Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Ole-Magnus Vian Norum

Exploring Self-Supervised Learning for Bird Vocalization Classification

Master's thesis in Computer Science
Supervisor: Keith Downing
July 2023

NTNU
Norwegian University of
Science and Technology

Ole-Magnus Vian Norum

Exploring Self-Supervised Learning for Bird Vocalization Classification

Master's thesis in Computer Science
Supervisor: Keith Downing
July 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Ole-Magnus Vian Norum

Exploring Self-Supervised Learning for Bird Vocalization Classification

Master of Science in Computer Science
Submission date: July 2023
Supervisor: Keith Downing, IDI

Data and Artificial Intelligence
Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering

Abstract

Birds play an essential role in the ecosystem they inhabit and can be used as indicator species for assessing the health of an ecosystem. Because of these factors, bird species monitoring has become a viable strategy for evaluating ecosystem health. In recent years bird vocalization classification has become a popular topic within machine learning competitions, where solutions using neural networks are the most popular. Training large neural network models to classify bird vocalizations faces the challenge of most of the available bird recordings being unlabeled or weakly labeled. In recent years, methods for using unlabeled datasets for pre-training machine learning models called self-supervised learning, have shown promise. Since most of the bird vocalization recordings are unlabeled, self-supervised learning might be a good application for bird vocalization classification.

This thesis investigates the use of self-supervised learning for the bird vocalization classification task. By using a self-supervised learning implementation, called SimSiam, the impact of different augmentations used during self-supervised learning is explored. The self-supervised models, using the best performing augmentations, are also compared to a model pre-trained on the image domain. The models are linearly fine-tuned and fine-tuned end-to-end, using varying amounts of training data to assess their performance.

The results from testing different augmentations found that using a combination of augmentations during self-supervised learning performs better than single augmentations. The best combination tested in this thesis were cropping and adding noise. From comparing the self-supervised models with an identical model pre-trained on images, it was found that the model pre-trained on images outperformed the self-supervised models on all tasks. From looking at the embeddings produced by the self-supervised models, it is clear that the self-supervision was not able to learn the models' appropriate feature extractions to create embeddings useful for the downstream task of bird vocalization classification.

Sammendrag

(This is a Norwegian translation of the abstract.)

Fugler har en essensiell rolle i økosystemet de tilhører og kan bli brukt som indikasjonsarter for å vurdere helsen til et økosystem. På grunn av disse faktorene, har fugleartmonitorering blitt en mulig strategi for å evaluere økosystemers helse. I senere år har fugleklassifikasjon blitt et populært emne blant maskinlæringskonkurranser, hvor de fleste bruker nevrale nettverk for å løse oppgaven. En utfordring med å trene store nevrale nettverk for å klassifisere fuglelyder er at mesteparten av de tilgjengelige dataene er ikke annotert eller bare delvis annotert. I nyere tid har metoder for å bruke ikke-annoterte data for å trene modeller, kalt selvstyrt læring, vist lovende resultater. Etter som mesteparten av fuglelydene ikke er annotert, kan selvstyrt læring være en passende metode å bruke for fuglesangklassifikasjon.

Denne masteroppgaven utforsker bruken av forhåndstrening via selvstyrt læring for fugleklassifikasjonsoppgaven. Ved å bruke en implementasjon av selvstyrt læring, kalt SimSiam, utforskes effekten av ulike augmentasjoner som blir brukt under den selvstyrte læringen. De selvstyrte modellene som brukte de mest effektive augmentasjonene, sammenlignes også med en modell som har blitt forhåndstrent på annoterte bilder. Modellene finjusteres med flere ulike mengder av et annotert fuglesang-datasett, og evalueres basert på deres klassifiseringsevne.

Resultatene fra å teste ulike augmentasjoner viser at det å bruke en kombinasjon av augmentasjoner under den selvstyrte læringen gir bedre resultater enn det å bruke enkeltstående augmentasjoner. Den beste kombinasjonen av augmentasjoner som ble testet i denne masteroppgaven var å trimme spektrogrammet og legge til støy. Ved å sammenligne modellene som brukte selvstyrt læring med modellen som var forhåndstrent på bilder, ble det funnet at modellen forhåndstrent på bilder gjorde det bedre enn modellene som brukte selvstyrt læring. Ved å se på representasjonene produsert av de selvstyrte modellene, er det tydelig at modellene ikke var i stand til å lære viktige egenskaper ved dataen som kunne vært nyttige for den påfølgende klassifiseringsoppgaven.

Preface

This paper was written as a master's thesis at the Department of Computer and Information Science, at the Norwegian University of Science and Technology (NTNU).

I would like to thank my supervisor Professor Keith L. Downing for his invaluable guidance and feedback during the making of this thesis. I also want to thank the maintainers and contributors of xeno-canto.org for sharing their recordings of bird vocalizations with the public.

Ole-Magnus Vian Norum
Trondheim, June 20, 2023

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Large parts of this chapter were taken from my project report (Norum 2022). This chapter covers the introduction to this thesis. Section 1.1 introduces the motivation behind this paper. Section 1.2 states the goal of this research, as well as the research questions this thesis intends to answer. Under section 1.3, the methodology for addressing the goals will be disclosed. In section 1.4, the contribution of the research is stated. Lastly, the structure of the thesis is detailed in section 1.5.

1.1 Background and Motivation

Birds play an important part in the ecosystem, providing several important services. Some of the most widespread and significant services birds provide are population control through predation, pollination through interaction with plants, and seed dispersion by consumption of fruits (Whelan et al. 2008). Because of human activity and habitat destruction, the number of bird species has seen a decline. Currently, the percentage of bird species vulnerable or under threat of extinction within Europe is about 13% (European Commission. Directorate General for Environment. et al. 2022). The decline of birds negatively impacts the ecosystem they inhabit because of the important services they provide, as mentioned above. Therefore, it is essential to monitor the population of different bird species to know when measures for species preservation must be taken.

The monitoring of bird species is not only used for inter-species health assessment but also general environmental health assessment. The monitoring of different bird species within an ecosystem can be used for assessing the health of the ecosystem (Kahl, Wood, et al. 2021). Birds are found living in a variety of ecological habitats across the earth as carnivores, herbivores, and omnivores. Ornithology is also an extensively studied subject of science, making the knowledge of bird ecology large compared to many other biological taxa (Koskimies 1989). These factors make birds a good candidate for being an indicator species for monitoring ecosystem health.

In recent years there have been multiple competitions for autonomous bird classification with the use of bird vocalization to try and further the progression of machine learning within bird vocalization classification and detection. These competitions have been hosted on websites such as Kaggle and ImageClef for the public to participate. New advances and architectures within deep learning have made neural network approaches the leading method for solving these tasks.

Some challenges of using deep learning for bird vocalization classification are the need for large labeled datasets and the handling of unbalanced datasets. For certain bird species, there are a vast number of recordings available, some having multiple hours of recordings, while other bird species only have minimal audio data. Many competition participants try to overcome these challenges by using pre-trained models (Kahl, Denton, et al. 2021). The pre-training for these models is usually done on large image datasets and not bird audio, which begs the question if pre-training on bird audio would be more beneficial for the models.

Large-scale supervised learning on bird audio is challenging, as most of the data available is unlabeled or weakly labeled, where one does not know what bird can be heard in what segment of the audio recording. With the new progress of self-supervised learning, this hurdle can be overcome as one can pre-train a model on unlabeled data. This pre-training method has shown promise in computer vision with implementations such as, SimSam (X. Chen and He 2020) and SimCLR (T. Chen et al. 2020), performing equally or close to its supervised counterpart. Self-supervised learning has not been extensively tested on the bird vocalization classification domain. Therefore, it is interesting to see how well models self-supervised on bird vocalization compare to models pre-trained on images for the task of bird vocalization classification.

1.2 Goals and Research Questions

This section states the goal and research questions that drive this master thesis.

Goal

The thesis goal is to investigate the effectiveness of self-supervised learning on the bird vocalization classification problem.

To further specify the means to achieve the goal, two research questions to be explored have been formulated.

Research question 1

What augmentations for the self-supervised model give the best results?

Research question 2

How does the self-supervised model compare against a supervised model pre-trained on ImageNet, with linear fine-tuning and end-to-end tuning on different amounts of training data?

1.3 Research Method

To explore the two research questions, a self-supervised learning scheme will be implemented to pre-train a model on an unlabeled dataset of bird vocalizations. Different combinations of data augmentations during the self-supervised training will be tested to find what augmentations produce the best performing model. This model will then be compared to a model pre-trained on the image dataset ImageNet by linearly fine-tuning and end-to-end tuning each model on a smaller labeled dataset of bird vocalizations.

To evaluate the self-supervised model, the model will be fine-tuned on a smaller, manually labeled dataset of bird vocalizations. To evaluate the data augmentations, the fine-tuned model will be run on an unseen dataset where the top-1 accuracy for each model will be calculated. For the comparison with a model pre-trained on ImageNet, the models will be run on an unseen dataset where the top-1 accuracy will be calculated, and the models' ability to cluster embeddings will be visualized.

1.4 Contributions

This thesis aims to contribute to the field of bird vocalization classification by investigating the use of self-supervised learning on unlabeled bird vocalizations for the downstream task of fine-tuning a bird vocalization classifier. This thesis will compare different combinations of data augmentations for the self-supervised learning scheme and test if models using self-supervision on bird vocalization can produce better results than models using supervised pre-training on images. To the best of the author's knowledge, such comparisons have not been made within the field of bird vocalization classification.

1.5 Thesis Structure

The master thesis is partitioned into six chapters. Chapter 1 introduces the topic of bird vocalization classification and lays out the objective of the master thesis. Chapter 2 presents the background information relevant to the thesis regarding bird vocalization, convolutional neural networks, self-supervision, and audio representation. Chapter 3 gives an overview of earlier papers and contributions to the field of bird vocalization classification and self-supervision. Chapter 4 goes through the methodology used in this thesis and describes how experiments will be conducted and how the results will be produced. Chapter 5 presents the results from the conducted experiments and discusses these findings in light of previous work on self-supervision and bird vocalization classification. Lastly, chapter 6 summarizes the work done in this thesis and discusses the key implication of the results with respect to the general research questions.

Chapter 2

Background Theory

Large parts of this chapter were taken from my project report (Norum 2022). This chapter covers the background information needed to understand the topics discussed in this thesis. Section 2.1 introduces the function of bird vocalization. Section 2.2 covers the general workings of the convolutional neural network. Section 2.3 introduces the concept of data augmentation when training machine learning algorithms. In section 2.4, self-supervised learning and contrastive learning are introduced. Section 2.5 introduces digital audio representations and how audio inputs to convolutional neural networks are generated. Lastly, an overview of the performance metrics used in this thesis to evaluate deep learning models is given in section 2.6.

2.1 Bird Vocalization

When doing machine learning tasks, it is beneficial to have some basic domain knowledge to base decisions on. Domain knowledge is often useful when examining the collected data and designing the pre-processing pipeline. While many species of birds have different vocalizations, this section will introduce general bird vocalization and its general characteristics.

Birds produce vocalizations through the syrinx. In birds, the syrinx is situated at the lower end of the trachea, splitting into two bronchi as depicted in figure 2.1. The vocalizations are generated by air flowing through the syrinx, making the tympanic membranes on the walls of the bronchus vibrate (Kumar 2003). The syrinx can vary among bird species, but its general anatomy stays consistent. Having two channels with tympanic membranes allows birds to produce sounds individually through each channel, giving increased vocalization complexity.

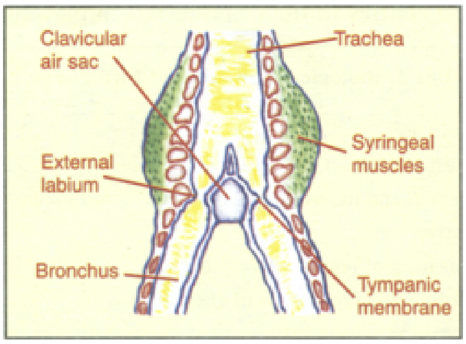


Figure 2.1: Depiction of the syrinx taken from (Kumar 2003).

Birds can be grouped into two subgroups. Songbirds and non-songbirds. The songbirds have a more complex syrinx with more muscles compared to the group of non-songbirds (Kumar 2003). Songbirds are vocal and have complex vocalizations. Some non-songbirds are also vocal but often with less complexity in their vocalizations. Bird species that use vocalization often have more than one distinct vocalization. These vocalizations vary from species to species, but the range of bird vocalization across species lies between 250 Hz and 8.3 kHz (Kahl 2020).

Bird vocalization can generally be divided into two groups, songs, and calls. Songs tend to be more complex than calls and are used to attract potential mates and agnostic signaling (Okanoya 2008). Bird songs consist of an ordered sequence of discrete elements called notes. A grouping of notes is called a syllable, and a group of these syllables is called a phrase.

Calls are shorter than songs and used for signaling. Birds usually have several types of calls, which are classified based on the context they are used in, such as distress, roosting, and begging (ibid.).

As mentioned in section 1.1, monitoring birds is important because of their contribution to their ecosystem and for being an indicator species. The monitoring of birds in an ecosystem is often done by point count surveys. Point count observations are limited to the area and time where the observers are and are, therefore, limited in their observations and costly (Clough 2020). Since birds are vocal animals, with the syrinx giving most species distinct calls and songs, recent trends of using autonomous recording units for remote monitoring of birds have taken place (Bobay et al. 2018). This eliminates the need for physical point count surveys but still requires hours of listening to recordings. Because of the large time consumption of manual bird surveying, an interest within the birding and artificial intelligence community has risen to try and use machine learning on bird vocalization for classification and labeling.

2.2 Convolutional Neural Networks

As mentioned in section 1.1, deep learning methods using neural networks are the approaches showing the most promise in the field of bird vocalization detection and classification. Of the deep learning architectures, the most commonly used architecture in competitions is the convolutional neural network. This architecture is often seen used among the top-ranking competitors on both Kaggle and BirdClef.

The convolutional neural network (CNN) is a subcategory of the artificial neural network. The general architecture of the convolutional neural network has taken inspiration from findings in how the primary visual cortex process visual stimuli (Lindsay 2021). CNNs have shown great results in the task of image classification and object detection, and are commonly used within computer vision. There are several different implementations of the CNN, but they all usually utilize three main components, the convolutional layer, the pooling layer, and the fully connected layer. Figure 2.2 shows a common convolutional neural network architecture layout starting with a convolutional layer followed by a sequence of alternating pooling layers and convolutional layers, before ending in a sequence of fully connected layers.

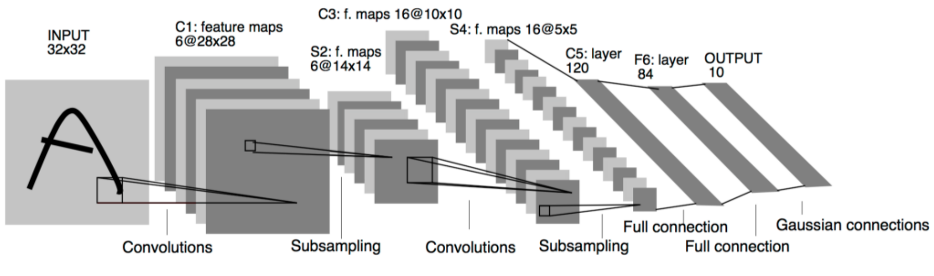


Figure 2.2: Overview of a general CNN architecture with a combination of convolutional layers, pooling layers and fully connected layers (Koushik 2016).

2.2.1 Convolutional Layer

The convolutional layer is the foundation of the convolutional neural network and plays a key part in extracting information and features from the input for later use in the fully connected layers. As mentioned, a convolutional layer can extract simple features. By having several convolutional layers downstream, the network can use previous convolutional layer outputs to build more complex and semantically meaningful features (Wu 2017).

The way these convolutional layers extract features from their inputs is with the use of filters called kernels. These kernels can be of any specified shape and slide over the input to calculate the output as the matrix product between the kernel and the current section the kernel is over. The movement of the kernel can be given by the stride, which tells the kernel how many rows and columns to move before doing a new calculation. Usually, each convolutional layer has several kernels. The kernel values are usually learned by the network. Therefore, there is no need to explicitly set these weights since the network finds suitable filters during training. The output shape of these layers is (x, y, number of kernels) where x and y are dependent on the shape and stride of the kernel. Figure 2.3 shows a simplified example of how the output of a convolutional layer is calculated.

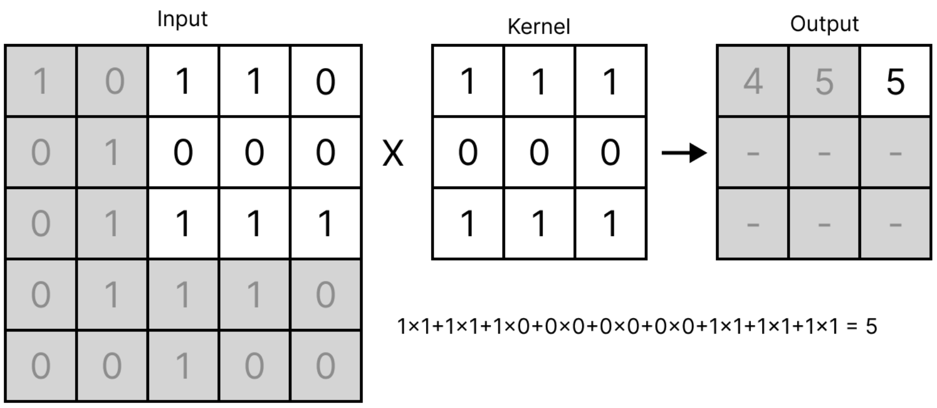


Figure 2.3: Simplified example of computing output of convolutional layer with kernel size equal to 3 x 3 and stride equal to 1 x 1.

2.2.2 Pooling Layer

The convolutional layers are often followed by pooling layers. The objective of the pooling layer is to reduce the dimensionality of the representation, reducing the computational complexity of the network (O'Shea and Nash 2015).

Like the convolutional kernel, the pooling layer uses a sliding window with a size and a stride parameter to calculate output values over its inputs. There are different methods for pooling, where max pooling and average pooling are the most common. During max pooling, the highest value within the window is chosen as the output, while all the other values are discarded. In average pooling, the average of all the input values within the window is computed and used as the output. Figure 2.4 illustrates how max pooling is done. Because the pooling method is given by the network architecture there are no trainable features in the pooling layers.

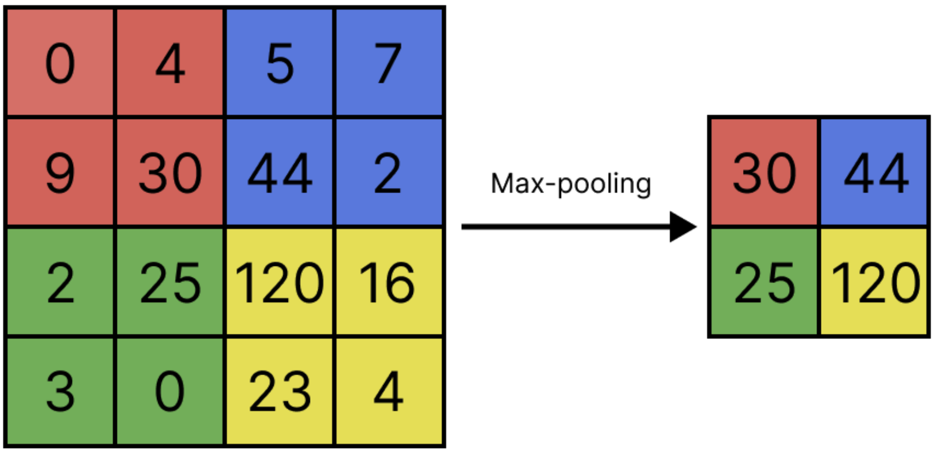


Figure 2.4: Example of max pooling with kernel size equal to 2 x 2 and stride equal to 2 x 2.

Since the pooling layer reduces the dimensionality of the input, some of the information from the input is lost. Using a large kernel size can lead to poor results as too much information is lost. Therefore, it is normal to use a small kernel size during pooling (ibid.). The use of overlapping sliding windows can also be used during this step.

2.2.3 Fully Connected Layer

After the convolutional and pooling layers, the output is usually flattened and sent through the fully connected layers. These fully connected layers are equal to the standard neural network, where all neurons in a layer are connected to all other neurons in the previous layer and the next layer. If the CNN is used in classification, the fully connected layers use the feature representation extracted from the previous convolutional layers to classify the input, where the size of the last fully connected layer is equal to the number of classes.

2.3 Data Augmentation

Data augmentation is a technique to increase the size of the training dataset without adding new data or labels to the dataset. This is done by taking instances from the collected dataset and augmenting some aspects of the data while giving the new augmented data the same label as the original.

Data augmentation is often used to combat overfitting on smaller datasets. Overfitting is when the model learns structures specific to the training dataset that are not representative of the general data it tries to learn. This leads to the model performing worse on data that was not in the training dataset. Overfitting can be observed during training when the loss on the training set decreases while the loss on the validation set increases.

Using data augmentation while training will make the smaller dataset represent more data points for what could be encountered later in unseen datasets, and lowers the possibility of the model learning dataset-specific structures. It can also be used to decrease the impact of unbalanced datasets by augmenting instances of classes with fewer instances to achieve a more balanced dataset.

For image recognition with convolutional neural networks data augmentations is often visual, such as cropping an image and resizing it to the original size, adding noise to the image, or augmenting the color distribution of the image. Several augmentations are often done on the same instance.

2.4 Self-Supervised Learning

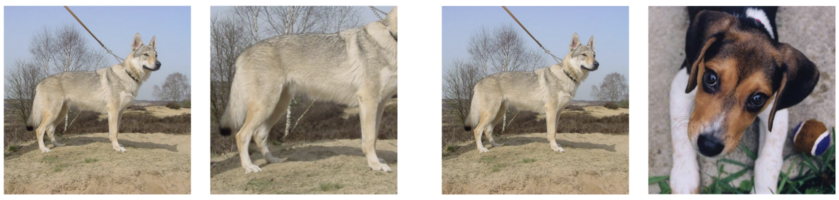
Most bird vocalization datasets, such as Xeno-canto (*xeno-canto* n.d.), are comprised of weakly labeled data. Often in these audio datasets, each audio file has a bird species label, but it is not specified where in the audio file the bird is. This can make it hard to do supervised learning as one has to manually go through the audio files to retrieve the bird vocalizations of the labeled bird. These audio files can often be several minutes long which leads to a time-consuming manual data preparation step. There is also a big discrepancy in the number of audio samples in these open bird vocalization datasets, with some bird species having several hundred samples, while others only have a couple. Therefore, one might not have enough data on certain species to train a supervised learning model adequately. In these circumstances, self-supervised learning can be applied.

The idea behind self-supervised learning is to leverage large amounts of unlabeled data by making a training scheme that does not require labels, called the pretext task. The pretext task uses the unlabeled data to train a model to extract useful features from the data given. After the pretext task is done, the model can then be trained on a smaller labeled dataset, called the downstream task, requiring less data and training time.

There are many different methods for constructing a self-supervised pretext task, but in recent years there have been made several advances in contrastive learning and non-contrastive learning for computer vision, with frameworks such as SimCLR and MoCo performing close to their supervised counterparts (T. Chen et al. 2020) (He, Fan, et al. 2019). Contrastive learning is a specific method for constructing a self-supervised pretext task where an encoder model is given pairs of positive and negative samples. The goal of the encoder is to generate embeddings where the positive samples are close to each other and the negative samples are far apart (Jaiswal et al. 2020). To measure the distance between the two embeddings a similarity metric, such as cosine similarity, is used. In computer vision, the positive pair is often derived from augmenting an image to produce two different views from the same ground image, while the negative pairs are derived from two different images from the unlabeled dataset. Figure 2.5 gives an example of a positive and negative pair of images. These positive and negative pairs are often passed through a siamese network architecture, depicted in figure 2.6, where each branch is given one image from the pair. In each branch, the encoder tries to produce embeddings that are far apart or close to each other depending on the type of image pair. A similarity metric is used to produce a loss from the embeddings that can then be used to update the weights of the neural network.

