COMBINING HIGH-LEVEL FEATURES OF RAW AUDIO AND SPECTROGRAMS FOR AUDIO TAGGING

Technical Report

Marcel Lederle[*]

University of Konstanz
78457 Konstanz, Germany
marcel.lederle@uni.kn

Benjamin Wilhelm[†]

University of Konstanz
78457 Konstanz, Germany
benjamin.wilhelm@uni.kn

ABSTRACT

We introduce a method for general-purpose audio tagging that combines high-level features computed from the spectrogram and raw audio data. We use convolutional neural networks with one-dimensional and two-dimensional convolutions to extract these useful high-level features and combine them with a densely connected neural network to make predictions. Our method performs in the top two percent of on the Freesound General-Purpose Audio Tagging Challenge.

***Index Terms*—** audio-tagging, CNN, raw-audio, mel-spectrogram

1. INTRODUCTION

In this report, we focus on building an audio tagging system that can assign one of 41 heterogeneous classes to an audio clip as of task 2 of the DCASE 2018 Challenge [1]. The system has to decide between classes drawn from the AudioSet Ontology [2] like "Acoustic guitar", "Bark", "Bus" and "Telephone".

For training and evaluation only the provided challenge dataset is used which contains diverse user-generated audio clips from Freesound (https://freesound.org) which feature unreliable labels. The challenge organizers labeled a subset of the audio clips manually while a larger set was labeled automatically. Therefore, the reliability of the labels varies. Additional information about the labeling process can be found in the publication describing the challenge task [1].

2. METHOD

For the classification, we train two convolutional neural networks on the raw audio input and the mel-scaled spectrogram and combine the learned features with a densely connected neural network. In the following we describe each model and the performed stacking separately.

2.1. CNN on raw audio (CNN_AUDIO)

For the CNN_AUDIO model, we use an architecture similar to common architectures for image classification like VGG16 (Simonyan et al. [3]) or AlexNet (Krizhevsky et al. [4]), but with one-dimensional convolutions and max pooling. The model takes a normalized array of audio samples with a sampling rate of 44.1 kHz as input.

The exact architecture is described in figure 1.

2.2. CNN on mel-scaled spectrogram (CNN_SPEC)

The CNN_SPEC model is a two-dimensional convolutional neural network on the mel-scaled spectrogram of the audio. The mel-scaled spectrogram was extracted using librosa (McFee et al. [5]) with a hop-length of 256.

The architecture is again similar to common image classification architectures and is described in detail in figure 2.

2.3. Stacking (CNN_COMB)

For the combined model we removed the dense layers of the CNN_AUDIO model and the CNN_SPEC model and concatenated the output features of the previous layers of both models. The concatenated features were then connected to five dense layers including the output layer with 41 classes. The hidden dense layers have 512, 256, 256 and 128 neurons respectively. The full model is illustrated in figure 3.

2.4. Data augmentation

To prevent our model from overfitting we use extensive data augmentation like random cropping and padding, time shifting and combining audio of the same or different classes. Each of these techniques is applied to the raw audio and the mel-scaled spectrogram.

2.4.1. Random Cropping

For audio samples that are longer than the model input, we use a crop of the size of the model input which is taken from a random position of the audio sample.

2.4.2. Random Padding

For audio files that are shorter than the model input, we pad it with zeros. The location of the original audio sample in the padded input is uniformly random. If an audio sample fits multiple times (e.g. n times) into the model input it appears $k \in \{1, \dots, n\}$ times with a probability of $1/n$.

[*] Shared first author
[†] Shared first author

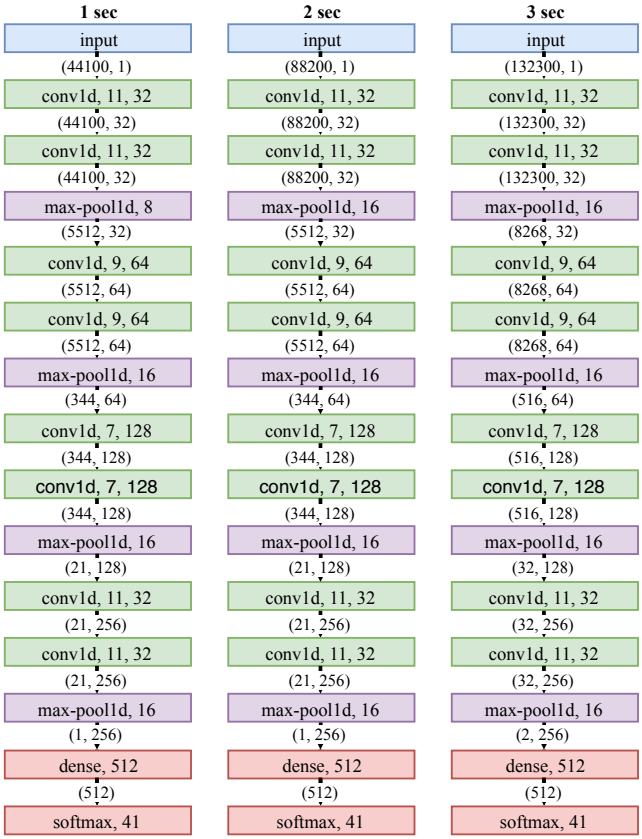

Figure 1: The architecture of the CNN_AUDIO model. After each convolution and the dense layer a ReLU activation function is applied and after each max-pooling and dense layer batch normalization is performed.

