CONVOLUTIONAL RECURRENT NEURAL NETWORK AND DATA AUGMENTATION FOR AUDIO TAGGING WITH NOISY LABELS AND MINIMAL SUPERVISION

Janek Ebbers, Reinhold Haeb-Umbach

Paderborn University, Department of Communications Engineering, Paderborn, Germany
{ebbers, haeb}@nt.upb.de

ABSTRACT

In this paper we present our audio tagging system for the DCASE 2019 Challenge Task 2. We propose a model consisting of a convolutional front end using log-mel-energies as input features, a recurrent neural network sequence encoder and a fully connected classifier network outputting an activity probability for each of the 80 considered event classes. Due to the recurrent neural network, which encodes a whole sequence into a single vector, our model is able to process sequences of varying lengths. The model is trained with only little manually labeled training data and a larger amount of automatically labeled web data, which hence suffers from label noise. To efficiently train the model with the provided data we use various data augmentation to prevent overfitting and improve generalization. Our best submitted system achieves a label-weighted label-ranking average precision (lwlrap) of 75.5% on the private test set which is an absolute improvement of 21.7% over the baseline. This system scored the second place in the teams ranking of the DCASE 2019 Challenge Task 2 and the fifth place in the Kaggle competition "Freesound Audio Tagging 2019" with more than 400 participants. After the challenge ended we further improved performance to 76.5% lwlrap setting a new state-of-the-art on this dataset.

Index Terms— audio tagging, label noise, data augmentation

1. INTRODUCTION

Environmental sound recognition has recently attracted increased interest, not only in academia. Numerous commercial applications can benefit from a reliable acoustic scene analysis, such as ambient assisted living, autonomous driving and various monitoring and diarization tasks. Within environmental sound recognition the tasks of Sound Event Detection (SED), Audio Tagging and Acoustic Scene Classification (ASC) [1] can be distinguished, which differ in the level of detail obtained about the acoustic environment. While SED makes predictions at frame level, Audio Tagging and ASC make predictions at sequence level.

Because frame level annotations (so-called *strong labels*) are difficult and time-consuming to obtain, current large-scale datasets, such as Google's AudioSet [2], only provide sequence level labels (*weak labels*). Further, for many applications frame level predictions are not required, which, together with the availability of weakly labeled large-scale datasets, results in an increased popularity of Audio Tagging.

Although weak annotations are easier to obtain than strong annotations, they still require human annotators. To avoid this necessity completely, one line of research is devoted to develop semi-supervised approaches which directly benefit from unlabeled data [3, 4], which is abundant. However, there are also tremendous amounts of data coming with other modalities and meta-data, which can potentially be exploited to derive labels automatically. In the dataset considered in this contribution, labels had been generated for the web data using video-level predictions. Note, that those automatically generated labels certainly contain errors, which is why they are called *noisy* labels. Until now there are only few works addressing the impact of noisy labels for sound recognition [5].

Data augmentation is another common approach to increase the amount of labeled training data and improve generalization. It has been shown to improve classifier performance on many tasks, including speech recognition [6] and audio classification [7, 8, 9].

In this contribution we tackle the DCASE 2019 Challenge Task 2 [10], where the main research objective is the following: How can we train a high performance Audio Tagging system given relatively little data with reliable labels but a larger amount of mismatched data with noisy labels? We primarily tackle this by exploring different methods for data augmentation, while we spend less time on neural network architecture tuning. We show that frequency warping and time and frequency masking for data augmentation significantly improve performance. Further, our experiments tend to suggest, that our model is robust against label noise, as it achieves state-of-the-art performance without any particular treatment of label noise. While the methods investigated for dealing with label noise did individually not result in classifier improvements, they nevertheless increased the diversity of the models trained, resulting eventually in performance improvement through system combination. Please note that our system is publicly available on github.[1]

The rest of the paper is structured as follows. After briefly describing the considered task in Section 2, we outline our neural network architecture in Section 3. After presenting the data augmentation methods in Section 4, our training procedure and experiments are given in Sections 5 and 6, respectively. Conclusions are drawn in Section 7.

