GENERAL-PURPOSE AUDIO TAGGING BY ENSEMBLING CONVOLUTIONAL NEURAL NETWORKS BASED ON MULTIPLE FEATURES

Kevin Wilkinghoff

Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE
Fraunhoferstraße 20, 53343 Wachtberg, Germany
kevin.wilkinghoff@fkie.fraunhofer.de

ABSTRACT

This paper describes an audio tagging system that participated in Task 2 "General-purpose audio tagging of Freesound content with AudioSet labels" of the "Detection and Classification of Acoustic Scenes and Events (DCASE)" Challenge 2018. The system is an ensemble consisting of five convolutional neural networks based on Mel-frequency Cepstral Coefficients, Perceptual Linear Prediction features, Mel-spectrograms and the raw audio data. For ensembling all models, score-based fusion via Logistic Regression is performed with another neural network. In experimental evaluations, it is shown that ensembling the models significantly improves upon the performances obtained with the individual models. As a final result, the system achieved a Mean Average Precision with Cutoff 3 of 0.9414 on the private leaderboard of the challenge.

Index Terms— audio tagging, acoustic event classification, convolutional neural network, score-based fusion

1. INTRODUCTION

In past years, deep convolutional neural networks (CNN) or variants thereof have taken over almost any area related to image classification and computer vision in general. As audio data can easily be converted into images by computing their spectrogram, they have also become popular for acoustic event detection and classification [1, 2, 3]. Moreover, CNNs trained on spectrograms are reported to outperform classical approaches as for example Hidden Markov Models (HMMs) [4]. In this work, an ensemble of CNNs for the purpose of acoustic event classification is presented. The ensemble consists of five different CNNs trained on multiple features derived from the acoustic data and another neural network used for fusing the scores. More concretely, Mel-frequency Cepstral Coefficients (MFCCs) [5], Perceptual Linear Prediction (PLP) features [6], Mel-spectrograms and the raw data are used as features.

The presented system participated in Task 2 "General-purpose audio tagging of Freesound content with AudioSet labels" of the "Detection and Classification of Acoustic Scenes and Events (DCASE)" challenge 2018. The audio dataset being used consists of a subset of the Freesound dataset (FSD) [7] which has been taken from the online database Freesound [8]. All files in FSD have been uploaded and tagged by many different users. A mapping from these tags to AudioSet [9] categories has been manually designed and all samples have been automatically annotated. After that, the annotations of some files have been manually verified although there are some cases where the annotators could not agree on a label.

As a result, the part of the dataset used for training consists of 9473 files belonging to one of 41 categories. The distribution is not uniform but ranges from 94 to 300 samples per category. In addition to that, the length of the audio files is between 300 ms and 30 s. Only 3710 samples are manually verified and for each file it is known if this is the case or not. According to the organizers of the challenge, at least two thirds of the non-verified samples are estimated to be labeled correctly for each category. The test set consists of 1600 manually verified samples and 7800 non-verified samples referred to as padding files. Furthermore, 301 files have been used as validation data for the public leaderboard and 1299 files have been used for the final evaluation of the challenge (private leaderboard). The padding files have not been used for any evaluation and were simply kept as test data to prevent participants from cheating. For a more detailed description of the challenge and dataset the reader is referred to [10].

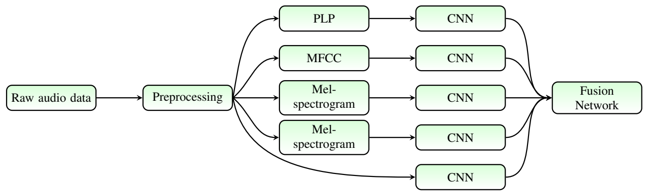

Figure 1: Structure of the audio tagging system

This paper is organized as follows. First, the global structure of the audio tagging system is presented. Then, all five CNNs based on all features including their hyperparameters as well as the Fusion Network are described. After that, experimental results are given in order to compare the performance of the CNNs and to show that creating an ensemble of them is highly beneficial.

2. STRUCTURE OF THE AUDIO TAGGING SYSTEM

An overview of the audio tagging system's structure is depicted in Fig. 1. The system is an ensemble consisting of five different CNNs trained on PLP features, MFCCs, Mel-spectrograms and the raw audio data. In the following, we will refer to all of them as features. First, the audio data has been preprocessed with Librosa [11]. All files have been downsampled from 44100 Hz to 24000 Hz and were normalized with respect to the maximum norm. Any part of the files that is 50 db below peak power is considered as silence and has been removed. After these steps, all features have been extracted with Librosa except the PLP features which were computed via Sidekit [12]. We used a dimension of size 22 for the PLP features, one of size 64 for the MFCC features and one of size 96 as well as 128 for the Mel-spectrograms.

