An effective analysis of Deep Learning based approaches for Audio based feature extraction and its Visualization

Dhiraj[1], Rohit Biswas[2], Nischay Ghattamaraju[3]

Emails: dhiraj@ceeri.res.in, biswasrohit143@gmail.com, nischay1234@gmail.com

Abstract: Visualizations help decipher latent patterns in music and garner a deep understanding of a song's characteristics. This paper offers a critical analysis of the effectiveness of various state-of-the-art Deep Neural Networks in visualizing music. Several implementations of auto encoders and genre classifiers have been explored for extracting meaningful features from audio tracks. Novel techniques have been devised to map these audio features to parameters that drive visualizations. These methodologies have been designed in a manner that enables the visualizations to be responsive to the music as well as provide unique visual experiences across different songs.

Keywords: Deep Neural networks, Convolutional Autoencoder, VGG, Alexnet, Audio Feature Extraction, Genre Classifiers, Audio Visualization, PCA, K-Means

1. Introduction

There has been extensive research on applying Digital Signal Processing to extract features from audio signals and using them to visualize music. A significant amount of this research focuses on frequency domain analysis [14]. While the results of these works have been outstanding, the features obtained by manual engineering are typically trivial, yielding similar visualizations. To make unique visualizations, it generally takes a lot of engineering on the graphics front and usually at the cost of responsiveness. Moreover, these approaches are very limited in terms of the types of features that they can capture. For example, the mood of a song, lyrical aspects and distant temporal relations in tracks are not captured by such features. Keeping these shortcomings in mind, in this research, we focus on the use of deep neural network architectures to extract features from audio tracks and use them to create audio visualizations that have the potential to either outperform conventional visualizations in terms of responsiveness or provide a new and unique visual experience.

The subsequent sections of this paper are organized as follows. *Section 2* covers details of related work which other researchers have attempted in the context of audio signal analysis and audio visualizations. In *Section 3*, we discuss the deep learning architectures that we have adopted for extracting features in the audio domain. We also delve into the details of the dataset and pre-processing techniques used for training these architectures. In *Section 4*, we explore the details of the audio visualizations that we have developed. *Section 5* demonstrates our analysis of the extracted audio features and explores various methodologies for mapping the audio features derived from deep learning systems to the space of parameters that drive audio visualizations. It contains our detailed observations and is well illustrated with the findings of our analysis. Finally, we draw our conclusions in *Section 6*.

2. Related work

Since the beginning of 20th century, people have devised techniques for visualizing music. Oskar Fischinger developed a mechanical instrument called Lumigraph to create imagery during his

performances. Robyn Taylor. et al. [1] used pre-programmed models to define how animations behave by extracting features from raw audio sequences. The harmonic relationship between colors and tones was highlighted by Ciuha et al. [2] by using a mathematical model. Similarly, a real-time animation of music based on tempo and loudness was developed by Dixon [3]. The efficiency and benefits of deep learning for audio analysis using various deep network architectures in terms of end-to-end learning was well presented by S.Sigtia et al [4]. A two-stage learning model was presented to effectively predict multiple labels for songs by J.Nam et al [5]. J. Schluter [6] focuses on unsupervised machine learning to learn features empirically from data and designed a music similarity estimation system based on the learned audio features. Eric J. Humphrey [7] emphasizes the importance of deep learning by using conceptual arguments for feature learning and deep architectures to overcome the drawbacks of traditional music signal analysis approaches. Kahng et al [8] have done tremendous work in the visualizations of neural networks. These works have facilitated researchers with better insight into neural networks. However, the application of deep learning in generating audio visualizations remains majorly unexplored. Gallagher M Downs [9] had employed PCA to visualize the training of neural networks in a lower dimensional space. There has recently been some research on audio visualizations in the context of deep learning systems [10].

3. Feature Extraction

It is a widely accepted notion that a well-trained neural network captures the essential features of data within its hidden layers. The success of deep learning architectures is accredited to their potential of capturing highly intricate features of data. The application of these features to other domains have greatly interested researchers. For example, Takahashi et al [11] have used features from a trained Audio Event Detection system to improve the performance of Video Activity Recognition. Mierswa et al [12] have extracted features for classifying audio data. The focus has been limited to only the deep learning domain. Our objective has been to apply these learned audio features to a different domain, namely, audio visualizations. This task is fairly new and unexplored. We have experimented with dynamic feature extraction from audio tracks [13] and mapping the extracted features to a small number of visual parameters that drive audio visualizations.

We have employed a deep convolutional neural network (CNN) [17] to extract features from segments of audio tracks. The two dimensional CNN allows us to capture intricate features from the audio track including temporal features from segments of the audio track. CNN's have performed exceptionally well on images [19] [22] and have been successfully applied to other domains as well. Wang et al [23] have recently demonstrated record-breaking results in Polarimetric synthetic aperture radar image segmentation using CNN's. Researchers have successfully employed CNN based architectures in the audio domain as well [16] [18].

We have explored two different kinds of deep learning architectures for feature extraction. For sensible mapping, the audio feature space must be similar in size to the visual parameters space which is typical of small dimensions. We have employed auto encoders [15] as they allow us to extract low dimensional features from data. We have also explored genre classifiers as feature extractors and looked at ways to reduce the dimensions of the extracted features. This section describes our training process with respect to the dataset used, pre-processing techniques employed, and the architectural details of the deep learning systems used in our study. All models have been developed using PyTorch [24] and trained on a single NVIDIA Geforce GTX 1080 Ti GPU.

3.1. Dataset

The Free Music Archive (FMA) [25] dataset is widely used for various MIR-related tasks [26] [28]. We have performed our experiments on the FMA Small dataset which consists of 8,000 tracks, 30 seconds each, distributed across 8 musical genres as well as the FMA Medium dataset which is similarly comprised of 25,000 tracks and 16 musical genres. All the experiments reported in this paper have been performed on the FMA Medium dataset. Most of the dataset, about 90%, has been utilized for training, 5% has been added to the cross-validation set which we have extensively used for tuning the hyper-parameters of our models and the remaining 5% has been added to the test set which has been primarily used for the analysis of the extracted audio features as explained in *Section 5*.

3.2. Pre-Processing

To create visualizations that are responsive to the music, we have divided the audio tracks into small segments. We have then extracted features for these short segments of tracks which have finally been used to update parameters that drive audio visualizations. This enables us to create visualizations that are in synergy with the audio track. Following the footsteps of Hershey et el. [29], we have split each track into 31 segments, 960 milliseconds in length. Research has shown that deep learning systems work well with audio signals transformed to the frequency domain [18]. We have transformed all the audio segments to their Mel-frequency cepstrum representations (MFCC). Using hop length of 10 milliseconds, Fast Fourier Transform (FFT) window size of 25 milliseconds and 64 mel spaced frequency bins, the pre-processed dataset consists of two-dimensional inputs of size 96 x 64. These numbers have made it easy for us to experiment with up to 5 layers of pooling. The dataset was finally scaled using Standard Normalization for the classifier related experiments and Minmax Normalization for the autoencoder.

3.3. CNN Autoencoder

An autoencoder can be used to generate feature-rich encodings of desired lengths [15] and hence they are the most obvious choice for this task. We have experimented with various convolutional autoencoder [33] architectures with up to 5 layers of pooling. After trying out various sets of hyper-parameters, our best performing models were trained for 40 epochs using Stochastic Gradient Descent (SGD) [34] with momentum 0.95, learning rate 0.0002 and batch size of 64. ReLU activation was used in all the hidden layers while Sigmoid activation was used in the final layer so that the autoencoder could accurately reconstruct the inputs which were scaled with Minmax Normalization between 0 and 1. The output of the encoder which is the middle layer of the network is of size 10 which allows us to perform a direct mapping with the visual parameters space which of the same size. Further details of the architecture are illustrated in *Figure 3.3.1*.

The choice of pooling layer has a great impact on the extracted features and was an important hyper-parameter in our experiments. The pooling operation adds translational invariance at the feature level and reduces the size of the feature maps [35] and a number of options were available with regard to pooling. Max Pooling and Average Pooling are popularly used across a wide range of architectures. Stochastic Pooling [39] has also been found to be very effective for regularizing convolutional neural networks and has achieved state of the art performance. We have experimented with Max and Average Pooling in our autoencoder architectures. Our observations were aligned with Zhang et al. [38] as we obtained a lower reconstruction error with Max Pooling and report the same results in this study.