2. TASK DESCRIPTION

The DCASE 2019 Challenge Task 2 "Audio Tagging with noisy labels and minimal supervision" [10] is a follow-up of the DCASE 2018 Challenge Task 2 "General-purpose audio tagging of Freesound content with AudioSet labels" [11]. As in the 2018 edition the challenge was hosted on Kaggle[2] with more than 400 participants. The provided dataset "FSDKaggle2019" consists of two subsets: a small *curated* set of 4970 manually labeled audio clips and a *noisy* set of 19815 audio clips where labels were automatically derived from video-level predictions from a variety of

This work has been supported by Deutsche Forschungsgemeinschaft under contract no. HA 3455/15-1 within the Research Unit FOR 2457 (acoustic sensor networks).

[1] https://github.com/fgnt/upb_audio_tagging_2019

[2] https://www.kaggle.com/c/freesound-audio-tagging-2019

https://doi.org/10.33682/57xx-t679

pre-trained audio models. A vocabulary of 80 sound events is used where multiple events can be active at a time resulting in a multi-label classification problem.

3. MODEL

3.1. Feature Extraction

First, we perform an STFT with a frame length of $40\,\mathrm{ms}$ (1764 samples) and a hop size of $20\,\mathrm{ms}$ (882 samples) on the provided $44.1\,\mathrm{kHz}$ audio signals without resampling. For each frame we then extract 128 log-mel-band-energy features with f_{min}=$50\,\mathrm{Hz}$ and f_{max}=$16\,\mathrm{kHz}$. Lastly, we substract the global mean of each feature and then divide by the global standard deviation over all features.

3.2. Neural Network Architecture

Our proposed deep learning based model is outlined in Table 1. The neural network consists of a convolutional (conv.), a recurrent, and a fully connected module. We expect a four dimensional input to our model of shape $B{\times}C{\times}F{\times}N_m$ with B, C, F, N_m being the mini-batch size, number of channels, number of features and number of frames in the m-th input signal, respectively, where C=1 and F=128 are fix. In the following the signal index m is neglected.

The convolutional module combines a 2d CNN and a 1d CNN. The 2d CNN consists of five conv. blocks, with each block comprising one or two conv. layers and a max pooling layer. While the first four blocks have two conv. layers the last block only has a single one. In each block the number of channels is doubled starting from 16 while the number of features are halfed by max pooling. The number of time steps are also halfed in the first three blocks while being unchanged in the last two blocks. This results in an output of shape $B{\times}C'{\times}F'{\times}N'$ with C'=256, F'=4 and $N' = \lceil \frac{N}{8} \rceil$. Each 2d conv. layer uses a kernel size of 3×3 and is followed by batch normalization and ReLU activation.

While the 2d CNN is meant to extract high-level feature maps from the log-mel-band-energy spectrogram, the 1d CNN (or TDNN) is meant to provide holistic representations by jointly processing all frequencies and channels of adjacent frames. Therefore it takes the reshaped output ($B \times C'{\cdot}F' \times N'$) of the 2d CNN as input and applies three 1d conv. layers with 256 hidden channels each. Each 1d conv. layer uses a kernel size of 3 and is followed by batch normalization and ReLU activation.

The output of the CNN is then fed into a recurrent sequence encoder. We use two layers of Gated Recurrent Units (GRUs) with 256 units per layer. Only the last output vector of each sequence in a batch is forwarded to the classification network.

The fully connected classification network conists of one hidden layer with 256 hidden units and ReLU activation function and the final classification layer with Sigmoid activation outputting scores between 0 and 1 for each of the 80 target event classes.

Table 1: Convolutional Recurrent Neural Network for Audio Tagging with output shapes of each block. Each ConvXd uses a kernel size of three and a stride of one and includes BatchNorm and ReLU.

