HAW HAMBURG

BACHELOR THESIS

Bat Call Classification in Audio Recordings with Drone Noise Using Deep Learning

submitted on 23 May 2024

Mira Hesse
[redacted]

First Examiner:	Prof. Dr. Tessa T. Taefi
Second Examiner:	Marc Roswag

HAMBURG UNIVERSITY OF APPLIED SCIENCES
Department of Media Technology
Finkenau 35
22081 Hamburg

Abstract

There are several approaches to bat call classification using deep learning. In this thesis, I compared existing architectures for bat call classification and for general audio classification on a dataset that contained acoustic drone noise in the recordings in addition to bat calls. Two pretrained models (without further training) showed no significant difference between data with and without drone noise. I also compared two pretrained models after transfer learning. For the training dataset I compared different levels of resampling and an augmented dataset to compensate for the high class imbalance between the three classes in the dataset. The best results were obtained using the Patchout faSt Spectrogram Transformer (PaSST) model with oversampling. The model achieved a f1-score of 94.9% on a binary classification task. On multilabel classification it achieved a micro f1-Score of 90.6% and a macro f1-score of 78.5%.

Zusammenfassung

Es gibt bereits einige Modelle für die Klassifikation von Fledermausrufen mithilfe von Deep Learning. In dieser Arbeit habe ich existierende Modelle für die Klassifizierung von Fledermausrufen und für die allgemeine Audioklassifikation für einen Datensatz verglichen, der zusätzlich akustisches Drohnenrauschen enthält. Zwei vortrainierte Modelle (ohne weiteres Training) zeigten keinen signifikanten Unterschied zwischen Daten mit und ohne Drohnenrauschen. Ich habe auch zwei Modelle nach „Transfer Learning“ verglichen. Für den Trainingsdatensatz habe ich verschiedene „Resampling“ und „Augmentation“ Kombinationen verglichen, um das hohe Ungleichgewicht zwischen den drei Klassen im Datensatz auszugleichen. Die besten Ergebnisse wurden mit dem Modell „Patchout faSt Spectrogram Transformer“ (PaSST) und „Oversampling“ erzielt. Das Modell erreichte einen F1-Score von 94, 3% für binäre Klassifizierung. Für „Multilabel“ Klassifizierung erreichte es einen Mikro F1-Score von 90, 6% und einen Makro F1-Score von 78, 5%.

Contents

Acronyms

adam	adaptive moment estimation
AST	Audio Spectrogram Transformer
BAT	BioAcoustic Transformer
CNN	Convolutional Neural Network
CPU	Central Processing Unit
ESC	Environmental Sound Classification
GPU	Graphical Processing Unit
IRLbl	Imbalance Ratio Per Label
LP-ROS	Label Powerset Random Oversampling
LP-RUS	Label Powerset Random Undersampling
MeanIR	Mean Imbalance Ratio
Myo	Myotis
Nyc	Nyctalus
PaSST	The Patchout faSt Spectrogram Transformer
Pip	Pipistrellus
RNN	Recurrent Neural Network
STFT	Short-Time Fourier Transform
UAS	Unmanned Aerial System
ViT	VisionTransformer

List of Figures

List of Tables

1. Introduction

The fight against climate change and the conservation of biodiversity can sometimes contradict each other. Wind turbines can contribute to climate protection by reducing carbon dioxide emissions, which is why the number of wind turbines is growing (Barthelmie & Pryor, 2021). At the same time operating wind turbines can cause mortality among bats, therefore they are a threat to biodiversity (Lehnert et al., 2014). Wind turbines have to be shut off during periods of high bat activity in the European Union, which protects bats, but reduces the amount of energy produced by wind farms (Smallwood & Bell, 2020; Voigt et al., 2022).

Monitoring of bats in potentially dangerous regions could improve the current state of bat protection actions. Presently used methods usually monitor bat activity with automatic ultrasonic detectors also called bat detectors (acoustic recording devices that automatically start recording when they detect a bat call). The bat detector is usually positioned at about ground level or at a fixed position at the wind turbines' nacelle resulting in zones that can not be covered by the bat detector (Voigt et al., 2021). The project Drones4Bats[1] by Hamburg University of Applied Sciences works towards a different approach, by assessing the suitability of Unmanned Aerial Systems (UAS's) as a tool to improve acoustic bat monitoring at wind farm sites. The UAS is used to carry a bat detector into the regions of interest and opens the opportunity of covering a larger area than with other monitoring methods (Roswag et al., 2024). UAS's emit acoustic noise that the bat detector may mistake for bat calls (Taefi et al., 2024). So using UAS's could be a chance to improve bat monitoring on wind farm sites, while at the same time introducing new challenges.

Classifying bat calls in audio recordings is a known challenge. Researchers have developed different deep learning models to approach the task (e.g. Aodha et al., 2022; Fundel et al., 2023 and Zualkernan et al., 2021, c.f. section 2.2.3). There are also commercial tools for the task. However, these models and tools are not optimized to classify bat calls in audio recordings that contain drone noise. The commercial software bcadmin[2] misclassifies a significant amount of recordings with drone noise according to Taefi et al., 2024. VisionTransformers (ViTs), a deep learning architecture, show promising results on a binary classification task (bat vs. noise) on a small dataset that was collected as part of the Drones4Bats project (Mauson, 2022;

[1]https://www.haw-hamburg.de/forschung/forschungsprojekte-detail/project/project/show/drones4bats/
[2]https://ecoobs.de/produkte/software/bcadmin

Taefi et al., 2024). Methods to classify bat calls exist, but a large dataset of bat call recordings that also contain drone noise, could not yet be classified with any of those successfully.

This thesis works towards multilabel classifying a larger bat echolocation dataset that contains drone noise. The research question I am working on is, which of the compared pretrained deep learning architectures achieves the highest f1-scores (c.f. section 3.3) on a dataset that contains bat call recordings contaminated with drone noise. I compare (1) different deep learning architectures pretrained for either bat call or general audio classification, (2) pretrained models as they are provided and pretrained models that I trained on the given dataset, (3) how interpreting the results as those of a binary classifier (bat vs. noise) changes the results, (4) how methods for handeling imbalanced datasets influence the results. (4) was chosen after analysing the dataset distribution (c.f. section 3.1).

In the first part of my thesis I describe the state of the art in bat call classification with deep learning, including an overview of deep learning architectures. Then I describe the deep learning models and methods I applied before, during and after training to answer my research question. At last I present and discuss the results of my experiments and draw a conclusion.

2. State of the Art

The topic of this thesis - bat call classification using deep learning - is an audio classification task. Section 2.1 focuses on the general concepts of deep learning and neural networks for classification tasks. Section 2.2 then describes existing approaches in audio classification and in bat call classification that use deep learning methods.

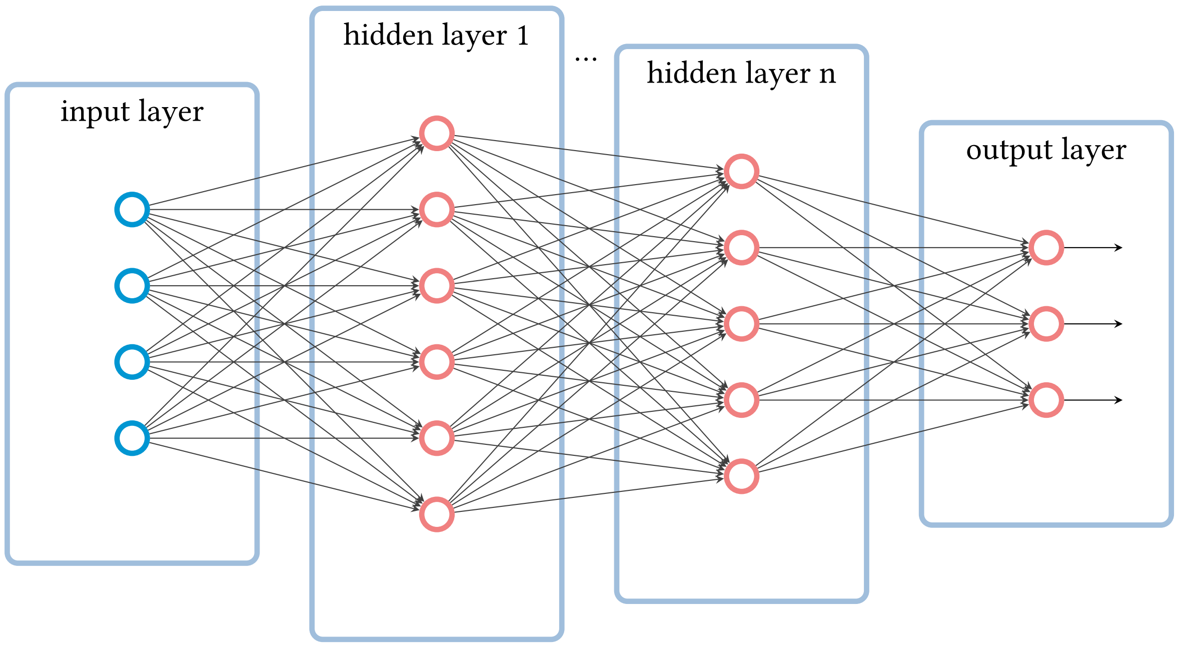
2.1. Neural Networks and Deep Learning

Deep learning is a type of machine learning that is part of the broader field of artificial intelligence. Deep learning - and other machine learning techniques - work with features that are generated from the input data. Whereas in traditional machine learning, the features are constructed manually, in deep learning the features are learned directly from the data. Therefore, deep learning does not require as much expert knowledge about the input data as traditional machine learning (LeCun et al., 2015). According to Zaman et al., 2023 deep learning mostly performs better than traditional machine learning, but requires large training datasets and more computational resources.

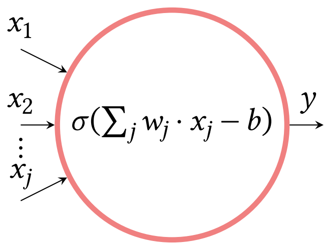
Deep learning is based on neural networks. In addition to the basic fully connected neural networks, there are several more specific neural network architectures with different strengths and weaknesses (Tufféry, 2023). After some general information about neural networks and deep learning, this section describes the specific model architectures relevant to this thesis - Convolutional Neural Networks (CNNs), transformer encoders and hybrid models.

2.1.1. Fully-Connected Neural Networks

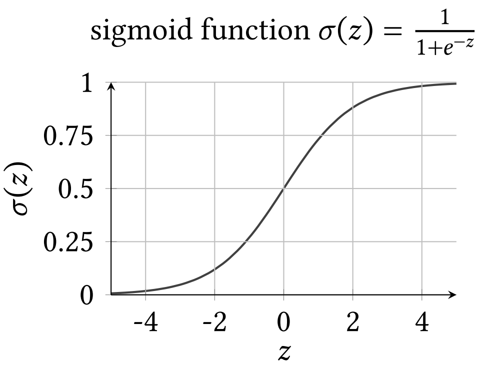
A neural network consists of layers (see figure 2.1a). There is an input layer, which contains the input data, followed by a number of hidden layers, followed by the output layer. A neural network is called a multilayer perceptron or a deep neural network (as opposed to a shallow neural network) if it has at least two hidden layers. Each hidden layer is made up of neurons that each take all outputs from the previous layer as input. A neuron outputs one value that is fed into each neuron of the following layer. This builds a (fully-connected) neural network



(a) Fully-connected neural network.



(b) The sigmoid neuron.



(c) The sigmoid activation function.

Figure 2.1.: Fully-connected neural network. Adapted from Nielsen, 2015; Tufféry, 2023.

of decision making units that rely on decisions made by previous units. The output layer is built the same way as the hidden layers, except that the output of those neurons is the result of the neural network. The output of a neuron is calculated using the inputs, a weight for each input and a bias. A basic form of neuron is the perceptron. Nielsen, 2015 explains how perceptrons are universal for computation, because they can be used to compute any logical function. The difference between traditional logic and neural networks is the introduction of learning algorithms that can find weights and biases, where it would be too complex a task for humans. For perceptrons a small value change can cause a large output change. To avoid this behaviour perceptrons can be replaced by sigmoid neurons. Figure 2.1b shows how the output of a sigmoid neuron is calculated and figure 2.1c shows the sigmoid function. The sigmoid function is an activation function and is used to map the output of each neuron to a scale from 0 to 1. There are other activation functions that could replace the sigmoid function in the formula, but the sigmoid function is the most common activation function, because it simplifies the computation of the derivatives needed for the backpropagation algorithm. So a key feature of deep learning is the use of hidden layers. They are called hidden, because the values of their parameters (weights and biases) are not directly influenced by a human, but instead determined by the training data during the process of training. The network described in this section is also an example for a feed forward neural network in which information is never moved to previous layers but only from one layer forward to the next layer (Nielsen, 2015). A neural network is the basic structure all models used in this thesis are based on.

2.1.2. Training of Neural Networks

The training process described in the following refers to supervised learning, where the model has information about the true labels for each sample (LeCun et al., 2015). The purpose of training a neural network is to modify the adjustable parameters (weights and biases) of the network in a way that, after training, the output of the neural network is closer to the true labels than it was before training (Nielsen, 2015). The training process uses backpropagation and gradient descent and in a first step computes the loss (or cost) function. The loss function is a multidimensional function of all the weights and biases in the network. The smaller the loss, the more accurate the output of the neural network. The goal is to find weights and biases for the network so that the network approximates the outputs for all inputs as closely as possible. The backpropagation algorithm works by calculating all partial derivatives of the loss function with respect to each weight and bias. The next step is to move in the opposite direction of the loss functions' gradient by a defined amount (the learning rate), which is called gradient descent. The amount of computation needed can be reduced by using stochastic gradient descent, where the gradient is averaged over random subsets (mini-batches) of the training data. When all samples have been considered one epoch ends. Ideally there are as many epochs, as it needs to reach the global minimum of the loss function (Nielsen, 2015). The key elements of the described training process are backpropagation and gradient descent, but there is also a number of hyperparameters (e.g. the number of epochs, the learning rate and the loss function) that can be chosen.

