General-Purpose Audio Tagging System Using Convolutional Neural Networks

Zongming Liu (B00784897)

August 14, 2018

Abstract

In this project, convolutional neural networks (CNN) are applied for the Freesound General-Purpose Audio Tagging Challenge [1] hosted on Kaggle. This challenge provides a labeled training set of around 9,500 sound samples unequally distributed among 41 categories. We propose three simple CNN models for audio tagging. Our model were evaluated on the unlabeled test set provided by the challenge, which consists of around 1,600 sound samples. The test accuracy can be retrieved from Kaggle after submission. In this approach, log-mel spectrogram feature vectors are extracted from audio waveforms and used as the input to the CNN model, and convolved in 1 dimension.

1 Introduction

The purpose of the audio tagging system is to classify audio samples into one of the 41 labels from Google's AudioSet Ontology [2], such as "Tearing", "Cello", "Gong", and etc. Given the increased number of sound events of very diverse nature nowadays, it is worthy to build a model that can recognize those events. Such models can be used for automatic description of multimedia or acoustic monitoring applications.

2 Related work

In recent years, CNNs have shown promising results on image classification. Due to the massive success of CNNs on image data, CNNs have been applied to audio data as well. In [3], Hershey *et al.* examined various deep CNN models for large-scale audio classification, and obtained good results. In [4], Fonseca *et al.* presented the Task 2 of the DCASE 2018 Challenge, the task that this project aims to complete, with a baseline system. In this project, we trained three shallow CNN models based on the baseline system. In [3, 4], each audio sample is preprocessed and transformed from waveform to log-mel spectrogram, then the log-mel spectrogram is divided into frames of equal lengths. Then, each audio sample would have a number of frames of same shape, which are the inputs to their CNN models. This approach would give more training samples out of one audio sample. But, it would require much longer preprocessing, training, and test time. Also, the relation between each frames in one audio sample is lost. Instead, we used log-mel spectrogram as the inputs to our CNN model with 1-D convolution. In this case, we have to zero pad some of the resulted log-mel spectrograms so that all inputs to our CNN model have the same shape.

3 Data

The dataset provided for this task is Freesound Dataset Kaggle 2018, a reduced subset of FSD [5]. It is an audio dataset containing 18,873 audio files annotated with labels from Google's AudioSet Ontology [2]. All audio samples are provided as uncompressed PCM 16 bit, 44.1 kHz, mono audio files. They are unequally distributed in 41 categories of Google's AudioSet Ontology.

The entire dataset is split into a train set and a test set. In the train set, there are around 9,500 samples unequally distributed among 41 categories. The minimum number of audio samples per category in the train set is 94, and the maximum is 300. The duration of the audio samples ranges from 300ms to 30s. Out of the 9,500 samples from the train set, about 3,700 samples have manually-verified annotations and the rest 5,800 samples have non-verified annotations. In the test set, there are around 1,600 samples with manually-verified annotations. The test set is complemented with around 7,800 padding sounds which are not used for scoring the systems [4].

In this project, we used all samples from the train set to train our model. Though approximately 5,800 samples have non-verified annotations, at least 65-70% of the non-verified annotations per category in the train set are indeed correct, which did not give us any reason to only use the samples with manually-verified annotations.

To train CNN models on audio data, we need to transform the audio data into some form that CNN models work with. log-mel spectrogram has shown good results on being feature vectors for audio data [3, 4]. To convert waveform to log-mel spectrogram, we used LibROSA [6]. We adopted the way in [4] that generates the log-mel spectrogram out of waveform: first, computing spectrogram with a window size of 25ms and a hop size of 10ms; second, mapping the spectrogram to 64 mel bins covering the range 125-7500 Hz; lastly, log-mel spectrogram is computed by applying log(mel spectrogram + 0.001). The resulting log-mel spectrogram has shape [time, 64]. The last step is to make all log-mel spectrograms have the same shape, we zero-padded all log-mel spectrograms whose first dimension is less than 3001, which is the maximum of the first dimension of all log-mel spectrograms. Finally, we transformed all our audio samples into corresponding log-mel spectrograms with a shape of [3001, 64].

