Emanuele Caprioli

A semi-supervised approach to bird song classification

Master's thesis in Computer Science
Supervisor: Keith L. Downing
June 2022

Master's thesis

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

NTNU
Norwegian University of
Science and Technology

Emanuele Caprioli

A semi-supervised approach to bird song classification

Master's thesis in Computer Science
Supervisor: Keith L. Downing
June 2022

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

NTNU
Norwegian University of
Science and Technology

Abstract

Ecosystems have, in the last years, seen dramatic changes than at any other times in human history. Bioindicators like birds are monitored by ornithologists, to screen the environment's quality. Deep learning methods showed promising results in the field of polyphonic sound events and bird song recognition. Nevertheless, such methods require extensive datasets, requiring many hours of field research as well as substantial domain knowledge to collect and label bird sound recordings. With the advance of electronics and microphone technology, recording units capable of recording 24/7 can be deployed virtually anywhere, reducing the need of human presence on site, and easily building big unlabeled datasets.

In this thesis, a Semi-Supervised (SSL) approach to bird song classification based on FixMatch is presented. FixMatch is a novel SSL method developed with the image classification domain in mind, and with the purpose of exploiting unlabeled recordings in order to increase a classifier performance.

Results showed that FixMatch is indeed both applicable to the domain of bird song classification and compatible with Transfer Learning, using weights computed on an image dataset. The findings showed that raw-audio augmentation yielded no particular advantages against audio-sensitive image augmentation. The method managed to increase the classifier performance, also when presented with an unlabeled set composed by half unknown classes. The system built in this thesis showed potential, suggesting that future work like building a model capable of classifying more bird types and testing on real soundscape recordings, should be investigated.

Sammendrag

Økosystemer har sett mer dramatiske endringer enn noen annen gang I menneskets historie. Bioindikatorer som fugler overvåkes av ornitologer, for screening av miljøets kvalitet. Dyplæringsmetoder har produsert lovende resultater innen polyfoniske lyderhendelser og gjenkjenning av fuglesangfeltet. Likevel krever slike metoder omfattende datasett, som krever mange timer med feltforskning samt betydelig domenekunnskap for å samle inn og merke fuglelydopptak. Med fremskritt innen elektronikk og mikrofonteknologi kan opptaksenheter som er i stand til å ta opp 24/7 plasseres praktisk talt overalt, noe som reduserer behovet for menneskelig tilstedeværelse og kan enkelt bygge store umerkede datasett.

I denne oppgaven presenteres en Semi-Supervised (SSL) løsning til klassifisering av fuglesang basert på FixMatch. FixMatch er en ny SSL-metode utviklet for bildeklassifisering, og med omfanget av å utnytte umerkede opptak for å øke klassifiseringsytelsen.

Resultatene viste at FixMatch er både anvendelig for fuglesangsklassifiseringsdomenet og kompatibel med Transfer Learning, ved å bruke vekter beregnet på et bildedatasett. Funnene viste at rålyd augmentering ikke ga noen spesielle fordeler mot lyd-tenkte bilde augmentering. Metoden klarte å øke klassifiseringsytelsen, også når den ble presentert med et umerkede datasett der halvparten av lydsporene stammer fra ukjente klasser. Systemet som ble laget i denne oppgaven viste potensiale, og antydet at fremtidig arbeid som å bygge en modell som er i stand til å klassifisere flere fugletyper og teste på ekte lydlandskapsopptak, bør undersøkes.

Preface

This work is the culmination of the Master's program TDT4900 at the Norwegian University of Science and Technology (NTNU), Department of Computer Science (IDI).

This thesis was supervised by professor Keith L. Downing, whom I would like to thank for all the invaluable guidance and feedback I received during the last year. I would like to thank the Xeno-Canto community for the retrieval of the high-quality bird recordings, and the people at Cornell Lab of Ornithology for compiling the dataset that makes this thesis possible. Lastly, I would like to thank my friends and family for all the support I have received, with a special thanks to my twin sister Licia.

Contents

Figures

Tables

Chapter 1

Introduction

This chapter introduces the research area and its relative difficulties, along with the steps this thesis is taking towards solving it. The problem background and some of the related work is presented in section 1.1, the research questions and goal for this thesis are formulated in section 1.2, the research method is introduced in section 1.3, and, finally, the thesis outline is listed in section 1.4.

1.1 Motivation

In modern times, both managed and natural ecosystems have seen dramatic changes more than in any other times in human history, leading to habitat destruction, soil erosion, pollution, climate change and species extinction [1]. Monitoring of all biotic and abiotic factors leads to a better understanding of the situation of the given environment. Nevertheless, due to the complexity of such task, bioindicators are instead monitored for screening of the environment's quality. The term bioindicator refers to living organisms such as animals, plants and microbes. Their high position in the food chain [2], capability of reflecting changes in other animals and plants, and high sensitivity to environmental contaminants make birds a valid bioindicator [3].

Domain experts traditionally mark the presence of birds in an environment either by asserting their visual traits or by listening to their sounds. Some limitation comes with this method, for instance the inability for humans to reach or operate in some environments with relative comfort and the requirement of a domain expert on the scene. With electronics becoming more affordable, increasing cellular coverage and the advancing of microphone technology, the positioning of recording units capable of 24/7 recording seems the most sensible way of monitoring birds [4]. Such units can have either local storage with months-worth capacity of audio recordings, or even transmitting high quality streams in real time, reducing the time a domain expert needs to employ on the field in order to classify bird species.

In the recent years, deep learning methods like CNN (e.g. [5] [6]), CRNN (e.g [7] [8]), ResNet (e.g. [4]), CapsNet (e.g. [9]) and pretrained architectures (e.g [10]) showed potential in classifying polyphonic sound events,i.e., sounds that can overlap. Deep learning methods require substantial quantity of labeled data to train in a Supervised Learning (SL) fashion, as data is not always available, or expensive to acquire. Alternatives lie within utilization of unlabeled data to improve the accuracy of the models. Such methods, called Semi Supervised Learning (SSL), yielded comparable state-of-the-art performance within image domain (e.g., [11] [12]) and audio domain (e.g., [13] [14][15]).

Xeno-Canto [16] is a community driven platform where ornithologists and bird watchers can share recordings of bird sounds. The data collection, containing over ten thousands different species, is weakly labeled, i.e., each recording is tagged with only the foreground and some of the recognized background species, lacking information such as on-offset, i.e., start and stop of the bird song in the recording. This limitation amplifies the need to explore alternative methods that rely on weakly labeled data, with motivation in both improving classifiers training on this specific dataset and developing classifiers for other audio domains, where sufficient labeled examples are resource-heavy to acquire, as opposed to unlabeled examples.

1.2 Goals and Research Questions

The project goal and review of the research questions are presented as it follows:

Goal *Reducing the labeled data needed to reach baseline performance on classification of bird sounds.*

The goal of this project is to test SSL methods on training a classifier that achieves the same performance of the baseline methods, while utilizing less labeled data.