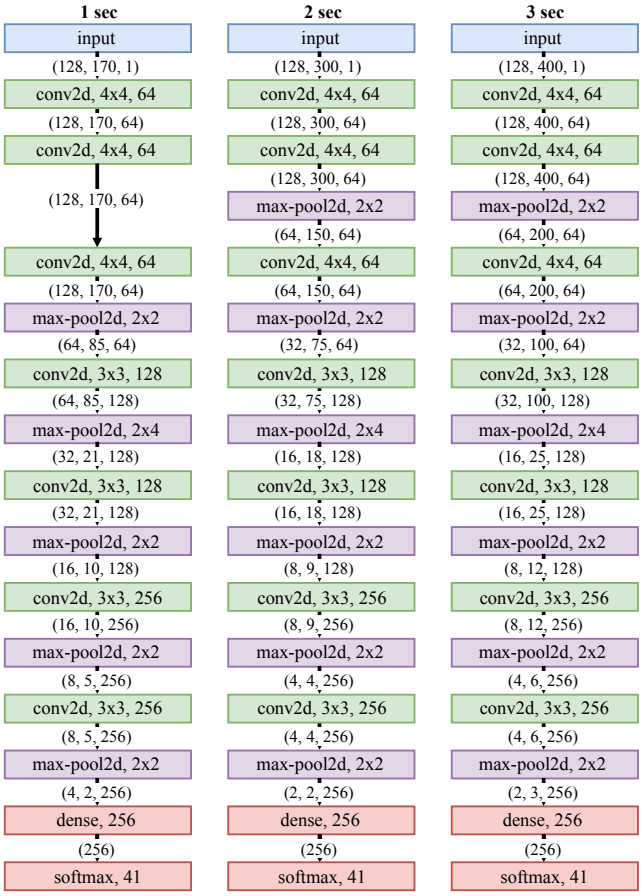

Figure 2: The architecture of the CNN_SPEC model. After each convolution and the dense layer a ReLU activation function is applied and after each max-pooling, dense or convolution without a following max-pooling layer batch normalization is performed.

2.4.3. Time Shifting

A uniformly random time shift is applied to the audio sample.

2.4.4. Same-Class Augmentation

For same class augmentation, multiple audio samples with the same class are summed up with random weights for each sample.

2.4.5. Different-Class Augmentation

For different class augmentation, multiple audio samples with different classes are summed up with random weights for each sample. The model is then required to predict the weights of each class included.

2.5. Implementation Details

We implemented the described method using Keras (Chollet et al. [6]) in Python.

To prevent overfitting and monitor the model performance during training we always excluded a part of the training data as validation data. In order to still make use of all training data, we trained five models on different stratified folds of the training data such that each training example is used once for validation and four times for training. For the final prediction, we use all five models and predict the three classes which yield the highest geometric mean over the models.

To fit the models we train the CNN_AUDIO and CNN_SPEC models from scratch. After that, the weights of these models are used for the feature computing parts of the CNN_COMB model and only the weights of the following dense layers are trained.

While the spectrogram is computed in advance the data augmentation described above is performant enough to be computed on the fly during the training. This saves disk-space and ensures a large amount of diverse training data.

When predicting on the test data we have to take into account that some audio tracks are longer than the desired model input and some tracks are shorter. For the longer tracks, it is not sufficient to only predict on one crop of the data because the main class might not be present in the selected crop. Therefore, we run the inference step on many crops of the audio where the step size is 5120 frames which is about 0.12 seconds and combine the predictions with the geometric mean. For shorter audio tracks the model might be able to recognize the class better in certain parts of the input. Therefore,