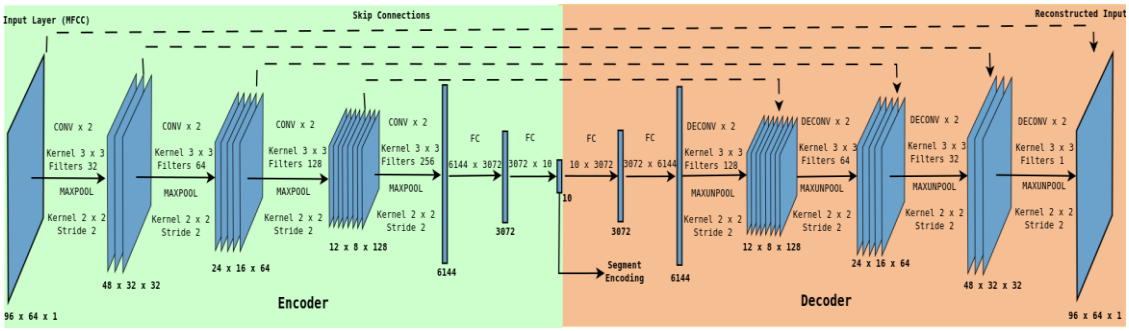

Figure 3.3.1 shows the architecture of NET-AE-SKIP (Abbreviations used for various operations: CONV = Convolution, MAXPOOL = Max Pooling, FC = Fully Connected, DECONV = Deconvolution, MAXUNPOOL = Max Unpooling). The input and output of the network is a single channel MFCC transformation of the audio segments. The middle layer is the smallest, of size 10, and its activations are used as features of the audio segments. Each arrow, emerging from the convolutional layers, represents a series of 3 operations, 2 valid convolutions followed by max pooling with a stride of 2x2. Notice that the encoder and decoder are symmetric, which allows us to to introduce skip connections from the convolutional blocks and to their corresponding deconvolutional blocks. The architecture of NET-AE, NET-AE-SHARED and NET-AE-SKIP *have been kept identical so that their extracted features are easy to compare. NET-AE and NET-AE-SHARED do not have the skip connections. NET-AE-SHARED additionally does not have separate weights for the decoder*

Architecturally, we have experimented with 3 kinds of autoencoders. Later, in *Section 5*, we have analyzed the features extracted using each of them in the context of audio visualizations. Following are the details of the autoencoder architectures used in this study:

1. A simple convolutional autoencoder in which the encoder and decoder have their own independent weights which we have referred to as NET-AE.
2. A convolutional autoencoder in which the weights of the encoder and decoder are tied [36]. Only the encoder is trained and the weights of each layer of the decoder are derived from the corresponding layer of the encoder. The tied weights have a regularizing effect on training. Moreover, this setup reduces the trainable parameters and thereby speeds up training. This architecture has been referred to as NET-AE-SHARED.
3. We have also experimented with skip connections in autoencoders as demonstrated by Mao et al. [45]. The skip connections made the task of input reconstruction easy and the optimization converged in just 20 epochs. This architecture has been referred to as NET-AE-SKIP.

3.4. CNN Genre Classifier

Features are primarily captured by an autoencoder for the purpose of reconstruction of audio data but this is not the only available option. We can also extract audio features from supervised deep learning systems such as a genre classifier [20] [21]. The features captured in the case of the genre classifier can be expected to be similar for tracks of the same genre. In contrast, the features obtained from an Autoencoder can vary drastically even within segments of the same track. Hence, the genre-specific features are bound to produce a very different visual experience.

For image classification tasks, the community has developed novel architectures and reported groundbreaking results [19] [22]. Some of these architectures have even been used in MIR-related tasks [29] [41]. We have borrowed some state-of-the-art CNN architectures, including AlexNet [19] and VGGNet [37], with pre-trained weights, adapting them to match the configurations of our pre-processed dataset, and fine-tuned them for the purpose of genre classification. We have experimented on Alexnet

(which we have referred to as NET-ALEX), VGG-11 (NET-VGG11), VGG-13 (NET-VGG13) and VGG-16 (NET-VGG16) with various sets of hyper-parameters and trained the best performing models for 25 epochs using SGD with momentum 0.95, learning rate 0.00001 and batch size of 64. The VGG architectures performed better with Batch Normalization [40]. When trained on the FMA Small dataset, VGG-13 and VGG-16, which are the larger models, overfit the training set very early. However, all models performed well when trained on FMA Medium. An example of the architectures is illustrated in *Figure 3.4.1*.

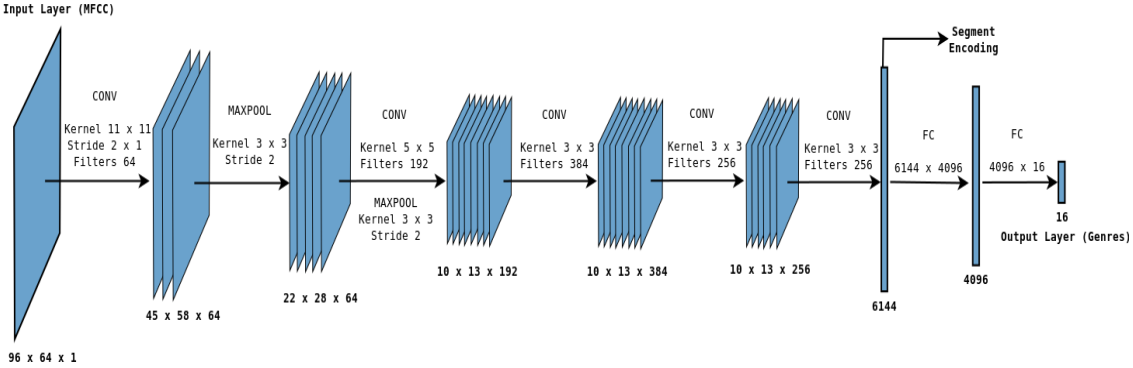

Figure 3.4.1 shows the architecture of Alexnet (NET-ALEX) adapted for our study (Abbreviations used for various operations: CONV = Convolution, MAXPOOL = Max Pooling, FC = Fully Connected). The first layer has been modified to make the model suitable for the single channel MFCC input of size 96 x 64 which is much smaller than the original input, of size 227 x 227, of the model. Similarly, the first fully connected layer had to be resized from 9216 units to 6144 units. The rest of the layers were initialized with pre-trained weights. Notice that the first fully connected layer is the feature extractor and its output, of size 4096, is considered as the encoding of an audio segment

Unlike the autoencoder, it is not straightforward to extract low dimensional features from a genre classifier. Each layer has a high dimensional activation size, and this makes it impractical to perform direct mapping of the audio features and visual parameters space. As we will explain in *Section 5*, we have employed various techniques to reduce the dimensionality of the features extracted from the genre classifiers. The choice of layer from which to extract the features is another hyper-parameter of the setup but we have found it computationally feasible to use only the final few layers of the model which are the smallest in size. For experimentation, we have extracted features from Layer 6 of NET-ALEX which is the 1st fully connected layer, Layer 8 of NET-VGG11 which is the 8th convolutional layer, Layer 10 of NET-VGG13 which is the 10th convolutional layer, Layer 13 of NET-VGG16 which is the 13th convolutional layer, Layer 14 of NET-VGG16 which is the 1st fully connected layer and Layer 15 of NET-VGG16 which is the 2nd fully connected layer.

4. Audio Visualizations

We have used the audio features extracted by using the deep learning systems to compute visual parameters for audio visualizations. To understand the impact of the transformations performed in the audio features space on the resulting audio visualizations, we have designed simplistic visualizations allowing us to observe subtle patterns in the visual parameters, the interactions between them as well as to study them independently. In practical scenarios, the visual parameters will be intricate in design, will often blend together and will be mapped to the feature space in more complex ways. However, for this study, we were compelled to keep them simple. The motivation was to critically analyze the audio

features mapped with visual parameters in a simple manner. We have created two types of visualizations and have made all our observations on both of them.

The first one is like a bar spectrogram (which we have referred to as VIZ-BAR) which can be seen in *Figure 5.3.1.2*. Each bar plays the role of a visual parameter instead of a frequency band. The length of the bar is determined by the activation of the corresponding visual parameter after it has been mapped against the audio feature space. VIZ-BAR has been very useful for making basic observations, for example, associating tracks with unique visual patterns, identifying inactive visual components and measuring the dynamicity across the visual components. However, VIZ-BAR fails to capture the inter-component interactions, variance in the individual visual components and in general is too simple in design to be compared against other audio visualizations.

To address the shortcomings of VIZ-BAR, we have designed a more realistic visualization (VIZ-REAL) which can be seen in *Figure 5.3.1.1*. It consists of visual elements that resemble typical audio visualizations more closely. Its visual elements include moving spheres with variable sizes, a colorful background, and a moving camera, all of which are parametrized by the extracted audio features. VIZ-REAL has been extremely effective in demonstrating the potential of our research in the mapping of audio features to visual parameters in a realistic manner. The resulting animations were interesting, intriguing and at the same time simple to study. It is a simple proof of concept of the use of audio features extracted from deep learning systems to audio visualizations.

Besides these two, we have also adapted several interesting visualizations designed by various artists by parameterizing their visual elements. The aforementioned audio visualizations have been developed in JavaScript using popular WebGL frameworks [27] due to the ease of rapid prototyping, ease of development and cross-browser support. There was an abundance of examples and resources in this regard which facilitated fast-paced development. Moreover, this has allowed us to showcase all our experiments over the internet for the benefit of the community.

5. Audio Feature Mapping

The primary focus of our research has been on exploring techniques using which we can derive parameters for driving the audio visualizations in interesting and meaningful ways. This is a very promising and, as of yet, an unexplored subject with a wide scope of research. The task of generating visualizations from deep learning systems can be approached from several varied paths. Below, we enumerate a few options that we had considered:

1. Generative models have been successfully used in the image [30] and speech [31] synthesis. The field has already gained immense popularity with its application in creative imagery [32]. There is immense potential in its application to inter-domain synthesis, for example, synthesizing images from audio signals.
2. An easier approach would be to derive parameters for audio visualizations from the audio signals by means of supervised learning. This would involve annotating small segments of songs with the desired visual effect. A lot of time and labor would be consumed but with this approach, we can gain a fine control over the visualizations.
3. Visual parameters for audio visualizations can also be determined from features accumulated from existing deep learning systems. This approach can work with both supervised and unsupervised systems and can be easily integrated with existing systems. It has the potential to yield a varied range of interesting visualizations across different deep learning systems.

