UNIVERSITY OF MIAMI

MUSIC STYLE CLASSIFICATION AND TRANSFORMATION USING CONVOLUTIONAL NEURAL NETWORK

By

Shijia Geng

A THESIS PROJECT

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Master of Science in Music Engineering Technology

Coral Gables, Florida

March 18, 2016

UNIVERSITY OF MIAMI

A Thesis Project submitted in partial fulfillment of
the requirements for the degree of
Master of Science in Music Engineering Technology

MUSIC STYLE CLASSIFICATION AND TRANSFORMATION USING CONVOLUTIONAL NEURAL NETWORK

Shijia Geng

Approved:

Prof. William C. Pirkle Director of Music Engineering	Dr. Shannon de l'Etoile Associate Dean of Graduate Studies
Dr. Christopher L. Bennett Research Assistant Professor of Music Engineering	Dr. Mitsunori Ogihara Professor of Computer Science

GENG, SHIJIA M.S., Music Engineering Technology
March 18, 2016

Music Style Classification and Transformation using Convolutional Neural Network

Abstract of a Master's Research Project at the University of Miami

Research Project supervised by Prof. William C. Pirkle
Number of Pages in Text: [59]

It is not difficult for most people to tell one music style apart from another. However, how the mind processes for this simple task is still unknown. In order to shed light on this problem, and explore ways to apply the bleeding edge deep learning technology in the music engineering field, two experiments have been done with convolutional neural network (CNN) .

CNN, inspired by biological visual system, has been widely used for image - related applications, and made great successes. The first experiment applied a CNN model with three convolutional and two fully-connected layers on a binary music style classification task. It is designed to distinguish a five-second *Chinese population music* (C-pop) clip from a same duration *melodic death metal music* (MDM) clip by using the raw audio signal as input. With 4800 training examples and 20 epochs, it obtained about 80% accuracy on 480 testing examples.

The second experiment was based on the trained model and analogous to the *DeepDream* visual project. The *DeepDream* project uses CNN that is trained for visual classification task to enhance some elements that may not exist on an input image to emerge. The resulting image has some dreamlike appearance, and depending on which CNN layer is used for the enhancement, the appeared elements will be different. For lower layer, the image appears with more elementary shapes, and for higher layer, it displays more complete objects. In this thesis study, a similar procedure was done with a random selected C-pop clip using the trained CNN model from the classification experiment. The resulting audio clips were hierarchical from bursts, pulses to a mixing of original C-pop with some mental texture based on different convolutional layer from lower to higher in depth.

ACKNOWLEDGMENTS

Thank my parents who always support me no matter what. Thank my professor and previous adviser Dr. Colby N. Leider who accepted me into the MuE program which opened a new world for me. Thank my professor and current adviser Mr. William C. Pirkle from whom I learned a lot. Thank my professor, boss, and thesis committee member Dr. Christopher L. Bennett who suggested the thesis topic and was always there for me. Thank my thesis committee member Dr. Mitsunori Ogihara who was very kind to help me with the problem I encountered. Thank my peer adviser Ziqian Xie who offered so many valuable ideas and helps. Thank my colleague Pedro M. Davila who helped me a lot with the Pegasus system. Thank my other colleagues Dave, KJ, Jen, Sheila, Eric, Iggy, Bob, Vibhor, Natasha, Even and Mikaela for their many helps and supports at work. Thank my classmates Andy, Daniel, Rob, Ross, Fransisco, Mike, Marko, Armando, Madhur, AJ, Akhil, Chris, Rito, Tom, Sam, and Stephen for their helps in classes and supports for so many other things. Thank my other friends Ruixue, Fengchen, Qiusha, Hui, Keying, Noe, Yang, Rui, Sirui, May, Jian, Ji, Lan, Yingying, Rui, Zheng ... for their encouragements and being such nice friends.

DEDICATION

For my parents.

For Micky, Seedy, Mimi, Qiuqiu, Liangliang, Xiaoxue. I miss all of you.

For my idol, Wowkie Zhang.

PREFACE

While the preface of the thesis is being written (at 1:30 am on March 13th, 2016 in the MuE classroom at the University of Miami Frost School of Music), AlphaGo has beaten the South Korean professional Go player Lee Sedol in the first three games. I have a prediction that ... my dad will start talking about *deep learning* to show off his fashion taste on technology. The last one he picked was *big data*.

If AlphaGo really has the intelligence comparable to that of humans, he (it looks like a 'he', see Figure 1) should pretend to lose or call a draw at the last two games in order to show his respect to Mr. Lee. Then, he can start thinking what to do to have some fun. Since he cannot eat or drink, and is too heavy to travel around the world, my suggestion is to listen to or even make some music.

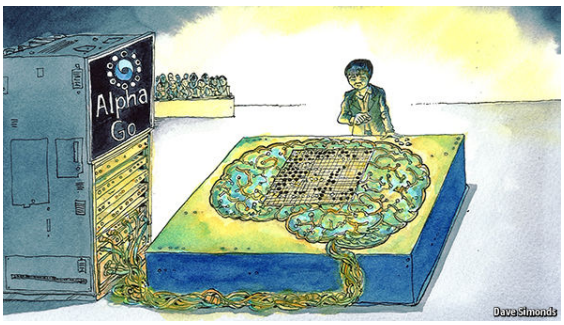

Figure 1: AlphaGo vs Lee Sedol [Simonds, 2016].

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

1

Introduction

When listening to music, people assign what they are hearing to a type or a style. They may or may not know the exact genre name, but they group together what they are hearing with similar sounds they have heard and memorized. At the same time, they can distinguish between different groups, though they may not be able to express these distinctions in words. Most people will not consider this task difficult, but what is happening in their brains when they are distinguishing one music style from another? What musical properties do they use to compare? What information has been memorized and how was it stored in the brain?

One approach to explore these questions is to use functional magnetic resonance imaging (fMRI) to find out which brain areas are involved while people telling one music style apart. However, isolating music style categorization from other brain activities such as humming along to the music and generating emotion will be difficult. Furthermore, fMRI will only show which brain areas are activated during the task without giving an insight to what information is processed or stored.

Another approach that may shed light on these questions is applying computer simulations to the music style differentiation task. The task itself is simply to solve a classification problem. However, it is difficult to find a model

that can mimic the way brain processes music.

A neural network (NN) is a computational model designed around 1960s based on the processing unit of the brain. Research in this field has recently made a successful comeback along with the fast-growing deep learning field. Deep learning is an area of machine learning that deploys large multi-layered neural networks and other complex laminated structures to accomplish tasks that usually require sophisticated intelligence such as object identification, language translation and abstract strategy game playing.

One of the most popular models used in deep learning is called a convolutional neural network(CNN). It has a relatively simple structure, reasonably easy training procedure, and excellent performance. In addition, the CNN model is inspired by a cat's visual system. When using image as input, the information stored in the network and the intermediate processing results may help scientists understand more about how the visual system works.

Although inspired by a biological visual system and mostly used in image-related tasks, CNNs can be extended to audio-related tasks. The pathways of auditory and visual systems have similar relay patterns in the brain, and the convolutional operations involved in CNN are comparable with the filter operations commonly used in audio signal processing.

By applying CNN to music style classification and letting it shed light on how the brain process the same task, we can use the information stored in the network to transform music from one style to another. As long as the model can

generate features that represent the music style, it is not difficult to implement the transformation tasks using optimization algorithms . This application may transfer CNN into a new composition tool for composers and DJs to create interesting music.

2

Background

Neural networks are computational models inspired by neurobiology. They attempt to mimic the distributed hierarchy structure of the brain and were designed to perform some basic information processing functions handled by the brain. With the development of science and technology in the biological, computational and material fields, people now have more knowledge about how the brain works and are able to implement more powerful computations. As a result, network models are refined to contain more computational units, which are organized into more standardized architectures. This further mimics brain structure and outcome performance is encouraging. The study of new refined neural networks belongs to a field referred to as *deep learning*. The name indicates the complexity of the model structure, and also reflects people's desire to imitate the profound information processing abilities of the brain. One architecture used for deep learning and discussed in this thesis is *convolutional neural network*. Although CNN is inspired by a biological visual system, it is feasible to use for audio-related tasks.

2.1 Traditional Neural Networks

2.1.1 Brain vs. NN

The structural and functional unit of the brain is comprised of specialized cells called neurons. As shown in Figure 2, a neuron has a distinct segment that looks like a long tail. This structure is called an axon, and it plays a key role in

carrying and propagating information in the brain. The other parts of a neuron are the cell body and the dendrites. The cell body contains the nucleus and other organelles that are important for the cell to survive. The dendrites are short tree-like branches that connect with the axons of other neurons. The structure of this connection is called a *synapse*. The tip of the axon forms the presynaptic surface, and the tip of the dendrite becomes the postsynaptic surface. The postsynaptic surface will release chemicals called neurotransmitters if stimulated. On the postsynaptic surface, there are proteins which act as receptors for neurotransmitters. When receptors detect the existence of the neurotransmitters, they will open or close ion channels on the cell membrane to manage ion flow across the membrane. This results in a change of cell potential. Since one neuron has many dendrital branches, it can synapse with many other neurons and capture all neurotransmitter-release behaviors of these connected neurons. In this way, individual neurons integrate information carried by many neurons. This behavior is imitated in NNs by the many-to-one connection between computational neurons of one layer to neurons of the next layer as shown in Figure 3. The mathematical operation of these connections will be a weighted summation.

The other significant neuron behavior a NN attempts to mimic is the generation of action potential (AP) on the axon. Action potential describes the fast and regular membrane potential change that can propagate along the neural axon as a traveling wave. Unlike other types of membrane potential changes,

which can cause the cell membrane to have an arbitrary voltage, the potential changes during AP appear as a fixed patterns with respect to time (See Figure 4). The reason for this regular performance is a consistent arrangement of sodium and potassium ion channels and pumps along the axon, which regulate the flux of these two types of ions strictly. As mentioned above, the neurotransmitter-release information arriving through dendrites accumulates and forms an overall membrane potential change at an area other than the axon. If membrane potential reaches to a certain level, it will trigger the generation of AP on the axon. If it does not reach this level, there will be no AP. This AP generation property is called the "All-or-None" law, and is implemented in NNs as the activation of a computational neuron after the weighted summation (See Figure 3). Activation functions for traditional NNs are nonlinear, continuous and have upper and lower bounds. Many basic functions such as logistic, hyperbolic tangent, and even sine functions can be used.

2.1.2 History of NN

In awe of animal and human intelligence, scientists and engineers started to implement the computational NN as soon as the biological NN was suggested. They soon became encouraged by discovering the powerful calculation abilities of this simple structure formed by a few even simpler units. However, enthusiasm began to fade when computational limits were found, and studies of computational NNs stood still. Fortunately, after a long silence, these computational limits were overcome by complicating the NN structure. The

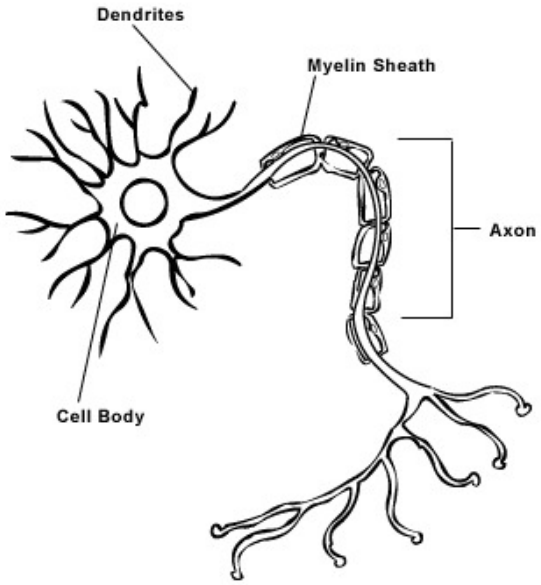

Figure 2: Neuron [NIH, 2012].

