AUDIO TAGGING WITH CONVOLUTIONAL NEURAL NETWORKS TRAINED WITH NOISY DATA

Technical Report

Fabian Paischer[1], *Katharina Prinz*[1,2]

[1]Institute of Computational Perception, Johannes Kepler University, Linz, Austria
[2]Austrian Research Institute for Artificial Intelligence, Vienna, Austria
paischer101@gmail.com, katharina.prinz@ofai.at

ABSTRACT

This report is a description of our submission to the 2019 DCASE Challenge, Task 2. The task at hand is to predict one or more audio-tags, out of the 80 available tags, for the audio clips of different lengths, originating from two different datasets. For training, a total number of 4970 audio clips is provided with trustworthy labels, whereas 19815 samples contain a substantial amount of label noise with unknown noise ratio. To tackle this task, we propose two different convolutional neural network (CNN) architectures trained on different features to capture different aspects of the data. Stochastic Weight Averaging is used in order to improve generalisation. By averaging over the predictions of all five networks, we obtain an ensemble that provides us with the likelihood of 80 different labels being present in an input audio clip. On the unseen data of the Public Kaggle Leaderboard, our system reaches a Label Weighted Label Ranking Average Precision (Lwlrap) of 0.722.

***Index Terms*—** Audio Tagging, Ensemble, Convolutional Neural Networks, Noisy Labels, Stochastic Weight Averaging

1. INTRODUCTION

Modern machine learning tasks often require a large amount of data to train. While manual labels are not a scalable way to obtain ground-truth, various sources, e.g. websites, provide information that can be used to infer labels automatically. These, however, can contain a significant amount of noise. The ”Freesound Audio Tagging 2019 Challenge”, carried out as Task 2 of this year's DCASE Challenge [1], tackles exactly this problem, and provides, next to a smaller curated dataset, also a larger amount of audio samples with noisy annotations [2]. Contrary to Task 2 of the 2018 DCASE Challenge [3], the number of possible audio tags to be predicted rose from 41 to 80 potential labels. In addition to that, multiple labels can be predicted for a single audio clip.

In this technical report, we describe our submission to Task 2 of the 2019 DCASE Challenge. The following sections contain a description of our audio data preprocessing, the architectures of the convolutional neural networks (CNNs) which we use for the task, and information about the basic training procedure.

2. AUDIO DATA

2.1. Datasets

Curated and noisy audio clips provided by the 2019 DCASE Challenge [1] originate from two different sources, namely the Freesound dataset and the Yahoo Flickr Creative Commons 100M dataset. The 4970 curated and 19815 noisy clips all have 0.3 to 30 seconds length and are annotated with one or multiple of 80 different labels. For each of the labels, we have around 80 % of noisy, and 20 % of curated audio clips available.

2.2. Audio Preprocessing

Instead of directly using raw audio data for the training of our system, we preprocess each available clip. More precisely, we compute variations of Mel-spectrograms and constant Q-power spectra. Prior to the transformation into the frequency domain, we remove silent parts within the audio data with silence clipping using the SoX toolbox[1]. After this step, the features with five different sets of parameters are extracted.

The first kind of feature, subsequently denoted by **mel_n**, is extracted by applying a Short Time Fourier Transform (STFT) to the resampled audio data with a sampling rate of 32 kHz. The STFT uses a hop length of 192, and a Fast Fourier Transform (FFT) window length of 1024. Finally, a mel-scaled filter bank is used to obtain 128 frequency-bins from the logarithmic STFT, before the spectrogram is normalised.

Two other features, **mel** and **mel_weighted**, are computed similarly. The differences here are the parameters of the STFT with a sampling rate of 44.1 kHz, an FFT window length of 2048 and a hop size of 512. Instead of applying the mel-scaled filter bank to the logarithmic STFT, **mel_weighted** is perceptually weighted[2].

The last two features, denoted by **cqt_weighted** and **cqt_n_weighted**, are constant Q-power spectra. After silence clipping, a constant Q-transform is applied with two sets of parameters. For **cqt_weighted**, the sampling rate is 44.1 kHz, the hop length equals to 512, and the number of frequency bins is 128. The constant Q-transform for **cqt_n_weighted** uses a sampling rate of 32 kHz, a hop length of 192, and a number of frequency bins equal to 84. After the transformation into the frequency domain, the spectra are perceptually weighted and normalised.

In order to simplify the processing of audio data with different lengths, we fix the number of frames for all spectrograms to be 2784. Shorter features are padded by repeating the spectrogram; longer features, on the other hand, are simply clipped. The fixed length is chosen based on frequently occurring feature lengths.

[1]`http://sox.sourceforge.net/`

[2]`https://librosa.github.io/librosa/generated/librosa.core.perceptual_weighting.html`

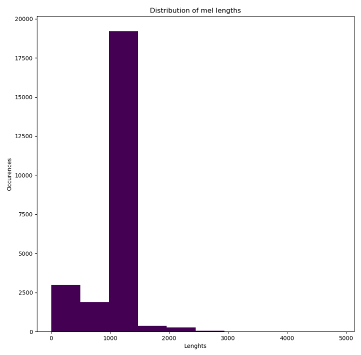

Figure 1: Lengths of Mel spectrograms (**mel**).

