

MACHINE LEARNING APPROACHES TO HISTORIC MUSIC RESTORATION

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Quinn Coleman

March 2021

© 2021
Quinn Coleman
ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE: Machine Learning Approaches to Historic
Music Restoration

AUTHOR: Quinn Coleman

DATE SUBMITTED: March 2021

COMMITTEE CHAIR: Dennis Sun, Ph.D.
Professor of Statistics

COMMITTEE MEMBER: John Clements, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Paul Anderson, Ph.D.
Professor of Computer Science

ABSTRACT

Machine Learning Approaches to Historic Music Restoration

Quinn Coleman

In 1889, a representative of Thomas Edison recorded Johannes Brahms playing a piano arrangement of his piece titled “Hungarian Dance No. 1”. This recording acts as a window into how musical masters played in the 19th century. Yet, due to years of damage on the original recording medium of a wax cylinder, it was un-listenable by the time it was digitized into WAV format. This thesis presents machine learning approaches to an audio restoration system for historic music, which aims to convert this poor-quality Brahms piano recording into a higher quality one. Digital signal processing is paired with two machine learning approaches: non-negative matrix factorization and deep neural networks. Our results show the advantages and disadvantages of our approaches, when we compare them to a benchmark restoration of the same recording made by the Center for Computer Research in Music and Acoustics at Stanford University. They also show how this system provides the restoration potential for a wide range of historic music artifacts like this recording, requiring minimal overhead made possible by machine learning. Finally, we go into possible future improvements to these approaches.

ACKNOWLEDGMENTS

Thanks to:

- Dennis Sun, my advisor, for your knowledge, guidance and helpfulness, and thank you for giving me the privilege to work on this exciting thesis.
- Mom, Dad, Julie, Ryan and Sarah for supporting and inspiring me every step of the way. I couldn't have done this without you and I love you all so much.
- Dana Clanin, my middle school band teacher.
- Andrew Guenther, for uploading this template

TABLE OF CONTENTS

	Page
LIST OF TABLES	viii
LIST OF FIGURES	ix
CHAPTER	
1 Introduction	1
2 Background	4
2.1 Sound and Digital Signal Processing	4
2.1.1 Sound, Music and Recording	4
2.1.2 Digital Signal Processing	7
2.1.2.1 The Fourier Transform	7
2.1.2.2 The Short-Time Fourier Transform	9
2.2 Machine Learning	11
2.2.1 Non-Negative Matrix Factorization	12
2.2.2 Deep Learning and Artificial Neural Networks	15
2.2.2.1 Artificial Neural Networks	16
2.2.2.2 Recurrent Neural Networks and Deep Learning	17
2.2.2.3 Gradient-Based Optimization and Backpropagation	19
2.2.3 Training Practice	20
3 Related Works	22
3.1 Classical Restoration Methods	22
3.2 Machine Learning Restoration Methods	23
3.2.1 Methods with Non-Negative Matrix Factorization	23
3.2.2 Methods with Deep Learning	27

4	Implementation	32
4.1	Pre and Post Processing Audio Data	32
4.2	Non-negative Matrix Factorization Approach	33
4.2.1	Learning Piano Basis Vectors	33
4.2.2	Learning Noise Basis Vectors	37
4.2.3	Using the Learned Basis Vectors for Source Separation	38
4.3	Deep Learning Approach	41
4.3.1	Synthesizing Training Data	42
4.3.1.1	Augmenting Noise Source Data	43
4.3.1.2	Augmenting Piano Source Data	44
4.3.2	Improving Upon a Baseline	44
5	Evaluation	48
5.1	Non-negative Matrix Factorization Approach	50
5.2	Deep Learning Approach	60
5.3	Blind Test Including Both Approaches	64
6	Conclusion	67
6.1	Future Work	67
6.2	Summary	68
	BIBLIOGRAPHY	69

APPENDICES

LIST OF TABLES

Table		Page
3.1	Optimal Hyperparameters for Speech Denoising Setting by Huang et al. [12]	29
4.1	Model Architectures using Novel Hyperparameters as Potential Improvements over the Baseline	47
5.1	Results for the Number of Noise Basis Vectors with Supervised NMF	52
5.2	Results for the Variations of Semi-Supervised NMF	54
5.3	Results for Various L1 Penalty Terms on Current Best NMF Approach	58

LIST OF FIGURES

Figure	Page
2.1 Waveform of a Piano Recording	6
2.2 Spectrogram of a Piano Recording	10
2.3 Non-Negative Matrix Factorization	13
2.4 Simple Neural Network Diagram	17
2.5 Recurrent Neural Network A (left) Unrolled in Time (right) [2] . . .	18
3.1 NMF Modeling a Recording of Mary Had a Little Lamb [23]	24
3.2 NMF Source Separation Pipeline by Sun and Bryan [23]	25
3.3 DL Source Separation Model Architecture by Huang et al. [12] . . .	30
4.1 Score Piano Basis Vectors	35
4.2 Damaged Piano Filter	35
4.3 Score Piano Basis Vectors with Damage	36
4.4 Prior-Learned Basis Vectors: Noise (left) and Piano (right)	37
4.5 Supervised NMF Results on the Brahms Recording: V (left), W and H (right)	39
4.6 Supervised NMF Results as Mixture of Piano and Noise Source of the Brahms Recording: Splitting W and H	39
4.7 NMF Restoration Pipeline	41
5.1 Spectrogram of Original Brahms Recording	49
5.2 Spectrogram of Supervised NMF Restoration (1 Noise Basis Vector)	51
5.3 Spectrogram of Current Supervised NMF Restoration with 1 Noise Basis Vector added (2 in Total)	53

5.4	Spectrogram of Current Semi-Supervised NMF Restoration, Learning the Noise Basis Vectors from Random Initialization	56
5.5	Spectrogram of Current Best NMF Restoration with L1 Penalty Term of $\gamma = 131,072$	57
5.6	Piano Activations Matrix of Current Best NMF Approach, Restricted to the Top-5 Activations Per Timestep (Before Excluding the Bottom Two Row-Vectors)	59
5.7	Best NMF Approach: Restricted to the Top-5 Piano Activations Per Timestep	60
5.8	Spectrogram of Initial Denoising Model Result	62
5.9	Spectrogram of Denoising Model Result, Trained on Data with a Damaged Piano Source	63
5.10	Spectrogram of Denoising Model Result, Trained on Data with a Damaged Piano Source and a Time-Stretched and Generated Noise Mix Source	65

Chapter 1

INTRODUCTION

Thomas Edison invented the phonograph: the first machine to record and reproduce audio. Knowing he had made a technology so unprecedented, he put it to use by sending his representatives on a tour to record famous people for marketing purposes [15]. In December of 1889, a representative recorded a classical composer all-time-great: Johannes Brahms. He was recorded playing a piano arrangement of his piece titled “Hungarian Dance No. 1 in G Minor” [4]. This is an important recording because it offers a priceless window through which to experience the earliest sound of a great musician. This recording was engraved on a hollow wax cylinder: the recording medium for the phonograph. Sadly, many early wax cylinders are damaged from years of natural deterioration from use and misuse. This recording was on one of those damaged cylinders, and by the time it was finally digitized it was nearly inaudible. A musicologist, Gregor Benko, wrote about this digitized recording stating “any musical value heard can be charitably described as the product of a pathological imagination” [4]. In this thesis we make approaches for a system to solve this problem.

The system’s goal is to automatically turn the Brahms piano recording into a higher quality one, and perhaps do this for any and many historic music recordings. The motivation for this is if we can receive better audio from poor-quality historic musical recordings like Brahms’, this will allow better or easier restoration of many historical recordings. Other benefits will be better historical education and/or appreciation of historical media. There are numerous collections of old recordings known and unknown[15] that with possibility could benefit.

There is arguably potential to restore music recordings of the present day. Poor quality audio is captured everyday due to resource limitations like poor-quality recording equipment or a poor recording environment. This leads to music-related fields in industry that can benefit from better music restoration. Music recognition, like used in popular applications like Shazam and SoundHound, is a technology that allows for song information to be returned to the user if they record the song being played in an unobtrusive enough surrounding. A poor recording environment is a big use case of this technology due to background noise that commonly covers up the music under consideration. Automatic music transcription is a technology in its infancy that receives musical audio, like music recognition, but it returns the sheet music (transcription) pertaining to the recording based on what can be inferred. Music restoration can fill this use case as a pre-processing step. Music production and other media can benefit because of "sampling": the use of snippets of old music tracks to create new and original music. Restoration could give a music producer the tool to make an old recording sound much better.

Established techniques in music restoration use meticulous and time-intensive methods of digital signal processing (DSP) [4]. These techniques haven't seen recent advances, but the modern emergence of big data and machine learning (ML) offers potential to take away the manual labor. ML is a paradigm in computing that is allowing software to solve problems that are intuitive for humans and have not been attempted to be solved by conventional programs out of sheer complexity [10].

ML is allowing software to retrieve more and more complex interpretations of input data. Given a useful data representation of a poor-quality audio recording, a machine learning technique can use it as input. Non-negative Matrix Factorization (NMF) is a relatively new machine learning technique. It lends itself well to physical data in which values are naturally non-negative (i.e. audio data). Other ML techniques like

Deep Learning, namely Deep Neural Networks (DNNs), have also shown promise in handling audio data such as with Recurrent Neural Networks (RNNs) which make use of sequential data like audio.

In this thesis we propose two approaches for restoring the damaged Brahms piano recording using machine learning techniques: NMF and a DNN. The remainder is structured as follows: a background section that will go over relevant topics of DSP and ML used in the implementation, related works describing how others have restored music audio, an overview of our implementation, evaluation of our results, and a conclusion.

Chapter 2

BACKGROUND

2.1 Sound and Digital Signal Processing

2.1.1 Sound, Music and Recording

Sound is the wave-like movement of air particles we can hear from the contact of these particles within our ears. Sounds can be defined by volume and pitch; volume is controlled by how far the air particles move (i.e. the amplitude) and pitch is controlled by how fast they move (the frequency). In musical pitches, the air particles move in coordination so that frequencies in the waveform remain consistent over time.

The origins of sound recording are connected with the application of the system in this thesis. Thomas Edison is known as the inventor of the first sound recording machine: the Phonograph. This device operates by facing an input apparatus towards the sound source(s) of interest. As an aside, a source means a body or process that generates a signal. For example, a piano is a source that can generate a composite piano signal, and a coin dropping on the floor is a source that generates the signal of a "clink" sound (even though unwanted if this is a special recording). The phonograph used the recording medium of a hollow wax cylinder, which is fed through a spinning axis. Being an analog recording device, it etches into the wax outer surface of the cylinder, making a natural representation of the signal. Developments in technology only made recordings higher quality but also more persistent as recording medium materials were refined. This eludes correctly that many early recordings, such as the Brahms, were permanently damaged, not only from misuse but also from too much

use. The extended use of the Brahms recording due to its popularity unfortunately damaged it more than other recordings of the same era.

The invention of digital recording and the digital medium offers persistence of data. Contrary to analog recording, digital recording doesn't capture an analog waveform but approximates it instead by taking samples of an incoming analog signal at small time intervals. This technique is called pulse-code modulation (PCM), and allows a continuous waveform to be represented in digital audio. This technique gives way to two errors: quantization error and aliasing. Quantization error is when a sampled value is not exactly the analog signal at that time, and this can be mended with higher-precision datatypes for samples. Another error is aliasing: sampling a signal that contains higher frequencies than what the sampling rate f_s can capture, resulting in a distorted representation. Aliasing can be prevented if the input signal only contains frequencies that are lower than $f_s/2$, where $f_s/2$ is known as the Nyquist frequency. If a frequency greater than or equal to $f_s/2$ is encountered during sampling, the peak amplitude of the waveform (the minimum defining feature of a frequency) won't be captured, and will lead to artifacts. The industry-standard sampling rate for a PCM signal is 44,100 Hz (commonly 44.1 kHz), yielding a Nyquist frequency comfortably above the common maximum hearing frequency of humans: 20,000 Hz. Stereo signals contain two channels, which in implementation are two signals - this makes for realistic listening for use-cases of two sound sources like when using headphones or speakers. Mono signals contain one channel and thus are one signal.

Broad types of noise in vintage recordings are: broad-band noise or impulsive noise [17]. Broad-band noise is observed as hissing throughout the recording, and can be solved with classical denoising methods. Impulsive noise is observed as random clicks or crackles which are artifacts of once-present parts of a recording medium gone missing from breakages, scratches, dust or dirt. This is solved by synthesizing material

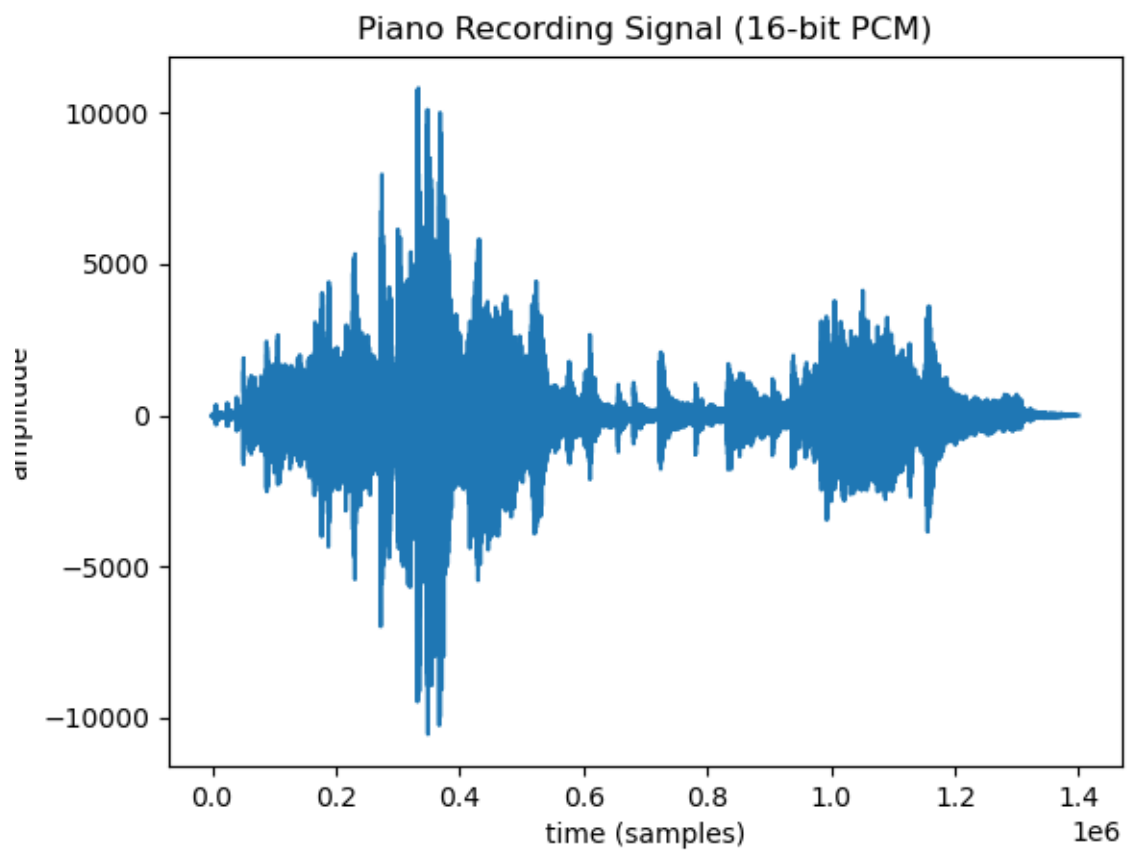


Figure 2.1: Waveform of a Piano Recording

to fill the missing gaps [17]. Solving impulsive noise moves into a controversial area that blurs the lines between restoration and dilution [15]; this is because synthesis of missing parts of an art-piece, like an audio signal, isn't technically the artist's (in our case Brahms') making. For this reason, we will refer to noise as broad-band noise, and ignore the impulsive noise problem.

2.1.2 Digital Signal Processing

Digital recording would be used in the field of digital signal processing (DSP). Digital signal processing is the portion of signal processing concerned with signals in the digital medium. Signal processing is a subfield in electrical engineering concerned with analyzing, modifying and producing signals such as sound, images and scientific measurements.

2.1.2.1 The Fourier Transform

The Fourier Transform (FT) is a commonplace operation in DSP. It allows a signal, which exists in the time-amplitude domain, to be transformed to the frequency-amplitude domain. Said differently, the characteristics of a signal in terms of its waveform over time are transformed directly into the characteristics of what frequencies exist in the waveform over said time. The more the frequencies vary within this time, the more inconclusive the result showing the active frequencies - this is why it is important that anything close to a stationary signal (a signal whose frequencies are constant through time) is used. If a non-stationary signal isn't available, a small enough time interval of the signal can approximate one well enough - this is given that the signal source characteristics are known well enough to deduce that the frequencies won't change in such a small amount of time. For a human-performed music source

signal this is doable, because there is a low enough limit to how fast a human can play/perceive a rhythm.

A PCM signal is transformed with a Discrete Fourier Transform (DFT). The Fast-Fourier Transform (FFT) is the algorithm of choice for calculating the DFT, and it requires a signal length that is a power of 2. If a signal's length is not a power of 2, it can be padded with zeros on either side until it is - this interpolates the frequencies and is harmless. The nature of the FFT makes its output complex-valued, inherently storing 2 characteristics of the signal's frequencies: the magnitudes and phases. One can use Euler's Identity to get these characteristics of the frequencies back to real values, but as artifact it gives both positive and negative frequencies. To combat this, the positive subset of their frequencies is taken. To get the magnitude of the frequencies, the absolute value of them is taken. To get the phases, the angle of them is taken.