However, it requires a detailed statistical analysis of the feature space and sophisticated algorithms to gain control over the visualizations.

In this research, we have focused only on the third approach listed above which involves the mapping of features extracted from deep learning systems to visual parameters. This was the most straightforward of the above approaches which seemed to be unexplored and have a lot of potential for generating interesting audio visualizations. The feature extractors that we have employed provide encodings of variable sizes depending on the architecture used. Using the autoencoders, we extract a feature size of 10 which is the same as our chosen number of visual parameters. We need to scale the extracted features to the range expected for the visual parameters before mapping them. *Section 5.2* discusses our approach towards feature scaling. Moreover, we can improve the mapping by incorporating the statistics of the extracted features. *Section 5.1* specifically illustrates the statistics of the raw encodings but the statistical analysis is continued in the remaining sections as and when deemed necessary. The genre classifiers yield high dimensional features which need to be transformed to a low dimensional space prior to mapping. Moreover, the low dimensional autoencoder features can also be majorly improved. *Section 5.3* discusses all the techniques for feature mapping that we have employed while attempting to address the aforementioned concerns. We progressively attempt to improve the mapping methodologies while trying to eliminate the drawbacks of previous approaches. In *Section 5.4*, we introduce some post-processing techniques that we have applied to the mapped features to improve the visual experience.

5.1. Feature Statistics

A naive approach is to directly map the extracted audio features to the visual parameters. While it might work and even produce interesting results, there is a great scope for improvement. By considering the underlying statistics of the extracted features, we can calibrate the mapping better and devise sophisticated mapping strategies. Two descriptors, namely mean and standard deviation, of the encoding data, played a very crucial role in our analysis. The mean of an encoding component determines how a corresponding visual parameter will be expressed. For example, if a component is mapped to the background color of the visualization or the speed of the moving camera, its mean will indicate how bright the background is or how fast the camera is moving at an average. The standard deviation of a component gives an estimate of how dynamic the corresponding visual parameter will be. A component with high standard deviation may result in rapidly changing background colors and accelerating cameras. Moreover, studying these statistics across all the components can give us an idea of the overall appearance of the visualization and its dynamicity which is very useful for comparing visualizations across tracks and musical genres.

We have analyzed features for individual tracks, for entire musical genres, and across all tracks in the test set with the objective of understanding the features extracted from all the models previously discussed in *Section 2*. *Figure 5.1.1*, *Figure 5.1.2* and *Figure 5.1.3* present some important results from our statistical analysis of the raw audio features in the context of creating audio visualizations. Other statistics are discussed in subsequent sections where they are more relevant.

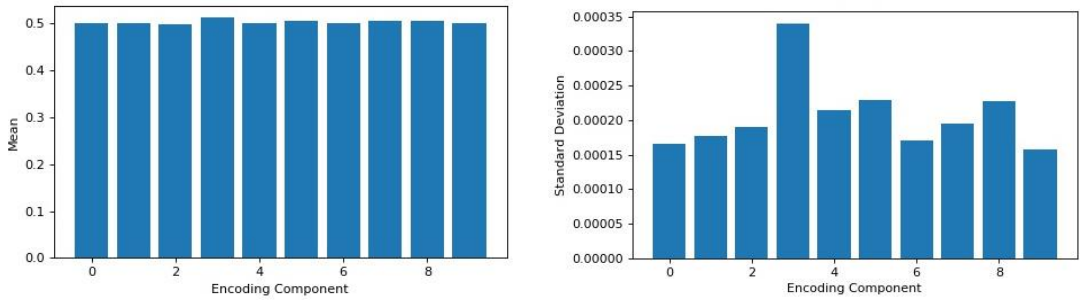

Figure 5.1.1 shows the raw encodings across all tracks obtained from NET-AE. Raw encodings generated by NET-AE and NET-AE-SKIP exhibit similar mean and extremely low standard deviation across the encoding components

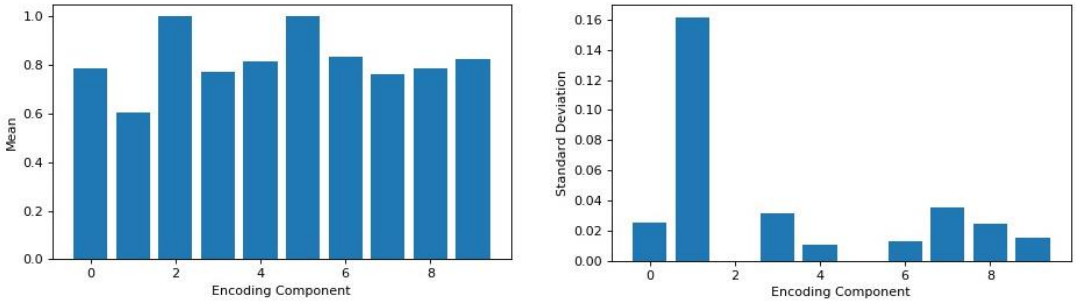

Figure 5.1.2 shows the raw encodings across all tracks obtained from NET-AE-SHARED. The features have varying means for the different encoding components as well as better variance, but these can still not be directly translated into visual parameters as those parameters will not considerably vary across the track and the resulting visualizations will lack dynamicity

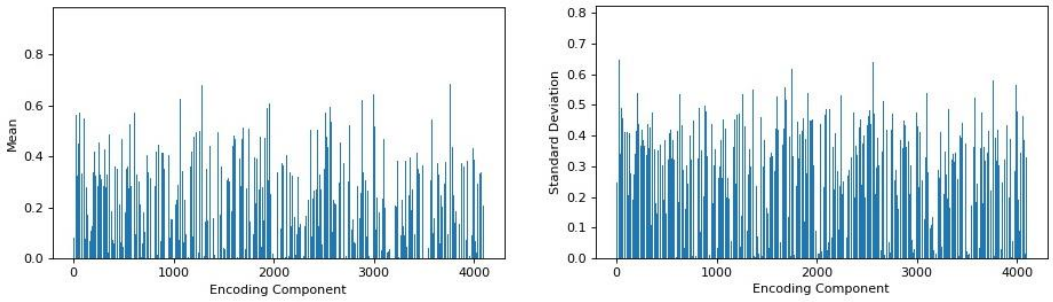

Figure 5.1.3 shows the raw encodings across all tracks obtained from the final layer of NET-ALEX. Notice the means varying across the encoding components and high standard deviation. These are potentially good features for generating dynamic audio visualizations. However, the features obtained from all the genre classifiers are very high dimensional, averaging at about 4000 features, and hence there is no simple way to map them directly to about 10 visual parameters

5.2. Feature Scaling

The range of feature values is different for different architectures. We have, for the sake of simplicity, set our desired values for the visual parameters to be in the range from 0 to 1 and need to map the feature values to the same range. Moreover, as observed from the statistics of the raw data in *Section 5.1*, the raw features are not suitable for mapping. To address these challenges, we can scale the raw features to our

desired range. We have employed Standard Normalization as well as Minimax Normalization to scale the encodings between 0 and 1. We have also experimented with scaling features independently for each encoding component as well as scaling the features across all the encoding components.

Minimax Normalization can be easily employed to scale data to a given range and it was thus our first choice. *Figure 5.2.1* shows that scaling the raw encodings also highlights the differences between each encoding component which are otherwise very subtle in the raw encoding data. One drawback of Minimax Normalization is that it is extremely susceptible to outliers and the standard deviation of the scaled encodings is still very low on account of being suppressed. We can improve the standard deviation of the encodings by using Standard Normalization. The aforementioned technique is typically used to standardized data to a mean of 0 and a standard deviation of 1. But, we have instead used a mean of 0.5 and standard deviation of 0.4 so that the encodings lie in the desired range from 0 to 1 and the standard deviation is scaled up to a reasonable value. Any values scaled beyond the desired range have been clipped. An example is illustrated in *Figure 5.2.2*. Both the aforementioned methodologies have been used in different parts of our study. In *Section 5.3*, we have illustrated several such examples.

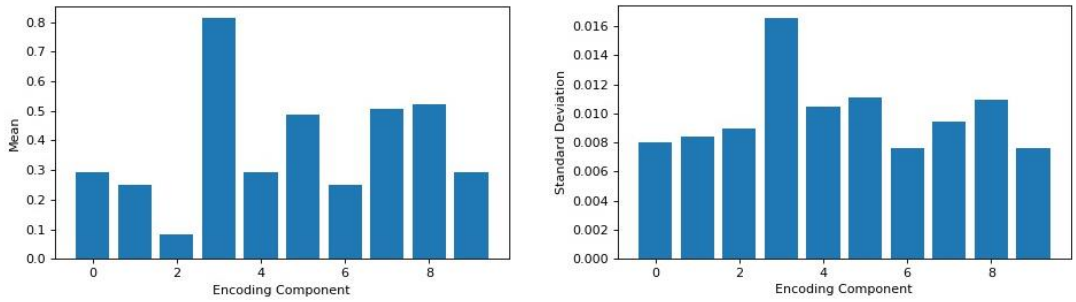

Figure 5.2.1 shows the features across all tracks obtained from NET-AE scaled across all the components using Minimax Normalization. It can be observed that the means vary considerably across the components and the standard deviations are also scaled up, though the deviations are still much smaller than desirable. These features will result in similar looking visualizations which are not very dynamic.

