is computed by scikit-learn [Pedregosa et al., 2011]; Accuracy and Error are computed using mir_eval [Raffel et al., 2014]. The Accuracy and Error scores are assume a global prediction threshold of 0.4.

dataset [Trabelsi et al., 2018]. Performance is highly sensitive to layer one. While a naive filterbank performs poorly, log-spaced, cosine-windowed filters approach the performance of a learned filterbank (see Table 3.6; compare "log + windows" to "learned filterbank"). Also, note the importance of the compressive non-linearity used for all models except the one marked "no compress."

The end-to-end architectures evaluated in Table 3.6 significantly underperform the corresponding architectures with a handcrafted layer-one filterbank. Because filterbanks are essentially realizable in an end-to-end architecture (see Section 3.3.4) we infer that these optimizations have converged to either a local minimum or a saddle point. In particular, the learned layer-one weights of the channel convolution model exhibit some structure between neighboring filters, but not the global frequency ordering exhibited by the hand-crafted

Model	Precision	Recall	Accuracy	Error
MIREX 2009 Dataset				
THK1	**82.2**	78.9	**72.0**	**.316**
KD1	72.4	**81.1**	66.9	.419
MHMTM1	72.7	78.2	65.5	.441
WCS1	64.0	80.6	59.3	.569
ZCY2	62.7	56.2	50.6	.601
Su Dataset				
THK1	**70.1**	**54.6**	**51.0**	**.529**
KD1	45.9	45.0	38.1	.745
WCS1	63.6	39.7	35.7	.700
MHMTM1	61.2	36.8	35.2	.676
ZCY2	40.9	28.2	26.2	.799

Table 3.7: MIREX 2017 results for the top 5 participants by accuracy in each category of the Multiple Fundamental Frequency Estimation challenge. THK1 is the wide layer 3 frequency-invariant model described in this document.

filterbanks.

We provide further evaluation of results for the frequency-invariant model obtained in the MIREX 2017 Fundamental Frequency Estimation challenge. This is a yearly challenge, in which entrants submit models that are then evaluated on two, privately held test-sets: the MIREX 2009 dataset and the aforementioned Su dataset. Our frequency-invariant model is competitive with state-of-the art on the MIREX dataset and far outperforms all other competitors on the Su dataset. Table 3.7 summarizes the results for 2017, the year that we submitted our entry. The next-best result on the Su dataset overall is the echo-state network [Steiner et al., 2020], reported in the 2019 contest (see Table 3.8) which achieves 45.5 accuracy on the Su test set. By comparison, the frequency-invariant network achieves 51.0 accuracy on the same test set in the 2017 contest.

While it is difficult to conclusively attribute empirical performance of models, we see in the extended results (Table 3.8) that both the frequency-invariant network (THK1) and echo-state networks (SBJ) were trained using MusicNet. The AR2 model, which is the most successful model for this challenge not trained on MusicNet, was also trained using a

dataset of human performances: the URMP dataset [Li et al., 2019]. The note templates used for the CB entries are extracted from the MAPS piano dataset [Benetos and Dixon, 2012], and the weak results here may reflect a distributional shift between the statistics of piano acoustics and the more diverse stringed instruments present in the Su dataset. Other supervised models submitted to this contest, including the MHMTM1 model [Mita et al., 2017] reported in Table 3.7 and DT1 [Troxel, 2016], MM1 [Marolt, 2004] are trained on synthesized datasets. Unsupervised methods also appear fundamentally limited, the best method in this class being SY1 [Su and Yang, 2015a].

3.4 Conclusion

We observed in Section 3.3.6 that current models supervised using synthesized data or piano datasets, as well as unsupervised models, do not generalize well to the Su chamber music dataset. This underscores the importance of training on datasets of in-domain human performances, which in turn motivates the work on alignments in Section 3.1 that expands our access to training data beyond synthetic datasets and Yamaha Disklavier recordings. However, even with robust automatic alignment algorithms, we may struggle to create in-domain, aligned datasets at a sufficiently large scale to fully solve the music transcription problem. This in turn motivates an interest in unsupervised generative models, which can be trained on large-scale quantities of unlabeled data. The unsupervised paradigm introduces a different set of challenges: how can we re-purpose an generic unsupervised model for a specific task using a limited amount of supervision? In Chapter 4, we develop unsupervised generative models of musical scores, motivated by the possible use of such models as priors for music transcription. In Chapter 5, we develop methodology for adapting generative models to conditional generation tasks.

Model	Year	Precision	Recall	Accuracy.	Error	Dataset
THK1	2017	**70.1**	54.6	**51.0**	**0.529**	MusicNet
SBJ4	2019	65.2	48.3	45.5	0.578	MusicNet
SBJ3	2019	64.7	46.5	43.9	0.594	MusicNet
SBJ2	2019	68.9	43.6	41.9	0.604	MusicNet
AR2	2019	63.1	42.9	40.6	0.628	URMP
SBJ1	2019	68.3	39.9	38.5	0.637	MusicNet
SY1	2015	51.6	**62.6**	38.5	0.779	Unsupervised
KD1	2017	45.9	45	38.1	0.745	Unknown
KD2	2017	45.9	45	38.1	0.745	Unknown
SY2	2015	50	62	37.5	0.795	Unsupervised
SY3	2015	53.5	56.7	36.9	0.742	Unsupervised
SY4	2015	53.2	55.6	36.4	0.732	Unsupervised
WCS1	2017	63.6	39.7	35.7	0.7	Unknown
BW1	2015	61.4	48	35.6	0.684	MAPS, RWC
MHMTM1	2017	61.2	36.8	35.2	0.676	Synthesized
MM1	2016	58.1	32	31	0.714	Synthesized
PR1	2017	46	33.2	30	0.77	Unsupervised
ZCY2	2017	40.9	28.2	26.2	0.799	Unknown
CB1	2015	61.7	31.5	25.9	0.736	MAPS, RWC
CB2	2015	58.5	29.9	24	0.757	MAPS, RWC
CB1	2019	61.7	23.6	23.4	0.773	MAPS, RWC
CB1	2018	61.7	23.6	23.4	0.773	MAPS, RWC
CB1	2017	61.7	23.6	23.4	0.773	MAPS, RWC
CB1	2016	61.7	23.6	23.4	0.773	MAPS, RWC
CB2	2019	58.6	22.4	22.1	0.788	MAPS, RWC
CB2	2018	58.6	22.4	22.1	0.788	MAPS, RWC
CB2	2017	58.6	22.4	22.1	0.788	MAPS, RWC
CB2	2016	58.5	22.4	22.1	0.788	MAPS, RWC
PRGR2	2017	26.3	10.2	9.6	1.023	Unsupervised
PRGR1	2017	23.8	6	6.2	0.926	Unsupervised
MHMTM2	2017	28.7	5.7	5.7	0.945	Synthesized
HH2	2019	6.5	6.2	5.3	1.086	Unknown
DT1	2016	7.1	1.6	1.6	0.986	Synthesized

Table 3.8: Full MIREX Su dataset results for the Multiple Fundamental Frequency Estimation challenge since the introduction of the Su dataset in 2015. THK1 is the wide layer 3 frequency-invariant model described in this document. There were no participants in this challenge in 2020.

Chapter 4

MODELING SYMBOLIC REPRESENTATIONS OF MUSIC

Like written language and speech, classical music has two canonical representations: musical scores and audio performances. And like natural language, the written form of music is amenable to symbolic digitization. In this chapter, we model the structure of symbolic representations of music. We consider two symbolic modeling tasks: unsupervised generative modeling (Section 4.2) and supervised composer attribution (Section 4.3). Looking back to Chapter 3 provides motivation for learning a generative model over symbolic music: a generative model could be used as a Bayesian prior over the structure of musical scores, breaking the strong statistical independence assumptions made by the frame-based formulation of the music transcription task. And looking forward to Chapter 5, we can motivate composer attribution models as Bayesian likelihoods that could be used to control sampling from a generative model based on a desired musical style.

Unlike written language, there is no canonical digital representation of written music. Whereas written language is comprised of linear sequences of characters from some fixed alphabet, musical scores are a visual medium with an open-ended vocabulary of symbols laid out across two dimensions. This complexity has lead to a proliferation of digital formats for encoding musical scores, none of which can claim to be the canonical encoding of written music. These encodings are not fully isomorphic, in the sense that we cannot construct lossless translations from one digital encoding to another. When we work with symbolic music, we must therefore make somewhat arbitrary choices in our digital encoding. In Section 4.1 we discuss a variety of approaches to encoding music. The modeling work in later sections of this Chapter uses data derived from the KernScores dataset, which is encoded using the Humdrum format described in Section 4.1.2.

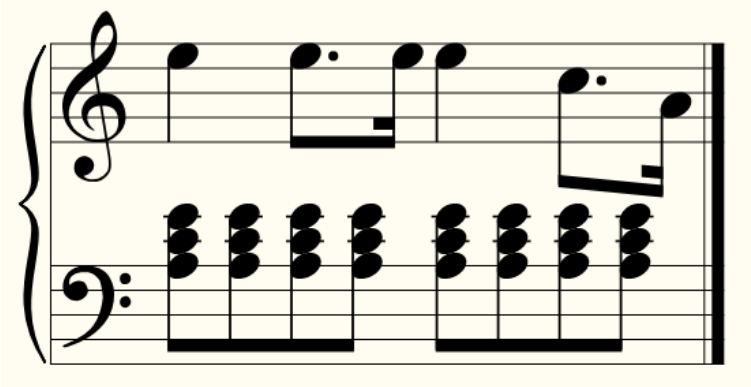


Figure 4.1: Mozart's piano sonata number 8 in A minor, movement 1, from measure 1.

In Section 4.2, we develop a generative model for a digital encoding of musical scores derived from KernScores [Sapp, 2005]. Our aim is to construct an autoregressive density estimator of the probability distribution over this data. Using the same data, in Section 4.3 we develop a supervised classification model for attributing composers to musical scores. Because digital scores are a limited resource, we are interested in constructing models that incorporate prior knowledge of music in order to bias learning towards structurally plausible solutions. For the classification task, the success of these methods is relatively easy to evaluate. But for generative modeling evaluation is subtle, resulting in a more nuanced discussion of generalization and quantitative metrics for the generative task.

4.1 Symbolic Encodings of Music

In this section we identify three archetypes of digital encodings for classical music scores, along with concrete examples of digital formats that illustrate these archetypes. Throughout the section, we use the fragment of the score pictured in Figure 4.1 as a concrete example to illustrate the behavior of each encoding. In Section 4.1.1 we consider sequential encodings, which are most similar to the canonical encoding of language. The MIDI format is a representative example of sequential encoding. In Section 4.1.2 we consider tensor encodings, which are most similar to the canonical encoding of images. The Humdrum format can be interpreted as a (serialized) example of a tensor encoding. Finally, in Section 4.1.3 we con-

sider hierarchical encodings, for which the MusicXml format is an (serialized) representative example.

The diversity of encodings for musical scores creates ongoing challenges for the music modeling community. Each encoding uses its own vocabulary, and these vocabularies often cannot be perfectly translated. This makes it difficult to compare modeling results across encodings; each encoding of music is effectively a unique data domain. In Section 4.2 we will attempt to address some of these challenges, but other aspects of the problem seem insurmountable. If one generative model's vocabulary includes meter but not dynamics, and another includes dynamics but not meter, how do we compare them? The natural language modeling community has overcome similar challenges using Unicode, which is a reasonably good approximation to a universal vocabulary for the characters of written natural language (see Section 4.1.4). But it is not clear whether we can define a universal vocabulary for musical scores, or if this is the right approach to comparing results in the music domain.

4.1.1 Sequential Encodings

A sequential encoding expresses the content of a score as a linear sequence of tokens. In a trivial sense, any digital encoding of scores is sequential because the fundamental digital storage medium is a sequence of bits. We will find it useful to distinguish sequential encodings, where the semantic content of the data is expressed using a linear structure, from serialized encodings of tensors (Section 4.1.2) and hierarchical structures (Section 4.1.3). The MIDI format [International MIDI Association, 1983] is an example of a sequential data format that is commonly used to encode musical scores. A Standard MIDI File [International MIDI Association, 1988] is a container that stores a collection of MIDI tracks;[1] Figure 4.2 presents an example of a MIDI track that encodes the top line of the score fragment shown in Figure 4.1. MIDI is a binary format; the binary MIDI data is transliterated into text in Figure 4.2 to make the structure of the format more comprehensible. For the discussion in this section

[1]MIDI is arguably a hierarchical encoding with two layers of hierarchy: a MIDI file contains a collection of tracks, each of with contains a sequence tokens.

we will refer to each line of the transliteration as a token, although for modeling purposes we would typically use a more fine-grained tokenization.

The MIDI format is an operational format, in the sense that each token can be interpreted as a command that manipulates a hidden state. The MIDI file as a whole can be thought of as a program that, when executed sequentially, constructs a musical performance of the file's content. This procedural, operational style reflects the history of the MIDI format, which was created as a network communication protocol for transmitting musical information between electronic instruments and other devices (sound boards, synthesizers, etc.). The Standard MIDI File format is a container for storing recording sequences of MIDI network traffic. The time annotations associated with each token reflect this history; each time stamp represents the time difference between receipt of the previous token and receipt of the current token.

When we read a MIDI file sequentially, we must keep track of a collection of hidden state. For example, we need to track an accumulating value of time as we process each token. The stateful, operational nature of MIDI is also reflected in how it encodes notes. Each note is encoded by a pair of tokens: a `note_on` token, followed later in the sequence by a `note_off` token. The `note_on` and `note_off` tokens manipulate a hidden state of active notes, which a parser (or model) must track as it processes the MIDI file. While the MIDI format is operational, this is not a general property of sequential encodings of music. It is easy to define a declarative sequential format that replaces relative time stamps with absolute timing information. The stateful `note_on` and `note_off` tokens could be replaced with a single `note` token, with absolute start and end times.

A general property of sequential formats is that the order of many tokens is arbitrary. Imposing a temporal order on tokens is canonical, but further constrains on the order of tokens is difficult to justify. This is illustrated in several parts of our MIDI example: any sequence of tokens with zero time delay (`time=0`) can be re-ordered without changing the semantic content of the MIDI file. This poses a dilemma for modeling: either we can try to learn a model that is invariant to these permutations of the sequence, or we can pre-process these sequences with an arbitrarily imposed sort-order. Neither solution is ideal. Allowing

```
1   meta_message start_of_track
2   program_change channel=0 program=0 time=0
3   note_on channel=0 note=76 velocity=80 time=0
4   note_off channel=0 note=76 time=455
5   note_on channel=0 note=76 velocity=80 time=25
6   note_off channel=0 note=76 time=341
7   note_on channel=0 note=76 velocity=80 time=19
8   note_off channel=0 note=76 time=113
9   note_on channel=0 note=76 velocity=80 time=7
10  note_off channel=0 note=76 time=455
11  note_on channel=0 note=72 velocity=80 time=25
12  note_off channel=0 note=72 time=341
13  note_on channel=0 note=69 velocity=80 time=19
14  note_off channel=0 note=69 time=113
15  meta_message end_of_track
16  meta_message start_of_track
17  note_on channel=0 note=57 velocity=80 time=0
18  note_on channel=0 note=60 velocity=80 time=0
19  note_on channel=0 note=64 velocity=80 time=0
20  note_off channel=0 note=57 time=227
21  note_off channel=0 note=60 time=0
22  note_off channel=0 note=64 time=0
23
24        ...
25
26  meta message end_of_track
```

Figure 4.2: A text re-encoding of the binary MIDI encoding of the top line of the score displayed in Figure 4.1. For brevity, a portion of the second track has been elided (...).

permutations of token order complicates the learning problem. But imposing an arbitrary order raises difficult questions: could we have learned better using a different order? This is a perennial question in the autoregressive modeling community [Larochelle and Murray, 2011] and has no easy answer: we cannot hope to exhaustively explore the space of orderings.

```
1    **kern          **kern
2    *staff2         *staff1
3    *clefF4         *clefG2
4    =1-             =1-
5    8A\ 8c\ 8e\L    4ee\
6    8A\ 8c\ 8e\     .
7    8A\ 8c\ 8e\     8.ee\L
8    8A\ 8c\ 8e\J    .
9    .               16ee\Jk
10   8A\ 8c\ 8e\L    4ee\
11   8A\ 8c\ 8e\     .
12   8A\ 8c\ 8e\     8.cc/L
13   8A\ 8c\ 8e\J    .
14   .               16a/Jk
15   ==|!            ==|!
16   *-              *-
```

Figure 4.3: Humdrum encoding of the score displayed in Figure 4.1. Each successive row indicates a new musical event in the time series (first axis of the tensor). Parts (staves) are organized in tab-separated columns (second axis of the tensor). Each tab-delineated item in a row consists of one or more space-delineated tokens (third axis of the tensor). For example, the notation `8A\ 8c\ 8e\L` in Line 5, Column 1 indicates an A minor triad of eighth-notes (the addition slashes and `L` indicate visual formatting instructions that we do not model in this work).

4.1.2 Tensor Encodings

A tensor encoding expresses the content of a score as a multi-dimensional array of tokens. Tensors can be a convenient way to encode musical score: one axis of the tensor can represent time, for example, another axis can represent pitch, and additional axes can represent instrument, articulation, dynamics, etc. By encoding information across multiple axes, we avoid the need to impose an arbitrary ordering on this information. The Humdrum format [Huron, 1993] can be viewed as a serialized tensor encoding of musical scores. Figure 4.3 presents an example of Humdrum data that encodes the score fragment shown in Figure 4.1. The piano-roll encoding discussed in Section 2.1 is another example of a tensor encoding.

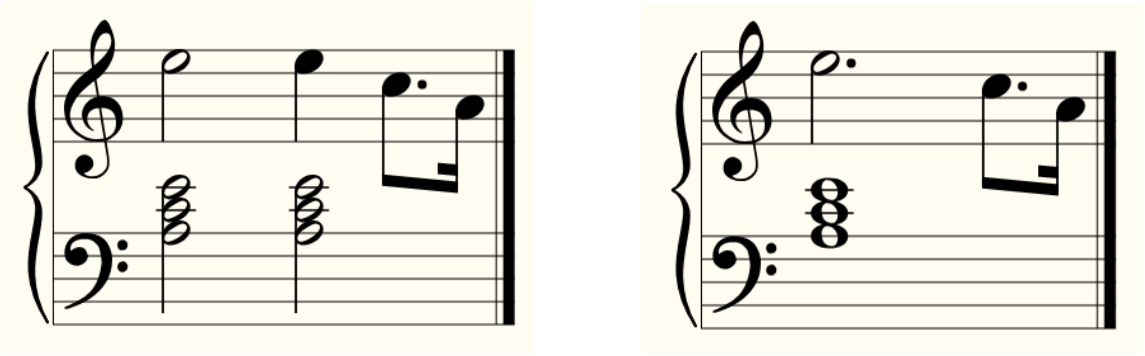


Figure 4.4: Two scores with the same piano-roll representation as the score fragment in Figure 4.1. The popular dataset introduced by Boulanger-Lewandowski et al. [2012] uses this single-bit representation. A second bit is introduced in some more recent work: Liang et al. [2017] refers to these as "Tie bits")

The Humdrum format is technically a sequential encoding; indeed, any file format is ultimately described by a linear sequence of bits. But we can view the Humdrum format as a serialization of a tensor. A Humdrum file is organized into T lines: this is the first axis of the tensor, which represents time. Each line of the file consists of V tab-separated columns: this is a second axis of the tensor, which represents distinct musical voices (analogous to MIDI tracks). Each column contains up to P space-separated tokens: this is a third axis of the tensor, which represents concurrent events within a part. And each token consists of up to A subtoken characters: this is a fourth axis of the tensor, which represents attributes of a single event. The Humdrum file can be de-serialized into a 4-dimensional tensor with dimensions $T \times V \times N \times A$, analogous to how a serialized image (stored on disk as a linear sequence of bytes) can be deserialized into 3-dimensional tensor with dimensions $H \times W \times C$.

A time-discretization of a piano-roll (Definition 2.1) is also a tensor encoding. In that case, the tensor has just two axes: time and notes, omitting part assignments and other details of a score. In Sections 4.2 and 4.3 we take $A = 2$ and adopt a working definition of a score as a binary tensor $\mathbf{x} \in \{0,1\}^{T \times V \times 2P}$. This definition still omits many details of a score, but it augments the simpler piano-roll representation with part information and a second attribute. At each time $t \in T$, in each voice $v \in V$, we have two values. The first bit

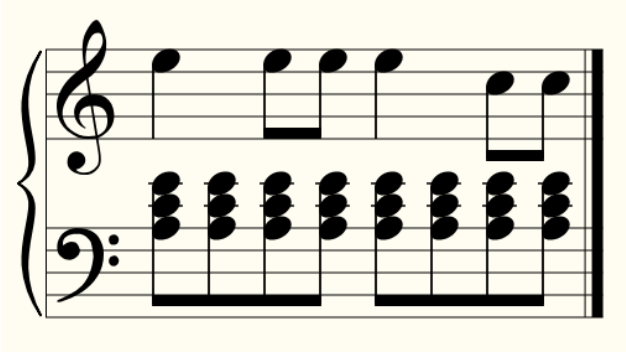


Figure 4.5: A corruption of the score shown in Figure 4.1, discretized at eighth-note resolution. Fine-grained rhythmic information, and the pitch of the final note in the treble part, are lost. Boulanger-Lewandowski et al. [2012] discretizes data at quarter-note resolution.

$\mathbf{x}_{t,v,p}$ is the standard piano-roll pitch attribute, which indicates that pitch $p \in P$ in voice v is present at time t. The second bit $\mathbf{x}_{t,v,P+p}$ indicates that pitch p in voice v begins at time t. While classical piano-roll representations omit the second onset bit, both bits are necessary to faithfully represent a musical score. Consider, for example, Figure 4.4: these two scores have identical piano-roll encodings if only a single bit is used to indicate a pitch. Many other scores also alias to this same piano-roll encoding. The addition of an onset bit delineates the boundaries between multiple notes of the same pitch, thus resolving the ambiguity.

One subtlety of piano-roll representations is the choice of discretization. The piano-rolls **pianoroll** : $[0, T) \to \{0, 1\}^N$ described by Definition 2.1 are continuous-time processes with a real-valued temporal indices $t \in [0, T)$. Discretizing this process at a sampling rate Δ can introduce distortions. We will argue in Section 4.2 that uniform discretization at a sufficiently small sampling rate Δ is information-preserving. From the perspective of Nyquist-Shannon theorem [Shannon, 1949] we can interpret this result as a statement that scores contain no high-frequency content. The consequences of choosing a discretization that is too coarse are illustrated by Figure 4.5.

```
<score-partwise>
  <part-list>
    <part-group type="start" number="1"/>
    <score-part id="P1"/>
    <part-group type="stop" number="1"/>
    </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>4</divisions>
        <time> <beats>4</beats> <beat-type>4</beat-type>
          </time>
        <staves>2</staves>
        <clef number="1"> <sign>G</sign> <line>2</line>
          </clef>
        <clef number="2"> <sign>F</sign> <line>4</line>
          </clef>
        </attributes>
      <note>
        <pitch> <step>E</step> <octave>5</octave>
          </pitch>
        <duration>4</duration>
        <voice>1</voice>
        <type>quarter</type>
        <stem>down</stem>
        <staff>1</staff>
        </note>

      ...

      <barline location="right">
        <bar-style>light-heavy</bar-style>
        </barline>
      </measure>
    </part>
  </score-partwise>
```

Figure 4.6: MusicXml encoding of the score displayed in Figure 4.1. For brevity, a portion of the MusicXml document has been elided (...).

4.1.3 Hierarchical Encodings

A hierarchical encoding expresses the content of a score using a nested tree structure. Hierarchies are a natural way to organize the content musical scores: a score contains parts (tracks; voices) which in turn contain measures, which contain notes, etc. The MusicXml format [Good, 2001] is an example of a hierarchical data encoding of musical scores. Like the Humdrum format (Section 4.1.2) MusicXml is technically a serialized form of the hierarchical semantics that it describes. Figure 4.6 presents an excerpt of MusicXml encoding for the score fragment shown in Figure 4.1. Hierarchical encodings hold the potential to mitigate some of the challenges of modeling long-range dependencies. Recent work has begun to explore structured models that process hierarchically encoded music both as inputs for conditional tasks [Jeong et al., 2019a,b] and as the targets for generative modeling [Wang et al., 2020]. There has also been analogous work in the natural language domain using parse trees [Dyer et al., 2016].

4.1.4 Encodings of Natural Language

Like with music, encoding difficulties have appeared in the history of modeling written languages. Language models based on word tokenizations face an out-of-vocabulary problem: for any fixed vocabulary of words, a sufficiently large corpus of data will contain words outside of the fixed vocabulary. This complication is traditionally addressed by "unk"-ing the corpus: each word in the corpus outside of the fixed vocabulary is replaced by a special `<unk>` token [Jurafsky and Martin, 2009]. However, this preprocessing turns the canonical task of modeling the distribution over natural language text into a non-canonical task: modeling the distribution over altered text with a specific unk'ing pattern. Modelers of musical scores face a similar challenge: encoding and pre-processing steps elide different details of scores, and our task is reduced to modeling a particular, non-canonical encoding of scores.

The vocabulary situation in language has improved recently with the introduction of

character-level models [Zhang et al., 2015] and subword tokenizations [Schuster and Nakajima, 2012, Sennrich et al., 2016, Wu et al., 2016, Kudo and Richardson, 2018]. While the vocabulary of words in a language is open-ended, the vocabulary of characters is fixed: to be sure that our vocabulary is comprehensive, we can take the entire base multilingual plane of the Unicode as a character set. Subword tokenizations (e.g., byte-pair encoding) simply compress the underlying character sequences by aggregating common chunks of characters into single tokens. These models avoid non-canonical, lossy pre-processing steps and directly model the canonical encoding of text as a sequence of characters. Unfortunately, whereas the language community has been able to resolve the encoding question by simply clarifying the canonical encoding of their data and using it, the encoding question for symbolic music appears to be more fundamental.

4.1.5 Encodings of Images

In contrast to music and natural language, image processing has used a largely consistent and stable tensor encoding: an image is encoded as a 3-dimensional tensor with height, width, and color channel axes. One exception to this consistent approach is recent exploration of coordinate encodings. These encodings augment the color channel information at each spatial location of an image with an additional encoding of the spatial coordinates themselves. This encoding is informationally redundant, because these locations are implicitly defined by the structure of the tensor, but explicitly introducing this information can be helpful for modeling. Encoding coordinates can re-introduce spatial information into convolutional models that are otherwise spatially invariant [Liu et al., 2018] and transformer models that are otherwise blind to spatial structure [Parmar et al., 2018].

4.2 Autoregressive Modeling of Musical Scores

The conceit of generative modeling is to view our data—in this case musical scores—as samples from an unknown probability distribution. We will "learn to compose" by constructing a density estimator for this probability distribution (see Chapter 2, Section 2.2) and gen-

erate new scores by sampling from our approximation. For a broad survey of approaches to automatic music composition, see Herremans et al. [2017]; for a more targeted survey of classical probabilistic approaches, see Conklin [2003]. Our focus in this work is on models of musical scores, in contrast to other recent and impressive work that models expressive musical performances [Oore et al., 2020, Huang et al., 2019]. We discuss the distinction between these two modeling tasks in Section 4.2.2.

Polyphonic scores consist of notes and other features of variable length that overlap each other in quasi-continuous time. Scores contain a vast heterogenous collection of information, much of which we will not attempt to model: time signatures, tempi, dynamics, etc. We will therefore give a working definition of a score that captures the essential aspects of a score that we will model: pitch, rhythm, and voice assignments. This extends the definition of a piano-roll (Chapter 2, Definition 2.1) to include voice and onset information. Like in Definition 2.1, we let N denote a vocabulary of pitch classes.

Definition 4.1. *A score of length T, consisting of V voices, is a discrete-valued function* $\mathbf{score} : [0, T) \to \{0, 1\}^{V \times 2P}$. *For each voice $v \in \{1, \ldots, V\}$ and each pitch $p \in \{1, \ldots, N\}$,*

$$\mathbf{score}_{t,v,p} = 1 \qquad \textit{if pitch } p \textit{ is on at time } t \textit{ in voice } v, \tag{4.1}$$

$$\mathbf{score}_{t,v,P+p} = 1 \qquad \textit{if pitch } p \textit{ begins at time } t \textit{ in voice } v. \tag{4.2}$$

Both "pitch" bits (4.1) and "onset" bits (4.2) are important to faithfully encode a score; the onset bit allows the encoding to distinguish between a sequence of repeated notes of the same pitch and a single sustained note (see Section 4.1.2, Figure 4.4). Incorporating voice annotations allows us to think of polyphonic scores as a collection of correlated single-voice, homophonic composition problems; this presents us with interesting opportunities for serializing and modeling scores, which we explore in Sections 4.2.3 and 4.2.4.

We will work with temporal discretizations of scores, uniformly sampled at a uniform rate Δ, given by the Nyquist rate of the corpus.

Definition 4.2. *The Nyquist rate [Nyquist, 1928] for a corpus of musical scores is the largest time value $\Delta \in \mathbb{R}$ such that all change-points in the corpus occurs at integer multiples of Δ. A discretized score is a tensor $\mathbf{x} \in \{0,1\}^{T/\Delta \times P \times 2N}$ defined by $\mathbf{x}_k = \mathbf{score}(k\Delta)$.*

For example, the Nyquist rate of a corpus containing 32nd notes and triplet patterns, but not 64th notes or more rhythmically complex quintuplets or septuplets, would be $\Delta = 1/48$ samples per beat. See Section 4.1.2 for an illustration of the consequences of sampling at a lower rate than the Nyquist rate. Proposition 4.3 shows that sampling scores at the Nyquist rate is sufficient, in the sense that a higher sampling rate yields no additional information about the score.

Proposition 4.3. *Let $\mathcal{P} = (0, \Delta, 2\Delta, \ldots, T)$ be the uniform partition of the interval $[0, T]$ at the Nyquist rate. For any refinement $\mathcal{R}$ of $\mathcal{P}$,*

$$H(\mathbf{x}) = H(\mathbf{score}_{\mathcal{R}_1}, \ldots, \mathbf{score}_{|\mathcal{R}|}) \tag{4.3}$$

Proof. Applying the chain rule for conditional probabilities to the entropy of a score sampled on a partition $\mathcal{R}$,

$$H(\mathbf{score}_{\mathcal{R}_1}, \ldots, \mathbf{score}_{|\mathcal{R}|}) = \mathop{\mathbb{E}}_{\mathbf{score} \sim q} \log q(\mathbf{score}_{\mathcal{R}_1}, \ldots, \mathbf{score}_{\mathcal{R}_{|\mathcal{R}|}}). \tag{4.4}$$

$$= \sum_{k=1}^{|\mathcal{R}|} \mathop{\mathbb{E}}_{\mathbf{score} \sim q} \log q(\mathbf{score}_{\mathcal{R}_k} | \mathbf{score}_{\mathcal{R}_1}, \ldots, \mathbf{score}_{\mathcal{R}_{k-1}}). \tag{4.5}$$

Consider terms $q(\mathbf{score}_{\mathcal{R}_k} | \mathbf{score}_{\mathcal{R}_1}, \ldots, \mathbf{score}_{\mathcal{R}_{k-1}})$ where $\mathcal{R}_k \notin \mathcal{P}$. There must be some n such that $n\Delta < \mathcal{R}_k < (n+1)\Delta$. We must have $\mathbf{score}_{\mathcal{R}_k} = \mathbf{score}_{n\Delta}$ because, by definition of the Nyquist rate Δ (Definition 4.2) all change-points in $\mathbf{x}$ occur at integer multiples of Δ. Clearly $n\Delta \in \mathcal{R}$ ($\mathcal{R}$ is a refinement of $\mathcal{P}$) and $n\Delta \in (\mathcal{R}_0, \ldots, \mathcal{R}_{k-1})$ ($n\Delta < \mathcal{R}_k$). It follows that

$$\mathop{\mathbb{E}}_{\mathbf{score} \sim q} \log q(\mathbf{score}_{\mathcal{R}_k} | \mathbf{score}_{\mathcal{R}_1}, \ldots, \mathbf{score}_{\mathcal{R}_{k-1}}) = \mathop{\mathbb{E}}_{\mathbf{score} \sim q} \log q(\mathbf{score}_{\mathcal{R}_k} | \mathbf{score}_{n\Delta}) = 0. \tag{4.6}$$

Dropping all such terms k with $\mathcal{R}_k \notin \mathcal{P}$ we see that

$$H(\mathbf{score}_{\mathcal{R}_1}, \ldots, \mathbf{score}_{|\mathcal{R}|}) = \sum_{k\,:\,\mathcal{R}_k \in \mathcal{P}} \mathop{\mathbb{E}}_{\mathbf{score} \sim q} \log q(\mathbf{score}_{\mathcal{R}_k} | \mathbf{score}_{\mathcal{R}_0}, \ldots, \mathbf{score}_{\mathcal{R}_{k-1}}) \tag{4.7}$$

$$= \sum_{k=1}^{T/\Delta} \mathop{\mathbb{E}}_{\mathbf{score} \sim q} \log q(\mathbf{score}_{k\Delta} | \mathbf{score}_0, \ldots, \mathbf{score}_{(k-1)\Delta}) \tag{4.8}$$

$$= \mathop{\mathbb{E}}_{\mathbf{score} \sim q} \log q(\mathbf{score}_0, \mathbf{score}_\Delta, \ldots, \mathbf{score}_T) = H(\mathbf{x}). \qquad \square$$

From the perspective of the Nyquist-Shannon theorem [Shannon, 1949] Proposition 4.3 states that scores contain no high frequency content. It may be illuminating to draw a contrast with processes such as Brownian motion, for which refinement of the sampling partition always reveals new details of the process. Indeed, Proposition 4.3 is a special property of scores that does not even hold for other forms of digital music, such as expressive MIDI performances (see Section 4.2.2). The difference is that in performances, unlike scores, musical events (e.g., note onsets and offsets) occur at fuzzy real-valued timestamps, rather than precise subdivisions of a metrical structure.

We parameterize our estimator of the distribution over tensors $\mathbf{x}$ with a neural network. Because scores have varying lengths, we find it natural to construct autoregressive models of these objects (see Chapter 2, Section 2.3) that parameterize conditional distributions of the evolution of the score over time. A precise discussion of the autoregressive factorization used in this work is presented in Section 4.2.3, along with discussion of alternative factorizations. Musical scores are highly structured objects, which motivates an exploration of neural architectures that can exploit this structure; we consider a variety of music-specific modeling techniques in Section 4.2.4. Evaluating generative models is difficult [Theis et al., 2016], and this difficulty has lead to a diversity of opinions (and substantial disagreement) over methodology for evaluating generative models. We discuss the challenge of evaluating generative models in Section 4.2.2, and develop a cross-entropy measure for studying the effects of the modeling techniques presented in Section 4.2.4. We conclude in Section 4.2.5 with an evalu-

ation of our best model using a user study, along with a broader discussion of methodology for evaluating generative models. Supplementary material including compositional samples and code for reproducing the experiments is available online.[2]

4.2.1 Related Work on Symbolic Generative Modeling

Early probabilistic models of music focused on single-voice, monophonic melodies. Possibly the first probabilistic model for music generation was proposed by Pinkerton [1956]. This work was followed concretely by Jr. et al. [1957], who built a Markov transition model estimated on small music corpora. A proliferation of work on computer-generated music and data-driven musicology followed these pioneering works in the 1960's and 1970's; see Roads [1980] for a survey. An important development during this era was the application of Chomsky-inspired grammatical analysis to music, exemplified by Kohonen [1989]; this latter work contemplates the generation of two concurrent musical parts, one of the earliest examples of polyphonic generation.

The first application of neural networks to melody composition was proposed by Todd [1989]. This work prompted followup [Mozer, 1994] using an alternative data representation inspired by pitch geometry ideas [Shepard, 1982]; the relative pitch and note-embedding schemes considered in Section 4.2.4 can be seen as a data-driven approach to capturing some of these geometric concepts. Neural melody generation was revisited by Eck and Schmidhuber [2002], using long short-term memory models. More recent work on melodic composition experiments with techniques to capturing longer-term structure than classic recurrent models provide. Jaques et al. [2017] explore reinforcement learning as a tool for eliciting long-term structure, expanding on ideas first considered by Franklin [2001]. Roberts et al. [2018] also attempt to capture long-term structure, proposing a variational auto-encoder for this purpose. For recent work on monophonic composition, see Sturm et al. [2016], Jaques et al. [2017], Roberts et al. [2018].

---

[2] http://homes.cs.washington.edu/~thickstn/ismir2019composition/

Work on polyphonic music composition has a shorter history. Early precursors include Kohonen [1989], which considers two-voice composition, and Ebcioğlu [1988], which proposes an expert system to harmonize 4-voice Bach chorales. The harmonization task became popular, along with the Bach chorales dataset [Allan and Williams, 2004]. Multiple voice polyphony is directly addressed by Lavrenko and Pickens [2003], albeit using a simplified preprocessed encoding of scores that throws away duration information. Maybe the first work with a fair claim to consider polyphonic music in full generality is Boulanger-Lewandowski et al. [2012]; that work proposes a coarsely-discretized tensor encoding of musical scores (see Section 4.1.2) and examines the cross-entropy of a variety of neural models on several music datasets (including the Bach chorales). Much subsequent work on polyphonic models uses the dataset, encoding, and quantitative metrics introduced by Boulanger-Lewandowski et al. [2012], notably Vohra et al. [2015] and Johnson [2017].

Several recent generative models of scores focuses exclusively on the Bach chorales dataset [Liang et al., 2017, Hadjeres et al., 2017, Huang et al., 2017]. Both Liang et al. [2017] and Hadjeres et al. [2017] evaluate their models using qualitative large-scale user studies. The system proposed by Hadjeres et al. [2017] optimizes a pseudo-likelihood, so its quantitative losses cannot be directly evaluated using the cross-entropies discussed in Section 4.2.2. The generative model proposed by Liang et al. [2017] could in principle report cross entropies, but that work also focuses on a user study evaluation. Quantitative cross-entropy results are reported for the Bach chorales by Huang et al. [2017]. Both Hadjeres et al. [2017] and Huang et al. [2017] propose non-sequential Markov chain sampling schemes for generation, in contrast to the ancestral sampler used by Liang et al. [2017] and our models in Section 4.2.4. Markov-chain Monte Carlo samplers are revisited in Chapter 5.

4.2.2 Evaluation Methodology

Our work uses a cross-entropy metric, calculated on a holdout subset of the score data, to study the effects of the various structural modeling assumptions discussed in Section 4.2.4. It is reasonable to question whether cross-entropy is the right metric for this study. On the

Bach	Beethoven	Chopin	Scarlatti	Early	Joplin	Mozart	Hummel	Haydn
191,374	476,989	57,096	58,222	1,325,660	43,707	269,513	3,389	392,998

Table 4.1: Notes in the KernScores dataset, partitioned by composer. The "Early" collection consists of Renaissance vocal music; a plurality of the Early music is composed by Josquin.

one hand, if we accept the conceit that our goal is to model a probability distribution over musical scores, then cross-entropy is a natural finite-sample estimate of the Kullback-Leibler divergence from our model to the data distribution. On the other hand, if we view this probabilistic conceit as a concrete approach to the broader automatic music composition problem, then we need to argue that a test-set cross-entropy is a good surrogate for measuring the quality of the samples we generate. In practice, we observe across many generative modeling tasks that models trained to minimize cross-entropy to produce qualitatively good samples [Brown et al., 2020, Radford et al., 2019, Esser et al., 2021, Dhariwal et al., 2020]. Because the cross-entropy is easily calculated, we are able to methodically ablate our structural modeling choices (Section 4.2.4). But in deference to the broader question of automatic music composition, we also analyze our best-performing model using a user-study (Section 4.2.2).

