Monitoring Bat Activity in Audio with Self-Supervised Learning

Jingyuan Wang

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2023

Abstract

In this study, we investigate the effectiveness of self-supervised learning techniques on bat echolocation call classification using a dataset contains calls from 17 bat species breeding in the UK. Self-supervised learning aims to learn useful features and representations from raw data without the need for labelled data. We implement two self-supervised learning architectures, SimCLR and autoencoder, and train them on the bat echolocation call spectrogram database we produced from the given dataset. By comparing their performance with a ResNet-18 supervised baseline, we find that the SimCLR model achieves a test accuracy of 75.49% using only 50% of the labelled dataset, which is competitive with the baseline model's performance of 75.92% using 100% of the labelled dataset. Moreover, we observe that the SimCLR model can learn visual representations that are competitive to those learnt by the supervised baseline without using any labelled data at the genus level, where the linear evaluation accuracy of the pretrained SimCLR model is only 2% different from the accuracy achieved by the supervised baseline trained end-to-end. Furthermore, we identify that the primary reason for misclassification in species discrimination tasks is the low inter-class differences between species, and self-supervised pretrained models have demonstrated improved performance in addressing this challenge. This report shows the potential of self-supervised learning method in the task of bat echolocation call classification and provides valuable insights for future research in this area.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Jingyuan Wang*)

Acknowledgements

First and foremost, I would like to thank my supervisor, Oisin Mac Aodha, for willing to take me under the unforeseen circumstances I was in. Thank you for providing your guidance and expertise, and most importantly, for helping me discover and nurture my interest in the field of computer vision.

Secondly, I also would like to thank my friends who made this city my home, providing me with continuous support and proofreading my dissertation report.

Lastly, it would be impossible to be here and complete this degree without the constant encouragement and support from my family. To my parents who have given me constant support and love throughout my degree. To my dogs, who never fail to bring a smile to my face.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

There is a critical need for robust measurements to monitor environmental change. A bioindicator, which is a living organism used to evaluate the health of an ecosystem, stands out from other candidates to monitor the ecosystem's health since it's more sensitive, economical and determined [29]. Bats have been identified as an excellent bioindicator as they are globally distributed, taxonomic stable, and particularly sensitive to habitat environmental change [19]. Thus, monitoring bat populations and species dynamics can be a powerful tool to monitor environmental change.

However, bats are nocturnal, elusive, and very sensitive to disturbance, which makes visually monitoring bats impossible [36]. As a result, audio-based monitoring has become a powerful tool for monitoring bat populations and species distribution [2, 38]. Acoustic monitoring of bats is an approach for estimating bat populations and distributions by analysing the echolocation calls emitted by them. Echolocation calls are the ultrasound signals that bats emit to navigate, acquire prey, and communicate [39]. This monitoring method stands out from other methods of bat monitoring as it offers a non-invasive way to monitor bats at a low cost. Furthermore, multiple large target areas can be monitored intensively at the same time using automated acoustic sensing devices [1, 11].

However, bat calls exhibit large variation [35], and many species have small inter-class differences [21]. These all make manually labelling bat echolocation calls challenging, and normally require professionals with years of training to label the echolocation calls individually. Figure 1.1 shows an example spectrogram containing bat echolocation calls belonging to two species with small inter-class differences.

The recent advancements in deep learning revolutionised the detection and classification of bat echolocation calls. Deep learning methods show the potential to build up a fully automated pipeline for bat echolocation call classification, which is particularly helpful in large-scale ecological studies. The contributions from researchers have demonstrated the capability of supervised machine-learning methods to detect and classify the bat echolocation calls from raw audio [26, 27, 24, 45]. For example, Tabak et al. [45]

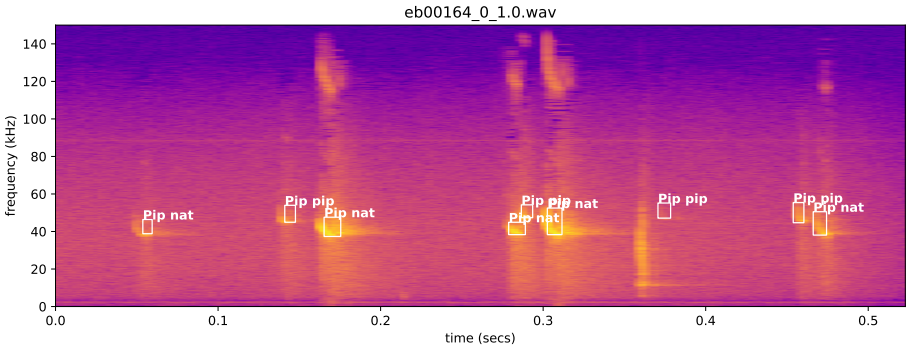

Figure 1.1: An example of a spectrogram containing bat echolocation calls, 'Pip nat' and 'Pip pip' in the graph are short for bat species names Pipistrellus nathusii and Pipistrellus pipistrellus respectively.

applied a ResNet-18 architecture to the echolocation calls of the 10 bat species in the United States and achieved a classification accuracy of 90%.

However, existing works that apply deep learning to bat echolocation call classification heavily rely on the availability of a sufficiently large, high-quality labelled dataset. Establishing such a dataset can be challenging, as labelling bat echolocation call data requires a high level of expertise and is time-consuming. Furthermore, the bat species distribution in distinct areas differ largely, such that even the same species distributed in different geographical environments can have different echolocation call features [18, 33]. This makes the model trained on the existing dataset hard to generalise on the bat population from a different geographical location.

Therefore, we wish to find a way to build an efficient deep learning-based approach that does not rely on large-scale, high-quality labelled datasets. Self-supervised learning stands out for its ability to learn meaningful features and representations from unlabelled data. Self-supervised learning is a machine learning method where models learn meaningful representations using unlabelled data to generate supervision for target tasks. The current state-of-the-art techniques in self-supervised learning are capable of producing visual representations that are competitive to those generated by fully supervised models in many downstream tasks [9]. Despite its potential, there are limited studies on the application of self-supervised techniques in bioacoustics classification, and to the best of our knowledge, no research has been conducted on applying self-supervised learning techniques in the task of bat echolocation call classification. In this report, we aim to explore the utility and effectiveness of self-supervised learning techniques in bat echolocation call classification, bridging the gap in the current research landscape.

1.2 Goal of this project

In this project, our primary objective is to develop an efficient deep learning-based approach for classifying bat echolocation events of interest in audio files, using limited species-level supervision, while maintaining performance comparable to the fully

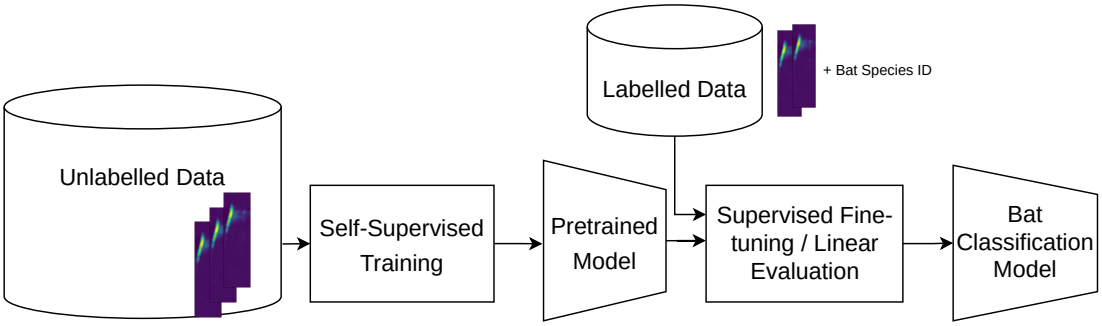

Figure 1.2: A high-level overview of the training process used to obtain the final bat echolocation call classification model. We developed the final bat echolocation call classification model by first pretraining the model using a self-supervised method with unlabelled data, followed by finetuning or linear evaluation of the model.

supervised baseline.

Through our experiments, we aim to address the following research questions:

- Can self-supervised pretraining improve the performance of the model when utilising all available audio and labels?
- Can self-supervised pretraining enable a model to achieve competitive performance using a smaller amount of labelled data compared to a baseline model that utilises all available labelled data? If so, by what extent can the amount of data be reduced while maintaining competitive performance?
- Can self-supervised pretraining help the model learn to produce meaningful and discriminative visual representations that are competitive to those learnt by fully supervised model?
- What insights can be derived from the experimental results regarding the challenges faced when performing bat echolocation call classification, and how can these insights inform future research in this area?

1.3 Achievements

In this report, we have achieved the stated goal by applying the SimCLR [6] and autoencoder self-supervised approaches and comparing their performance with a fully supervised, end-to-end trained, ResNet-18 baseline. The SimCLR model achieves an average testing accuracy of 75.49% using 50% of the labelled dataset, which is competitive to the performance achieved by the baseline model (75.92%) while using 100% of the labelled dataset. By linearly evaluating and visualising the representation learnt by supervised baseline and self-supervised models, we found both self-supervised models have learnt a meaningful representation both at the species and genus level without using any labelled data. Especially for the SimCLR model, which learnt competitive representations to those learnt by the supervised baseline at the genus level, achieving a linear evaluation accuracy which is only 2% different from the accuracy

achieved by the supervised baseline. An overview of our main pipeline can be seen in Figure 1.2.

In order to achieve the main targeted objectives, the following steps have been undertaken:

- Production of a spectrogram database containing 32,651 bat echolocation calls from the UK bat echolocation call database constructed by Mac Aodha et al. [27].
- Creation of the first self-supervised machine learning pipeline for training and evaluating bat echolocation call classification models, from raw audio data to a complete and efficient machine learning model.
- Investigation of the challenges faced when applying deep learning methods to bat echolocation call classification, providing insights for future research.
- Assessment of the visual representation quality learnt by the two self-supervised models, demonstrates the potential of self-supervised learning approaches for this task.

Our results showcase the viability of self-supervised learning approaches for bat echolocation call classification and provide valuable insights for future research in this area.

Chapter 2

Background

In this chapter, we discuss the technical background and concepts which this report is based on. We also cover the related work in this field, which demonstrate how our contributions fit within the broader research context.

2.1 Bioindicator

A bioindicator, also known as biological indicator, is a living organism that can be used to screen the health of an ecosystem and its subsequent effect on human society [29]. Common bioindicators includes birds, lichens, and amphibians. Compared to the traditional way of screening the ecosystem, using bioindicator to screen the environment have advantages including the biological impacts being determined, its easier to be monitored and it can diagnose the ecosystem change in the early stage.

Bioindicators can be used to monitor environmental change as they are normally sensitive to habitat environmental changes. As a result, the biodiversity of the environment will be affected, which can be monitored by population and species distribution fluctuations [15].

Bats have shown great potential to be excellent bioindicators with the reason of they are extremely sensitive to environmental changes [19]. Moreover, bats inhabit all six continents except Antarctica, with a total of over 1,400 species overall, which is more than one-fourth of the total mammal species [42]. This implies that bats can be used as a general bioindicator around most places on Earth. Furthermore, with the advancement and wide application of audio based monitoring, collecting bat calls has become extremely easy.

2.1.1 Biological Taxonomy

Taxonomy is the scientific theory and practice of classifying groups of biological organisms based on their shared characteristics [43]. It follows a hierarchical structure, with the most basic unit of classification being the species. Species are then further grouped together to form higher levels of classification, known as taxonomic ranks.

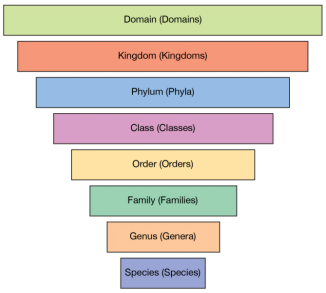

Figure 2.1: Taxonomic ranks in descending order. Image taken from [5]

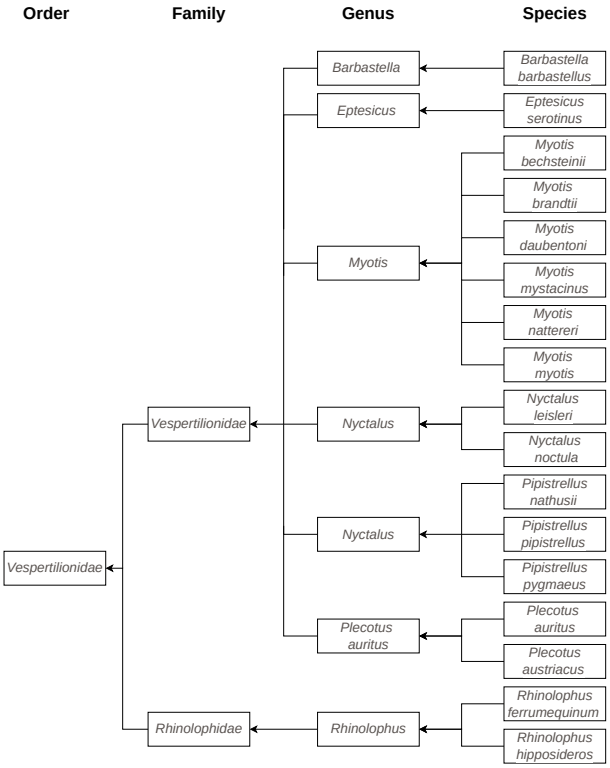

Figure 2.2: The taxonomy tree of the 17 bat species we investigated in the report.

As the level of classification increases, organisms within the same group share fewer common characteristics. A detailed representation of taxonomic ranks are shown in Figure 2.1.

In this report, we aim to classify 17 bat species which are known to be breeding in the UK at the species level, these 17 bat species can be further grouped into 7 genera, 2 families and 1 order. A specific taxonomy tree of the 17 bat species is shown in Figure 2.2.

