TUC Media Computing at BirdCLEF 2024: Improving Birdsong Classification Through Single Learning Models

Notebook for the Media Computing Lab at CLEF 2024

Arunodhayan Sampath Kumar*, Tobias Schlosser and Danny Kowerko

Junior Professorship of Media Computing, Chemnitz University of Technology, 09107 Chemnitz, Germany

Abstract

This work presents our contribution to the BirdCLEF 2024 competition, aimed at enhancing birdsong classification through the implementation of single learning models. Our primary objective was to develop a robust model capable of processing continuous audio data to accurately identify bird species from their calls. Key challenges included addressing imbalanced training data, managing domain shifts between training and test samples, and processing extensive soundscape recordings within a limited time frame. Our proposed approach leverages transformer-based architectures, incorporating positional encodings to enhance the spatial context of the input spectrograms. Data augmentation techniques were employed to mitigate the effects of noisy labels and domain shifts. The model training process involved the use of various libraries and frameworks, with a focus on optimizing performance through strategies such as cosine annealing and weighted sampling. Our results indicate the potential in improving the accuracy and efficiency of birdsong classification to support avian population monitoring and conservation efforts. From a total of 974 teams that ranked between 45.8 % and 69.0 % in terms of ROC-AUC on the private leaderboard, our best model achieved a ROC-AUC score of 62.9 %, ranking us at 300th place among all submissions. On the public leaderboard, our model achieved a score of 63.9 %.

Keywords

Audio Classification, Birdsong Soundscapes, Computer Vision and Pattern Recognition, Convolutional Neural Networks (CNN), Data-Efficient Image Transformers (DeiT), Vision Transformers (ViT)

1. Introduction and motivation

The BirdCLEF 2024 [1, 2] competition aimed to identify bird species in a global biodiversity hotspot in the Western Ghats, also known as the Sahyadri. The broader goals of BirdCLEF 2024 included developing a deep learning model capable of processing continuous data to recognize bird species by their calls. Specifically, the objectives were to identify endemic bird species in the soundscape data of the sky-islands, detect and classify endangered bird species (species of conservation concern) despite having limited training data, and detect and classify nocturnal bird species, which are currently poorly understood.

This year's primary challenges involved a significant imbalance in the training set, a shift in the domain between the clean training samples and the soundscape test samples, and the time constraint of testing 73.3 hours of diverse soundscape recordings in only 2 hours of time. Nevertheless, advancements in deep learning models are expected to aid in monitoring avian populations, facilitate more effective threat evaluation, and allow for timely adjustments to conservation actions. Ultimately, this will benefit avian populations and support long-term sustainability efforts.

CLEF 2024: Conference and Labs of the Evaluation Forum, September 09–12, 2024, Grenoble, France

*Corresponding author.

arunodhayan.sampath-kumar@cs.tu-chemnitz.de (A. S. Kumar); tobias.schlosser@cs.tu-chemnitz.de (T. Schlosser); danny.kowerko@cs.tu-chemnitz.de (D. Kowerko)
https://arunodhayan.github.io (A. S. Kumar); https://www.tobias-schlosser.net (T. Schlosser); https://tu-chemnitz.de/cs/mc (D. Kowerko)
0000-0002-2654-7050 (A. S. Kumar); 0000-0002-0682-4284 (T. Schlosser); 0000-0002-2411-8877 (D. Kowerko)

Table 1
Dataset used for model pre-training (IDs 2, 3, and 4) and data augmentation (IDs 5 and 6).

ID	Name	Classes	Files
1	BirdCLEF 2024 [5]	182	24 459
2	BirdCLEF 2021 [6]	397	62 874
3	BirdCLEF 2022 [7]	152	14 852
4	BirdCLEF 2023 [8]	264	16 940
5	DCASE [9]	1	22 012
6	ESC-50 [10]	4	500
Total		**1000**	**141637**

2. Fundamentals and implementation

The implementation of transformer-based bird species recognition presented in this work comprises data preparation, feature extraction, model architecture, data augmentation, and training methods. This work builds on our previous implementations for BirdCLEF 2021 and 2022, which utilized convolutional neural networks (CNN) [3, 4].