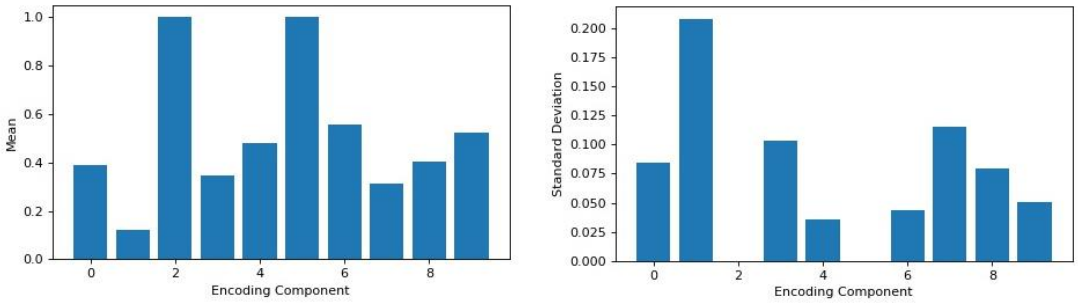

Figure 5.2.2 shows the features across all tracks obtained from NET-AE-SHARED scaled across all the components using Standard Normalization. As compared to the raw encodings, illustrated in Figure 5.1.2, the means are well distributed between 0 and 1. Moreover, the standard deviations have also reasonably scaled up. These features will result in dynamic visualizations. Note that scaling across the components has resulted in suppression of the mean of component 1 and the standard deviations of components 2 and 5.

Since the inputs to the feature extractors were normalized and considering the symmetry of the deep learning architectures it is reasonable to expect that the output features are of the same scale. The feature statistics further reinstate our assumption as all the encoding components have means and standard deviations of approximately the same scale. Considering this, we could scale the feature values across all

the encoding components as illustrated in *Figure 5.2.1* and *Figure 5.2.2*. This has the additional effect of preserving the relative means and deviations between the encoding components. *Figure 5.2.3* compares the scaled encodings obtained from all the autoencoders. A downside of scaling across all the components is that if the distribution of a component differs from the average distribution across the components, then it can get suppressed. Such a component can potentially exhibit characteristic features which we might lose by adopting this method of scaling. To address this problem, we can scale each feature independently to the desired range as shown in *Figure 5.2.4*. This method ignores the inter-feature means and deviations. However, the deviations within each feature are highlighted and each component is expressed well. This section discussed scaling techniques applied to the raw encodings but in the subsequent sections, we will see that the same techniques can be used on transformed encodings.

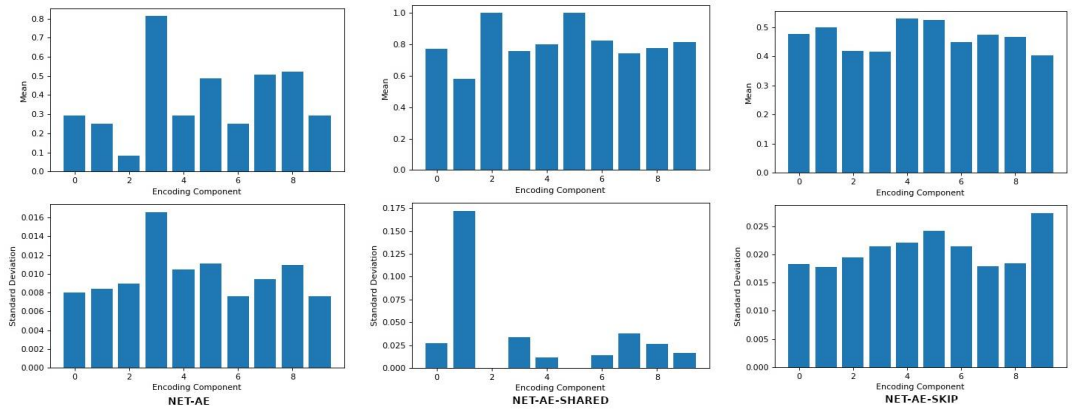

Figure 5.2.3 compares the features across all tracks obtained from the 3 autoencoder architectures after scaling them across all the components using Minmax Normalization. NET-AE (Left) shows varying means across the components but the standard deviation of the components is still very low. NET-AE-SHARED (Middle) shows a lot of variation in the standard deviation of the components. NET-AE-SKIP (Right) shows the least variation in both mean and standard deviation across the components

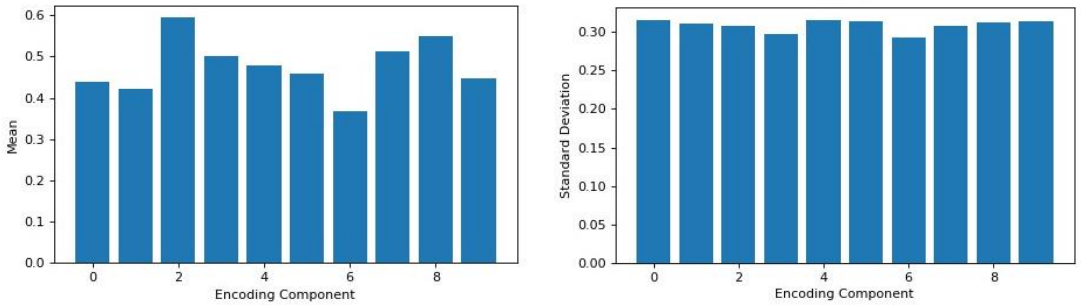

Figure 5.2.4 shows the encodings across all tracks obtained from NET-AE scaled independently for each component using Standard Normalization. The standard deviation has now scaled better than when scaling across the components, as was illustrated in Figure 5.2.2, and none of the components are suppressed

5.3. Feature mapping methodologies

In this section, we describe all the feature mapping methodologies that we have experimented with. Methods I and II involve direct mapping and hence are only applicable to the features extracted from the autoencoders. Methods III explores a more sophisticated approach to feature mapping and a technique of dimensionality reduction that has been experimented on features obtained from all the architectures. Methods IV attempts at making distinctive visualizations for different songs and, though applicable to features obtained from all the architectures, it has yielded better results in the case of the genre classifiers. In Method V, we combined the previous two approaches to yield a more sophisticated feature mapping technique.

5.3.1. Method I - Deterministic Mapping

The most straightforward way to map the raw audio features to the visual parameters is to perform a deterministic one to one mapping between them. The mapping is arbitrarily performed but it does not change throughout the analysis. However, the features need to be scaled before mapping them. We have experimented with scaling by and across features both yielding interesting results. Note that direct mapping requires the feature space to be identical in size to the visual parameters space and hence it can only be applied to the features extracted from the autoencoders.

The features obtained from all the autoencoder architectures produce dynamic visualizations when an appropriate feature scaling methodology is used as illustrated in *Figure 5.3.1.1 and Figure 5.3.1.2*. All the scaling methods performed well except Standard Normalization across the components on NET-AE and Minmax Normalization as they suppressed the standard deviation of the encodings.

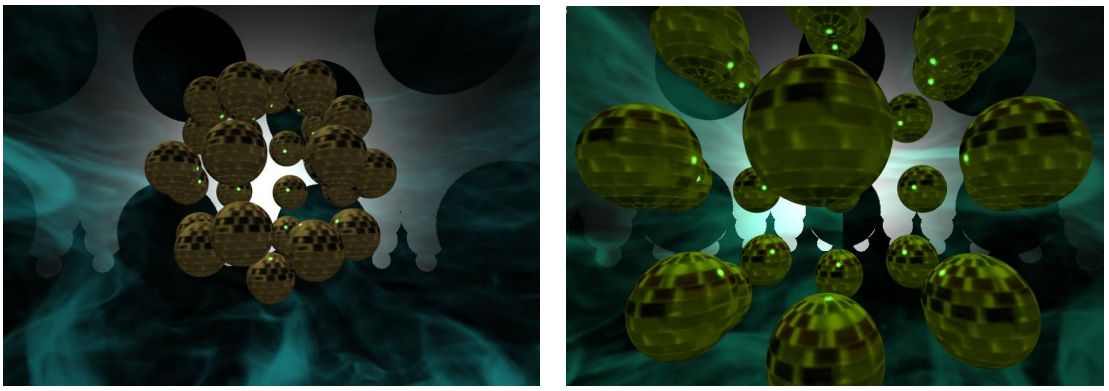

Figure 5.3.1.1 shows the visualizations of two different segments of the track 'Homemade Rap' by 'Pot-C' on VIZ-REAL. The features used were extracted from NET-AE-SKIP and scaled across the encoding components using Standard Normalization. Each audio segment has a different encoding and consequently varying visual parameters. This results in interesting and dynamic visualizations changing throughout the track

