SPECMIX: A SIMPLE DATA AUGMENTATION AND WARM-UP PIPELINE TO LEVERAGE CLEAN AND NOISY SET FOR EFFICIENT AUDIO TAGGING

Technical Report

Eric Bouteillon

France
ericspamkiller-kaggle@yahoo.fr

ABSTRACT

This paper presents a semi-supervised **warm-up pipeline** used to create an efficient audio tagging system as well as a novel data augmentation technique for multi-labels audio tagging named by the author **SpecMix**. These new techniques were applied to our submitted audio tagging system to the *Freesound Audio Tagging 2019* challenge carried out within the *DCASE 2019 Task 2 challenge* [3]. Purpose of this challenge consist of predicting the audio labels for every test clips using machine learning techniques trained on a small amount of reliable, manually-labeled data, and a larger quantity of noisy web audio data in a multi-label audio tagging task with a large vocabulary setting. Results are reproducible, description of requirements, steps to reproduce and source code are available on GitHub[1]. Source code is released under an open source license (MIT).

Index Terms— Audio Tagging, Fully Convolutional Neural Networks, Noisy Labels, Warm-up Pipeline, SpecMix

1. INTRODUCTION

Reliable automatic general-purpose audio tagging systems are difficult to build due to the complexity of sounds around us. A significant amount of manual effort goes into tasks like annotating sound collections and providing captions for non-speech events in audiovisual content. To tackle this problem, Freesound (an initiative by MTG-UPF that maintains a collaborative database with over 400,000 Creative Commons Licensed sounds) and Google Research's Machine Perception Team (creators of AudioSet, a large-scale dataset of manually annotated audio events with over 500 classes) have teamed up to develop the dataset for *Freesound Audio Tagging 2019* challenge carried out within the *DCASE 2019 Task 2 challenge* [3]. One of the challenge of this dataset is the small amount of reliable manually labeled data compared to the important number of unreliable automatically labeled data. The second challenge of this dataset is to correctly identify more than one label for a single audio sample.

Two common solutions to tackle the small number of samples with good label are: 1/ fully exploit these samples with correct labels by using *data augmentation* to generate artificial training data; 2/ filter the samples with unreliable label in a *semi-supervised* way before accepting them in training set. Let's dig further into these techniques.

Data augmentation has been proposed as a method to generate additional training data for Automatic Speech Recognition. For example, in [5], artificial data was augmented for low resource speech recognition tasks. Vocal Tract Length Normalization has been adapted for data augmentation in [6]. Augmentation can also be applied on spectral domain like SpecAugment [1] but these techniques augment only the sample without changing the associated label. We present in this paper a novel data augmentation technique for multi-labels audio tagging, named by the author **SpecMix**, which allows to create additional training samples with new labels well suited for multi-labels problems.

Semi-supervised learning (SSL) has a large literature due to the different techniques to leverage unlabeled data. A broad class of approaches contains feature learning with unlabeled data, based on generative models including variational auto-encoders [7], or generative adversarial networks [8]. Another technique is self-training, which refers to retraining a model based on its own predictions on unlabeled data. Self-training is widely applied in practice. Data distillation [9] proposed an approach based on self-training to visual structure prediction problems. This paper presents a semi-supervised **warm-up pipeline** used to create an efficient audio tagging system by filtering unreliable samples in a multi-stage process and assembling model checkpoints to improve the global accuracy of the system.

2. AUDIO DATA PREPROCESSING

In our approach, audio clips were first trimmed of leading and trailing silence (threshold of 60 dB), then converted into 128-bands mel-spectrogram using a 44.1 kHz sampling rate, hop length of 347 samples between successive frames, 2560 FFT components and frequencies kept in range 20 Hz – 22,050 Hz. Last preprocessing consisted in normalizing (mean=0, variance=1) the resulting images and duplicating to 3 channels.

3. DATA AUGMENTATION

One important technique to leverage a small training set is to augment this set using data augmentation. For this purpose we created a new augmentation named ***SpecMix***. This new augmentation is an extension of *SpecAugment* [1] inspired by *mixup* [2].

SpecAugment applies 3 transformations to augment a training sample: time warping, frequency masking and time masking on mel-spectrograms.