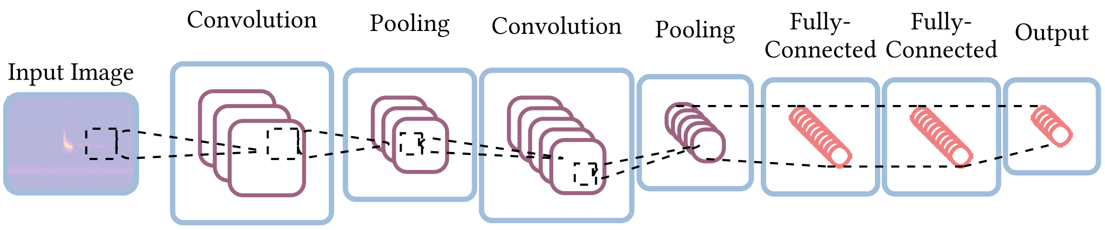
Problems that may occur when hyperparameters are chosen poorly could be overfitting, when the model does not generalize well to unseen data or underfitting, when the training was not extensive enough (Zaman et al., 2023). In the following I name hyperparameters and how they are chosen. Before training a neural network, the dataset is usually split into data used for training and data used for testing, to ensure that the network can be tested on data it has never seen before. The test set can also be split again into a validation set, for monitoring performance during training and the remaining test set, for evaluating performance after training. Typical split proportions could be around two thirds for training and one third for testing (Sechidis et al., 2011). The total number of epochs can be either set before training or the training is stopped when a certain condition is met, e.g the results on the validation set do not improve any longer. The number of epochs is optimal, when the model does not overfit the training data and reached its maximum performance. A higher number of epoch needs more training time. Changing the loss function can influence which errors the training process prioritizes. A widely used loss function is the cross-entropy loss function. The learning rate determines how fast a model learns. Generally faster learning could be considered better, but if the learning rate is to high, the loss function may skip its global minimum (Nielsen, 2015). An optimizer enables the use of an adaptive learning rate. A common optimizer (e.g. Tufféry, 2023) is adaptive moment estimation (adam) optimizer (Kingma & Ba, 2014). Since classification models output a value between 0 and 1 for each class, another parameter to choose after training is the threshold. All classes with probabilities larger than the threshold are considered true (Tufféry, 2023). Training a neural network is a complex process that needs data, computational resources and careful setting of hyperparameters.

2.1.3. Pretraining and Transfer Learning

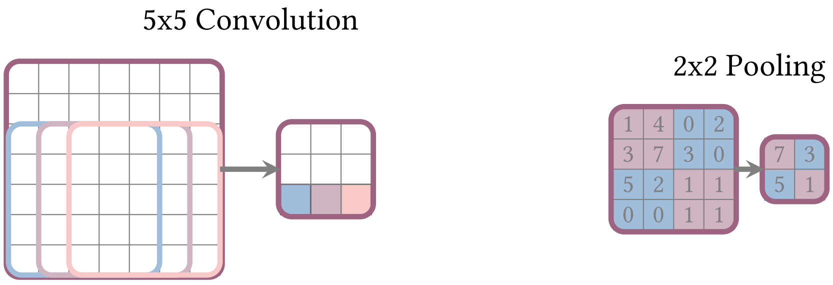
When a neural network is trained, the weights and biases are usually initialized randomly. Transfer learning takes advantage of a network that has already been trained (pretrained) by using its weights and biases for initialization. When training a pretrained model on another dataset it is possible to either train (finetune) all layers or freeze some layers to reduce training time (Iman et al., 2023). Tsalera et al., 2021 compared different CNNs for sound classification trained from scratch and from pretrained models and consistently achieved higher accuracy with the models that used transfer learning. Pretraining and transfer learning can be interpreted as a head start to reduce the resources needed.

2.1.4. Convolutional Neural Networks

A CNN is a feedforward neural network that has been adapted to work with higher dimensional data, such as two dimensional images. The neural network described in section 2.1.1 uses fully connected layers, where each neuron in a layer is connected to each neuron in the previous and following layers. This basic neural network architecture does not respect spatial relations of input values, which CNNs do. Every neuron in the same layer of a CNN uses the same shared weights (a two dimensional matrix in the case of an image) and the same shared bias, together forming the filter. Using a two dimensional weight matrix ensures



(a) Example for a convolutional neural network architecture.



(b) Convolution and Pooling Operations

Figure 2.2.: Convolutional Neural Network. Based on Park and Lee, 2020.

that every neuron in that layer would detect the same feature in any location of the input, so patterns can be recognized even if they are shifted in location. Every layer can have several filters and outputs a feature map for each. Figure 2.2a shows an example architecture of a CNN. The first convolution layer has three filters, one for each of the created feature maps. The outputs of a convolution layer are determined by moving the filter along the input as shown in figure 2.2b at a defined step size. A convolution layer is usually followed by a pooling layer that reduces the layer size by focusing on the most important information. An example is max-pooling as shown in figure 2.2b (Nielsen, 2015). After the convolution and pooling layers follows a flattening operation moving the data from two dimensional to one dimensional space. The last layer(s) are fully-connected as described in section 2.1.1 (Zaman et al., 2023). CNNs are specialized on finding patterns in more dimensional input data (e.g. audio spectrograms) without processing the relationship between those patterns.

2.1.5. Transformer Encoder

The self-attention mechanism and transformers were introduced in Vaswani et al., 2017 and have since experienced increasing popularity for natural language processing (Dosovitskiy et al., 2020). The attention mechanism was used before the introduction of the transformer, but the novelty of the transformer is that it is solely based on attention. The model proposed in Vaswani et al., 2017 improves performance while reducing training time on natural language processing tasks compared to existing sequence-based models (usually Recurrent Neural

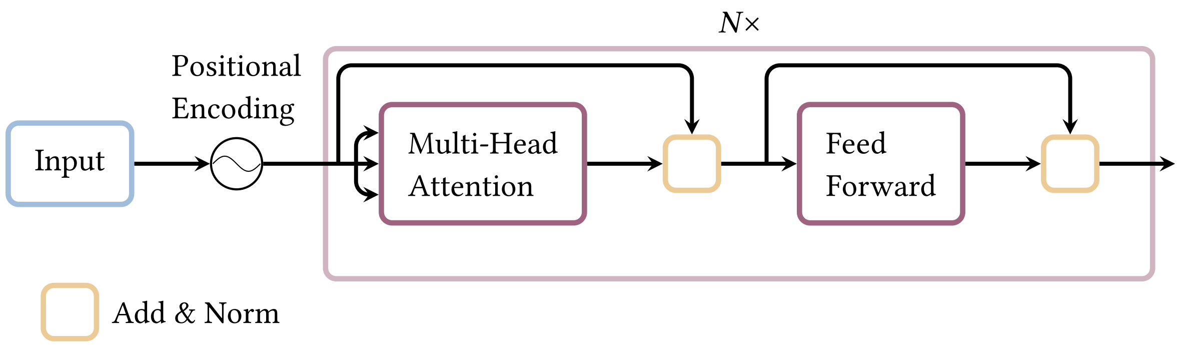


Figure 2.3.: Transformer Encoder. Adapted from Fundel et al., 2023; Vaswani et al., 2017.

Networks (RNNs)[1] until then). The transformer is an encoder-decoder architecture, where the encoder creates a low level representation of the input and the decoder translates that representation into the output (Vaswani et al., 2017). Classification tasks can use the encoder part of the transformer (Soyalp et al., 2021), which figure 2.3 shows the basic architecture of. The encoder consists of N identical layers. First, the input is split into tokens, with each token receiving some positional encoding. This is followed by the multi-head attention, which (compared to single-head attention) is computed several times in parallel, allowing the model to learn different representations of the data (Vaswani et al., 2017). For every token the model computes a query vector, a key vector and a value vector that depend on every token of the input. Between all token pairs the model then calculates an attention weight. The output of the multi-head attention is for each token the weighted sum of all value vectors. The outputs are added to the input tokens and normalized. After that follows a feed forward neural network. The output is again added to the networks input and normalized. Finally the model outputs probabilities for each class (Fundel et al., 2023). An advantage of the transformer architectures is its ability to handle variance in the input length and to keep track of the global context of information (Zaman et al., 2023). The transformer encoder can find relationships between different parts of the input data, but do not have the CNN's capability of finding patterns in more dimensional data.

2.1.6. Hybrid Deep Learning Models

In hybrid models different deep learning architectures are combined to take advantage of strengths from each of the used models (Zaman et al., 2023). E.g. a CNN-transformer hybrid would have the CNN's capability of processing spatial information and the transformer's self-attention mechanism to find correlations between different parts of the input data.

[1]RNNs are another popular deep learning architecture, where the output of a neuron can also be influenced by the output of the same neuron at earlier points in time. This enables the network to learn temporal information (Nielsen, 2015).

2.2. Audio Classification Using Deep Learning

As I am working with acoustic bat call data in my thesis, this section is about the classification of audio data in general and the classification of acoustic bat recordings in particular - both in the context of deep learning.

2.2.1. Audio Input Features

Before a deep learning model is used to learn on or classify an audio signal, the signal is usually transformed from the raw signal into a different representation. This representation is called a feature[2]. The starting point is a digital signal quantized and sampled at a certain samplerate (Serizel et al., 2017). The most common features are time frequency representations like spectrograms extracted by Short-Time Fourier Transform (STFT) and mel-spectrograms (Zaman et al., 2023). A spectrogram displays requency bands over time. In the mel-spectrogram the frequency axis is adjusted to closer approximate human hearing. Spectrograms and mel-spectrograms are only two of many possible audio features. There other two dimensional features as well as one dimensional features in both time and frequency space (Serizel et al., 2017). Zualkernan et al., 2020 compared different input features including spectrograms and mel-spectrograms with a CNN architecture for bat call classification and achieved the highest f1-score using mel-spectrograms.

2.2.2. Models for Audio Classification

Bat call classification is an Environmental Sound Classification (ESC) task (Bansal & Garg, 2022) in the broader context. This section gives an overview of existing deep learning methods in the ESC domain and in general audio classification. Zaman et al., 2023 is a survey on audio classification using deep learning including publications until 2023. The survey found models for ESC using various architectures. The basic architectures CNN, RNN and transformers as well as hybrid models have been applied to ESC tasks. The hybrid deep learning models found in the survey are combinations of CNN-RNN, CNN-transformers and CNN-RNN-transformers. The transformer architecture (section 2.1.5) was also adapted for audio classification tasks. The Audio Spectrogram Transformer (AST) (Gong et al., 2021) and the The Patchout faSt Spectrogram Transformer (PaSST) (Koutini et al., 2022) are two of those specifically for audio spectrograms. The PaSST model reduces the needed computational

[2]Note that the same term (feature) is used for features learned by deep learning models during training as described in section 2.1. Both features have a similar meaning - a different representation of the raw data, but are different from a user perspective. An input feature as described in this section is usually created before training.

resources for transformers in audio classification (Zaman et al., 2023). According to the survey, more hybrid models and transformers will be used in the future of deep learning (Zaman et al., 2023). Pretrained audio classification models could be used for bat call classification with transfer learning.

2.2.3. Models for Bat Call Classification

There are several approaches to bat call classification using machine learning techniques, some of which use traditional machine learning (e.g. Armitage and Ober, 2010; Roemer et al., 2021; Ruiz et al., 2019; Zamora-Gutierrez et al., 2016). There are also some commercial tools (e.g. bcAdmin[3], Sonobat[4], Kaleidoscope Pro[5]). The models used in this thesis are all deep learning models and the most recently published models also mainly use deep learning. In the following I describe existing deep learning models for bat call classification.

There are early neural network based approaches using a basic feed forward neural network (Parsons & Jones, 2000) or an evolutionary neural network (Mirzaei et al., 2012). They mostly use one hidden layer which does not qualify the neural networks as deep learning architectures.

In recent years many deep learning models for bat call classification were published. With the increasing popularity of CNNs for audio classification in the last decade (Zaman et al., 2023), many approaches using CNNs for bat call species classification have been developed (Chen et al., 2020; Dierckx et al., 2022; Hu et al., 2020; Kobayashi et al., 2021; Zualkernan et al., 2020, 2021). They all classify bat species with a varying amount of classes and slightly different CNN architectures. Schwab et al., 2022 use a two-stage CNN architecture, first classifying bat vs. not bat and then classifying the species. The most recent models mostly use transformer and/or hybrid models. Bellafkir et al., 2022 use DeiT-tiny, a ViT architecture. Alipek et al., 2023 and Vogelbacher et al., 2023 both use a combination of CNN and RNN. Aodha et al., 2018 started with a CNN model, for detecting bat calls (without classification) by annotating the training data with bounding boxes. In Aodha et al., 2022, the results of the 2018 paper could be improved by using a CNN-based encoder and decoder with self-attention and adding the ability to classify the bat call species. The authors report a performance improvement when adding self-attention, that they attribute to the attentions' ability of connecting information even over a longer time span. The model was trained and tested on five different datasets, achieving a minimum accuracy of 78% on a dataset recorded in the UK containing 17 species. The new model is called BatDetect2 (Aodha et al., 2022). Fundel et al., 2023 also use a transformer-CNN combination calling it BioAcoustic Transformer (BAT). The

[3]https://ecoobs.de/produkte/software/bcadmin
[4]https://sonobat.com
[5]https://www.wildlifeacoustics.com/products/kaleidoscope

model achieves f1-scores of up to 76% (micro) and 70% (macro) on a multilabel dataset with 18 species. The dataset was synthetically generated from a dataset where each sample only contained one of 18 german bat species (Fundel et al., 2023). All these models classify bat calls with different aspects as their main focus, but none of them has the additional challenge of drone noise in the audio recordings.