Dataset. The models discussed in this paper are trained on the KernScores dataset [Sapp, 2005], a collection of early modern, classical, and romantic era digital scores encoded in Humdrum format 4.1.2 and assembled by musicologists and researchers associated with Stanford's CCARH.[3] We use a subset of this dataset consisting of over 2,300 scores containing approximately 2.8 million note labels. Tables 4.1 and 4.2 give a sense of the contents of the dataset. We discretize the dataset using $\Delta = 1/48$ samples per beat (see Definition 4.2); this sampling rate captures most—but not all—of the content of this corpus but will create aliasing in rare instances of 64'th notes and quintuplet patterns. We hold out 200 randomly chosen scores as a test set for calculating model cross-entropies.

We contrast this dataset's Humdrum-encoded scores with the MIDI encoded expressive

[3] http://kern.ccarh.org/

Vocal	String Quartet	Piano
1,412,552	820,152	586,244

Table 4.2: Notes in the KernScores dataset, partitioned by ensemble type.

performances used by many recent large-scale music modeling efforts.[4] As discussed in Section 4.1.1, MIDI was designed as a protocol for communicating digital performances, rather than scores. This is exemplified by the MAPS [Emiya et al., 2010] and MAESTRO [Hawthorne et al., 2019] datasets, which consist of symbolic MIDI data aligned to expressive performances. While this data is symbolic, it cannot be interpreted as scores because it is unaligned to a grid of beats and does not encode note-values (quarter-note, eighth-note, etc). Some MIDI datasets are aligned to a grid of beats, including MusicNet (Section 3.2). But heuristics are still necessary to interpret this data as visual scores. For example, many MIDI files encode "staccatto" articulations by shortening the length of notes, thwarting simple rules that identify note-values based on length.

Evaluation. Let $\hat{q}$ be an estimate of the unknown probability distribution over scores q. We want to measure the quality of $\hat{q}$ by its cross-entropy to q (Definition 2.5). Because the entropy of a score grows with its length T is is natural to consider a cross-entropy rate that measures the average cross-entropy over a unit of time. By convention, we measure time in units of beats, so the cross-entropy rate will have units of bits per beat. Defining cross-entropy for a continuous-time processes such as $\mathbf{score} : [0, T) \to \{0, 1\}^{P \times 2N}$ (Definition 4.1) generally requires some care. But in light of Proposition 4.3, a cross entropy defined on discretized scores $\boldsymbol{x} \in \{0, 1\}^{T/\Delta \times P \times 2N}$ sampled at the Nyquist rate (Definition 4.2) is sufficient.

Definition 4.4. *The rate-adjusted cross entropy from between two distributions $\hat{q}$ to q over*

[4] A notable exception is Lavrenko and Pickens [2003], which uses data derived from the KernScores collection considered here.

scores of length T sampled at rate Δ is given by

$$H(q, \hat{q}) \equiv \mathop{\mathbb{E}}_{\boldsymbol{x} \sim q} \left[-\frac{1}{T\Delta} \log \hat{q}(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{T/\Delta}) \right].$$

This is a cross entropy measure of joint distributions over musical scores. Notably, it is agnostic to autoregressive factorization of this distribution. As we will see in Section 4.2.3, there are many possible autoregressive factorizations of the joint distribution over scores. These choices model different conditional distributions, with distinct corresponding conditional cross entropies. Measuring cross entropy in units of bits per beat according to Definition 4.4, we can compare cross entropy results across models constructed using different factorizations. In practice, calculating $H(q, \hat{q})$ from the next-token cross entropy of a given autoregressive model is a simple change of units, calculated via a multiplicative factor.

4.2.3 Factoring the Distribution over Scores

We are interested in building a neural autoregressive model (Chapter 2, Section 2.3) of the unknown distribution $q(\mathbf{x})$ over discretized musical scores $\mathbf{x} \in \{0, 1\}^{T/\Delta \times P \times 2N}$. The idea is to serialize the tensor $\mathbf{x}$, and factor the joint distribution over scores into a sequence of conditional distributions. We parameterize this collection of conditional distributions with a neural network. To emphasis this point: we use a single neural network with a common set of weights to parameterize each of the conditional distributions arising from the autoregressive factorization. This reflects the modeling assumption that the evolution of a serialized score is dependent only on its history, rather than an absolute position in the sequence; formally, we assume that the sequence is stationary. The stationarity assumption can be weakened by appending absolute position information to the encoding of the score (see Section 4.1.5 for analogous weakening of the translation-invariant assumption imposed by convolutional models in the vision domain).

There are many ways to serialize a score tensor. This choice of serialization, and the corresponding factorization of $q(\mathbf{x})$, has broad implications for both the computational ef-

ficiency and the inductive biases of our autoregressive models. In the remainder of this section we review raster and sparse serializations, and introduce new serializations inspired by run-length encoding that we use for the modeling work in Section 4.2.4. The serializations presented below are not comprehensive, but they are chosen to illustrate a variety of choices and the consequences these choices have for modeling.

Raster Serializations. Rasterization serializes the score tensor along the time axis in uniform discrete steps of length Δ. An autoregressive factorization of a rasterized score has the following form:

$$q(\mathbf{x}) = \prod_{k=0}^{T/\Delta} q(\mathbf{x}_k|\mathbf{x}_{1:k}) = \prod_{k=0}^{T/\Delta} q(\mathbf{score}_{k\Delta}|\mathbf{score}_{0:k\Delta}) \tag{4.9}$$

Throughout this work, a slice $a:b$ is inclusive of the first index a but does not include the final index b. Rasterized time is a common serialization of scores, notably used to construct an autoregressive factorization of polyphonic music by Boulanger-Lewandowski et al. [2012]. The values $\mathbf{x}_k \in \{0,1\}^{P\times 2N}$ are still high-dimensional objects; if we impose an order on parts and notes, we can further factor this distribution into

$$q(\mathbf{x}) = \prod_{k=1}^{T/\Delta} \prod_{v=1}^{V} \prod_{p=1}^{2P} q(\mathbf{x}_{k\Delta,v,p}|\mathbf{x}_{1:k\Delta}, \mathbf{x}_{k\Delta,1:v}, \mathbf{x}_{k\Delta,v,1:p}). \tag{4.10}$$

The values $\mathbf{x}_{k\Delta,v,p}$ are simply Bernoulli random variables, which can be directly modeled using, for example, a parameterized sigmoid. The orders imposed on voices and pitch classes are somewhat arbitrary; pitches are typically ordered either low-to-high or high-to-low. Raster serialization of score tensors is directly comparable to the serializations of images adopted for autoregressive image modeling [van den Oord et al., 2016b, Salimans et al., 2017, Parmar et al., 2018]. Like musical scores, image serialization requires an arbitrary choice of order: the spatial dimensions are typically ordered left-to-right, top-to-bottom, and the color channels of each pixel are ordered using the conventional red-green-blue ordering.

Raster models are expensive to train and evaluate on rhythmically diverse music. Rasteri-

zation at the Nyquist frequency creates sequences of length $O(1/\Delta)$; a full serial factorization of the score given by Equation (4.10) creates sequences of length $O(TPV/\Delta)$. While the costs associated with these serializations can be somewhat mitigated by parallelism during training, the exact, ancestral algorithm for sampling from autoregressive models is a fundamentally serial operation (see Chapter 5 for a discussion of approximate sampling from autoregressive models, which can sometimes mitigate this sampling cost). To reduce the computational burden of raster serialization, raster models are typically trained using a coarse sampling rate Δ. Unfortunately, as we saw in Section 4.1.2, coarse discretization can create significant aliasing effects.

Run-Length Serializations. Definition 4.5 introduces one possible variant of run-length serialization, which exploits the temporal stability of musical scores to reduce sequence length.

Definition 4.5. *Let $c_0, \ldots, c_L$ denote the temporal change points in a score $\mathbf{x}$. The run-length serialization of $\mathbf{x}$ is the tensor $\mathbf{e} \in \left(\mathbb{N} \oplus \{0,1\}^{P \times 2N}\right)^L$ defined by $\mathbf{e}_{k,0} = \frac{c_{k+1} - c_k}{\Delta} \in \mathbb{N}$ and $\mathbf{e}_{k,1} = \mathbf{x}_{c_k}$.*

The sequence $\mathbf{e}$ is non-linear in its time index k: entry $\mathbf{e}_{k+1}$ occurs $d_k \Delta$ beats after entry $\mathbf{e}_k$, in contrast to raster serializations where $\mathbf{x}_{t+1}$ occurs at a constant time interval Δ following $\mathbf{x}_t$. We can factor the distribution over run-length encoded scores into conditional distributions over natural-number durations $\mathbf{e}_0$ and binary note values $\mathbf{e}_1$:

$$q(\mathbf{e}) = \prod_{k=1}^{L} q(\mathbf{e}_k | \mathbf{e}_{1:k}) = \prod_{k=1}^{L} q(\mathbf{e}_{k,0} | \mathbf{e}_{1:k}) \prod_{v=1}^{V} \prod_{p=1}^{2P} q(\mathbf{e}_{k,v,p} | \mathbf{e}_{1:k}, \mathbf{e}_{k,0}, \mathbf{e}_{k,1,1:v}, \mathbf{e}_{k,1,v,1:p}). \tag{4.11}$$

These serializations have length $O(LPV)$, in contrast to raster serializations which have length $O(TPV/\Delta)$. Because the number of change points L in a score is much smaller than the number of samples T/Δ, run-length serializations save a substantial amount of computation by predicting a single run-length duration $\mathbf{e}_{k,0}$ rather than re-iterating a sequence of identical predictions $\mathbf{x}_{t,p,n}$ at small incremental time steps. One criticism of the run-length

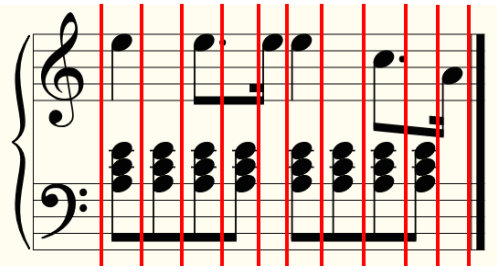


Figure 4.7: The Mozart from Figure C.1, with red lines that indicate the boundaries of events under a run-length factorization of the score. Notes in the treble staff are chopped up into eight-note runs, so instead of predicting note durations (quarter, dotted-eighth, sixteenth, etc.) we instead predict fragments of notes (eighth, continue eighth, continue eighth, etc.).

factorization is that, when notes of different durations overlap in a score, the longer notes are chopped up along the boundaries of the short notes as illustrated in Figure 4.7. Rather than predicting conditional likelihoods of musically meaningful quantities like notes, we instead predict artificial vertical slices of a score.

Homophonic Run-Length Serializations. Putting full scores aside for a moment, consider a single voice v, i.e., a slice $\mathbf{x}_{1:T/\Delta,v,1:2P}$ of a score tensor.

Definition 4.6. *Let $c_0^v, \ldots, c_{L_v}^v$ denote the temporal change points in voice v of a score $\mathbf{x}$. The run-length serialization of $\mathbf{x}_{1:T/\Delta,v,1:2P}$ is the tensor $\mathbf{r} \in \left(\mathbb{N} \oplus \{0,1\}^{\times 2N}\right)^L$ defined by $\mathbf{r}_{k,0} = \frac{c_{k+1}^v - c_k^v}{\Delta} \in \mathbb{N}$ and $\mathbf{r}_{k,1} = \mathbf{x}_{c_k^v,v}$.*

And we can factor the distribution over run-length encoded voices as

$$q(\mathbf{r}) = \prod_{k=1}^{L_v} q(\mathbf{r}_k|\mathbf{r}_{1:k}) = \prod_{k=1}^{L} q(\mathbf{r}_{k,0}|\mathbf{r}_{1:k}) \prod_{p=1}^{2P} q(\mathbf{r}_{k,p}|\mathbf{r}_{1:k}, \mathbf{r}_{k,0}, \mathbf{r}_{k,1,1:p}). \tag{4.12}$$

By definition of the Humdrum data format, voices in the KernScores dataset are homophonic: no notes are sustained across a change point. To illustrate this point, in Figure 4.7 the score as a whole is polyphonic because some notes are sustained across change points. The treble staff is monophonic, because only one note is performed at a given time. The bass staff is homophonic: while multiple notes are performed concurrently, they proceed in rhythmic lock-

step. This means that, for an individual voice, run-length serialization behaves very nicely: it reduces the length of our serialization (compared to rasterization) without breaking up the score into non-musical slices. Indeed, the durations $\mathbf{r}_{k,0}$ are highly musically salient: they are the durations of notes. The average voice in the KernScores dataset has 1.25 notes per beat, so run-length serialization of these voices requires just 1.25 tokens per beat, in contrast to the Δ tokens per beat required by raster serialization. In the next section we consider an serialization of full polyphonic scores that interleaves run-length serializations of individual homophonic voices. In the remainder of this section, we will address some technical details about homophonic voice serializations.

Informally, it is convenient to think of a voice as a musical part assigned to an individual musician. For some music—piano music in particular—it is necessary to draw a distinction between a voice and a part. Consider the piano score given in Figure 4.8. This single piano part is more comparable to a complete score than the individual parts of, for example, a string quartet. Indeed, a sophisticated musician might parse this score as four distinct voices: a high sequence of quarter and eighth notes, two middle sequences of sixteenth notes, and a low sequence of quarter notes. In the last measure, the lowest two parts combine into a single bass line of sixteenth notes. These voice divisions are indicated in the score with a combination of beams, slurs, and other visual queues. We do not model these visual indicators; instead we rely on the voice annotations provided by the KernScores dataset. These voice annotations are a unique aspect of the KernScores dataset and Humdrum format; while in principle formats like MIDI could encode this information, in practice they typically collect all notes for a single instrument into a single track, or possibly two tracks (for the treble and bass staves, as seen in the figure) in the case of piano music.

In rare cases, the distinction between voices and musical parts must also be made for stringed instruments; a notable example is Beethoven's string quartet number 14, in the fourth movement between measures 165 and 173, where the four instruments each separate into two distinct parts creating brief moments of 8-part harmony. The physical constraints of stringed instruments discourage more widespread use of these polyphonies. For vocal music,

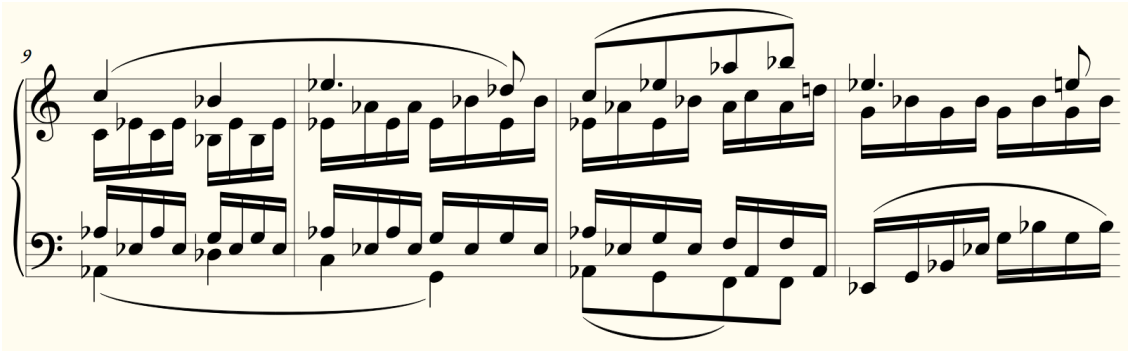


Figure 4.8: Beethoven's piano sonata number 8 (Pathetique) movement 2, from measure 9, rendered by the Verovio Humdrum Viewer. Although visually rendered on two staves, this sonata consists of four parts: a high sequence of quarter and eighth notes, two middle sequences of sixteenth notes, and a low sequence of quarter notes.

of course, physical constraints prevent intra-instrument polyphony entirely.

As Figure 4.8 illustrates, these abstract homophonic voices can weave in and out of existence. Two voices can merge with each other; a single voice can split in two; new voices can emerge spontaneously. The KernScores data provides annotations that describe this behavior. We can represent these dynamics of parts as a $V \times V$ flow matrix at each time step that describes where each part moves in the next step (V is an upper bound on the number of voices; for the KernScores corpus used in this work, we take $V = 6$). At most time steps, this flow matrix is the identity matrix. The state-based, recurrent models proposed in Section 4.2.4 can easily be adjusted to accommodate these flows. If two parts merge, sum their states; if a part splits in two, duplicate its state. These operations amount to multiplying the vector of state estimates for the parts with the flow matrix at each time step. However, we do not model the flow matrix. Because the flow matrix for piano music contains some (small) amount of entropy, we therefore exclude piano music from the results reported in Table 4.4. We do include the piano music in training.

Interleaved Polyphonic Run-Length Serialization. We now propose a serialization of full polyphonic scores that interleaves homophonic run-length serializations of each of a score's constituent voices. The idea is to serialize each voice using run-lengths (Definition

4.6) and interleave the events in each voice sequence, sorted by time. When events occur in multiple voices at the same time, we use an arbitrary numerical order assigned to each voice to determine the order of these concurrent events (concurrent events occur quite frequently, for example at the beginning of each measure). This interleaved serialization ensures that, when we sample from our model, the sequence of events for any particular voice never advances further than one note-value ahead of the partially generated sequence for any other voice. Formally, we can describe this serialization using Definition 4.7.

Definition 4.7. *Let $c_0^v, \ldots, c_{L_v}^v$ denote the temporal change points in voice v of a discretized score $\mathbf{x}$, with $L \equiv \sum_{v=1}^{V} L_v$. Define a total order on change points according to the rule $c_k^v < c_{k'}^u$ for all v, u if $k < k'$, and $c_k^v < c_k^u$ if $v < u$. For each $i \in \{1, \ldots, L\}$ let α_i and β_i denote the voice-index and voice of the i'th change point according the total ordering of change points. The interleaved voice run-length serialization of $\mathbf{x}$ is the tensor $\mathbf{s} \in (\{0,1\}^D \oplus \{0,1\}^N)^L$ defined by $\mathbf{s}_{k,0} = \frac{c_{\alpha_k+1}^{\beta_k} - c_{\alpha_k}^{\beta_k}}{\Delta} \in \mathbb{N}$ and $\mathbf{s}_{k,1} = \mathbf{x}_{c_{\alpha_k}^{\beta_k}}$.*

We can factor the distribution over scores according to this serialization as

$$q(\mathbf{s}) = \prod_{k=1}^{L} q(\mathbf{s}_{k,0}|\mathbf{s}_{1:k}) \prod_{p=1}^{2P} q(\mathbf{s}_{k,1,p}|\mathbf{s}_{1:k}, \mathbf{s}_{k,0}, \mathbf{s}_{k,1,1:p}). \tag{4.13}$$

This factorization produces a ragged frontier when sampling, where generation of any particular voice advances no further than one note-value ahead of any other voice; in contrast, the raster sampling advances with a smooth frontier, one Δ-slice of time after another. The KernScores multi-voice corpus averages approximately 5 notes per beat, so only 5 predictions per beat are required for to sample from an autoregressive model of interleaved polyphonic run-length encoded scores. This is an order of magnitude less computation than the $\Delta = 48$ predictions per beat required for a raster model.

Sparse Serializations. Another family of serializations based on MIDI has become popular in recent work on expressive modeling of MIDI performances [Oore et al., 2020, Huang et al., 2019]. Like MIDI, the idea is to serialize a score as a sequence of events. A

simple example of this type of serialization is to represent a score as a list of notes, where each note is a tuple consisting of a pitch class, voice, start time, and duration. This serialization takes advantage of the sparsity of musical scores to eliminate the number of pitch classes as a factor in serialization length; indeed, these serializations can be compared to classical methods for sparse matrix storage [Gilbert et al., 1992]. Recall that raster serializations have length $O(TPV/\Delta)$, and run-length serializations have length $O(LPV)$ (where L is the number of change points in the score). In contrast, MIDI-style serializations have length proportional (equal, for the serialization proposed here) to the number of notes; like L, the number of notes is proportional to the length of the performance T, and the number of notes is also proportional to the number of voices V. But this serialization breaks the dependence on the number of pitch classes P; the number of active pitches at any time in a score is proportional to the number of voices, not the size of the space of pitch classes. Furthermore, whereas the run-length serialization proposed in Definitions 4.6 and 4.7 rely on the homophonic voice structure of the Humdrum encoding, these sparse serializations generalize more easily to the other encodings of music discussed in Section 4.1. Sparse serializations merit strong consideration for future work on generative models of symbolic music, with the caveat that it is unclear how to combine these serializations with the relative pitch weight-sharing techniques proposed by Johnson [2017] (see Section 4.2.4).

4.2.4 Structure-Aware Models of Scores

In this section we consider neural architectures for parameterizing an autoregressive model of musical scores (Definition 4.2) using the run-length serializations discussed in Section 4.2.3. First, we consider how to encode the conditional history used as input for out models. Second, we introduce a single-voice, homophonic modeling task: estimation of the marginal distribution over a single voice in the score tensor. Learning the marginal distribution over voices is similar in spirit to classical monophonic generation tasks; homophonic composition generalizes monophonic composition to allow for chords. Results for this task are summarized in Table 4.3. Third and finally, we consider multi-voice, polyphonic estimation of the joint

distribution over scores. Results for this task are summarized in Table 4.4.

Encoding the History

Like the choice of factorization, there are many options for encoding the history of a score used for prediction of values $\mathbf{s}_{1:k}, \mathbf{s}_{k,0}, \mathbf{s}_{k,1,1:p}$ (the serialization described by Definition 4.7). One natural choice of encoding is the serialization used to define the autoregressive factorization (Equation (4.13)) [Oore et al., 2020, Huang et al., 2019] but the serialization used for autoregressive factorization and the encoding of the history in the conditional distributions $q(\mathbf{s}_{k,0}|\mathbf{s}_{1:k})$ and $q(\mathbf{s}_{k,1,p}|\mathbf{s}_{1:k}, \mathbf{s}_{k,0}, \mathbf{s}_{k,1,1:p})$ are not inherently connected. Indeed, while serialization is necessary for autoregressive factorization, these sequences are lengthy, somewhat arbitrarily ordered (see the discussion at the end of Section 4.1.1) and can obscure the structure of a score. Another natural choice of encoding is a masked version of the score tensor $\mathbf{x}$ (Definition 4.2). See Chapter 2, Section 2.3 for a discussion of masking in neural autoregressive models, as well as Germain et al. [2015]. Rasterization of the time axis of the score tensor means that this encoding is also quite lengthy.

In this work, we use a masked run-length encoding of the score tensor to encode history: $\mathbf{r}$ (Definition 4.6) for single-voice modeling, and $\mathbf{e}$ (Definition 4.5) for multi-voice modeling. Even using a run-length encoding, full musical scores are quite lengthy. For both the single-voice and multi-voice tasks, we truncate the history at a fixed number of time indices. We explore several history lengths in the experiments and observe diminishing improvement in quantitative results for windows beyond the range of 10-20 run-lengths: see Experiment group (1,2,6,7) in Table 4.4. These window sizes are generous compared to the lengths of traditional n-gram models, but quite short compared to modern neural models of symbolic music inspired by work in language modeling [Huang et al., 2019] (2,048 tokens, using a sparse serialized encoding of the history; see Section 4.2.3).

For single-voice modeling we also consider the encoding $\tilde{\mathbf{r}} \equiv \mathbf{e}_{:,v}$, a single-voice slice of the run-length encoded score tensor. Observe that $\tilde{\mathbf{r}} \neq \mathbf{r}$ because the note durations $\mathbf{r}_{k,0}$ are subdivided in $\tilde{\mathbf{r}}$ by the change points of other voices in the full score. With the single-

voice encoding $\mathbf{r}$, simple linear filters can be learned that are sensitive to particular rhythmic patterns: e.g., groups of four eighth notes, or three triplet-quarter notes. This is not the case for $\tilde{\mathbf{r}}$, for which a single rhythmic pattern can be broken up in a variety of ways. These observations might lead us to consider raster encodings for multi-voice history, restoring effectiveness of simple linear filters at a cost of increasing the length of the history encoding. We find that recurrent network architectures for the single-voice task are unhampered when retrained on $\tilde{\mathbf{r}}$: compare Experiments 21 and 22 in Table 4.3. Performance falls slightly when learning on $\tilde{\mathbf{r}}$, but this is to be expected because history interspersed with continuations is effectively a shorter-length history. So at least for recurrent networks, we find that trading off linearity for a reduction in sequence length is desirable.

Single-voice models

Using the autoregressive factorization (Equation 4.12) of the run-length serialization given by Definition 4.6, we explore fully connected, convolutional, and recurrent models for learning the conditional distributions $q(\mathbf{r}_{k,0}|\mathbf{r}_{1:k})$ over note-values and $q(\mathbf{r}_{k,1,p}|\mathbf{r}_{1:k}, \mathbf{r}_{k,0}, \mathbf{r}_{k,1,1:p})$ over pitches. We build separate models that estimate $\mathbf{r}_{k,0}$ and $\mathbf{r}_{k,1,p}$, with respective losses Loss_t and Loss_n. In the remainder of this section, we consider opportunities to exploit the structure of music by sharing weights in our models. Quantitative results for these single-voice models are summarized in Table 4.3.

Autoregressive modeling. To build a generative model over conditionally stationary sequential data, we make the conditional stationarity assumption $q(\mathbf{r}_k|\mathbf{r}_{1:k}) = q(\mathbf{r}_{k'}|\mathbf{r}_{1:k'})$ for all $k, k' \in \mathbb{N}$ (see Section 2.3). We can then learn a single family of conditional distributions that share model parameters across all time translations. Scores are not quite conditionally stationary; the distribution of notes and rhythms varies substantially depending on the sub-position within a beat. To address this, we follow the lead of Johnson [2017] and Hadjeres et al. [2017] and augment our history tensor with a one-hot location feature vector ℓ that

indicates the subdivision of the beat at which we are presently making predictions.[5] Compare the loss of duration models (Loss_t) with and without these features in Experiment pairs (3,4), (6,7), (10,11), (12,13), and (15,16) in Table 4.3.

Relative pitch. We can perform a similar weight-sharing scheme with pitches as we do with time. Instead of building an individual predictor for each pitch conditioned on the notes in the history tensor, we adopt an idea proposed by Johnson [2017]: build a single predictor that conditions on a shifted version of the history tensor centered around the note we want to predict. Convolving this predictor over the pitch axis of the history tensor lets us make a prediction at each note location, as visualized by Figure 4.9. As with time, the distribution over notes is not quite conditionally stationary. For example, a truly relative predictor would generate notes uniformly across the note-class axis, whereas the actual distribution of notes concentrates around middle C. Therefore we augment our history tensor with a one-hot feature vector $\mathbf{1}_p$ that indicates the pitch p for which we are making a prediction. This allows us to take full advantage of all available information when making a prediction, while borrowing strength from shared harmonic patterns in different keys or octaves. We compare absolute pitch-indexed classifiers ($\mathbf{lin}_p$) to a single, relative pitch classifier (**lin**) in Table 4.3: compare the loss of pitch models (Loss_p) in Experiment groups (2,3,4), (5,6,7), (8,9,10), (11,12,13), and (15,16) in Table 4.3. Relative pitch models serve a similar purpose to key-signature normalization Liang et al. [2017] or data augmentation via transposition Hadjeres et al. [2017]. Building this invariance into the model offers an alternative approach that avoids data preprocessing or the introduction of hyper-parameters. We find that training with transpositions in the range ± 5 semi-tones yields no performance increase for relative pitch models.

Pitch embeddings. Borrowing the concept of a word embedding from natural language processing, we consider learned embeddings $\mathbf{c}$ of the pitch vectors $\mathbf{r}_{k,1}$ ($\mathbf{e}_{k,v,1}$ for the multi-voice models). For recurrent models, we do not see performance benefits to learning these

[5] Location can always be computed from the full history tensor. But we truncate the history, so this information is lost unless it is explicitly reintroduced.

#	Params	Rhythm Model	Notes Model	Loss_t	Loss_p	$\text{Loss}_{\text{total}}$
1	112	$\hat{\mathbf{r}}_{k,0} = \mathbf{bias}_0$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{bias}_{1,p}$	2.92	7.15	10.07
2	21k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin}(\mathbf{r}_{(1)})$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin}_p(\mathbf{r}_{(1)}, \mathbf{r}_+)$	2.00	6.05	8.05
3	9k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin}(\mathbf{r}_{(1)})$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin}(\mathbf{r}_{(1)}, \mathbf{r}_+)$	2.00	4.29	6.29
4	11k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin}(\mathbf{r}_{(1)}, \ell)$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin}(\mathbf{r}_{(1)}, \mathbf{r}_+, \mathbf{1}_p)$	1.83	4.29	6.12
5	149k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin} \circ \mathbf{fc}(\mathbf{r}_{(1)})$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin}_p \circ \mathbf{fc}(\mathbf{r}_{(1)}, \mathbf{r}_+)$	1.99	3.93	5.92
6	135k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin} \circ \mathbf{fc}(\mathbf{r}_{(1)})$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin} \circ \mathbf{fc}(\mathbf{r}_{(1)}, \mathbf{r}_+)$	1.99	4.07	6.05
7	172k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin} \circ \mathbf{fc}(\mathbf{r}_{(1)}, \ell)$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin} \circ \mathbf{fc}(\mathbf{r}_{(1)}, \mathbf{r}_+, \mathbf{1}_p)$	1.80	3.90	5.70
8	72k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin}(\mathbf{r}_{(5)})$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin}_p(\mathbf{r}_{(5)}, \mathbf{r}_+)$	1.86	6.05	7.91
9	36k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin}(\mathbf{r}_{(5)})$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin}(\mathbf{r}_{(5)}, \mathbf{r}_+)$	1.86	3.91	5.76
10	38k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin}(\mathbf{r}_{(5)}, \ell)$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin}(\mathbf{r}_{(5)}, \mathbf{r}_+, \mathbf{1}_p)$	1.73	3.91	5.63
11	418k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin} \circ \mathbf{fc}(\mathbf{r}_{(5)})$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin}_p \circ \mathbf{fc}(\mathbf{r}_{(5)}, \mathbf{r}_+)$	1.64	3.26	4.90
12	497k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin} \circ \mathbf{fc}(\mathbf{r}_{(5)})$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin} \circ \mathbf{fc}(\mathbf{r}_{(5)}, \mathbf{r}_+)$	1.64	3.16	4.80
13	535k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin} \circ \mathbf{fc}(\mathbf{r}_{(5)}, \ell)$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin} \circ \mathbf{fc}(\mathbf{r}_{(5)}, \mathbf{r}_+, \mathbf{1}_p)$	1.59	3.10	4.69
14	228k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin} \circ \mathbf{fc}(\mathbf{f}(\mathbf{r}_{(5)}), \ell)$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin} \circ \mathbf{fc}(\mathbf{c}(\mathbf{r}_{(5)}), \mathbf{r}_+, \mathbf{1}_p)$	1.58	3.05	4.63
15	134k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin}(\mathbf{r}_{(10)})$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin}(\mathbf{r}_{(10)}, \mathbf{r}_+)$	1.83	6.05	7.88
16	71k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin}(\mathbf{r}_{(10)}, \ell)$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin}(\mathbf{r}_{(10)}, \mathbf{r}_+, \mathbf{1}_p)$	1.71	3.83	5.53
17	372k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin} \circ \mathbf{fc}(\mathbf{f}(\mathbf{r}_{(10)}), \ell)$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin} \circ \mathbf{fc}(\mathbf{c}(\mathbf{r}_{(10)}), \mathbf{r}_+, \mathbf{1}_p)$	1.55	3.00	4.55
18	250k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin} \circ \mathbf{conv}_5(\mathbf{f}(\mathbf{r}_{(10)}), \ell)$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin} \circ \mathbf{conv}_5(\mathbf{c}(\mathbf{r}_{(10)}), \mathbf{r}_+, \mathbf{1}_p)$	1.55	3.01	4.56
19	769k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin} \circ \mathbf{conv}_{5,3}(\mathbf{f}(\mathbf{r}_{(10)}), \ell)$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin} \circ \mathbf{conv}_{5,3}(\mathbf{c}(\mathbf{r}_{(10)}), \mathbf{r}_+, \mathbf{1}_p)$	1.50	2.92	4.42
20	342k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin} \circ \mathbf{rnn}(\mathbf{r}_{(10)}, \ell)$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin} \circ \mathbf{rnn}(\mathbf{r}_{(10)}, \mathbf{r}_+, \mathbf{1}_p))$	1.48	2.89	4.37
21	283k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin} \circ \mathbf{rnn}(\mathbf{f}(\mathbf{r}_{(10)}), \ell)$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin} \circ \mathbf{rnn}(\mathbf{c}(\mathbf{r}_{(10)}), \mathbf{r}_+, \mathbf{1}_p))$	1.48	2.88	4.36
22	301k	$\hat{\mathbf{r}}_{k,0} = \mathbf{lin} \circ \mathbf{rnn}(\mathbf{f}(\tilde{\mathbf{r}}_{(10)}), \ell)$	$\hat{\mathbf{r}}_{k,1,p} = \mathbf{lin} \circ \mathbf{rnn}(\mathbf{c}(\tilde{\mathbf{r}}_{(10)}), \mathbf{r}_+, \mathbf{1}_p))$	1.59	2.93	4.52

Table 4.3: Single-voice (homophonic) results. Loss is the cross-entropy described in Section 4.2.2. Loss_t and Loss_p are the decompositions this loss for the distributions of $\mathbf{r}_{k,0}$ and $\mathbf{r}_{k,1}$ in Equation (4.12). For succinctness, define $\mathbf{r}_{(m)} \equiv \mathbf{r}_{k-m:k}$ (a truncated history of length k) and $\mathbf{r}_+ \equiv \mathbf{r}_{k,0} \oplus \mathbf{r}_{k,1,1:p}$ (the current frame, masked above pitch p). $\mathbf{lin}_p$ indicates a log-linear classifier (softmax for $\hat{\mathbf{r}}_{k,0}$ and sigmoid for $\hat{\mathbf{r}}_{k,1,p}$); $\mathbf{lin}$ indicates the relative pitch log-linear classifier; inputs $\mathbf{1}_p$ indicate pitch-class features. The inputs ℓ indicate location features. $\mathbf{fc}$ indicates a fully connected layer. $\mathbf{c}$ indicates learned pitch embeddings and $\mathbf{f}$ indicates fixed (octave) embeddings. $\mathbf{conv}_k$ indicates 1d convolution of width k. $\mathbf{conv}_{k_1,k_2}$ indicates two layers of convolutions ($\mathbf{conv}_{k_1,k_2} = \mathbf{conv}_{k_2} \circ \mathbf{conv}_{k_1}$). $\mathbf{rnn}$ indicates a recurrent layer. All hidden layers are parameterized with 300 nodes. Models were regularized with early stopping.

embeddings: compare Experiments 20 and 21 in Table 4.3. However, we do find that we can learn compact embeddings (16 dimensions for the experiments presented in this paper) without sacrificing performance. Using these embeddings reduces computational cost, and low dimensional embeddings may be helpful when adapting these models for conditional

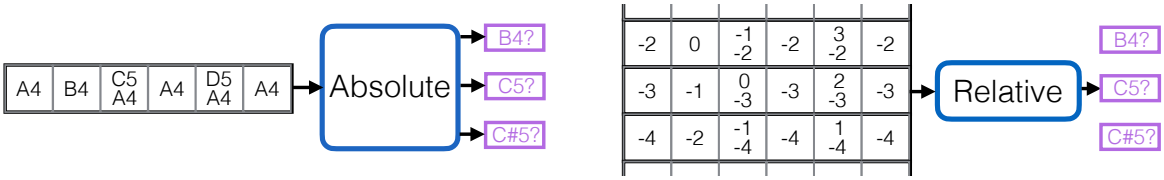


Figure 4.9: Left: an absolute pitch predictor uses a distinct model for each pitch-class. Right: a relative pitch predictor learns a single model and computes a prediction for each pitch class based on a translated view of the conditional history. For example, an absolute model predicts the presence or absence of pitch-classes C#5, C5, B4, etc. in the current event $\mathbf{r}_k$ (Definition 4.6) given the presence of A4 in the previous event $\mathbf{r}_{k-1}$. In contrast, a relative model predicts the presence or absence of C#5 given $\mathbf{r}_{k-1}$ contained a note 4 steps below, C5 given $\mathbf{r}_{k-1}$ contained a note 3 steps below, B4 given $\mathbf{r}_{k-1}$ contained a note 2 steps below, etc.

sampling tasks using Langevin dynamics (looking ahead to Chapter 5). We also find that using a 12 dimensional fixed embedding of pitches $\mathbf{f}$, in which we quotient each pitch class by octave, reduces overfitting for the rhythmic model while preserving predictive accuracy.

Multi-voice models

Using the autoregressive factorization (Equation 4.11) of the run-length serialization given by Definition 4.5, we now explore ways to capture the correlations between voices in a full score. We build models of the conditional distributions $q(\mathbf{s}_{k,0}|\mathbf{s}_{1:k})$ over note-values and $q(\mathbf{s}_{k,1,p}|\mathbf{s}_{1:k}, \mathbf{s}_{k,0}, \mathbf{s}_{k,1,1:p})$ over pitch classes, with respective cross-entropy losses Loss_t and Loss_p. Quantitative results for these experiments are summarized in Table 4.4. The same structural observations that we made for the single-voice models apply to multi-voice modeling; all multi-voice models considered in this chapter use the three weight-sharing schemes discussed in the previous section. We also propose an additional weight-sharing opportunity that is specific to the multi-voice modeling task, voice coupling, which is described at the end of this section.

