3-2017

Advanced Music Audio Feature Learning with Deep Networks

Madeleine Daigneau
mad1948@rit.edu

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation
Advanced Music Audio Feature Learning with Deep Networks

By

Madeleine Daigneau

A Thesis Submitted in Partial Fulfillment of the Requirements for Degree of Master Science in Computer Engineering

Department of Computer Engineering
Kate Gleason College of Engineering

Rochester Institute of Technology
Rochester, NY

March 2017

Committee Approval:

Date:
Dr. Raymond Ptucha
Primary Advisor – R.I.T. Dept. of Computer Engineering

Date:
Dr. Andreas Savakis
Secondary Advisor – R.I.T. Dept. of Computer Engineering

Date:
Dr. Sonia Lopez Alarcon
Secondary Advisor – R.I.T. Dept. of Computer Engineering
ABSTRACT

Music is a means of reflecting and expressing emotion. Personal preferences in music vary between individuals, influenced by situational and environmental factors. Inspired by attempts to develop alternative feature extraction methods for audio signals, this research analyzes the use of deep network structures for extracting features from musical audio data represented in the frequency domain. Image-based network models are designed to be robust and accurate learners of image features. As such, this research develops image-based ImageNet deep network models to learn feature data from music audio spectrograms. This research also explores the use of an audio source separation tool for preprocessing the musical audio before training the network models. The use of source separation allows the network model to learn features that highlight individual contributions to the audio track, and use those features to improve classification results.

The features extracted from the data are used to highlight characteristics of the audio tracks, which are then used to train classifiers that categorize the musical data for genre and auto-tag classifications. The results obtained from each model are contrasted with state-of-the-art methods of classification and tag prediction for musical tracks. Deeper networks with input source separation are shown to yield the best results.
ACKNOWLEDGEMENTS

I would first like to thank my thesis adviser, Dr. Raymond Ptucha, for his continued guidance throughout the development of this thesis. I would also like to thank the members of my thesis committee, Dr. Sonia Lopez-Alarcon and Dr. Andreas Savakis, for their support during my academic classes and for participating in my graduate research.

This research would not have been possible without the valuable resources of the Machine Intelligence Lab at Rochester Institute of Technology (RIT) and the support and feedback from my fellow classmates in the lab.

I would also like to thank Dr. Zhiyao Duan and his students from the Audio Information Research (AIR) Lab at the University of Rochester for their inspiring insight into the field of audio processing and analysis.
# Table of Contents

ABSTRACT ................................................................................................................................. ii  
ACKNOWLEDGEMENTS ........................................................................................................... iii  
INTRODUCTION .......................................................................................................................... 1  
  Music Information Retrieval ............................................................................................. 2  
  Deep Networks .................................................................................................................... 3  
  Audio/Music Information ...................................................................................................... 7  
BACKGROUND .......................................................................................................................... 14  
  Music and Deep Learning ................................................................................................... 14  
  Datasets ................................................................................................................................. 18  
APPROACH ................................................................................................................................. 22  
  Framework ............................................................................................................................. 22  
  Audio Pre-Processing .......................................................................................................... 23  
  Deep Learning Models ......................................................................................................... 26  
RESULTS ..................................................................................................................................... 32  
  Genre Classification ............................................................................................................. 32  
  Tag Prediction ....................................................................................................................... 47  
FUTURE WORK .......................................................................................................................... 56  
CONCLUSIONS ........................................................................................................................... 58  
BIBLIOGRAPHY .......................................................................................................................... 60
INTRODUCTION

The human brain, arguably the most complex entity in the universe, is directly responsible for our successes at developing and adapting throughout life. Our ability to learn based on our experiences affects the evolution of our society. Inspired by the human brain, deep neural networks strive to attain some of its incredible potential. In the past few years, deep learning has made it possible to train computers to recognize and understand image patterns, which in many cases surpass human capabilities. Since humans can comprehend visual and audio stimulus equally well, there is no reason why the same models that excel at image analysis cannot be trained to understand audio data. In particular, this thesis will investigate the performance of deep networks developed for image data when they are applied to musical audio data.

Music is an integral part of the entertainment market in the world. Every culture has developed their own variety of music. Music is distributed on its own, in album collections of tracks, and as an integral part of movies. People listen to music at different times of the day, during different activities, and for different reasons. Some listen to softer style of music to fall asleep, others listen music with a faster beat during exercise, while others simply listen for enjoyment during work and recreation. Everyone listens to music in his or her own way and everyone has his or her own musical preferences.

This research analyzes the application of deep network models for musical genre classification and tag prediction. Deep networks designed for image data are refactored and retrained towards understanding the features that exist in musical audio data. In addition, this research explores the use of audio source separation as a means of preprocessing before the training stage of the deep network model.
**Music Information Retrieval**

Music Information Retrieval (MIR) is the field of research that describes extracting, analyzing, and using information from music. MIR tasks include, but are not limited to, musical classifications, audio track separation, musical score following, automatic music transcription, and music recommendation systems.

Musical classifications are a central problem of MIR, which include predictions of a musical sample’s genre, artist, and mood information. The Music Information Retrieval Evaluation eXchange (MIREX) is an annual competition held as a part of the International Society for Music Information Retrieval (ISMIR) Conference that accepts entries for a variety of MIR tasks, including several categories of classifications. Genre classifications in MIREX for the most recent competition utilize the K-POP dataset. The K-POP dataset is comprised of 30-second audio clips taken from the middle of the original tracks. The task is to determine which of seven musical genres the track is associated with. The seven genres are ballad, dance, folk, hip-hop, R&B, rock, and trot. The K-POP dataset only contains 1894 samples in the dataset, too small for a defined training and testing dataset, so accuracy is evaluated using 3-fold cross validation. Mood classifications on the K-POP dataset generally look to cluster music samples into one of five clusters of mood categories. The first cluster includes music of passion and confidence, the second cluster is for cheerful or joyous feelings, the third contains songs of a wistful or brooding nature, the fourth contains whimsical or silly music, and the fifth contains songs that are intense and aggressive. Tag classification is conducted on a separate dataset called MajorMinor, which contains 2,300 ten-second audio clips collected from 1,400 different tracks of 500 unique artists. Of the 73,000 total unique tags in this dataset, 12,000 were used by at least two users, and only 43 were verified at least 35 times. Those 43 tags are the descriptive ground
truth used for the tag prediction task. Entries to this competition analyze the ten-second audio clips of musical tracks to produce basic description and mood tags with some ranking of the results. This competition provides its own dataset of audio tracks for training and testing classification models, provided only after registration in that year’s competition [1].

Music recommendation is another popular MIR task, because it can relate directly to the music industry as a means of promotion. Many companies that produce and distribute music use some form of music recommendation algorithms to encourage additional purchases. These algorithms largely utilize the user preferences on music, albums, and artists. Most recommender models use collaborative filtering, which recommends products based on information gathered on products previously enjoyed by the same customer. This information can be used to recommend products determined to be similar to previously liked products, or it can be used to determine which customers have similar tastes, and then recommend products based on what those similar customers have liked.

Most MIR tasks rely heavily on the quality of music feature data available, and many of the tasks are accomplished using predefined metadata of each musical track. This approach is limited to the data available to each track which often does not have any in-depth analysis of the music audio. Currently, there is a growing interest in the application of deep learning techniques to audio data towards improving predictive models for MIR tasks.

**Deep Networks**

The function of Artificial Neural Networks (ANNs) is inspired by the biological neurons in the human brain. The human brain is comprised of billions of neurons, each of which is connected up to 10,000 other neurons. These neurons receive, process, and transmit information
necessary for various biological functions, from basic muscle movements to complex organ operations. ANNs attempt to model something similar to this function without being constrained by the real-world interactions of the neurons in the brain. Inputs to the ANN, which includes a bias term, are applied through weighted connections to an activation function, which produces an output value from the ANN. Figure 1 shows a comparison between a biological neuron and an artificial neuron.

![Diagram of a biological neuron and an artificial neuron connection.](image)

**Figure 1.** (Left) Diagram of a biological neuron. (Right) Diagram of an artificial neuron connection. [2]

In a biological neuron, impulses are received via dendrites from across the synapse, or gap, between neuron cells. The impulses are carried by the axon connection to the axon terminals before being transmitted to the next neuron cell.

In the right diagram of Figure 1, X is the input to the neuron, or the ‘impulse’ received by the neuron. A weighted matrix, W, is applied to the input, X, before reaching the ‘cell body’ where the weighted input and bias are evaluated. An activation function, shown in the right of Figure X as \( f \), is used to constrain the value of the output, such as the sigmoid function \( f(x) = \frac{1}{1+e^{-x}} \) to constrain the output range to \([0, 1]\) or TANH \( \tanh(x) = 2\sigma(2x) - 1 \), a scaled sigmoid) to constrain the output to the range \([-1, 1]\). The basic structure of an ANN uses multiple layers made up of these artificial neurons. The initial layer is the input layer, where the input to
the network is applied. The middle layers are called hidden layers. A neural network can contain any number of hidden layers with any number of artificial neurons per layer. The final layer of the network is the output layer, where the final output of the network is produced [2].

Yann LeCun’s seminal paper [3] introduced the convolutional neural network (CNN), a type of ANN, in 1998. The CNN learns features in the form of kernel windows that convolve with an input image to produce feature maps. Figure 3 shows an example of some of these kernel window features. These feature maps are eventually passed into fully connected layers, and finally a softmax classification layer. Using supervised learning, CNN models learn from many exemplar images, each image having an associated ground truth label. Figure 2 contains the network from LeCun’s paper with a breakdown of his network structure and how an input image is processed through it.

