Audio Classification using HPSS Features and Data Augmentation with Convolutional Neural Network

Lejun Min
Computer Science, Zhiyuan College
Shanghai Jiao Tong University
519030910326
`aik2mlj@sjtu.edu.cn`

Abstract

This paper presents a proposed model using pretrained imagenet model EfficientNet based on convolutional neural network model and multi-channel audio feature extracted by harmonic percussive source separation (HPSS).

1 Introduction

This paper is aimed at proposing a model[1] to classify different kinds of sounds. It heavily refers to this work[2]. We use ESC-50[3] as our training and validation data set. It is a small-sized audio data set with 2000 5-second 44.1khz-sampled audio files categorized into 50 kinds. In the past few years, the mainstream method of audio classification transforms from dealing with audio features like MFCCs (Mel-frequency cepstral coefficients) to focusing on convolutional neural network structure. The latter proves to be more adaptive and performs better in most cases. But feature extraction still remains to be explored further.

Google has prompted image classification a lot by EfficientNet model which surpasses other existing models. In this work, We try to extract multiple features from each audio and find a way to route the input into EfficientNet.

2 Feature Extraction

The audio files in the data set is in WAV form. We use *librosa* package to manipulate audio information and extract features that we want.

2.1 Log-Mel-Spectrogram

In order to convert an audio file to a accessible matrix to put into the train model, we need to transform it into a spetrogram that encodes its frequency and time information. Given many kinds of spectrograms, the log-mel-spectrogram stands out for its human-friendly features. It compresses the raw audio data to dB and frequency according to time. Compared to normal spetrogram, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal spectrum[4].

The 5-second 44.1hhz-sampled raw audio is transformed into numpy ndarray with shape $(128, 216)$. It contains both time and frequency information, and is convenient for further manipulation. The used parameters are Mel-filter 128, FFT size 1024, Hop size 512.

[1]See codes at https://github.com/aik2mlj/Audiobia
[2]Jaehun Kim, Urban Sound tagging using multi-channel audio feature with convolutional neural networks
[3]ESC-50: Dataset for Environmental Sound Classification, https://github.com/karolpiczak/ESC-50
[4]Mel-frequency cepstrum, wikipedia, https://en.wikipedia.org/wiki/Mel-frequency_cepstrum

Table 1: Categories of ESC-50

Animals	Natural soundscapes	Human sounds	Domestic sounds	Urban noises
Dog	Rain	Crying baby	Door knock	Helicopter
Rooster	Sea waves	Sneezing	Mouse click	Chainsaw
Pig	Crackling fire	Clapping	Keyboard typing	Siren
Cow	Crickets	Breathing	Door, wood creaks	Car horn
Frog	Chirping birds	Coughing	Can opening	Engine
Cat	Water drops	Footsteps	Washing machine	Train
Hen	Wind	Laughing	Vacuum cleaner	Church bells
Insects (flying)	Pouring water	Brushing teeth	Clock alarm	Airplane
Sheep	Toilet flush	Snoring	Clock tick	Fireworks
Crow	Thunderstorm	Drinking, sipping	Glass breaking	Hand saw

2.2 HPSS

On a very coarse level, sounds can be divided into either harmonic sounds or percussive sounds. Harmonic sounds are perceived to have a certain pitch, while percussive sounds are more likely to have a clicking or a punching pattern without a specific pitch. The goal of harmonic percussive source separation (HPSS) is to separate the harmonic and the percussive parts from the original audio source to be

