FEW-SHOT BIOACOUSTIC EVENT DETECTION: ENHANCED CLASSIFIERS FOR PROTOTYPICAL NETWORKS

Ren Li[1], Jinhua Liang[1], Huy Phan[1,2]

[1] School of Electronic Engineering and Computer Science, Queen Mary University of London, United Kingdom
[2] The Alan Turing Institute, United Kingdom
{ml20986, jinhua.liang, h.phan}@qmul.ac.uk

ABSTRACT

Few-shot learning has emerged as a novel approach to bioacoustic event detection since it is useful when training data is insufficient, and the cost of labelling data is high. In this paper, we explore the Prototypical Networks for developing a few-shot learning system to detect mammal and bird sounds from audio recordings. To enhance the deep networks, we use a ResNet-18 variant as the classifier, which can learn the embedding mapping better with stronger architecture. Another method is proposed to focus on domain shift problem during learning the embedding by taking advantage of autoencoders to learn the low-dimensional representations of input data. A reconstruction loss is added to the training loss to perform regularization. We also utilize various data augmentation techniques to boost the performance. Our proposed systems are evaluated on the validation set of DCASE 2022 task 5 and improve the F1-score from 29.59% to 47.88%.

Index Terms— Few-shot learning, sound event detection, Prototypical Networks, embedding space

1. INTRODUCTION

Bioacoustic event detection is the task of recognizing biological sound events present in a set of audio recordings and predicting their time boundaries [1]. This technology is now benefitting from the power of deep learning and becomes an effective way to gain information on the activities of animals that reflects human's impact on the environment [2]. Traditionally, researchers have conducted the work through manually labelling on huge datasets, which is consuming both in time and resources [2]. In addition, collecting labelled data in some certain animal sounds can be challenging, and the scarcity of supervised data can lead to poor generalization and overfitting problem [1].

To address the data scarcity and reduce the cost of labelling data, few-shot learning has been proposed; this approach learns a classifier that can recognize new classes with a limited amount of labelled data [3]. One applicable advantage of few-shot learning is its ability to gain experience from prior similar tasks, so few-shot learning can be characterized as a kind of meta-learning [4]. A meta-learning algorithm gains experience over a set of learning "episodes" and uses this experience to improve its future performance for a new task [4]. For *N*-way-*K*-shot classification, each episode includes *N* classes with *K* examples. For the DCASE 2022 task 5, the first *K*=5 events are used for the class of interest for each test file to detect all the events of this class in the rest of the recording [1].

In recent years, an increasing number of meta-learning approaches for few-shot learning have been proposed and applied in many domains, such as sound event detection, image classification and text classification [4]. Among them, the Prototypical Network (ProtoNet) proposed by Snell *et al.* [3] is simple in principle but effective in practice. The ProtoNet transforms the input into an embedding space where the embeddings are simply clustered to the nearest "prototype" [3]. Therefore, it is desirable for the deep networks to produce adequate embedded features and calculate a useful prototype for each class.

In this work, we propose two enhanced methods to build a stronger ProtoNet. The first method uses a ResNet-18 variant as the embedding features extractor, which is capable of learning and extracting more advanced features with deeper and wider residual networks. The second method merges the ideas from autoencoders and ProtoNet to learn low-dimensional representations and to preserve the information contained in original low-level features. We also apply various spectrogram augmentation techniques to increase the amount of training data for model generalization. Our proposed systems are evaluated on the validation set of DCASE 2022 task 5 and achieve the best F1-score of 47.88%.

2. RELATED WORK

The use of convolutional neural layers allows feature extractor to extract complex features that express the raw data in much more detail and learn representations more efficiently. However, as the layers get deeper, the learned features can deteriorate due to vanishing gradient, leading to performance deterioration [5]. For this reason, residual networks (ResNets) were proposed and widely applied in deep learning tasks [5]. Sharma *et al.* [6] used a pre-trained ResNet-50 model for bird song classification, producing an accuracy of 97.1%, which was far superior to that of the VGG16 model. Soumya *et al.* [7] improved the ProtoNet using a customized ResNet as the feature embedding network for facial emotion recognition and proved its capability of extracting minute details.