2.2 Representing Audio

2.2.1 Fourier Transformation

Fourier Transform is a mathematical transformation that transforms a function of time, into a function of frequency [12]. In the audio processing domain, it decomposes a complex waveform into individual frequency components, each in time segment of the audio wavefront for analysis. The Fourier transformation of a function $f(t)$ is defined below.

$$F(k) = \int_{-\infty}^{\infty} f(t)e^{-ikt}dt \tag{2.1}$$

The output value of Fourier transformation will be plotted as a spectrogram, and the bat echolocation calls classification will be carried out on the spectrogram of each bat call.

2.2.2 Spectrograms

A spectrogram is a visual representation of an audio signal that displays the variation of the magnitude of each frequency band over time. An example of the spectogram with the audio input waveform is shown in Figure 2.3, the x-axis is the time, the y-axis is the frequency, and the colour shows the magnitude of the signal at each frequency band and time segment.

The spectrogram is generated from the raw audio by first splitting the audio into numbers of small and overlapping segments. After that, a Fourier transformation is calculated for each segment. Then, the absolute value of the output of the Fourier transformation is taken, which results in a series of frequency spectra. After plotting each spectrum following the time order on the x-axis, the resulting plot is the spectrogram of the input raw audio. The implementation details of this process is shown in section 4.2.

The classification of the bat echolocation calls is performed on the spectrogram generated from the audio instead of the raw audio itself because: First, the spectrogram visualises the variation of the magnitude of each frequency band over time, which makes analysing the characteristics of the audio easier. This is particularly essential for bat echolocation call classification as the bat echolocation call frequency is closely related to the species it belongs [18]. Second, the spectrogram could help with noise reduction by focusing the frequency bands related to the classification tasks, and the constant noise can be easily removed by deducting the mean value from each frequency band.

2.3 Related Work

As stated in the motivation section, there is only a limited amount of research carried out about self-supervised bioacoustic monitoring, and no research has been carried out about self-supervised bat echolocation call classification. Therefore, in this section, we talk about the previous works carried out in machine learning-based bat acoustic

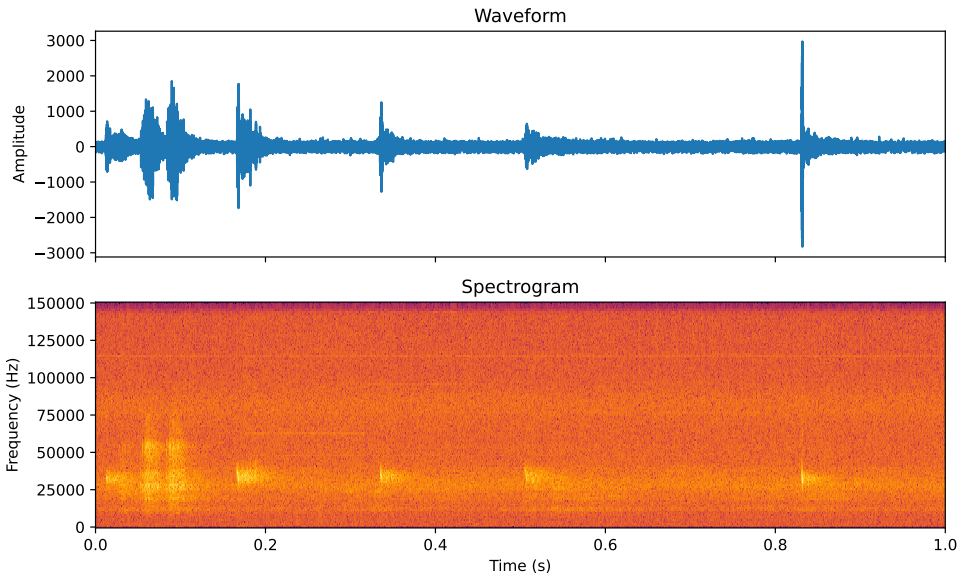

Figure 2.3: The waveform of an example audio and the spectrogram computed.

monitoring, as well as self-supervised learning in bioacoustic monitoring. The aim of this section is to discuss the limitations and the breakthrough of the previous related studies.

2.3.1 Machine Learning in Bat Echolocation Call Classification

Recent advancements in bioacoustic sensing techniques and machine learning resulted in various attempts to establish a fully automatic pipeline for bioacoustic monitoring. In the supervised approach, there are two main directions. The first one is applying machine learning-based approaches to the acoustic features extracted manually from the audio recording and classifying the species present in the audio recording. The second one is applying deep learning-based approaches directly to the audio recording.

There is a rich history of applying machine learning methods to manually extracted features from the bat echolocation call recordings [30, 47, 34], These researches are all leveraging the fact that bat echolocation calls have less diversity as their primary function is objective locating, and the difference in extracted features like highest frequencies and duration, which makes the identification of most spices in a community possible [20, 34]. The extracted features in the previously mentioned work are manually extracted discriminative features related to the frequency-based and time-based characteristics of the bat echolocation calls, such as duration, highest frequency and sonotypes. Compared with deep learning feature, the manually extracted feature method usually require less amount of data, and it's a quicker and easier approach to distinguish the main bat species.

However, the structure of bat echolocation calls varies largely on various factors, including sex, age and habitat, making the manually extracted feature methods hard to

predict bat species with high accuracy [20]. Furthermore, the presence of background noise and other vocalising species further increases the challenge of manually extracted feature methods. The deep learning-based methods aim to solve this problem by learning meaningful representations that could be used to discriminate calls from each species directly to the raw audio recording. The recent applications of deep learning-based approaches to the bat bioacoustic classification all yield promising results, which outperform traditional feature extraction classification methods on many datasets [8, 24, 27].

Kobayashi et al. [24] applied CNN based MobileNet-V1 model to a dataset containing 30 species which is collected from Japan and South Korea from 1999 to 2019. They applied short-time Fourier Transformation to the bat echolocation calls to generate spectrograms. Next, they apply an automatic algorithm to detect peaks in the recording and trim out spectrograms with 20ms long centred on the peak point detected. i.e., the peak point is in the middle of the trim-out 20ms spectrogram. After manually inspecting all the result spectrograms and filtering wrongly detected noises, the resulting 20ms spectrograms will be the input of the CNN-based model. The model successfully outperformed the feature extraction baseline from the previous studies and achieved an accuracy of 98.1%.

Mac Aodha et al. [27] developed CNN-based U-Net-Style architecture with skip connections and a transformer-based self-attention layer in the middle. The model is tested and shown promising results on 5 datasets collected from 4 geographically distinct areas. The model also outperformed the feature extraction baseline on the UK dataset. Be noted, compared with previous works, the model proposed by Mac Aodha et al. is able to take longer audio input (duration of 1 second long) effectively because of the existence of the self-attention layer.

However, all these works heavily rely on the availability of a large-scale, fully labelled dataset, which makes it challenging to apply to the area where datasets with similar quality are not established.

2.3.2 Self-supervised Learning

Self-supervised learning is a subfield of unsupervised learning in which the model learns to produce meaningful representations by solving proxy tasks created by the self-supervised learning method. Solving these proxy tasks does not require any human-labelled examples, as the model leverages the inherent structure of the input data to create its own supervision signals.

Self-supervised learning has emerged as a potent pretraining technique for classification tasks. Current state-of-the-art self-supervised methods can generate competitive representations compared to their supervised counterparts in various downstream tasks [10]. In many general classification tasks, the self-supervised learning method outperforms the supervised counterpart such as CIFAR100, ImageNet and places [6, 7, 13]. Self-supervised techniques are applied to classification tasks through a two-stage process. In the first stage, the model utilises a large unlabelled dataset. Self-supervised learning creates supervision signals for proxy tasks using unlabelled data, enabling the model

to capture meaningful features or representations. In the second stage, the pretrained model is fine-tuned on a smaller labelled dataset specific to the classification task. The representation learnt during the self-supervised pretraining stage is leveraged at this point. Consequently, the model is adapted and optimized for the target classification task, allowing it to effectively perform the task using the meaningful features extracted during the self-supervised learning phase. The self-supervised technique helps the model utilise a large amount of unlabelled data, which supervised tasks cannot. This usually results in an improvement in classification task performance with limited supervision, making self-supervised learning a powerful approach in scenarios where labelled data is expensive to obtain. Though self-supervised learning has been proven to be effective and sometimes outperforms the fully supervised models, there are only limited applications of self-supervised learning in bioacoustic monitoring, which mainly focus on the detection of bioacoustic events.

Bermant et al. [4] applied self-supervised learning in detecting sperm whale coda clicks. They trained the network in a self-supervised manner by feeding the model with clips of audio created by the sliding windows approach with no overlapping and train the model to maximise the agreement between adjacent window data and minimise the agreement between randomly sampled non-adjacent window data. Next, they applied a peak-finding algorithm to the output of the trained model. The model successfully outperformed the baseline energy amplitude threshold detector.

Although there have not been any applications of self-supervised learning in bioacoustic classification, there are several successful applications in related fields, such as environmental sound classification, that can provide valuable insights and inspiration for our research.

Tripathi et al. [46] applied the self-supervised learning technique to environmental sound classification. They chose ResNet-18 to perform the self-supervised learning. They pretrained the model to recognise the type of data augmentation they applied to the signal before feeding it to the network. The model successfully outperformed both CNN-based and feature extraction baselines.

Although self-supervised learning has demonstrated effectiveness in environmental sound classification, the task of bat echolocation call classification presents a unique set of challenges. Unlike environmental sounds, which typically exhibit higher inter-class differences, bat echolocation calls often have lower inter-class variations, making the classification task more challenging. Therefore, the effectiveness of self-supervised learning in the context of bat echolocation call classification remains to be investigated.

Chapter 3

Methodology

In this chapter, we discuss the methodologies used in the experiments, including model architectures, loss functions we applied and a list of data augmentation we applied.

3.1 Supervised Baseline Model

In this section, we outline the architecture and implementation details of the baseline models we choose in this report: ResNet-18.

3.1.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) is a special type of Artificial Neural Network (ANN). Compared with normal ANNs, CNN uses a special type of layer called "convolutional layers", which apply convolution operations on input data. In the convolutional layers, a weight matrix, also known as the kernel, strides through the layer input and performs the convolution operation to produce feature maps. Be noted, the weight matrix will not change while striding through the layer input, which means the same weights are applied at different locations using convolution. Normally, the average pooling and nonlinear activation will be applied after the convolution operation.

Through convolution, the network learns to extract patterns and features from the input data such as edges and shapes. As more and more convolutional layers have been passed, the features that are extracted by the convolutional layers will be more specific and higher level, such as face or hands. Finally, the Neural Network could do classification or detection tasks based on the features extracted from convolutional layers.

CNN is widely used in image-processing tasks mainly because the convolutional layers in the CNN could reduce high-dimension images to extract features of the image while not losing any information [40]. It address the challenges of dealing with massive trainable parameters and computational complexity, making them efficient for image recognition and classification tasks.

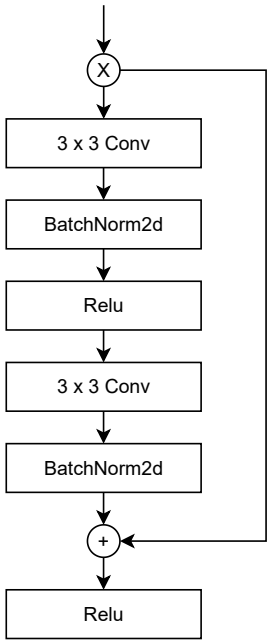

Figure 3.1: A detailed architecture of a ResNet block.

3.1.2 ResNet

ResNet (Residual Networks) is a special type of convolutional neural network proposed by He et al. [14] to solve the vanishing gradient problem in deep convolutional neural networks. The ResNet architecture is formed by blocks of several convolutional layers, an example of a ResNet block is shown in the Figure 3.1. The ResNet solve the vanishing gradient problem by adding a residual connection for each block which adds the input of the block to its output. This effectively prevents the gradient from getting too small, as it is really easy for the block to learn identity mapping since the input is directly added to the output.

In this report, we specifically implement a ResNet-18 architecture and refer to it as the baseline model. Additionally, we use ResNet-18 as the backbone for implementing the two self-supervised models explored in this report to make them comparable with the baseline. ResNet-18 is a specific type of ResNet that include 18 layers in total. It is chosen as the baseline model because ResNet architectures are commonly used in self-supervised tasks. ResNet-18 is selected among all ResNet networks because of the objective of building an efficient deep learning model and the constraints of computational resources.

The main body of the ResNet-18 neural network is a 17-layer CNN, which consists of 8 residual blocks with a single CNN layer at the beginning. A complete architecture of ResNet-18 is shown in Figure 3.2. Following the 17-layer CNN is a fully connected layer to output the classification probabilities for each bat species. Within each residual block, each convolutional layer is followed by a 2D batch normalisation layer [17] with a ReLU layer, and a residual connection adds the input to the output of the second batch normalisation layer to prevent the gradient in the network from getting too small. A detailed architecture of the residual block is shown in Figure 3.1. All convolutional

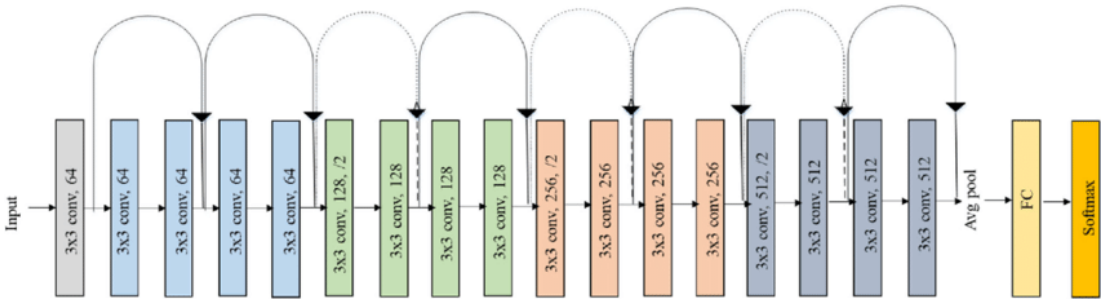

Figure 3.2: A detailed description of the ResNet-18 architecture. Image taken from [32].

layers in the ResNet-18 architecture have a kernel size of 3×3, and the number of channels for the initial convolutional layers starts with 64. The number of channels doubles every 4 layers, with the output channels of the 17 convolutional layers totaling 512. Follows the convolutional layers, an average pooling and a fully connected layer map the extracted features to the probabilities of the input data belongs to each class. The final softmax function normalise these probabilities so that their sum equals 1 across all classes. In our implementation, the output of the entire ResNet-18 network is a one dimensional vector with length 18, where each entry represent the probability of the input data belongs to that class.

