Few-Shot Bioacustic Event Detection through Deep Learning and Bayesian Optimisation

Francesco Stella
Department of Computer Science
University of Milan, Milan, Italy
francesco.stella@studenti.unimi.it

***Abstract*—Sound Event Detection (SED) and classification is a challenging task aimed at automatically identifying and annotating events of interest within a sound recording. Typically, in supervised contexts few data is available due to the expensiveness of the manual annotation task, thereby few-shot learning (FSL) methods are commonly used to counter data scarcity and obtain a model capable of generalising from few examples. In this work, the DCASE 2023 bioacustic SED task on mammal and bird vocalisations is performed by means of two models, namely a Siamese Network (SN) and a Relation Network (RN), both sharing a common embedding module selected through a state-of-the-art algorithm based on Bayesian Optimisation. Mel-spectrograms are generated from audio signals and improved through Per-Channel Energy Normalisation (PCEN), that suppresses background noise while enhancing foreground sounds. Results for training, validation and finetuning seem promising, although further investigations must be made concerning the preliminary validation performed using the DCASE script.**

***Index Terms*—SED, Few-Shot Learning, BOHB, Bioacustics**

1. Introduction

Sounds are ubiquitous in everyday life, typically coming from multiple sources simultaneously and providing useful information about the environment. In Sound Event Detection (SED) methods, also referred as Audio Event Detection (AED), automatic means are adopted in order to detect and recognise events in a sound signal, thus enabling, e.g., urban soundscape, biodiversity or underwater ambient monitoring for risk assessment [1], [2], [3] among many other possible uses. Typically, a distinction is made between SED and *speech-* or *music-recognition* tasks due to the intrinsic differences in the sound characteristics [1].
In the field of Bioacustics, SED methods can be used to monitor through sensor networks the wildlife of a specific region over long period of times, thus enabling viable, scalable and effective species detection and annotation [2], although some major challenges specifically related to this field must be addressed. For example, background noise represents a factor that hinders the classification performance of SED algorithms.
Since the manual detection and annotation of relevant sound events is highly time-consuming [4], Few-Shot Learning (FSL) applied to SED represents a very effective way to build models capable of generalising from few examples. In fact, Machine Learning algorithms typically require large amount of data in order to learn a model capable of generalising to new instances. FSL applied to SED is commonly conducted as a Few-Shot Image Classification (FSIC) task on the extracted features, typically consisting of spectrogram-based features, such as log-Mel Spectrograms or Per-Channel Energy Normalisation (PCEN). In this work, experiments involving the extraction of log-Mel Spectrograms and PCENs and the training of two type of deep networks are performed. Moreover, although with some limitations, two novel approaches are explored, namely model selection through Bayesian Optimisation and HyperBand (BOHB) and the training of the embedding module through the Triplet Loss. To the best of my knowledge, experiments using Bayesian Optimisation or Triplet Loss have not been performed in the SED task for the bioacustic domain.

2. Related Work

Deep learning techniques are largely adopted in SED tasks due to their promising capabilities of learning useful and complex features. In the context of few-shot bioacustic SED, typical approaches consist of *meta-learning*, that leverages past experience to foster learning (e.g. through prototypical networks), and *transfer-learning* that involves the transfer of knowledge by means of finetuning from a large dataset to a smaller one containing new classes [5]. With regard to the architectures, CNNs are commonly used to learn useful features, for example in [6] four CNNs are adopted to extract feature vectors from PCEN inputs, while a self-attention Transformer is used in conjuction with a fully connected layer to provide the prediction. This system achieved the best performance in the DCASE 2023 challenge. Differently from the previous approach, in [7] the meta-learning framework is adopted, using the Bidirectional Encoder representation from Audio Transformers (BEATs) [8] as a feature extractor, followed by a prototypical network aiming at learning prototypes in a latent space. Mel-filter banks are used as inputs to the BEATs model and euclidean distance between prototypes and the embedded sample is used to perform the prediction. A different strategy is followed in [9], where transfer learning is adopted using the ResNet model, consisting of nine CNNs, as feature extractor and Mel spectrograms as input features. Moreover, data augmentation is performed and the Supervised Contrastive Learning (SCL) framework is adopted during training, whose loss tries to separate the class representations in the latent space. Similarly to [8], an Audio Spectrogram Transformer (AST) is adopted leveraging Mel-spectrograms in [10] with the aim of learning to generalise over sounds from different animal species. Instead, an increasingly adopted approach entailing Discrete Wavelet Transform (DWT) is proposed in [11] with regard to the medical acoustics field, although bioacustics SED is among the potential applications. A siamese network is trained on features extracted through a three-level DWT from respiratory sounds, leveraging the multiresolution analysis enabled by the wavelets, which provide details useful to characterise the audio signals at different levels.

3. System Overview

Two network architectures are devised in this work, namely a Siamese Network (SN) and a Relation Network (RN), both sharing a common embedding module consisting of 3 CNNs. The choice of the networks depends on the promising results of the contrastive learning and metric learning approaches, where the latter has been shown to reduce overfitting and often achieved state-of-the-art performance in the few-shot setting [12]. SN and RN both estimate the distances between input images, the former using a fixed metric, specifically the Euclidean distance, hence belonging to the contrastive learning framework, while the latter contains a module that learns a metric during training, thus providing greater generalisation capabilities. A simplified version of the architecture of SN is shown in Figure 1, while the architecture of RN is shown in Figure 2. More accurate representations of the deep networks are provided in the project's repository on GitHub[1].

