Learning to Recognize Musical Genre from Audio

Wang Yuanjing

***Abstract*—In this article, we investigate a new 1D convolutional neural network (CNN) - SwishNet. It is a lightweight architecture and runs based on MFCC functionality, suitable for adding to the front-end of audio processing pipelines. This article demonstrates that it can improve the performance of our network by extracting knowledge from 2D CNN pre trained on ImageNet. We investigated the performance of our network on the GTZAN corpus, which is a publicly available comprehensive collection of noise, music, and speech samples suitable for deep learning. In speech/non speech recognition tasks, the proposed network achieved high overall accuracy in clip classification and frame by frame segmentation tasks. To verify the robustness of our model, we trained it on GTZAN and found that its accuracy was very high with almost no fine-tuning. We also demonstrated that our model is fast on both CPU and GPU, consumes very little memory, and is suitable for implementation in embedded systems.**

***Index Terms*—Audio Segmentation, Convolutional Neural Network, Voice Activity Detection, Multimedia Indexing.**

I. INTRODUCTION

Convolutional neural networks (CNNs) have been frequently used in various music classification tasks, such as genre classification, music labels, and prediction of recommended features for potential users. In the field of computer vision, since the breakthrough of CNNs, deep learning methods have become the standard. These methods come from feature learning, and each parameter is learned to reduce the loss function. CNNs can be extracted through convolutional kernels to assume features at different levels. During supervised training, learn hierarchical features to achieve a given task. CNN based methods have also been used for music information retrieval. They use 2D convolutions to perform for music-noise segmentation[1] and chord recognition while 1D (times-axis) convolutions are performed for automatic tagging in. When the target shape is known, the mechanism for learning filters in cellular neural networks is relatively clear. What remains to be unraveled is how cellular neural networks accomplish tasks such as emotion recognition or type classification. These tasks are related to subjective perceptual impressions, and the relationship between subjective perceptual impressions and acoustic characteristics, as well as whether neural network models can learn the correlation and optimal representation of sound that helps improve the performance of these tasks, is an unresolved issue. As a result, researchers currently lack on understanding of what is learnt by CNNs when CNNs are used in those tasks, even if it show state-of-the-art performance[2]. One effective way to examine CNNs was introduced in[3], which visualizes deeper features through a method called deconvolution. Deconvolution and unspooling layers allow people to see which part of the input image each filter focuses on. However, it does not provide a relevant explanation of CNNs for music, as the behavior of CNNs are task specific and data dependent. Unlike visual image recognition tasks where image contours play an important role, spectrograms are mainly composed of continuous and smooth gradients. There is not only local correlation, but also global correlation, such as harmonic structures and rhythmic events. In this article, we introduce the process and results of deconvolution and naturalization to expand our understanding of CNNs in music. We use the public datasets GTZAN, which includes 10 types of music such as blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock. We not only apply deconvolution to spectrograms, but also propose auditioning trained filters to achieve time-domain reconstruction. In Section two, the background of CNNs and deconvolution was explained. Section three introduces the proposed listening method. The experimental results are discussed in Section four. Conclusion can be found in Section five.

II. BACKGROUND

A. Visualization of CNNs

Multiple convolutional layers are located at the core of CNN. The output of a layer which is called a feature map is fed back into the input of the next layer. This stacking structure enables each layer to learn filters in different levels of the hierarchy. The sub sampling layer is also an important part of CNNs. It adjusts the size of the feature maps and utilizes the network see the data at different scales. Subsampling is typically implemented by a max-pooling layers, which adds positional invariants. The behavior of CNN is uncertain because the maximum pooling operation varies depending on the input. That is why analyzing the learning weights of convolutional layers cannot provide a satisfactory explanation. One way to understand a CNN is to visualize the features given different inputs. This article introduces the visualization of CNNs was introduced in[3], which showed how high-level features (postures/objects) are constructed by combining low-level features (lines/curves), as illustrated in Figure 1. In the figure, the shapes represented that features are evolving. In the first layers, each feature simply responds to lines with different directions. By combining them, the features in the second and third layers can capture certain shapes - a circle, textures, honeycombs, etc. During this forward path, the features not only become more complex but also allow slight variances, and that is how the similar but different faces of dogs can be recognized by the same feature in Layer 5 in Figure 1. Finally, the features in the last layer successfully captured the contours of the target objects such as cars, dogs, and human faces. The visualization of CNNs helps not only to be used to understand the process internal the black box model, but also to determine the hyper parameters of the networks. For example, the redundancy or inadequacy of the