The DFT size N , is the term for the length in samples in the DFT's input, which results in an N -length array. The output of the DFT is discrete, which means it approximates the continuous frequency spectrum by binning, or grouping values of similar frequencies into N "bins". The larger the DFT size, the longer amount of time is consumed, but the larger amount of frequency bins results. Thus, DFT size makes the compromise between time and frequency precision. The frequency precision or bin size f , can be calculated using the sampling rate of the signal f_s in the equation:

$$f = \frac{f_s}{N}$$

It's possible to transform the frequency-amplitude domain output of the FFT back into the time-frequency domain with the inverse FFT (iFFT). It takes in the FFT result and returns the original signal. To prepare the complex-valued input from

positive magnitudes and phases, they can each be concatenated with their respective mirrors, then joined back together into complex-values.

2.1.2.2 The Short-Time Fourier Transform

The FFT alone is useless if you want to know frequencies of a signal over time. This can be solved for by splitting the signal into small consecutive segments, and performing the FFT on and taking the positive frequencies of each. This is called the Short-Time Fourier Transform (STFT). This is valuable for a visual representation if the magnitudes are taken from each segment, called a spectrogram or magnitude spectrum, but also for an ML representation. It is biologically found that our inner-ears transform a signal waveform to a frequency-over-time representation for hearing, which gives clues to why the STFT is a valuable representation for ML. Figure 2.2 shows a spectrogram of the same piano recording in Figure 2.1. Also, the spectrograms shown in this paper only cover frequencies in the fundamental frequency range of piano notes (for visual ease).

Given that this STFT transformation has been modified by a ML algorithm, it is important that it can be transformed back to time-amplitude format to be heard. This is solved by taking the iFFT of each segment concatenated with its mirror, and connecting them back into a signal. This is called the inverse STFT (iSTFT).

An unwanted artifact from operating on segments of a signal can come from the fact that the frequencies are disrupted on each end of the segment where cuts are made. After manipulation with a ML algorithm, a segment will have different frequencies. This means in the later iSTFT, each segment won't be able to match its borders' frequencies with its neighbors' borders' frequencies, causing unpleasant sound distortion in the result. This is solved by the overlap-add method in which each positive

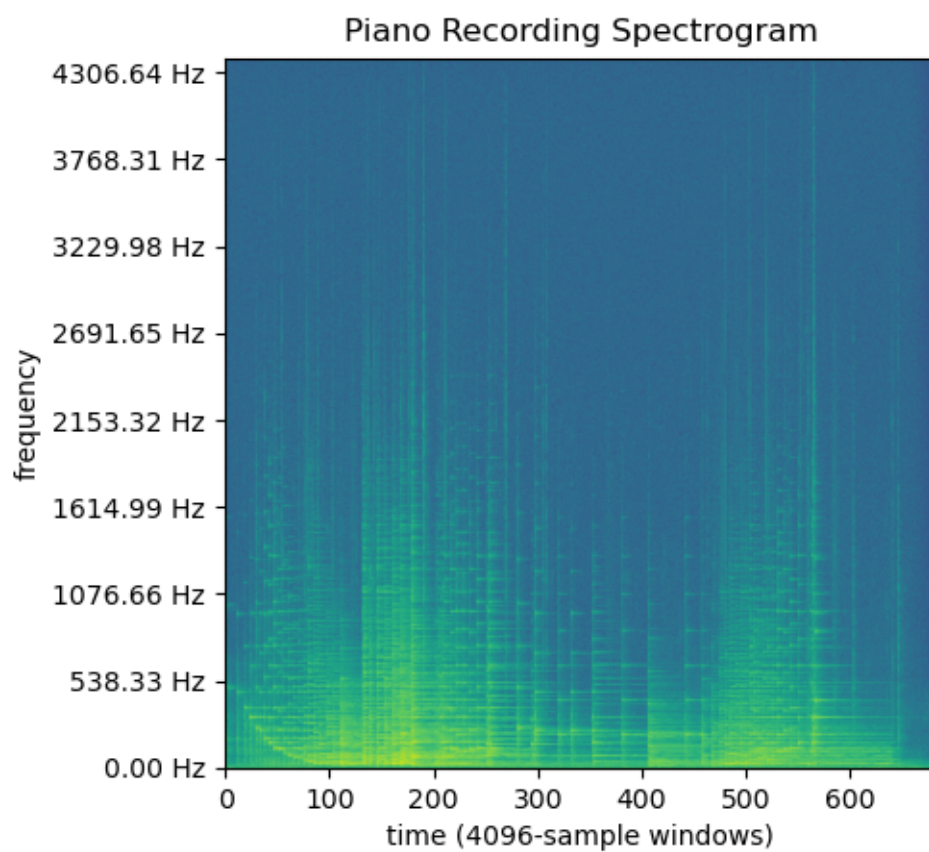


Figure 2.2: Spectrogram of a Piano Recording

magnitude segment is applied to a window function (e.g. Hanning) so each the frequencies of the segment reach 0 at each border. The window function also needs to leave the frequencies symmetric around the segment center, because the segments are then overlapped by half of their length with neighboring segments. The constructive interference of the frequencies inside each overlap, rebuild the frequencies back to their original magnitudes. Because the overlap-add method would shorten the signal length almost by half, it's crucial that the segments created in the STFT have enough redundancy so time isn't modified. This is done in the STFT with a hop size R , which controls how far a window is shifted on the signal before making the next segment. R in hand controls overlap-adding such that if R is half the length of a segment ($R = N/2$), the length of overlap needs to be half of a segment. By common default, $R = N/2$.

Filters are used in DSP, which change characteristics of a signal when the signal is passed through one. More specifically, filters can reduce or enhance any frequencies of the signal. Another word for filter is a mask, which is terminology commonly seen for filters which apply to frequencies variably at different points in time. As such, masks can be applied to time-frequency representations like spectrograms to control how much how much of a frequency is present at each timestep.

2.2 Machine Learning

Machine learning (ML) is a subset of the practice of Artificial Intelligence (AI): which today boils down to automating tasks that require "intelligence". ML is understood well when contrasted with the rest of AI which can be called Symbolic AI. Symbolic AI describes when domain-knowledge is hard-coded into a program, letting it accomplish what a human expert in this knowledge domain can do. On the other hand, ML relies

on a substantial sample size of data within a knowledge domain, and uses learning methods to derive the domain-knowledge. Of the two, symbolic AI systems were the first created but are still powerful in specific tasks. ML is powerful in broader tasks, in knowledge domains that are intuitive to us but the inner-workings are still unsolved [10].

The recent emergence of big data and advances in hardware allowing storage and processing of it, has made for rapid advances in ML. In ML the goal is to "train" a model, or let a model "learn" on a substantial amount of data called training data, until it converges. A training data needs to be as much as possible, which helps it be as representative of the population as possible. ML models also have some form of loss function and optimizer. The loss function tells the model how good of an approximation it makes, or how far off it is from convergence (having error at or very close to the global minimum of the loss function). The optimizer's goal is to tweak the model's parameters until it converges, at which point the model can be used on unseen data like symbolic AI.

Two categories of ML are used in this thesis: supervised learning and unsupervised learning. In unsupervised learning, the model comes up with novel interpretations of the data fed into it. In supervised learning, training data is annotated, as in each sample is paired with its ground-truth or target output (the result of some unknown function). Thus, supervised learning allows the programmer control over a model to approximate some function.

2.2.1 Non-Negative Matrix Factorization

Non-Negative Matrix Factorization (NMF) is an ML algorithm developed by Lee and Seung in 1999 [14]. It is an unsupervised algorithm similar to Principal Component

Analysis (PCA), and it is a matrix factorization algorithm like Singular Value Decomposition (SVD). What sets NMF apart from similar methods is its non-negative constraint which means that the input data to NMF cannot hold negative values, and its output is also non-negative. In this way, NMF can transform a dataset into not only a simpler one, but importantly an additive and parts-based representation. This makes it an intuitive choice for dealing with non-negative data captured in the “natural world” like audio, images, etc. It has many uses, including denoising audio, topic extraction in text mining, clustering gene expressions, and general-case dimensionality reduction.

How NMF works is that for an input matrix $V \in \mathbb{R}_+^{m \times n}$, it tries to find two low-rank matrices $W \in \mathbb{R}_+^{m \times k}$ and $H \in \mathbb{R}_+^{k \times n}$ such that $V \approx WH$. This makes W and H the approximate factors to input matrix V , hence the “matrix factorization.” Given a column-vector $h \in H$, Wh is a linear transformation that approximates column-vector $v \in V$, so NMF can be called a linear function approximator. The variable k (called the “rank”) dictates how much dimensionality reduction to perform, the smaller it gets. Dimensionality reduction maps input features into fewer ones, which is helpful for simplifying data to perform good approximations. Typically, $k < \min(m, n)$ for NMF to give meaningful results.

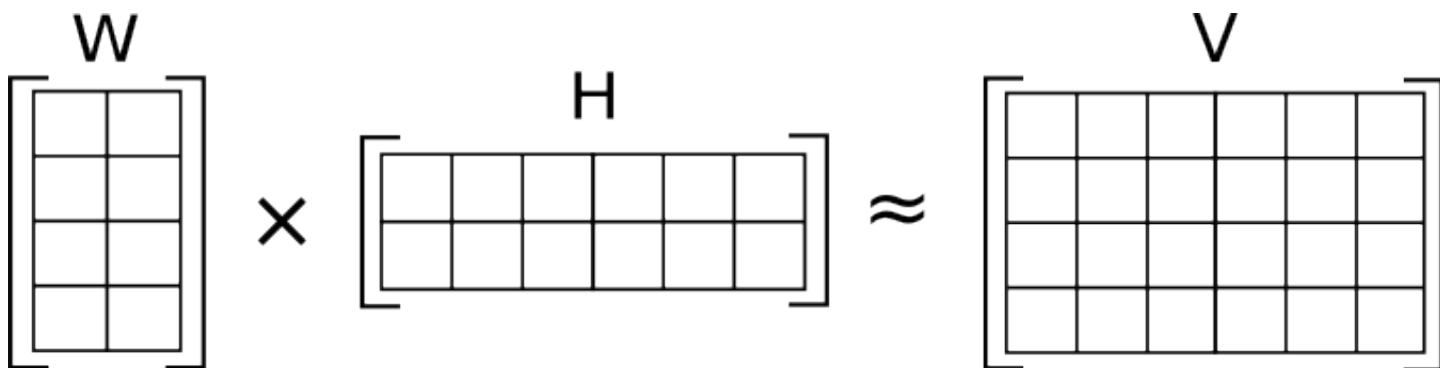


Figure 2.3: Non-Negative Matrix Factorization

As an important linear algebra refresher, a matrix's "rank" means the number of basis vectors needed to create it. This means each vector (row/column) in a rank- k matrix is made up of a linear combination of k basis vectors. Basis vectors are defined as the set of linearly independent vectors that span a vector space. For example, basis vectors $i = [1, 0]$ and $j = [0, 1]$ span the length-2 vector space, and any vector in this space can be made with a linear combination of i and j (e.g. $a = [2, -0.5] = 2i - 0.5j$).

Most NMF implementations work in this way: it begins by randomly initializing W and H , and then iterating over a number of alternating updates (AUs) to W and H , for example W is updated with a fixed H , then H is updated with a fixed W , and repeat. This goes on until their matrix product "converges" or makes a good enough approximation to V .

Because the NMF problem is non-convex, multiplicative updates (MUs) are used which implicitly minimize a loss function to yield a close approximation to V . It is commonly known that MUs can be derived from the Kullback-Leibler (KL) Divergence $D_{KL}(V\|WH)$:

$$D_{KL}(V\|WH) = \sum_{i,j} V_{ij} \log \frac{V_{ij}}{(WH)_{ij}} - \sum_{i,j} (WH)_{ij} + \sum_{i,j} V_{ij}$$

which can be treated as the NMF loss function. The MUs are below (where \odot is element-wise multiplication), where l denotes the current update we are on:

$$H^{(l+1)} \leftarrow H^l \odot \frac{W^T \frac{V}{WH^l}}{W^T \mathbf{1}}$$

$$W^{(l+1)} \leftarrow W^l \odot \frac{\frac{V}{W^l H} H^T}{\mathbf{1} H^T}$$

You can see these formulas in lines 3 & 4 of algorithm 1: the algorithm for NMF with MUs derived from KL Divergence, called KL-NMF.

Algorithm 1 KL-NMF

Require: $1 \in \{1\}^{m \times n}$, V

- 1: Initialize W, H
 - 2: **while** W, H not converged **do**
 - 3: $W \leftarrow W \odot \frac{\frac{V}{W^T H} H^T}{1 H^T}$
 - 4: $H \leftarrow H \odot \frac{W^T \frac{V}{W H^T}}{W^T 1}$
 - 5: **end while**
 - 6: **return** W, H
-

The MUs in NMF are sensitive to the initial values of W and H , not working well when they are close to 0. So values are initialized to the range between 0 and 1, and added 1.

2.2.2 Deep Learning and Artificial Neural Networks

Deep learning is a category of machine learning that is characterized by learning successive representations of data that are increasingly more meaningful. Most ML models rely on appropriate representations (features) of input data before it can be fed in - this task exists in a field called feature engineering. What makes deep learning shine is that it doesn't need feature engineering, but instead makes useful representations out of data and then maps it to an output in one go.

In a deep learning model, a representation of data is called a "layer" which is simply the result of an operation on the data. What makes a deep learning model "deep" is the high number of layers and thus high number of successive operations on the input, exactly like a large nested function. The input data is known as the first layer. Besides the output layer, all others are called hidden layers because they contain values not given in the input data. The size of a hidden layer is known as the hidden units. If

the hidden units of a layer are less than the size of the input layer (the number of input features), dimensionality reduction is automatically performed which maps the features into newer ones which are fewer in number.

2.2.2.1 Artificial Neural Networks

Artificial neural networks (ANNs), or neural networks (NNs), is the specific algorithm used in most deep learning. In NNs, input x is a vector of features that constitutes a sample of data, and the goal is to approximate some function $f^*(x)$, with help of learned parameters θ . This makes $f(x, \theta)$ to output y . NNs also consist of layers, where each layer is a non-linear function - specifically an affine transformation controlled by learned parameters W and b , followed by a fixed non-linear transformation called an activation function σ . This is why NNs are called non-linear function approximators. Non-linear functions allow more representation possibilities, which gives NNs the edge over related linear models like linear regression. The equation for the l th layer of an NN is below, in terms of the layer's activation h :

$$h^l = \sigma(W h^{l-1} + b)$$

where matrix W is called the weights and vector b is called the bias of the layer, and count towards θ . If $l = 2$, replace h^{l-1} with x . We will refer to these layer types as densely-connected or dense layers. The name "NN" is loosely inspired by neuroscience in the way that each layer is analogous to many neurons, which act at once by taking in inputs from many other neurons and each firing off their own "activation" value.

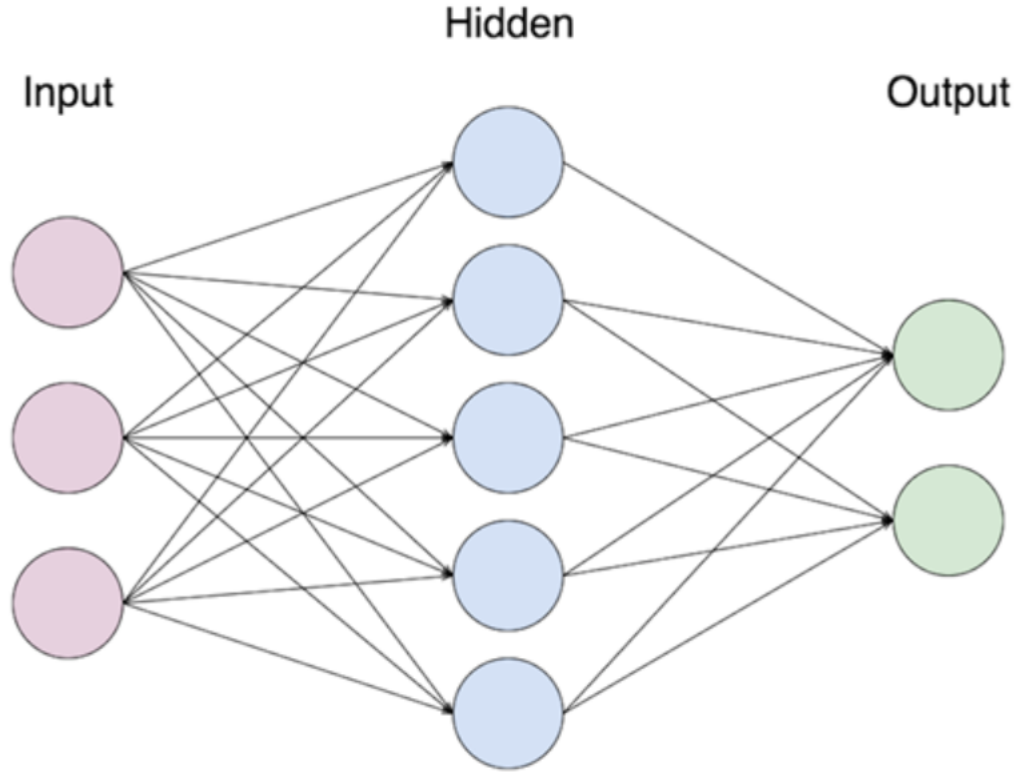


Figure 2.4: Simple Neural Network Diagram

2.2.2.2 Recurrent Neural Networks and Deep Learning

Recurrent neural networks (RNNs) are a type of NN that are designed to handle sequential data x_1, \dots, x_T . Thus x_t now corresponds to a feature vector at a point t in a sequence (often time). Most RNNs can also process sequences of variable length, and are designed to handle long sequences standard DNNs cannot handle. Its ability to handle long sequences comes from sharing the parameters θ across the model, which manifests itself in a single hidden layer and thus a single representation of the data sequence. Sometimes it is helpful to visualize an RNN not as composed of one hidden layer but instead many hidden layers, where each one in reality is the same hidden layer at a specific point on the input sequence. This is known as "unrolling" the computational graph of an RNN through time and drawing each data point in sequence like a unique sample, which makes it appear similar to a DNN.