In addition, most few-shot learning methods can suffer from domain shift problems during learning the embedding [8]. Since the embedding is only learned from the seen classes, when performing testing with the unseen classes, the embedding features are likely to be shifted due to the bias of the seen classes used for training [8]. Sometimes it can make the query data points far away from the correct corresponding unseen class prototypes, thus affecting the accuracy of the *k*-NN search. An effective Semantic Autoencoder (SAE) [8] is proposed to solve this problem by adding a reconstruction constraint to learn the low-dimensional representation. Moreover, Liu *et al.* [9] improved the SAE using graph structure and another L2-norm constraint, which preserves

the intrinsic data structure and has more discriminating power. Inspired by the ideas of above works, we enhance the ProtoNet for the few-shot bioacoustic event detection.

3. METHOD

This section introduces our methods developed upon the baseline system, including model design and data augmentation. First, we present a modified version of ResNet-18 used as a feature embedding network. After that, we describe how to combine autoencoders to perform data reconstruction from embeddings for better generalization to new unseen classes. Finally, we describe spectrogram augmentation techniques to boost the system's performance.

3.1. ResNet-based prototypical network

For the task of detecting bird and mammal sounds, it is important for the embedding module to extract adequate features since the sound samples are often too short and imperceptible to detect and distinguish them. However, the embedding module of the original ProtoNet only consists of four Conv blocks. If we use the original ProtoNet with multiple Conv blocks to learn the sound's features, it is prone to encounter gradient vanishing problems, which reduces the quality of the embeddings.

We thus choose residual networks as the embedding encoder, which can avoid the vanishing gradients thanks to skip connections. The skip connections add the output from a preceding layer to a later layer, allowing information to get fast-forwarded and go deeper with less deterioration [6]. Another salient feature of ResNets is the use of batch normalization (BN) to normalize the input of the activation function of the previous layer, which helps mitigate the covariate shift problem [6].

Our implementation is based on the ResNet-18. We modify the network to obtain a less deep model which only has 3 residual blocks to fit the size of the features. Each residual block contains 3 convolution layers using a kernel size of 3 × 3, followed by a batch normalization and a Leaky ReLU activation. Importantly, a shortcut with a 1 × 1 convolutional layer is added over the 3 layers. The architecture of our residual network is shown in Table 1, while Table 2 shows the architecture of the original embedding module, which provides a comparison.

Table 1. Architecture of the presented residual network

Encoder		Residual Block	
Layers	Channels	Layers	Kernel
Conv2D+ BN +ReLU	16	Conv2D+ BN +ReLU	3 × 3
Residual Block	64	Conv2D+ BN +ReLU	3 × 3
Residual Block	128	Conv2D+ BN	3 × 3
Residual Block	64	Shortcut: Conv 2D+BN	1 × 1
Adaptive AvgPooling+SoftMax	-	ReLU+MaxPool-ing+Dropout	1 × 1

Table 2. Architecture of original encoder and Conv block

Encoder		Conv Block	
Layers	Channels	Layers	Kernel Size
Conv Block	64	Conv2D	3 × 3
Conv Block	64	BatchNorm	-
Conv Block	64	ReLU	-
Conv Block	64	Max pool	2 × 2

3.2. Combination of Autoencoder and ProtoNet

To overcome the domain shift problems described in Section 2, we enhance the network based on the encoder-decoder paradigm. The encoder compresses the spectrograms of input data into an embedding space while the decoder reconstructs the expected original input features from the embedding space [10]. The output of the decoder is then compared with the original input features. This additional reconstruction task imposes a new constraint in learning the input features that guarantees the embedding features preserve more distinctive information contained in the original input features [8]. Therefore, it is effective in mitigating the domain shift problem. Although the appearance of features changes from seen classes to unseen classes, the demand for a more truthful reconstruction of the input features is unchanged; thus, the embedding function is generalizable across seen and unseen domains [8].

Inspired by AutoProtoNet proposed by Sandoval-Segura *et al.*[11], we use the 4 original sequential convolution blocks for the encoder and 4 sequential transpose convolution blocks for the decoder. The transpose convolution blocks are utilized to reproduce the high-dimensional low-level features by deconvolutional operations. The detail of these blocks is displayed in Table 2. Each convolutional block consists of a Conv2D layer, a Batch Normalization, a ReLU activation and followed by a Max Pooling layer with pool-size of 2×2. Each transpose convolutional block is made up of a transpose layer, a Batch Normalization layer, and a Conv2D layer and followed by a ReLU activation.