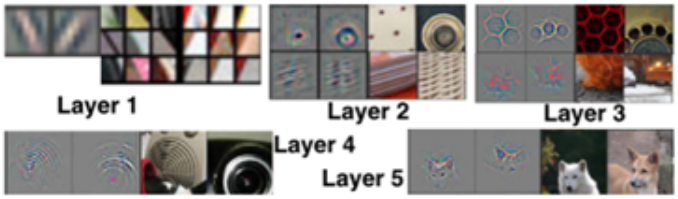

Fig. 1. Deconvolution results of CNNs trained for image classification. In each layer, the responses of the filters are shown on the left with gray backgrounds and the corresponding parts from images are shown on the right. Image is courtesy of[3].

capacity of the networks, which is limited by hyper parameters such as the number of layers and filters, can be judged by inspecting the learnt filters. Network visualization provides useful information since fine tuning hyper parameters is a crucial factor in obtaining cutting-edge performance.

B. Datasets

We validate the effectiveness of the proposed method on different sizes of datasets for GTZAN. The GTZAN datasets contains 1000 music clips, each with a length of 30 seconds. There are 10 different genre categories. The sampling rate for music clips is 22.5kHz. The datasets does not have a formal separation of training and validation, so we used 20

C. Audition Of Learnt Filters

Much research of CNNs on audio signal uses 2D time-frequency representations as input data. And various types of representations have been used including Short Time Fourier transform (STFT), Mel spectrogram and Constant Q transform (CQT). CNNs have demonstrated state-of-the-art performance on many tasks such as music structure segmentation[4], music labeling[5], and music classification. Those performances empirically indicate that CNNs are powerful models to solve many music-related problems. From many perspectives, the gauge diagram is very consistent with the assumptions of CNNs. They are locally correlated and typically require shift/-translation invariances, and the output labels may depend on local, sparse features[6]. Although we can obtain spectrograms through deconvolution, deconvolution spectrograms do not necessarily facilitate an intuitive explanation. This is because seeing a spectrogram does not necessarily provide clear intuition that is comparable to observing an image. To address this issue, we suggest to reconstructing audio signals from deconvoluted spectrograms, which is called auditory processing. This requires an additional stage of inverse transformation of a deconvoluting spectrograms. The phase information is provided by the phase of the original time frequency representations, following the generic approach in spectrogram-based sound source separation algorithms[7]. Therefore, it is recommended to use STFT as it allows us to easily obtain time-domain signals. The pseudocode of the auralisation is described in Listing 1. Line 1 indicates that we have a convolutional neural network that is trained for a target task. In line 2-4, an STFT representation of a music signal is provided. Line 5 computes the weights of the neural networks with the input STFT representation and the result is used during the deconvolution of the filters in line 6 (for more details). Line 7-9 shows that the deconvolved filters can be converted into time-domain signals by applying the phase information and inverse STFT.

```
cnn_model = train_CNNs (*args) # model
src = load(wavfile)
SRC= stft(src)
aSRC , pSRC = SRC.mag , SRC.phase
weights = unpool_info(cnn_model , aSRC)
deconved_imgs = deconv(weights , aSRC)
for img in deconved_imgs:
signal = inverse_stft(img * pSRC)
wav_write(signal)
```

