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

Marcel Lederle[*]

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

Benjamin Wilhelm[*]

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

ABSTRACT

In this paper, we describe our contribution to Task 2 of the DCASE 2018 Audio Challenge [1]. While it has become ubiquitous to utilize an ensemble of machine learning methods for classification tasks to obtain better predictive performance, the majority of ensemble methods combine predictions rather than learned features. We propose a single-model method that combines learned high-level features computed from log-scaled mel-spectrograms and raw audio data. These features are learned separately by two Convolutional Neural Networks, one for each input type, and then combined by densely connected layers within a single network. This relatively simple approach along with data augmentation ranks among the best two percent in the Freesound General-Purpose Audio Tagging Challenge on Kaggle.

Index Terms— audio-tagging, convolutional neural network, raw audio, mel-spectrogram

1. INTRODUCTION

For humans, it seems to be effortless to associate sounds with events or categories that describe the perceived sound best. However, the complex structure and the large amount of information transmitted through sound makes it particularly difficult to extract that information automatically.

Recognizing a wide variety of sounds has many applications in our today's life. These include surveillance [2, 3, 4], acoustic monitoring [5], and automatic description of multimedia [6]. Due to the diversity of sounds belonging to the same category, a reliable recognition of manifold sound categories is still under ongoing research.

Carefully hand-crafted features such as Mel Frequency Cepstral Coefficients (MFCCs) were the dominant features used for speech recognition [7, 8] and music information retrieval [9], but the trend is now shifting toward deep learning [10]. The approach of manual feature engineering has drawbacks compared to deep learning based methods because it requires considerable effort and expertise to manually create features for a specific purpose. In particular, most of the engineered features, such as MFCCs and spectral centroids [11], are non-task specific, whereas all deep learning approaches are task-specific due to its formulation as a minimization process on task-specific training examples. Since Convolutional Neural Networks (CNNs) have shown a remarkable progress in visual recognition tasks [12] over the last years, it has become common to use CNNs for feature extraction and classification in the audio domain [13, 14]. Several CNN architectures, such as

AlexNet [15], VGG [16], ResNet [17], and Inception-v3 [18], have been proposed for image classification, which are also well suited to the task of audio tagging.

The goal of Task 2 of the DCASE 2018 Challenge [1] was to predict the category of an audio clip belonging to one out of 41 heterogeneous classes, such as "Acoustic guitar", "Bark", "Bus" and "Telephone", drawn from the AudioSet Ontology [19]. Training and testing data contain a diverse set of user-generated audio clips from Freesound (https://freesound.org) [1].

In this paper, we mainly focus on building an audio-tagging system that uses both the raw audio data and the corresponding mel-spectrogram rather than ensembling [20] or stacking [21] multiple classifiers.

2. METHOD

Our audio-tagging system comprises two separately trained Convolutional Neural Networks on raw audio and mel-spectrogram, respectively. The learned high-level features are then combined by a densely connected neural network to form the system. In the following, we describe each model in detail.

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 [16] or AlexNet [15], but with one-dimensional convolutions and one-dimensional max pooling. As described in Table 1, we use four blocks, each consisting of two convolutional and one max-pooling layer. The number of filters is increased in each consecutive block, while the kernel size is decreased. The pool-size of the max-pooling layers is chosen to quickly reduce the large time dimension. After each block and after the dense layer, we apply batch normalization [22], as experiments have shown that it reduces the training time and increases the model accuracy. To introduce nonlinearities into our network, we apply a ReLU activation function [23] after each convolutional layer and dense layer.

2.2. CNN on Mel-Spectrogram (*cnn-spec*)

The *cnn-spec* model is a two-dimensional convolutional neural network taking mel-spectrograms as input. The architecture is again similar to common image classification architectures and is described in detail in Table 2. As with the one-dimensional model, we apply batch normalization after each block and the dense layer and use the ReLU activation function after convolutional and dense layers.

---

[*]Both authors contributed equally to this work.