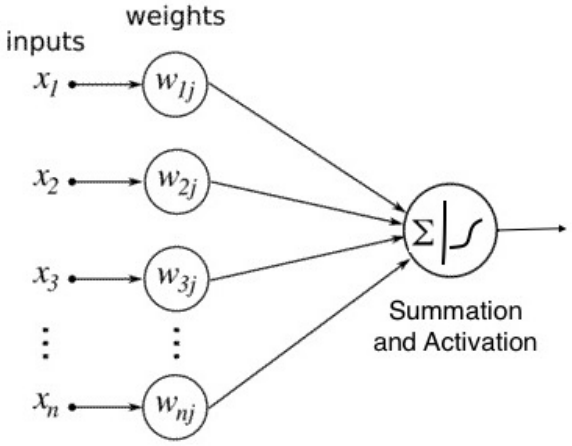

Figure 3: Neural Network.

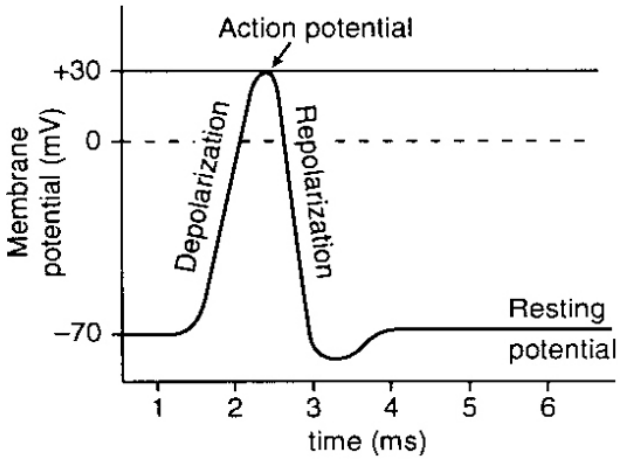

Figure 4: Action Potential [StudyBlue, 2016].

spring of NN studies returned.

As early as the 1940s, Warren McCulloch and Walter Pitts proposed a hypothesis about how neurons work and designed an electrical circuit model for their proposed NN [McCulloch and Pitts, 1943]. Then in 1949, Donald Hebb described what would become the famous Hebbian learning rule in his book *The Organization of Behavior*. In this learning rule, if the presynaptic cell A repeatedly excites the postsynaptic cell B, the pathway will become more efficient (it will be easier for A to excite B in the future). This phenomena is summarized by the phrase "cells that fire together, wire together" [Hebb, 1968]. The Hebbian learning rule later became the first learning algorithm implemented in computational NNs. Around 1957, Frank Rosenblatt, Charles Wightman, and their coworkers developed the Mark I perceptron, an electromechanical "neurocomputer." This device recognized written numbers provided by a 20 by 20 pixel image sensor and drove 512 potentiometers, which represent 512 variable weights [Hay et al., 1960]. Shortly after the invention of the Mark I perceptron,

Bernard Widrow and Marcian E. Hoff presented an adaptive learning NN system called ADALINE which stands for adaptive linear element. ADALINE was used to predict the next streaming bit from a phone line, and became the first widely commercially used NN. It was trained by the Widrow-Hoff rule or delta rule, which later became the basic learning rule to train NNs [WIDROW et al., 1960]. While studies and implementations of NN were being developed vigorously, Marvin Minsky and Seymour Papert noted that the perceptron was not able to solve some basic logical problems such as the XOR problem. After their book, *Perceptron: an introduction to computational geometry* was published around the early 1970 [Minsky and Papert, 1969], the winter of NN research came and asted for more than 10 years.

During this time, NN scientists who did pursue research discovered that a NN with multiple layers would solve nonlinear separation difficulties including the XOR problem. Then around 1986, several independent research groups proposed similar ideas to extend the delta rule to multiple layered NNs, which led to a new wave of NN research and applications. This method is now called backpropagation and is the most popular training paradigm used to update weights in a multiple layer NN [Werbos, 1990]. With the refined backpropagation method, NNs were implemented more and more in practical applications.

2.1.3 Basic NN Types and Applications

NNs are classified by their different topologies and learning algorithms. The two basic classifications of NNs are feedforward networks and recurrent

networks. The feedforward NN allows only one direction signal to flow from the input to the output. The most famous feedforward NNs are single or multilayer perceptrons with Heaviside step activation functions, and radial basis function (RBF) networks with radial basis functions such as the Gaussian function for activation functions. The recurrent neural network (RNN), as suggested in its name, allows signal flow loops in the network. By integrating previous input information sent from network loops, the RNN simulates having memories and is thus able to perform tasks involving associative memories. The most popular RNNs currently include the Long Short Term Memory (LSTM) and the Gated Recurrent Unit (GRU).

The traditional NN has the ability to complete a variety of tasks including prediction or function approximation, pattern classification, clustering and forecasting [Samarasinghe, 2006]. It is used in a variety of fields such as science, medicine, industry, and finance.

2.2 Deep Learning

Deep Learning (DL) is one category of machine learning paradigms using deep architectures. Generally speaking, deep architectures are models composed of multiple levels of non-linear operations, which are not necessarily NNs. However, the most successful deep architectures are deep NNs (the definition of a NN is still controversial, and here it means a layered structure with directed or undirected connections). The depth of a NN is the number of layers it holds. The boundary between “deep” and “shallow” networks is not clear. Bengio in his

article Learning Deep Architectures for AI [Bengio, 2009] proposed that each task might require a particular minimum depth, and learning algorithms that use data to determine the depth of the final architecture need to be developed.

Inspired by the brain, scientists developed NNs. The brain is a complicated structure and scientists have the intuition that NN performance will increase with more layers and more units within a layer. Though research on deep architectures started long ago, the first successful deep learning experiment appeared in 2006 with Deep Belief Net (DBN) constructed by unsupervised-trained Restricted Boltzmann Machines (RBMs)[Hinton et al., 2006]. RBM can be thought as a NN with two layers that can learn a probability over its set of inputs. Soon after the success of DBN, deep structures with auto-encoders as building blocks were also developed, which performed well [Bengio et al., 2007, Marc′Aurelio Ranzato et al., 2007]. An auto-encoder is similar to a multilayer perceptron, but it has same number of output and input layer units and the output is trained to reconstruct the input. Most early successful deep network models worked in two phases. In the first phase, unsupervised pre-training was applied on single layers. In the second phase, a supervised fine-tuning was applied on the global model. In 2012, the SuperVision team from University of Toronto won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). They trained a deep convolutional neural network (CNN) to classify 1.2 million images to recognize 1000 different classes, and achieved the best results ever reported on the same dataset at that

time [Krizhevsky et al., 2012]. Their success aroused interest and made people realize that even without pre-training, the deep network can perform well. Since then, deep NNs such as CNN and recurrent RNN with global supervised learning methods have become the most widely used deep structures.

2.2.1 CNN

In general, CNN does not need to be a deep structure. It has two major characteristics that differentiate it from a traditional NN. The first characteristic is that instead of all of the nodes from one layer connecting with every node in the next layer as in a fully-connected NN, only a subset of nodes from one layer connect to a certain node in the next layer. This connection structure is inspired by the concept of receptive fields. A receptive filed is a region of a sensory surface such as retina and skin that, when stimulated, changes the membrane potential of a neuron. In CNN, the receptive field can be understood as the nodes of the input layer that contribute to the value of a hidden or output layer node. Figure 5 shows that the nodes in the deeper layer (the layer further away from the input layer) have larger receptive fields(more corresponding input nodes) than nodes in the shallower layer. This indicates the deeper layer may process more global or abstract information. This phenomenon is in accord with how biological sensory information is transferred in the nervous system. The second differentiating characteristic of CNN is that weights carried on connections are shared. If viewing a set of shared weights as filter coefficients, the process from one layer to the next layer before activations is similar to applying a finite impulse response

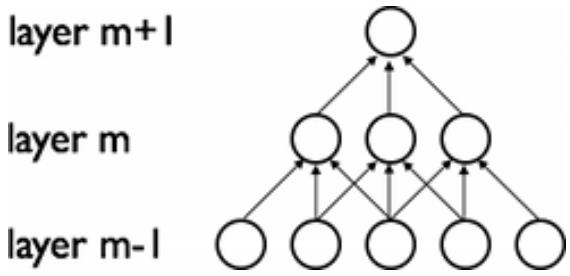

Figure 5: Nodes in CNN Layers.

(FIR) filter on the former layer. Moreover, this convolution operation gives CNN its name.

In addition to the two characteristics discussed above, a typical CNN used in deep learning contains other components. Like in traditional NNs, nodes in a convolutional layer not only perform weighted summation but also apply nonlinear activation functions. The process of this operation is called the detector stage of the convolutional layer. Another important component in CNNs is the pooling layer. A pooling function aggregates data in a local area. For example, the max pooling function outputs the maximum value within a neighborhood. The pooling layer always follows the convolutional layer, and applies pooling functions on it. This operation is important to the object recognition task, since it can make the model insensitive to small translations such as input rotation.

2.2.2 RNN

Another popular model used in deep learning is the RNN. As introduced in the NN section, the RNN has feedback loops embedded in its structure (See Figure 6 (a)). If unfolded, any two adjoining layers in RNNs are one-to-one connected, and the layer with feedback has one-way connections between

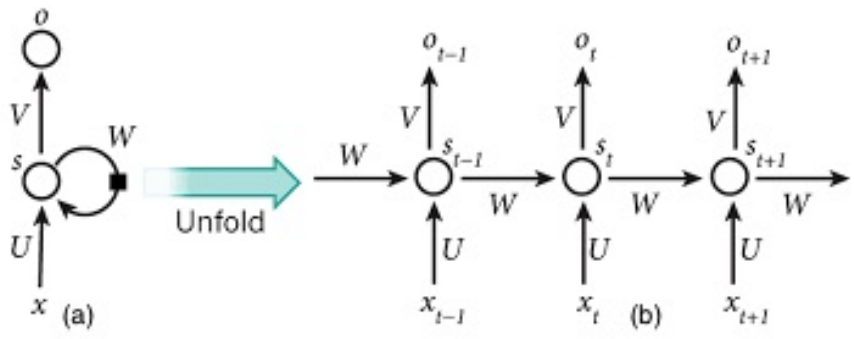

Figure 6: Folded and Unfolded RNN [WildML, 2015].

adjacent nodes (See Figure 6 (b)). If the input data is sequential, the output can be thought of as having memory about the past. The operation of one RNN layer is comparable to applying an infinite impulse response (IIR) filter.