![Image](image-url)

**Figure 2.** Convolutional Neural Network model from LeCun [3].

The network in Figure 2 contains six hidden layers in addition to an input layer and an output layer. The input layer is a 32×32 pixel greyscale image. The first hidden layer is a convolutional layer, which learns six 5×5×1 convolution windows to generate six feature maps of size 28×28. The second hidden layer is a pooling layer, which reduces the scale of the image, with a 2×2 pooling window and a stride of two, to 14×14 feature maps. The third hidden layer
learns sixteen 5×5×6 convolution windows to generate sixteen 10×10 feature maps. The fourth hidden layer is a second pooling layer with a 2×2 pooling window and a stride of two, reducing the feature maps to 5×5. The fifth hidden layer is a fully connected layer, which takes all the information from the feature maps and generates a 120×1 feature vector. The sixth hidden layer is a fully connected layer, which generates an 84×1 feature vector. The final layer uses softmax to generate a 10×1 output vector for classification. [3]

CNN models performed well, but their performance was limited because the computer systems at the time were not capable of processing large amounts of data. Neither the compute power, nor the memory required to store the millions of network weights were feasible. Thus, CNNs were set aside for more computationally-friendly models until advancements were made in computer systems and making CNNs less expensive to implement.

With the release of AlexNet [4], CNNs were reintroduced for more practical use in image recognition with the development of faster processors with more memory capabilities. AlexNet popularized the use of Rectified Linear Units (ReLUs, $ReLU(x) = \max(0, x)$) as a simpler means of activation in the layers. In addition, the usage of dropout was demonstrated to be a powerful method for parameter regularization, which offered numerous benefits over the traditional L1 and L2 regularization techniques.
As CNNs became a more popular tool for training feature detection for multiple problem domains, researchers designed networks for faster training and even more accurate classifications. Over the last few years, CNNs were advanced even further, with researchers releasing deeper and more complex networks such as GoogLeNet[5], VGGNet[6], and ResNet[7]. These models were able to advance deep learning to the point that today; computers have surpassed human level image recognition.

**Audio/Music Information**

In the field of audio analysis, low level features that define audio data fall under the categories of timbre and temporal features. Timbre is associated with the frequency domain and defines features such as existing frequencies in an audio track, as well as identifying prevalent and harmonic frequencies. Temporal features are defined over the time domain.

Many methods of extracting features from audio signals have been researched for audio content analysis. Mel-frequency cepstral coefficients (MFCCs) are particularly popular for extracting the power spectrum of an audio signal. The basic process is to take the spectrogram of the audio signal, convert it to a Mel scale \( (Mel\ scale = 2595 \times \log_{10} \left( 1 + \frac{frequency}{700} \right)) \) [8], take
the log of the powers at each Mel-frequency, and apply the discrete cosine transform to generate the Mel-frequency cepstrum (MFC). The MFCCs are the amplitudes of the resulting cepstrum. This process has many parameters that may be adjusted for results customized to a specific function. MFCCs are most often used as features in speech recognition and analysis. Another, more simple, timbre feature extraction method is the Zero-Crossing Rate, or the rate at which a signal changes from a positive to a negative value and back.

These expertly defined methods of extracting audio information have had many recorded successes, and have improved performance of audio analysis across various problem domains. However, these methods were not designed for musical data. They were designed for speech audio, which is only a single contribution to a musical track (more in choral/a Capella pieces, and less in instrumental pieces). Despite being designed for speech audio, research conducted with these methods of feature extraction has proved that they do extract some meaningful information from music audio. In fact, many papers have been published using these methods to perform musical analysis and classification tasks [9-12].

When analyzing musical audio, the raw audio signal is most often transformed into its spectrogram representation prior to its analysis. A spectrogram is generated by analyzing the existing frequency components of an audio signal over a given frame of time of the signal. These frequencies are then analyzed for their individual magnitudes which are then translated to values on a two dimensional matrix. The dimensions of the spectrogram are the time domain and the frequency domain, and the values of the spectrogram are the magnitudes of the associated frequency at that time. Figure 4 contains a spectrogram representation of a short audio track generated by using the short-time Fourier transform (STFT).
Figure 4. Linear spectrogram representation of a 30-second stereo music audio clip. The two separate spectrograms represent each of the two channels of the audio signal. The two channels give a stereo effect to the audio signal though the differences between each signal are minute in magnitude. The spectrograms used in this research are converted to a single channel input signal.

Transforming an audio signal into a spectrogram increases the amount of information through the addition of the frequency domain to the data. In addition to the transform, the magnitude values of the spectrogram are converted to a logarithmic scale, often converting the data into decibels or into a Mel scale. The original linear scale places emphasis on the harmonic relationships of the spectrogram while the logarithmic scaling emphasizes the tonal relationships that are key to musical pieces. In this research, the spectrogram magnitudes are scaled to decibels: \[ \text{Decibel Scale} = 20 \times \log(\text{Magnitude}) \].

In music, there exists the concept of harmonic frequencies, which make each note more pleasant to hear than that of pure tones. A pure tone in music is a single frequency, easily visualized as a smooth sinusoidal waveform of the raw audio signal or a single spike on the
spectrogram representation of the signal [13]. Figure 5 compares a pure tone middle ‘C’ note to the same note played on other instruments, such as a piano keyboard or a guitar. The spectrogram function used to generate these figures in MATLAB uses windows to separate the signal into segments. The vertical lines on the spectrograms are noise caused by the type of windows used to generate the spectrograms. These spectrograms were generated using a Hamming window function, which are not zero-ended, causing the vertical line noise.

All three spectrograms display the strongest magnitude at the same frequency, which is the frequency of the middle ‘C’ note. However, the real world instruments also show additional frequencies present in their spectrograms, which are the harmonic frequencies. The waveforms of the real world instrument’s audio signals also show some distortion to their waveforms, though they maintain the same fundamental period as the waveform of the pure middle ‘C’ note (the pure sinusoid). This is because the frequencies that are distorting the waveforms are harmonically related to the fundamental frequency of the middle ‘C’ note. In other words, the harmonic frequency, $f_k$, is related to the fundamental frequency, $f_0$, by the equation: $f_k = k \times f_0$, for some integer $k$. By this same equation, the fundamental frequency, or the note played, can be determined visually by the spectrogram seeing which of the lines appears at the lowest frequency and with the greatest magnitude [14].

These harmonic frequencies determine how the same note sounds when played from different instruments, and they are just one of many features associated with audio analysis.
In regards to the field of music, terms such as pitch or notes define the lower level features. Genre and subgenre, mood, rhythm, melody, and other more complex arrangements of music describe the higher level features.
In deep learning applications, the traditional medium of feature analysis is image data. Image data is represented by a matrix of pixels, which can be of various heights and widths. The image data also has a depth of either a single value (for a greyscale image) or a depth of three values for each pixel (one for each color channel of red, green, and blue, for a color-scale image). In order to train deep learning models for audio data, the simplest strategy is to introduce the audio spectrograms as greyscale images.

Spectrograms and image data are interchangeable in their original forms. However, with deep networks, there are differences between the two that must be addressed prior to the training phase. First, images have no set order to their dimensions, thus, they can be stretched, scaled, mirrored, cropped, and have many other types of transforms applied with little to no negative impact on their provided information. This is an advantage for data augmentation, which can apply various means of transforming images to gain multiple sources of information from even a single data sample. However, this does not apply to audio spectrograms, whose dimensions show ordered information of frequency and time.

Considering the frequency domain, the information provided at different locations indicates completely different pitches, and therefore any change to the scale of this dimension would significantly change the information the spectrogram contained. On the time domain, only a single direction that indicates the flow of time, and changing the direction or the scale of the time domain again changes the information provided by the spectrogram. However the settings of the beginning and end of the time domain do not affect the relative information of the spectrogram, so cropping the spectrogram along the axis of time is an acceptable means of data augmentation.
One major concern from using musical audio features is the risk of overfitting the features to the training data. This problem occurs due to the lack of a large enough dataset for the potentially 100+ million parameters in neural networks to learn effective features. This problem is propagated by the fact that the various means of data augmentation popular for image data is not feasible for audio data, especially in the case of music. For most trained models using these smaller audio datasets, researchers resolve to use other means of avoiding overfitting such as regularization parameters and dropout layers, or utilize $k$-fold cross validation during model training.
BACKGROUND

Music and Deep Learning

Since their introduction, deep network models have offered advanced feature learning and detection across various problem domains for several different types of data, including audio data. In the field of music, content-based musical analysis is a growing interest in audio research.

In musical audio analysis, a popular way to accomplish tasks is to extract features from musical data, using MFCCs or the like, and then have machine learning applications use those features as inputs to a system. Another popular method is to use machine learning applications to determine the best combination of feature extractions to accomplish a goal [9-11]. Utilizing deep neural networks combines the stages of feature learning and extraction with the classifier for the original task.

Current popular music recommendation systems implement a collaborative filtering approach for their recommendations. Such consumption-based methods recommend music based on what music the user listens to and who else has listened to the same music. This data mining approach assumes that music popular with one individual will be popular with another consumer who has listened to and enjoyed similar music. This solution performs well for users and musical tracks with lots of user metadata associated with them. However, collaborative filtering techniques do not account for new users and for new or unpopular music releases or artists, as these have little to no user metadata attributed to them. This conflict defines the cold start problem. Collaborative filtering recommendations are limited to the most-listened to musical tracks and the most active users.
Content-based methods recommend songs with similar content features extracted from the audio tracks, and therefore the cold start problem does not apply to them. Audio content has been fed into deep learning frameworks in an attempt to produce a better music recommendation system by learning important content features [15, 16].

One such network used a bottleneck architecture to better detect features associated with musical chords within an audio track. The bottleneck architecture design uses a larger number of feature weights in the first hidden layers of the network, and a decreased amount in the middle layers of the network, and then another increase in the number again in the last hidden convolutional layers. This architecture is implemented mainly to reduce the likelihood of overfitting the features extracted to the training data [17].