The effectiveness of recurrent models for the single-voice modeling task, and representational considerations (Section 4.2.4, Encoding the History) motivate us to consider extensions

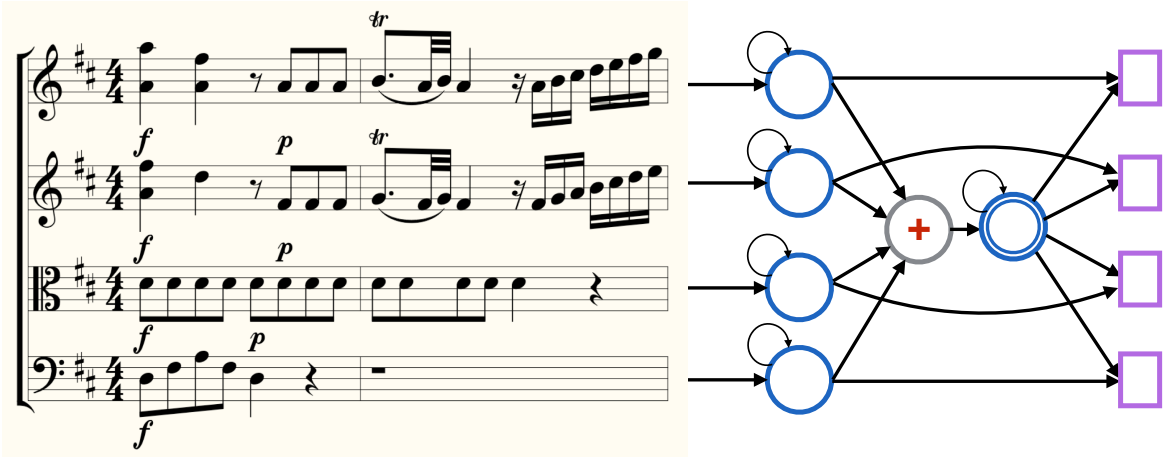


Figure 4.10: Coupled state estimation of Mozart's string quartet number 2 in D Major, K155, movement 1, from measure 1, rendered by the Verovio Humdrum Viewer. A recurrent network models the state $h_{k,v}$ of each voice v at step k, based on the previous state $h_{k-1,v}$ and the current content of the voice. Another recurrent network models of the global state g_k of the score at step k based on the previous global state g_{k-1} and a sum of the current states of each voice. Subsequent notes (purple) in each voice are predicted using features of the global state and the state of the relevant voice. See Equations 4.14 for a formal description of this model.

of the recurrent architecture to capture structure in the multi-voice setting. Figure 4.10 illustrates one natural extension of the standard recurrent neural network to multiple, concurrent voices. The idea is to construct a hierarchical state-based model, where the state of each part evolves individually, and these states contribute the global evolving state of the score. Formally,

$$\begin{aligned} h_{k,v}(\mathbf{e}) &\equiv \mathbf{a}\left(W_v^\top h_{k-1,v}(\mathbf{e}) + W_e^\top \mathbf{c}\left(\mathbf{e}_{k,v}\right)\right), \\ g_k(\mathbf{e}) &\equiv \mathbf{a}\left(W_g^\top g_{k-1}(\mathbf{e}) + W_{hv}^\top \sum_u h_{k,u}(\mathbf{e})\right). \end{aligned} \tag{4.14}$$

The first equation describes a standard recurrent neural network that builds a state estimate $h_{k,v}$ of a voice v at time index k based on transition weights W_v, an input embedding $\mathbf{c}$, input weights W_e, and non-linear activation $\mathbf{a}$ (we use a ReLU activation). We integrate the state of each voice (weights W_{hv}) into a global state g_k given the previous global state

g_{k-1} (weights W_g). Because voice order is arbitrary in our dataset, we sum (i.e., pool) their states before feeding them into the global network. At each time k, we use the learned state of each voice together with the global state to make a note-value prediction: $\hat{\mathbf{s}}_{k,0} = \mathbf{lin}(h_{k,\beta_k}(\mathbf{e}), g_k(\mathbf{e}))$, where **lin** is a log-linear classifier. We make pitch predictions $\hat{\mathbf{s}}_{k,1,p} \in \{0, 1\}$ using the same architecture. We learn a single, relative-pitch classifier for all multi-voice experiments (see Section 4.2.4, Relative Pitch). We do not share weights between the note-value and pitch models.

An alternate extension of a recurrent voice model to scores directly integrates the state of the other voices' states into each individual voice's state, resulting in a distributed state architecture

$$h_{k,v}(\mathbf{e}) = \mathrm{a}\left(W_v^\top h_{k-1,v}(\mathbf{e}) + W_x^\top \mathbf{c}\left(\mathbf{e}_{k,v}\right) + W_{hv}^T \sum_u h_{k,u}(\mathbf{e})\right). \tag{4.15}$$

At each time k, for each voice v, we use the learned state of voice v to make a note-value prediction $\hat{\mathbf{s}}_{k,0} = \mathbf{lin}(h_{k,\beta_k}(\mathbf{e}))$, where **lin** is a log-linear classifier. We make predictions for $\hat{\mathbf{s}}_{k,1,n} \in \{0, 1\}$ using the same architecture and again we do not share weights between the note-value and pitch models.

We find that the distributed architecture underperforms the hierarchical architecture (see Table 4.4; Experiments 2 and 3) although the difference is not so extreme that we should consider this result conclusive. For the hierarchical model, we can consider whether the global state representation is as sensitive to history-length as the voices. Could we make successful predictions using only the final state of each voice, rather than coupling the states at each step? Experiment group (4,5,6) in Table 4.4 suggests that this is not the case: we observe significant gains by integrating voice information at each time step.

Voice coupling. Decomposing a score into multiple voices presents us with an opportunity to share weights between voice models by learning a single set of weights W_v in Equation (4.14), rather than learning unique voice-indexed weights W_{v_i} for each voice v_i. Indeed, because voice indices are arbitrary, the weights W_{v_i} will converge to the same values for all i;

#	History (voice/global)	Architecture	Loss (total)	Loss_t (time)	Loss_p (notes)
1	3 / 3	hierarchical	14.05	5.65	8.40
2	5 / 5	hierarchical	13.40	5.35	8.04
3	5	distributed	13.82	5.41	8.41
4	10 / 1	hierarchical	13.20	5.22	7.98
5	10 / 5	hierarchical	12.94	5.13	7.81
6	10 / 10	hierarchical	12.87	5.12	7.75
7	20 / 20	hierarchical	12.78	5.01	7.76
8	10	independent	18.63	6.56	12.08

Table 4.4: Multi-voice (polyphonic) results. Loss is the cross-entropy described in Section 4.2.2. Loss_t and Loss_p are decompose this loss into contributions from the models of $q(\mathbf{s}_{k,0}|\mathbf{s}_{1:k})$ and $q(\mathbf{s}_{k,1,p}|\mathbf{s}_{1:k}, \mathbf{s}_{k,0}, \mathbf{s}_{k,1,1:p})$ respectively (using the factorization described by Equation (4.13)). The hierarchical architecture is defined by Equations (4.14). The distributed architecture is defined by Equation (4.15). Voice and global history refer to the number of time steps used to construct the states $h_{k,v}$ and g_k respectively. Experiment 8 is a baseline where the voice models are completely decoupled (equivalent to single-voice Experiment 22 in Table 4.3; the average number of voices per score is 4.12). Results are reported on non-piano test set data (see the discussion of Homophonic Run-Length Serializations in Section 4.2.3).

sharing a single set of weights W_v accelerates learning by enforcing this property. All score models presented in Table 4.4 share these weights. Extending a loose analogy between recurrent neural networks and hidden Markov models, the coupled recurrent models considered in this section can be compared to factorial hidden Markov models [Ghahramani and Jordan, 1995]. A crucial distinction is that the distributed latent state of a coupled recurrent model is determined by the distributed input structure of a score, whereas the distributed structure of a factorial hmm only appears in the latent state.

4.2.5 Revisiting Evaluation of Generative Models

In Section 4.2.4 we studied a variety of neural architectures for modeling musical scores under the cross entropy metric developed in Section 4.2.2. This allowed us to compare the relative performance of a wide variety of models, but doesn't offer much insight into their

quality in an absolute sense. In this section, we present the results of a user study, which seeks to quantify the absolute performance of one of our best models: Experiment 6 in Table 4.4. We conclude with an open-ended discussion about techniques for evaluating generative models of symbolic music. Better methodology is clearly needed, but the path forward is unclear. We contrast the current state of evaluation in the music domain with more mature evaluation methodology in the vision and language domains.

A User Study

To gain insight into the quality of samples from our models, we recruited twenty study participants from the Computer Science and Engineering graduate program at the University of Washington. Participants were asked to listen to a variety of audio clips, each synthesized from either a real composition or from generated scores sampled using the model presented in Experiment 6 in Table 4.4. Our models do not predict tempo, so each score was synthesized at a tempo of 120bpm. For each clip, participants were asked to judge whether the clip was written by a computer or by a human composer, following a procedure comparable to Pearce and Wiggins [2001]. Each participant was asked to listen to 5 sets of 10 audio clips, which varied in length from a 10-index slice of the run-length encoded score tensor $\mathbf{e}$ (2-4 seconds; the length of history conditioned on by the model) to a 50-index slice of the score tensor (10-20 seconds). Each participant was presented with their own set of audio clips, randomly sampled from either the training data or generated output from the model.

We also asked the participants to identify if they recognized a specific audio clip, because we were concerned that knowledge of the classical music canon could confound the model's ability to fool a human. Despite several of our study participants having extensive musical training, only one user reported recognition of a single clip in our study. This may be partially explained because our models do not predict tempo; performing each score at 120bpm could obscure otherwise recognizable pieces. However, this also highlights a limitation of this study, because all audio clips (real or artificial) could sound foreign when normalized to this common tempo. Participants were informed of the tempo normalization, but it is not

clear how and to what extent they were able to act upon this knowledge when making their judgements.

Users were given the following prompt before beginning the study:

> *This is a musical Turing test. You will be presented with a selection of audio clips, beginning with short clips and progressing to longer clips. For each audio clip, you will be asked whether you believe the clip was composed by a human or a computer. Half the clips you will be presented with belong in each category. This data contains many famous classical compositions, ranging from the Renaissance to early 20th century. If you specifically recognize a piece, please let me know. Finally, all recordings you hear–both human and artificial–are performed at a tempo of 120bpm.*

Additionally, we asked users two questions about their background:

- Do you self-identify as musically educated? (8 responded 'yes')
- Do you self-identify as educated in machine learning? (13 responded 'yes')

Table 4.5 summarizes the results of this listening study, including conditional results for the educated subgroups. Participants become more confident in their judgements of the longer clips, but even among the longest clips (around 20 seconds) participants often identified an artificial clip as a human composition. These results superficially suggest that we have done well in modeling the short-term structure scores (we make no claims to have captured long-term structure; indeed, the truncated history input to our models precludes this).

Concluding Thoughts on Evaluation Methodology

In Sections 4.2.4 and 4.2.5, we have attempted to evaluate generative models from two perspectives: an automated cross-entropy metric and a user study. User studies are sometimes considered a gold standard for evaluation, but it unclear whether humans are good at evaluating the outputs of a generative model. In the music domain, see Pearce and Wiggins [2007],

Length	10	20	30	40	50
All	5.3	5.7	6.6	6.7	6.8
Music	4.9	6.0	6.4	6.9	7.0
ML	4.8	5.5	6.2	6.7	6.8

Table 4.5: Qualitative evaluation of a hierarchical model conditioned on 10 steps of history: Experiment 6 in Table 4.4. Twenty participants were asked to judge 50 audio clips of varying length (where length indicates a number of time indices in the score tensor $\mathbf{e}$; see Definition 4.5). The scores indicate participants' average correct discriminations out of 10 (5.0 would indicate random guessing; 10.0 would indicate perfect discrimination). The categories indicate breakdowns for listeners who identified as educated in music or educated in machine learning.

Jordanous [2012] for criticism of musical Turing tests like the one presented here. One perspective on this problem is that, while humans are generally good at rejecting implausible compositions, they may not properly penalize a model that lacks creativity or diversity of outputs [Hashimoto et al., 2019]. The foreign sounds of synthesis and tempo-normalization introduced by the user study presented here magnify the fallibility of human judgement.

One possible way forward is demonstrated by the computer vision community, which has made tremendous advances in generative modeling using automated metrics. In particular, the Inception Distance [Salimans et al., 2016] and Frechet Inception Score [Heusel et al., 2017] are regularly reported on standardized CIFAR-10 [Krizhevsky, 2009] and ImageNet [Krizhevsky et al., 2012] test sets. Because these scores are completely automated they can be compared across papers, fostering competition where teams of researchers strive outperform each others' best efforts. In contrast, user studies generally require head-to-head comparisons, so the same team that invents a new method is also responsible for running the study that compares it to previous work; this creates perverse incentives, even for the most ethical researcher. The language modeling community has also begun to focus on automated metrics, with recent demonstrations that aggressive minimization of a cross-entropy metric leads to qualitatively strong language generation results [Radford et al., 2019, Brown et al., 2020].

Unfortunately, it may be difficult for the symbolic music modeling community to follow the lead of the vision and language communities using automated metrics. The problem is that there are no standardized corpora on which to report metrics. The best current example of a standardized corpus is the Bach chorales dataset, but even with this dataset differences in discretization make cross-entropy comparisons across papers difficult [Huang et al., 2017]. Furthermore, the richness of musical scores works against us: if one work models tempo but not dynamics, and another models dynamics but not tempo, it is difficult to imagine an automated metric (or a user study) that can compare between them. Not only are there no standard datasets, there is no self-contained vocabulary for musical scores. The community cannot even agree on a vocabulary of pitches: most work encodes pitch using 12-tone equal temperament, but others encode note names (e.g., F♯ or G♭) that distinguish between enharmonic equivalents [Hadjeres et al., 2017]. In this sense, the music community faces similar challenges to the language community discussed in Section 4.1.4, which have been resolved by adopting subword tokenizations that can encode any unicode text, in contrast to open-ended word-based vocabularies. But it is unclear what the musical analog to an alphabet of characters would be.

4.3 Classification Models for Musical Scores

In this section, we turn our attention from generative models to discriminative, classification models. We focus in particular on a composer attribution task: predicting the composer of a given score. Before attending to the details of this task, we ought to ask: why should we care about this task? Nearly every musical score has composer metadata attached. And in rare cases where this information has been lost, a thoughtful (and interpretable) scholarly musicological analysis is likely to be more interesting than the prediction of a black-box classifier. In light of this, we motivate the composer attribution task in two ways: (1) it is an archetypical example of a discriminative modeling task for musical scores, with unambiguous labels, and (2) an end-to-end, differentiable neural classifier can be useful tool for guiding a posterior sampler, using the methods proposed in Chapter 5.

Models for attributing composers to musical scores have been extensively studied in the music information retrieval community. The composer classification question has been posed for a variety of corpora, from Renaissance composers [Buzzanca, 2002, Brinkman et al., 2016], to the narrow (and challenging) case of Haydn and Mozart string quartets [Hillewaere et al., 2010, Herlands et al., 2014, Van Kranenburg and Backer, 2005, Kempfert and Wong, 2020], and to various collections of classical era compositions (most of the other papers discussed in Section 4.3.1). In this work we study an expansive collection of scores, described in Section 4.3.2, from 13th century sacred music by Guillaume Du Fay to 20th century ragtimes by Scott Joplin.

A major challenge for the composer attribution task is learning from limited data. While the corpus considered here is larger than most, this is mostly due to the number of composers considered (19): for specific composers, we have at most 466 scores (Bach) and as few as 22 (Japart). Small datasets are an inherent challenge for composer attribution: the corpus used in this work contains, for example, all of the Bach chorales and all of the Mozart string quartets. We cannot resurrect these composers and have them write us more scores to include in our corpus. This situation contrasts starkly with many learning problems, where substantial progress can be made by collecting massive datasets and exhaustively training an expressive model (usually a deep neural network) with "big data." Further complicating this task, an individual score is itself a high dimensional object: the average score in our corpus consists of thousands of notes, each of which is encoded as a high dimensional vector to represent its pitch and value. Learning from a small number of examples in a high dimensional space is a formidable problem; thus much work on composer classification focuses on feature engineering, feature selection, dimensionality reduction, or some combination of these approaches to construct low-dimensional representations of scores to learn from.

In this work we take a different approach: we dispense with feature engineering and explore end-to-end classifiers that operate directly on scores (Section 4.3.4). Specifically, we investigate shallow convolutional neural networks with an aggressive pooling operation. In this setting, all but the most impoverished linear classifiers achieve 100% training accu-

racy. We rely on implicit regularization introduced by the network structure and first-order optimization with early stopping to avoid overfitting to training data. While theoretical understanding of such an approach is in its early stages [Soudry et al., 2018], we find empirically that this works quite well for composer attribution (Section 4.3.5).

4.3.1 Related Work

The earliest work on composer classification analyzed highly preprocessed corpora of melodic fragments [Pollastri and Simoncelli, 2001, Buzzanca, 2002]. Much of the subsequent work focuses on engineering features to summarize full scores. These approaches can be broadly categorized, using terminology introduced by Hillewaere et al. [2009], into "global" summarization approaches which compute small sets of summary statistics as a feature set for each score [Van Kranenburg and Backer, 2005, Kaliakatsos-Papakostas et al., 2010, Herlands et al., 2014, Herremans et al., 2016, Sadeghian et al., 2017, Kempfert and Wong, 2020, Brinkman et al., 2016] and local "event" featurizations which extract n-gram counts of a score as features [Wołkowicz et al., 2008, Hillewaere et al., 2010, Kaliakatsos-Papakostas et al., 2011, Hontanilla et al., 2013, Wolkowicz and Keselj, 2013]. There is also a line of work that applies compression-based dissimilarity metrics [Anan et al., 2012, Takamoto et al., 2016, 2018] to this task, which offers a substantially different perspective on classification problems. Our work can be seen as a unified framework for learning global and event features. In Section 4.3.4 we will draw analogies between linear convolutional filters and n-gram features, and also demonstrate how convolutional models can express many popular global features. We also introduce a global pooling operation that can be interpreted as an counter that tracks the number of occurrences of learned features, which is directly analogous to the count and ratio statistics that comprise the bulk of metrics used in human-engineered featurizations.

The present work is most similar in spirit to Buzzanca [2002] and Velarde et al. [2016]. Following Buzzanca [2002], we adopt an end-to-end approach to feature learning using neural architectures. In contrast with Buzzanca [2002], we learn on full scores with minimal preprocessing and consider a multi-class classification task over a broad variety of composers; this

approach is made possible by modern hardware unavailable to researchers in 2002. We also take a more systematic approach to architecture exploration, and identify effective architectures that are simpler than hybrid convolutional-recurrent approach taken by Buzzanca [2002]. Like Velarde et al. [2016], we exploit structure in musical scores using convolutional models. But where Velarde et al. [2016] use a fixed Morlet or Gaussian convolution filters, the convolutional filters in this work are parameterized and learned from the data to maximize classification accuracy. We also explore multi-layer "deep" convolutional models and demonstrate improvements using such architectures versus the single layer of convolutions explored by Velarde et al. [2016].

4.3.2 Corpus and Data Representation

We train and evaluate models on a corpus of 2,500 scores spanning five centuries of choral, piano, and chamber compositions from the KernScores collection ([Sapp, 2005]; see Section 4.1.2 for an example of the Humdrum encoding used by KernScores). An overview of this collection is provided in Table 4.6. In this work, we consider each movement of a multi-movement composition to be a distinct score. Our models extract only the note data (pitch, note-value, and voicing) from scores, ignoring all other markings such as time signatures, key signatures, tempo markings, instrumentation, and movement names. For the Renaissance composers in this collection (Du Fay through Japart) we shorten the length of all note-values by a factor of 4 to crudely account for the shift in duration conventions between mensural and modern notation [Cumming, 2000].

We represent a score by lossless encoding of its pitch, voice, and note-value contents, transliterated from a Humdrum file to a binary representation suitable for input to a neural network.

Definition 4.8. *A transliterated Humdrum score of length L, consisting of V voices, with N distinct pitch classes and D distinct note-values is a binary tensor* $\mathbf{w} \in \{0,1\}^{L\times V\times(N+D+1)}$

such that

$$\mathbf{w}_{k,v,n} = 1 \quad \textit{if pitch } n \textit{ occurs at row (time index) } k \textit{ in column (voice index) } v, \tag{4.16}$$

$$\mathbf{w}_{k,v,N+d} = 1 \quad \textit{if note-value } d \textit{ occurs at row } k \textit{ in column } v, \tag{4.17}$$

$$\mathbf{w}_{k,v,N+D} = 1 \quad \textit{if notes continue at row } k \textit{ in column } v. \tag{4.18}$$

For the KernScores corpus used in this work, $N = 78$ and $D = 55$.

It may be helpful to compare this encoding $\mathbf{w}$ with the run-length encoding $\mathbf{r}$ described in Section 4.2.3, Definition 4.5. Definition 4.8 more faithfully transliterates the Humdrum format described in Section 4.1.2. In contrast to run-length encoding $\mathbf{r}$, which explicitly encodes the length of elapsed time between each index k and $k+1$, the encoding $\mathbf{w}$ instead explicitly encodes the durations of notes. In the $\mathbf{w}$-encoding, run-lengths are implicit: the elapsed time between $\mathbf{w}_k$ and $\mathbf{w}_{k+1}$ can be calculated as the distance between the time at $\mathbf{w}_k$ and the minimum end-time of all active notes in the score at time k (end-times can be calculated because the durations of notes are explicitly encoded in $\mathbf{w}$. In contrast, for the run-length encoding $\mathbf{r}$, the duration of a note is implicit but this durations can be calculated by summing the run-lengths of the time indices it spans.

As we saw in Section 4.2, converting from Humdrum text to a structured tensor encoding allows us to exploit structure in musical scores. In Section 4.3.4, we will use this structure to write convolution operations along the time and pitch axes of the data tensor $\mathbf{w}$. Encoding pitches with binary indicators in N-dimensional vectors is consistent with piano roll representations [Velarde et al., 2016] but departs from the example of Buzzanca [2002], which encodes pitch as a single numerical magnitude. The binary pitch encoding is required to support convolutions along the pitch domain, which we will introduce later in the models defined by Equations (4.27) and (4.28).

Binary note-value encoding also differs from the numerical magnitude encoding used by Buzzanca [2002]. We will not introduce models that convolve over durations (there is no translation invariant structure to exploit) so the motivation above for representing pitches

Composer	Dates	Sub-Collection	Scores
Du Fay	1397-1474	Choral	35
Ockeghem	1410-1497	Choral	98
Busnois	1430-1492	Choral	68
Martini	1440-1497	Choral	122
Compere	1445-1518	Choral	27
Josquin	1450-1521	Choral	423
de la Rue	1452-1518	Choral	178
Orto	1460-1529	Choral	43
Japart	1474-1507	Choral	22
Corelli	1653-1713	Trio Sonatas	188
Vivaldi	1678-1741	Concertos	33
Bach	1685-1750	Chorales	370
		Well-Tempered Clavier	96
D. Scarlatti	1685-1757	Keyboard Sonatas	59
Haydn	1732-1809	String Quartets	209
Mozart	1756-1791	Piano Sonatas	69
		String Quartets	82
Beethoven	1770-1827	Piano Sonatas	102
		String Quartets	67
Hummel	1778-1837	Preludes	24
Chopin	1810-1849	Preludes and Mazurkas	76
Joplin	1868-1917	Ragtimes	47

Table 4.6: Details of the KernScores collection used for training and evaluation in this paper.

with indicators does not apply to durations. Rather, we are motivated by the observation that note-values of similar duration may not be more alike in any musical sense than note-values with less similar duration. We avoid imposing this notion of similarity a-priori by encoding durations as categorical indicators: in this encoding, all note-values are equally distant in the Euclidean sense. We also contrast our note-value encodings with piano roll representations, such as the representation used by Velarde et al. [2016]. In a piano roll representation, time is discretized: the value of a note is indicated implicitly by the number of discrete time-slices over which it is sustained. We choose an explicit representation of note-values because it more directly reflects the contents of a written score, and results in shorter time series overall than discretized representations.

4.3.3 A Composer Attribution Task

Our aim is to learn a classifier that predicts a composer y given a score $\mathbf{w}$ encoded according to Definition 4.8. There are $C \equiv 19$ composers in our corpus, and we assign each composer a label from 1 to C. We will construct a model $f_\theta : \mathcal{S} \to \mathbb{R}^C$ that assigns vector $f_\theta(\mathbf{w})$ to a score $\mathbf{w}$ where each component $f_\theta(\mathbf{w})_i$ indicates the model's (un-normalized) confidence that composer i wrote score $\mathbf{w}$. We predict $\hat{y}_\theta(\mathbf{w}) \equiv \arg\max_i f_\theta(\mathbf{w})_i$, the composer the model has most confidence in.

We evaluate our models via accuracy on holdout sets $\mathbf{w}^{\text{holdout}}$, where accuracy is the zero-one loss defined by

$$\text{Accuracy}(\mathbf{w}^{\text{holdout}}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}(\hat{y}_\theta(\mathbf{w}_i^{\text{holdout}}) = y_i). \tag{4.19}$$

Here $\mathbf{1} : \text{Bool} \to \{0, 1\}$ is the indicator function: $\mathbf{1}(p) = 1$ if the proposition p is true, otherwise $\mathbf{1}(p) = 0$. The results in Section 4.3.5 report 10-fold cross-validated accuracies. It is standard practice in the machine learning community to report results on a single holdout set. But for for the small datasets considered in composer attribution, cross-validation is essential to reduce the variance of estimated accuracy.

Given a collection of labeled scores (training data) $\{(\mathbf{w}_1, y_1), \ldots, (\mathbf{w}_n, y_n)\}$ and a parameterized family of models $\{f_\theta : \theta \in \Theta\}$ we learn an optimal model f_θ by empirical risk minimization of the negative log-likelihood under a softmax-normalized probability distribution of model outputs:

$$\min_{\theta\in\Theta} \sum_{i=1}^{n} -\log\left(\frac{\exp(f_\theta(\mathbf{w}_i)_{y_i})}{\sum_{k=1}^{C}\exp(f_\theta(\mathbf{w}_i)_{y_k})}\right). \tag{4.20}$$

For each of iteration of 10-fold cross-validation, in addition to the holdout fold $\mathbf{w}^{\text{holdout}}$, we hold out a second fold as validation data and optimize the objective (4.20) on the remaining 8 folds. We train our models using the Adam optimizer [Kingma and Ba, 2015], regularizing

with retrospective early stopping at the point with best accuracy on the validation fold.

4.3.4 Model Architectures

Every model class f_θ proposed in this work uses the following general template:

1. Compute a set of local features at or around each time index in the score.

2. Average these features across time ("pool" the features, in neural networks parlance) into a single global feature vector.

3. Construct a linear classifier on top of this global feature vector to predict the composer of the score.

This approach is motivated by the need to manage the high-dimensionality of a score: given even the first 5 indices of the score tensor $\mathbf{w}$ described in Section 4.3.2, we can easily fit a classifier that achieves 100% training accuracy but fails to generalize to new data. As discussed in Section 4.3.1, the classical approach to this overfitting phenomenon is to reduce a score to a low-dimensional summary of pre-determined features and fit a classifier to this summary. The present work aims to learn features from scratch, but if we permit our model to learn any features it wants then it will simply overfit to the training data.

We therefore regularize our models in two ways. First, Step 1 of the general template above limits our model to learn features that are local in scope. We will allow our models to learn features that are sensitive to correlations between co-occurring notes (harmony), or between short sequences of notes (melody, rhythm). But by construction, our models will not be able to learn features that capture correlations between (for example) the first and last notes of a score. This precludes us from learning certain high-level patterns that could have predictive power (e.g., Mozart is more likely to repeat a section verbatim than Bach) but saves of us from learning a multitude of spurious patterns that may appear to have predictive power on the training data but fail to generalize to new observations. Second,

	Sample Size					
Model	10	20	50	100	250	500
Histogram (Equation 4.21)	50.0	59.0	62.0	63.0	66.1	64.2
Voices (Equation 4.24)	60.0	61.6	63.9	72.0	75.5	76.9
Hybrid (Equation 4.28)	59.3	62.1	68.9	77.1	79.9	81.7

Table 4.7: Comparison of model accuracies at various samples sizes: accuracy increases uniformly with sample size. See the referenced equations for formal model definitions.

Step 2 of the general template prevents our model from overfitting to features that occur at fixed time locations. As discussed above, even knowing the first 5 indices of the score tensor is enough to easily identify every score in the training data. By pooling features together across time, we force our models to classify based on the overall prevalence of the features it learns, rather than the occurrence of a particular feature at a particular time.

Classical approaches to feature engineering largely follow the same modeling restrictions proposed in this template. The features used by e.g., Herremans et al. [2016] (Table 1, Page 7) or Brinkman et al. [2016] (Table 1, Page 2) consist primarily of overall frequencies, prevalences, and rates of occurrence. These features capture properties of either a single time index or short sequences, aggregated across an entire score. The use of n-gram features also fits this mold: an n-gram is by definition a local feature of n time indices, and an n-gram featurization computes aggregate (i.e., pooled) counts of the occurrences of each particular n-gram across a score. Non-local features are used by Kempfert and Wong [2020], e.g., the "maximum fraction of overlap with opening material within first half of movement." The use of these features may account for the effectiveness of Kempfert and Wong [2020] in the Haydn versus Mozart classification task, which our models underperform on.

Sub-sampling Scores

The approach outlined above requires us to average features across entire scores. Each score in the KernScores corpus (Section 4.3.2) has a unique length, ranging from 10 to 4000 time indices. As a practical matter, it is difficult to deal with such variable-length data

in machine-learning systems; our tools are designed to operate efficiently on homogeneous batches of data. One solution to this problem is to sub-sample scores; for example Velarde et al. [2016] train models on the first s quarter notes where $s = 70$ or $s = 400$. That work found that the larger sample consistently outperforms the smaller one. We confirm this finding with the experiments in Table 4.7, which show that our models consistently perform better with larger samples of the score.

We therefore make the following compromise between using all available information from a score and operating on homogeneous inputs: we sample the first s, middle s, and last s indices from our score $\mathbf{w}$, resulting in $3s$ time indices sampled from each score. We use $s = 500$ for all experiments except the experiments in Table 4.7 that explore how models behave as we vary this hyperparameter. The average score in the KernScores corpus has 534 time indices, so for most scores this means we sample the entire score (for scores shorter than 500 time indices we pad out our sample with zeros). Only for scores longer than $1,500$ time indices (there are 117 in our corpus) do we lose any information with this approach.

Histogram Models

The simplest models we consider are histogram models. Averaging the input data $\mathbf{w}$ over voices and time gives us a histogram vector $h \in \{0,1\}^{N+D+1}$:

$$h(\mathbf{w}) = \frac{1}{TP} \sum_{t=1}^{T} \sum_{p=1}^{P} \mathbf{w}_{t,p}.$$

Multiplying this histogram by a weight matrix $W_\theta \in \mathbb{R}^{(N+D+1)\times C}$ with parameterized entries gives us a simple linear model:

$$f_\theta(\mathbf{w}) = W_\theta^\top h(\mathbf{w}). \tag{4.21}$$

No features are learned in this model; all that is learned are the linear weights W_θ on the histogram features. The model can be interpreted as a simplified version of the global feature models discussed in Section 4.3.1. In this case, the global features are the prevalences at

which each of the $N + D + 1$ note and duration symbols occur in a score.

Voice Convolutional Models

We now consider a simple neural model inspired by n-gram features. Let k be a number of features we desire to learn and n be a number of time indices. Define the function $\text{relu} : \mathbb{R} \to \mathbb{R}$ by $t \mapsto t\mathbf{1}(t > 0)$. Given a weight matrix $W_\theta^1 \in \mathbb{R}^{n(N+D+1)\times k}$ we can construct a "convolutional" feature representation $h_{t,p} \in \mathbb{R}^{T\times P\times k}$ at each time index t in each voice p defined by

$$h_{t,p}(\mathbf{w};\theta) = \text{relu}\left((W_\theta^1)^\top \mathbf{w}_{t:t+n,p}\right). \tag{4.22}$$

We define $\mathbf{w}_{t:t+n}$ to be a slice of $\mathbf{w}$ from index t to index $t+n$ (non-inclusive); when $t+n > T$, we pad $\mathbf{w}$ with zeros. We then pool these features across voices and time to construct a single, global feature representation $h \in \mathbb{R}^k$, to which we can apply a linear classifier with weights $W_\theta \in \mathbb{R}^{k\times C}$:

$$\begin{aligned} h(\mathbf{w};\theta) &= \frac{1}{TP}\sum_{t=1}^{T}\sum_{p=1}^{P} h_{t,p}(\mathbf{w};\theta), \\ f_\theta(\mathbf{w}) &= (W_\theta)^\top h(\mathbf{w};\theta). \end{aligned} \tag{4.23}$$

This is a non-linear model (because of the non-linear relu "activation") and we can view h as a learned feature representation of the score $\mathbf{w}$. The weights ("filters") W_θ^1 learn to extract k relevant patterns of length n from voices, analogous to—but more expressive and compact than—classical n-gram featurizations. In our experiments we set $k = 500$ and $n = 3$; the choice of n is consistent with the pervasive use of 3-grams features in prior work [Hillewaere et al., 2010, Hontanilla et al., 2013, Wolkowicz and Keselj, 2013, Kempfert and Wong, 2020].

Deeper Representations

A natural way to extend the convolutional feature extraction discussed in Section 4.3.4 is to stack multiple layers of convolutions. Given the feature representation $h_{t,p}$ given by Equation

4.22 and a parameterized weight tensor $W_\theta^2 \in \mathbb{R}^{nk_1 \times k_2}$, we can construct a second layer of features

$$h_{t,p}^2(\mathbf{w};\theta) = \text{relu}\left((W_\theta^2)^\top h_{t:t+n,p}(\mathbf{w};\theta)\right).$$

We can loosely interpret such a representation as building hierarchical features of features. In principle we can build arbitrarily deep stacks of features in this way; in our experiments, we were unable to realize significant gains using architectures with more than two convolutional layers. Building a classifier over these features proceeds identically to the shallower models:

$$\begin{aligned} h_{\text{conv}}(\mathbf{w};\theta) &= \frac{1}{TP}\sum_{t=1}^{T}\sum_{p=1}^{P} h_{t,p}^2(\mathbf{w};\theta) \\ f_\theta(x) &= (W_\theta)^\top h_{\text{conv}}(\mathbf{w};\theta). \end{aligned} \tag{4.24}$$

For this model we set $n = 3$, $k = 300$, and $k_2 = 300$. Of course it is possible to stack more layers of convolutions following this pattern, but we were unable to achieve strong performance in our experiments using deeper representations.

Full-Score Convolutional Models

The models described in Equations (4.23) and (4.24) are largely monophonic: they extract features from individual voices (although they classify the score based on a pool of features gathered from all the voices). Notably, those models have no ability to capture harmonic patterns in the interactions between voices. We now consider a model that can capture these interactions. Let $W_\theta^1 \in \mathbb{R}^{nP(N+D+1)\times k}$, $W_\theta^2 \in \mathbb{R}^{nk\times k_2}$ and consider the model

$$\begin{aligned} h_t(\mathbf{w};\theta) &= \text{relu}\left((W_\theta^1)^\top \mathbf{w}_{t:t+n}\right) \in \mathbb{R}^{T\times k}, \\ h_t^2(\mathbf{w};\theta) &= \text{relu}\left((W_\theta^1)^\top h_{t:t+n}\right) \in \mathbb{R}^{T\times k2}, \\ h(\mathbf{w};\theta) &= \frac{1}{T}\sum_{t=1}^{T} h_t^2(\mathbf{w};\theta), \\ f_\theta(\mathbf{w}) &= (W_\theta)^\top h(\mathbf{w};\theta). \end{aligned} \tag{4.25}$$

We parameterize this model with $n = 3$, $k = 300$ and $k_2 = 300$. This model is strictly more expressive than the part-wise models (4.23) or (4.24), capable of capturing patterns that the part models can't. However, looking ahead to Section 4.3.5, the underperformance of this model (4.25) relative to less expressive models (4.23) and (4.24) suggests that it is prone to capture spurious patterns, leading to overfitting (see results in Table 4.8).

Harmonic Models

All the models considered so far treat pitch classes as categorical data. We recognize, for example, that C4 is distinct from E4 or G4, but not that C4 is 4 semi-tones below E4 and 7 semi-tones below G4. We now consider a model that exploits this structural order of pitch-classes, by convolving along the pitch-axis of the input tensor. For notational convenience, we decompose the input tensor $\mathbf{w} = \mathbf{f} \oplus \mathbf{d}$ into separate pitch components $\mathbf{f} \in \{0,1\}^{T\times P\times N}$ and note-value components $\mathbf{d} \in \{0,1\}^{T\times P\times (D+1)}$. Let $W^1_\theta \in \mathbb{R}^{jP\times k}$ and convolve along the pitch-axis to construct a features $h_{t,n}(\mathbf{f};\theta) \in \mathbb{R}^{T\times N\times k}$:

$$h_{t,u}(\mathbf{f};\theta) = \text{relu}\left((W^1_\theta)^\top \mathbf{f}_{t,:,u:u+j}\right).$$

Here j is a hyper-parameter indicating the height of the convolution; analogous to the width-n hyperparameter in our time-domain convolutions for models (4.23), (4.24), and (4.25). Unlike the time domain, we find that setting a large value of j (in our models, $j = N/2$) is desirable; a similar parameterization is used for frequency-domain convolutions in Chapter 3, Section 3.3.5.

We proceed to pool the features $h_{t,u}$ together across the pitch domain to construct $h_t \in \mathbb{R}^{T\times k}$:

$$h_t(\mathbf{f};\theta) = \frac{1}{N}\sum_{u=1}^{N} h_{t,u}(\mathbf{f},\theta). \tag{4.26}$$

The idea of this pooling is to construct a feature-set that is invariant to pitch translation. We are interested in learning features such as, for example, the occurrence of general major

chords rather than the occurrence of a particular major chord, such as the one rooted at A3. The pooling operation above precludes us from learning the latter type of feature. We then construct a second layer of features to integrate the harmonic features h_t together with the note-value features $\mathbf{d}_t$. Using weights $W_\theta^2 \in \mathbb{R}^{k \times k_2}$ and $W_\theta^3 \in \mathbb{R}^{(D+1) \times k2}$ we build $h_t^2(\mathbf{w}; \theta) \in \mathbb{R}^{T \times k_2}$. We then pool the representations h_t^2 across time and construct a linear classifier on the resulting representation:

$$\begin{aligned} h_t^2(\mathbf{w}; \theta) &= \text{relu}\left((W_\theta^2)^\top h_t(\mathbf{f}; \theta) + (W_\theta^3)^\top \mathbf{d}_t\right), \\ h_{\text{harmonic}}(\mathbf{w}; \theta) &= \frac{1}{T} \sum_{t=1}^{T} h_t^2(\mathbf{w}; \theta), \\ f_\theta(\mathbf{w}) &= (W_\theta)^\top h_{\text{harmonic}}(\mathbf{w}; \theta). \end{aligned} \tag{4.27}$$

We parameterize this model with $k = 64$ and $k_2 = 500$.

Hybrid Models

Looking back at the models we've introduced, observe that the voice models (4.23) and (4.24) exploit temporal structure within voices, but pool away any harmonic patterns between voices. In contrast, the harmonic model (4.27) exploits harmony between voices but pools away any sequential patterns across time indices. The full-score convolutional model can capture both types of structure, but is prone to capture spurious patterns and overfit. This motivates the introduction of our final, hybrid model that weakly combines temporal and harmonic models to increase predictive power without overfitting. The idea is to feed the input tensor separately through temporal and harmonic models to construct features representations h_{conv} (4.24) and h_{harmonic} (4.27) respectively. We combine these features in a final, linear layer using weights $W_\theta^c \in \mathbb{R}^{k_2 \times C}$ and $W_\theta^h \in \mathbb{R}^{k_2 \times C}$ to make a prediction:

$$f_\theta(\mathbf{w}) = (W_\theta^{\text{c}})^\top h_{\text{conv}}(\mathbf{w}; \theta) + (W_\theta^{\text{h}})^\top h_{\text{harmonic}}(\mathbf{w}; \theta). \tag{4.28}$$

	Models					
Composer	(4.21)	(4.23)	(4.24)	(4.25)	(4.27)	(4.28)
Japart	0.0	13.6	13.6	9.1	18.2	13.6
Hummel	41.7	54.2	66.7	62.5	87.5	91.7
Compere	0.0	25.9	22.2	25.9	40.7	37.0
Vivaldi	30.3	94.4	91.6	54.5	45.5	54.5
Du Fay	45.7	82.9	74.3	71.4	80.0	74.3
Orto	0.0	18.6	37.2	25.6	46.5	48.8
Joplin	85.1	91.5	93.6	93.6	95.7	91.5
D. Scarlatti	44.1	59.3	62.7	78.0	79.7	72.9
Busnois	13.2	48.5	48.5	51.5	60.3	60.3
Chopin	55.3	54.2	64.5	72.4	76.3	68.4
Ockeghem	13.3	55.1	69.4	52.0	66.3	72.4
Martini	44.3	68.0	75.4	59.8	68.0	73.8
Mozart	34.8	56.3	61.6	63.6	70.2	67.5
Beethoven	72.2	82.2	83.4	78.7	84.0	89.3
de la Rue	27.5	57.9	71.3	63.5	73.6	79.2
Corelli	89.4	89.9	86.2	93.1	93.6	95.2
Haydn	85.6	75.6	71.3	79.9	82.3	83.7
Josquin	81.1	78.7	76.4	75.9	77.3	82.3
Bach	92.3	95.7	96.1	97.2	97.2	97.6
Overall	64.2	75.4	76.9	75.5	79.8	81.7

Table 4.8: Results of the 19-way classification problem on the full corpus for each model considered in this work. Reported results are percent accuracy, as defined by Equation (4.19), calculated using the 10-fold cross-validation procedure described in Section 4.3.3.