Since CNNs need input data of fixed size and the audio files vary greatly in length, we partitioned the sequence of features into overlapping windows of length 500 with an overlap of 50 (ordered with respect to time). To enforce a more different behavior of both CNNs trained on Mel-spectrograms, a window size of 200 with overlap 20 has been used for the 128 dimensional features instead. Much larger windows of length 48000 (i.e. clips of 2 seconds) with an overlap of 4800 have been used for the raw data. In case that any of the feature windows was too short, the window has been repeated until the desired length was reached. Later, the geometric mean of the scores obtained with all windows belonging to the same file has been taken as the final score for that file. Note that a similar windowing approach is also used for the baseline system of the challenge presented in [10].

The output scores of all CNNs obtained with the validation data are fused with a neural network. It basically applies Logistic Regression (see [13]) with only a single parameter per model and one global bias. Thus, the final scores, used for identifying the classes by returning the argument of the maximum score, are computed as a weighted linear combination of all models' scores. This has the advantage that better performing models have a higher influence than weaker ones and usually leads to a better performance than taking the mean (see [14]). A weighted geometric mean of the probabilities corresponds to a weighted sum (i.e. a standard linear transformation of a neural network) in log-space. Therefore, we converted all softmax probabilities to log-space before applying the neural network as this improved the overall performance of the system. Additionally, we standardized the scores of each model i.e. we subtracted the mean and divided by the standard deviation of all scores.

All hyperparameters involved have been tuned using Stratified 5-Fold Cross Validation as implemented in Scikit-learn [15]. After training all models on the five data splits, also five ensembles were obtained and each of them has been applied to the test dataset instead of retraining each model on the full training dataset. Hence, the full ensemble consists of 25 submodels. To have a single final output, we simply took the geometric mean of the fused output probabilities obtained by evaluating the five model ensemble with the test dataset. By doing so, the whole training dataset has been used while also having the possibility to monitor the system's performance using the validation data. Thus, we could apply Early Stopping when training each CNN by monitoring the validation loss. Furthermore, the scores obtained with the validation data could be used to train the neural networks used for fusing the scores which would not have been possible otherwise. As a result, a slightly better performance was reached.

3. NEURAL NETWORK ARCHITECTURES

All CNNs and the Fusion Network have been implemented using Keras [16] with Tensorflow [17]. Their structures are depicted in Tab. 1, 2, 3, 4, 5 and 6 which can be found in the appendix. To be able to compare the complexities of all models, we included their total number of parameters in Tab. 7.

As stated in [2, 18], data augmentation is of great help in order to have a well performing system. Therefore, Mix-up [19] with $\alpha = 1$, Cutout [20], Dropout [21] and vertical shifts up to 60% of the total width have been applied randomly in each batch. When using the Mel-spectrogram based CNNs, we also used feature-wise centering as well as featurewise normalization of the standard deviation. All CNNs have been trained for 800 epochs with a batch size of 64 by minimizing the Categorical Crossentropy. To speed up the training process, Adam [22] with a learning rate of 0.001 and weight decay of 0.0001 as well as Batch-Normalization [23] have been used. Due to the non-uniform distribution of training samples per class, we used balanced class weights during training to put more emphasis on the classes having fewer samples as implemented in Scikit-learn (which is loosely following [24]).

As noted above, many labels are not guaranteed to be correct due to the automatic labeling procedure. To compensate for this, Label Smoothing [25] has been applied. This means that categorical labels are altered by reducing the single 1 to $1 - \alpha_{nv}$ and replacing all zeros with $\frac{\alpha_{nv}}{N-1}$ where $\alpha_{nv} \in [0, 1]$ and $N \in \mathbb{N}$ denotes the number of classes. According to the organizers of the challenge, at least two thirds of the non-verified labels are correct. Because of that, a value of $\alpha_{nv} = 0.3$ has been used for all non-verified samples. As the annotators could not agree on a label for some of the files and thus at least some labels are probably wrong, we used a value of $\alpha_v = 0.05$ for the manually verified data. In addition to that, sounds recorded in realistic environments, naturally, contain more than one class although the organizers of the challenge tried to prevent this. For example, a barking dog could have been recorded outside in the streets where buses or other vehicles are also present or a snare drum, base drum and hi-hat are played together in one recording of a drummer. Thus, it is not preferable that the system tries to make a hard prediction per file which is another reason to include Label Smoothing and Mix-up.