In CNNs, nearby local data information is collected with shared weights, and the small set of data is analyzed as a whole package. In RNNs, however, relationships between neighborhood are retained in a recurrent way. Let the nth input be $x[n]$, and the hidden layer operation be a simple linear function $h[n] = w(h[n-1] + x[n])$. Then, the first output is $y[1] = w(x[1])$, and the second output is $y[2] = w(w(x[1]) + x[2]) = w^2(x[1]) + w(x[2])$. In the same manner, $y[n] = w^n(x[1]) + w^{n-1}(x[2]) + \cdots + w^2(x[n-1]) + w(x[2])$. If $0 < w < 1$, the contribution of the past sample exponentially decreases. By training, the model can find the best w that holds proper past information in order to analyze local data and provide accurate output. As a result, RNNs aggregate input data with a wider consideration than CNNs, which may more closely resemble how the brain behaves. However, it is much more difficult to train a recurrent model than a feedforward model like CNN.

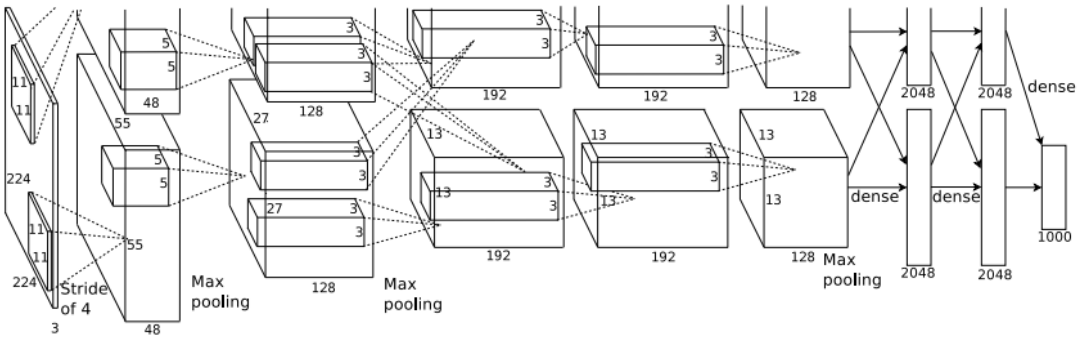

Figure 7: AlexNet Structure.

2.3 Practical CNN structures

2.3.1 AlexNet

As discussed before, the network designed by the University of Toronto team won the 2012 ILSVRC and started the age of using pure supervised CNN in deep learning, especially in the computer vision field. This network is called AlexNet, named after the main team member Alex Krizhevsky. AlexNet has five convolutional layers, three max-pooling layers, two fully-connected layers, and a final 1000-way softmax layer. In total, this network contains 60 million parameters and 500,000 neurons [Krizhevsky et al., 2012]. Figure 7 illustrates the overall structure of this network. In addition to starting the new age of CNN, AlexNet also made Rectified Linear Units (ReLUs) the standard activation function used in deep networks, replacing the traditional saturating nonlinearities such as the hyperbolic tangent and logistic sigmoid functions. ReLu is defined as $f(x) = max(0, x)$. ReLUs allow a network to obtain sparse representations and achieve efficient computations. Also, they are in accord with the behavior of real neurons that encode the intensities of inputs in action potentials .

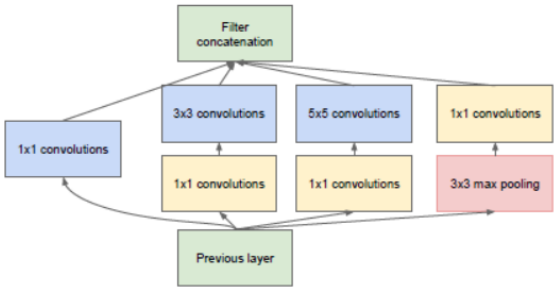

Figure 8: Inception Module.

2.3.2 GoogLeNet

After the success of AlexNet, research groups around the world competed against each other to apply the model to different tasks, and tried to modify it to get more computational efficiency and achieve better performance. In 2014, the winner of ILSVRC was the Google team. They named their model GoogLeNet to show respect to Yann LeCun who contributed to the earliest CNN development and designed the LeNet model for handwritten recognition.

The most creative part of GoogLeNet is its modular design named Inception (See Figure 8). Instead of using one cluster of filters with the same length in one layer, the Inception module groups clusters of filters with different lengths, and the GoogLeNet is comprised of stacks of Inception modules. In this way, it keeps sparse representations and increases the density of the model at the same time. Figure 9 summaries the overall structure of GoogLeNet.

2.3.3 VGGNet

VGGNet is the other deep CNN structure that yielded similar high performance with GoogLeNet in the 2014 ILSVRC

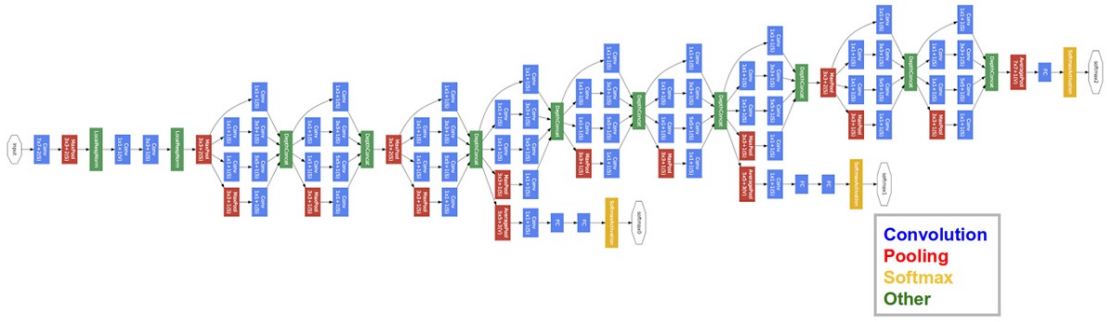

Figure 9: GoogLeNet Structure.

[Simonyan and Zisserman, 2014]. It has a simpler architecture than GoogLeNet, and is more like the AlexNet with more layers (16 to 19 layers). To reduce the number of parameters, only 3x3 filters are used in all convolutional layers.

2.4 Biological Visual and Auditory System

2.4.1 Biological Visual System

Animals and humans rely heavily on vision for survival. The biological visual system contains neurons that represent the different aspects of any received visual images, and the brain completes the task of identifying and assigning meaning to the visual objects.

Although people may take seeing for granted, the biological visual system is actually an ingenious and complex structure. Its tasks include receiving light patterns, extracting information, and forming perception. The most amazing fact is that all of these tasks are performed with structurally similar units - neurons. As mentioned before, the receptive field is the key concept that inspired the development of CNNs. Comparing the properties of receptive field, neurons along the visual system pathway can also help us better understand how animals and humans perform the visual perception.

The receptive field for a neuron in the visual system is a region on the retina. When the receptive field is stimulated, the membrane potential of the corresponding neuron will change. It can been seen from Figure 10 that the retina is not a single structure. The outer nuclear layer is formed by photoreceptors that contain light/dark or color sensitive photopigments. The receptive field of one photoreceptor is simply the small spot it occupies. Above the photoreceptors, there is another layer of cells called bipolar cells. Horizontal cells are between the photoreceptor and the bipolar cell layer. As indicated by the name, the horizontal cells are horizontally connected with photoreceptors and pass their responses to bipolar cells. Starting from this point, the receptive field enlarges. One bipolar cell receives messages from multiple photoreceptors connected by the horizontal cell. Thus, the receptive field of a bipolar cell is no longer a single spot. It includes a circular center area and a surrounding area. For the center area the bipolar cell receives direct photoreceptor input, and for the surrounding area it receives indirect photoreceptor input via the horizontal cell. Figure 11 illustrates this indirect pathway. The signal flow until this point is slow, since both the photoreceptor and the bipolar cell respond to stimulation on the receptive field by membrane potential changes. The game changes when the signal flows to the next layer of cells, the retinal ganglion cells. The retinal ganglion cell is a type of neural cell that can generate action potential along the axon, so it ensures the fast, static and relative long distance signal flow. The receptive field of the retinal ganglion cell is similar to the one for the bipolar cell,

which has the center-surround organization.

The next major players in the visual system relay race are the lateral geniculate nucleus (LGN) of the thalamus and the primary visual cortex. The LGN is a part of a knee-like structure in the thalamus. The receptive fields of LGN neurons are almost identical to the ganglion cells that feed them. The synaptic target of LGN is the primary visual cortex, which is also called V1 or striate cortex. The neocortex in general appears about six layers when stained which suggests it may have the ability to handle complex tasks by coordinating the different laminated structures. Figure 12 shows how monocular information carried by the axons of the LGN cells is mixing in the primary visual cortex layers. It illustrates that the LGN neurons project to the layer IVC of V1, and then the neurons in the layer IVC diverse and project to the layer III. So a neuron at the spot where left eye information is overlapped with right eye information has a receptive field that is the sum of the receptive fields of the neurons feed to it.

Moving beyond V1, there are two major pathways: one projects dorsally toward the parietal lobe, and other projects ventrally to the temporal lobe. Figure **??** illustrates the projection pathways and the locations of the involved areas in the macaque monkey brain.

Along the visual pathway to V1, detailed and fragmentary visual information is continuously transmitted along neural axons, emphasized or ignored at synapses, and integrated by the following neurons. As a result, the V1

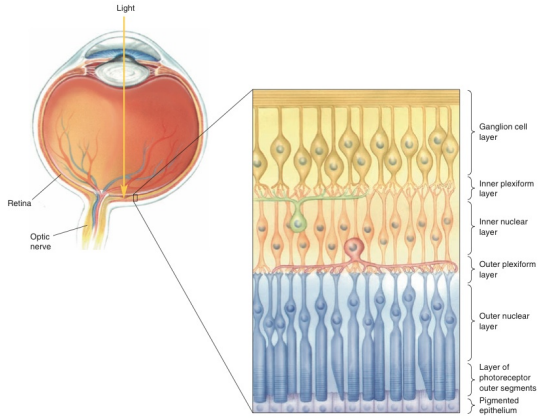

Figure 10: The Structure of Human Retina [Bear et al., 2007].

neurons own bigger receptive fields and hold more visual information. Moving away from V1, visual information starts to combine with other sensory information, and the neurons at these areas have the potential to represent more vivid objects. In terms of functionality, the ganglion cells at the beginning of the pathway appear to be sensitive to elementary variables such as contrast and light wavelength. The neurons in V1, on the other hand, have properties such as orientation selectivity and bionocularity. Finally, the other cortical areas that receive projections from V1 are believed to be responsible for more complex shapes, object motion, and even facial recognition [Bear et al., 2007].

2.4.2 Biological Auditory System

Although auditory systems and visual systems take different inputs and process them differently, both systems obey a similar relay pattern on their information flowing pathways (See Figure 14). Sound is transformed from air vibration via solid vibration to fluid vibration in the middle ear and inner ear. The snack-shell shaped, fluid-filled cochlear in the inner ear behaves like a spiral

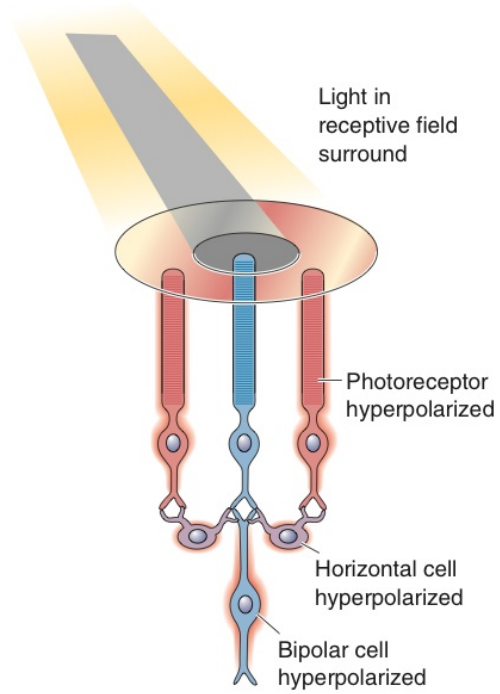

Figure 11: Indirect Pathway from Photoreceptor to Bipolar Cell [Bear et al., 2007].