Because temporal and harmonic information are only combined in the final linear layer, this model is unable to learn expressive relationships between these features, such as the classical XOR relationship Minsky and Papert [1987]. As we see in Table 4.8, this combination increases accuracy over either the temporal or harmonic models on their own.

4.3.5 Results and Conclusions

The results for all models discussed in this paper, evaluated on the full corpus using the cross-validation procedure described in Section 4.3.3, are presented in Table 4.8. We sort the

	Bach	Orto	Fay	Ock.	Josq.	Rue
Bach	370	0	0	0	0	0
Orto	0	17	0	3	22	1
Du Fay	0	0	29	4	2	0
Ockegham	0	2	5	80	9	2
Josquin	3	6	5	14	357	38
de la Rue	2	0	0	1	46	129

Model	Bach	Orto	Fay	Ock.	Josq.	Rue
Hybrid (Equation (4.28))	**100.0**	**39.5**	**82.9**	**81.6**	**84.4**	72.5
KNN [Brinkman et al., 2016]	94.5	38.9	42.9	70.0	60.6	80.6
SVM [Brinkman et al., 2016]	98.5	33.3	25.0	60.0	60.0	**87.1**

Table 4.9: (Top) Confusion matrix for the hybrid model defined by Equation (4.28), trained and evaluated on a 6-composer subset of the corpus; rows indicate the true composer and columns indicate the model's prediction. Compare to the results in Tables 3 and 4 (page 6) of Brinkman et al. [2016]. (Bottom) Accuracy (Equation (4.19)) of our hybrid model comparisoned to the KNN and SVM models from Brinkman et al. [2016].

rows in this table by the number of scores for each composer; we observe a trend towards increasing accuracy when we have more data (with some outliers).

To compare with previous work, we train additional models on subsets of the corpus. We invite comparisons between the results in Table 4.9 and the results of Brinkman et al. [2016], and between the results in Table 4.10 and the results of Herremans et al. [2016]. These comparisons are imperfect: neither Brinkman et al. [2016] nor Herremans et al. [2016] report the precise scores used in their experiments. Nevertheless our corpus is derived from the same KernScores sources as Brinkman et al. [2016] and Herremans et al. [2016], and contains a comparable number of scores to the counts reported by Herremans et al. [2016]. Therefore we believe our subsets are similar to the corpora used in these works and that comparison is meaningful. For future reference, the exact dataset used for the present work can be found online.[6]

[6] http://homes.cs.washington.edu/~thickstn/ismir2019classification/

	Bach	Haydn	Beethoven
Bach	369	1	0
Haydn	7	195	7
Beethoven	5	18	146

Model	Bach	Haydn	Beethoven
Hybrid (Equation (4.28))	**99.8**	**93.3**	**86.4**
SVM [Herremans et al., 2016]	94.6	80.3	64.8

Table 4.10: (Top) Confusion matrix for the hybrid model (4.28), trained and evaluated on a 3-composer subset of the corpus; rows indicate the true composer and columns indicate the model's prediction. Compare to the results in Table 9 (page 18) of Herremans et al. [2016]. (Bottom) Accuracy (Equation (4.19)) comparisons of our hybrid model to the SVM model from Herremans et al. [2016].

For the popular Haydn versus Mozart string quartet classification task [Hillewaere et al., 2010, Herlands et al., 2014, Van Kranenburg and Backer, 2005, Kempfert and Wong, 2020], we were unsuccessful. The standard evaluation metric for this task is LOOCV, which we could not perform due to the computational expense of our models. With 10-fold cross validation, we observed exceedingly high variance upon repeat optimizations of the same model. However none of our optimizations exceeded 80%. Due to imbalance between Haydn and Mozart quartets (209 versus 82 scores) a classifier that simply predicts Haydn given any input achieves 71.8% so, despite the variance, it is unlikely that our classifiers are particularly effective for this task. Overall, we conclude that the convolutional models proposed in this paper perform quite well. We find this notable, given that success in neural modeling is often associated with much larger datasets. Furthermore, we do not believe that the potential of these methods has been exhausted; further investigation may yield even better end-to-end neural architectures for composer attribution.

4.4 Conclusion

Having gone to the effort to construct generative models (Section 4.2) and composer attribution models (Section 4.3) we should emphasize that we are not interested in these models

as ends unto themselves. Our interest in a generative model of music is motivated by their power as a prior over musical form. For example, difficult or ambiguous transcription questions could be more readily resolved given prior over the form of a transcribed performance: any ambiguities can be resolved to make the transcribed output most likely under the prior. Concrete applications of generative models is priors for Bayesian posterior sampling are discussed next in Chapter 5. Likewise, we are not particularly interested in the composer attribution task in its own right, but the task is appealing as an unambiguous benchmark classification task for musical scores.

Arguably, the most challenging aspect of working with symbolic musical data is not the modeling, but rather the diversity of representations and encodings discussed in Section 4.1. Digital symbolic music is unlike images or audio, which have standard discretized encodings as pixel or sample arrays, nor is symbolic music like language, which, through the hard work of the Unicode consortium, has a relatively standardized universal character set. This lack of standardization makes it difficult to choose an encoding to model, and difficult to make comparisons between models based on different encodings. We hope that the tensor definition of symbolic music proposed in Definition 4.1, and the factorization-agnostic cross entropy metric proposed in Definition 4.4 can help to bring clarity to these modeling questions. But we also acknowledge that encodings are likely to remain a difficulty for symbolic music modeling work in the foreseeable future.

Chapter 5

CONDITIONAL SAMPLING FROM GENERATIVE MODELS

Generative models can serve as a powerful primitive for creative interaction with data. Conditional generative models $p(\mathbf{x}|\mathbf{y})$ allow us to synthesize or re-synthesize multimedia, using conditioning variables $\mathbf{y}$ as adjustable knobs to steer the model's outputs in service to a creative vision. While training a conditional generative model is a straightforward generalization of unconditional generative modeling, there are several reasons we might want to avoid training task-specific conditional generative models. First, generative models are expensive to train: building a large suite of conditional generative models for a variety of related tasks could be an inefficient use of computational resources. Second, we may want to take advantage of large quantities of unlabeled data to train an unconditional generative model, and adapt this model to a conditional task using a much smaller quantity of labeled data. Third, training a conditional generative model requires labeled data during training: given a general-purpose generative model trained on unlabeled data, we may want to later adapt this model for conditional sampling using a labeled dataset. Finally, while injecting conditional variables into a generative model is mathematically straightforward, this change requires modifications to the neural architecture used to parameterize the model. The precise nature of these changes may be non-obvious, specific to a particular conditioning variable, and the hyper-parameter settings of the modified architecture may be different than the original unconditional architecture, requiring further hyper-parameter optimization for each conditional model to achieve strong results.

In this chapter we develop a procedure for conditional sampling from posterior distributions of the form $p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$, given an unconditional generative model $p(\mathbf{x})$ and a classifier $p(\mathbf{y}|\mathbf{x})$. The idea is to construct a Markov chain, based on Langevin dynamics, with

a steady state distribution that approximates the target distribution $p(\mathbf{x}|\mathbf{y})$, while avoiding a direct (and intractable) explicit calculation of the posterior density $p(\mathbf{x}|\mathbf{y})$. This approach to sampling decouples the generative modeling problem from the details of specific conditional generation tasks, giving us a general way to construct conditional samples $\mathbf{x}$, controlled by labels $\mathbf{y}$. We treat $p(\mathbf{x})$ as a black box, which allows us to take advantage of pre-trained generative models trained using vast quantities of un-labeled data and industrial-scale computing resources.

In Section 5.1, we review Langevin dynamics [Neal et al., 2011] and noise-annealed Langevin dynamics [Song and Ermon, 2019] along with their application to general likelihood-based models and extension to the posterior sampling setting [Jayaram and Thickstun, 2020, Song et al., 2021]. Langevin dynamics follow the gradients of a continuous log-likelihood function, and so a Langevin-based sampler cannot be directly applied to discrete distributions. We propose an approach to smoothing discrete distributions, which allows Langevin samplers to be applied to discretized generative models of continuous data. We focus in particular on smoothing of discretized autoregressive models. Defined over a discrete lattice within a continuous space, these models occupy a middle ground between continuous and discrete models. Our interest in discretized autoregressive models is motivated by their success as unconditional models of audio waves [van den Oord et al., 2016a, Mehri et al., 2017, Dhariwal et al., 2020].

In Section 5.2 we discuss linear inverse problems of recovering $\mathbf{x}$ given $\mathbf{y} = g(\mathbf{x})$, for some linear function $g : \mathcal{X}^n \to \mathcal{Y}^n$. We propose to use deep generative models as priors for linear inverse problems, using Langevin dynamics to sample from the posterior distribution over possible inverses $\mathbf{x}$. We focus in particular on source separation, which can be framed as a linear inverse problem. For some inverse problems, the posterior distribution $p(\mathbf{x}|\mathbf{y})$ is quite peaked, and sampling recovers the only plausible inverse $\mathbf{y}$. But in many cases, the inverse is underdetermined: a ground truth label $\mathbf{x}$ for input $\mathbf{y}$ represents just one of a variety of plausible inverses. This motivates our interest in posterior sampling, by which we can explore the space of plausible inverses under the prior. At the same time, this ambiguity

poses a challenge traditional evaluation metrics, which often presume that a ground-truth $\mathbf{x}$ is identifiable given $\mathbf{y}$. We therefore discuss evaluation methodology, and propose the use of generative modeling metrics to evaluate source separation results.

In Section 5.3 we undertake an investigation of the empirical behavior of conditional Langevin sampling in the visual and audio domains. We consider unconditional Langevin sampling, as well as linear inverse problems. We focus in particular on source separation, which we frame as a linear inverse problem, with additional results for other linear inverse problems: in-painting, super-resolution, and image colorization. Our empirical study focuses on three questions. First, how *fast* and *accurate* is the Langevin sampling procedure? Second, how does the *quality* of the prior model affect the quality of results? And third, how does the quality of Langevin posterior sampling compare to task-specific solutions to various linear inverse problems?

5.1 Conditional Sampling via Langevin Dynamics

We can sample from an unconditional, continuous probability distribution $p(\mathbf{x})$ over objects $\mathbf{x} \in \mathbb{R}^n$ via Langevin dynamics. Let $\mathbf{x}_0 \sim \text{Uniform}(\mathbb{R}^n)$, $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, I_n)$, and define a Markov chain

$$\mathbf{x}^{(t+1)} \equiv \mathbf{x}^{(t)} + \eta \nabla_{\mathbf{x}} \log p(\mathbf{x}^{(t)}) + \sqrt{2\eta}\boldsymbol{\varepsilon}_t. \tag{5.1}$$

As we discuss in Chapter 2 (Section 2.7), under regularity conditions on $p(\mathbf{x})$ and for a sufficiently small step size η, the iterates $\mathbf{x}^{(t)}$ converge in distribution to $p(\mathbf{x})$ as $t \to \infty$. When the distribution $p(\mathbf{x})$ is parameterized by a neural network, then gradients $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ can be computed by automatic differentiation with respect to the inputs of the generator network. This family of likelihood-based models includes autoregressive models [Salimans et al., 2017, Parmar et al., 2018], the variational autoencoder [Kingma and Welling, 2014, van den Oord et al., 2017], and flow-based models [Dinh et al., 2017, Kingma and Dhariwal, 2018]. Alternatively, if gradients of the distribution are modeled [Song and Ermon, 2019],

then an estimate of $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ can be used directly.

For conditional generation tasks, we are interested in sampling from the posterior of a joint distribution $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$, where $p(\mathbf{x})$ is an autoregressive model over $\mathbf{x} \in \mathbb{R}^n$ and $p(\mathbf{y}|\mathbf{x})$ is a conditional likelihood. The posterior distribution is given by the density

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{\int_{\mathcal{X}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x})\, d\mathbf{x}}. \tag{5.2}$$

In general the denominator $p(\mathbf{y}) = \int_{\mathcal{X}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x})\, d\mathbf{x}$ has no simple closed form, and calculating this density is typically intractable. However $p(\mathbf{y})$ is constant as a function of $\mathbf{x}$, so $\nabla_{\mathbf{x}} \log p(\mathbf{y}) = 0$, and therefore the Langevin dynamics for sampling from the posterior distribution are

$$\begin{aligned} \mathbf{x}^{(t+1)} &\equiv \mathbf{x}^{(t)} + \eta \nabla_{\mathbf{x}} \log p(\mathbf{x}^{(t)}|\mathbf{y}) + \sqrt{2\eta}\boldsymbol{\varepsilon}_t \\ &= \mathbf{x}^{(t)} + \eta \nabla_{\mathbf{x}} \left(\log p(\mathbf{x}^{(t)}) + \log p(\mathbf{y}|\mathbf{x}^{(t)}) \right) + \sqrt{2\eta}\boldsymbol{\varepsilon}_t. \end{aligned} \tag{5.3}$$

This is a convenient Markov chain for posterior sampling, because it avoids explicit calculation of the denominator of the posterior density. We are particularly interested in measurement models of the form $\mathbf{y} = g(\mathbf{x})$, for some linear function $g : \mathbb{R}^n \to \mathbb{R}^m$. This family of linear measurement models describes the conditional generation tasks featured in Section 5.3, as well as other linear inverse problems including sparse recovery.

For rich, multi-modal distributions $p(\mathbf{x}, \mathbf{y})$, Langevin dynamics mixes slowly. In Section 5.1.2 we develop a procedure for smoothing $p(\mathbf{x}, \mathbf{y})$ that accelerates this mixing process. Given a temperature parameter σ, we introduce distributions $p_\sigma(\mathbf{x}, \mathbf{y})$ that smooth out the structure of $p(\mathbf{x}, \mathbf{y})$. At high temperatures, $p_\sigma(\mathbf{x}, \mathbf{y})$ is unimodal and irreducible, so Langevin dynamics will mix quickly. And $p_\sigma(\mathbf{x}, \mathbf{y}) \to p(\mathbf{x}, \mathbf{y})$ in total variation as we cool the temperature σ to zero. This motivates introduction of a modified Langevin dynamics, replacing the

Algorithm 3: Noise-Annealed Langevin Sampling

Input: $\mathbf{y}$, $\{\sigma_i\}_{i=1}^{L}$, δ, T
Sample $\mathbf{x}^{(0)} \sim \mathcal{N}(0, \sigma_1^2 I_n)$
for $i \leftarrow 1$ **to** L **do**
  $\eta_i \leftarrow \delta \cdot \sigma_i^2 / \sigma_L^2$
  for $t = 1$ **to** T **do**
    Sample $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, I_n)$
    $\mathbf{g}^{(t)} \leftarrow \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x}^{(t)}) + \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{y}|\mathbf{x}^{(t)})$
    $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} + \eta_i \mathbf{g}^{(t)} + \sqrt{2\eta}\boldsymbol{\varepsilon}_t$
  end for
end for

posterior likelihood $p(\mathbf{x}|\mathbf{y})$ with its smoothed counterpart $p_\sigma(\mathbf{x}|\mathbf{y})$:

$$\mathbf{x}^{(t+1)} \equiv \mathbf{x}^{(t)} + \eta \nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x}^{(t)}|\mathbf{y}) + \sqrt{2\eta}\varepsilon_t. \tag{5.4}$$

These dynamics approximate samples from $p(\mathbf{x}|\mathbf{y})$ as $\eta \to 0$, $\sigma^2 \to 0$, and $t \to \infty$.

We appeal to simulated annealing [Kirkpatrick et al., 1983] using the heuristic proposed by Song and Ermon [2019] to turn down the temperature as the Markov chain (5.4) mixes. In contrast to classical Markov chain sampling, for which samples $\mathbf{x}^{(t)}$ converge in distribution to p, annealed Langevin dynamics converges asymptotically to a single point distributed approximately according to p. Algorithm 3 describes these annealed Langevin dynamics given a smoothed prior $p_\sigma(\mathbf{x})$ and smoothed conditional likelihood $p_\sigma(\mathbf{y}|\mathbf{x})$. Details of the annealing schedule $\sigma_1, \ldots, \sigma_L$ and step size η are discussed in Section 5.1.5.

In Section 5.1.3 we develop a Langevin sampler for neural autoregressive models [Larochelle and Murray, 2011]. This is a popular family of generative models, with wide-ranging applications in a variety of domains including audio [van den Oord et al., 2016a, Dhariwal et al., 2020], images [van den Oord et al., 2016b, Salimans et al., 2017, Parmar et al., 2018, Razavi et al., 2019], and text [Radford et al., 2019, Brown et al., 2020]. Continuous autoregressive models [Uria et al., 2013] are likelihood-based models and, as such, they are amenable to the

smoothing techniques developed in Section 5.1.2. But many popular autoregressive models of continuous data are defined over a discretized support set [van den Oord et al., 2016b, Salimans et al., 2017, Parmar et al., 2018, Razavi et al., 2019, van den Oord et al., 2016a], for which the smoothing procedure described in Section 5.1.2 cannot be directly applied [Frank and Ilse, 2020].

Neural autoregressive models parameterize the conditional distribution over a token in an ordered sequence, given previous tokens in the sequence. The standard approach to sampling from an autoregressive model iteratively generates tokens, according to a conditional distribution over tokens defined by the model, conditioned on the partial sequence of previously generated tokens. We will refer to this approach to sampling as the *ancestral sampler*. Ancestral sampling has time complexity that scales linearly in the length of the generated sequence. For data such as high-resolution images or audio, ancestral sampling from an autoregressive model (where the tokens are pixels or sound pressure readings respectively) can be impractically slow. In Section 5.1.4, we present a stochastic variant of the Langevin sampler for autoregressive models based on stochastic gradient Langevin dynamics [Welling and Teh, 2011]. This is an embarrassingly parallel, asynchronous distributed algorithm for autoregressive sampling. In Section 5.3 we will see that stochastic Langevin sampling can approximate the quality of ancestral sampling to arbitrary accuracy, with compute time that is inverse proportional to the number of computing devices. This allows Langevin sampling to take full advantage of modern, massively parallel computing infrastructure.

5.1.1 Related Work on Sampling

Noise-annealed Langevin posterior sampling is based on the annealed Langevin dynamics introduced by Song and Ermon [2019], which accelerates standard Langevin dynamics [Neal et al., 2011, Du and Mordatch, 2019] using a smoothing procedure in the spirit of simulated annealing [Kirkpatrick et al., 1983] and graduated optimization [Blake and Zisserman, 1987]. The extension of annealed Langevin dynamics to posterior sampling, which we develop in this work for linear inverse problems, is discussed for the fully general case of posterior sampling

by Song et al. [2021]. Markov-chain Monte Carlo posterior samplers based on Gibbs sampling rather than Langevin dynamics are proposed by Theis and Bethge [2015] and Hadjeres et al. [2017] as solutions for inpainting tasks.

The slow speed of ancestral sampling is a persistent obstacle to the adoption and deployment of autoregressive models, leading to algorithmic innovations that seek to parallelize the sampling process. Parallel WaveNet [van den Oord et al., 2018] and ClariNet [Ping et al., 2019] train generative flow models to mimic the behavior of an autoregressive model. Sampling from a flow model requires only one pass through a feed-forward network that can be distributed across multiple devices; these models can themselves be adapted for conditional sampling tasks using the Langevin sampler, as we demonstrate with the Glow model [Kingma and Dhariwal, 2018] in Section 5.3. Wiggers and Hoogeboom [2020] and Song et al. [2020] propose fixed-point algorithms for sampling from autoregressive models that, like Langevin sampling, iteratively refine an initial sample from a simple distribution into a sample from the target distribution. These iterative algorithms are easily adapted to completion problems given partial observations (e.g., inpainting and super-resolution) but are not easily adaptable to linear inverse problems or more general conditional sampling tasks.

Similar to anytime sampling [Xu et al., 2021], Langevin sampling offers a tradeoff between sample quality and computational budget. The Langevin iterates (5.1) mix to the target distribution as the iteration count $t \to \infty$; by stopping early, we can approximate samples from this distribution using less computation. We explore the empirical tradeoff between sample quality and computation for Langevin sampling from autoregressive models in Section 5.3.3. The anytime sampler proposed by Xu et al. [2021] requires a specific model architecture based on the VQ-VAE [van den Oord et al., 2017, Razavi et al., 2019]. In contrast, we can use Langevin sampling with any likelihood-based model. But unlike an anytime sampler, the computational budget for annealed Langevin sampling must be specified in advance: halting prior to completing the annealing schedule will result in noisy samples.

5.1.2 Smoothing a Joint Distribution

To accelerate mixing of the Markov chain given by Equation (5.3), we adopt a simulated annealing schedule over smoothed approximations to the model $p(\mathbf{x})$, extending the unconditional sampling algorithm proposed by Song and Ermon [2019]. The reason for smoothing is that rich distributions $p(\mathbf{x}|\mathbf{y})$ are multi-modal, and it is difficult for Langevin dynamics to hop between well-separated modes and properly explore the sample space. By smoothing out the distribution, we create an easier distribution to sample; in the extreme this distribution becomes approximately Gaussian and theoretical guarantees ensure efficient mixing. As we gradually reduce the smoothing temperature, we incrementally commit to sampling a particular cluster of modes in the underlying distribution; as temperature approaches zero, annealed sampling converges to a neighborhood of a single mode of $p(\mathbf{x}|\mathbf{y})$. Inspiration and analogies to this procedure can be found in previous work on simulated annealing [Kirkpatrick et al., 1983], graduated optimization [Blake and Zisserman, 1987], and coarse-to-fine methods [Raphael, 2001, Felzenszwalb and Huttenlocher, 2006, Kiddon and Domingos, 2011].

We smooth a joint density $p(\mathbf{x}, \mathbf{y})$ by convolving $\mathbf{x}$ with a spherical Gaussian $\mathcal{N}(0, \sigma^2 I_n)$. Let $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\varepsilon}_\sigma$ where $\boldsymbol{\varepsilon}_\sigma \sim \mathcal{N}(0, \sigma^2 I_n)$. Note that $\tilde{\mathbf{x}}$ is conditionally independent of $\mathbf{y}$ given $\mathbf{x}$ and therefore the joint distribution over $\mathbf{x}$, $\tilde{\mathbf{x}}$, and $\mathbf{y}$ can be factored as

$$p_\sigma(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{x}}) = p_\sigma(\tilde{\mathbf{x}}|\mathbf{x})p(\mathbf{y}|\mathbf{x})p(\mathbf{x}). \tag{5.5}$$

We will work with the smoothed marginal $p_\sigma(\tilde{\mathbf{x}}, \mathbf{y})$ of the joint distribution $p_\sigma(\mathbf{x}, \mathbf{y}, \tilde{\mathbf{x}})$. If $\varphi_\sigma(\mathbf{x})$ denotes the density of a spherical Gaussian $\mathcal{N}(0, \sigma^2 I_n)$ on $\mathbb{R}^n$ then the marginal $p_\sigma(\tilde{\mathbf{x}}, \mathbf{y})$ of Equation (5.5) can be expressed by

$$p_\sigma(\tilde{\mathbf{x}}, \mathbf{y}) = \int \varphi_\sigma(\tilde{\mathbf{x}} - \mathbf{x})p(\mathbf{x}, \mathbf{y})\, d\mathbf{x}. \tag{5.6}$$

This density approximates the original distribution $p(\mathbf{x}, \mathbf{y})$ in the sense that $p_\sigma(\tilde{\mathbf{x}}, \mathbf{y}) \to p(\mathbf{x}, \mathbf{y})$ in total variation as $\sigma^2 \to 0$. The smoothed density $p_\sigma(\tilde{\mathbf{x}}, \mathbf{y})$ can be factored as

$p_\sigma(\mathbf{y}|\tilde{\mathbf{x}})p_\sigma(\tilde{\mathbf{x}})$. The density $p_\sigma(\tilde{\mathbf{x}})$ is simply the smoothed Gaussian convolution of $p(\mathbf{x})$:

$$p_\sigma(\tilde{\mathbf{x}}) = \int p_\sigma(\tilde{\mathbf{x}}, \mathbf{y})\, d\mathbf{y} = \iint \varphi_\sigma(\tilde{\mathbf{x}} - \mathbf{x}) p(\mathbf{x}, \mathbf{y})\, d\mathbf{x}\, d\mathbf{y} \tag{5.7}$$

$$= \int \varphi_\sigma(\tilde{\mathbf{x}} - \mathbf{x}) p(\mathbf{x}) \left(\int p(\mathbf{y}|\mathbf{x})\, d\mathbf{y} \right) d\mathbf{x} = \int \varphi_\sigma(\tilde{\mathbf{x}} - \mathbf{x}) p(\mathbf{x})\, d\mathbf{x}. \tag{5.8}$$

And because $\mathbf{y}$ is conditionally independent of $\tilde{\mathbf{x}}$ given $\mathbf{x}$, marginalizing over $\mathbf{x}$ we see that

$$p_\sigma(\mathbf{y}|\tilde{\mathbf{x}}) = \int p_\sigma(\mathbf{y}|\tilde{\mathbf{x}}, \mathbf{x}) p_\sigma(\mathbf{x}|\tilde{\mathbf{x}})\, d\mathbf{x} = \int p(\mathbf{y}|\mathbf{x}) \varphi_\sigma(\mathbf{x} - \tilde{\mathbf{x}})\, d\mathbf{x}. \tag{5.9}$$

The integrals in Equations (5.8) and (5.9) are difficult to calculate directly in general. For noise-conditioned score networks, which directly parameterize the gradients of the prior log-likelihood given a noise level σ, we can directly compute $\nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}})$ by evaluating the score network at the desired noise level. For the linear inverse problems discussed in Section 5.2, the smoothed likelihood $p_\sigma(\mathbf{y}|\tilde{\mathbf{x}})$ has a convenient closed form. But for generic likelihood-based models $p(\mathbf{x})$ and $p(\mathbf{y}|\mathbf{x})$, these smoothed distributions are not directly accessible.

We can adapt models $p(\mathbf{x})$ and $p(\mathbf{y}|\mathbf{x})$ to approximate the quantities in Equations (5.8) and (5.9) using a fine-tuning procedure inspired by transfer learning [Yosinski et al., 2014]. A direct approach to estimating the distributions $p_\sigma(\tilde{\mathbf{x}})$ and $p_\sigma(\mathbf{y}|\tilde{\mathbf{x}})$ is to train models from scratch on noisy data $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\varepsilon}_\sigma$ where $\boldsymbol{\varepsilon}_\sigma \sim \mathcal{N}(0, \sigma^2 I_n)$. But this not always practical; generative models are expensive to train. Instead of training models from scratch, we can finetune pre-trained models $p(\mathbf{x})$ and $p(\mathbf{y}|\mathbf{x})$ on noise-perturbed data. Empirically, this procedure quickly converges rapidly to an estimate of $p_\sigma(\mathbf{x})$.

Fine-tuning requires us to store a copy of the model for each of L noise levels σ. This can be avoided by training a single noise-conditioned generative model as described in Section 5 of Song et al. [2021]. The advantage of using copies of the model is that we can directly use standard generative models, without any adjustment to the network architecture or subsequent hyper-parameter tuning of a modified architecture. This approach cleanly decouples the conditional sampling problem from neural architecture design questions. Note that while

we store L copies of the model, there is no algorithmic memory overhead: these models are loaded and unloaded serially during optimization as we anneal the noise levels, so only one model is resident in memory at a time. While GPU memory is a scarce resource, disk space is generally abundant.

5.1.3 Discretized Autoregressive Smoothing

We now consider Langevin sampling for autoregressive generative models over indexed sequences of values $\mathbf{x} \in \mathcal{X}^n$ where

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(\mathbf{x}_i | \mathbf{x}_{<i}). \tag{5.10}$$

In particular, we are interested in developing a sampler for *discretized* autoregressive models, where $\mathcal{X} = \mathbb{R}$ and each conditional $p(\mathbf{x}_i|\mathbf{x}_{<i})$ has support on a finite set of scalar values $\mathcal{D} = \{e_1, \ldots, e_d\} \subset \mathbb{R}$. The set $\mathcal{D}^n$ could represent, for example, an 8-bit encoding of an image or audio wave. For discrete models, the gradients $\nabla_{\mathbf{x}} \log p(\mathbf{x}^{(t)})$ required for Langevin dynamics are undefined. In this Section, we propose a smoothing procedure for discretized autoregressive models, creating a differentiable density on which the Markov chain described by (5.1) can mix. This procedure is visualized in Figure 5.1.

We will work with models that parameterize the conditional distribution $p(\mathbf{x}_i|\mathbf{x}_{<i})$ with a categorical softmax distribution over $d = |\mathcal{D}|$ discrete values. Given functions $f_i : \mathcal{X}^i \to \mathbb{R}^d$, we define

$$p(\mathbf{x}_i = e_k | \mathbf{x}_{<i}) = \frac{\exp(f_{i,k}(\mathbf{x}_{<i}))}{\sum_{\ell=1}^{d} \exp(f_{i,\ell}(\mathbf{x}_{<i}))}. \tag{5.11}$$

The functions f_i are typically given by a neural network, with shared weights across the sequential indices i. Collectively, these conditional models define the joint distribution $p(\mathbf{x})$ according to Equation (5.10).

Following the smoothing procedure in Section 5.1.2, we smooth $p(\mathbf{x})$ by convolving it with a spherical Gaussian $\mathcal{N}(0, \sigma^2 I_n)$. This smoothing relies on the fact that $e_k \in \mathcal{D}$ represent

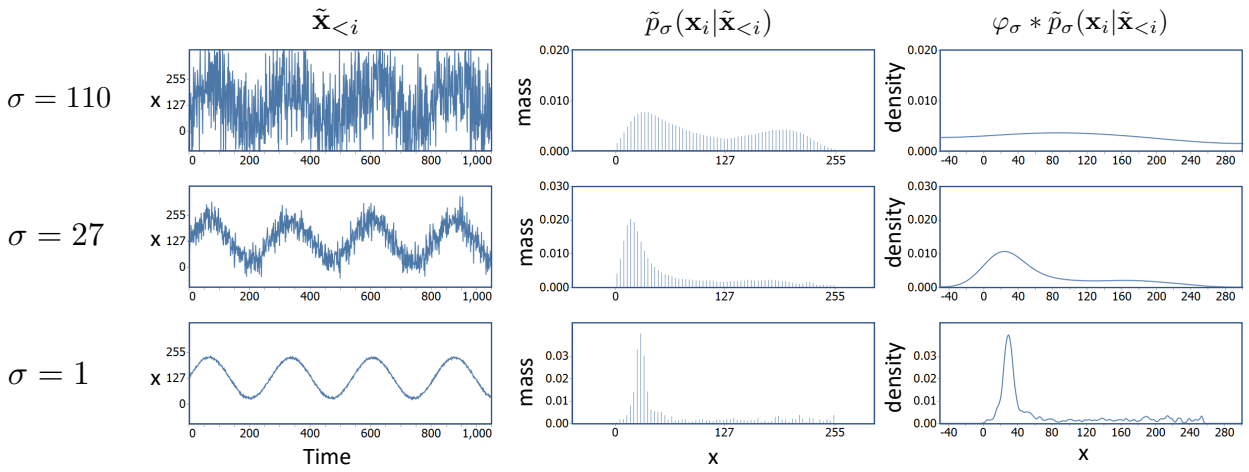


Figure 5.1: A visual summary of discretized autoregressive smoothing. Given a noisy history $\tilde{\mathbf{x}}_{<i} = \mathbf{x}_{<i} + \boldsymbol{\varepsilon}_{<i}$ (left column) where $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2 I)$, we train a model to predict the un-noised distribution over $\mathbf{x}_i \in \mathbb{R}$ (middle column). This distribution is discrete and non-differentiable in $\tilde{\mathbf{x}}$; we convolve with a Gaussian $\varphi_\sigma(t) = \mathcal{N}(t; 0, \sigma^2)$ to produce a continuous estimate of $\tilde{\mathbf{x}}_i$ (right column). We can run Langevin dynamics on the continuous distribution, and gradually anneal the smoothing to approximate the target distribution.

scalar values on the real line, and therefore the discrete distribution $p(\mathbf{x}_i|\mathbf{x}_{<i})$ can be viewed as a linear combination of weighted Dirac spikes on $\mathcal{D} \subset \mathbb{R}$. In Section 5.1.2 we approximated the smoothed density $p_\sigma(\tilde{\mathbf{x}})$ by fine-tuning a model $p(\mathbf{x})$ on noisy data $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\varepsilon}_\sigma$ where $\mathbf{x} \sim p$ and $\boldsymbol{\varepsilon}_\sigma \sim \mathcal{N}(0, \sigma^2 I_n)$. This approach cannot be directly applied to discrete autoregressive models as defined by Equation (5.11). The obstruction is that noisy samples $\tilde{\mathbf{x}}_i \in \mathbb{R}$ are not supported by the discretization $\mathcal{D}$. One way to address the problem is to replace the discrete model $p(\mathbf{x})$ with a continuous autoregressive model of $p_\sigma(\tilde{\mathbf{x}})$, e.g., RNADE [Uria et al., 2013]. We avoid this approach because fine-tuning $p(\mathbf{x})$ to $p_\sigma(\tilde{\mathbf{x}})$ becomes complicated when these models have different architectures.

Instead of directly fine-tuning $p(\mathbf{x})$ to a model $p_\sigma(\tilde{\mathbf{x}})$, we combine an analytic calculation with an auxiliary model learned via fine-tuning. Let $\tilde{p}_\sigma(\mathbf{x}_i|\tilde{\mathbf{x}}_{<i})$ denote a (discrete) conditional model trained to predict $\mathbf{x}_i$ given noisy covariates $\tilde{\mathbf{x}}_{<i} = \mathbf{x}_{<i} + \boldsymbol{\varepsilon}_{\sigma,<i}$. If φ_σ denotes the density

of $\mathcal{N}(0, \sigma^2)$ then we can re-write the factored density $p_\sigma(\tilde{\mathbf{x}})$ as

$$p_\sigma(\tilde{\mathbf{x}}) = \prod_{i=1}^{n} p_\sigma(\tilde{\mathbf{x}}_i|\tilde{\mathbf{x}}_{<i}) = \prod_{i=1}^{n} \big(\varphi_\sigma * \tilde{p}_\sigma(\,\cdot\,|\tilde{\mathbf{x}}_{<i})\big)(\tilde{\mathbf{x}}_i). \tag{5.12}$$

On the right-hand side, we decompose the smoothed conditional densities $p_\sigma(\tilde{\mathbf{x}}_i|\tilde{\mathbf{x}}_{<i})$ into Gaussian convolutions of discrete conditionals $\tilde{p}_\sigma(\,\cdot\,|\tilde{\mathbf{x}}_{<i})$ evaluated at $\tilde{\mathbf{x}}_i$. This suggests the following approach to evaluating $p_\sigma(\tilde{\mathbf{x}}_i|\tilde{\mathbf{x}}_{<i})$:

- Learn a (discrete) model $\tilde{p}_\sigma(\mathbf{x}_i|\tilde{\mathbf{x}}_{<i})$, trained to predict the un-noised value $\mathbf{x}_i$ given noisy history $\tilde{\mathbf{x}}_{<i}$. This model can be learned efficiently by finetuning a pre-trained model $p(\mathbf{x}_i|\mathbf{x}_{<i})$ on noisy covariates $\tilde{\mathbf{x}}_{<i}$. See Figure 5.1 (middle column).

- Evaluate the Gaussian convolution $\varphi_\sigma * \tilde{p}_\sigma(\,\cdot\,|\tilde{\mathbf{x}}_{<i})$ at $\tilde{\mathbf{x}}_i$ to compute $p_\sigma(\tilde{\mathbf{x}}_i|\tilde{\mathbf{x}}_{<i})$. This convolution can be calculated in closed form given $\tilde{p}_\sigma(\mathbf{x}_i|\tilde{\mathbf{x}}_{<i})$. See Figure 5.1 (right column).

The convolution $p_\sigma(\tilde{\mathbf{x}}_i|\tilde{\mathbf{x}}_{<i}) = \big(\varphi_\sigma * \tilde{p}_\sigma(\,\cdot\,|\tilde{\mathbf{x}}_{<i})\big)(\tilde{\mathbf{x}}_i)$ has a simple closed form given by a Gaussian mixture model

$$p_\sigma(\tilde{\mathbf{x}}_i|\tilde{\mathbf{x}}_{<i}) = \sum_{k=1}^{d} \tilde{p}_\sigma(e_k|\tilde{\mathbf{x}}_{<i})\varphi_\sigma(\tilde{\mathbf{x}}_i - e_k). \tag{5.13}$$

Using the softmax parameterization of $\tilde{p}(\mathbf{x}_i|\mathbf{x}_{<i})$ given by Equation (5.11), with fine-tuned logits $f_{\sigma,i} : \mathcal{X}^{\otimes i} \to \mathbb{R}^d$, the log-density of this smoothed conditional density can be written in a numerically stable form:

$$\log\, p_\sigma(\mathbf{x}_i|\mathbf{x}_{<i}) = -\log \sum_{\ell=1}^{d} \exp(f_{\sigma,i,\ell}(\mathbf{x}_{<i})) + \log \sum_{k=1}^{d} \exp\left(f_{\sigma,i,k}(\mathbf{x}_{<i}) - \frac{1}{2\sigma^2}(\mathbf{x}_i - e_k)^2\right) + C. \tag{5.14}$$

5.1.4 Stochastic Gradient Langevin Sampling

The Langevin updates described in Equations (5.1) or (5.3) require calculation of $\log p(\mathbf{x})$. This calculation is an $O(n)$ operation for an autoregressive model of length-n sequences. Unlike ancestral sampling, calculating $\log p(\mathbf{x})$ for a given sequence $\mathbf{x} \in \mathbb{R}^n$ decomposes into an embarrassingly parallel set of calculations $\log p_\sigma(\mathbf{x}_i|\mathbf{x}_{<i})$. For moderate sequence lengths n, the cost of computing $\log p(\mathbf{x})$ is essentially constant using a modern parallel computing device (this is what allows autoregressive models to be efficiently trained using the maximum likelihood objective). But when n is large, e.g., for WaveNet models where just a minute of audio has $n > 10^6$ samples, a single device cannot fully parallelize all n conditional likelihood calculations. In this case, it can be convenient to distribute sampling across multiple computing devices. We will now describe a variant of Langevin sampling for autoregressive models (Algorithm 4) that is easily distributed across a cluster of devices.

