THE AALTO SYSTEM BASED ON FINE-TUNED AUDIOSET FEATURES FOR DCASE 2018 TASK2 —— GENERAL PURPOSE AUDIO TAGGING

Zhicun Xu, Peter Smit, Mikko Kurimo

Aalto University, Department of Signal Processing and Acoustics, Espoo, Finland,
{zhicun.xu, peter.smit, mikko.kurimo}@aalto.fi

ABSTRACT

In this paper, we presented a neural network system for DCASE 2018 task 2, general purpose audio tagging. We fine-tuned the Google AudioSet feature generation model with different settings for the given 41 classes on top of a fully connected layer with 100 units. Then we used the fine-tuned models to generate 128 dimensional features for each 0.960s audio. We tried different neural network structures including LSTM and multi-level attention models. In our experiments, the multi-level attention model has shown its superiority over others. Truncating the silence parts, repeating and splitting the audio into the fixed length, pitch shifting augmentation, and mixup techniques are all used in our experiments. The proposed system achieved a result with MAP@3 score at 0.936, which outperforms the baseline result of 0.704 and achieves top 8% in the public leaderboard.

Index Terms— audio tagging, AudioSet, multi-level attention model

1. INTRODUCTION

Sound contains various information that could indicate the sound sources, surrounding environment, music genres, possible dangers or even the emotions of the speakers. Thus sound plays a crucial part in our daily communication and interaction with the world. Teaching machines to listen, such as recognizing the sound events, would benefit humans in many ways. Relevant applications include public surveillance, sound print, auditory medical information monitoring and multimedia content analysis.

General purpose audio tagging is a task that infers descriptive labeling from these sounds such as musical instruments, domestic animals, and human activities. Recognizing and labelling these sound events with appropriate tags can provide a powerful tool to categorize the extensively large amount of audio data from the internet. With the labels, the content providers can give better services such as providing audio descriptions for visually or hearing impaired people, and providing powerful searching tools for the people working in the entertainment industries.

The traditional methods for doing the audio classification and audio tagging are adapted from speech recognition such as the Mel-frequency cepstral coefficients (MFCC) features and simple Gaussian mixture model (GMM) classifiers [1]. Recently, deep neural networks have proven its great usefulness in feature engineering, classification, detection, and audio synthesis. Almost all the submissions in DCASE 2017 [2] used some forms of neural networks such as long short-term memory units (LSTM) and convolutional neural networks (CNN). Thus deep learning is our main research approach for this task. Google has created a dataset called AudioSet with a structured hierarchical ontology [3], which provides the proper way to annotate the sounds. Instead of releasing the original audio files, each sample from the AudioSet is represented by 10 instances of 128 dimensional features. Google also provides a pre-trained model to generate the 128 features for 0.960 seconds audio. Kong *et al*. proposed a single-level attention model on this dataset, which outperformed the Google' baseline [4]. Later, Yu *et al*. proposed a multi-level attention model as an extension to previous single-level attention models, the results outperformed both the single-level attention model and the baseline [5].

Thus we came up with the idea to fine-tune the feature generation model first for the 41 classes of this DCASE task, and then try the multi-level attention model on the generated compact features. We aimed at proving the usability of these compact features and the superiority of multi-level attention model. We also incorporated pitch shifting augmentation and mixup techniques.

The structure of this paper is as follows. Section 2 describes methods on how to fine-tune the existing CNN model for the given 41 classes. Section 3 describes the design of our proposed system. The experimental results and conclusions can be seen in section 4 and 5 respectively.

2. FINE-TUNED VGGISH MODEL

2.1. Structure

VGGNet [6] with deep CNN structures has worked greatly well in image classification. Since the spectral representation of an audio signal can be used directly as an image, this deep CNN structure is also a promising techniques in many machine listening tasks, such as audio tagging, audio event detection and acoustics scene classification. VGGish model [7] is a variant of the VGG model with minor modifications. Table 1 shows the detailed structure of the modified version. Google trained this model on the YouTube-100M dataset with a total of 100 million videos. The training set contains 70 million videos and they were further split into non-overlapping 960 ms audio frames. Log-mel spectrograms were then computed as 96×64 images for the input of the VGGish model. The evaluation results of VGGish model showed its great usability in the audio domain.

The pre-trained VGGish model can act as a feature extractor. The provided 128 embedding features for 0.960 seconds are very compact, high level and semantically meaningful as well. These

This work was supported by the Kone foundation, and the European Union's Horizon 2020 research and innovation programme via the project MeMAD (GA780069). Computational resources were provided by the Aalto Science-IT project.