Non-contrastive learning is almost identical to contrastive learning, only having three minor differences. The first difference is that non-contrastive learning does not utilize negative pairs, using only positive pairs during the self-supervised training. The second difference is that non-contrastive models use an extra predictor module at the end of one of the branches. The last difference is that the network only propagates the calculated gradients through the branch with the predictor module, as depicted in figure 2.7.

When using contrastive learning or non-contrastive learning for pre-training, there is a possibility for the training scheme to produce a collapsed solution. A collapsed solution means that the model produces the same output regardless of the input. When the contrastive learning scheme collapses the encoder is not able to learn good representations of the input, and the performance of the model on the downstream task will decrease.



(a) A positive pair of images. (b) A negative pair of images.

Figure 2.5: Example of negative and positive pairs (images taken from the Standford Dogs dataset (Khosla et al. 2011)).

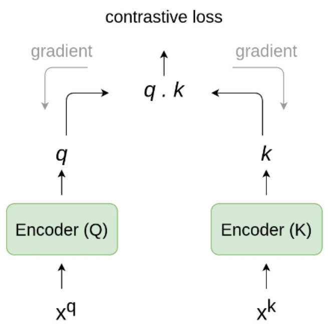


Figure 2.6: Contrastive learning where x^q and x^k are a pair of images going through separate branches of the network. (Jaiswal et al. 2020).

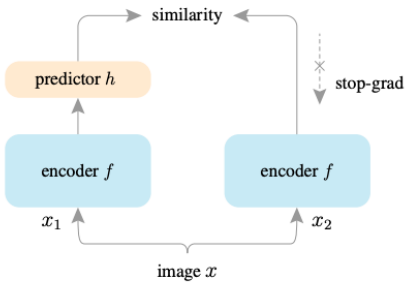


Figure 2.7: Non-contrastive learning that only uses positive pairs, a predictor, and a stop-gradient on one of the branches (X. Chen and He 2020).

The augmentations that are done during the pretext task greatly influence the model's performance on the downstream task, as the model will learn to ignore the applied augmentations when learning the underlying structure. Therefore, it is important to pick augmentations that do not impede the model's ability to learn good embeddings. For example, if one is using contrastive learning for the downstream task of classifying healthy and sick leaves, one should probably not use grayscale as an augmentation, as the color of the leaf could be an important feature for classifying a leaf as healthy or sick.

The purpose of self-supervised learning is always to apply the trained model on a downstream task. One can think of self-supervised training as pre-training a model using unlabeled data, for then to use the encoder part of the pre-trained model to further train on a smaller labeled dataset, as depicted in figure 2.8. By having learned the underlying structures of the unlabeled dataset, learning on a labeled dataset will be quicker requiring a smaller dataset to get the same performance as a larger labeled dataset.

A common way of applying a self-supervised model on the downstream task is either to add a new classification head at the end of the encoder model and freeze the encoder model weights, training only the last layers on the labeled dataset, or add the classification head and fine-tuning the whole model on the labeled dataset.

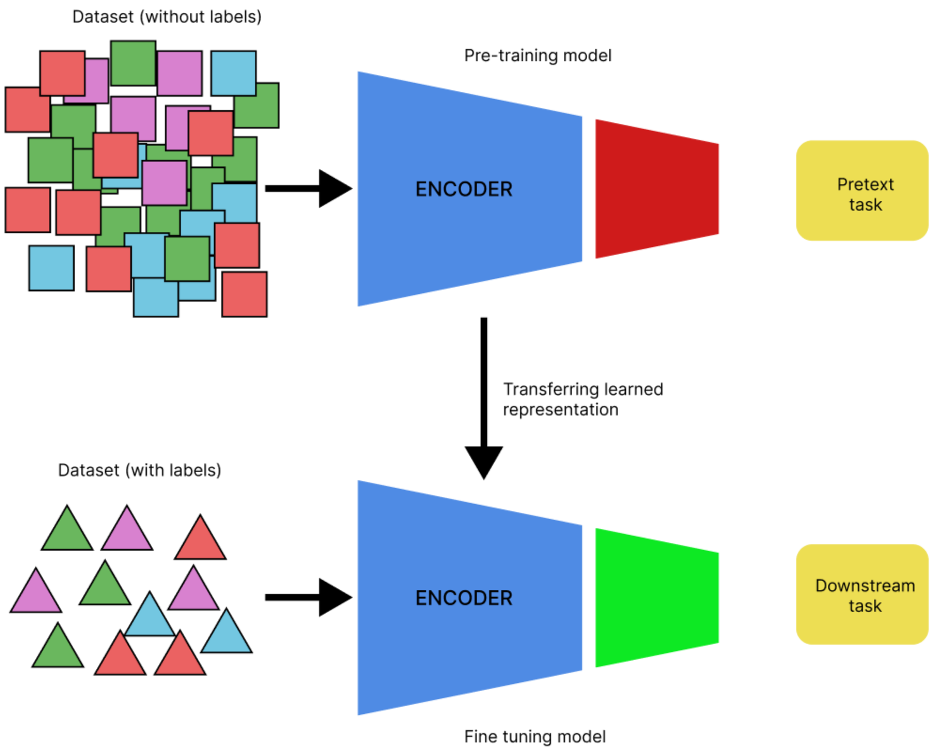


Figure 2.8: Pre-trained self-supervised model used on a downstream tasks.

2.5 Audio Representation

This thesis is concerned with contributing to the ongoing research on bird vocalization classification and detection. Seeing that the media used in this field of research is audio, it is important to discuss and explore what the best representations of these vocalizations are, as the performance of a neural network is highly dependent on the quality of the input and training data.

Sound is constructed by the propagation of pressure differentials through a medium, such as air. The different frequencies of the pressure differentials and the difference of the high- and low-pressure zones, determine what we perceive as different tones, sounds, and loudness. All these features of sound are continuous. Since computers and neural networks are not able to use continuous values, a discrete representation for these sounds must be presented as inputs to the neural network.

2.5.1 Digital Audio

Most of the publicly available datasets of bird vocalization are in the format of digital audio, either as .wav or .mp3 files, where .mp3 is compressed while .wav is not. The way these files store the information from the original audio is by sampling the original sound wave at a fixed sampling rate within a given bit depth. From these files, the audio information can be fetched using tools for audio extraction. The resulting output is often a one-dimensional array of length $sr \cdot s$, where sr is the sampling rate and s is the duration of the audio in seconds. This array contains a time series representing the amplitude of the waveform of the different sample instances.

Audio files from bird vocalization datasets can have different durations, all from a few seconds to several minutes. To make this research simpler it is a good idea to decide on a fixed duration for the model's input. Choosing the correct duration for the audio input is not a simple task, as different birds might have different durations for their vocalizations. Having the input be too short might lead to longer bird vocalizations being cut early, losing out on valuable information for the classification task. Making the input longer makes the model larger and can lead to unnecessary noise that does not help with classification or detection. Many previous papers covering the topic of bird vocalization classification use audio chunks from 3 seconds to 7 seconds, which suggests using audio of duration much greater than 7 seconds does not lead to improved results (Kahl 2020) (Y. Zhang et al. 2022) (Yan et al. 2021).

It is not only the duration of the audio segment that affects the input size. As mentioned, the sample rate of the audio also affects the length of the array fetched. Therefore, it is important that all the audio files are re-sampled to the same sample rate, to give the audio segments of fixed duration the same-sized array. The decision of the fixed sample rate is also an important one. Because of the Nyquist-Shannon sampling theorem (Shannon 1949), the choice of sampling rate will affect the highest frequency captured from the audio. This frequency is called the Nyquist frequency and is half the sampling rate.

These decisions of audio duration and sampling rate are important decisions and need to be chosen with care. Using the resulting time series as input is often not sufficient for getting good results. One issue is the length of the array. One audio segment at 7 seconds sampled at a low sample rate of 16,000 will produce a time series array of length 112,000. Therefore, it is usual to take this generated array and turn it into a different representation of the audio.

2.5.2 Spectrograms

For general audio classification and detection in deep learning, it is normal to use a spectrogram as input to the model. This is also a popular representation used in the bird vocalization classification field. The spectrogram changes the audio representation from a one-dimensional array to a two-dimensional array and makes it more visually comprehensible, as seen in figure 2.9. This visual representation also makes it easier to sift through a dataset to get a better understanding of the general traits of the audio samples.

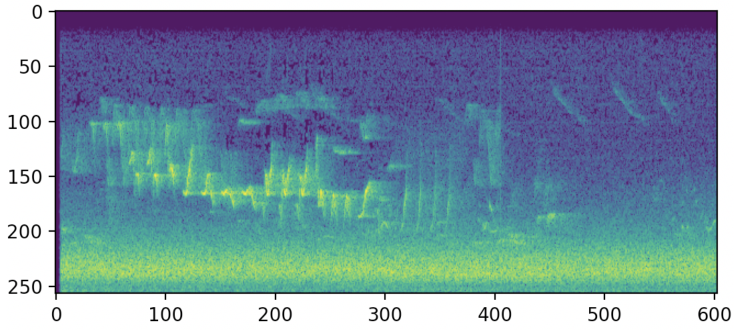


Figure 2.9: Spectrogram of a 7-second audio clip containing the vocalization of a Willow Warbler.

The spectrogram is a time-frequency representation of the audio waveform, where the x-axis is time, and the y-axis is frequency. The conversion from waveform to a frequency representation is done with the Fourier transform, shown in equation 2.1 (Oppenheim 1970). This equation transforms the waveform from the time domain to the frequency domain, where t is time and w is the frequency, and the output $X(w)$ is a complex number where the real component is the magnitude of the frequency w, while the complex component is the phase of the frequency.

$$X(w) = \int_{-\infty}^{\infty} x(t)e^{-i2\pi wt}dt \tag{2.1}$$

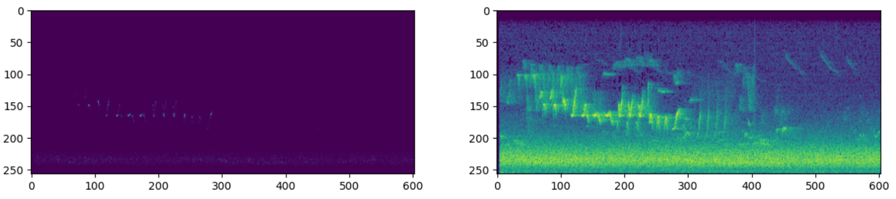
The Fourier transform in equation 2.1 is for continuous time. As digital audio uses discrete time, the use of the discrete Fourier transform is used instead, shown in equation 2.2 (Duraisamy 2001). Here frequencies are not continuous but separated into bins, where N is the length of the array containing the amplitude values, x_n is the amplitude at position n, and k is the frequency bin for the computed value. The number of bins is the sampling frequency divided by N.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \tag{2.2}$$

The discrete Fourier transform gives only information about the frequencies of the waveform. To get information about both time and frequency over discrete time, as spectrograms do, the discrete short-time Fourier formula is used. The discrete short-time Fourier transform, shown in equation 2.3, takes the discrete Fourier transform over separate parts of the sampled waveform. This is done with a sliding window, and the discrete Fourier transform is calculated within the window (Mertins 1999). Here N is the frame size of the window, and H is the hop size deciding how much the window slides per step. $w(n)$ is the sliding window function which is often a Hann window. The Hann function is a smoothing function making the signal inside the window periodic. This lowers spectral leakage as discontinuities in the window results in additional frequencies (*Leakage* n.d.). This equation will give us both information of time and frequency as the position of the sliding window changes.

$$X(m,k) = \sum_{n=0}^{N-1} x(n+mH)w(n)e^{-i2\pi kn/N} \tag{2.3}$$

When computing the discrete short-time Fourier transform over the different time steps the resulting output is a two-dimensional matrix of complex values, where frequency bins are on the y-axis and time, or time steps, are on the x-axis. To get the real-valued magnitude of the different frequency bins, the absolute value is computed on the values in the matrix $|X(m,k)|$. If these values are squared the resulting matrix can then be displayed as a power spectrogram. This power spectrogram has large distances between the high values and low values of the matrix and is not representative of how we hear different noise levels (Foley and Matlin 2015). To make the spectrogram more representative of human hearing it is normal to apply a logarithmic function to the matrix values, seen in figure 2.10.



(a) Before applying logarithmic function. (b) After applying logarithmic function.

Figure 2.10: Applying a logarithmic function to the spectrogram.

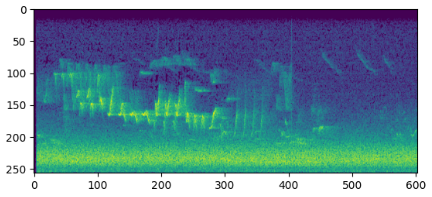
When using the discrete short-time Fourier transform there is a trade-off between frequency resolution and time resolution that must be taken into consideration. The number of frequency bins for the spectrogram is given by the frame size, as seen in equation 2.4. While the number of frames (or time steps) is given by equation 2.5. The larger the frame size is the larger the number of frequency bins get, but the smaller the number of frames will be. It is possible to adjust for this by decreasing the hop size, but then the spectrogram will have more overlap of the previous time steps. Figure 2.11 shows different configurations of frame length and hop size.

$$number\ of\ bins = \left\lfloor \frac{frame\ size}{2} \right\rfloor + 1 \tag{2.4}$$

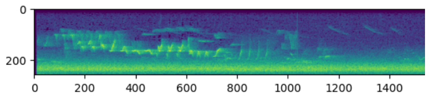
$$number\ of\ frames = \left\lfloor \frac{samples - frame\ size}{hop\ size} \right\rfloor + 1 \tag{2.5}$$

The spectrograms this paper is most interested in will be generated from the vocalizations of birds. Most birds can hear frequencies spanning from 500 to 6000 Hz, where hearing outside of this range decreases rapidly (Okanoya 2008). Birds also have a high sensitivity to frequency changes, with studies showing some birds are capable of discriminating a change of 1% in frequency (Kahl

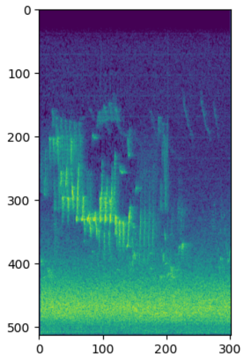
2020). This ability in birds to discern frequencies, points towards that having a high-frequency resolution in spectrograms can be beneficial for classification. Bird songs are usually made up of a sequence of discrete vocalizations called notes (Okanoya 2008). These pauses between notes tend to be small. To be able to get a good representation of the different notes in a spectrogram, the time resolution on the spectrogram should be high. This leads to a difficult decision where a compromise between the frequency resolution and time resolution must be made.



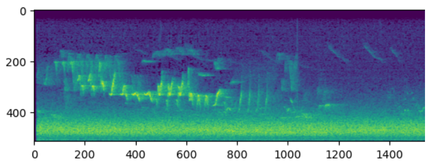
(a) Frame size 512 and hop size 256.



(b) Frame size 512 hop size 100.



(c) Fame size 1024 hop size 512.

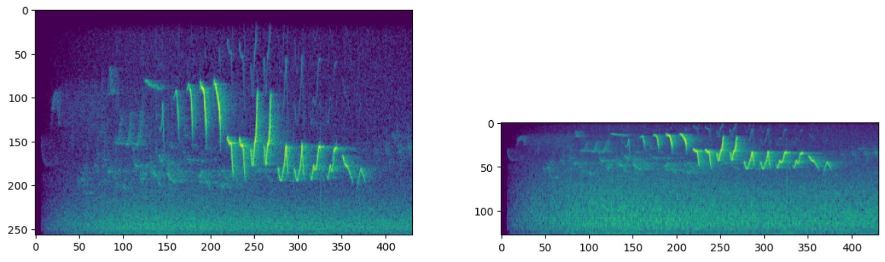


(d) Frame size 1024 hop size 100.

Figure 2.11: Spectrograms of the same audio recording with different frame sizes and hop sizes.

In addition to the spectrogram, the mel-spectrogram is a popular choice for audio representation. The spectrogram depicts frequencies on a linear scale. Humans do not perceive changes in frequencies as linear, having more of a logarithmic perception for higher frequencies (Kahl 2020). To reflect the human perception of frequencies the mel scale is used. The mel scale transforms the frequency scale to a logarithmic scale using equation 2.6. When applied to a spectrogram one gets the mel-spectrogram, shown in figure 2.12, which can also be used in audio classification tasks.

$$mel = 2595 \cdot log_{10}(1 + \frac{frequency}{700}) \tag{2.6}$$



(a) Spectrogram. (b) Mel-spectrogram.

Figure 2.12: Comparison of spectrogram and mel-spectrogram.

2.6 Performance Metrics

In order to compare and evaluate different deep learning models, it is important to have good performance metrics. Given that this paper is concerned with the performance of self-supervised models trying to identify different bird species based on their vocalizations, metrics for classification will be used.

When doing classification, there are four outcomes a classification can be grouped into, which is depicted in table 2.1. The table depicts a binary classification task. If the predicted class is true and the ground truth was true, the model produced a true positive (TP). If the model predicted true and the ground truth was false, the model produced a false positive (FP). If the model predicted false and the ground truth was false, the model produced a true negative (TN). Lastly, if the model predicted false, but the ground truth was true, the model produced a false negative (FN). For a multi-class classification task, one only looks at one class at a time in order to count the size of each group with the help of a confusion matrix.

Table 2.1: Possible outcomes of binary classifications.

		Ground truth	
		True	False
Predicted	True	TP	FP
	False	FN	TN

2.6.1 Accuracy

Accuracy is a metric calculated over all classes and looks at the overall classification accuracy of the model. This is done by creating a confusion matrix, summing the diagonal, and dividing by all cases. When calculating accuracy, it is usual to calculate top-1 accuracy, where in order to be counted as a true positive the correct class must be the top suggestion of the model. This is in contrast to top-5 accuracy where a true positive is counted if the correct class is among the top 5 suggestions of the model.

2.6.2 Linear Evaluation

When testing the performance of a self-supervised model, it is normal to use a linear evaluation on the pre-trained model. During the linear evaluation, the model weights are frozen and linearly fine-tuned by appending a linear layer to the head of the model and training it on a downstream dataset. After the linear fine-tuning is done, the encoder model with the linear layer appended to it is tested on an unseen validation set, and the accuracy of the model is calculated.

2.7 Summary

This chapter introduced the basics of bird vocalizations, describing how bird vocalizations can be classified into songs and calls, and the tendency for vocalizations to lie between 250 Hz and 8.3 kHz. This chapter has also given an introduction to convolutional neural networks and audio representations, describing how audio can be represented as two-dimensional spectrograms with the Fourier transform, making it possible to use convolutional neural networks for classification tasks on audio recordings. Self-supervised learning has also been presented, describing how contrastive and non-contrastive learning can be used to pre-train a neural network model with the use of large unlabeled datasets, for then to be fine-tuned on a smaller dataset with labels. Lastly, a method of evaluating classification models has been presented.

Chapter 3

Related Work

Large parts of this chapter were taken from my project report (Norum 2022). Bird vocalization classification is a popular machine learning topic, having been the focus of many public competitions over the last few years. As a consequence of its popularity, several papers have been written, contributing to the domain. This chapter will describe how the research on bird vocalization classification and self-supervision was conducted. It will introduce and compare different papers related to this research to give the reader a better view of the current landscape of bird vocalization classification and self-supervision. It is important to note that one can not directly compare the performance of architectures against each other, as different datasets were used. In section 3.1, a description of the research conducted to gain domain knowledge is described. Section 3.2 gives an overview of the current state-of-the-art within bird vocalization classification. In Section 3.3, previous solutions from bird vocalization classification competitions are presented. Section 3.4 gives examples of different self-supervised learning applications related to this research. Lastly, section 3.5 describes the use of transfer learning on bird vocalization classification in other related papers.

3.1 Structured Literature Review

When researching and reading about new topics, it is important to be structured. Structuring and planning the research makes it easier to find relevant papers, as well as to avoid wasting time sifting through non-relevant topics.

3.1.1 Search Process

To get an introduction to the topic of bird vocalization classification, the description of the Kaggle competition introducing this master's was read (*Cornell Birdcall Identification* n.d.). The Kaggle description gave insight into the motivation for the task of bird vocalization classification, and how the raw data of bird audio were handled. The Kaggle site contained links and acknowledgments to datasets and other similar competitions, such as the BirdClef competition (*BirdClef competition 2021* n.d.).

After reading the Kaggle competition an extensive literature search was performed. The literature search started first with one query, "bird song classification", to serve as test search terms to get a couple of relevant articles regarding the research topic. These search terms were used on the Google Scholar and Google search engines. The papers discovered through this search gave the author a deeper insight into the research topic through citations to other papers and a broader vocabulary that was then used to construct several search queries. These search queries were,"bird song detection", "audio classification", "bird acoustics", "bird vocalization classification", "bird sound classification", "avian acoustics classification", and "avian audio classification".

Other bird vocalization classification competitions were also researched to get a better understanding of the current deep learning methods used. The BirdClef competition was shown to have been held several times in previous years, where for each competition the organizers released a report on the outcome of the contest and cited papers of the top solutions. These reports from the competitions were read and the papers with the most relevant solutions were also read.

With the information gained from the first period of literature research, the author was able to find new possible research topics within bird vocalization classification that could be of interest. After exploring different possibilities, the author landed on the topic of self-supervised learning for bird vocalization classification. After deciding on the topic, another round of literature search was done, focusing on the intersection between audio classification and self-supervised learning. To start with, the author read papers about self-supervised learning previously handed out in another course, before searching for new papers. The search queries used to find relevant papers during this second period of literature research were, "self-supervised learning", "self-supervised audio classification", "self-supervised bird song classification", "self-supervised bird vocalization classification", "pre-training bird song classification" and "unsupervised audio classification". These search terms were used on the Google Scholar and Google search engines.

After the last round of literature research was performed, enough domain knowledge was gained to formulate a research goal and research questions that could be seen as a contribution to the topic of bird vocalization classification.

3.1.2 Selection Criteria

During the literature search, only articles with known authors were taken into account. The papers had to be published with credible results, and a good description of the methods used in the papers.

Since the field of bird vocalization in deep learning is in constant evolution, more recent papers were preferred, whereas all papers before 2015 would not be considered. For self-supervision, this cut-off was even stricter, as there is only in recent years that self-supervised models have shown results close to their supervised counterparts. Therefore, all papers published before 2018 were not considered.

Because of the large number of contributions to the field of bird vocalization classification, only the most relevant papers were taken into consideration. Therefore, papers regarding convolutional neural networks on bird vocalization classification were prioritized. The same holds for papers regarding self-supervision, where papers of self-supervision on the audio domain were prioritized.

3.2 State of the Art Bird Vocalization Classification

One of the largest contributions to the field of bird vocalization classification using deep learning is written by one of the organizers of the BirdClef competitions. Stefan Kahl developed a research platform named BirdNet, using expert domain knowledge. Kahl trained on a dataset containing 987 different classes and 80,000 distinct vocalizations, which is described in the paper Identifying Birds by Sound: Large-scale Acoustic Event Recognition for Avian Activity Monitoring (Kahl 2020).

To create a dataset of bird vocalizations, Kahl acquired data from the publicly available datasets Xeno-canto, eBird, and Macaualay Library (ibid.). To decide what species would be featured in the dataset he curated a list of the 595 bird species in America considered to be vocal and likely to be observed during monitoring. A list of common European bird species was also curated through data from the eBird project. Due to large differences in incidences per class in the Xeno-canto dataset, the max number of cases per species downloaded was set to

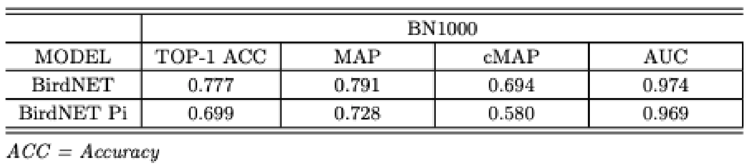
250 and a lower threshold of 10 species per class was also implemented to avoid data bias in the model. Non-bird vocalizations were also added to the dataset, with audio from the Google AudioSet and FreeField, containing audio such as humans, locomotion, and insects, being added to the dataset.