For the Fusion Network slightly different parameters and no data augmentation techniques have been used. It has been trained for 1000 epochs with a batch size equal to the size of the full dataset (i.e. with the exact gradient) using Adam with a learning rate of 0.01. Instead of Label Smoothing and Weight Decay, L2 Regularization with a penalty of 0.0005 has been applied. When using batch normalization, it was important to disable the additional beta and gamma parameters in Keras. Otherwise, there is an additional linear transformation with independent weights for each class and each model. This gives the Fusion Network too much power and the generalization to unseen test data is much worse.

4. EXPERIMENTAL RESULTS

The mean Average Precisions with Cutoff 3 (mAP@3s) [8], we obtained in the challenge, can be found in Fig. 2 and have been depicted for each model individually. They relate to the dataset used for the public leaderboard and the private leaderboard, which correspond to the final challenge results, as well as our own five validation splits of the training dataset. To make it more clear, since all five validation sets of the splits together result in the whole training set again, each mAP@3 is based on all 9473 training files. Note that the data split used for the public leaderboard only consists of 301 files and thus is a less accurate estimate of the true mAP@3.

As expected, the obtained mAP@3s are much higher when only evaluating with the manually verified data but still using the non-verified samples for training. Thus, the assumption that the weakly-labeled files are by far not optimally labeled seems to be true. Since complete the test set is manually verified, the results of the challenge are closer to the presented mAP@3s of the manually verified subset than to the full validation set. It is also clear that the performance obtained with the validation data is a bit too optimistic because all parameters have been adapted to these data splits. The CNNs trained on Mel-spectrograms perform better than the ones trained on PLP and MFCC features which both result in similar mAP@3s. In addition to that, the CNN trained on the raw data performed worst but still reasonably well. Interestingly, a relatively small number of model parameters seems to be sufficient since both

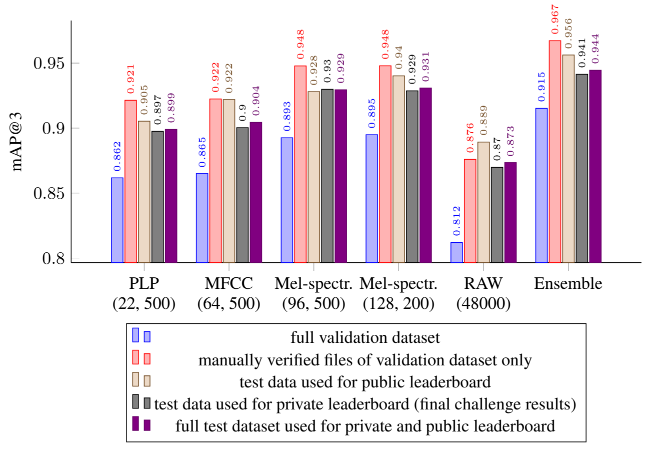

Figure 2: Mean Average Precision with Cutoff 3 obtained with the CNNs and the ensemble.

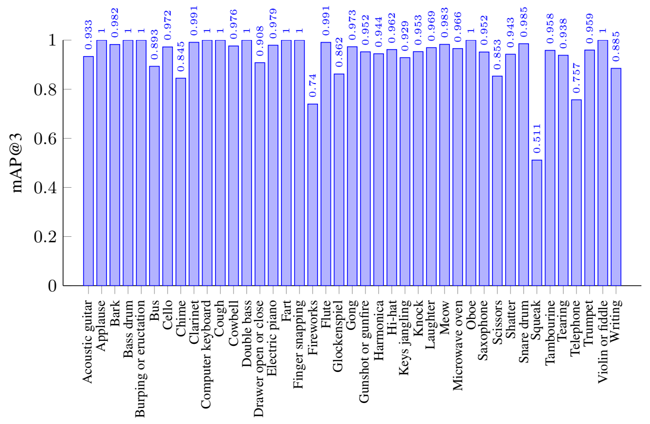

Figure 3: Mean Average Precision with Cutoff 3 per class obtained with all test files used for the public and private leaderboard.