Research question 1 *How do SSL methods applied in other domains perform on bird songs in comparison to a sensible baseline?*

Semi Supervised Learning methods have been extensively applied on image classification tasks. While some work in the sound classification task has been conducted, few and outdated works benchmark purely on bird songs, where similarities between calls and intraspecies dialectal differences can have an impact on learning from unlabeled features.

Research question 2 *To which extent can the labeled-to-unlabeled examples ratio be reduced before degradation of the system?*

Semi-supervised learning methods utilize both labeled and unlabeled examples during training, where the ratio of labeled-to-unlabeled can be tweaked and adapted to the dataset available. In the case where the unlabeled set includes external sounds and noises, reducing the aforementioned ratio could result to a worse classifier performance compared to only using labeled examples.

1.3 Research Method

The project will focus on FixMatch, a rather novel algorithm for training models in a semi-supervised fashion, and its applicability on the bird song domain. The performance of such model will be benchmarked against a baseline trained only on the labeled subset and fine-tuning of a pretrained model, due to the similarity of application between transfer learning and semi-supervised learning.

1.4 Thesis Outline

The rest of this thesis is organized as follows: theoretical knowledge required for a better understanding of the concepts discussed in this thesis are presented in chapter 2, related research regarding the problem area of bird sound classification and semi-supervised learning approaches are briefly listed in chapter 3, the system pipeline is described in chapter 4, while the experiment plan, results and analysis are presented and discussed in chapter 5. Finally, this thesis' conclusion, final remarks and suggestions for future work is available in chapter 6.

Chapter 2

Background Theory

This chapter presents the background knowledge this thesis is based on. Audio signal theory is introduced in section 2.1, and an insight into the bird song domain is presented in section 2.2. Deep learning architectures are explained in section 2.3, while Semi-supervised learning methods are discussed in section 2.4. Finally, the evaluation metrics used in this thesis are presented in section 2.5.

2.1 Sound Representation

2.1.1 Signals and Sampling

Sound is the repetition of a signal during time. In Figure 2.1, a sinus wave is plotted over time. Here, the "Period" represents the time needed for the signal to repeat itself, while the "Amplitude" shows the signal's intensity from the reference point. The "Frequency" of the signal is 1 over period and it represents the number of times the signal repeats itself in the span of one second.

A continuous signal consists of infinite data points and is impossible to represent on a digital medium without any kind of manipulation. "Sampling" is a technique where the signal is divided in evenly spaced samples and one data point per sample is saved for the digital representation of the signal. The number of samples per one second is defined as "Sampling Rate", while the length of a sample is defined as "Sampling Period" [17]. Figure 2.2 exemplifies a sampling, where the same signal is sampled with a lower sample rate on the left and a higher rate on the right.

Good sound representations are required in order to construct a good machine learning system.A good representation is simultaneously small in size and rich in information. To put things in perspective, a five-second-long clip sampled at 44.1kHz produces 220500 data points, a representation too big for the majority of applications.

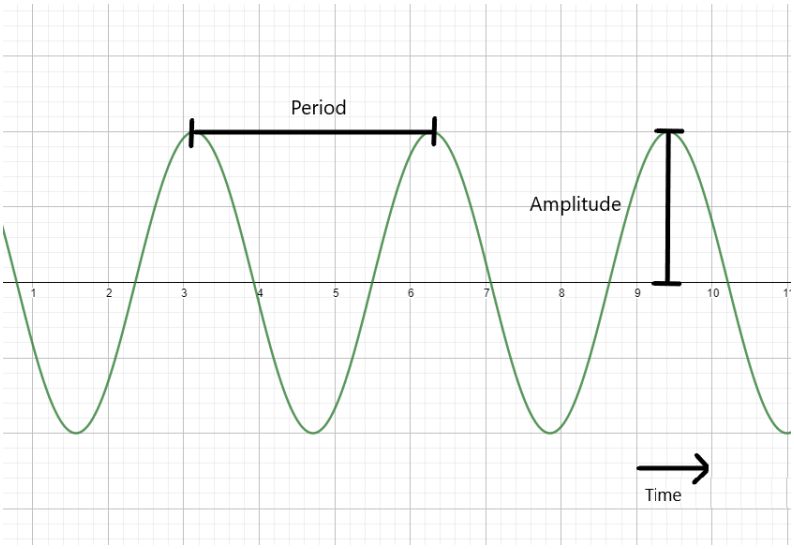


Figure 2.1: A sinus signal plotted over time, with its period and amplitude marked out.

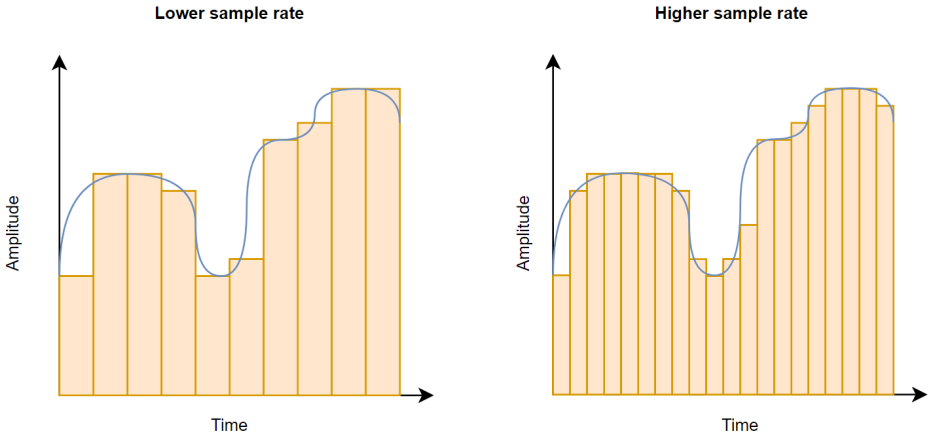


Figure 2.2: Digital sampling of the same audio signal. In the graph on the left, the sampling rate is lower than in the right. It can be noticed that more of the signal features are retained when a higher sample rate is used.

2.1.2 Frequency Domain

The majority of audio signals are not periodic signals. Fourier Theorem states that every non-periodic signal is composed of sums of multiple periodic signals, each with their own frequency and intensity. Fourier Transform is a mathematical function that decomposes a signal into its distinct components, resulting in the "Spectrum", namely a plot of all of the frequencies of a signal and their intensities within a given time frame. The Fourier Transform maps a signal from the time domain to the frequency domain. An example of such mechanism can be seen in Figure 2.3, where the original non-periodic signal is plotted in red. Fourier Transform then proceeds by separating each of the composing periodic signals, plotted in different shades of purple, producing the frequency spectrum, plotted in blue.