$$D = D_{harmonic} + D_{percussive}$$

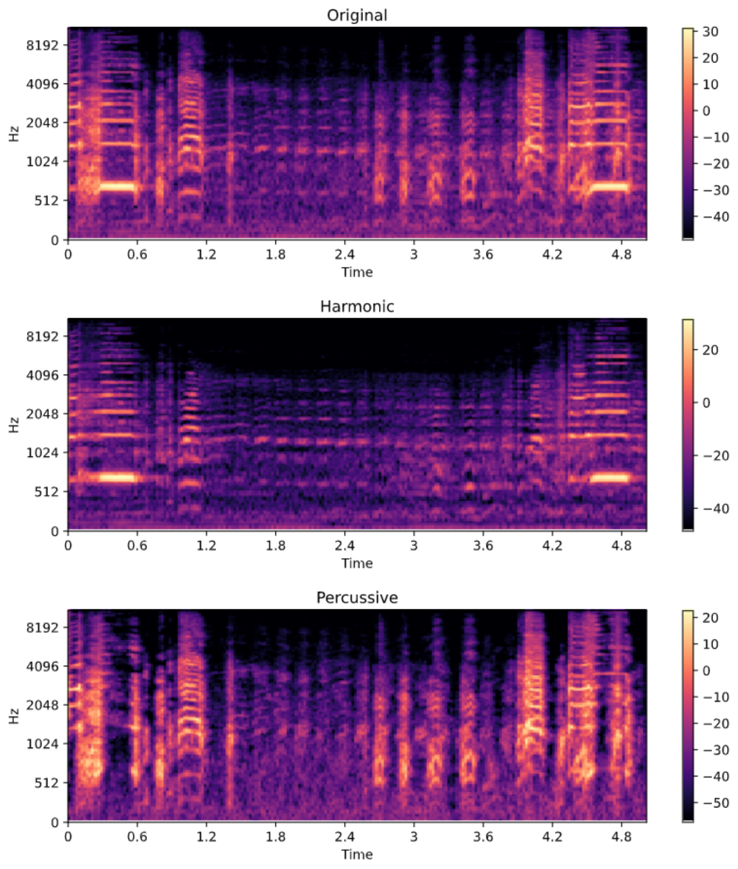


Figure 1: Source mel-spectrogram and its harmonic and percussive mel-spectrogram

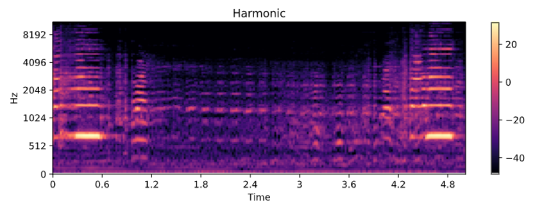
Figture 1 shows that in the form of mel-spetrogram, the harmonic part has more horizontal patterns and the percussive part has more vertical patterns. The main idea is to first apply short-time Fourier transform (STFT) to the audio source, and then apply some filter to the power spectrogram that enchances horizontal structures and suppresses vertical structures to extract horizontal patterns. Extracting vertical patterns is the same.

After HPSS and calculating log-mel-spectrogram, the extracted feature shape is $(128, 431, 3)$, where the third axis indicates the three channels (original, harmonic, percussive) which is very similar to the three channels of an RGB image. This is essential for routing into the EfficientNet model since the EfficientNet model is originally proposed for classifying 3-channel RGB images. In real implementation, the extracted feature should be dimention-expanded to fit the input shape.

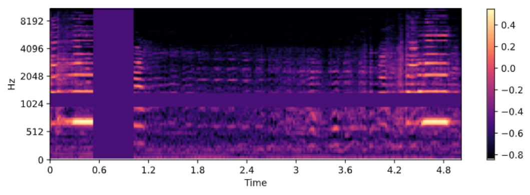
2.3 Spectrogram Data Augmentation

Considering that the size is small (2000 audio samples), we try to apply data augmentation to enlarge the training set. Raw audio data augmentation focuses mainly on raw audio data manipulation, including time shifting, pitch shifting, speed changing, etc. However, in order to decrease the training time, we separate feature extraction from the training part. The data loaded when training is already log-mel-spectrogram data that is pre-saved in feature extraction, so applying raw audio data augmentation is not practical.

We use a spectrogram data augmentation method called SpecAugment[5]. It ramdomly put some horizontal and vertical mask bars on the spectrogram to create an augmented spectrogram data.