The network is trained with the objective to minimise the cross entropy loss. The cross entropy loss shows the difference between true probability distribution and output probability distribution of the input data, the cross entropy loss of data X is calculated as follows:

$$\text{Loss} = -\sum_{k=1}^{K} y_k \log P(Y = k|X) \tag{3.1}$$

where K represent the total number of classes. The indicator variable y_k is set to 1 if data X belongs to class k and 0 otherwise, $P(Y = k|X)$ is the output probability of data X belongs to class K.

3.2 Self-Supervised Models

In this section, we outline the architecture and implementation details of the two self-supervised models investigated in this report: SimCLR and the autoencoder.

3.2.1 SimCLR

SimCLR (Simple Framework for Contrastive Learning of Visual Representations) is a contrastive learning framework first proposed in 2020 by Chen et al. [6]. The SimCLR model learns to produce meaningful visual representations by maximising the agreement between different augmented versions of the same image. As shown in Figure 3.3, SimCLR consists of four components: a data augmentation scheme T, a neural network encoder f, a small neural network projection head g and a contrastive loss function L.

The data augmentation scheme T generates two different augmented views, x_i and x_j, for every input data x. During training, x_i and x_j are considered as a positive pair, while x_i or x_j with all other augmented views generated from all other data in the same training batch are considered negative pairs. In our implementation, 7 data augmentations are applied to generate different augmented views, but we assigned a 50% probability for each to be applied, thereby enhancing randomness and diversity during training. The *torchvision.transforms* and *torchaudio.transforms* packages are used for all data augmentations. The list of augmentations applied is described in section 3.4.

The encoder f is a neural network that extracts a representation h_i from the augmented data view x_i. The SimCLR architecture allows flexibility in the design of the encoder network. In our implementation, we employed a ResNet-18 architecture as the encoder network f due to its widespread use in self-supervised tasks and our goal of constructing an efficient network. Moreover, we aimed to make the SimCLR model comparable to the ResNet-18 baseline model.

The projection head g is a small fully connected neural network which projects the visual representation h_i extracted by encoder f to the space where the contrastive loss is applied. The projection of the visual representation h_i is defined as z_i, where $z_i = g(h_i)$. The original design of the projection head proposed in [6] consists of two fully connected layers with a ReLU layer in between. However, we adopted the approach suggested in [7] which involves increasing the width and depth of the projection head, as adding another fully connected layer improved the performance of ResNet-based SimCLR models by 5% on ImageNet. Consequently, we implemented a projection head comprising three fully connected layers, with a ReLU layer between each pair of dense layers. In terms of width, aside from the first dense layer with an input and output size of 512, the remaining two dense layers have an input size that is double their output size.

For the contrastive loss function L, follows [6] and [7], we employed the NT-Xent (the normalised temperature-scaled cross-entropy loss) as follows.

$$\text{sim}(A,B) = \frac{A^\top B}{||A||||B||} \tag{3.2}$$

$$L_{i,j} = -log\frac{\exp(\text{sim}(z_i,z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]}\exp(\text{sim}(z_i,z_k)/\tau)} \tag{3.3}$$

Where $\mathbb{1}_{[k\neq i]} \in \{0,1\}$ returns 1, iff $k \neq i$, z_i and z_j is the projected visual representation, and τ indicate the temperature constant.

The NT-Xent loss awards high similarity between positive pairs (the two augmented views generated from the same data) in the batch and penalizes high similarities between negative pairs (all other augmented views generated from different data in the batch). Consequently, the positive pairs are attracted, and the negative pairs are repelled.

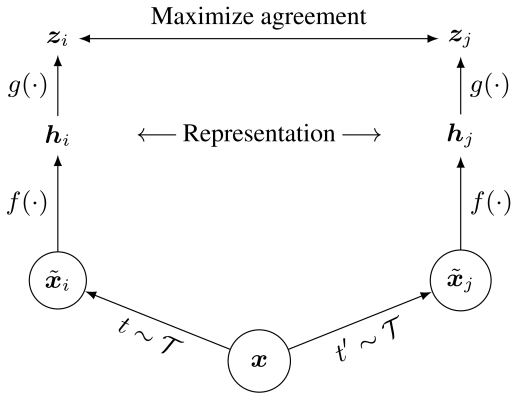

Figure 3.3: An overview of the SimCLR framework. Image taken from [6].

3.2.2 Autoencoders

Autoencoder is another popular type of self-supervised representation learner. The autoencoder model learns to produce meaningful visual representations by minimising the difference between input data y_i and reconstructed data $\hat{y}_i$. An autoencoder has three major components, a neural network-based encoder f, a neural network-based decoder g and a loss function L. An overview of the autoencoder architecture is shown in Figure 3.4.

The neural network-based encoder f extracts a visual representation h_i from the data y_i provided. In our implementation, we adopt the commonly used ResNet, more specifically we selected a ResNet-18 architecture due to the limitation of computation resources and the aim of making the autoencoder model comparable to the SimCLR model and baseline model.

The neural network-based decoder g is employed to project the representation h_i back into the data space. To ensure that the encoder and decoder have a similar capacity for learning to produce representations and reconstructing data, our decoder is designed to mirror the architecture of the encoder, which is a mirrored ResNet-18. The mirrored ResNet-18 decoder comprises ResNet blocks that consist of deconvolutional layers, batch normalisation, ReLU activation functions, and residual connections. The ResNet blocks are arranged in reverse order compared to the ResNet-18 encoder. This means the decoder starts with the final layer of the encoder and ends with the encoder's first layer. The decoder operates in reverse compared to the encoder, upscaling the lower-dimensional representation back to the original size. Similar to the encoder, each deconvolutional layer in the decoder has a kernel size of 3×3. This design choice ensures that both the encoder and decoder can effectively learn and process the data in a symmetrical manner, helping the autoencoder learn to produce meaningful representations.

The loss function L compares the difference between data y_i and reconstructed data $\hat{y}_i$. We applied a commonly used mean squared error shown as follows.

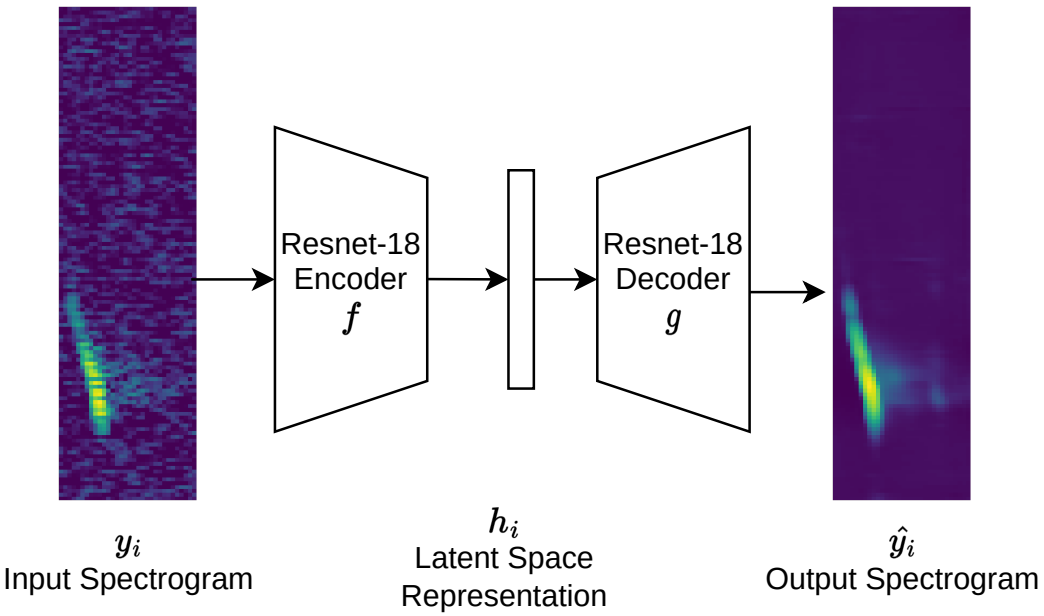

Figure 3.4: An overview of the autoencoder framework.

$$\text{MSE}(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.4}$$

3.3 Multinomial Logistic Regression

In this report, the multinomial logistic regression classifier is employed to linearly evaluate the visual representations extracted by self-supervised models. This approach is preferred over the commonly used attaching a linear layer at the end of the network for linear evaluation because multinomial logistic regression is convex, which can produce more stable and optimal results.

Multinomial logistic regression is a classifier that generalises the logistic regression to classification problems with more than two classes. Multinomial logistic regression maps the input data to a set of probabilities that the data belongs to each class using the softmax function. The probability of the data X_i belonging to class k when applying a softmax function is shown below:

$$P(Y = k|X_i) = \frac{e^{(b_{k0}+b_{k1}X_{i1}+b_{k2}X_{i2}+\cdots+b_{kn}X_{in})}}{\sum_{j=1}^{K} e^{(b_{j0}+b_{j1}X_{i1}+b_{j2}X_{i2}+\cdots+b_{jn}X_{in})}} \tag{3.5}$$

Here, $P(Y = k|X_i)$ represents the probability that data X_i belongs to class k given the data X_i. K denotes the total number of classes, and $b_{k0}, b_{k1}, ..., b_{kn}$ are the parameters to be learnt during training. $X_{i1}, X_{i2}, ..., X_{in}$ are the features of the input data X_i.

The softmax function calculates the probability of input data belonging to each class and normalizes these probabilities so that their sum equals 1 across all classes.

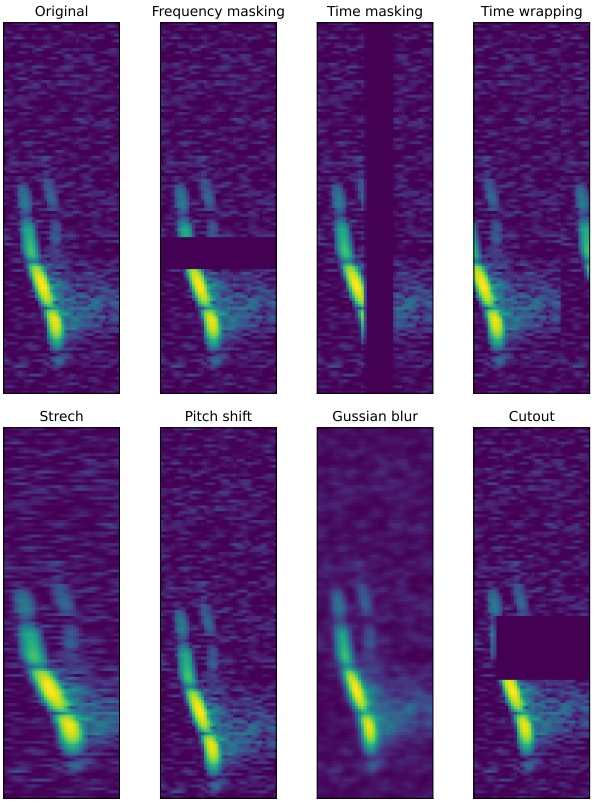

Figure 3.5: Illustration of all data augmentations applied in this report, showing the original spectrogram and 7 augmented views.

During training, a set of optimal parameters $b_{k0}, b_{k1}, ..., b_{kn}$ are learnt by minimising the cross entropy loss, which is the same loss function the baseline model applied during training. The formula of cross entropy loss is shown in equation 3.1.

3.4 Data Augmentation

Data augmentation is a machine learning technique that increases the diversity of the dataset by applying various transformations to existing data to generate slightly different copies and then added them to the dataset. The common transformations used in data augmentation include rotation, colour jittering, cropping and masking. Data augmentation is an effective regularization technique and could effectively reduce overfitting in the model training [41].

In our experiments, data augmentation is used in the training of the three models mentioned above (supervised ResNet-18, autoencoder and SimCLR). In the training of supervised ResNet-18 and autoencoder, data augmentation is used to increase the training dataset and reduce overfitting. In the training of SimCLR, data augmentation is used to generate positive and negative pairs, which is used to learn the meaning of visual representation by training the model to be invariant to transformations.

In this report, we employed 7 data augmentations in total, including time masking,

frequency masking, and time wrapping which are proposed by [28], these three augmentations are proposed for speech recognition tasks on log mel spectrogram. Following [44] we added strech and pitch shift, these two augmentations are applied in bioacoustics classification tasks on spectrograms. We also added other two types of popular data augmentation in the field of image classification, which are gaussian blur and random erasing. Illustrations of all the data augmentations applied in this report are shown in Figure 3.5

3.5 Machine Learning Pipeline

Finally, we introduce the pipeline for developing deep learning-based self-supervised models we employed in this report to pretrain the self-supervised models, starting with the raw audio input data and ending with a model capable of classifying bat echolocation calls. A flowchart illustrating the pipeline is provided in Figure 3.6.