Mel-spectrogram based CNNs performed equally well although one has about five times as many parameters as the other (see Tab. 7). Furthermore, it can be seen that the ensemble of all CNNs has a significantly higher mAP@3 than all of its components alone across all datasets. Hence, each of them has a slightly different view on the data providing additional information. In conclusion, building an ensemble of CNNs based on all the features is highly beneficial. But compared to the baseline system which yields an mAP@3 of 0.6945 on the private leaderboard and of 0.7049 on the public one, even the individual CNNs perform much better.

Next, the mAP@3s have been evaluated for each class individually to see which classes are easily identified and which ones are not. See Fig. 3, for a visualization. The mAP@3s for most classes are decent and ten classes are even identified without any errors. However, there are some classes whose mAP@3s are much lower than those of the other classes. The worst performing class is “Squeak” which has an mAP@3 of 0.511. A reason may be that squeaks are diversely sounding depending on the objects causing them. Looking at the confusion matrix (see Fig. 4), it can be seen that squeaks are fairly often confused with many other classes which supports this assumption. Additionally, there are also classes which are almost exclusively confused with each other as for example “Chime”

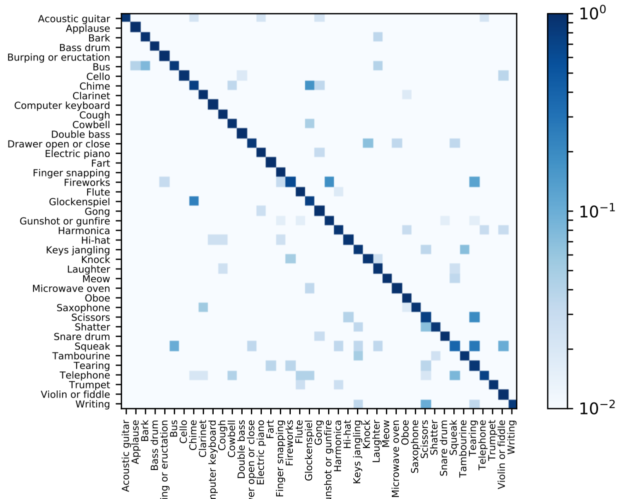

Figure 4: Confusion matrix obtained with all test files used for the public and private leaderboard.

and “Glockenspiel”. Both have a relatively low mAP@3 of approximately 0.85. From a human’s perspective it seems to be reasonable to confuse them as they are mostly perceived as similarly sounding.

5. CONCLUSIONS AND FUTURE WORK

In this work, an audio tagging system consisting of five CNNs and a neural network for fusing their scores has been presented and described. In the experimental evaluations conducted for Task 2 of the DCASE Challenge 2018, it has been shown that ensembling all models significantly improves upon the performances of each individual model. Furthermore, we examined the performance per class and were able to identify classes which are very easy to classify and some that are difficult to classify leading to more insights about the dataset used in this challenge.

An improvement of the presented audio tagging system could be achieved by using additional data augmentation techniques. Some techniques to be named are Pitch-Shifting, Time-Stretching, class conditional data augmentation as suggested in [18] or Equalized Mixture Data Augmentation [2] which is an extension of Mix-up. Another possibility is to make use of Transfer Learning and replace some models of the ensemble with pretrained deep neural networks as for example VGG or ResNet (as done in [3]). This will probably improve the performances of the individual models and thus also the capabilities of the whole audio tagging system. Last but not least, another improvement may be accomplished by designing custom features or classifiers (e.g. other CNNs) for differentiating among those classes which are hard to tell apart and add them to the ensemble.

6. ACKNOWLEDGMENT

The author would like to thank Daisuke Niizumi who also participated in this challenge and shared his insights publicly in the Kaggle discussion forum. His comments on data augmentation techniques such as Mix-up and Cutout have been of great help to improve the performance of the presented audio tagging system.

7. REFERENCES

[1] O. Gencoglu, T. Virtanen, and H. Huttunen, "Recognition of acoustic events using deep neural networks," in *Signal Processing Conference (EUSIPCO), 2014 Proceedings of the 22nd European*. IEEE, 2014, pp. 506–510.

[2] N. Takahashi, M. Gygli, B. Pfister, and L. Van Gool, "Deep convolutional neural networks and data augmentation for acoustic event detection," in *Interspeech*, 2016, pp. 2982–2986.

[3] 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.

[4] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *Signal Processing Conference (EUSIPCO), 2010 Proceedings of the 18th European*. IEEE, 2010, pp. 1267–1271.

[5] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980.