Listing 1. A pseudo-code of audition procedure

D. Description of the Corpora

The MUSAN corpus[8] is compiled from Creative Commons and US Public Domain sources. It is freely available (at OpenSLR[9]) and redistributable. It consists of approximately 109 hours of audio in three categories. It has 660 music files of different genres, 930 noise files and 426 voice files (in 12 different languages) from LibriVox recordings and US Govt. hearings. All the audio files are in 16 kHz WAV format. The GTZAN[10] Music and Voice Collection contains 60 music files from different genres and 60 voice files from different sources. Each of the tracks is 30 seconds long. All files are in 22050Hz WAV format.

III. EXPERIMENTAL SETUP

We implemented a CNN-based genre classification algorithm using a dataset obtained from GTZAN Music 2 and based on Keras and Theano. All audio signal processing was done using librosa. Three genres (ballad, dance and hiphop) were classified using 8,000 songs in total. In order to maximize the utilization of data, 10 clips of 4 seconds were extracted for each song, generating 80,000 data samples by STFT. STFT is computed with a 512 point windowed Fast Fourier Transform with a jump size of 50The CNN architecture consists of 5 convolutional layers of 64 feature maps and 3-by-3 convolution kernels, with a maximum pooling size and stride of (2,2), and two fully connected layers as illustrated in the figure 2. Dropout(0.5) is applied to the all convolutional and fully-connected layers to increases generalization ability. This system showed 75 It is noteworthy that although a homogeneous size of convolutional kernels (3×3) are used, the effective coverages are increasing due to the subsampling, as in the table 1.

A. Evaluation Strategy

The files in the MUSAN corpus were randomly divided into three groups. 65%, 10% and 25% of the all files in each category (i.e. speech, music and noise) were allocated to training, validation and testing sets respectively. The same

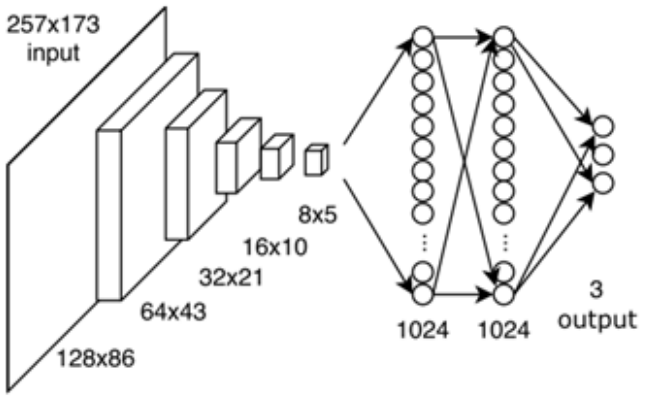

Fig. 2. Block diagram of the trained CNNs.

TABLE I
THE EFFECTIVE SIZES OF CONVOLUTIONAL KERNELS

Layer	Convolution	Effective width	Effective height
1	3×3	93ms	86Hz
2	3×3	162ms	151Hz
3	3×3	302ms	280Hz
4	3×3	580ms	538Hz
5	3×3	1137ms	270Hz