Block	output shape
LogMel(128)	$B{\times}1{\times}128{\times}N$
GlobalNorm	$B{\times}1{\times}128{\times}N$
2×Conv2d(16)	$B{\times}16{\times}128{\times}N$
Pool2d(2×2)	$B{\times}16{\times}64{\times}\lceil N/2 \rceil$
2×Conv2d(32)	$B{\times}32{\times}64{\times}\lceil N/2 \rceil$
Pool2d(2×2)	$B{\times}32{\times}32{\times}\lceil N/4 \rceil$
2×Conv2d(64)	$B{\times}64{\times}32{\times}\lceil N/4 \rceil$
Pool2d(2×2)	$B{\times}64{\times}16{\times}\lceil N/8 \rceil$
2×Conv2d(128)	$B{\times}128{\times}16{\times}\lceil N/8 \rceil$
Pool2d(2×1)	$B{\times}128{\times}8{\times}\lceil N/8 \rceil$
Conv2d(256)	$B{\times}256{\times}8{\times}\lceil N/8 \rceil$
Pool2d(2×1)	$B{\times}256{\times}4{\times}\lceil N/8 \rceil$
Reshape	$B{\times}1024{\times}\lceil N/8 \rceil$
3×Conv1d(256)	$B{\times}256{\times}\lceil N/8 \rceil$
2×GRU(256)	$B{\times}256$
fc_{ReLU}(256)	$B{\times}256$
$\text{fc}_{\text{Sigmoid}}$(80)	$B{\times}80$

4. DATA AUGMENTATION

Because there is only little data available, efficient data augmentation is crucial to prevent overfitting and improve generalization capabilities of the system. In the following we outline the data augmentation methods that we combined during model training. All augmentation methods are performed on the fly during training yielding an extremely large number of possible training samples.

4.1. Mixup

Mixup [12] is a data augmentation technique originating from classification tasks where a new training sample is generated as a weighted average of two samples from the dataset:

$$\tilde{\mathbf{x}}_i = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j,\ \lambda \in [0,1].$$

Similarly their one-hot encoded targets are combined to a soft target vector:

$$\tilde{\mathbf{y}}_i = \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j.$$

Although for classification tasks mixup results in ambiguous samples (which probably wouldn't make a lot of sense to a human either) it has shown to improve generalization and robustness of the trained classifier networks. Mixup has successfully been used for general purpose audio tagging, e.g., in [9] in the DCASE 2018 Challenge.

For audio tagging (as opposed to classification) the input audio may already be a superposition of multiple sources. Thus, mixing two or more audio signals together yields a new valid audio signal with an increased number of active events. Therefore, instead of building a weighted average we superpose two waveforms as follows:

$$\tilde{x}_i(t) = \lambda_0 x_i(t) + \gamma \lambda_1 \frac{\max(|x_i|)}{\max(|x_j|)} x_j(t-\tau)$$

with

$$\begin{aligned}
&\gamma \sim \mathcal{B}(2/3),\\
&\tau \sim \mathcal{U}(\max(-T_j, T_i - 30\,\mathrm{s}), \min(T_i, 30\,\mathrm{s} - T_j)),\\
&T_j \leq 1.1 \cdot T_i,\\
&\lambda_m = a_m^{2b_m - 1};\ a_m \sim \mathcal{U}(1,2);\ b_m \sim \mathcal{B}(1/2);\ m{\in}\{0,1\}
\end{aligned}$$

where $\mathcal{B}$ denotes the Bernoulli distribution. Putting this equation into words we

- perform mixup only with a probability of $2/3$,
- only mixup signals which are shorter than 1.1 times the base signal x_i,
- allow mixup to lengthen the signal as long as it does not exceed the maximum length of $30\,\mathrm{s}$,
- normalize signals to the maximum value of the base signal x_i,
- attenuate or amplify each normalized signal by a random factor.

We also do not build a weighted average of the individual targets, but simply combine all tags into a single n-hot encoded target $\tilde{\mathbf{y}}_i$.