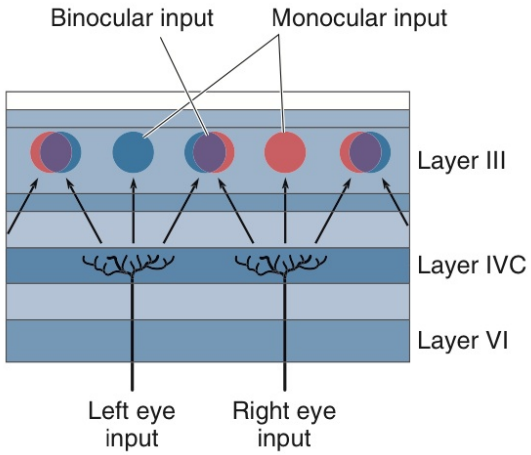

Figure 12: The Mixing of Monocular Information in V1 [Bear et al., 2007].

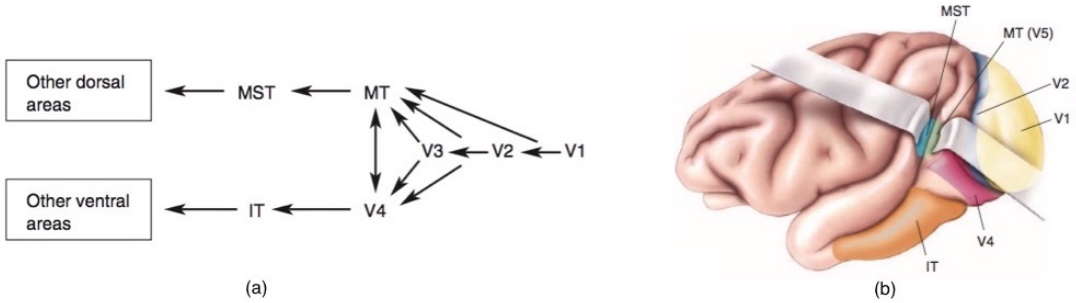

Figure 13: (a) The Major Projection Pathways from V1 and (b) the Related Areas Locations in the Macaque Monkey Brain [Bear et al., 2007].

piano. It plays the notes of the sound source by maximally deforming the corresponding key positions on its basilar membrane. Then, spiral ganglion cells connected to different positions pass the separated frequency information along with other sound properties to the nuclei in the brain stem. After that, the information is sent to the medial geniculate nucleus (MGN) in the thalamus, and projects to the primary auditory cortex (A1) from there.

As in the visual pathway, neurons along the auditory pathway can response to more complicated patterns. For example, some neurons in the cochlear nuclei are sensitive to sounds varying in frequency with time, and some neurons in the MGN can respond to complex sounds such as vocalizations. However, it seems the simple frequency selectivity property is carried on from the auditory nerve to A1 (See Figure 15) [Bear et al., 2007].

2.5 Tools for Deep Learning Implementation

2.5.1 Deep Learning Libraries

Constructing a deep learning network is like stacking building blocks, but the operations to strengthen the structure are not easy. Deep learning libraries

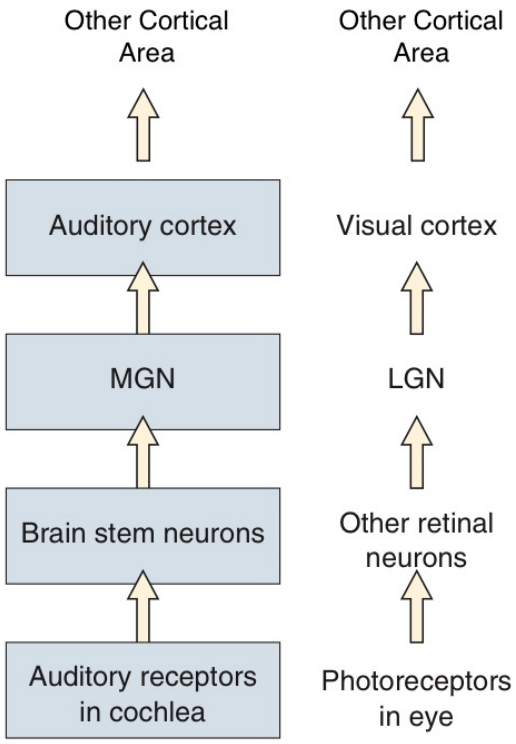

Figure 14: Auditory and Visual Pathways Comparison

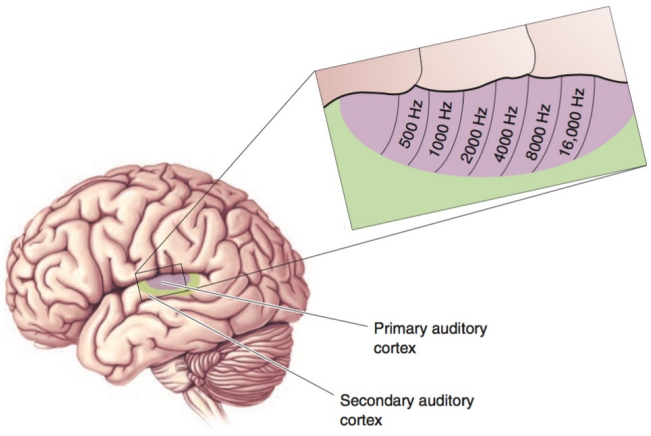

Figure 15: Tonotopic organization in A1 [Bear et al., 2007].

provide the building blocks and simplify the strengthening calculations.

The deep learning libraries can be grouped into two categories: the imperative style and the symbolic style. Imperative style programs run the computations line by line as instructed. Symbolic style programs first construct the rule pattern symbolically, then convert the pattern into a function so it can be called with different parameters.

The imperative program is more flexible since it is easier to access conditionals and loops in the host language. The most famous imperative style deep learning library is Torch. Torch is written in Lua, a fast and light weight scripting language. It has been used in large tech companies such as Google and Facebook due to its flexibility and fast speed.

In contrast, the symbolic program is more restricted which allows it to optimize the underline calculations. A large portion of deep learning libraries belongs to the symbolic style, including Theano and TensorFlow. Developed by University of Montreal since 2007, Theano is one of the most widely used Python deep learning libraries. There are many higher-level Theano-depent libraries such as Lasagne, Blocks, and Keras that simplify model definition and updating. TensorFlow was released by Google in 2015. It views data feeding and training as tensors (multidimensional arrays) flow via the mathematical operation nodes. It has both Python and C++ interfaces.

MXNet is another useful deep learning library. It mixes the imperative and symbolic styles together by including symbolic programs as part of

imperative programs. Apart from taking advantages from both of the styles, it also performs very well on memory saving.

2.5.2 Graphics Processing Unit

The graphic processing unit (GPU) is a specialized processor, which is originally designed for rendering images on a computer monitor. Rendering images especially the high-resolution 3-D graphics for videos and games, is a mathematically intensive task. Massive calculations need to be done on redrawing properties such as lighting, effects and object transforming. GPUs are designed to have many parallel processing cores, which allows for efficient calculations on large amounts of data.

Scientists and engineers soon realized they can use the power of GPUs to accelerate vector and matrix calculations, which account for a large proportion of today's scientific projects and practical applications. The GPU-accelerated computing approach is called general-purpose computing on graphics processing units (GPGPU). In order to enable GPGPU, the programmer needs to use specific programming frameworks to access the GPU's computational elements. Currently, the major GPU programming framework for GPGPU is CUDA, which is designed to work with C, C++ and Fortran. CUDA only supports NVIDIA GPUs. Another popular framework is OpenCL which is not vendor dependent.

3

Previous Work

3.1 Previous Studies in the Imaging Field

A great many imaging-related tasks have been done using CNN. The works discussed in this section inspired the transformation task for this thesis.

3.1.1 DeepDream

Although the performances of CNNs on visual object classification tasks approaches to human accuracy, it is still not fully understood why these models work and how to improve them. The most straightforward way to study CNNs is to visualize what happens in each layer. Engineers at Google implemented a program to visualize different layers in CNN. Because of the hallucinatory appearance of the images generated, the program was named DeepDream.

In order to visualize a particular layer of a trained CNN, the program runs a gradient ascent process that maximizes the root mean square of the activations for that layer. The source code is posted on https://github.com/google/deepdream/blob/master/dream.ipynb. The input can be any image or even 2D random noise. In the example shown with the open source code, the input is an image of a sky with clouds (Figure 16). If a lower layer is chosen, the generated image obtains many basic ornamented patterns as seen in Figure 17. In contrast, the image that enhances a higher layer tends to include complex patterns or even a complete object (Figure 18). These results are in accord with what researchers expected. One hypothesis of CNN is that the

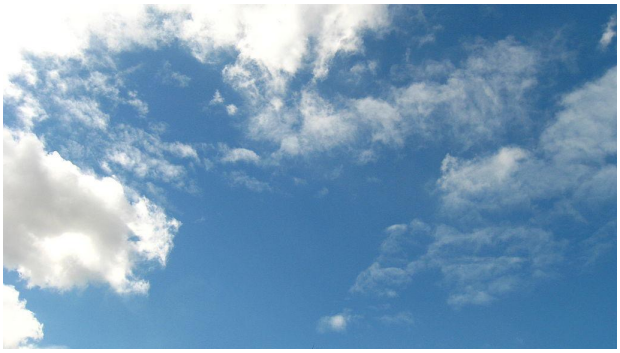

Figure 16: Original Clouds Image

lower layers are more detail orientated, and as the structure goes deeper it will extract more abstract concepts. This hypothesis was developed according to the discoveries of the structure and function of mammalian visual systems. The DeepDream program shows that the CNNs work in a way similar to how animals and humans process images.

By changing the objective function that the gradient ascent algorithm tries to maximize, an input image can have objects that are from a reference image. In order to update the input image to make its output in a particular layer closer to the output of the reference image in the same layer, the objective function can be rewritten as the dot product between these outputs. Since the dot product measures how close two vectors are, the gradient ascent process will make the input image have similar features with the reference image. Thus, the input image will change to look like the reference image in some way (Figure 19). This achievement indicates the DeepDream technique not only helps scientists understand how a CNN works but also has the potential to become a tool for artists to generate creative art works.