![Bottleneck Architecture](image)

**Figure 6.** Example Bottleneck Architecture in the hidden layers of a neural network.

Sander Dieleman explored another angle to traditional audio processing and designed his network to extract features from time domain audio signals, as opposed to extracting features from their spectrograms [15]. While his network did not attain the performance attained by networks trained on the frequency domain, he did prove their potential in classifications of musical data.
Since their introduction, Convolutional Deep Belief Networks (CDBNs) have been applied to data from the image, audio, and graphical data domains. CDBNs are neural networks designed for unsupervised feature extraction using layers of restricted Boltzmann machines (RBMs). Unsupervised in machine learning indicates that there are no ground truth values associated with training data, and the network learns features from the data using internal error functions, similar to Principle Component Analysis (PCA) or $k$-means clustering. RBMs are bipartite, undirected graphical models with a single input and a single hidden layer with a weight matrix connection between them. During training, the network performs Gibbs sampling on the input data model before forward propagating the input through the weight matrix, sampling the output, and back propagating the result through the weight matrix again. This forward and back propagation occurs a finite number of iterations before the result is compared to the original input, the difference between which the weight matrix is updated in the model. Multiple layers of RBMs can be trained in succession. Each RBM layer in the CDBN model trains independently of each other, until the output of the last layer is run through some form of classifier, which can be another fully connected neural network or something simple such as a smart vector machine (SVM). In 2009, CDBN models were introduced for use with extracting features from images in the MNIST handwritten digit dataset and select categories of the CalTech101 dataset [18] and have also been trained for facial recognition [19].

In the field of audio analysis, CDBN models have proved useful for unsupervised feature extraction from spectrograms of audio signals. Modified CDBN model results were published for training with audio data for speech detection, audio recognition, and musical classification [20, 21]. Another research team proposed a modified CDBN model designed for learning harmonies and percussive features of audio signals [22].
In deep models designed for image analysis, the kernels in the convolutional and pooling layers traditionally have square dimensions for robust feature learning. This is generally due to the subject having the potential for a limitless number positions in the image. In deep models designed for audio analysis, the kernel windows are traditionally given rectangular dimensions to learn features. The ordered nature of the spectrogram dimensions fixes the potential for significant features to exist along the time axis or the frequency axis. For frequency-based features, the convolutional kernels are shaped to span the entire frequency axis of the spectrogram. These rectangular kernels encourage the learning of features between the frequencies at a point in time. Similarly, pooling operations take place on the dimension of time. For time-based features, the convolutional kernels are shaped to cover some extent of the time axis in order to learn features related to rhythm, or patterns of frequencies appearing over the course of time [20, 23].

One side-benefit of using rectangular kernel windows is the reduction of memory requirements at the later stages of the network model. Rectangular kernel windows are naturally larger than square kernel windows, so they require more memory to store and train. However, because of their size, they further reduce the size of the output to the layer, which reduces the memory required to store the output of that layer and all successive layers of the network. For networks that use rectangular convolutional kernels that span the entire frequency axis, the output of the layer has a frequency dimension of one. As such, all successive kernel windows need only cover that single dimension, which naturally reduces their size and therefore memory requirements for storage and training.
Datasets

The available audio datasets for research are not as well developed as the datasets that exist for image or video related research projects, especially audio datasets dedicated to musical analysis. The largest known of the publicly available music datasets, the Million Song Dataset [24], is composed entirely of manually defined metadata and do not contain the raw audio tracks, which restricts deep content analysis for music. Unfortunately, the memory requirements to load the audio files of a dataset and the copyright restrictions have limited the size and availability of music audio datasets for music content analysis. This is particularly unfortunate in regards to deep learning applications, which yield higher levels of performance when they are trained with large amounts of data, and tend to overfit their models to training data when the datasets are too small.

One effect of the restricted size of the datasets available is the lack of any predefined training, validation or test partitions to the data. The absence of these pre-separated sets makes direct performance comparison from one model to the next unreliable. When different models train their feature weights from different datasets, even in the case of different sections of the same dataset, an increase in predictive success on one model compared to another model does not indicate that the prior is the better model. The only accurate comparisons are made by using the same sets of data as another to train the model, or by training that other model on the sets of data defined for the new model.

The GTZAN dataset is a collection of 1000 total musical audio tracks, uniformly distributed over ten genre classes, published by George Tzanetakis in 2002. The genres included are blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock. Each audio file is thirty seconds long, sampled from the middle of the original music tracks [9].
Due to the restricted size of the GTZAN dataset, most research that utilizes this dataset evaluates their models using \( k \)-fold cross validation. In its introductory research, George Tzanetakis achieved an accuracy of 61% with a 4% standard deviation over 10-fold cross validation using Gaussian Mixture Models. This was utilizing the tool that was the main contribution of his research; the MARSYAS (Music Analysis, Retrieval and Synthesis for Audio Signals) feature extraction toolbox. He proposed a specific set of feature extraction functions, including MFCC’s, in a publically available toolbox for defining the important information in a musical track [9].

Another paper proposed by Li et al. researched how specific sets of expertly-defined features for musical content analysis, using different combinations in conjunction with Daubechies wavelet coefficient histograms (DWCHs), would perform on musical content analysis. They compared each combination of features using several different models, including Gaussian Mixture Models and K-Nearest Neighbors, for evaluating each collaboration. This research uses the MARSYAS toolbox for a particular set of its feature extraction functions, again including MFCC’s, filtering the audio track down to a descriptive histogram for classification tasks. They achieved their highest accuracy on the GTZAN dataset using Smart Vector Machines (SVMs) to get 78.5% accuracy with a 4.07% standard deviation over 10-fold cross validation [10].

Lidy et al. proposed psycho-acoustic analysis for musical content analysis. Their approach used two novel feature representations of Statistical Spectrum Descriptors and Rhythm Histogram features in addition to previously implemented rhythm feature detection. They reported their feature detection performance using SVMs on the GTZAN dataset and achieved 74.9% accuracy [11].
Lee et al. [25] proposed using modulation spectral analysis of the spectral and cepstral features of music audio. They used MFCCs, Octave-Based Spectral Contrast (OSC), and Normalized Audio Spectral Envelope (NASE) for their feature extraction stage. They then applied long-term modulation spectral analysis to extract information on the variances of the musical track over time from these short-term feature extractions. Using these features with a Linear Discriminant Analysis (LDA) classifier, Lee et al. achieved 90.6% accuracy with 3.06% standard deviation on the GTZAN dataset over ten-fold cross validation.

Wülfing et al. [26] explored the use of k-means clustering towards an unsupervised learning strategy for music genre classification. Their approach used convolutional 16×16 windows for k-means clustering with bootstrapping tested on spectrograms generated using the Constant Q Transform. Their best model, using a linear SVM classifier, achieved 85.25% accuracy with 3.5% standard deviation over ten-fold cross validation on the GTZAN dataset.

Behûn [27] proposed an approach similar to image feature detection for music genre classification. His method extracted Scale-Invariant Feature Transform (SIFT) features from the music spectrograms. These 32×32 SIFT features formed a Bag-of-Words feature vector that, when combined with a linear SVM classifier, achieved 86.4% accuracy on the GTZAN dataset over ten-fold cross validation.

Law et al. introduced the MagnaTagATune dataset, which contains extensive metadata for more than 25,000 musical data samples. The samples in this dataset include 29 seconds of the actual audio tracks of the songs, with some nearly complete tracks divided into 29-second partitions. The metadata includes a ground truth tag description that is some combination of 188 possible tags given by over 1,000 unique users. The dataset continues to grow as users sample and tag tracks from the Magnatune label via the TagATune game [28]. The metadata included
with the audio tracks defines each tracks’ associated song title, artist, album, the times of the original track it was sampled from, and the original source of the song on TagATune. In addition to the tag descriptions and the audio clip information, the dataset also provides features extracted using the Echo Nest API 1.0 [29]. The extracted features include the track’s tempo, time signature, pitch, timbre, and more with a confidence value assigned to each attribute being extracted.

The metadata of the MagnaTagATune dataset also includes inverted similarity data from a bonus game on TagATune. The game allows the user to place a single vote for the most dissimilar track out of three options. The record of votes is included in the MagnaTagATune dataset as the inverse similarity data [28]. This inverse similarity data has been processed and the constraints data is published for the comparison of musical tracks through the Music Informatics Research Group at the City University of London, which currently hosts the dataset [30].

Unfortunately, the manner in which tags were collected on the TagATune game, with users manually typing each descriptive term, allowed the data gathered subject to spelling errors. These tags were not cleaned prior to being published. One such spelling error included is the separate tags ‘classical’ and ‘clasical’. The tags are also not uniformly distributed. There are some tags associated with thousands of tracks and others that only appear a few dozen times. There is unpublished research on how to combine potentially similar tags and misspelled tags, though most published results using MagnaTagATune use only the most popular tags for training and evaluating their tag prediction models to avoid issues related to training for the less often occurring tags [12, 15, 31-33].
**APPROACH**

This research analyzes and repurposes deep neural network models, originally designed for image feature analysis, to learn key features of musical audio data. Experimentation includes the network models in their original unaltered forms, and additionally a modified version using traditional approaches to deep networks for audio data. The extracted features are identifying markers for more effective classifications of musical data into categories defined by genre and tag prediction.

The network models will learn features from the preprocessed analysis of the frequency features from the spectrograms of musical tracks. The experiment data includes the spectrograms from the original audio signals and the audio signals preprocessed using a source separation toolbox. Figure 7 shows the basic structure of the process for the research.

![Figure 7](image)

**Figure 7.** The overall system architecture of this research. The main contributions are the proposed methods of Data Pre-Processing and analyzing the results for different models in the Deep Network stages.

**Framework**

In order to train the deep network models on audio data, a framework capable of robust calculations, utilization of a general processing unit (GPU) for faster calculations, and flexible network parameters was required. The convolutional architecture for fast feature embedding (Caffe) provides a robust framework designed for ease of developing deep neural network models for training on unique data from various domains [34].

22
In addition to the robust framework, the Caffe installation included tools for loading a customized dataset and evaluating any trained model on a given input image.

Caffe not only provides the appropriate tools required for training and testing neural network models, it also provides the structures of several published network models via the Caffe Model Zoo. These network models are available for users to download and adjust to train for official datasets or using their own data.

**Audio Pre-Processing**

Neural networks produce better results when large amounts collections of data samples are used in the training stages. In contrast to the models designed for image data, which included cropping as a parameter in the data layer of the network models, all preprocessing of the audio tracks was conducted manually and separately from training the model. The general approach to the full audio preprocessing is shown in Figure 8. Some stages are omitted for given datasets or training phases for each model.