With regard to the embedding module, it is trained using Triplet Loss [13], which has been shown to be useful for learning meaningful representations, e.g. in the face recognition task [14]. More precisely, Triplet Loss requires three inputs, namely an *anchor* image, a *positive* image that has the same label (belongs to the same class) as the anchor and a *negative* image, which has a different label than the previous two. These inputs are then passed through the embedding module and the learned representations are then used to estimate the *triplet loss*, computed as shown in equation 1, where x_i^a, x_i^p and x_i^n are, respectively, the anchor, the positive and the negative images belonging to the i-th triplet, f is the embedding module, while α is the margin, a positive real number that enforces a minimum distance between positive and negative images.

$$L = \sum_i^N [||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha]_+ \quad (1)$$

The final goal is that given a learned representation of an anchor image, this must be closer to the representations of the positive images than to any negative image representation. In order to achieve this result, different strategies have been designed to generate the triplets of inputs and hence compute the loss. More specifically, a distinction can be made between *easy*, *semi-hard* and *hard* triplets, in which the former entails the fact that the anchor is closer to the positive example than to the negative one, that does not violate the margin. The semi-hard triplets are characterised by a negative example that is farther from the anchor than the positive example, although it violates the margin. Finally, the hard triplets are generated selecting negative examples that are closer to the anchor than the positive ones. Triplets are generated using *online mining* (i.e. on-the-fly) in order to avoid negatively impacting performances and this is achieved using the PyTorch Metric Learning framework [15], which provides the implementation of several metric learning losses and mining strategies.

3.1. Feature Extraction

Two kinds of features (sections 3.1.1 and 3.1.2) are extracted from resampled time-domain signals corresponding to the events of interest. More precisely, first the channels of the time-domain signals are merged computing their mean, the signals are then normalised by subtracting the mean and dividing by the standard deviation and then they are resampled to 22050 Hz, finally chunks corresponding to the hard labeled parts are extracted from the resulting signals. Chunks are then cut to 1 second (if they exceed this threshold) in order to ensure inputs with the same size, then they are centered on the time axis. Similarly, shorter signals are padded with zeros or random noise (zero mean and unit variance) in order to reach the one-second duration. Subsequently, a Short-Time Fourier Transform (STFT) is applied using the torchaudio Python package[2] with the main parameters set as shown in table 1. The power parameter is set to 1, meaning that magnitude instead of power is estimated, while the chosen window is the Hann window, a symmetric and sinusoidal function that reduces the spectral leakage and better preserves spectral resolution compared to other commonly used windows.

Afterwards, the obtained spectrogram is transformed into the Mel scale using 128 mel filterbanks. Regardless of the selected feature (i.e. PCEN or log-Melspec), augmentation consisting in time and frequency masking is applied to the training data in order to prevent overfitting.

3.1.1. Per-Channel Energy Normalisation. Per-Channel Energy Normalisation (PCEN) is a dynamic procedure originally designed for the far-field keyword spotting task [16],

1. https://github.com/FrancescoStella1/Few-Shot-Bioacustic-SED

2. https://pytorch.org/audio/stable/index.html

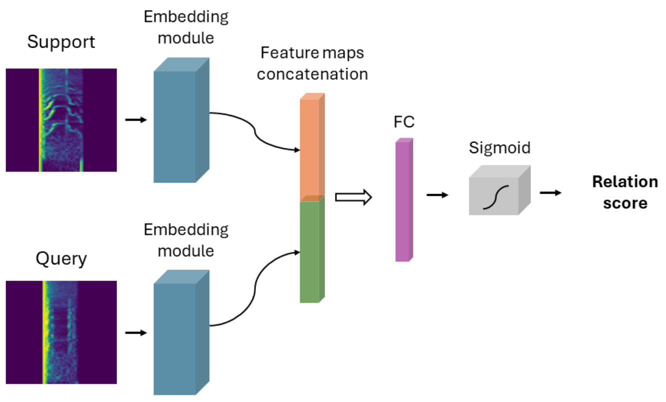


Figure 1: Siamese Network architecture.

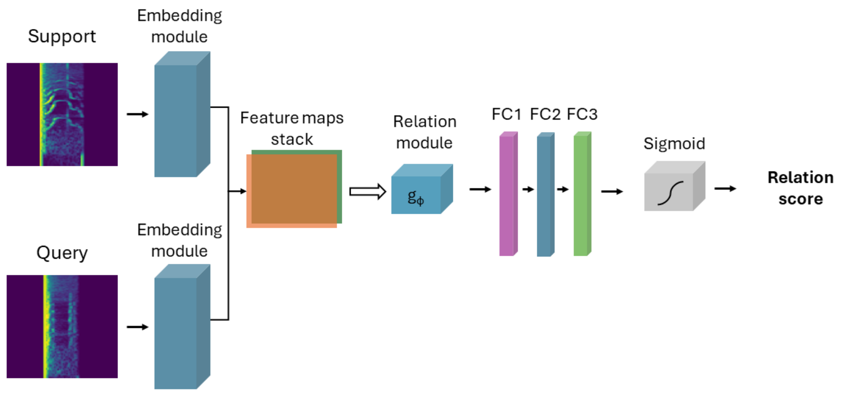


Figure 2: Relation Network architecture.

Parameter	Value
n_fft	1024
win_length	1024
hop_length	256
window_fn	Hann window
power	1
normalised	False

TABLE 1: Main parameters of the STFT applied to obtain the spectrograms.

shown to be an effective and robust means to suppress background noise while enhancing animal vocalisations [17], outperforming other features such as spectral flux or log-mel spectrograms [17], [16]. More details about PCEN are provided in Appendix A.