(a) Harmonic Data



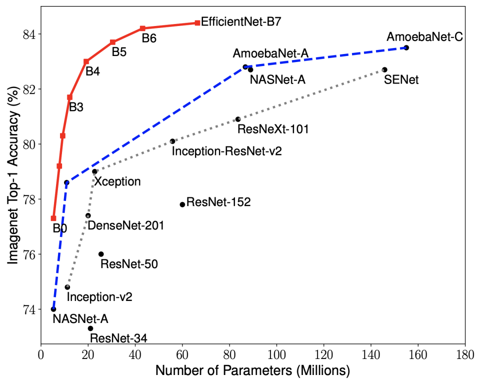
(b) Augmented Harmonic Data

Figure 2: SpecAugment applied on a Harmonic Spectrogram

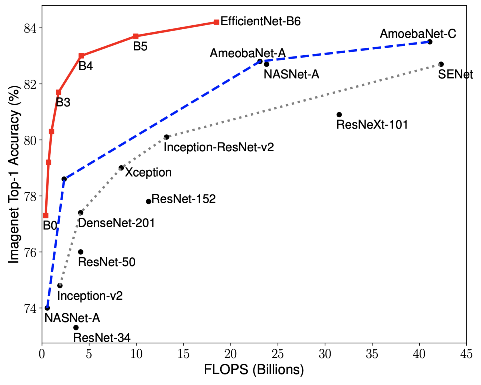
3 Network Architecture

We use a highly user-friendly package *keras* with *tensorflow* backend to construct our model.

3.1 EfficientNet



(a) Model Size vs. ImageNet Accuracy



(b) FLOPS vs. ImageNet Accuracy

Figure 3: EfficientNet: performance on ImageNet

[5]Daniel S. Park, William Chan, al. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition, arXiv preprint arXiv:1904.08779, 2019

Figure 3[6] shows the surpassing performance EfficientNet has on ImageNet. Due to the limitation of GPU power, we use *EfficientNetB4* preloaded with "noisy-student" weights as the base training layer. The performance will be even better for bigger EfficientNet model.

3.2 Dropout

Deep neural architectures have a natural tendency to overfitting. Dropout is a simple yet highly effective method to tackle this problem. In each training iteration, each unit is randomly removed with a predefined probability. This method can prevent the network from learning spurious dependencies, along with fast performance.

In our proposed model, we connect a dropout layer with 0.5 propability to the base EfficientNet layer.

Table 2: Summary of the preposed model

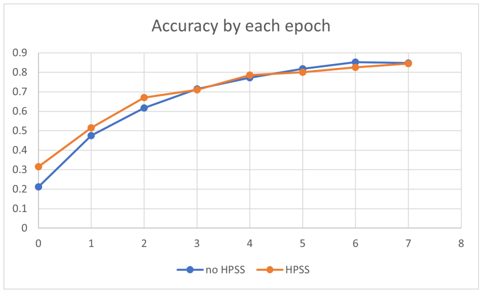
Layer (type)	Output Shape	Param
input (InputLayer)	[(None, 1, 128, 431, 3)]	0
time_distributed (TimeDistributed)	(None, 1, 1792)	17673816
time_distributed_1 (TimeDistributed)	(None, 1, 1792)	0
time_distributed_2 (TimeDistributed)	(None, 1, 50)	89650
output (AutoPool1D)	(None, 50)	50

4 Experiments and Results

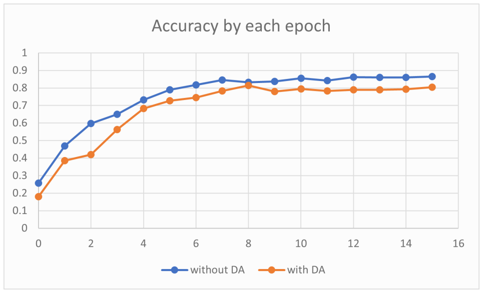
We want to find out the impact the 3-channel feacture extraction and data augmentation on the final prediction accuracy. The parameters are batch size 8, epochs 8 / 16, learning rate 10^{-4}, l2 regularization 10^{-4} and Adam optimizer.

Table 3: Evaluation result on ESC-50 (Accuracy %)

Model	Accuracy
Proposed Model without HPSS & SpecAugment	84.8
Proposed Model without SpecAugment	86.5
Proposed Model	80.3



(a) With / without HPSS



(b) With / without SpecAugment

Figure 4: Training histories of varients of proposed model

5 Conclusion

It shows that compared to single raw audio spectrogram, 3-channel extracted features performs slightly better. Its accuracy increases more quickly and smoothly.

[6]TAN, Mingxing; LE, Quoc V. Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946, 2019.

However, applying SpecAugment seems to decrease the accuracy during 16 epochs. Due to the limited time, we haven't found out what causes the degradation. One possible reason is that the model generated too much noise into the augmented data so that the network learned some wrong patterns. Later work on choosing parameters and polishing model structure is needed.