2.1. Data Preparation

The BirdCLEF 2024 training set consisted of 24 459 audio recordings provided by xeno-canto [5], covering 182 bird species. The test dataset consisted of 1 100 recordings, each of 4 minutes in length. Table 1 provides an overview of the individual datasets utilized. Datasets with IDs 2, 3, and 4 were used for model pre-training. Datasets with IDs 5 and 6 were employed as a data augmentation technique for background noise. In total, 141 637 recordings across 1 000 classes were collected for training and data augmentation. The BirdCLEF 2024 train set consisted of audio files sampled at 32 kHz using a single channel (mono) and compressed using lossy Ogg compression. Similarly, the datasets used for pre-training and augmentation were converted to 32 kHz and a single channel. The training dataset was weakly labeled, meaning there was no precise information on the presence or absence of the labeled birdsong within the recordings. However, hearing the labeled birdsong at the beginning or the end of each audio file is typically highly probable. These audio files were trimmed for the first 5 and last 5 seconds and saved as NumPy arrays to speed up the data loading procedure since we do not need to load the complete audio file. For training and cross-validation, the dataset is split into 5 stratified folds. For pre-training, we made sure that the species present in BirdCLEF 2024 are not present in our pre-training dataset in order to avoid leakage.

2.2. Feature extraction

We used the librosa [11] Python library to convert a 1D audio signal into 2D log-mel spectrograms. The spectrogram representation used the following parameters:

- Width (W) = 576 or 256
- Height (H) = 256 or 196
- Sampling rate (SR) = 32 000 Hz
- hop_length = 284
- mel_bins = W // 2
- frequency_min = 50 Hz
- frequency_max = 16 000 Hz
- power = 2.0
- top_db = 100

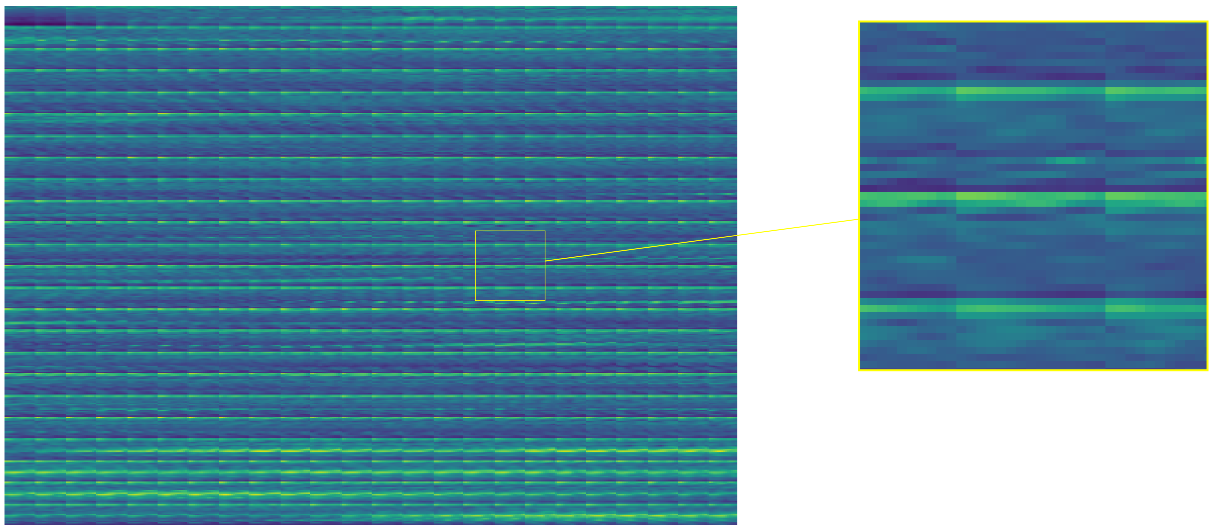

Figure 1: Input Spectrogram: This comprises a 16×16 patch spectrogram arranged in a 24×24 grid. The 16×16 patch spectrogram has a time axis (x-axis) ranging from 0 to 5 seconds, and a frequency axis (y-axis) ranging from 50 to 16 000 Hz.