3.1.2. Log-Mel Spectrogram. Log-Mel Spectrograms lean their foundations on the Mel-scale, a perceptually motivated logarithmic scale. Log-Mel Spectrograms turned out to be effective in SED tasks and are largely adopted. In this work, log-Mel Spectrograms are extracted from audio signals, however PCENs are used for training and finetuning since upon the completion of a few experiments they seemed to provide better performance, enabling a better feature space clustering.

4. Experimental Set-up

4.1. Dataset

The dataset is available on the DCASE 2023 Challenge website, in the task 5 section[3]. The dataset consists of an aggregation of other datasets containing 192 field recordings of hard-labeled mammals and bird vocalisations. The dataset is pre-splitted into training and validation sets, the former containing multi-class annotations divided into positive (POS), negative (NEG) and unknown (UNK) events, whereas the latter provides only single-class annotations with positive (POS) or unknown (UNK) events. Moreover, an evaluation set containing 66 audio files from different sources is provided in order to test the chosen models. In validation and evaluation sets, only the first 5 shots of each class/sound recording can be used to fine-tune the model or compute the prototypes in order to perform the inference with the remaining part of the recording. The training split contains 46 classes and it is unbalanced, as Figure 3 shows.

4.2. Model Selection

Model selection is performed through the BOHB algorithm, in order to find promising hyperparameter configurations for the embedding module. The Ray Tune framework[4] is used to implement the search for the embedding module and the RN, while SN does not require further model selection due to structural reasons. Due to time constraints, for the purpose of this work the search is performed only on the embedding module and in a limited way, sampling 20 configurations from the space and training the resulting models for 6 epochs on the training split, using the Triplet Loss and the validation split to evaluate the models. Further details on the BOHB algorithm are given in appendix B. The search space consists in parameters such as dropout values, kernel sizes, output channels (for CNNs) or features (for FC layers), batch normalisation momentum and weight decay for the optimiser, while their bounds are pre-specified using suitable functions.

4.3. Training

The training consists in two phases, namely the actual training on the larger training set and the finetuning on the validation set, where the latter follows a 5-way 5-shot approach. During training, the dataset is divided into two other splits, namely a training split and a validation split, with ratios 0.7 and 0.3 respectively, in order to collect metrics on the validation split and perform early stopping accordingly. It is important to note that this validation split is different from the validation set provided by the DCASE challenge, which instead requires to follow a 5-way 5-shot task and it is used for finetuning the models. Moreover, during the training of the embedding module, several values for the margin of the Triplet Loss, different types of triplets and two optimisers, namely SGD and NAdam, are tested in order to find the best configurations. A margin equal to 0.2, all possible triplets (i.e. hard, semi-hard and easy ones) and the NAdam optimiser with learning rate equal to 1e-4 are identified as the best values, since the learned feature space can be better clustered, although overlapping between clusters is present (see Figure 4). Batch size is fixed to 256 samples for the training of the embedding module, while the maximum number of epochs is fixed to 150 and early stopping is implemented using the validation loss as the metric to minimise with a delta value of 2e-4 and a patience of 50 epochs. After training, feature maps obtained from the first eight channels of each of the three CNN layers of the embedding module are visualised for a few classes, in order to foster an intuitive understanding of the extracted features. Figure 8 shows such an example for class *c_6* from the training set.
Training of SN and RN is performed freezing the weights of the embedding module, since this resulted in better performance. Also, the training of SN and RN involves a custom sampler useful to mitigate the class imbalance problem and it is performed as a binary classification task, using Binary Cross-Entropy (BCE) Loss. More precisely, in each mini-batch 40 classes are sampled and 6 pairs per class are generated. Each pair consists of a support and a query example, 3 pairs contain positive queries (i.e. support and query examples belong to the same class), while 3 pairs contain negative queries. Negative examples for queries are generated using annotated events from other classes and randomly sampling chunks of the audio signals from two annotated positive events. Thus, the assumption that between any two positive events (POS) there is a negative one (NEG) is made. Importantly, unknown (UNK) events may be present in these intervals and it cannot be excluded that these are actually POS events, hence they are preemptively discarded as a precautionary measure. Overall, this approach should foster the capability of the models to discriminate between similar and dissimilar examples. Training and validation loss histories for the three models are provided in Figure 5, while Table 2 provides metrics collected on the validation split during training. Furthermore, the Receiver Operating Characteristic (ROC) curve and the Area Under the ROC Curve (AUC) are estimated for both SN and RN on the validation split, as shown in Figure 6 and Figure 7.

Model	Loss	Accuracy	Precision	F1 score
SN	0.212	0.931	0.857	0.906
RN	0.196	0.938	0.878	0.908

TABLE 2: Training metrics estimated on the validation split. The loss is the Binary Cross-Entropy (BCE) loss.

4.4. Finetuning

Finetuning is performed on the validation set, which contains five classes and each class has five examples available, as required by the FSL framework, more precisely