Table 2. Components of Conv Block and Transpose Conv Block

Conv Block		**Transpose Conv Block**	
Layers	Kernel Size	Layers	Kernel Size
Conv2D	3 × 3	Conv2D Transpose	2 × 2, stride 2
BatchNorm	-	BatchNorm	-
ReLU	-	Conv2D	3 × 3
Max pool	2 × 2	ReLU	-

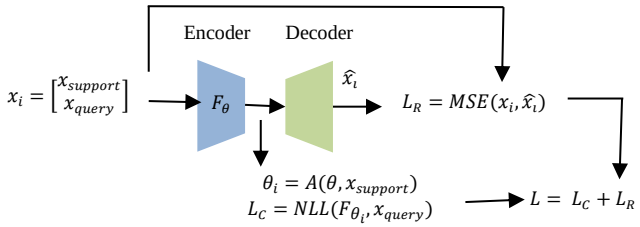

Figure 1: Overview of the forward pass through the autoencoder model

The training procedure of the autoencoder model is based upon the original training framework presented by Snell *et al* [3]. The main improvement is that we tailor the training loop with a reconstruction loss to regularize the embedding features and purpose it to preserve more useful details. The overview of the forward pass through the autoencoder is shown in Figure 1.

In the new training framework, the support data $x_{support}$ and query data x_{query} are randomly sampled from the current episode in form of 5 classes with 5 examples. They are then passed through the encoder and decoder to produce a reconstruction set

$\hat{x}_i$. This reconstruction set is then compared with the original input features x_i using mean square error (MSE) loss [11], defined as:

$$MSE = (x_i - \hat{x}_i)^2. \quad (1)$$

The finetuning algorithm A computes a set of prototypes p_k for each class k by computing the class-wise mean of embedded support examples and updates the model's parameters [3], and both are contained in θ_i. Eq. (2) defines the computation of prototypes:

$$p_k = \frac{1}{|S_k|} \sum_{x \in S_k} F_\theta(x), \quad (2)$$

where the S_k denotes the set of support samples for class k and x denotes the embeddings of class k.

Given the Euclidean distance function d and a set of query sound samples, the ProtoNet produces a probability distribution over classes for a query sample x belonging to true class k by Eq. (3) [3]. Then the training proceeds by minimizing the negative log-likelihood (NLL) of the true class k by Eq. (4). Finally, the classification loss L_C and the reconstruction loss L_R are summed to jointly optimize the training.

$$p_\theta(y = k|x) = \frac{\exp(-d(F_\theta(x), p_k))}{\sum_{k'} \exp(-d(F_\theta(x), p_{k'}))} \quad (3)$$

$$L(\theta) = -log\, p_\theta(y = k|x). \quad (4)$$

3.3. Data augmentation

In order to increase the diversity of data and the generalization ability of the model, we use *SpecAugment* [12] as the data augmentation technique. It essentially consists of three transformations: time warping, frequency masking, and time masking. Specifically, they modify a spectrogram by warping it in the time direction with a distance factor, masking blocks of consecutive frequency channels, and masking blocks of time steps, respectively [12]. In our case, we warp the feature to the left by 0.5 s and mask one block of one frequency mel bin and one block of 10 time steps. We choose these values according to the size of the time-frequency representation, which is appropriate to produce the diversity of the training data. If the values are too large or small, the augmented features could be very different from the originals or not changing enough; thus, the model is unable to achieve improved performance.

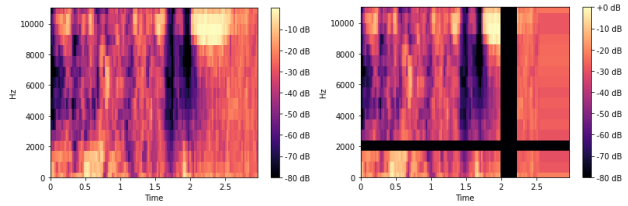

Figure 2: Example of spectrogram augmentation: orginial spectrogram (left) and augmented spectrogram with SpecAugment (right).

4. EXPERIMENTS

4.1. Dataset

The DCASE 2022 challenge provides a development set which is predefined as a training set and validation set. They were acquired from multiple bioacoustic sources, including sounds of worldwide birds, spotted hyenas, jackdaws, meerkats, and wetlands birds [1]. As a result, the sounds can be long or very short across the subsets; the sampling rate of each audio varies from 6 kHz to 44 kHz [1]. The training set consists of 174 audio recordings, 47 classes and 14,229 event instances. In addition, multi-class annotations are provided for the training set with positive, negative, and unknown; we only extracted and made use of the positive event instances for training. The validation set consists of 18 audio recordings, 5 classes and 1,077 positive event instances [1].

