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

Technical Report

Janek Ebbers[*], *Reinhold Haeb-Umbach*

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

ABSTRACT

This report presents our Audio Tagging system for the DCASE 2019 Challenge Task 2. Our proposed neural network architecture consists of a convolutional front end using log-mel-energies as input features, a recurrent neural network sequence encoder outputting a single vector for a whole sequence and finally a fully connected classifier network outputting an activity probability for each of the 80 event classes. Due to the limited amount of available data we use various data augmentation techniques to prevent overfitting and improve generalization. Our best system achieves a label-weighted label-ranking average precision (lwlrap) of 73.0% on the public test set which is an absolute improvement of 19.3% over the baseline.

Index Terms— audio tagging, label noise, data augmentation

1. MODEL

The model is outlined in Tab. 1.

1.1. Feature Extraction

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

1.2. Neural Network Architecture

It consists of a convolutional (conv.), a recurrent and a fully connected module. We expect a four dimensional input to our model of shape $B{\times}C{\times}F{\times}N$ with B, C, F, N being the mini-batch size, number of channels, number of features and number of frames, respectively, where $C{=}1$ and $F{=}128$ are fix.

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 are 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.

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

Table 1: Convolutional Recurrent Neural Network for Audio Tagging with output shapes of each block. For simplicity we write $\lceil N/4 \rceil$ instead of $\lceil \lceil N/2 \rceil /2 \rceil$. 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)	$B{\times}80$

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.

2. DATA AUGMENTATION

Because there is only little data available, an efficient data augmentation is crucial to prevent overfitting and improve generalization of the system. In the following we outline the data augmentation methods that we combined during model training. All of them have shown to improve model performance evaluated on a held out validation set.

2.1. Mixup

Mixup [1] 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:

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

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 Gerneral purpose audio tagging, e.g. in [2], in the DCASE 2018 Challenge [3].

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 equations 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 between one and two.

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

2.2. Frequency Warping

Recently, SpecAugment [4] 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 stretching/compressing the whole spectrogram.

It is worth noting that the frequency warping performed here is very similar to Vocal Tract Length Pertubation [5].

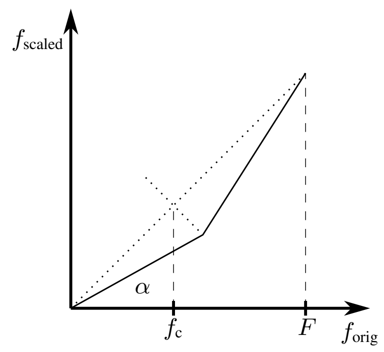

Figure 1: piece-wise linear frequency warping function.

2.3. 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

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

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

2.4. Time Masking

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

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

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

3. TRAINING

3.1. Data

For taining we use both the data with curated labels as well as the data with noisy labels. Recognizing that the train portion with noisy labels primarily contains audio signals of length 15 s, we randomly split the audio signals into two at length $r_n \cdot T_n$ with T_n being the length of the n-th signal and $r_n \sim \mathcal{U}(0.1, 0.9)$. 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.

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. To prevent training from being dominated by noisy labels, we iterate the curated portion R times as often as the noisy portion.

3.2. 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 [6] was used for optimization. Training is performed for 150K iterations with a learning rate of $3 \cdot 10^{-4}$ followed by 50K iterations with a learning rate of 10^{-4}.

We perform Stochastic Weight Averaging (SWA) [7] 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. Another advantage is that with SWA there is no need for held-out data to determine the best performing checkpoint.

4. EXPERIMENTS

The random splitting of the noisy examples, presented in Sec. 3.1, was independently performed three times resulting in three different datasets which we refer to as splits in the following.

4.1. Relabeling

For each of the three splits we train a model on five different folds using the provided noisy labels with $R = 8$ (this was chosen such that the curated portion makes more than 50% of the training). 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.

4.2. Results

On each relabeled split we trained a new model for a certain value of R using the whole dataset for training (we do not need held-out data to determine the best checkpoint as we use SWA).

Tab. 2 presents performance of our submitted systems. Model performance is evaluated in terms of label-weighted label-ranking average precision (lwlrap) [8]. The marked single model system is submitted as our candidate for the judges award. Combining multiple models (trained on different splits) into ensembles, however, significantly improves performance. The best performing system presented in the last row combines all submodels from the systems presented in the previous two rows into a single ensemble.

Table 2: Performance of different (ensemble) systems. Corresponding filenames of submitted systems are given as Ebbers_UPB_task2_<FileIdx>.output.csv.

#models	R	FileIdx	Judges Candidate	lwlrap
Baseline				0.537
1	4	3	✓	0.707
3	4	2		0.726
3	2	N/A		0.719
6	2;4	1		**0.730**

5. CONCLUSIONS

In this report we presented our system for the DCASE 2019 Challenge Task 2. We suggested a convolutional recurrent neural network architecture operating on log-mel spectrogram features. We put our main focus, however, on efficient data augmentation. We adjusted and combined the recently proposed mixup data augmentation and SpecAugment resulting in a high performance improvement compared to the baseline. We also presented a scheme to randomly split noisy labeled data to create differing datasets. We showed that ensembling models trained on these different datasets further improved performance.

6. REFERENCES

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

[2] I.-Y. Jeong and H. Lim, "Audio tagging system using densely connected convolutional networks."

[3] 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

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.