3. DCASE Challenge task 5
4. Ray Tune

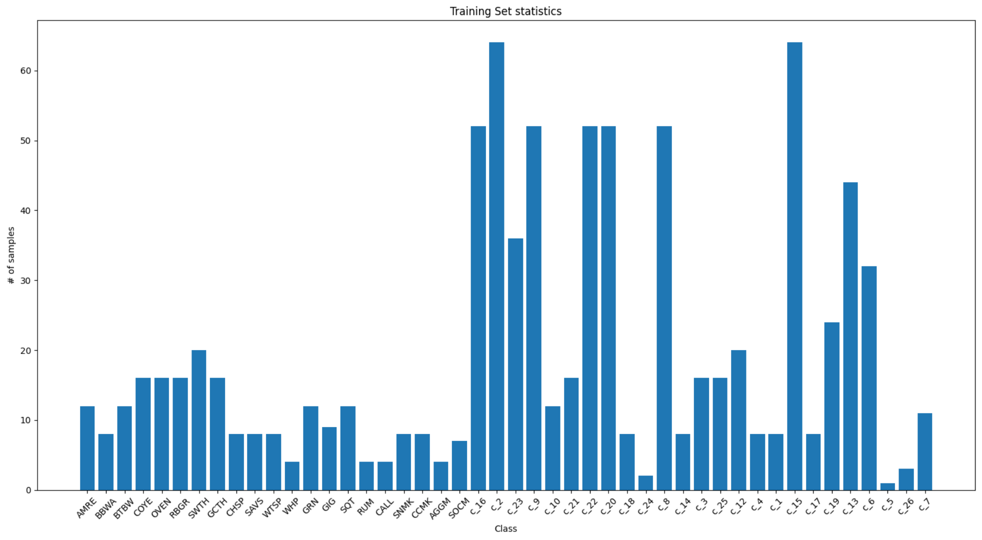


Figure 3: Class distribution in the training split of the dataset.

Model	Loss	Accuracy	Precision	F1 score
SN	0.315	0.864	0.903	0.884
RN	0.343	0.824	0.934	0.838

TABLE 3: Finetuning metrics collected on the validation dataset, following the 5-way 5-shot task.

following the 5-way 5-shot approach. During finetuning, five random pairs for each class are considered, three with positive queries and two with negative ones. SN and RN are then trained for 20 epochs, using early stopping with patience equal to 10. Finetuning metrics for SN and RN are shown in Table 3, while complete history comprising accuracy, precision and f1 score is available on the project's GitHub repository.

5. Results

Evaluation relies on the event-based F1 score and is performed on the validation dataset (which is used as a surrogate for the final evaluation set) using the evaluation script available on the DCASE 2023 challenge website. Results for SN and RN are shown in Table 4 and Table 5, while a comparison of the overall event-based F-scores for SN, RN and three other models submitted to DCASE 2023 is provided in Table 6.

6. Conclusion

In this work, two deep networks are proposed for the bioacustic SED task in the FSL framework. Bayesian optimisation is adopted through the BOHB algorithm in order to perform model selection on the embedding module and the obtained model is then used in the SN and RN models. Training is performed separately for the embedding module and then for SN and RN, freezing the weights of the embedding module and implementing early stopping in order to prevent overfitting, using the validation split's loss as reference. Moreover, the class imbalance problem is addressed through sampling techniques and metrics consisting in loss, precision, accuracy and F1-score are collected during all the phases. Despite the promising results obtained during training, validation and finetuning, the preliminary evaluation on the validation set provided by the DCASE challenge leads to questionable results, perhaps due in part to problems with the storing of the predicted annotations, that must be further investigated.
Finally, some of the possible improvements or experiments involve a more exhaustive search in the hyperparameter space, the adoption of alternative losses for the embedding module training and the extraction of other features to feed the network, e.g. using DWT.

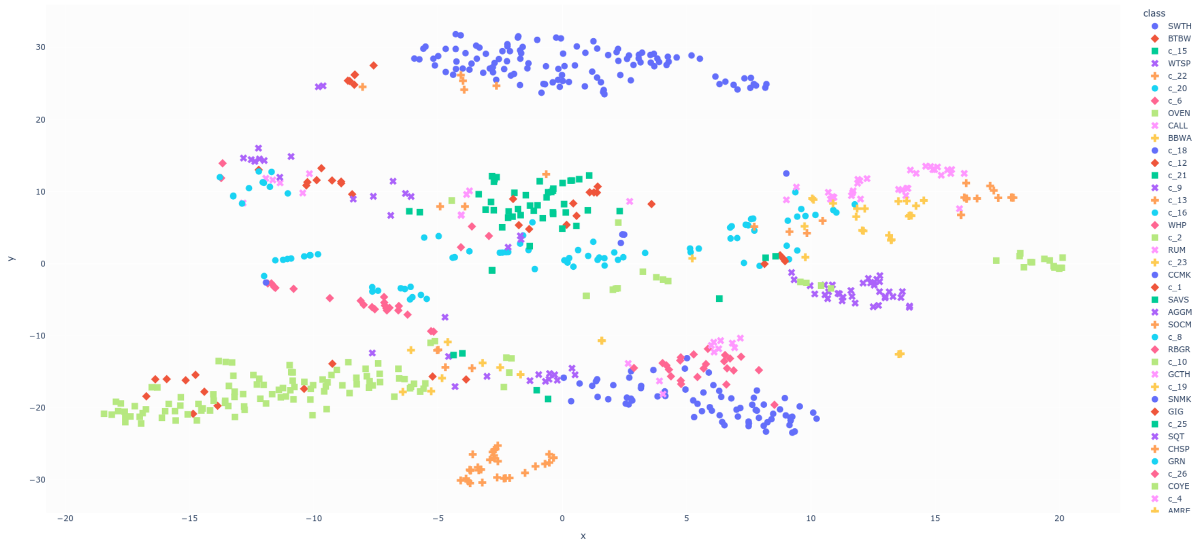


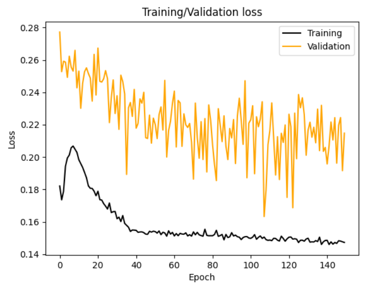
Figure 4: Feature space learned by the embedding module using the Triplet Loss. PCA followed by t-SNE are used for performing dimensionality reduction and feature visualisation.