4.2. Data pre-processing

Mel-spectrogram. All audio files in both the training set and validation set were first resampled to a sampling rate of 22,050 Hz. The audio files were then transformed to Mel-spectrograms with 128 Mel bins using an FFT size of 1024 samples and a hop size of 256 samples. The *librosa* library was employed for this purpose. Afterwards, spectrogram images of size $F \times T$ where F=17 by T=128 were used as inputs.

PCEN. PCEN has been proposed to normalize a time-frequency representation by performing automatic gain control, followed by nonlinear compression [13]. Former research used PCEN to mitigate the effects of background noise, demonstrating its effectiveness as a preprocessing step prior to convolutional methods in sound event detection [13]. Bioacoustic data recorded in the wild often have multiple sound sources and uncleaned background. Therefore, we utilized PCEN to reduce noise presented in the Mel-spectrograms and improve robustness to channel distortion.

4.3. Training

Prototypical networks adopt an episodic training procedure where in each episode, a mini batch is randomly sampled from the training data [4]. A subset of mini batch was used as the support set and the remaining is used as query set. The models were trained with 2,000 episodes and 5 classes in each minibatch with the Adam optimizer and the learning rate of 0.001. Euclidean distance was selected as the metric that measures the distance between query samples to a prototype.

4.4. Post-processing

The preliminary experiments confirms that the task of detecting bioacoustics events from nature is challenging; most classification methods can produce a large number of false-positive predictions, substantially reducing the model's F1-score [14]. Therefore, we applied post-processing to the outputs to remove possible false positives. Specifically, we removed the predictions that were shorter than 20% of the average duration calculated by the first 5 shots for each audio file. It was because participants were expected to treat the task as a 5-shot setting.

Table 3. Comparison of models using different classifier and feature. The best results are highlighted in **boldface**.

Model components			Validation set scores (%)			Subset F1-score (%)		
Exp No.	Classifer	Feature	F1-score	Precision	Recall	HB	ME	PB
1	CNN (Baseline)	PCEN	29.59	36.34	24.96	/	/	/
2	ResNet	PCEN	45.64	48.34	43.22	50.00	57.14	26.18
3	Autoencoder	PCEN	37.94	38.95	36.97	44.53	52.05	25.68
4	CNN	PCEN+Augment	37.16	42.09	33.26	38.86	**72.01**	15.33
5	ResNet	PCEN+Augment	**47.88**	**52.11**	**44.30**	**53.45**	**50.98**	**17.65**
6	Autoencoder	PCEN+Augment	**47.61**	**50.18**	**45.34**	**52.68**	**53.10**	**22.44**

5. RESULTS AND DISCUSSIONS

We conducted several ablation experiments on the validation set to verify the effectiveness of the components and tricks in our proposed models. To further investigate the capabilities of our models, we computed the F1-score for three difference subsets. The experiments results are shown in Table 3. The brief information about each subset is as follows: the HB subset records the mosquito's events that are very long with low noise; the ME subset contains the sounds of meerkats that are short with low noise; the PB subset records the bird flight calls that are very short and unclear with high noise.

5.1. Effects of using ResNets

From the results of Experiments 1 and 2, it can be seen that the ResNet model outperforms the baseline CNN and achieves a noticeable improvement of over 15%. It is primarily due to the ResNet's deep architecture, which has many more parameters to capture the features better, allowing the learned features to fit the input data better. In addition, the residual networks make use of skip connections, enabling the model to carry gradients to a very deep layer. It also allows the model optimally tuning the number of the layers during training, so that the model parameters can be updated more optimally. However, since it adopts a complex networks architecture, the computation and memory cost increase intensively. To compare the model complexity, we measure the number of trainable parameters for different models as shown in Table 4. Compared to CNN with 112k parameters, the number of parameters for ResNet has grown significantly to 724k, and the number of parameters for the autoencoder is roughly twice that of the CNN.

Table 4. The number of parameters for different models

Model	Parameters
CNN (Baseline)	111,936
ResNet	724,096
Autoencoder	272,717

5.2. Effects of adding reconstruction loss

As shown by the results of Experiments 1, 2 and 3, the autoencoder model also improves the performance but with a slightly lower gain compared to that of the ResNet model. By adding a reconstruction loss, it is likely that the autoencoder model can learn the high-level low-dimensional representation and preserve more useful details, resulting in a better generalization to unseen classes for few-shot learning task. However, the reconstruction loss is served as a constraint during learning, so its influence is limited. Making an embedding classifier to learn the representation of the input data in a fundamentally different way is more meaningful and challenging. That could be the reason why the ResNet model outperforms the autoencoder in this case.