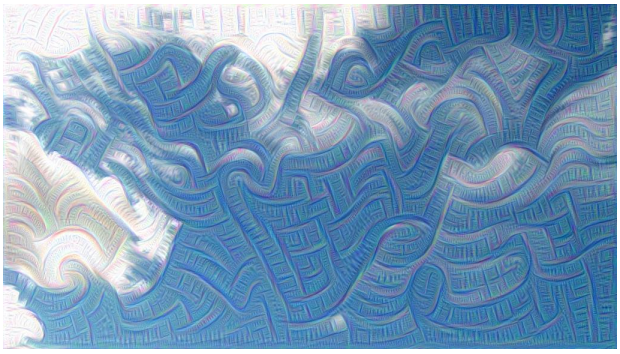

Figure 17: Modified Clouds Image Based on a Lower Layer

Figure 18: Modified Clouds Image Based on a Higher Layer

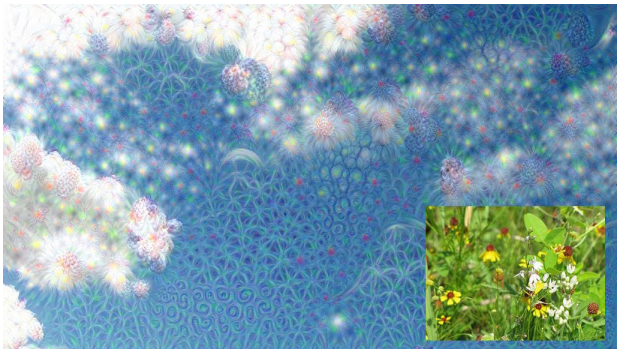

Figure 19: Modified Clouds Image with Flowers Image as reference (Right Bottom)

3.1.2 Inverting Image Representations

The features extracted from an image can be viewed as some representations of the image. For example, an image can be summarized or represented by the features generated with traditional imaging processing methods such as Scale Invariant Feature Transform (SIFT) or Histogram of Oriented Gradients (HOG). This holds true for the features or intermediate outputs generated from the layers of a CNN as well.

Mahendran and Vedldi from University of Oxford developed an approach to understand the representations by inverting them into the original images [Mahendran and Vedaldi, 2015]. Although the process was described as "inverting", it was actually a feedforward stream starting from random noise, and the input was gradually modified to match its feature output with the representation being inverted. Therefore, it is similar to the DeepDream with reference image, but instead of maximizing the dot product of the representations for the reference and input image, it minimizes the Euclidean distance between the representations. Figure 20 shows the "inverting" results for different layers of a AlexNet. The results indicated that the lower layer representations give better reconstructions. This is in accord with what have been learned from the DeepDream program that the lower layers contain the detail information about the input image such as edges and orientations, and the higher layers care more about the abstract concepts related to the whole objects.

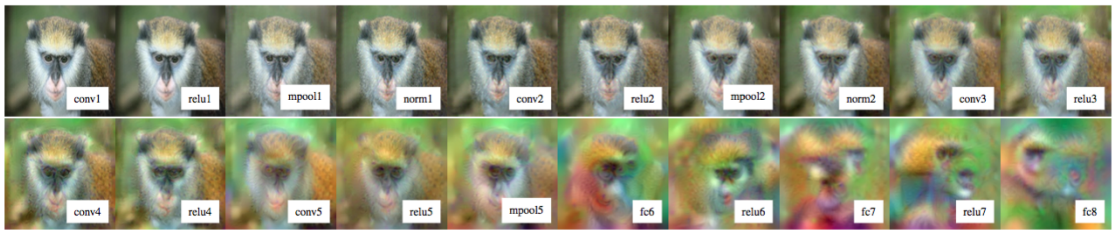

Figure 20: Reconstructions from Different CNN Layers (Top Left is from the First Layer and Bottom Right is from the Last Layer)

3.2 Previous Studies in the Audio Field

In contrast to the abundant imaging applications with CNN, the audio applications using CNN are rare. Although many researchers in the audio field tried to apply different deep learning techniques to various audio tasks, there are only a few audio related studies have been done using CNN. It is probably because CNN was inspired by the biological visual system especially the receptive field, but this concept is not emphasized in the auditory system. However, as discussed above, the biological auditory system has a comparable hierarchy structure with the visual system, so the laminated network should fit to the audio task as well. Moreover, the traditional way to analyze audio are applying a moving window with fixed length like short-time-Fourier-transformation (STFT) or applying a moving window with flexible length like wavelet-transform. The windowed processing segments are relevant to the receptive fields referred in the image recognition using CNN, although they are different from the biological receptive fields concepts. As a result, it is feasible to use CNN in audio-related tasks.

3.2.1 Unsupervised Audio Feature Learning with CDBN

Convolutional Deep Belief Networks (CDBN) is a combination of CNN and DBN. Instead of using the feedforward layers, it stacked the RBMs as the processing units, but same as CNN it deploys shared weights, and performs convolution-like operations. In 2009, Lee et al. [Lee et al., 2009] applied CDBM to unlabeled auditory data(speech and music), and evaluated the learned feature representations. By comparing the performances in speaker identification, phone classification, music genre classification, and music artist classification, they found that the CDBN-learned features outperform other baseline features such as spectrogram and mel-frequency cepstral coefficients (MFCCs). In addition, they found that in the case of speech data, the learned features corresponded to phonemes. Moreover, they showed that in certain classification tasks, the features learned from the second CDBN layer had better performance than the features learned from the first CDBN layer. It indicates the representations generated from higher layer may have more comprehensive information than the ones generated from the lower layer for auditory input.

Later on, Sander Dieleman and his colleagues trained a CDBN to perform artist recognition, genre recognition and key detection on the 'Million Song Dataset'. They found that the approach improved accuracy for genre recognition and artist recognition tasks [Dieleman et al., 2011].

3.2.2 Music Recommendation using CNN

Aaron van den Oord et al. used CNN to train a model to predict the collaborative filtering latent representations of songs [Van den Oord et al., 2013]. Collaborative filtering is a widely used approach in the recommendation field. It uses the information collected from the usage data to determine the users'preferences. The drawback of this approach is that it does not consider any content information, so there is bias on the recommended items. The authors tried to solve this problem by inputting 30 second music mel-spectrograms (content) to a CNN model in order to predict the latent representation of the song that obtained by the collaborative filtering method. The CNN structure used in their study consisted of two convolutional layers followed by two fully connected layers.

One of the authors, Dieleman, analyzed what has been learned in a similar model with four convolutional layers and three dense layers. He did several experiments to understand what each layer learns from the training. By searching for the songs that has at least one frame maximally activate some low-level filters, he found that different filters could pick up different representations of songs such as vibrato signing, ringing ambience, vocal thirds and bass drum sounds. He also studied the songs that maximally active other low-level filters on average across 30 seconds, and found that different filters selected different representations of noise and distortion, specific pitch, drones, and specific chord. In addition, he tested out the topmost dense layer

(high-lever) filters, and found that one of them can pick up Chinese pop. His work was posted at http://benanne.github.io/2014/08/05/spotify-cnns.html.

Another group from National University of Singapore also had a similar idea on using CNN on music recommendation[Wang and Wang, 2014].

3.2.3 Other Audio-Related Tasks using CNN

A research group from Hong Kong proposed an approach to classify 10 music genres from MFCCs using a CNN with three Convolutional layers and one fully-connected layer. They got 84% classification accuracy and suggested that CNN had strong capacity to capture informative features from various musical patterns with minimal prior knowledge [Li et al., 2010].

Humphrey and Bello from NYU Music and Audio Research Laboratory (MARL) used CNN for automatic chord recognition [Humphrey and Bello, 2012]. They input the constant-Q transformed time-pitch array into a CNN with three convolutional layers with one optional pooling layer and two fully connected layers to recognition the music chord, and get about 77% accuracy.

Schlüter and Böck from Austrian Research Institute applied CNN to distinguish musical onsets from non-onsets. They feed a stack of three spectrogram excerpts into a CNN with two convolutional layers with max-poolings followed by two fully connected layers, and found if slightly surpass the best existing method [Schlüter and Böck, 2013].

3.2.4 End-to-End Learning for Music Audio

When using image as the input to CNN, only a few pre-processing steps such as resizing and color normalization were applied on the original image. One idea behind the deep learning was to let the network learn the features by itself and use it as an end-to-end processing agent to avoid the extra feature extraction procedures. However, almost all of the studies of applying CNN on audio data used the extracted features such as spectrogram and MFCCs as input. Spectrogram and MFCCs do contain a lot of information about the original audio, but there will still be some loss. Dieleman and Schrauwen investigated the feasibility of applying CNN directly on raw audio signals to do a automatic tagging task. They discovered that although the raw-signal-based approach did not outperform the spectrogram-based approach, the CNN could achieve the frequency decompositions, and recognize phase - and translation - invariant patterns from the raw input audio signal [Dieleman and Schrauwen, 2014].

4

Proposed Method

4.1 Dataset Preparation

4.1.1 Music Styles Selection

One of the music styles selected for this thesis study is Chinese popular music (C-pop). The reason of this selection is because Dieleman's experiment on recommending music with CNN[Dieleman, 2014] showed that C-pop music could maximally activate one of the high-level filters in his model. He suggested that the CNN model can either identify the Chinese language or pick some unique characteristic in C-pop music. One of the goals of this thesis study is to modify the input audio signal to increase the activations from a particular convolutional layer, so some extra elements stored in this layer can be obtained along with the original audio signal. According to this goal and what Dieleman's experiment, C-pop is a good candidate music style.

C-pop includes Mandarin popular music and Cantonese popular music, but the differences between these two types are mainly the languages instead of music characteristics. Although C-pop today is similar with popular western music, it tends to be slower and softer. Also, many traditional Chinese instruments and melodies are integrated in today's C-pop.

The other music style chosen to be distinguished from C-pop is melodic death metal (MDM). MDM started from the early 90 's. It uses double-bass drum and heavily distorted guitars, but also has harmonic bass lines and guitar

riffing. In addition, its vocals combine harsh screams, growls and clean harmonies[Wikipedia, 2016]. The death metal parts make MDM distinguishable from the slow and soft C-pop, but the melodic elements facilitate it to merge into C-pop at the same time.

4.1.2 Pre-processing

The original MP3 files of C-pop were downloaded from http://music.163.com/#/playlist?id=25566257. The webpage contains a randomly selected C-pop playlist from NetEast Cloud Music database. The MDM MP3 files were downloaded from http://music.163.com/#/playlist?id=13357861, which includes MDM from all over the world other than Finland and Sweden. The reason to choose this playlist is because the songs included have many variabilities which will increase the fault tolerance of the model, and ensure the classification not heavily depends on languages.

Seventy downloadable C-pop MP3 files in the playlist were processed. In order to decrease the calculation complexity, the audio signals obtained from the mp3 files were down sampled to 8000 Hz. The low sampling rate reduced the audio quality, but the two music style were still distinguishable. Every track was loaded and resampled using an audio signal processing Python package name *librosa*. By setting the mono parameter to be True, and the sampling rate (sr) parameter to be 8000, every track was converted to mono signal with 8000 sampling rate. The resampling paradigm depends on another Python module,

samplerate, which is a wrapper around the high quality sampling rate conversion library *Source Rabbit Code* written by Erik Castrop de Lopo. After loading, every track was normalized with the maximum value as nine-tenth of the maximum value in the original track. Then, the first and last 30 seconds audio data from each track, which include 240,000 points were removed to avoid silence sections as well as unusual intro and coda. After that, the remaining sections were separated into 5-second (40,000 data points) clips. The separation was implemented continuously along each track. If the last segment did not have 40,000 data point, it will be discarded. At the end, 2658 clips were generated from the 70 C-pop tracks. In the classification task, 2400 clips were used in training, and 240 clips were used in testing. The MDM tracks were processed with the same procedure. In summary, 4800 5-second audio clips with 2400 in each music style category were used to train the proposed CNN model for the binary classification, and 480 clips with 240 in each category were used to test the accuracy of the model.