	Siamese Network			Relation Network		
Filename	Precision	Recall	F-measure	Precision	Recall	F-measure
ME1	0.05	0.36	0.095	0.27	0.27	0.27
ME2	0.66	0.15	0.24	0.87	0.34	0.49
R4_cleaned recording_16-10-17	0.06	0.37	0.10	0.03	0.13	0.04
R4_cleaned recording_TEL_24-10-17	0.40	0.41	0.41	0.28	0.97	0.44
R4_cleaned recording_TEL_25-10-17	0.41	0.41	0.41	0.21	0.91	0.34
R4_cleaned recording_17-10-17	0.27	0.61	0.38	0.06	0.19	0.09
R4_cleaned recording_TEL_20-10-17	0.36	0.08	0.13	0.46	0.73	0.57
file_97_113	0.07	0.2	0.10	0.08	0.2	0.11
file_423_487	0.30	0.79	0.44	0.24	0.86	0.37
R4_cleaned recording_TEL_23-10-17	0.37	0.89	0.53	0.33	0.90	0.49
R4_cleaned recording_TEL_19-10-17	0.40	0.54	0.46	0.17	0.78	0.28
R4_cleaned recording_13-10-17	0.625	0.53	0.57	0.11	1.00	0.19
BUK5_20180921_015906a	0.00	0.00	0.00	0.00	0.00	0.00
BUK5_20161101_002104a	0.5	0.02	0.04	0.01	0.06	0.01
BUK4_20171022_004304a	0.11	0.06	0.08	0.17	0.06	0.09
BUK4_20161011_000804	0.00	0.00	0.00	0.00	0.00	0.00
BUK1_20181013_023504	0.00	0.00	0.00	0.00	0.00	0.00
BUK1_20181011_001004	0.00	0.00	0.00	0.00	0.00	0.00

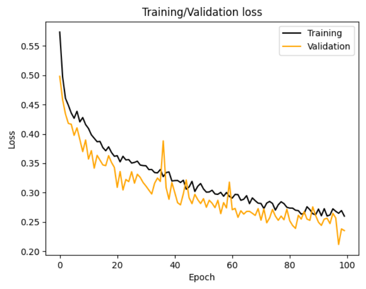
TABLE 4: The audiofile-based results provided by the DCASE script.

	Siamese Network			Relation Network		
Subset	Precision	Recall	F-measure	Precision	Recall	F-measure
ME	0.12	0.19	0.15	0.63	0.37	0.43
HB	0.24	0.46	0.31	0.20	0.68	0.30
PB	0.02	0.01	0.01	0.01	0.02	0.01

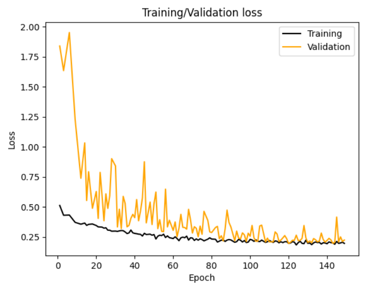
TABLE 5: The subset-based results provided by the DCASE script.



(a) Training and Validation loss (Triplet Loss with margin=0.2) for the embedding module.



(b) Training and Validation BCE Loss for Siamese Network.



(c) Training and Validation BCE Loss for Relation Network.

Figure 5: Training and Validation losses for the three models.

Model	Event-based F-score (%)
Du_NERCSLIP_task5_3	76.4
Moummad_IMT_task5_4	58.3
Gelderblom_SINTEF_task5_2	36.6
SN	**4.051**
RN	**3.495**

TABLE 6: Overall validation results including SN, RN and three other models from DCASE 2023.

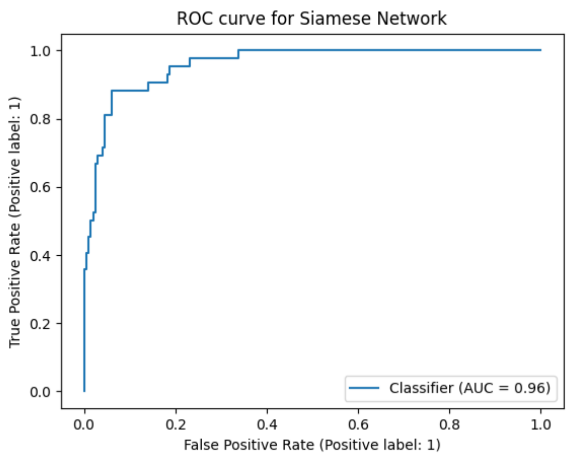


Figure 6: ROC-AUC for the Siamese Network, estimated on the validation split of the training set.

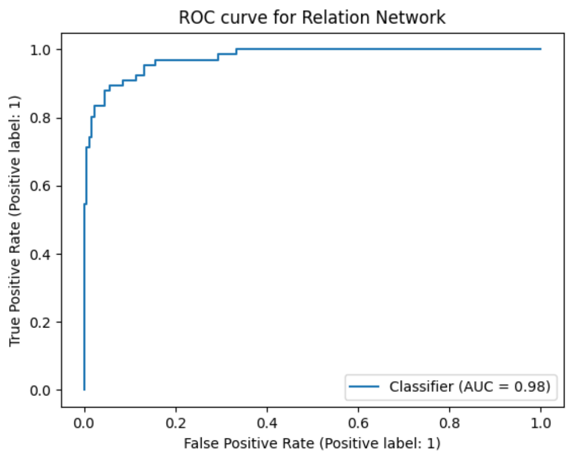


Figure 7: ROC-AUC for the Relation Network, estimated on the validation split of the training set.