2.2.4. Models for Bat Call Classification with Drone Noise

The bachelor thesis by Mauson, 2022 compares pre-trained CNNs and ViTs on a binary classification task (bat vs. no bat) using a dataset that contains drone noise in addition to the bat calls. The bat calls are extracted manually from spectrograms. The best results are obtained using a ViT with an accuracy of 94.4% and a Matthews correlation coefficient of 88.8% (see also Taefi et al., 2024). To the best of my knowledge Mauson, 2022; Taefi et al., 2024 is the only work on a deep learning model for bat call classification were the audio data also contains drone noise and is therefore the approach closest to the one in my thesis.

3. Methods

For this thesis I tested some of the existing architectures mentioned in section 2.2. I tested two models (BatDetect2 and BAT) that were designed specifically for bat call classification and I tested one model (PaSST) that was designed for audio classification tasks in general. This chapter describes the dataset and how I modified it, the models and how I trained and evaluated them and the used software and hardware.

3.1. Dataset

The data I used to train and test the deep learning models in this thesis was collected and labeled as part of the Drones4Bats project mentioned in the introduction (chapter 1). The following subsections describe the recording setup, the labeling, the dataset distribution, how I handled class imbalances and preprocessing steps.

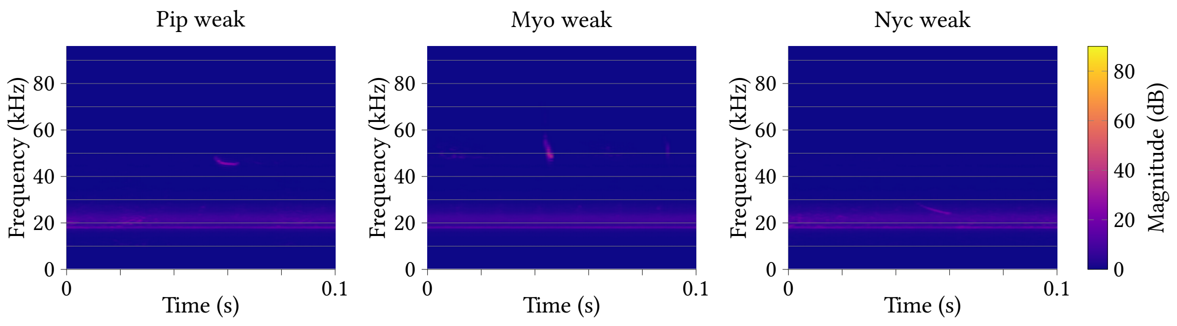
3.1.1. Data Collection

The data (with and without drone noise) was recorded over several months in three different locations within Germany in 2022. The recordings were made with an automatic bat detector (Song Meter SM4BAT FS, Wildlife Acoustics, Inc. using a sampling rate of 500 kHz) two meters above ground level. For the recordings with drone noise there was a multicopter (ConVecDro by Third Element Aviation GmbH) at a height of 20 meters (Roswag et al., 2024).

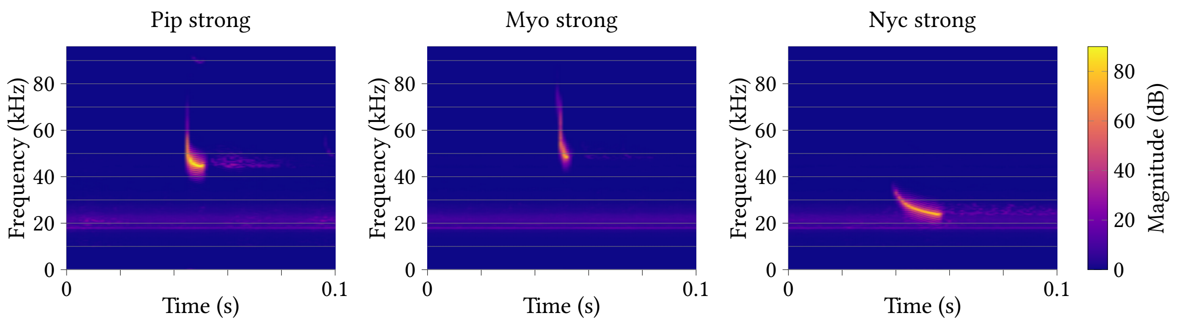
3.1.2. Labeling

Before labeling, the recordings were split into one second long files. The data was labeled by an expert as either noise or as one to three groups of bat species, one of each mainly containing calls from Pipistrellus, Myotis and Nyctaloid. I call the groups Pip, Myo and Nyc respectively in my thesis. The species were grouped based on their call characteristics (Roswag et al., 2024), resulting in a multilabel dataset with three possible labels and 8 different labeling options (labelsets). Before I used the data with the neural networks I created a csv file with the file names and class information[1].

Inspired by Aodha et al., 2022 I calculated the average of all spectrograms from files containing drone noise (and no bat calls). I then added exemplary bat calls from each of the three groups (from files without drone noise). Figure 3.1 shows the resulting spectrograms. The spectograms show 100 ms long of a full 1 s recording. Figure 3.1a shows that the drone noise can potentially blend with weak calls from the Nyc group. The exemplary calls cover a frequency range of about 20 kHz at the lower and to about 80 kHz at the upper end. The drone noise is strongest at around 20 kHz.

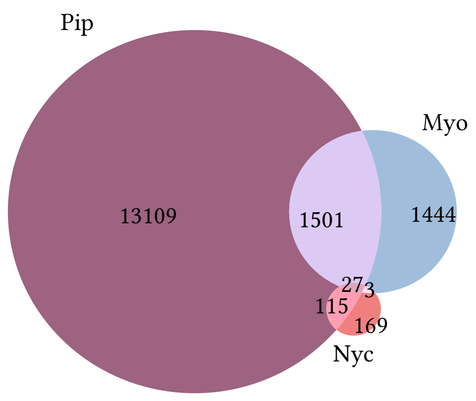


(a)

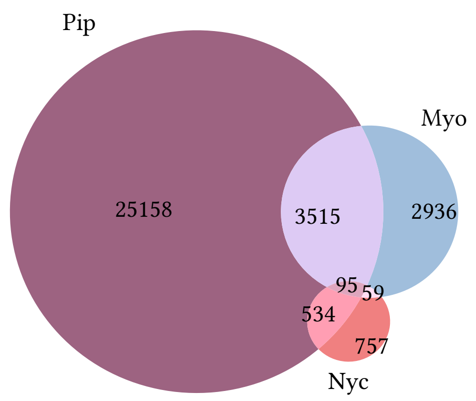


(b)

Figure 3.1.: Spectrograms of bat calls with drone noise. Each spectrogram contains an exemplary bat call from each of the three groups - either particularly (a) weak or (b) strong. Additionally the spectrograms contain the average noise level over all drone noise files (horizontal band at around 20 kHz).



(a) Dataset with drone noise



(b) Dataset without drone noise

Figure 3.2.: Dataset distributions.

3.1.3. Class Distribution

The venn diagrams in figure 3.2 show a visual representation of the class label distribution in the dataset. Pip is the most common label followed by Myo and at last Nyc with significant differences between all three stages. The numbers also show that there is about double the amount of data without drone noise, than there is with drone noise. The tables 3.1 show the number of samples and the Imbalance Ratio Per Label (IRLbl) for each labelset. The higher the IRLbl the more underrepresented that label is compared to the most common label. The Mean Imbalance Ratio (MeanIR) is the mean of the IRLbl for all labels (Tarekegn et al., 2021). The number of noise samples is similar to the Pip class in both datasets. The IRLbl is high for all labelsets containing Myo or Nyc, with the dataset with drone noise having an even higher MeanIR than the dataset without drone noise. In summary, the dataset is highly imbalanced with Pip being overrepresented and Myo and Nyc being underrepresented.

3.1.4. Managing Class Imbalances

Class imbalance in the training dataset can have a significant impact on the performance of an audio classification model, highlighting the importance of balancing the dataset (Abayomi-Alli et al., 2022). Since most classifiers' training is based on reducing a global error rate the models tend to be biased towards majority classes, when training on an imbalanced dataset (Charte et al., 2013). To address the class imbalance problem I used specific evaluation metrics (c.f. section 2.2.3) and tested two different approaches to modify the dataset before training.

[1]each class is either 0 or 1 for every sample, so a sample containing all three species groups would be encoded as [1 1 1] and a file with only noise and no bat calls as [0 0 0] in the format [Pip Myo Nyc].

Table 3.1.: The number of recordings and the imbalance ratio for each label(-set) in the dataset.

(a) Dataset with drone noise

Label(-set)	**# Samples**	**IRLbl**
Pip	13 109	1
Myo	1444	9
Nyc	169	78
Pip/Myo	1501	9
Pip/Nyc	115	114
Myo/Nyc	3	4370
Pip/Myo/Nyc	27	486
(Drone) Noise	8737	2

Number of recordings = 25105
MeanIR = 633.38

(b) Dataset without drone noise

Label(-set)	**# Samples**	**IRLbl**
Pip	25 158	1
Myo	2936	9
Nyc	757	33
Pip/Myo	3515	7
Pip/Nyc	534	47
Myo/Nyc	59	426
Pip/Myo/Nyc	95	265
Noise	20 242	1

Number of recordings = 53297
MeanIR = 98.63

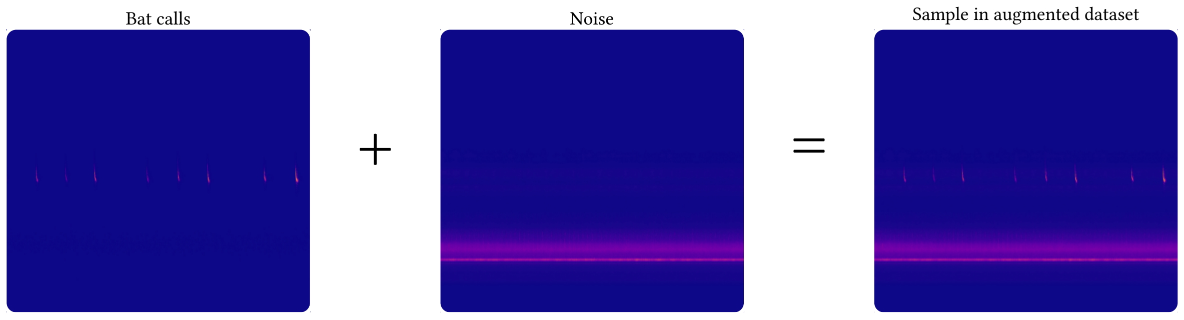


Figure 3.3.: Augmented dataset.

A method to handle class imbalances is data augmentation, where the existing data is modified to obtain artificially constructed data, to increase the number of samples in the dataset. Class-specific data augmentation has helped with a number of problems that arise with imbalanced datasets (Abayomi-Alli et al., 2022). For this thesis I took the data from the dataset without drone noise - that was less imbalanced and significantly larger - and added each sample with a drone noise sample without bat calls from the smaller dataset. Since there were more files in the dataset without drone noise than files containing drone noise, I had to use the drone noise files repeatedly. To keep evaluation with validation and test set unbiased I only used drone noise files from the training set. I randomly assigned one of the drone noise files to each of the files inside the larger dataset. This way I generated a larger dataset especially with more samples of the underrepresented classes.

Another straightforward method to solve the imbalance problem in a dataset is resampling, where samples of majority classes are dropped and samples of minority classes are duplicated.

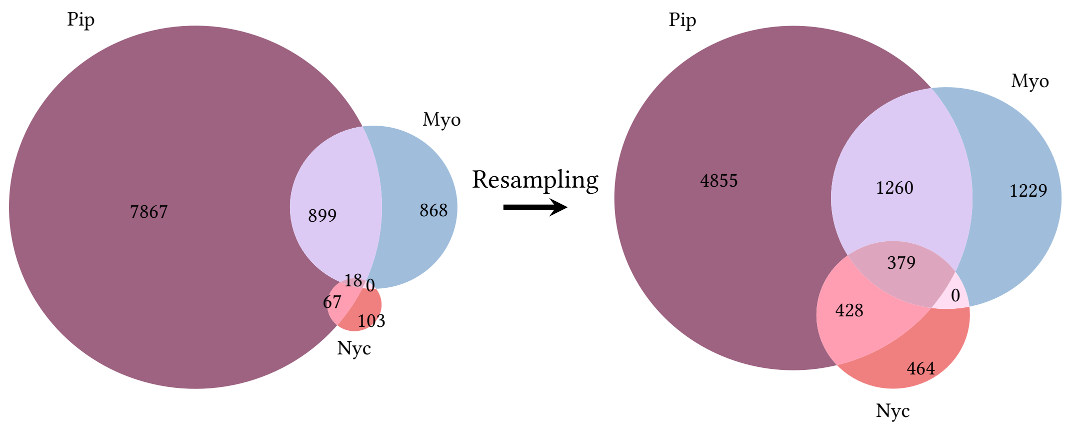


Figure 3.4.: Resampling Example on the training dataset with 40% Label Powerset Random Undersampling (LP-RUS) and 20% Label Powerset Random Oversampling (LP-ROS).