Appropriately, x_t is processed in order of place in the sequence. This leads to a major factor in RNNs: the computation of activation h depends not only on x_t but on all data encountered so far. This means the data flow in this algorithm is cyclic. The equation for an RNN at the t th timestep is below, in terms of the it's activation h :

$$h_t = \sigma(Uh_{t-1} + Wx_t + b)$$

where U and W are the weights for the hidden-hidden connections and input-hidden connections, respectively.

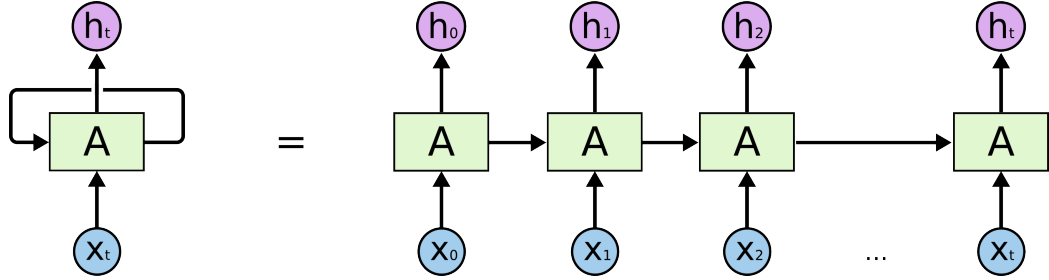


Figure 2.5: Recurrent Neural Network A (left) Unrolled in Time (right) [2]

Deep neural networks (DNNs), is simply a NN algorithm with many layers, which unlocks the power of deep learning. Because RNNs alone process a sequence once, they only create a single representation of it. They lack the ability to process a point in sequence more than once and create hierarchical representations of a sequence. For this reason, a deep recurrent neural network (DRNN) can be made by including RNNs as layers in a DNN. The equation for the l th layer of a DRNN, an RNN, is below in terms of the layer's activation h :

$$h_t^l = \sigma(U^l h_{t-1}^l + W^l h_t^{l-1} + b^l)$$

where U and W are the weights for the recurrent connections and hidden-hidden layer connections, respectively. We will refer to these layer types as recurrent layers.

2.2.2.3 Gradient-Based Optimization and Backpropagation

During the supervised training of a DNN, it is driven to match $f^*(x)$ to $f(x)$. This is known as the learning or training process, which happens by updating the parameters of the DNN. For this to happen, each input x is paired with its target answer $y \approx f^*(x)$; after x passes through the DNN we get $f(x)$ (or \hat{y}), which are used in the loss function. The loss function quantifies how far off prediction \hat{y} is from expected output y . Like the training process of other ML methods, an optimizer algorithm is used, whose goal is minimize the loss function by updating the parameters θ of the model.

What sets DNNs apart from other ML models is they contain non-linear transformations. This makes most loss functions become non-convex, meaning DNN models are learned with iterative gradient-based optimizers. These aren't guaranteed to drive function $f(x)$ to match $f^*(x)$ (find the global minimum of the loss function), but instead find an approximation to $f^*(x)$ (find a local minimum). Thus, DNNs are called parametric function approximation algorithms.

Loss functions need to be differentiable, so their gradients can be computed. A gradient is the term for the derivative of a multi-variable function, which tells the direction of steepest ascent for the input. Thus the negative gradient of the loss function is used, to travel in the direction of steepest descent, which minimizes the loss function. To descend the gradient, we need to change the network parameters θ in proportion to the loss function output. Gradient-descent is sensitive to the initial values of θ , which are normally initialized to small random numbers.

Gradients are computed with the backpropagation algorithm. It is needed to find the loss function as not a function of the predictions \hat{y} and targets y but as a function of the networks parameters θ . It does this with the chain-rule, as the algorithm backpropogates through the network. Thus the backpropagation algorithm returns the gradient of the loss function with respect to the parameters θ of the network.

Stochastic gradient descent is a foundational optimization algorithm, but there exist more sophisticated ones like Adam, which include the concept of momentum when descending. Computing the gradient for an RNN is no different than computing it for a DNN, because the algorithm unrolls the RNN's computation graph. Then the gradients can be used like those of DNNs by the optimization algorithm.

The averaged changes to each of the parameters θ , averaged over all training samples, is the true negative gradient of the loss function. But for computational efficiency, the optimizer works repeatedly on portions of the train data called mini-batches. This means that the mini-batch size in addition to number of features (including sequence length for RNNs), defines the width of the network, which combined with depth in layers can measure its size.

2.2.3 Training Practice

Parallel processing n-D matrices of large NNs drives high demand on computer hardware. This is why GPUs are generally a hardware requirement for training DNNs.

Common training practice in deep learning and ML is to separate data into a train set and test set. The train set is training data, and the test set is used to test the trained model. In supervised learning, the train set can be divided into a train set and validation set. The validation set is used like the test set but since it is annotated, it can give a performance measure of how well the model approximates

outputs. Also during training, we can get a performance measure on the train set after each parameter update. Performance measures on both train and validation sets diagnose overfitting, which is when the model trains too specifically to the train set, leading to poor performance on anything but the train set. Overfitting shows when validation error is much higher than train error. DNNs can be built in ways to combat overfitting, and help generalize the model.

Building DNNs in different ways translates to picking hyperparameters that define the model architecture and training process. They are called hyperparameters to differentiate from the parameters being learned inside of the DNN model; so they are the term for parameters of a DNN program.

It usually isn't apparent what hyperparameters to pick for a model. Good practitioners can choose these mostly based on theory or convention formed by repeated empirical results. Otherwise, experimentation is required which involves testing possible combinations of them. Due to computational intensity making for low turn-around time of DNN training, quick experimentation can be unfeasible. Instead, a large experiment containing many combinations is set off in a single hyperparameter-tuning search, which can run anywhere from hours to weeks long.

Chapter 3

RELATED WORKS

This thesis work falls into the area of machine learning applied to music signal processing, a subfield of DSP. In this section, we will cover related works on music restoration and surrounding work in music signal processing. This will include ML's role, because ML is allowing the computational power for advancements in this field among many others.

Related works in music restoration boil down to a source separation task called denoising. Source separation is the act of, given an input signal that is a mix of signals produced by different sources, separating and outputting the signals corresponding to each source. A mixture is a signal that is the sum of other signals. To denoise music, we want to separate two sources: the target music source, and the noise source that interferes with the music.

In music denoising of a recording, the noise source is generally unknown because it is made up of random factors of in recording environment or technology. This rules out these source separation methods: microphone arrays and adaptive signal processing because they require knowing all sources, and independent component analysis because it requires a recording for each source [23].

3.1 Classical Restoration Methods

Classical restoration/denoising methods revolve around attenuating noisy parts of a signal's frequency spectrum. Some methods are wiener filtering, spectral subtraction,

and minimum mean square error short time spectral attenuation (MMSE STSA) [23]. One of the earliest classical restorations of a musical recording was in 1975, by applying digital techniques to deconvolve the effects of horn on wax cylinder recordings of Enrico Caruso from 1907 [17]. In 1994, DSP was used in a challenging music denoising task by Berger, Coifman and Goldberg at the Center for Computer Research in Music and Acoustics (CCRMA) at Stanford [4]. That piece of audio is the same that we use in our system: Johannes Brahms’ personal piano performance of his piece titled ”Hungarian Dance No. 1”. The algorithm chooses an optimal basis for the Brahms signal from a library of local trigonometric and wavelet packet bases. After choosing the optimal basis, the algorithm splits the signal into clean source and noisy source, then recurses on the noisy source. The end result is the sum of the clean parts. This method’s shortcomings were remaining impulsive noise, and that it worked unevenly in regions of the signal with high and low activity. The results of this system are the best restoration of this recording, and act as a benchmark.

3.2 Machine Learning Restoration Methods

The work described so far is a manual process, and could benefit from automation by ML. ML also has the advantage to model non-stationary noise, as opposed to previous classical denoising approaches which assume stationary noise [19].

3.2.1 Methods with Non-Negative Matrix Factorization

NMF’s usage on music audio data was pioneered by Smaragdis et al. in 2003 for automatic music transcription [21]. Since then, NMF has been applicable for simple and effective music source separation, specifically denoising. This follows from NMF’s non-negativity constraint, which allows a spectrogram to be modeled as a mixture of

prototypical spectra [23]; each of which are the frequency spectrum of a unique music note, which together make building blocks of music audio.

For source separation applications, $V \in \mathbb{R}_+^{f \times t}$ is the input spectrogram of the audio of concern. Thus V is made up of t time segments of the recording, each containing f bins or features making up the frequency spectra. $W \in \mathbb{R}_+^{f \times k}$ is the matrix of basis vectors or dictionary elements, which are vectors that each encode the frequency spectrum of an elementary source in the recording. $H \in \mathbb{R}_+^{k \times t}$ is the matrix of activations, weights or gains, which contains column-vectors that denote how much of which of each basis vector is being used (activated) in the current time segment. From another view, H contains row vectors that each denote how much of its corresponding basis vector is being used (activated) in each time segment. This means (where $0 < i < k$), basis vector w_i multiplied by activations row vector h_i^T gives a spectrogram of one audio source. Typically $k < f < t$ [23]. This can be seen in Figure 3.1.

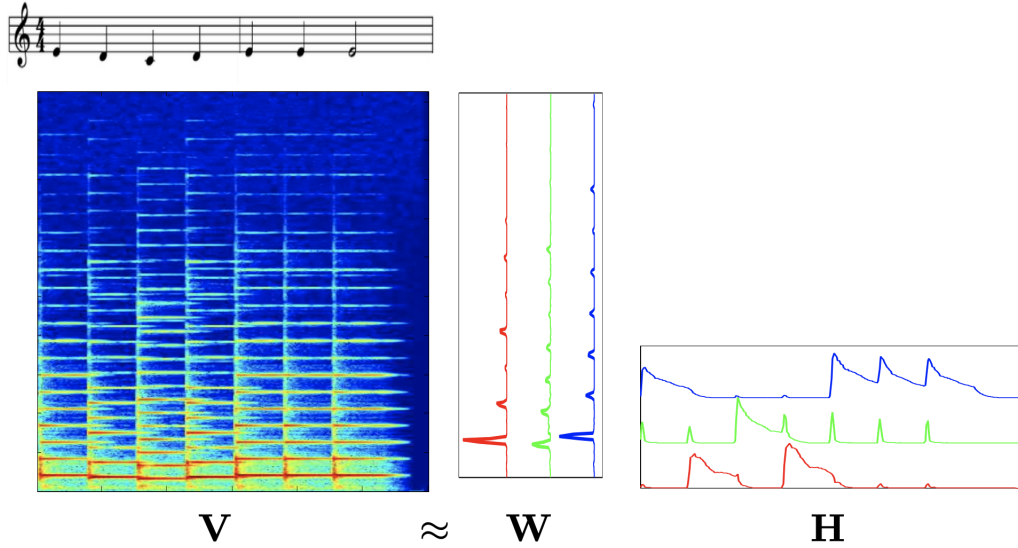


Figure 3.1: NMF Modeling a Recording of Mary Had a Little Lamb [23]

Denoising is done by picking the subset of basis vectors W_s and activations row-vectors H_s corresponding to the target music source, then multiplying them together to get \hat{V}_s which approximates the target source magnitudes. To enforce that the sources

summed together equals the sum of the mixture, a time-frequency (T-F) mask M_s is made from \hat{V}_s , which is applied to the input mixture V . Thus, \tilde{V} is the masked V and is the answer. T-F masks, specifically soft masks, denote how much of a target source is prevalent in a T-F unit of the T-F representation, or spectrogram, of a signal [24]. The mask M_s shown previously is a soft mask which is made by dividing \hat{V}_s by $\hat{V} = WH$. M_s is applied by element-wise multiplying M_s with V . This process, including the pre and post processing of audio data, is illustrated in Figure 3.2.

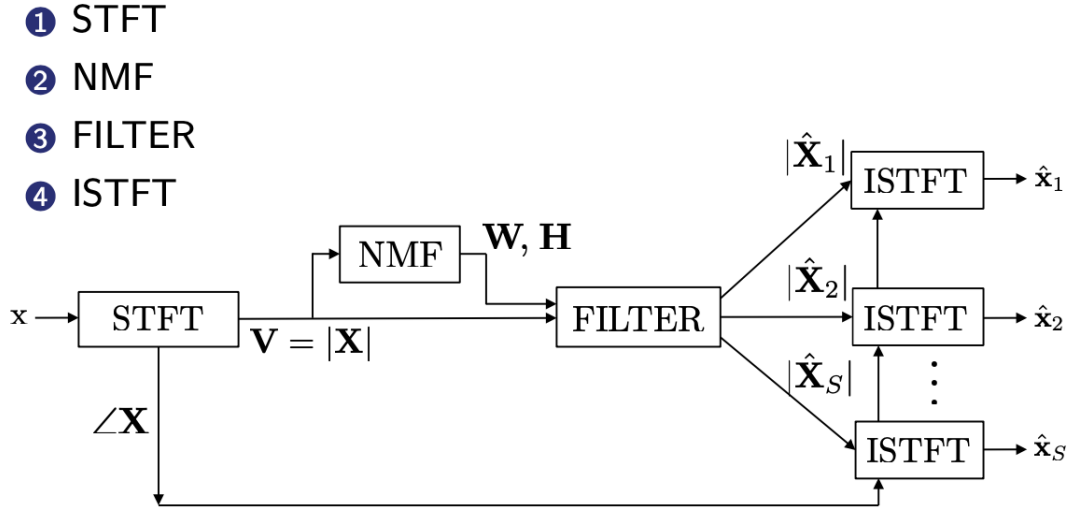


Figure 3.2: NMF Source Separation Pipeline by Sun and Bryan [23]

In real music audio, a source rarely consists of a single NMF component (e.g. a piano source requires 88 components for notes from 88 piano keys). This calls for incorporating prior information, done by Shoji et al. [9], which makes so-called supervised NMF in which sources are known prior and encoded into W , which could allow only H to be learned while W remains fixed. Pre-specified W are made by learning their basis vectors from isolated sounds. They also termed semi-supervised NMF, in which some sources known prior and encoded into a subset of W , meaning the subset is

fixed while the rest of W is learned with H . The MU for W in the semi-supervised case are below, where only a *subset* is updated:

$$W^{(l+1)}[:, subset] \leftarrow W^l[:, subset] \odot \frac{\frac{V}{W^l H} H[subset, :]^T}{1 H[subset, :]^T}$$

Another variation is penalized NMF introduced by Févotte and Idier [8], where W and/or H are regularized. Regularization is an ML practice in which a model's weight values are forced to take only small values. They specifically introduce l_1 -norm regularization (l_1 -penalization), which drives values to near-exactly zero. The KL-NMF loss function with an added l_1 -penalty on H is shown below:

$$\min_H D_{KL}(V \| WH) + \lambda \|H\|_1$$

where λ is a tuning parameter. If $\lambda = 0$, H is not penalized. For l_1 -penalization, as λ grows H will be forced to become smaller until it eventually contains all zeros, because that will provide the lesser total loss in training. This translates to λ added to the denominator in the MU for H :

$$H^{(l+1)} \leftarrow H^l \odot \frac{W^T \frac{V}{WH^l}}{W^T 1 + \lambda}$$

NMF works in music denoising include Cabras et al. 2010 [5], who used a semi-supervised approach in which noise source activations are learned by applying a statistical model to the input mixture, which are used to learn noise basis vectors. They also use a Bayesian suppression rule to make a T-F soft mask from the noise source approximation and the input mixture, to make the music source mask. Later in 2016, Canadas-Quesada et al. [6] used a supervised, score-informed NMF approach. The

musical score of the piece in the recording is used to inform the learning of the activations, specifically by only allowing possible combinations of basis vectors at each time step. Both piano basis vectors were prior-learned: the piano from recordings of isolated acoustic piano notes, and noise from snippets of recordings with common types of vinyl noise.

3.2.2 Methods with Deep Learning

Deep learning (DL) is said to have solved audio enhancement/denoising tasks previously addressed by NMF [19]. It made advances in many areas of music signal processing. A jointly-trained NN can perform polyphonic music transcription [11]. A "context encoder" has been created for audio inpainting [18], a convolutional neural network (CNN) has performed audio super-resolution [13], and an RNN can recover uncompressed audio from a low-bitrate encoding [7].

DL models designed for supervised source separation, and specifically audio enhancement/denoising, fit our use case appropriately. Supervised source separation was inspired by the concept of T-F masking in computational auditory scene analysis [24]. These models are categorized into mapping and masking-based approaches: mapping-based output the frequency spectra of a source, while masking-based output a T-F mask that when applied to the input mixture yields the frequency spectra of a source.

Huang et. al [12] were among the first to introduce DL for the task of source separation. Their model is a DRNN in order to exploit time-series data, and learn successive representations of a mixture containing 1 or more sources. They soon extended the application of their model to speech denoising, getting substantial results. For denoising, this model considers the input as a mixture of a target source and a noisy

source. Thus at time t , the input x_t is a vector of audio features (i.e. timesteps of a spectrogram) of the window corresponding to t of the degraded mixture. The two output targets $y_{1t} \in \mathbb{R}^F$ and $y_{2t} \in \mathbb{R}^F$ and the two output predictions \hat{y}_{1t} and \hat{y}_{2t} of the model are the magnitude spectra of the different sources, where F is the number of features. They synthesized training data for the denoising task by mixing wanted vocal samples with unwanted environmental noise like airport terminal, subway station, or drilling ambience.