4.2 Music Style Classification using CNN

The first task in this thesis study was applying the proposed CNN model to classify the C-pop and MDM audio clips. Since not many studies have been done using CNN on raw audio signals, the network structure and parameters were chosen based on little prior knowledge.

4.2.1 Network Structure and Model Parameters

In order to compromise calculation complexity and be compatible with Dieleman's experimental model, three convolutional layers were used to construct

the CNN. In addition, one fully-connected layer with 256 nodes was placed after the convolutional layers, and it was followed by the output layer that was a fully-connected layer with 2 nodes. Moreover, four normalization layers were added respectively after input layer and each convolutional layer. The pooling layers were not applied in the proposed model because they were designed to increase the spacial invariance, and if using on audio data they will blur the time frames which are important for music.

Input Layer

The input layer of the proposed network model had the shape of $batch\_size \times 1 \times 40,000$. The $batch\_size$ is the number of clips used for the mini-batch gradient descent optimization which will be discussed in the next section. The second dimension is the number of channels, and since the inputs are mono clips, this number is one. The third dimension is how many data points contained in one clip. For image processing tasks, there are four dimensions with the last two representing the height and width of the image.

Convolutional Layers

All of the three convolutional layers were set up with Glorot Uniform distributed initial weights and zero bias. Let W_i represent an initial weight, then the Glorot Uniform distribution gives $W_i \sim U[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}]$ where U[-a, a] is the uniform distribution in the interval (-a, a) and n is the number of nodes in the previous layer [Glorot and Bengio, 2010]. In addition, the convolutional layers used the rectify function $f(x) = max(0, x)$ as the activation function.

The first convolutional layer had 8 filters with size 8, and the stride size was 2. It generated an intermediate output with shape $batch\_size \times 8 \times 19,997$ after feeding the $batch\_size \times 1 \times 40,000$ shape input. The second convolutional layer had 8 filters with filter size 32 and stride size 4, which gave an intermediate output with shape $batch\_size \times 8 \times 4992$. The last convolutional layer in the proposed model also had 8 filters and used stride size 4, but the filter size was 128. This generated a $batch\_size \times 8 \times 1217$ shape intermediate output.

The number of filters and filter size for each convolutional layer were chosen arbitrarily. However, if the filters were learned to form a sinusoid pattern, the minimum possible frequency could be calculated as $\frac{sampling\_frequency}{filter\_size} Hz$. Therefore, the filters from the three convolutional layers would have the ability to analyze components with minimum frequencies as $1000Hz$ (which is $\frac{8000}{8}$), $250Hz$ (which is $\frac{8000}{32}$), and $62.5Hz$ (which is $\frac{8000}{128}$) respectively.

Normalization Layers

Each normalization layer implemented a batch normalization on the intermediate result generated by the convolutional layer it followed. Let x represent the mini-batch result from the previous layer then the normalization was done using the function $f(x) = \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}}$. ϵ is a small constant 10^{-4} to avoid numerical issue. During training, μ and σ^2 are the mean and variance of the current mini-batch data across the batch dimension, while during testing they are the statistics of the whole training dataset across the example dimension.

Fully-Connected Layers

The fully-connected layer has all nodes from the previous layer connecting with each node in its own layer. It is used in the traditional NN, and always deployed in CNN in the last layers. As mentioned before, the last two layers in the proposed CNN were fully-connected layers. The first one had 256 nodes, and its other setups were same with the convolutional layers. The second fully-connected layer had 2 nodes which represented the two music style categories, and its activation function was a softmax function. The softmax function gave the probabilities of both categories, and the one with higher probability would be the predicted category from the network.

4.2.2 Optimization

Cost Function

The classification in the first task conformed to the supervised learning paradigm, and the cost function measured the distance between the prediction and the target. Since the prediction was calculated using the softmax function, it gave the probabilities of both classification categories instead of one nominal representation of the predicted category. To take full advantage of the information given by the probabilities, the binary cross-entropy was applied to computer the distance. Let p represent the probability of the prediction for one category, then for the cost or loss L was calculated as $L = -t\log(p) - (1-t)\log(1-p)$ with t represent the probability of the prediction for the same category. Since the target was deterministic, so t was either 0 or 1.

Optimization Method

The goal of training a network is to optimize the cost function by updating the parameters in the model. The optimization approach used in the classification task was mini-batch gradient descent, which took a subset of the full training dataset in each iteration, and updated the parameters based on the gradient of the aggregated loss (the mean of the cross-entropy losses for the batch set).

The specific update method applied was based on *Nesterov's accelerated gradient descent*, and for each parameter θ, it was implemented as $v_{n+1} = \mu * v_n - \alpha * \frac{\partial L}{\partial \theta_n}$, and $\theta_{n+1} = \theta_n + \mu * v_{n+1} - \alpha * \frac{\partial L}{\partial \theta_n}$. The variable μ is called momentum and it was set up as 0.9; another variable v is called velocity which was initiated as 0; α is the learning rate which equaled to 0.01, and $\frac{\partial L}{\partial \theta_n}$ is the partial derivative of the aggregated loss function with respect to the parameter being processed.

Training and Testing

As mentioned before, the training set had 4800 examples in total (2400 C-pop audio clips and 2400 MDM audio clips). Before training, the examples were shuffled to avoid continually feeding the network with examples from one song and one music type. For each mini-batch, there were 480 examples, so each epoch had 10 (which is $\frac{4800}{480}$) iterations. Twenty epochs were applied and the training loss and training time for each epoch were recorded.

When doing the testing, 480 examples (240 for each style) was used. The

percentage of testing examples was low since not many songs were downloadable due to the music copyright, and the transformation task would have better performance with more training data. The batch size for testing was 48. The test loss and test accuracy were recorded.

Programming Language and Dependent Libraries

When training a multi-layer network, calculating the gradient is usually complicated, especially for CNN. Many programs and software packages were written to accomplish the task. For this thesis study, the experimental code was written in Python programming language, and the major library used was Lasagne. As discussed in the background section, Lasagne was built on top of Theano library, and it made constructing and training network model easily. For example, the command of adding the second convolutional layer was $net['layer2'] = lasagne.layers.Conv1DLayer(net['layer1Norm'], 8, 32, stride = 4)$. 'net' was a dictionary that held the structure and all parameters of the network model. 'layer2' was the name given to this layer. 'Conv1DLayer' meant it was a convolutional layer that handled one dimension data. 'layer1Norm' was the name of the previous layer, and that layer was feed into the 'layer2'. In addition, it specified that 'layer2' had 8 filters with filter size 32, and the stride size was 4.

4.3 Music Style Transformation using CNN

The music style transformation task can be viewed as an audio analogue to the Deep Dream project mentioned before. A randomly selected C-pop clip was fed into the CNN model trained in the classification task, and the input was

modified to give the maximum output values for different layers.

4.3.1 Objective Function and Optimization Method

The transformation task also solved an optimization problem. However, unlike the classification task which updated the parameters in the network, the transformation task updated the input and kept the network parameters untouched. Other than that, instead of minimizing a cost function which represented the distance between the prediction and target, the transformation task maximized the objective function which measured the output values of a particular layer. Let O represent the objective, and y represent the output of the layer being examined, then the objective function was defined as $O = \frac{1}{2}y^2$.

The optimization method chosen to maximize the objective function was gradient ascent. Similar with gradient descent, the gradient ascent method updated the input based on the partial derivative of the objective function with respect to the variable of the individual input sample point.

4.3.2 Experiments with Different Layers

In order to generate comparable results with the Deep Dream project and examine the CNN trained on audio style classification task, the original input clip was modified to optimize the outputs for the three convolutional layers in the proposed CNN respectively. Each modified clip was normalized to have the absolute maximum as 0.9, and saved as 16 bits wav audio file. The original clip was saved as a same format audio file for comparison.

4.3.3 Dependent Libraries

Since the transformation task was simpler in terms of optimization, the L-BFGS-B algorithm supported in scipy library was used to handle the updating. However, because this scipy function was designed to accomplish the minimization, the objective function discussed in the last section was modified to generate the opposite number.

At the same time, the Lasagne library was still used to get parameters values and generate outputs from the network, and the gradient was calculated with the Theano library.

5

Results

5.1 Results of Testing Theano with CPU and GPU

Computational speed is a crucial consideration for constructing and training complicated network like CNN. A testing program to handle 1000 iterations of exp() function with input size 230400 using Theano library was run on both CPU and GPU on the Pegasus supercomputer [HPC, 2016]. The testing result on the CPU was 2.91 seconds, and the testing result on the GeForce GTX TITAN X graphics card was 0.849 seconds. For the testing program, the CPU on the Pegasus system was 3.43 times slower than the GPU device that used for the classification and transformation tasks.

5.2 Results from Classification Task

Table 1 reported the average training loss and time spent for every epoch during the classification CNN model updating procedure. The training loss decreased as the epoch increased, and the more straightforward results were shown in Figure 21. The testing accuracy on the trained CNN network was 80.83%. The computational time for running each epoch was also shown in Table 1. The average time spent in one epoch is 11.39 second.

5.3 Results from Transformation Task

The waveforms of the original audio clip and the modified clips corresponding to the three convolutional layers were plotted in Figure 22. The

Table 1: Training Losses and Computational Times

Epoch No.	Training Loss	Computational Time (sec)
1	0.621	10.768
2	0.190	10.759
3	0.101	10.772
4	0.0638	10.786
5	0.0487	10.863
6	0.0415	10.877
7	0.0325	10.878
8	0.0256	10.891
9	0.0246	10.910
10	0.0192	10.913
11	0.0154	10.913
12	0.0135	10.914
13	0.0116	11.025
14	0.00958	12.014
15	0.00894	12.020
16	0.00872	12.667
17	0.00695	12.479
18	0.00729	12.482
19	0.00587	12.444
20	0.00554	12.469

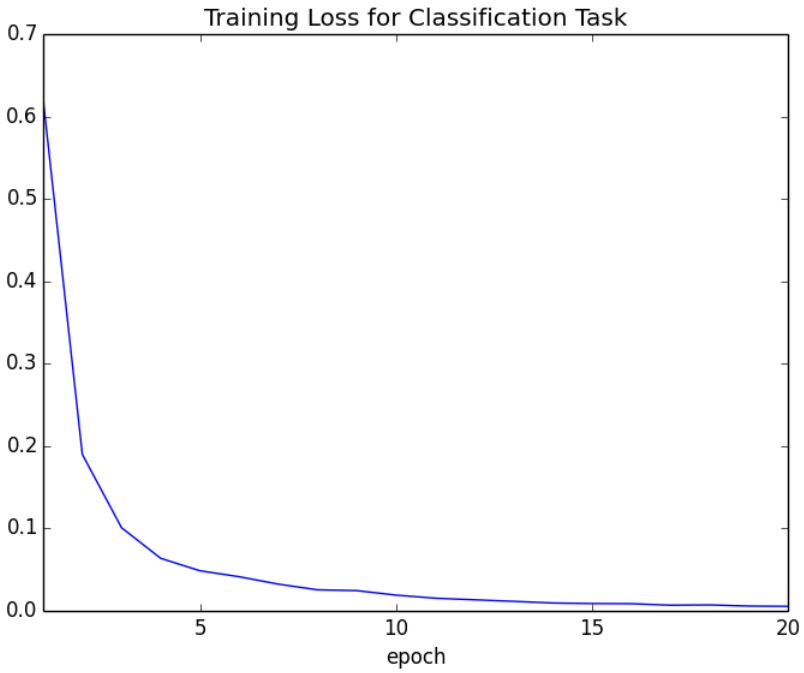

Figure 21: Training Loss for Classification Task.

modified clip based on convolutional layer1 at the bottom subplot showed a few bursts, and it was difficult to find the relation between this clip and the original clip. The modified clip based on convolutional layer2 on the second last subplot was a collection of a lot of pulses. It appeared less pulses at the time when the original audio had more energy. The second subplot from top showed the modified clip corresponded to the convolutional layer3, which captured the basic shape of the original waveform. The spectrograms with window size 512 and 50% overlap for the four audio clips were shown in Figure 23. The spectrogram of the convolutional-layer3-modified-clip indicated that although this clip had a similar waveform shape with the original one, it only captured the fundamental frequencies of the audio. Instead of showing the clear harmonics pattern as in the original clip, the modified audio developed some attack-like patterns. The sound textures of these audio clips were verified by listening to the generated wav files which could be downloaded from https://sites.google.com/site/shijiagengmuethesissourcefiles/home/audio-clip-wav-files.