During the data preprocessing, we looked at the implementation of transforming audio from waveform to log-mel spectrogram provided in [7].

4 Methods

Our approach to build the audio tagging system is to use a shallow CNN. In our approach, the inputs to the model are log-mel spectrograms, with time being the first dimension. We would like to consider the change of the audio during time, thus, we decide to use 1-D convolutional layers to slide through our log-mel spectrograms from left to right.

We tried three similar models:

Our first model is based on the baseline system provided by the challenge [4]: the input layer is of shape [batch, 3001, 64]; followed by a 1-D convolutional layer with 100 filters, kernel size of 7, strides of 1, 'same' padding strategy, and ReLU as the activation functions; followed by a 1-D max pooling layer with pooling size of 3, strides of 2, and 'same' padding strategy; followed by a 1-D convolutional layer

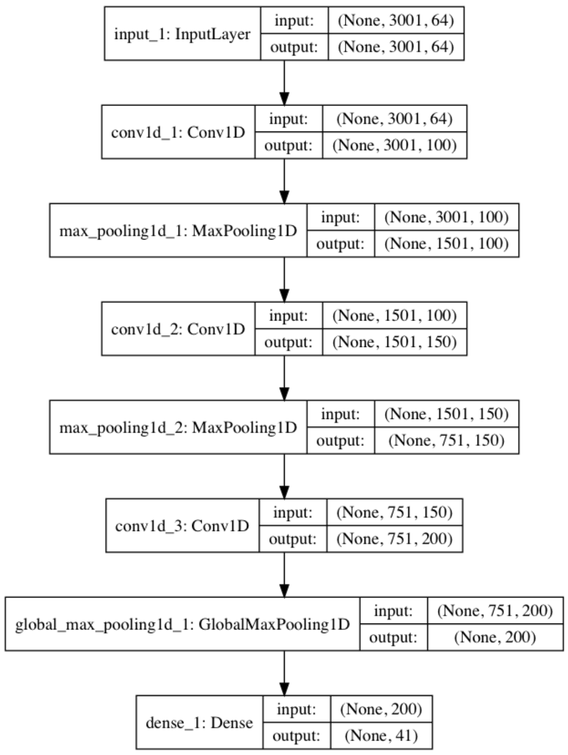

Figure 1: CNN 1D.

with 150 filters, kernel size of 5, strides of 1, 'same' padding strategy, and ReLU as the activation functions; followed by a 1-D max pooling layer with pooling size of 3, strides of 1, 'same' padding strategy; followed by a 1-D convolutional layer with 200 filters, kernel size of 3, strides of 1, 'same' padding strategy, and ReLu as the activation functions; followed by a 1-D Global Max Pooling layer; followed by a Dense layer with softmax as the activations functions, and output a tensor of shape [batch, 41]. This model is called CNN-1D, see its Figure 1.

Our second model is based on the first model - CNN-1D, we added a layer of dropout before the Dense layer. This model is called CNN-1D-Dropout, see its Figure 2.

Our third model is also based on the first model - CNN-1D, however, it adds another Dense layer before the original Dense layer. Also, before both Dense layers, one Dropout layer is added. This model is called CNN-1D-Dense-Output, see its Figure 3