Table 1: VGGish model structure. The kernel size for CNN is (3, 3) and the kernel size for pooling is (2, 2). The activation function is 'relu' for all the layers.

layers	filters@size	layers	filters@size
1. input	1@96×64	**9. conv**	512@12×8
2. conv	64@96×64	**10. conv**	512@12×8
3. pooling	64@48×32	**11. pooling**	512@6×4
4. conv	128@48×32	**12. flatten**	12288
5. pooling	128@24×16	**13. fc**	4096
6. conv	256@24×16	**14. fc**	4096
7. conv	256@24×16	**15. fc**	128
8. pooling	256@12×8	**16. ...**	...

compact features can then be fed into a shallower model for classification. VGGish model can also become part of a larger model where more layers are added upon the model, which makes it possible to adapt and fine-tune this model for different datasets.

2.2. Balanced and Unbalanced Fine-tuning

The original VGGish model is trained for a multi-label classification task with the Google AudioSet ontology [3], which has 632 classes in a hierarchy tree structure. This challenge is a multi-class classification task, which means there is only one correct label for a sample. The 41 classes for this task all belongs to the AudioSet ontology. The training data is provided by FreeSound Dataset (FSD) [8]. The dataset has in total 9473 training files with the smallest class having only 94 training samples and the biggest class having 300 training samples. The average length of the audio files is 6.7 seconds. The more detailed description of the task setup, dataset and baseline can be seen in [9]. We added one hidden layer with 100 units after the VGGish model, and the final classification layer has 41 units with softmax activations.

To fine-tune the VGGish model, we must divide the data into a training part and a validation part. The validation loss is then used as the stopping criterion in case of overfitting. Firstly, we used the first 8000 audio files as training data and the remaining 1473 audio files as validation data. To fully use the data, we then used the last 8000 audio files as training data and the first 1473 audio files as validation data. Thus for one balancing technique we would get two models, one of which is fine-tuned on the first 8000 audio files and the other is fine-tuned on the last 8000 audio files.

Mini-batch balancing is a technique which assigns equal number of training samples for each class in each mini-batch. It has been proven useful on AudioSet to deal with the unbalanced dataset [4]. For the balanced fine-tuning, we followed the mini-batch balancing method by choosing an audio sample from each class for every training batch. However, it is not perfect balancing since audio samples may contain different number of 0.960 seconds length segments. In total, there would be 41 audio files, around 280 log-mel spectrograms, for each training batch. On the other hand, we also fine-tuned a unbalanced model by randomly choose 41 audio files in each batch for comparison. So we got 4 models in total.

The fine-tuned model are used as the feature extractors for the audio. Each audio file is firstly split into several non-overlapping 0.960 seconds segments. And for each segment, the log-mel spectrogram with dimension 96×64 is computed, then the spectrogram is fed into the fine-tuned model to get the 128 dimensional features.

3. THE PROPOSED SYSTEM

3.1. Preprocessing

The provided audio files are provided as PCM 16 bits, 44.1 kHz, mono format. However, the original quality might be quite different since they are uploaded by users all around the world. For preprocessing, we trimmed the silence parts only in the beginning and the end to avoid the influence of these irrelevant silence parts. We used the librosa[1] toolbox for the trimming here. The normalizing and pitch shifting mentioned in the following section are also implemented using this toolbox. The silence parts that are in the middle might contain important dynamic information for classification. We then normalized the amplitude of the audio files to [-1, 1]. The trimmed and normalized audio files are then split into non-overlapping 0.960 seconds segments. For each segment the log-mel spectrogram with size 96×64 is computed with 25ms window size and 10ms hop size. The 96 represent the frame size and 64 represents the number of frequency bands. The spectrograms are then fed into the fine-tuned VGGish model for features extraction.

3.2. Data Augmentation

The data is unbalanced where only around half of the classes have 300 audios. The imbalance problem could make the model emphasize more on the classes with more training samples and neglect to learn from the classes with less samples. To deal with this problem, we used the pitch shifting, repeat and split strategies for augmentation.

For the implementations of pitch shifting, we randomly choose an integer number between (-12, 12) for each audio file, and then shift the correspond number of semitones to create the new file. The shifting does not affect much on the melodic or stable classes, since the maximum shift is only one octave. For the noise-like classes such as 'fireworks' and 'fart', the perception is still relatively okay even for those with maximum amount of shift.