[6] H. Hermansky, "Perceptual linear prediction (plp) analysis of speech," *the Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990.

[7] 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.

[8] F. Font, G. Roma, and X. Serra, "Freesound technical demo," in *Proceedings of the 21st ACM international conference on Multimedia*. ACM, 2013, pp. 411–412.

[9] 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 *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 776–780.

[10] 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. URL: https://arxiv.org/abs/1807.09902*.

[11] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

[12] A. Larcher, K. A. Lee, and S. Meignier, "An extensible speaker identification sidekit in python," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5095–5099.

[13] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. Springer, 2006.

[14] K. Wilkinghoff, P. M. Baggenstoss, A. Cornaggia-Urrigshardt, and F. Kurth, "Robust speaker identification by fusing classification scores with a neural network," 2018, preprint, to appear in: Proceedings of the 13th ITG Symposium on Speech Communication.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[17] M. Abadi *et al.*, "Tensorflow: a system for large-scale machine learning," *OSDI*, vol. 16, pp. 265–283, 2016.

[18] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

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

[20] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, 2015, pp. 448–456.

[24] G. King and L. Zeng, "Logistic regression in rare events data," *Political analysis*, vol. 9, no. 2, pp. 137–163, 2001.

[25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

Table 1: Architecture of the PLP based CNN.

Layer	Output Shape	#Parameters
Input	(22, 500)	0
Convolution (kernel size: 5x5, strides: 2x3, ReLU)	(11, 167, 64)	1,664
Max-Pooling (pool size: 2x2, strides: 1x2)	(10, 83, 64)	0
Batch Normalization	(10, 83, 64)	256
Convolution (kernel size: 5x5, ReLU)	(10, 83, 128)	204,928
Max-Pooling (pool size: 2x2, strides: 1x2)	(9, 41, 128)	0
Batch Normalization	(9, 41, 128)	512
Convolution (kernel size: 3x3, ReLU)	(9, 41, 192)	221,376
Batch Normalization	(9, 41, 192)	768
Convolution (kernel size: 3x3, ReLU)	(9, 41, 256)	442,624
Batch Normalization	(9, 41, 256)	1,024
Convolution (kernel size: 3x3, ReLU)	(9, 41, 256)	262,400
Max-Pooling (pool size: 2x2, strides: 1x2)	(8, 20, 256)	0
Batch Normalization	(8, 20, 256)	1,024
Flatten	40,960	0
Dropout (0.5)	40,960	0
Dense (ReLU)	256	10,486,016
Batch Normalization	256	1,024
Dropout (0.5)	256	0
Dense (ReLU)	128	32,896
Batch Normalization	128	512
Dense (Softmax)	41	5,289

Table 2: Architecture of the MFCC based CNN.

Layer	Output Shape	#Parameters
Input	(64, 500)	0
Convolution (kernel size: 5x5, strides: 2x3, ReLU)	(32, 167, 64)	1,664
Max-Pooling (pool size: 2x2, strides: 1x2)	(31, 83, 64)	0
Batch Normalization	(31, 83, 64)	256
Convolution (kernel size: 5x5, ReLU)	(31, 83, 128)	204,928
Max-Pooling (pool size: 2x2, strides: 1x2)	(30, 41, 128)	0
Batch Normalization	(30, 41, 128)	512
Convolution (kernel size: 3x3, ReLU)	(30, 41, 192)	221,376
Batch Normalization	(30, 41, 192)	768
Convolution (kernel size: 3x3, ReLU)	(30, 41, 256)	442,624
Batch Normalization	(30, 41, 256)	1,024
Convolution (kernel size: 3x3, ReLU)	(30, 41, 256)	262,400
Max-Pooling (pool size: 2x2, strides: 1x2)	(29, 20, 256)	0
Batch Normalization	(29, 20, 256)	1,024
Flatten	148,480	0
Dropout (0.5)	148,480	0
Dense (ReLU)	256	38,011,136
Batch Normalization	256	1,024
Dropout (0.5)	256	0
Dense (ReLU)	128	32,896
Batch Normalization	128	512
Dense (Softmax)	41	5,289

Table 3: Architecture of the Mel Spectrogram based CNN.