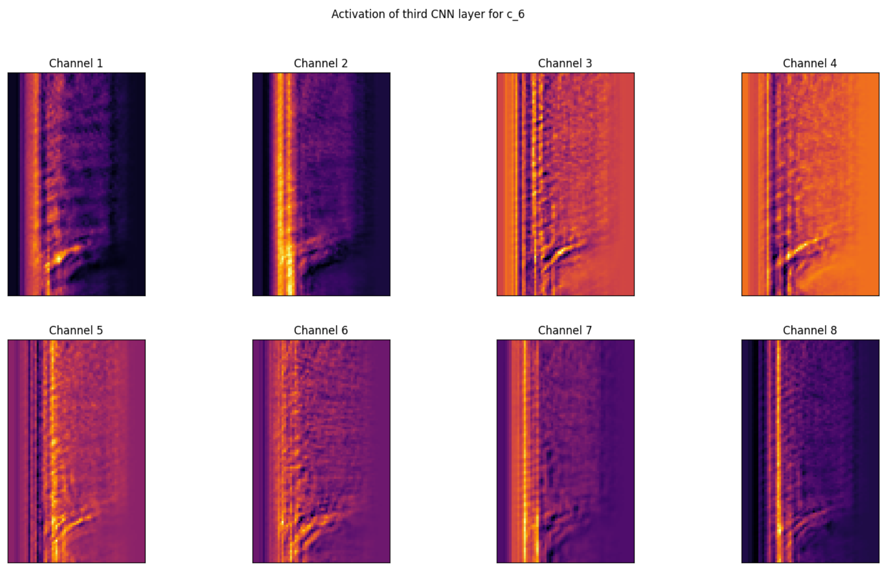


Figure 8: First eight channels of the feature map obtained from the third CNN layer of the embedding module. The considered class is the *c_6* from the training dataset.

References

[1] A. Mesaros, T. Heittola, T. Virtanen, and M. D. Plumbley, "Sound event detection: A tutorial," *IEEE Signal Processing Magazine*, vol. 38, no. 5, p. 67–83, Sep. 2021. [Online]. Available: http://dx.doi.org/10.1109/MSP.2021.3090678

[2] V. Lostanlen, J. Salamon, A. Farnsworth, S. Kelling, and J. P. Bello, "Robust sound event detection in bioacoustic sensor networks," *PLOS ONE*, vol. 14, no. 10, p. e0214168, Oct. 2019. [Online]. Available: http://dx.doi.org/10.1371/journal.pone.0214168

[3] Z. Zhao, E. A. D'Asaro, and J. A. Nystuen, "The sound of tropical cyclones," *Journal of Physical Oceanography*, vol. 44, no. 10, pp. 2763 – 2778, 2014. [Online]. Available: https://journals.ametsoc.org/view/journals/phoc/44/10/jpo-d-14-0040.1.xml

[4] Y. Wang, J. Salamon, N. J. Bryan, and J. Pablo Bello, "Few-shot sound event detection," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 81–85.

[5] I. Nolasco, S. Singh, V. Morfi, V. Lostanlen, A. Strandburg-Peshkin, E. Vidaña-Vila, L. Gill, H. Pamuła, H. Whitehead, I. Kiskin, F. H. Jensen, J. Morford, M. G. Emmerson, E. Versace, E. Grout, H. Liu, B. Ghani, and D. Stowell, "Learning to detect an animal sound from five examples," *Ecological Informatics*, vol. 77, p. 102258, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S157495412300287X

[6] G. Yan, R. Wang, L. Zou, J. Du, Q. Wang, T. Gao, and X. Fang, "Multi-task frame level system for few-shot bioacoustic event detection," DCASE2023 Challenge, Tech. Rep., June 2023.

[7] F. Gelderblom, B. Cretois, P. Johnsen, F. Remonato, and T. A. Reinen, "Few-shot bioacoustic event detection using beats," DCASE2023 Challenge, Tech. Rep., June 2023.

[8] S. Chen, Y. Wu, C. Wang, S. Liu, D. Tompkins, Z. Chen, and F. Wei, "Beats: Audio pre-training with acoustic tokenizers," 2022.

[9] I. Moummad, R. Serizel, and N. Farrugia, "Supervised contrastive learning for pre-training bioacoustic few shot systems," DCASE2023 Challenge, Tech. Rep., June 2023.

[10] L. You, E. P. Coyotl, S. Gunturu, and M. V. Segbroeck, "Transformer-based bioacoustic sound event detection on few-shot learning tasks," in *ICASSP 2023*, 2023. [Online]. Available: https://www.amazon.science/publications/transformer-based-bioacoustic-sound-event-detection-on-few-shot-learning-tasks

[11] S. Ntalampiras, "Explainable siamese neural network for classifying pediatric respiratory sounds," *IEEE Journal of Biomedical and Health Informatics*, vol. 27, no. 10, pp. 4728–4735, 2023.

[12] X. Li, X. Yang, Z. Ma, and J.-H. Xue, "Deep metric learning for few-shot image classification: A review of recent developments," *Pattern Recognition*, vol. 138, p. 109381, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320323000821

[13] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," 2018.

[14] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2015. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2015.7298682

[15] K. Musgrave, S. J. Belongie, and S.-N. Lim, "Pytorch metric learning," *ArXiv*, vol. abs/2008.09164, 2020.