As mentioned in section 2.5.2, spectrograms are a widely used representation in audio classification. Since there are many hyperparameters when creating spectrograms, such as audio duration and frame size, Kahl used domain knowledge to explore different parameters. Kahl tested three different audio durations, 2.0 seconds, 2.5 seconds, and 3.0 seconds. From these tests, he found that the 3-second spectrograms significantly improved the classification of single spectrogram prediction, while shorter durations led to better mono-species prediction in soundscapes recordings. Kahl also tested the trade-off between frequency and temporal resolution of spectrograms, discussed in section 2.5.2. Through empirical testing, the result showed that higher temporal resolution had some performance benefits and that these benefits were most noticeable in noisy environments. Kahl also concluded that temporal resolution is more important than frequency resolution (Kahl 2020). It is important to note that Kahl did not use standard spectrograms but a mel-like spectrogram with a self-defined mel frequency scale given by equation 3.1 with a cut-off at 150Hz and 15kHz and binning them into 64 mel bins. After empirical testing, Kahl ended up using 3 seconds of audio and a frame size of 512, resulting in a spectrogram shape of (64, 384).

$$f_{scaled} = 4581 \cdot \log_{10}(1 + \frac{f}{1750}) \tag{3.1}$$

For the architecture, Kahl uses a self-defined convolutional neural network based on the ResNet architecture. During testing, he concluded that deeper topologies perform better than shallow ones, but require more computing power. Therefore, two models were built. One shallow model for deployment on a raspberry pi, and one deeper and wider not intended for mobile use. While training the network, data augmentations such as frequency masking, time masking, adding noise, and distortion, were applied to the spectrograms. These data augmentations were used to train the model on more diversified data to make the model generalize better. When training the models, the training set contained a total of 1,727,234 spectrograms with a maximum of 3000 cases per class.

After training, the two resulting models were evaluated on a balanced test set with a total of 2,868 spectrograms where each class had a maximum of 3 cases. For mono-species classification, the larger model and the smaller model achieved a top-1 accuracy of 0.777 and 0.699 respectively, as seen in figure 3.1.



	BN1000			
MODEL	TOP-1 ACC	MAP	cMAP	AUC
BirdNET	0.777	0.791	0.694	0.974
BirdNET Pi	0.699	0.728	0.580	0.969

ACC = Accuracy

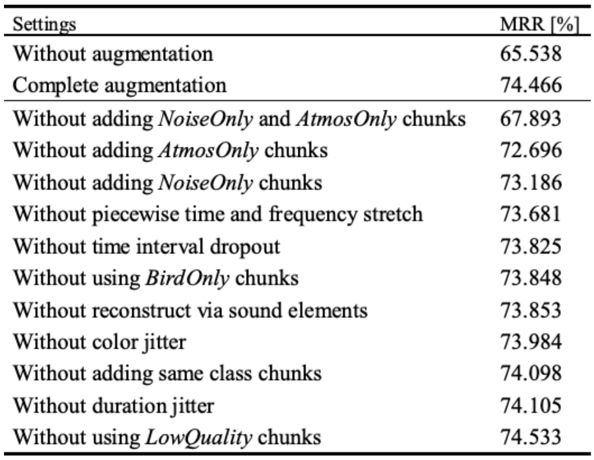
Figure 3.1: Evaluation of the two BirdNet models on the test set (Kahl 2020).

3.3 Bird Vocalization Competition

As mentioned in the introduction to this chapter there have been numerous open competitions on bird vocalization classification, and many of the competitors with the highest scores have published their methods and results through open publisher sites such as Ceur (*CEUR-WS* n.d.).

From the BirdClef competition of 2021, the first-place participants published their methods (Kahl, Denton, et al. 2021). BirdClef 2021 had the challenge of identifying all bird calls in soundscape recordings, where each recording was divided into 5-second audio clips (ibid.). The winners of the competition were Murakami et al. Their solution to the task was to use a CNN for bird audio prediction combined with a decision tree for the associated metadata of the audio to give a more powerful prediction (Murakami et al. 2021). In addition to the dataset provided by BirdClef, they used audio from freefield1010 to use as no-call training samples. The audio clips were converted to mel-spectrograms with 128 bins, using a frame size of 3200 samples and a hop size of 800 samples. The training of the classification system was separated into three parts. In the first part, a no-call detector was trained using the ResNeXt50 architecture. This model was then used in the second part to train a multi-label classifier. For training the multi-label classification model, ResNeSt50 was used. Here the labels were multiplied with the output of the no-call detector, as the audio recordings were weakly labeled and did not have any timestamp for the vocalization. For the third part, a decision tree was constructed, using the top 5 most likely bird species, given by the multi-label model, and audio clip metadata as input to classify the birds present. This combination of multiple deep learning models and a decision tree gave an F1 score of 0.6932 on the hidden dataset and first place in the competition.

In an earlier BirdClef competition from 2018, there were two tasks, mono-species prediction and soundscape multi-species prediction (Goëau et al. 2018). The dataset used in this competition was based on Xeno-canto and contained 1500 different bird species. The participant taking first place in both mono- and multi-species prediction tasks was Lasseck. Lasseck used ensemble CNNs where the networks were pre-trained on the ImageNet dataset. What stood out about Lasseck's submissions was the number of augmentations done during training. Lasseck used a combination of 15 different augmentations on the dataset (Lasseck 2018). Some of the augmentations were, adding background noise to the audio, stretching time and frequency intervals, dropping chunks of samples from the bird audio, and lowering audio quality. To assess the effects of different augmentations on performance, Lasseck compared a model using all augmentations with models where one augmentation was absent, seen in figure 3.2. From these tests, it was shown that almost all augmentations, except for lowering audio quality, had a positive effect on prediction accuracy. According to Lasseck, the data augmentations were found to combat overfitting and increase model accuracy.



Settings	MRR [%]
Without augmentation	65.538
Complete augmentation	74.466
Without adding *NoiseOnly* and *AtmosOnly* chunks	67.893
Without adding *AtmosOnly* chunks	72.696
Without adding *NoiseOnly* chunks	73.186
Without piecewise time and frequency stretch	73.681
Without time interval dropout	73.825
Without using *BirdOnly* chunks	73.848
Without reconstruct via sound elements	73.853
Without color jitter	73.984
Without adding same class chunks	74.098
Without duration jitter	74.105
Without using *LowQuality* chunks	74.533

Figure 3.2: Evaluation of different data augmentations (Lasseck 2018).

3.4 Self-Supervised Learning

One of the more recent works within self-supervised learning is the paper by DeepMind called Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning (Grill et al. 2020). In this paper, the researchers introduce a new self-supervised learning approach they call BYOL, which is used for image representation.

The researchers claim that BYOL surpasses its predecessors, like SimCLR and MoCo, while also avoiding using negative image pairs. When using self-supervised methods, it is essential to avoid a collapsed solution where all outputs of the model become equal regardless of the input. The way SimCLR and MoCo avoid this is by using contrastive learning with negative image pairs, as mentioned in section 2.4. BYOL does not use negative pairs of images and still avoids a collapsed solution (ibid.).

The researchers developing BYOL took inspiration from SimCLR and MoCo, and still use a siamese neural network architecture. BYOL uses an online network and a target network as shown in figure 3.3. The researchers built the target network up of three modules. The encoder module is a ResNet50 without the last classification layer. The projector module is a multi-layer perceptron with two layers of size 4092 and then 256. The predictor module is also a multi-layer perceptron, with the same architecture as the projector. The target network has the same architecture as the online network but does not implement the predictor module. To update the networks the researchers of the paper chose to use backpropagation for the online network, and momentum update for the target network. The equation for the momentum update can be seen in equation 3.2, where θ is the online network weights, ξ is the target network weights and τ is the momentum decay rate. The evaluation of the online network is done with a similarity loss function seen in equation 3.3. The authors describe that the purpose of this architecture is for the encoder to learn a good representation, y_θ seen in figure 3.3, that can later be used on a downstream task.

$$\xi = \xi\tau + \theta(1 - \tau) \tag{3.2}$$

$$L_{\theta,\xi} = 2 - 2 \cdot \frac{\left\langle q_\theta(z_\theta),\ z'_\xi \right\rangle}{\|q_\theta(z_\theta)\|_2 \ \cdot \ \|z'_\xi\|_2} \tag{3.3}$$

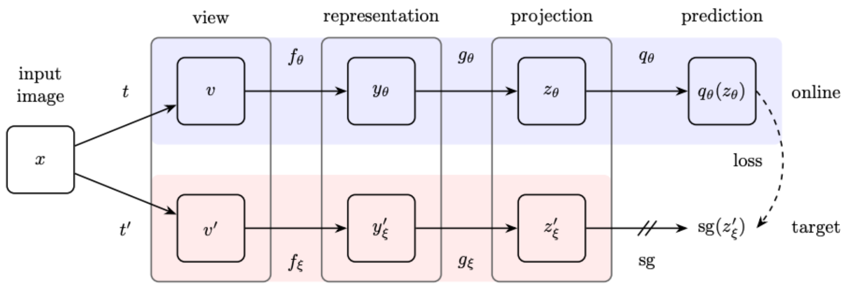
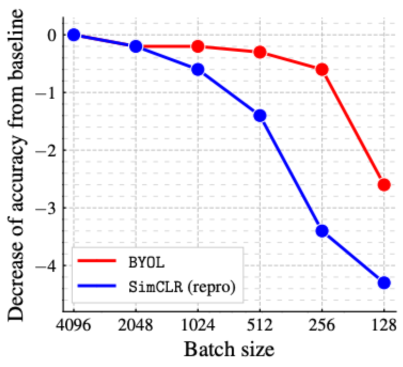


Figure 3.3: BYOL setup with encoder (f_θ), projector (g_θ), predictor (q_θ) (Grill et al. 2020).

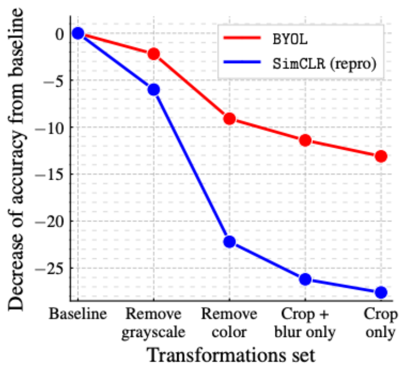
Even though BYOL does not use negative image pairs, it still avoids a collapsed solution. The authors of the papers hypothesize that the reason for no collapsed solution is that there is no loss $L_{\theta,\xi}$ that is jointly minimized over both the online network and the target network.

Through ablation studies of the BYOL architecture, the researchers found valuable information about the batch size, target network decay rate τ, and the impact of different augmentations. The researchers found that the batch size of BYOL had an impact on the performance, where higher batch size led to higher accuracy, but that the accuracy did not drastically decrease before a batch size of 256, as shown in figure 3.4. The researchers also found that BYOL is much less affected when removing image augmentations, than previous architectures as seen in figure 3.4.

To evaluate the performance of BYOL the authors used linear evaluation and fine-tuning of the whole model on ImageNet. For the linear evaluation, the researchers added a prediction head on top of the encoder, froze the encoder weights, and trained only the prediction head. For fine-tuning the whole model, the authors added a prediction head to the encoder and fine-tuned the whole model on the training set. The models' performance was measured on top-1 and top-5 accuracy, seen in figure 3.5. BYOL achieved a top-1 accuracy of 74.3% on ResNet50, with larger backbones scoring higher.

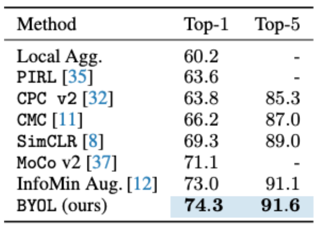


(a) Impact of batch size



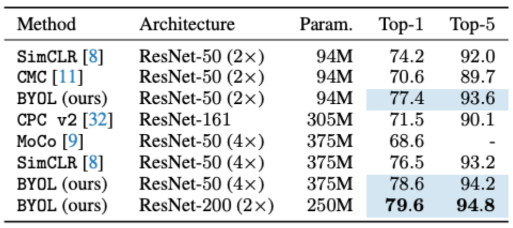
(b) Impact of progressively removing transformations

Figure 3.4: BYOL ablation study of batch size and image augmentation (Grill et al. 2020).



Method	Top-1	Top-5
Local Agg.	60.2	-
PIRL [35]	63.6	-
CPC v2 [32]	63.8	85.3
CMC [11]	66.2	87.0
SimCLR [8]	69.3	89.0
MoCo v2 [37]	71.1	-
InfoMin Aug. [12]	73.0	91.1
BYOL (ours)	**74.3**	**91.6**

(a) ResNet-50 encoder.



Method	Architecture	Param.	Top-1	Top-5
SimCLR [8]	ResNet-50 (2×)	94M	74.2	92.0
CMC [11]	ResNet-50 (2×)	94M	70.6	89.7
BYOL (ours)	ResNet-50 (2×)	94M	77.4	93.6
CPC v2 [32]	ResNet-161	305M	71.5	90.1
MoCo [9]	ResNet-50 (4×)	375M	68.6	-
SimCLR [8]	ResNet-50 (4×)	375M	76.5	93.2
BYOL (ours)	ResNet-50 (4×)	375M	78.6	94.2
BYOL (ours)	ResNet-200 (2×)	250M	**79.6**	**94.8**

(b) Other ResNet encoder architectures.

Figure 3.5: Top-1 and top-5 accuracy for BYOL (Grill et al. 2020).

The paper Exploring Simple Siamese Representation Learning from Facebook AI Research (FAIR) presents a simple siamese network for self-supervised learning. According to the researchers, this architecture learns meaningful representations without relying on negative sample pairs, large batch sizes, or a momentum encoder (X. Chen and He 2020). The proposed architecture, named Simple Siamese (SimSiam), is built up of two identical encoders (f) and one predictor (h) appended to one of the encoders. Negative cosine similarity is used to compute embedding similarity and a stop-gradient is applied on the branch without the predictor. A depiction of the SimSiam architecture can be seen in figure 3.6. The researchers note that the SimSiam architecture is similar to BYOL, with the authors thinking of SimSiam as “BYOL without the momentum encoder” (ibid.).

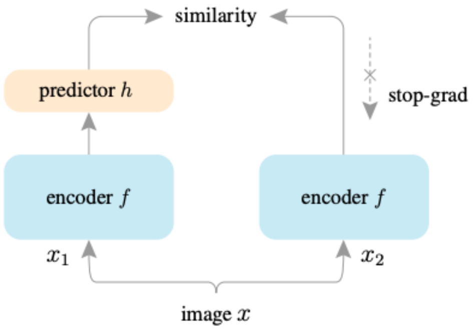


Figure 3.6: Depiction of the Simple Siamese architecture (X. Chen and He 2020).

In the paper the encoders f are built up of a backbone network followed by a multi-layer preceptor, while the predictor is also a multi-layer perceptron. For the empirical study of SimSiam the researchers used a ResNet50 as the backbone, three fully connected linear layers of size 2048 as the multi-layer perceptron in f, and two fully connected linear layers of size 512 and 2048 for the predictor h.

During the empirical study, the researchers tested SimSiam with a variety of different batch sizes. The results showed SimSiam to perform reasonably well over a wide range of batch sizes, with a drop of 2.0% for a batch of size 64 compared to a batch size of 256, as seen in figure 3.7.

batch size	64	128	256	512	1024	2048	4096
acc. (%)	66.1	67.3	68.1	68.1	68.0	67.9	64.0

Figure 3.7: SimSiam performance across different batch sizes (X. Chen and He 2020).

To test for collapsed solutions during training the researchers computed the channel average standard deviation of the l2-normalized output of the encoder over each epoch. According to the researchers the expected value for a non-collapsed solution is $\frac{1}{\sqrt{d}}$, where d is the output dimension of the encoder. When the stop-gradient was used during training the researchers did not observe any collapse, but without the stop-gradient SimSiam produced a collapsed solution. A run of SimSiam with and without the stop-gradient can be seen in figure 3.8.

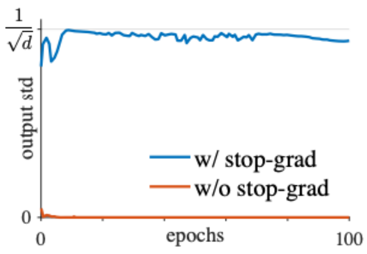
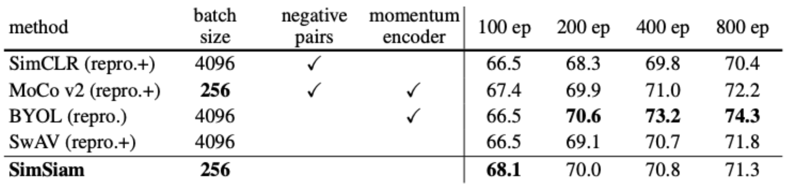


Figure 3.8: SimSiam testing for collapsed solutions with and without stop-gradient (X. Chen and He 2020).

The researchers compared SimSiam to other self-supervision methods on ImageNet with linear evaluation. The researchers found that SimSiam can perform on par with previous contrastive learning methods, while using smaller batch sizes, with it performing the best on 100 epochs, but performing comparatively worse as the epochs increase, as shown in figure 3.9.



method	batch size	negative pairs	momentum encoder	100 ep	200 ep	400 ep	800 ep
SimCLR (repro.+)	4096	✓		66.5	68.3	69.8	70.4
MoCo v2 (repro.+)	**256**	✓	✓	67.4	69.9	71.0	72.2
BYOL (repro.)	4096		✓	66.5	**70.6**	**73.2**	**74.3**
SwAV (repro.+)	4096			66.5	69.1	70.7	71.8
SimSiam	**256**			**68.1**	70.0	70.8	71.3

Figure 3.9: Comparison of SimSiam to other self-supervision methods (X. Chen and He 2020).

While the authors of BYOL used images for self-supervised learning, the authors of BYOL for Audio: Self-Supervised Learning for General-Purpose Audio Representation, BYOL-A for short, researched the BYOL architecture for the audio domain (Niizumi et al. 2021). The researchers of BYOL-A wanted to study if BYOL could be used for learning foreground audio representations without the use of negative samples.

As BYOL uses images as its input, the researchers of BYOL-A had to make some minor changes to the architecture as seen in figure 3.10. To make BYOL applicable to the audio domain the researchers produced mel-spectrograms from audio segments to use as inputs. The backbone used in BYOL-A was also changed from the ResNet50, used in BYOL, to a smaller CNN model previously used in the DCASE 2020 challenge (ibid.).

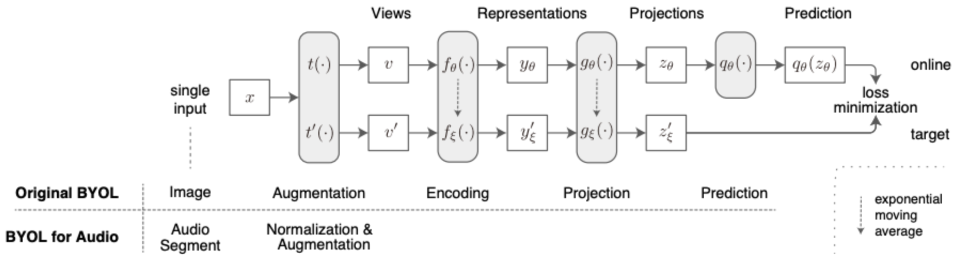
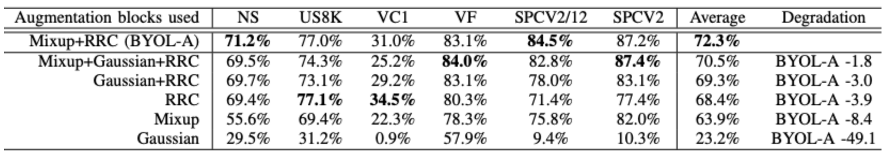


Figure 3.10: Comparison of BYOL and BYOL-A (Niizumi et al. 2021).

The researchers wanted to see if BYOL-A could be used to learn foreground representations regardless of background noise, as to make downstream tasks easier to learn regardless of the background audio. To do self-supervised learning on audio, the researchers of BYOL-A changed the augmentations used to produce the different views for the inputs. For the first step of the augmentation chain, pre-normalization was used on the mel-spectrogram. The spectrograms were normalized based on the mean and standard deviation of the training set of spectrograms. After normalization, random background noise was added to the spectrogram by taking a weighted average of the audio sample and a background noise sample (mixup). These noise samples were kept in a memory bank with a size of 2048. After adding the background noise, cropping and resizing using bicubic interpolation (RRC) were done to approximate frequency and time shifting. At the end of the augmentation chain, the spectrogram was again normalized to keep the values bounded.

To test the effectiveness of BYOL-A the researchers evaluated the self-supervised model on six different downstream tasks by doing a linear evaluation. The self-supervised learning was done with the AudioSet dataset with no labels, which had a total number of 1,963,807 audio samples. For pre-training, a target decay rate of $\tau = 0.99$, a batch size of 256, and 100 epochs were used. For the six downstream tasks, the datasets NSynth (NS), UrbanSound8k (US8k), VoxCeleb1 (VC1), VoxForge (VF), Speech Commands V2 (SPCV2), and Speech Commands V2 with 12 classes (SCV2/12) were tested. To assess the performance of the model, BYOL-A was compared against other self-supervised audio architectures. On average over all downstream tasks, BYOL-A performed the best with an accuracy of 77.8%.

To test the impact of the different combinations of augmentations an ablation study was done, using 10% of the AudioSet dataset for pre-training. In addition to RRC and mixup, Gaussian noise was also used, as shown in figure 3.11. The combination of mixup and RRC gave the best accuracy on average, while the top augmentations for each downstream task varied.

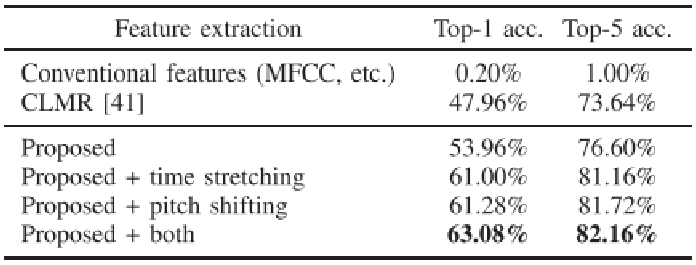


Augmentation blocks used	NS	US8K	VC1	VF	SPCV2/12	SPCV2	Average	Degradation
Mixup+RRC (BYOL-A)	**71.2%**	77.0%	31.0%	83.1%	**84.5%**	87.2%	**72.3%**	
Mixup+Gaussian+RRC	69.5%	74.3%	25.2%	**84.0%**	82.8%	**87.4%**	70.5%	BYOL-A -1.8
Gaussian+RRC	69.7%	73.1%	29.2%	83.1%	78.0%	83.1%	69.3%	BYOL-A -3.0
RRC	69.4%	**77.1%**	**34.5%**	80.3%	71.4%	77.4%	68.4%	BYOL-A -3.9
Mixup	55.6%	69.4%	22.3%	78.3%	75.8%	82.0%	63.9%	BYOL-A -8.4
Gaussian	29.5%	31.2%	0.9%	57.9%	9.4%	10.3%	23.2%	BYOL-A -49.1

Figure 3.11: Ablation study of BYOL-A augmentations (Niizumi et al. 2021).

Another paper using self-supervision on the audio domain is the paper by Yakura et al. (Yakura et al. 2022). In the paper, the researchers propose a new self-supervised learning approach for classifying singing voices. The researchers used an augmented version of MoCo for the downstream task of classifying singers based on 5-second audio segments. The researchers used different segments from the same audio clip as positive pairs, while negative pairs were constructed by generating two augmented views of the same audio segment. The data transformations done during the self-supervised training were pitch shifting and time stretching. The idea behind the change in MoCo's architecture was that the new augmentations applied to the audio segments were key features in telling singing voices apart.

To test the self-supervised model, the researchers compared their model against another self-supervision approach for music information retrieval called CLMR. The pre-trained weights of the models were frozen and a classification head of two fully connected linear layers was added. The models were fine-tuned on a dataset of singing voices with 500 classes and 20 audio clips per song. The models' top-1 accuracy can be seen in figure 3.12.



Feature extraction	Top-1 acc.	Top-5 acc.
Conventional features (MFCC, etc.)	0.20%	1.00%
CLMR [41]	47.96%	73.64%
Proposed	53.96%	76.60%
Proposed + time stretching	61.00%	81.16%
Proposed + pitch shifting	61.28%	81.72%
Proposed + both	**63.08%**	**82.16%**

Figure 3.12: Accuracy of augmented MoCo and CLMR (Yakura et al. 2022).