Label Powerset Random Undersampling (LP-RUS) and Label Powerset Random Oversampling (LP-ROS) are resampling techniques specifically for multilabel classification and work based on labelsets. They were proposed by Charte et al., 2013. LP-RUS deletes samples until the dataset is reduced to the specified percentage and LP-ROS duplicates samples until the dataset size is increased by the specified percentage. I applied[2] both techniques alone and in combination with 0 and 40% for LP-RUS and 0, 20 and 40% for LP-ROS. I applied LP-RUS with 60% to the augmented dataset to reduce the amount of data, while still having more distinct samples of the minority classes. I applied the same three stages of LP-ROS to the augmented dataset (0, 20 and 40%). It is to be noted that oversampling can lead to overfitting, while undersampling can lead to loss of useful information (Tarekegn et al., 2021). Applying resampling heavily reduced the MeanIR from 633.38 to one digit numbers. To better compare the results I calculated the density for each dataset. The density as a measure for imbalanced multilabel datasets is the mean number of labels that belong to each sample divided by the maximum number of labels (Bernardini et al., 2014). The exact label distributions with number of samples, IRLbl per label(-set) and density can be found in appendix A.2 for each tested combination. I refer to the dataset that originally contained drone noise as “dro” and to the artificially created dataset as “aug” in graphics and tables. I tested different combinations of LP-ROS and LP-RUS amounts to find a good trade-off between the negative and positive effects on the models’ performance, but these methods are not able to increase the number of distinct samples for less represented classes.

When splitting a dataset (e.g. into training, validation and test sets), the original class distribution should be preserved by using stratified sampling (Sechidis et al., 2011). To achieve

[2]Python implementation: https://github.com/manuel-rdz/multilabel-dataset-resampling-algorithms

this for the given multilabel dataset, I used the Python library skmultilearn (Szymański & Kajdanowicz, 2017) and used the same subsets for each modified training set to achieve better comparability. I used 60% of the data for training and 20% for each validation set and test set. Assuming that the dataset reflects a real world distribution of the class labels I only applied resampling and augmentation to the training set and not to the validation and test set (Shorten & Khoshgoftaar, 2019).

3.1.5. Preprocessing

The data was recorded at a sampling rate of $sr_0 = 500\,\text{kHz}$. However, the highest relevant frequency of the bat calls contained in the dataset does not exceed 80 kHz (section 3.1.2 and Roswag et al., 2024). To eliminate redundant data while reducing the amount of data by 61.6%, the recordings were downsampled to $sr_1 = 192\,\text{kHz}$.

As mentioned the recordings were split into one second long files. Therefore, the last file from each recording is usually shorter than one second. For models that require equal input lengths I used zero padding to correct shorter file lengths.

3.2. The Models

Following the prognosis of Zaman et al., 2023 (see also section 2.2.2) I focused on hybrid models and a transformer model. I tested three different models that were all pretrained - BatDetect2[3] (Aodha et al., 2022) and BAT[4] (Fundel et al., 2023) for bat call classification and the PaSST[5] model for general audio classification (Koutini et al., 2022). BatDetect2 and BAT I tested without further training on both the dataset with drone noise and the dataset without drone noise. Both models were trained on a larger number of output classes compared to the dataset used in this thesis (see section 3.1). Therefore when I applied an existing model to the dataset I matched the output classes to the classes in the dataset following the mapping in Roswag et al., 2024. The mapping for both models can be found in appendix A.1. I also tested BAT and PaSST with transfer learning. This section describes the three models and how I trained and/or tested them. Table 3.2 shows an overview of all three models including basic information about and how I used each model.

[3]https://github.com/macaodha/batdetect2
[4]https://github.com/FrankFundel/BAT
[5]https://github.com/kkoutini/PaSST

Table 3.2.: Overview of the three models.

	BatDetect2 (Aodha et al., 2022)	**BAT** (Fundel et al., 2023)	**PaSST** (Koutini et al., 2022)
Architecture	CNN-based encoder-decoder with attention	CNN-transformer	Transformer (for audio spectrograms)
Task	Classification & detection	Classification	Classification
Pretraining dataset	Bats (17 species)	Bats (18 species)	ESC-50
Tested without transferlearning	✗	✗	
Tested with tansferlearning		✗	✗

3.2.1. BatDetect2

BatDetect2 uses a CNN based encoder-decoder for bat call detection and classification. The encoder extracts features from the input spectrogram and the decoder generates the predicted location on time and frequency axis and the species label for each call. Skip connections are used to share features between encoder and decoder. The model has a self-attention layer between encoder and decoder that only operates along the time axis (Aodha et al., 2022).

I tested the BatDetect2 model only without further training, because the annotations of the training dataset of that model included bounding boxes around each call in a file (Aodha et al., 2022), which exceeded the annotations of the given dataset.

3.2.2. BAT

BAT is a CNN-transformer hybrid model designed for bat call classification. The CNN takes sequences of spectrogram patches and extracts features from each patch. The transformer encoder then finds correlations between the patches (Fundel et al., 2023).

Before applying transfer learning to the pretrained model, I reduced the number of output classes to three and froze all layers except the last (classification) layer. I trained the model for 25 epochs with a learning rate of $lr = 5 \cdot 10^{-4}$ like in Fundel et al., 2023. I resampled my data to the samplerate the source model was trained on $sr = 220.5\,\text{kHz}$ and used a batch size of 32, binary cross entropy loss and adam optimizer.

3.2.3. PaSST

PaSST is based on the ViT (Dosovitskiy et al., 2020), which is an adjusted transformer architecture for image processing, while PaSST is specialized on audio spectrograms (Koutini et al., 2022). The PaSST architecture with focus on the adjusted positional encoding is shown in figure 3.5. The model takes mel spectrograms as input. The positional encoding used in the transformer encoder is adapted to include both frequency and time encoding. This adaption enables the model to train on datasets with an input length deviating from the dataset used for pretraining. The model then uses patchout, which reduces the amount of data by dropping parts of the input sequence. The authors tested different forms of patchout, the model used in this thesis uses structured patchout, where the model drops random frequency bins. There is no overlap between patches. At this point the classification token is added. Then follows the transformer encoder (see also section 2.1.5) including self-attention. The PaSST has a higher performance and a higher training speed than CNNs on a large audio dataset. PaSST is also faster than AST (Koutini et al., 2022).

Koutini et al., 2022 provides pretrained models trained on different datasets. I used the model[6] pretrained on fold 4 of ESC-50[7], a large open source dataset for ESC. Before applying transfer learning to the pretrained model, I reduced the number of output classes to three and froze all layers except the last (classification) layer. I trained the model for 5 epochs with a learning rate of $lr = 5 \cdot 10^{-4}$. Then I chose the model with the highest f1-scores after 5 epochs and continued the training to a total of 16 epochs. I used a batch size of 8, binary cross entropy loss and adam optimizer. The samplerate was set to $sr = 192\,\text{kHz}$ and the minimum and maximum frequencies to $f_{min} = 10\,\text{kHz}$ and $f_{max} = 80\,\text{kHz}$ respectively.

3.3. Evaluation Metrics

This section introduces the evaluation metrics I used to evaluate and compare the models. The metrics were calculated using the python library scikit-learn version 1.3.2 (Pedregosa et al., 2012). I tested the models from two different perspectives. First I interpreted the results from a multilabel classification view. Second I interpreted the results as those of a binary classifier. In addition to the metrics described in the following sections I compared model sizes and prediction times.

[6] https://github.com/kkoutini/PaSST/releases/tag/v.0.0.6
[7] https://github.com/karolpiczak/ESC-50

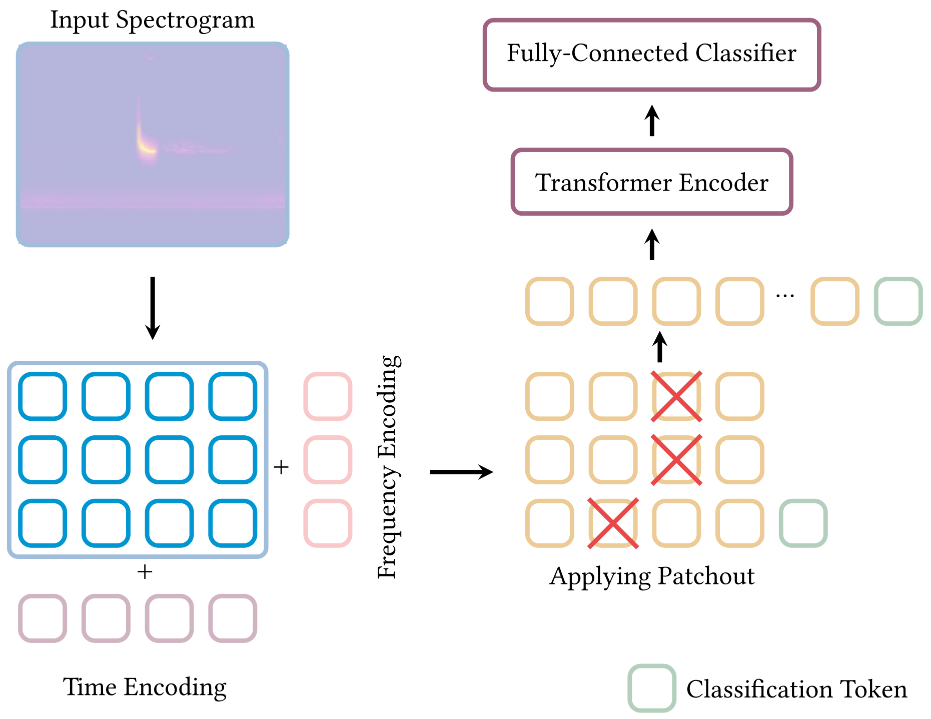


Figure 3.5.: The PaSST model. Adapted from Dosovitskiy et al., 2020; Koutini et al., 2022. For more details on the transformer encoder see figure 2.3.

3.3.1. The Metrics

This section introduces the metrics that I used.

F1-score, Precision, Recall

The f1-score is the harmonic mean of precision and recall, where precision is the ratio of true positive predictions to all positive predictions and recall is the ratio of true positive predictions to all true labels. A f1-score value of 100% means that all positive labels were recognized as positive and there were no false-positives (Owusu-Adjei et al., 2023).

ROC-Curve

The ROC-curve (Receiver Operating Characteristic Curve) measures the models' capability to differentiate between classes. It is calculated with the raw probability outputs from the model, so it does not depend on a threshold (unlike all other metrics that I used). The ROC-Curve is a plot of recall against false positive rate for all thresholds. If the area under the ROC-curve

(AUC) is 0.5 the classifier acts like a random classifier, while an AUC of 1 would describe a perfect classifier (Rainio et al., 2024).

Confusion Matrix

The confusion matrix shows the number of samples predicted for each pair of true label and predicted label (Owusu-Adjei et al., 2023). The confusion matrix allows deeper insight into what causes the results of the before described metrics, because it contains information about true and false positives, as well as true and false negatives for each label.

3.3.2. Multilabel Evaluation

To evaluate a multilabel classifier the evaluation metrics can be either calculated for each label separately or averaged. Macro averaging means calculating the metric for each class separately and then taking the average. Micro averaging means to calculate the metric over all samples and then taking the average (Tarekegn et al., 2021). The macro f1-score enabled evaluation without a bias introduced by the datasets imbalance.

3.3.3. Binary Evaluation

Assuming a higher cost for misclassifications between bat and noise than between two bat label groups I also evaluated the models after grouping all three labels into one bat group.

The metrics described in the last section can be applied to results of binary classifiers without further adjustments. I summarized the results from my multilabel classifiers to a bat and a noise class. The bat class was true, if the sample was classified as any of the three bat groups and the noise class was true if none of three bat groups was classified.

3.4. Hardware and Software Specifications

The models were written in Visual Studio Code using Python on an Intel i7-11800H @ 2.30GHz, 32 GB RAM and a NVIDIA GeForce RTX 3050 Ti Laptop GPU. I executed the scripts from a Windows PowerShell console.

The models were trained using the Python libraries PyTorch (Paszke et al., 2019) and torchudio (Yang et al., 2021), because the chosen models (Aodha et al., 2022; Fundel et al., 2023; Koutini et al., 2022), were built using those libraries. I used the PyTorch Graphical Processing Unit (GPU) Acceleration. The code and dataset can be found on the appended storage medium.

4. Results

This chapter presents the results of my tests on the pretrained models without further training (BatDetect2 and BAT) both on data with and without drone noise and the results on the transferlearned models (BAT and PaSST), including the effects of resampling and augmentation on the training dataset. The results for BatDetect2, BAT and PaSST are reported at thresholds of 0.3, 0.5 and 0.2 respectively. The thresholds were chosen by comparing f1-scores at different thresholds. The results for all experiments are reported on the testset.

4.1. Pretrained Models

Table 4.1 shows the results for the pretrained models without further training. The top half of the table shows the results on the testset (with drone noise). Comparing the results on the testset shows that of the two pretrained models BatDetect2 outperformed BAT for all reported f1-scores and recall values and for most precision values. BAT achieved slightly higher precision values for the Pip and Nyc classes. The results on the dataset without drone noise (bottom half of the table) are not significantly different from the results on the dataset with drone noise. The largest differences are the results for Myo and Nyc classes, where the f1-scores on the dataset with drone noise are 0.022 and 0.030 higher respectively. The Pip class achieved the highest scores among all three classes. Both models achieved higher scores in binary classification than in multilabel classification.

Figure 4.1 shows the ROC-curves for both pretrained models on the testset, with BatDetect achieving an AUC value of 97.3% and BAT of 91.1%.

4.2. Transferlearned Models

This section first presents the results for BAT and PaSST for all tested resampling-augmentation combinations, followed by a more detailed evaluation of the model that achieved the highest f1-scores.

Table 4.1.: Results for pretrained models. The testset without drone noise had the same dimensions as the dataset with drone noise.