Figure 1 shows, as an example, the histogram of Mel-spectrogram lengths.

Instead of using the entire spectrogram as an input to audio tagging models, we work with windows having a frame length of 348, and train a network on half-overlapping windows (i.e. with hop size of 174). This means that we do not obtain a single, but rather multiple outputs of a model for one audio clip. The final prediction of a model is the average prediction of all windows within a clip.

3. MODEL ARCHITECTURES

For our audio tagging system, previously described audio features are used as an input to two different CNN architectures. The first one is an adaptation of the 2018 DCASE Challenge's second place [4]. The second architecture is a CNN with fewer layers and average-pooling (in contrast to the max-pooling layers in previous model).

3.1. The Max-Pooled CNN

Figure 2 shows the architecture of the adapted CNN of Dorfer and Widmer [4]. It consists of 5×5, 3×3 and 1×1 convolutions with Rectified Linear Units (ReLUs), followed by batch normalisation and max-pooling layers. For regularisation, dropout is used. The last layers of the CNN consist of a global pooling layer, which averages over the first dimension of the feature map, and maximises over the second, and finally a dense layer. In order to obtain a final prediction that describes the likelihood of each label being present in an audio clip, a softmax activation function is used.

3.2. The Average-Pooled CNN

The second CNN which we trained is visualised in Figure 3. Besides its depth, the main difference is the prevalent usage of average-pooling as opposed to max-pooling between convolutional layers. The amount of hidden layers in this network is reduced in order to obtain a smaller receptive field in the last layer for regularisation, allowing the network to focus on local parts of the input, as discussed in [5]. Also for this network, batch normalisation and dropout are used.

Input $1 \times 348 \times 128$ (84)
5×5 Conv(stride = 2, activations = relu, filters = 64), BN() 3×3 Conv(stride = 1, activations = relu, filters = 64), BN() 2×2 MaxPooling(stride = 2) Dropout(0.3)
3×3 Conv(stride = 1, activations = relu, filters = 128), BN() 3×3 Conv(stride = 1, activations = relu, filters = 128), BN() 2×2 MaxPooling(stride = 2) Dropout(0.3)
3×3 Conv(stride = 1, activations = relu, filters = 256), BN() Dropout(0.3) 3×3 Conv(stride = 1, activations = relu, filters = 256), BN() Dropout(0.3) 3×3 Conv(stride = 1, activations = relu, filters = 384), BN() Dropout(0.3) 3×3 Conv(stride = 1, activations = relu, filters = 384), BN() 2×2 MaxPooling(stride = 2) Dropout(0.3)
3×3 Conv(stride = 1, activations = relu, filters = 512), BN() 3×3 Conv(stride = 1, activations = relu, filters = 512), BN() 1×2 MaxPooling(stride = (1, 2)) Dropout(0.3)
3×3 Conv(stride = 1, activations = relu, filters = 512), BN() 3×3 Conv(stride = 1, activations = relu, filters = 512), BN() 1×2 MaxPooling(stride = (1, 2)) Dropout(0.3)
3×3 Conv(stride = 1, activations = relu, filters = 512), BN() Dropout(0.5) 1×1 Conv(stride = 1, activations = relu, filters = 512), BN() Dropout(0.5)

GlobalPooling()
Dense(80)

Softmax()

Figure 2: Architecture of max-pooled CNN, adapted from [4].

4. TRAINING PROCEDURE

4.1. Setup and Integration of Noisy Data

As a way to reduce the possibility of over-fitting and memorisation of corrupt labels, mixup-data augmentation [6] with $\alpha = 0.3$ is applied in our training procedure. To learn the parameters of any network, we use a categorical cross-entropy loss function and an Adam optimiser with a starting learning rate of 0.001. In addition to that, we use a linear learning rate decay after 50 epochs for the max-pooled CNN; for the average-pooled CNN, on the other hand, we drop the learning rate by a factor of 10 after 150 epochs. Each model in our final audio tagging system is training for 200 epochs with a mini-batch size of 7.

As we do not know the exact amount of noise present in the non-curated audio clips, the network is trained on solely curated data for the majority of epochs. Every fifth epoch we switch to the noisy dataset, and train on a randomly selected subset of the noisy data with decreased learning-rate for an equal amount of steps per epoch. Note here, that the network is thus not seeing the entire set of noisy samples in one such epoch.

4.2. Stochastic Weight Averaging

To further improve the generalisation of our audio tagging system, we use stochastic weight averaging (SWA) [7] for our average-pooled CNN architecture. The original SWA recomputes the batch-norm statistics in every forward pass, in order to match the new averaged weights of the SWA model. However, due to the computational complexities, we skip this step and solely average over the weights of the last 15 epochs instead.