There have also been attempts at using self-supervised transfer learning on the audio domain. The paper by Shin et al. explores self-supervised learning on the image domain, for transfer learning on the audio domain (Shin et al. 2021). The authors of the paper used SimCLR for their self-supervision framework, training a ResNet50 on unlabeled images from ImageNet, for then to fine-tune the model on audio datasets.

To evaluate the self-supervised model, the authors compared their model with a supervised pre-trained ResNet50, trained on ImageNet, and a ResNet50 that had not been pre-trained. All models were fine-tuned on several audio datasets, with k-fold cross-validation, using the validation folds for benchmarking. From the evaluation the authors found that the self-supervised pre-trained model performed better than the untrained model, and on par with the supervised pre-trained model on all audio datasets. On average over all datasets the self-supervised pre-training got 87.98%, the supervised pre-training got 88.13%, and the untrained model got 80.29%.

3.5 Transfer Learning from Image to Audio

There have not been many papers using self-supervised pre-training for the bird vocalization classification task, but transfer learning from labeled images to the audio domain has been attempted. Usually, these papers use well-known network architectures, such as ResNet, and DenseNet (He, X. Zhang, et al. 2015) (Huang et al. 2016), which have been previously trained on some other large labeled dataset of images.

One of the papers using transfer learning from another domain to bird vocalization classification is the paper by Incze et al. The authors of this paper present a CNN for bird sound recognition and analyze its performance based on different configurations of image color scales and numbers of classes (Incze et al. 2018). The researchers used a known convolutional neural network called MobileNet, which is a smaller CNN designed for mobile computer vision applications (Howard et al. 2017). The instance of MobileNet used in the paper was pre-trained on the ImageNet Large Scale Visual Recognition Challenge 2012 dataset (ILSVRC-2012-CLS), consisting of 1,2 million images with 1000 classes (*ImageNet (ILSVRC2012)* n.d.). This model was then trained on a custom dataset compiled from Xeno-canto. To make the dataset compatible with the pre-trained MobileNet the audio clips were transformed into spectrograms and stored as png files.

The authors of the paper found the network to perform better when saving the png files with the jet color scale compared to greyscale, with an average of 7,4% higher accuracy for jet-scaled png files. The authors hypothesize that the performance difference was due to the MobileNet being pre-trained on color images (Incze et al. 2018). When classifying between 2 classes the models managed to score an accuracy above 80%. When increasing the number of classes the models performed worse and the difference between using jet-scaled and grey-scaled images increased, as seen in figure 3.13.

Another paper that also used pre-training from the image domain was the paper by Disabato et al. This paper explores the use of transfer learning to train a model for birdsong detection on edge devices (Disabato et al. 2021). The paper presents ToucaNet, a convolutional neural network based on ResNet18, which is designed for birdsong detection with the constraint of IoT devices in mind. ToucaNet uses ResNet18 pre-trained on the ImageNet classification task as its CNN base. The authors argue that using weights from pre-training decreases the duration of the training phase (ibid.). To generate the spectrograms used as inputs to the model, the authors sampled 10 seconds of audio at 22050 Hz with

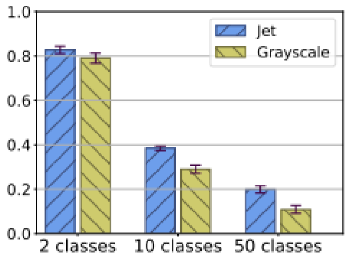
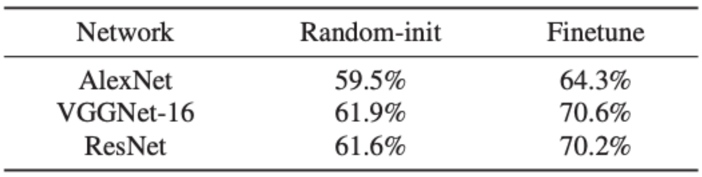


Figure 3.13: Accuracy of jet scaled and grey scaled png files over variable amounts of classes (Incze et al. 2018).

a frame size of 512 and a hop size of 512, producing spectrograms with a shape of (257, 431). According to the authors, this is still too demanding for IoT devices. Because of this, an approximation of ToucaNet was made called BarbNet. BarbNet is a smaller model where some of the convolutional layers of ToucaNet are removed dependent on the hardware specifics of the IoT unit. BarbNet also lowers the input dimensionality by setting the sampling rate to 2205, frame size to 64, and hop size to 64.

These models were tested on three datasets, Warblbr, Free-field, and BirdVox-DCASE-20k. According to the authors, ToucaNet is able to perform at the state-of-the-art level with an accuracy of 92,25% for birdsong detection, while using 80% of the computational demand of its peers. BarbNet is able to achieve 77,8% accuracy while consuming even fewer resources.

The paper by Zhou et al. investigates the use of pre-training on images to use for acoustic scene classification (Zhou et al. 2018). The authors note that a problem for most acoustic scene classification tasks is the lack of training data. As a solution to this problem, the authors explored different convolutional networks, pre-trained on ImageNet, and compared their performance against the same networks without pre-training. Alexnet, VGGNet-16, and ResNet50 were tested on the TUT Urban Acoustic Scenes 2018 dataset, containing 10 classes. The recordings from the dataset were processed into mel-spectrograms and used as input to the different models. By comparing the accuracy of the pre-trained models with models randomly initialized Zhou et al. found that models pre-trained on images performed better on all model instances, as seen in figure 3.14. With further data augmentation and changes to the last multi-layer perceptron, the authors were able to achieve an accuracy of 77.8% with a pre-trained VGGNet-16.



Network	Random-init	Finetune
AlexNet	59.5%	64.3%
VGGNet-16	61.9%	70.6%
ResNet	61.6%	70.2%

Figure 3.14: Accuracy of models pre-trained on ImageNet compared to random initialization of the models (Zhou et al. 2018).

3.6 Summary

This chapter has given an overview of previous work related to the topic of bird vocalization classification and self-supervised learning. State-of-the-art bird vocalization classification systems have been presented, such as BirdNet, and descriptions of key components such as spectrogram generation and neural network architecture have been given. Submissions to bird vocalization classification competitions have also been discussed, presenting classification systems that performed well in open competitions. This chapter has also presented recent works on self-supervised learning, such as BYOL and SimSiam, and shown how these architectures can be used to pre-train on unlabeled audio, such as the case with BYOL-A. Lastly, examples of how transfer learning from the image domain have been used on bird classification tasks have been displayed, showing different works using neural networks pre-trained on image datasets for bird vocalization classification.

Chapter 4

Methodology

This chapter covers in detail the methodology used in this thesis. Section 4.1 covers the method used to acquire the unlabeled and labeled datasets and describes their contents. Section 4.2 describes the process of preparing the datasets for self-supervised pre-training and fine-tuning. In section 4.3, the different augmentations to be tested for the self-supervised learning scheme are presented. Section 4.4 describes in detail the architecture of the chosen self-supervised learning scheme. In section 4.5, the different experiments to be conducted are described in detail.

4.1 Data Collection

As mentioned in section 2.4, self-supervised learning requires large amounts of data. To be able to explore the use case of self-supervision on bird vocalization classification, a large amount of unlabeled data must be acquired. The web page xeno-canto.org contains a vast amount of weakly labeled audio of bird vocalizations, containing more than 750,000 recordings of varying lengths. To build the large unlabeled dataset for self-supervision, recordings from Xeno-canto will be used.

Recordings on Xeno-canto contain metadata describing audio quality ranging from A to F, recording origin, and recording duration. Since the downloaded recordings will need to be cut into smaller equally sized clips, there is a possibility of generating audio segments with no birds present. To lower the amount of empty audio clips, only recordings between 3 and 60 seconds will be downloaded from Xeno-canto, with the hypothesis that shorter recordings have a higher ratio of bird vocalization to non-bird vocalization.

The recordings on Xeno-canto are sourced from volunteers sending in their personal recordings. This means that the quality of the audio recordings can vary. To avoid low-quality recordings in the unlabeled dataset, only recordings with a quality rating of C or higher will be downloaded. To restrict the number of recordings downloaded from Xeno-canto, only recordings from Europe and the United States of America will be used.

To evaluate the performance during self-supervised learning, a manually labeled bird vocalization dataset will be constructed. This dataset will contain 10 different classes of birds, with approximately 250 instances for each class. To generate the dataset, vocalizations of the 10 birds with the largest amount of recordings in the United Kingdom will be downloaded from Xeno-Canto. These files will be converted to spectrograms and split into durations of 2.6 seconds, and manually labeled. To ensure that the spectrograms are labeled correctly each spectrogram under consideration will be compared to samples of spectrograms corresponding to the bird species available on Xeno-canto. This manually labeled dataset will be split into a training dataset for fine-tuning and a validation dataset to measure model performance. 80% of the labeled data will be used to construct the training data, while the remaining 20% will be used to construct the validation dataset. A more detailed description of the supervised dataset can be seen in the appendix A.1.

The bird species in the labeled dataset will be, Common blackbird, Common chaffinch, Common chiffchaff, Common whitethroat, Corn bunting, Eurasian blue tit, Eurasian tree sparrow, Great tit, Willow warbler, and Yellowhammer.

4.2 Pre-Processing

Before the acquired audio data can be used to explore self-supervised learning on bird vocalization, it has to be standardized and converted to a format that is suitable for self-supervision. From section 3, the papers by Kahl, Lasseck, and Disabato et al. show that converting audio recordings to spectrograms is a good representation for the use case of bird vocalization classification (Kahl 2020) (Disabato et al. 2021) (Lasseck 2018). A spectrogram also bears resemblance to a grayscale image, with it being a two-dimensional representation, which is beneficial as the self-supervised methods mentioned in chapter 3 such as BYOL and SimSiam were designed for self-supervision on images (Grill et al. 2020) (X. Chen and He 2020). This makes it possible to use an already existing self-supervision framework without extensive changes, as BYOL-A presented in section 3 is an example of.

Before the audio recordings can be converted to spectrograms, all the recordings have to be at the same sample rate, as the sample rate of a recording influences the final spectrogram. According to Stephan Kahl, having a high temporal resolution is important when classifying bird vocalizations (Kahl 2020). Therefore, all audio recordings will be re-sampled to a high sample rate of 44100 Hz before being converted to spectrograms.

As mentioned in section 2.5, there is a trade-off between frequency and time resolution. To avoid having to test different hyperparameter values, such as window size, hop length, and scaling, when creating the spectrograms, the spectrograms in this thesis will be constructed according to the description in BirdNet by Stefan Kahl. Each audio recording will be constructed with a frame size of 512, a hop length of 256, and converted to a mel-spectrogram with 64 mel-bands between 150 Hz and 15000 Hz. After the mel-spectrogram is produced, it will be scaled according to the non-linear scaling given in the paper by Kahl, Identifying Birds by Sound: Large-scale Acoustic Event Recognition for Avian Activity Monitoring (ibid.). The scaling formula is shown in equation 4.2. For this thesis, the hyperparameter a will be equal to 1.2.

$$\sigma(a) = \frac{1}{1 + exp(-a)} \tag{4.1}$$

$$y = x^{\sigma(a)} \tag{4.2}$$

As mentioned in section 4.1, the downloaded bird vocalizations are between 3 and 60 seconds. Therefore, after converting the audio recordings to mel-spectrograms, each spectrogram will be cut into segments of 2.6 seconds. The resulting representation for one sample will be a mel-spectrogram of shape (64, 448). When cutting the mel-spectrograms, the last spectrogram from one recording might be shorter than 2.6 seconds. If the last recording is longer than 2.08 seconds, the spectrogram will be padded with zeros at the end to reach a length of 2.6 seconds. If the mel-spectrogram is shorter than 2.08 seconds, it will be discarded.

After the mel-spectrograms have been split, they will be normalized in the range of 0 to 1. An overview of the pre-processing pipeline can be seen in table 4.1, with an example of a resulting mel-spectrogram seen in figure 4.1.

Table 4.1: Overview of the pre-processing pipeline for constructing the unlabeled dataset.

- Resample audio to 44100 Hz
- Compute mel-spectrogram with window size 512, hop length 256, and 64 mel-bands between 150 and 15000 Hz
- Scale mel-spectrogram
- Split mel-spectrogram into a shape of (64, 448)
- Normalize mel-spectrogram in the range between 0 and 1

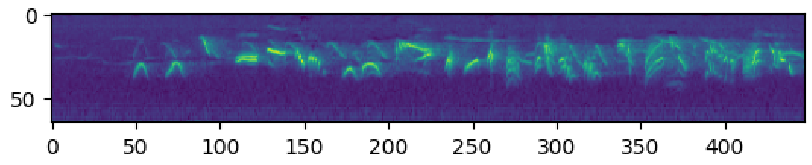


Figure 4.1: Mel-spectrogram after pre-processing pipeline.

4.3 Spectrogram Augmentations

Augmentation is an important step during self-supervised learning and can impact the backbone's ability to learn adequate embeddings. The paper by Chen et al. explores how the composition of different types of image augmentations can impact the top-1 accuracy for downstream linear evaluation, as displayed in figure A.3 in the appendix (T. Chen et al. 2020). The paper shows that choosing the type of augmentation and combination of augmentations is an important decision in the construction of a self-supervision training scheme. The authors of BYOL note in their conclusion that to generalize BYOL for other domains, such as audio and text, it is necessary to find suitable augmentations for each domain (Grill et al. 2020). Therefore, using existing image augmentations from the image domain might not be suitable for self-supervision on bird vocalizations.

To explore the topic of self-supervised learning on bird vocalization classification, this thesis will not use the augmentations used in SimCLR, SimSiam, or BYOL. Instead, this thesis will test augmentations used within the audio domain. These augmentations will be cropping, adding background audio, adding noise, frequency masking, and time masking.

4.3.1 Cropping

The cropping used in this thesis is different from the standard random crop used in the image domain. The crop augmentation is taken from BYOL-A, where a random crop is done on the spectrogram, but the crop boundary extends outside both sides of the spectrogram by 25% (Niizumi et al. 2021). If parts of the crop are outside of the spectrogram, that area will be filled with zeros. The crop boundary does not extend outside the frequency range, as depicted in figure 4.2. After a crop is taken, it is resized to the same shape as the original spectrogram. The crop height and width are chosen by uniformly choosing a height and width multiplier between 0.6 and 1.5 and multiplying the values with the spectrogram's original width and height. The value for the crop height is clipped to the original spectrogram height so as to not go outside the crop boundary.

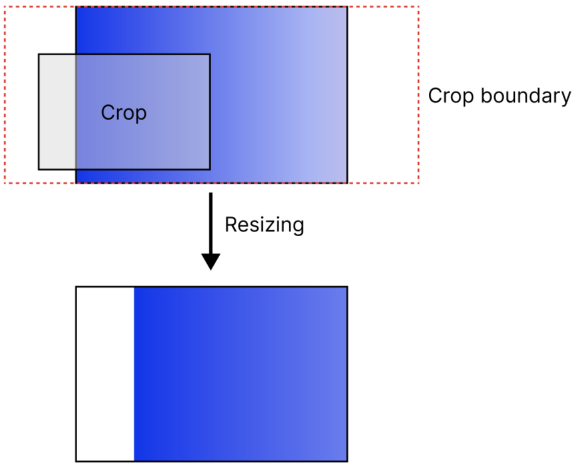
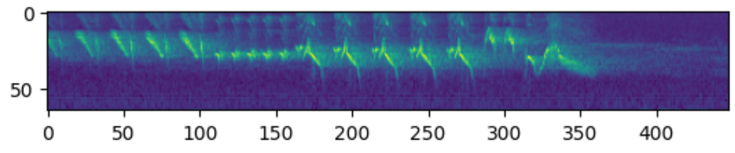
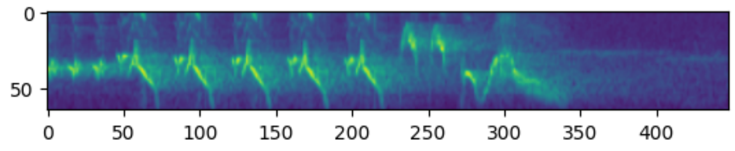


Figure 4.2: Cropping and resizing within the crop boundary.

According to the authors of BYOL-A the crop augmentation approximates pitch shifting and time stretching on the mel-spectrogram, expecting details spread over the spectrogram to be learned regardless of the distortion of time and pitch (ibid.). During an ablation study, the authors of BYOL-A found cropping to perform well across different audio datasets. An example of the crop augmentation on the self-supervised dataset can be seen in figure 4.3.



(a) Spectrogram before augmentation.



(b) Spectrogram after crop augmentation.

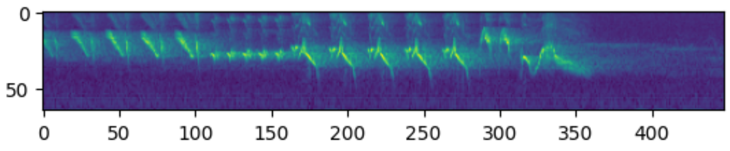
Figure 4.3: Example of applying crop to a mel-spectrogram.

4.3.2 Background Audio

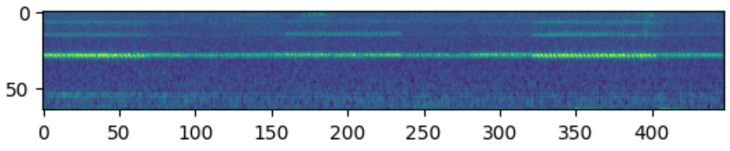
Adding background audio to the spectrogram is also an augmentation done in BYOL-A and will also be tested in this thesis. A separate smaller dataset of bird vocalizations will be constructed and used to add background audio to the spectrograms. This will be done by converting the mel-spectrograms back to a linear scale, for then to take a weighted average of the foreground spectrogram and the background spectrogram, shown in equation 4.3. In the equation, the value a is sampled uniformly between 0 and 0.4. After the background audio is added to the spectrogram, the same non-linear scaling as described in section 4.2 is re-applied, resulting in spectrograms similar to the one in figure 4.4.

$$spectrogram = (1 - a) \cdot spectrogram + a \cdot background \tag{4.3}$$

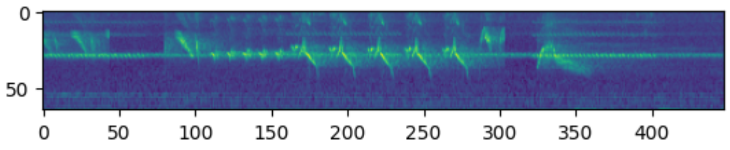
This augmentation creates views of a spectrogram with the same foreground sound but different background audio. The idea is that this augmentation encourages learning of foreground representation invariant of the background audio. During the ablation study of BYOL-A, the background augmentation performed well on its own, scoring an average of 63.9% on linear evaluation over different datasets, and when combined with cropping achieving the highest average accuracy of 72.3% (Niizumi et al. 2021).



(a) Spectrogram before augmentation.



(b) Spectrogram of background audio.



(c) Spectrogram after background audio augmentation.

Figure 4.4: Example of applying background audio to a mel-spectrogram.

4.3.3 Random Noise

When training a classification model, the addition of noise to the input features can be beneficial. The use of random noise often helps the model reduce overfitting by introducing small random variances in similar images. According to Kahl, the addition of noise when training has been shown to increase overall detection performance (Kahl, Wood, et al. 2021). Lasseck also found that the addition of noise while training increased the performance of the model (Lasseck 2018).

The addition of noise has previously been tested for self-supervision in the image domain. The authors of SimCLR tested adding Gaussian noise as an augmentation in their ablation study. They found Gaussian noise to perform not as well alone, achieving only 9.8% accuracy on the linear evaluation on ImageNet, but when noise was combined with cropping, the accuracy increased from 33.1% to 39.9% on the linear evaluation (T. Chen et al. 2020).

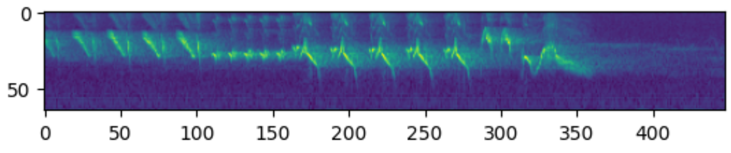
BYOL-A used noise augmentation to compare against adding background audio (Niizumi et al. 2021). The authors found that combining the noise augmentation with cropping improved accuracy on the linear evaluation by 0.9%, while using background audio instead improved accuracy by 3.9%.

The implementation of random noise in this thesis is similar to Gaussian noise. A matrix of shape (64, 448) is first constructed, containing values sampled from a Gaussian distribution with a mean of 0 and a standard deviation of 0.13. A bitmask with the same shape is also constructed, with values sample from a binomial distribution with p equal to 0.7. The noise matrix is then multiplied with the bitmask before being added to the spectrogram. In the end, all values in the spectrogram will be clipped between 0 and 1. An example of the noise augmentation can be seen in figure 4.5.

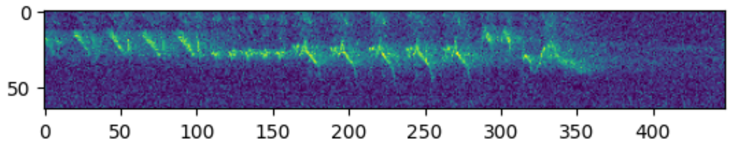
4.3.4 Frequency Masking and Time Masking

Frequency masking and time masking are two widely used augmentation methods within deep learning in the audio domain. Frequency and time masking are used on spectrograms, where frequency masking masks a number of consecutive frequency bands, and time masking masks a number of consecutive time frames.

In the paper by Park et al. time masking, frequency masking, and time warping are used for speech recognition (Park et al. 2019). During the training of a Listen, Attend, and Spell (LAS) network, time masking, frequency masking, and time warping were applied to mel-spectrograms containing human speech. By using these augmentations, the researchers were able to achieve state-of-the-art per-



(a) Spectrogram before augmentation.



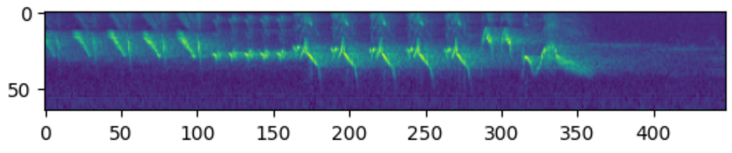
(b) Spectrogram after noise augmentation.

Figure 4.5: Example of applying noise to a mel-spectrogram.

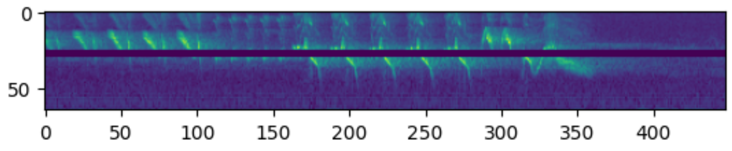
formance on the LibriSpeech 960h and Switchboard 300h tasks, outperforming prior results. In their discussion, the authors point out that time warping is not a major factor in improving performance, leaving time masking and frequency masking as the largest contributors. The authors note that adding these augmentations converts an overfitting problem to an underfitting problem, making it possible to train longer and use larger networks.

Kahl discusses time and frequency masking in the context of bird vocalization. According to Kahl, not every utterance holds the same importance when trying to classify speech, and this is also applicable to bird vocalizations (Kahl 2020). Kahl notes that bird trills repeatedly encode the same signals, and by masking entire frequency bands or time frames, the neural network is forced to focus on semantically significant features that are invariant of the information lost.

The implementation of frequency masking in this thesis is similar to that of the paper by Park et al. First, a random height of the frequency mask, h, is chosen from the uniform distribution [0, 20], making the largest possible mask approximately 31% of the total number of frequency bands. After the height of the frequency mask has been chosen, the starting point h_0 is drawn from the uniform distribution [0, 64-h], making sure the mask stays within the bounds of the spectrogram. After the size and indices of the frequency mask are chosen, the corresponding area on the spectrogram is filled with zeros. An example spectrogram with frequency masking is shown in figure 4.6.