Model	**Metric**	**Pip**	**Myo**	**Nyc**	**Micro**	**Macro**	**Binary Bat**
	Dataset with drone noise [*]						
BatDetect2	F1-score	**0.943**	**0.772**	**0.464**	**0.906**	**0.726**	**0.955**
	Precision	0.902	**0.881**	0.344	**0.881**	**0.709**	**0.934**
	Recall	**0.987**	**0.687**	**0.714**	**0.933**	**0.796**	**0.978**
BAT	F1-score	0.894	0.431	0.403	0.814	0.576	0.907
	Precision	**0.942**	0.501	**0.410**	0.869	0.617	0.896
	Recall	0.851	0.378	0.397	0.765	0.542	0.919
	Dataset without drone noise [**]						
BatDetect2	F1-score	0.946	0.751	0.434	0.900	0.710	0.954
	Precision	0.903	0.765	0.295	0.857	0.654	0.921
	Recall	0.992	0.737	0.822	0.947	0.850	0.989
BAT	F1-score	0.894	0.389	0.356	0.800	0.546	0.901
	Precision	0.932	0.422	0.285	0.834	0.547	0.867
	Recall	0.858	0.360	0.475	0.769	0.564	0.938

[*] Results on testset (results on full set in appendix B.1).
[**] Reduced dataset with same dimensions as dataset with drone noise.

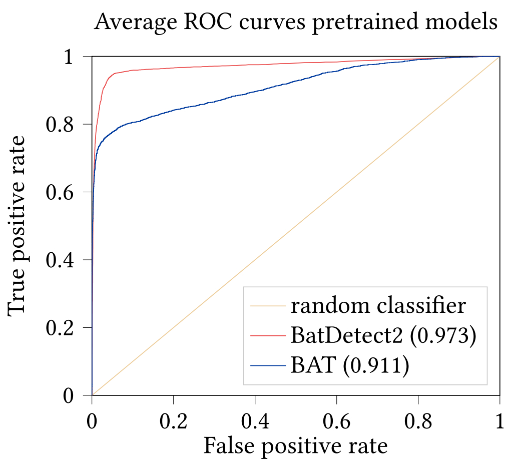
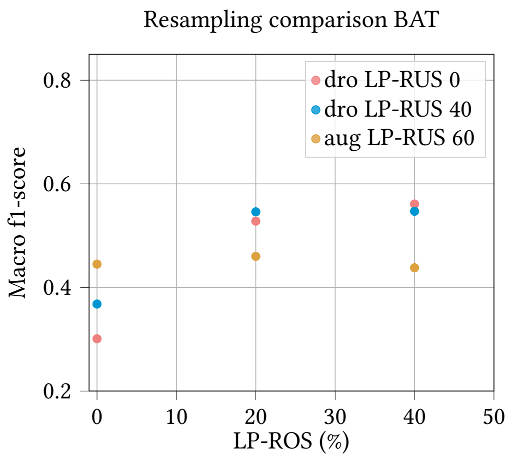


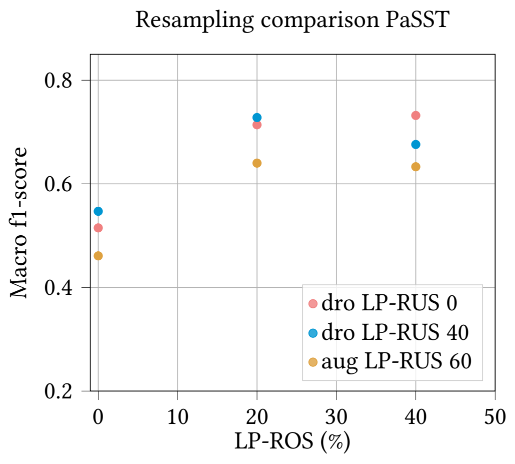
Figure 4.1.: ROC-curves and AUC for pretrained models on the testset.

4.2.1. Resampling and Augmentation

I compared different levels of LP-ROS and LP-RUS on the dataset with drone noise and on the augmented dataset for training. Figure 4.2 shows the resulting macro f1-score for all combinations. The plot shows the level of LP-ROS on the x-axis versus the macro f1-scores on the y-axis for the dataset with drone noise at 0 and 40% LP-RUS and for the augmented dataset at 60% LP-RUS. The results for BAT are shown in figure 4.2a. On the dataset with drone noise both LP-ROS and LP-RUS increase the macro f1-score. For LP-ROS the difference between 20% and 40% is minimal. At 0% LP-ROS the augmented dataset achieved the highest macro f1-score. For 20 and 40% LP-ROS the augmented dataset achieved the lowest macro f1-scores compared to the dataset with drone noise at the same oversampling level. Figure 4.2b shows the comparison results for PaSST. The highest macro F1-scores for the BAT model are at a similar level as the lowest results for PaSST, but the general distribution of results across different resampling levels follows a similar pattern for both models. An exception is the augmented dataset at 0% LP-ROS, which scored lowest compared to the drone noise dataset at the same level of oversampling for PaSST.



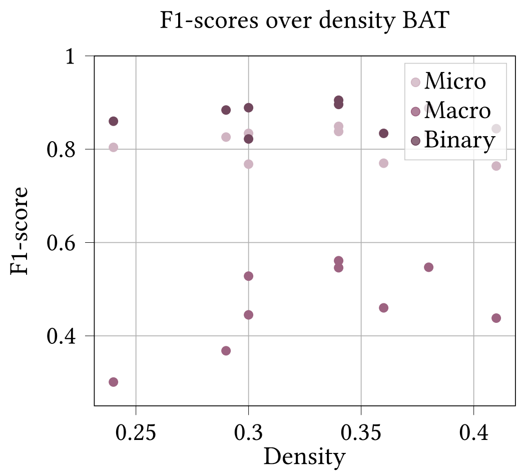
(a)



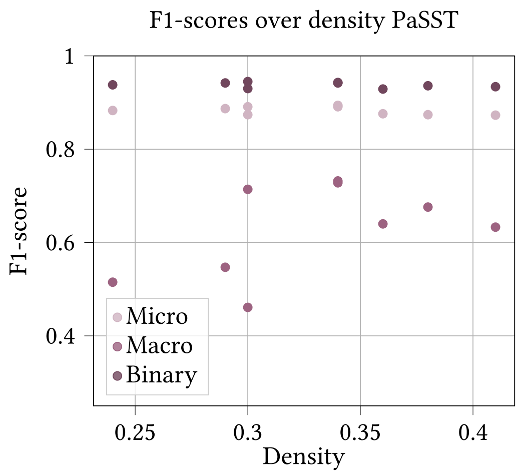
(b)

Figure 4.2.: Macro f1-scores for resampling-augmentation combinations for both models.

Figure 4.3 shows the micro and macro f1-scores and the f1-score on binary classification over the density of the training dataset. The density values for each combination are in appendix A.2. The macro f1-scores in the plots are the same as in 4.2, but with the density instead of LP-ROS on the x-axis. Additionally the plots show results for micro and binary f1-scores. The values for PaSST are generally higher than those for BAT. For both models micro and binary f1-scores remain at about the same level for all densities, but there is an increase of macro f1-score with density.

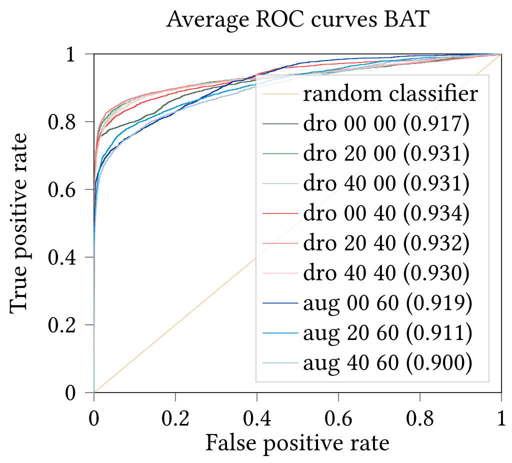


(a)

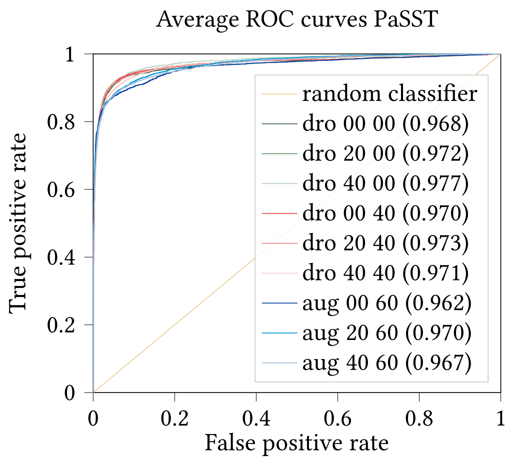


(b)

Figure 4.3.: Micro, macro and binary f1-scores over dataset density for both models.



(a)



(b)

Figure 4.4.: ROC-curves and AUC for both models for resampling-augmentation combinations.
The labels show: dataset, LP-ROS (%), LP-RUS (%), (AUC).

Figure 4.4 shows the ROC-curves and the AUC for all combinations. The ROC-curves and AUC values for BAT are more widely spread then those for PaSST are generally lower. For PaSST all AUC values are between 96.8% and 97.7% and for BAT between 90.0% and 94.3%. For both models the lowest AUC values result from training on the drone noise dataset without any resampling and on the augmented dataset for each level of oversampling.

4.2.2. Multilabel and Binary Evaluation

Table 4.2.: Results after transfer learning.

Model	Metric	Pip	Myo	Nyc	Micro	Macro	Binary Bat
BAT	F1-score	0.907	0.450	0.327	0.838	0.561	0.896
	Precision	**0.981**	**0.941**	0.178	**0.956**	0.733	**0.991**
	Recall	0.843	0.296	0.327	0.745	0.512	0.817
PaSST (5 ep.)	F1-score	0.934	0.731	0.530	0.894	0.732	0.943
	Precision	0.934	0.857	0.407	0.906	0.733	0.955
	Recall	0.935	0.637	0.762	0.883	0.778	0.930
PaSST (16 ep.)	F1-score	**0.941**	**0.756**	**0.658**	**0.906**	**0.785**	**0.949**
	Precision	0.937	0.803	**0.562**	0.908	**0.768**	0.957
	Recall	**0.945**	**0.714**	**0.794**	**0.904**	**0.818**	**0.941**

For both BAT and PaSST the highest macro f1-scores could be achieved applying 40% LP-ROS and no augmentation or undersampling to the dataset. The precision, recall and f1-scores for this combination for both models are reported in table 4.2. The results for all resampling and augmentation combinations can be found in appendix B.2. The results are reported after 5 epochs for PaSST and 25 epochs for BAT. The last rows show the results after training the PaSST model with the highest macro f1-scores after 5 epochs with continued training up to 16 epochs.

PaSST after 16 epochs constantly outperformed BAT and PaSST after 5 epochs. PaSST after 5 epochs mostly outperformed BAT. However, BAT achieved the highest precision values, except for Nyc class and macro precision, where PaSST after 16 epochs scored highest. The highest recall values were all achieved by PaSST. PaSST after 16 epochs achieved an AUC value of 98.2%. So PaSST after 16 epochs mostly achieved the highest scores with exceptions precision.

The results for binary evaluation were generally higher than those for multilabel evaluation. The highest f1-score after transferlearning was achieved by PaSST after 16 epochs with 94.9%. The highest f1-score on binary evaluation of all models tested in this thesis was achieved by BatDetect2 with 95.5% (table 4.1).

The confusion matrix in figure 4.6 for the PaSST model after 16 epochs for multilabel classification shows that 92.2% and 90.9% of noise and Pip samples respectively were classified correctly. Mispredictions for noise mainly resulted in Pip and vice versa. 18.1% of Myo samples were misclassified as noise. Most mispredictions for all other classes resulted in Pip or a labelset containing one correct class but missing or adding another one. Both the

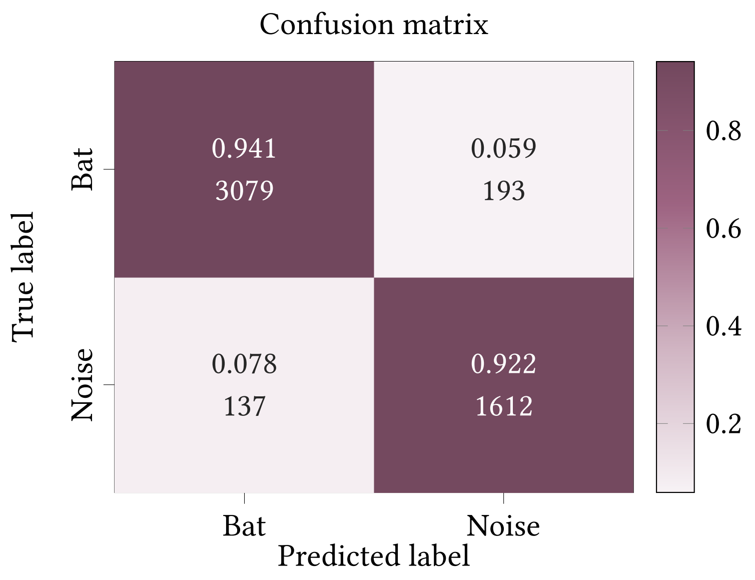


Figure 4.5.: Binary confusion matrix for the PaSST model with the highest f1-scores.

combination Myo-Nyc and Pip-Myo-Nyc stand out in the confusion matrix, because they were highly underrepresented in the testset. However, mispredictions followed the same pattern for these combinations.

Figure 4.5 shows the confusion matrix for binary classification where all bat labelsets are summarized into one bat class. The bat class was predicted correctly for 94.1% of samples.

4.3. Model Properties

Table 4.3 shows prediction times on Central Processing Unit (CPU) and GPU as well as the model size for all three models. BAT was the smallest model in size and the fastest in prediction time, followed by BatDetect2 and then PaSST. The prediction time for one file using PaSST on the CPU was 0.73 s. Using the GPU reduced the prediction time to 0.129 s. On BAT there was no significant difference in prediction times between CPU and GPU. BatDetect2 could only be evaluated on CPU with a prediction time 0.220 s faster than PaSST. There were significant differences in model size and prediction time between all three models.