![Figure 8](image)

*Figure 8.* Overview of data pre-processing architecture for source separated audio spectrograms. If testing the model on the spectrograms of the original audio track, the first stage of preprocessing, ‘FASST’, is omitted. If preprocessing the MagnaTagATune dataset, the second stage, ‘Crop Audio’, is omitted.

The musical tracks from the GTZAN dataset were sampled at a rate of 22kHz, and the spectrograms were generated using a Hamming window size of 1024, or about 46ms, and a stride of 512, or about 23ms. Prior to training on the Caffe models, the spectrogram images were resized to 256×256 pixels so that the image-based deep networks would be able to process the images as they were intended.
In order to increase the amount of data in the GTZAN dataset (1000 samples), simple audio cropping was implemented for data augmentation. The cropping stage took the raw audio data and cropped overlapping 10-second frames of audio from the file before generating the audio spectrograms. The dataset was then separated into training, validation, and testing sets. The training partition contained 60% of the data samples, the validation partition contained 20% of the data samples, and the testing set contained the final 20% partition of the data samples.

Prior to training with the dataset, cleaning the 188 unique tags in the MagnaTagATune dataset was a priority. Rather than rely on an unpublished research project for grouping similar tags, code was written for grouping tags that had been analyzed to have similar meaning or intended to have the same meaning. One example of tags intended to have the same meaning are the ‘classical’ and ‘clasical’ tags. The second spelling was incorrect, however both meant the classical genre could be an applicable description to that specific track. This pre-processing stage reduced the total number of possible tags from the original 188 tags to 132 unique tags.

In addition, no cropping augmentation was implemented for these audio tracks. Due to the nature of the tags associated with each track, there was no guarantee that the same tags could be applied to every section of the track sample. For example, one clip might have a person singing at the start of the clip, but stop before the end of the clip, or vice versa. Therefore, a tag associated with a person singing (‘vocals’) would apply to only the section of the clip where the person was singing, and cropping data augmentation would not account for that change in the ground truth.

The musical tracks from the MagnaTagATune dataset were sampled at a rate of 16kHz, and the spectrograms were generated using a Hamming window size of 1024, or about 46ms, and a stride of 512, or about 23ms. Prior to training on the Caffe models, the spectrogram images
were resized to 256×256 pixels so that the image-based deep networks would be able to process the images as they were intended. The dataset was then partitioned into training, validation, and testing sets. The training partition contained 60% of the data samples, the validation partition contained 20% of the data samples, and the testing set contained the final 20% partition of the data samples.

One strategy for audio pre-processing utilizes existing source separation tools before training audio features on individual ‘voices’ from the track. The Flexible Audio Source Separation Toolbox (FASST) [35] is a source isolation resource used to separate individual ‘voices’ or contributions to a single audio track. For the input data, FASST separated out four sources from the original audio tracks. The four sources include the track’s main melody, the bass notes, the drums, and all other sounds. While the term ‘melody’ generally describes a vocal part, in this toolbox, the melody is the main tonal and rhythm patterns in the musical track. Then, the short-time Fourier transform (STFT) converted the original and source separated audio data into their spectrogram representations.

![Figure 9](image)

Figure 9. (Left) Spectrogram representation of 10 seconds of an example audio track. (Right) Four spectrograms, each generated from the resulting audio tracks from using FASST to separate four sources of melody, bass, drums, and all other sounds from the same example audio track as on the left.
Figure 10. Conversion of source separated spectrograms into an RGB image. Each channel of the depth of the RGB image is a different source separated spectrogram. Here, red is the ‘melody’ spectrogram, green is the ‘bass’ spectrogram, blue is the ‘drums’ spectrogram, and where the spectrograms overlap, is cyan, magenta, white, etc. The final channel (‘others’) is not used in this form (due to the three channel limit on an RGB image), but can be included as a fourth channel.

**Deep Learning Models**

In this research, the models tested are benchmark models on the ImageNet dataset and models specifically designed to improve memory requirements and performance without sacrificing accuracy.

The introduction of AlexNet [4] revolutionized image processing with deep networks. The structure and parameters of this model are included with the initial Caffe download.

The SqueezeNet [36] model and the Network-In-Network [37] ImageNet model (hereafter referred to as NIN-ImageNet) were developed as alternative networks to produce AlexNet-level results with fewer parameters and memory requirements. Both of these model structures were downloaded from Caffe Model Zoo. In each of these publications, the authors focused on improving the model memory requirements instead of improving the accuracy on the dataset.

Iandola et al. proposed the Fire module for the SqueezeNet model, which implemented two sublayers of ‘squeeze’ and ‘expand’, in order to achieve three network goals. First, use 1×1
convolution filters instead of 3×3 convolution filters to reduce learnable parameters. Second, reduce memory requirements further by reducing the number of channels, or input data depth, on the input of the 3×3 filters. Third, delay downsampling the data until the later layers of the network to allow the convolutional layers have large activation maps, which theoretically increases the network accuracy. Downsampling in this network involved the use of increased strides for the filters in the final convolutional layers. Comprised of 1×1 convolution filters, the ‘squeeze’ layer design contributes to the first and second goals, achieving the second as it contains fewer filters than the number of filters in the ‘expand’ layer, which used a combination of 1×1 and 3×3 layers [36].

Figure 11. SqueezeNet Fire Module [36].

The Network-In-Network (NIN) model proposed by Lin et al. [37] stacked micro-networks, multi-perceptron (MLP) layers, as the layers of the network. The final layers of this network rely on dropout regularization and the final output are confidence values from a final MLP layer using global average pooling [37].
The GoogLeNet [5] network structure was also provided with the Caffe package download. GoogLeNet implemented more complex network layers than AlexNet, using the Inception Layer developed to increase the depth of the network without sacrificing its accuracy or overfitting the training data, achieving a total network depth of 22 layers. Rather than follow classic CNN architectures, which used convolutional layers often followed by activation and pooling layers, Szegedy et al. proposed the Inception layer, inspired by the Network-In-Network model before.

All four of these network models were designed for the ImageNet dataset, so their convolution and pooling layers use square kernel windows. For audio processing, these networks’ convolution and pooling layers were modified for rectangular kernel windows. In the modifications, the first convolutional layer uses the kernel height to span the entire height of the input data, effectively covering the frequency axis of the input data. The output of that layer has a height of one, so the remaining layers use a kernel window with a height of one as well. This results in data with a dimension of width and depth, but no height, so the remaining data layers are all two-dimensional.
In his research, Sander Dieleman proposed a deep network model specifically designed for extracting audio features from spectrograms of musical tracks. His network was developed through his work with Spotify, towards improving music recommendation. This model will hereafter be referred to as SDNet.

Unlike the other network models discussed here, Dieleman specifically designed SDNet for audio data, so the convolution and pooling layers are already set for rectangular kernel windows. He implemented a global pooling layer before his fully connected layers, which pooled the learned features over the remaining values in the time domain. This method argued that where in a track the features appeared was not as significant as the fact that they existed, and that pooling along the time axis would allow for more robust results [38]. The model was edited
slightly at the final fully connected layer to fit the tasks of this research, and the global temporal pooling layer here uses only the maximum and average pooling.

Figure 15. Sander Dieleman’s original network model (SDNet) for extracting musical features from audio spectrograms [23]. Prior to the global temporal pooling layer, the time axis is on the vertical axis. The network ends at a fully connected layer that outputs a 40-value feature vector, which is the feature dimension used by Spotify for their music recommendation network. Each layer (rectangle) shows the output of the previous layer, while the red rectangles indicate the convolutional kernel windows. The number at the bottom of these rectangular layers is the height of the layer input, and the number in red next to the red kernel rectangle is the size of the convolutional kernel window. After the first layer (spectrogram), the numbers above the layers indicate the layer depth, or the number of feature windows trained in the previous layer. For the first layer, the width of the kernel window spans the frequency domain (the width) of the spectrogram image, and for all successive layers, the kernel width is one. For simplicity, each layer after the first and before the global pooling layer shows the output’s height by its depth. In this figure, MP indicates Max Pooling. After the global temporal pooling layer are the fully connected layers.

All deep network models were trained from scratch using the spectrogram images derived from music datasets. In addition, all models were modified in order to disable any internal data augmentations, such as regional cropping, during training. For example, AlexNet was designed to crop the upper left, upper right, lower left, lower right, and center regions of the training images, and randomly mirror the images as well.
In order to implement the previous models for tag prediction, the final layers of the network for accuracy and loss need to be modified for the unique structure of the ground truth. Tag prediction requires adjusting the final output of the fully connected layers to 264, which accounts for each possible value (logical 1 being present, logical 0 being absent) of each of possible tags for each clip (132). This output is then reshaped to fit the required input shape to the accuracy and loss layers.

The scripts provided by Caffe for loading images into a custom dataset did not allow for multiple labels for each track. As such, customized code was written for the MagnaTagATune dataset. Ordinarily, one data batch contained the training data and ground truth and other contained the validation data and ground truth. In the case of the MagnaTagATune data, however, four data batches were loaded. The first contained the images into an ordered training data partition and the ground truth was loaded into a separate data batch that maintained the order identical to the image data batch. Then the same was done for the validation data partition. Each dataset was loaded using a separate input layer in the Caffe model, which loaded the images as ‘data’ and the ground truth tags as ‘label’, the same way a traditional data layer in a Caffe model would.
RESULTS

All experiments were conducted on GPUs large enough to process the data layer’s indicated batch sizes of the Caffe models, containing the output of the batch at each layer in addition to the memory required for each of the learned feature windows of the network model. In some cases, the original value of the data layer’s batch size was too large for the GPU to process without raising memory capacity errors. When these errors occurred, the value of the data layer’s batch size was lowered to an appropriate value such that the memory required to train or test the network model did not exceed the available memory of the GPU.