preprocessing and data preparation steps were followed for all three sets. Silent parts of the files were removed using power threshold. Balance the volume through loudness was equalized with a window length of 250ms. For speech/music classification on the GTZAN corpus, background noise was removed using the Log MMSE[11] method. For evaluation on GTZAN, we selected 25For segmentation, we used the same networks trained for classification purposes. We have created an artificial datasets consisting of 500 files of randomly chosen lengths ranging 20 seconds to 120 seconds; by randomly concatenating mute, noise, speech and music signals from the test set with average lengths are 0.5s, 5s, 10s and 12s respectively. Some of the transitions were abrupt, while others were separated by silence. Instead of zeroing out the silenced parts, we used natural silence portions clipped from within the dataset. We apply neural networks on a windowed signal around a particular frame to gather contextual information. So, the decision of a frame depends not only on that frame but also on other frames within the contextual window. Window length was varied between 0.5s and 2.0s in order to evaluate its impact on the segmentation performance. Then, median filtering (filter length of 200 frames) was applied on the frame-wise predictions (probabilities) which slightly improved final accuracy. Accuracy was measured frame-wise against true annotations. The silent frames were excluded from the evaluation. As a baseline, we compared our results with the widely used Gaussian Mixture Model (GMM) and Fully-Connected Neural Network (FNN). They are all both trained in a frame-wise manner, and MFCC with first and second order deltas were used as input features. The final decision on division classification was taken by majority voting. We used three GMMs for these three categories, each with 256 components with diagonal covariance. As for the FNN, we used a 4-layered Self-Normalizing Neural Network (SNN)[12] with Alpha dropout (which is the current state-of-the-art for FNNs). The design choices of these models were focused on improving classification performance rather than speed.

B. Optimization Strategy

We used the Adam optimizer[13] to train all our networks. The gating activations used in SwishNet seemed to cause vanishing gradient problems to some extent and training was relatively slow. Therefore, in order to accelerate the training we used cosine annealing and warm restarts as described in[14]. The batch size was tuned for fastest training. For smaller segments, the number of training samples increased, so we increased the batch size proportionately to keep the number of gradient updates per epoch same. We trained SwishNet for 120 epochs. MobileNet trained really fast when initialized with ImageNet weights and reached convergence within very few (8-10) periods. Train GMM until convergence and train SNN until the validation loss stabilizes. In order to fine tune GTZAN, we used a network trained on MUSAN and trained it on GTZAN training data at a low learning rate for 50 periods.

C. Distillation Strategy

For distillation, first train the pre trained MobileNet (on ImageNet) on the training data. Then, we use the output logits of MobileNet to extract SwishNet (on the same training data). Validation data is used to select the optimal temperature for distillation. We formed a weighted loss from 90

IV. RESULTS AND DISCUSSION

A. Classification Results

Both SwishNet and MobileNet can handle variable input lengths. Therefore, we have kept the network structures fixed for all tests. Table I compares the sizes prediction speeds of the three networks as well as the baseline models. To simulate comparison of real-time computational cost, we fixed batch size to 1 for all models and compared their speed on CPU and GPU. Our CPU was Core i7 8700K, a 6 core processor and our GPU was NVIDIA GTX 1080ti and the networks were implemented in Tensorflow. We witness that SwishNet-slim is actually faster on CPU than on GPU and the width of SwishNet for shorter clip lengths. This is due to the fact that, when parallelization requirement is minimal, clock speed plays the major role in computational speed. Also, latency is introduced when data is passed from CPU memory to GPU memory. SwishNet slim has a computational speed on CPU comparable to SNN while being much smaller in size. The fully-connected SNN enjoys an advantage in speed performance due to its lower depth and simplistic computational requirement. But, as we will show presently, the performance gain of SwishNet justifies the use of a deep convolutional architecture. SwishNet is 4 to 8 times faster than MobileNet on CPU, due to its low parallelization requirement. For all clip lengths, the SwishNet model is much faster on CPU than traditional GMM methods. Among all the models considered,