Input $1 \times 348 \times 128$
5×5 Conv(stride = 2, activations = relu, filters = 64), BN()
3×3 Conv(stride = 1, activations = relu, filters = 64), BN()
2×2 AveragePooling(stride = (2, 2))
Dropout(0.3)
3×3 Conv(stride = 1, activations = relu, filters = 128), BN()
3×3 Conv(stride = 1, activations = relu, filters = 128), BN()
2×3 AveragePooling(stride = (2, 3))
Dropout(0.3)
3×3 Conv(stride = 1, activations = relu, filters = 256), BN()
3×3 Conv(stride = 1, activations = relu, filters = 256), BN()
2×3 AveragePooling(stride = (2, 3))
Dropout(0.3)
1×1 Conv(stride = 1, activations = relu, filters = 512)
BN(), Dropout(0.5)
1×1 Conv(stride = 1, activations = relu, filters = 80)
GlobalPooling()
Softmax()

Figure 3: Architecture of the CNN using average pooling.

feature	architecture	loss	lwlrap
mel	avg-pooled	1.384	0.825
mel_weighted	avg-pooled	1.437	0.817
mel_n	max-pooled	0.168	0.835
cqt_n_weighted	max-pooled	0.138	0.776

Table 1: Average 4-fold cross validation loss and lwlrap for four trained CNNs, contained in every of our three ensembles.

4.3. 4-Fold Cross-Validation

In addition to the evaluation the public test-set offers, we use a 4-fold cross validation (CV). To set this up, the curated data is split into four folds with comparable distributions over the different labels. For training, three folds of curated, as well as the noisy data are used; the remaining fold, on the other hand, serves as validation set. As every fold is used exactly once for evaluation, we end up training four models, which we combine for our final submission by averaging over their respective predictions.

4.4. Ensembles of Final Submissions

With our final audio tagging system, we aim to obtain one prediction that describes the likelihoods of 80 labels being present in a given audio clip. We create an ensemble of different models by averaging over their respective predictions for a given audio clip.

Table 1 shows four models with their respective features, architectures, as well as the average loss and label weighted label ranking average precision (lwlrap) over the four folds in our CV setup. These four models are used in every ensemble of our three submissions.

The first submission is an ensemble of five networks. In addition to the four models from Table 1, we include a model with the avg-pooled architecture trained on *cqt_weighted* features. This system achieves our highest lwlrap of 0.722 on the public Kaggle leaderboard[3].

The second ensemble, with a lwlrap of 0.721 on the public leaderboard, combines six different networks. More precisely, the five models from submission 1 are enhanced with a fully-convolutional version of the max-pooled architecture (Figure 3). For training we use the *mel_n* features.

Finally, the last submission consists of the four networks in Table 1. Additionally, a model with a max-pooled architecture is fed with a perceptually weighted version of the *mel_n* features. On the public leaderboard, this ensemble results in a lwlrap of 0.715.

Table 2 summarises the additional models with their average performance during cross validation for each of the three submissions.

submission	feature	architecture	loss	lwlrap
1, 3	cqt_weighted	avg-pooled	1.865	0.825
2	mel_n_weighted	max-pooled	2.475	0.724
3	mel_n	max-pooled*	1.382	0.889

Table 2: Additional CNNs that complete the ensembles for all three submissions.

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

5. CONCLUSION

In this report, we proposed an ensemble of multiple CNNs as an audio tagging system that provides us with a prediction of how likely it is, that any one of 80 different audio tags is present in a given audio clip. For training we have audio samples with curated, as well as noisy ground-truth labels available. Every fifth epoch, we train on a fraction of the noisy data. Furthermore, we work with different Mel-spectrograms and constant Q-transform spectra, and two different CNN architectures. Mixup-data augmentation and stochastic weight averaging are two methods we use to increase generalisation. Our best ensemble with respect to the public Kaggle leaderboard combines five networks and achieves a lwlrap of 0.722.

6. ACKNOWLEDGMENT

Thanks to everyone at the Institute of Computational Perception at the JKU, for the supervision, discussions, advice and also the computational resources. In particular, thanks to Andreas Arzt, Hamid Eghbal-zadeh, Khaled Koutini, Rainer Kelz and Gerhard Widmer.

This work is supported by the Austrian National Science Foundation (FWF, P 31988).

7. REFERENCES

[1] http://dcase.community/challenge2019/task-audio-tagging.

[2] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, and X. Serra, "Audio tagging with noisy labels and minimal supervision," 2019.

[3] http://dcase.community/challenge2018/index.

[4] M. Dorfer and G. Widmer, "Training general-purpose audio tagging networks with noisy labels and iterative self-verification," DCASE2018 Challenge, Tech. Rep., September 2018.

[5] K. Koutini, H. Eghbal-zadeh, M. Dorfer, and G. Widmer, "The Receptive Field as a Regularizer in Deep Convolutional Neural Networks for Acoustic Scene Classification," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, A Coruña, Spain, 2019.

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

[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.