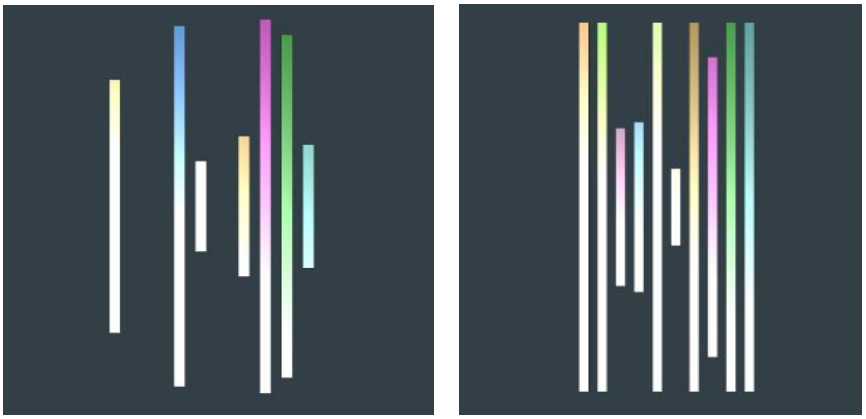

Figure 5.3.1.2 shows the visualizations of two consecutive segments of the track 'L's' by 'Fleslit'. The features used were extracted from NET-AE-SHARED and scaled by features using Minmax Normalization. The changes in the visualizations are generally smooth but they often change abruptly along with the track. These abrupt changes have been dealt with using post-processing techniques discussed in Section 5.4

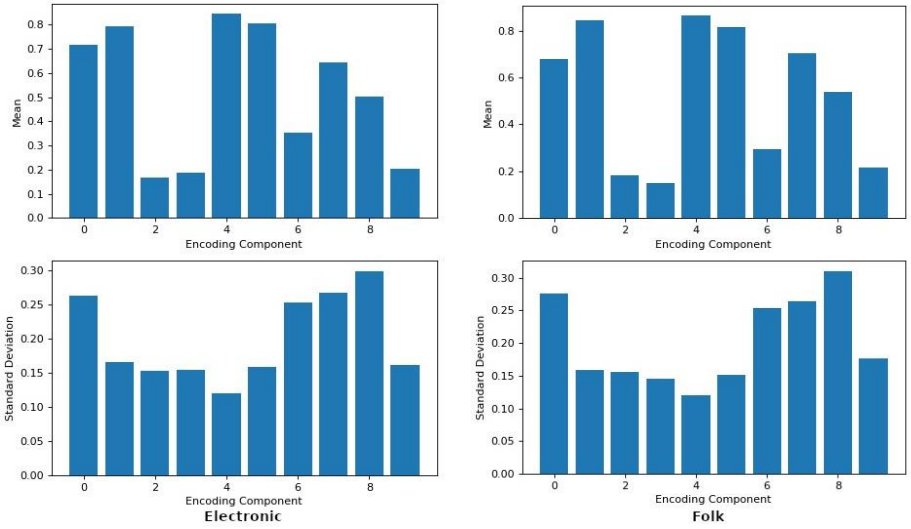

Figure 5.3.1.3 shows the features across two different musical genres, Electronic (Left) and Folk (Right), obtained from NET-AE-SKIP and scaled by features using Standard Normalization. The statistics are extremely similar for the two musical genres suggesting that the autoencoder architecture did not capture genre-specific features and the resulting visualization of tracks from these genres will be rather similar

We observed a few drawbacks of mapping the autoencoder features using Deterministic Mapping. It was illustrated in *Figure 5.2.2*, that when scaling across features, the mean and standard deviation of some components are suppressed. Correspondingly, the mapped visual parameters are not expressed well. This can be handed by scaling each feature independently, though, at the cost of variaitions between the encoding components. Moreover, *Figure 5.3.1.3* also shows that the statistics of entire musical genres can be very similar. Even our track level analysis revealed that the means and deviations are not too different for different tracks. Though the intrinsic variation of the features yields dynamic visualizations

but these visualizations tend to similar across the tracks irrespective of the scaling methodology used. In *Section 5.3.4*, we will explore methods for creating different visual experiences for different tracks. Finally, the approach of deterministic mapping does not allow us to control the extent to which a visual parameter is expressed. We could sort the components by their mean or variance and selectively map them to visual parameters but in *Section 5.3.3*, we will look at a more sophisticated technique for prioritizing the visual parameters.

5.3.2. Method II - Random Mapping

In this experiment, we randomize the mapping of the audio feature space to the visual parameters space with the remaining experimental setup being identical to that employed in Method I. A one to one mapping is performed between the encoding components and visual parameters, but each time at random. A major drawback of performing a deterministic mapping is that some of the visual parameters are expressed less than others. Random mapping clearly addresses this problem.

Figure 5.3.2.1 shows the visualization of the same segment of the track 'Max Awe' by 'Pot-C' on VIZ-BAR using Random Mapping. The features used have been extracted from NET-AE and scaled across all the encoding components using Standard Normalization. Notice how each bar in the left image corresponds to an identical bar in the right image, for example, the 6th bar (Left) corresponds to the 8th bar (Right) and the 2nd bar (Left) corresponds to the 5th bar (Right)

The mean and variance are distributed across all the visual parameters. Consequently, over multiple iterations of the same or different tracks, all visual parameters are expressed equally. The total number of possible random mappings is large (equal to the factorial of 10 which is the number of visual parameters). Hence, we got to see various interesting shapes in VIZ-BAR for the same segment of a track, an example being illustrated in *Figure 5.3.2.1*. Though the visualizations produced in this experiment are interesting, it can very hard to visually identify patterns and compare visual components across tracks. Moreover, If the intention is to make similar visualizations for similar tracks, then this mapping approach is not appropriate.

5.3.3. Method III – Mapping the Principal Components

One of the limitations, highlighted in the previous experiment with Deterministic Mapping, was related to the lack of control over the visual parameters. Typically, in a visualization, certain visual elements need to be very dynamic, for example, the motion of a bouncing ball while others, such as the color or size of the ball, can be subtler. In this experiment, we incorporated the underlying variance of the data while performing the mapping. We employed Principal Component Analysis (PCA) to transform the encodings of all the tracks in the test set to its principal components. This resulted in a new feature space in which the encoding components were ordered by variance. *Figure 5.3.3.1* shows the mean and variance of the transformed encodings. The visual parameters were also ordered by relevance such that

the visual elements which were required to be more dynamic were mapped to the initial principal components and then a one to one mapping was performed deterministically between them.

An additional advantage of PCA is that it enables us to bring down the dimensionsionality of the feature space. This is extremely useful in the context of the features extracted from the genre classifiers which are very large, averaging at around 4000 features. After the PCA transformation, we have reduced the number of components to 10 which is equal to the number of visual parameters. This results in the loss of up to 60% of the variance of the data but our experiments show that the retained components can easily capture the nuances between genre-specific features. *Figure 5.3.3.2* shows that some genre level differences are captured by the PCA transformations on the features extracted from the genre classifiers. However, these differences are subtle and can be accredited to the feature extractor rather than the mapping methodology as such differences are absent in the case of the features extracted from the autoencoders. Moreover, the visualizations produced by the features obtained from different classifiers also yielded slightly different visual experiences, as shown in *Figure 5.3.3.3.*

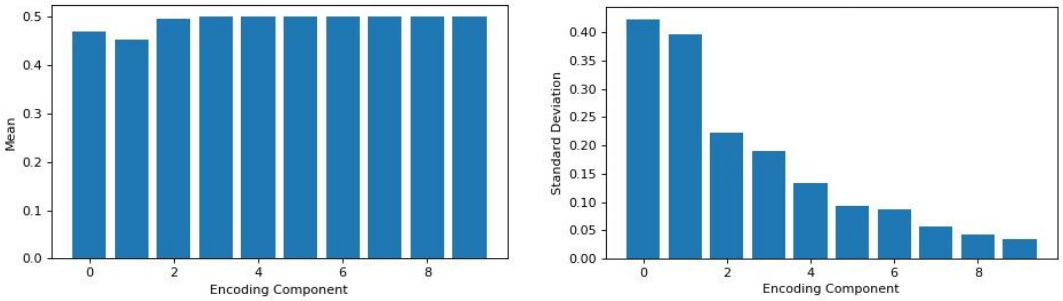

Figure 5.3.3.1 shows the mean and standard deviation of the encodings across all the tracks after being PCA transformed. The features have been obtained from NET-AE-SHARED and scaled across all the components using Standard Normalization after the PCA transformation. The mean of the of each component is around 0.5 but the standard deviations exhibit decent variation which is also ordered by the principal components. Note that the visual parameter mapped to component 0 will take up nearly all possible values in its range while the one mapped to component 9 will deviate only slightly around its mean