Layer	Output Shape	#Parameters
Input	(96, 500)	0
Convolution (kernel size: 5x5, strides: 2x3, ReLU)	(48, 167, 64)	1,664
Max-Pooling (pool size: 3x3, strides: 1x2)	(46, 83, 64)	0
Batch Normalization	(46, 83, 64)	256
Convolution (kernel size: 5x5, ReLU)	(46, 83, 128)	204,928
Max-Pooling (pool size: 3x3, strides: 2x2)	(22, 41, 128)	0
Batch Normalization	(22, 41, 128)	512
Convolution (kernel size: 3x3, ReLU)	(22, 41, 192)	221,376
Batch Normalization	(22, 41, 192)	768
Convolution (kernel size: 3x3, ReLU)	(22, 41, 256)	442,624
Batch Normalization	(22, 41, 256)	1,024
Convolution (kernel size: 2x2, ReLU)	(22, 41, 256)	262,400
Max-Pooling (pool size: 2x2)	(11, 20, 256)	0
Batch Normalization	(11, 20, 256)	1,024
Flatten	56,320	0
Dropout (0.5)	56,320	0
Dense (ReLU)	256	14,418,176
Batch Normalization	256	1,024
Dropout (0.5)	256	0
Dense (ReLU)	128	32,896
Batch Normalization	128	512
Dense (Softmax)	41	5,289

Table 4: Architecture of the Mel Spectrogram based CNN.

Layer	Output Shape	#Parameters
Input	(128, 200)	0
Convolution (kernel size: 5x5, strides: 2x3, ReLU)	(64, 67, 64)	1,664
Max-Pooling (pool size: 3x3, strides: 1x2)	(62, 33, 64)	0
Batch Normalization	(62, 33, 64)	256
Convolution (kernel size: 5x5, ReLU)	(62, 33, 128)	204,928
Max-Pooling (pool size: 3x3, strides: 2x2)	(30, 16, 128)	0
Batch Normalization	(30, 16, 128)	512
Convolution (kernel size: 3x3, ReLU)	(30, 16, 192)	221,376
Batch Normalization	(30, 16, 192)	768
Convolution (kernel size: 3x3, ReLU)	(30, 16, 256)	442,624
Max-Pooling (pool size: 2x2)	(15, 8, 256)	0
Batch Normalization	(15, 8, 256)	1,024
Convolution (kernel size: 3x3, ReLU)	(15, 8, 256)	590,080
Max-Pooling (pool size: 2x2)	(7, 4, 256)	0
Batch Normalization	(7, 4, 256)	1,024
Convolution (kernel size: 3x3, ReLU)	(7, 4, 512)	1,180,160
Batch Normalization	(7, 4, 512)	2,048
Global Average Pooling	512	0
Dropout (0.5)	56,320	0
Dense (ReLU)	256	131,328
Batch Normalization	256	1,024
Dropout (0.5)	256	0
Dense (ReLU)	128	32,896
Batch Normalization	128	512
Dense (Softmax)	41	5,289

Table 5: Architecture of the raw data based CNN.

Layer	Output Shape	#Parameters
Input	48000	0
Convolution (kernel size: 31, strides: 4, ReLU)	(12000, 64)	2,048
Batch Normalization	(12000, 64)	256
Max-Pooling (pool size: 16)	(750, 64)	0
Dropout (0.2)	(750, 64)	0
Convolution (kernel size: 15, strides: 2, ReLU)	(375, 128)	123,008
Batch Normalization	(375, 128)	512
Max-Pooling (pool size: 4)	(93, 128)	0
Dropout (0.2)	(93, 128)	0
Convolution (kernel size: 8, ReLU)	(93, 192)	196,800
Batch Normalization	(93, 192)	768
Dropout (0.2)	(93, 192)	0
Convolution (kernel size: 4, ReLU)	(93, 256)	196,864
Batch Normalization	(93, 256)	1,024
Global Average Pooling	256	0
Dropout (0.5)	256	0
Dense (ReLU)	256	65,792
Batch Normalization	256	1,024
Dropout (0.5)	256	0
Dense (ReLU)	128	32,896
Batch Normalization	128	512
Dense (Softmax)	41	5,289

Table 6: Architecture of the Fusion Network.

Layer	Output Shape	#Parameters
Input	205	0
Batch Normalization	205	410
Fusion Layer (Softmax)	41	6

Table 7: Total number of parameters of each neural network.

Model	Input size	Total #Parameters
MFCC based CNN	(64, 500)	39,184,873
PLP based CNN	(22, 500)	11,659,753
Mel-spectrogram based CNN	(96, 500)	15,591,913
Mel-spectrogram based CNN	(128, 200)	2,817,513
raw data based CNN	48000	624,745
Fusion Network	205	416