Instead of making batch updates on a full sequence $\mathbf{x} \in \mathbb{R}^n$, consider updating a single coordinate $j \in \{1, \dots, n\}$:

$$\mathbf{x}_j^{(t+1)} = \mathbf{x}_j^{(t)} + \eta \nabla_{\mathbf{x}_j} \log p_\sigma(\mathbf{x}^{(t)}) + \sqrt{2\eta}\boldsymbol{\varepsilon}_j^{(t)}. \tag{5.15}$$

This coordinate-wise derivative is only dependent on the tail of the sequence $\mathbf{x}_{\geq j}$:

$$\nabla_{\mathbf{x}_j} \log p_\sigma(\mathbf{x}) = \sum_{i=j}^{n} \nabla_{\mathbf{x}_j} \log p_\sigma(\mathbf{x}_i|\mathbf{x}_{<i}). \tag{5.16}$$

This does not yet yield a computational advantage: calculating an update on a single coordinate $\mathbf{x}_j$ required $n - j$ inference calculations $p_\sigma(\mathbf{x}_i|\mathbf{x}_{<i})$. But models over long sequences, including WaveNets, usually make a Markov assumption $p(\mathbf{x}_i|\mathbf{x}_{<i}) = p(\mathbf{x}_i|\mathbf{x}_{i-w}, \dots, \mathbf{x}_{i-1})$ for some limited contextual window of length w. In this case, the coordinate-wise derivative requires only w calls:

$$\nabla_{\mathbf{x}_j} \log p_\sigma(\mathbf{x}) = \sum_{i=j}^{j+w} \nabla_{\mathbf{x}_j} \log p_\sigma(\mathbf{x}_i|\mathbf{x}_{<i}). \tag{5.17}$$

Calculating a gradient on a contiguous block of c coordinates leads to a more efficient update

$$\nabla_{\mathbf{x}_{j:j+c}} \log p_\sigma(\mathbf{x}) = \sum_{i=j}^{j+c+w} \nabla_{\mathbf{x}_{j:j+c}} \log p_\sigma(\mathbf{x}_i|\mathbf{x}_{<i}). \tag{5.18}$$

Calculating Equation (5.18) requires transmission of a block $\{\mathbf{x}_{j-w}, \ldots, \mathbf{x}_{j+c+w}\}$ of length $c + 2w$ to the computing device, and $c + w$ calculations $p_\sigma(\mathbf{x}_i|\mathbf{x}_{<i})$ in order to compute the gradient of a block of length c. If we partition a sequence of length n into n/c blocks of length c, then we can distribute computation of $\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})$ with an overhead factor of $1 + w/c$. This motivates choosing c as large as possible, under the constraint that $c + w$ calculations $p_\sigma(\mathbf{x}_i|\mathbf{x}_{<i})$ can still be parallelized on a single device.

We can calculate $\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})$ by aggregating n/c blocks of gradients according to Equation (5.18), with synchronous communication between n/c machines for every update Equation (5.1); this is a MapReduce algorithm [Dean and Ghemawat, 2008]. We propose a different approach in Algorithm 4 based on block-stochastic Langevin dynamics [Welling and Teh, 2011]. If $j \in \{1, \ldots, n\}$ is chosen uniformly at random then Equation (5.18) is an unbiased estimate of $\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})$. This motivates block-stochastic updates on patches, which multiple devices can perform asynchronously, a Langevin analog to Hogwild! [Niu et al., 2011].

For general likelihoods $p(\mathbf{y}|\mathbf{x})$ (e.g., a classifier) the conditioning values $\mathbf{y}$ may depend on the whole sequence $\mathbf{x}$. In this case, stochastic PnF must read the entire sequence $\mathbf{x}$ in order to calculate the posterior

$$\nabla_{\mathbf{x}_{j:j+c}} \log p(\mathbf{x}|\mathbf{y}) = \nabla_{\mathbf{x}_{j:j+c}} \big(\log p(\mathbf{x}) + \log p(\mathbf{y}|\mathbf{x}) \big). \tag{5.19}$$

But for long sequences $\mathbf{x}$ such as audio, the conditioning information $\mathbf{y}$ is often a local function of the sequence $\mathbf{x}$. In this case, $\mathbf{x} \in \mathcal{X}^n$, $\mathbf{y} \in \mathcal{Y}^n$, $\mathbf{y}_i = g(\mathbf{x}_{N(i)})$ where $N(i)$ is a local neighborhood of indices near i, and the likelihood decomposes via conditional independence

Algorithm 4: Stochastic Autoregressive Langevin Sampling

Input: $\mathbf{y}$, $\{\sigma_i\}_{i=1}^{L}$, δ, T
Sample $\mathbf{x} \sim \mathcal{N}(0, \sigma_1^2 I_n)$
for $i \leftarrow 1$ **to** L **do**
  $\eta_i \leftarrow \delta \cdot \sigma_i^2 / \sigma_L^2$
  Fork()
  for $t = 1$ **to** T **do**
    Sample $j \sim \text{Uniform}\{1, \ldots, n\}$
    Sample $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, I_c)$
    Read $\mathbf{x}^{(t)}_{j-w:j+c+w} \leftarrow \mathbf{x}_{j-w:j+c+w}$
    $\mathbf{g}_j^{(t)} \leftarrow \nabla_{\mathbf{x}_{j:j+c}} \log p_{\sigma_i}(\mathbf{x}^{(t)}_{j-w:j+c+w})$
      $+ \nabla_{\mathbf{x}_{j:j+c}} \log p_{\sigma_i}(\mathbf{y}^{(t)}_{j:j+c} | \mathbf{x}^{(t)}_{j-w:j+c+w})$
    Write $\mathbf{x}_{j:j+c} \leftarrow \mathbf{x}^{(t)}_{j:j+c} + \eta_i \mathbf{g}_j^{(t)} + \sqrt{2\eta}\boldsymbol{\varepsilon}_t$
  end for
  Synchronize()
end for

into

$$\log p(\mathbf{y}|\mathbf{x}) = \sum_{i=1}^{n} \log p(\mathbf{y}_i | \mathbf{x}_{N(i)}). \tag{5.20}$$

All experiments presented in Section 5.3 feature this conditioning pattern. For spectrogram conditioning, $N(i)$ is the set of indices (centered at i) required to compute a short-time Fourier transform. For source separation, super-resolution, and in-painting, $N(i) = i$. This allows us to compute block gradients of the conditional likelihood (Algorithm 4).

5.1.5 Setting the Step Size

We broadly adopt the geometric annealing schedule for Langevin dynamics introduced by Song and Ermon [2019] and elaborated upon by Song and Ermon [2020]. The relationship between the step size η and the smoothing parameter σ is governed by the signal-to-noise ratio (SNR) heuristic proposed by Song and Ermon [2019]. Namely, we maintain a constant signal-to-noise ratio (SNR) between the expected size of the posterior log-likelihood gradient

term $\eta \nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}}|\mathbf{y})$ and the expected size of the Langevin noise $\sqrt{2\eta}\varepsilon$:

$$\mathop{\mathbb{E}}_{\tilde{\mathbf{x}} \sim p_\sigma} \left[\left\| \frac{\eta \nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}}|\mathbf{y})}{\sqrt{2\eta}} \right\|^2 \right] = \frac{\eta}{4} \mathop{\mathbb{E}}_{\mathbf{x} \sim p_\sigma} \left[\left\| \nabla_{\mathbf{x}} \log p_\gamma(\mathbf{y}|\mathbf{x}) + \nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x}) \right\|^2 \right]. \tag{5.21}$$

Assuming that gradients w.r.t. to the likelihood and the prior are uncorrelated, the SNR is approximately

$$\frac{\eta}{4} \mathop{\mathbb{E}}_{\mathbf{x} \sim p_\sigma} \left[\left\| \nabla_{\mathbf{x}} \log p_\gamma(\mathbf{y}|\mathbf{x}) \right\|^2 \right] + \frac{\eta}{4} \mathop{\mathbb{E}}_{\mathbf{x} \sim p_\sigma} \left[\left\| \nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x}) \right\|^2 \right]. \tag{5.22}$$

Empirical work has found that the gradients of a smoothed model $p_\sigma(\tilde{\mathbf{x}}|\mathbf{y})$ are inverse-proportional to the variance of the noise: $\mathbb{E}\|\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x}^{(t)})\|^2 \propto 1/\sigma^2$ [Song and Ermon, 2019]. This relationship has been consistently observed across a variety of datasets, data domains and families of generative models [Jayaram and Thickstun, 2020, Song and Ermon, 2020, Jayaram and Thickstun, 2021]. The SNR heuristic, together with this inverse-proportionality relationship, motivates choosing a step size $\eta = \delta \cdot \sigma^2$ for some problem-dependent constant of proportionality δ.

The empirical inverse proportionality of posterior likelihood gradients and σ^2 could be surprising to the reader, and clearly cannot hold for general probability densities. For example, gradients of the smoothed prior densities $p_\sigma(\tilde{\mathbf{x}})$ (Equation (5.8)) can be described by convolution of p with a Gaussian kernel:

$$\nabla_{\tilde{\mathbf{x}}} \log p_\sigma(\tilde{\mathbf{x}}) = \nabla_{\tilde{\mathbf{x}}} \log \mathop{\mathbb{E}}_{\boldsymbol{\varepsilon} \sim \mathcal{N}(0,I)} \left[p(\tilde{\mathbf{x}} - \sigma\boldsymbol{\varepsilon}) \right]. \tag{5.23}$$

From this expression, assuming p is continuous, we clearly see that the gradients are asymptotically independent of σ:

$$\lim_{\sigma \to 0} \nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}). \tag{5.24}$$

Maintaining proportionality $\mathbb{E}\|\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})\|^2 \propto 1/\sigma^2$ would require the gradients to grow

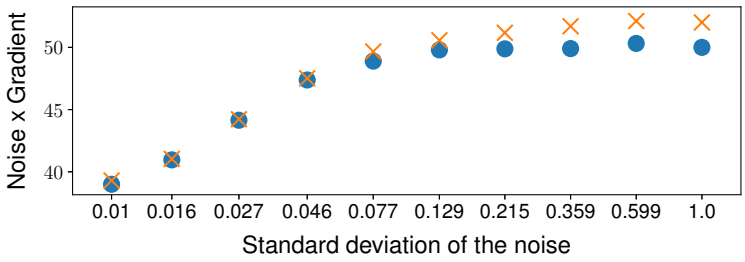


Figure 5.2: The behavior of $\sigma \times \|\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})\|$ in expectation for the NCSN (orange) and Glow (blue) models trained on CIFAR-10 at each of 10 noise levels as σ decays geometrically from 1.0 to 0.01. For large σ, $\|\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})\| \approx 50/\sigma$. This proportional relationship breaks down for smaller σ. Because the expected gradient of the noiseless density $\log p(\mathbf{x})$ is finite, its product with σ must asymptotically approach zero as $\sigma \to 0$.

unbounded as $\sigma \to 0$, but the gradients of the noiseless distribution $\log p(\mathbf{x})$ are finite. Therefore, proportionality must eventually break down as $\sigma \to 0$. Similar reasoning apples to the likelihood $p_\sigma(\mathbf{y}|\tilde{\mathbf{x}})$.

We conjecture that the proportionality between the gradients and the noise is a consequence of severe non-smoothness in the model $p(\mathbf{x})$. For example, in the visual domain the probability mass of this distribution is peaked around plausible images $\mathbf{x}$, and decays rapidly away from these points in most directions. Consider the extreme case where the prior has a Dirac delta point mass. The convolution of a Dirac delta with a Gaussian is itself Gaussian so, near the point mass, the noisy distribution p_σ will be proportional to a Gaussian density with variance σ^2. If p_σ were exactly Gaussian then analytically

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim p_\sigma}\left[\|\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})\|^2\right] = \frac{1}{\sigma^4} \mathop{\mathbb{E}}_{\mathbf{x}\sim p_\sigma}\left[\mathbf{x}^2\right] = \frac{1}{\sigma^2}. \tag{5.25}$$

Because the distribution $p(\mathbf{x})$ does not contain actual point masses—only approximations thereof—we would expect this proportionality to eventually break down as $\sigma \to 0$. Indeed, Figure 5.2 shows that both for NCSN and Glow models of CIFAR-10, after maintaining a consistent proportionality $\mathbb{E}\left[\|\nabla_{\mathbf{x}} \log p_\sigma(\mathbf{x})\|^2\right] \propto 1/\sigma^2$ at the higher noise levels, the decay of σ^2 to zero eventually outpaces the growth of the gradients.

We conclude with some comments about two special cases. First, for the discretized

autoregressive models discussed in Section 5.1.3, the noiseless distribution $p(\mathbf{x})$ is genuinely a mixture of Dirac spikes. In this case, the analysis above applies without caveats and the proportionality relationship between gradients and the smoothing parameter does not break down as $\sigma \to 0$ (the precise constant of proportionality remains application dependent, and is discussed in the experimental details in Section 5.3). Second, when $\mathbf{y} = g(\mathbf{x})$ is given by a linear measurement model, $\log p_\sigma(\mathbf{y}|\tilde{\mathbf{x}})$ is a simple concave quadratic. In this case, we can show analytically that $\mathbb{E}\left[\|\nabla_{\tilde{\mathbf{x}}} \log p_\gamma(\mathbf{y}|\tilde{\mathbf{x}})\|^2\right] \propto 1/\sigma^2$.

5.2 Linear Inverse Problems and Source Separation

Linear inverse problems ask us to recover an unobserved object $\mathbf{x}$ given measurements $\mathbf{y} = g(\mathbf{x})$ for a known linear function $g : \mathbb{R}^n \to \mathbb{R}^m$. Many practical visual and audio task can be cast in this framework: source separation, image colorization, inpainting and outpainting, super-resolution, and sparse recovery. We will generally be interested in the undetermined setting, where $m < n$. In this setting many solutions of $\mathbf{x} = g^{-1}(\mathbf{y})$ exist, and we must impose some regularization on the problem to resolve the ambiguity, such as sparsity [Elad, 2010, Mairal et al., 2014]. In our setting, this regularization takes the form of a Bayesian prior $p(\mathbf{x})$ learned from data.

Bayesian inverse problems are explored extensively in theoretical settings, where the prior is given by a simple analytical distribution [Tropp and Wright, 2010, Knapik et al., 2011, Wang et al., 2017]. Linear inverse problems have also been studied using learned priors given by GAN's [Chang et al., 2017, Bora et al., 2017, Raj et al., 2019]. GAN-based approaches are tailored to the latent variable architecture of the model, performing latent space optimizations that find codes corresponding to desired outputs. There is no obvious analog of these latent variable approaches for general likelihood-based models, for example autoregressive models. Langevin sampling offers a general approach to using likelihood-based models as priors for Bayesian inverse problems.

We will focus our attention on the single-source separation task [Davies and James, 2007], a representative example of a linear inverse problem with broad practical importance.

Single-channel source separation requires us to decompose a mixed signal $\mathbf{y} \in \mathbb{R}^m$ into a linear combination of k components $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^m$ with scalar mixing coefficients $\alpha_i \in \mathbb{R}$:

$$\mathbf{y} = g(\mathbf{x}) \equiv \sum_{i=1}^{k} \alpha_i \mathbf{x}_i. \tag{5.26}$$

This is motivated by, for example, the "cocktail party problem" of isolating the utterances of individual speakers $\mathbf{x} \in \mathbb{R}^{k \times m}$ from an audio mixture $\mathbf{y} \in \mathbb{R}^m$ captured at a busy party, where multiple speakers are talking simultaneously. We review related work on source separation in Section 5.2.1.

We will solve a Bayesian inverse problem by sampling from the posterior distribution $p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$. We can view a linear measurement model $\mathbf{y} = g(\mathbf{x})$ as degenerate likelihoods $p(\mathbf{y}|\mathbf{x})$ of the form

$$p(\mathbf{y}|\mathbf{x}) = \delta(\mathbf{y} - g(\mathbf{x})), \tag{5.27}$$

where δ denotes the Dirac delta function. In this case, the smoothed densities $p_\sigma(\mathbf{y}|\tilde{\mathbf{x}})$ discussed in Section 5.1.2 can be calculated in closed form. Writing the linear function g as a matrix $g(\mathbf{x}) = A\mathbf{x}$, it can be shown that

$$\mathbf{y} = g(\mathbf{x}) = g(\tilde{\mathbf{x}}) + g(-\boldsymbol{\varepsilon}_\sigma) \sim \mathcal{N}\left(g(\tilde{\mathbf{x}}), \sigma^2 AA^T\right). \tag{5.28}$$

The smoothing $p_\sigma(\mathbf{y}|\tilde{\mathbf{x}}) = \mathcal{N}\left(g(\tilde{\mathbf{x}}), \sigma^2 AA^T\right)$ generalizes the smoothing proposed by Jayaram and Thickstun [2020] for source separation. In that work, we proposed separately smoothing the prior and likelihood, resulting in a smoothed likelihood $p(\tilde{\mathbf{y}}|\mathbf{x})$ over $\tilde{\mathbf{y}} = \mathbf{y} + \boldsymbol{\varepsilon}_\mathbf{y}$, where $\boldsymbol{\varepsilon}_\mathbf{y} \sim \mathcal{N}(0, \sigma^2 I_n)$. This is equivalent to Equation (5.28) in the case of source separation, for which $g(\mathbf{x}) = \frac{1}{2}\mathbf{x}_1 + \frac{1}{2}\mathbf{x}_2$ and therefore $p_\sigma(\mathbf{y}|\tilde{\mathbf{x}}) = \mathcal{N}\left(g(\tilde{\mathbf{x}}), \sigma^2 I\right)$.

For some mixtures the posterior distribution $p(\mathbf{x}|\mathbf{y})$ is quite peaked, and sampling from this distribution recovers the only plausible inverse of the observation $\mathbf{y}$. Even with a strong generative prior, inverse problems including source separation can be ambiguous. Visual examples of this ambiguity are illustrated in Figures 5.3 and 5.4. This motivates our interest

in sampling, which explores the space of plausible separations. It also poses a challenge for traditional source separation metrics, which presume that the original mixture components are identifiable and compare the separated components to ground truth. For ambiguous mixtures of rich data, recovery of ground truth $\mathbf{x}$ is not a well-posed problem. Instead, we should ask whether proposed inverses are consistent with an observed mixture under a given prior data distribution. Motivated by this perspective, we discuss evaluation metrics for source separation in Section 5.2.2.

For practical source separation tasks, where we are interested in generating a single separation result, it is natural consider providing a maximum a-posteriori (MAP) estimate rather than a sample from the posterior distribution. As we will see in Section 5.2.3, a greedy effort to generate a MAP estimate by ascending gradients of the posterior log-likelihood produces undesirable results. We will see in Section 5.3.5 that a more cautious approach, using an empirical estimate of the MAP over multiple samples, can improve results for practical tasks such as source separation. But there are reasons to prefer sampling over an aggressive attempt to obtain a MAP estimate; maximum-likelihood training methods optimize a model's likelihood over typical data, but have been observed to behave strangely at modes of the model distribution [Holtzman et al., 2020].

5.2.1 Related Work on Source Separation

Blind separation. Classical "blind" approaches to single-channel source separation resolve ambiguity by privileging solutions to (5.26) that satisfiy mathematical constraints on the components $\mathbf{x}$, such as statistical independence [Comon, 1994, Bell and Sejnowski, 1995, Davies and James, 2007] sparsity [Lee et al., 1999, Zibulevsky and Pearlmutter, 2001, Li et al., 2006, Wilson et al., 2008] or non-negativity [Lee and Seung, 1999, Schmidt and Olsson, 2006, Erdogan and Grais, 2010]. These constraints can be be viewed as weak priors on the structure of sources, but the approaches are blind in the sense that they do not require adaptation to a particular dataset. Because blind methods have no access to sample components, they face the challenging task of modeling the distribution over unobserved components while

simultaneously decomposing mixtures into likely components. It is difficult to fit a rich model to latent components, so blind methods often rely on simple models such as dictionaries to capture the structure of these components.

One promising recent work in the blind setting is Double-DIP Gandelsman et al. [2019]. This work leverages the unsupervised Deep Image Prior Ulyanov et al. [2018] as a prior over signal components, similar to our use of a trained generative model. But the authors of this work document fundamental obstructions to applying their method to single-channel source separation; they propose using multiple image frames from a video, or multiple mixtures of the same components with different mixing coefficients α. This multiple-mixture approach is common to much of the work on blind separation. In contrast, our approach is able to separate components from a single mixture.

Supervised regression. In contrast to the blind approach, most recent work on source separation is data-driven. To separate a mixture of sources, it is natural to suppose that we have access to samples $\mathbf{x}$ of individual sources, which can be used as a reference for what the source components of a mixture are supposed to look like. This data can be used to regularize solutions of Equation (5.26) towards structurally plausible solutions. The prevailing way to do this is to construct a supervised regression model that maps an input mixture $\mathbf{y}$ to components $\mathbf{x}_i$ [Huang et al., 2014, Halperin et al., 2019]. Paired training data $(\mathbf{y}, \mathbf{x})$ can be constructed by summing randomly chosen samples from the component distributions $\mathbf{x}_i$ and labeling these mixtures with the ground truth components.

Regression models have been extensively studied for separation of images [Halperin et al., 2019], audio spectrograms [Huang et al., 2014, 2015, Nugraha et al., 2016, Jansson et al., 2017], and raw audio [Lluís et al., 2019, Stoller et al., 2018b, Défossez et al., 2019], as well as more exotic data domains, e.g., medical imaging [Nishida et al., 1999]. By learning to predict components (or equivalently, masks on a mixture) this approach implicitly builds a generative model of the signal components. This connection is made more explicit in recent work that uses GAN's to force components emitted by a regression model to match the distribution of a given dataset [Zhang et al., 2018a, Stoller et al., 2018a].

The supervised approach takes advantage of expressive deep models to capture a strong prior over signal components. But it requires specialized model architectures trained specifically for the source separation task. In contrast, our approach leverages standard, pre-trained generative models for source separation. Furthermore, our approach can directly exploit ongoing advances in likelihood-based generative modeling to improve separation results.

Signal Dictionaries. Much work on source separation is based on the concept of a signal dictionary, most notably the line of work based on non-negative matrix factorization (NMF) [Lee and Seung, 2000]. These approaches model signals as combinations of elements in a latent dictionary. Decomposing a mixture into dictionary elements can be used for source separation by (1) clustering the elements of the dictionary and (2) reconstituting a source using elements of the decomposition associated with a particular cluster. Dictionaries are typically learned from data of each source type and combined into a joint dictionary, clustered by source type [Schmidt and Olsson, 2006, Virtanen, 2007]. The blind setting has also been explored, where the clustering is obtained without labels by e.g., k-means [Spiertz and Gnann, 2009]. Recent work explores more expressive decomposition models, replacing the linear decompositions used in NMF with expressive neural autoencoders [Smaragdis and Venkataramani, 2017, Venkataramani et al., 2017].

When the dictionary is learned with supervision from labeled sources, dictionary clusters can be interpreted as implicit priors on the distributions over components. Our approach makes these prior explicit, and works with generic priors that are not tied to the dictionary model. Furthermore, our method can separate mixed sources of the same type, whereas mixtures of sources with similar structure present a conceptual difficulty for dictionary-based methods.

Generative adversarial separation. Recent work by Sübakan and Smaragdis [2018] and Kong et al. [2019] explores the intriguing possibility of optimizing $\mathbf{x}$ given a mixture $\mathbf{m}$ to satisfy (5.26), where components $\mathbf{x}_i$ are constrained to the manifold learned by a GAN. The GAN is pre-trained to model a distribution over components. Like our method, this approach leverages modern deep generative models in a way that decouples generation from

source separation. We view this work as a natural analog to our likelihood-based approach in the GAN setting.

Likelihood-based approaches. Our approach is similar in spirit to older ideas based on maximum a posteriori estimation [Geman and Geman, 1984], likelihood maximization [Pearlmutter and Parra, 1996, Roweis, 2000], and Bayesian source separation [Benaroya et al., 2006]. We build upon their insights, with the advantage of increased computational resources and modern expressive generative models.

5.2.2 Evaluation Methodology

Much of the previous work on source separation evaluates results using peak signal-to noise ratio (PSNR) or structural similarity index (SSIM) Wang et al. [2004]. These metrics assume that the original sources are identifiable; in probabilistic terms, the true posterior distribution $p(\mathbf{x}|\mathbf{m})$ is presumed to have a unique global maximum achieved by the ground truth sources (up to permutation of the sources). Under the identifiability assumption, it is reasonable to measure the quality of a separation algorithm by comparing separated sources to ground truth mixture components. PSNR, for example, evaluates separations by computing the mean-squared distance between pixel values of the ground truth and separated sources on a logarithmic scale.

For CIFAR-10 source separation, the ground truth source components of a mixture are not identifiable. As evidence for this claim, we call the reader's attention to Figure 5.3. For each mixture depicted in Figure 5.3, we present separation results that sum to the mixture and (to our eyes) look plausibly like CIFAR-10 images. However, in each case the separated images exhibit high deviation from the ground truth. This phenomenon is not unusual; Figure 5.4 shows an un-curated collection of samples from $p(\mathbf{x}|\mathbf{m})$ using BASIS, illustrating a variety of plausible separation results for each given mixture. We will later see evidence again of non-identifiability in Figure 5.13. If we accept that the separations presented in Figures 5.3, 5.4, and 5.13 are reasonable, then source separation on this dataset is fundamentally underdetermined; we cannot measure success using metrics like PSNR that

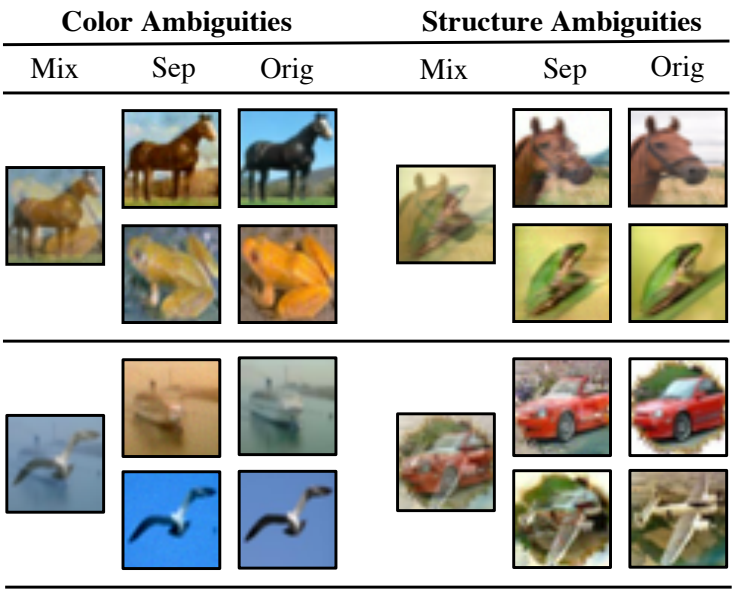


Figure 5.3: A curated collection of examples demonstrating color and structural ambiguities in CIFAR-10 mixtures. In each case, the original components differ substantially from the components separated by BASIS using NCSN as a prior. But in each case, the separation results also look like plausible CIFAR-10 images.

compare separation results to ground truth.

Instead of comparing separations to ground truth, we propose instead to quantify the extent to which the results of a source separation algorithm look like samples from the data distribution. If a pair of images sum to the given mixture and look like samples from the data distribution, we deem the separation to be a success. This shift in perspective from identifiability of the latent components to the quality of the separated components is analogous to the classical distinction in the statistical literature between estimation and prediction [Shmueli et al., 2010, Bellec et al., 2018]. To this end, we borrow the Inception Score (IS) [Salimans et al., 2016] and Frechet Inception Distance (FID) [Heusel et al., 2017] metrics from the generative modeling literature to evaluate CIFAR-10 separation results. These metrics attempt to quantify the similarity between two distributions given samples. We use them to compare the distribution of components produced by a separation algorithm to the distribution of ground truth images.

In contrast to CIFAR-10, the posterior distribution $p(\mathbf{x}|\mathbf{m})$ for an MNIST model is

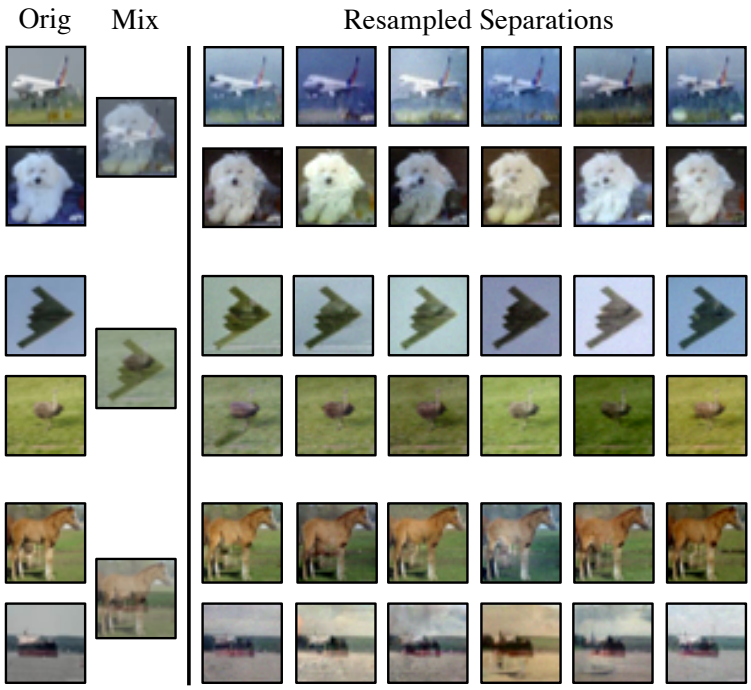


Figure 5.4: Repeated sampling using BASIS with NCSN as a prior for several mixtures of CIFAR-10 images. While most separations look reasonable, variation in color and lighting makes comparative metrics like PSNR unreliable. This challenges the notion that the ground truth components are identifiable.

demonstrably peaked. Moreover, BASIS is able to consistently identify these peaks. This constitutes a constructive proof that components of MNIST mixtures are identifiable, and therefore comparisons to the ground-truth components make sense. We report PSNR results for MNIST, which allows us to compare the results of BASIS to other recent work on MNIST image separation Halperin et al. [2019], Kong et al. [2019].

Although we do not directly model the posterior likelihood $p(\mathbf{x}|\mathbf{m})$, we can compute the log-likelihood of the output samples $\mathbf{x}$. The log-likelihood is a function of the artificial variance hyper-parameter γ, so it is more informative to look at the unweighted square error $\|\mathbf{m}-g(\mathbf{x})\|^2$; this quantity can be interpreted as a reconstruction error, and measures how well we approximate the hard mixture constraint. Because we geometrically anneal the variance σ of the posterior distribution, by the end of optimization the inverse constraint is rigorously enforced; per-pixel and per-sample reconstruction error is smaller than the quantization level

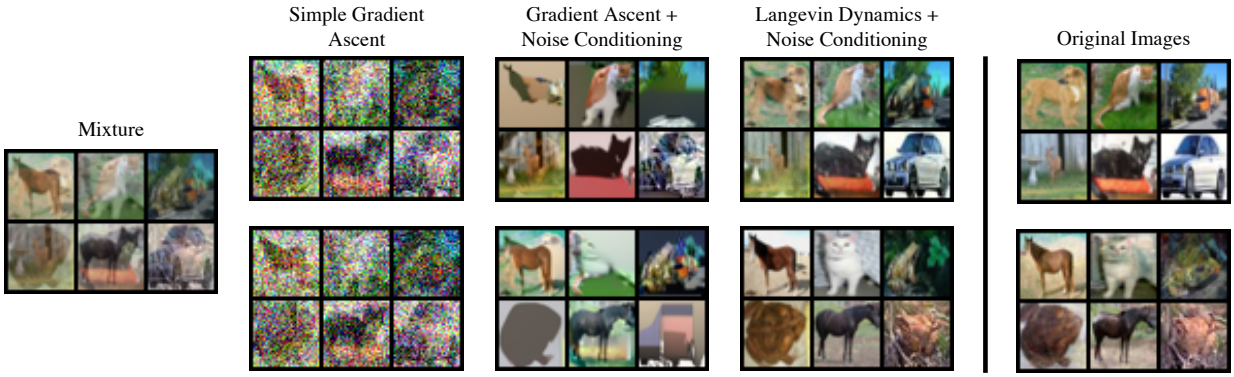


Figure 5.5: Non-stochastic gradient ascent produces sub-par results. Annealing over smoothed-out distributions (Noise Conditioning) guides the optimization towards likely regions of pixel space, but gets stuck at sub-optimal solutions. Adding Gaussian noise to the gradients (Langevin dynamics) shakes the optimization trajectory out of bad local optima.

of 8-bit color, resulting in pixel-perfect visual reconstructions.

5.2.3 The Importance of Stochasticity

Injecting Gaussian noise into the gradients for Langevin sampling is essential. Setting aside the formal setting of Bayesian posterior sampling, it is tempting to simply run gradient ascent on the pixels of the components to maximize the likelihood of these components under the prior, with a Lagrangian term to enforce the constraint $g(\mathbf{x}) = \mathbf{y}$:

$$\mathbf{x} \leftarrow \mathbf{x} + \eta \nabla_{\mathbf{x}} \left[\log p(\mathbf{x}) - \lambda \|g(\mathbf{x}) - \mathbf{y}\|^2 \right]. \tag{5.29}$$

But this does not work. As demonstrated in Figure 5.5, there are many local optima in the loss surface of $p(\mathbf{x})$ and a greedy ascent procedure simply gets stuck. Pragmatically, the noise term in Langevin dynamics can be seen as a way to knock the greedy optimization (5.29) out of local maxima.

In the recent literature, pixel-space optimizations by following gradients $\nabla_{\mathbf{x}}$ of some objective are perhaps associated more with adversarial examples than with desirable results [Goodfellow et al., 2015, Nguyen et al., 2015]. We note that there have been some successes

of pixel-wise optimization in texture synthesis [Gatys et al., 2015] and style transfer [Gatys et al., 2016]. But broadly speaking, pixel-space optimization procedures often seem to go wrong. We speculate that noisy optimizations (5.4) on smoothed-out objectives like p_σ could be a widely applicable method for making pixel-space optimizations more robust.

5.3 Empirical Sampling Results

We we now investigate the empirical behavior of Langevin posterior sampling for a variety of datasets and models in the visual and audio domains. In Section 5.3.1, we discuss the datasets used for the experiments. In Section 5.3.2 we review the models used in the experiments, and the hyper-parameters of Langevin sampling when each of these models is used as a prior. In Section 5.3.3 we explore the extent to which the Langevin sampler can accurately approximate samples from the prior $p(\mathbf{x})$ and posterior $p(\mathbf{x}|\mathbf{y})$. In Section 5.3.4 we show that the stochastic Langevin sampler introduced in Section 5.3.4 is faster than the ancestral sampler for autoregressive models, when parallelized across a modest number of devices. Finally, we show how Langevin sampling can be applied to a variety of visual and audio posterior sampling tasks: source separation (Section 5.3.5), super-resolution (Section 5.3.7), and inpainting (Section 5.3.8). Qualitative results for visual posterior sampling are shown in Figure 5.6. Code and instructions for reproducing these experiments is available online for visual source separation using NCSN and Glow models [1] and autoregressive Langevin sampling using PixelCNN++ and WaveNet models.[2]

5.3.1 Datasets

For visual experiments we work with three datasets of images: the MNIST 28×28 grayscale dataset [LeCun et al., 1998], CIFAR-10 32×32 RGB color dataset [Krizhevsky, 2009], and LSUN dataset [Yu et al., 2015], downsampled to 64×64 RGB color. In each case, pixel channels are represented using scalar values normalized to $[0, 1]$ with 8-bit linear quantization.

[1] https://github.com/jthickstun/basis-separation

[2] https://grail.cs.washington.edu/projects/pnf-sampling/

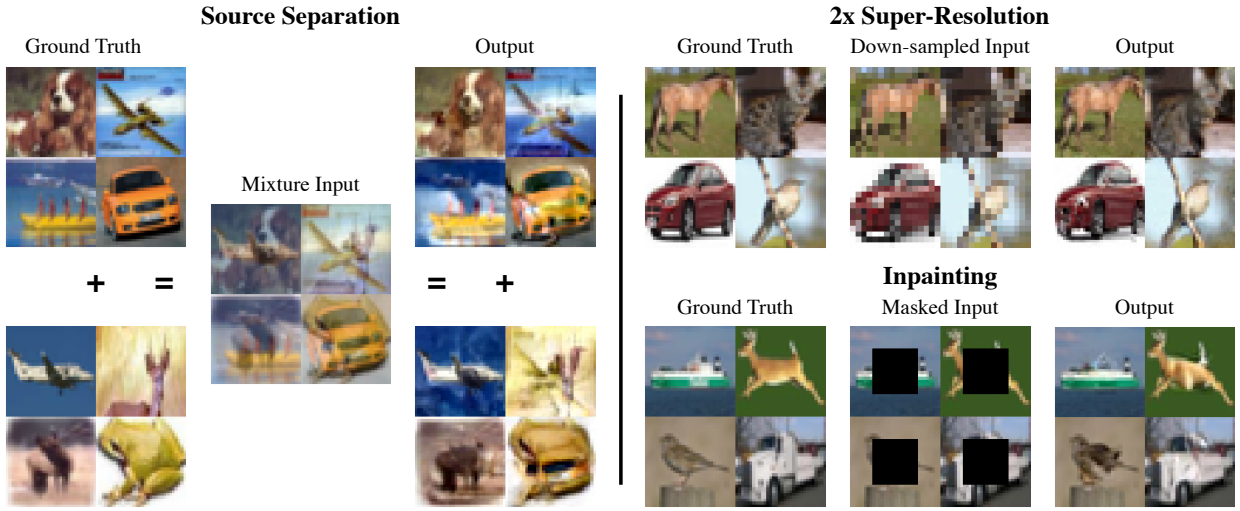


Figure 5.6: Langevin sampling applied to visual source separation (Section 5.3.5) super-resolution (Section 5.3.7) and inpainting (Section 5.3.8) using a PixelCNN++ prior trained on CIFAR-10 images. Ground-truth images are taken from the CIFAR-10 test set.

Generative priors $p(\mathbf{x})$ are trained on the dataset's standard training split, and evaluated on images in the test set.

For audio experiments we use the VCTK dataset [Veaux et al., 2016] consisting of 44 hours of speech and the Supra Piano dataset [Shi et al., 2019] consisting of 52 hours of piano recordings. We use a random 80-20 train-test split of VCTK speakers and piano recordings for evaluation. Audio sequences are sampled at a 22kHz with sound pressure levels normalized to $[0, 255]$. These pressure levels are quantized using 8-bit μ-law encoding [CCITT, 1988], except for the source separation task, in which case 8-bit linear encoding is used. Sequences used for quantitative evaluation are 50k sample excerpts, approximately 2.3 seconds of audio, chosen randomly from the longer test set recordings.

5.3.2 Generative Priors

We investigate four likelihood-based generative models as priors for posterior sampling tasks: the score-based NCSN model [Song and Ermon, 2019], the flow-based Glow model [Kingma and Dhariwal, 2018], and autoregressive models PixelCNN++ [Salimans et al., 2017] and WaveNet [van den Oord et al., 2016a]. Details of the experimental setup and

hyper-parameters used for Langevin sampling in conjunction with each model are discussed below.

(NCSN) We use the official implementation of NCSN, using the pre-trained weights published by the authors.[3] These noise-conditioned models are designed to be used with the noise-annealed Langevin sampler. We use the standard geometric annealing schedule proposed by Song and Ermon [2019], beginning from $\sigma_1 = 1.0$ and annealing to $\sigma_L = 0.01$ over $L = 10$ discrete noise levels. We set $\delta = 2 \times 10^{-5}$, and $T = 100$. All these hyper-parameters are the same for both the MNIST and CIFAR-10 sampling experiments.

(Glow) We use the official implementation of Glow, using the pre-trained weights published by the authors.[4] We adopt the hyper-parameters proposed for NCSN, for each of the MNIST, CIFAR-10, and LSUN sampling experiments. Using the pre-trained models of $p(\mathbf{x})$ published by the Glow authors, we fine-tune these models on noise-perturbed data $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. Empirically, this procedure quickly converges to an estimate of $p_\sigma(\tilde{\mathbf{x}})$ within 10 epochs.