Layer	1sec shape	2sec shape	3sec shape
input	(44100, 1)	(88200, 1)	(132300, 1)
conv1d, 11, 32	(44100, 32)	(88200, 32)	(132300, 32)
conv1d, 11, 32	(44100, 32)	(88200, 32)	(132300, 32)
max-pool1d, 8/16	(5512, 32)	(5512, 32)	(8268, 32)
conv1d, 9, 64	(5512, 64)	(5512, 64)	(8268, 64)
conv1d, 9, 64	(5512, 64)	(5512, 64)	(8268, 64)
max-pool1d, 16	(344, 64)	(344, 64)	(516, 64)
conv1d, 7, 128	(344, 128)	(344, 128)	(516, 128)
conv1d, 7, 128	(344, 128)	(344, 128)	(516, 128)
max-pool1d, 16	(21, 128)	(21, 128)	(32, 128)
conv1d, 5, 256	(21, 256)	(21, 256)	(32, 256)
conv1d, 5, 256	(21, 256)	(21, 256)	(32, 256)
max-pool1d, 16	(1, 256)	(1, 256)	(2, 256)
dense, 512	(512)	(512)	(512)
softmax, 41	(41)	(41)	(41)

Table 1: The architecture of the *cnn-audio* model. Note that for the one-second model, the first max-pooling layer uses a pool-size of 8, while for other models a pool-size of 16 is used.

The mel-spectrogram is extracted using librosa [24] with the original sampling frequency of 44.1 kHz, 2048 FFT points, 128 mel-bins, and a hop-length of 256. The amplitude of the mel-spectrogram is scaled logarithmically, and the scaled mel-spectrogram is resized in time dimension to fit the model input size.

2.3. Joining CNNs (*cnn-comb*)

We remove the softmax and dense layer of both the trained *cnn-audio* and *cnn-spec* model and then concatenate the output features of the previous layer of both models so as to join them. The concatenated features are then connected to a densely connected neural network with four hidden layers. The hidden dense layers have 512, 256, 256, and 128 neurons, respectively. The complete model is illustrated in Figure 1.

We train *cnn-audio* and *cnn-spec* from scratch. Afterward, the weights of these models are transferred to the *cnn-comb* model and only the newly added dense layers are trained. The splitting of the training of *cnn-comb* into three steps, facilitates the procedure.

2.4. Data Augmentation

To prevent our model from overfitting, we make use of extensive data augmentation during training (time shifting, cropping, padding, and blending clips of same and different categories). Each of these augmentation techniques is applied to the raw audio wave and the mel-spectrogram. In the remaining section, we explain the augmentation methods based on the raw audio wave.

First, we apply a uniformly random time shift to the audio clip. To ensure that the audio clips fit the size of the model input, crops are taken from too long audio files and too short audio files are padded. For audio clips that are longer than the model input size, we use a crop with the size of the model input taken from a random position. If an audio sample fits multiple times (n times) in the input size, it is replicated such that it appears $k \in \{1, \ldots, n\}$ times with

Layer	1sec shape	2sec shape	3sec shape
input	(128, 170, 1)	(128, 300, 1)	(128, 400, 1)
conv2d, 4×4, 64	(128, 170, 64)	(128, 300, 64)	(128, 400, 64)
conv2d, 4×4, 64	(128, 170, 64)	(128, 300, 64)	(128, 400, 64)
max-pool2d, 1×1/2×2	(128, 170, 64)	(64, 150, 64)	(64, 200, 64)
conv2d, 4×4, 64	(128, 170, 64)	(64, 150, 64)	(64, 200, 64)
max-pool2d, 2×2	(64, 85, 64)	(32, 75, 64)	(32, 100, 64)
conv2d, 3×3, 128	(64, 85, 128)	(32, 75, 128)	(32, 100, 128)
max-pool2d, 2×4	(32, 21, 128)	(16, 18, 128)	(16, 25, 128)
conv2d, 3×3, 128	(32, 21, 128)	(16, 18, 128)	(16, 25, 128)
max-pool2d, 2×2	(16, 10, 128)	(8, 9, 128)	(8, 12, 128)
conv2d, 3×3, 256	(16, 10, 256)	(8, 9, 256)	(8, 12, 256)
max-pool2d, 2×2	(8, 5, 256)	(4, 4, 256)	(4, 6, 256)
conv2d, 3×3, 256	(8, 5, 256)	(4, 4, 256)	(4, 6, 256)
max-pool2d, 2×2	(4, 2, 256)	(2, 2, 256)	(2, 3, 256)
dense, 256	(256)	(256)	(256)
softmax, 41	(41)	(41)	(41)

Table 2: The architecture of the *cnn-spec* model. Note that the first max-pooling layer does not exist in the case of the one-second model.

a probability of $1/n$, and the remaining space before, after, and in between replications is filled with zeros.

Additionally, we enhance the robustness of our model by blending multiple audio clips of same or different categories. This method is referred to as mixup [25]. We blend them by assigning a random weight to each sample (weights sum up to one) and taking the weighted sum. If the blended samples are of the same class, the model should still predict the common class for the newly generated training sample. In the other case, if the blended samples are of distinct classes, the model is trained to predict the weight of each included class.