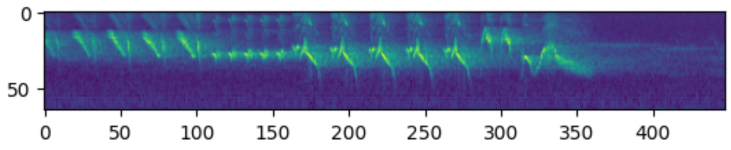
(a) Spectrogram before augmentation.



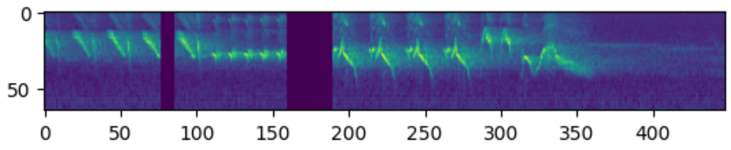
(b) Spectrogram after frequency masking augmentation.

Figure 4.6: Example of applying frequency masking to a mel-spectrogram.

The implementation of time masking in this thesis is also based on the paper by Park et al. First, a random width of the time mask, w, is chosen from a uniform distribution in the range [0, 40], making the largest single time mask possible approximately 9% of the total number of time frames in the spectrogram. After the width of the time mask has been chosen, the starting point of the mask w_0 is drawn from a uniform distribution in the range [0, 448-w], making sure the mask stays within the spectrogram. After the indices of the time mask are chosen, the values within the bounds of the mask are set to zero. Because the time dimension is larger than the frequency dimension in the mel-spectrogram, frequency masking is applied twice to have a bigger impact. The second application of frequency masking does not take into account the first application. Therefore, the second time mask can be placed on top of the first. An example spectrogram with time masking applied is shown in figure 4.7.



(a) Spectrogram before augmentation.



(b) Spectrogram after time masking augmentation.

Figure 4.7: Example of applying time masking to a mel-spectrogram.

4.4 Self-Supervised Learning Architecture

For self-supervision on the constructed unlabeled bird vocalization dataset described in section 4.2, the SimSiam architecture will be used as a starting point. The SimSiam architecture was chosen because of its simpler implementation and hardware requirements. Simsiam does not use explicit negative samples in contrast to MoCo and SimCLR (T. Chen et al. 2020) (He, Fan, et al. 2019), and therefore does not require a large memory bank of negative samples. SimSiam is also smaller than BYOL, with BYOL using a momentum encoder while SimSiam uses identical encoders for both augmented views. This makes SimSiam faster to pre-train compared to BYOL, as the forward pass only uses one encoder instead of two and there is no need to update the weights of a momentum encoder. SimSiam has also been shown to perform well on smaller batch sizes, as presented in section 3.4. Being able to use smaller batch sizes lessens the hardware requirements on the GPU, making it possible to train on GPUs with less memory.

The specific architectural implementation of SimSiam will be constructed similarly to the architecture for the CIFAR-10 dataset from the paper by Chen and He (X. Chen and He 2020). To implement the self-supervised training scheme the machine learning library TensorFlow will be used.

The implemented SimSiam architecture can be split into three parts, backbone, projector, and predictor, where the original encoder in SimSiam is split into the backbone and the projector. The backbone is the convolutional neural net being pre-trained. In this thesis, the backbone will be the convolutions of a ResNet18, producing an output dimension of 512. Following the backbone is the projector, which will consist of two fully connected linear layers of size 2048. Each layer in the projector will be followed by a batch normalization layer. Only the first layer of the projector will have a ReLu activation function. The predictor will only be applied to one of the branches. The predictor will consist of a fully connected linear layer of size 512 with batch normalization and ReLu activation function, and a last fully connected linear layer of size 2048 without batch normalization or activation function. A graphical representation of the architecture can be seen in figure 4.8. Note that the figure only depicts spectrogram 1 going through the branch with the predictor, but during training, both augmented spectrograms will be sent down both branches.

The reason for using a smaller SimSiam architecture compared to the original SimSiam architecture used for ImageNet (X. Chen and He 2020) is that the spectrograms used in this thesis are smaller than images from ImageNet. ImageNet contains color images and is usually cropped to a shape of (224, 224, 3), while the mel-spectrograms used in this thesis have a shape of (64, 448). Therefore, a smaller architecture might be more appropriate.

The reason for choosing a ResNet as the backbone neural network is that ResNets are commonly used as backbones for contrastive and non-contrastive learning architectures. BYOL, SimSiam, MoCo, and SimCLR all use a ResNet50 as their backbone (Grill et al. 2020), (X. Chen and He 2020), (He, Fan, et al. 2019), (T. Chen et al. 2020).

To produce two different views from the same input, the spectrogram will be fed through an augmentation module. This module takes in a spectrogram and applies the selected augmentations, making two differently augmented views of the same spectrogram. The augmentations to be used in this thesis are detailed in section 4.3. Since ResNet18 expects a three-dimensional input shape, a duplication operation is appended after the augmentation module, as depicted in listing 4.1. This operation copies the spectrogram two times and stacks them together to create a shape of (64, 448, 3).

The loss function to be used when evaluating the similarity of the outputs from the two branches will be negative cosine similarity, shown in equation 4.4. p and z are the output vectors from the two branches of the SimSiam architecture. The triangle bracket denotes the dot product of the two vectors produced from each branch, while the denominator is the product of the Euclidean distance of the two vectors. This loss function is bounded between 0 and -1, where the closer to -1 the loss is, the more similar the vectors outputted from the two branches are.

$$Loss = -\frac{\langle z,\ p\rangle}{\|z\|_2\ \cdot\ \|p\|_2} \tag{4.4}$$

To symmetrize the loss, both of the augmented views of the spectrogram will be passed through both branches of the SimSiam architecture and the computed similarity loss will be averaged, as shown in equation 4.5. Doing this lets both augmented spectrograms be a pseudo target for the other spectrogram.

$$AveragedLoss = -(0.5 \cdot \frac{\langle z_1,\ p_2\rangle}{\|z_1\|_2\ \cdot\ \|p_2\|_2} + 0.5 \cdot \frac{\langle z_2,\ p_1\rangle}{\|z_2\|_2\ \cdot\ \|p_1\|_2}) \tag{4.5}$$

When pre-training on bird vocalizations the hyperparameters for the SimSiam training scheme will be set to those of the implementation for the CIFAR-10 dataset (X. Chen and He 2020). The optimizer will be stochastic gradient descent with a batch size of 256, a weight decay of 0.0005, and a momentum of 0.9. The initial learning rate will be set to 0.03 and follow a cosine decay schedule during training.

Listing 4.1: Pseudocode for handling of spectrograms during pre-training.

```
# loading batches of spectrograms from the dataset
for x in dataset:
    # using augmentation module to get different views
    spec_1, spec_2 = augmentation(x), augmentation(x)
    # stacking of spectrograms to get shape [batch, 64, 448 ,3]
    spec_1 = stack([spec_1, spec_1, spec_1])
    spec_2 = stack([spec_2, spec_2, spec_2])
    # passing the spectrograms through the SimSiam encoder
    z_1, z_2 = projector(backbone(spec_1)), projector(backbone(spec_2))
    # passing the embeddings through the SimSiam predictor
    p_1, p_2 = predictor(z_1), predictor(z_2)
    # computing negative cosine similarity between (z_1,p_2) and (z_2,p_1)
    loss = loss(stop_grad(z_1),p_2)*0.5 + loss(stop_grad(z_2), p_1)*0.5
    # updating SimSiam model
    update(loss, predictor, projector, backbone)
```

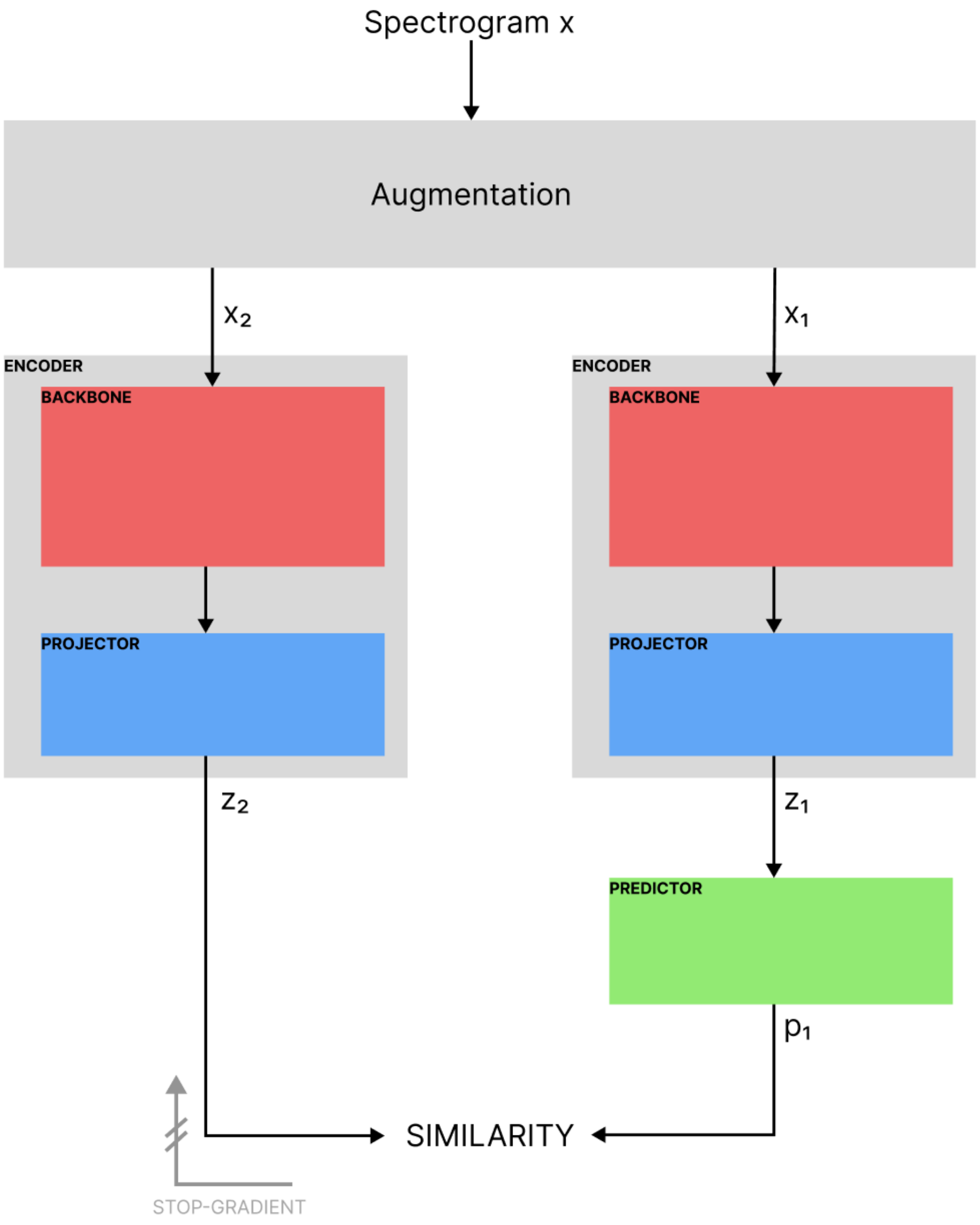


Figure 4.8: Diagram of SimSiam architecture used during self-supervision.

4.5 Experimental Setup

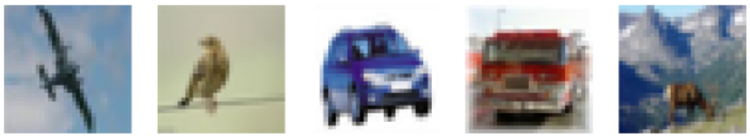
4.5.1 Implementation Validation

To validate that the implementation of SimSiam is correct, a reproduction of a simple result from the original SimSiam paper will be done by testing it on the CIFAR-10 dataset. Before applying this thesis implementation of SimSiam on bird vocalizations, it will be pre-trained on the CIFAR-10 dataset and evaluated with linear fine-tuning to see if the backbone model learns good representations. The architecture of the backbone, projector, and predictor will be the same as for the bird vocalization domain.

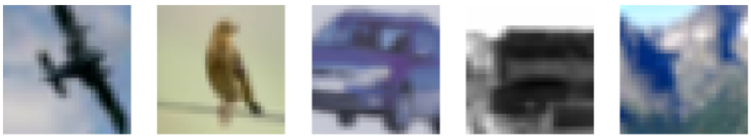
The CIFAR-10 dataset is an image dataset consisting of 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck), where each class contains 6000 color images of size (32, 32, 3) (Krizhevsky 2009). The dataset is divided into 50000 training samples and 10000 test samples. Example images from CIFAR-10 can be seen in figure 4.9.

Figure 4.9: Example images from CIFAR-10.

The augmentations done on the CIFAR-10 images while pre-training will follow those of the simsiam paper. The augmentations are horizontal flipping, color jitter, and grayscale. Random cropping and horizontal flipping will always be applied, while color jitter and grayscale will be applied with a probability of 80% and 20% respectively. An example of the augmented images can be seen in figure 4.10. The self-supervised pre-training will run for a total of 900 epochs with a batch size of 512 and an initial learning rate of 0.06 for the cosine decay schedule.



(a) Example images from CIFAR-10 before augmentations.



(b) Example images from CIFAR-10 after applying augmentations.

Figure 4.10: Augmented example images from CIFAR-10.

During pre-training on CIFAR-10, the per epoch loss and channel average standard deviation of the l2-normalized embedding will be recorded. If the loss converges close to -1 fast during pre-training, it could be a sign that the solution collapses and produces the same embedding regardless of input. If the channel average standard deviation is close to $\frac{1}{\sqrt{d}}$, where d is the size of the output dimension during training, the model most likely avoided a collapsed solution, finding a non-trivial embedding.

The value of $\frac{1}{\sqrt{d}}$ is derived by noting the output embedding from the encoder as z and the l2-normalized output embedding as z' where $z'_j = \frac{z_j}{\sqrt{\sum_{n=j}^{d} z_j^2}}$. If z_j is assumed to be an independent identically distributed Gaussian distribution with a zero mean and standard deviation of 1 for all j. Then the squared sum of all channels $\sum_{n=j}^{d} z_j^2$ follows the chi-squared distribution and have an expected value of d (Ross 2010). Thus $z'_j = \frac{z_j}{\sqrt{d}}$ and $std(z'_j) = std(\frac{z_j}{\sqrt{d}})$, and because $\sqrt{d}$ is a constant we can write $std(z'_j) = \frac{1}{\sqrt{d}} \cdot std(z_j)$. Following the assumption that z_j have a standard deviation of 1 we get $std(z'_j) = \frac{1}{\sqrt{d}}$. Since this is assumed for all the channels in the embedding, the expected average standard deviation over all channels will be $\frac{1}{\sqrt{d}}$. In this thesis, the embedding dimension d has a size of 2048, making the expected average standard deviation over all channels $\frac{1}{\sqrt{2048}} = 0.02209708691$.

After self-supervised pre-training of the model, images from the CIFAR-10 dataset will be passed through the backbone, and the embeddings will be saved. T-SNE will then be used to reduce the dimensionality of the image embeddings and plot them in a two-dimensional space. This visualization will be compared to the embeddings of a ResNet18 backbone that have been initialized with random weights to see if there are any visual indications that the self-supervised ResNet18 model clusters images from the same classes together.

To test the performance of the self-supervised Resnet18 backbone, a linear evaluation will be conducted. For the linear evaluation of the SimSiam backbone, the ResNet18 backbone weights will be frozen and a fully connected linear layer of 10 neurons will be appended to the head of the backbone, as depicted in figure 4.11. This final linear layer will be trained on the CIFAR-10 training set for a total of 100 epochs with the Adam optimizer, using a learning rate of 0.0025 and categorical entropy as the loss function. The resulting linearly fine-tuned network will then be evaluated on the center crop of the CIFAR-10 test set.

4.5.2 Augmentation Exploration

To explore the impacts of the different augmentations, described in section 4.3, on the self-supervised pre-training scheme, multiple models using different spectrogram augmentations during the self-supervised learning will be trained.

For each augmentation, seven models using the setup described in section 4.4 will be pre-trained with self-supervised learning using only one of the augmentations. The dataset used to pre-train these models will be a subset of the unlabeled dataset, containing 274,567 spectrograms, approximating 28% of the entire unlabeled dataset. Each model will be pre-trained for a total of 100 epochs, taking approximately 8 hours to run on an NVIDIA A100 40GB GPU.

After all the models of one augmentation are done pre-training, a downstream task will be used to evaluate the models and augmentations. This entails using the pre-trained backbone, freezing the ResNet18 model weights, and appending a single fully connected linear layer of 10 neurons to the head of the model. The downstream task used in the evaluation will be to classify the bird vocalizations in the manually labeled dataset, described in section 4.1. The final linear layer appended to the frozen backbone will be fine-tuned on the labeled training dataset and then evaluated on the labeled validation dataset. The fine-tuning will be done for a total of 100 epochs with the Adam optimizer using a learning rate of 0.0025 and a categorical cross-entropy loss function. A depiction of fine-tuning the frozen backbone and linear layer can be seen in figure 4.11.

Each model will be fine-tuned and evaluated five times. Since each augmentation will be used to pre-train seven models, the total amount of evaluations per augmentation will be 35. These evaluations will then be averaged to find the average performance for each augmentation, and their standard deviation will be calculated.

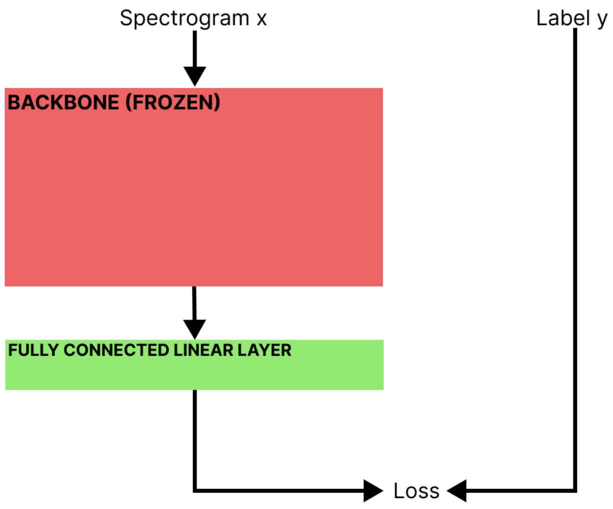


Figure 4.11: Depiction of linearly fine-tuning on the downstream task. The ResNet18 backbone for the SimSiam architecture is frozen, and a linear layer of 10 neurons is appended to the head of the backbone.

After the performance of all augmentations has been calculated, the three best performing augmentations will be chosen for further testing. The three best performing augmentations will be combined into pairs of augmentations to find if any of the combinations of augmentations enhances the performance. These tests will be carried out with the same procedure as the single augmentations.

4.5.3 Fine-Tuning Evaluation

The purpose of self-supervised learning schemes like SimCLR, BYOL, and SimSiam is to be able to use large datasets of unlabeled data to train a neural network backbone to produce good representations that can later be used in a downstream task that uses supervised learning. This is similar to using the weights of a neural network that has been trained on a large labeled dataset, such as ImageNet, and then fine-tuning it on a downstream task dataset.

After the combinations of augmentations have been evaluated, seven models, using the best performing augmentation setup, will be pre-trained on the full size of the unlabeled dataset, containing 987,305 spectrograms [1]. The self-supervision will run for a total of 100 epochs, taking approximately 29 hours for each model on an NVIDIA A100 40GB GPU.

Convolutional neural networks trained on ImageNet are readily available through platforms such as TensorFlow, PyTorch, and other open-source projects, making it easy to implement pre-trained models for classification tasks. Since using models pre-trained on ImageNet has shown good results within the bird vocalization classification task and general audio classification, as discussed in section 3.5, a pre-trained Resnet18 model will be used as the baseline model when evaluating the performance of the model pre-trained with self-supervision on the unlabeled bird vocalization dataset from Xeno-canto.

After the self-supervised models are done pre-training on the full size of the unlabeled dataset, they will be compared against a ResNet18 pre-trained on ImageNet. Each model will be linearly evaluated where the ResNet18 backbones are frozen, and a single fully connected linear layer with 10 neurons is appended to the model. This layer will then be trained on different amounts of the manually labeled training dataset to see how well the self-supervised model compares to a model pre-trained on ImageNet. The different amounts of training data will be 5%, 10%, 25%, 50%, and 100% of the manually labeled training dataset, which is approximately 10, 20, 50, 100, and 200 samples per class. The networks will be fine-tuned for a total of 100 epochs and 200 epochs with a learning rate of 0.0025 to explore their behaviors on different training durations. Each self-supervised model will be linearly evaluated five times for each experiment, making a total of 35 linear evaluations for the self-supervised models. These evaluations will be averaged and used to compare against the model pre-trained on ImageNet.

The self-supervised models and the model pre-trained on images will also be fully fine-tuned where the ResNet18 backbones are unfrozen such that all weights in the models are trained, as depicted in figure 4.12. This test will be conducted with the same setup as the linearly fine-tuned evaluation to see if self-supervision on bird vocalizations produces better weight initialization than pre-training on ImageNet.

[1]Ideally, more than seven models would be trained on the full size of the unlabeled dataset to get more statistically significant results, but because of the long training time and hardware waiting queue this was not feasible.

The ResNet18 model pre-trained on ImageNet used in this thesis will be taken from the GitHub repository of Pavel Iakubovskii (Iakubovskii 2023). The pre-trained ResNet18 is implemented in TensorFlow and achieves a top-1 accuracy of 68.24% and top-5 accuracy of 88.49% on the center crop of the ImageNet validation dataset.

When fine-tuning and evaluating the self-supervised model and the model pre-trained on ImageNet, each model will be fine-tuned and evaluated a total of 35 times, and the average accuracy of the models will be calculated over all fine-tuning runs. The standard deviation for all the fine-tuning runs will also be calculated.

In the end, embeddings produced by the ResNet18 backbones of the self-supervised models and the Resnet18 pre-trained on ImageNet will also be evaluated. This will be done by using t-SNE dimension reduction to visualize the embeddings in a two-dimensional space to get a visual understanding of how the different models cluster the embeddings.

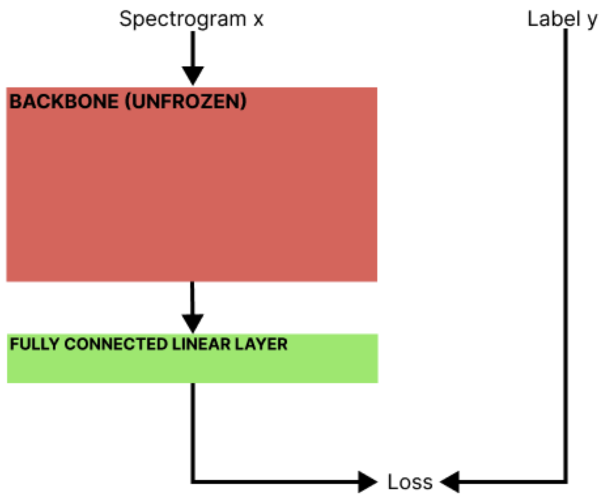


Figure 4.12: Depiction of fully fine-tuning on the downstream task. The ResNet18 backbone is unfrozen and a linear layer of 10 neurons is appended to the head of the backbone.

4.6 Summary

This chapter has given an overview of the methodology used in this thesis. A description of how the large unlabeled bird vocalization dataset and the manually labeled dataset were obtained has been given, as well as how these datasets were pre-processed and converted to mel-spectrograms. The five data augmentations cropping, addition of noise, background audio, frequency masking, and time masking have been presented as well as the reason for their selection. This chapter has also presented a detailed description of the implemented SimSiam architecture, as well as a description for validating its implementation. Lastly, a protocol for evaluating the different augmentations and the final self-supervision scheme has been given.

Chapter 5

Results and Analysis

This chapter presents the results from the experiments described in chapter 4, and discusses the results in the context of previous work within bird vocalization and self-supervised learning. Section 5.1 presents the results from running the experiments explained in section 4.5 in the previous chapter. Section 5.2 discusses the results presented in section 5.1 and relates the results to other relevant work.

5.1 Results

5.1.1 Validation of Implementation

By testing the thesis's implementation of SimSiam on the CIFAR-10 dataset, it is possible to know if the implementation is correct by seeing if the implementation produces the expected behavior. It is also possible to explore and display the different behaviors of the model.