(PixelCNN++) We use a public implementation of PixelCNN++ written by Lucas Caccia.[5] We used the pre-trained CIFAR-10 weights for this model shared by Lucas Caccia (at the link above) with a reported log loss of 2.95 bits per dimension on the CIFAR-10 test set. For the models $p_\sigma(\mathbf{x})$, we fine-tuned the pre-trained model for 10 epochs at each noise level σ^2. We adopt a geometric annealing schedule for σ^2, beginning at $\sigma_1 = 1.0$ and ending at $\sigma_L = 0.01$ using $L = 19$ noise levels. This is twice the number of noise levels used for NCSN and Glow. We also found that sample quality improved using a smaller learning rate and mixing for more iterations for the NCSN and Glow models. For conditional sampling tasks, we set $\delta = 3e-06$ and $T = 300$ in contrast to $\delta = 2e-05$ and $T = 100$ used in previous work. In wall-clock time, we find that conditional PixelCNN++ sampling tasks require approximately 60 minutes to generate a batch of 16 samples using a 1080Ti GPU. We

[3] https://github.com/ermongroup/ncsn.

[4] https://github.com/openai/glow.

[5] https://github.com/pclucas14/pixel-cnn-pp.

speculate that the need for more levels of annealing and slower mixing may be attributable to the autoregressive model parameterization; similar adjustments to L and δ are required for the WaveNet models.

(WaveNet) Our audio sampling experiments are performed using a WaveNet model $p(\mathbf{x})$ trained on both the VCTK and Supra Piano datasets. We used the public implementation of Wavenet written by Ryuichi Yamamoto.[6] While we focus on WaveNet, due to its prominence as a generative model for audio, Langevin sampling could be applied more generally with other likelihood-based models. In the audio space, this includes recent diffusion models [Kong et al., 2021, Chen et al., 2021]. Note however that audio vocoder models [Prenger et al., 2019, Kim et al., 2019, Ping et al., 2020], which rely on spectrogram conditioning, cannot be adapted as priors for the source separation, super-resolution, and inpainting experiments. In addition, GAN based models [Donahue et al., 2019a, Kumar et al., 2019], which are not likelihood based, cannot be sampled using Langevin dynamics.

For all audio experiments, where data is encoded with values $\{0, \ldots, 255\}$, we use $L = 15$ noise levels geometrically spaced between $\sigma = 175.9$ and $\sigma = 0.15$. The same noise levels are also used for the sampling speed and quality results presented in Figures 5.7 and 5.8. For all experiments, the number of Langevin steps per noise level $T = 256$, except for Figure 5.7 where that parameter is varied to highlight changes in likelihood. The learning rate multiper $\delta = 0.05$ for all experiments. The Markov window w is based on the underlying architecture. When training the fine-tuned noise models, all training hyperparameters are kept the same as the original WaveNet implementation. For the WaveNet implementation used in this paper, this is 6139 samples which is roughly 0.3 seconds at a 22kHz sample rate Please refer to the WaveNet paper or the public WaveNet implementation for training details.

As discussed by [van den Oord et al., 2016a], 8-bit μ-law encoding results in a higher fidelity representation of audio than 8-bit linear encoding. For most experiments, the observation constraint $y = g(x)$ is still linear even under a μ-law encoding of x. However,

[6] https://github.com/r9y9/wavenet_vocoder.

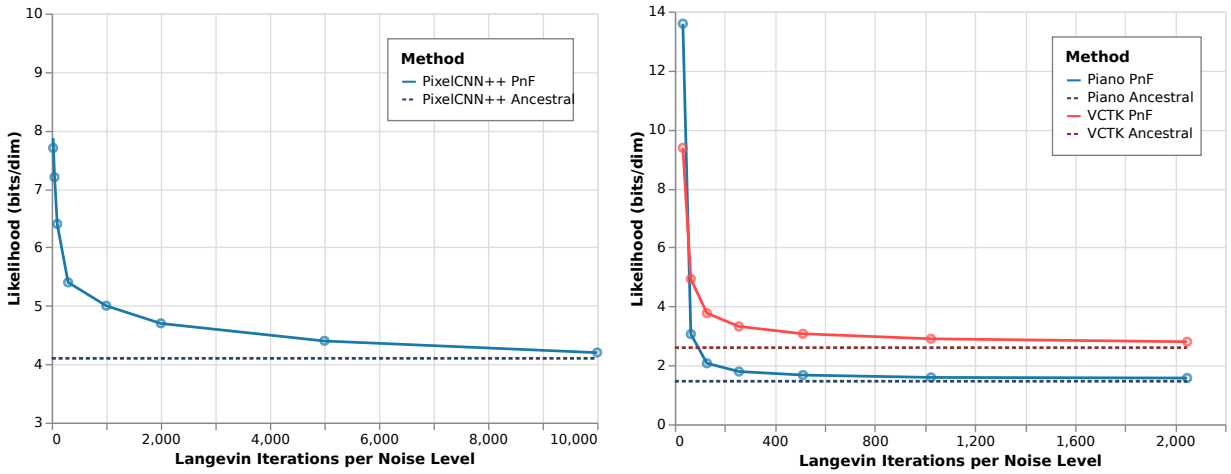


Figure 5.7: As the number of Langevin iterations T increases, the log-likelihood of sequences generated by stochastic Langevin sampling approaches the log-likelihood of test set sequences. Left: sampling from a PixelCNN++ model trained on CIFAR-10. Right: sampling from WaveNet models trained on the Supra Piano and VCTK speech datasets.

for source separation, the constraint $y = x_1 + x_2$ is no longer linear under μ-law encoding. Consequently, we use an 8-bit linear encoding of x for source separation experiments to avoid a change of variables calculation. To facilitate a fair comparison, all ground truths and baselines shown in the demos use the corresponding μ-law or linear 8-bit encoding.

5.3.3 Quality of Generated Samples

To evaluate the quality of samples generated by Langevin sampling, we follow a similar procedure to Holtzman et al. [2020]. We compare log-likelihoods, calculated using the noiseless model $p(\mathbf{x})$, of sequences generated by Langevin sampling to sequences generated by ancestral sampling from the lowest-noise model $p_{\sigma_L}(\mathbf{x})$. Because Langevin-sampled sequences are continuous, we quantize these samples to 8-bit values when evaluating their likelihood under the noiseless model. We consider the sampling procedure to be successful if it generates sequences with comparable log-likelihoods to ancestral generations.

In Figure 5.7 we present quantitative results for Langevin sampling using an unconditional PixelCNN++ model of CIFAR-10, and a spectrogram-conditioned WaveNet model of both voice and piano datasets. We evaluate $1,000$ generations (length $n = 50,000$ sequences for

Dataset	$p(\mathbf{x}_{\text{test}})$	$p_{\sigma_L}(\mathbf{x}_{\text{test}})$	$p_{\sigma_L}(\mathbf{x}_{\text{Langevin}})$
MNIST	0.5	3.6	3.6
CIFAR-10	3.4	4.5	4.7
LSUN (bed)	2.4	4.2	4.4
LSUN (crh)	2.7	4.4	4.4

Table 5.1: The mean log-likelihood under the minimal-noise Glow prior $p_{\sigma_L}(\mathbf{x})$ for the test set $\mathbf{x}_{\text{test}}$, and for samples of 100 Langevin separations $\mathbf{x}_{\text{Langevin}}$. The log-likelihood of each test set under the noiseless prior $p(\mathbf{x}_{\text{test}})$ is reported for reference.

the WaveNet models) using various numbers of Langevin iterations, and report the median log likelihood of quantizations of these sequences under the noiseless model. Asymptotically, as the iterations T of Langevin dynamics increase, the likelihood of Langevin samples approaches the likelihood of ancestral samples. Generated audio samples for various T are available on the website.

In Table 5.1, we consider the quality of conditional samples generated for the source separation task using Glow as a prior. How do the probabilities of sources $\mathbf{x}_{\text{Langevin}}$ generated by Langevin posterior sampling compare to the probabilities of data $\mathbf{x}_{\text{test}}$ taken directly from a dataset's test set? Because we anneal the noise to a fixed level $\sigma_L > 0$, we find it most informative to ask this question using the minimal-noise, fine-tuned prior $p_{\sigma_L}(\mathbf{x})$. As seen in Table 5.1, the outputs of the Langevin posterior sampler are generally comparable in log-likelihood to test set images under $p_{\sigma_L}(\mathbf{x})$; Langevin posterior sampling separates a mixture into components that are typical under the prior. The bits/dim reported in Column 2 of Table 5.1 are substantially higher than the standard test-set log-likelihoods reported in Column 1 because, even at $\sigma = 0.01$, the Gaussian noise injects considerable (albeit imperceptible) entropy into the models.

5.3.4 Speed and Parallelism

Ancestral sampling has $O(n)$ serial runtime in the length n of the generated sequence. Using the stochastic Langevin sampler described in Section 5.1.4, serial runtime is $O(T)$, where T is the number of Langevin iterations at each level of smoothing. We find empirically that

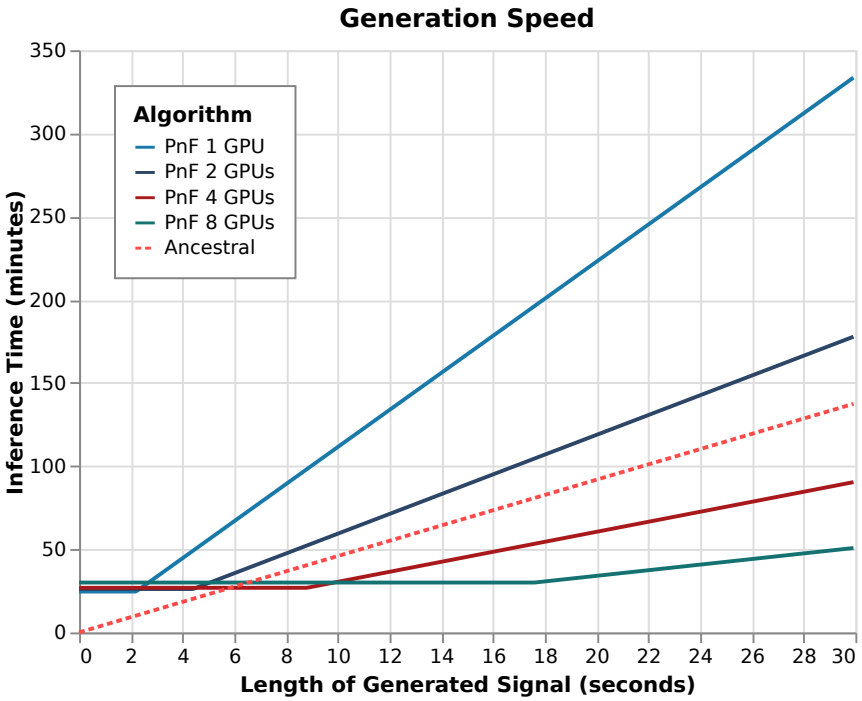


Figure 5.8: Stochastic Langevin sampling can be parallelized across multiple devices, resulting in faster inference time than ancestral sampling. Beyond a threshold level of computation, Langevin sampling time is inversely proportional to the number of devices.

we can set T independent of n, and therefore the theoretical serial runtime of stochastic Langevin sampling is constant as a function of sequence length. In practice, we do not have an infinite supply of parallel devices, so the serial runtime of stochastic Langevin sampling grows inversely proportional to the number of devices. This behavior is demonstrated in Figure 5.8 for WaveNet sampling using a cluster of 8 Nvidia Titan Xp GPU's and $T = 256$. Each GPU can calculate Equation (5.18) for a block of $c = 50,000$ samples (approximately 2.3 seconds of audio).

Stochastic Langevin sampling depends upon asynchronous writes being sparse so that memory overwrites, when two workers update overlapping blocks, are rare. This situation is analogous to the sparse update condition required for Hogwild! If blocks are length c and the number of devices is substantially less than n/c, then updates are sufficiently sparse. But if the number of devices is larger than n/c, memory overwrites become common, and sampling fails to converge. This imposes a floor on generation time determined by c, exhibited in

Figure 5.8. We cannot substantially reduce this floor by decreasing c because of the tradeoff between c and the model's Markov window w described in Section 5.1.4.

In general, annealed Langevin sampling becomes faster than ancestral sampling for autoregressive models of very long sequences n, in which case the stochastic variant of the Langevin sampler becomes necessary in order to distribute the calculation of n conditional likelihoods. For autoregressive models of short sequences, accurate unconditional samples can be achieved more quickly with the ancestral sampler. For example, unconditional 32×32 CIFAR-10 generation using the PixelCNN++ mode requires $T = 10,000$ Langevin iterations per noise level (Figure 5.7) for accurate samples; annealing through $L = 20$ levels (Section 5.3.2) requires a total of $L \times T = 200,000$ serial queries to the PixelCNN++ model, far more than $n = 3 \times 32 \times 32$ serial queries to PixelCNN++ for ancestral sampling of a CIFAR-10 size image. Note also that PixelCNN++ conditions on a full image ($w = n$; see Section 5.1.4) so the stochastic variant of Langevin sampling is not applicable to this model.

5.3.5 Source Separation

For our quantitative source separation experiments, we consider the recovery of two unobserved sources $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}^n = \mathbb{R}^{2 \times n}$ given an observed mixture $\mathbf{y} = g(\mathbf{x}) = \frac{1}{2}\mathbf{x}_1 + \frac{1}{2}\mathbf{x}_2$ with equal weights on components. As outlined in Section 5.2, we view source separation as a linear Bayesian inverse problem given a prior $p(\mathbf{x})$. Assuming that $\mathbf{x}_1$ and $\mathbf{x}_2$ are chosen independently,

$$p(\mathbf{x}) = p(\mathbf{x}_1, \mathbf{x}_2) = p_1(\mathbf{x}_1)p_2(\mathbf{x}_2). \tag{5.30}$$

Note that we can also apply Langevin posterior sampling to separation of dependent sources, but in this case we require a joint model over components; an example of separation of dependent sources is presented in Section 5.3.6.

We consider to variants of the source separation task: *class agnostic* separation, where mixtures are generated from two components $\mathbf{x}_1$ and $\mathbf{x}_2$ drawn from the same distribution, and and *class conditional* separation, where the components are drawn from different dis-

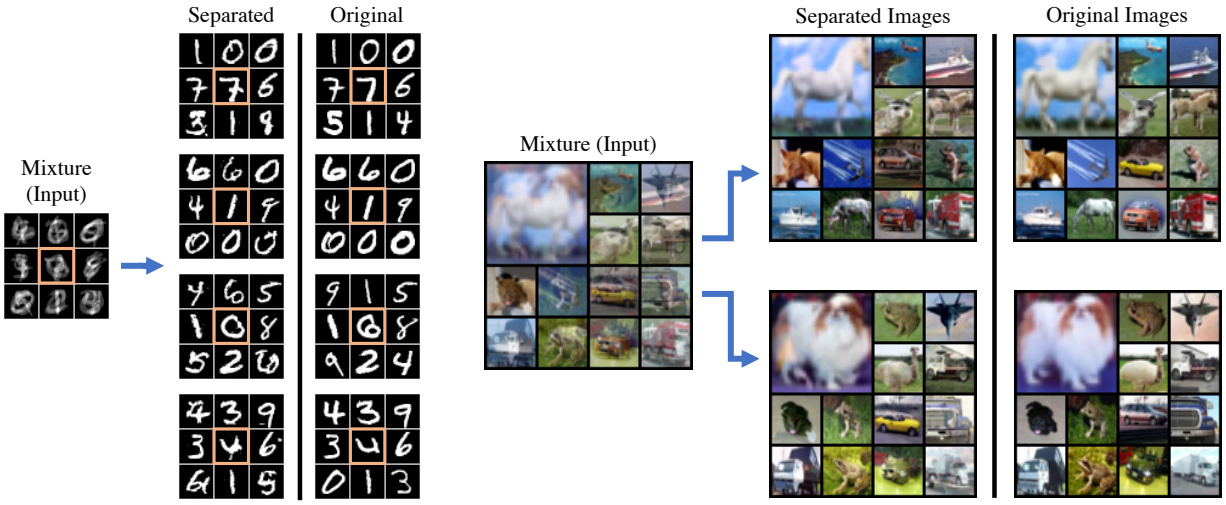


Figure 5.9: Separation results for mixtures of four images from the MNIST dataset (Left) and two images from the CIFAR-10 dataset (Right), using annealed Langevin sampling with the NCSN Song and Ermon [2019] generative model as a prior over images. We draw attention to the central panel of the MNIST results (highlighted in orange), which shows how sometimes a mixture can be plausibly separated in multiple ways.

tributions. For MNIST and CIFAR-10 experiments, we consider both class-agnostic and class-conditional separation tasks. For class-conditional separation, we partition the test sets of these datasets into two groupings: digits $0-4$ and $5-9$ for MNIST, "animals" and "machines" for CIFAR-10. For LSUN we consider class-conditional separation, using the churches and bedrooms classes. For audio, we consider class-conditional separation using the VCTK voice dataset and Supra piano dataset as classes.

Several select examples of class-agnostic visual source separation are shown in Figure 5.9. In the class-agnostic case, separation results are only defined up to a permutation of the components; to facilitate visual comparisons, visual separation results are sorted to minimize PSNR with the original images. Likewise, we evaluate PSNR for class-agnostic separation using the permutation of the components that results in minimal PSNR. This convention usually results in the separated components being visually paired with the most similar original components, but not always; the alert reader may have noticed that the yellow and silver car mixture in Figure 5.9 appears to have been displayed in reverse order. This happens because the separated yellow car component takes the light sky from the

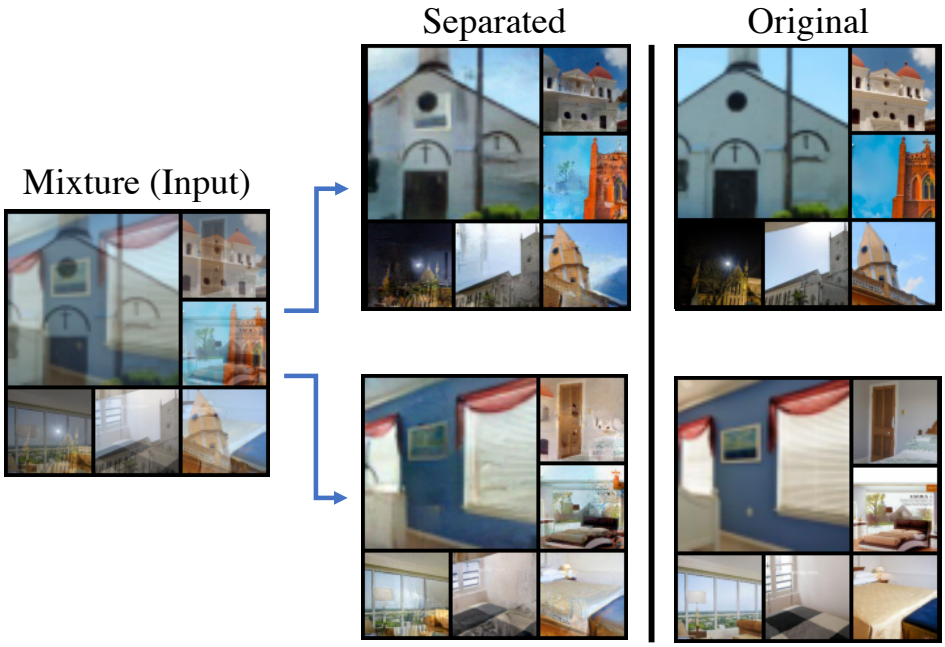


Figure 5.10: 64×64 class-conditional LSUN separation results using Glow as a prior. One mixture component is sampled from the LSUN churches category, and the other component is sampled from LSUN bedrooms.

original silver car component, and the lightness of the sky dominates the PSNR metric. For class-conditional separation, the symmetry of components is broken by the distinct priors, so no post-processing is required for visualization or evaluation. See, for example, the LSUN results in Figure 5.10, for which the priors naturally sort separated components into churches and bedrooms.

There are two different ways to apply a prior for class-conditional separation. In the class agnostic setting, $\mathbf{x}_1$ and $\mathbf{x}_2$ are drawn from the same distribution (the empirical distribution of the test set) so it makes sense to use a single prior $p = p_1 = p_2$. In the class conditional setting, we could potentially use separate priors over components $\mathbf{x}_1$ and $\mathbf{x}_2$. For the MNIST and CIFAR-10 experiments, we use pre-trained models trained on unconditional distribution of the training data for both the class agnostic and class conditional setting. It is possible that better results could be achieved in the class conditional setting by re-training the models on class conditional training data. For LSUN, the authors of Glow provide separate pre-trained models for the Church and Bedroom categories, so we are able to demonstrate class-

conditional LSUN separations using distinct priors in Figure 5.10.

Visual Separation Baselines. On the MNIST dataset, we compare to results reported for the GAN-based "S-D" method [Kong et al., 2019] and the fully supervised version of Neural Egg separation "NES" [Halperin et al., 2019]. The S-D method is a class-agnostic separation algorithm, whereas the NES method is class-conditional. To the best of our knowledge there are no previously reported quantitative metrics for CIFAR-10 separation, so as a baseline we ran Neural Egg separation on CIFAR-10 using the authors' published code. We remark that we cannot compare our algorithm to the separation-like task reported for CapsuleNets [Sabour et al., 2017]. The segmentation task discussed in that work is similar to source separation, but the mixtures used for the segmentation task are constructed using the non-linear threshold function $h(\mathbf{x}) = \max(\mathbf{x}_1 + \mathbf{x}_2, 1)$, in contrast to our linear function g. While extending the techniques of this paper to non-linear relationships between $\mathbf{x}$ and $\mathbf{m}$ is intriguing, we leave this to future this work.

We also consider results for a simple baseline, "Average," that separates a mixture $\mathbf{y}$ into two 50% masks $\mathbf{x}_1 = \mathbf{x}_2 = \mathbf{y}/2$. This is a surprisingly competitive baseline. Observe that if we had no prior information about the distribution of components, and we measure separation quality by PSNR, then by a symmetry argument setting $\mathbf{x}_1 = \mathbf{x}_2$ is the optimal separation strategy in expectation. In principle we would expect Average to perform very poorly under IS/FID, because these metrics purport to measure similarity of distributions and mixtures should have little or no support under the data distribution. But we find that IS and FID both assign reasonably good scores to Average, presumably because mixtures exhibit many features that are well supported by the data distribution. This speaks to well-known difficulties in evaluating generative models [Theis et al., 2016] and could explain the strength of "Average" as a baseline.

Visual Source Separation. Quantitative results for MNIST image separation are reported in Table 5.2, and a panel of visual separation results are presented in Figure 5.12. For quantitative results, we report mean PSNR over separations of $12,000$ separated components. The distribution of PSNR for class agnostic MNIST separation is visualized in Figure

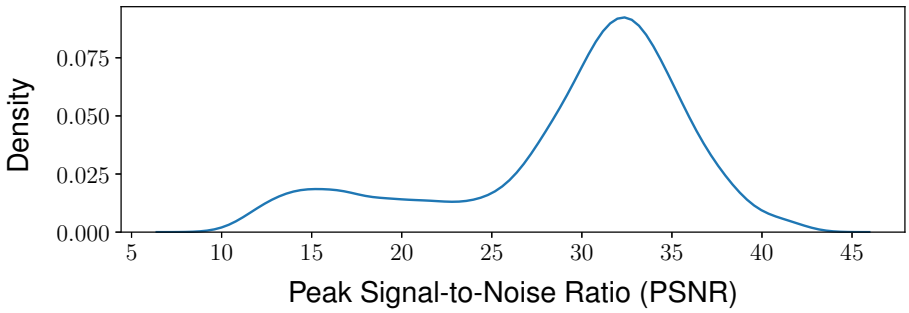


Figure 5.11: The empirical distribution of PSNR for 5,000 class agnostic MNIST digit separations using BASIS with the NCSN prior (see Table 5.2 for comparison of the central tendencies of this and other separation methods).

Algorithm	Class Split	Class Agnostic
Average	14.8	14.9
NMF	16.0	9.4
S-D	-	18.5
BASIS (Glow)	22.9	22.7
NES	24.3	-
BASIS (Glow, 10x)	27.7	27.1
BASIS (NCSN)	**29.5**	**29.3**

Table 5.2: PSNR results for separating 6,000 pairs of equally mixed MNIST images. For class split results, one image comes from label $0-4$ and the other comes from $5-9$. We compare to S-D Kong et al. [2019], NES Halperin et al. [2019], convolutional NMF (class split) Halperin et al. [2019] and standard NMF (class agnostic) Kong et al. [2019].

5.11. We observe that approximately 2/3 of results exceed the mean PSNR of 29.5, which to our eyes is visually indistinguishable from ground truth.

A natural approach to improve separation performance is to sample multiple $\mathbf{x} \sim p(\cdot|\mathbf{y})$ for a given mixture $\mathbf{m}$. A major advantage of models like Glow, that explicitly parameterize the prior $p(\mathbf{x})$, is that we can approximate the maximum of the posterior distribution with the maximum over multiple samples. By construction, Langevin posterior sampling approximately satisfies $g(\mathbf{x}) = \mathbf{y}$, so for the noiseless model $p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{x})$. We demonstrate the effectiveness of resampling in Table 5.2 (Glow, 10x) by comparing the expected PSNR of $\mathbf{x} \sim p(\cdot|\mathbf{y})$ to the expected PSNR of $\arg\max_i p(\mathbf{x}_i)$ over 10 samples $\mathbf{x}_1, \ldots, \mathbf{x}_{10} \sim p(\cdot|\mathbf{y})$. Even moderate resampling dramatically improves separation performance. Unfortunately

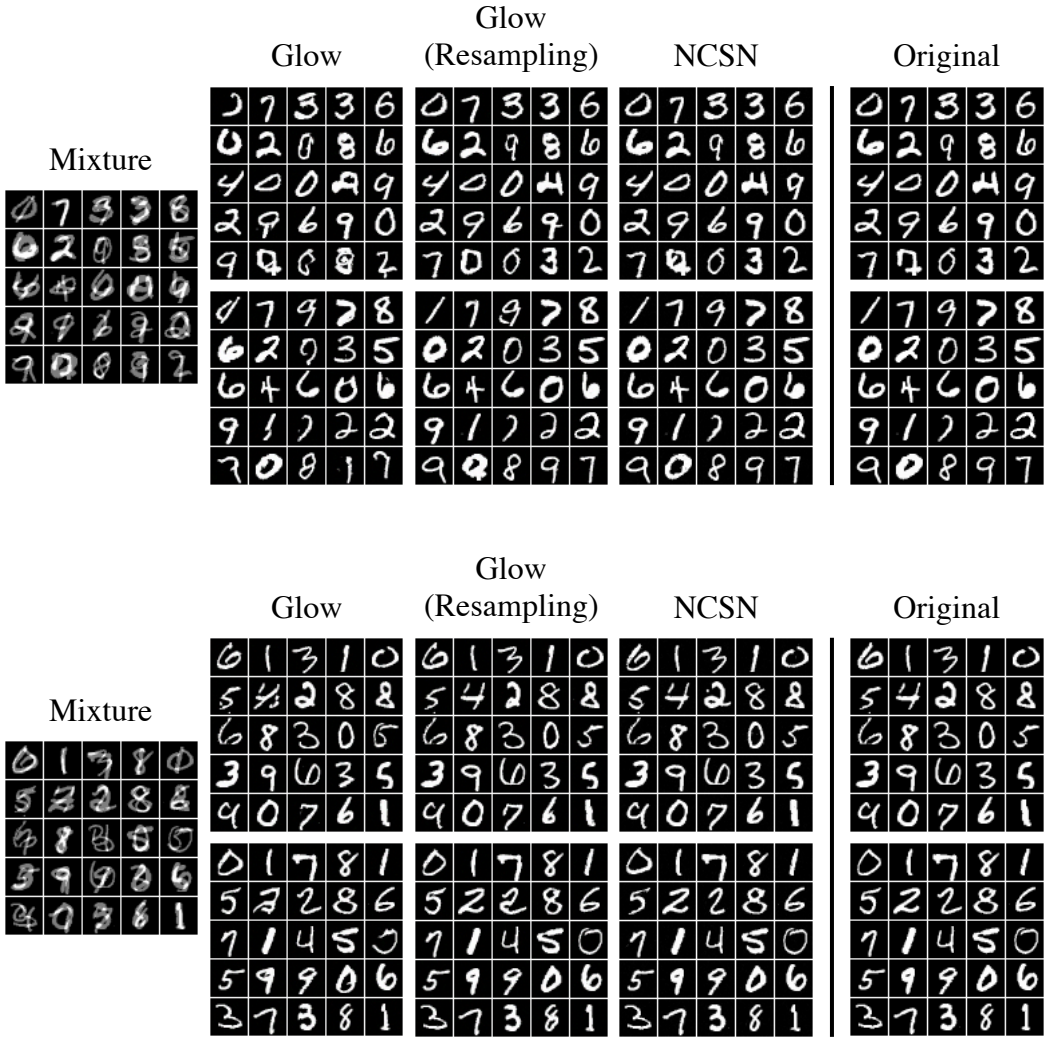


Figure 5.12: Uncurated class-agnostic separation results using: (1) samples from the posterior with Glow as a prior (2) an approximate MAP estimate using the maximum over 10 samples from the posterior with Glow as a prior (3) samples from the posterior with NCSN as a prior.

this approach cannot be applied to the otherwise superior NCSN model, which does not model explicit likelihoods $p(\mathbf{x})$.

Without any modification, we can apply Langevin sampling can separate mixtures of $k > 2$ component images. We contrast this with regression-based methods, which require re-training to target varying numbers of components. Figure 5.9 shows the results of Langevin posterior sampling using the NCSN prior applied to mixtures of four randomly selected MNIST images. For more mixture components, we observe that identifiability of ground

Algorithm	Inception Score	FID
Class Split		
NES	5.29 ± 0.08	51.39
Langevin (Glow)	5.74 ± 0.05	40.21
Langevin (PixelCNN++)	5.86 ± 0.07	40.66
Average	6.14 ± 0.11	39.49
Langevin (NCSN)	7.83 ± 0.15	29.92
Class Agnostic		
Langevin (Glow)	6.10 ± 0.07	37.09
Langevin (PixelCNN++)	6.14 ± 0.15	37.89
Average	7.18 ± 0.08	28.02
Langevin (NCSN)	8.29 ± 0.16	22.12

Table 5.3: Quantitative results for visual sources separation on CIFAR-10. Results are measured using Inception Score / FID Score of 25,000 separations (50,000 separated images) of two overlapping CIFAR-10 images. In Class Split one image comes from the category of animals and other from the category of machines. The NES baseline results are computed using the procedure described by Halperin et al. [2019].

truth sources begins to break down. This is illustrated by looking at the central item in each panel of Figure 5.9 (highlighted in orange).

Quantitative results for CIFAR-10 separation are tabulated in Table 5.3. Of the three priors used for Langevin posterior separation, we see that Glow and PixelCNN++ perform comparably (with PixelCNN++ performing slightly better) with NCSN performing substantially better. This is consistent with the unconditional performance of these generative models. Given the strong correlation between the strength of a generative model and the quality of separations using that model as a prior, we anticipate that more recent innovations in autoregressive image models based on transformers [Parmar et al., 2018, Child et al., 2019] will lead to stronger separation results once implementations of these models that match the results reported in these papers become public.

Qualitative results for LSUN separation are visualized in Figure 5.10. While the separation results in Figure 5.10 are imperfect, Table 5.1 shows that the mean log-likelihood of the separated components is comparable to the mean log-likelihood that the model assigns to images in the test set. This suggests that the model is incapable of distinguishing these

Algorithm	Test SI-SDR (dB)		
	All	Piano	Voice
Langevin (WaveNet)	17.07	13.92	20.25
Conv-Tasnet	17.48	20.02	15.50
Demucs	14.18	16.67	12.75

Table 5.4: Quantitative results for audio source separation of mixtures of Supra piano and VCTK voice samples. Results are measured using SI-SDR (higher is better).

separations from better results, and the imperfections are attributable to the quality of the model rather than to the separation algorithm. This is encouraging, because it suggests that the artifacts are due to the Glow model rather than the Langevin separation algorithm, and that better separation results will be achievable with improved generative models.

Audio Separation. We compare Langevin audio separation to results using the Demucs [Défossez et al., 2019] and Conv-Tasnet [Luo and Mesgarani, 2019] source separation models. Both Demucs and Conv-Tasnet are supervised models, trained specifically for the source separation task, that learn to output source components given an input mixture. An advantage of Langevin sampling is that it does not rely on pairs of source signals and mixes like these supervised methods. We train the supervised models to separate mixtures of VCTK and Supra Piano samples and measure results using the standard Scale Invariant Signal-to-Distortion Ratio (SI-SDR) metric for audio source separation [Roux et al., 2019]. All mixtures are created with 1/2 gain on each source component; due to the natural variation of loudness of training data, we find that our model generalizes to mixtures without exactly 1/2 gain on each source; source separation result on the website demonstrate that we can separate real-world mixtures, even when we have no information about the relative loudness of each component. Results in Table 5.4 show that Langevin sampling is competitive with these specialized source separation models. Qualitative comparisons are provided in the supplement. We do not compare results on the popular MusDB dataset [Rafii et al., 2017] because this dataset has insufficient single-channel audio to train WaveNet generative models.

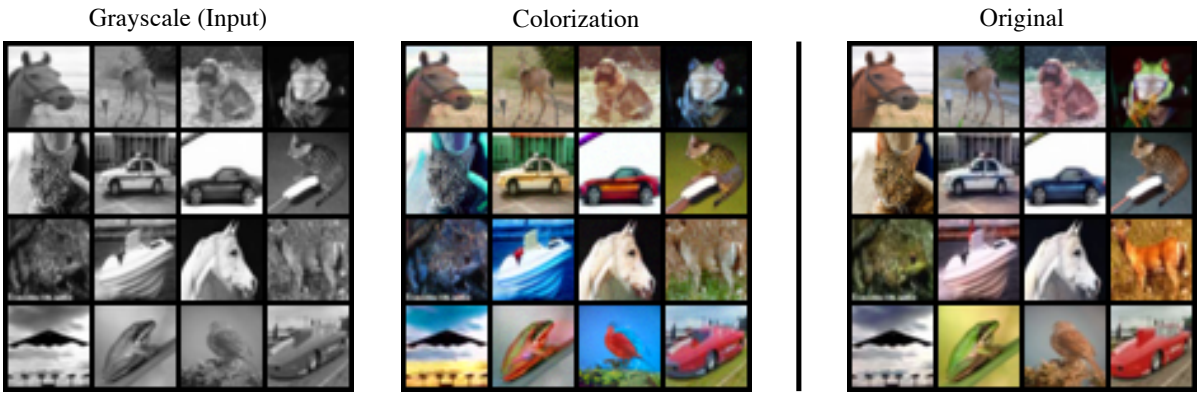


Figure 5.13: Colorizing CIFAR-10 images. Left: original CIFAR-10 images. Middle: greyscale conversions of the images on the left. Right: imputed colors for the greyscale images, found by BASIS using NCSN as a prior.

5.3.6 Image Colorization

We can also view image colorization Levin et al. [2004], Zhang et al. [2016] as a source separation problem by interpreting a grayscale image as a mixture of the three color channels of an image $\mathbf{x} = (\mathbf{x}_r, \mathbf{x}_g, \mathbf{x}_b)$ with

$$g(\mathbf{x}) = (\mathbf{x}_r + \mathbf{x}_g + \mathbf{x}_b)/3. \tag{5.31}$$

Unlike our previous separation problems, the channels of an image are clearly not independent, and the factorization of p given by Equation 5.30 is unwarranted. But conveniently, a generative model trained on color CIFAR-10 images itself models the joint distribution $p(\mathbf{x}) = p(\mathbf{x}_r, \mathbf{x}_g, \mathbf{x}_b)$. Therefore, the same pre-trained generative model that we use to separate images can also be used to color them.

Qualitative colorization results are visualized in Figure 5.13. The non-identifiability of ground truth is profound for this task (see Section 5.2.2 for discussion of identifiability). We draw attention to the two cars in the middle of the panel: the white car that is colored yellow by the algorithm, and the blue car that is colored red. The colors of these specific cars cannot be inferred from a greyscale image; the best an algorithm can do is to choose a reasonable color, based on prior information about the colors of cars.

Data Distribution	Inception Score	FID Score
Input Grayscale	8.01 ± 0.10	68.52
BASIS (Glow)	8.69 ± 0.15	28.70
BASIS (NCSN)	**10.53 ± 0.17**	**11.58**
CIFAR-10 Test Set	11.24 ± 0.12	0.00

Table 5.5: Inception Score / FID Score of 50,000 colorized CIFAR-10 images. As measured by IS/FID, the quality of NCSN colorizations nearly matches CIFAR-10 itself.

Quantitative coloring results for CIFAR-10 are presented in Table 5.5. We remark that the IS and FID scores for coloring are substantially better than the IS and FID scores of 8.87 and 25.32 respectively reported for unconditional samples from the NCSN model; conditioning on a greyscale image is enormously informative. Indeed, the Inception Score of NCSN-colorized CIFAR-10 is close to the Inception Score of the strongest possible baseline: the CIFAR-10 test set.

5.3.7 Super-Resolution

The super-resolution problem asks us to recover unobserved data $\mathbf{x}$ given a down-sampled observation $\mathbf{y} = g(\mathbf{x})$. For 1-dimensional (audio) super-resolution, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^{n/r}$, and $\mathbf{y}_i = \mathbf{x}_{ri}$ [Kuleshov et al., 2017, Eskimez et al., 2019]. For 2-dimensional (image) super-resolution $\mathbf{x} \in \mathbb{R}^n = \mathbb{R}^{k \times k \times 3}$, $\mathbf{y} \in \mathbb{R}^{k/r \times k/r \times 3}$, and $\mathbf{y}_{i,j} = \mathbf{x}_{ri,rj}$ [Dahl et al., 2017, Zhang et al., 2018b]. Like source separation, super-resolution can be viewed as a Bayesian linear inverse problem, and we can recover solutions to this problem via Langevin sampling. In the audio domain, the down-sampling operation $g(\mathbf{x})$ can be interpreted as a low-pass filter.

We measure audio super-resolution performance using peak signal-to-noise ratio (PSNR) and compare against a deep learning baseline [Kuleshov et al., 2017] as well as a simple cubic B-spline. Quantitative audio results are presented in Table 5.6, which show that we outperform these baselines on piano data and produce similar quality reconstructions on voice data. Qualitative audio samples are available in the supplement, where we also show examples of 32x super resolution - beyond the reported ability of existing methods. Select qualitative visual results for image super-resolution using a PixelCNN++ model are

	Piano			Voice		
Ratio	Spline	KEE	Langevin	Spline	KEE	Langevin
4x	23.07	22.25	29.78	15.8	16.04	15.47
8x	13.58	15.79	23.49	10.7	11.15	10.03
16x	7.09	6.76	14.23	6.4	7.11	5.32

Table 5.6: Quantitative results for audio super-resolution at three different scales on the Supra piano and VCTK voice datasets. Results are measured using PSNR (higher is better). KEE refers to the method described by Kuleshov et al. [2017]

presented in Figure 5.6.

5.3.8 Inpainting

Inpainting problems involve recovering unobserved data $\mathbf{x}$ given a masked observation $g(\mathbf{x}) = \mathbf{m} \odot \mathbf{x}$, where $\mathbf{m} \in \{0, 1\}^n$ [Adler et al., 2012, Pathak et al., 2016]. This family of problems includes completion tasks (finishing a sequence given a prime) pre-completion tasks (generating a prefix to a sequence) and outpainting tasks. Ancestral sampling can only be applied to completion tasks, whereas Langevin sampling can be used to fill in any pattern of masked occlusions. Qualitative results for audio inpainting are available in the supplement. Select qualitative results for image inpainting using a PixelCNN++ model are presented in Figure 5.6.