SwishNet has the smallest number of parameters and the smallest weight file size. The scale of SwishNet slim and SwishNet wide is approximately 2.5The classification results are presented in Table 2, 3 and 4. We see that SwishNet performs better than the conventional GMM approach and also the fully connected SNN for all clip lengths. The performance of SwishNet is only masked by MobileNet which is a much larger network, pre-trained on a huge datasets such as ImageNet. From the confusion matrices, we can see that recall for noise is low for all models, especially for smaller clip lengths, even though the speech/non-speech discrimination accuracy is high for all clip lengths. This result indicates that for smaller contextual windows, it is difficult to distinguish between noise and music. This is because, some of the noises, such as ringtones or phones ringing are slightly musical in nature and short segments of music may sound like noise. So, a longer context is necessary for discriminating noise from music. Also, we see that although SNN can reasonably distinguish speech from non-speech, its performance in discriminating between noise and music is not at all satisfactory. This is because of the inability of an FNN to properly utilize contextual information, which is essential for distinguishing noise from music. This result justifies our use of a deep convolutional architecture which ensures proper utilization of contextual information. When using ImageNet weights instead of random weights to initialize MobileNet, we see a significant improvement in its classification performance. This result validates our hypothesis of transfer learning from ImageNet. When knowledge is extracted from MobileNet to SwishNet, we see a more significant improvement in performance across a wider range of versions. For shorter clip lengths, SwishNet slim cannot benefit greatly from distillation due to its low capacity. The results of undistilled SwishNet dilute and distilled SwishNet wide are presented in the following sections.

B. Classification Results on GTZAN

Table 2 presents the music/speech classification results of 2-second long clips exported from the GTZAN corpus. From the results, we can see that even with only fine-tuning 25% of the new datasets, SwishNet can still achieve very good results. This is impressive because GTZAN voice data has severe background noise, but the MUSAN corpus used to train the network has almost no background noise. Although there are music genres in the test data that have not even been trained by the network, the recall rate of music is still very high. This indicates the robustness of the network to input data.

TABLE II
CLASSIFICATION RESULTS FOR 2S LONG CLIPS DERIVED FROM GTZAN.

Network	Overall Accuracy	Speech Recall	Average Recall	Music Recall	F1 Recall
SwishNet-slim	98.00%	98.17%	97.84%	98.01%	97.90%
SwishNet-wide	98.26%	98.33%	98.20%	98.27%	98.17%
MobileNet	98.83%	98.28%	99.45%	98.86%	98.77%

C. Discussions and Trade-offs

SwishNet-slim is an extremely fast and lightweight network. It can be easily integrated into the front end of a real-time processing pipeline while introducing very small latency. The SwishNet-wide has slightly speed and size slightly in favour of higher performance and robustness which increases with distillation. The graceful performance improvement with network size also proves the scalability of our architecture and provides a very balanced trade-off between speed and performance. The performance of SwishNet is much better than frame methods such as GMM or SNN, as it can pick up and retain information in long contextual windows. However, compared to pre trained MobileNet, its performance is much lower. Without pre training, the performance of MobileNet is actually similar to that of SwishNet. However, when using ImageNet for transfer learning, it can achieve excellent accuracy scores and robustness. while comparing our network to pretrained MobileNet, we should keep in mind that, MobileNet is a much more heavyweight and slower network compared to ours. Also, it enjoys the advantage of being pretrained on a massive datasets i.e. ImageNet. While it may be practical to use MobileNet in some off-line indexing tasks, the increased performance may not be justifiable when speed and efficiency is the priority and also when computational resources are limited. In this case, SwishNet would be a much better choice in those scenarios. It is worth noting that we can extract some knowledge from MobileNet (initialized with ImageNet weights) to the SwishNet range. It can be assumed that we can indirectly transfer knowledge from the natural image database ImageNet to SwishNet, which is an interesting discovery.

V. CONLUSIONS

We introduced audition of CNNs, which is an extension of CNNs visualization. We proposed a novel 1D convolutional neural network – SwishNet designed for efficient classification and segmentation of audio signals into three major categories Speech, Music and Noise. We also use this method to make the music classification. This is done by inverse-transformation of deconvolved spectrograms to obtain time-domain audio signals. Listening to the audio signal enables researchers to understand the mechanism of CNNs that are trained with audio signals. In the experiments, we trained a 5-layer CNN to classify genres. Selected learnt features are reported with interpretations from musical and music information aspects. The comparison of correlations of feature responses showed how the features evolve and become more invariant to the chord and instrument variations. Further research will include computational analysis of learnt features.