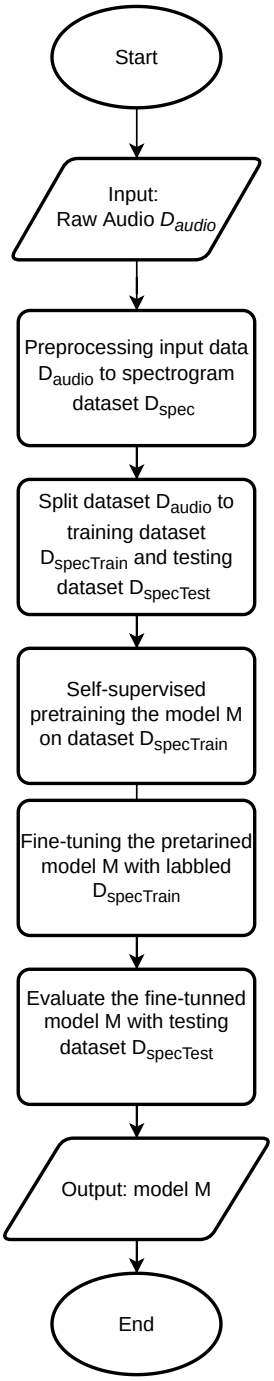

Figure 3.6: Machine learning pipeline to develop the deep learning model in this report.

Chapter 4

Dataset

In this chapter, we described the dataset used in this report, as well as outlined all the preprocessing steps we carried out in order to convert the raw audio recordings to spectrograms containing bat echolocation calls which we did the classification with.

4.1 Dataset Description

The dataset contains 2,737 recordings with an average length of 2 seconds. All recordings have a frequency range from 0kHz to 150kHz with a sample rate of 300,000. The dataset includes 32,651 calls from all 17 known breeding bat species in the UK, the class distribution of the dataset is shown in Table 4.1, and the mean average image of each class of the dataset is shown in Figure 4.1. The recordings are collected by different devices and were provided by six different sources, where each source corresponds to an organisation or individuals that provide multiple audio files. The multi-sources feature of the dataset maximises the variation in the dataset, which increases the challenge and improves the generalisation ability.

The annotations of the data are from [27], where they manually annotate each bat call using the audio annotation interface they developed. The species class information at the file level is provided to the annotators. The annotators draw boxes and assign a species class label to each call on each recording's spectrogram; an example of the spectrogram of a recording with the labels is shown in Figure 4.2. Under the condition where it is not possible to to assign the correct class or there are multiple species present in a single file, the unknown class calls will be marked to a generic class 'Bat'.

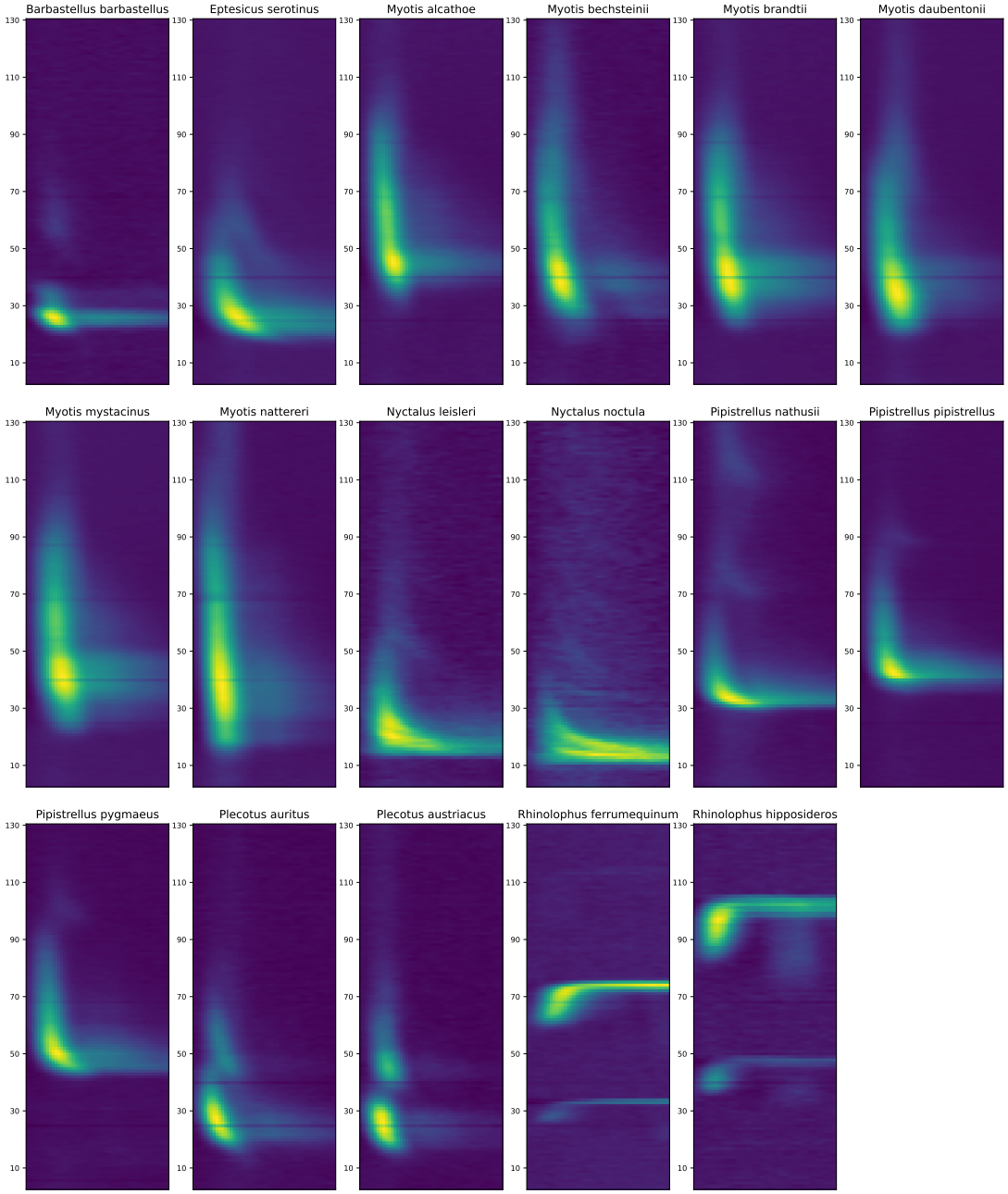

Figure 4.1: Visualisation of the average spectrogram of the echolocation calls from each species in the dataset. The y-axis represents kHz, which spans from 10kHz to 140kHz, the time duration for each spectrogram is 20ms.

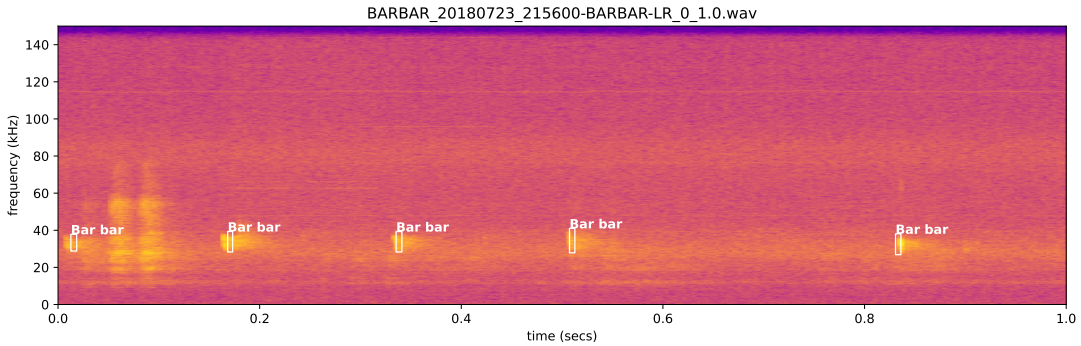

Figure 4.2: Example of the recording and annotations in the dataset, ”Bar bar” in the graph is short for bat species name Barbastellus barbastellus.

Class Name	Numbers	Num Train Calls	Num Test Calls
Bat	1389	1389	0
Barbastellus barbastellus	984	791	193
Eptesicus serotinus	2585	2025	560
Myotis alcathoe	866	702	164
Myotis bechsteinii	880	718	162
Myotis brandtii	1936	1553	383
Myotis daubentonii	6337	4992	1345
Myotis mystacinus	2757	2184	573
Myotis nattereri	2627	2090	537
Nyctalus leisleri	1150	879	271
Nyctalus noctula	402	302	100
Pipistrellus nathusii	1442	1145	297
Pipistrellus pipistrellus	1891	1493	398
Pipistrellus pygmaeus	2346	1903	443
Plecotus auritus	1120	814	306
Plecotus austriacus	842	680	162
Rhinolophus ferrumequinum	1637	1322	315
Rhinolophus hipposideros	1460	1173	287

Table 4.1: Class distribution of the dataset used.

4.2 Data Prepossessing

In this section, we outline each step applied to convert raw audio recordings into cut spectrograms, which serve as the input for our models. After loading all the audio recording files from the disk, we apply a short-time Fourier transformation with a window size of 1024 and an 80% window overlap to the audio to compute the magnitude spectrogram. We then trim out the frequency bands higher than 140 kHz or lower than 10 kHz, as most of the bat classes only appear in the frequency range between 10 kHz and 140 kHz. Similar to [27] and [3], we deduct the mean value of each frequency band from the frequency band to remove the constant background noise. Next, we crop out each bat call from the spectrogram by cropping out 20 milliseconds of data starting from 2.5 milliseconds before the call start. Although some bat calls have a duration longer than 20ms, the starting 20ms of the bat calls contain features most relevant to bat species, such as start frequencies and frequency with maximum energy [37]. As a final step, we resample each cropped spectrogram into 128 frequency bins. We assume only one bat call appears in one cropped spectrogram. Ultimately, when given an audio file containing mutiple bat echolocation calls, this preprocessing process generates multiple spectrograms each contraining one bat echolocation call with dimensions of 128 by 32.

Similar to [22], we add noise samples to the training data to increase the robustness to background noise. The noise samples are extracted from the space between adjacent calls in training data (i.e., after the end of the last call, before the start of the next call). All noise samples undergo the exact same preprocessing steps as the training data, with one exception: when cropping the spectrogram, we start cropping 20ms before the start of the call instead of at the call's ending. This approach is chosen because some bat calls exceed 20ms in duration. To make the noise sample representative, we collect 70 noise samples from each data source of the dataset, ensuring that each noise sample is collected from a different file. This results in 420 noise samples, which serve as the 18th class 'noise' of our classification task. It is important to note that these noise samples are only added to the training data and not the testing data.

4.3 Data Split

In this study, we have split our dataset into non-overlapping training and testing datasets with an 80:20 ratio in order to train and evaluate our models. To ensure that the training and testing data remain distinct, we have performed a file-level split, meaning that the calls from the training dataset and calls from the testing dataset always come from different files. This could prevent information in the testing dataset from leaking into the training dataset, ensuring the reliability of the evaluation of the model performance. To split the dataset on a file level while maintaining a similar class distribution in the training and testing data, we randomly assigned recording files to the training dataset while keeping track of the number of calls per species added. Once the number of calls of a species added to the training dataset reaches the desired threshold, all other files containing calls of this species are added to the testing dataset. This could maintain the species distribution in both datasets, which makes the evaluation of the model performance fair. Consequently, our dataset consists of 25,944 calls in the training set

and 6,496 calls in the testing set. A detailed breakdown of the class distribution in the training and testing datasets is shown in Table 4.1. It is important to note that, data from class "bat" will only be used for unlabelled pretraining, thus no data from class "bat" will be split into a test dataset.

To prevent class bias and generalise well on new data, we balanced our training dataset by downsampling each class's data to 300 data per class. This results in a balanced labelled dataset containing 5,270 bat calls in total, by adding 300 noise samples from training data, we result in a balanced labelled dataset containing 5,400 spectrograms. To be noted, this balanced dataset will only be used to train the fully supervised model and fine-tune/linear-evaluate the self-supervised model. The entire training dataset (without the balance) will be used for self-supervised pretraining.

To simulate the condition with limited labelled data, we created a subsampled version of the training dataset where each has 1%, 5%,10%, and 50% of data from the balanced training dataset. The subsampled versions are selected uniformly at random for each class. The subsampled version datasets with increasing size are nested, i.e. the version with 50% of the data will contain all the data in the version with 10% of the data. To make the result comparable, the dataset is static across the experiments of different models, i.e. the exact same data will be fed to three models for each set of experiments.

In the end, we have 7 datasets in total, one testing dataset contains 6,496 calls to evaluate the model performance, an unbalanced training dataset contains 25,944 calls to do the self-supervised pretraining, a balanced dataset contains 5,400 spectrogram and 4 subsampled versions of the balanced dataset which contains 2,700, 540, 270, 54 (50%, 10%, 5% 1%) of calls respectively, the balanced dataset will be used to train the fully supervised baseline and fine-tune/linear evaluate the models trained using a self-supervised manner.

Chapter 5

Experiments and Results

In this chapter, we discuss the settings for each experiment, the rationale behind conducting them, and the results obtained. Additionally, we provide an analysis of the results to offer insights into the performance of the models and the implications of the findings.

We conduct the first experiment to compare the classification performance between supervised baseline and pretrained self-supervised models under different amounts of labelled data. We aim to investigate the effectiveness of self-supervised pretraining under different sizes of labelled dataset in this experiment.

The second experiment is conducted to assess the quality of the visual representation learnt by the two self-supervised models during pretraining. We compare the visual representation learnt by two self-supervised models to that learnt by the supervised baseline and a randomly initialised baseline model without any training.

5.1 Implementation Details

In this report, all experiments have been carried out on the University of Edinburgh's teaching cluster. Each task within the cluster operates on the Ubuntu 20.04 LTS operating system, equipped with an NVIDIA GTX 1060 GPU 6GB and 16GB of RAM. The open-source PyTorch library [31] has been used for the development of all associated scripts.

5.2 Fully Supervised Baseline Experiment

In this report, we refer to the ResNet-18, which is a CNN-based model proposed in [14], as the baseline model.