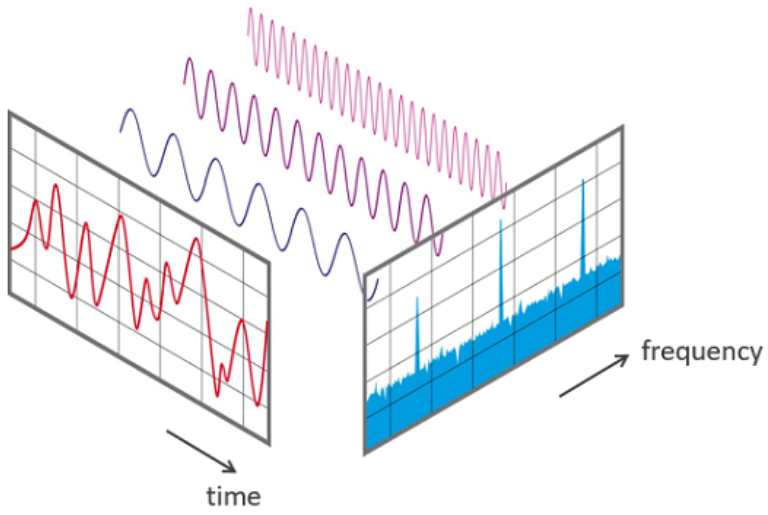


Figure 2.3: Fourier Transform maps the signal from the time domain to the frequency domain. Illustration by Phonical [18], shared under the CC BY-SA 4.0 license.

2.1.3 Short Time Fourier Transform

In the case of non-periodic signals, the frequencies that compose said signal change over time, making the spectrum of the entire audio signal a poor representation. Short Time Fourier Transform (STFT) is an algorithm that divides a signal into overlapping windows of time and computes the Fast Fourier Transform of each window, resulting in a "spectrogram" [19]. Fast Fourier Transform efficiently implements the Discrete Fourier Transform, a mathematical function that applies the Fourier Transform on discrete signal rather than continuous, such as the result of sampling of signal, discussed in subsection 2.1.1. Figure 2.4a shows a 7.5 seconds-long signal corresponding to the call of a Blue Jay, sampled at 22.050kHz.

Applying the STFT to the signal, with a window size of 2048 and 25% overlap, the 7.5 seconds long signal produces a resulting vector of $\frac{7.5s*22.050kHz}{2048*0.25} = 323$ columns, as seen in Figure 2.4b. Each column of the spectrogram includes the magnitude - also referred to as energy levels - of all the frequencies present during that specific window of time.

2.1.4 Mel Spectrograms

The Mel Spectrogram is a spectrogram of a signal where the frequencies are mapped to the "mel scale". The mel scale is a scale of pitches and perceived frequencies where humans asses them to be equally distant [20]. The Mel scale, illustrated in Figure 2.5, has a quasi-logarithmic nature compared to a linear Hertz scale, due to the fact that humans perceive the change between 100Hz and 200Hz to be of higher significance over, for instance, between 1000Hz and 1100Hz or between 10000Hz and 10100Hz, despite them being of the same differential quantity. The reference point between the Mel and Hertz scale is set by equating a pitch of 1000 Mels to 1000Hz. Mels can be computed by converting Hz frequencies using the formula found in Equation 2.1.

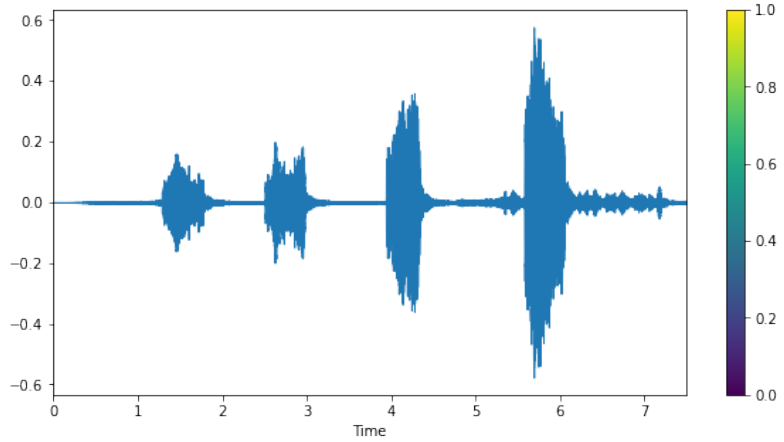
$$m = 2595 log_{10}(1 + \frac{f}{700}) \tag{2.1}$$

Energy levels in the mel spectrogram are mapped to decibels (dB), a logarithmic scale expressing relative change between two signals, where each 3dB increase represents a doubling of the intensity.

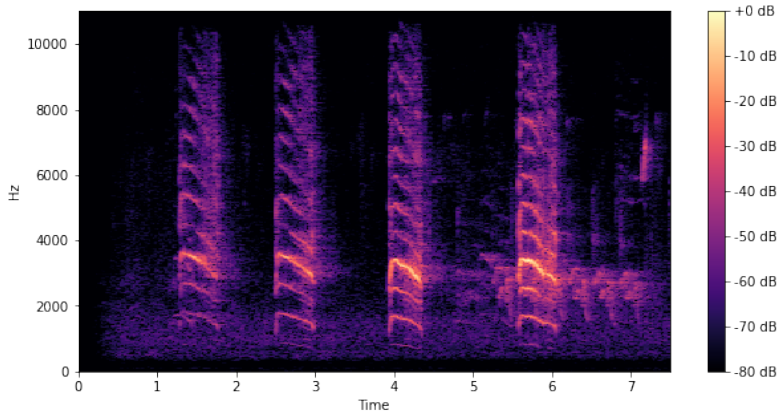
2.2 Bird Songs Variations

Birds produce a large variation of sounds, each for a different occasion. Some of those variations are: calls, mating calls, songs and duets. At the same time, more variations of such sounds exists, like between male and female exemplars of the same species, during different life stages, and dialects, making the bird song domain a non trivial one.

An example of dialectal differences can be observed in the blujay call spectrograms showed in Figure 2.4b and Figure 2.6a, where the former is computed on a recording retrieved just outside Montréal (Canada), while the latter was retrieved in Florida (USA), a distance of around 2500Km between the two locations. The two spectrograms differ greatly, with almost no similarities, even though they represent the same species call. On the other hand, the spectrogram of the brown-crested flycatche call from the Honduras, shown in Figure 2.6b, shares some similarities with the blujay call from Florida. Similarly to the blujay example, the brown-crested flycatche call from Arizona, shown in Figure 2.6c, differs also from the counterpart from the Honduras, and shows resemblance to the Canadian blujay call.



(a) Signal when magnitude plotted against time. Sampling rate of 22.050kHz.



(b) Short Time Fourier Transform computed with a window length of 2048 samples (about 93ms of length) and energy levels normalized against the dB scale.

Figure 2.4: "Blue Jay call from Canada" signal (a) and it's spectrogram representation (b). Clip from the Xeno-Canto collection.