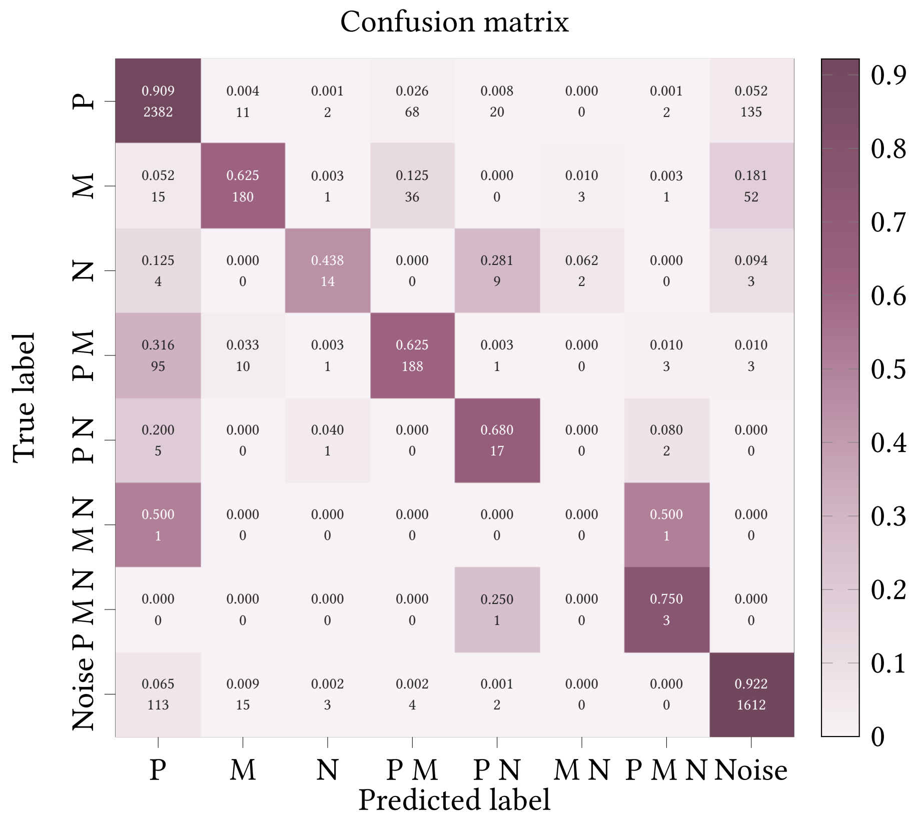


Figure 4.6.: Multilabel confusion matrix for the PaSST model with the highest f1-scores. The confusion matrix compares true labels to predicted labels for each labelset. Each cell shows the relative amount of all true labels of a class that were predicted as a label and the absolute number of calls for each combination in the matrix. P, M and N stand for Pip, Myo and Nyc respectively.

Table 4.3.: Prediction times and model size. The prediction time was tested on the same subset of 500 files with each model. The listed results are the average processing time for one file.

Model	**Prediction time (s)**		**Model size (MB)**
	CPU	GPU	
BatDetect2	0.510	N./A.	7.660*
BAT	0.040	0.046	0.292
PaSST	0.730	0.129	333.516

* Size of complete model package including scripts.

5. Discussion

The highest macro f1-score of the compared models was achieved with PaSST after 16 epochs using a training dataset with 40% LP-ROS. PaSST outperforms BAT in all f1-scores. The BatDetect2 model could achieve the highest binary f1-score (closely followed by PaSST) even without further training on a dataset with drone noise. Testing BatDetect2 and BAT without further training on datasets with and without drone noise showed no significant differences between the two datasets. In this section I discuss the comparison of the different pretrained and transferlearnend architectures, of binary vs. multilabel classification and how resampling and augmentation on the training dataset affect the results.

5.1. Architectures

A direct comparison of the results with those of existing models is not meaningful, as they mostly classify between more distinct species than the three classes used in this thesis (e.g. Bellafkir et al., 2022; Vogelbacher et al., 2023; Zualkernan et al., 2020). I compared three different architectures - a CNN with attention (BatDetect2), a CNN-transformer (BAT) and a transformer (PaSST). While the CNN can process spatial information and find patterns in the two-dimensional spectrogram (Nielsen, 2015), the transformer can find relationships between those patterns (Vaswani et al., 2017). There are many CNN based approaches to bat call classification. The high scores on the PaSST model as well as with other transformer/attention based methods (Bellafkir et al., 2022) indicate that transformers may be at least equal to CNNs on bat call classification tasks. A combination of attention and a CNN was used in Aodha et al., 2022 achieving promising results. Fundel et al., 2023 uses a CNN-transformer hybrid with lower scores than BatDetect2, but still achieving up to 91.5% on binary classification. Apart from the architecture the models differ in a few other parameters. Another difference are the input features. While BAT uses spectrograms, PaSST uses mel-spectrograms that are also superior to regular spectrograms in Zualkernan et al., 2020.

Hestness et al., 2017 find that increasing model size often increases accuracy, which aligns with the results for the models in this thesis. The f1-scores for all three models increased with model size. The larger models required a longer prediction time per sample.

The BatDetect2 model detects calls before classification (Aodha et al., 2022) and the BAT model performs peak detection to identify regions of interest (Fundel et al., 2023). The PaSST model does not use any form of detection before classification (Koutini et al., 2022) and could still outperform both other models. However, the results for BatDetect2 were promising even without further training, whether this was due to the detection mechanism or another parameter of the model requires further testing. PaSST generally achieved the highest f1-scores and recall values. Where the precision values was higher on BAT the distance to the values achieved by PaSST was relatively small. PaSST is pretrained on a dataset for ESC, that is not bat specific (Piczak, 2015). This shows that a pretraning dataset closer to the downstream task is not necessarily superior to another pretraning dataset.

The f1-scores of the BAT model did not improve after transfer learning. This could be due to a poor choice of hyperparameters or the model architecture in general. Further testing is needed to confirm. However the results showed that using resampling methods can improve the results when working with an imbalanced dataset.

5.2. Multilabel and Binary Evaluation

Both BatDetect2 and BAT before transfer learning performed significantly worse on Myo and Nyc as compared to Pip. Aodha et al., 2022 reported the Myotis species as challenging and the per-species precision-recall curves included in the paper indicate the same for Nyctalus species. When I trained the models on the Drones4Bats dataset the models still performed significantly better on Pip than on the other two classes. There was an increase in f1-score with an increase in number of samples in the training dataset. This can be supported by Aodha et al., 2022 who also report an increasing performance on species with a increasing number of calls from that species in the training set. However, the call characteristics may also influence the results.

Similar results for Nyc for the dataset with drone noise and the dataset without drone noise were unexpected. This leads to the assumption that the drone noise does not have as much of an effect as expected, even with the spectrograms in figure 3.1 showing that Nyctalus calls are within the same spectral range as the drone noise.

The scores when interpreting the models as a binary classifier are generally higher than when interpreting them as multilabel classifiers, because the binary evaluation accepted a classification as correct as long as the model discriminated correctly between bat and noise. A misclassification between the bat groups was not relevant. PaSST achieved generally higher f1-scores than BAT, same as with multilabel classification, but the highest f1-score (with little distance to PaSST) was achieved by BatDetect2. The model has a greater focus towards finding calls through its ability to detect calls before classifying them (Aodha et al., 2022).

The PaSST model has no information about locations in time or frequency of the bat calls in the files.

5.3. Imbalance

In figure 4.2 I compared resampling-augmentation combinations for both models. The results showed an increasing macro f1-score from no oversampling to 20% LP-ROS, but no significant difference from 20% to 40% LP-ROS. Since oversampling only duplicates samples there is no new information which can also introduce the negative effect of overfitting (Tarekegn et al., 2021). So the oversampling may improve the macro f1-score, as long as the positive effects from balancing outweigh the negative effects. For the PaSST model with the drone noise dataset at 40% LP-RUS the macro f1-score decreased from 20% to 40% LP-ROS. Further testing could reveal if this is a general trend for all combinations. Applying LP-RUS with 40% achieved higher scores for 0 and 20% LP-ROS, even though the training dataset is smaller than the one without LP-RUS. So a larger dataset can achieve lower scores compared to a smaller dataset that is more balanced. At 40% LP-ROS the macro f1-score was higher without undersampling, further testing is needed to see if the results could be improved with more oversampling. The augmented dataset achieved the lowest macro f1-scores (except for the BAT model at 0% LP-ROS). The augmented dataset was larger than the drone noise training dataset, but it was less similar to the testset. This could be the reason for the lower performance of the models trained on the augmented dataset. Testing on different datasets could confirm this assumption. The plots in figure 4.3 show a correlation between density and macro f1-scores. This aligns with the findings of Bernardini et al., 2014 who also found a correlation between f1-score and density. The lowest density dataset mostly received the lowest macro f1-score. From a certain point there was no longer an increasing macro f1-score with higher density. This could be explained by the fact that oversampling simply duplicates existing samples and does not add new distinct information to the dataset. Micro f1-score and binary f1-score did not show large differences for different densities. This could be, because both are mainly influenced by the Pip class, because it is overrepresented in the testset. Increasing the density increased the f1-score for underrepresented classes, but had little to no effect on overrepresented classes.

Using the augmented dataset without oversampling and 60% LP-RUS as training dataset is the only test were the results did not follow the same pattern on BAT and PaSST (figure 4.2). For BAT it achieved the highest macro f1-score of the three tests without oversampling and on PaSST it achieved the lowest of the three. Looking at the precision and recall values in appendix B.2 shows that this is mainly due to PaSST achieving 0% precision and recall on Nyc class, while BAT was able to achieve 44.4% precision. The recall value is also near 0%, but the resulting f1-score is still 11.1%. All other tests for both PaSST and BAT without oversampling received 0% on both precision and recall. An explanation for the high precision value could be that BAT was pretrained on bat call data with most represented species in the training dataset Pipistrellus, Myotios and Nytaloid (Fundel et al., 2023) and therefore had information about the Nyc class prior to first training on the drone noise dataset, while PaSST did not (Koutini et al., 2022). So the larger number of distinct samples of the Nyc class in the augmented dataset may have been sufficient for the BAT model to improve performance compared to a model trained on the drone noise dataset. This would however not explain the results for BAT with the augmented dataset and 20 and 40% LP-ROS where the precision is reduced to under 20%.

The ROC-curve measures the model performance without the bias of a threshold and showed a similar distribution of results for the different resampling and augmentation combinations. I only reported micro averaged ROC-curves, so the resulting curves and values were biased towards Pip class, but the curves can still indicate that the choice of threshold for the model evaluation did not introduce a bias towards any of the models.

6. Conclusion

The use UAS could improve the monitoring of bats at wind farm sites (Taefi et al., 2024). The monitoring is important, because wind turbines can be a threat to bats (Lehnert et al., 2014). But UAS add the challenge of acoustic drone noise in the recordings, which may reduce the monitoring accuracy (Taefi et al., 2024). One previously used method for evaluating acoustic bat call recordings are deep learnig models (e.g. Aodha et al., 2022; Fundel et al., 2023 and Zualkernan et al., 2021). For my thesis I worked on answering the research question, which of the compared pretrained deep learning architectures achieves the highest f1-scores. More specifically, I compared neural network architectures, existing pretrained models and transferlearned models, interpreted the results as a binary classifier and a multilabel classifier and compared methods to handle imbalanced datasets and the input features.

The overall highest f1-scores were achieved by the PaSST model after 16 epochs using the oversampling method LP-ROS with 40% on the training dataset, with a micro f1-score of 90.6% and a macro f1-score of 78.5%. The highest binary f1-score was achieved by BatDetect2 with 95.5%.

The results of my experiments follow Taefi et al., 2024 and Mauson, 2022 in showing how the classification of echolocation bat calls could be achieved despite the challenge of drone noise in the audio files. Comparing models on data with and without drone noise showed that the drone noise may not impact the results of bat call classification with deep learning as much as expected. In my tests I used a dataset that groups bat species into three groups which is a smaller number compared to most other existing models that distinguish between e.g. 17 species for BatDetect2 (Aodha et al., 2022). PaSST outperformed BatDetect2 in macro f1-score, so if three classes are sufficient for a particular task the reduction in classes can improve the model performance. It would be interesting to see whether PaSST could still outperform the other models, when it is trained and tested on a dataset with more classes. Testing resampling and augmentation to address the imbalance problem showed that especially oversampling of minority classes improved the results on those classes. Of the models tested, PaSST (Koutini et al., 2022) achieved the highest macro f1-score, while BatDetect2 (Aodha et al., 2022) achieved the highest binary f1-score. Interpreting the results as a binary classifier over a multilabel classifier increases the f1-score by about 5% as compared to the micro f1-score for all models. My results could contribute to the current state of the art by demonstrating the potential of PaSST for bat call classification and the potential of methods to reduce class imbalance for a bat call dataset.

The resulting models could be adapted for use in the field for classifying bat calls that are recorded with a microphone and a UAS. The lightweight BAT model could still score 91.5% on binary classification and may be of interest for practical applications where a lower computational cost is of interest. The more computationally expensive model PaSST was also able to score relatively high on per label metrics and could therefore be of interest, when classification into the labels groups or generally a higher performance is important.