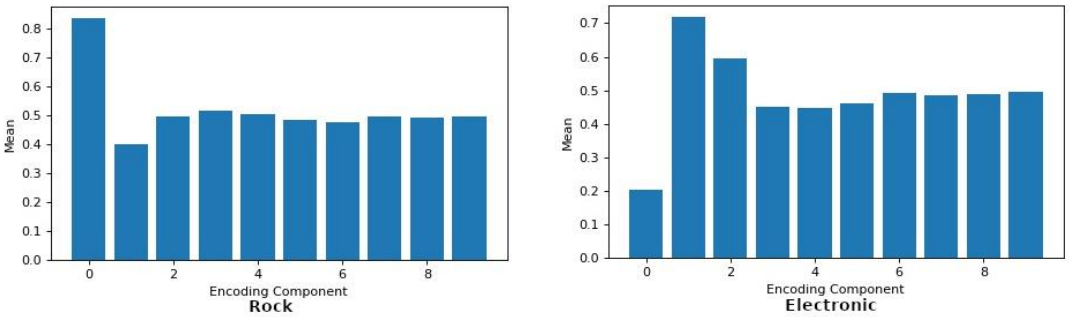

Figure 5.3.3.2 compares the means of the PCA transformed encodings of two musical genres, Rock (Left) and Electronic (Right), using features obtained from Layer 10 of NET-VGG13, scaled across the components using Standard Normalization. Note that there is a slight difference in the means of the first 3 principal components. Since these are also the most dynamic components, these features will produce noticeably different visual experiences across these musical genres

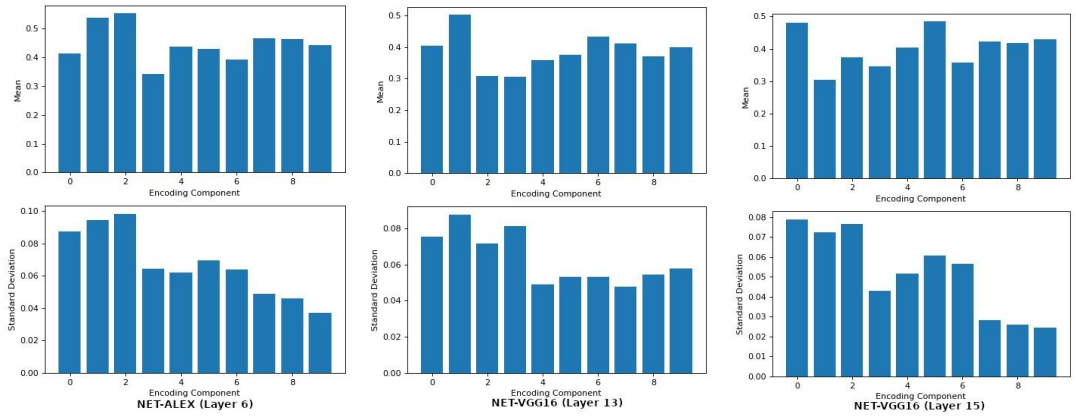

Figure 5.3.3.3 compares the PCA transformed features across the genre Folk, obtained from three different classifier layers, Layer 6 of NET-ALEX (Left), Layer 13 of NET-VGG16 (Middle) and Layer 15 of NET-VGG16 (Right). The features have been scaled across the components using Minmax Normalization. While the PCA transformed features from the genre classifiers, when analyzed across all the tracks, exhibit only subtle differences, a genre level analysis reveals slight differences in the distribution of the encoding components. Hence, the visualization produced by each set of features will vary.

Though the effect of the PCA transformation is apparent in VIZ-BAR, the analysis was more effective on VIZ-REAL, illustrated in *Figure 5.3.3.4*, which consists of realistic visual elements with emphasized dynamicity. We have scaled the PCA transformed features across all the components using either Minmax or Standard Normalization. The unscaled PCA transformed encodings have negative values and are on different scales than required which is not desirable. Moreover, scaling by features destroys the captured order of standard deviations and it is thus not suitable for use with this mapping technique

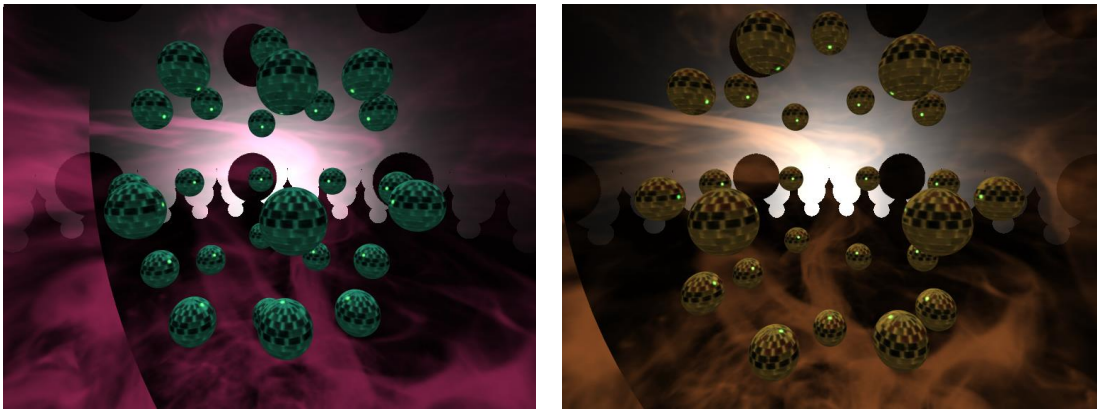

Figure 5.3.3.4 shows the visualizations of two segments of the same track 'It Was Me' by 'Derek Clegg' on VIZ-REAL created using the PCA transformed features obtained from NET-AE-SHARED and scaled across the components using Standard Normalization. The colors are mapped to the initial principal components and hence they are very dynamic while the positions and sizes of the spheres are mapped to the final components and they do not vary much across the track

5.3.4. Method IV – Cluster centroids as visual parameters

After the experiment with principal components, we were able to gain a finer control over the mapping between audio features and visual parameters. We were able to prioritize the visual components better and optimize their dynamicity. The resulting visualizations were interesting and in excellent

synchronicity with the audio. But the visualizations of different audio tracks still looked the same which was a big drawback of the previous approaches.

In this experiment, we have tried to capture the similarity between audio segments and to generate similar visualizations for similar segments. We have used K-Means to cluster the raw encodings into 10 clusters so that for each cluster, there is a corresponding visual parameter. Furthermore, we had to devise a way for the encodings to be mapped to the visual parameters. A naive approach was to use the cluster membership criterion for activating the visual parameters. This would result in only one active visual parameter at a time which might not be very useful. A better approach is to calculate the membership of an encoding to each cluster using its Euclidean distances from the cluster centroids in the encoding space. The encodings were thereby transformed into a lower dimensional space of Euclidean distances. We have used the inverse of the Euclidean distances to compare the similarity of the features to the cluster centroids. The underlying intuition behind these transformations was that each encoding would activate multiple visual parameters, and the magnitude of activation would be determined by the similarity score of the encoding with the corresponding cluster centroid. Thus, similar encodings would tend to yield similar activations in the visual parameters space.

The K-Means transformation is scalable across different feature dimensions and hence it could easily be applied to all the architectures. An analysis of best clusters suggests that the ideal number of clusters varies by the model used. For example, the ideal number of clusters were found to be around 7 for NET-VGG13. However, we have chosen 10 clusters for our study to make the visualizations comparable across the experiments. Analyzing the K-Means transformed feature space across all tracks did not make much sense as its mean and standard deviation were uniformly distributed across all the tracks. However, genre level analysis, as in shown *Figure 5.3.4.1* and *Figure 5.3.4.*2, revealed contrasting statistics for the features obtained from both the autoencoder and genre classifiers, with the latter exhibiting more prominent distinctions between genres. The resulting visualizations now vary not only within the track but also across them. Moreover, the K-Means approach works well with the different combinations of scaling methodologies previously discussed. *Figure 5.3.4.3* exemplifies the observation that the visualizations produced with this method are especially contrasting when using the features obtained from the genre classifiers. It should, however, be noted that information is stored in the encoding data in a non-linear way and hence a non-linear clustering strategy might yield better results than K-Means Clustering and we intend to revisit this idea in the future.

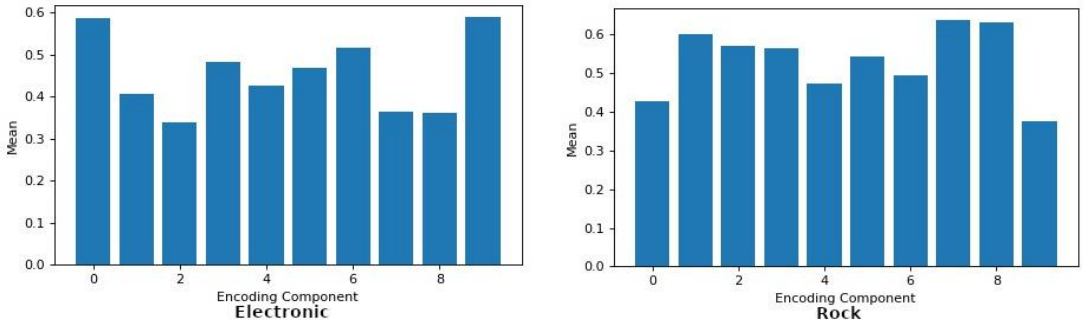

Figure 5.3.4.1 shows the means of the components of the encodings transformed with K-Means across two genres, Electronic (Left) and Rock (Right), obtained from NET-AE and scaled by features using Standard Normalization. The variation in the means of few of the components is apparent. An autoencoder will typically capture characteristic features of songs and be genre agnostic. However, it is not completely unreasonable to assume that there will be some characteristic similarities between songs of the same genre. The K-Means transformation highlights such genre level similarities in the features captured even from the autoencoder models