Further, this model mixes mapping and masking approaches by jointly training with a fixed T-F mask as the output layers. Thus, previous predictions \hat{y}_{1t} and \hat{y}_{2t} pass through it. The T-F masking layers contains no weights, but the weights in remaining layers are optimized for the error metric between mask function outputs \tilde{y}_{1t} , \tilde{y}_{2t} and y_{1t} , y_{2t} . The mask is a soft mask, which also enforces the constraint that the sum of the model's predicted sources is equal to the sum of the input mixture. Thus, this mask works the same as the one in NMF source separation. The masking layers are as follows:

$$\tilde{y}_{1t} = \frac{|\hat{y}_{1t}|}{|\hat{y}_{1t}| + |\hat{y}_{2t}|} \odot x_t \quad \tilde{y}_{2t} = \frac{|\hat{y}_{2t}|}{|\hat{y}_{1t}| + |\hat{y}_{2t}|} \odot x_t \quad (3.1)$$

The model architecture is one-input two-outputs. The input layer is followed by one or more recurrent layers, followed by a fork by which the last recurrent layer's output is shared to two densely-connected layers. Lastly, each densely-connected layer is followed by a T-F masking layer.

The l th recurrent hidden layer's activation for time t , h_t^l , is defined as:

$$h_t^l = \text{relu}(U^l h_{t-1}^l + W^l h_t^{l-1})$$

and the dense l th hidden layer’s activation h_t^l is defined as:

$$h_t^l = \text{relu} (W^l h_t^{l-1})$$

where relu is the rectified linear unit activation function ReLU, W^l is the weight matrix for layer l , and U^l is the recurrent weight matrix for layer l . The diagram for this model is below:

The model’s loss function allows it to discriminate between predicted sources. It aims to minimize the difference between the prediction and target spectra of each source, but also to enforce a difference between the sources’ predictions. This amount of enforcement is tuned with a constant γ in the loss function. Below is the loss function:

$$\frac{1}{2} \sum_{t=1}^T (\|y_{1t} - \tilde{y}_{1t}\|^2 + \|y_{2t} - \tilde{y}_{2t}\|^2 - \gamma \|y_{1t} - \tilde{y}_{2t}\|^2 - \gamma \|y_{2t} - \tilde{y}_{1t}\|^2)$$

Hyperparameter Name	Value
Features	magnitude spectra
DFT size N (of features)	1024
# Hidden (Recurrent) Layers	2
# Hidden Units per Hidden Layer	1000
Input Normalization	false

Table 3.1: Optimal Hyperparameters for Speech Denoising Setting by Huang et al. [12]

In terms of other related works, it is unclear which DL architecture is superior in the audio enhancement/denoising domain: CNNs, RNNs or convolutional recurrent neural networks (CRNNs), because different research groups have claimed state-of-the-art results with different models [19]. In this domain, use of bidirectional recurrent layers, and residual connections (skip-connections) are shown to increase performance as well [24].

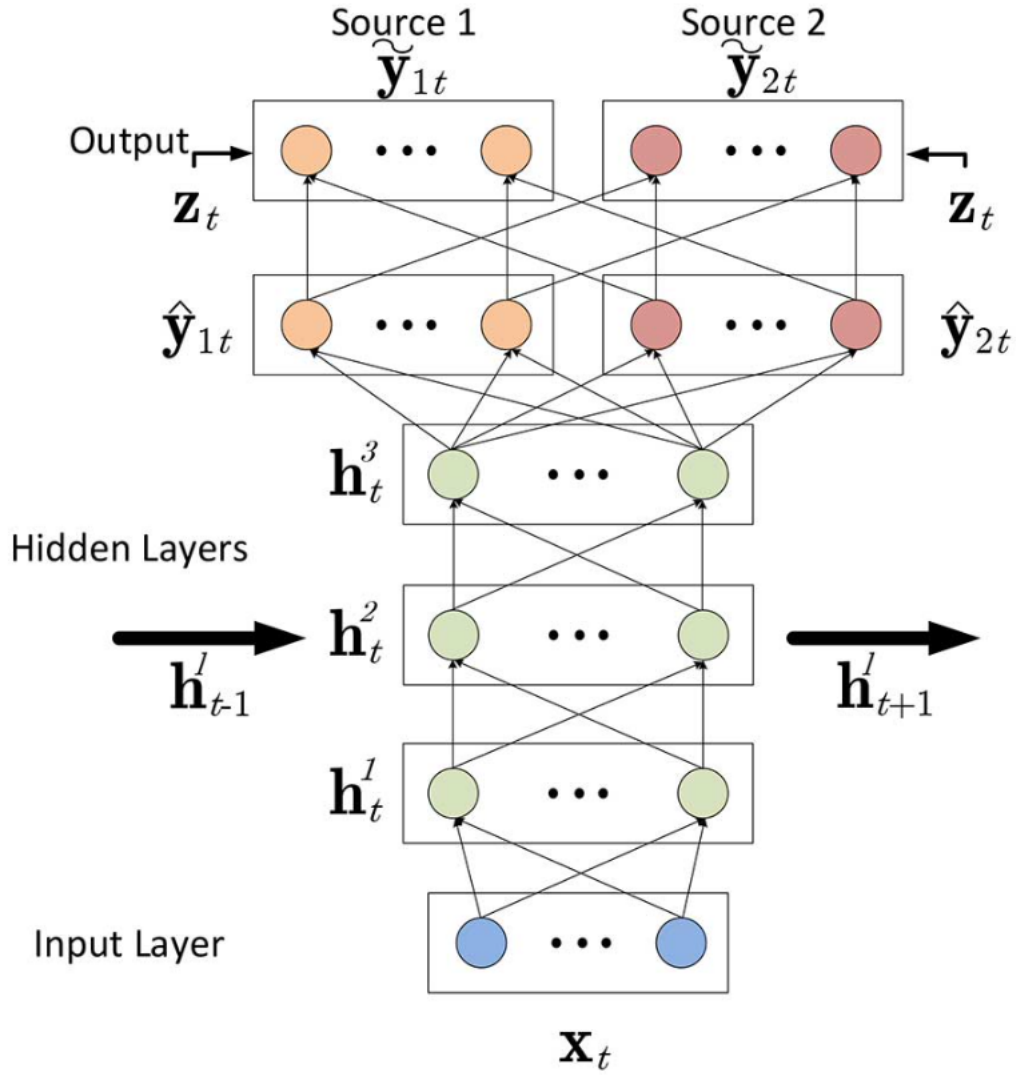


Figure 3.3: DL Source Separation Model Architecture by Huang et al. [12]

The residual connection layer combines the output of the recurrent-unit with the input. The residual connection layer’s activation h_t^3 is defined as:

$$h_t^3 = h_t^2 + x_t$$

Open-Unmix is an open-source near state-of-the-art music source separation model, released for research purposes by Stöter et al. [22]. This model has similarities to Huang et al. in how it incorporates a DRNN and a T-F mask, but is more involved with hyperparameters, including residual connection after the recurrent layers of the network, a fully-connected layer with a tanh activation as the first hidden layer and input layer scaling/output layer unscaling and batch normalization.

Chapter 4

IMPLEMENTATION

The following section describes the implementation details of our restoration approaches. Each approach uses basic DSP and an ML method as its core technology: either NMF, or Deep Learning.

4.1 Pre and Post Processing Audio Data

Both of our approaches share use of DSP to pre and post process audio data. Signals are read in from and written out to WAV files using the input and output API of the SciPy Python library.

On read in, signal values and datatypes are converted to that of the WAV format of the largest value range we deal with: 16-bit integer PCM, which encodes signals with the value range $[-32,768, 32767]$ (The Brahms recording WAV format is 8-bit integer PCM, with a range of $[0, 255]$, so its values are scaled up to the range of 16-bit integer PCM). Then, stereo signals are converted to mono by averaging the 2 channels together.

Then signals are transformed into spectrograms using DFT size $N = 4096$ and the hanning window function. $N = 4096$ was a good compromise because it guarantees to capture all rhythms as small as quarter-note triplets, and all notes as low as G3 into unique frequency bins, which are the majority of rhythms and notes in the recording. As an additional step, for spectrograms fed into the deep learning model, their values are converted from 64-bit to 32-bit floating point decimals for performance.

Post-processing converts the model result spectrogram into a waveform which is written back into hearable WAV format. If a signal’s original format isn’t int-16 integer PCM, its values are converted to its original format’s before write out to WAV. Remember that spectrograms don’t include the phase information. This means in order for post-processing, we preserved the positive phases of the Brahms recording input, and combined them with the model result spectrogram (positive magnitudes) to perform an iSTFT and retrieve a waveform.

4.2 Non-negative Matrix Factorization Approach

We began our NMF approach framed as a denoising source separation task with supervised NMF. In this way, if we can put prior information into the basis vectors that denote the make-up of wanted and/or unwanted sources, with appropriate use of the basis vectors we can pick out the wanted source. In our case, the wanted source is Brahms’ piano and the unwanted source is the noise that interferes with it. Like done by Canadas-Quesada [6], the basis vectors used for Brahms’ piano $W_p = W_s \subseteq W$ are ”prior-learned” from isolated, high-quality recordings and are associated with the source of interest. These offer hope of not only denoising, but improving the recording with high quality basis vectors for a stronger restoration. ”Prior-learned” is the term we will use for basis vectors which are learned prior to their use in supervised NMF for source separation of the Brahms recording. The following sections go into the first steps which are how the piano and noise basis vectors are prior-learned.

4.2.1 Learning Piano Basis Vectors

We learned 88 piano basis vectors, for the 88 possible notes played on a piano. High-quality isolated piano note recordings, 16-bit PCM signals sampled at 44.1 kHz, were

taken from the University of Iowa Electronic Music Studios [3]. These recordings were labeled by volume level, so the ones at loudest volume were taken. After reading in a recording signal, the silence on the either end is trimmed off using a threshold of 1% of its maximum waveform amplitude. Then after transforming the signal into a spectrogram, we average across the time segments to create the averaged frequency magnitudes of the piano note - this is the basis vector for that piano note.

Unlike the samples in [6], the Brahms recording departs from the score because of his improvisation, so we can't attempt a score-informed approach. Yet, we still thought to use the score by only learning piano basis vectors for the piano notes (in all octaves) appearing in the score. Although Brahms departed from the score, it is unlikely he added notes outside of its key: G Minor. G minor leaves out notes of B and Ab, yielding 73 notes (in all octaves) for 73 piano basis vectors total. Only necessary notes minimizes the chances of piano basis vectors representing noise.

In hopes of better picking up Brahms' piano again, we made a degraded copy of the piano basis vectors. They were degraded in order to emulate the muffled sound of Brahms' piano, much like how musical sources sound in other vintage recordings. This was done by passing the piano basis vectors through a filter which removed frequencies less than 400 Hz and more than 3,000 Hz, and enhanced the upper-end of the remaining frequencies by a small amount. It was found that removing these frequencies, with some artistic tweaks, achieved this effect. This was done in the free digital audio workstation, Audacity, and the filter in full-detail is shown in Figure 4.2. The proposed method for using the degraded piano basis vectors is only to use them within the updates of NMF to learn the piano activations. Once the piano activations are made, we then use the corresponding high-quality piano basis vectors in matrix multiplication with the piano activations for the wanted source synthesis.

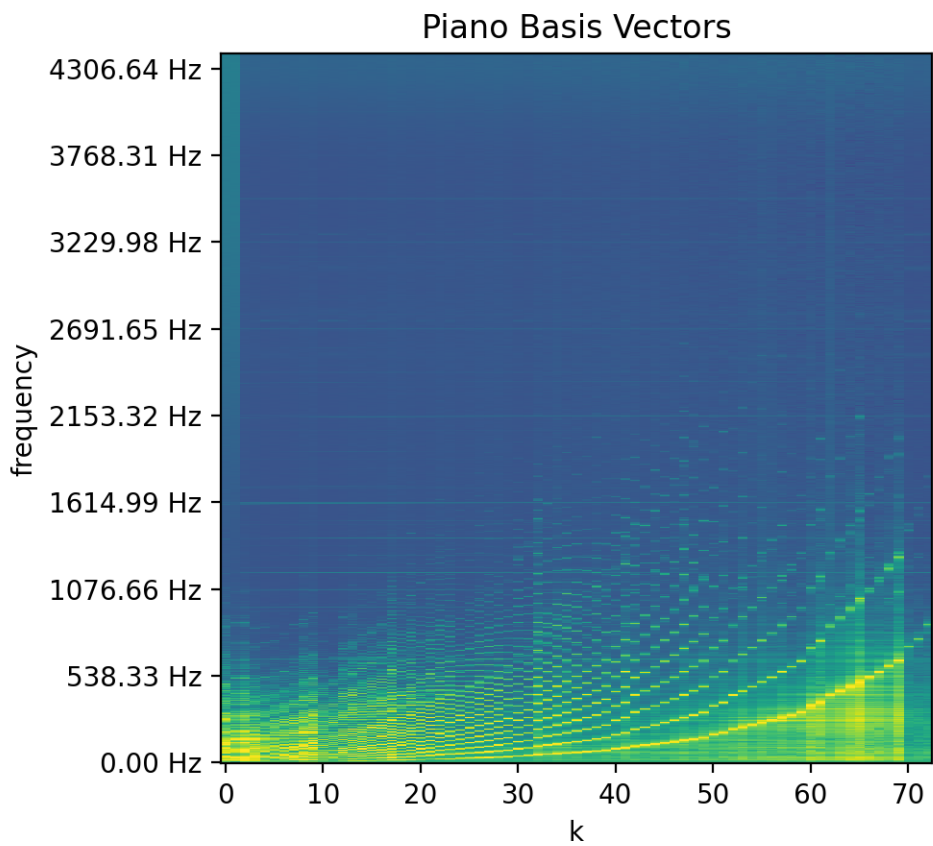


Figure 4.1: Score Piano Basis Vectors

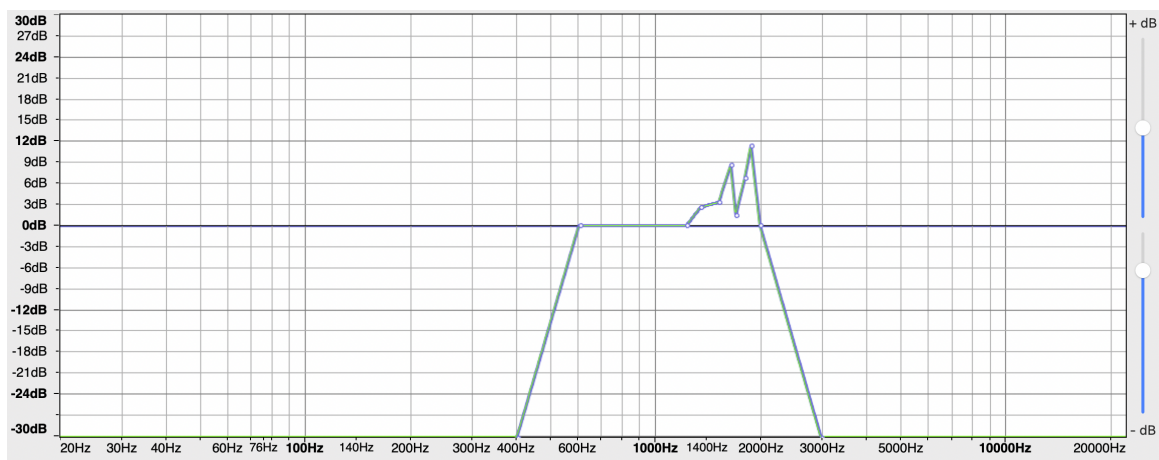


Figure 4.2: Damaged Piano Filter

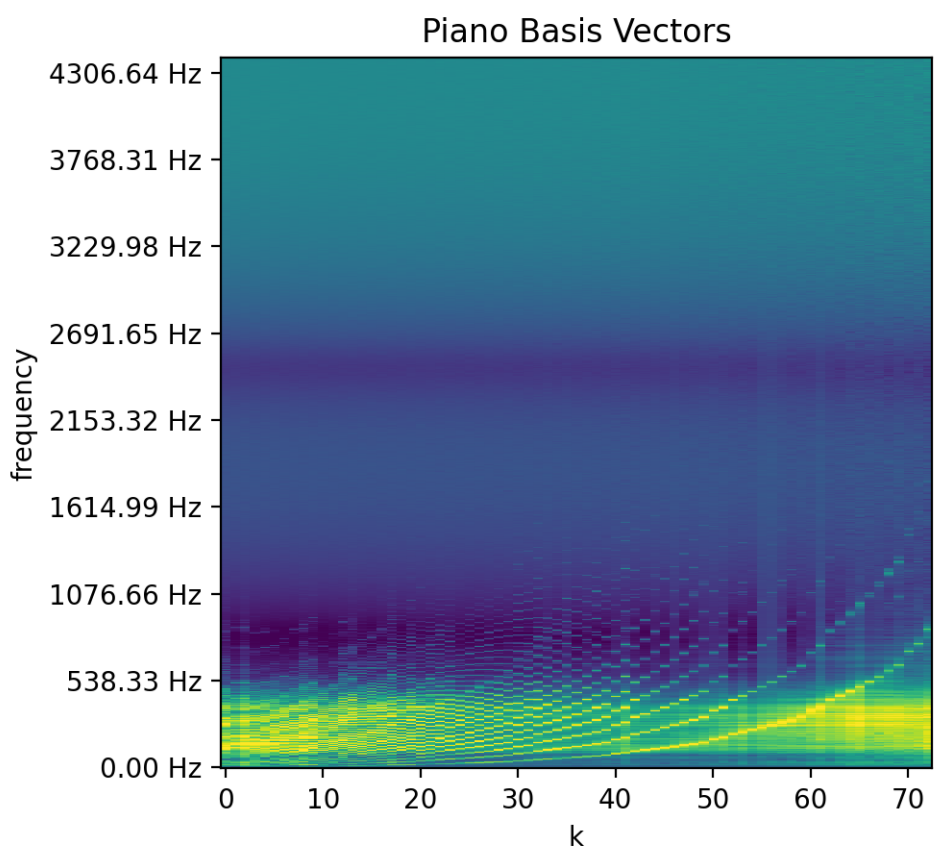


Figure 4.3: Score Piano Basis Vectors with Damage

4.2.2 Learning Noise Basis Vectors

The piano basis vectors W_p and noise basis vectors W_n , will be the make-up of our entire W . Noise basis vectors were learned from unsupervised NMF on a signal that is most-similar to that of the noise source(s) interfering with piano in the Brahms recording. We retrieved this signal by taking the longest continuous portion of only noise from the Brahms recording. This portion was found by-ear, which ended up being the portion from the beginning up until the announcer’s voice.