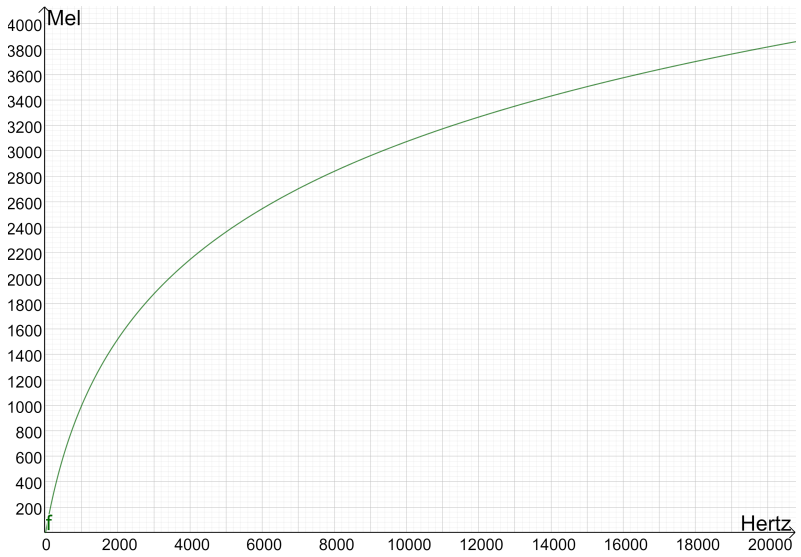


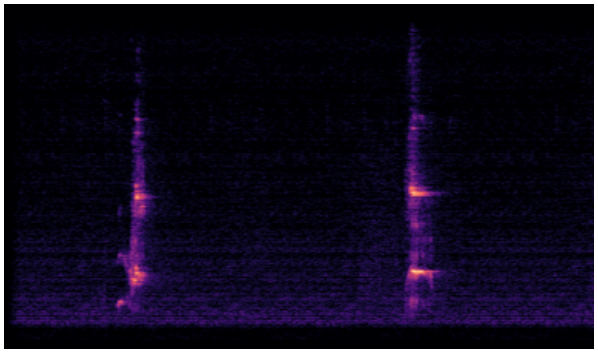
Figure 2.5: Mel scale plotted against the linear Hertz scale.

2.3 Deep Learning Architectures

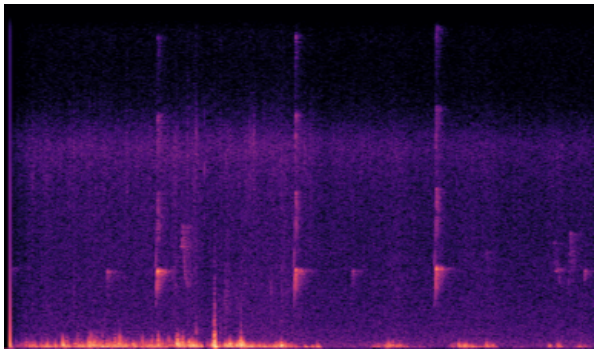
2.3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) is a subset of machine learning designed for processing grid like data, by learning spacial invariant features from lower to higher level patterns [21]. Convolutional, Pooling and Fully Connected (FC) layers are the building blocks of the CNN architecture: the first two are repeatedly stacked, followed by one or multiple FC layers, as seen in Figure 2.7. Their design is inspired by the animal visual cortex, where the first layers learn low level features like lines and edges, while the subsequent layers detect progressively more complex features such as eyes, nose and faces. In CNNs, convolutional and pooling layers are responsible for feature extraction, while the FC layers are utilized as a classifier head. Training occurs with the "Backpropagation" algorithm, where the network updates its learnable parameters in order to minimize the loss, computed via a "Loss fuction" from the network output and the ground truth.

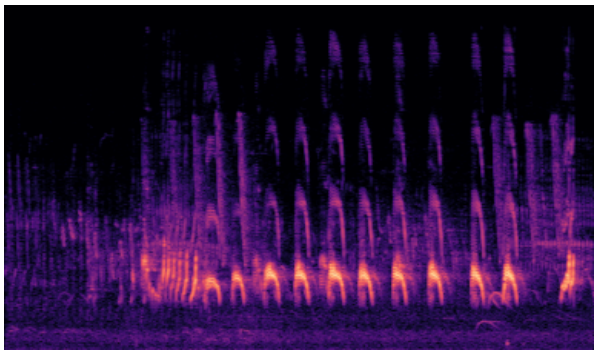
The convolutional Layer performs a linear convolutional operation by applying a small array of numbers called "kernel" on its input tensor. The Input tensor has a dimension of C x W x H, where C is the number of channels. In an image classification task, the input channel C represents the number of color channels, while W and H are the pixel width and height of the image, respectively. Each convolutional layer has K "Kernel stacks" of a predefined shape of I x N x M, where I is the number of channels of the input (for the first layer $I = C$ applies), while N and M are the size of the convolutional filter, as shown in Figure 2.8. Each kernel stack is then translated over the input tensor, computing an element-wise product of each element of the input tensor and the kernel, summing it up along the I



(a) Blujay "call" from Florida



(b) Brown-crested Flycatcher "call" from the Honduras



(c) Brown-crested Flycatcher "call" from Arizona.

Figure 2.6: Dialects differences showed in the spectrograms computed on the blujay and brown-crested flycatcher calls. A strong resemblance between the blujay call from Florida and the brown-crested flycatcher from the Honduras can be observed here. On the other hand, those similarities do not apply when inspecting the intraspecies calls retrieved from samples found in different locations, like between the brown-crested flycatcher calls from the Honduras and Arizona, and between the blujay calls from Florida and Canada, the latter blujay call shown in Figure 2.4b.

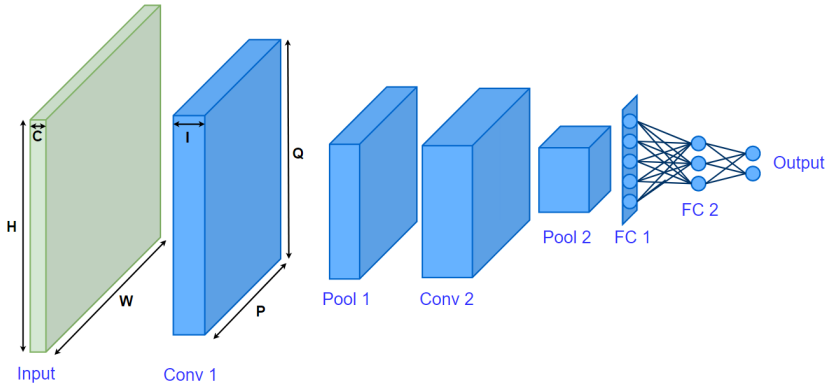


Figure 2.7: The CNN architecture is achieved by stacking multiple convolutional and pooling layers, to then finish with one or muliple fully connected (FC) layers.

dimension, to then produce K outputs called feature map. The output of a convolutional layer is then of shape K x P x Q, where P and Q are dependent on the size of the kernel, the hopping length "stride" and the input padding technique used. The equations for calculating the output P and Q are shown in Equation 2.2 and Equation 2.3.

$$P = \frac{W - N + 2 * padding}{stride} + 1 \tag{2.2}$$

$$Q = \frac{H - M + 2 * padding}{stride} + 1 \tag{2.3}$$

Each kernel can be thought of as a feature extractor.