Algorithm 1 Resizing and rearranging spectrograms method for feature extraction.

```
Require: Input spectrogram $x$ of shape either $(576, 256)$ or $(256, 196)$
Ensure: Output tensor $y$ of shape either $(384, 384)$ or $(224, 224)$
if $x$.shape $= (576, 256)$ then
    Initialize $y \leftarrow$ zeros$(384, 384, \text{dtype} = x.\text{dtype})$
    $x1 \leftarrow x^T.\text{view}(24, 24, 16, 16)$
    for $i = 0$ to $23$ do
        for $j = 0$ to $23$ do
            $y[i \times 16 : (i+1) \times 16, j \times 16 : (j+1) \times 16] \leftarrow x1[i, j]$
        end for
    end for
else if $x$.shape $= (256, 196)$ then
    Initialize $y \leftarrow$ zeros$(224, 224, \text{dtype} = x.\text{dtype})$
    $x1 \leftarrow x^T.\text{view}(14, 16, 16, 16)$
    for $i = 0$ to $13$ do
        for $j = 0$ to $15$ do
            $y[i \times 16 : (i+1) \times 16, j \times 16 : (j+1) \times 16] \leftarrow x1[i, j]$
        end for
    end for
else
    Raise error: Unsupported input shape
end if
    return $y$
```

Vision transformers (ViT) oftentimes require spectrograms that are either 384×384 or 224×224 in size. Resizing spectrograms from 576×256 to 384×384 or from 256×196 to 224×224 did not yield any positive results. Therefore, we utilized a resizing and rearranging spectrograms method to ensure that spectrograms are properly provided to the transformer model. The detailed algorithm for this approach is shown in Algorithm 1. Its spectrogram realization is illustrated in the accompanying Fig. 1.

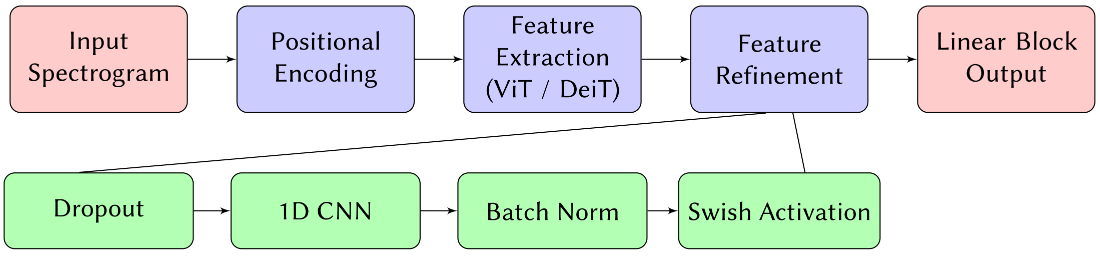

Figure 2: Block diagram of the BirdCLEF transformer model architecture.

2.3. Model Architecture

Figure 2 depicts our model architecture used. It incorporates positional encodings into the input spectrograms to enhance the spatial context of the data. Upon receiving the input spectrogram, an additional singleton dimension is added. A positional encoding is then generated by linearly interpolating values from 0 to 1 along the input height. This encoding is converted to half-precision and reshaped to match the spectrogram dimensions. The positional encoding is concatenated with the input spectrogram along the channel dimension, resulting in a combined input that includes both spectral and positional information. This enhanced input is then passed through the backbone network for feature extraction. The feature extractor used here is ViT / data-efficient image transformers (DeiT) [12, 13]. Post-feature extraction, the first element along the middle dimension is selected to reshape the feature map accordingly. The feature refinement stage processes these reshaped features through a series of operations, including a dropout layer, a 1D convolutional layer, batch normalization, and a swish activation function, to further refine the feature representation. The final classification logits are obtained by passing these refined features through the linear block, a fully connected layer. The output contains the logits, enabling the model to make improved predictions by leveraging spectral and spatial information.