1 https://github.com/ebouteillon/freesound-audio-tagging-2019

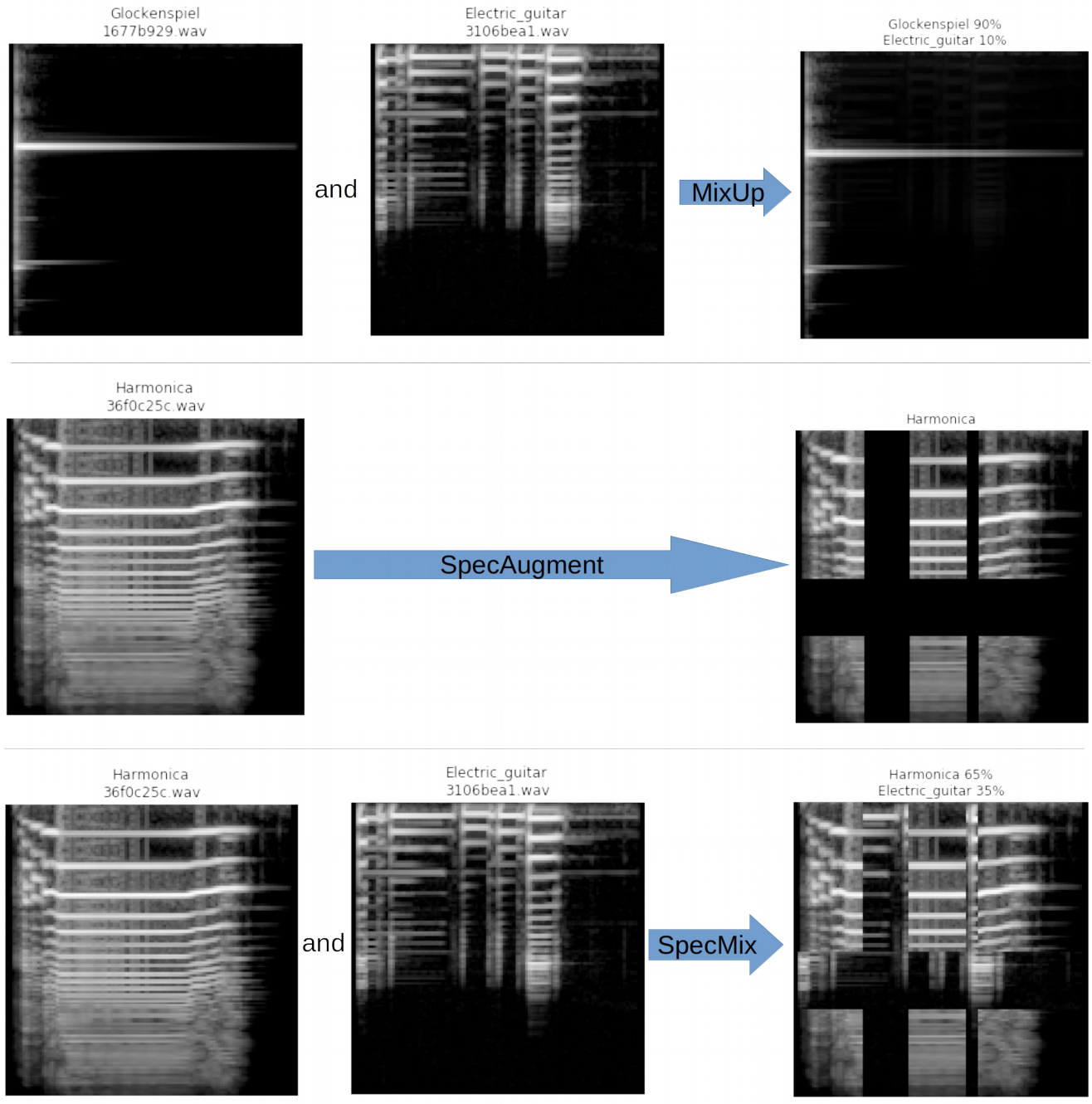

Figure 1: Comparison of mixup, SpecAugment and SpecMix.

mixup: create a virtual training example by computing a weighted average of two samples inputs and targets.