Genre Classification

Each model was trained and evaluated over four different model specifications. The first training phase had each model using their original structures, as applied to the original audio track’s generated spectrograms as the input images. The second training phase had each model use a modified network structure for rectangular kernel windows in their convolution and pooling layers, again applied to the original audio track’s generated spectrograms as the input images. The third training phase had the models use their original structures with the source separated audio spectrograms as the input data. Finally, the fourth training phase had the models use the modified network structure for rectangular kernel windows, again, with the source separated audio spectrograms as the input data. Table 1 summarizes the results for training on the GTZAN dataset and includes each models’ original performance on the ImageNet dataset as reported in their respective documentation.
Table 1. Validation accuracy. ImageNet Accuracy is the accuracy each model reports on the ImageNet dataset from their original conference papers and from the documentation on Model Zoo, if applicable. GTZAN accuracy reported from four separate conditions of the model and input data. ORIG indicates the input data to the model was the greyscale image of the spectrogram generated from the raw audio. SS indicates the input to the model was the concatenated spectrograms generated for each of the sources separated from the original audio file. SQ indicates the model maintained its original square kernel windows in the convolution and pooling layers. REC indicates the model parameters in the convolution and pooling layers were adjusted to have rectangular kernel windows, which spanned the entire height of the spectrogram (the frequency domain) in each layer. The final column Model AVG is the mean value of the validation accuracy for each model given the four (or two for SDNet) methods of input data (ORIG or SS) and model feature shape (SQ or REC) versions. The final row, Method AVG, is the mean value of the validation accuracy for each method over the five (or four, considering SDNet) models’ performance.

<table>
<thead>
<tr>
<th>Model</th>
<th>ImageNet Validation Accuracy</th>
<th>GTZAN Validation Accuracy</th>
<th>Model AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Top 1</td>
<td>Top 5</td>
<td>ORIG  + SQ</td>
</tr>
<tr>
<td>AlexNet[4]</td>
<td>57.1%</td>
<td>80.2%</td>
<td>66.2%</td>
</tr>
<tr>
<td>SqueezeNet[36]</td>
<td>57.5%</td>
<td>80.3%</td>
<td>64.8%</td>
</tr>
<tr>
<td>NIN-ImageNet[37]</td>
<td>59.36%</td>
<td>-</td>
<td>67.4%</td>
</tr>
<tr>
<td>GoogLeNet[5]</td>
<td>68.7%</td>
<td>88.9%</td>
<td>72.8%</td>
</tr>
<tr>
<td>SDNet[38]</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Method AVG</td>
<td></td>
<td>67.8%</td>
<td>61.2%</td>
</tr>
</tbody>
</table>

Even though the SDNet was designed for the analysis of musical audio spectrograms, the models designed for image data performed rather well even without any modification. The performance of GoogLeNet even managed to surpass that of SDNet by more than 3% in validation accuracy.

As indicated by the results of Table 1, the rectangular windows did not always result in improved performance for genre classification. Figure 16 shows the accuracy during the training phase for each of the network models as a plot of training iteration versus validation accuracy. For AlexNet and GoogLeNet, using the rectangular kernel windows decreases the accuracy while it improves the accuracy of NIN-ImageNet. In SqueezeNet, using the rectangular windows with the original spectrograms does not alter the accuracy by much, but it still significantly impacts the accuracy for the source separated spectrograms. SqueezeNet’s Fire module did not
seem to have any improvement on the audio data spectrograms, despite its slight improvement over AlexNet on the ImageNet dataset.

(a) Training curve for AlexNet
Figure 16. Plots showing the validation accuracy during each training setting of each network. ORIG indicates the input data to the model was the greyscale image of the spectrogram generated from the raw audio. SS indicates the
input to the model was the concatenated source separated spectrograms. SQ indicates the model maintained its original square kernel windows in the convolution and pooling layers. REC indicates the model parameters in the convolution and pooling layers were adjusted to have rectangular kernel windows, which spanned the entire height of the spectrogram (the frequency domain) in each layer.

Utilizing source separation as a pre-processing stage yielded improvements in overall classification for all models, as shown in Figure 16. Source separation adds details and information to the original spectrograms by indicating which frequencies are attributed to certain ‘voices’ of the track. In this case, the only information provided was the three sources of ‘melody’, ‘bass’ and ‘drums’. Increasing the depth of the spectrograms with additional ‘voice’ extractions may improve results even further.

The normalized confusion matrices shown in Figure 17 correspond to the best validation results from the models in Figure 16. The ‘metal’ genre was most successful classification in all of the models presented here. Predicting the ‘classical’ genre also presented successes across the networks. The most unsuccessful genre varied between the networks, though several had difficulty with the ‘rock’ genre. NIN-ImageNet, and SDNet, often mislabeled as the ‘rock’ genre as ‘metal’ or ‘country’. This confusion is understandable, considering the two genres do have many similarities in terms of instruments, tempo, etc. Another difficult classification was the ‘country’ genre, sometimes misclassified as ‘rock’. AlexNet and SqueezeNet misclassified ‘country’ as ‘reggae’ as often or more often than it was mistakenly labeled ‘rock’. The ‘blues’ genre was also particularly difficult for NIN-ImageNet and SDNet. They often misclassified the ‘blues’ genre as ‘country’ or ‘classical’.

37
(a) AlexNet Validation Confusion Matrix

(b) SqueezeNet Validation Confusion Matrix
Figure 17. Confusion Matrices of the best validation results from the AlexNet (a), SqueezeNet (b), NIN-ImageNet (c), GoogLeNet (d), and SDNet (e) models.

Figure 18 shows a select few of the 256 filters trained for SDNet’s first convolutional layer. These filters span the whole frequency domain of the input spectrograms and therefore can be interpreted as very short term spectrograms. Higher values are represented in white, neutral and low values are represented by greys, and black represents negative values, so white and black lines in the kernel windows indicate the edges of the frequencies that activate that kernel filter.

The kernel window shown in Figure 18(a) detects a harmonized increase in pitch in the music, and (d) detects a lowering of the pitch. The kernel in (e) has several pitches detected that indicates strong harmonics or a chord, which is a combination of pitches or notes played simultaneously. The similarity between the two mean this kernel may be activated by either.
kernel in (b) detects a single lower note in the musical track’s melody, and the kernel in (c)
detects lower pitched drums.

![Figure 18](image)

**Figure 18.** Visualization of five random feature kernel windows in the first convolutional layer of SDNet. The frequency domain is along the vertical, and the time domain is on the horizontal axis. Each filter is four time segments wide by 256 frequency segments high. Analysis is based on previous knowledge of filters and data.

An additional experiment was conducted to test the validity of using a dual stream
approach to the network models. In CNNs, a dual stream are two parallel CNNs that learn in
tandem independently of each other until the final layers of the network, where some fusion
method (typically a concatenation) integrates the information between the two before the
classifier. In this case, the dual stream used identical networks in each stream, the only difference
between the two being that one stream used the modified rectangular kernel windows, and the
other used the original square kernel windows. This model uses one stream to learn meaningful
features related to the frequencies of the audio spectrogram from the rectangular kernel windows,
and a second stream to learn the square kernel window features that have had surprisingly good
performance on the dataset.
Table 2 shows how the results compare when each model is the base of the dual stream. The overall results of AlexNet did slightly improve its validation accuracy with the use of a dual stream method. However, the other models suffered a decrease in overall accuracy from the dual stream approach. Intuitively, while a rectangular kernel may learn more meaningful frequency relationships in the audio spectrogram, the square kernel may learn features such as rhythm.

**Table 2.** Results from testing a dual stream approach. One stream used square kernel windows and the second used rectangular kernel windows. The network was concatenated before the final fully connected layers and the classifier.

<table>
<thead>
<tr>
<th>Model</th>
<th>Square Kernels</th>
<th>Rectangular Kernels</th>
<th>Dual Stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet[4]</td>
<td>66.2%</td>
<td>57.6%</td>
<td>68.4%</td>
</tr>
<tr>
<td>SqueezeNet[36]</td>
<td>64.8%</td>
<td>54.2%</td>
<td>50.6%</td>
</tr>
<tr>
<td>NIN-ImageNet[37]</td>
<td>67.4%</td>
<td>63.2%</td>
<td>60.2%</td>
</tr>
<tr>
<td>GoogLeNet[5]</td>
<td>72.8%</td>
<td>65.99%</td>
<td>72.4%</td>
</tr>
</tbody>
</table>

Another implementation of the classifier passes the features learned from the deep network models into a separate SVM classifier to learn the individual classes. This robust system has been used in several deep learning systems for improvement of classification. Table 3 shows the comparison between the deep neural network model’s classification results and the best SVM model for each model’s learned features.

**Table 3.** Comparison of CNN classifier versus a SVM classifier. SS indicates the input to the model was the concatenated source separated spectrograms. REC indicates the model parameters in the convolution and pooling layers were adjusted to have rectangular kernel windows, which spanned the entire height of the spectrogram (the frequency domain) in each layer.

<table>
<thead>
<tr>
<th>Model</th>
<th>CNN</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet[4]-SS</td>
<td>69.2%</td>
<td>70.66%</td>
</tr>
<tr>
<td>SqueezeNet[36]-SS</td>
<td>66.2%</td>
<td>64.67%</td>
</tr>
<tr>
<td>NIN-ImageNet[37]-SS+REC</td>
<td>68.2%</td>
<td>67.86%</td>
</tr>
<tr>
<td>GoogLeNet[5]-SS</td>
<td>75.2%</td>
<td>72.26%</td>
</tr>
<tr>
<td>SDNet[38]-SS+REC</td>
<td>72.0%</td>
<td>71.26%</td>
</tr>
</tbody>
</table>
In Table 3, the AlexNet SVM used the features from the second-to-last fully connected layer before the classifier, ‘FC6’, which outputs a 4096 element feature vector. The SVM used a second order polynomial kernel along with a one-versus-all model. The same type of SVM model is used for SDNet, whose 2048 element feature vector comes from its ‘FC6’ layer, being the last fully connected layer before the classifier.

Table 4 compares the best performance of each model in this research with the models proposed in previous research with the same GTZAN dataset. The models from other research papers in this table report accuracy using 10-fold cross validation due to the minimal size of the dataset. Tzanetakis’s [9] used a 30 dimensional feature extraction from the MARSYAS toolbox with Gaussian Mixture Models for his results. All the deep network models surpassed the original model proposed by Tzanetakis. GoogLeNet was additionally tested using ten-fold cross validation on the GTZAN dataset to better compare its performance to the published results from other models. GoogLeNet successfully surpassed the use of manually extracted features with pairwise and one-versus-all SVMs published by Li et al. and Lidy et al. However the additional models proposed by ___ did exceed the accuracy results from GoogLeNet even using the source separated input spectrograms.