There were limitations to my work in different areas. The following gives an overview on those limitations and provides further research suggestions. There were some limitations regarding the dataset. In general a larger dataset can increase the performance of deep learning models (Cayir & Navruz, 2021), which can be supported by my results on resampling. So generally using more distinct training samples especially for minority classes could improve the results. Due to the small amount of samples for some labelsets the testset only contained a few samples of those labelsets. This may have introduced a bias to the results on those labelsets (however, testing BatDetect2 and BAT before transfer learning on the full drone noise dataset showed no significant differences. Refer to tables 4.1 and B.1). Testing and potentially retraining the model on data that was recorded with a microphone suspended from a drone could show how the model performs in a situation closer to a practical application. The results on the augmented dataset did not improve the results compared to the drone noise dataset, however data augmentation in general has shown to improve models (e.g. Abayomi-Alli et al., 2022). So testing other augmentation methods or the same augmented dataset, but without LP-RUS could be interesting, as well as testing more combinations of resampling, as my test results may have produced an incomplete picture. Other limitations were introduced through model choice. I only worked with pretrained models and only applied transfer learning to the last layer of each model. Future research could train models from scratch, which would require more computational resources. The number of tested architectures could be extended. Transformers have shown to be promising on the task, but e.g. CNN-RNN hybrids have recently been applied to bat call classification (Alipek et al., 2023; Vogelbacher et al., 2023). I only worked with supervised methods, but unsupervised methods could supplement a model. Both the CNN-RNN hybrid and unsupervised learning have been applied to bat call classification e.g. in Alipek et al., 2023 showing promising results. Though the most recent approaches to bat call classification mostly use deep learning methods, traditional machine learning has been successfully applied to the task before. According to Roemer et al., 2021 there is no extensive comparison of random forest and deep learning for bat call classification. It may be worth exploring traditional machine learning architectures on alone or in a hybrid with a deep learning model. In the training process I was only able to test a limited number of hyperparameter configurations. One method that might be worth exploring, because it can improve training with imbalanced datasets is a weighted loss function (Rengasamy et al., 2020). My training was limited to a number of epochs for each model. At the end for some training processes the validation loss was still decreasing, so there may be potential for further finetuning. The tested models only worked with spectrograms (Fundel et al., 2023) and mel spectrograms (Koutini et al., 2022). There exists a large variety of input features for neural networks computed from the raw audio waveform and it may be worth exploring different input features, concatenated feature vectors or feature learning (Serizel et al., 2017). The high scores for the BatDetect2 model suggest that exploring a model and dataset annotations that include information about detection in time and frequency in addition to classification annotations may have improved the results. Though it should be noted that the high scores could also be caused by any other parameter (combination) in the model.

The results of my thesis may contribute to future research on protecting biodiversity by reducing bat mortality caused by wind turbines while maintaining green energy production.

Bibliography

Abayomi-Alli, O. O., Damaševičius, R., Qazi, A., Adedoyin-Olowe, M., & Misra, S. (2022). Data Augmentation and Deep Learning Methods in Sound Classification: A Systematic Review. *Electronics*, *11*(22), 3795. https://doi.org/10.3390/electronics11223795

Alipek, S., Maelzer, M., Paumen, Y., Schauer-Weisshahn, H., & Moll, J. (2023). An Efficient Neural Network Design Incorporating Autoencoders for the Classification of Bat Echolocation Sounds. *Animals*, *13*(16), 2560. https://doi.org/10.3390/ani13162560

Aodha, O. M., Gibb, R., Barlow, K. E., Browning, E., Firman, M., Freeman, R., Harder, B., Kinsey, L., Mead, G. R., Newson, S. E., Pandourski, I., Parsons, S., Russ, J., Szodoray-Paradi, A., Szodoray-Paradi, F., Tilova, E., Girolami, M., Brostow, G., & Jones, K. E. (2018). Bat detective—Deep learning tools for bat acoustic signal detection. *PLOS Computational Biology*. https://doi.org/10.1371/journal.pcbi.1005995

Aodha, O. M., Martínez Balvanera, S., Damstra, E., Cooke, M., Eichinski, P., Browning, E., Barataud, M., Boughey, K., Coles, R., Giacomini, G., Swiney G., M. C. M., Obrist, M. K., Parsons, S., Sattler, T., & Jones, K. E. (2022). Towards a General Approach for Bat Echolocation Detection and Classification [This article is a preprint and has not been certified by peer review.]. *bioRxiv*. https://doi.org/10.1101/2022.12.14.520490

Armitage, D. W., & Ober, H. K. (2010). A comparison of supervised learning techniques in the classification of bat echolocation calls. *Ecological Informatics*, *5*(6), 465–473. https://doi.org/10.1016/j.ecoinf.2010.08.001

Bansal, A., & Garg, N. K. (2022). Environmental Sound Classification: A descriptive review of the literature. *Intelligent Systems with Applications*, *16*, 200115. https://doi.org/10.1016/j.iswa.2022.200115

Barthelmie, R. J., & Pryor, S. C. (2021). Climate Change Mitigation Potential of Wind Energy. *Climate*, *9*(9), 136. https://doi.org/10.3390/cli9090136

Bellafkir, H., Vogelbacher, M., Gottwald, J., Mühling, M., Korfhage, N., Lampe, P., Frieß, N., Nauss, T., & Freisleben, B. (2022). Bat Echolocation Call Detection and Species Recognition by Transformers with Self-attention. In *Intelligent systems and pattern recognition* (pp. 189–203). Springer International Publishing. https://doi.org/10.1007/978-3-031-08277-1_16

Bernardini, F. C., Silva, R. B. d., Rodovalho, R. M., & Meza, E. B. M. (2014). Cardinality and density measures and their influence to multi-label learning methods. *Learning and Nonlinear Models*, *12*(1), 53–71. https://doi.org/10.21528/lnlm-vol12-no1-art4

Cayir, A. N., & Navruz, T. S. (2021). Effect of dataset size on deep learning in voice recognition. *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*. https://doi.org/10.1109/hora52670.2021.9461395

Charte, F., Rivera, A., del Jesus, M. J., & Herrera, F. (2013). A First Approach to Deal with Imbalance in Multi-label Datasets. In *Hybrid artificial intelligent systems* (pp. 150–160). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-40846-5_16

Chen, X., Zhao, J., Chen, Y.-h., Zhou, W., & Hughes, A. C. (2020). Automatic standardized processing and identification of tropical bat calls using deep learning approaches. *Biological Conservation*, *241*, 108269. https://doi.org/10.1016/j.biocon.2019.108269

Dierckx, L., Beauvois, M., & Nijssen, S. (2022). Detection and Multi-label Classification of Bats. In *Advances in intelligent data analysis xx* (pp. 53–65). Springer International Publishing. https://doi.org/10.1007/978-3-031-01333-1_5

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. https://doi.org/10.48550/ARXIV.2010.11929

Fundel, F., Braun, D. A., & Gottwald, S. (2023). Automatic bat call classification using transformer networks. *Ecological Informatics*, *78*, 102288. https://doi.org/10.1016/j.ecoinf.2023.102288

Gong, Y., Chung, Y.-A., & Glass, J. (2021). Ast: Audio spectrogram transformer. https://doi.org/10.48550/ARXIV.2104.01778

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., & Zhou, Y. (2017). Deep learning scaling is predictable, empirically. https://doi.org/10.48550/ARXIV.1712.00409

Hu, J., Huang, W., Su, Y., Liu, Y., & Xiao, P. (2020). Batnet++: A robust deep learning-based predicting models for calls recognition. *2020 5th International Conference on Smart Grid and Electrical Automation (ICSGEA)*. https://doi.org/10.1109/icsgea51094.2020.00062

Iman, M., Arabnia, H. R., & Rasheed, K. (2023). A review of deep transfer learning and recent advancements. *Technologies*, *11*(2), 40. https://doi.org/10.3390/technologies11020040

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. https://doi.org/10.48550/ARXIV.1412.6980

Kobayashi, K., Masuda, K., Haga, C., Matsui, T., Fukui, D., & Machimura, T. (2021). Development of a species identification system of Japanese bats from echolocation calls using convolutional neural networks. *Ecological Informatics*, *62*, 101253. https://doi.org/10.1016/j.ecoinf.2021.101253

Koutini, K., Schlüter, J., Eghbal-zadeh, H., & Widmer, G. (2022). Efficient Training of Audio Transformers with Patchout. *Interspeech 2022*. https://doi.org/10.21437/interspeech.2022-227

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lehnert, L. S., Kramer-Schadt, S., Schönborn, S., Lindecke, O., Niermann, I., & Voigt, C. C. (2014). Wind Farm Facilities in Germany Kill Noctule Bats from Near and Far (J. M. Ratcliffe, Ed.). *PLoS ONE*, *9*(8), e103106. https://doi.org/10.1371/journal.pone.0103106

Mauson, K. (2022). *Künstliche neuronale Netze für die Bildklassifikation von Fledermausrufen*.

Mirzaei, G., Majid, M. W., Ross, J., Jamali, M. M., Gorsevski, P. V., Frizado, J. P., & Bingman, V. P. (2012). The BIO-acoustic feature extraction and classification of bat echolocation calls. *2012 IEEE International Conference on Electro/Information Technology*. https://doi.org/10.1109/eit.2012.6220700

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press. http://neuralnetworksanddeeplearning.com

Owusu-Adjei, M., Ben Hayfron-Acquah, J., Frimpong, T., & Abdul-Salaam, G. (2023). Imbalanced class distribution and performance evaluation metrics: A systematic review of prediction accuracy for determining model performance in healthcare systems (A. Reiner-Benaim, Ed.). *PLOS Digital Health*, *2*(11), e0000290. https://doi.org/10.1371/journal.pdig.0000290

Park, G., & Lee, S. (2020). Environmental noise classification using convolutional neural networks with input transform for hearing aids. *International Journal of Environmental Research and Public Health*, *17*(7), 2270. https://doi.org/10.3390/ijerph17072270

Parsons, S., & Jones, G. (2000). Acoustic identification of twelve species of echolocating bat by discriminant function analysis and artificial neural networks. *Journal of Experimental Biology*, *203*(17), 2641–2656. https://doi.org/10.1242/jeb.203.17.2641

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. https://doi.org/10.48550/ARXIV.1912.01703

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2012). Scikit-learn: Machine learning in python. https://doi.org/10.48550/ARXIV.1201.0490

Piczak, K. J. (2015). ESC: Dataset for Environmental Sound Classification. https://doi.org/10.7910/DVN/YDEPUT

Rainio, O., Teuho, J., & Klén, R. (2024). Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, *14*(1). https://doi.org/10.1038/s41598-024-56706-x

Rengasamy, D., Jafari, M., Rothwell, B., Chen, X., & Figueredo, G. P. (2020). Deep learning with dynamically weighted loss function for sensor-based prognostics and health management. *Sensors*, *20*(3), 723. https://doi.org/10.3390/s20030723

Roemer, C., Julien, J.-F., Ahoudji, P. P., Chassot, J.-M., Genta, M., Colombo, R., Botto, G., Negreira, C. A., Djossa, B. A., Ing, R. K., Hassanin, A., Rufray, V., Uriot, Q., Participants, V.-C., & Bas, Y. (2021). An automatic classifier of bat sonotypes around the world.

Methods in Ecology and Evolution, *12*(12), 2432–2444. https://doi.org/10.1111/2041-210x.13721

Roswag, M., Roswag, A., Roswag, M. S., Fietz, J., & Taefi, T. T. (2024). Advancing bat monitoring: Assessing the impact of unmanned aerial systems on bat activity [In review]. *Plos One*.

Ruiz, A. T., Equihua, J., Martínez, S., Robredo, E., Palm, G., & Schwenker, F. (2019). Detection of bat acoustics signals using voice activity detection techniques with random forests classification. In *Lecture notes in computer science* (pp. 253–261). Springer International Publishing. https://doi.org/10.1007/978-3-030-13469-3_30

Schwab, E., Pogrebnoj, S., Freund, M., Flossmann, F., Vogl, S., & Frommolt, K.-H. (2022). Automated bat call classification using deep convolutional neural networks. *Bioacoustics*, *32*(1), 1–16. https://doi.org/10.1080/09524622.2022.2050816

Sechidis, K., Tsoumakas, G., & Vlahavas, I. (2011). On the stratification of multi-label data. In *Lecture notes in computer science* (pp. 145–158). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-23808-6_10

Serizel, R., Bisot, V., Essid, S., & Richard, G. (2017, September). Acoustic features for environmental sound analysis. In *Computational analysis of sound scenes and events* (pp. 71–101). Springer International Publishing. https://doi.org/10.1007/978-3-319-63450-0_4

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*(1). https://doi.org/10.1186/s40537-019-0197-0

Smallwood, K. S., & Bell, D. A. (2020). Effects of wind turbine curtailment on bird and bat fatalities. *The Journal of Wildlife Management*, *84*(4), 685–696. https://doi.org/10.1002/jwmg.21844

Soyalp, G., Alar, A., Ozkanli, K., & Yildiz, B. (2021). Improving text classification with transformer. *2021 6th International Conference on Computer Science and Engineering (UBMK)*. https://doi.org/10.1109/ubmk52708.2021.9558906

Szymański, P., & Kajdanowicz, T. (2017). A scikit-based python environment for performing multi-label classification. https://doi.org/10.48550/ARXIV.1702.01460

Taefi, T. T., Roswag, M., & Peklar, G. (2024). Wingbeat Over Wind Turbines: Autonomous Drones for Acoustic Bat Detection in Operational Wind Farms. *2024 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*. https://doi.org/10.1109/acdsa59508.2024.10467290

Tarekegn, A. N., Giacobini, M., & Michalak, K. (2021). A review of methods for imbalanced multi-label classification. *Pattern Recognition*, *118*, 107965. https://doi.org/10.1016/j.patcog.2021.107965

Tsalera, E., Papadakis, A., & Samarakou, M. (2021). Comparison of Pre-Trained CNNs for Audio Classification Using Transfer Learning. *Journal of Sensor and Actuator Networks*, *10*(4), 72. https://doi.org/10.3390/jsan10040072