While the spectrogram is computed in advance, the computationally inexpensive data augmentation techniques can be computed on-the-fly during training. This saves disk-space and guarantees a large amount of diverse training data.

2.5. Implementation Details

We implemented the described method using Keras [26] in Python.

To monitor overfitting and the model performance during training, we exclude a part of the training data as validation data. To still make use of all training data, we train five models on 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 accumulate the predictions of all five models using the geometric mean.

All models are trained using the Adam optimizer [27] with a fixed learning rate of 0.001 and a mini-batch size of 32 for a maximum of 300 epochs, but stopping earlier if the validation loss hasn't improved for 35 epochs. We use the categorical cross-entropy loss function and weight the loss according to the distribution of training examples per class, thereby ensuring that the models pay more attention to samples from an under-represented class.

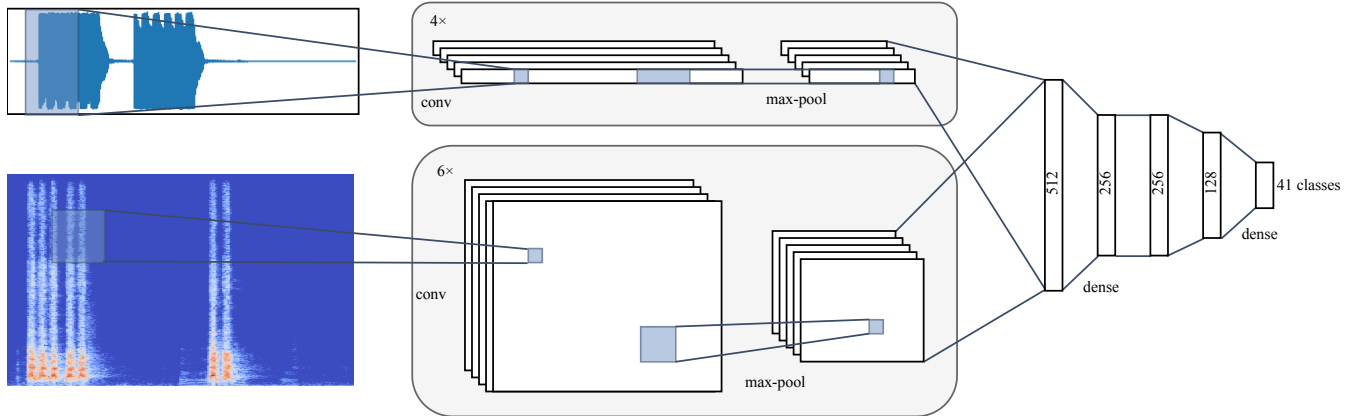Because many clips contain silence, we cut off silent parts at

Figure 1: Illustrated architecture of the complete model.

the beginning and at the end of an audio clip that do not exceed a volume of 40 decibel.

When predicting the test data, we have to take the varying length of audio files into account. It is not sufficient to only predict on one crop of too long tracks because important features might not be present in the selected crop. Therefore, we run the inference on many crops of the audio file with a step size of 5120 frames, which is approximately 0.12 seconds. For too short audio tracks, the model might be able to better recognize class-specific features in certain parts of the input. Therefore, we generate multiple inputs by padding the audio file with zeros such that the real audio appears at different positions in the input. Again, we use a step size of 5120 frames. Multiple predictions for one audio file are combined by means of the geometric mean.

3. EVALUATION

3.1. Dataset

We evaluate our method on the dataset provided for Task 2 of the DCASE 2018 Challenge [1], which comprises 9473 training and 1600 test samples. The test data set has been manually verified, whereas the training data features labels of different reliability. Each mono audio file has a bit-depth of 16, a sampling rate of 44.1 kHz, and is associated with one out of 41 classes of the AudioSet Ontology [19].

The class distribution of both training and test set is not balanced and ranges from 94 to 300 and from 25 to 110 samples per class, respectively. The duration of the shortest audio file is 300 ms and 30 seconds for the longest clip, while the average length is 6.8 seconds for the train set and 5.2 seconds for the test set.