Some of our models used fixed length audio as input, but the audio files have variable length. To fully use the full length of the audio, we split them into several fixed length audios as input, which also gave us more training data. For the audio files that are shorter, we repeat them and concatenate them together to the fixed length.

3.3. Mixup

Mixup means training the neural networks using the convex combinations from pairs of examples and their labels. It helps the neural network to emphasize the linear combination between the training samples, which can improve the generalization ability, reduce memorization of the corrupted labels, increase the robustness to adversarial examples [10]. The training data provided by the organizer has another label representing if the audio is manually verified. Thus trying this method can help reduce the effect of the potential corrupted or misclassified audio in the unverified audios. Besides, this will also allow the model learn to distinguish between classes. The following equation [10] explains how the algorithms works:

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1-\lambda) x_j \\ \tilde{y} &= \lambda y_i + (1-\lambda) y_j \\ \lambda &\sim \text{Beta}(\alpha, \alpha) \\ \alpha &\in (0, \infty) \end{aligned} \tag{1}$$

[1] https://github.com/librosa/librosa

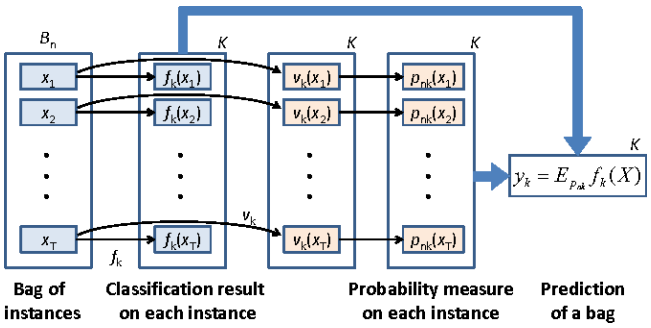

Figure 1: Attention model for Audio Set [4]

The (x_i, y_i) and (x_j, y_j) are two training and label pairs, which are randomly sampled form the training data. λ is drawn from a Beta distribution and lies in the region [0,1]. The experiments in the work [10] shows that increasing the α would increase the training error on real data and minimize the generalization gap. They also found that $\alpha \in [0.1, 0.4]$ would give an improved performance and large α will leads to underfitting.

3.4. Multi-level Attention Model

The dynamic changes are crucial in audio tagging challenges. Only considering the frequency structure might be useful for the recognition of the melodic instruments, but for the tags such as 'gunshort', and 'knock', the temporal changes are much more important. An attention model, which assign different weights for the instances in a time series, is a good strategy for considering the dynamics of sound. An attention model structure [4] presented for the AudioSet can be seen in Fig. 1. B_n represents an audio file with 10 instances $x_1 \sim x_{10}$, $f_k(x_n)$ represents the classification results for class K. The weights for each instance are firstly computed as $v_k(x_1) \sim v_k(x_{10})$ and then normalized to $p_{nk}(x_1) \sim p_{nk}(x_{10})$ so that $\sum_i^{10} p_{nk}(x_i) = 1$. Finally, multiplying the weights with the relevant prediction results and summing them together gives the final prediction for class K.

Combining the features from different levels and different timescale can provide more accurate descriptions. A CNN-based architecture [11] has shown great performance for music tagging by aggregating the multi-level and multi-scale features. Concatenated features extracted from different levels of CNN has also been proven useful in computer vision tasks [12]. Inspired by the works [4] [5] on Google Audio Set, we decided to choose a similar multi-level attention structure for audio. We did not try multi-scale methods because the fine-tuned VGGish models can only generate features for the 0.960 seconds fixed length audio. We prepared each training sample as 6 concatenated segments, each segment contains 0.960 seconds audio. Then the 6×128 dimensional features extracted using the fine-tuned VGGish model are fed into the neural network for training.

Fig. 2 shows the model structure. Each of the 6 instances are fed into a fully connected neural networks with 3 layers. Different color boxes represent the different levels features, and the features from the same level are then fed into an attention model, shown in Fig. 1, to get the predictions. We then concatenated all three predictions from different levels and forward them into the final classification layer with the softmax activation to get the 41 probabilities for the 41 classes.