<table>
<thead>
<tr>
<th>Model</th>
<th>Validation Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet[4] + SS</td>
<td>69.2%</td>
</tr>
<tr>
<td>SqueezeNet[36] + SS</td>
<td>66.2%</td>
</tr>
<tr>
<td>NIN-ImageNet[37] + SS + REC</td>
<td>68.2%</td>
</tr>
<tr>
<td>GoogLeNet[5] + SS</td>
<td>75.2%</td>
</tr>
<tr>
<td>SDNet[38] + SS</td>
<td>72.002%</td>
</tr>
<tr>
<td>GoogLeNet[5] + SS (10-fold Cross-Val)</td>
<td>79.1% (2.6% std. dev.)</td>
</tr>
<tr>
<td>G. Tzanetakis [9]</td>
<td>61% (4%)</td>
</tr>
<tr>
<td>Li et al. [10] SVM-Pairwise</td>
<td>74.9 % (4.97%)</td>
</tr>
</tbody>
</table>
Further testing investigated the effects of the dimensions of the deep network’s kernel window size and its overall performance of the network models. For this set of experiments, a simplified version of SDNet was implemented, hereafter called SimpleNet, with a series of different kernel windows in its convolutional and pooling layers. Table 5 shows the details of the network layers and the dimensions of the learned filters in each layer.

<table>
<thead>
<tr>
<th>Source</th>
<th>Model Type</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Li et al. [10]</td>
<td>SVM-1-vs-all</td>
<td>78.5% (4.07%)</td>
</tr>
<tr>
<td>Lidy et al. [11]</td>
<td>SVM-Pairwise</td>
<td>74.9%</td>
</tr>
<tr>
<td>Lee et al. [25]</td>
<td></td>
<td>90.6%</td>
</tr>
<tr>
<td>Wülfing et al. [26]</td>
<td></td>
<td>85.25% (3.5%)</td>
</tr>
<tr>
<td>Behún [27]</td>
<td></td>
<td>86.4%</td>
</tr>
</tbody>
</table>

The model ‘Full Freq’ uses similar kernel windows to the original SDNet that spans the entire frequency domain, immediately reducing the dimensions of the output data to a single frequency bin, or an image height of one. The model ‘Partial Freq’ implements kernel windows in its convolutional layers that spans only half of the frequency range of each input data layer. In this version, the height of the kernel window reduces by half at each successive layer. Also, the pooling layers only implement max pooling operations along the width of the data, or the time dimension, preserving the information in the frequency layer. The ‘Square’ model implements kernel windows as most image-based models do. In this structure, each layer’s kernel window uses the same height and width, allowing for robust, translation-invariant feature learning. In the ‘Partial Time’ model architecture, the kernel windows are shaped and scaled the same as those in the ‘Partial Freq’ model layers, except that they are oriented so that the larger kernel dimension is aligned with the image width, or the time dimension. Similarly, the ‘Full Time’ model architecture mirrors the kernel window shapes of the ‘Full Freq’ model layers, except that the kernel windows are oriented so the larger dimension is aligned with the image width, so that the
learned filters span the entire time range. The output data layer of the first convolutional layer of this model has a width of one, effectively 'squashing' the time dimension.

Table 5. Simplified SDNet (SimpleNet) Architecture with five versions, covering different shapes and spans of the kernel windows in each models’ convolutional and pooling layers. Here, Conv-X indicates a convolutional layer with X learned filters, and MaxPool xY indicates a max pooling layer of scale Y. Each convolutional layer is succeeded by a ReLU activation layer and a Batch Normalization layer.

<table>
<thead>
<tr>
<th></th>
<th>Full Freq</th>
<th>Partial Freq</th>
<th>Square</th>
<th>Partial Time</th>
<th>Full Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv-256</td>
<td>256×4</td>
<td>129×4</td>
<td>4×4</td>
<td>4×129</td>
<td>4×256</td>
</tr>
<tr>
<td>MaxPool x4</td>
<td>1×4</td>
<td>1×4</td>
<td>4×4</td>
<td>4×1</td>
<td>4×1</td>
</tr>
<tr>
<td>Conv-256</td>
<td>1×4</td>
<td>65×4</td>
<td>4×4</td>
<td>4×65</td>
<td>4×1</td>
</tr>
<tr>
<td>MaxPool x2</td>
<td>1×2</td>
<td>1×2</td>
<td>2×2</td>
<td>2×1</td>
<td>2×1</td>
</tr>
<tr>
<td>Conv-512</td>
<td>1×4</td>
<td>33×4</td>
<td>4×4</td>
<td>4×33</td>
<td>4×1</td>
</tr>
<tr>
<td>MaxPool x2</td>
<td>1×2</td>
<td>1×2</td>
<td>2×2</td>
<td>2×1</td>
<td>2×1</td>
</tr>
<tr>
<td>Conv-512</td>
<td>1×4</td>
<td>17×4</td>
<td>4×4</td>
<td>4×17</td>
<td>4×1</td>
</tr>
<tr>
<td>MaxPool x2</td>
<td>1×2</td>
<td>1×2</td>
<td>2×2</td>
<td>2×1</td>
<td>2×1</td>
</tr>
<tr>
<td>FC5-2048</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC6-2048</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FC7-10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each model was tested for the original grayscale input spectrograms and the source separated RGB input spectrograms, with the results for each shown in Table 6. Here, performance is not only reported for each model on each input data type, but the average of the performance for each model and for each input data type is shown in the final column and row of the table, respectively.

Table 7 shows the validation accuracy from the same models when implemented in a ‘voting’ classifier. In this implementation, each trained model ‘votes’ on a genre for each segment of a single input song’s audio spectrogram, these segments being taken as overlapping ten-second windows over the entire 30 second long audio spectrogram. This allows five votes to be cast on the same track, with the majority of the votes being the final model classification for that data sample.
Table 6. Reported validation accuracy for each version of the SimpleNet model with different kernel window shapes and sizes on the GTZAN dataset. Here ORIG indicates the input data was the original grayscale audio spectrograms and SS indicates the input was the RGB source separated spectrograms. Each model’s performance was averaged into the last column, and each method’s performance (ORIG or SS) was averaged into the last row.

<table>
<thead>
<tr>
<th>SimpleNet Model</th>
<th>GTZAN Validation Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ORIG</td>
</tr>
<tr>
<td>Full Freq</td>
<td>76.2%</td>
</tr>
<tr>
<td>Part Freq</td>
<td>69.2%</td>
</tr>
<tr>
<td>Square</td>
<td>76.2%</td>
</tr>
<tr>
<td>Part Time</td>
<td>69.2%</td>
</tr>
<tr>
<td>Full Time</td>
<td>55.6%</td>
</tr>
<tr>
<td>Avg method</td>
<td>69.3%</td>
</tr>
</tbody>
</table>

Table 7. Reported validation accuracy for each version of the SimpleNet model with different kernel window shapes and sizes on the GTZAN dataset when using a voting classifier. Here ORIG indicates the input data was the original grayscale audio spectrograms and SS indicates the input was the RGB source separated spectrograms. Each model’s performance was averaged into the last column, and each method’s performance (ORIG or SS) was averaged into the last row.

<table>
<thead>
<tr>
<th>SimpleNet Model</th>
<th>GTZAN Validation Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ORIG</td>
</tr>
<tr>
<td>Full Freq</td>
<td>80%</td>
</tr>
<tr>
<td>Part Freq</td>
<td>72%</td>
</tr>
<tr>
<td>Square</td>
<td>78%</td>
</tr>
<tr>
<td>Part Time</td>
<td>72%</td>
</tr>
<tr>
<td>Full Time</td>
<td>62%</td>
</tr>
<tr>
<td>Avg method</td>
<td>72.8%</td>
</tr>
</tbody>
</table>

Table 8 shows the results from including each version of SimpleNet in an ensemble classifier. Here, an ensemble classifier runs the same data sample through multiple models to get a predicted genre for each model. The majority of these classifications is the ensemble’s predicted genre for that data sample. As shown in Table 8, various implementations were tested, including utilizing all models for each version of the input data, and the omission of a single model in the subsequent testing. This method of testing enables the detection of the models that contribute the most meaningful predictions to the classifier, or which ones negatively impact the classifier’s performance. As shown in Tables 6, 7, and 8, the models with the greatest performance in isolation and models with the greatest contributions to the overall ensemble
classifier are the ‘Full Freq’ and ‘Square’ kernel window models. Also, the results in Table 8 suggest that the removal of the partial time kernel windows, ‘Part Time’, actually improves the overall ensemble classification accuracy. And the omission of the partial frequency, ‘Part Freq’, and ‘Full Time’ hardly impacts the ensemble performance at all. All these observations suggest that the most effective kernel window shapes for learning audio features are the full frequency range spanning kernels and the translation-invariant square kernel windows.

Table 8. SimpleNet ensemble classifier performance on GTZAN validation set. Here, O indicates the inclusion of the model in the ensemble and – indicates the omission of the model from the ensemble. The red text indicates the greatest loss of accuracy from the baseline accuracy (all models included) and the bold text indicates the least loss, or greatest gain, from the baseline accuracy.

<table>
<thead>
<tr>
<th>Model</th>
<th>Original Grayscale Input Spectrograms</th>
<th>Source Separated RGB Spectrograms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Freq</td>
<td>O - O O O O - O O O O O - O O O O O</td>
<td></td>
</tr>
<tr>
<td>Part Freq</td>
<td>O O O - O O O O O O O O O - O O O O O</td>
<td></td>
</tr>
<tr>
<td>Square</td>
<td>O O O - O O O O O O O O O - O O O O O</td>
<td></td>
</tr>
<tr>
<td>Part Time</td>
<td>O O O O O O O O O O O O O O O O O O</td>
<td></td>
</tr>
<tr>
<td>Full Time</td>
<td>O O O O O O O O O O O O O O O O O O</td>
<td></td>
</tr>
<tr>
<td>GTZAN Validation Accuracy</td>
<td>79% 75% 79% 76% 80% 78% 86% 83% 86% 82% 87% 85%</td>
<td></td>
</tr>
</tbody>
</table>

Tag Prediction

Tag prediction results were gathered from the ImageNet models and SDNet, modified for multi-class prediction. The ground truth of each clip is a vector of length 132, each tag is represented in this vector as either being present in the clip description (logical 1) or absent (logical 0).