During training, the model's weights are updated for each input batch of 32 samples, leveraging backpropagation to minimize cross-entropy loss. An Adam optimiser [23] with an initial learning rate of 1e-3 and a weight decay of 1e-3 is used. A cosine decay

Dataset Size		Macro-averaged Metric			
Number	Percentage	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
54	1%	37.23	35.53	36.52	33.29
270	5%	52.60	48.24	50.12	47.51
540	10%	58.15	53.25	56.73	53.84
2700	50%	71.15	62.62	67.26	62.51
5400	100%	75.92	66.96	72.02	66.04

Table 5.1: The macro-averaged metrics for the baseline model trained fully supervised on balanced datasets of varying sizes. The dataset size column displays the number of samples in the dataset and its percentage relative to the full balanced dataset.

learning rate scheduler [25] is also applied during training. The model is trained for 40 epochs.

The ResNet-18 model is trained on the full balanced training dataset (300 calls per class) and its subsampled versions, which contain 50%, 10%, 5%, and 1% of data from the balanced training dataset (i.e., each subsampled version contains 150, 30, 15, and 3 calls per class, respectively). All training is then tested on the testing dataset mentioned in section 4.3, comprising 6,496 calls while maintaining the original class distribution (i.e., the testing dataset is not balanced).

We run the experiments five times and the average results are displayed in Table 5.1, with the confusion matrix for the model trained on the full balanced dataset (300 samples per class) shown in Figure 5.1.

The confusion matrix reveals that most misclassifications occur in species with minimal inter-class differences, primarily within Myotis, Nyctalus, and Plecotus genera. For example, 37% of calls from Nyctalus noctula are misclassified as Nyctalus leisleri, 19% of calls from Myotis bechsteinii are misclassified as Myotis brandtii, and 14% of calls from Pipistrellus pipistrellus are misclassified as Pipistrellus pygmaeus. Based on average call images shown in Figure 4.1, these genera exhibit very subtle inter-class differences, which we interpret as the primary cause of the misclassifications.

5.3 Self-Supervised Experiments

This section explains the training details and the two self-supervised models (SimCLR and autoencoders), and the results of the experiments carried out on these two models.

5.3.1 SimCLR Model

To evaluate the SimCLR's performance on the task of bat echolocation calls, we applied a two-phase training approach. In the first phase, we trained the SimCLR model in a self-supervised manner using unlabelled data. Subsequently, in the second phase, we fine-tuned the model with labelled data to evaluate its classification performance.

The SimCLR model was pre-trained in a self-supervised fashion using the 25,944 calls in

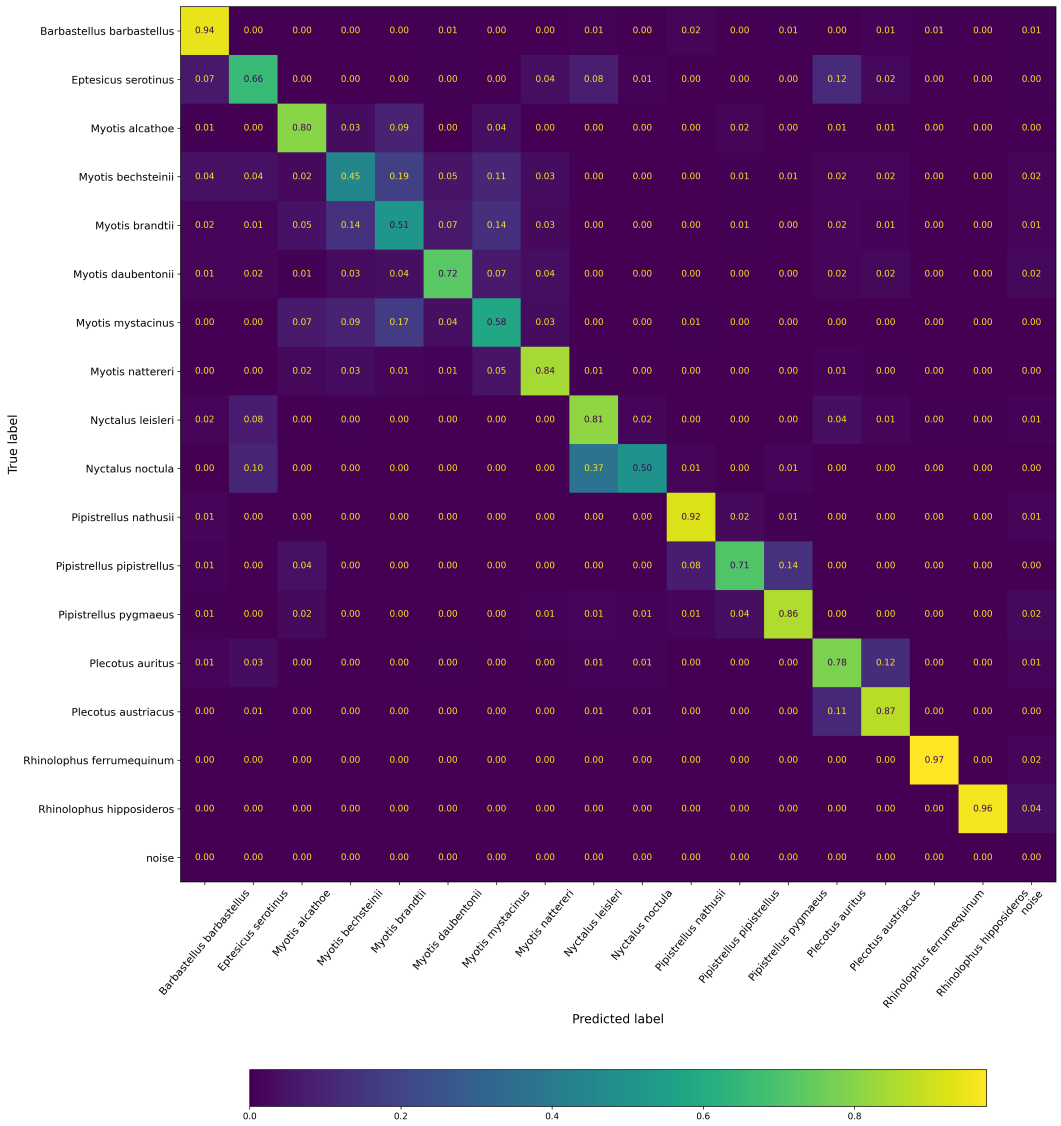

Figure 5.1: Confusion Matrix of the baseline model trained fully supervised on the complete balanced dataset (300 calls per class).

the training dataset, with only call data and no class labels. Model weights were updated per batch of 32 samples. Although both [6] and [7] indicated that the SimCLR model greatly benefits from larger batch sizes for more negative pair sampling, we opted for a batch size of 32 due to graphics card memory constraints. Furthermore, since our dataset contains only 18 classes, increasing the number of negative samples would raise the likelihood of same-class calls being labelled as negative samples, causing the contrastive loss to drive apart the representations of these same-class calls. We pre-trained the model for 200 epochs in a self-supervised manner, utilizing an Adam optimiser with an initial learning rate of 1e-3 and a weight decay of 1e-3. A ReduceLROnPlateau learning rate scheduler is also employed with a patience of 5 epochs. The temperature constant for the NT-Xent loss function is set to 0.5.

We proceed to fine-tune the pre-trained SimCLR network for the downstream task of

Dataset Size		Macro-averaged Metric			
Number	Percentage	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
54	1%	43.83	41.26	41.22	38.03
270	5%	61.18	57.33	69.67	56.89
540	10%	66.19	60.91	63.20	67.14
2700	50%	75.49	68.32	71.89	68.31
5400	100%	76.56	69.77	72.73	70.85

Table 5.2: The macro averaged metrics of the pretrained SimCLR model fine-tuned on the balanced datasets of varying sizes. The dataset size column displays the number of samples in the dataset and its percentage relative to the full balanced dataset.

bat echolocation call classification. Following the methods in [7], we fine-tune the network from the first dense layer of the projection head, discarding the remainder of the projection head. Subsequently, we add a dense layer to classify the output from the SimCLR network. The model is fine-tuned for 40 epochs with a batch size of 32 to minimise the cross-entropy loss. During training, we employ the Adam optimiser and CosineAnnealingLR scheduler, with an initial learning rate of 1e-3 and a weight decay of 1e-3.

Similar to baseline experiments, the pre-trained SimCLR model is fine-tuned on a balanced training dataset and its four subsampled versions (50%, 10%, 5%, 1%). The model is then tested on the testing dataset, which includes 6,496 calls.

Due to the limitation of computational power and the computational complexity of the pre-trained model. We only pretrained the model once and fine-tune on the pretrained model five times to collect results. The average results are presented in Table 5.2, and the confusion matrix for fine-tuning on the complete balanced dataset is displayed in Figure 5.2.

From the confusion matrix, we found that most of the misclassification still happens within the genus with small inter-class differences, where 33% of calls from Myotis bechsteinii are misclassified to Myotis brandtii, 22% of calls from Nyctalus leisleri are misclassified to Nyctalus noctula. However, 10% of Eptesicus serotinus calls are misclassified into Plecotus genus, which is not within the species group with small inter-class differences.

5.3.2 Autoencoder Model

Similar to SimCLR network, the autoencoder network is pre-trained using the 25,944 calls in the training dataset, with only call data and no class labels. The model weights are updated every batch with the objective of minimising the Mean Square Error between the input call spectrogram and recreated call spectrogram, where the batch size is 32. The model is trained with an Adam optimiser with an initial learning rate of 1e-3 and weight decay of 1e-3. A ReduceLROnPlateau scheduler is also employed with a patient of 5. The autoencoder model is trained for 200 epochs.

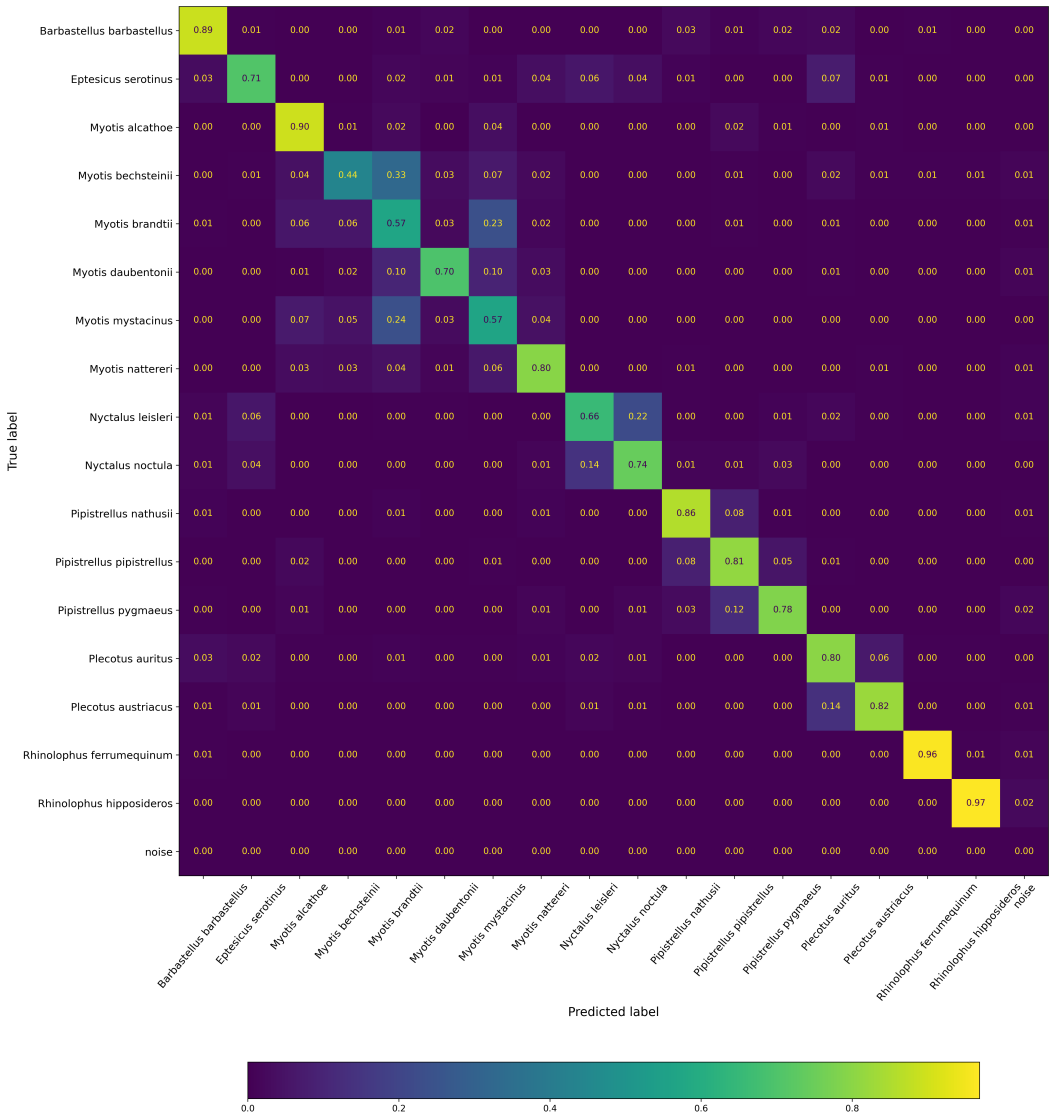

Figure 5.2: Confusion Matrix of the pretrained SimCLR model finetune on the complete balanced dataset (300 calls per class).

The pre-trained autoencoder network is then fine-tuned on the balanced training dataset and its four subsampled versions and tested on the testing dataset includes 6,496 calls. During fine-tuning, the decoder network is replaced with a classification layer. An exact same fine-tuning setup described in section 5.3.1 is used. Similar to the SimCLR model, the autoencoder model is also only pretrained once and fine-tuned five times. The average result is present in Table 5.3, and the confusion matrix when fine-tuning on the complete balanced dataset is shown in Figure 5.3.