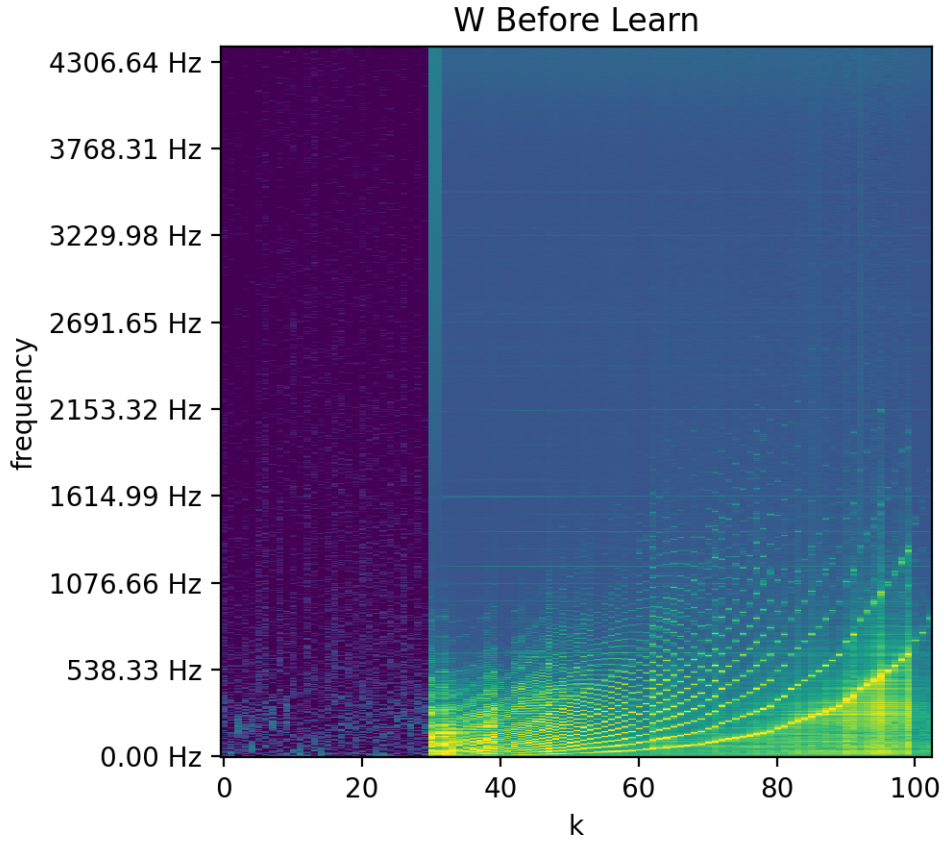


Figure 4.4: Prior-Learned Basis Vectors: Noise (left) and Piano (right)

4.2.3 Using the Learned Basis Vectors for Source Separation

Now that we have learned these basis vectors (a.k.a. they have been prior-learned), we are presented with many possibilities of how we can restore the Brahms recording. Unlike in classical (unsupervised) NMF, the basis vectors we have learned can be used as priors for the basis vector matrix W in restoring the recording. This specifically means the basis vector matrix W can be initialized with prior information that remains fixed during the updates of NMF, unlike in classical NMF where the entire basis vector matrix W is initialized to random values and included in the updates. In this implementation, our convention when putting together the basis vector matrix W is putting noise basis vectors on the left and piano basis vectors on the right.

The most straight-forward approach for us to first explore is the use of both our prior-learned piano and noise basis vectors. In this case, because our entire basis vectors are prior-learned, only the activations matrix H is updated in NMF, while the basis vectors W remain fixed. This approach is called supervised NMF, and after it runs, H has been trained, containing the activations which tell how much of each basis vector is in the signal at each time step. V , W and H at this point is illustrated by Figure 4.5.

The prior-learned piano basis vectors W_p (right side of W) in Figure 4.5 should look similar to Figure 4.1 because they are the same. It's important to realize that each basis vector and corresponding activations row-vector technically encodes a source of audio. So, if we only wanted to hear when Brahms played a middle-C on the piano in his recording, we would pick the middle-C piano basis vector and its corresponding activations row vector. Then we would multiply these vectors together to give us the corresponding spectrogram. More practically, if we only want to hear Brahms' piano playing, we pick all piano basis vectors W_p and their corresponding activations H_p ,

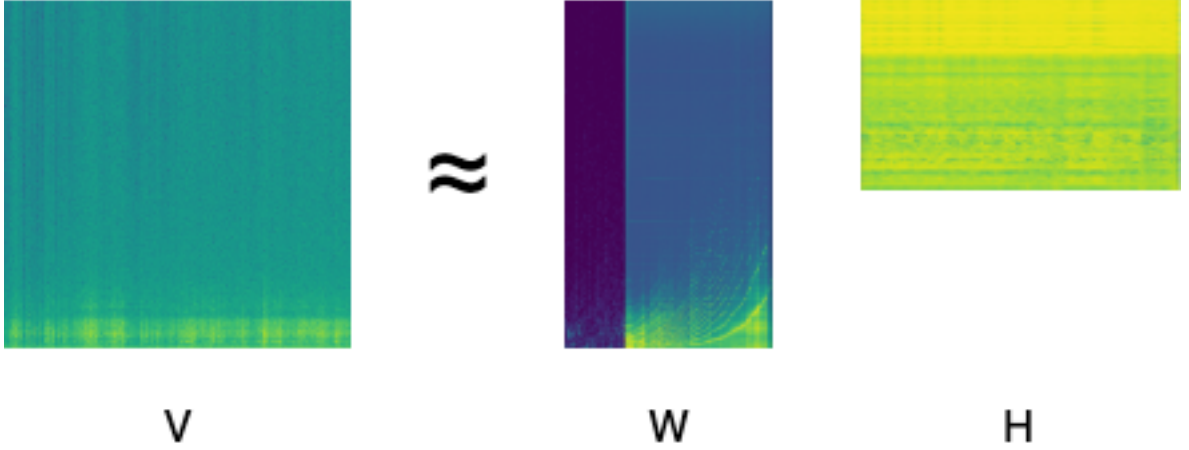


Figure 4.5: Supervised NMF Results on the Brahms Recording: V (left), W and H (right)

and then multiply them together to make the spectrogram of Brahms' piano playing \hat{V}_p ; $W_p H_p \approx \hat{V}_p$. Separating the piano basis vectors W_p and activations H_p from the noise basis vectors W_n and activations H_n allows us to pick this out, and is the source separation process of our NMF restoration pipeline. This process is shown in Figure 4.6.

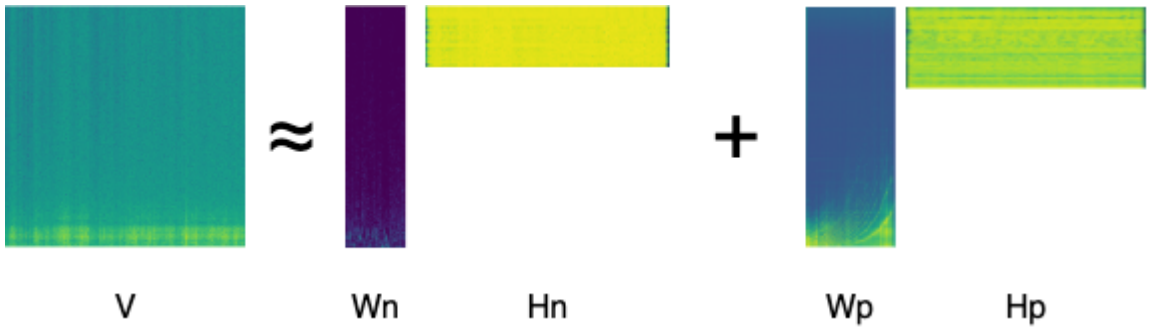


Figure 4.6: Supervised NMF Results as Mixture of Piano and Noise Source of the Brahms Recording: Splitting W and H

In supervised NMF, we are able to do source separation because our prior-learned basis vectors tell us the boundary where to split W and H . But in fact, we do not need both of them prior-learned to do source separation. We only need one of these sources. The basis vectors for the other source can be learned in the process of NMF. This approach is called semi-supervised NMF. One possibility for this approach, is

only using prior-learned piano basis vectors, and allowing the noise basis vectors to be learned along with H . In this way, the NMF algorithm will learn noise basis vectors that can fill in the noise sound which is too far off for use of the piano basis vectors. Another possibility is only using prior-learned noise basis vectors, allowing NMF to learn the piano sound which isn't fitting for the noise basis vectors.

Note that in semi-supervised NMF mentioned above, if the basis vectors for one source are prior-learned, then the rest of the basis vectors are initialized to random values. The downside of that is, for example, if we don't use the prior learned piano basis vectors, we miss out on their high-quality sound because instead the piano basis vectors are randomly-initialized. To try solving this, we can still use prior-learned piano basis vectors and prior-learned noise basis vectors by initializing them as such in W for NMF, and then allow NMF to update W_p from it's prior-learned initialization. Thus, the high quality spectra in W_p might be able to persist while also adapting to the recording at hand. The same thing can be done with W_n if we wanted to.

With supervised and semi-supervised NMF, this leaves five ways we can use our prior-learned basis vectors in NMF: both W_p and W_n (W) being prior-learned and fixed (1 way - supervised NMF), and either W_p or W_n as prior-learned and fixed leaving W_n or W_p to be learned respectively (2 ways - semi-supervised NMF), and initializing W_n or W_p respectively with random values or their prior-learned selves to be learned on top of (2 more ways - still, semi-supervised NMF). Lastly, we will also try l_1 -penalizing the piano activations H_p in order to restrict the number of non-zero piano activations allowed per timestep. This will naturally cause a realistic amount of piano notes to be played at any time: a maximum of 10 in theory since Brahms only had 10 fingers to work with. This will be possible by only using the l_1 -penalty term in the update for the subset of the activations for piano H_p .

We built our NMF algorithm to support supervised as well as semi-supervised learning and $l1$ -penalty. For this reason we will call it extended NMF and penalized extended NMF when using $l1$ -penalty, as seen in our NMF approach restoration pipeline in Figure 4.7.

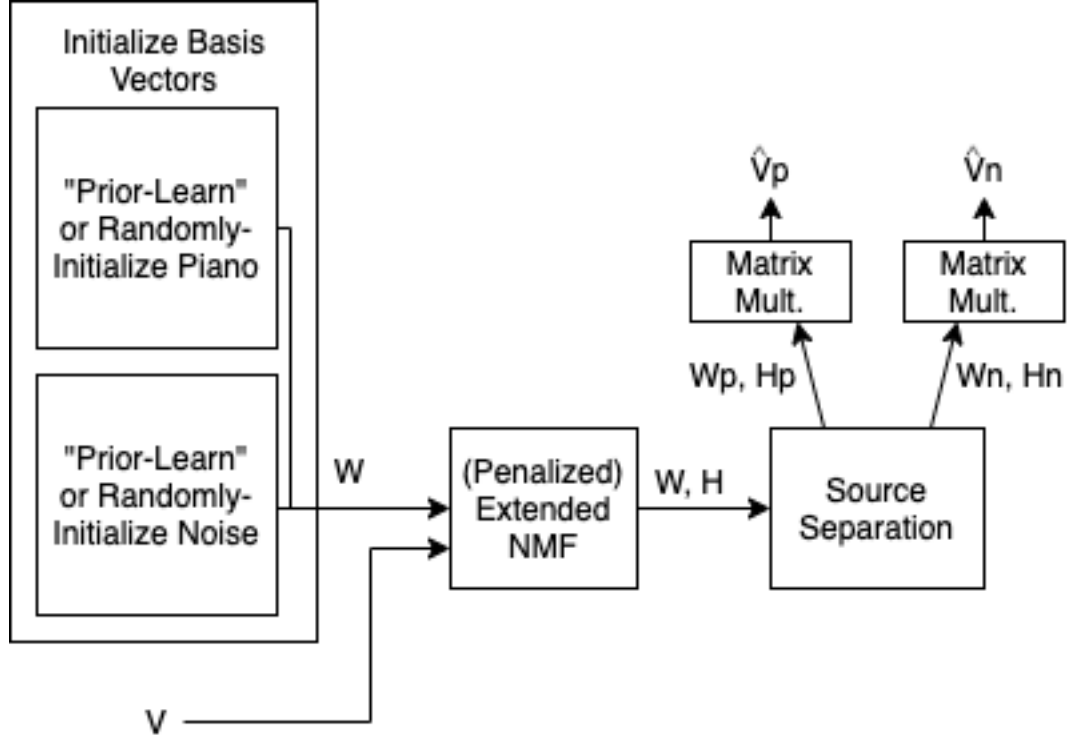


Figure 4.7: NMF Restoration Pipeline

4.3 Deep Learning Approach

Our deep learning approach is based off the supervised source separation model by Huang et al. which we will configure for denoising. This is a DRNN which forks into two outputs: the restored piano source, and the unwanted noise source. We begin by explaining training data synthesis, and then go into the model building process.

4.3.1 Synthesizing Training Data

To configure this model for denoising, we will give it training data as a mixture of piano source and an interfering noise source for the denoising task. Because this is a supervised model, we need to supply this training data: firstly a mix of piano source and noise source, and secondly the same piano source and noise source by themselves. Since we can't access the latter from the Brahms recording, or any other damaged historical recordings, we need to synthesize our training data. The goal of synthesizing training data is for a mix to sound as similar to a damaged historic recording as we can. Further, because this model is only for running on the Brahms recording, the goal is simplified to make a mix to sound as similar to the Brahms recording as we can.

Training data samples are synthesized by gathering audio excerpts of piano and audio excerpts of noise, and mixing them together. For our use case, the piano excerpts are gathered from recordings of solo piano renditions of Brahms' Hungarian Dance No. 1 in G minor, and only the sections of these renditions which are the same sections heard played in the Brahms recording. Because the model ours is based off of trains on voice utterances that are sequences of at most 100 timesteps [12], our piano excerpts are made to be sequences of 100 timesteps (spectrogram timesteps). We made the noise excerpts from the same noise signal used to make the noise basis vectors in our NMF approach. This was the longest continuous portion of only noise we could find by-ear in the Brahms recording, which was the portion from the beginning of the recording up until the announcer's voice. Since the noise signal was shorter than 100 timesteps, it was looped starting from a random point within it, until it matched the length of it's corresponding piano excerpt. This made for unique samples of the same length.

This is how we made the mixes from each piano excerpt and noise excerpt (we can refer to as the sources), and thus the final training data. Piano excerpt and noise excerpt signals were scaled until they sounded around the same volume as the piano and the noise in the Brahms recording, respectively. Since the SNR of the Brahms recording is unknown, this process aimed to replicate the SNR via this scaling. Then the piano and noise excerpt signals were summed together to make the noisy mixture. We then came back to the scaling factors of the excerpts and tweaked them by-ear, after comparing our mixtures to the Brahms recording. Finally the piano source, noise source and mix signals are turned into spectrograms, where the mix serves as the input and the piano and noise sources as the annotations. For DNN performance, the spectrograms are saved to numpy files which are loaded from in the training sessions.

4.3.1.1 Augmenting Noise Source Data

At this point, our training data mixes sounded pretty similar to the Brahms recording. In attempt to make them more similar, we experimented with augmenting our piano and noise source training samples before combining them into mixes. This was realized as two techniques for the noise source, and one for the piano source. For the noise source, a problem heard was that it didn't have variation because it was a loop of the noise at the beginning of the Brahms recording. We tried adding variation by time-stretching this signal. Time stretching is altering the speed of, either speeding up or slowing down, the contents of a signal without altering the frequencies. This was done with the effects API of the Librosa [16] Python library for audio and music processing. The process operated on adjacent random-length portions of the looped noise signal, each of which which were stretched in time by a random factor between 0.3 and 2.0. The result was more natural patterns of noise that took on different

duration. The signal was then sliced or looped again to have the same length as its corresponding piano source.

Our second technique for augmenting the noise source data was by mixing it with synthetic noise. This was made with an online vinyl noise generator (cit dust n scratches), which lets you create a custom audio mix of different types of noise you would hear on an old record. The parameters of this generator were tweaked, again by-ear, until it most closely matched the noise in the Brahms recording. After either of our noise source augmentations, we would again scale the noise signal to roughly the same noise volume heard in the Brahms recording.