[16] Y. Wang, P. Getreuer, T. Hughes, R. F. Lyon, and R. A. Saurous, "Trainable frontend for robust and far-field keyword spotting," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5670–5674, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:3100199

[17] V. Lostanlen, K. Palmer, E. Knight, C. Clark, H. Klinck, A. Farnsworth, T. Wong, J. Cramer, and J. Bello, "Long-distance detection of bioacoustic events with per-channel energy normalization," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, ser. DCASE 2019. New York University, 2019. [Online]. Available: http://dx.doi.org/10.33682/ts6e-sn53

[18] S. Falkner, A. Klein, and F. Hutter, "Bohb: Robust and efficient hyperparameter optimization at scale," *ArXiv*, vol. abs/1807.01774, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49571505

[19] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," 2018.

[20] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 1. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 115–123. [Online]. Available: https://proceedings.mlr.press/v28/bergstra13.html

[21] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," 2012.

[22] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, and F. Hutter, "Towards automatically-tuned neural networks," in *Proceedings of the Workshop on Automatic Machine Learning*, ser. Proceedings of Machine Learning Research, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., vol. 64. New York, New York, USA: PMLR, 24 Jun 2016, pp. 58–65. [Online]. Available: https://proceedings.mlr.press/v64/mendoza_towards_2016.html

[23] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," 2010.

[24] Z. Karnin, T. Koren, and O. Somekh, "Almost optimal exploration in multi-armed bandits," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, p. III–1238–III–1246.

[25] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," 2015.

[26] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf

Appendix A. PCEN

PCEN was first proposed in [16] and provides an alternative way to process spectrograms with the goal to enhance robustness to channel distortion. PCEN is constituted of an Adaptive Gain Control (AGC) stage followed by a Dynamic Range Compression (DRC) stage. In the AGC stage, a first order infinite-impulse-response (IIR) filter is applied on the filterbank energy, as shown in Equation 2:

$$M(t, f) = (1 - s)M(t - 1, f) + sE(t, f) \tag{2}$$

where $E(t, f)$ is the filterbank energy and $M(t, f)$ is a smoothed version of it. The AGC phase can be represented as follows:

$$\frac{E(t,f)}{(\epsilon+M(t,f))^{\alpha}}$$

where $\alpha \in [0,1]$ controls the gain normalisation strength in the AGC stage (the larger α, the stronger the gain normalisation), while ϵ is a small constant. AGC is performed in a channel-independent way and emphasises changes in the spectral history, tending to enhance speech onsets [16]. Following, the DRC stage aims at reducing the dynamic range using an offset δ and an exponent r. Overall, PCEN can be represented using Equation 3.

$$PCEN(t,f) = \left(\frac{E(t,f)}{(\epsilon + M(t,f))^{\alpha}} + \delta\right)^{r} - \delta^{r} \tag{3}$$

Appendix B. Model Selection through BOHB

Bayesian Optimisation and HyperBand (BOHB) was first proposed in [18] and, as the name suggests, it is based on Bayesian Optimisation and the Hyperband algorithm proposed in [19], leveraging the points of strength of both approaches.

B.1. Bayesian Optimisation

Bayesian Optimisation (BO) has been successfully applied in past years to optimise hyperparameters of deep models. For example, in [20] an highly parameterised CNN is optimised, also in [21] state-of-the-art performance was achieved on the CIFAR-10 dataset, analysing different BO approaches, while in [22] BO was applied during the 2016 autoML challenge to find a promising architecture and to optimise fully connected layers' parameters. The idea underlying BO is that the objective function to optimise is unknown and it is thus seen as a black-box function, however we can sample it using a probabilistic model based on the Bayes' theorem. More precisely, given an hypothesis M, it can be updated accordingly to new evidence E:

$$P(M|E) = P(E|M)P(M) \tag{4}$$

Where $P(M)$ is the prior knowledge we have on the problem. Essentially, since the objective function is unknown, a *surrogate model* is used in place of the objective black-box function and different strategies have been designed to guide its sampling. Specifically, an *acquisition function* is adopted to balance between exploration and exploitation and to sample accordingly the objective function. Acquisition functions such as Upper Confidence Bound (UCB), Expected Improvement (EI) or Probability of Improvement (PI) [23] provide different means to optimise the objective function, as represented in Figure 9. A generalised form of EI is provided in [23] and shown in Equation 5, where ξ is a small positive parameter, $\phi(\cdot)$ and $\Phi(\cdot)$ represent respectively the PDF and the CDF of the standard normal distribution, x^{+} is the best observation among the collected ones, while $Z = \frac{\mu(x)-f(x^{+})-\xi}{\sigma(x)}$ if $\sigma(x) > 0$, otherwise ($\sigma(x) = 0$) $Z = 0$.

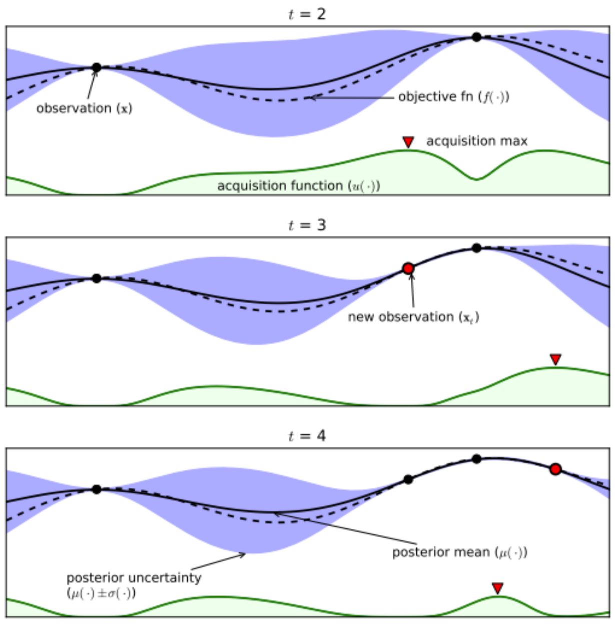


Figure 9: Representation of BO on a toy problem, from [23]. The dashed line is the surrogate model of the objective function (in bold), while the blue area represents uncertainty. The green shaded plot is the acquisition function which guides the sampling of the objective function towards promising points (in black and red dots) using a chosen policy to balance exploitation-exploration.