2.4. Data Augmentation

Data augmentation techniques were utilized to address the domain shift between the training and test sets, as well as to handle weak and noisy labels. A concise overview of our used augmentation strategies is provided in [14]. These include:

- Noise augmentations
- Short-noise bursts
- General mix up
- tanh-based distortion
- Gaussian noise
- Loudness normalization

2.5. Training Methods

In the training process, we used various tools and libraries. The main framework we utilized was the machine learning library PyTorch, along with additional libraries such as the Pytorch Image Models timm for transformer backbone models [15], soundfile [16] and librosa [11] for audio and signal processing, audiomentations for data augmentation, and scikit-learn for calculating metrics and creating stratified folds for training / validation data splits.

The depth and number of heads of ViT and DeiT were reduced to decrease the required computation times. This reduction enabled us to omit the usage of pre-trained weights, such as from the ImageNet dataset. As a result, we opted to train the model from scratch using the previous year's BirdCLEF competition dataset for 70 training epochs. These weights were in turn used as our pre-trained weights

Table 2
BirdCLEF 2024 competition results. Note that rank 300 was not the best model submission and is thus not seen in the official ranking. Rank 300 would be the position corresponding to the ROC-AUC of our best transformer-based approach. Submission of TUC was done under the user name Arunodhayan.

Rank (private LB)	Team name	ROC-AUC (private LB)	ROC-AUC (public LB)
1	Team Kefir	0.690391	0.738566
2	adsr	0.690354	0.727939
3	NVBird	0.689970	0.742124
4	Team Cerberus	0.687770	0.746911
5	coolz	0.687173	0.743960
300	**TUC (Transformer)**	**0.629135**	**0.638547**

for BirdCLEF 2024. ViT / DeiT require the image to be either 384×384 or 224×224 pixels in size, for which we extracted non-overlapping 16×16 patches, arraigned in a 24×24 grid. Initially, we resized our spectrogram to meet these dimensions. However, we encountered disappointing results due to diluted essential features caused by noise domination and misallocation of attention. Therefore, we resized and rearranged our spectrograms as described in Algorithm 1. After performing the spectrogram extraction and model training, the obtained results could be further improved since transformers lack an inherent notion of the order of the input spectrograms. Hence, we introduced a positional encoding block that addresses this by injecting information about the positions of elements in the sequence directly into the model. The model was trained with a learning rate of $5 \cdot 10^{-4}$ with a cosine annealing learning rate scheduler to optimize performance over epochs. To prevent overfitting, a weight decay of $1 \cdot 10^{-6}$ was applied. The training process utilized a weighted sampler based on the number of samples per class to ensure balanced representation during training. For loss computation, binary cross-entropy with logits loss function was employed, incorporating secondary labels for a more nuanced learning. The model utilized the AdamW optimizer. The number of epochs for training was set to 70, with a batch size of 32.

3. Test results, evaluation, and discussion

While our submissions with the name Arunodhayan using EfficientNetB0 ranked 76 and 440 in the public and private leaderboard, respectively, in our report documentation, we will focus on systematic studies obtained with vision transformers. We achieved a macro average ROC score of 63 % on the public test set (public leaderboard) and 62 % on the private test set (private leaderboard) by utilizing vision transformers, excluding classes with no true positive labels. The top-ranked team achieved a score of 69 % on the private test set. Our best transformer model would have achieved a rank of 300 among 972 participants on the private leaderboard (LB) (team TUC in Table 2).

The main goal of this competition was to evaluate the test set in under 2 hours of runtime without using a graphics processing units. The primary challenge we faced was the inconsistency in Kaggle's hardware, which made it difficult to predict the runtime of our models. Therefore, we could not complete the test runs without altering our transformer models when the execution time exceeded 2 hours. To address this issue, we converted our model via OpenVINO [17]. After making the submission, the execution time was reduced to 117 minutes, and we achieved a score of 61 %. Subsequently, we experimented with reducing the number of depth and heads of the transformer models, which further reduced the inference time. The respectively obtained results are presented in Table 2.