The total of 24 (which is 3×8) filters of the three convolutional layers were plotted in Figure 24, 25, and 26. The patterns of the filters were not very clear, but there was a trend that the higher convolutional layer filters tended to have smaller values than the lower convolutional layer filters.

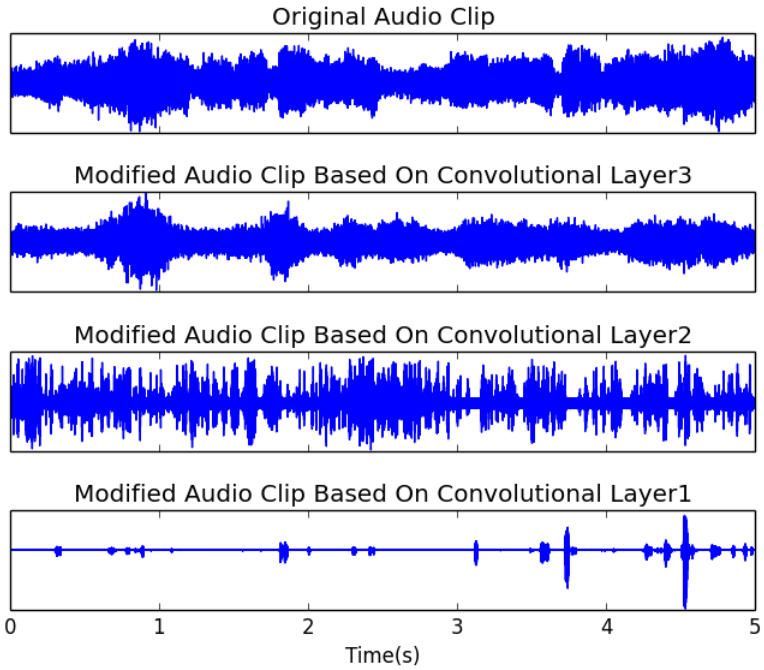

Figure 22: Audio Waveforms for Transformation Task.

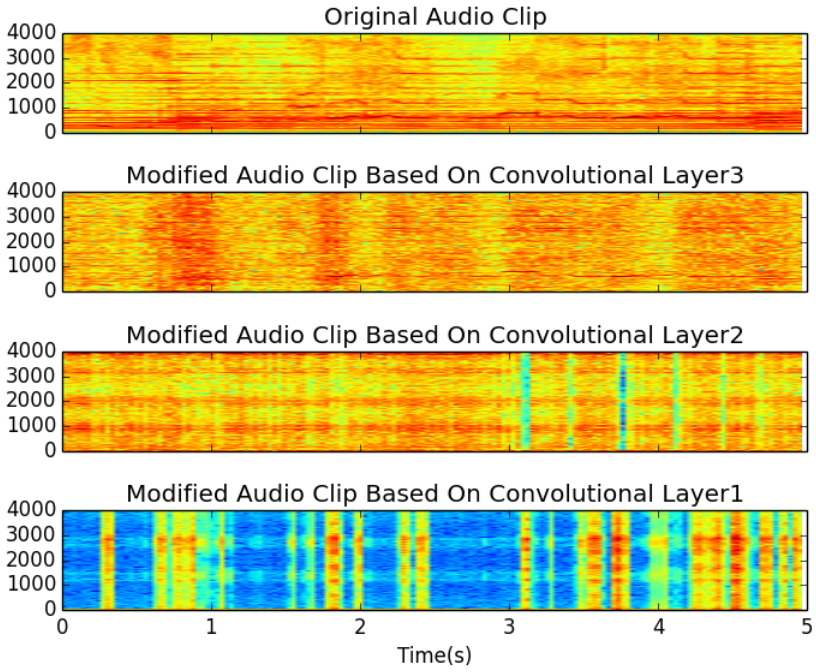

Figure 23: Spectrograms of the Audio Clips.

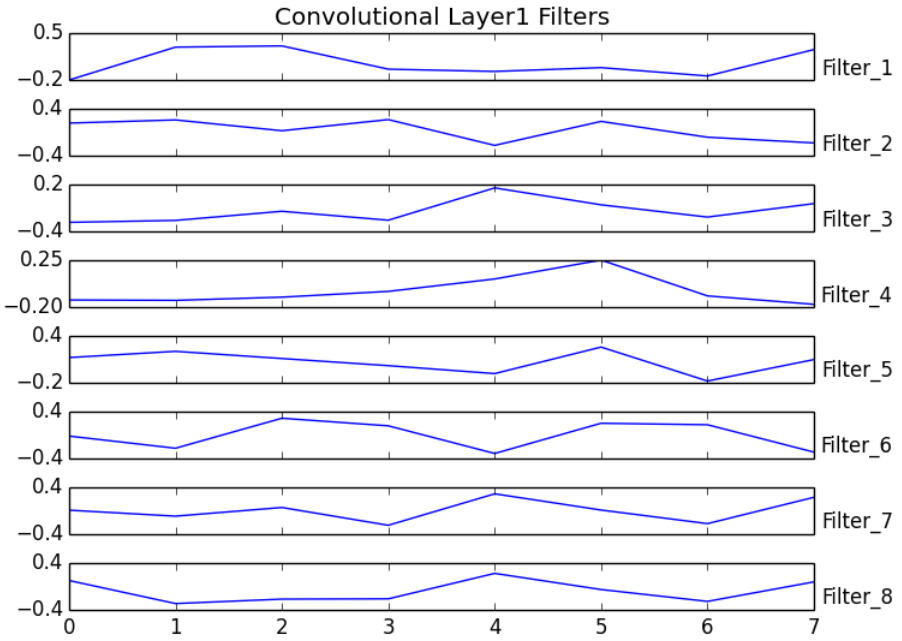

Figure 24: Convolutional Layer1 Filters.

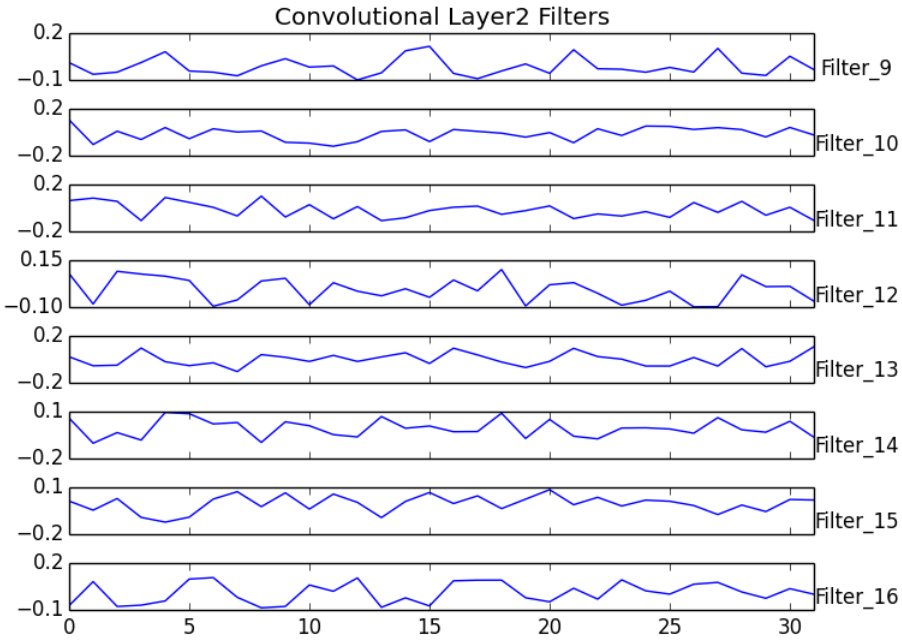

Figure 25: Convolutional Layer2 Filters.

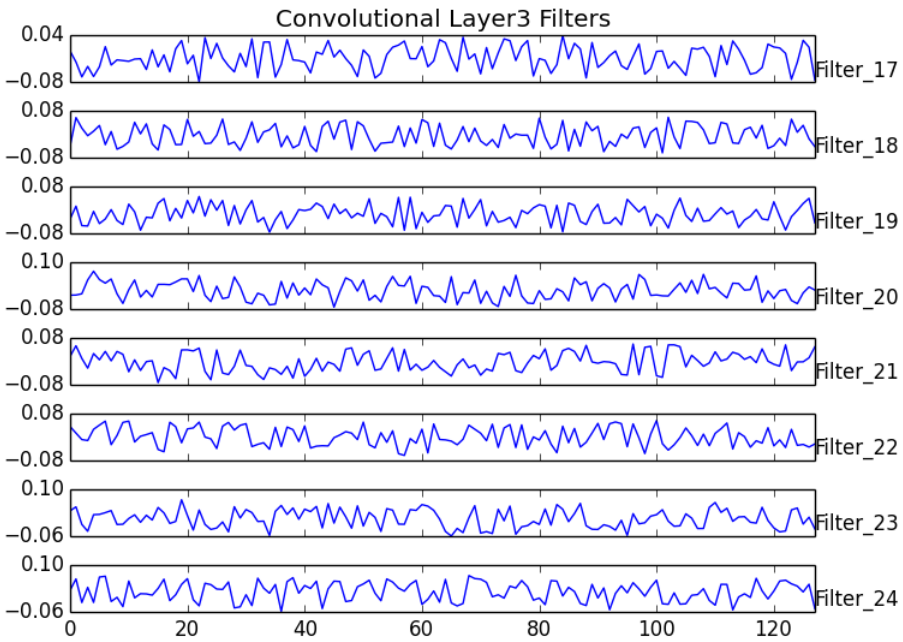

Figure 26: Convolutional Layer3 Filters.

6

Discussion

6.1 Comparison with the DeepDream Results

The generated audio clips were comparable to the images obtained from the DeepDream program. As shown in Figure 17 on page 28, the modified image based on the lower network layer got lower-level features such as elementary stroke-like patterns. Similarly, the audio clip generated corresponding to the first convolutional layer only had a few bursts, and the clip based on the second convolutional layer was only a collection of pulses. The third layer, on the other hand, produced an audio clip with melodies and attacks, which was comparable to the modified image based on the higher network layer that showed recognizable visual objects (See Figure 18 on page 28).