4.3.1.2 Augmenting Piano Source Data

The piano source was augmented by degrading it exactly like how we degraded the piano basis vectors in our NMF approach. This was by passing each piano excerpt through a degradation filter (shown in figure 4.2) which largely removes frequencies outside of the range of 400-3,000 Hz. This technique is found to result in a vintage sound in which musical sources are muffled, like the piano in Brahms’ recording. With this augmentation, it is our hope that the model can better train for the low-quality piano in Brahms’ recording, and more importantly learn how to enhance it to the level of quality modern piano recordings have. In order for the latter to happen, the high quality piano excerpts remain as the annotations for their sample, while a copy of them is damaged for use in the mix.

4.3.2 Improving Upon a Baseline

Our first deep learning approach was to recreate the source separation model by Huang et al. [12] as a baseline. We specifically created the model they used in the

speech denoising setting of their experiments. This was implemented as a two-output model, where a spectrogram of a mixture can be fed in, and the two outputs are returned: a spectrogram of the wanted source, and spectrogram of the interfering source. This model is specifically a DNN that is jointly trained with a T-F mask, which remains fixed throughout the training process but has the purpose to filter the outputs of the DNN. Although the T-F mask isn't actually the output layer of the model, the optimizer pretends it is by tweaking the network weights under the masking constraint. This constraint, like mentioned earlier, is that the sum of the outputs of the model (a.k.a. the sum of the sources) is equal to the sum of the model input (a.k.a. the sum of the mixture), which conforms to qualities of mixtures of signals in the real world. This T-F mask is specifically a soft mask. The soft mask for an output source is made by dividing the source by the sum of itself and the other output source; this tells what proportion of the frequency in a T-F unit of the input spectrogram (noisy mixture) belongs to the output source. The code for the T-F masking layer is in the appendix.

The architecture of the baseline model is a DRNN, which begins with two consecutive recurrent layers, followed by a fork in the network made by giving the output of the recurrent layers to two dense layers, each followed by a fixed T-F masking layer. This was displayed in Figure 3.3 of the related works. The activations of the weighted network layers are ReLU to enforce non-negativity in the representations of magnitude frequency spectra, in parallel to how NMF enforces the same non-negativity. Like in Huang et al. all hidden layers have the same number of hidden units as features in the input layer, so no dimensionality reduction is done. We used the Keras API of the Tensorflow Python library to implement this model as quickly as possible [1].

Once the baseline model was created, upon initial training we immediately encountered the exploding gradient problem. This was detected by loss values reported as

NaNs and silent signals as model output. We hypothesized this was because our samples with timesteps $T = 100$ were still too lengthy in time for the model to handle, so we first explored hyperparameters shown to alleviate exploding gradient. We ended up settling on using the Adam optimizer with a reduced learning rate, which together gave us consistent non-NaN loss values and hearable model outputs.

We seek to improve this baseline by (1) training it on our synthetic data and (2) trying novel hyperparameters not mentioned in Huang et al., via a broad hyperparameter tuning search. This was for both the possibility of expanding upon their search, and finding out how to best fit to our training data. Novel hyperparameters explored were a dense layer with a TanH activation as the first hidden layer described in [22], and bidirectional RNNs and a skip-connection after the recurrent layers as described in [24]. These were joined by additional hyperparameters: the discriminative loss function parameter γ , mini-batch size, number of contiguous recurrent layers, residual dropout in the RNNs, input layer scaling and output layer un-scaling paired with batch normalization. The model architectures using these novel hyperparameters are displayed in Table 4.1. Note in the table, that the forking point of the model is denoted by "[forking point]" in a layer, which means that this layer is duplicated into two layers which receive the output of the previous layer as their inputs. This means every layer afterward is duplicated by two and only receives input from the layers in their branch of the network.

Model	Baseline	Baseline-Bidir-Skip	Baseline-Dense
Layer 1	RNN(100x2049)	Bi-RNN(100x4098)	Dense(100x2049)
-	ReLU	ReLU	TanH
Layer 2	RNN(100x2049)	Bi-RNN(100x4098)	RNN(100x2049)
-	ReLU	ReLU	ReLU
Layer 3	Dense(100x2049)	Concat(100x8196)	RNN(100x2049)
-	ReLU	ReLU	ReLU
-	[forking point]	-	-
Layer 4	T-F Masking	Dense(100x2049)	Dense(100x2049)
-	-	ReLU	ReLU
-	-	[forking point]	[forking point]
Layer 5	-	T-F Masking	T-F Masking

Table 4.1: Model Architectures using Novel Hyperparameters as Potential Improvements over the Baseline

Chapter 5

EVALUATION

This chapter outlines the evaluation throughout the development of our approaches. Being that this is an audio restoration problem, all evaluation is qualitative and aims to describe how the music is restored. These evaluations take place in succession, only considering a limited amount of parameters enabled by the system, and sticking to the parameters giving best results, before moving onto evaluating more of them. This technique aims to enforce order in the many possible parameter combos existing, and promote productivity in finding the best parameter combinations. These evaluations have been recorded in a designated experiment results document, and they are transferred into this section as the tables.

Each restoration evaluation will fill in two observation criteria: (1) the improvement of piano sound and (2) the amount of noise removed or denoising. Also each restoration is run on the original recording of Johannes Brahms playing a piano arrangement of his music piece titled "Hungarian Dance No. 1" in G Minor. Specifically, this is the bose acetate transfer of the original wax cylinder recording, in WAV file digital format [4].

Before we get into evaluation let's observe the status of the damaged Brahms recording, so we can have these observations to compare restoration results with. Firstly, the Brahms recording has constant noise that exists for the whole duration. We will call this noise "global noise", and some of this is in the form of low rumbling bass frequencies which we will call "rumbling" noise. The rest of the constant noise has medium frequencies which we will call "crunching noise", because the sound is

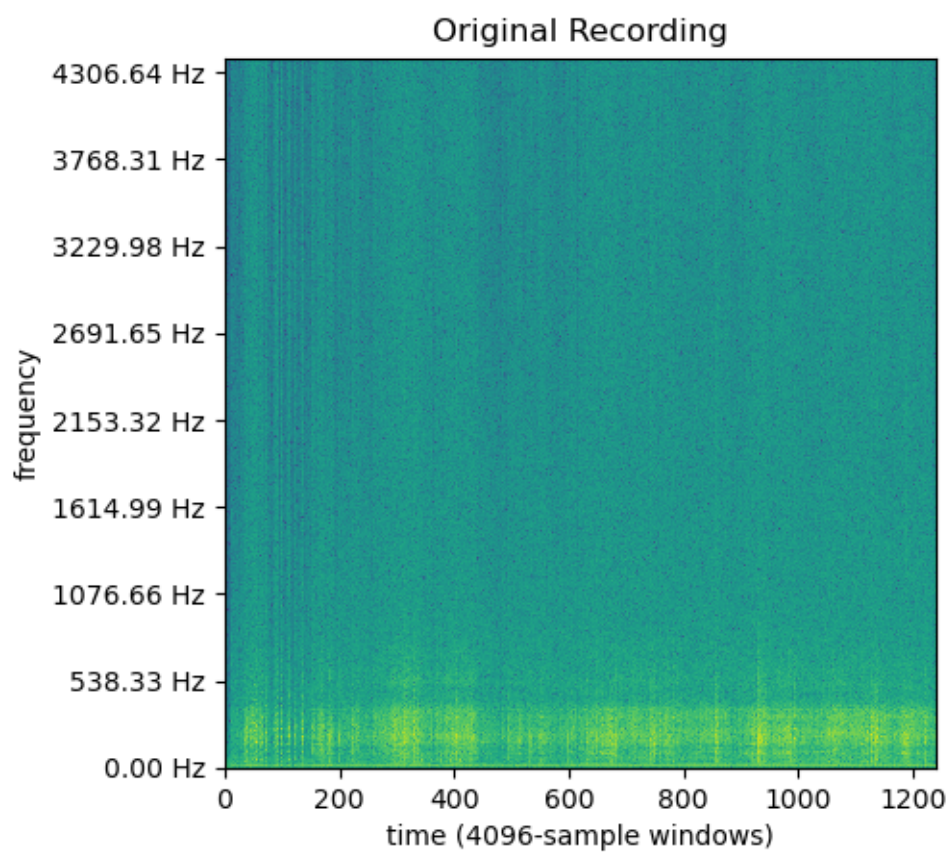


Figure 5.1: Spectrogram of Original Brahms Recording

analogous to how a car tire could sound when rolling and crunching over gravel on a dirt road. The Brahms recording also has some higher-frequency noise that is less constant throughout the recording, but is equally as noticeable as the others because it rises in volume during the phrases of piano. This noise sounds like hissing in ways comparable to white noise, so we will call this "hissing noise". Overall, most of the noise seems correlated to the piano because at points of highest piano volume, the hissing noise becomes present and the crunching noise comes forward in volume. Also note that we will refer to piano notes that sound closely grouped together in time as well as in musical structure as "piano phrases".

5.1 Non-negative Matrix Factorization Approach

We began evaluation on supervised NMF using both prior-learned piano and noise basis vectors. Immediate observations are that gaps between piano phrases are completely silent. This is positive, except for the fact that it removes the quiet piano that was once existing under noise in between these louder phrases. As for the noise that still exists in the non-silent portions, it no longer contains the rumbling noise from the original. Unfortunately the rest of the noise substantially persists, and now the hissing noise contains random pitches in a way analogous to how a wind chime sounds when exposed to a constant gust of wind. This is an artifact of our process we will call "wind-chime noise", and entails that piano basis vectors are being used to approximate the noise that noise basis vectors couldn't. In terms of if the high quality piano basis vectors improved the piano, there are hints of high quality sound that are still covered with too much noise to tell if it improves the original recording piano. The spectrogram of this result is in Figure 5.2.

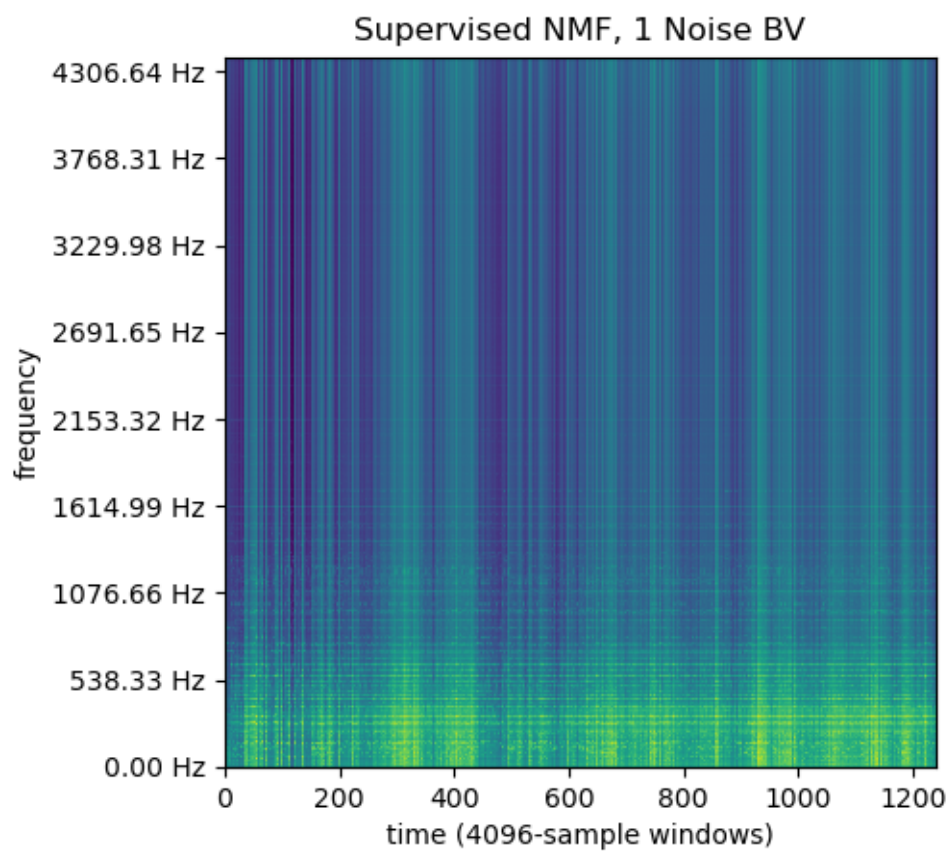


Figure 5.2: Spectrogram of Supervised NMF Restoration (1 Noise Basis Vector)

To solidify our evaluation on supervised NMF, we experimented with different amounts of noise basis vectors in ascending order, starting from none. The observations of this are put in Table 5.1, with evaluation criteria in the right-most columns. This table’s order is consistent with the order we made observations: the first row’s observations were the first made and are by comparison to the original recording, while remaining row’s observations are always by comparison to the previous row’s for ease of reading. Specifically, if the row says ”No change.”, adding noise basis vectors didn’t change things from the previous result.

# Noise Basis Vectors	(1) Piano Enhancement	(2) Denoising
0	Piano is covered by too much noise to detect improvement.	No noise is removed and it is instead replaced by wind-chime noise.
1	The piano underneath the noise might sound clearer, but is still covered with too much noise to detect improvement.	Noise between piano phrases completely removed, resulting in gaps of silence. But, quiet piano once existing in these gaps is gone. Rumbling noise removed. Crunching noise decently removed. Noise still covers up piano, and includes wind-chime noise.
2	No change.	More hissing noise is removed from over the piano notes.
3	No change.	No change.
5	No change.	No change.
10	No change.	No change.
15	No change.	No change.

Table 5.1: Results for the Number of Noise Basis Vectors with Supervised NMF

With no noise basis vectors, noise was approximated with piano sound as expected. With any noise basis vectors, the missing piano problem doesn’t grow any worse. Noise is consistently removed from over the piano phrases until three or more noise

basis vectors are used. We concluded that two noise basis vectors was optimal because adding any more noise basis vectors made no further benefit or damage. This shows how this number maximizes their representative abilities. The best results so far are in Figure 5.3

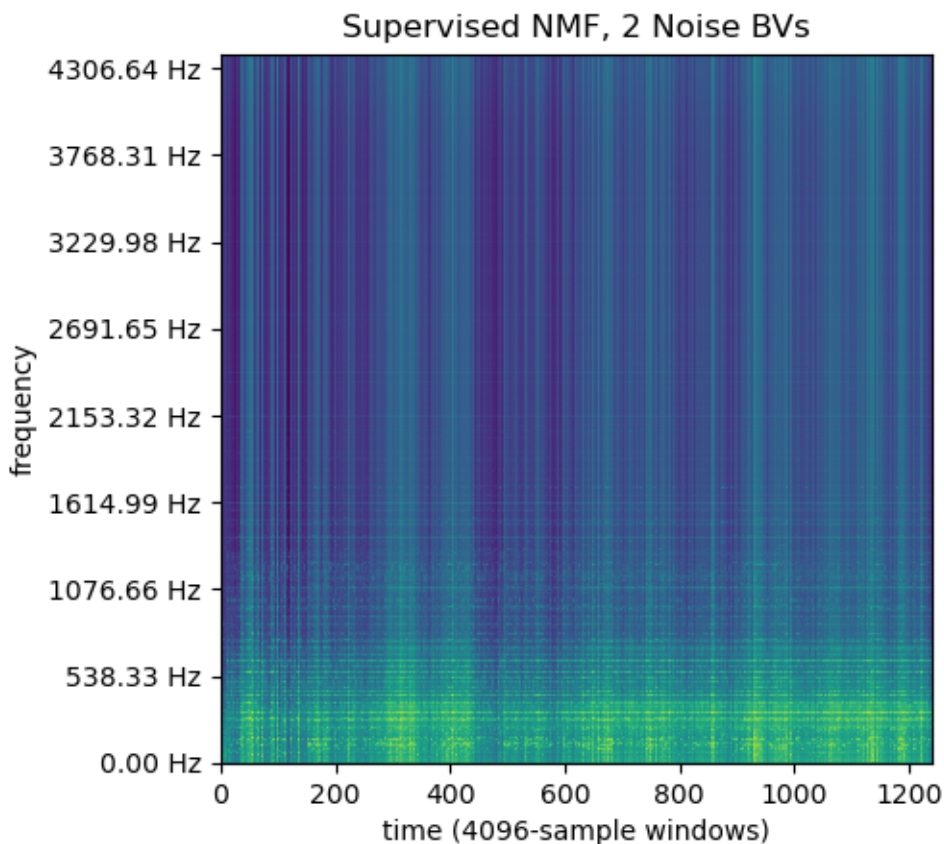


Figure 5.3: Spectrogram of Current Supervised NMF Restoration with 1 Noise Basis Vector added (2 in Total)

At this point we also evaluated supervised NMF like so far but by learning the piano activations with the damaged piano basis vectors instead of high quality ones. Immediate results are lackluster: they showed that in addition to the wind-chime noise our prior-learned piano basis vectors seem to make, a consistent high pitch seemed to approximate a global noise and lasted the whole duration. Along with this, the piano is covered by more noise. It is found later that this trend continues with damaged

piano basis vectors, so we don't consider them in the remainder of this evaluation chapter.

Next, we experimented with semi-supervised approaches while keeping 2 noise basis vectors. We tested either learning piano basis vectors or learning noise basis vectors as well as initializing the basis vectors with random values or as being prior-learned. Like our initial evaluation on supervised NMF, these observations are made in comparison to the original recording. They are in Table 5.2.

Learned	Prior-Learned Init.	(1) Piano Improvement	(2) Denoising
Piano	False	None detected.	Some rumbling noise removed, and noise in-between notes is somewhat removed.
Piano	True	None detected.	Some rumbling noise removed.
Noise	False	Piano underneath the noise might sound clearer, but is still covered with too much noise to detect improvement.	Rumbling noise is removed. Most of the crunching noise remains. All hissing noise seems to be replaced with wind-chime noise. Noise decently removed from between piano phrases - seemingly preserves most of the piano.
Noise	True	Piano underneath the noise might sound clearer, but is still covered with too much noise to detect improvement.	Rumbling noise is removed. Most of the crunching noise remains. All hissing noise seems to be replaced with wind-chime noise. Noise decently removed from between piano phrases - seemingly preserves most of the piano.