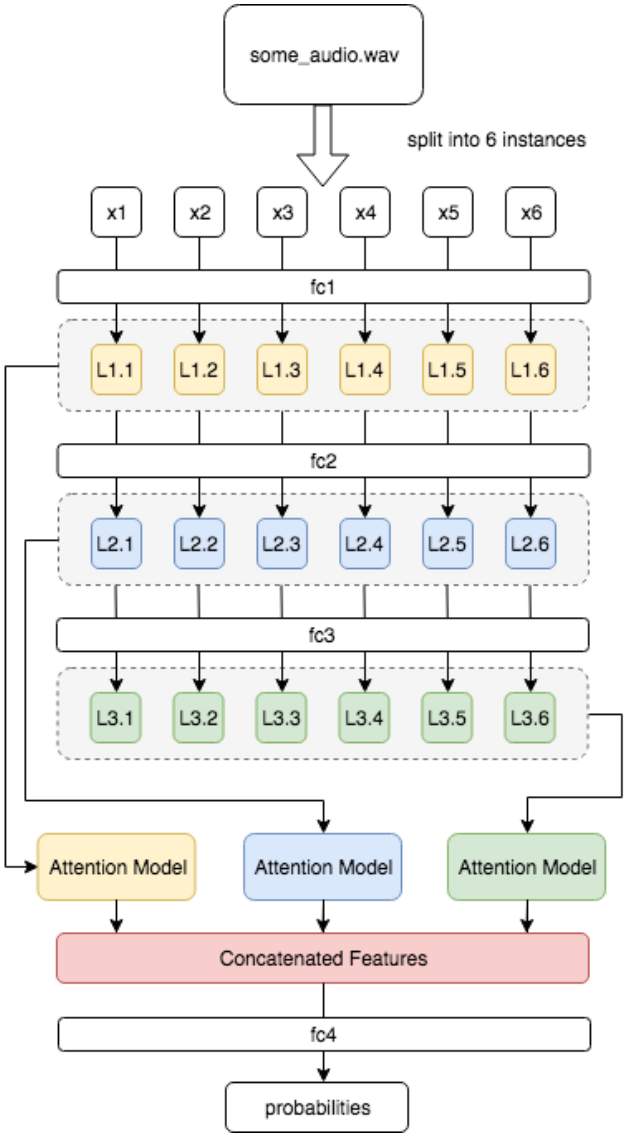

Figure 2: Model Structure for Multi-level Attention model

3.5. Evaluation Metric

The evaluation uses the Mean Average Precision @ 3 (MAP@3)[2]. The detailed implementation and explanation can be seen in the link provided in the footnote. Simply speaking, up to three predictions can be given even though there is only one correct label. The order of the predictions matters in this setting. If the correct label is predicted in the 1st position among those three predictions, the system would get a score 1. 2nd position would get a score 1/2, 3rd position would get a score 1/3. If there is no correct answer in the three predictions, the score would be zero. The average of the scores for all the testing samples would be the final evaluation score. However, the public leaderboard only has the score for around 19% of the testing data. So the scores in this paper are all evaluated upon

[2] https://github.com/benhamner/Metrics/blob/master/Python/ml_metrics/average_precision.py

Table 2: The one second DNN evaluation results on different fine-tuned VGGish models

Feature Extractor	MAP@3
Baseline	0.704
Original VGGish	0.606
Balanced First Part	0.870
Balanced Last Part	0.864
Balanced All	0.891
Unbalanced First Part	0.858
Unbalanced Last Part	0.872
Unbalanced All	0.892
Ensemble All	**0.903**

Table 3: Evaluation results on different model structures and hyper-parameters.

Index	Model Structure	MAP@3
A	1-Segment Multi-level	0.903
B	LSTM	0.914
C	6-Segment Multi-level Attention	0.925
D	C + Pitch shifting Augmentation	0.930
E	D + Mixup ($\alpha = 0.1$)	0.930
F	D + Mixup ($\alpha = 0.2$)	**0.936**
G	D + Mixup ($\alpha = 0.4$)	0.931
H	D + Mixup ($\alpha = 1.0$)	0.930

those 19% and the final results might be different.