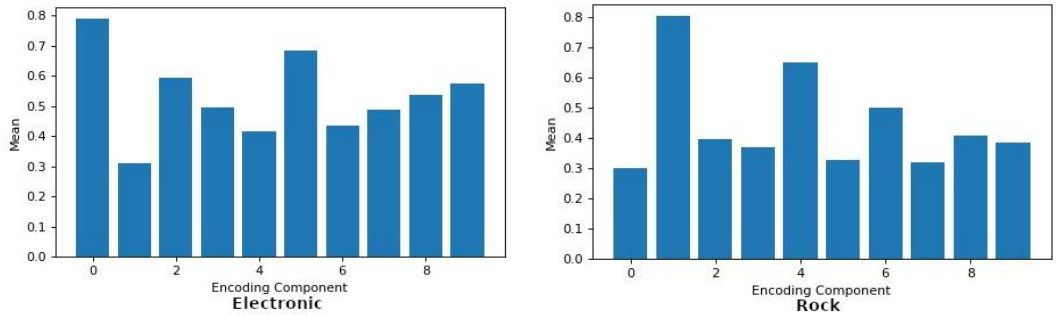

Figure 5.3.4.2 shows the means of the components of the encodings transformed with K-Means across the two genres, Electronic (Left) and Rock (Right), obtained from Layer 15 of NET-VGG16 and scaled by features using Standard Normalization. The difference between the genres, as expected, is more prominent than in the case of the autoencoders, as was depicted in Figure 5.3.4.1. Some of the components have contrasting means between the two genres. The resulting visualizations will look very different for tracks in the two genres

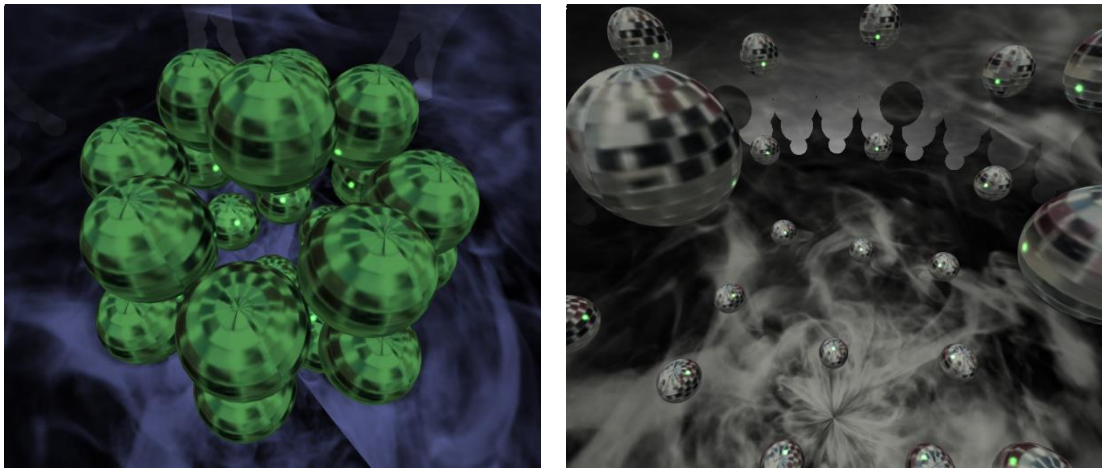

Figure 5.3.4.3 shows the visualizations of tracks from 2 different genres, Electronic - 'ELI' by 'Borful Tang' (Left) and Rock - 'Prototype' by 'The Modern Airline' (Right). The features used were obtained from Layer 15 of NET-VGG16, processed using K-Means and scaled by features using Standard Normalization. The variation in many of the visual components, like the positions of the spheres and color are apparent. The visualizations produced by these tracks are different from each other in terms of both in terms of the overall appearance and dynamicity of features

5.3.5. Method V - Combining PCA and K-Means

The PCA transformations allowed us to control the expression of visual parameters but the differences between the visualizations of different tracks were not very prominent. The K-Means transformations allowed us to produce different visual experiences for different tracks but it does not come with the aforementioned control over the expression of visual parameters. In this method, we try to combine the previous two approaches to create different visualizations for different tracks as well as control the expression of visual parameters. The methodology is simple and it involves, first, transforming the raw features using K-Means, followed by reduction of the newly transformed space into its principal components. *Figure 5.3.5.1* shows how the transformed feature space has different means across the genres and additionally the standard deviations are ordered by their magnitude across the components. The visualizations produced, as illustrated in *Figure 5.3.5.2*, vary across the genres and show higher dynamicity in the important visual parameters.

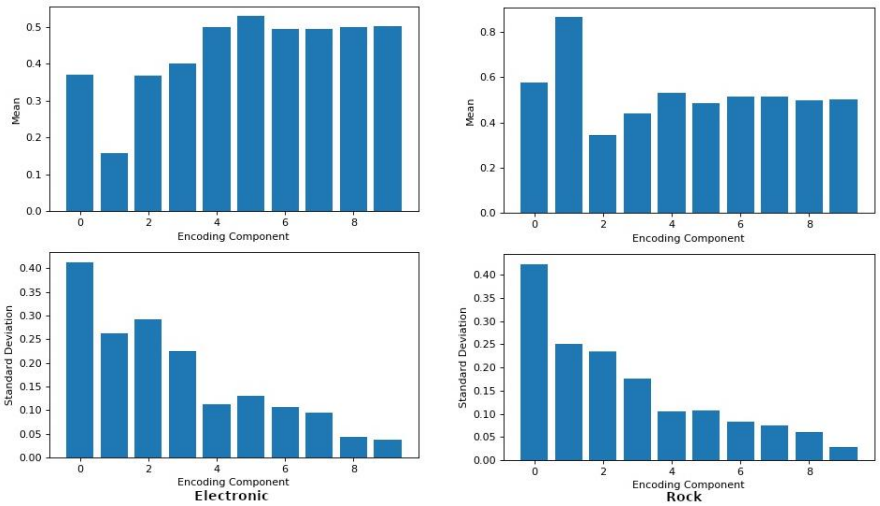

Figure 5.3.5.1 shows the statistics of the encoding components transformed with K-Means followed by PCA across two genres, Electronic (Left) and Rock (Right), obtained from Layer 15 of NET-VGG16 and scaled across the encoding components using Standard Normalization. The means are different across the encoding components and hence the visualizations produced for the two genres will look very different. Moreover, the variance is ordered by the components because of the PCA transformation and hence we can prioritize visual parameters accordingly

Figure 5.3.5.2 shows the visualizations of tracks from 2 different genres, Electronic - 'ELI' by 'Borful Tang' (Left) and Rock - 'Prototype' by 'The Modern Airline' (Right). The features used were obtained from Layer 15 of NET-VGG16, processed using K-Means followed by PCA and scaled across the encoding components using Standard Normalization. The position and size of the spheres were mapped to the initial principal components while the color of the sphere to the final components. The dynamicity in the former was apparent in both the tracks and their overall appearances are clearly different.

5.4. Post Processing

After mapping the feature space to the visual parameters space a couple of post-processing techniques are necessary to achieve a decent visual experience.

5.4.1. Linear extrapolation of visual parameters

Each feature corresponds to a segment of a track that is approximately one-second-long as described in *Section 2*. If the visual parameters, determined by these features, update every second, the visualization will be static with abrupt changes every second. To fix this problem, we linearly extrapolate the feature value at any given time using the features of the current segment, next segment and the time elapsed. This results in smooth changes.

5.4.2. Exponentially averaged visual parameters

Consecutive segments often have very distinct features which lead to abrupt changes in the visualization even after extrapolating the features. An example is illustrated in *Figure 5.3.1.2.* Averaging out the visual parameters over the last few segments smoothens out these changes. We have experimented with exponentially weighted moving averaging of the visual parameters with variable values of the smoothing coefficient. Various configurations yield interesting results. We observed decent results with linear extrapolation enabled and a smoothing coefficient below 0.5.

6. Conclusion

In our study, we have explored methodologies to extract features from audio signals and map them to parameters that drive audio visualizations. We have experimented with features obtained from state-of-the-art deep learning architectures including autoencoders and genre classifiers. Each set of extracted features were statistically analyzed which facilitated the design of numerous techniques for mapping features to visual parameters.

We explored three autoencoder architectures. NET-AE was a basic autoencoder with independent encoder and decoder units, NET-AE-SHARED had the weights of the decoder unit tied to those of the encoder and NET-AE-SKIP had skip connections from the convolutional blocks to the corresponding deconvolutional blocks. It was observed that NET-AE produced the most well-balanced features that exhibited varying statistics across the components, NET-AE-SHARED produced features that exhibited skewed standard deviations while NET-AE-SKIP showed very similar distributions across all the components and turned out to be the least interesting of the three architectures. We also borrowed several state-of-the-art image classification models for feature extraction in the audio domain. These models were trained on the task of musical genre classification and we picked various layers across these models whose activations were considered as the features extracted by the models. The features extracted from all the classifiers showed similar characteristics when analyzed across all the tracks in the test set but a genre level analysis revealed differing feature statistics for the same genre. Across the experiments, it was observed that the performance of the models improved with their complexity with NET-VGG16 performing the best.