From the confusion matrix, we found most of the misclassifications all happen within the species group which has less inter-class difference. 26% of the calls belonging to Myotis brandtii are misclassified to Myotis mystacinus, 20% of Plecotus austriacus calls are misclassified to Plecotus auritus.

Dataset Size		Macro-averaged Metric			
Number	Percentage	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
54	1%	42.19	39.27	41.64	38.62
270	5%	60.30	55.84	57.76	54.48
540	10%	65.18	59.72	52.64	58.14
2700	50%	74.11	69.40	70.01	68.79
5400	100%	77.02	70.41	73.85	71.51

Table 5.3: The macro averaged metrics of the pretrained autoencoder model finetune on the balanced datasets of varying sizes. The dataset size column displays the number of samples in the dataset and its percentage relative to the full balanced dataset.

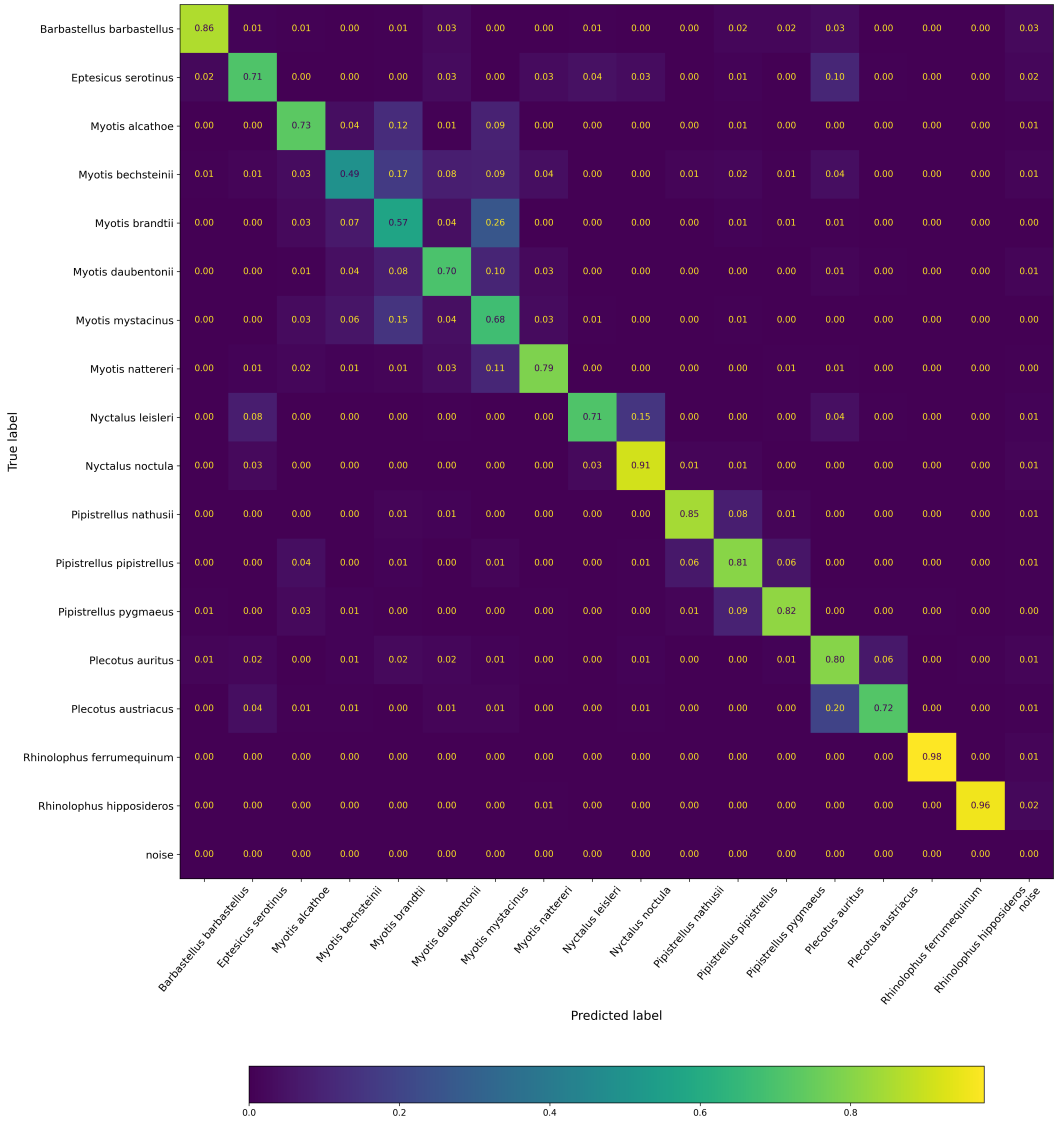

Figure 5.3: Confusion Matrix of the autoencoder model trained fully supervised on the complete balanced dataset (300 calls per class).

5.4 Comparison of the Different Models

A summary of the macro accuracy for each model trained on varying dataset sizes is presented as line plot in Figure 5.4 with the error bar presenting the standard division. The line plot demonstrates that the two self-supervised models consistently outperform the baseline ResNet-18 network across all dataset sizes, with SimCLR emerging as the top-performing model. The performance gap between models decreases as the training dataset size increases. The gap between SimCLR and the baseline is 6.5% when training with 1% of the dataset, which narrows down to 1% when training on the full dataset. Moreover, we observe that the self-supervised models achieve comparable performance to the fully supervised baseline while utilizing a significantly smaller amount of labelled data, particularly in the case of the SimCLR model. The SimCLR model, when trained on 50% of the dataset, attains performance similar to the fully supervised baseline trained on the entire dataset. The SimCLR model achieves a macro accuracy of 66% when trained on just 10% of the dataset, which is only 9% lower than the fully supervised baseline trained on 100% of the dataset.

In the previous sections, we identified that the primary challenge in this dataset is distinguishing bat species with small inter-class differences, as most misclassifications occurred within groups of bat species with minimal inter-class variations. From the three confusion matrices Figure 5.1, Figure 5.2 and Figure 5.3, we found that, compared to the fully supervised baseline, both self-supervised models exhibit better performance in differentiating calls belongs to species within genera with minor inter-class differences. For instance, the average accuracy of Nyctalus genus of the baseline model is 65%, the SimCLR and autoencoder achieved an average accuracy of 70% and 81% across the Nyctalus genus. For Myotis genus, which is the most challenging genus to differentiate, the baseline model achieved an average accuracy of 65%, while the SimCLR and autoencoder rise this to 68% and 66% respectively. However, self-supervised models fail to perform better on Plecotus genus, where the baseline achieved 82%, the SimCLR and autoencoder achieved 81% and 76% respectively. It is worth noting that the confusion matrices of the three models when trained on 100% of the dataset are compared since the differences in macro metrics are the smallest (up to 1%), allowing for the identification of differences in each class's performance through comparison.

5.5 Linear Evaluation of the Learnt Representations

We have compared the fine-tuning performance of the two self-supervised models in the downstream task of bat echolocation classification. In this section, we further evaluate the quality of the visual representation learnt by each self-supervised model by performing linear evaluation on the representation extracted by the self-supervised pretrained models. We then analyze the linear evaluation performance with the aid of t-SNE plots of the extracted representations to provide better insight.

The linear evaluation method involves training a linear classifier on the representations extracted by the pretrained models to produce the final classification model. Unlike the end-to-end fine-tuning in the previous sections, the linear evaluation does not change the pretrained model weights during the training process. Instead, only the weights of

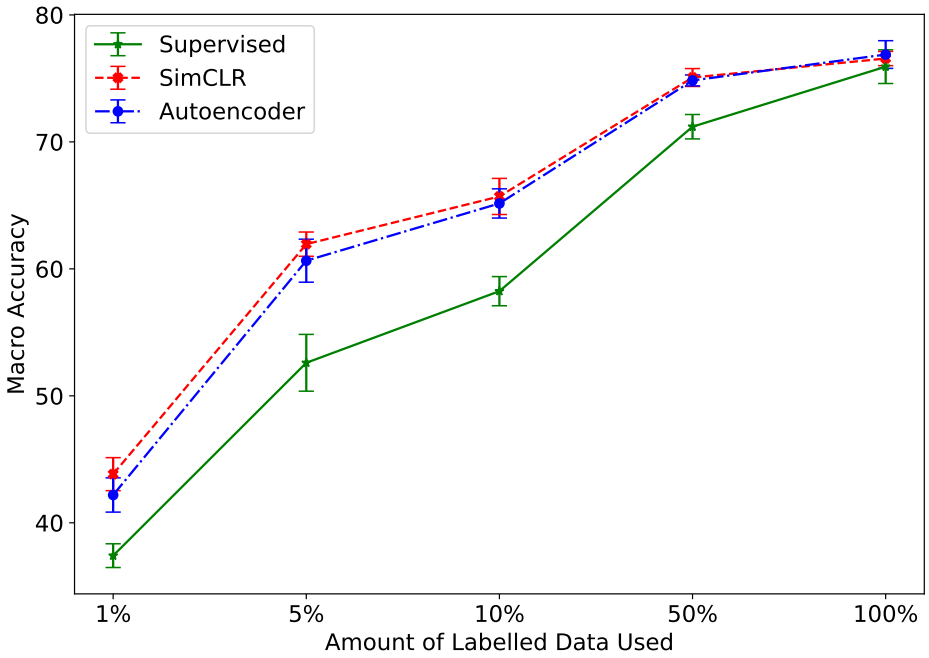

Figure 5.4: Macro average accuracy of supervised ResNet-18 training from scratch, pretrained SimCLR, and autoencoder when fine-tuning on datasets with different sizes. The error bar shows the standard division.

the linear classifier are adjusted.

We applied a multinomial logistic regression as the linear classifier to classify the visual representations extracted by the pretrained models. For the autoencoder, the output of the encoder h_i is used as the representation being evaluated. For SimCLR, the output of the encoder network h_i is used instead of the output of the projection head z_i, as [6] found classes represented by h_i are better separated. We implemented the multinomial logistic regression classifier using the *LogisticRegression* classifier from the *sklearn.linear_model* package. We trained and tested the classifier on the visual representations extracted by the pretrained models. We trained the classifier on the full balanced dataset (300 calls per class) and its four subsampled versions (50%, 10%, 5%, and 1%). The classifier was then tested on the testing dataset containing 6,496 calls.

To make the performance of the two self-supervised models comparable, we assessed the visual representation extracted by a ResNet-18 model with randomly initialised weights without any training. The performance of the end-to-end finetuned baseline ResNet-18 is also added to make the performance of the self-supervised models comparable.

5.5.1 Species Level Performance

The multinomial logistic regression is applied to classify the representations extracted by the pretrained SimCLR, pretrained autoencoder, and randomly initialized ResNet-18

model. Species-level labels are used during training. The average test macro accuracies, along with the standard deviation, for the logistic regression classifier applied to the representations extracted by the three models, are shown in Figure 5.5. The performance of the end-to-end trained supervised baseline is also included for comparison.

From Figure 5.5, we observe that both the SimCLR model and autoencoder achieve significantly better performance compared to the randomly initialised ResNet-18 network. This implies that both self-supervised models have learnt meaningful visual representations without using any labels. Compared with the supervised baseline, both self-supervised models outperform the supervised baseline when training using 1% and 5% of the labelled data. The baseline start outperforms the SimCLR model when training with 10% of the labelled data, and starts outperforms the autoencoder model when training with 50% of the labelled data. The performance difference keeps increasing until training with 100% of the labelled data. The performance gap between the supervised baseline and self-supervised methods suggests that though a meaningful representation has been learnt, there is still a gap compared with the representation learnt by the supervised baseline.

We also generated t-SNE plots for the representations extracted by the four models from the testing data. The t-SNE plots were generated using the *TSNE* function from the *sklearn.manifold* package, with the *n_component* parameter set to 2. A detailed explanation of how the t-SNE processes and how t-SNE plots are generated is in Appendix 7.1. The resulting t-SNE plots are shown in Figure 5.7.

From the t-SNE plots, we found that compared to the randomly initialized ResNet-18, both autoencoder and SimCLR models have learnt meaningful representations that can differentiate most bat species without any labels. In the t-SNE plot for autoencoder, though most of the classes have been separated, the distribution between data is relatively loose and no distinct clusters are formed. In the t-SNE plot for SimCLR, a few distinct clusters can be observed. However, unlike the t-SNE plot for the supervised baseline, some clusters in the t-SNE plot for SimCLR contain multiple species within the same genus, such as Pipistrellus pygmaeus and Pipistrellus pipistrellus, which are species with small inter-class differences. This makes it difficult for the logistic regression classifier to separate calls from these classes. This can be further explained by Figure 5.5, where the SimCLR outperform the autoencoder by 3% when trained with 1% of the dataset, but the autoencoder starts to outperform the SimCLR from training with 5% dataset with the performance gap keeps increasing as the amount of training data increase. We interpret this as the distinct clusters formed by SimCLR can be easily classified with a small amount of data. But as the amount of training dataset increase, the close clusters formed by the SimCLR make it difficult for the logistic regression classifier to classify, but the loose distribution in the t-SNE plots for autoencoder allows the logistic regression classifier to draw a more complex decision boundary when more data is used in training.