5.3. Effects of data augmentation

Applying *SpecAugment* has been the popular choice for sound event detection. When training data is unbalanced and insufficient, it has shown to be effective. Overall, this technique also workes well for our systems. Despite being augmented by *SpecAugment*, the performance of detecting the very short bird sounds in PB subset decreases. The reason could be that since the masking and warping were randomly applied from a uniform distribution over the value of factors, the blocks and warping steps could be too excessive in some cases. This could lead to some useful information in the minor sounds being masked, or even the augmented features became quite different from the originals, thus preventing the networks from learning. It gets even worse under high-noise conditions. Furthermore, the model trained on CNN and augmented data was observed to achieve much higher results for the ME subset than other approaches. This is because ResNet and autoencoder have many more parameters to optimize, likely leading to overfitting the training set. In contrast, the CNN with a simple architecture is just capable of generalizing better to ME data.

6. CONCLUSION

In this paper, we present two solutions to enhance Prototypical Networks for the task of bioacoustics sound event. We show a list of ablation studies and discussed the effects of each component or trick used in our systems. Overall, using ResNets and autoencoder (or construction loss) contribute to learn a more adequate embedding space and boost the model's performance. Joining with data augmentation techniques, our enhanced models achieve the best F1-score of 47.88%, which improves over the baseline by a large margin.

Our studies also imply that it is challenging to detect the very short and unclear bioacoustics sound events. This can be an important subject to be explored in the future work. Recent research show that *model adaptation* is an effective solution to improve the general robustness of SED method [15].

7. REFERENCES

[1] https://dcase.community/challenge2022/task-few-shot-bioacoustic-event-detection

[2] C. Chalmers, P. Fergus, S. Wich and S. N. Longmore, "Modelling Animal Biodiversity Using Acoustic

Monitoring and Deep Learning," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1-7.

[3] J. Snell, K. Swersky, R. Zemel, “Prototypical networks for few-shot learning,” Advances in neural information processing systems. 2017, vol. 30.

[4] Z. Zhan, J. Zhou, and B. Xu, “Fabric defect classification using prototypical network of few-shot learning algorithm,” Computers in Industry. 2022, vol. 138, p.103628.

[5] C. S. Wickramasinghe, D. L. Marino and M. Manic, "ResNet Autoencoders for Unsupervised Feature Learning From High-Dimensional Data: Deep Models Resistant to Performance Degradation," *IEEE Access*, 2021 vol. 9, pp. 40511-40520.

[6] N. Sharma, A. Vijayeendra, V. Gopakumar, P. Patni and A. Bhat, "Automatic Identification of Bird Species using Audio/Video Processing," in *2022 International Conference for Advancement in Technology (ICONAT)*, 2022, pp. 1-6.

[7] K. Soumya and S. Palaniswamy, "Emotion Recognition from Partially Occluded Facial Images using Prototypical Networks," in *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 2020, pp. 491-497.

[8] E. Kodirov, T. Xiang, and S. Gong, “Semantic autoencoder for zero-shot learning,” *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3174-3183.

[9] Y. Liu, D.Y. Xie, Q. Gao, J. Han, S. Wang, and X. Gao, Graph and autoencoder based feature extraction for zero-shot learning. In IJCAI, 2019, vol.1(2), pp.6.

[10] S. Hong and S. -K. Song, "Kick: Shift-N-Overlap Cascades of Transposed Convolutional Layer for Better Autoencoding Reconstruction on Remote Sensing Imagery," *IEEE Access*, 2020, vol. 8, pp. 107244-107259.

[11] P. Sandoval-Segura, W. Lawson. AutoProtoNet: Interpretability for Prototypical Networks. arXiv preprint arXiv:2204.00929. 2022 Apr 2.

[12] 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. 2019, arXiv preprint arXiv:1904.08779.

[13] C. Ick and B. McFee, "Sound Event Detection in Urban Audio with Single and Multi-Rate Pcen," *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 880-884.

[14] V. Morfi, I. Nolasco, V. Lostanlen, S. Singh, A. Strandburg-Peshkin, L.F. Gill, H. Pamula, D. Benvent, and D. Stowell. Few-Shot Bioacoustic Event Detection: A New Task at the DCASE 2021 Challenge. In DCASE, 2021, November, pp. 145-149.

[15] A. Mesaros, T. Heittola, T. Virtanen and M. D. Plumbley, "Sound Event Detection: A tutorial," in *IEEE Signal Processing Magazine*, 2021, vol. 38, pp. 67-83.