3.1. SpecMix

SpecMix is inspired from the two most effective transformations from *SpecAugment* and improve them by replacing simple masking by another sample excerpt replacement. Moreover we extends this technique by creating virtual multi-labels target like *mixup*:

1. *Frequency replacement* is applied so that f consecutive mel-frequency channels $[f0, f0+f)$ are replaced from another training sample, where f is first chosen from a uniform distribution from minimal to maximum the frequency mask parameter F, and $f0$ is chosen from $[0, \nu-f)$. ν is the number of mel frequency channels.
2. *Time replacement* is applied so that t consecutive time steps $[t0, t0+t)$ are replaced from another training sample, where t is first chosen from a uniform distribution from 0 to the time mask parameter T, and $t0$ is chosen from $[0, \tau-t)$. τ is the number of time samples.
3. *Target* of the new training sample is computed as the weighted average of each original samples. The weight

for each original sample is proportional to the number of pixel from that sample. Our implementation uses same replacement sample for *Frequency replacement* and *Time replacement*, so it gives us a new target computed based on:

$$y = (1-a)\cdot y_1 + a\cdot y_2 \qquad (1)$$

where $a = (f/v) + (t/\tau) - ((f/v)\cdot(t/\tau))$

3.2. Others data augmentation

We added other data augmentation techniques:

- *mixup* before SpecMix. A small improvement is observed (lwlrap increased by +0.001). mixup is first applied on current batch, generating new samples for the current batch and then SpecMix is applied on these newly created samples. In the end, combining mixup and SpecMix, up to four samples are involved in the generation of one single sample.
- *zoom and crop*: a random zoom between 1. and max 1,05 is applied with probability 75%, a small improvement is seen (lwlrap increased by +0.001).
- *lighting*: a random lightning and contrast change controlled is applied with probability 75%.

4. MODEL

In this section, we describe the neural network architectures used:

Version 1 consists in an ensemble of a custom CNN "CNN-model-1" defined in Table 1 and a VGG-16 with batch-normalization. Both are trained in the same manner.

Version 2 consist of only our custom CNN "CNN-model-1", defined in Table 1.

Version 3 is evaluated for Judge award and it is same model as version 2.

Input 128 × 128 × 3
3 × 3 Conv(stride=1, pad=1)−64−BN−ReLU
3 × 3 Conv(stride=1, pad=1)−64−BN−ReLU
3 × 3 Conv(stride=1, pad=1)−128−BN−ReLU
3 × 3 Conv(stride=1, pad=1)−128−BN−ReLU
3 × 3 Conv(stride=1, pad=1)−256−BN−ReLU
3 × 3 Conv(stride=1, pad=1)−256−BN−ReLU
3 × 3 Conv(stride=1, pad=1)−512−BN−ReLU
3 × 3 Conv(stride=1, pad=1)−512−BN−ReLU
concat(AdaptiveAvgPool2d + AdaptiveMaxPool2d)
Flatten−1024-BN-Dropout 25%
Dense-512-ReLU-BN-Dropout 50%
Dense-80

Table 1: CNN-model-1 architecture. BN: Batch Normalisation, ReLU: Rectified Linear Unit.

5. TRAINING

Training is done in 4 stages, each stage generating a $model_x$ using the same CNN architecture but trained on a different subsets of curated or noisy datasets. A $model_x$ is generated at a stage using the 10-folds cross-validation process, and we save for each of the 10 folds the weights of trained models. These weights are named here model checkpoint. Batches of 128 augmented excerpts of randomly selected sample mel-spectrograms are used. The implementation was done using the fastai library [4]. These $model_x$ are used for 3 things:

- **warm-up** the model training in the next stage
- help in a semi-supervised way to **select** noisy elements
- participate in the **test prediction** (except model 1)

An important point of this competition, is that we are not allowed to use neither external data nor pretrained models. So our pipeline presented below only used curated and noisy sets from the competition.