Table 3
Experiment overview. Entries with "-" could not be completed due to runtime requirements.

ID	Description	Depth	Head	Testing time (minutes)	CV	Positional encoding	Private LB	Public LB
M1	ViT_tiny_384	12	3	314	0.68	False	-	-
M1.1	ViT_tiny_384	12	3	314	0.74	True	-	-
M2	ViT_tiny_224	12	3	263	0.69	True	-	-
M3	ViT_tiny_384	10	3	219	0.71	True	-	-
M4	ViT_tiny_224	10	3	130	0.66	True	-	-
M5	ViT_tiny_384	8	3	119	0.70	True	0.595397	0.582498
M6	ViT_tiny_224	8	3	94	0.68	True	0.596327	0.581465
M7	ViT_tiny_384	8	2	72	0.72	True	**0.596547**	**0.586987**
M7.1	ViT_tiny_384	8	2	72	0.72	True	**0.629135**	**0.638547**
M8	ViT_tiny_224	8	2	55	0.58	True	0.592785	0.571654
M9	ViT_tiny_384	6	2	44	0.63	True	0.602487	0.598741
M10	DeiT_tiny_224	8	2	65	0.69	True	0.609154	0.597128
M11	DeiT_tiny_384	6	2	62	0.72	True	0.618745	0.602874

Ablation study

Table 3 outlines the various aspects and approaches related to model performance we investigated. M1 and M1.1 are the baseline models, both supporting a resolution of 384×384. Unfortunately, we encountered issues in submitting these models, as the inference time exceeded the allotted time frame. Our cross-validation (CV) results improved macro-averaged mean average precision (cmAP) when positional encoding was introduced. We then attempted to utilize ViT, which operates at a resolution of 224×224. With the inclusion of positional encoding, we achieved a CV score of 69 %. However, our submission failed due to the inability to complete the execution within the allocated time frame. To address this, we experimented by reducing the complexity of the ViT transformer model, resulting in models M3 to M11. Model M5 was successfully submitted, although we experienced inconsistent Kaggle hardware, leading to unsuccessful runs with the same model. Subsequently, we opted to reduce the head of the transformer to 2, which proved to be successful. Model M5 served as our baseline, and we fine-tuned the model through data augmentation. Traditionally, adding soundscape noise as data augmentation has improved performance. However, this year's soundscape noise augmentation led to a drop in model performance. Consequently, we decided to incorporate noise augmentations from the ESC-50 dataset, focusing on insects, cars, human noises, and non-bird events from the DCASE 2018 dataset. Following this principle, we utilized an unlabeled soundscape dataset as pseudo labels using the M7 transformer model, leading to an overall performance increase by 4 % (model M7.1).

4. Conclusion and outlook

The BirdCLEF 2024 competition presented a unique challenge compared to its previous years. The use of the metric "a macro-averaged ROC-AUC that excludes classes with no true positive labels" made it difficult to compare cross-validation results and leaderboard scores, Furthermore, the importance of developing efficient models that achieve a good balance between accuracy and speed was highlighted. Our work explored the use of vision transformers and data-efficient image transformers as well as the optimization of these models to run on CPU hardware by converting them via OpenVINO. Since transformers require standard shapes such as 384×384 or 224×224, directly resizing the spectrograms to the desired shape did not show the expected impact. As an alternative, we attempted to resize and reshape the spectrograms by organizing 16×16 patch spectrograms into a 24×24 grid, which yielded promising results but did not outperform the scores of competing submissions that used ensemble models that included EfficientNets. Due to the reduction in the complexity of the transformer, we could not use pre-trained weights. Therefore, we trained a model using data from a previous competition, facilitating faster model convergence. The model's efficiency depends on the data's quantity and quality. In BirdCLEF 2024, we trained a model using a dataset recorded with high-quality microphones. However, when we tested it using a soundscape dataset recorded with long-distance microphones, there was a domain shift, meaning the test set was not representative of the training dataset. To address this, we added non-bird event sounds to the training set as an augmentation, which made it more similar to the test set and improved model performance.