4.2. Frequency Warping

Recently, SpecAugment [13] was introduced as a simple yet efficient augmentation method of log-mel spectrograms for automatic speech recognition. It uses three different distortions namely time warping, frequency masking, and time masking. In our experiments, however, we found that for audio tagging warping the spectrogram on the frequency axis yielded better performance than time warping. Hence, we exchange the time warping by frequency warping in our version of SpecAugment which is explained in this and the following two sections.

We consider the log-mel spectrogram as an image here with width T and height F. Warping the vertical (frequency) axis of the image is controlled by the cutoff frequency

$$f_\mathrm{c} \sim \mathcal{E}(0.5 \cdot F),$$

where $\mathcal{E}$ denotes the exponential distribution parameterized by the scale $\beta = 0.5 \cdot F$, and by the warping factor

$$\alpha = (1+u)^{2s-1}; \quad u \sim \mathcal{E}(0.07);\ s \sim \mathcal{B}(1/2).$$

Fig. 1 shows the resulting piece-wise linear warping function. Note that f_c can be larger than F, which results in pure stretching/compressing the whole spectrogram.

It is worth noting that the frequency warping performed here is very similar to Vocal Tract Length Pertubation [14].

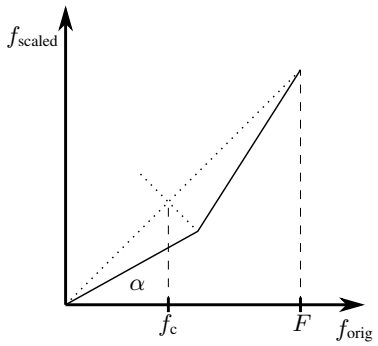

Figure 1: piece-wise linear frequency warping function.

4.3. Frequency Masking

We randomly mask H consecutive mel frequencies in the range $[f_0, f_0 + H]$, where H and f_0 are drawn from uniform distributions

$$H \sim \mathcal{U}(0, H_\mathrm{max})$$
$$f_0 \sim \mathcal{U}(0, F - H)$$

with $H_\mathrm{max} = 16$.

4.4. Time Masking

We randomly mask W consecutive time frames in the range $[n_0, n_0 + W]$, where W and n_0 are drawn from uniform distributions

$$W \sim \mathcal{U}(0, \min(W_\mathrm{max}, p \cdot N))$$
$$n_0 \sim \mathcal{U}(0, T - W)$$

with $W_\mathrm{max} = 70$ and $p = 0.2$.

5. TRAINING

5.1. Data

For training we use both the data with curated labels as well as the data with noisy labels.

As the provided train portion with noisy labels primarily contains audio signals of length 15 s and our model processes whole sequences, there is a bias towards sequences with a length of 15 s. To overcome this bias, we generate, before starting training, new audio excerpts of varying lengths by randomly splitting each of the audio signals with noisy labels into two at length $r_m \cdot T_m$ with T_m being the length of the m-th signal and $r_m \sim \mathcal{U}(0.1, 0.9)$. This results in audio excerpt lengths which are approximately uniformly distributed between 1 s and 13.5 s. As mixup data augmentation may lengthen audio signals, mixup of the noisy excerpts yields signals distributed between 1 s and 27 s. Each audio excerpt copies the event tags of the original audio, which results in some additional label noise. The random splitting is performed three times resulting in three different datasets which we refer to as splits 0-2 in the following[3].

We mixup curated only data which we refer to as the curated portion in the following as well as we mixup combined curated and noisy data which we refer to as the noisy portion in the following. In each training epoch the curated portion is repeated an integer number of times to prevent training from being dominated by noisy labels. The ratio of noisy labels to the total number of labels in one epoch is referred to as $R = \frac{M_\mathrm{noisy}}{M_\mathrm{total}}$.

5.2. Optimization

The training criterion is the binary cross entropy between the model predictions $\hat{\mathbf{y}}$ and the n-hot target vector $\tilde{\mathbf{y}}$:

$$L(\hat{\mathbf{y}}, \tilde{\mathbf{y}}) = -\sum_{k=0}^{K-1} \Big(\tilde{y}_k \log(\hat{y}_k) + (1 - \tilde{y}_k) \log(1 - \hat{y}_k) \Big)$$

with $K = 80$ denoting the number of target event classes.