3.2. Metric

The Challenge uses mean Average Precision at three (mAP@3) for evaluating test results, which allows up to three predictions per audio clip. Full credit is given if the first prediction matches the label of the clip, while less credit is given if one of the other predictions is correct. The evaluation metric is defined as

$$\text{mAP@3} = \frac{1}{U} \sum_{i=1}^{U} \sum_{j=1}^{\min(3,n_i)} \frac{[\![y_{ij} = \hat{y}_i]\!]}{j},$$

Model	Crop length	Public score (mAP@3)	Private score (mAP@3)	Total (mAP@3)
cnn-audio	1sec	0.920	0.888	0.894
	2sec	0.921	0.884	0.891
	3sec	0.935	0.889	0.898
cnn-spec	1sec	0.930	0.923	0.924
	2sec	0.950	0.928	0.932
	3sec	0.935	0.930	0.931
cnn-comb	1sec	0.955	0.939	0.942
	2sec	**0.966**	**0.944**	**0.948**
	3sec	0.956	**0.944**	0.946

Table 3: Evaluation results of the individual models on the public (301 samples), private (1299 samples), and full test set.

where U is the total number of scored audio files, y_{ij} is the predicted label for file i at position j, $\hat{y}_i$ is the ground-truth label for file i, n_i is the total number of predicted labels for file i, $[\![\texttt{True}]\!] = 1$, $[\![\texttt{False}]\!] = 0$. No label may be predicted multiple times for one audio file.

3.3. Results

We have trained all models on inputs of one, two, and three seconds, as described in Section 2.5, and evaluated the model performance on the test set (see Table 3). We have observed that for each crop length, the combined model performs significantly better compared to models with a single input. Combined models with an input size of two and three seconds, perform best and rank in the upper two percent on the Private Leaderboard on Kaggle.

Additionally, we determined the per-category mAP@3 on the complete test set showing that some classes are more challenging to predict than others (see Table 4). Our model primarily struggles with the classes "Squeak", "Telephone" and "Fireworks", but it still beats the baseline system [1] in every per-category mAP@3 score.

To verify that the performance gain of the combined model results from combining the extracted high-level features from both models, we compare *cnn-audio* and *cnn-spec* to the *cnn-comb*

Name	samples	time	mAP@3	Name	samples	time	mAP@3	Name	samples	time	mAP@3
Acoustic_guitar	300	52.2	0.893	Electric_piano	150	25.5	1.000	Microwave_oven	146	25.1	0.966
Applause	300	58.2	1.000	Fart	300	18.6	0.944	Oboe	299	15.3	0.976
Bark	239	44.6	0.982	Finger_snapping	117	5.9	1.000	Saxophone	300	33.7	0.942
Bass_drum	300	12.8	1.000	Fireworks	300	48.2	0.786	Scissors	95	15.7	0.927
Burping_or_eructation	210	11.7	1.000	Flute	300	46.2	1.000	Shatter	300	26.1	0.960
Bus	109	28.4	0.953	Glockenspiel	94	8.4	0.856	Snare_drum	300	17.9	0.912
Cello	300	37.3	0.951	Gong	292	41.8	0.968	Squeak	300	38.2	0.603
Chime	115	23.8	0.891	Gunshot_or_gunfire	147	11.1	0.950	Tambourine	221	10.1	0.975
Clarinet	300	34.7	0.991	Harmonica	165	18.6	0.970	Tearing	300	38.7	0.981
Computer_keyboard	119	23.0	1.000	Hi-hat	300	18.6	0.957	Telephone	120	16.2	0.788
Cough	243	22.4	1.000	Keys_jangling	139	18.8	0.929	Trumpet	300	28.3	0.959
Cowbell	191	10.9	1.000	Knock	279	19.6	0.957	Violin_or_fiddle	300	26.6	0.986
Double_bass	300	16.9	0.946	Laughter	300	36.3	0.974	Writing	270	48.3	0.948
Drawer_open_or_close	158	18.0	0.925	Meow	155	18.7	1.000				

Table 4: Per category mAP@3 score of the *cnn-comb* 2sec model on the full test set and the number of samples along with time in minutes of the respective class in the train set.
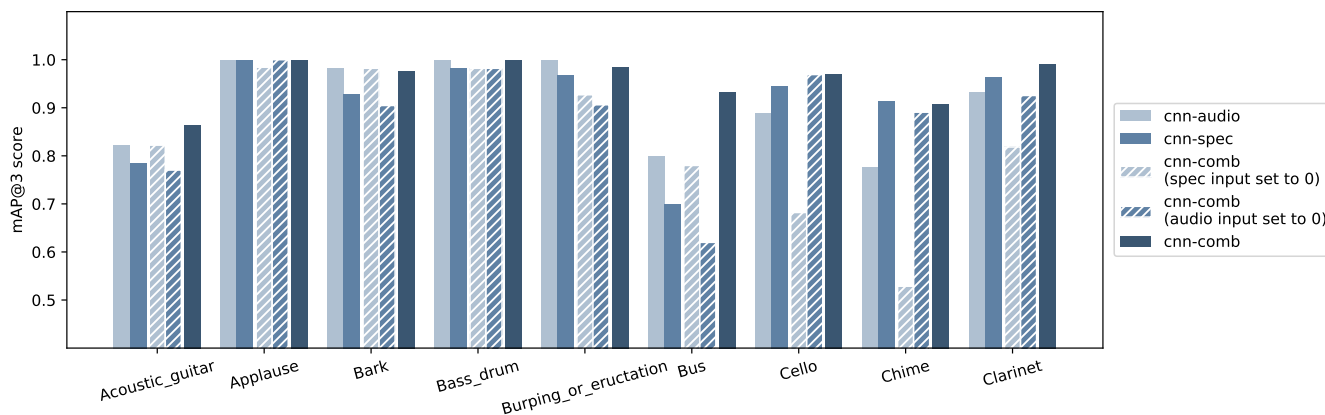