5.4 Conclusion

In Section 5.1 of this chapter, we proposed a posterior sampler (with a novel smoothing procedure for discretized autoregressive models) that we applied to linear likelihood models $p(\mathbf{y}|\mathbf{x})$ with a deep generative prior $p(\mathbf{x})$ in Sections 5.2 and 5.3. Nothing about the methods proposed in Section 5.1 requires a linear relationship between $\mathbf{x}$ and $\mathbf{y}$. If the likelihood $p(\mathbf{y}|\mathbf{x})$ is instead given by a deep generative model, then we simply replace analytic calculations of the gradients $\nabla_x \log p_\sigma(\mathbf{y}|\tilde{\mathbf{x}})$ with gradients calculated using automatic differentiation. The smoothed likelihoods $p_\sigma(\mathbf{y}|\tilde{\mathbf{x}})$ described by Equation (5.9) can themselves be approximated using the same fine-tuning procedure used to smooth $p_\sigma(\tilde{\mathbf{x}})$, as described in Section 5.1.2.

This raises the possibility of using a family of likelihoods $p(\mathbf{y}_i|\mathbf{x})$ to control the outputs of a generative model $p(\mathbf{x})$, ranging from the fine-grained control of a mixture constraint (which imposes strong conditions on the value of every sample in $\mathbf{x}$), to more abstract and high-level control induced by a classifier. For example, we can imagine using the composer attribution classifier developed in Chapter 4, Section 4.3 to steer the outputs of the unconditional music composition model developed in Chapter 4, Section 4.2 to generate compositions in the style of a particular composer. Posterior sampling with a likelihood composed of multiple, independent classifiers and constraints could provide us with a set of high-level knobs for steering the output of a generative model towards a desired artistic vision.

Finally, we observe that while the *visual* source separation results presented in this chapter are state-of-the-art, the *audio* are merely competitive with state-of-the-art models. In Section 5.3.5 (in particular, Table 5.3) we observed that, for visual separation, quality of our algorithm varied substantially with the strength of the generative model $p(\mathbf{x})$ used as a prior. In contrast, in the audio domain we demonstrate results for only a single model, WaveNet, which quite old by the standards of deep generative models. Based on the trend for visual source separation, we anticipate that results for audio source separation using the posterior sampling approach will dramatically improve as stronger generative audio models become available.

Chapter 6

CONCLUSION AND PERSPECTIVES

In the preceding chapters, we studied three problems in music and signal processing: music transcription (Chapter 3), music composition (Chapter 4), and source separation (Chapter 5). In Chapter 1, we motivated these problems by their widespread recognition as foundational questions in the music information retrieval field. But we should also consider *why* these problems are considered foundational, and how the methods developed in this dissertation (or more mature future variants of these methods) could find their way into user-facing tools for communities of musicians, audio engineers, composers, and music educators.

When we consider applications of models in practical tools, we should make a distinction between *analytical* tools that provide some form of information to a user, and *synthetic* tools that assist a user in a creative process. This dissertation has largely focused on generative models, which are most naturally applied to the synthesis problems faced by composers, producers, and audio engineers. Musicians and music educators may be more interested in analytical tools that could provide feedback about a musical performance, or explain the structure of a musical composition. But this broad characterization of interests is not sharp: a solo musician, for example, might be interested in a generative model that synthesizes an automatic real-time accompaniment to their musical performance [Kaliakatsos-Papakostas et al., 2012]. Conversely, audio engineering tools increasingly incorporate analytical models to facilitate music production: for example, Apple's Logic Pro software incorporates an analytical beat tracking model [Ellis, 2007], which can be used to synchronize multiple stems of audio to a common tempo (see Logic Pro's Smart Tempo technology).

The music transcription models developed in Chapter 3 blur the distinction between tools for analysis and synthesis. Music transcription has already been commercialized in

audio engineering software, including Celemony's Melodyne product, which we evaluated in Section 3.3.6. Like beat tracking annotations, a transcript can provide insights into the content of an audio recording for a music producer, which they can then act on (perhaps using another generative model) to edit the recording. Looking to the future, accurate transcription software could become a valuable tool for music education. While interaction with an expert educator is essential to musical education, contact with an expert is intermittent: perhaps an hour a week, or less for economically disadvantaged students. Automated tools that transcribe and analyze a musical performance could offer student musicians a tight feedback loop in their practice.

In Chapter 4 we developed both generative models that synthesize musical scores (Section 4.2), and discriminative models for analyzing scores (Section 4.3). As we discussed in Chapter 1, there is little artistic value in the outputs of an unconditional generative model. But there is substantial community interest in models that can be steered by user input. A conditionally-generated composition is a human artistic creation, where the artistic process simply happens at a higher level of abstraction: rather than specifying individual notes, the artist specifies constraints and attributes of the desired composition. Simple generative models, for example arpeggiators [Robinson and Howell, 1983], are already popular building-blocks of music creation found in popular production tools including Logic Pro and Ableton Live. More sophisticated conditional generative models, that can generate more sophisticated and varied motifs, will surely be integrated into future music composition workflows.

The method developed in Chapter 5 for guiding and controlling the outputs of a generative model has numerous potential applications to music synthesis (in conjunction with an unconditional generative model, e.g., the model developed in Chapter 4). Concretely, we showed how this method can be used to synthesize audio subject to a mixture constraint, leading to an algorithm for source separation. Source separation has diverse applications in a range of fields including wireless communications [Madhow, 1997], astrophysics [Funaro et al., 2003], and seismology [Zhou et al., 2016]. In the music domain, we can imagine future applications of source separation as a production tool: high-quality source separation

tools would allow engineers and producers to decompose (de-mix) an audio recording into a collection of sources (stems), process these stems in isolation, remove undesired stems (e.g., an errant cough), and re-mix the stems to create an updated audio recording.

Thinking more broadly about a future with robust solutions to the problems addressed in this dissertation, we can imagine a 21st century audio production environment powered by data-driven models. Current audio processing techniques are largely powered by a digital signal processing toolbox. While these techniques can be robust and powerful, they are also low-level metaphors for engaging with music. Beyond an arpeggiator, we can ask for a model that can help us find interesting chord progressions for our melody [Huang et al., 2016]. Beyond autotune, we can ask for a model that re-styles our voice with a particular vocal quality. These desires can only be met with data-driven generative models trained on large collections of music that capture the range and diversity of human expression.

The largest obstacle to this vision may be sociological, rather than technical: audio recordings are subject to copyright. The legal status of creations generated by models trained on copyrighted data remains uncertain. Are these creations derivative works? Are the authors of the recordings used for training models entitled to royalties? Generative models are known to memorize training data [Carlini et al., 2019] and a generative model's creations may unintentionally plagiarize copyrighted material (to be clear, human composers are also susceptible to this problem). For the experiments described in this dissertation, we largely avoided copyright concerns: the MusicNet dataset introduced in Chapter 3 was deliberately constructed using permissively licensed recordings, which release rights to derivative works and redistribution. The classical scores considered in Chapter 4 are hundreds of years old and have entered the public domain. The WaveNet models in Chapter 5 were trained using small-scale academic datasets. But future high-quality, production iterations of these models will require large-scale datasets. Use of copyrighted material to train these models may become unavoidable and legal considerations must be taken into account. Idealistically, I hope our legal frameworks evolve in a way that respects the rights of artists and protects their livelihood, while also fostering a supportive environment for data-driven creative tools.

BIBLIOGRAPHY

Amir Adler, Valentin Emiya, Maria G. Jafari, Michael Elad, Rémi Gribonval, and Mark D. Plumbley. Audio inpainting. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 20(3):922–932, 2012. 5.3.8

Moray Allan and Christopher K. I. Williams. Harmonising chorales by probabilistic inference. In *Proceedings of the 17th Advances in Neural Information Processing Systems (Neurips)*, pages 25–32, 2004. 4.2.1

Yoko Anan, Kohei Hatano, Hideo Bannai, Masayuki Takeda, and Ken Satoh. Polyphonic music classification on symbolic data using dissimilarity functions. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 229–234, 2012. 4.3.1

Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, 2017. 2.5

Andreas Arzt and Stefan Lattner. Audio-to-score alignment using transposition-invariant features. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 592–599, 2018. 3.1.2, 3.1.3

Mert Bay, Andreas F. Ehmann, and J. Stephen Downie. Evaluation of multiple-f0 estimation and tracking systems. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 315–320, 2009. 3

Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995. 5.2.1

Pierre C Bellec, Guillaume Lecué, Alexandre B Tsybakov, et al. Slope meets lasso: improved oracle bounds and optimality. *The Annals of Statistics*, 46(6B):3603–3642, 2018. 5.2.2

Laurent Benaroya, Frédéric Bimbot, and Rémi Gribonval. Audio source separation with a single sensor. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 14 (1):191–199, 2006. 1, 5.2.1

Emmanouil Benetos and Simon Dixon. A shift-invariant latent variable model for automatic music transcription. *Computer Music Journal*, 36(4):81–94, 2012. 3.3.6

Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: Breaking the glass ceiling. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 379–384, 2012. 1

Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 2013. 1, 3.3.1

Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30, 2019. 1

Yoshua Bengio and Samy Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. In *Proceedings of the 12th Advances in Neural Information Processing Systems (Neurips)*, pages 400–406, 1999. 2.3

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research (JMLR)*, 3:1137–1155, 2003. 2.3

Gregory W. Benton, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. Learning invariances in neural networks from training data. In *Proceedings of the 33rd Advances in Neural Information Processing Systems (Neurips)*, 2020. 3.3.5

Rachel M. Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan Pablo Bello. Deep salience representations for F0 estimation in polyphonic music. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 63–70, 2017. 3.3.1

Harold S Black and JO Edson. Pulse code modulation. *Transactions of the American Institute of Electrical Engineers*, 66(1):895–899, 1947. 2.1

Andrew Blake and Andrew Zisserman. *Visual Reconstruction*. MIT Press, 1987. 5.1.1, 5.1.2

Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G. Dimakis. Compressed sensing using generative models. In *34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 537–546, 2017. 5.2

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *29th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, 2012. (document), 4.4, 4.5, 4.2.1, 4.2.3

G. E. P. Box and Mervin E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29:610–611, 1958. 2.2

Andrew Brinkman, Daniel Shanahan, and Craig Sapp. Musical stylometry, machine learning and attribution studies: A semi-supervised approach to the works of josquin. In *Proceedings of the Biennial International Conference on Music Perception and Cognition*, pages 91–97, 2016. (document), 4.3, 4.3.1, 4.3.4, 4.3.5, 4.9

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner,

Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 33rd Advances in Neural Information Processing Systems (Neurips)*, 2020. 4.2.2, 4.2.5, 5.1

Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *4th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2016. 2.4, 2.4

Giuseppe Buzzanca. A supervised learning approach to musical style recognition. In *2nd International Conference on Music and Artificial Intelligence*, page 167, 2002. 4.3, 4.3.1, 4.3.2

Julio José Carabias-Orti, Francisco J. Rodríguez-Serrano, Pedro Vera-Candeas, Nicolás Ruiz-Reyes, and Francisco J. Cañadas-Quesada. An audio to score alignment framework using spectral factorization and dynamic time warping. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 742–748, 2015. 3.1.2

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium*, pages 267–284. USENIX Association, 2019. 6

CCITT. Pulse code modulation (pcm) of voice frequencies. International Telecommunication Union, 1988. 5.3.1

Celemony. Melodyne. http://www.celemony.com/en/melodyne/what-is-melodyne. 3.3.6, 3.3.5

Jen-Hao Rick Chang, Chun-Liang Li, Barnabás Póczos, and B. V. K. Vijaya Kumar. One network to solve them all - solving linear inverse problems using deep projection models. In *IEEE International Conference on Computer Vision (ICCV)*, pages 5889–5898, 2017. 5.2

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J. Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. In *9th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2021. 2.7, 5.3.2

E Colin Cherry. Some experiments on the recognition of speech, with one and with two ears. *The Journal of the Acoustical Society of America*, 25(5):975–979, 1953. 1

Kin Wai Cheuk, Kat Agres, and Dorien Herremans. The impact of audio input representations on neural network based music transcription. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2020a. (document), 3.11

Kin Wai Cheuk, Yin-Jyun Luo, Emmanouil Benetos, and Dorien Herremans. The effect of spectrogram reconstruction on automatic music transcription: An alternative approach to improve transcription accuracy. In *Proceedings of the 25th International Conference on Pattern Recognition (ICPR)*, pages 9091–9098. IEEE, 2020b. 3.3.1

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *OpenAI blog*, 2019. 5.3.5

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Proceedings of the 28th Advances in Neural Information Processing Systems (Neurips)*, pages 2980–2988, 2015. 2.4

Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A. Smith. Creative writing with a machine in the loop: Case studies on slogans and stories. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces (IUI)*, pages 329–340, 2018. 1

Pierre Comon. Independent component analysis, A new concept? *Signal processing*, 36(3): 287–314, 1994. 5.2.1

Darrell Conklin. Music generation from statistical models. In *Proceedings of the AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 30–35, 2003. 1, 4.2

Arshia Cont, Diemo Schwarz, Norbert Schnell, and Christopher Raphael. Evaluation of real-time audio-to-score alignment. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, pages 315–316, 2007. 3.1.2, 3.1.2

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2001. 2.2

Julie E Cumming. Motet & cantilena. In *A Performers Guide to Medieval Music*. Indiana University Press, 2000. 4.3.2

Ryan Dahl, Mohammad Norouzi, and Jonathon Shlens. Pixel recursive super resolution. In *IEEE International Conference on Computer Vision (ICCV)*, pages 5449–5458, 2017. 5.3.7

Roger B. Dannenberg and Ning Hu. Polyphonic audio matching for score following and intelligent audio editors. In *Proceedings of the 2003 International Computer Music Conference (ICMC)*. Michigan Publishing, 2003. 3.2.2

Mike E Davies and Christopher J James. Source separation using single channel ica. *Signal Processing*, 87(8):1819–1832, 2007. 5.2, 5.2.1

Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. 5.1.4

Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis R. Bach. Music source separation in the waveform domain. *arXiv Preprint arXiv:1911.13254*, 2019. 5.2.1, 5.3.5

Johanna Devaney. Estimating onset and offset asynchronies in polyphonic score-audio alignment. *Journal of New Music Research*, 43(3):266–275, 2014. 3.1.2

Johanna Devaney and Daniel P. W. Ellis. Handling asynchrony in audio-score alignment. In *Proceedings of the 2009 International Computer Music Conference (ICMC)*. Michigan Publishing, 2009. 3.1.2

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv Preprint arXiv:2005.00341*, 2020. 2.3, 2.4, 4.2.2, 5, 5.1

Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968. IEEE, 2014. 3.3, 3.3.4

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *International Conference on Learning Representations Workshop*, 2015. 2.6

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2017. 2.2, 2.6, 2.6, 5.1

Simon Dixon. An on-line time warping algorithm for tracking musical performances. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1727–1728. Professional Book Center, 2005. 3.1

Chris Donahue, Julian J. McAuley, and Miller S. Puckette. Adversarial audio synthesis. In *7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019a. 2.5, 5.3.2

Chris Donahue, Ian Simon, and Sander Dieleman. Piano genie. In *Proceedings of the 24th International Conference on Intelligent User Interfaces (IUI)*, pages 160–164, 2019b. 1

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Proceedings of the 32nd Advances in Neural Information Processing Systems (Neurips)*, pages 3603–3613, 2019. 2.7, 5.1.1

Zhiyao Duan, Bryan Pardo, and Changshui Zhang. Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 18(8):2121–2133, 2010. 3.1.3

Alain Durmus and Eric Moulines. Nonasymptotic convergence analysis for the unadjusted langevin algorithm. *The Annals of Applied Probability*, 27(3):1551–1587, 2017. 2.7

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 199–209. The Association for Computational Linguistics, 2016. 4.1.3

Kemal Ebcioğlu. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12(3):43–51, 1988. 4.2.1

Douglas Eck and Jürgen Schmidhuber. Finding temporal structure in music: blues improvisation with LSTM recurrent networks. In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing (NNSP)*, pages 747–756. IEEE, 2002. 4.2.1

Michael Elad. *Sparse and Redundant Representations - From Theory to Applications in Signal and Image Processing*. Springer, 2010. 5.2

Daniel PW Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007. 6

Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010. 3.1.3, 3.2.1, 3.3, 4.2.2

Jesse H. Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. In *7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019. 2.5

Hakan Erdogan and Emad M. Grais. Semi-blind speech-music separation using sparsity and continuity priors. In *20th International Conference on Pattern Recognition (ICPR)*, pages 4573–4576. IEEE Computer Society, 2010. 5.2.1

Sefik Emre Eskimez, Kazuhito Koishida, and Zhiyao Duan. Adversarial training for speech super-resolution. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):347–358, 2019. 5.3.7

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12873–12883, 2021. 4.2.2

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 3.3.6

Sebastian Ewert and Meinard Müller. Refinement strategies for music synchronization. In *Proceedings of the 5th International Symposium on Computer Music Modeling and Retrieval (CMMR)*, volume 5493, pages 147–165. Springer, 2008. 3.1.2, 3.1.2

Sebastian Ewert, Meinard Müller, and Peter Grosche. High resolution audio synchronization using chroma onset features. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1869–1872. IEEE, 2009. 2, 3.1.1, 3.1.2, 3.1.2

Sebastian Ewert, Meinard Müller, Verena Konz, Daniel Müllensiefen, and Geraint A. Wiggins. Towards cross-version harmonic analysis of music. *IEEE Transactions on Multimedia*, 14(3-2):770–782, 2012. 3.1.3

Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2013. 3.1.1

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 70(1):41–54, 2006. 5.1.2

Francesco Foscarin, Andrew McLeod, Philippe Rigaux, Florent Jacquemard, and Masahiko Sakai. ASAP: a dataset of aligned scores and performances for piano transcription. In *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, pages 534–541, 2020. (document), 3.2.3, 3.4, 3.5

Maurice Frank and Maximilian Ilse. Problems using deep generative models for probabilistic audio source separation. In *"I Can't Believe It's Not Better!" Workshop at Advances in Neural Information Processing Systems*, 2020. 5.1

Judy A. Franklin. Multi-phase learning for jazz improvisation and interaction. *Proceedings of the Eighth Biennial Symposium for Arts and Technology*, pages 51–60, 2001. 4.2.1

Christian Fremerey, Meinard Müller, and Michael Clausen. Handling repeats and jumps in score-performance synchronization. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, pages 243–248. International Society for Music Information Retrieval, 2010. 3.2.4

Maria Funaro, Erkki Oja, and Harri Valpola. Independent component analysis for artefact separation in astrophysical images. *Neural Networks*, 16(3-4):469–478, 2003. 6

Yossi Gandelsman, Assaf Shocher, and Michal Irani. "double-dip": Unsupervised image decomposition via coupled deep-image-priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11026–11035, 2019. 5.2.1

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Proceedings of the 28th Advances in Neural Information Processing Systems (Neurips)*, pages 262–270, 2015. 5.2.3

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016. 5.2.3

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984. 5.2.1

Robert Gens and Pedro M. Domingos. Deep symmetry networks. In *Proceedings of the 27th Advances in Neural Information Processing Systems (Neurips)*, pages 2537–2545, 2014. 3.3.5

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: masked autoencoder for distribution estimation. In *32nd International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pages 881–889, 2015. 2.3, 4.2.4

Zoubin Ghahramani and Michael I. Jordan. Factorial hidden markov models. In *Proceedings of the 8th Advances in Neural Information Processing Systems (Neurips)*, pages 472–478, 1995. 4.2.4

John R. Gilbert, Cleve Moler, and Robert Schreiber. Sparse matrices in MATLAB: design and implementation. *SIAM Journal on Matrix Analysis and Applications*, 13(1):333–356, 1992. 4.2.3

Michael Good. Musicxml for notation and analysis. In *The Virtual Score: Representation, Retrieval, Restoration*. MIT Press, 2001. 4.1.3

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th Advances in Neural Information Processing Systems (Neurips)*, pages 2672–2680, 2014. 2.2, 2.5

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations (ICLR)*. OpenReview.net, 2015. 5.2.3

Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Music genre database and musical instrument sound database. In *Proceedings of the 4th International Society for Music Information Retrieval Conference (ISMIR)*, 2003. 3.2.1, 3.3

Ulf Grenander. Tutorial in pattern theory. *Report, Division of Applied Mathematics*, 1983. 2.7

Ulf Grenander and Michael I Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(4):549–581, 1994. 2.7

Gaëtan Hadjeres, François Pachet, and Frank Nielsen. Deepbach: a steerable model for bach chorales generation. In *34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1362–1371, 2017. 4.2.1, 4.2.4, 4.2.4, 4.2.5, 5.1.1

Tavi Halperin, Ariel Ephrat, and Yedid Hoshen. Neural separation of observed and unobserved distributions. In *36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 2566–2575, 2019. (document), 5.2.1, 5.2.2, 5.3.5, 5.2, 5.3

Paul Hand and Vladislav Voroninski. Global guarantees for enforcing deep generative priors by empirical risk. *IEEE Transactions on Information Theory*, 66(1):401–418, 2020. 2.2

Tatsunori B. Hashimoto, Hugh Zhang, and Percy Liang. Unifying human and statistical evaluation for natural language generation. In *Proceedings of the 2019 Conference of*

the North American Chapter of the Association for Computational Linguistics (NAACL), pages 1689–1701. Association for Computational Linguistics, 2019. 4.2.5

Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse H. Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019. 3.1.3, 3.1.4, 3.2.1, 3.3, 3.3.1, 4.2.2

Simon Haykin and Zhe Chen. The cocktail party problem. *Neural Computation*, 17(9): 1875–1902, 2005. 1

W Heisenberg. Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. *Zeitschrift fur Physik*, 43(3-4):172–198, 1927. 3.3.3

William Herlands, Ricky Der, Yoel Greenberg, and Simon A. Levin. A machine learning approach to musically meaningful homogeneous style classification. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 276–282. AAAI Press, 2014. 4.3, 4.3.1, 4.3.5

Dorien Herremans, David Martens, and Kenneth Sörensen. Composer classification models for music-theory building. In *Computational Music Analysis*, pages 369–392. Springer, 2016. (document), 4.3.1, 4.3.4, 4.3.5, 4.3.5, 4.10

Dorien Herremans, Ching-Hua Chuan, and Elaine Chew. A functional taxonomy of music generation systems. *ACM Computing Surveys*, 50(5):69:1–69:30, 2017. 1, 4.2

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 30th Advances in Neural Information Processing Systems (Neurips)*, pages 6626–6637, 2017. 4.2.5, 5.2.2

Ruben Hillewaere, Bernard Manderick, and Darrell Conklin. Global feature versus event models for folk song classification. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 729–734. International Society for Music Information Retrieval, 2009. 4.3.1

Ruben Hillewaere, Bernard Manderick, and Darrell Conklin. String quartet classification with monophonic models. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, pages 537–542. International Society for Music Information Retrieval, 2010. 4.3, 4.3.1, 4.3.4, 4.3.5

Geoffrey Hinton, Simon Osindero, Max Welling, and Yee-Whye Teh. Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive science*, 2006. 2.7

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 2002. 2.2, 2.7, 2.7

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2020. 5.2, 5.3.3

María Hontanilla, Carlos Pérez-Sancho, and José Manuel Iñesta Quereda. Modeling musical style with language models for composer recognition. In *Proceedings of the 6th Iberian Conference on Pattern Recognition and Image Analysis*, volume 7887 of *Lecture Notes in Computer Science*, pages 740–748. Springer, 2013. 4.3.1, 4.3.4

Ning Hu, Roger B Dannenberg, and George Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 185–188, 2003. 3.1.3, 2

Cheng-Zhi Anna Huang, David Duvenaud, and Krzysztof Z. Gajos. Chordripple: Recommending chords to help novice composers go beyond the ordinary. In *Proceedings of the*

21st International Conference on Intelligent User Interfaces (IUI), pages 241–250. ACM, 2016. 6

Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron C. Courville, and Douglas Eck. Counterpoint by convolution. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 211–218, 2017. 4.2.1, 4.2.5

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer: Generating music with long-term structure. In *7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019. 2.3, 4.2, 4.2.3, 4.2.4

Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Singing-voice separation from monaural recordings using deep recurrent neural networks. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, pages 477–482, 2014. 5.2.1

Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Joint optimization of masks and deep recurrent neural networks for monaural source separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(12):2136–2147, 2015. 5.2.1

Eric J. Humphrey, Juan Pablo Bello, and Yann LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 403–408, 2012. 3.2

Yun-Ning Hung, Yi-An Chen, and Yi-Hsuan Yang. Multitask learning for frame-level instrument recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 381–385. IEEE, 2019. 3.3.1

David Huron. The humdrum toolkit: Software for music research. 1993. 4.1.2

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research (JMLR)*, 6:695–709, 2005. 2.7

International MIDI Association. Midi musical instrument digital interface specification 1.0. 1983. 2.1, 4.1.1

International MIDI Association. Standard midi files. 1988. 4.1.1

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017. 3.1.1

Özgür Izmirli and Roger B. Dannenberg. Understanding features and distance functions for music sequence alignment. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, pages 411–416, 2010. 3.2.2

Andreas Jansson, Eric J. Humphrey, Nicola Montecchio, Rachel M. Bittner, Aparna Kumar, and Tillman Weyde. Singing voice separation with deep u-net convolutional networks. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 745–751, 2017. 5.2.1

Natasha Jaques, Shixiang Gu, Richard E. Turner, and Douglas Eck. Tuning recurrent neural networks with reinforcement learning. In *5th International Conference on Learning Representations (ICLR) Workshop Track Proceedings*. OpenReview.net, 2017. 4.2.1

Vivek Jayaram and John Thickstun. Source separation with deep generative priors. In *37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 4724–4735, 2020. 5, 5.1.5, 5.2

Vivek Jayaram and John Thickstun. Parallel and flexible sampling from autoregressive models via langevin dynamics. In *38th International Conference on Machine Learning*

(ICML), volume 139 of *Proceedings of Machine Learning Research*, pages 4807–4818, 2021. 5.1.5

Dasaem Jeong, Taegyun Kwon, Yoojin Kim, Kyogu Lee, and Juhan Nam. Virtuosonet: A hierarchical rnn-based system for modeling expressive piano performance. In *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, pages 908–915, 2019a. 4.1.3

Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. Graph neural network for music score data and modeling expressive piano performance. In *36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 3060–3070, 2019b. 4.1.3

Cyril Joder and Björn W. Schuller. Off-line refinement of audio-to-score alignment by observation template adaptation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 206–210. IEEE, 2013. 3.1.2, 3.1.3

Cyril Joder, Slim Essid, and Gaël Richard. An improved hierarchical approach for music-to-symbolic score alignment. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, pages 39–45, 2010. 3.1.3

Cyril Joder, Slim Essid, and Gaël Richard. A conditional random field framework for robust and scalable audio-to-score matching. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 19(8):2385–2397, 2011. 3.1.2, 3.1.3

Cyril Joder, Slim Essid, and Gaël Richard. Learning optimal features for polyphonic audio-to-score alignment. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 21(10):2118–2128, 2013. 3.1.2, 3.1.3, 3.2.2

Daniel D. Johnson. Generating polyphonic music using tied parallel networks. In *Proceedings of the 6th International Conference on Computational Intelligence in Music, Sound, Art*

and Design, Lecture Notes in Computer Science, pages 128–143, 2017. 4.2.1, 4.2.3, 4.2.4, 4.2.4

Anna Jordanous. A standardised procedure for evaluating creative systems: Computational creativity evaluation based on what it is to be creative. *Cognitive Computation*, 4(3): 246–279, 2012. 4.2.5

Frederick P. Brooks Jr., Albert L. Hopkins Jr., Peter G. Neumann, and William V. Wright. An experiment in musical composition. *IRE Transactions on Electronic Computers*, 6(3): 175–182, 1957. 4.2.1

Dan Jurafsky and James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition.* Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009. 4.1.4

Maximos A. Kaliakatsos-Papakostas, Michael G. Epitropakis, and Michael N. Vrahatis. Musical composer identification through probabilistic and feedforward neural networks. In *European Conference on the Applications of Evolutionary Computation*, pages 411–420. Springer, 2010. 4.3.1

Maximos A Kaliakatsos-Papakostas, Michael G Epitropakis, and Michael N Vrahatis. Weighted markov chain model for musical composer identification. In *European Conference on the Applications of Evolutionary Computation*, volume 6625 of *Lecture Notes in Computer Science*, pages 334–343. Springer, 2011. 4.3.1

Maximos A. Kaliakatsos-Papakostas, Andreas Floros, and Michael N. Vrahatis. Intelligent real-time music accompaniment for constraint-free improvisation. In *IEEE 24th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 444–451, 2012. 6

T Anderson Keller, Jorn WT Peters, Priyank Jaini, Emiel Hoogeboom, Patrick Forré, and Max Welling. Self normalizing flows. *Beyond Backpropagation Workshop at Neural Information Processing Systems*, 2020. 2.6

Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 475–481, 2016. 3.3.1

Katherine C Kempfert and Samuel WK Wong. Where does haydn end and mozart begin? composer classification of string quartets. *Journal of New Music Research*, 49(5):457–476, 2020. 4.3, 4.3.1, 4.3.4, 4.3.4, 4.3.5

Joseph Keshet, Shai Shalev-Shwartz, Yoram Singer, and Dan Chazan. A large margin algorithm for speech-to-phoneme and music-to-score alignment. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 15(8):2373–2382, 2007. 3.1.2, 3.1.3

Chloé Kiddon and Pedro M. Domingos. Coarse-to-fine inference and learning for first-order probabilistic models. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2011. 5.1.2

Jong Wook Kim. *Automatic Music Transcription in the Deep Learning Era: Perspectives on Generative Neural Networks*. PhD thesis, New York University, 2020. 1

Sungwon Kim, Sang-gil Lee, Jongyoon Song, Jaehyeon Kim, and Sungroh Yoon. Flowavenet : A generative flow for raw audio. In *36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 3370–3378, 2019. 2.6, 5.3.2

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, (ICLR)*. OpenReview.net, 2015. 4.3.3

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Proceedings of the 31st Advances in Neural Information Processing Systems (Neurips)*, pages 10236–10245, 2018. 2.6, 2.6, 5.1, 5.1.1, 5.3.2

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, (ICLR)*. OpenReview.net, 2014. 2.2, 2.4, 2.4, 5.1

Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. 2.7, 5.1, 5.1.1, 5.1.2

Bartek T Knapik, Aad W Van Der Vaart, J Harry van Zanten, et al. Bayesian inverse problems with gaussian priors. *The Annals of Statistics*, 39(5):2626–2657, 2011. 5.2

Teuvo Kohonen. A self-learning musical grammar, or 'associative memory of the second kind'. *International Joint Conference on Neural Networks*, pages 1–5, 1989. 4.2.1

Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990. 3.3.5

Qiuqiang Kong, Yong Xu, Philip J. B. Jackson, Wenwu Wang, and Mark D. Plumbley. Single-channel signal separation and deconvolution with generative adversarial networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2747–2753. ijcai.org, 2019. (document), 5.2.1, 5.2.2, 5.3.5, 5.2

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *9th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2021. 2.7, 5.3.2

Rainer Kress. *Numerical analysis*. Springer, 1998. 3.1.2

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. 4.2.5, 5.3.1

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th Advances in Neural Information Processing Systems (Neurips)*, pages 1106–1114, 2012. 3.3.1, 4.2.5

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 66–71. Association for Computational Linguistics, 2018. 4.1.4

Volodymyr Kuleshov, S Zayd Enam, and Stefano Ermon. Audio super-resolution using neural nets. In *International Conference on Learning Representations (Workshop Track)*, 2017. (document), 5.3.7, 5.6

Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C. Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. In *Proceedings of the 32nd Advances in Neural Information Processing Systems (Neurips)*, pages 14881–14892, 2019. 2.5, 5.3.2

Taegyun Kwon, Dasaem Jeong, and Juhan Nam. Audio-to-score alignment of piano music using rnn-based automatic music transcription. In *14th Sound and Music Computing Conference, (SMC), Espoo, Finland, 5-8 July 2017*, 2017. 3.1.2, 3.1.3

Rémi Lajugie, Damien Garreau, Francis R. Bach, and Sylvain Arlot. Metric learning for temporal sequence alignment. In *Proceedings of the 27th Advances in Neural Information Processing Systems (Neurips)*, pages 1817–1825, 2014. 3.2.2

Rémi Lajugie, Piotr Bojanowski, Philippe Cuvillier, Sylvain Arlot, and Francis R. Bach. A weakly-supervised discriminative model for audio-to-score alignment. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2484–2488. IEEE, 2016. 3.1.2, 3.1.3, 3.2.2

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pages 29–37, 2011. 2.2, 2.3, 2.3, 4.1.1, 5.1

Victor Lavrenko and Jeremy Pickens. Polyphonic music modeling with random fields. In *Proceedings of the Eleventh ACM International Conference on Multimedia (ACMMM)*, pages 120–129. ACM, 2003. 4.2.1, 4

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. 2.3

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pages 2278–2324, 1998. 3.3.5, 5.3.1

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting Structured Data*, 2006. 2.2

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999. 5.2.1

Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Proceedings of the 13th Advances in Neural Information Processing Systems (Neurips)*, pages 556–562, 2000. 5.2.1

Te-Won Lee, Michael S Lewicki, Mark Girolami, and Terrence J Sejnowski. Blind source separation of more sources than mixtures using overcomplete representations. *IEEE Signal Processing Letters*, 6(4):87–90, 1999. 5.2.1

Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23(3):689–694, 2004. 5.3.6

Bochen Li, Xinzhao Liu, Karthik Dinesh, Zhiyao Duan, and Gaurav Sharma. Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications. *IEEE Transactions on Multimedia*, 21(2):522–535, 2019. 3.3.6

Yuanqing Li, Shun-Ichi Amari, Andrzej Cichocki, Daniel WC Ho, and Shengli Xie. Underdetermined blind source separation based on sparse representation. *IEEE Transactions on Signal Processing*, 54(2):423–437, 2006. 5.2.1

Feynman T. Liang, Mark Gotham, Matthew Johnson, and Jamie Shotton. Automatic stylistic composition of bach chorales with deep LSTM. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 449–456, 2017. (document), 2.3, 4.4, 4.2.1, 4.2.4

Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Proceedings of the 31st Advances in Neural Information Processing Systems (Neurips)*, pages 9628–9639, 2018. 4.1.5

Francesc Lluís, Jordi Pons, and Xavier Serra. End-to-end music source separation: Is it possible in the waveform domain? In *20th Annual Conference of the International Speech Communication Association (Interspeech)*, pages 4619–4623. ISCA, 2019. 5.2.1

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015. 3.1.1

Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988. 2.6

Hans-Dieter Lüke. The origins of the sampling theorem. *IEEE Communications Magazine*, 37(4):106–108, 1999. 2.1

Yi Luo and Nima Mesgarani. Conv-tasnet: Surpassing ideal time-frequency magnitude masking for speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(8):1256–1266, 2019. 5.3.5

Siwei Lyu. Interpretation and generalization of score matching. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 359–366, 2009. 2.7

Fangchang Ma, Ulas Ayaz, and Sertac Karaman. Invertibility of convolutional generative networks from partial measurements. In *Proceedings of the 31st Advances in Neural Information Processing Systems (Neurips)*, pages 9651–9660, 2018. 2.2

Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, 1971. 3.1.2

Upamanyu Madhow. Blind adaptive interference suppression for the near-far resistant acquisition and demodulation of direct-sequence CDMA signals. *IEEE Transactions on Signal Processing*, 45(1):124–136, 1997. 6

Akira Maezawa and Hiroshi G. Okuno. Bayesian audio-to-score alignment based on joint inference of timbre, volume, tempo, and note onset timings. *Computer Music Journal*, 39 (1):74–87, 2015. 3.1.3

Julien Mairal, Francis R. Bach, and Jean Ponce. Sparse modeling for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, 8(2-3):85–283, 2014. 5.2

Matija Marolt. A connectionist approach to automatic transcription of polyphonic piano music. *IEEE Transactions on Multimedia*, 6(3):439–449, 2004. 3.2, 3.3.6

Brian McFee, Eric J. Humphrey, and Juan Pablo Bello. A software framework for musical data augmentation. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 248–254, 2015a. 3.3.1

Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th Python in Science Conference*, volume 8, pages 18–25, 2015b. 3.2.3

Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron C. Courville, and Yoshua Bengio. Samplernn: An unconditional end-to-end neural audio generation model. In *5th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2017. 2.3, 5

Yoram Meron and Keikichi Hirose. Automatic alignment of a musical score to performed music. *Acoustical Science and Technology*, 22(3):189–198, 2001. 3.1.2, 3.1.3

Marvin Minsky and Seymour Papert. *Perceptrons - an Introduction to Computational Geometry*. MIT Press, 1987. 4.3.4

Marius Miron, Julio José Carabias-Orti, and Jordi Janer. Audio-to-score alignment at the note level for orchestral recordings. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, pages 125–130, 2014. 3.1.3

Shinjiro Mita, Gaku Hatanaka, Alexis Meneses, Nattapong Thammasan, and Daiki Miura. Mirex 2017 : Multi-instrumental end-to-end convolutional neural network for multiple f0 estimation. *MIREX Multiple Fundamental Frequency Estimation & Tracking Task*, 2017. 3.2, 3.3.6

Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv Preprint arXiv:1610.03483*, 2016. 2.5

Michael C. Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 6(2-3):247–280, 1994. 4.2.1

Meinard Müller. Dynamic time warping. In *Information Retrieval for Music and Motion*. Springer, 2007. 3.2.2

Eita Nakamura, Kazuyoshi Yoshii, and Haruhiro Katayose. Performance error detection and post-processing for fast and accurate symbolic music alignment. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, pages 347–353, 2017. 3, 3.1.1, 1, 3.1.4

Juhan Nam, Jiquan Ngiam, Honglak Lee, and Malcolm Slaney. A classification-based polyphonic piano transcription approach using learned feature representations. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 175–180, 2011. 3.3.1

Radford M Neal et al. MCMC using hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*. Chapman & Hall, 2011. 5, 5.1.1

Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436, 2015. 5.2.3

XuanLong Nguyen, Martin J. Wainwright, and Michael I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010. 2.5

Bernhard Niedermayer. Improving accuracy of polyphonic music-to-score alignment. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 585–590, 2009. 3.1.2, 3.1.3

Bernhard Niedermayer and Gerhard Widmer. A multi-pass algorithm for accurate audio-to-score alignment. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, pages 417–422, 2010. 3.1.2, 3.1.3

Shigeto Nishida, Masatoshi Nakamura, Akio Ikeda, and Hiroshi Shibasaki. Signal separation

of background eeg and spike by using morphological filter. *Medical Engineering & Physics*, 21(9):601–608, 1999. 5.2.1

Feng Niu, Benjamin Recht, Christopher Ré, and Stephen J. Wright. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Proceedings of the 24th Advances in Neural Information Processing Systems (Neurips)*, pages 693–701, 2011. 5.1.4

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Proceedings of the 29th Advances in Neural Information Processing Systems (Neurips)*, pages 271–279, 2016. 2.5

Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent. Multichannel audio source separation with deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(9):1652–1664, 2016. 5.2.1

Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 1928. 4.2

Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. This time with feeling: learning expressive musical performance. *Neural Computing and Applications*, 32(4):955–967, 2020. 4.2, 4.2.3, 4.2.4

Nicola Orio and Diemo Schwarz. Alignment of monophonic and polyphonic music to a score. In *Proceedings of the 2001 International Computer Music Conference (ICMC)*. Michigan Publishing, 2001. 1, 3.1.2, 3.1.3, 3.2.2, 3.2.2

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 4052–4061, 2018. 4.1.5, 4.2.3, 5.1, 5.1, 5.3.5

Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544, 2016. 5.3.8

Grigorios A Pavliotis. *Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations*. Springer, 2014. 2.7