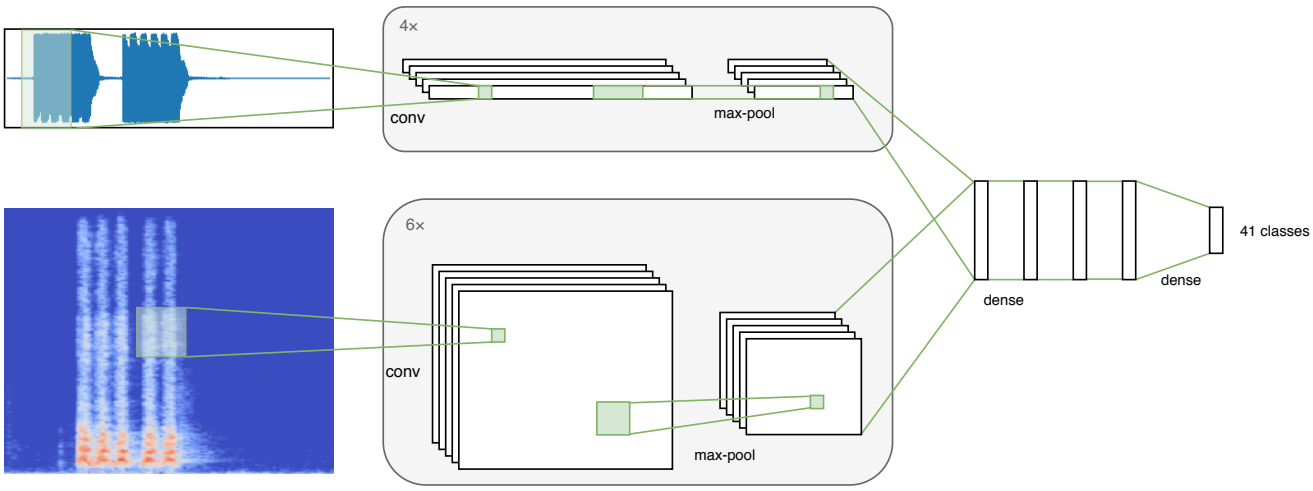

Figure 3: The architecture of the whole model.

we generate multiple inputs by padding the audio file with zeros such that the real audio appears in different positions of the input. Again we use a step size of 5120 frames.

3. EVALUATION

We trained all the described models for inputs of one, two and three seconds as described in section 2.5 and uploaded the predictions of them to Kaggle where they were scored with mean Average Precision at three on about 300 samples. The scores can be found in table 1.

We observe that the combined model performs significantly better than both single models. This suggests that the models make different errors and some relevant high-level features are easier to extract from the raw audio while others are easier to extract from the mel-scaled spectrogram.

The two-second combined model yields the best public leaderboard score and ranks in the top 2 percent of all participants.

4. JUDGES AWARD

To improve the computational efficiency of our model we can just use one of the five two-second combined models. Additionally, we can increase the step size of the crops to 51200 frames which are about 1.2 seconds.

With this step size and only using one model running inference on all 9397 test audio clips with a total length of almost 15 hours took only about 11 minutes on a Dell notebook with a Nvidia GTX 1050 GPU. Therefore, the prediction of one minute worth of audio clips only takes about 0.74 seconds. The model has a total number of 3,136,649 parameters.

While the computational time is greatly improved the model is still competitive on the public leaderboard with a mAP@3 score of **0.956**.

The corresponding submission CSV file is called `Wilhelm_UKON_task2_2.output.csv` and the Kaggle team name is “Deep Lake”.

5. CONCLUSION

In this report, we introduced a powerful and easy method for audio-tagging that adds the usage of the raw audio wave to classical approaches which only make use of the spectrogram and showed that this improves the accuracy of the model significantly. We demonstrate the possibilities of our model by submitting to the Freesound General-Purpose Audio Tagging Challenge and scoring in the top two percent of all participants.

6. ACKNOWLEDGMENT

We thank Christian Borgelt and Christoph Doell for their lecture which challenged us to take part in this competition.

7. REFERENCES

[1] 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.

[2] 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.

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

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] B. McFee, M. McVicar, S. Balke, C. Thomé, C. Raffel, O. Nieto, E. Battenberg, D. Ellis, R. Yamamoto, J. Moore, R. Bittner, K. Choi, F.-R. Stöter, S. Kumar, S. Waloschek, Seth, R. Naktinis, D. Repetto, C. F. Hawthorne, C. Carr, hojinlee,

Model	Crop length	Validation score (mAP@3) min	max	avg	Public leaderboard score (mAP@3)
CNN_AUDIO	1sec	0.879	0.901	0.887	0.920
	2sec	0.876	0.905	0.890	0.921
	3sec	0.895	0.894	0.884	0.935
CNN_SPEC	1sec	0.928	0.947	0.933	0.930
	2sec	0.932	0.944	0.932	0.950
	3sec	0.928	0.945	0.936	0.935
CNN_COMB	1sec	0.943	0.963	0.953	0.955
	2sec	**0.949**	0.958	**0.954**	**0.966**
	3sec	0.945	**0.966**	**0.954**	0.956

Table 1: Evaluation Results. For the validation score, the prediction was done by all five models on different subsets of the training data where the respective model wasn't trained on. For the public leaderboard score, the classes resulting from the geometric mean of all five models have been uploaded to Kaggle where they were evaluated on around 300 test samples.

W. Pimenta, P. Viktorin, P. Brossier, J. F. Santos, JackieWu, Erik, and A. Holovaty, "librosa/librosa: 0.6.0," Feb. 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1174893

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