Tufféry, S. (2023). *Deep learning: From big data to artificial intelligence with R*. Wiley.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. https://arxiv.org/pdf/1706.03762.pdf

Vogelbacher, M., Bellafkir, H., Gottwald, J., Schneider, D., Muhling, M., & Freisleben, B. (2023). Deep Learning for Recognizing Bat Species and Bat Behavior in Audio Recordings. *2023 10th IEEE Swiss Conference on Data Science (SDS)*. https://doi.org/10.1109/sds57534.2023.00014

Voigt, C. C., Kaiser, K., Look, S., Scharnweber, K., & Scholz, C. (2022). Wind turbines without curtailment produce large numbers of bat fatalities throughout their lifetime: A call against ignorance and neglect. *Global Ecology and Conservation*, *37*, e02149. https://doi.org/10.1016/j.gecco.2022.e02149

Voigt, C. C., Russo, D., Runkel, V., & Goerlitz, H. R. (2021). Limitations of acoustic monitoring at wind turbines to evaluate fatality risk of bats. *Mammal Review*, *51*(4), 559–570. https://doi.org/10.1111/mam.12248

Yang, Y.-Y., Hira, M., Ni, Z., Chourdia, A., Astafurov, A., Chen, C., Yeh, C.-F., Puhrsch, C., Pollack, D., Genzel, D., Greenberg, D., Yang, E. Z., Lian, J., Mahadeokar, J., Hwang, J., Chen, J., Goldsborough, P., Roy, P., Narenthiran, S., … Shi, Y. (2021). Torchaudio: Building blocks for audio and speech processing. https://doi.org/10.48550/ARXIV.2110.15018

Zaman, K., Sah, M., Direkoglu, C., & Unoki, M. (2023). A Survey of Audio Classification Using Deep Learning. *IEEE Access*, *11*, 106620–106649. https://doi.org/10.1109/access.2023.3318015

Zamora-Gutierrez, V., Lopez-Gonzalez, C., MacSwiney Gonzalez, M. C., Fenton, B., Jones, G., Kalko, E. K. V., Puechmaille, S. J., Stathopoulos, V., & Jones, K. E. (2016). Acoustic identification of Mexican bats based on taxonomic and ecological constraints on call design (R. Freckleton, Ed.). *Methods in Ecology and Evolution*, *7*(9), 1082–1091. https://doi.org/10.1111/2041-210x.12556

Zualkernan, I., Judas, J., Mahbub, T., Bhagwagar, A., & Chand, P. (2020). A tiny cnn architecture for identifying bat species from echolocation calls. *2020 IEEE / ITU International Conference on Artificial Intelligence for Good (AI4G)*. https://doi.org/10.1109/ai4g50087.2020.9311084

Zualkernan, I., Judas, J., Mahbub, T., Bhagwagar, A., & Chand, P. (2021). An AIoT System for Bat Species Classification. *2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)*. https://doi.org/10.1109/iotais50849.2021.9359704

Appendix A.

Dataset

A.1. Species Mapping

Table A.1.: Mapping of species in pretrained models to species in Drones4Bats dataset (Aodha et al., 2022; Fundel et al., 2023; Roswag et al., 2024).

Species	**Drones4Bats dataset equivalent (Roswag et al., 2024)**	**BAT (Fundel et al., 2023)**	**BatDetect2 (Aodha et al., 2022)**
Barbastellus barbastellus	Myo		✗
Eptesicus serotinus	Nyc	✗	✗
Eptesicus nilssonii	Nyc	✗	
Miniopterus schreibersii	N/A	✗	
Myotis alcathoe	Myo		✗
Myotis bechsteinii	Myo		✗
Myotis brandtii	Myo	✗	✗
Myotis daubentonii	Myo	✗	✗
Myotis dasycneme	Myo	✗	
Myotis emarginatus	Myo	✗	
Myotis myotis	Myo	✗	
Myotis mystacinus	Myo	✗	✗
Myotis nattereri	Myo	✗	✗
Nyctalus leisleri	Nyc	✗	✗
Nyctalus noctula	Nyc	✗	✗

Continued on next page

Species	Drones4Bats dataset equivalent	BAT	BatDetect2
Pipistrellus kuhlii	Pip	✗	
Pipistrellus nathusii	Pip	✗	✗
Pipistrellus pipistrellus	Pip	✗	✗
Pipistrellus pygmaeus	Pip		✗
Plecotus auritus	Myo		✗
Plecotus austriacus	Myo		✗
Rhinolophus ferrumequinum	N/A	✗	✗
Rhinolophus hipposideros	N/A	✗	✗
Vespertilio murinus	Pip	✗	

Aodha et al., 2022 also had a class called bat, I classified it as noise in multilabel classification and as bat in binary classification.

A.2. Class Distributions

Table A.2.: Class distributions on different training set configurations and the validation set.

(a) Drone noise dataset distribution LP-RUS 0%

Label(-set)	**LPROS 0**		**LPROS 20**		**LPROS 40**	
	#	**IRLbl**	**#**	**IRLbl**	**#**	**IRLbl**
Pip	7867	1.00	7867	1.00	7867	1.00
Myo	868	9.06	1470	5.35	2073	3.79
Nyc	103	76.38	705	11.16	1308	6.01
Pip/Myo	899	8.75	1501	5.24	2104	3.74
Pip/Nyc	67	117.42	669	11.76	1272	6.18
Myo/Nyc	0	/	0	/	0	/
Pip/Myo/Nyc	18	437.06	620	12.69	1223	6.43
(Drone) Noise	5241	1.50	5241	1.50	5241	1.50
MeanIR		93.02		6.96		4.10
Density	0.24		0.30		0.34	

(b) Drone noise dataset distribution LP-RUS 40%.

Label(-set)	**LPROS 0**		**LPROS 20**		**LPROS 40**	
	#	**IRLbl**	#	**IRLbl**	#	**IRLbl**
Pip	4855	1.00	4855	1.00	4855	1.00
Myo	868	5.59	1229	3.95	1292	3.76
Nyc	103	47.14	464	10.46	967	5.02
Pip/Myo	899	5.40	1260	3.85	1292	3.76
Pip/Nyc	67	72.46	428	11.34	931	5.21
Myo/Nyc	0	/	0	/	0	/
Pip/Myo/Nyc	18	269.72	379	12.81	882	5.50
(Drone) Noise	2229	2.18	2229	2.18	2229	2.18
MeanIR		57.64		6.51		3.78
Density	0.29		0.34		0.38	

(c) Augmented dataset distribution LP-RUS 60%.

Label(-set)	**LPROS 0**		**LPROS 20**		**LPROS 40**	
	#	**IRLbl**	#	**IRLbl**	#	**IRLbl**
Pip	7966	1.00	7966	1.00	7966	1.00
Myo	2936	2.71	2936	2.68	2936	2.68
Nyc	757	10.52	1883	4.18	2816	2.79
Pip/Myo	3515	2.27	3515	2.24	3515	2.24
Pip/Nyc	534	14.92	1660	4.74	2816	2.79
Myo/Nyc	59	135.02	1185	6.64	2366	3.33
Pip/Myo/Nyc	95	83.85	1221	6.44	2401	3.28
(Drone) Noise	6662	1.20	6662	1.18	6662	1.18
MeanIR		31.44		3.64		2.41
Density	0.30		0.36		0.41	

(d) Validationset distribibution (the testset distribution is almost the same).

Label(-set)	#
Pip	2622
Myo	288
Nyc	34
Pip/Myo	301
Pip/Nyc	23
Myo/Nyc	1
Pip/Myo/Nyc	5
(Drone) Noise	1747

Appendix B.

Results

B.1. Results Pretrained Models on Full Dataset

Table B.1.: Results for pretrained models with the full dataset with drone noise.

Model	Metric	Pip	Myo	Nyc	Micro	Macro	Binary Bat
BatDetect2	F1-score	0.946	0.789	0.515	0.921	0.750	0.958
	Precision	0.908	0.900	0.380	0.888	0.729	0.934
	Recall	0.987	0.703	0.799	0.937	0.830	0.982
BAT	F1-score	0.908	0.453	0.438	0.831	0.600	0.916
	Precision	0.947	0.541	0.461	0.884	0.650	0.903
	Recall	0.872	0.390	0.417	0.784	0.560	0.929

The results in table B.1 only slightly differ from those on the testset (table 4.1).

B.2. Results on All Resampling Combinations

Table B.2.: Results for BAT for all resampling-augmentation combinations.

Set/LP-RUS/LP-ROS	Metric	Pip	Myo	Nyc	Micro	Macro	Binary Bat
dro/0/0	F1-score	0.902	0.0	0.0	0.804	0.301	0.860
	Precision	0.984	0.0	0.0	**0.984**	0.328	0.995
	Recall	0.832	0.0	0.0	0.680	0.277	0.758
dro/0/20	F1-score	0.903	0.430	0.252	0.834	0.528	0.889
	Precision	0.984	**0.954**	0.292	0.970	0.743	0.993
	Recall	0.835	0.277	0.222	0.732	0.445	0.805
dro/0/40	F1-score	0.907	0.450	**0.327**	0.838	**0.561**	0.896
	Precision	0.981	0.941	0.178	0.956	0.733	0.991
	Recall	0.843	0.296	0.327	0.745	0.512	0.817
dro/40/0	F1-score	0.912	0.191	0.0	0.826	0.368	0.884
	Precision	0.982	0.865	0.0	0.978	0.616	0.991
	Recall	0.852	0.108	0.0	0.714	0.320	0.798
dro/40/20	F1-score	**0.917**	0.440	0.280	**0.849**	0.546	**0.905**
	Precision	0.977	0.940	0.341	0.965	**0.753**	0.988
	Recall	**0.864**	0.287	0.238	**0.758**	0.463	**0.835**
dro/40/40	F1-score	0.903	**0.462**	0.276	0.831	0.547	0.888
	Precision	0.984	0.929	0.212	0.948	0.739	0.985
	Recall	0.834	**0.308**	**0.397**	0.739	**0.513**	0.809
aug/60/0	F1-score	0.839	0.385	0.111	0.768	0.445	0.822
	Precision	**0.991**	0.688	**0.444**	0.960	0.708	**0.996**
	Recall	0.727	0.267	0.063	0.639	0.352	0.700
aug/60/20	F1-score	0.853	0.316	0.212	0.770	0.460	0.834
	Precision	**0.991**	0.771	0.175	0.945	0.646	0.994
	Recall	0.749	0.198	0.212	0.650	0.406	0.719
aug/60/40	F1-score	0.863	0.276	0.174	0.764	0.438	0.844
	Precision	0.989	0.703	0.114	0.906	0.602	0.986
	Recall	0.766	0.171	0.365	0.661	0.434	0.737

Table B.3.: Results for PaSST for all resampling-augmentation combinations after 5 epochs. For results after more epochs refer to section 4.2.2.

Set/LP-RUS/LP-ROS	Metric	Pip	Myo	Nyc	Micro	Macro	Binary Bat
dro/0/0	F1-score	0.932	0.614	0.0	0.883	0.515	0.938
	Precision	0.922	0.919	0.0	0.921	0.614	0.952
	Recall	0.943	0.461	0.0	0.847	0.468	0.925
dro/0/20	F1-score	0.934	0.680	0.528	0.891	0.714	**0.945**
	Precision	0.922	0.930	0.409	0.891	0.753	0.952
	Recall	0.948	0.536	0.746	0.876	0.743	0.938
dro/0/40	F1-score	0.934	**0.731**	0.530	**0.894**	**0.732**	0.943
	Precision	**0.934**	0.857	0.407	0.906	0.733	**0.955**
	Recall	0.935	0.637	0.762	0.883	0.778	0.930
dro/40/0	F1-score	0.927	0.713	0.0	0.887	0.547	0.942
	Precision	0.897	0.787	0.0	0.882	0.561	0.931
	Recall	**0.960**	0.652	0.0	**0.892**	0.537	**0.953**
dro/40/20	F1-score	0.933	0.685	**0.568**	0.891	0.728	0.942
	Precision	0.919	0.879	**0.494**	0.904	0.764	0.944
	Recall	0.947	0.561	0.667	0.879	0.725	0.939
dro/40/40	F1-score	0.927	0.730	0.370	0.874	0.676	0.936
	Precision	0.926	0.766	0.236	0.860	0.643	0.945
	Recall	0.927	**0.697**	**0.857**	0.888	**0.827**	0.928
aug/60/0	F1-score	**0.940**	0.443	0.0	0.874	0.461	0.930
	Precision	0.932	**0.945**	0.0	**0.932**	0.626	**0.955**
	Recall	0.948	0.443	0.0	0.823	0.412	0.906
aug/60/20	F1-score	0.928	0.564	0.427	0.876	0.640	0.929
	Precision	0.899	0.938	0.463	0.895	**0.767**	0.924
	Recall	0.958	0.403	0.397	0.857	0.586	0.933
aug/60/40	F1-score	0.934	0.537	0.428	0.873	0.633	0.934
	Precision	0.925	0.915	0.323	0.902	0.721	0.948
	Recall	0.944	0.380	0.635	0.846	0.653	0.921

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit mit dem Titel

Bat Call Classification in Audio Recordings with Drone Noise Using Deep Learning

selbstständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.
Neben den in der Arbeit genannten Quellen habe ich Large Language Models verwendet (DeepL Write beta[1] für sprachliche Verbesserungen, ChatGPT 3.5[2] für das Verständnis von Zusammenhängen und GitHub Copilot[3] zur Erstellung und Korrektur von Python Code).

Hamburg, 23. Mai 2024

[1] https://www.deepl.com/write
[2] https://chat.openai.com/
[3] https://docs.github.com/en/copilot