SUPERVISED CONTRASTIVE LEARNING FOR PRE-TRAINING BIOACOUSTIC FEW-SHOT SYSTEMS

Technical Report

Ilyass Moummad[1]*, Romain Serizel*[2]*, Nicolas Farrugia*[1]

[1]IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France,
[2]University of Lorraine, CNRS, Inria, Loria, F-54000, Nancy, France

ABSTRACT

We show in this work that learning a rich feature extractor from scratch using only official training data is feasible. We achieve this by learning representations using a supervised contrastive learning framework. We then transfer the learned feature extractor to the sets of validation and test for few-shot evaluation. For few-shot validation, we simply train a linear classifier on the negative and positive shots and obtain a F-score of 63.46% outperforming the baseline by a large margin. We don't use any external data or pretrained model. Our approach doesn't require choosing a threshold for prediction or any post-processing technique. Our code is publicly available on Github : `https://github.com/ilyassmoummad/dcase23_task5_scl`

Index Terms— Contrastive learning, representation learning, transfer learning, few-shot learning

1. INTRODUCTION

Few-Shot Learning (FSL) is a machine learning problem where a model has to learn to adapt to new categories of data unseen during training with only few labeled samples. FSL is adapted for many applications where acquisition or annotation is expensive or time consuming. Sound Event Detection (SED) is the problem of locating onsets and offsets of certain sounds. In bioacoustics for instance, recordings are usually long with few occuring events which can be time consuming for labeling. FSL can be a solution to detect these events with few labeled samples. This is the framework of DCASE task 5. In the baseline of the challenge, prototypical networks (ProtoNets) [1] are proposed as a learning framework to solve FSL problem of detecting animal sound events. ProtoNets, a meta-learning framework, have been state-of-the-art FSL audio systems in the recent years. However, in Computer Vision, simple transfer learning methods are still outperforming sophisticated meta-learning methods in FSL [2]. We follow the trend of using transfer learning to solve FSL problems for this DCASE task [3].

As the quality of the feature extractor model is crucial for efficient transfer learning, we propose to train a model on the training set using the supervised contrastive learning (SCL) framework [4]. The idea behind SCL is to learn a latent space where samples sharing the same class labels are pulled together whereas samples with different class labels are pushed apart. This framework has shown competitive transfer learning capabilities of its learned representations. After training the model using SCL, for each audio file we extract features of the positive and negative prototypes as well as the query samples of the validation set independently. We then train a binary classifier to output the query class either by freezing the encoder layers or by finetuning them.

This work was co-funded by the AI@IMT program and the company OSO-AI.

2. METHOD

2.1. Framework

SCL consists in learning an embedding space in which the samples with the same class labels are close to each other, and the samples with different class labels are far from each other. Formally, a composition of an encoder f and a shallow neural network h called a projector (usually a MLP with one hidden layer) are trained to minimize the distances between representations of samples of the same class while maximizing the distances between representations of samples belonging to different class. After convergence, h is discarded, and the encoder f is used for transfer learning for downstream tasks. The supervised contrastive loss (SCL) is calculated as follows:

$$\mathcal{L}^{SCL} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_p / \tau\right)}{\sum\limits_{n \in N(i)} \exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_n / \tau\right)} \quad (1)$$

where $i \in I = \{1...2N\}$ is the index of an augmented sample within a training batch, containing two views of each original sample. These views are constructed by applying a data augmentation function A twice to the original samples (see section 2.2). $\boldsymbol{z}_i = h(f(A(\boldsymbol{x}_i))) \in \mathbb{R}^{D_P}$ where $n \in \{1...N\}$ with N being batch size and D_P is the projector's dimension. $P(i) = \{p \in I : y_p = y_i\}$ is the set of indices of all positives in the two-views batch distinct from i sharing similar label with i, and $|P(i)|$ is its cardinality, $N(i) = \{n \in I : y_n \neq y_i\}$ is the set of indices of all negatives in the two-views batch having different label from i, the $\cdot$ symbol denotes the dot product, and $\tau \in \mathbb{R}^{+*}$ is a scalar temperature parameter.