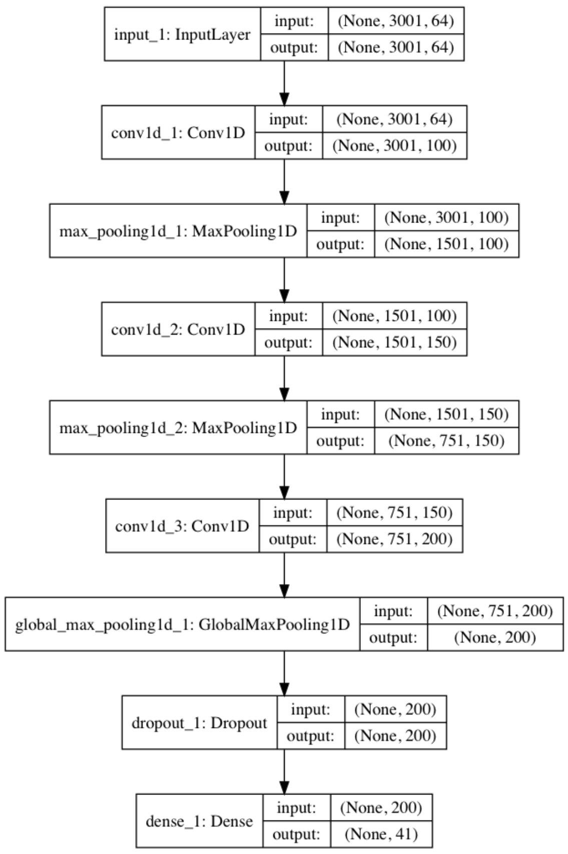

Figure 2: CNN 1D Dropout.

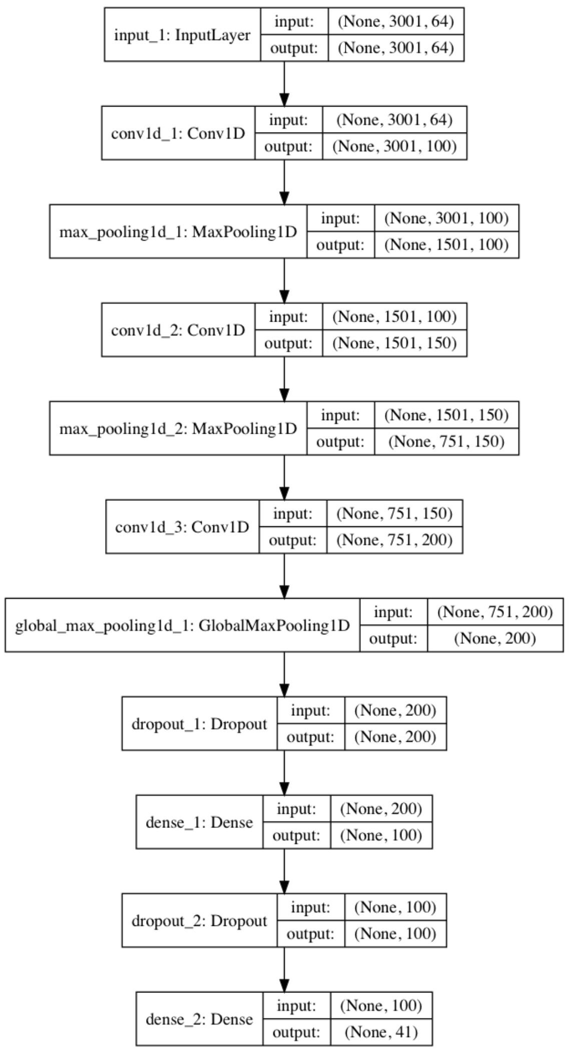

Figure 3: CNN 1D Dense Dropout.

5 Experiments

We trained all three models in 30 epochs, with a batch size of 64. The final training accuracy of these three models are: 84%, 81%, 76% respectively. However, the test accuracy of these three models are: 75%, 80%, 77%. As we can see that the first model showed some level of overfitting, which is reasonable, given we did not introduce any regularization methods to it. The second model and the third model did not show evidence of overfitting, and gave good results based on the training time, though the second model gave better result than the third model. See the Table below that records the comparison of the test accuracy of the baseline model and our models.