References

[1] S. Kahl, T. Denton, H. Klinck, V. Ramesh, V. Joshi, M. Srivathsa, A. Anand, C. Arvind, H. CP, S. Sawant, V. V. Robin, H. Glotin, H. Goëau, W.-P. Vellinga, R. Planqué, A. Joly, Overview of BirdCLEF 2024: Acoustic identification of under-studied bird species in the western ghats, Working Notes of CLEF 2024 - Conference and Labs of the Evaluation Forum (2024).

[2] A. Joly, L. Picek, S. Kahl, H. Goëau, V. Espitalier, C. Botella, B. Deneu, D. Marcos, J. Estopinan, C. Leblanc, T. Larcher, M. Šulc, M. Hrúz, M. Servajean, et al., Overview of lifeclef 2024: Challenges on species distribution prediction and identification, in: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer, 2024.

[3] A. Sampathkumar, D. Kowerko, Tuc media computing at birdclef 2021: Noise augmentation strategies in bird sound classification in combination with densenets and resnets, 2021. URL: https://arxiv.org/abs/2106.10856, arXiv preprint arXiv:2106.10856.

[4] A. Sampathkumar, D. Kowerko, Tuc media computing at birdclef 2022: Strategies in identifying bird sounds in complex acoustic environments, in: Proceedings of the Working Notes of CLEF, 2022, pp. 2189–2198.

[5] H. Klinck, M. Sohier, D. S. Kahl, T. Denton, V. Ramesh, Birdclef 2024, https://kaggle.com/competitions/birdclef-2024, 2024.

[6] A. Howard, A. Joly, H. Klinck, S. Dane, S. Kahl, T. Denton, Birdclef 2021 - birdcall identification, 2021. URL: https://kaggle.com/competitions/birdclef-2021, kaggle.

[7] A. Howard, A. Navine, H. Klinck, S. Dane, S. Kahl, T. Denton, Birdclef 2022, 2022. URL: https://kaggle.com/competitions/birdclef-2022, kaggle.

[8] H. Klinck, S. Dane, S. Kahl, T. Denton, Birdclef 2023, 2023. URL: https://kaggle.com/competitions/birdclef-2023, kaggle.

[9] D. Stowell, Y. Stylianou, M. Wood, H. Pamuła, H. Glotin, Automatic acoustic detection of birds through deep learning: the first bird audio detection challenge, Methods in Ecology and Evolution (2018). URL: https://arxiv.org/abs/1807.05812. arXiv:1807.05812.

[10] K. J. Piczak, Esc: Dataset for environmental sound classification, in: Proceedings of the 23rd ACM International Conference on Multimedia, ACM, 2015, pp. 1015–1018. doi:10.1145/2733373.2806390.

[11] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, O. Nieto, librosa: Audio and music signal analysis in python, in: Proceedings of the 14th python in science conference, volume 8, 2015.

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, CoRR abs/2010.11929 (2020). URL: https://arxiv.org/abs/2010.11929. arXiv:2010.11929.

[13] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou, Training data-efficient image transformers & distillation through attention, CoRR abs/2012.12877 (2020). URL: https://arxiv.org/abs/2012.12877. arXiv:2012.12877.

[14] A. S. Kumar, T. Schlosser, S. Kahl, D. Kowerko, Improving learning-based birdsong classification by utilizing combined audio augmentation strategies, Ecological Informatics (2024) 102699. URL: https://www.sciencedirect.com/science/article/pii/S1574954124002413. doi:https://doi.org/10.1016/j.ecoinf.2024.102699.

[15] R. Wightman, Pytorch image models, https://github.com/rwightman/pytorch-image-models, 2019. doi:10.5281/zenodo.4414861.

[16] M. R. Gogins, Soundfile, https://pypi.org/project/soundfile/, 2024. Version 0.12.1.

[17] O. T. Contributors, Openvino™ toolkit, 2024. URL: https://github.com/openvinotoolkit/openvino, accessed: 2024-06-20.