Our primary focus was on establishing a number of strategies for mapping the raw extracted features to visual parameters. We started with direct mapping that could only be applied to the features extracted from the autoencoders and progressively moved towards more sophisticated strategies that could be applied to features extracted from all the models. The raw features were not found suitable for mapping

and we devised some scaling methodologies to rescale the audio features to the range of the visual parameters, which we had chosen to be from 0 to 1. We experimented with scaling across features which was crucial for the PCA methodology but in general suppressed some of the components and with scaling by features which performed well in most of the experiments and had the advantage of highlighting the variations in each of the components. Moreover, we employed both Minmax and Standard Normalization, the latter being crucial for scaling up the standard deviation of the encodings and thereby making dynamic visualizations feasible.

Our experiments with deterministic mapping on VIZ-BAR showed that audio segments are associated with unique shapes which change throughout the track, producing dynamic visualizations. To solve the drawback of under-expressed visual components, we introduced random mapping, using which we observed interesting visual patterns. To prioritize the expression of the visual elements, we employed PCA and fine-tuned our mapping to produce more variance in the important visual components. We experimented with an approach, using K-Means Clustering, that explored the similarity between encodings of audio segments and allowed us to make audio visualizations that appear different across different tracks. Finally, we combined the methods of PCA and K-Means and created visualizations that are not only different across tracks but also allow prioritization of expression of visual elements. We conducted the mapping experiments III to V with features from all the architectures. The visualizations obtained when using the autoencoders were found to be dynamic with a lot of variation within the same track but they were similar in overall appearance across the tracks. The visualization produced by the genre classifiers were clearly different across different genres, which was further emphasized by employing K-Means Clustering.

We have developed two simple visualizations that have facilitated our analysis and discussed their merits and shortcomings. All our experiments have involved extensive statistical analysis at track, genre and dataset level as well as a visualization on both VIZ-BAR and VIZ-REAL. Our code has been made available online (*https://github.com/rbiswas143/deep-audioviz-experiments*) for the benefit of the community where one can even find a web playground demonstrating our experiments.

This has been a fundamental analysis of the application of deep learning systems to the domain of audio visualizations. In the future, we intend to improve our findings by researching on new feature mapping methodologies, experimenting with other supervised and unsupervised deep learning architectures, as well as improving the existing methodologies. A non-linear clustering strategy might work better than K-Means on extracted audio features. Instead of genre classification, we can use other supervised learning methods, like mood classification, extract mood-specific features and use our methodologies to design visualizations that capture the mood of a song. Moreover, though the appearance of the visualization of a song varies across our methodologies, we do not yet have the necessary control to achieve a pre-defined visual experience. We have introduced a novel approach to creating audio visualizations and, through our meticulously designed experiments and deterministic results, have laid a concrete foundation for future research in the realm of audio visualizations.

References

1. Robyn Taylor, Pierre Boulanger, and Daniel Torres. Real-time Music Visualizations using Responsive Imagery.

2. P. Ciuha, B. Klemenc and F. Solina Visualization of Concurrent Tones in Music with Colours. Univ. of Ljubljana, Slovenia.

3. S. Dixon, W. Goebl, and G. Widmer. The performance worm: Real time visualisation based on langner's representation. In M. Nordahl, editor, Proceedings of the 2002 International Computer Music Conference, pages 361{364, San Francisco, CA, 2002. International Computer Music Association.

4. S. Sigtia and S. Dixon. Improved music feature learning with deep neural networks. In Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2014.

5. J. Nam, J. Herrera, and K. Lee. A Deep Bag-of-Features Model for Music Auto-Tagging. Eprint arXiv: 1508.04999. 2015.

6. J. Schluter. Unsupervised Audio Feature Extraction for Music Similarity Estimation. Technische Universit at Munchen, Fakultat fur Informatik.

7. E.J. Humphrey, J.P. Bello, Y. LeCun. Feature Learning and Deep Architectures: New Directions for Music Informatics. Journal of Intelligent Information Systems 41 (3), 461-481.

8. Kahng, M., Andrews, P. Y., Kalro, A., & Chau, D. H. (2018). ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models. IEEE Transactions on Visualization and Computer Graphics, 24(1), 88-97. doi:10.1109/tvcg.2017.2744718

9. Gallagher M Downs T 1997 Visualisation of learning in neural networks using principal component analysis Proceedings of International Conference on Computational Intelligence and Multimedia Applications, Gold Coast, Australia, 10–12 February 1997, edited by B. Verma and X. Yao, pp. 327–331

10. https://bgodefroyfr.github.io/Deep-Audio-Visualization/web-app

11. N. Takahashi, M. Gygli, and L. V. Gool, “AEnet: Learning deep audio features for video analysis,” 2017, arXiv: 1701.00599.

12. Ingo Mierswa , Katharina Morik, Automatic Feature Extraction for Classifying Audio Data, Machine Learning, v.58 n.2-3, p.127-149, February 2005 [doi>10.1007/s10994-005-5824-7]

13. K. Umapathy, S. Krishnan and R. K. Rao, "Audio Signal Feature Extraction and Classification Using Local Discriminant Bases," in *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1236-1246, May 2007.

14. Foote, J. (2018). Visualizing music and audio using self-similarity.

15. Y. Chung, C. Wu, C. Shen, H. Lee, L. Lee, "Audio Word2Vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder", Proc. Interspeech, pp. 410-415, 2016.

16. Hui-Hui Wang, Jia-Ming Liu, Mingyu You and Guo-Zheng Li, "Audio signals encoding for cough classification using convolutional neural networks: A comparative study," *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Washington, DC, 2015, pp. 442-445.

17. S. Dieleman, B. Schrauwen, "End-to-end learning for music audio", Proc. IEEE Int. Conf Acoust. Speech Signal Process, pp. 6964-6968, 2014.

18. L. Wyse, “Audio spectrogram representations for processing with convolutional neural

networks," arXiv: 1706.09559, 2017.

19. Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. InNIPS, pp. 1106–1114, 2012.

20. B. K. Baniya, J. Lee and Z. N. Li, "Audio feature reduction and analysis for automatic music genre classification," *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, San Diego, CA, 2014, pp. 457-462.

21. Annesi, Paolo, Roberto Basili, Raffaele Gitto, Alessandro Moschitti, and Riccardo Petitti. "Audio feature engineering for automatic music genre classification." In *Large Scale Semantic Access to Content (Text, Image, Video, and Sound)*, pp. 702-711. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 2007.

22. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., & Rabinovich, A. (2014). Going deeper with convolutions. Technical report.

23. Wang, S., Sun, J., Phillips, P., Zhao, G. and Zhang, Y. (2017). Polarimetric synthetic aperture radar image segmentation by the convolutional neural network using graphical processing units. Journal of Real-Time Image Processing.

24. Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In NIPS 2017 Autodiff Workshop: The Future of Gradient-based Machine Learning Software and Techniques, Long Beach, CA, US, December 9, 2017, 2017.

25. K. Benzi, M. Defferrard, P. Vandergheynst, and X. Bresson, "Fma: A dataset for music analysis," arXiv preprint arXiv:1612.01840, 2016

26. Benjamin Murauer and Günther Specht. 2018. Detecting Music Genre Using Extreme Gradient Boosting. In WWW

27. Congote, John, Alvaro Segura, Luis Kabongo, Aitor Moreno, Jorge Posada, and Oscar Ruiz. "Interactive visualization of volumetric data with webgl in real-time." In *Proceedings of the 16th International Conference on 3D Web Technology*, pp. 137-146. ACM, 2011.

28. Jaehun Kim, Minz Won, Xavier Serra, and Cynthia C. S. Liem. 2018. Transfer Learning of Artist Group Factors to Musical Genre Classification. In WWW

29. S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, et al. , "CNN architectures for large-scale audio classification," arXiv preprint arXiv: 1609.09430, 2016.

30. S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. arXiv preprint arXiv:1605.05396, 2016.

31. Santiago Pascual, Antonio Bonafonte, and Joan Serrà. 2017. SEGAN: Speech Enhancement Generative Adversarial Network arXiv: 1703.09452.

32. David Ha and Douglas Eck. A neural representation of sketch drawings. CoRR, 2017.

33. X. Mao, C. Shen, and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In NIPS 2016.

34. I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton. On the importance of initialization and momentum in deep learning. In ICML, volume 28 of JMLR Proceedings, pages 1139–1147. JMLR.org, 2013.

35. D. Scherer, A. Muller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in Proc. of the Intl. Conf. on Artificial Neural Networks, 2010, pp. 92–101

36. D. J. Im, M. I. D. Belghazi, and R. Memisevic, Conservativeness of untied auto-encoders, CoRR, abs/1506.07643 (2015).

37. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

38. Zhang, Yu-Dong & Dong, Zhengchao & Chen, Xianqing & Jia, Wenjuan & Du, Sidan & Muhammad, Khan & Wang, Shuihua. (2017). Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation. Multimedia Tools and Applications. 1-20. 10.1007/s11042-017-5243-3.

39. M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. CoRR, abs/1301.3557, 2013.

40. S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015

41. N. Takahashi, M. Gygli, L. Van Gool, Aenet: Learning deep audio features for video analysis, 2017.