The output of the convolutional operation is then passed through a non-linear activation function that maps the output values to another scale. Example of non-linear functions are Sigmoid, where outputs get mapped between 0 and + 1, Tahn, between -1 and +1, and ReLU, where negative values are set to zero while the positive ones are left unchanged.

Pooling layers are responsible for selecting the extracted features from the previous layer, while simultaneously reducing the dimensionality of the input and therefore the number of learnable parameters in the subsequent layers [21]. Pooling layers have the capability of selecting "translational invariant features", features that are resilient to small shifts and distortions. Analogously to convolutional layers, the output of a pooling layer is dependent on the stride, the filter size and the padding technique, and therefore the calculations of the output shape are the same as in Equation 2.2 and Equation 2.3.

2.3.2 Deep Residual Networks

Deep Residual Network (ResNet) [22] is a CNN based architecture, born from the assumption that very deep networks should perform at least as good as their

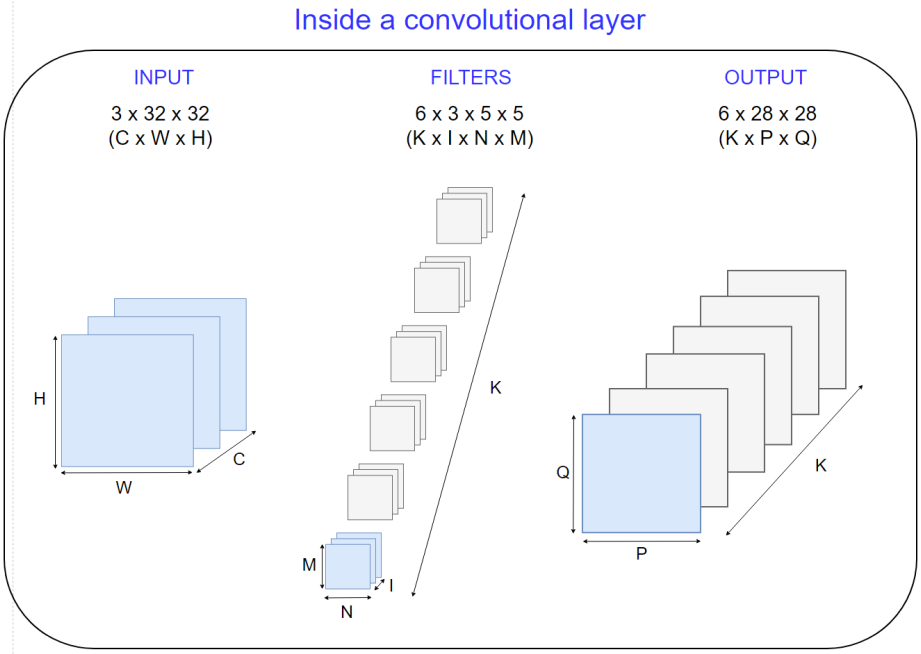


Figure 2.8: Inside the first convolutional layer of the network showed in Figure 2.7. The convolutional layer has a predefined number of kernels K that, applied to the input tensor, produce K output feature maps. In this example, six kernels of shape 3x5x5 (same amount of channels as the input, I = C) are applied to the input of shape 3x32x32 to produce an output tensor of shape 6x28x28, when a stride of 1 is used and no padding is being applied. Marked in blue is the component used to produce one feature map.

shallow counterpart. The introduction of residual blocks, depicted in Figure 2.9, allows the network to skip connection within the block and acting as an identity layer when needed, by so avoiding degradation of the network performance. With this design, very deep networks can be conceptualised that are not subject to the vanishing gradient problem, namely a learning freeze in the earlier layers of the network due to gradients updates becoming smaller during backpropagation. The original ResNet architecture designed for solving the ImageNet dataset incorporated a total of 152 layers, while newer and popular variants are less deep, like ResNet50 and ResNet18 which incorporate 50 and 18 layers, respectively.

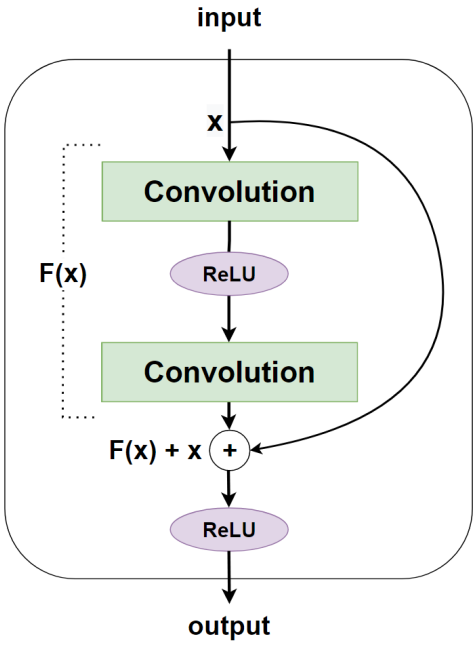


Figure 2.9: Illustration of a Residual block. The input x is passed trough a function f() composed by two convolutional layers with a ReLU activation in between. The input x is then added directly to the function output F(x) and the result activated.

2.3.3 Transfer Learning

Traditionally, a model is trained from scratch on a specific task, but in a scenario where not enough labeled training examples nor enough resource exist, good performance can be challenging to achieve. Transfer learning (TF) is the concept of transferring the knowledge acquired from a task in a domain to another task in another domain [23]. In neural networks-based architecture like CNN, transfer learning can be applied by utilizing a state-of-the-art architecture design like VGG-16 or ResNet, "pretrained" on a vast and balanced dataset like ImageNet. As seen in subsection 2.3.1, CNNs are build ups of convolutional and FC layers with each its trainable parameters called weights. An implementation of TL is freezing

all the convolutional layers of the pretrained model, stopping weights update at these levels, and design a new classifier head of FC layers on top of it that matches the desired application domain, discarding the old one. The transferred layers are in this case utilized as the feature extractor, while only the new classifier head is permitted to update weights.

CNNs learn to extract simpler features in the earlier layers. Another application of transfer learning is "fine-tuning", where only the weights from the earlier layers of the pretrained model are transferred and frozen, leaving other feature extraction and FC layers able to update the weights. This is particularly useful the more the domains differs, for instance when applying knowledge from a pretrained model on an image domain to a sound classification task: features learned in the earlier layers, like lines, curves, angles and points can be useful, while higher level and more complex features like eyes, faces and cars are irrelevant.

2.3.4 Data Augmentation

Data augmentation is a technique used in order to increase the number of examples in a small dataset, without the need of creating new synthetic data. It does so by applying a single transformation, or a combination thereof, to the original data, producing more or less altered versions that the model can utilize during training, making it more capable of generalizing. Figure 2.10 shows an example of data augmentation, where a figure of a dog has been flipped and cropped, resulting in three times the amount of training data. Even after the application of the transformation, the label "dog" has not been changed. Other noteworthy image augmentation techniques apply rotation, translation, flipping, and change of contrast, or brightness, to the input. The only constraint to data augmentation is that the transformed output must retain enough information for the model to make a sensible prediction.