- **Stage 1**: Train a model (named "$model_1$") from scratch only using the noisy set. Then compute 10-folds cross-validated per-clip label-ranking average precision (lrap) [10] on noisy set and noted as $lrap_1$. $lrap_1$ is a set of values, one for each clip in a dataset and shall not be confused with the competition metric (lwlrap) which is a single value computed for a whole dataset.
- **Stage 2**: Train a model (named "$model_2$") only on curated set but use $model_1$ checkpoints as pretrained model (to initialize CNN weights for each fold). Then compute 10-folds cross-validated per-clip lrap on noisy set ($lrap_2$).
- **Stage 3**: Let's start semi-supervised learning: our algorithm selects samples from noisy set that are (almost) correctly classified by both $model_1$ and $model_2$: we simply keep sample from noisy set having a geometric mean of ($lrap_1$, $lrap_2$) higher or equal to 0,5. A maximum of 5 samples per fold and per label is selected. Then train a model ($model_3$) on curated plus selected noisy samples and use $model_2$ checkpoints as pretrained model. In the end, compute 10-folds cross-validated per-clip lrap on noisy set ($lrap_3$).
- **Stage 4**: Let's continue semi-supervised learning: our algorithm selects that time samples from noisy set which are strictly correctly classified by $model_3$, i.e. samples from noisy set getting a $lrap_3$ equal to 1. Then train a model ($model_4$) on curated plus selected noisy samples and use $model_3$ checkpoints as pretrained model.
- **Last step:** final predictions are the average of predictions on test set from $model_2$, $model_3$ and $model_4$.

Before training, noisy and curated datasets were split in 10 sets each for cross-validation. These sets were kept unchanged for all models training. When training a model checkpoint on the i^{th} fold of the cross-validation, the i^{th} set of noisy dataset and i^{th} set of curated dataset are kept out of fold; the 9 remaining sets of noisy and curated are kept eligible for training. Then these eligible training data were additionally filtered as described above (sum-up in Figure 2) to form the training data. It means that the i^{th} set never leaks in during the training of the i^{th} fold whatever the $model_x$ checkpoint. Then the model trained for i^{th} fold computes predictions for the i^{th} split of curated (10% of noisy). Once we have all predic-

tions, we are able to compute the per-clip ($lrap_x$) to select noisy samples for following stages' instance selection.

Notice that most semi-supervised techniques are doing instance selection based on the prediction probabilities. Instead, we select the valid instances by thresholding the per-clip lrap metric.

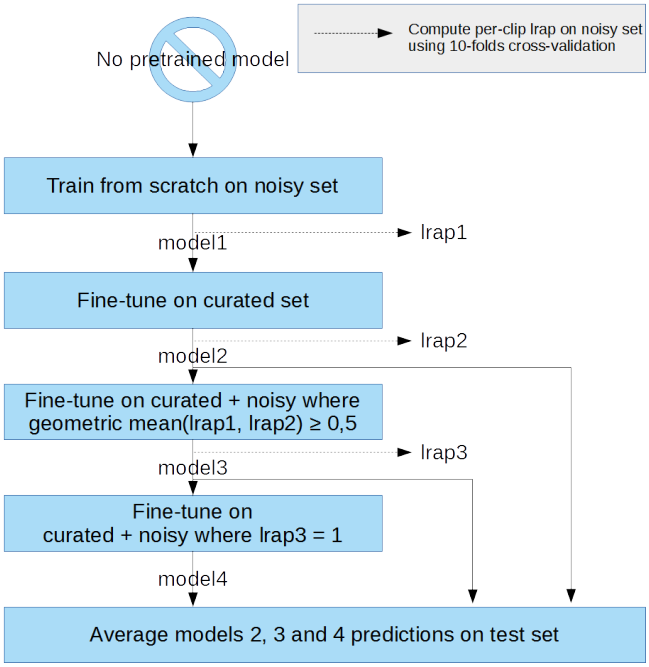

Figure 2: Warm-up pipeline

6. INFERENCE

During inference (prediction on test set), we split the test audio clips in windows of 128 time samples (2 seconds), windows were overlapping. Then these samples are fed into our models to obtain predictions. All predictions linked to an audio clip are averaged to get the final predictions to submit.

This competition had major constraints for test prediction inference: submission must be made through a Kaggle kernel with time constraints. As our solution requires a GPU, the inference of the whole unseen test set shall be done in less than an hour.