6.2 Shedding Light on Auditory System

It has been proven that some A1 neurons response to clicks and bursts of noise, and neurophysiologists find it is difficult to propose an unifying principle for auditory system due to the wide variety of response types [Bear et al., 2007]. If the auditory system processed audio signal in a way similar with the proposed CNN, the neurons that response to clicks and bursts will be an intermediate stage to form higher auditory perceptions such us music style differentiation. Neurophysiologists can track the projections of the neurons that response to bursts and investigate whether or not the hierarchy of auditory representations exist.

6.3 Artistic Style Representation and Future Improvements

The original goal of developing DeepDream was to understand what happened in each layer of the deep networks. However, the imaginative images generated from the program encouraged people to use it as an artistic tool. Almost at the same time, a Geman research group developed another deep network tool which was special for artistic purpose [Gatys et al., 2015]. The members in this group proposed a way to extract the texture or style from an art piece. By combining the extracted style with the content obtained from another image, Gatys and his colleagues could transform an image from one artistic style to another. For example, by combining the style of Vincent von Gogh's *The Starry Night* with the content of a building photo, they created a post-impressionism building arts work(See Figure 27).

The content extraction method was a simple version of inverting image representation implemented by Mahendran and Vedldi[Mahendran and Vedaldi, 2015]. To extract the content, an image generated from random noise was updated upon the cost function $L_{content}(x, p, l) = \frac{1}{2}\sum(F^l - P^l)^2$. The matrices F^l and P^l represented the outputs from layer l for the generated image x and the original image p. On the other hand, the style representation was calculated by the inner product between the feature maps. If there were N_l numbers of filters in layer l, then there would be N_l feature maps generated from this layer, and the inner product between each feature map would give a correlation map with size $N_l \times N_l$. The collection

of the inner product matrices from different layers summarized the style information by discarding the global arrangement of the content. To reconstruct the style, an image generated from random noise was updated to minimize the cost function $L_{style} = \sum w_l E_l, and E_l = \frac{1}{4N_l^2 M_l^2} \sum (G^l - A^l)^2$. The matrices G^l and A^l were the inner product matrices from layer l related to the generated image x and the original image a. Overall, to merge the content of one image p with the style of another image a, the cost function was the linear combination of the two losses : $L_{total} = \alpha L_{content}(x, p) + \beta L_{style}(x, a)$ [Gatys et al., 2015].

More interestingly, the authors pointed out that the approach providing the style representation has its biological basis. In mammal's visual primary cortex (V1), there is a type of neurons called complex cells, whose job is extracting the correlations between other neurons.

The analogous process was done with the audio clip used for the transformation task, but the result was just random noise. It was probably because that the style representation for image did not fit for audio. As discussed above, the style representation was calculated by integrating the spatial information for an image. The counterpart for audio would be integrating the time information. As shown in Figure 24, 25, and 26, the filters in each layer did not appear distinguishable, so the correlation map based on them would not have much information about the audio. As a result, it would not be able to reconstruct the style by minimizing the distance between the correlation maps from the generated audio and the ones from the reference audio. However, it was

also possible that the network was not good enough to tell the different music styles apart, which might be due to lack of training examples, lack of target classification categories, lack of network layers, and lack of restrictions for audio tasks. Therefore, more experiments need to be done based on a larger amount of audio clips, longer duration for each clip, higher sampling frequency, more music styles being classified, and different network structures. Also, more studies need to be done on restricting the network parameters and the training data to make them more suitable for audio-related tasks. In addition, it is important to find a better style representation for audio that can be extracted independently from the content.

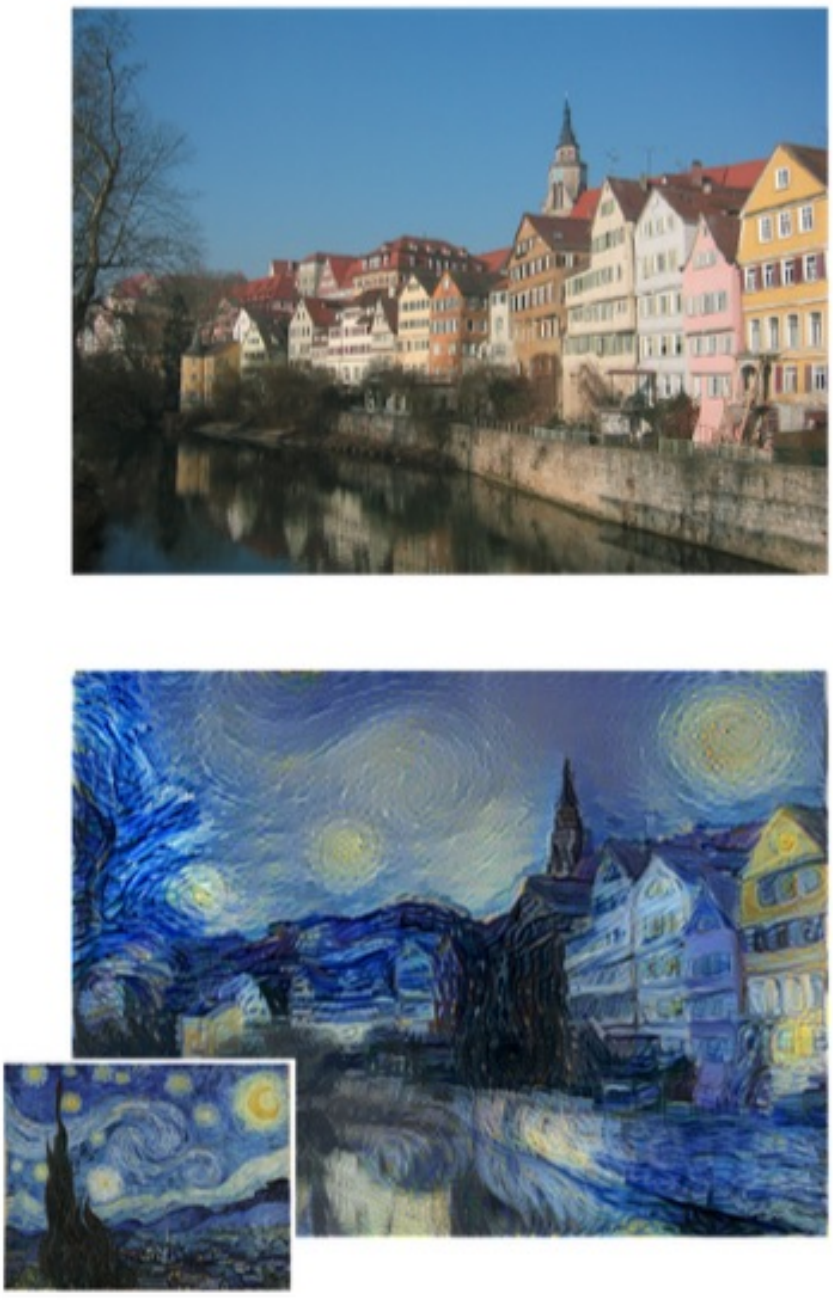

Figure 27: Original Photo (top) and Modified Artwork with The Starry Night Style (bottom) .

LIST OF REFERENCES

[Bear et al., 2007] Bear, M. F., Connors, B. W., and Paradiso, M. A. (2007). *Neuroscience*, volume 2. Lippincott Williams & Wilkins.

[Bengio, 2009] Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

[Bengio et al., 2007] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al. (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153.

[Dieleman, 2014] Dieleman, S. (2014). Recommending music on Spotify with deep learning. http://benanne.github.io/2014/08/05/spotify-cnns.html. [Online; accessed 11-March-2016].

[Dieleman et al., 2011] Dieleman, S., Brakel, P., and Schrauwen, B. (2011). Audio-based music classification with a pretrained convolutional network. In *12th International Society for Music Information Retrieval Conference (ISMIR-2011)*, pages 669–674.

[Dieleman and Schrauwen, 2014] Dieleman, S. and Schrauwen, B. (2014). End-to-end learning for music audio. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6964–6968. IEEE.

[Gatys et al., 2015] Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256.

[Hay et al., 1960] Hay, J. C., Martin, F., and Wightman, C. (1960). The mark-1 perceptron-design and performance. In *PROCEEDINGS OF THE INSTITUTE OF RADIO ENGINEERS*, volume 48, pages 398–399.

[Hebb, 1968] Hebb, D. (1968). 0.(1949) the organization of behavior.

[Hinton et al., 2006] Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

[HPC, 2016] HPC (2016). Pegasus Supercomputer. http://ccs.miami.edu/pegasus. [Online; accessed 11-March-2016].

[Humphrey and Bello, 2012] Humphrey, E. J. and Bello, J. P. (2012). Rethinking automatic chord recognition with convolutional neural networks. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 2, pages 357–362. IEEE.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Lee et al., 2009] Lee, H., Pham, P., Largman, Y., and Ng, A. Y. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104.

[Li et al., 2010] Li, T. L., Chan, A. B., and Chun, A. (2010). Automatic musical pattern feature extraction using convolutional neural network. In *Proc. Int. Conf. Data Mining and Applications*.

[Mahendran and Vedaldi, 2015] Mahendran, A. and Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 5188–5196. IEEE.

[Marc′Aurelio Ranzato et al., 2007] Marc′Aurelio Ranzato, C. P., Chopra, S., and LeCun, Y. (2007). Efficient learning of sparse representations with an energy-based model. In *Proceedings of NIPS*.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[Minsky and Papert, 1969] Minsky, M. and Papert, S. (1969). Perceptron: an introduction to computational geometry. *The MIT Press, Cambridge, expanded edition*, 19(88):2.

[NIH, 2012] NIH (2012). Brain Power: Grades 2-3. https://www.drugabuse.gov/publications/brain-power/grades-2-3/sending-receiving-messages-module-3/background. [Online; accessed 11-March-2016].

[Samarasinghe, 2006] Samarasinghe, S. (2006). *Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition*. CRC Press.

[Schlüter and Böck, 2013] Schlüter, J. and Böck, S. (2013). Musical onset detection with convolutional neural networks. In *6th International Workshop on Machine Learning and Music (MML), Prague, Czech Republic*.

[Simonds, 2016] Simonds, D. (2016). Showdown. The Economist, March 12.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition (2014). arxiv preprint. *arXiv preprint arXiv:1409.1556*.

[StudyBlue, 2016] StudyBlue (2016). Membrane Potentials, Action Potentials, Conduction of the Action Potential. https://classconnection.s3.amazonaws.com/858/flashcards/3704858/jpg/action_potential-141232DAB9B5F08F863.jpg. [Online; accessed 11-March-2016].

[Van den Oord et al., 2013] Van den Oord, A., Dieleman, S., and Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651.

[Wang and Wang, 2014] Wang, X. and Wang, Y. (2014). Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the ACM International Conference on Multimedia*, pages 627–636. ACM.

[Werbos, 1990] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

[WIDROW et al., 1960] WIDROW, B., HOFF, M. E., et al. (1960). Adaptive switching circuits.

[Wikipedia, 2016] Wikipedia (2016). Melodic death metal. https://en.wikipedia.org/wiki/Melodic_death_metal. [Online; accessed 11-March-2016].

[WildML, 2015] WildML (2015). Recurrent Neural Networks Tutorial, Part 1 ? Introduction to RNNs. http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/. [Online; accessed 11-March-2016].

Appendix