Figure 5.1 depicts the negative cosine similarity loss during self-supervised learning on the CIFAR-10 training dataset. The loss during pre-training rapidly decreases in the first couple of epochs before stabilizing around -0.8 and decreasing slowly before ending on a loss of -0.85 after 900 epochs. As mentioned in section 4.4, the negative cosine similarity is bounded between 0 and -1, where similar outputs are closer to -1.

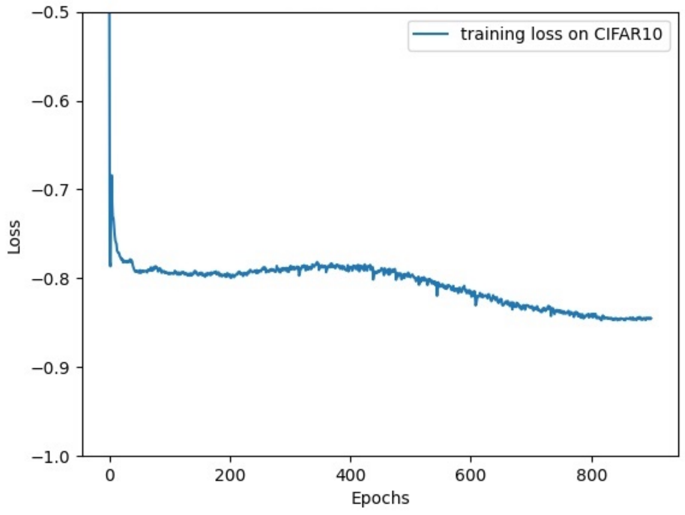
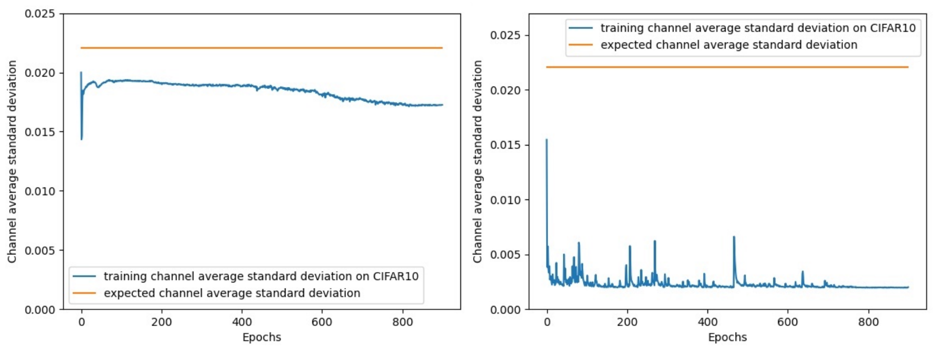


Figure 5.1: Loss values during pre-training on **CIFAR-10**.

Figure 5.2a shows the channel average standard deviation for the SimSiam model while pre-training on the CIFAR-10 training dataset. The dimension size of the output from the projection layer is 2048. Therefore, the expected channel average standard deviation is about 0.22, shown in orange. The channel average standard deviation for the SimSiam model during pre-training can be seen hovering around 0.020 and 0.015, before ending at 0.017 after 900 epochs.

The SimSiam model was also pre-trained on CIFAR-10 with the stop-gradient removed, allowing gradients to flow through both branches. Figure 5.2b records the channel average standard deviation during pre-training of the SimSiam model with no stop-gradient. With the stop-gradient removed, the channel average standard deviation quickly decreases towards zero, before stabilizing and ending at 0.002 after 900 epochs.

The ResNet18 backbone used during the SimSiam self-supervised training on CIFAR-10, was linearly evaluated on the CIFAR-10 dataset. Figure 5.3 depicts the performance of the self-supervised ResNet18 backbone during linear fine-tuning on the CIFAR-10 dataset. The backbone is able to achieve an accuracy of 75.80% on the training data and an accuracy of 75.14% on the validation data after 100 epochs of training.



(a) With stop-gradient (b) Without stop-gradient

Figure 5.2: Channel average standard deviation during self-supervised learning on **CIFAR-10**.

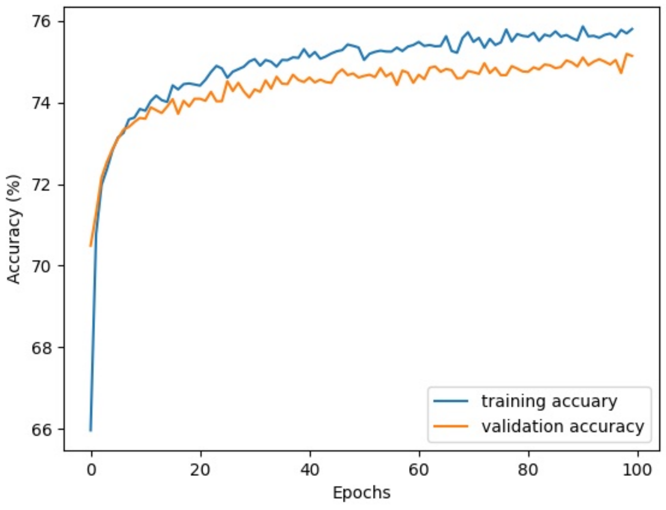
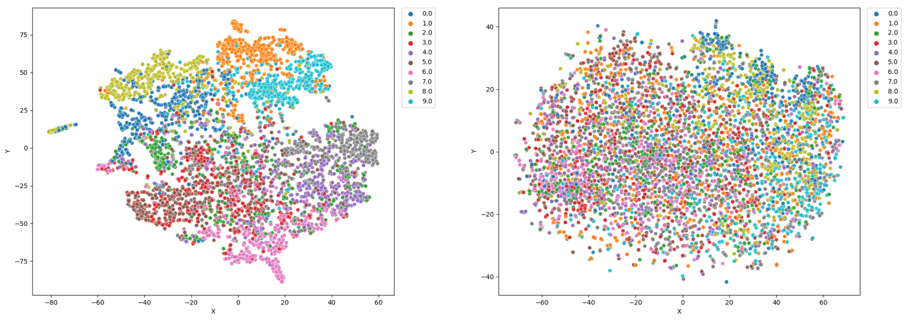


Figure 5.3: Accuracy of linear evaluation of ResNet18 on CIFAR-10.



(a) After self-supervision.

(b) Random initialization

Figure 5.4: Comparing t-SNE plots over Renset18 embeddings on **CIFAR-10**.

As mentioned in section 4.5.1, t-SNE was used on the embeddings created by the self-supervised ResNet18 to visualize how the different samples were clustered. Figure 5.4 shows a comparison between the t-SNE plot from the self-supervised backbone and a ResNet18 initialized with random weights, where each class of CIFAR-10 is color coded. From comparing the different plots, one can see that the plot generated from the self-supervised ResNet18 embeddings has a more defined clustering between the different classes than the plot generated from a ResNet18 initialized with random weights. Still, the plot generated with the embeddings from the self-supervised backbone does not have all the classes completely separated, with the green, red, purple, and gray classes not being clearly separated but are bounded within the same area.

5.1.2 Study of Different Individual Augmentations

By using the self-supervised architecture to pre-train the ResNet18 backbone with different data augmentations, it is possible to test how each augmentation affects the performance of the backbone on the downstream task. As mentioned in section 4.5.2, each augmentation was used to pre-train seven models on a subset of the unlabeled bird vocalization dataset, and each model was linearly fine-tuned five times. The results of these tests can be seen in table 5.1 as the average and standard deviation over all the different fine-tuned models on the validation dataset.

Table 5.1: Linear evaluation for the different augmentations on the supervised dataset.

Augmentations	Accuracy (%)	Standard deviation
Crop	**24.69**	**3.33**
Time mask	19.07	2.46
Frequency mask	19.71	1.75
Noise	19.66	1.06
Background audio	**17.49**	1.61

The results in table 5.1 show that the crop augmentation gives the highest accuracy out of all the augmentations with 24.69% on average over the validation data. Cropping also has the highest standard deviation with 3.33. Time masking, frequency masking, and the addition of noise perform similarly to each other. The addition of background audio has the lowest average accuracy of the augmentations, with an average of 17.49% on the validation set.

Figure 5.5 depicts the average accuracy on the validation dataset while linearly fine-tuning with the ResNet18 backbones. From figure 5.5, one can see that cropping achieves the highest average accuracy regardless of the number of epochs. The four other augmentations are closer to each other during the fine-tuning. One can also see that the augmentations start off with similar performances, between 11.63% and 14.34% after the first epoch, with the difference increasing as the number of epochs increases.

From these results, a decision can be made on which augmentations to further test and combine with each other. Since cropping performed significantly better than the other augmentations, it will be one of the three augmentations to be tested further. The augmentation of adding background audio performed the worst and will, therefore, not be tested further. Time masking, frequency masking, and adding noise performed similarly during the test, making it harder to choose between them. Because the time mask augmentation has the lowest accuracy and the highest standard deviation of the three, it will not be used in further testing. This leaves cropping, frequency masking, and adding noise to be the three augmentations used in further testing.

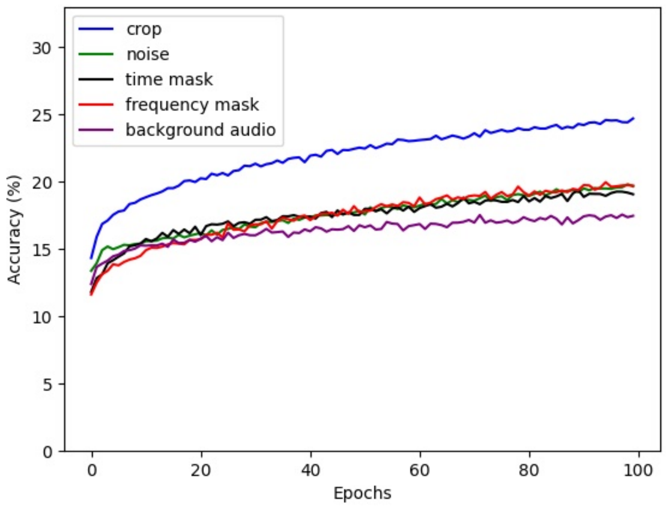


Figure 5.5: Accuracy from linear evaluation of the different **individual** augmentations.

5.1.3 Study of Combinations of Augmentations

After choosing the three augmentations, self-supervised models using combinations of the augmentations were trained and linearly fine-tuned on the manually labeled dataset with the same setup previously used for the testing of individual augmentations. The result of these tests can be seen in table 5.2 as the average and standard deviation over the validation set.

The results in table 5.2 show that combining the crop augmentation with the noise augmentation gives the highest accuracy out of all augmentation combinations, with an average accuracy of 35.66%. This combination of augmentations also has the highest standard deviation of 3.71, slightly higher than cropping alone. The combination of crop and frequency mask also performs better than only cropping and only frequency masking, with an average accuracy of 25.86%

and a standard deviation of 1.73. The same holds for the combination of noise and frequency masking, with an increase in accuracy of 1.56 percentage points compared to using only frequency masking.

Table 5.2: Linear evaluation for the different combinations augmentations on the supervised dataset.

Augmentations	Accuracy (%)	Standard deviation
Crop + Noise	**35.66**	**3.71**
Crop + Frequency mask	25.86	1.73
Noise + Frequency mask	21.27	0.80

Figure 5.6 depicts the average accuracy of the different combinations of augmentations on the validation dataset. The backbones pre-trained with the combinations of augmentations start with similar performances between 12.67% and 15.36%. This starting point is similar to the single augmentations that can be seen in figure 5.5, but during training, the accuracy increases faster for all the models using a combination of augmentations compared to the models with a single augmentation.

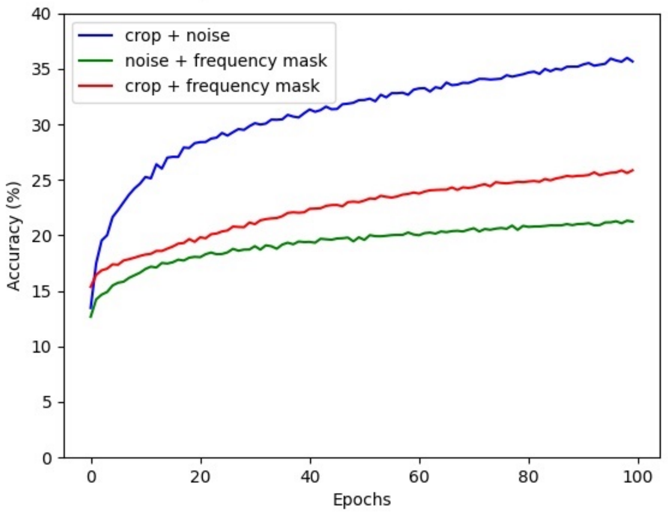


Figure 5.6: Accuracy from linear evaluation from **combinations** of augmentations.

5.1.4 Comparison Between Self-Supervision and ImageNet

From comparing different combinations of augmentations, the combination of cropping and adding noise was chosen for self-supervised learning on 100% of the unlabeled dataset. After doing self-supervised learning on the whole unlabeled dataset, linear fine-tuning and fine-tuning of the whole models were conducted, varying the size of the fine-tuning dataset. These results were compared to an identical ResNet18 backbone pre-trained on ImageNet.

For the linear evaluation of the self-supervised ResNet18 models and the ResNet18 model pre-trained on ImageNet, it is clear that the model pre-trained on ImageNet outperforms the self-supervised models, both for 100 and 200 epochs. Table 5.3 and 5.4 shows that for all the different amounts of training data, the model pre-trained on ImageNet achieves significantly higher accuracies, with the highest difference being 28.33% percentage points when linearly fine-tuning with 25% of the labeled training data for 100 epochs. The standard deviation is higher for the self-supervised models compared to the model pre-trained on ImageNet, with the highest standard deviation being 2.38 when linearly fine-tuning on 100% of the training data on 200 epochs. The size of the training data has a significant effect on the models, with the models performing better with larger training sets. The average validation accuracy for the self-supervised models when training on 100% of the training data increases from 54.53% to 58.42% when running for 200 epochs. In contrast, the accuracy of the model pre-trained on ImageNet does not change significantly from 100 epochs to 200 epochs.

Table 5.3: **Linear evaluation** of pre-training with self-supervision and pre-training with ImageNet using different amounts of training data. Linearly fine-tuning for **100** epochs.

Pre-training	Portion of training data (%)	Accuracy (%)	Standard deviation
Self-supervised	100	**54.53**	**2.32**
	50	49.32	2.56
	25	44.05	2.61
	10	33.38	2.17
	5	30.74	1.67
ImageNet	100	**76.47**	**0.68**
	50	73.94	0.47
	25	72.38	0.81
	10	61.47	0.92
	5	53.59	1.00

Table 5.4: **Linear evaluation** of pre-training with self-supervision and pre-training with ImageNet using different amounts of training data. Linearly fine-tuning for **200** epochs.

Pre-training	Portion of training data (%)	Accuracy (%)	Standard deviation
Self-supervised	100	**58.42**	**2.38**
	50	53.37	2.30
	25	48.41	3.27
	10	36.47	1.83
	5	33.93	1.53
ImageNet	100	**76.06**	**0.49**
	50	73.55	0.47
	25	72.18	0.54
	10	62.07	0.85
	5	53.56	0.88

Figure 5.7 shows the average accuracy and standard deviation of the models during linear fine-tuning. The model pre-trained on ImageNet has a consistently small standard deviation during training, while the self-supervised models' standard deviation increases with higher epochs. The average accuracy of the self-supervised models does not seem to plateau after 200 epochs, while the model pre-trained on ImageNet plateaued after 50 epochs.

For the comparisons where the whole models were fine-tuned on the training dataset, the ResNet18 model pre-trained on ImageNet outperforms the self-supervised models, as shown in table 5.5 and 5.6. For all the different amounts of training data, the model pre-trained on ImageNet archives higher accuracies. Still, the difference between the models is smaller than the results from table 5.3 and 5.4. The largest difference between the models is 9.70% when using 10% of the training set for 100 epochs. The impact from the number of epochs is smaller for the self-supervised models when fine-tuning the whole model compared to linear fine-tuning, with the biggest increase being from an average validation accuracy of 59.78% to 63.26% using 10% of the training data.

Figure 5.8 shows the calculated negative cosine similarity loss for the seven models during the self-supervised learning of the ResNet18 backbones on the large unlabeled bird vocalization dataset. All the models have a similar loss pattern. The loss begins around -0.92 and ends at -0.98 after 100 epochs of self-supervision.

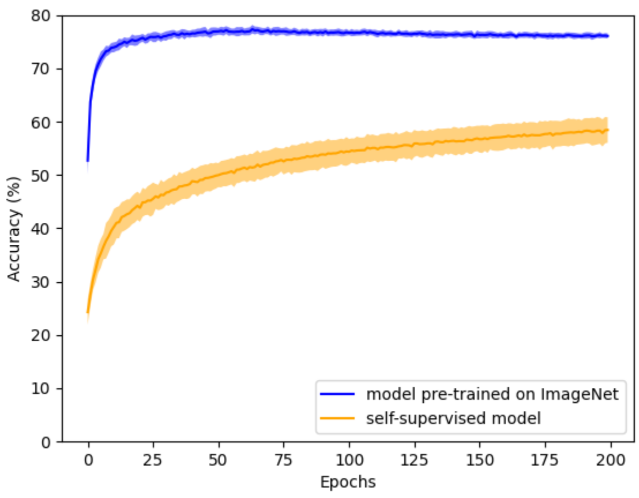


Figure 5.7: Validation Accuracy of the self-supervised and ImageNet models during **linear evaluation** using 100% of the training data for **200** epochs.

Table 5.5: **Full fine-tuning** comparison of pre-training with self-supervision and ImageNet on different amounts of training data. Fine-tuning for **100** epochs.

Pre-training	Portion of training data (%)	Accuracy (%)	Standard deviation
Self-supervised	100	**90.03**	**1.17**
	50	84.75	2.24
	25	78.18	2.37
	10	**59.78**	3.99
	5	51.79	2.74
ImageNet	100	**92.00**	**1.45**
	50	89.82	1.31
	25	85.71	2.00
	10	69.48	2.97
	5	59.21	2.88

Table 5.6: **Full fine-tuning** comparison of pre-training with self-supervision and ImageNet on different amounts of training data.Fine-tuning for **200** epochs.

Pre-training	Portion of training data (%)	Accuracy (%)	Standard deviation
Self-supervised	100	**90.64**	**0.98**
	50	85.94	1.58
	25	79.05	2.40
	10	**63.26**	2.82
	5	52.91	3.75
ImageNet	100	**92.22**	**1.26**
	50	89.55	3.17
	25	86.03	3.87
	10	68.91	2.65
	5	60.71	2.88

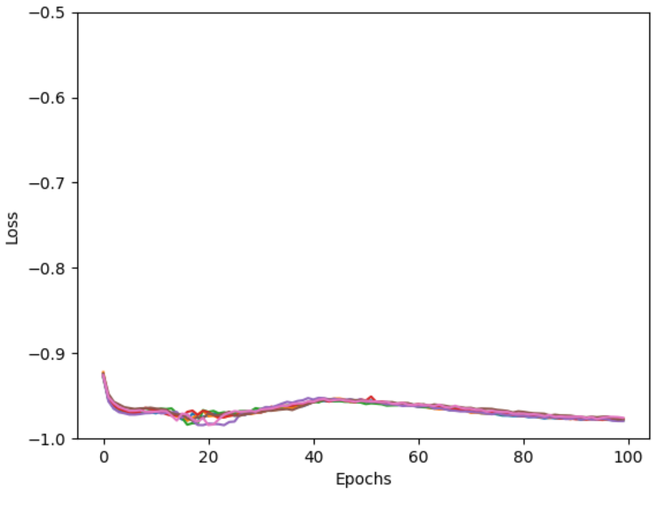


Figure 5.8: Loss values from the seven models during self-supervised learning on the **large unlabeled bird vocalization dataset**. Using the crop and noise augmentations.

Figure 5.9 shows the channel average standard deviation of the seven models during the self-supervised learning on the large unlabeled bird vocalization dataset. All the models have a similar channel average standard deviation pattern, ending on a channel average standard deviation of 0.019.

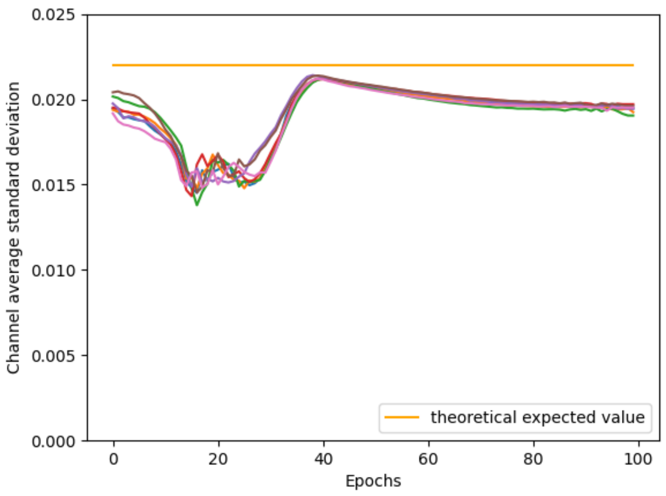
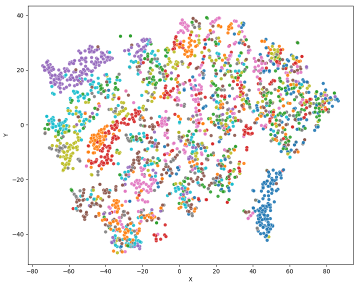
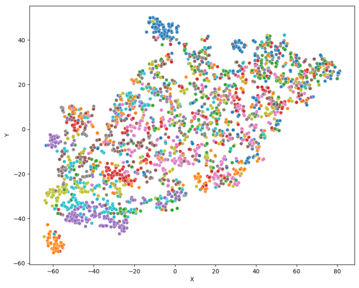


Figure 5.9: Channel average standard deviation from the seven models during self-supervised learning on the **unlabeled bird vocalization dataset**.

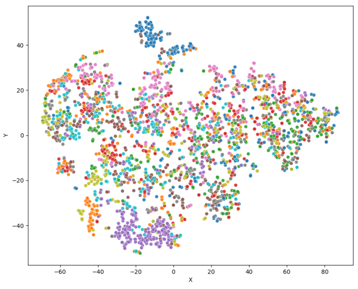
Figures 5.10 and 5.11 show t-SNE plots of the embeddings produced by the self-supervised ResNet18 backbones and the ResNet18 pre-trained on ImageNet. The embeddings are created by passing the manually labeled dataset through the ResNet18 models. Each bird vocalization class is color coded. Figure 5.10 shows no clear clustering for any of the seven self-supervised models except for parts of the blue and purple classes. For the embeddings produced by the ResNet18 pre-trained on ImageNet, more class-based clusters can be seen, but many of the samples are still not clustered well, with clusters blending together.



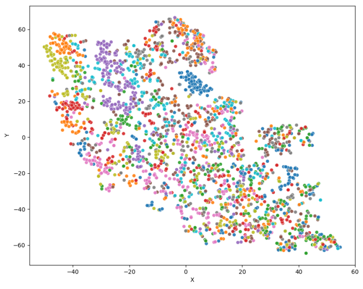
(a)



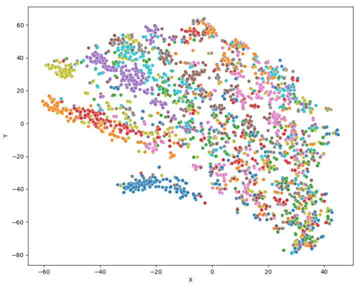
(b)



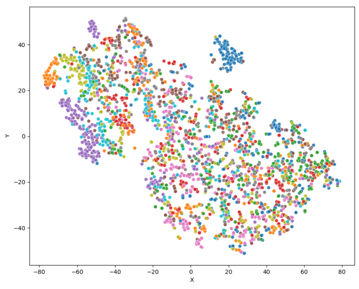
(c)



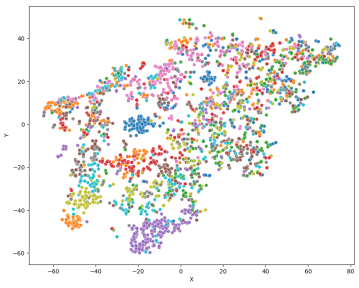
(d)



(e)



(f)



(g)

Figure 5.10: T-SNE plots of the embeddings from the seven **self-supervised models**.