(a) Original **(b)** Flipped **(c)** Cropped

Figure 2.10: A sample picture for a class "dog". Transformations have been applied to the original sample to produce two new samples, by flipping and cropping.

2.4 Semi-Supervised Learning

Deep learning architectures learn by backpropagating the gradients update obtained by minimizing the loss computed through a loss function, the network output and the ground truth. In "Supervised Learning (SL)", the ground truth is available for the training set in its entirety. In "Unsupervised Learning (UL)", algorithms discover patterns and similarities without the need of labeled examples. Situated between SL and UL, "Semi-Supervised Learning (SSL)" utilizes a small amount of labeled examples and a larger amount of unlabeled examples during training [24]. SSL methods are often applied to domains where labeled examples are costly to acquire, while, at the same time, an abundance of unlabeled examples exist. SSL is based on the assumption that points that are close to each other are likely to share the same label. The top half of Figure 2.11 shows the separation function found when only two labeled example (one for each class) is utilized. Despite it being a valid separation, when unlabeled examples are introduced, a more correct separation evolves out of the assumption that the points close to each other share the same label, better separating the two clusters.

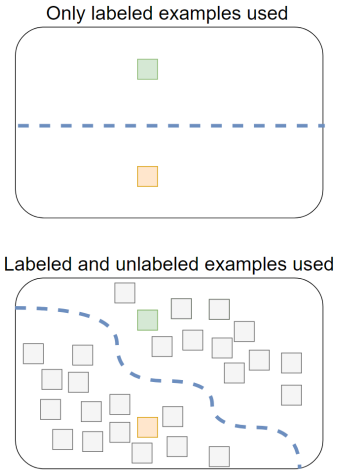


Figure 2.11: Learned separation function when only labeled examples are used versus introducing unlabeled data in a semi-supervised fashion.

2.4.1 Pseudo-Labeling

Pseudo-Labeling [25] is a semi-supervised technique where unlabeled data is utilized during training in a supervised fashion. Firstly, the model is trained with SL on a labeled subset. Secondly, for each weight update, pseudo-labels for the unlabeled subset are generated by selecting the class with the highest value out of the probability produced by the same model: see Figure 4.8 for a graphic visualization of this mechanism. The pseudo-labels are then incorporated in the training set if above a predefined threshold, meaning that the model is confident enough

about those predictions. Lastly, the loss is calculated by the loss function in Equation 2.4, where the first part is the averaged labeled mini-batch loss and the second part is the averaged unlabeled mini-batch loss adjusted by a coefficient α. Such coefficient is adjusted to leverage the importance of the unlabeled data. A good strategy is to gradually increase the value with the training epoch, due to the fact that pseudo-labels should be more accurate as the model trains.

$$L = \frac{1}{n}\sum_{m=1}^{n}\sum_{i=1}^{C} L(y_i^m, f_i^m) + \alpha(t)\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{C} L(y_i'^m, f_i'^m) \tag{2.4}$$

Pseudo-labeling struggles in situations where datasets are of higher dimensionality and clusters are more complex than the one showed in Figure 2.11. This is due to a a phenomenon called "Confirmation Bias", namely overfitting on incorrect pseudo-labels predicted by the model [26]. In such cases, the network wrongly labels unlabeled examples and incorporates such examples into the learning process, leading to incorrect boundaries being learned, seen in Figure 2.12.

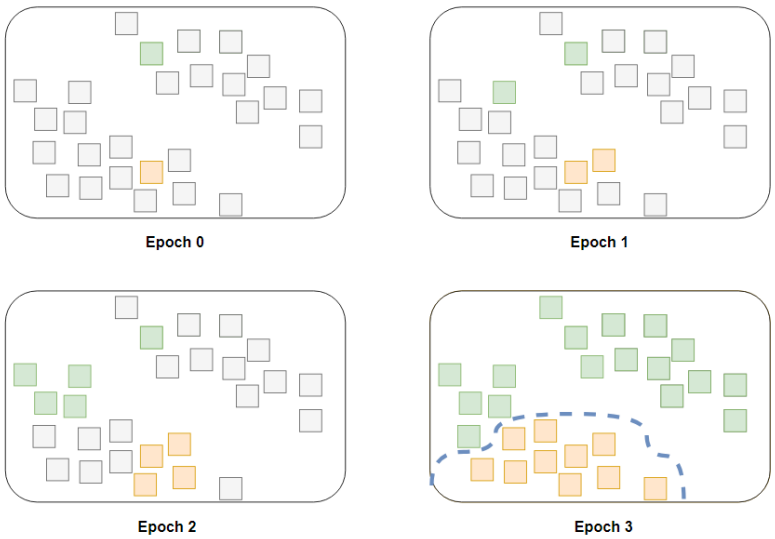


Figure 2.12: A confirmation error example. In the training epoch 0, the model is presented with one labeled example from each class, and many unlabeled examples. In epoch 1, the model mislabels one of the unlabeled examples, assigning the class green. The error is propagated in epoch 2, where the green label is wrongly assigned to the neighbouring unlabeled examples, leading to learning a more incorrect separation function than the one found in Figure 2.11.

2.4.2 Consistency Regularization

Consistency Regularization is a SSL method that encourages the network to produce consistent output probabilities between an unlabeled example and a perturbed version of it [27]. The perturbed example is generated by applying some

kind of transformation to the original example. The model is trained by trying to minimize the distance between predictions of augmented versions of the same unlabeled example, using the loss function shown in Equation 2.5, where $\alpha()$ represents a stochastic augmentation function.

$$\sum_{b=1}^{\mu B} \| p_m(y|\alpha(u_b)) - p_m(y|\alpha(u_b)) \|_2^2 \tag{2.5}$$

2.5 Evaluation Metrics

When training a machine learning model, a metric for knowing how well that model is performing is required.

2.5.1 Binary Classification

In binary classification, each instance to be predicted is member of either the "positive" or "negative" class, two arbitrary and non-overlapping classes. Predictions and the true labels can be plotted together in a confusion matrix, as in Figure 2.13, where:

True Positive The classifier correctly predicts the positive class

False Positive The classifier incorrectly predicts the positive class (also called Type 1 error)

False Negative The classifier incorrectly predicts the negative class (also called Type 2 error)

True Negative The classifier correctly predicts the negative class

Precision and recall are metrics for evaluating a classifier. Formulas for calculating the aforementioned metrics are the following:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

Precision is the ratio of correctly identified instances of the positive class over all the positive predictions. Recall, also referred to as sensitivity, tests the classifier ability of identifying the true positive.

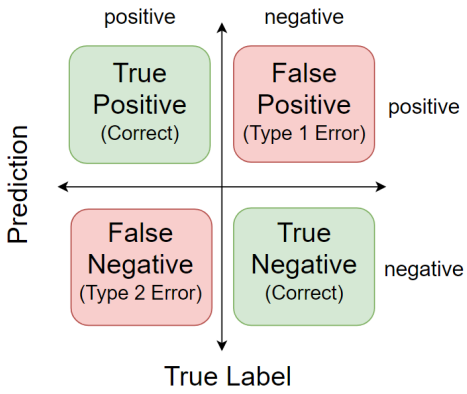


Figure 2.13: Confusion matrix representing the relation between predictions and their true label. The case where the classifier incorrectly predicts a positive class, a False Positive conclusion has been committed, also known as Type 1 Error in statistics. When the classifier fails to identify the positive class, a False Negative is produced, analogous to Type 2 Error in statistics.