Table 5.2: Results for the Variations of Semi-Supervised NMF

When learning piano basis vectors, even when they are initialized as prior-learned, we observe no improvement of piano sound. Noise is either minimally or not removed between the piano phrases, preserving all piano, and some rumbling noise is removed, for not great results.

When learning noise basis vectors, there are hints of high quality piano sound that is too hard to distinguish from the noise, like in supervised NMF. Unlike in supervised NMF, noise is decently removed from between louder piano phrases while preserving most of the quieter piano. Also, the crunching noise isn't removed as well as in supervised NMF, which might be the trade-off needed to preserve all of the piano. For learning noise, no difference is heard between prior-learned and random initialization.

We concluded that learning noise basis vectors from random initialization give best results so far, because decently denoises while preserving the piano notes, and allows prior-learned piano basis vectors to remain fixed and improve quality, as seen in Figure 5.4. We carry on with this combination of parameters into $l1$ -penalty evaluation.

Next we experimented with $l1$ -penalty on piano activations. The $l1$ -penalty tuning parameter λ needed to be experimented with until it hopefully allows a few correct piano activations to persist while bringing the many incorrect ones to zero. The λ tuning search began by finding its upper-bound that makes for an all-zero H_p , resulting in a silent restoration. As an aside, immediately it was found in earlier experiments that when corresponding $W_p = W_{learn}$ in semi-supervised learning, W_p would make-up for H_p by learning larger values which canceled-out $l1$ -penalization. Because of this, H_p can only be penalized when corresponding W_p is fixed, which happens to work with our best NMF approach which fixes W_p . Our $l1$ -penalty evaluation is done on semi-supervised NMF which only uses prior-learned piano basis vectors, and learns two noise basis vectors. Observations are made with respect to the original recording, and are in Table 5.3.

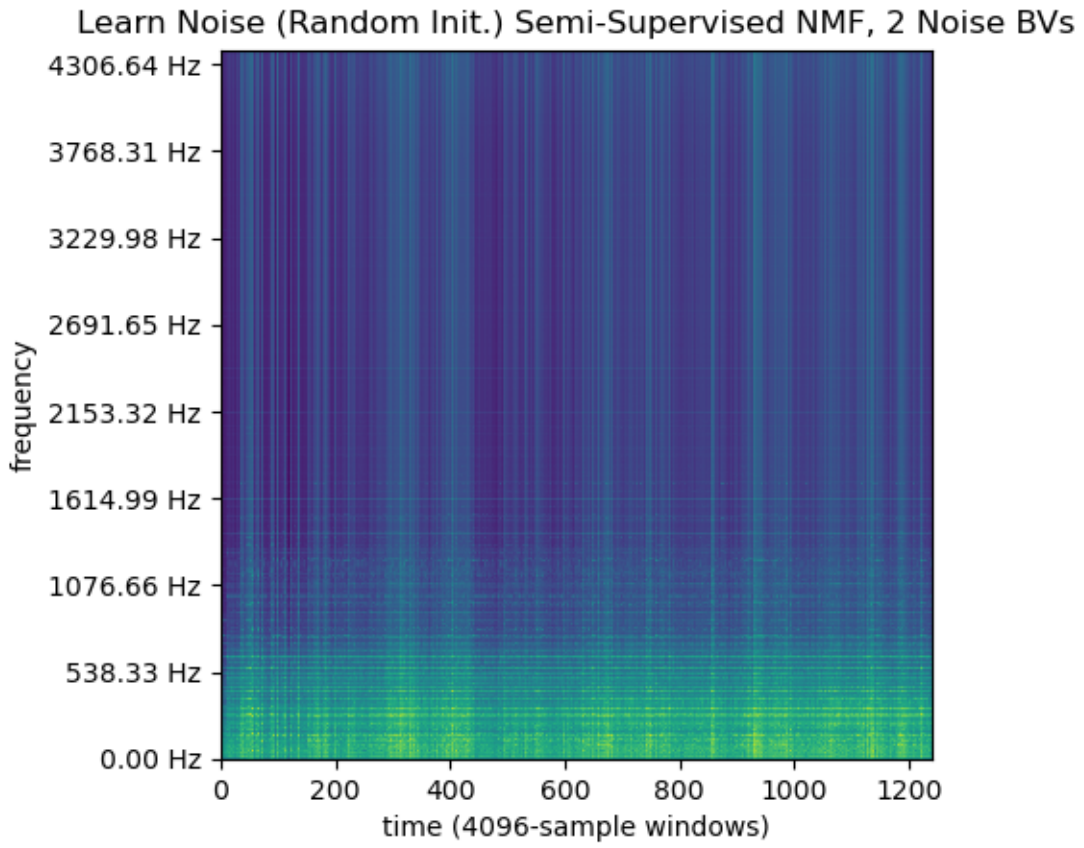


Figure 5.4: Spectrogram of Current Semi-Supervised NMF Restoration, Learning the Noise Basis Vectors from Random Initialization

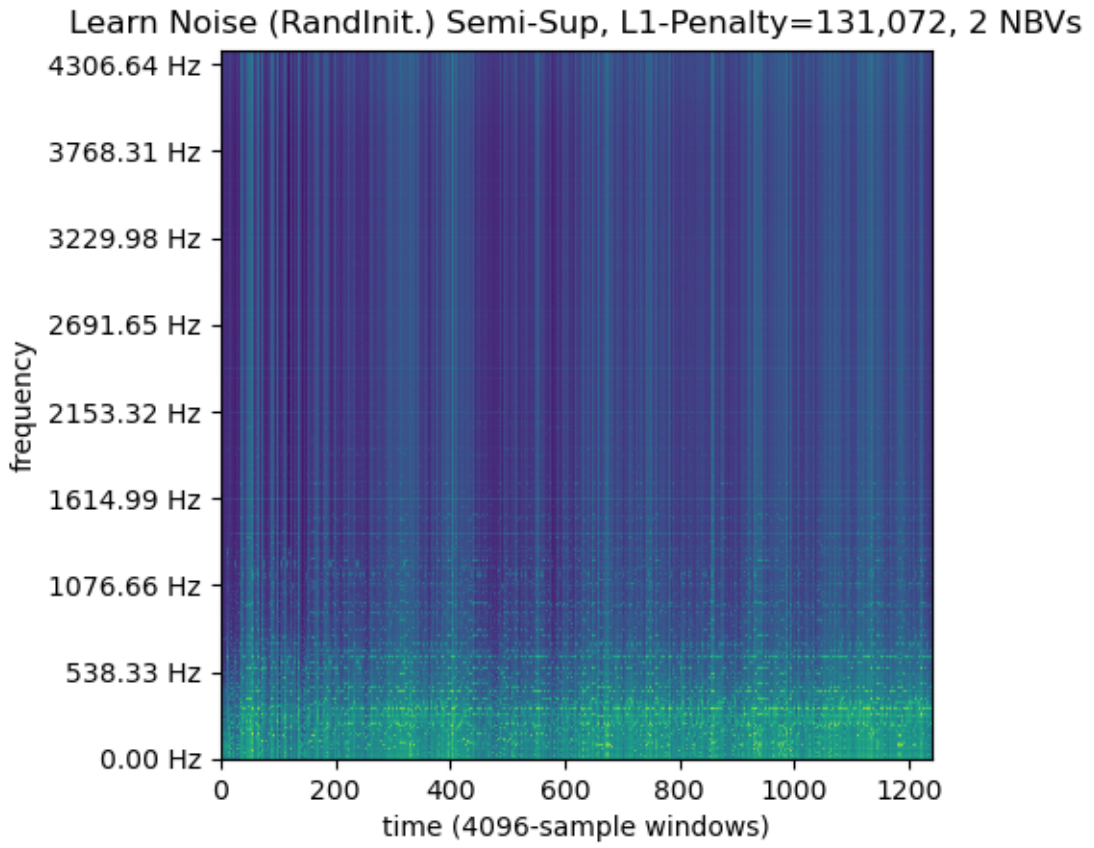


Figure 5.5: Spectrogram of Current Best NMF Restoration with L1 Penalty Term of $\gamma = 131,072$

L1 Penalty Term	(1) Piano Improvement	(2) Denoising
$> 4,000,000$	Silence.	Silence.
$\approx 500,000$	Perfect.	Occasional notes appear.
$\approx 200,000$	Great - a little noise on top.	Occasional piano phrases.
$\approx 100,000$	Piano improvement is finally noticeable - only some noise on top.	Very broken piano phrases. Notes have fringes of noise.
$\approx 60,000$	Piano underneath the noise might sound clearer, but is still covered with too much noise to detect improvement.	Broken piano phrases that are mostly covered in noise.
$\approx 30,000$	Piano underneath the noise might sound clearer, but is still covered with too much noise to detect improvement.	Rumbling noise is removed. Most of the crunching noise remains. All hissing noise seems to be replaced with wind-chime noise.

Table 5.3: Results for Various L1 Penalty Terms on Current Best NMF Approach

We observed that the largest penalty without silencing the recording yielded high quality piano sound with no overlaying noise. Unfortunately, the overlaying noise approximated with piano sound made this difficult to do without completely removing most of the piano. As the penalty is lowered, more piano-approximated noise leaks through as more piano is preserved. We found a good compromise using $\gamma = 131,072$, which allows enough high quality piano sound to be revealed and enough piano activations to be preserved.

As a final experiment, we tried replacing $l1$ -penalty with only allowing the few piano activations with the highest values to persist while zeroing out the rest of them, for each timestep. Into trying this, we saw these highest-value activations consistently included those in the bottom two row-vectors of the piano activations matrix H_p , for the entirety of the signal. More so, these activations correspond to piano notes that are not played in the recording, so it was apparent they were being used to

approximate noise. To mend this, we excluded the bottom activations while only including the top five activations per timestep. The result allows some piano phrases to last a little longer than with our best compromise with $l1$ -penalty, and thus this result is better. The piano activations matrix (before excluding the bottom two row-vectors) is in Figure 5.6.

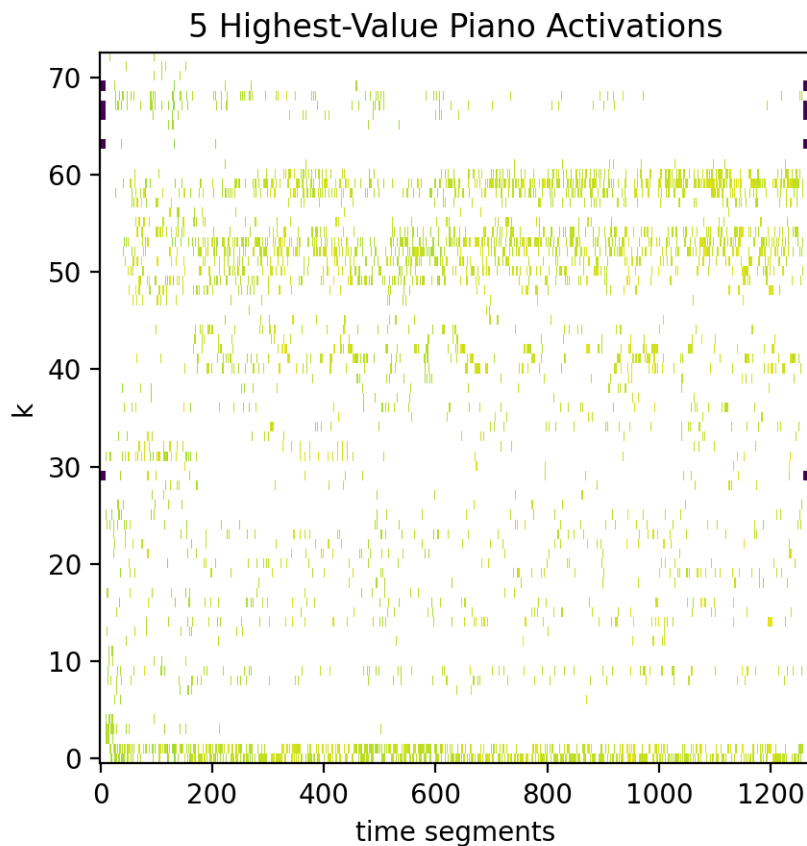


Figure 5.6: Piano Activations Matrix of Current Best NMF Approach, Restricted to the Top-5 Activations Per Timestep (Before Excluding the Bottom Two Row-Vectors)

The best configuration for our NMF approach is semi-supervised NMF which learns two noise basis vectors from random initialization, while restricting the piano activations to only include the top five values per timestep (excluding those in the bottom two row-vectors). This allows the clear piano activations to be revealed from the

noise, with the trade-off of excess denoising which removes most of the piano. This final result is in Figure 5.7

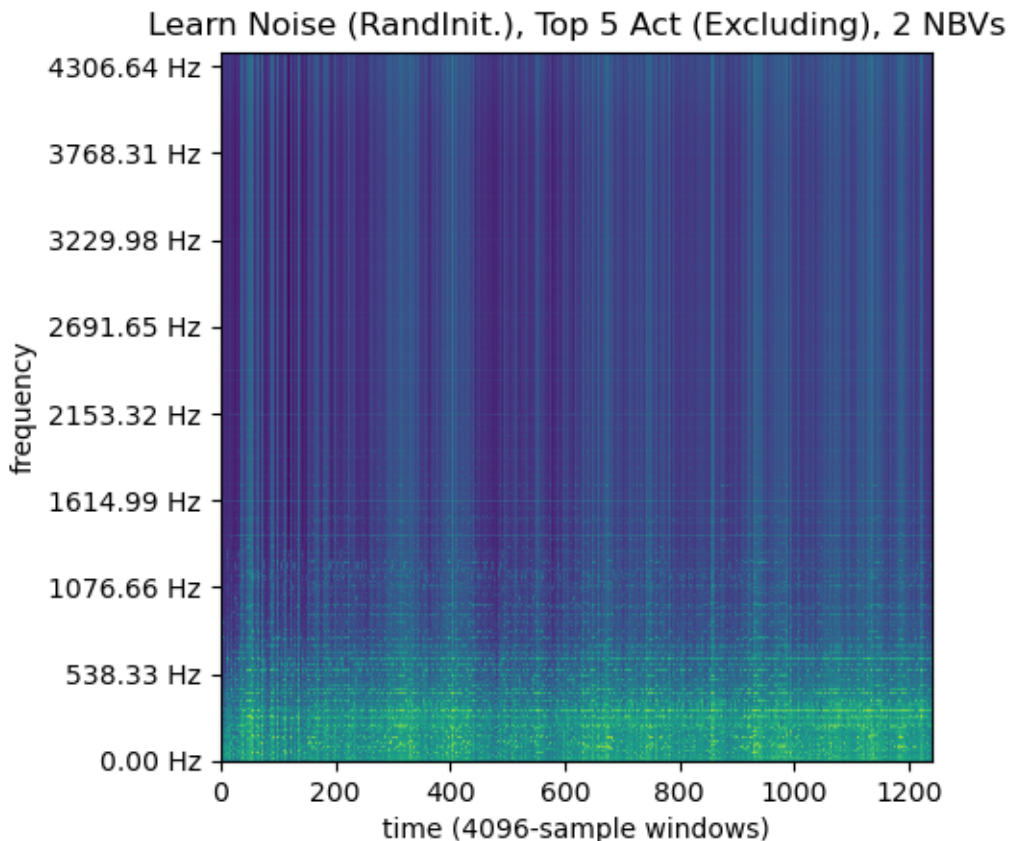


Figure 5.7: Best NMF Approach: Restricted to the Top-5 Piano Activations Per Timestep

5.2 Deep Learning Approach

As a recap, we sought to improve our deep learning model over the baseline in Huang et al. [12]. This was (1) by synthesizing training data with unique augmentations and (2) by trying novel hyperparameters in a hyperparameter-tuning search. The hyperparameter-tuning search was performed on the Cal Poly high-performance com-

puter servers on a Nvidia V100 GPU, while remaining evaluation was performed on a personal workstation with an Nvidia GeForce GTX 970 GPU.

The hyperparameter tuning search was realized as a grid-search over all possible combinations which totaled to 3072 and ran for one month. The grid search ran on our initial training data set, which included no data augmentations: solo piano excerpts of Brahms’ Hungarian Dance No. 1 in G Minor mixed with excerpts of the noise retrieved from the beginning of the Brahms recording which was looped over. To evaluate these grid search results, we retrieved the models corresponding to those which returned a validation loss within 10% of the minimum validation loss of the search. Of these models, we listened to the results of those with the most varying hyperparameters. Of these results, they all sounded practically the same. Upon comparing these results to the that of the baseline model, it was found that they sounded the same as well. The results of the model with the optimal hyperparameter combination is the baseline model that is trained on our data. The final code implementation of this model is in the appendix. Other than the baseline hyperparameters, these are the hyperparameters for this model: the discriminative loss function parameter $\gamma = 0.1$, 2 contiguous recurrent layers, mini-batch size of 50, and 40 epochs. This result is in Figure 5.8.

Because this is a denoising model, the piano can’t be enhanced but only uncovered by noise, so we do not give subjective observations on the quality of the piano. The evaluation of this result shows that rumbling noise is removed, and hissing noise is decently removed. The spaces between piano phrases are decently denoised, resulting in gaps of silence, but quiet piano is occasionally taken out.

Next, we tried evaluation on the same hyperparameter combination but trained on training data where the piano source is damaged. Results of this are in Figure 5.9.