In order to match this hard constraint, we took following decisions:

- use same preprocessing and inputs for all models,
- limit the final ensemble to two models architecture only,
- limit the overlapping of windows,
- as the unseen test set was reported to be three times the public test set by organizers, then we made sure to infer the public test set in less than 1,000 seconds, which should allow the kernel to infer the unseen test set in about 3,000 seconds and keep a 20% time margin for safety.

7. EXPERIMENTS AND RESULTS

To asses the performance of our system, we provide results in Table 2. The metric used is lwlrap (*label-weighted label-ranking average precision*). In one hand, evaluation of performances on noisy set (lwlrap noisy) set and curated set (lwlrap curated) are computed using the competition metric on generated $model_x$'s prediction. Prediction for one clip in noisy or curated sets is produced by one single model checkpoint (outcome of each of the 10 iteration of the cross-validation process). On another hand, evaluation on test set predictions are values reported by the public leaderbord during the Kaggle competition on public test set. Prediction for one public test element is the average of the predictions of the 10 checkpoints for $model_x$. We did not score the $model_1$ with public set during the competition, hence the N/A value reported for lwlrap leaderboard.

Model	lwlrap noisy	lwlrap curated	lwlrap leaderboard
$model_1$	0.65057	0.41096	N/A
$model_2$	0.38142	0.86222	0.723
$model_3$	0.56716	0.87930	0.724
$model_4$	0.57590	0.87718	0.724
average	N/A	N/A	0.733

*Table 2: Empirical results of **version 2** (single CNN-model-1) using proposed warm-up pipeline*

Each stage of the **warm-up pipeline** generates a model with excellent prediction performance on the test test. Each model would give us a silver medal with the 25th position on the public leaderboard. Moreover these warm-up models bring sufficient diversity on their own, as a simple averaging of their predictions (lwlrap .733) gives 16th position on the public leaderboard.

Final 12th position of the author was provided by **version 1**, which is an average of the predictions given by *CNN-model-1 and VGG-16,* both trained the same way.

8. CONCLUSION

This paper presents a semi-supervised **warm-up pipeline** used to create an efficient audio tagging system as well as a novel data augmentation technique for multi-labels audio tagging named by the author **SpecMix**. These techniques leveraged both clean and noisy sets and were shown to give excellent results.

These results are reproducible, description of requirements, steps to reproduce and source code are available on GitHub[1]. Source code is released under an open source license (MIT).

9. ACKNOWLEDGMENT

These results were possible thanks to the infinite support of my 5 years-old boy, who said while I was watching the public leaderboard: "*Dad, you are the best and you will be at the very top*". ❤

I also thank the whole kaggle community for sharing knowledge, ideas and code. In peculiar daisuke for his kernels during the competition and mhiro2 for his simple CNN-model and all the competition organizers.

1 https://github.com/ebouteillon/freesound-audio-tagging-2019

10. REFERENCES

[1] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, Quoc V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition", arXiv:1904.08779, 2019.

[2] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. "*mixup: Beyondempirical risk minimization*". arXiv preprint arXiv:1710.09412, 2017.

[3] Eduardo Fonseca, Manoj Plakal, Frederic Font, Daniel P. W. Ellis, and Xavier Serra. "Audio tagging with noisy labels and minimal supervision". Submitted to DCASE2019 Workshop, 2019. URL: https://arxiv.org/abs/1906.02975

[4] fastai, Howard, Jeremy and others, 2018, URL: https://github.com/fastai/fastai

[5] A. Ragni, K. M. Knill, S. P. Rath, and M. J. F. Gales, "Data augmentation for low resource languages," in INTERSPEECH, 2014,

[6] N. Jaitly and G. Hinton, "Vocal Tract Length Perturbation (VTLP)improves speech recognition," in ICML Workshop on Deep Learn-ing for Audio, Speech and Language Processing, 2013.

[7] D. Kingma, S. Mohamed, D.J. Rezende, and M. Welling,"Semi-supervised learning with deep generative models," in NIPS, 2014.

[8] J.T. Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial net-works," inICLR, 2015.

[9] I. Radosavovic, P. Dollar, R. Girshick, G. Gkioxari, and K. He, "Data distillation: Towards omni-supervised learning," inCVPR, 2018

[10] Label Ranking Average Precision (lrap). Sklearn description. URL: https://scikit-learn.org/stable/modules/model_evaluation.html#label-ranking-average-precision