We randomly sample mini batches of size 16 from the training data such that

1. no signal in the mini batch is padded by more than 20%,
2. no example in the mini batch includes the same events as another example in the same minibatch,

and compute gradients of the average loss in the mini batch. We clip gradients at a threshold of 15. Adam [15] was used for optimization. Training is performed for 200K iterations with a learning rate of $3 \cdot 10^{-4}$.

We perform Stochastic Weight Averaging (SWA) [16] for the last 50K iterations with a frequency of 1K iterations. At the end of training we exchange the model weights for the averaged weights. Finally we update the statistics of all batch normalization layers by making a forward pass on our (unaugmented) training data using the SWA model. SWA has shown to improve generalization and hence performance on unseen data [16]. Another advantage is that with SWA there is no need for held-out data to determine the best performing checkpoint.

6. EXPERIMENTS

In the following we evaluate the usefulness of the proposed data augmentation techniques and different methods for label noise handling. While Section 6.1 and Section 6.2 only report the performance of single model systems (which were trained on split 0),

[3]The generated splits are available at `https://github.com/fgnt/upb_audio_tagging_2019`

Section 6.3 presents performance of ensembles combining models trained on different splits.

System performance is measured in terms of label-weighted label-ranking average precision (lwlrap) evaluated on the private test set. The lwlrap metric measures the capablity of the model to rank active events over non-active events (see [10] for details). The reported scores are one-shot results and were obtained from (late) submissions on the Kaggle competition website.

6.1. Data Augmentation

To evaluate the different data augmentation methods we trained single model systems using curated and noisy labeled data using the provided labels. In this section, all models were trained with a noisy label rate of R=0.56. Table 2 shows the performance gains due to adding the proposed data augmentation methods. It can be seen that each augmentation method significantly increases the model performance in terms of lwlrap on the private test set.

Table 2: Performance when gradually adding data augmentation methods.

Data Augmentation	lwlrap
-	0.611
Mixup	0.665
+ Freq. Warp.	0.701
+ Freq. & Time Mask.	0.721

6.2. Label Noise Handling

In this Section we evaluate different methods for dealing with label noise when using all proposed data augmentation techniques. While for our challenge submission we used relabeling, which is explained first, we adopted Multi Task Learning [17] from the winning team after the challenge has ended, which is explained second.

6.2.1. Relabeling

For each of the three splits (as explained in Section 5.1) we trained models on five different folds using a noisy label rate of $R = 0.5$. This results in a total of 15 different models which were all combined into an ensemble to make predictions for the noisy excerpts from all three splits. Here the event scores of the individual models were averaged to obtain the ensemble output scores.

For each event we then determined the decision threshold yielding the best error rate jointly evaluated on the set of all noisy excerpts from all splits. These decision thresholds were used to relabel the noisy excerpts, where excerpts without any active event were discarded.

We then used the whole relabeled data of a split for training a new model as we do not need held-out data to determine the best checkpoint due to SWA.

6.2.2. Multi Task Learning

The winning team of the DCASE 2019 Challenge Task 2 proposed Multi Task Learning (MTL) to deal with noisy labels [17]. Here different classifier layers are used during training to predict curated and noisy labels, respectively. After the challenge ended we adopted this approach for our model, i.e., we trained different fully connected layers (the last two layers in Table 1) for curated and noisy labels. At test time only the classifier layers trained on the curated labels were used to make predictions.

6.2.3. Results

Results for the different methods of label noise handling are shown in Table 3. It can be seen that using the noisy labels results in a significant performance gain. However, using relabeling and MTL yield only small improvement of the lwlrap. In the next section these methods are further evaluated when performing ensembling.

Table 3: Performance for different treatments of label noise.