Marcus Pearce and Geraint Wiggins. Towards a framework for the evaluation of machine compositions. In *Proceedings of the AISB 2001 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 22–32, 2001. 4.2.5

Marcus T Pearce and Geraint A Wiggins. Evaluating cognitive models of musical composition. In *Proceedings of the 4th International Joint Workshop on Computational Creativity*, pages 73–80. Goldsmiths, University of London, 2007. 4.2.5

Barak A. Pearlmutter and Lucas C. Parra. Maximum likelihood blind source separation: A context-sensitive generalization of ICA. In *Proceedings of the 9th Advances in Neural Information Processing Systems 9 (Neurips)*, pages 613–619, 1996. 5.2.1

Fabrizio Pedersoli, George Tzanetakis, and Kwang Moo Yi. Improving music transcription by pre-stacking A u-net. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 506–510. IEEE, 2020. 3.3.1

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. volume 12, pages 2825–2830, 2011. (document), 3.6

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019. 2.5

Wei Ping, Kainan Peng, and Jitong Chen. Clarinet: Parallel wave generation in end-to-end text-to-speech. In *7th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2019. 5.1.1

Wei Ping, Kainan Peng, Kexin Zhao, and Zhao Song. Waveflow: A compact flow-based model for raw audio. In *37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 7706–7716, 2020. 2.6, 5.3.2

Richard C Pinkerton. Information theory and melody. *Scientific American*, 1956. 4.2.1

Mark D. Plumbley, Samer A. Abdallah, Juan Pablo Bello, Mike E. Davies, Giuliano Monti, and Mark B. Sandler. Automatic music transcription and audio source separation. *Cybernetics & Systems*, 33(6):603–627, 2002. 1

Graham E. Poliner and Daniel P. W. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Applied Signal Processing*, 2007. 3.2.1, 3.3.1, 3.3.6

Emanuele Pollastri and Giuliano Simoncelli. Classification of melodies by composer with hidden markov models. In *First International Conference on WEB Delivering of Music (WEDELMUSIC*, pages 88–95. IEEE Computer Society, 2001. 4.3.1

Jordi Pons and Xavier Serra. Designing efficient architectures for modeling temporal features with convolutional neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2472–2476. IEEE, 2017. 3.3.1

Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE, 2019. 2.6, 5.3.2

Lawrence R. Rabiner and Ronald W. Schafer. Introduction to digital speech processing. *Foundations and Trends in Signal Processing*, 1(1/2):1–194, 2007. 3.3.3, 3.3.3

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019. 4.2.2, 4.2.5, 5.1

Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, 2016. 3.2.1, 3.3

Colin Raffel and Daniel P. W. Ellis. Optimizing dtw-based audio-to-midi alignment and matching. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 81–85. IEEE, 2016. 3.1.3, 3.1.4, 3, 3.2.3, 3.2.3

Colin Raffel and Daniel PW Ellis. Intuitive analysis, creation and manipulation of midi data with pretty midi. In *15th International Society for Music Information Retrieval Conference Late Breaking and Demo Papers*, pages 84–93, 2014. 3.2.3

Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. mir_eval: A transparent implementation of common MIR metrics. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, pages 367–372, 2014. (document), 3.6

Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The MUSDB18 corpus for music separation, 2017. URL https://doi.org/10.5281/zenodo.1117372. 5.3.5

Ankit Raj, Yuqi Li, and Yoram Bresler. Gan-based projector for faster recovery with convergence guarantees in linear inverse problems. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5601–5610. IEEE, 2019. 5.2

Marc'Aurelio Ranzato, Y-Lan Boureau, Sumit Chopra, and Yann LeCun. A unified energy-based framework for unsupervised learning. In *11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 2 of *Proceedings of Machine Learning Research*, pages 371–379, 2007. 2.7

Christopher Raphael. Automatic segmentation of acoustic musical signals using hidden markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4): 360–370, 1999. 3.2.2

Christopher Raphael. Coarse-to-fine dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1379–1390, 2001. 5.1.2

Christopher Raphael. A hybrid graphical model for aligning polyphonic audio with musical scores. In *Proceedings of the 5th International Society for Music Information Retrieval Conference (ISMIR)*, pages 387–394, 2004. 3.1.2

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pages 14866–14876, 2019. 5.1, 5.1.1

Curtis Roads. Artificial intelligence and music. *Computer Music Journal*, 1980. 4.2.1

Adam Roberts, Jesse H. Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In *35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 4361–4370, 2018. 2.4, 4.2.1

Gareth O Roberts and Richard L Tweedie. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363, 1996. 2.7

John W Robinson and Stephen L Howell. Automatic arpeggiator. *The Journal of the Acoustical Society of America*, 74(5):1666–1666, 1983. 6

Thomas Rossing. *Springer handbook of acoustics*. Springer, 2007. 2.1

Jonathan Le Roux, Scott Wisdom, Hakan Erdogan, and John R. Hershey. SDR - half-baked or well done? In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 626–630. IEEE, 2019. 5.3.5

Sam T. Roweis. One microphone source separation. In *Proceedings of the 13th Advances in Neural Information Processing Systems (Neurips)*, pages 793–799, 2000. 5.2.1

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 3.2

Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. Dynamic routing between capsules. In *Proceedings of the 30th Advances in Neural Information Processing Systems (Neurips)*, pages 3856–3866, 2017. 5.3.5

Pasha Sadeghian, Casey Wilson, Stephen Goeddel, and Aspen Olmsted. Classification of music by composer using fuzzy min-max neural networks. In *12th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 189–192. IEEE, 2017. 4.3.1

Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(1):43–49, 1978. 3.1, 3.1.4

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Proceedings of the 29th Advances in Neural Information Processing Systems (Neurips)*, pages 2226–2234, 2016. 4.2.5, 5.2.2

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *5th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2017. 4.2.3, 5.1, 5.1, 5.3.2

Gerard Salton and Michael McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1984. 3.3.6

Craig Stuart Sapp. Online database of scores in the humdrum file format. In *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR)*, pages 664–665, 2005. 3.1.4, 4, 4.2.2, 4.3.2

Mikkel N. Schmidt and Rasmus Kongsgaard Olsson. Single-channel speech separation using sparse non-negative matrix factorization. In *9th International Conference on Spoken Language Processing (Interspeech)*. ISCA, 2006. 5.2.1

Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE, 2012. 4.1.4

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. The Association for Computer Linguistics, 2016. 4.1.4

Shai Shalev-Shwartz, Joseph Keshet, and Yoram Singer. Learning to align polyphonic music. In *Proceedings of the 5th International Society for Music Information Retrieval Conference (ISMIR)*, 2004. 3.1.2, 3.1.3

Claude Elwood Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 1949. 4.1.2, 4.2

Roger N. Shepard. Geometrical approximations to the structure of musical pitch. *Psychological Review*, 89(4):305, 1982. 4.2.1

Zhengshan Shi, Craig Sapp, Kumaran Arul, Jerry McBride, and Julius O. Smith III. SUPRA: digitizing the stanford university piano roll archive. In *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, pages 517–523, 2019. 5.3.1

Galit Shmueli et al. To explain or to predict? *Statistical Science*, 25(3):289–310, 2010. 5.2.2

Rui Shu, Hung H Bui, Shengjia Zhao, Mykel J Kochenderfer, and Stefano Ermon. Amortized inference regularization. In *Advances in Neural Information Processing Systems*, 2018. 2.4

Siddharth Sigtia, Emmanouil Benetos, Nicolas Boulanger-Lewandowski, Tillman Weyde, Artur S. d'Avila Garcez, and Simon Dixon. A hybrid recurrent neural network for music transcription. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2061–2065. IEEE, 2015. 3.3.1

Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):927–939, 2016. 3.3.1

Patrice Y. Simard, David Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *7th International Conference on Document Analysis and Recognition (ICDAR)*, pages 958–962. IEEE Computer Society, 2003. 3.3.1

Paris Smaragdis and Shrikant Venkataramani. A neural network alternative to non-negative audio models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 86–90. IEEE, 2017. 5.2.1

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Proceedings of the 32nd Advances in Neural Information Processing Systems (Neurips)*, pages 11895–11907, 2019. (document), 1, 2.2, 2.7, 2.7, 2.7, 5, 5.1, 5.1, 5.1.1, 5.1.2, 5.1.5, 5.1.5, 5.3.2, 5.9

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Proceedings of the 33rd Advances in Neural Information Processing Systems (Neurips)*, 2020. 5.1.5, 5.1.5

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Proceedings of the Thirty-Fifth Conference*

on Uncertainty in Artificial Intelligence (UAI), volume 115 of *Proceedings of Machine Learning Research*, pages 574–584, 2019. 2.7, 2.7

Yang Song, Chenlin Meng, Renjie Liao, and Stefano Ermon. Nonlinear equation solving: A faster alternative to feedforward computation. *arXiv Preprint arXiv:2002.03629*, 2020. 5.1.1

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2021. 5, 5.1.1, 5.1.2

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19:2822–2878, 2018. 4.3

Ferréol Soulez, Xavier Rodet, and Diemo Schwarz. Improving polyphonic and poly-instrumental music to score alignment. In *Proceedings of the 4th International Society for Music Information Retrieval Conference (ISMIR)*, 2003. 3.1.3, 3.2.2

Martin Spiertz and Volker Gnann. Source-filter based clustering for monaural blind source separation. In *Proceedings of the 12th International Conference on Digital Audio Effects*, 2009. 5.2.1

Peter Steiner, Simon Stone, Peter Birkholz, and Azarakhsh Jalalvand. Multipitch tracking in music signals using echo state networks. In *28th European Signal Processing Conference (EUSIPCO)*, pages 126–130. IEEE, 2020. 3.3.6

Daniel Stoller, Sebastian Ewert, and Simon Dixon. Adversarial semi-supervised audio source separation applied to singing voice extraction. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2391–2395. IEEE, 2018a. 5.2.1

Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pages 334–340, 2018b. 5.2.1

Bob L. Sturm, Joao Felipe Santos, Oded Ben-Tal, and Iryna Korshunova. Music transcription modelling and composition using deep learning. *Conference on Computer Simulation of Musical Creativity*, 2016. 4.2.1

Li Su and Yi-Hsuan Yang. Combining spectral and temporal representations for multipitch estimation of polyphonic music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(10):1600–1612, 2015a. 3.3.6

Li Su and Yi-Hsuan Yang. Escaping from the abyss of manual annotation: New methodology of building polyphonic datasets for automatic music transcription. In *International Symposium on Computer Music Multidisciplinary Research (CMMR)*, pages 309–321, 2015b. 1, 3.2.1, 3.3.6

Y. Cem Sübakan and Paris Smaragdis. Generative adversarial source separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 26–30. IEEE, 2018. 5.2.1

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th Advances in Neural Information Processing Systems (Neurips)*, pages 3104–3112, 2014. 3

Ayaka Takamoto, Mayu Umemura, Mitsuo Yoshida, and Kyoji Umemura. Improving compression based dissimilarity measure for music score analysis. In *International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA)*, pages 1–5. IEEE, 2016. 4.3.1

Ayaka Takamoto, Mitsuo Yoshida, Kyoji Umemura, and Yuko Ichikawa. Feature selection for composer classification method using quantity of information. In *Proceedings of the*

10th International Conference on Knowledge and Smart Technology (KST), pages 30–33. IEEE, 2018. 4.3.1

Lucas Theis and Matthias Bethge. Generative image modeling using spatial lstms. In *Proceedings of the 28th Advances in Neural Information Processing Systems (Neurips)*, pages 1927–1935, 2015. 5.1.1

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. In *4th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2016. 4.2, 5.3.5

John Thickstun, Zaïd Harchaoui, and Sham M. Kakade. Learning features of music from scratch. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2017. (document), 3.1, 3.6

Peter M. Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4):27–43, 1989. 4.2.1

Jakub M. Tomczak and Max Welling. VAE with a vampprior. In *21st International Conference on Artificial Intelligence and Statistics, (AISTATS)*, volume 84 of *Proceedings of Machine Learning Research*, pages 1214–1223, 2018. 2.4

Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J. Pal. Deep complex networks. In *6th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2018. 3.3.1, 3.3.4, 3.3.6, 3.3.5

Joel A. Tropp and Stephen J. Wright. Computational methods for sparse solution of linear inverse problems. *Proceedings of the IEEE*, 98(6):948–958, 2010. 5.2

Daylin Troxel. Music transcription with a convolutional neural network 2016. *MIREX Multiple Fundamental Frequency Estimation & Tracking Task*, 2016. 3.2, 3.3.6

Robert J. Turetsky and Daniel P. W. Ellis. Ground-truth transcriptions of real music from force-aligned MIDI syntheses. In *Proceedings of the 4th International Society for Music Information Retrieval Conference (ISMIR)*, 2003. 3.1, 3.2.2, 3.2.2

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Deep image prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9446–9454, 2018. 5.2.1

Benigno Uria, Iain Murray, and Hugo Larochelle. RNADE: the real-valued neural autoregressive density-estimator. In *Proceedings of the 26th Advances in Neural Information Processing Systems (Neurips)*, pages 2175–2183, 2013. 2.3, 5.1, 5.1.3

Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *Journal of Machine Learning Research (JMLR)*, 17:205:1–205:37, 2016. 2.3

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop*, page 125. ISCA, 2016a. 2.3, 5, 5.1, 5.3.2

Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *33nd International Conference on Machine Learning (ICML)*, volume 48 of *Proceedings of Machine Learning Research*, pages 1747–1756, 2016b. 2.3, 4.2.3, 5.1

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Proceedings of the 30th Advances in Neural Information Processing Systems (Neurips)*, pages 6306–6315, 2017. 2.4, 5.1, 5.1.1

Aäron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen,

Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. Parallel wavenet: Fast high-fidelity speech synthesis. In *35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 3915–3923, 2018. 2.6, 5.1.1

Peter Van Kranenburg and Eric Backer. Musical style recognition: a quantitative approach. In *Handbook of Pattern Recognition and Computer Vision*, pages 583–600. World Scientific, 2005. 4.3, 4.3.1, 4.3.5

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 30th Advances in Neural Information Processing Systems (Neurips)*, pages 5998–6008, 2017. 2.3

Christophe Veaux, Junichi Yamagishi, Kirsten MacDonald, et al. Superseded-cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit. 2016. 5.3.1

Gissel Velarde, Tillman Weyde, Carlos Eduardo Cancino Chacón, David Meredith, and Maarten Grachten. Composer recognition based on 2d-filtered piano-rolls. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 115–121, 2016. 4.3.1, 4.3.2, 4.3.4

Shrikant Venkataramani, Y. Cem Sübakan, and Paris Smaragdis. Neural network alternatives to convolutive audio models for source separation. In *27th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2017. 5.2.1

Cédric Villani. *Optimal Transport: Old and New*. Springer, 2009. 2.5

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research (JMLR)*, 11:3371–3408, 2010. 2.4

Tuomas Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 15(3):1066–1074, 2007. 5.2.1

Raunaq Vohra, Kratarth Goel, and Jajati Keshari Sahoo. Modeling temporal dependencies in data using a DBN-LSTM. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–4. IEEE, 2015. 4.2.1

Zheng Wang, Johnathan M. Bardsley, Antti Solonen, Tiangang Cui, and Youssef M. Marzouk. Bayesian inverse problems with l_1 priors: A randomize-then-optimize approach. *SIAM Journal on Scientific Computing*, 39(5), 2017. 5.2

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 5.2.2

Ziyu Wang, Yiyi Zhang, Yixiao Zhang, Junyan Jiang, Ruihan Yang, Gus Xia, and Junbo Zhao. PIANOTREE VAE: structured representation learning for polyphonic music. In *Proceedings of the 21th International Society for Music Information Retrieval Conference (ISMIR)*, pages 368–375, 2020. 4.1.3

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *28th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, pages 681–688, 2011. 5.1, 5.1.4

Auke J. Wiggers and Emiel Hoogeboom. Predictive sampling with forecasting autoregressive models. In *37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 10260–10269, 2020. 5.1.1

Kevin W. Wilson, Bhiksha Raj, Paris Smaragdis, and Ajay Divakaran. Speech denoising using nonnegative matrix factorization with priors. In *Proceedings of the IEEE Interna-*

tional Conference on Acoustics, Speech, and Signal Processing, (ICASSP), pages 4029–4032. IEEE, 2008. 5.2.1

Jacek Wolkowicz and Vlado Keselj. Evaluation of n-gram-based classification approaches on classical music corpora. In *Proceedings of the 4th International Conference on Mathematics and Computation in Music (MCM)*, Lecture Notes in Computer Science, pages 213–225. Springer, 2013. 4.3.1, 4.3.4

Jacek Wołkowicz, Zbigniew Kulka, and Vlado Kešelj. N-gram-based approach to composer recognition. *Archives of Acoustics*, 33(1):43–55, 2008. 4.3.1

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv Preprint arXiv:1609.08144*, 2016. 4.1.4

Wei Xiong, Jiahui Yu, Zhe Lin, Jimei Yang, Xin Lu, Connelly Barnes, and Jiebo Luo. Foreground-aware image inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5840–5848, 2019. 1

Yilun Xu, Yang Song, Sahaj Garg, Linyuan Gong, Rui Shu, Aditya Grover, and Stefano Ermon. Anytime sampling for autoregressive models via ordered autoencoding. In *9th International Conference on Learning Representations (ICLR)*. OpenReview.net, 2021. 5.1.1

Adrien Ycart, Andrew McLeod, Emmanouil Benetos, and Kazuyoshi Yoshii. Blending acoustic and language model predictions for automatic music transcription. In *Proceedings of*

the 20th International Society for Music Information Retrieval Conference (ISMIR), pages 454–461, 2019a. 3.3.1

Adrien Ycart, Daniel Stoller, and Emmanouil Benetos. A comparative study of neural models for polyphonic music sequence transduction. In *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, pages 470–477, 2019b. 3.3.1

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Proceedings of the 27th Advances in Neural Information Processing Systems (Neurips)*, pages 3320–3328, 2014. 5.1.2

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv Preprint arXiv:1506.03365*, 2015. 5.3.1

Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In *Proceedings of the 14th European Conference on Computer Vision (ECCV)*, volume 9907, pages 649–666. Springer, 2016. 5.3.6

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 28th Advances in Neural Information Processing Systems (Neurips)*, pages 649–657, 2015. 4.1.4

Xuaner Cecilia Zhang, Ren Ng, and Qifeng Chen. Single image reflection separation with perceptual losses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4786–4794, 2018a. 5.2.1

Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2472–2481, 2018b. 5.3.7

Yanhui Zhou, Jinghuai Gao, Wenchao Chen, and Pascal Frossard. Seismic simultaneous source separation via patchwise sparse representation. *IEEE Transactions on Geoscience and Remote Sensing*, 54(9):5271–5284, 2016. 6

Michael Zibulevsky and Barak A Pearlmutter. Blind source separation by sparse decomposition in a signal dictionary. *Neural Computation*, 13(4):863–882, 2001. 5.2.1

Appendix A

EXTENDED LANGEVIN SAMPLING RESULTS

We fine-tuned the MNIST, CIFAR-10, and LSUN Glow models at 10 noise levels σ^2 for 50 epochs each on clusters of 4 1080Ti GPU's. This procedure converges rapidly, with no further decrease of the negative log-likelihood after the first 10 epochs. Although Glow models theoretically have full support, the noiseless pre-trained models assign vanishing probability to highly noisy images. In practice, this can cause invertibility assertion failures when fine-tuning directly from the noiseless model. To avoid this we took an iterative approach: first fine-tune the lowest noise level $\sigma = .01$ from the noiseless model, then fine-tune the $\sigma = .016$ model from the $\sigma = .01$ model, etc. For PixelCNN++ CIFAR-10 models, we fine-tuned 20 noise models for 10 epochs each on a single 1080Ti GPU.

In concrete detail, a batch of 50 Langevin separation results for MNIST or CIFAR-10 using NCSN takes < 5 minutes on a single 1080Ti GPU. Running Langevin sampling with Glow or PixelCNN++ is much slower. A batch of 50 Langevin separation results on MNIST or CIFAR-10 using Glow takes about 30 minutes on a 1080Ti. A batch of 9 Langevin separation result on LSUN using Glow takes 2-3 hours on a 1080Ti. We observe that substantial time is spent loading and unloading each of the noisy models p_σ from memory, so the wall-clock sampling time for Glow could potentially be improved with engineering effort. A batch of 16 Langevin separation results on CIFAR-10 using PixelCNN++ takes about 60 minutes on a 1080Ti.

A.1 Intermediate Results During Noise-Annealed Langevin Sampling

Figure A.1: Intermediate CIFAR-10 separation results taken at noise levels σ during the annealing process of Langevin separation with an NCSN prior.

A.2 Additional PixelCNN++ Sampling Results

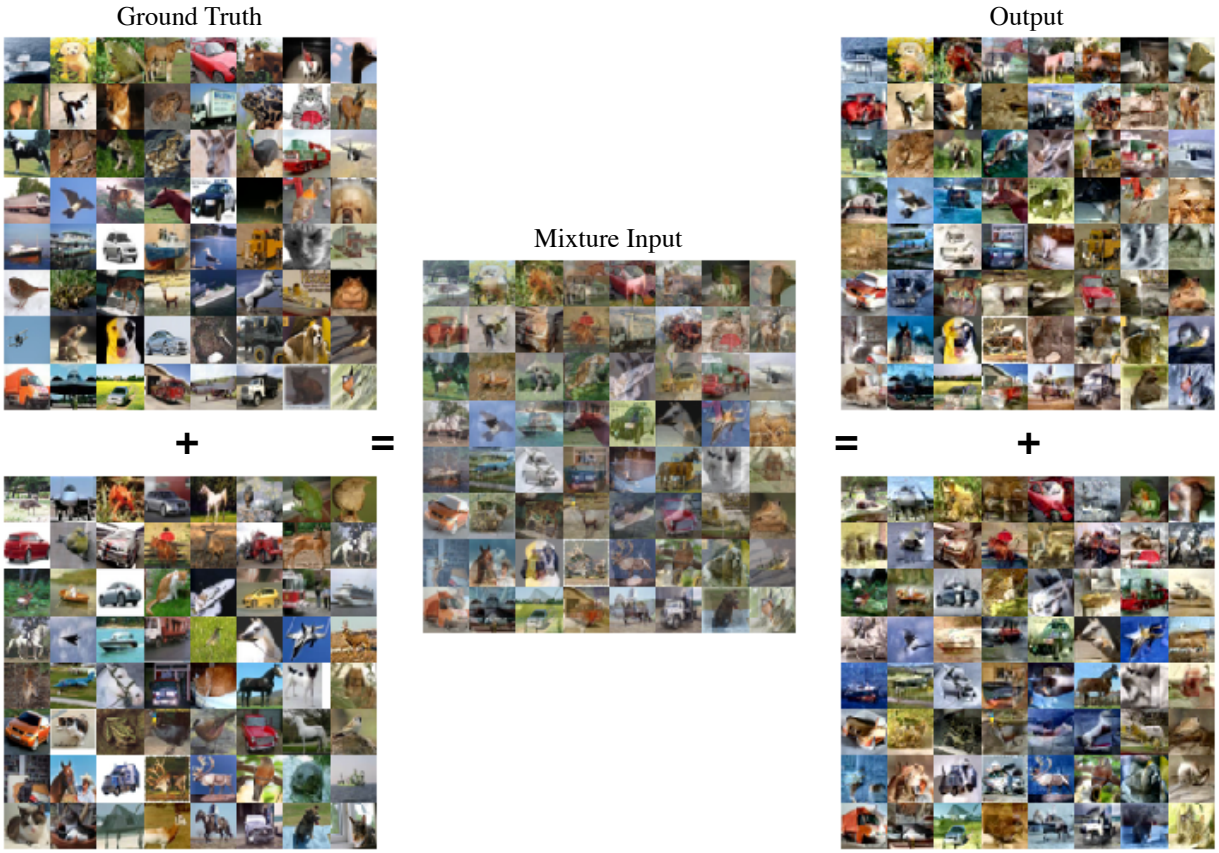


Figure A.2: Additional uncurated results of Langevin source separation (Section 5.3.5) for mixtures of CIFAR-10 test-set images using a PixelCNN++ prior trained on CIFAR-10.

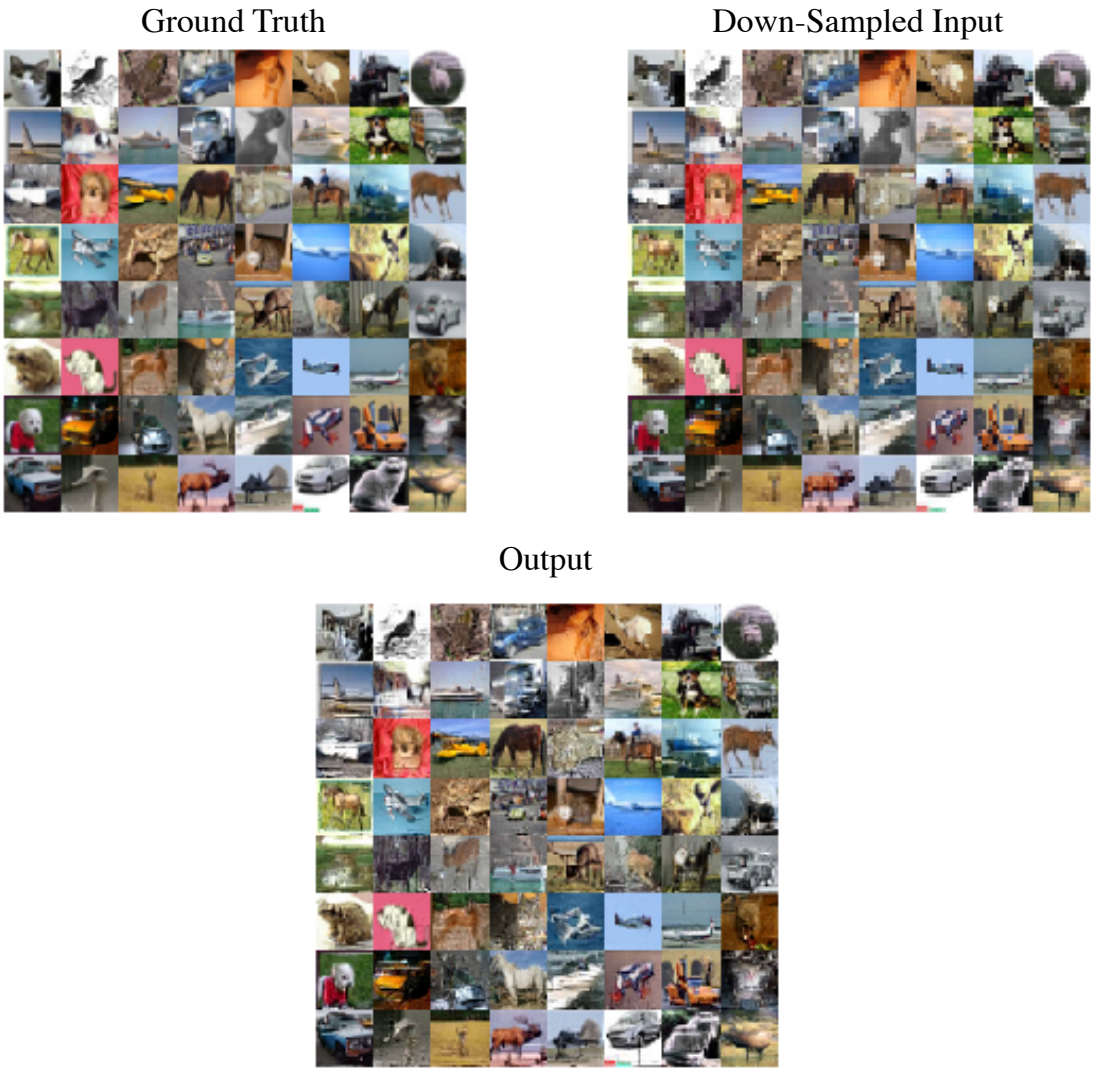


Figure A.3: Additional uncurated results of Langevin super-resolution (Section 5.3.7) applied to down-sampled CIFAR-10 test-set images using a PixelCNN++ prior trained on CIFAR-10.

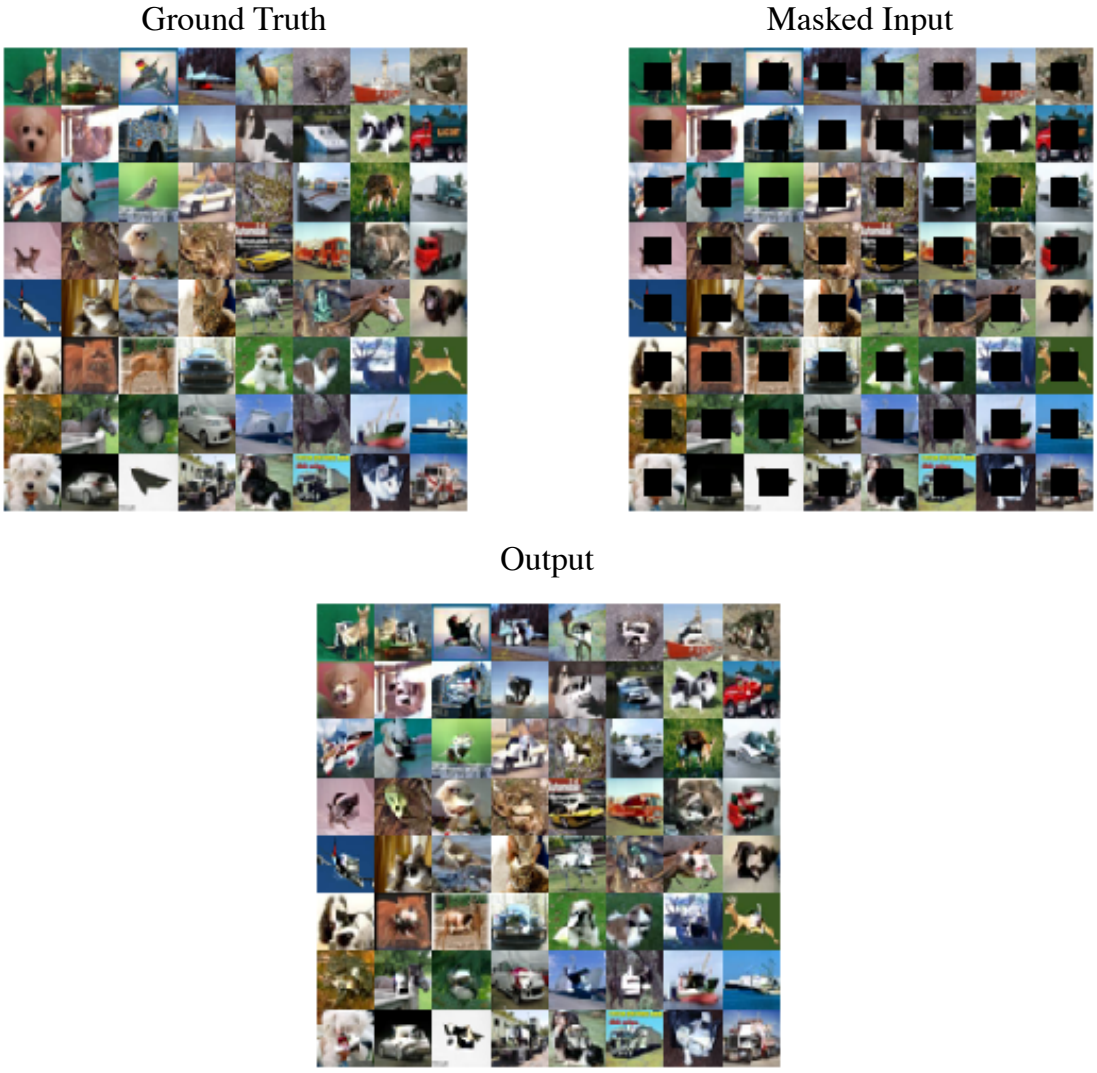


Figure A.4: Additional uncurated results of Langevin inpainting (Section 5.3.8) applied to masked CIFAR-10 test-set images using a PixelCNN++ prior trained on CIFAR-10.

A.3 Extended Glow LSUN Separation Results

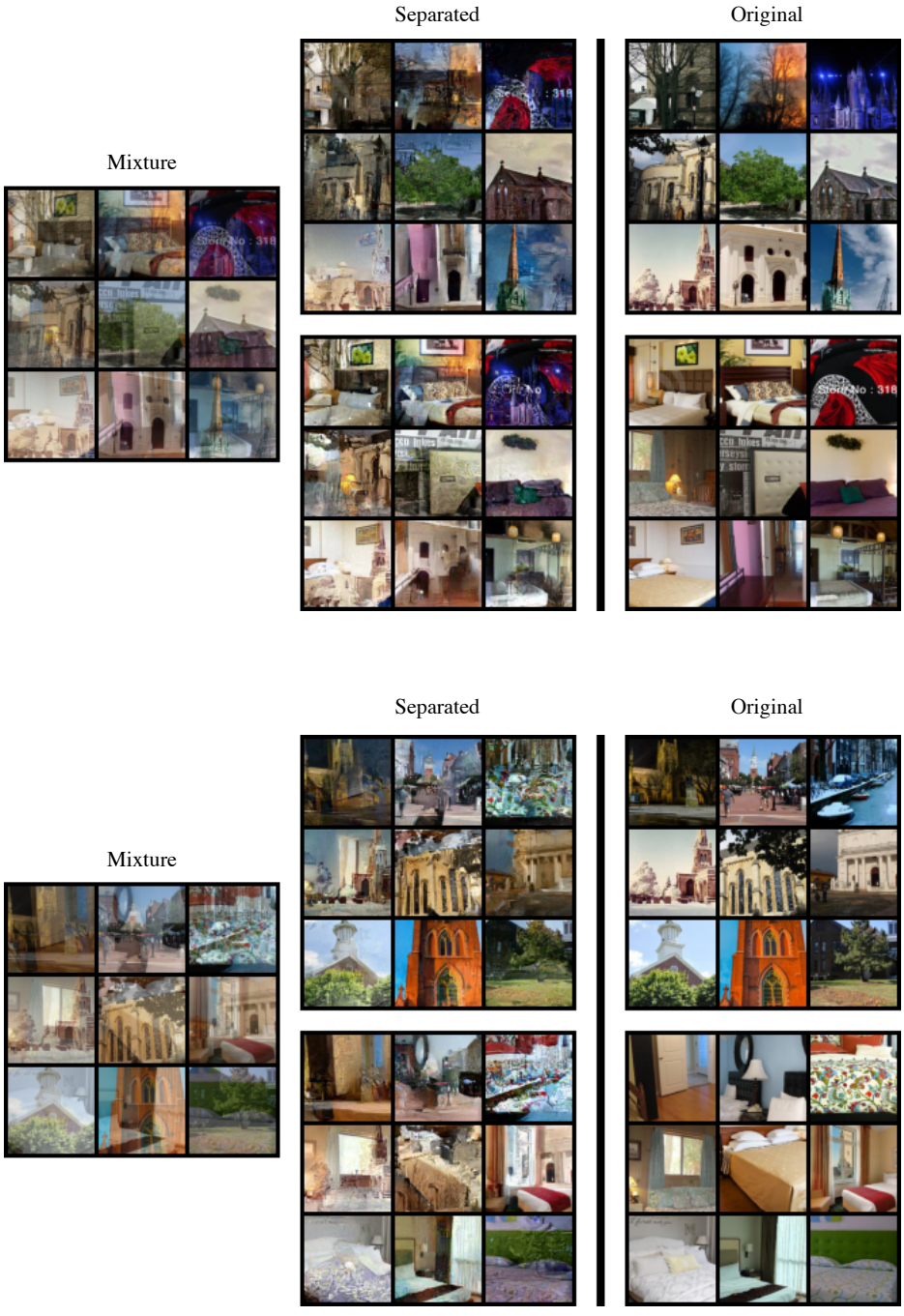


Figure A.5: Uncurated church/bedroom LSUN separation results using Glow as a prior.

A.4 Extended NCSN CIFAR-10 Separation Results

Figure A.6: Uncurated class-agnostic CIFAR-10 separation results using NCSN as a prior.

A.5 Extended Glow CIFAR-10 Separation Results

Figure A.7: Uncurated class-agnostic CIFAR-10 separation results using Glow as a prior.

A.6 Extended NCSN CIFAR-10 Colorization Results

Grayscale

Colorized

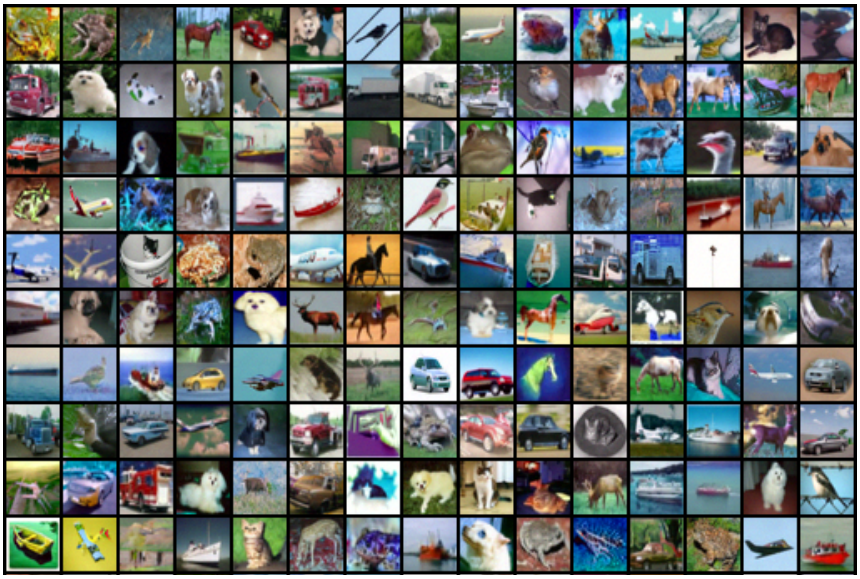


Figure A.8: Uncurated CIFAR-10 colorization results using NCSN as a prior.

A.7 Extended Glow CIFAR-10 Colorization Results

Grayscale

Colorized

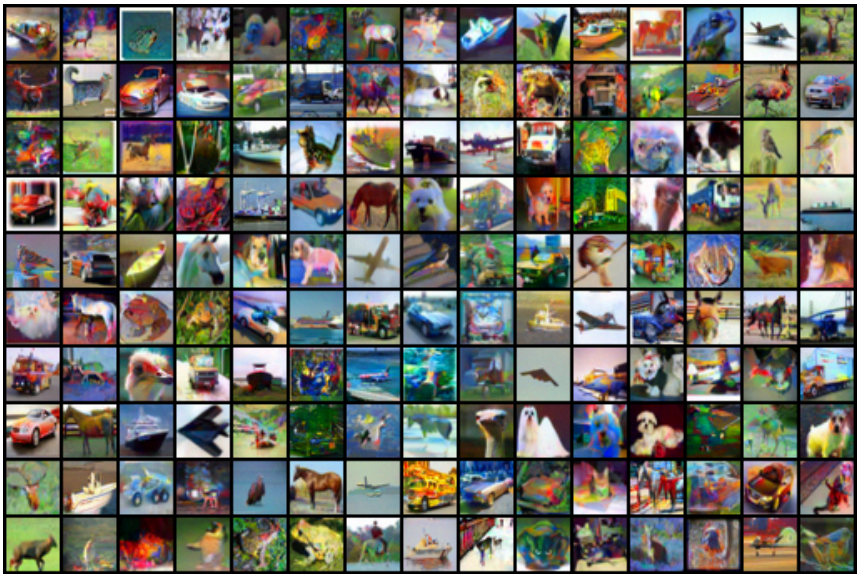


Figure A.9: Uncurated CIFAR-10 colorization results using Glow as a prior.