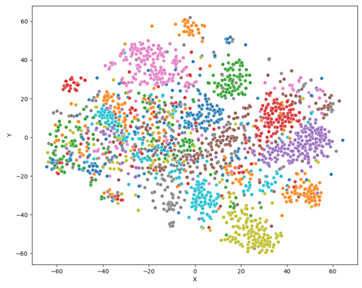


Figure 5.11: T-SNE plot of the embeddings from the ResNet18 **pre-trained on ImageNet**.

5.2 Analysis

5.2.1 Validation of Implementation

From the results displayed in section 5.1, one can argue that this thesis implementation of the SimSiam architecture is correct. The implemented training scheme displays the expected behavior with respect to the stop-gradient detailed in the SimSiam paper (X. Chen and He 2020). When the stop-gradient is applied, the output of the encoder does not collapse but stays close to the theoretically expected value of 0.02209708691. Whereas when the stop-gradient is removed, the output of the encoder collapse to similar outputs regardless of input, as shown in figure 5.2.

The results from linearly evaluating the self-supervised model on the CIFAR-10 test set show that the ResNet18 backbone is able to learn good feature extractions. The self-supervised ResNet18 accuracy starts at 66% after one epoch and ends at 75% after 100 epochs, indicating that the self-supervised ResNet18 backbone produces good embeddings that the last linear layer can make use of. From figure 5.4, it is clear that the self-supervised ResNet18 backbone is able to cluster embeddings from the same class better than a ResNet18 initialized with random weights. Although most of the classes are not completely separated from each other in the self-supervised ResNet18 embeddings.

With these results, it is reasonable to assume that the SimSiam architecture is implemented correctly. Therefore, the validity of the implementation will not be discussed in future subsections. The results from the validation experiment will be used to compare the results of the other experiments to discuss the system's behavior.

5.2.2 Study of Different Individual Augmentations

According to the results presented in table 5.1, the cropping augmentation performs the best out of all the single augmentations, averaging an accuracy of 24.69% on the validation set. This result is in line with previous self-supervision papers like SimCLR and BYOL-A, with both of these papers reporting cropping as the best performing augmentation in their ablation studies (T. Chen et al. 2020) (Niizumi et al. 2021). From looking at figure 5.5, the cropping augmentations also start at a higher accuracy after the first epoch compared to the other augmentations. This could be a sign that the embeddings produced by the models pre-trained with the crop augmentation are easier for the last linear layer to separate and classify, as it takes fewer training steps to increase its performance.

The crop augmentation has a higher standard deviation when calculating the average accuracy over the seven self-supervised models. The larger standard deviation could be explained by the crop augmentation being the strongest augmentation, changing the view of the original spectrogram more than any of the other augmentations. From a visual inspection of the unlabeled bird vocalization dataset, shown in figure A.2 in the appendix, one can see that the bird vocalizations often are not present in the entirety of the spectrogram. Therefore, the bird vocalization might not be present in the spectrogram after cropping and resizing it. This could lead to worse performance on the downstream task. Since the experiment self-supervised seven backbones, the amount of cropping leading to empty spectrograms during each self-supervision could vary, leading to varying performance between the seven models.

Although finding cropping to be the highest performing augmentation is in line with BYOL-A, the performance of cropping in BYOL-A and this thesis differ significantly. It is important to note that the linear evaluation done with BYOL-A and the models in this thesis use different datasets and is, therefore, not an ideal numerical comparison. The UrbanSound8K (US8K) dataset used in BYOL-A is the most similar to the manually labeled bird vocalization dataset used in this thesis. The US8K dataset has 10 different audio scene classes and 8,732 samples, while the manually labeled dataset has 10 bird vocalization classes and 2,500 samples. On the US8K dataset, BYOL-A achieves 77.1% on linear evaluation, while the self-supervised model pre-trained with cropping in this thesis achieves 24.69% on the manually labeled bird vocalization dataset.

The difference in accuracy for the cropping augmentation between BYOL-A and this thesis can be the result of several factors. One explanation could be that the crop augmentation is not as effective on the bird vocalization dataset as on the AudioSet used by BYOL-A. As mentioned earlier, the bird vocalizations in the unlabeled dataset often do not cover the whole spectrogram. Therefore a crop might not contain any bird vocalizations, making the augmentation less effective. Another plausible explanation could be the quality of the unlabeled dataset. The unlabeled dataset could contain too many spectrograms that do not contain any bird vocalizations, as subfigure A.2g in the appendix depicts. Since the unlabeled dataset is generated by taking longer audio recordings and splitting them into smaller segments, as described in section 4.2, there is a possibility that many of the spectrograms generated do not contain bird vocalizations. If there is a significant amount of empty spectrograms, the backbone's ability to learn specific features of bird vocalization could be negatively impacted.

The augmentation of adding background audio performed the worst out of all individual augmentations. The background audio augmentation was included in this thesis because of its results during the ablation study of BYOL-A. A possible explanation for why the background audio augmentation performed worse than expected is that the audio recordings constituting the unlabeled dataset are field recordings. These field recordings often have natural background audio, as birds and other animals can often be heard in the background. Therefore, adding more background audio to the recording might not impact the view of the spectrogram in a meaningful way, producing a sub-optimal self-supervision task.

5.2.3 Study of Combinations of Augmentations

The results from combining two augmentations, displayed in table 5.2, show that all the tested combinations of augmentations outperform their single augmentation counterpart. This is in line with the results from the papers by Chen et al, Grill et al, and Niizumi et al, finding that contrastive learning and non-contrastive learning benefits from using combinations of augmentations (T. Chen et al. 2020) (Grill et al. 2020) (Niizumi et al. 2021). The same seems to be true for the self-supervision of bird vocalizations. According to Tian et al., good augmentations generates views that retain task-relevant information and reduce irrelevant information (Tian et al. 2020). The reason for the higher accuracies for combinations of augmentations in this thesis might be that applying more augmentations lowers the amount of irrelevant information from the views, making the encoder focus more on the task-relevant information.

In this thesis, the combination of augmentations giving the highest accuracy on the downstream task was cropping and adding noise. Even though cropping and adding noise performed better than any other augmentation setup, the starting accuracy after one epoch was the same as the other augmentations, seen in the figures 5.5 and 5.6. If the backbone was able to learn good embeddings, it would be expected for the validation accuracy to start at a higher accuracy after one epoch, as is the case with the validation implementation depicted in figure 5.3. Instead, the accuracy of cropping and noise starts near the other augmentations and increases faster during the first couple of epochs. This could be because the backbones that were self-supervised with cropping and noise were not able to learn good feature extractions to produce clustered embeddings but were able to learn partial features, making it easier for the linear layer to learn faster during fine-tuning.

5.2.4 Comparison Between Self-Supervision and ImageNet

From the comparison of the self-supervised models pre-trained on the whole unlabeled bird vocalization dataset and the model pre-trained on ImageNet, it is clear that the self-supervised models are inferior to the model pre-trained on ImageNet on all tasks. There are many different factors that could explain these results, and these are discussed below.

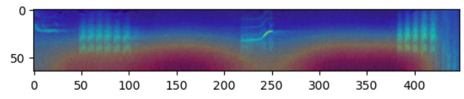
ImageNet achieves surprisingly high accuracy on the validation set for the different training dataset sizes, managing an average of 53.56% when linearly fine-tuning on only 5% of the original training dataset size. The model also starts at a high validation accuracy when training on 100% of the training set, reaching an accuracy above 50% after the first epoch of linear fine-tuning. This behavior is similar to the self-supervised model pre-trained on CIFAR-10, shown in figure 5.3. The large increase in accuracy could be a sign that the backbone is able to produce useful embeddings for the last linear layer, making training of the last layer easier and faster during fine-tuning. An explanation of this could be that some of the feature extractions learned from training on images are still applicable for spectrograms of bird vocalizations, such as edge detection or certain pattern extractions.

Although the ResNet18 pre-trained on ImageNet performed well on the linear evaluation, fine-tuning the whole model improves the average accuracy from 76.06% to 92.22% when training on 100% of the training data. This shows that the embeddings produced by a frozen ResNet18 model pre-trained on ImageNet are not optimal for bird vocalization classification, showing that another set of weights can achieve higher accuracies. It is important to note that the classification task used in this paper is relatively easy, with only 10 classes of birds. The performance of the model pre-trained on ImageNet might differ if the number of classes were to increase.

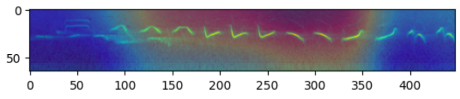
From figure 5.7, one can see that the average validation accuracy of the self-supervised models starts lower than the model pre-trained on ImageNet. This could be a sign that the self-supervised backbones did not learn beneficial feature extractions during self-supervision, leading to embeddings that are hard to classify for the last linear layer. Therefore, the last linear layer would need more training steps to increase the model's accuracy. This could explain why there is an increase in validation accuracy between training 100 and 200 epochs for the self-supervised models, while the accuracy of the model pre-trained on ImageNet stays more or less constant. The t-SNE plots in figure 5.10 support the explanation of hard-to-classify embeddings, with figure 5.10 showing that there is no clear clustering among the majority of the classes in the embeddings produced by the seven self-supervised models.

One factor for the poor performance of the self-supervised models could be the augmentations applied during self-supervised learning. As shown in figure A.3, different augmentations during self-supervision can lead to large differences in performance on the downstream task. Even though cropping and adding noise were the best augmentations in this thesis, they might not be adequate augmentations for learning bird vocalization-specific feature extractions. If the augmentations are too weak and do not remove information irrelevant to bird vocalization, the backbone might learn to look at task-irrelevant information in the spectrogram when generating the embeddings. This could lead to the loss decreasing rapidly during self-supervised learning, as seen in figure 5.8, while not learning valuable feature extractions, leading to poor performance on the downstream task. The attention heatmaps depicted in figure 5.12 support this hypothesis, showing the self-supervised backbones often fail to focus on the bird vocalizations when trying to classify the bird species.

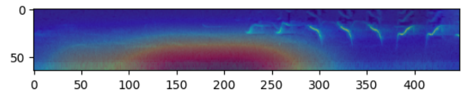
During self-supervision on the large unlabeled dataset, the seven self-supervised models quickly achieve low loss values, as depicted in figure 5.8. Each self-supervised model achieves similar loss values, suggesting that this behavior is likely not an outlier. The models achieve a relatively low loss compared to the self-supervised model used on CIFAR-10, shown in figure 5.1. Because of the previously discussed poor performance, this likely does not come from the backbones' ability to learn task-important features. The more likely explanation is that the self-supervised models have found a way to satisfy the similarity metric without learning downstream task-specific features. When looking at the channel average standard deviation, depicted in figure 5.9, the values during self-supervised learning do not tend toward zero, as depicted in subfigure 5.2b, suggesting that the models do not reach a collapsed solution. Instead, the self-supervised model might produce a partially collapsed solution. According to Li et al., a partial dimensional collapse is when parts of the output vector are constant across the dataset, or parts of the output vector covary with other parts of the vector (Li et al. 2022). These partial collapses would not show up when calculating the channel average standard deviation but would impact the performance of the backbone on the downstream task. Therefore, it could be that the reason for the poor performance of the self-supervised models is partial dimensional collapse.



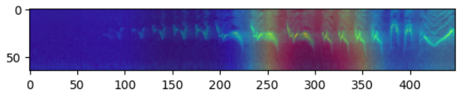
(a) Wrongly classified



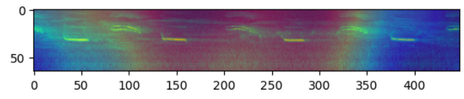
(b) Correctly classified



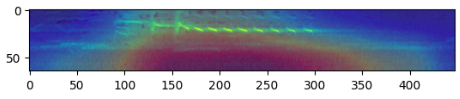
(c) Wrongly classified



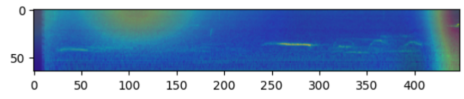
(d) Correctly classified



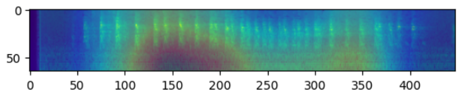
(e) Wrongly classified



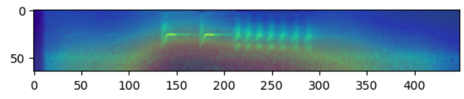
(f) Correctly classified



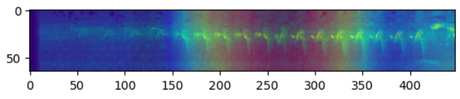
(g) Wrongly classified



(h) Correctly classified



(i) Wrongly classified



(j) Correctly classified

Figure 5.12: Examples of attention heatmaps for one of the self-supervised ResNet18 models when classifying bird species. The ResNet18 pre-trained using self-supervised learning often does not focus on the focal bird vocalization when classifying. Focusing instead on the edges of the spectrogram. The heatmaps were made by using the Grad-CAM technique (Selvaraju et al. 2020).

Another factor that could explain the self-supervised models' poor performance and inability to cluster embeddings could be the unlabeled dataset itself. Unlike datasets such as CIFAR-10 and ImagenNet used in previously discussed papers, the large unlabeled dataset used in this thesis is truly unlabeled. As mentioned in section 4.1, the large dataset was generated by downloading bird vocalization recordings and splitting them up into smaller spectrograms. This means that some of the spectrograms might not contain any bird vocalizations, and the dataset can contain a large amount of noise. When training neural networks, the quality of the dataset used during training has a huge impact on the model's performance. If the unlabeled bird vocalization dataset contained too many empty spectrograms or too much noise and not enough bird vocalizations, then learning bird vocalization-specific feature extractions would be difficult for the self-supervised model.

The use of a relatively small convolutional network as a backbone could also be a factor for the self-supervised models' low performance. According to Li et al. SimSiam is extraordinarily sensitive to model size, with smaller backbones being more prone to partially collapse on larger and more complex datasets, as depicted in A.4 in the appendix (Li et al. 2022). Therefore, the ResNet18 backbone might be too small for the large unlabeled dataset, leading to partial collapse and lower performance. This could also explain why SimSiam with a ResNet18 performs better on the CIFAR-10 dataset than on the bird vocalization task, as the dataset is smaller.

5.3 Summary

The best combination of augmentations tested in this thesis was cropping and adding noise to the spectrogram. The self-supervised models trained on a large unlabeled bird vocalization dataset with these augmentations still performed worse than an equivalent model pre-trained on ImageNet. This most likely comes from the self-supervised scheme's inability to learn the ResNet18 backbone relevant feature extractions. There can be many different explanations for this. The self-supervision scheme could have experienced partial collapse during self-supervised learning because of the dataset and the backbone size. The unlabeled dataset could have also contained too many samples not of bird vocalizations, making it hard for the self-supervised model to learn relevant features. Lastly, the augmentations applied during self-supervised learning might not have been effective for learning bird vocalization-specific feature extractions.

Chapter 6

Conclusion

This thesis used the SimSiam architecture to test the use of self-supervised learning on the bird vocalization classification task. The goal set forth by this thesis was to investigate the effectiveness of self-supervised learning on the bird vocalization classification problem. Different data augmentations were tested, and self-supervised models were compared against a model pre-trained on images. Section 6.1 in this chapter will present an overview of the thesis. Section 6.2 discusses the findings of this thesis with respect to the goal and research questions laid out in section 1.2. In section 6.3, the contributions of this thesis to the bird vocalization classification field are discussed. Section 6.4 gives suggestions for future work for self-supervised learning on bird vocalization classification. Lastly, a few final words are given in section 6.5

6.1 Overview

Chapter 1 introduced why bird classification and detection are important and explained the motivation for testing self-supervised learning on the bird vocalization classification problem. This chapter also laid out the goal of the thesis and the two research questions used to achieve the goal.

Chapter 2 contained information about bird vocalization and popular audio representations for machine learning. An introduction to convolutional neural networks, self-supervised learning, contrastive learning, and non-contrastive learning was also presented to give the reader the background knowledge needed for this thesis.

A review of related work was conducted in chapter 3. The chapter starts by introducing papers related to bird vocalization classification, reviewing papers from state-of-the-art bird vocalization detection, and papers with results ranking high in bird vocalization competitions. Then the chapter reviewed papers related to self-supervised learning for both the image and audio domains. Lastly, papers concerning transfer learning from the image domain to the audio domain were reviewed.

Chapter 4 explained in detail the method and architecture used to answer the research questions presented in chapter 1. The data collection method was detailed at the start of the chapter, explaining how the bird vocalizations were downloaded from Xeno-canto. Then a description of the pre-processing pipeline was given, describing the methods for producing mel-spectrograms from bird audio. The different data augmentations used during the self-supervised learning were also described before the specific SimSiam architecture used to conduct the experiments was detailed. Lastly, a description of the experimental setup for the three main experiments was given.

The key results of the experiments were displayed in Chapter 5. The first part of the chapter presented the results of running the validation experiment, showing that the implementation of SimSiam was most likely correct. After this, the chapter lays out the results of using the SimSiam architecture with different individual augmentations and combinations of augmentations for deciding the final augmentations for the self-supervised learning scheme. The last section of the results compares the downstream performance of self-supervised models using cropping and noise augmentations against a model pre-trained on ImageNet, using linear fine-tuning and fine-tuning the whole model. At the end of the chapter, the results were analyzed. From the results and analysis, the best performing combination of augmentations was cropping and adding noise. The results also showed that the self-supervision scheme set up in this thesis did not perform as well as the model pre-trained on ImageNet on the downstream task of bird vocalization classification.

6.2 Discussion

The goal of this thesis was to investigate the effectiveness of self-supervised learning on the bird vocalization classification problem. This section discusses the key implications of the results and analysis with respect to the thesis research questions.

6.2.1 Reseach Question 1

What augmentations for the self-supervised model give the best results?

From the results and analysis in chapter 5, the best performing single augmentation was Cropping with an average accuracy of 24.69%. When testing combinations of augmentations, all the combinations tested performed better than any single augmentation. The best combination of augmentations was cropping and adding noise with an average accuracy of 35.66%. Although the combination of cropping and adding noise performed the best in this thesis, it does not mean that this is the optimal combination of augmentations. When comparing its performance to BYOL-A with a similar dataset, there is a substantial difference, with BYOL-A managing an accuracy of 77.1% while the cropping and noise augmentations in this thesis average an accuracy of 35.66%. Still, the results produced by the augmentation experiments show that the self-supervised model benefits from using more than one augmentation. This makes a combination of augmentations the most likely augmentation scheme to perform well on contrastive or non-contrastive learning schemes intended for the downstream task of bird vocalization classification.

Another interesting result was that the augmentation of adding background noise to the spectrograms performed the worst out of all the single augmentations. This is in contrast to the results from BYOL-A, in which adding background audio to the spectrograms performed significantly better than adding random noise. The reason for this could be that there is already background noise in many of the bird vocalization spectrograms, and therefore, adding background audio does not augment the spectrograms significantly.

There can be many reasons why the augmentations of cropping and adding noise perform worse than expected on the self-supervised model. As mentioned, one reason could be that cropping and adding noise might not be suitable augmentation schemes for bird vocalization classification and that some other combinations not tested in this thesis might perform better. Another probable explanation, as mentioned in section 5.2.2, is that the dataset might not be ideal for self-supervision, as there may be too many spectrograms containing noice or that do not contain any bird vocalizations. If there are too many empty spectrograms, the backbone might not be able to sufficiently learn to extract features that are important for bird vocalization classification.

6.2.2 Research Question 2

> *How does the self-supervised model compare against a supervised model pre-trained on ImageNet, with linear fine-tuning and end-to-end tuning on different amounts of training data?*

From the results and analysis in chapter 5, it is clear that the model pre-trained on ImageNet outperforms the models using self-supervision. When comparing the models on the linear fine-tuning task, it is clear that the model pre-trained on ImageNet performs better, having higher accuracy regardless of training data size and the number of epochs. The model pre-trained on ImageNet also performs better than the self-supervised models in the experiments where the entire models were fine-tuned, showing that the weights from ImageNet are also a better weight initialization than the weights learned from self-supervision in this thesis.

As mentioned in section 5.2, there can be many reasons for the poor performance of the self-supervised model. The way the unlabeled dataset was constructed led to a possibly significant amount of noisy or empty spectrograms that could impact the effectiveness of the self-supervised learning scheme. Another possible reason could be that the augmentations chosen in this thesis were not appropriate for the bird vocalization classification task. It could also be that the relatively small size of the backbone used during self-supervision with SimSiam was not ideal for the size of the unlabeled dataset used. Most likely, a combination of these factors affects the performance of the self-supervised models constructed in this paper.

The results of this thesis can not say anything about the general performance of self-supervision on the bird vocalization task, as many different variables can be changed, such as backbone, self-supervised algorithm, datasets, and augmentations, that can change the performance of the self-supervised model. But the results of these experiments suggest that the use of multiple augmentations improves the performance on the downstream task. It also seems like using audio from Xeno-canto without a method for filtering out noisy and empty spectrograms leads to poor performance on the downstream task, as the model struggles to learn relevant features. It is important to note that the experiments in this thesis used a limited amount of runs to acquire statistical averages and standard deviations because of long training time and high hardware requirements. Several more runs and further experiments would be needed to verify any of the explanations given.

Although the self-supervised learning scheme implemented in this thesis did not do as well as a model pre-trained on ImageNet, another self-supervision scheme could likely manage similar or even surpass the performance of the model pre-trained on ImageNet. By testing more augmentations, backbones, and unlabeled dataset pre-processing pipelines, one is likely to find a better performing self-supervised learning scheme than the one laid out in this thesis.

6.3 Contributions

This thesis contributes to the field of bird vocalization classification in two aspects. The first contribution is the gathering of information through chapter 2 and chapter 3. Chapter 2 serves as an introduction to the information needed for working with self-supervised learning on the bird vocalization classification problem. This chapter describes popular self-supervised learning structures and audio representations for machine learning. Chapter 3 contains a detailed survey of state-of-the-art bird vocalization classification papers, self-supervised learning papers regarding both the image and audio domains, and papers concerning transfer learning from the image domain to the audio domain. These chapters give the reader needed background information for researching the intersection between self-supervised learning and bird vocalization classification.

The second contribution comes from this thesis implementation and experiments. To the best of the author's knowledge, this is the first paper using the non-contrastive learning scheme SimSiam on the bird vocalization classification task. Therefore, anyone wanting to research similar self-supervised learning implementations for the bird vocalization classification task could benefit from reading this thesis. The results and discussions from this thesis can be used as a starting point for other researchers exploring self-supervision on the bird vocalization classification problem.

6.4 Future Work

To the author's knowledge, this was the first attempt at self-supervised learning for bird vocalization classification. There are a lot of changes and improvements that can be explored for future research that builds on this thesis. Some possible changes for future work based on the results and discussions from this thesis are described below.

6.4.1 Using Different Unlabeled Dataset

The dataset used during self-supervised learning significantly impacts the backbone's performance on the downstream task. During the analysis and discussion, the unlabeled dataset was mentioned as a possible factor for the poor performance of the self-supervised models because of noise and empty spectrograms. For future work, it could be worth improving the quality of the unlabeled dataset. This could be done by finding or developing a metric for noisy spectrograms or detecting empty spectrograms and removing them.

Another approach to ensure a good quality of the unlabeled dataset could be to use a labeled audio dataset and remove the labels, such as in BYOL-A. By using previously labeled datasets, one can be assured that the amounts of noise or empty spectrograms would be a minimum. It could be interesting to test if doing self-supervision on large labeled audio datasets, such as AudioSet, increases the performance on the downstream task of bird vocalization classification.

6.4.2 Futher Testing of Augmentations

As mentioned in section 5.2, the augmentations applied during self-supervision have a significant impact on the performance of the downstream task. Due to hardware limitations and time constraints, this paper only tested five different augmentations and only did combinations of two augmentations at a time. Using more than two augmentations would most likely benefit the self-supervised backbones on the downstream task. Therefore, a good topic for future work within self-supervision for bird vocalization classification would be to test more data augmentations and test the application of more than two augmentations at a time during self-supervision.