Figure 2: Comparison of per-category scores of single input models, combined models with one input alternately set to zero, and the combined model with both inputs. The mAP@3 score is reported on a single fold for each model.

model, where one of its inputs is set to zero (See Figure 2).

For categories in which the single *cnn-audio* model outperforms the single *cnn-spec* model, the combined model performs better if the audio input is present and not set to zero. Otherwise, if *cnn-spec* outperforms *cnn-audio*, *cnn-comb* has a higher score if the mel-spectrogram is present and not set to zero. Setting either the mel-spectrogram or the audio wave to zero forces the *cnn-comb* model to make predictions based on a single input.

cnn-comb performs equally or better than *cnn-comb* with one single input set to zero because it has learned to utilize meaningful high-level features of both inputs jointly, which are not given by zeroed inputs. For the same reason, *ccn-comb* with one zeroed input usually performs worse than the corresponding model with a single input.

We conclude that *cnn-comb* makes use of the learned high-level features from both *cnn-audio* and *cnn-spec*, but it focuses more on features belonging to the superior single model.

4. CONCLUSION

In this paper, we have proposed a method for audio-tagging that extends current Convolutional Neural Network approaches that only make use of a frequency representation by adding a second input that incorporates the raw audio wave. Adding the additional input, has improved the mAP@3 score significantly. We have demonstrated the capabilities of our model by competing in the Freesound General-Purpose Audio Tagging Challenge on Kaggle and ranking in the top two percent of all participants.

5. ACKNOWLEDGMENT

6. 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] A. Harma, M. F. McKinney, and J. Skowronek, "Automatic surveillance of the acoustic activity in our living environment," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*. IEEE, 2005, pp. 4–pp.

[3] M. Crocco, M. Cristani, A. Trucco, and V. Murino, "Audio surveillance: a systematic review," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, p. 52, 2016.

[4] G. Valenzise, L. Gerosa, M. Tagliasacchi, F. Antonacci, and A. Sarti, "Scream and gunshot detection and localization for audio-surveillance systems," in *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*. IEEE, 2007, pp. 21–26.

[5] S. Goetze, J. Schroder, S. Gerlach, D. Hollosi, J.-E. Appell, and F. Wallhoff, "Acoustic monitoring and localization for social care," *Journal of Computing Science and Engineering*, vol. 6, no. 1, pp. 40–50, 2012.

[6] Q. Jin and J. Liang, "Video description generation using audio and visual cues," in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. ACM, 2016, pp. 239–242.

[7] C. Ittichaicharoen, S. Suksri, and T. Yingthawornsuk, "Speech recognition using MFCC," in *International Conference on Computer Graphics, Simulation and Modeling (ICGSM'2012) July*, 2012, pp. 28–29.

[8] B. Logan *et al.*, "Mel frequency cepstral coefficients for music modeling." in *ISMIR*, vol. 270, 2000, pp. 1–11.

[9] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.

[10] O. Abdel-Hamid, L. Deng, and D. Yu, "Exploring convolutional neural network structures and optimization techniques for speech recognition." in *Interspeech*, vol. 2013, 2013, pp. 1173–5.

[11] A. J. Eronen, V. T. Peltonen, J. T. Tuomi, A. P. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi, "Audio-based context recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 321–329, 2006.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[13] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[14] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*. IEEE, 2015, pp. 1–6.

[15] 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.

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

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] 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.

[19] 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.

[20] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[21] S. Džeroski and B. Ženko, "Is combining classifiers with stacking better than selecting the best one?" *Machine learning*, vol. 54, no. 3, pp. 255–273, 2004.

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[23] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[24] 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, 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

[25] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

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

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