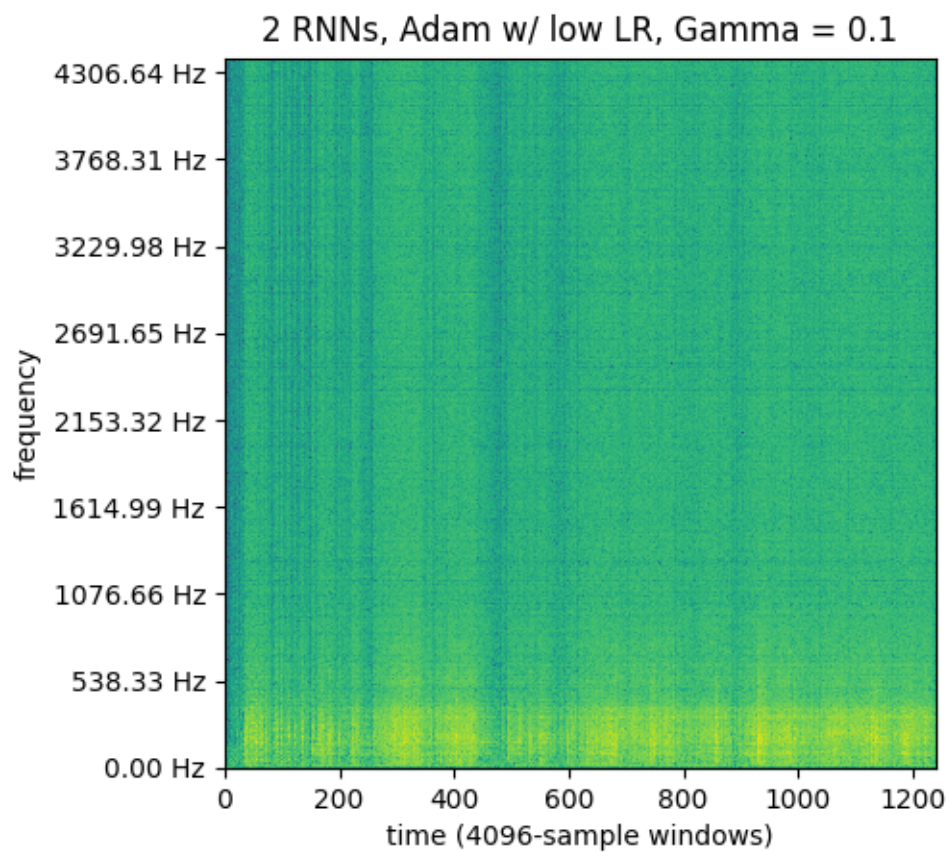


Figure 5.8: Spectrogram of Initial Denoising Model Result

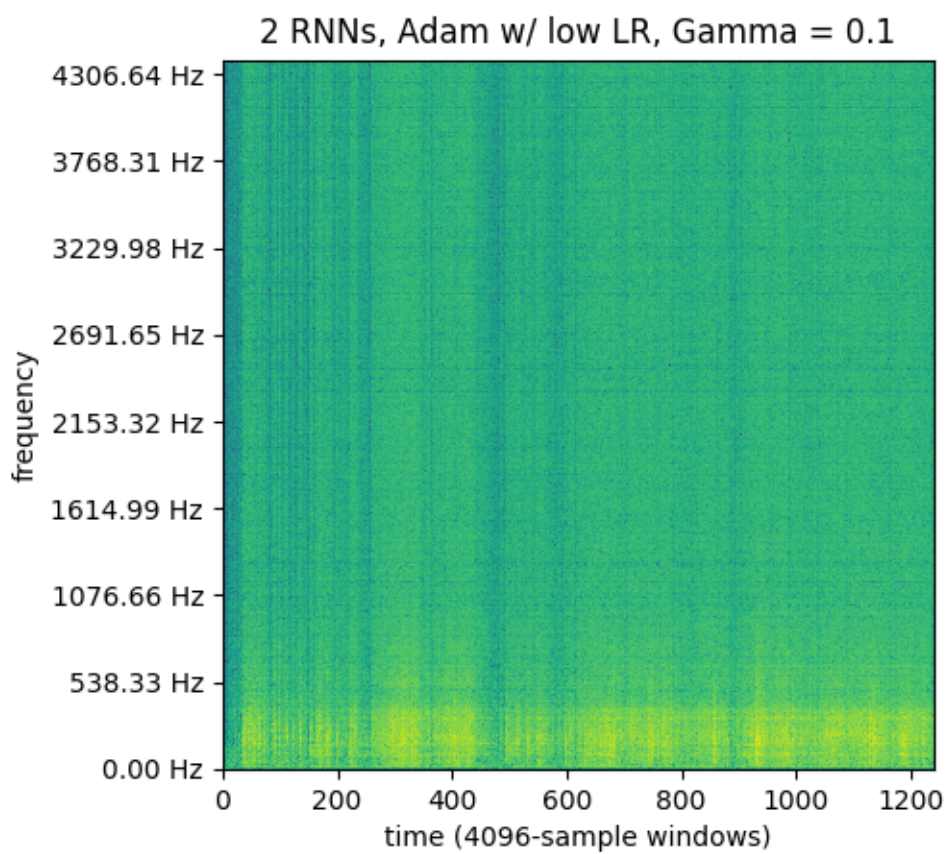


Figure 5.9: Spectrogram of Denoising Model Result, Trained on Data with a Damaged Piano Source

Upon meticulous observation, this model gives results that are almost identical to the previous model's, except for that these results take out more hissing noise. This might reflect that the damaged piano source in the training data prepared the model to more precisely identify piano through the noise. Given that the damaged piano source is absent of higher frequencies, this could have helped the Brahms piano be differentiated more accurately from the higher frequency hissing noise that interferes with it.

Lastly, we evaluated this model on the augmented training data that contains the damaged piano source, and the time-stretched and/or generated artificial mix noise source. This results in a very good denoising of the piano phrases, because practically no noise is heard on the piano. This is paired with the drawback that an immense amount of piano is removed, only allowing occasional piano phrases appear. These results occur when either of the augmented noise sources are in the training data, which shows how sensitive the model was to a noise that is more representative of the recording content. These results are in Figure 5.10.

5.3 Blind Test Including Both Approaches

Our results do not sound alike to the benchmark, so they don't warrant a large-scale blind test done by volunteers. Besides this fact, a blind-test script was created, which allowed us to compare the results of both of our approaches to the CCRMA benchmark and the original recording, without being told which recording is which. This was done by taking a small snippet from each signal, and allowing the user to play any of these snippets however many times they like. After listening to the snippets, the user can rank each one before being presented with the actual recordings that they ranked.

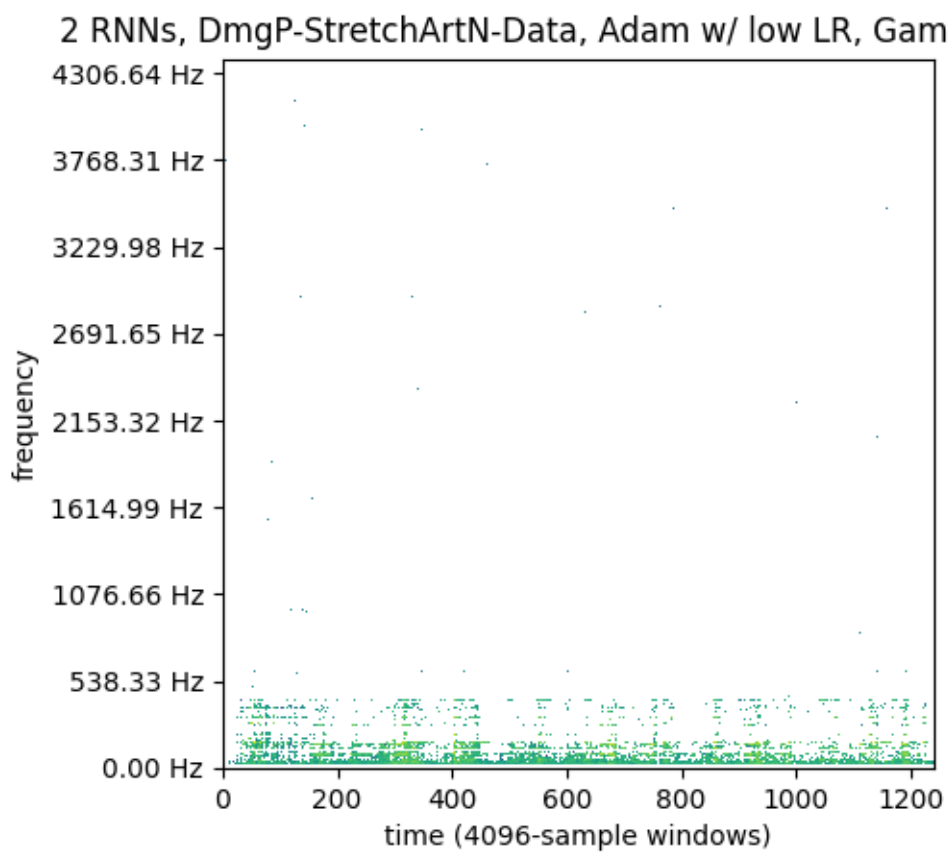


Figure 5.10: Spectrogram of Denoising Model Result, Trained on Data with a Damaged Piano Source and a Time-Stretched and Generated Noise Mix Source

This blind test becomes useless because our final restorations sound obviously different than each other, the benchmark and the original recording. Besides this, both our NMF and DL approach are uncomparable to the benchmark and original, because they remove too much piano, whereas the benchmark preserves all piano. Thus, removing too much piano is the main disadvantage of our approaches.

The question comes up of if our approaches offer advantages when wanting to seek more information than we can get from either the benchmark or the original recording. Although it reveals denoised piano at occasional phrases, the DL approach has no advantages because it yields sound quality no better than that in the benchmark. On the contrary, the NMF approach uses high quality piano sound to emulate exactly what Brahms was playing, so the smallest advantage is that the high quality piano sound lets us get a sense for the sound of Brahms' playing on a modern day piano. More so, because the NMF approach allows most of the main piano phrases of the recording to remain intact, the higher quality piano sound lets you hear the specific times at which Brahms plays the notes (onsets) throughout the recording. This is opposed to the CCRMA benchmark, in which the piano onsets are blurred by the artifacts of the restoration. As a final evaluation remark, this is the main advantage of our approach: the clearer onsets allowed by our NMF approach could allow someone to hear a more definite musical structure in timings, whereas the benchmark could leave listeners with a vague idea of when piano was played at moments.

Chapter 6

CONCLUSION

In the conclusion we go over possible future improvements to our approaches, and leave off with a summary of what we have done.

6.1 Future Work

Different signal transformations have shown the ability to encode the frequencies of a signal more precisely than what we use in our approach: pure magnitude frequency spectra [20]. For our NMF approach, we could learn noise basis vectors in more accurate ways, which could allow more piano to come through in the results. Some ideas are providing a more accurate noise source to learn from such as made by learning a filter that can synthesize noise alike to that in the Brahms recording.

Our DL approach missed the high quality piano sound potential exploited in our NMF approach. We explored ways of incorporating the high quality piano frequency spectra that the NMF piano basis vectors are made of, into the DNN. This included ideas of concatenating the same piano basis vectors used in NMF, to each DNN sample which could possibly give the model components of high quality piano sound with which it can use to synthesize high-quality sounding piano in the output. These experiments failed, but the potential for successful incorporation is still there. Although DL allows modeling of phase in addition to magnitude of frequencies (unlike NMF), it is a grey area on if the addition of phase features can increase source separation model performance [22]. Recent research is optimistic of this possibility if simpler features such as either complex-valued frequencies or waveforms are model input [19].

Lastly, audio denoising models also show competitive results with only one output, as opposed to source separation models that give one output per source [19].

6.2 Summary

In summary, we explored two machine learning approaches to restoring the damaged 1889 recording of Johannes Brahms playing a piano arrangement of his piece titled "Hungarian Dance no. 1". This involved DSP in pairing with with an NMF approach which uses high quality piano basis vectors, and a DL approach built upon a founded source separation model. Our benchmark to match was the restoration of the same recording by Berger et al. at CCRMA, which wasn't met. Our best approaches can't be compared to our benchmark, because they remove too much piano in the Brahms recording, whereas the benchmark preserves it all. That said, an advantage in our best approach brought by NMF, is the high-quality piano sound providing the ability to better observe the note onsets of Brahms' piano playing. If the application is to observe some clearer timing in a historic music recording, our approach offers a solution. Lastly, in terms of the overhead involved in our approach to restoring the Brahms recording, it is far less than processes of classical restoration. Because of this, our best approach also offers to enable slight but quicker restoration on the many historic music recordings still waiting to be uncovered.

BIBLIOGRAPHY

- [1] TensorFlow.
- [2] Understanding LSTM Networks – colah’s blog.
- [3] University of Iowa Electronic Music Studios.
- [4] J. Berger, R. R. Coifman, and M. J. Goldberg. Removing Noise from Music Using Local Trigonometric Bases and Wavelet Packets. *Journal of the Audio Engineering Society*, 42(10):808–818, Oct. 1994. Publisher: Audio Engineering Society.
- [5] G. Cabras, S. Canazza, P. L. Montessoro, and R. Rinaldo. RESTORATION OF AUDIO DOCUMENTS WITH LOW SNR: A NMF PARAMETER ESTIMATION AND PERCEPTUALLY MOTIVATED BAYESIAN SUPPRESSION RULE. page 8.
- [6] F. J. Canadas-Quesada, P. Vera-Candeas, D. Martinez-Munoz, N. Ruiz-Reyes, J. J. Carabias-Orti, and P. Cabanas-Molero. Constrained non-negative matrix factorization for score-informed piano music restoration. *Digital Signal Processing*, 50:240–257, Mar. 2016.
- [7] J. Deng, B. Schuller, F. Eyben, D. Schuller, Z. Zhang, H. Francois, and E. Oh. Exploiting time-frequency patterns with LSTM-RNNs for low-bitrate audio restoration. *Neural Computing and Applications*, 32(4):1095–1107, Feb. 2020.
- [8] C. Févotte and J. Idier. Algorithms for nonnegative matrix factorization with the beta-divergence. *arXiv:1010.1763 [cs]*, Mar. 2011. arXiv: 1010.1763.

- [9] C. Févotte, E. Vincent, and A. Ozerov. Single-Channel Audio Source Separation with NMF: Divergences, Constraints and Algorithms. In S. Makino, editor, *Audio Source Separation*, pages 1–24. Springer International Publishing, Cham, 2018. Series Title: Signals and Communication Technology.
- [10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
- [11] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck. Onsets and Frames: Dual-Objective Piano Transcription. *arXiv:1710.11153 [cs, eess, stat]*, June 2018. arXiv: 1710.11153.
- [12] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. 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, Dec. 2015. Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing.
- [13] V. Kuleshov, S. Z. Enam, and S. Ermon. Audio Super Resolution using Neural Networks. *arXiv:1708.00853 [cs]*, Aug. 2017. arXiv: 1708.00853.
- [14] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, Oct. 1999.
- [15] S. magazine. Digging Up Sound.
- [16] B. McFee, V. Lostanlen, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, O. Nieto, J. Mason, D. Ellis, E. Battenberg, S. Seyfarth, R. Yamamoto, K. Choi, Viktorandreevichmorozov, J. Moore, R. Bittner, S. Hidaka, Z. Wei,

Nullmightybofo, D. Hereñú, F.-R. Stöter, P. Friesch, A. Weiss, M. Vollrath, and T. Kim. librosa/librosa: 0.8.0, July 2020.

- [17] J. A. Moorer. DSP Restoration Techniques for Audio. In *2007 IEEE International Conference on Image Processing*, volume 4, pages IV – 5–IV – 8, Sept. 2007. ISSN: 2381-8549.
- [18] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context Encoders: Feature Learning by Inpainting. *arXiv:1604.07379 [cs]*, Nov. 2016. arXiv: 1604.07379.
- [19] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S. Chang, and T. Sainath. Deep Learning for Audio Signal Processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):206–219, May 2019. Conference Name: IEEE Journal of Selected Topics in Signal Processing.
- [20] J. Sleep. Automatic Music Transcription with Convolutional Neural Networks using Intuitive Filter Shapes. *Master’s Theses and Project Reports*, Oct. 2017.
- [21] P. Smaragdis and J. Brown. Non-negative matrix factorization for polyphonic music transcription. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No.03TH8684)*, pages 177–180, New Paltz, NY, USA, 2003. IEEE.
- [22] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji. Open-Unmix - A Reference Implementation for Music Source Separation. *Journal of Open Source Software*, 4(41):1667, Sept. 2019.
- [23] N. B. D. Sun. Source Separation Tutorial Mini-Series II: Introduction to Non-Negative Matrix Factorization. page 106.

- [24] D. Wang and J. Chen. Supervised Speech Separation Based on Deep Learning: An Overview. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1702–1726, Oct. 2018. Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing.

```

input_layer = Input(shape=(sequences , features))
x = SimpleRNN(features , activation='relu ',
               return_sequences=True) (input_layer)
x = SimpleRNN(features , activation='relu ',
               return_sequences=True) (x)
piano_hat = TimeDistributed(Dense(features)) (x)
noise_hat = TimeDistributed(Dense(features)) (x)
piano_pred = (TimeFreqMasking(epsilon)
              ((piano_hat , noise_hat , input_layer)))
noise_pred = (TimeFreqMasking(epsilon)
              ((noise_hat , piano_hat , input_layer)))

```

```

class TimeFreqMasking(Layer):
    def __init__(self , epsilon , **kwargs):
        super(TimeFreqMasking , self).__init__(**kwargs)
        self.epsilon = epsilon

    def call(self , inputs):
        y_hat_self , y_hat_other , x_mixed = inputs
        mask = (tf.abs(y_hat_self) /
                (tf.abs(y_hat_self) + tf.abs(y_hat_other) +
                 self.epsilon))
        y_tilde_self = mask * x_mixed
        return y_tilde_self

```