Method	R	lwlrap
Curated only	0.00	0.687
Provided labels	0.56	0.721
Relabeled	0.62	0.722
MTL	0.56	0.724

6.3. Ensembling

Finally, we evaluate ensembles of different models. In particular, we combine models trained on different splits and with different noisy label rates. System combination is achieved by averaging the output scores of the contributing systems. Each row in Table 4-6 adds three models which were trained on the three different splits. The results marked by an asterisk in Table 5 represent our submissions to the challenge. It can be seen that neither relabeling nor multi-task learning individually improve performance over the provided labels. Combining all the models from Tables 4-6 into a large ensemble of 27 models, however, raises performance to 76.5% lwlrap, setting a new state-of-the-art for this task as shown in Table 7. Do note that this ensemble has a total excecution time < 2 h using CPU, which meets the challenge constraints.

Table 4: Provided labels

#models	R	lwlrap
3	0.56	-
+3	0.64	-
+3	0.75	0.759

Table 5: Relabeled

#models	R	lwlrap
3	0.62	0.746*
+3	0.77	0.755*
+3	0.53	0.757

Table 6: MTL

#models	R	lwlrap
3	0.56	-
+3	0.64	-
+3	0.75	0.759

Table 7: System Comparison

System	lwlrap
Baseline [10]	0.537
DCASE winner [17]	0.758
Kaggle winner[4]	0.760
Our best subm. [18]	0.755
Our best late subm.	**0.765**

7. CONCLUSIONS

In this paper we presented our system for the DCASE 2019 Challenge Task 2. Our experiments carried out on the "FSDKaggle2019" data highlight the importance of data augmentation techniques for achieving high classification performance. On the other hand, particular consideration of the label noise did not prove effective in our case. Furthermore, performance was boosted by system combination raising the performance from 0.724 lwlrap of our best single model system to 0.765 by combining a total of 27 models, setting a new state of the art for this task.

[4] https://www.kaggle.com/c/freesound-audio-tagging-2019/leaderboard

8. REFERENCES

[1] T. Virtanen, M. D. Plumbley, and D. Ellis, *Computational analysis of sound scenes and events*. Springer, 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 *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[3] R. Serizel, N. Turpault, H. Eghbal-Zadeh, and A. P. Shah, "Large-scale weakly labeled semi-supervised sound event detection in domestic environments," *arXiv preprint arXiv:1807.10501*, 2018.

[4] N. Turpault, R. Serizel, A. Parag Shah, and J. Salamon, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," June 2019, working paper or preprint. [Online]. Available: https://hal.inria.fr/hal-02160855

[5] E. Fonseca, M. Plakal, D. P. Ellis, F. Font, X. Favory, and X. Serra, "Learning sound event classifiers from web audio with noisy labels," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 21–25.

[6] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. Sainath, and M. Bacchiani, "Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home," in *Proc. INTERSPEECH*, 2017, pp. 379–383.

[7] 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.

[8] T. Inoue, P. Vinayavekhin, S. Wang, D. Wood, N. Greco, and R. Tachibana, "Domestic activities classification based on cnn using shuffling and mixing data augmentation," *DCASE 2018 Challenge*, 2018.

[9] I.-Y. Jeong and H. Lim, "Audio tagging system using densely connected convolutional networks," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, 2018.

[10] E. Fonseca, M. Plakal, F. Font, D. P. Ellis, and X. Serra, "Audio tagging with noisy labels and minimal supervision," *arXiv preprint arXiv:1906.02975*, 2019.

[11] 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," *CoRR*, vol. abs/1807.09902, 2018. [Online]. Available: http://arxiv.org/abs/1807.09902

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

[13] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[14] N. Jaitly and G. E. Hinton, "Vocal tract length perturbation (vtlp) improves speech recognition," in *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language*, vol. 117, 2013.

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

[16] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[17] O. Akiyama and J. Sato, "Multitask learning and semi-supervised learning with noisy data for audio tagging," DCASE2019 Challenge, Tech. Rep., June 2019.

[18] J. Ebbers and R. Haeb-Umbach, "Convolutional recurrent neural network and data augmentation for audio tagging with noisy labels and minimal supervision," DCASE2019 Challenge, Tech. Rep., June 2019.