Comparing the performance of each model in tag prediction is complicated by the difference in the evaluation of the models. Accuracy is the measure of true positives and true negatives over all the predictions. However, there are 132 total tags in the dataset, most of which are marked as absent in the ground truth for each sample. Therefore, a prediction of mostly
absent tags results in an accuracy value of greater than 90% for the majority of the predictions, and, as mentioned previously, the absence of a tag in the ground truth does not indicate that the tag description does not apply to the track. As such, the false positive and true negative counts of the model are not completely reliable values. The accuracy metric used in Table 10 is what Caffe training reports for the model validation accuracy while ignoring the tags absent from the description.

Another means of evaluating tag classification, without a probabilistic model, is the F1-score, which uses the precision and recall values of the resulting model to determine its overall performance. Precision (1) is a ratio of the correctly classified present tags to the total number of tags that are present in the ground truth. Recall (2) is the ratio of correctly classified present tags to the total number of the tags predicted to be present [39]. The value of this metric is that it focuses only on the tags that the ground truth says is present without focusing on the tags the ground truth says are absent. As previously mentioned, a tag absent from the ground truth does not necessarily mean the tag does not apply to the track description.

\[
\text{Precision/Sensitivity} = \frac{TP}{TP + FN} \quad (1)
\]

\[
\text{Recall} = \frac{TP}{TP + FP} \quad (2)
\]

\[
F1\text{Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)
\]

Yet another evaluation metric is the G-Mean (5), which combines the sensitivity and the specificity into a single value. An advantage of this metric is that it accounts for the large ratio of absent tags to the number of present tags. The sensitivity (1), also called precision or the true
positive rate, is the ratio of the correctly classified present tags to the total number of present tags. The specificity \((4)\), or the true negative rate, is the ratio of the correctly classified absent tags to the total number of absent tags. The G-Mean is the square root of the product of the two prior resulting values [39].

\[
Specificity = \frac{TN}{TN + FP} \quad (4)
\]

\[
GMean = \sqrt{Sensitivity \times Specificity} \quad (5)
\]

When evaluating a multi-class model, usually the evaluation includes the classes correctly labeled as well as the incorrectly labeled classes. However, the collection of the MagnaTagATune dataset was not conducted by music ‘experts’, nor by a single user, nor using a fixed list of tags for users to select from. The dataset is not completely labeled, which means a tag that is absent from the ground truth does not mean that tag does not apply to the track in question. Evidence of this fact is the presence of tracks in the dataset with no tags given to them, even though a different segment of the same song does as shown by clip IDs 61 and 71 in Table 14.

Therefore, the multi-class models used a different implementation of the loss layer during the training phase. The model separated the loss layer into two layers, one that evaluated the loss concerning the present tags and the other evaluated the loss concerning the absent tags in each label. The loss for the present tags was weighted normally and the loss for the absent tags was weighted less to lessen the impact of the absent tags. This allows the absent tags to still apply to the samples, while not discounting them entirely, as they may have been omitted without intention. To determine the best results from the models, three separate training simulations were
conducted on AlexNet with the weights for the absent tags set to three different values, 0.25, 0.5, and 0.75. For a weight of 0.25, the validation accuracy was 94.263%, for a weight of 0.5, the validation accuracy was 94.988% and for a weight of 0.75, the validation accuracy reduced to 95.519%. Table XXX shows the results for some of the tag predictions of the test set. Table 9 shows the results of testing on the validation set for the different loss weights with AlexNet.

Table 9. AlexNet results using different loss weights for the absent tags.

<table>
<thead>
<tr>
<th>Loss Weight</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
<th>G-Mean</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.0907</td>
<td>0.1587</td>
<td>0.1155</td>
<td>0.3907</td>
<td>0.9426</td>
</tr>
<tr>
<td>0.5</td>
<td>0.0963</td>
<td>0.1342</td>
<td>0.1122</td>
<td>0.3607</td>
<td>0.9499</td>
</tr>
<tr>
<td>0.75</td>
<td>0.0967</td>
<td>0.1077</td>
<td>0.1019</td>
<td>0.3242</td>
<td>0.9552</td>
</tr>
</tbody>
</table>

The general trend shown in Table 5 is that the precision improves with the increase in the loss weight and the recall improves with the decrease in the loss weight. The G-Mean and the F1-score both improve with the decrease in loss weight while the accuracy and precision improve with the increase in the loss weight. The increase in the F1-score indicates that the model more accurately predicts present tags without predicting additional tags absent from the track description, or omitting tags present in the tag description. The increase in the G-Mean indicates an overall improvement in the prediction of present tags out of all present tags, and the prediction of absent tags out of all absent tags from the tag description. The increase in the precision indicates that the model is predicting more present tags out of all present tags, which is better for a model whose data is not completely labeled. Ultimately, using a weight of 0.5 for the loss of the absent tags provides a better balance for predicting present tags while not predicting too many tags not present in the ground truth.

Table 10 reports the validation accuracy from training the models on the MagnaTagATune dataset. The validation accuracy is dependent only on the tags present, as
reported by the Caffe model during training, to account for the large ratio of absent to present tags out of 132 for each track. Table 11 reports the F1-score, which is a popular method of evaluation of the MagnaTagATune dataset. Table 12 reports the G-Mean of the validation data, which is a metric of accuracy that balances the accuracy of the present tags with the accuracy of the absent tags. The F1-score and the G-Mean both indicate better performance the closer the results are to one.

Table 10. Validation accuracy. ImageNet Accuracy is the accuracy each model reports on the ImageNet dataset from their original conference papers and from the documentation on Model Zoo, if applicable. MagnaTagATune accuracy is the accuracy reported on the validation set during Caffe training of the network ignoring the absent tags. ORIG indicates the input data to the model was the greyscale image of the spectrogram generated from the raw audio. SS indicates the input to the model was the concatenated source separated spectrograms. SQ indicates the model maintained its original square kernel windows in the convolution and pooling layers. REC indicates the model parameters in the convolution and pooling layers were adjusted to have rectangular kernel windows, which spanned the entire height of the spectrogram (the frequency domain) in each layer.

<table>
<thead>
<tr>
<th>Model</th>
<th>ImageNet Accuracy</th>
<th>MagnaTagATune Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Top 1</td>
<td>Top 5</td>
</tr>
<tr>
<td>AlexNet[4]</td>
<td>57.1%</td>
<td>80.2%</td>
</tr>
<tr>
<td>SqueezeNet[36]</td>
<td>57.5%</td>
<td>80.3%</td>
</tr>
<tr>
<td>NIN-ImageNet[37]</td>
<td>59.36%</td>
<td>-</td>
</tr>
<tr>
<td>GoogLeNet[5]</td>
<td>68.7%</td>
<td>88.9%</td>
</tr>
<tr>
<td>SDNet[38]</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

As shown in Table 10, the accuracy seems to have differing trends across the models for the modified kernel windows and for the application of the source separated data. These results differ from the noticeable patterns detected in the results from the genre classification testing. The validation accuracy for SqueezeNet and GoogLeNet seem to generally improve with the inclusion of the rectangular kernel windows, which is the opposite of their trends for the GTZAN dataset. The NIN-ImageNet’s validation accuracy seems to decrease with the use of the rectangular kernel windows, which is also the opposite of its trend from GTZAN genre classification. In addition, the use of the source separated audio spectrograms decreases the
model accuracy for both NIN-ImageNet and AlexNet, which is again, in opposition to their trends following genre classification with the GTZAN dataset.

Table 11. Validation F1-scores for each model on the MagnaTagATune dataset. ORIG indicates the input data to the model was the greyscale image of the spectrogram generated from the raw audio. SS indicates the input to the model was the concatenated source separated spectrograms. SQ indicates the model maintained its original square kernel windows in the convolution and pooling layers. REC indicates the model parameters in the convolution and pooling layers were adjusted to have rectangular kernel windows, which spanned the entire height of the spectrogram (the frequency domain) in each layer.

<table>
<thead>
<tr>
<th>Model</th>
<th>MagnaTagATune F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ORIG + SQ</td>
</tr>
<tr>
<td>AlexNet[4]</td>
<td>0.1121</td>
</tr>
<tr>
<td>SqueezeNet[36]</td>
<td><strong>0.1342</strong></td>
</tr>
<tr>
<td>NIN-ImageNet[37]</td>
<td>0.1044</td>
</tr>
<tr>
<td>GoogLeNet[5]</td>
<td><strong>0.1208</strong></td>
</tr>
<tr>
<td>SDNet[38]</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 12. Validation G-Mean scores for each model on the MagnaTagATune dataset. ORIG indicates the input data to the model was the greyscale image of the spectrogram generated from the raw audio. SS indicates the input to the model was the concatenated source separated spectrograms. SQ indicates the model maintained its original square kernel windows in the convolution and pooling layers. REC indicates the model parameters in the convolution and pooling layers were adjusted to have rectangular kernel windows, which spanned the entire height of the spectrogram (the frequency domain) in each layer.

<table>
<thead>
<tr>
<th>Model</th>
<th>MagnaTagATune G-Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ORIG + SQ</td>
</tr>
<tr>
<td>AlexNet[4]</td>
<td>0.3607</td>
</tr>
<tr>
<td>SqueezeNet[36]</td>
<td>0.5383</td>
</tr>
<tr>
<td>NIN-ImageNet[37]</td>
<td>0.6395</td>
</tr>
<tr>
<td>GoogLeNet[5]</td>
<td>0.5752</td>
</tr>
<tr>
<td>SDNet[38]</td>
<td>-</td>
</tr>
</tbody>
</table>

The F1-score of the models in Table 11 shows how the prediction of the present tags relates to all incorrect model predictions. This evaluation metric indicates that models using square kernel windows are more accurate in predicting the presence of tags, excepting that AlexNet again performs better using the rectangular kernel windows.
The G-Mean value shown in Table 12 balances the model’s true positive accuracy with its true negative accuracy. This metric shows the same trend as Table 10, that the models are trending toward the use of rectangular kernel windows for tag prediction. The best model according to this metric was GoogLeNet, modified for rectangular kernel windows, and using source separated spectrograms as the input.