However, during end-to-end fine-tuning, the SimCLR model outperforms the autoencoder across all dataset sizes, which contrasts with the trend in logistic regression accuracy. We explain this observation by noting that pretraining helps the SimCLR model differentiate between genus groups with large inter-class differences. When

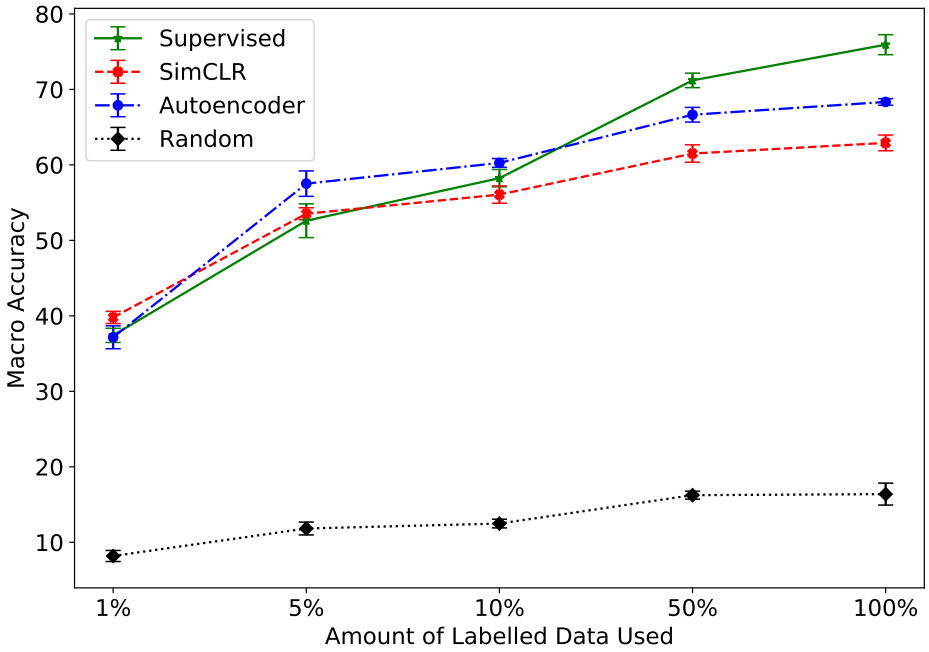

Figure 5.5: The species level macro average accuracy of the logistic regression classifier, when trained on the representations extracted by the randomly initialized ResNet-18, SimCLR, and autoencoder with datasets of various sizes. Additionally, the macro average accuracy of the ResNet-18 baseline model, which is trained end-to-end on datasets with different sizes, is shown. The error bars represent the standard deviation of the accuracy measurements.

fine-tuning, the SimCLR model can focus on differentiating calls with small inter-class differences. For the autoencoder, no clear clusters are formed during pretraining, so when fine-tuning, the autoencoder still has to differentiate calls with large inter-class differences. Considering during pretraining, the autoencoder model has a model size which is nearly double that of SimCLR, the result suggests that the SimCLR model is more effective and efficient in learning meaningful representations that can be fine-tuned for the downstream task of bat echolocation call classification.

5.5.2 Genus Level Performance

As we found that, through self-supervised pretraining, both self-supervised models cluster some bat species within the same genus closely. Especially for the SimCLR model, many distinct clusters containing multiple species within the same genus have been formed during pretraining. Thus, we further explore the linear evaluation performance of the self-supervised models at the genus level.

The trained models utilise the species-level labels from the above section are used, but instead of reporting the species-level accuracy, the performance at the genus level is reported, i.e., the misclassification within the genus will be considered correct

classification. The genus-level macro average accuracies of the four models (SimCLR, autoencoder, randomly initialised ResNet-18 and supervised ResNet-18) are shown in Figure 5.6.

From Figure 5.6, we find that both self-supervised models significantly outperform the randomly initialised baseline, indicating a meaningful representation has been learnt at the genus level. We observe that compared with species-level performance, the self-supervised pertaining has a more significant effect at genus-level performance. At the genus level, the SimCLR outperform the supervised baseline for 16% when training with 1% of the dataset. but at the species level, the SimCLR only outperform the baseline for 3%. The SimCLR model outperforms the supervised baseline for all the dataset sizes below 100%. Even when training with 100% of the dataset, the supervised baseline only outperforms the SimCLR model by 2%. This shows the SimCLR model has learnt a competitive representation compared with the supervised baseline without using any labelled data. In contrast with species-level performance, the SimCLR outperforms the autoencoder under all dataset sizes, which could be explained by the more distinct clusters at the genus level has been formed during the pretraining for SimCLR.

We also generate t-SNE plots at the genus level for the visual representations extracted by the four models (SimCLR, autoencoder, fully supervised ResNet-18, and randomly initialized ResNet-18) to further analyze their performance. The t-SNE graphs are shown in Figure 5.8 Compared with the randomly initialised ResNet-18 network, both self-supervised models have learnt meaningful representations at the genus level. We observe that SimCLR produces distinct, high-quality clusters that are competitive with those of the fully supervised model. These results demonstrate the effectiveness of SimCLR in learning discriminative visual representations at the genus level, even in the absence of labelled data. This is also shown in Figure 5.6, where there is only a 2% performance difference between SimCLR and supervised baseline when training with 100% of the dataset. In contrast, the autoencoder also produces clear clusters at the genus level, but these clusters are not as well-separated as those generated by SimCLR or the fully supervised model. Most of the genus clusters in the autoencoder's t-SNE plot interact with each other, indicating that the model might struggle to distinguish between different genera effectively. This could be further explained by SimCLR outperforming the autoencoder under all dataset sizes for genus-level linear evaluation accuracy shown in Figure 5.6.

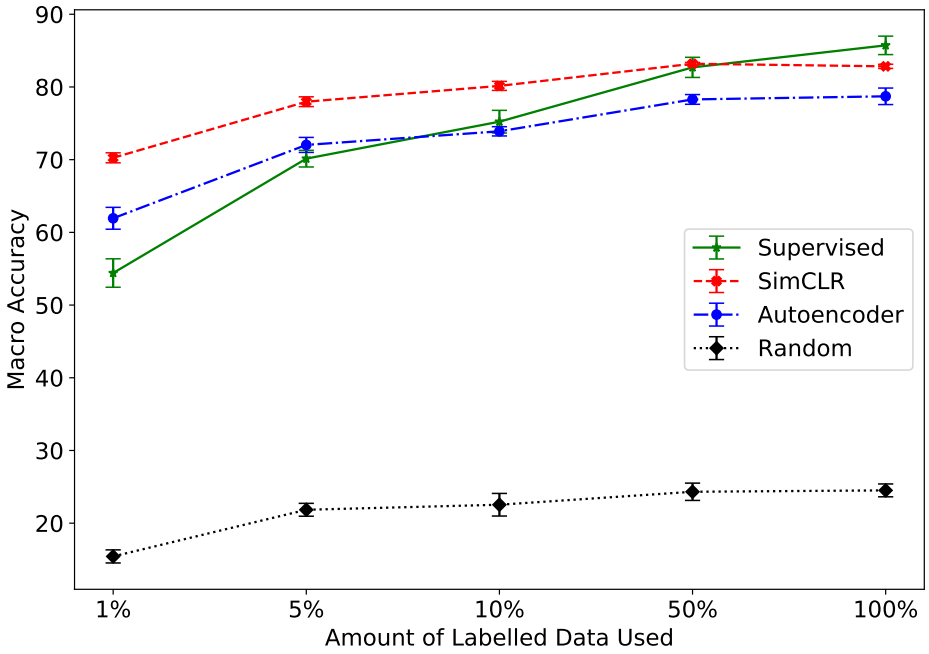

Figure 5.6: The genus level macro average accuracy of the logistic regression classifier, when trained on the representations extracted by the randomly initialized ResNet-18, SimCLR, and autoencoder with datasets of various sizes. Additionally, the macro average accuracy of the ResNet-18 baseline model, which is trained end-to-end on datasets with different sizes, is shown. The error bars represent the standard deviation of the accuracy measurements.

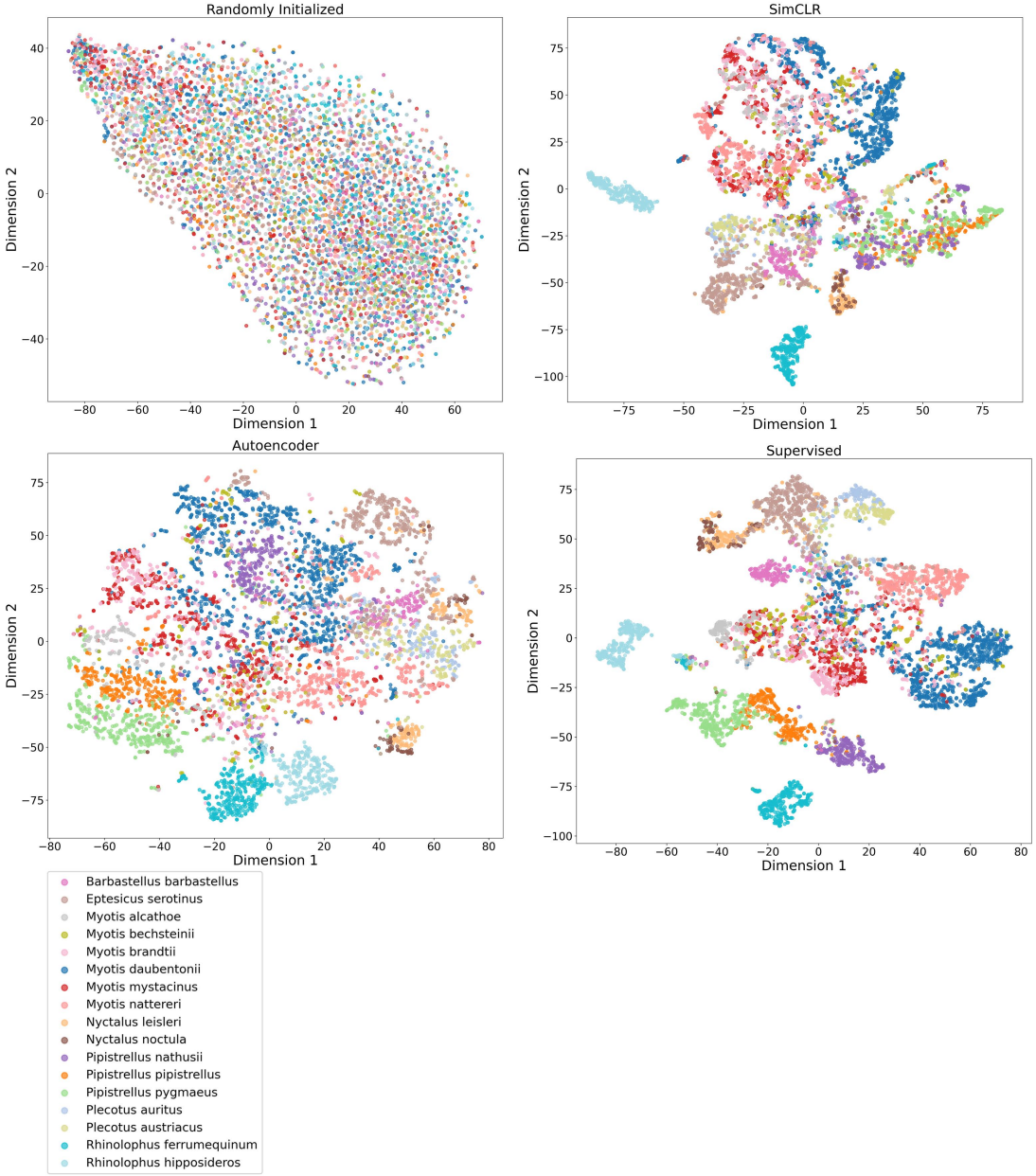

Figure 5.7: The t-SNE plots on species level for the visual representation extracted by Supervised ResNet-18, Randomly initialised ResNet-18, autoencoder and SimCLR model from the testing dataset.

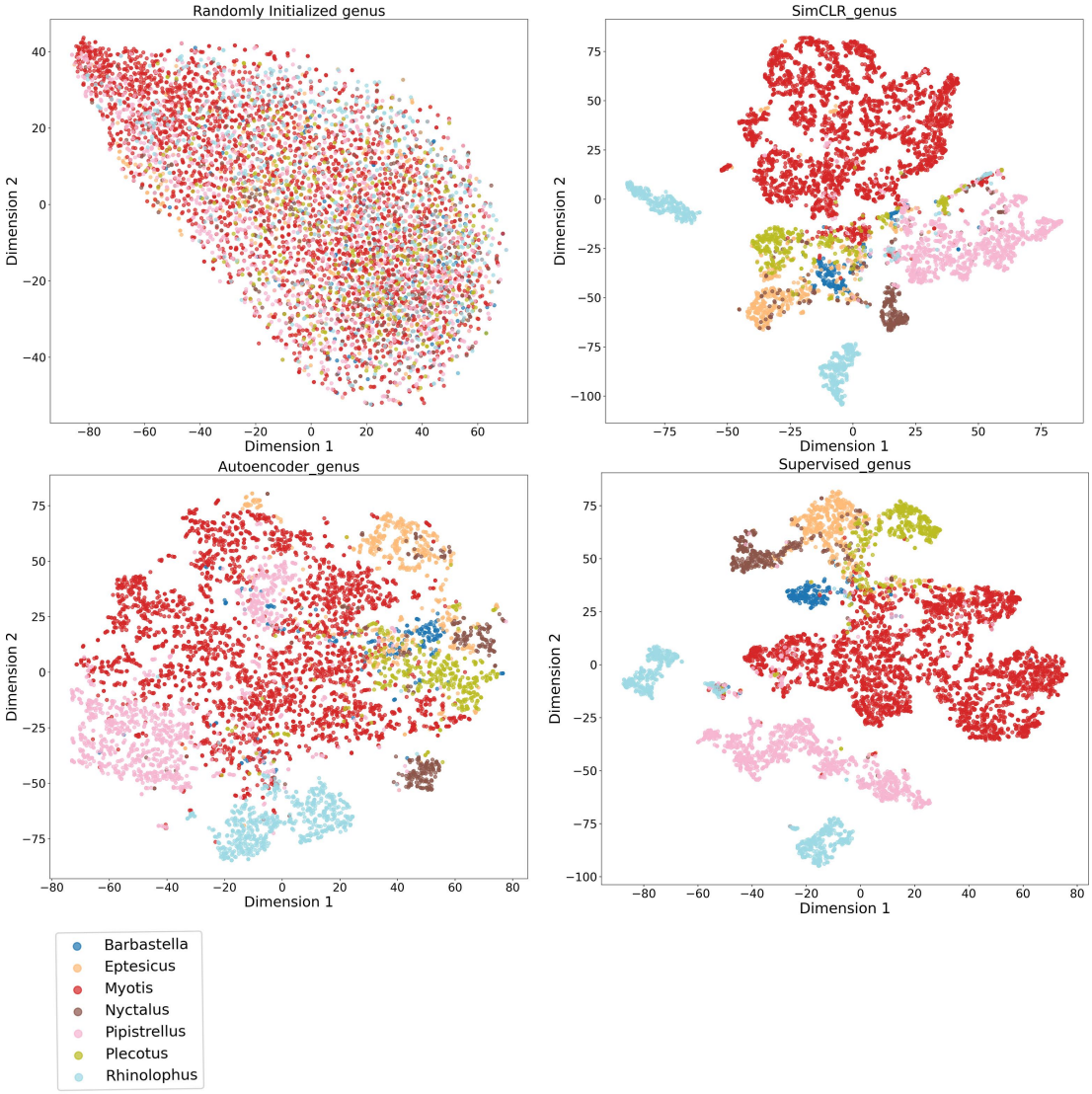

Figure 5.8: The t-SNE plots on genus level for the visual representation extracted by Supervised ResNet-18, Randomly initialised ResNet-18, autoencoder and SimCLR model from the testing dataset.