4. EXPERIMENTS

4.1. Different fine-tuned VGGish models

Firstly, we tested the performance of different fine-tuned VGGish models on a simpler model structure. In this experiment, we built a simple 3-layer fully connected neural networks with each layer containing 600 units. The classification results from each layer are concatenated together as input to the final layer. The activation function is 'relu' for all the layers, except that the classification layers use 'softmax' activation function. Batch normalization and dropout with ratio 0.4 are used after each middle layer. The input is the 128 dimensional features for 0.960 seconds. For evaluation on files with different length, we took the average results as the score. As can be seen from Table 2, all of the fine-tuned models outperform the original VGGish model. 'Balanced first part' means the feature extractor is trained using mini-batch balancing on the first 8000 training samples and validated on the remaining ones. 'Balanced last part' means the feature extractor is trained using mini-batch balancing on the last 8000 training samples and validated on the remaining ones. 'Balanced all' means taking the geometric means of the results from the 'Balanced first part' and 'Balanced last part'. 'Unbalanced' means the model is trained without mini-batch balancing. The remaining names can be interpreted in similar way. There is no clear difference between the models trained with mini-batch balancing and those without. However, using geometric mean results of all 4 models will give the best results.

Table 4: Evaluation results on different number of segments multi-level attention models

Number of Segments	1	2	4	6
MAP@3	0.917	0.919	0.931	0.936
Number of Segments	8	10	Ensemble All	
MAP@3	0.929	0.931	0.931	

4.2. Different neural network structures and hyper parameters

Section 4.1 has shown that utilizing the results from all 4 fine-tuned models would give the best results. Thus for each model structure in this section, we also used the same strategy. The results for different model structures and different random mix factors α can be seen in Table 3. Model A has the same setting as the best results from Table 2. Model B uses one LSTM layer with 600 hidden units upon the original variable length input. Model C has the same 6-segment multi-level attention structures shown in Section 3.4, other parameter are same as A. Model D is based on model C with pitch shifting augmentation. Model E, F, G, H are based on D with different mixup factors.

From the comparison between model A, B, C, LSTM is better than the 1-segment multi-level model. The 6-segment multi-level attention model performs the best. Thus temporal information plays an important role in recognition, and multi-level attention model has better ability to model the temporal information than LSTM in our settings. Using the pitch shifting augmentation to generate a relevantly more balanced training data also improves the results a little bit. The comparison between model E, F, G, H shows mixup with the $\alpha = 0.2$ has the best performance among these 4 choices. However, since the difference among C-H is quite small, further significance test might still be needed.

4.3. Different number of segments

We also tried the multi-level attention model with different number of segments as input. Other training settings, such as the pitch shifting augmentation and mixup, are the same as the best results in Table 3. Results can be seen in Table 4. For the ensemble results, we took the geometric mean of all the results. We can see that the 6-segment model has the best performance and the ensemble does not improve the overall score. The reason might be that these models are not diverse enough.

5. CONCLUSIONS

In this paper, we tried different fine-tuning methods on the AudioSet VGGish model for generating 128 features for 0.960s audio. The results show that the combination of the 4 models trained with different train-validation splitting and balanced/unbalanced techniques would give the best results. We also implemented different neural network structures for comparison and found that multi-level attention model performs the best among all. This shows the importance of modeling the temporal information. The 6-segment multi-level attention model with pitch shifting augmentation and mixup method using $\alpha = 0.2$ has the best MAP@3 at 0.936 in public leaderboard. Further research might include more thorough fine-tuning, building own CNN model for feature generation, utilizing multi-scale features along with multi-level features for the attention model.

6. REFERENCES

[1] H. Aronowitz, "Segmental modeling for audio segmentation," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4. IEEE, 2007, pp. IV–393.

[2] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "Dcase 2017 challenge setup: Tasks, datasets and baseline system," in *DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017.

[3] J. F. Gemmeke, D. P. 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 *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 776–780.

[4] Q. Kong, Y. Xu, W. Wang, and M. D. Plumbley, "Audio set classification with attention model: A probabilistic perspective," *arXiv preprint arXiv:1711.00927*, 2017.

[5] C. Yu, K. S. Barsim, Q. Kong, and B. Yang, "Multi-level attention model for weakly supervised audio classification," *arXiv preprint arXiv:1803.02353*, 2018.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[7] 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," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 131–135.

[8] 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, 2017, pp. 486–493.

[9] E. Fonseca, M. Plakal, F. Font, D. P. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline," *arXiv preprint arXiv:1807.09902*, 2018.

[10] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[11] J. Lee and J. Nam, "Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging," *IEEE signal processing letters*, vol. 24, no. 8, pp. 1208–1212, 2017.

[12] X. Meng, B. Leng, and G. Song, "A multi-level weighted representation for person re-identification," in *International Conference on Artificial Neural Networks*. Springer, 2017, pp. 80–88.