Neither of the evaluation metrics used for tag prediction mirrored the results from the genre classification. General model performance did not follow any identifiable pattern and results were poor comparatively. The F1-score for all models did not extend past 0.14 and the G-Mean did not reach even 0.63, and both metrics indicate perfect prediction when their values are equivalent to one. The error in these results could partially be the result of the incomplete labelling of the dataset, though most likely the error was largely due to a mistake in the processing of the spectrogram images to fit the deep network input parameters.

Therefore, even though intuition implies otherwise, additional testing was performed to compare the results of the models after cropping the audio input into ten-second segments of the original track. Intuitively, a tag that applies to a track does not necessarily apply to the entirety of the track. However, the scale of the spectrogram images does impact the quality of features that are learned. Table 13 shows a comparison between the accuracy of the present tags for AlexNet in the validation set for input of the original track’s 29-second span and a cropped 10-second span. Both inputs were converted to spectrograms and resized to 256×256 pixels as input images to AlexNet.

**Table 13.** Comparison of the MagnaTagATune validation accuracy (ignoring absent tags) after training the AlexNet model on ten-second cropped segments of the audio tracks. ORIG indicates the input data to the model was the greyscale image of the spectrogram generated from the raw audio. SS indicates the input to the model was the concatenated source separated spectrograms. SQ indicates the model maintained its original square kernel windows in the convolution and pooling layers. REC indicates the model parameters in the convolution and pooling layers.
were adjusted to have rectangular kernel windows, which spanned the entire height of the spectrogram (the frequency domain) in each layer.

<table>
<thead>
<tr>
<th>Model</th>
<th>MagnaTagATune Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ORIG + SQ</td>
</tr>
<tr>
<td>AlexNet[4]</td>
<td>48.7%</td>
</tr>
<tr>
<td>AlexNet[4] With Cropped Input</td>
<td>86.92%</td>
</tr>
</tbody>
</table>

Table 14 shows the tags predicted for some of the test cases of the MagnaTagATune dataset. Each entry in the table reports the track’s song title, artist, the tag description (ground truth), and the predicted tags for three variations of the AlexNet model. The variations of the AlexNet model use different loss weights for the predicted absent tags, because tags absent from the ground truth does not indicate that those tags do not apply to the track in question. While it is not true for every case, decreasing the loss weight generally increases the number of tags predicted, which does increase the number of incorrect tags, but also includes more tags that may be present in the ground truth or that are still applicable to the track. For example, in track 71, the tags ‘Choir’ and ‘FemaleVocals’ in the ground truth imply that the tag ‘Singing’ is also applicable to the track. However, predicting more tags does not necessarily mean all of them will be applicable, again shown in track 71, where ‘singing’ and ‘novocals’ are both predicted tags.
Table 14. Examples of tag prediction results for test images on multi-label AlexNet as compared to their ground truth values. A dash indicates there was no tags predicted being present or no tags present in the ground truth of the dataset. The tags from the ground truth in the predictions are highlighted.

<table>
<thead>
<tr>
<th>Track Clip ID</th>
<th>Song Title</th>
<th>Artist Name</th>
<th>Ground Truth</th>
<th>Prediction (loss weight = 0.25)</th>
<th>Prediction (loss weight = 0.5)</th>
<th>Prediction (loss weight = 0.75)</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>Lesson 1 in G Major_ Prelude (James Nares)</td>
<td>Steven Devine</td>
<td>Harpsichord Classical Loud Fast</td>
<td>Harpsichord Classical Piano Baroque</td>
<td>Harpsichord Classical NoVocals Piano</td>
<td>Harpsichord Classical</td>
</tr>
<tr>
<td>61</td>
<td>Musicalische Exequien SWV 279 Teil I_ Concert in Form einer deutschen Begrabnis-Missa</td>
<td>American Bach Soloists</td>
<td>-</td>
<td>Vocals Classical Strings Opera Slow Singing FemaleVocals</td>
<td>Classical Guitar Strings Opera Violin Calm</td>
<td>Vocals Classical Strings Opera Calm FemaleVocals</td>
</tr>
<tr>
<td>71</td>
<td>Musicalische Exequien SWV 279 Teil I_ Concert in Form einer deutschen Begrabnis-Missa</td>
<td>American Bach Soloists</td>
<td>Classical Opera FemaleVocals Choir</td>
<td>Vocals Classical Solo Strings Opera NoVocals Harp Piano MaleVocals Slow Singing Calm</td>
<td>Classical Guitar Strings Piano Calm</td>
<td>Classical Opera Slow Calm</td>
</tr>
<tr>
<td>105</td>
<td>Under The Bright Lights</td>
<td>Rocket City Riot</td>
<td>Guitar Fast ElectricGuitar Rock</td>
<td>HardRock Loud Fast MaleVocals Punk Rock Metal</td>
<td>Guitar Loud Fast Rock Metal</td>
<td>Loud Rock</td>
</tr>
<tr>
<td>5329</td>
<td>Till My Cup Runs Over</td>
<td>Four Stones</td>
<td>Jazz Trumpet</td>
<td>Jazz Guitar Trumpet Drums NoVocals Slow Techno Beat</td>
<td>Jazz Trumpet Drums Slow Beat</td>
<td>Jazz Drums Slow Beat</td>
</tr>
</tbody>
</table>
FUTURE WORK

In this research, audio source separation proved to be a valuable pre-processing tool. Experimentation has shown that using FASST to separate three different audio sources to a musical track improved genre classification using neural networks by up to 7.2%. Means of extracting additional audio sources could be explored for future research involving audio pre-processing. At present, FASST is only used to extract three types of audio from a musical track: melody, bass notes, and drums. If information of other types of instruments, such as piano, strings or brass instruments, were extracted from an audio track, the performance of classification systems may improve even further. Additionally, research could be conducted on which separated sources provide the most meaningful information to feature learning and for which tasks, including but not limited to genre and tag prediction.

The final layer of most neural networks, before the output layer, provides a larger feature vector that summarizes the information provided from the audio track. This feature vector could be used for musical comparison towards the goal of music recommendation systems. With the improvements in classification tasks using source separation and neural networks, musical comparison and recommendation tasks could yield significant improvements as well.

The MagnaTagATune dataset provides similarity data for the musical tracks from a side game to TagATune. The objective of the game was for the users to select which of three sample tracks was the ‘odd one out’ or the least similar to the other two tracks. As such, the similarity data provided by the dataset is in the form of ‘upvotes’, so that the higher the vote count for a song in a triplet, the more dissimilar the game users perceived that song to be relative to the other two. Most recommendation systems are based on a similarity metric that recommends items that
are the most similar (or least feature-defined distance) to what a user is known to have liked, viewed or purchased, depending on the environment the recommender serves. In order to use the MagnaTagATune dataset for music recommendation, the complimentary published metadata from the Music Informatics Research Group at the City University of London is required, for it has preprocessed the inverse similarity data and converted it to constraints metadata [30].
CONCLUSIONS

In recent years, deep networks have greatly improved performance in image and audio recognition. The objective of this research was to analyze deep networks designed for image feature recognition in their application to learning features from audio spectrograms and explore an additional means of audio pre-processing for musical classification tasks.

Image-based deep network models have achieved classification accuracies that rival networks designed for audio data, and other published networks on musical genre classification. Four image-based network models were analyzed for their performance on musical audio data; AlexNet, SqueezeNet, NIN-ImageNet, and GoogLeNet. In addition, SDNet was analyzed as a means of comparison against a music-based deep network model.

An interesting observation is the networks impressive performance using their original square kernel windows, often exceeding their performance when modified for rectangular kernel windows. Additional testing with a simplified network model trained using different kernel window shapes further enforces this observation. Research into audio spectrogram analysis currently focuses on systems designed with rectangular kernel windows, which makes intuitive sense due to the ordered structure of the audio spectrograms. However with the impressive performance shown in this research by image-based network models with square kernel windows, audio feature training should also be open to deep networks with square kernel windows. Furthermore, experimentation has shown that the most effective rectangular kernel windows span either the entire frequency domain, or a portion of the time domain of the audio spectrograms.
This research has shown that utilizing audio source separation tools as a means of pre-processing the musical tracks provided significantly improved results to genre classification. All models analyzed for genre prediction in this research that used the source separated audio spectrograms increased in their classification accuracy.

In terms of the multi-class modifications to the network models for tag prediction, it is possible the variance between the performance of the deep networks in the tag prediction and genre classification occurred because of the change in the pre-processing of the audio data. The images for the GTZAN audio spectrograms were loaded into a Caffe model via the framework’s provided tool, while the images from the MagnaTagATune dataset were loaded via a custom Python script to account for the shape of the ground truth vectors.

The Python script created two collections of data for every partition of the dataset. The first collection contained the resized, reshaped spectrogram images, and the second contained the vectors indicating the presence of the tag descriptors. The raw audio files for each of the datasets were the same length, though the time segment audio cropping for data augmentation was only applied to the GTZAN data. However, the spectrograms for both datasets were resized to 256×256 dimensions, so the scale is not the same between the input data for the datasets. GTZAN used ten seconds for the spectrograms, while MagnaTagATune used the entire length of the audio track (29 seconds) for the spectrograms. The time domain of the spectrogram is resized to 256 pixels, therefore the GTZAN spectrograms represent about 0.04 seconds of audio frequencies for each column, and the MagnaTagATune spectrograms represent about 0.11 seconds of audio frequencies per column. Ultimately, to gain conclusive evidence for this MIR task requires more extensive testing with preparation for the more complicated prediction task that Caffe network models were not inherently designed to handle.
BIBLIOGRAPHY