Chapter 6

Conclusions

6.1 Summary of Result

In this report, we applied self-supervised learning to the task of bat echolocation call classification for the 17 bat species that breed in the UK. Specifically, we implemented two self-supervised learning architectures: SimCLR and autoencoder, trained them on the bat call spectrogram database we established for this purpose, and compared their end-to-end finetuning performance with a fully supervised ResNet-18 baseline. By comparing the end-to-end finetuning classification performance on bat echolocation calls, we found that the SimCLR model achieved a test accuracy of 75.49% using 50% of the labelled dataset. This is very close to the testing performance achieved by the baseline model using 100% of the labelled dataset (75.92%). However, compared to the baseline, both self-supervised models only achieved a marginal performance improvement (around 1%) when using a complete labelled training dataset. By comparing the linear evaluation performance of visual representations learnt by the self-supervised models through pretraining with those of the supervised baseline model, we found that, at the species level, both self-supervised model have learnt a meaningful representation through self-supervised pertraining. At the genus level, the self-supervised pretraining has shown a larger effect, where the SimCLR model can learn to produce visual representations that are competitive with those learnt by the supervised baseline without using any labelled data. This is also shown by the linear evaluation accuracy of the SimCLR model when training with 100% of labelled data is only 2% lower than that of the supervised baseline trained end-to-end. We identify that the primary reason for misclassification in species discrimination tasks is the low inter-class differences between species, and self-supervised pretrained models have demonstrated improved performance in addressing this challenge.

In conclusion, we have achieved the goal of this project while addressing the four research questions proposed in section 1.2. The findings demonstrate the potential of self-supervised learning approaches for bat echolocation call classification and provide valuable insights for future research in this area.

6.2 Limitations and Future Work

One limitation of the experiments in this report is that we tested our model on a single dataset, which raises questions about the generalisability of our conclusions to bat echolocation call datasets collected from geographically distinct areas. Therefore, it would be worthwhile to investigate whether the same conclusions hold when tested on datasets collected from different geographical locations.

Another limitation of the pipeline proposed in this report is its reliance on a large annotated dataset containing information about the starting and ending time of each call. To reduce this dependency, a self-supervised approach could be applied to detect bat echolocation calls without requiring any labelled data. Such as the method proposed by Bermant et al. [4], which using self-supervised learning in detecting sperm whale coda click.

For future work, it is essential to consider the limited computational power of most devices used for audio-based monitoring tasks, such as Raspberry Pi. In many cases, these devices may not have sufficient computational resources to run a model with a ResNet-18 architecture in real-time, especially considering the preprocessing steps required to convert raw audio into spectrograms. Thus, it would be valuable to test the proposed pipeline on smaller neural network architectures (e.g., MobileNet [16]) to determine if self-supervised pretraining remains effective under resource-constrained settings.

Bibliography

[1] Nature-Smart Cities. https://naturesmartcities.com/about/, accessed 18 March 2023.

[2] Amanda M Adams, Meredith K Jantzen, Rachel M Hamilton, and Melville Brockett Fenton. Do you hear what i hear? implications of detector selection for acoustic monitoring of bats. *Methods in Ecology and Evolution*, 3(6):992–998, 2012.

[3] T Mitchell Aide, Carlos Corrada-Bravo, Marconi Campos-Cerqueira, Carlos Milan, Giovany Vega, and Rafael Alvarez. Real-time bioacoustics monitoring and automated species identification. *PeerJ*, 1:e103, 2013.

[4] Peter C Bermant, Leandra Brickson, and Alexander J Titus. Bioacoustic event detection with self-supervised contrastive learning. *bioRxiv*, pages 2022–10, 2022.

[5] A.J. Cain. Taxonomy.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[7] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

[8] Xing Chen, Jun Zhao, Yan-hua Chen, Wei Zhou, and Alice C Hughes. Automatic standardized processing and identification of tropical bat calls using deep learning approaches. *Biological Conservation*, 241:108269, 2020.

[9] Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5414–5423, 2021.

[10] Linus Ericsson, Henry Gouk, and Timothy M. Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5414–5423, June 2021.

[11] Stanley D Gehrt and James E Chelsvig. Bat activity in an urban landscape: patterns at the landscape and microhabitat scale. *Urban ecology: An international perspective on the interaction between humans and nature*, pages 437–453, 2008.

[12] Robert M Gray and Joseph W Goodman. *Fourier transforms: an introduction for engineers*, volume 322. Springer Science & Business Media, 2012.

[13] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] EA Holt and SW Miller. Bioindicators: Using organisms to measure environmental impacts. nature education knowledge 3 (10): 8, 2011.

[16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[18] Gareth Jones and Marc W Holderied. Bat echolocation calls: adaptation and convergent evolution. *Proceedings of the Royal Society B: Biological Sciences*, 274(1612):905–912, 2007.

[19] Gareth Jones, David S Jacobs, Thomas H Kunz, Michael R Willig, and Paul A Racey. Carpe noctem: the importance of bats as bioindicators. *Endangered species research*, 8(1-2):93–115, 2009.

[20] Gareth Jones and Emma C Teeling. The evolution of echolocation in bats. *Trends in Ecology & Evolution*, 21(3):149–156, 2006.

[21] K Jung, EKV Kalko, and O von Helversen. Echolocation calls in central american emballonurid bats: signal design and call frequency alternation. *Journal of Zoology*, 272(2):125–137, 2007.

[22] Stefan Kahl, Thomas Wilhelm-Stein, Hussein Hussein, Holger Klinck, Danny Kowerko, Marc Ritter, and Maximilian Eibl. Large-scale bird sound classification using convolutional neural networks. *CLEF (working notes)*, 1866, 2017.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Keigo Kobayashi, Keisuke Masuda, Chihiro Haga, Takanori Matsui, Dai Fukui, and Takashi Machimura. Development of a species identification system of japanese bats from echolocation calls using convolutional neural networks. *Ecological Informatics*, 62:101253, 2021.

[25] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[26] Oisin Mac Aodha, Rory Gibb, Kate E Barlow, Ella Browning, Michael Firman, Robin Freeman, Briana Harder, Libby Kinsey, Gary R Mead, Stuart E Newson, et al. Bat detective—deep learning tools for bat acoustic signal detection. *PLoS computational biology*, 14(3):e1005995, 2018.

[27] Oisin Mac Aodha, Santiago Martinez Balvanera, Elise Damstra, Martyn Cooke, Philip Eichinski, Ella Browning, Michel Barataud, Katherine Boughey, Roger Coles, Giada Giacomini, et al. Towards a general approach for bat echolocation detection and classification. *bioRxiv*, pages 2022–12, 2022.

[28] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

[29] Trishala K Parmar, Deepak Rawtani, and YK Agrawal. Bioindicators: the natural indicator of environmental pollution. *Frontiers in life science*, 9(2):110–118, 2016.

[30] Stuart Parsons and Gareth Jones. Acoustic identification of twelve species of echolocating bat by discriminant function analysis and artificial neural networks. *Journal of experimental biology*, 203(17):2641–2656, 2000.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[32] Farheen Ramzan, Muhammad Usman Ghani Khan, Asim Rehmat, Sajid Iqbal, Tanzila Saba, Amjad Rehman, and Zahid Mehmood. A deep learning approach for automated diagnosis and multi-class classification of alzheimer's disease stages using resting-state fmri and residual neural networks. *Journal of medical systems*, 44:1–16, 2020.

[33] Orly Razgour, Hugo Rebelo, Mirko Di Febbraro, and Danilo Russo. Painting maps with bats: species distribution modelling in bat research and conservation. *Hystrix*, 27(1):1–8, 2016.

[34] Charlotte Roemer, Jean-François Julien, and Yves Bas. An automatic classifier of bat sonotypes around the world. *Methods in Ecology and Evolution*, 12(12):2432–2444, 2021.

[35] Danilo Russo, Leonardo Ancillotto, and Gareth Jones. Bats are still not birds in the digital era: echolocation call variation and why it matters for bat species identification. *Canadian Journal of Zoology*, 96(2):63–78, 2018.

[36] Danilo Russo, Geoff Billington, Fabio Bontadina, Jasja Dekker, Markus Dietz, Suren Gazaryan, Gareth Jones, Angelika Meschede, Hugo Rebelo, Guido Reiter, et al. Identifying key research objectives to make european forests greener for bats. *Frontiers in Ecology and Evolution*, 4:87, 2016.

[37] Danilo Russo and Gareth Jones. Identification of twenty-two bat species (mammalia: Chiroptera) from italy by analysis of time-expanded recordings of echolocation calls. *Journal of Zoology*, 258(1):91–103, 2002.

[38] Danilo Russo and Christian C Voigt. The use of automated identification of bat echolocation calls in acoustic monitoring: A cautionary note for a sound analysis. *Ecological Indicators*, 66:598–602, 2016.

[39] Hans-Ulrich Schnitzler, Cynthia F Moss, and Annette Denzinger. From spatial orientation to food acquisition in echolocating bats. *Trends in Ecology & Evolution*, 18(8):386–394, 2003.

[40] Neha Sharma, Vibhor Jain, and Anju Mishra. An analysis of convolutional neural networks for image classification. *Procedia computer science*, 132:377–384, 2018.

[41] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[42] NB Simmons and AL Cirranello. Bat species of the world: a taxonomic and geographic database. *Accessed on*, 7(10):2020, 2020.

[43] Michael G Simpson. *Plant systematics*. Academic press, 2019.

[44] Elias Sprengel, Martin Jaggi, Yannic Kilcher, and Thomas Hofmann. Audio based bird species identification using deep learning techniques. Technical report, 2016.

[45] Michael A Tabak, Kevin L Murray, Ashley M Reed, John A Lombardi, and Kimberly J Bay. Automated classification of bat echolocation call recordings with artificial intelligence. *Ecological Informatics*, 68:101526, 2022.

[46] Achyut Mani Tripathi and Aakansha Mishra. Self-supervised learning for environmental sound classification. *Applied Acoustics*, 182:108183, 2021.

[47] Charlotte L Walters, Alanna Collen, Tim Lucas, Kim Mroz, Catherine A Sayer, and Kate E Jones. Challenges of using bioacoustics to globally monitor bats. *Bat evolution, ecology, and conservation*, pages 479–499, 2013.

Chapter 7

Appendix

7.1 T-SNE (t-distributed Stochastic Neighbour Embedding)

T-distributed Stochastic Neighbour Embedding (t-SNE) is a dimension-reduction method, which could represent high-dimensional data in low-dimensional space (usually 2 or 3 dimensions) while preserving the cluster structures and distance between data points.

There are three main steps in t-SNE to convert high-dimensional representation into low-dimensional space:

1. Calculating pairwise similarity p_{ij} in high dimensional space using Gaussian distribution. The similarity p_{ij} is calculated using the formula shown below.

$$p_{ij} = \frac{\exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} \exp(-||x_k - x_l||^2/2\sigma^2)} \tag{7.1}$$

   Where x_i and x_j in the formula are the data points, sigma is the standard division of the Gaussian distribution.

2. Optimising the low-dimensional representation. We first randomly initialise the low-dimensional representation y_i and y_j, which are the low-dimensional counterpart of the high-dimensional data points x_i and x_j. Then the pairwise similarity q_{ij} is calculated in low dimensional space using t-distribution. The similarity q_{ij} is calculated using the formula shown below.

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l} (1 + ||y_k - y_l||^2)^{-1}} \tag{7.2}$$

   The low-dimensional representations are then optimised with the objective to minimise the divergence between similarity in high and low dimensional space

(q_{ij} and p_{ij}). Kullback-Leibler (KL) divergence is applied to calculate the divergence, the divergence is optimised using gradient descent. The formula for KL divergence is shown below.

$$\mathrm{KL}(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{7.3}$$

In this report, t-SNE is employed as a visualization technique to evaluate the quality of representations extracted by the models. By examining the t-SNE plots, we can discern whether distinct clusters are formed for different bat species. Well-separated clusters indicate that the models have learnt meaningful representations that can distinguish between the various species.

However, t-SNE plots have its limitation in representing high-dimensional data. The t-SNE algorithm attempts to preserve the cluster information while compressing high-dimensional data into a low-dimensional space. Despite its effectiveness in maintaining cluster structure, some information will inevitably be lost during the dimensionality reduction process. As a result, t-SNE plots may not represent the original high-dimensional data entirely.