2.2. Training with Data Augmentation

We train our model from scratch on the training set using SCL. Specifically, in each training batch, we use data augmentation to create two views for each sample. We train a model to output similar representations for each view created from the same original sample as well as views sharing the same labels, and to output dissimilar representations for views from samples with different class

Table 1: Results on the validation set

System	Precision	Recall	F1-score	HB			ME			PB		
				Pr	Re	F1	Pr	Re	F1	Pr	Re	F1
Submission_1	71.41	55.19	62.26	77.14	81.57	79.29	65.45	69.23	67.28	72.64	36.17	48.29
Submission_2	73.93	55.59	63.46	82.95	82.32	82.63	67.69	84.61	75.21	72.72	33.33	45.71
Submission_3	72.90	55.14	62.79	79.73	89.72	84.43	74.60	90.38	81.73	65.57	31.06	42.19
Submission_4	67.08	51.58	58.32	81.20	91.38	**85.99**	58.75	90.38	71.21	65.00	27.65	38.80
2022 Winners S_1 [5]	**77.5**	**71.5**	**74.4**	-	-	77.0	-	-	**90.0**	-	-	**53.7**
2022 Winners S_2 [5]	75.6	62.1	68.2	-	-	85.8	-	-	79.2	-	-	48.1
2022 Winners S_3 [5]	66.5	51.8	58.5	-	-	76.7	-	-	64.2	-	-	42.4

labels. We use the following augmentations from the audio representation learning litterature [6, 7, 8]:

- Spectrogram mixing : we add background sounds using other samples from the same batch using the formula : $\hat{x_1} = \alpha x_1 + (1-\alpha)x_2$, where $\hat{x_1}$ is considered a view of x_1 and x_2 is a random sample from the batch
- Frequency shift : we approximate frequency shift by shifting upwards the spectrogram by few bands
- Random crop : we crop a patch from the spectrogram in the time axis, preserving the global audio semantic
- Spectrogram resize : we resize the spectrogram to the original size, combined with random crop approximate time stretching
- Power gain : we attenuate the power of the spectrogram by multiplying it with a coefficient sampled uniformly between 0.75 and 1.
- Additive white gaussian noise : we add a small additive white gaussian noise to the view

The above augmentations are applied sequentially in the presented order and are applied twice on the same data, apart from spectrogram mixing which is applied only on one view (in our experiments, this worked best). Details about the implementation are to be found in our github page.

2.3. Evaluation

For each audio file, we retrieve the first five shots as the positive prototypes. As for the negative prototypes, we consider the intervals preceding each of the five shots. We follow the work of Tang et al. [5] for the choice of adaptive window length, as for the window shift we take half the window length as in the protonet baseline. We train a linear binary classifier on top of the encoder using segments from the negative and positive prototypes. Afterwards, we use the final model (composition of the encoder and the classifier) to predict the class (negative or positive) for each query segment. We do not use any postprocessing technique.

3. IMPLEMENTATION DETAILS

3.1. Data

We use only the official training set for all the positive annotated segments. We compute Mel spectrogram features with a FFT of size 512, a hop length of 128, a number of mels of 128 and a sampling rate of 22.05 kHz. Each positive annotated segment from the training set is chunked into patches of length 200 ms with a sliding window of length 100ms. We apply min-max normalization on each patch.

3.2. Model architecture

We use a ResNet [9] of three blocks of three convolutions each with feature maps of size 64, 128 and 256. Each convolution is followed by a batch normalization and a leaky ReLU activation. We apply max pooling after each block, with a kernel of size 2x2 for the first and seconds blocks and 1x2 for the third block as not to pool too much the frequency bands to preserve frequency information as suggested by Hertkorn [10]. In order to maintain the same output dimension with different input lengths, we apply an adaptive max pooling with a desired output size of (8, 1) at the end of the network to get a latent vector of size 8 x 256 = 2048. We add to the convolutional encoder, a multi-layer perceptron projector with a hidden layer of dimension 2048 and an output layer of dimension 512.

3.3. Data Augmentation

We use the following parameters for the chosen augmentations :

- Spectrogram mixing : α is sampled from $\beta(5, 2)$ distribution
- Frequency shift : the shift size is sampled uniformly between 0 and 10
- Random crop : the crop is sampled uniformly between 60% and 100% in the time axis
- Spectrogram resize : we resize the spectrogram to the original size
- Power gain : we multiply the mel spectrogram with a coefficient sampled uniformly between 0.75 and 1
- Additive white gaussian noise : we add a gaussian noise with zero mean and a variable standard deviation chosen uniformly between 0 and 0.1