Comparison of baseline model and our models

Model	Training time	Test accuracy	Train accuracy
DCASE Baseline	12hrs	70%	N/A
CNN-1D	2hrs	75%	84%
CNN-1D-Dropout	3hrs	80%	81%
CNN-1D-Dense-Dropout	3hrs	77%	76%

As we can see from the table, all of our models beat the test accuracy of the baseline model, with weaker CPUs in much shorter training time. We believe this could be attributed to the form of our input feature vectors. To be more specific, in the baseline model, the log-mel spectrogram of one audio sample is divided into some number of frames with equal shape, but that actually prevents the model to see the audio sample's change in time from those frames since they will be randomly shuffled before feeding into the network. In our approach, the log-mel spectrogram is zero-padded and used as the input feature vectors directly. Since we used 1-D convolutional layer, our models actually slide through the input log-mel spectrogram from left to right, therefore, the change in the audio sample in time could be captured by our model, and that could be the reason why our models gave better test result in a much shorter training time than the baseline model.

In the meantime, as we can see that in CNN-1D, our first model, the training accuracy is much better than the test accuracy, which suggested overfitting. We believe this is because we did not apply any form of regularization to it, unlike what we did to CNN-1D-Dropout and CNN-1D-Dense-Dropout.

6 Future work

First, we would like to split the training data into training and validation set, then we can do grid search to tune hyperparameters regardless what kind of model we eventually end up with.

Second, right now, our second model - CNN-1D-Dropout gave the best test accuracy. It only has one dense layer, and we would like to add another dense layer before the final dense layer, so that we have more than one fully-connected layer. Also, we would like to replace the global max pooling 1d with a flatten layer, so we would get a lot more trainable parameters in the last three layers: flatten - dense - dense. Then, we need to impose enough regularizations to the model since we might see pretty good training accuracy on this new model, but it might just be overfitting, given the number of parameters are increased a lot.

Third, we would like to try to implement a Deep CNN, which should be ResNet. Since we found out that in audio data, the changes over time are pretty important, and they are not isolated, but related. Therefore, our model should be able to capture the effect of the changes over time in audio data. With ResNet, we could achieve that by adding identity in previous layers to current layers.

Lastly, we would like to try a recurrent model, such as LSTM, to learn from the training data. Since the audio is in the from of sequence naturally, unlike image. Therefore, a recurrent model would be a better fit for task dealing with audio than CNN.

7 Conclusion

In this project, we present three different shallow CNN models that give accepted test result with much shorter training time compared to the baseline model provided by DCASE 2018. Our best model reported a test accuracy of 80% in 3 hours of training time, compared to the baseline model with a test accuracy of 70% in 12 hours of training time. We believe the form of the input feature vectors of the audio data and the way the CNN model convolve would make the learning process much different for audio data. Given audio data is naturally sequential, using 1-D convolutional layer to slide through the log-mel spectrogram actually works relatively well in short training time with shallow CNN models. Future work could be done to examine the effectiveness of using 1-D convolutional layer in large-scale audio classification task.

References

[1] https://www.kaggle.com/c/freesound-audio-tagging.

[2] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 776–780, March 2017.

[3] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[4] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline." Submitted to DCASE2018 Workshop, 2018.

[5] E. Fonseca, J. Pons, X. Favory, F. Font, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: a platform for the creation of open audio datasets," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, (Suzhou, China), pp. 486–493, 2017.

[6] B. McFee, M. McVicar, O. Nieto, S. Balke, C. Thome, D. Liang, E. Battenberg, J. Moore, R. Bittner, R. Yamamoto, D. Ellis, F.-R. Stoter, D. Repetto, S. Waloschek, C. Carr, S. Kranzler, K. Choi, P. Viktorin, J. F. Santos, A. Holovaty, W. Pimenta, and H. Lee, "librosa 0.5.0," Feb. 2017.

[7] https://github.com/DCASE-REPO/dcase2018_baseline/tree/master/task2.