2.5.2 F1-Score

It is not possible to maximize both precision and recall, as increasing the one will often decrease the other. However, precision and recall can be combined into a single number, called F1-score, that can be maximized to increase the classifier performance. The formula for the F1-score is the following:

$$F1score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

2.5.3 Multi-Class Classification

When dealing with multi-class classification problems, techniques called "micro averaging" and "macro averaging" are introduced. In micro averaging, the precision is the ratio of all the true positive against all of the true positive and false positive instances. Analogously, the micro-recall is the ratio of all the true positives against all the true positives and false negatives instances. Micro-precision and micro-recall equations are thus the following:

$$microPrecision = \frac{\sum_c^C TruePositive(c)}{\sum_c^C TruePositive(c) + FalsePositive(c)}$$

$$microRecall = \frac{\sum_c^C TruePositive(c)}{\sum_c^C TruePositive(c) + FalseNegative(c)}$$

where c represents one of the C classes the classifier is able to identify.

The downside of micro-F1-score is the inability of dealing with imbalance datasets, as it combines all the instances without any class considerations, hiding

the model performance on the minority classes. On the other side, in macro averaging", the precision and recall are calculated for each class (the positive class) against all the other classes combined (representing the negative class). Macro-Precision and macro-recall are then the non weighted averaged of each class respective precision and recall, as shown in the equations:

$$macroPrecision = \frac{\sum_c^C Precision(c)}{C}$$

$$macroRecall = \frac{\sum_c^C Recall(c)}{C}$$

The macro/micro F1-score is thus the combination of macro/micro precision and recall, as described in subsection 2.5.2.

2.6 Chapter Summary

This chapter presented how relevant features are extracted from audio signals, thus producing spectrogram images. Some examples of intra-/interspecies differences and similarities have been presented and discussed. Then, an insight into the works of CNN and ResNet blocks, popular modern architectures used in image and audio classification, has been provided. Given the data-hungry nature of deep learning methods, relevant strategies for increasing the amount of data and taking advantage of existing datasets have been presented: data augmentation and transfer learning, respectively. Additionally, this chapter discussed basic semi-supervised learning concepts and approaches such as pseudo-labeling and consistency regularization, and concluded with the theory behind evaluation metrics for machine learning models.

Chapter 3

Related Work

Due to famous open machine learning competitions having been available each year since 2014, like LifeCLEF Bird task, later BirdCLEF [28] and, since 2016, DCASE task 3 [29], many methods have been tested and results published, tackling the monophonic and polyphonic sound event detection problem (SED). Support-vector machines, Bayesian classifiers and random forest-based solutions were submitted with discrete results. In the recent years, a number of submissions based on deep learning methods, like Convolutional Neural Networks, Recurrent Neural Networks (RNN), Convolutional Recurrent Neural Networks (CRNN) and ResNet [4] increased, ultimately dominating the solutions landscape for this kind of problem.

Each year, new submissions are heavily based on the previous years' winner approach, often a deep learning method. Because of this, the need of big datasets is crucial, albeit labour and domain-knowledge expensive to acquire.

In this chapter, some of the deep learning aproaches to audio classification are presented in section 3.1, while semi-supervised and unsupervised learning research in this domain discussed in section 3.2. Finally, the structured literature review protocol is presented in section 3.3.

3.1 Deep learning Approaches in Audio Domain

Convolutional Neural Network (CNN) was the first deep learning method to be applied to the LifeCLEF Bird task, which later became BirdCLEF, laying stepping stones for the majority of the succeeding attempts. Sprengel et al. [5] won the LifeCLEF 2016 Bird task with a rather shallow five-layers CNN and multiple data augmentation techniques. The authors split each training audio file into a pure signal part and a noise part, computed the Short Time Fourier Transform (STFT) of each part and divided the resulting spectrograms into chunks of three seconds in length. Different data augmentation techniques were implemented, such as time and pitch shift, shifting the spectrograms on the X and Y axes, respectively, and with the purpose of teaching the network irregular patterns and that the occurrence of the song could be at anytime within the frame. Combination of the

same-class bird sounds was introduced in order to simulate multiple instances of the same species vocalizing at the same time, while adding noise arbitrary chosen from previously separated training files reported good results and reduced the generalization error. The trained network by Sprengel et al. [5] performed well on classification of the foreground species, winning first price. When classifying the background species, other approaches performed better due to less sounding species being classified as noise earlier in the preprocessing pipeline. The model performed poorly during the landscape test, since it was only trained on examples of simultaneous vocalization of multiple same-species birds, rather than of different species [30].

A noteworthy point is that, in the approach by Sprengel et al. [5], each spectrogram was fed to the network singularly, as in an image classification task. The temporal context of the audio sequence is then discarded with this type of learning. Cakir et al. [7] combined the CNNs capabilities of extract translational invariant features with the Recurrent Neural Networks (RNN) temporal features recognition. STFT spectral representations were produced from the raw sounds, but the calculated energies were converted on the MEL scale. When training, each spectrogram is divided into overlapping sequences, giving the model the possibility to sample sound like in an NLP problem. Cakir et al.[7] composed a hybrid Convolutional Recurrent Neural Network (CRNN) by chaining the output of the CNN to the input of the RNN and subsequently to a fully connected layer (FC). In the CNN, Max Pooling was allowed to downsample only on the frequency domain, in order to feed the extracted features on the original time representation to the RNN. The CRNN was compared to plain CNN, RNN, feedforward neural network (FNN) and Gaussian mixture model (GMM). All models were tested on four SED datasets (TUT Sound Events Synthetic 2016, TUT-SED 2009, TUT-SED 2016 and CHiME-Home), showing improvements of CRNN over all the other tested methods. Performance of all tested models was noticeably worst on one of the datasets (TUT-SED 2016), due to the considerably smaller size. Transfer learning and Semi supervised learning have been suggested by the authors as future improvements regarding this problem, although no image augmentation methods were used on any of the datasets.

Boddapati et al. [6] successfully applied GoogLeNet and AlexNet CNNs, designed with image classification in mind, to the polyphonic SED classification problem. Testings were conducted over three datasets (ESC-50, ESC-10, and UrbanSound8K), with different sampling rates and feature representation fed to the networks, such as Spectrograms, Mel-Frequency Cepstral Coefficients (MFCC), Cross Recurrence Plot (CRP) and a mixture of the three, a different feature representation for each input color channel. GoogleLeNet performed the highest when pure STFT spectrograms were fed to the network. Boddapati et al. [6] theorizes that the results could be due to GoogleLeNet being deeper than AlexNet, the two being 22 and eight layers long, respectively. Poorer accuracy was yielded when feeding

the network with the mixed feature representation, rather than each feature representation alone. The pre-designed networks were then compared with a CRNN originally intended for an input of the size 96 x 1366, adapted to accept a 256 x 256 sized input representation. The results were not impressive, with GoogleLeNet and AlexNet performing better. This is probably due to feeding the network a single image per sound event, instead of a series of images. Due to the limited size of the two ESC datasets, "Time stretching" was implemented as an augmentation technique, producing six samples with different degrees of stretching for each training frame.