3.4. Training and Evaluation

We train our model from scratch on the training set using SCL framework with a temperature τ = 0.06 using SGD optimizer with a batch size of 128, a learning rate of 0.01 with a cosine decay schedule, momentum of 0.9, and a weight decay of 0.0001 for 50 epochs. We then discard the MLP projector and transfer the encoder on the validation and evaluation sets. We submit four systems to the challenge: the first consists in freezing the learned layers from the training set and training only a linear binary classifier on top of it on the five positive and negative prototypes. As for the second, third and fourth submissions, we finetune the last, two last, and all the layers, respectively, on the five positive and negative prototypes using random resized crop in the time axis with a crop of size 90% and 100%

Table 2: Comparison to baseline systems

Method	Precision	Recall	F-score
Template Matching	2.42	18.32	4.28
Prototypical	36.34	24.96	29.59
Ours (Submission_2)	73.93	55.59	63.46
2022 Winners [5]	77.5	71.5	74.4

of the original size. We optimize our systems using Adam optimizer with a learning rate of 0.01 for 20 epochs for the first system and a learning rate of 0.001 for 40 epochs for the other systems. These hyperparameters were chosen on the validation set.

4. RESULTS

We report the performance of our four systems on the validation set in the upper part of Table 1. For PB dataset, where events are short, the first system outperforms the others, suggesting that finetuning hurts the performance when only few positive patches are present. the third and fourth systems outperform the other systems on HB dataset, where events are mostly long, suggesting that finetuning more layers is practical when more positive patches are present. For the second system where we finetune only the last layer, does okay for all dataset but outperforms the other systems on average obtaining an F1 score of 63.46%.

In the lower part of Table 1, we show the scores for DCASE 2022 challenge task 5 winners [5]. Please note that the result reported includes the ML dataset in the validation set, which is not present in the final validation set of the 2022 and 2023 challenge. S_1, S_2, and S_3 are their submitted systems for frame-level, seg-level and event-filter approaches, respectively. Our systems are competitive with their S_2 (segment level) and S_3 (event filter), but get outperformed by their S_1 (frame level). Table 2 shows the scores of our best system with the baselines on the validation set as well as the best system of the winners of the challenge 2022 edition.

5. CONCLUSION

We describe in this technical report the systems we submitted to the DCASE 2023 challenge task 5. Our systems rely on learning a useful feature extractor on the training set using data augmentation and supervised contrastive learning, and training a binary classifier on the positive and negative prototypes on each audio file of the validation/evaluation set, independently. Importantly, we do not use any other data source than the official training set, and our system does not use any postprocessing. Our experiments show that our approach outperforms the baseline by a large margin.

6. ACKNOWLEDGMENT

We would like to thank Vincent Gripon, Yassir Bendou and Yassine El Ouahidi for their feedbacks and discussions about few-shot learning.

7. REFERENCES

[1] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[2] Y. Bendou, Y. Hu, R. Lafargue, G. Lioi, B. Pasdeloup, S. Pateux, and V. Gripon, "Easy—ensemble augmented-shot-y-shaped learning: State-of-the-art few-shot classification with simple components," *Journal of Imaging*, vol. 8, no. 7, p. 179, 2022.

[3] I. Nolasco, S. Singh, E. Vidana-Villa, E. Grout, J. Morford, M. Emmerson, F. Jensens, H. Whitehead, I. Kiskin, A. Strandburg-Peshkin, *et al.*, "Few-shot bioacoustic event detection at the dcase 2022 challenge," *arXiv preprint arXiv:2207.07911*, 2022.

[4] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in neural information processing systems*, vol. 33, pp. 18 661–18 673, 2020.

[5] J. Tang, X. Zhang, T. Gao, D. Liu, X. Fang, J. Pan, Q. Wang, J. Du, K. Xu, and Q. Pan, "Few-shot embedding learning and event filtering for bioacoustic event detection," *iFLYTEK Research Institute, Hefei, China, Tech. Rep*, 2022.

[6] E. Fonseca, D. Ortego, K. McGuinness, N. E. O'Connor, and X. Serra, "Unsupervised contrastive learning of sound event representations," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.

[7] L. Wang and A. v. d. Oord, "Multi-format contrastive learning of audio representations," *arXiv preprint arXiv:2103.06508*, 2021.

[8] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Byol for audio: Self-supervised learning for general-purpose audio representation," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Jul 2021. [Online]. Available: http://dx.doi.org/10.1109/IJCNN52387.2021.9534474

[9] 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.

[10] M. Hertkorn, "Few-shot bioacoustic event detection : Don ' t waste information technical report," DCASE2022 Challenge, Tech. Rep., June 2022.