$$EI(x) = \begin{cases} (\mu(x) - f(x^{+}) - \xi)\Phi(Z) + \sigma(x)\phi(Z) \; if \; \sigma(x) > 0 \\ 0 \;\; if \; \sigma(x) = 0 \end{cases} \tag{5}$$

Alongside the acquisition function, a prior distribution must be specified in BO. Typically, Gaussian Processes (GPs) are chosen for their properties. A GP is a distribution over a random variable and can be represented as follows:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{6}$$

Where m is the mean function and k is the covariance function, typically the squared exponential function and the returned value by the GP is the mean and variance of a normal distribution.
Despite its success, nowadays BO may become computationally infeasible in high-dimensional problems [18] due to the increasing models' complexity.

B.2. Hyperband

The Hyperband algorithm is used as a sub-routine in BOHB and is in turn based on the Successive Halving algorithm, originally formulated for the multi-armed bandit problem in the stochastic setting [24], then adapted to the non-stochastic setting [25]. For the purpose of this work, the idea is that given a set of hyperparameter configurations, Successive Halving allocates resources in an exponentially growing way to the most promising ones. Once the performance of the configurations are evaluated, the algorithm discards the worst half and repeats the steps until a single configuration remains. The dynamic allocation of resources is the major advantage over the classical Bayesian Optimisation algorithms, that can lead to the exploration of orders-of-magnitude more configurations [19]. The main disadvantage of the Successive Halving algorithm is that it requires the number of configurations at the beginning in order to uniformly allocate resources. Indeed, given a finite budget B (e.g. training time) and n initial configurations, $\frac{B}{n}$ resources are allocated to each of them. However, there is the problem of the trade-off between running many configurations with a small average budget (large n) and running fewer configurations with larger allocated budget (small n). Intuitively, if a training procedure converges fast, then we should be able to promptly discriminate between good and bad configurations, thus using large n is reasonable, while in presence of slow convergence we may need more resources for each configuration to test. However, typically we do not know a priori which strategy is better. Hyperband aims to solve this problem by trying different values of n following a grid search style, given a value for B. The pseudocode of Hyperband is presented in Algorithm 1, where R is the maximum amount of resource allocated to each configuration, η controls the discarding of configurations in each iteration, while the inner loop is the Successive Halving sub-routine, where r is the minimum amount of resource allocated to each configuration. Each run of the sub-routine uses approximately B resources and, since $(s_{max}+1)$ values for n are tested, a single execution of Hyperband takes a budget equal to $(s_{max}+1)B$. The execution of Hyperband starts in the most explorative way, setting n to a suitable value using s_{max}, then n is reduced by a factor of η until $s = 0$ (outer loop) imposes a random search allocating equal resources to all configurations.

Algorithm 1 Hyperband algorithm, from [25].

```
Input: $R$, $\eta$ (default: $\eta$ = 3)
Initialisation: $s_{max} = \lfloor log_{\eta}(R) \rfloor$, $B = (s_{max}+1)R$
1: for $s \in \{s_{max}, s_{max}-1, ..., 0\}$ do
2:     $n = \lceil \frac{B}{R} \frac{\eta^s}{s+1} \rceil$
3:     $r = R\eta^{-s}$
4:     T = get_hyperparameter_configuration(n)
5:     for $i \in \{0, ..., s\}$ do
6:         $n_i = \lfloor n\eta^{-i} \rfloor$
7:         $r_i = r\eta^{i}$
8:         L = {run_then_return_val_loss($t, r_i$) : $t \in T$}
9:         T = top_k($T, L, \lfloor \frac{n_i}{\eta} \rfloor$)
10:    end for
11: end for
12: return configuration with the best intermediate loss.
```

The major limitation of Hyperband is that it does not perform a targeted search on the objective function, contrarily to the BO approaches, thus leaving room for further resource optimisations.

B.3. BOHB

The resource optimisation provided by the Hyperband algorithm is leveraged together with the targeted search of BO methods. BOHB performs early stopping for unpromising configurations, thus reallocating resources in an optimised way and using larger budgets on fewer configurations as the search progresses. An important aspect of BOHB is that it uses the Tree Parzen Estimator (TPE) approach [26], a BO technique that models the probability densities over the configuration space as shown in Equation 7

$$p(x|y) = \begin{cases} l(x) & if\ y < y* \\ g(x) & otherwise \end{cases} \tag{7}$$

Where $l(x)$ is a density estimated using the observations that lead to a loss less than y^*, while $g(x)$ derives from the complementary case. TPE selects y^* such that $p(y < y^*) = \gamma$, i.e. y^* is a quantile γ of the observed values. This method is used in place of the EI acquisition function discussed in Appendix B.1 and it is equivalent to it, sampling the next candidate point by maximising $\frac{l(x)}{g(x)}$. Since TPE does not require a model for $p(y)$, it provides important computational advantages with respect to GP-based approaches [18], thus enabling BOHB to be more efficient than other BO techniques.