6.4.3 Testing Different Self-Supervised Architectures

There are many different self-supervision architectures that can be used for self-supervision for the bird vocalization classification task. This thesis only considered the SimSiam architecture, as this architecture was less hardware-demanding and time-consuming to train. As mentioned in section 5.2, the SimSiam architecture used in this thesis did not perform well on the downstream task. The use of a larger backbone or a different self-supervised learning architecture might increase the downstream performance of the self-supervised model. As discussed by Li et al. (Li et al. 2022), SimSiam performs better on larger backbones. Therefore, it would be interesting to compare the performance of SimSiam on the bird vocalization classification task using different-sized backbones and see if there is a significant improvement in accuracy on the downstream task.

It would also be interesting to test other self-supervision architectures to see if there is a significant difference in downstream task performance between them. Self-supervised architectures that could be interesting to test are the contrastive architectures SimCLR and MoCo and the non-contrastive architecture BYOL. These self-supervised learning architectures use the same principle of generating two augmented views for two different branches. Therefore, one can use identical pre-processing pipelines across architectures, making the architecture the only variable. It is important to note that these implementations often require larger GPU memory and might take longer to train.

6.5 Final Words

This thesis has investigated the use of self-supervised learning for bird vocalization classification, introducing the intersection between bird vocalization and self-supervised learning. This thesis has given general introductions to how bird audio can be processed and how self-supervised learning can be used to leverage large unlabeled datasets. Novel experiments have also been conducted, using the SimSiam architecture with unlabeled bird vocalization data collected from Xeno-canto. The results from these experiments suggest that contrastive and non-contrastive learning on bird vocalizations benefits from using several data augmentations during self-supervision. The results also suggest that using unfiltered audio clips from Xeno-canto as the unlabeled dataset with the SimSiam architecture described in this thesis is not sufficient for a good performance on the downstream task of bird vocalization classification. This thesis has also laid out possible improvements and further work within the self-supervision for bird vocalization classification.

Appendix A

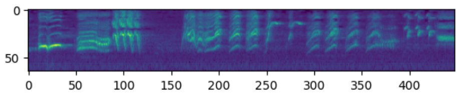
Additional Figures and Tables

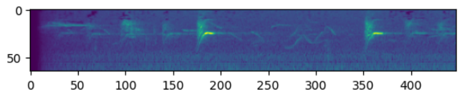
A.1 Manually Labeled Dataset

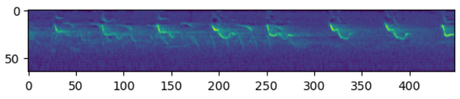
The labeled dataset has a size of 2514 instances, classified into 10 different bird classes. All the recordings used to generate this dataset originates from the United Kingdom. Each bird species is native to the United Kingdom but can also be found in large parts of Europe. Table A.1 presents the number of instances in each class in the dataset. An example spectrogram from each class can be seen in figure A.1. It is important to note that each bird can have several distinct vocalizations. Therefore, the spectrograms seen in figure A.1 are not a complete representation of all spectrograms in the labeled dataset.

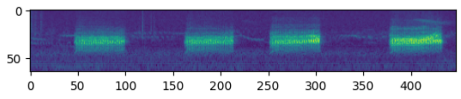
Table A.1: Number of instances for each class in the labeled dataset.

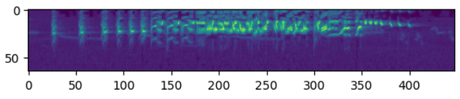
Bird class	Number of instances
Common Blackbird	251
Common Chaffinch	256
Common Chiffchaff	263
Common Whitethroat	250
Corn Bunting	250
Eurasian Blue Tit	253
Eurasian Tree Sparrow	255
Great Tit	227
Willow Warbler	257
Yellowhammer	252
Total	**2514**

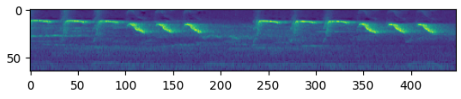

(a) Common Blackbird

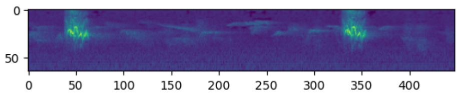

(b) Common Chaffinch

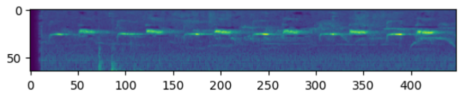

(c) Common Chiffchaff

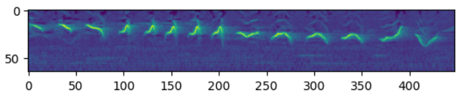

(d) Common Whitethroat


(e) Corn Bunting


(f) Eurasian Blue Tit


(g) Eurasian Tree Sparrow


(h) Great Tit


(i) Willow Warbler

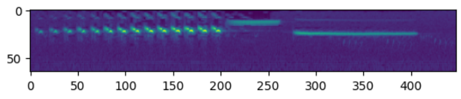
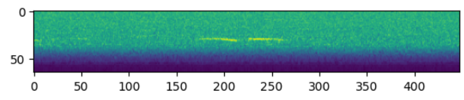
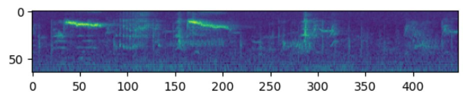
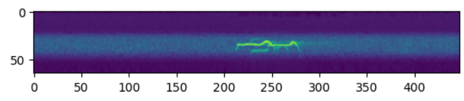

(j) Yellowhammer

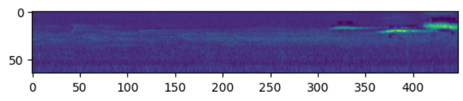
Figure A.1: Example of spectrograms from each class in the labeled dataset.

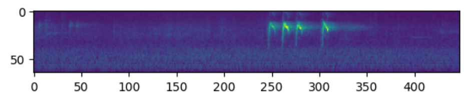
A.2 Unlabeled Dataset

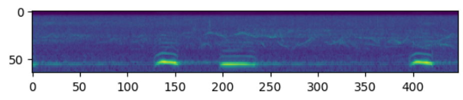

(a)

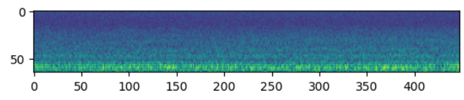

(b)

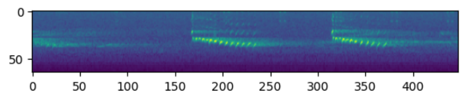

(c)

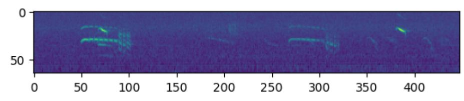

(d)

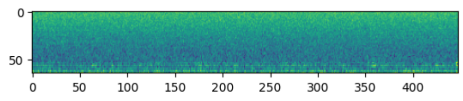

(e)

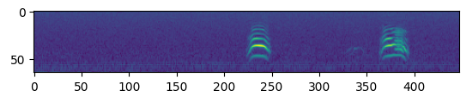

(f)


(g)


(h)


(i)


(j)


(k)

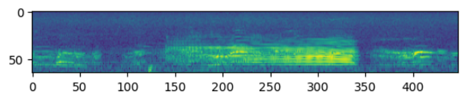
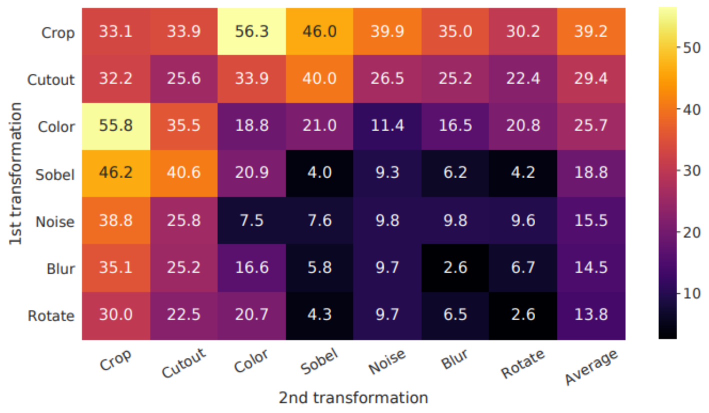

(l)

Figure A.2: Sample of spectrograms from the unlabeled dataset. The dataset contains noisy spectrograms and spectrograms without bird vocalizations, and the bird vocalizations often do not cover the whole spectrogram.

A.3 Figures From Other Papers



	Crop	Cutout	Color	Sobel	Noise	Blur	Rotate	Average
Crop	33.1	33.9	56.3	46.0	39.9	35.0	30.2	39.2
Cutout	32.2	25.6	33.9	40.0	26.5	25.2	22.4	29.4
Color	55.8	35.5	18.8	21.0	11.4	16.5	20.8	25.7
Sobel	46.2	40.6	20.9	4.0	9.3	6.2	4.2	18.8
Noise	38.8	25.8	7.5	7.6	9.8	9.8	9.6	15.5
Blur	35.1	25.2	16.6	5.8	9.7	2.6	6.7	14.5
Rotate	30.0	22.5	20.7	4.3	9.7	6.5	2.6	13.8

Figure A.3: Exploration of different data augmentation combinations for ImageNet on SimCLR. The values are the linear evaluation accuracy on ImageNet. The last column is the average over the row (T. Chen et al. 2020).

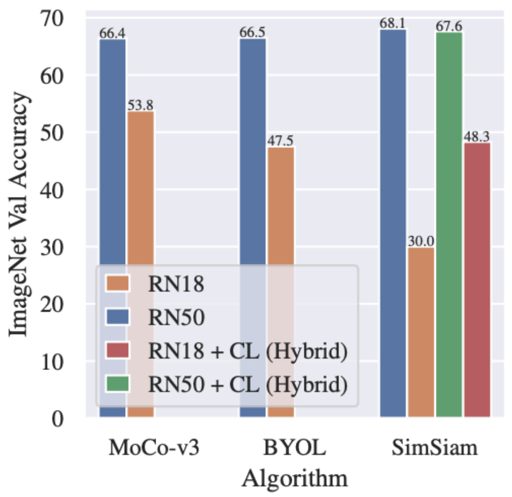


Figure A.4: Results from testing different contrastive and non-contrastive learning architectures on ImageNet with different sized backbones, with ResNet18 in brown and ResNet50 in blue (Li et al. 2022).

Bibliography

BirdClef competition 2021 (n.d.). URL: https://www.imageclef.org/BirdCLEF2021MainTask. (visited on 11/04/2022).

Bobay, Lucas R., Paul J. Taillie, and Christopher E. Moorman (Dec. 2018). "Use of autonomous recording units increased detection of a secretive marsh bird". en. In: *Journal of Field Ornithology* 89.4, pp. 384–392. ISSN: 0273-8570, 1557-9263. DOI: 10.1111/jofo.12274. URL: https://onlinelibrary.wiley.com/doi/10.1111/jofo.12274 (visited on 12/03/2022).

CEUR-WS (n.d.). URL: https://ceur-ws.org/. (visited on 11/24/2022).

Chen, Ting et al. (2020). "A Simple Framework for Contrastive Learning of Visual Representations". In: Publisher: arXiv Version Number: 3. DOI: 10.48550/ARXIV.2002.05709. URL: https://arxiv.org/abs/2002.05709 (visited on 11/08/2022).

Chen, Xinlei and Kaiming He (2020). "Exploring Simple Siamese Representation Learning". In: Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2011.10566. URL: https://arxiv.org/abs/2011.10566 (visited on 11/08/2022).

Clough, Lindsay (2020). "Autonomous Recording Units as an Alternative Method for Monitoring Songbirds". In: Publisher: University of Massachusetts Amherst. DOI: 10.7275/17574947. URL: https://scholarworks.umass.edu/masters_theses_2/923 (visited on 12/03/2022).

Cornell Birdcall Identification (n.d.). URL: https://www.kaggle.com/c/birdsong-recognition/overview. (visited on 11/04/2022).

Disabato, Simone et al. (Aug. 2021). "Birdsong Detection at the Edge with Deep Learning". In: *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*. Irvine, CA, USA: IEEE, pp. 9–16. ISBN: 978-1-66541-252-0. DOI: 10.1109/SMARTCOMP52413.2021.00022. URL: https://ieeexplore.ieee.org/document/9556234/ (visited on 11/14/2022).

Duraisamy, Sundararajan (2001). *Discrete Fourier Transform, The: Theory, Algorithms And Applications*. World Scientific. ISBN: 9789810245214. URL: https:

//search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=235873&site=ehost-live&scope=site.

European Commission. Directorate General for Environment., IUCN., and BirdLife International. (2022). *European red list of birds 2021.* eng. LU: Publications Office. URL: https://data.europa.eu/doi/10.2779/959320 (visited on 12/03/2022).

Foley, Hugh and Margaret Matlin (Aug. 2015). *Sensation and Perception.* en. 0th ed. Psychology Press. ISBN: 978-1-315-66506-1. DOI: 10.4324/9781315665061. URL: https://www.taylorfrancis.com/books/9781317350996 (visited on 12/10/2022).

Goëau, Hervé et al. (Sept. 2018). "Overview of BirdCLEF 2018: monospecies vs. soundscape bird identification". In: *CLEF 2018 - Conference and Labs of the Evaluation Forum.* Vol. 2125. CEUR Workshops Proceedings 9. Avignon, France. URL: https://hal.archives-ouvertes.fr/hal-02189229.

Grill, Jean-Bastien et al. (2020). "Bootstrap your own latent: A new approach to self-supervised Learning". In: Publisher: arXiv Version Number: 3. DOI: 10.48550/ARXIV.2006.07733. URL: https://arxiv.org/abs/2006.07733 (visited on 11/10/2022).

He, Kaiming, Haoqi Fan, et al. (2019). "Momentum Contrast for Unsupervised Visual Representation Learning". In: Publisher: arXiv Version Number: 3. DOI: 10.48550/ARXIV.1911.05722. URL: https://arxiv.org/abs/1911.05722 (visited on 11/10/2022).

He, Kaiming, Xiangyu Zhang, et al. (2015). "Deep Residual Learning for Image Recognition". In: Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385 (visited on 11/14/2022).

Howard, Andrew G. et al. (2017). "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.1704.04861. URL: https://arxiv.org/abs/1704.04861 (visited on 11/14/2022).

Huang, Gao et al. (2016). "Densely Connected Convolutional Networks". In: Publisher: arXiv Version Number: 5. DOI: 10.48550/ARXIV.1608.06993. URL: https://arxiv.org/abs/1608.06993 (visited on 11/14/2022).

Iakubovskii, Pavel (Apr. 2023). *Classification models Zoo - Keras (and TensorFlow Keras).* original-date: 2018-05-21T22:07:06Z. URL: https://github.com/qubvel/classification_models (visited on 04/27/2023).

ImageNet (ILSVRC2012) (n.d.). URL: https://www.image-net.org/challenges/LSVRC/2012/. (visited on 11/14/2022).

Incze, Agnes et al. (Sept. 2018). "Bird Sound Recognition Using a Convolutional Neural Network". In: *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY).* Subotica: IEEE, pp. 000295–000300.

ISBN: 978-1-5386-6841-2. DOI: `10.1109/SISY.2018.8524677`. URL: `https://ieeexplore.ieee.org/document/8524677/` (visited on 11/14/2022).

Jaiswal, Ashish et al. (2020). "A Survey on Contrastive Self-supervised Learning". In: Publisher: arXiv Version Number: 3. DOI: `10.48550/ARXIV.2011.00362`. URL: `https://arxiv.org/abs/2011.00362` (visited on 04/26/2023).

Kahl, Stefan (2020). *Identifying birds by sound: large-scale acoustic event recognition for avian activity monitoring.* en. Wissenschaftliche Schriftenreihe Dissertationen der Medieninformatik Band 10. Chemnitz: UniversitÃ¤tsverlag Chemnitz. ISBN: 978-3-96100-110-1.

Kahl, Stefan, Tom Denton, et al. (2021). "Overview of BirdCLEF 2021: Bird call identification in soundscape recordings". In: *Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021.* Ed. by Guglielmo Faggioli et al. Vol. 2936. CEUR Workshop Proceedings. CEUR-WS.org, pp. 1437–1450. URL: `http://ceur-ws.org/Vol-2936/paper-123.pdf`.

Kahl, Stefan, Connor M. Wood, et al. (Mar. 2021). "BirdNET: A deep learning solution for avian diversity monitoring". en. In: *Ecological Informatics* 61, p. 101236. ISSN: 15749541. DOI: `10.1016/j.ecoinf.2021.101236`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S1574954121000273` (visited on 12/03/2022).

Khosla, Aditya et al. (June 2011). "Novel Dataset for Fine-Grained Image Categorization". In: *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition.* Colorado Springs, CO.

Koskimies, Pertti (1989). "Birds as a tool in environmental monitoring". In: *Annales Zoologici Fennici* 26.3, pp. 153–166. ISSN: 0003455X, 17972450. URL: `http://www.jstor.org/stable/23734578` (visited on 12/03/2022).

Koushik, Jayanth (2016). "Understanding Convolutional Neural Networks". In: Publisher: arXiv Version Number: 1. DOI: `10.48550/ARXIV.1605.09081`. URL: `https://arxiv.org/abs/1605.09081` (visited on 11/14/2022).

Krizhevsky, Alex (2009). "Learning Multiple Layers of Features from Tiny Images". In.

Kumar, Anil (June 2003). "Acoustic communication in birds: Differences in songs and calls, their production and biological significance". en. In: *Resonance* 8.6, pp. 44–55. ISSN: 0971-8044, 0973-712X. DOI: `10.1007/BF02837868`. URL: `http://link.springer.com/10.1007/BF02837868` (visited on 10/29/2022).

Lasseck, Mario (2018). "Audio-based Bird Species Identification with Deep Convolutional Neural Networks". In: CEUR-WS.org. URL: `https://ceur-ws.org/Vol-2125/paper_140.pdf`.

Leakage (n.d.). URL: `https://mathworld.wolfram.com/Leakage.html`. (visited on 10/28/2022).

Li, Alexander C., Alexei A. Efros, and Deepak Pathak (2022). "Understanding Collapse in Non-Contrastive Siamese Representation Learning". In: Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2209.15007. URL: https://arxiv.org/abs/2209.15007 (visited on 05/25/2023).

Lindsay, Grace W. (Sept. 2021). "Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future". en. In: *Journal of Cognitive Neuroscience* 33.10, pp. 2017–2031. ISSN: 0898-929X, 1530-8898. DOI: 10.1162/jocn_a_01544. URL: https://direct.mit.edu/jocn/article/33/10/2017/97402/Convolutional-Neural-Networks-as-a-Model-of-the (visited on 10/26/2022).

Mertins, Alfred (1999). *Signal analysis: wavelets, filter banks, time-frequency transforms, and applications*. eng. English rev. ed. OCLC: 53925029. New York: J. Wiley. ISBN: 978-0-470-84183-9.

Murakami, Naoki, Hajime Tanaka, and Masataka Nishimor (2021). "Birdcall Identification Using CNN and Gradient Boosting Decision Trees with Weak and Noisy Supervision". In: CEUR-WS.org. URL: https://ceur-ws.org/Vol-2936/paper-136.pdf.

Niizumi, Daisuke et al. (2021). "BYOL for Audio: Self-Supervised Learning for General-Purpose Audio Representation". In: Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2103.06695. URL: https://arxiv.org/abs/2103.06695 (visited on 11/11/2022).

Norum, Ole-Magnus Vian (Dec. 2022). "Exploring Self-Supervised Learning for Bird Vocalization Classification". In.

O'Shea, Keiron and Ryan Nash (Dec. 2015). *An Introduction to Convolutional Neural Networks*. en. arXiv:1511.08458 [cs]. URL: http://arxiv.org/abs/1511.08458 (visited on 10/26/2022).

Okanoya, Kazuo (2008). "Avian Bioacoustics". en. In: *Handbook of Signal Processing in Acoustics*. Ed. by David Havelock, Sonoko Kuwano, and Michael VorlÃ¤nder. New York, NY: Springer New York, pp. 1887–1895. ISBN: 978-0-387-77698-9 978-0-387-30441-0. DOI: 10.1007/978-0-387-30441-0_103. URL: http://link.springer.com/10.1007/978-0-387-30441-0_103 (visited on 10/27/2022).

Oppenheim, A.V. (Sept. 1970). "Speech Spectrograms using the Fast Fourier Transform". In: *Spectrum, IEEE* 7, pp. 57–62. DOI: 10.1109/MSPEC.1970.5213512.

Park, Daniel S. et al. (2019). "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: Publisher: arXiv Version Number: 3. DOI: 10.48550/ARXIV.1904.08779. URL: https://arxiv.org/abs/1904.08779 (visited on 04/11/2023).

Ross, Sheldon M. (2010). "Distributions of Sampling Statistics". en. In: *Introductory Statistics*. Elsevier, pp. 297–330. ISBN: 978-0-12-374388-6. DOI: 10.1016/

B978-0-12-374388-6.00007-7. URL: https://linkinghub.elsevier.com/retrieve/pii/B9780123743886000077 (visited on 04/19/2023).

Selvaraju, Ramprasaath R. et al. (Feb. 2020). "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". en. In: *International Journal of Computer Vision* 128.2, pp. 336–359. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://link.springer.com/10.1007/s11263-019-01228-7 (visited on 06/05/2023).

Shannon, C.E. (Jan. 1949). "Communication in the Presence of Noise". In: *Proceedings of the IRE* 37.1, pp. 10–21. ISSN: 0096-8390. DOI: 10.1109/JRPROC.1949.232969. URL: http://ieeexplore.ieee.org/document/1697831/ (visited on 10/26/2022).

Shin, Sungho et al. (2021). "Self-Supervised Transfer Learning from Natural Images for Sound Classification". In: *Applied Sciences* 11.7. ISSN: 2076-3417. DOI: 10.3390/app11073043. URL: https://www.mdpi.com/2076-3417/11/7/3043.

Tian, Yonglong et al. (2020). "What Makes for Good Views for Contrastive Learning?" In: Publisher: arXiv Version Number: 3. DOI: 10.48550/ARXIV.2005.10243. URL: https://arxiv.org/abs/2005.10243 (visited on 05/22/2023).

Whelan, Christopher J., Daniel G. Wenny, and Robert J. Marquis (June 2008). "Ecosystem Services Provided by Birds". en. In: *Annals of the New York Academy of Sciences* 1134.1, pp. 25–60. ISSN: 00778923, 17496632. DOI: 10.1196/annals.1439.003. URL: http://doi.wiley.com/10.1196/annals.1439.003 (visited on 12/03/2022).

Wu, Jianxin (2017). "Introduction to convolutional neural networks". In: *National Key Lab for Novel Software Technology. Nanjing University. China* 5.23, p. 495.

xeno-canto (n.d.). URL: https://xeno-canto.org/about/xeno-canto. (visited on 11/10/2022).

Yakura, Hiromu, Kento Watanabe, and Masataka Goto (2022). "Self-Supervised Contrastive Learning for Singing Voices". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 30, pp. 1614–1623. DOI: 10.1109/TASLP.2022.3169627.

Yan, Na et al. (Dec. 2021). "Birdsong classification based on multi-feature fusion". en. In: *Multimedia Tools and Applications* 80.30, pp. 36529–36547. ISSN: 1380-7501, 1573-7721. DOI: 10.1007/s11042-021-11396-9. URL: https://link.springer.com/10.1007/s11042-021-11396-9 (visited on 10/26/2022).

Zhang, Yixiao et al. (2022). "Spectrogram Transformers for Audio Classification". In: *2022 IEEE International Conference on Imaging Systems and Techniques (IST)*, pp. 1–6. DOI: 10.1109/IST55454.2022.9827729.

Zhou, Hengshun, Xue Bai, and Jun Du (Nov. 2018). "An Investigation of Transfer Learning Mechanism for Acoustic Scene Classification". In: *2018 11th Interna-*

tional Symposium on Chinese Spoken Language Processing (ISCSLP). Taipei City, Taiwan: IEEE, pp. 404–408. ISBN: 978-1-5386-5627-3. DOI: 10.1109/ISCSLP.2018.8706712. URL: https://ieeexplore.ieee.org/document/8706712/ (visited on 04/27/2023).