The task of SED is a non-trivial one, especially within the bird song domain, where hundreds of species exist and dialects and different songs are developed both intraspecies-wise and between males and females [31]. All those nuances can be learned in longer networks with a superior weight count, but longer networks suffer the vanishing gradient problem. Kahl et al. [4] implemented a ResNet with 157 layers to solve the task of monophonic and polyphonic SED of bird songs. The dataset used for training was a set of 984 classes with the highest number of samples obtained from Xeno-canto and Macaulay Library of Natural Sounds. This approach was heavy on the preprocessing and augmentation side, due to previous results leading to acknowledge that incorporating ornithology domain knowledge at those stages was crucial for achieving best results. For each sample, the Fast Fourier Transform (FFT) was computed and the frequency range restricted to the bird vocalizing range, with some wiggle room for image augmentation (150 Hz - 15 kHZ). Due to the weakly-labeled nature of the dataset, only the focal part of the recordings were used in the training, discarding the background species. Kahl et al. implemented three different data augmentation techniques, namely: "Vertical Roll", shifting the spectrogram along the Y (frequency) axis, "Warping", due to the popularity achieved within speech recognition application, and adding arbitrary noise from other examples, as seen in the work of Sprengel et al. [5]. The network has been tested on focal recording of high quality (high sound-to-noise ratio SNR) and landscape recordings with both higher and lower SNR. Results demonstrate that short FFT windows frames performed better when it comes to bird sounds, Mixup training [32] improved training on all scenarios, while deeper networks did not perform better than wider networks, albeit still outperforming their shallow counterparts. Due to the imbalanced nature of the datasets, oversampling improved the overall scores whereas cost-sensitive learning did not.

Vesperini et al. [9] and Iqbal et al. [33] evaluated Capsule Neural Network [34] (CapsNet) for the audio classification task. The motivation behind this was to exploit the architectural advantage CapsNet held over CNN, namely the ability of identifying both invariant and equivariant features without any loss of information as a consequence to Max Pooling operations. CapsNet therefore performs better than traditional CNNs on less data and without the need of image augmentation, making it a reasonable approach for acoustic domain where labeled

data is expensive and few augmentation methods can be utilized due to domain constrains. Vesperini et al. compared the performance of CapsNet with those of CNN, previous DCASE challenge baseline model, and the state-of-the-art CRNN. No image augmentation was implemented on the three evaluated datasets (TUT Sound Events 2016, TUT Sound Events 2017, TUT Rare Sound Events 2017), and preprocessing consisted of generating STFT and logMel STFT spectrograms, with regular STFT performing better on CapsNet. The vanilla version of routing by agreement was implemented, with the sole change of initialization of the coefficient B_{ij} to be equal to the calculated value for the previous frame, retaining temporal information. Results showed that CNN performed better on monophonic SED, while CapsNet outperformed previous methods on the polyphonic SED with overlapping sounds, being consistent with the results of CapsNet on the MultiMNIST dataset found in the work of Sabour et al. [34]. Additionally, CapsNet performed better on the TUT Sound Event 2016 due to the considerably smaller size of this dataset, as seen in the work of Cakir et al. [7], consolidating the fact that a lot of labeled data is needed in order to achieve decent results when training in a Supervised Learning fashion.

Henri et al. [10] evaluated different pretrained CNN architectures on a small subset of bird sounds retrieved from the Xeno-Canto collection. The subset was limited to 36 bird species found in the republic of Mauritius. Due to the imbalanced nature of the dataset, classes with less than five instances were collected together under the "Other Mauritius Bird" label, resulting in a total of 19 classes. The data augmentation techniques "Time Stretching" and "Pitch Scaling" were implemented into the work with the purpose of increasing the training set. Details pertaining the aforementioned augmentations can be observed in chapter 4. Henri et al. finetuned three pretrained models (Inception V3, MobileNet V2 and ResNet50) and trained a custom model from scratch, all for an amount of 20 epochs and different parameter combinations. Except for the poor results of ResNet50 (not discussed in the work), the best pretrained model (MobileNet V2) yielded an accuracy on the validation set of 84.21%, over 2% higher than the custom CNN model trained from scratch, proving that transfer learning can indeed be applied to the domain of classifying bird songs. Future work presented by the authors include the retrieval of more data meant to increase the robustness of the model, test other data augmentation techniques and increase the dataset by producing synthetic bird sounds.

3.2 Semi-Supervised and Unsupervised Approaches in Audio Domain

Rowe et al. [35] utilized autoencoders to explore other feature representations than Spectral Acoustic Indices and MFCC spectrograms, with the goal of small scale analysis of long continuous recordings. The audio was split into non-overlapping

one-second-long clips, FFT computed for each clip and each resulting spectrogram converted to 128x128 pixel. The features produced with the autoencoders were proven to be more separable than Spectral Acoustic Indices, but outperformed by MFCC spectrograms. The models were trained only a few epochs due to hardware constraint, but the time to produce the feature representation after training is comparable to producing MFCC spectrograms, thus having the potential to be a valuable alternative, if trained more.

Zhong et al. [13] evaluated transfer learning and pseudo-label on a multi-label classification task on self collected data. One VGG16 CNN and two ResNet50 architectures were compared, with both ResNet50 being pretrained on ImageNet dataset and one of the networks finetuned with pseudo-labeling. Preprocessing consisted in generating Mel spectrograms with energies converted to dB scale and resized to 224x224 pixels. Image augmentation methods were not utilized in the work of Zhong et al. Both ResNet architectures outperformed the VGG16 CNN, with the combination of transfer learning and pseudo-labeling yielding the highest accuracy. The results show that there is a latent potential within unlabeled data and that pre-learned features from other domain like images can be utilized on sound classification tasks. Nonetheless, a comparison with a non-pretrainined ResNet50 instead of VGG16 CNN would have been more fair.

Grollmisch et al. [14] evaluated the performance of Semi-supervised (SSL) methods that reached state-of-the-art level on image classification, when applied to the audio domain. The chosen SSL methods, FixMatch [36] and Mean Teacher [37], were tested against Transfer Learning (TL) and Supervised Learning (SL). The methods were evaluated on three datasets (IDMT ISA METAL BALL, TUT2017 and NSynth), and the same preprocessing pipeline and CNN architecture were utilized across all datasets. Performance was tested with different ratios of labeled data (1%, 5%, 10% and 20%) and, for some of the datasets, one-shot-learning was also tested. In order for FixMatch to work properly, weak augmentation techniques were to be selected for each datasets. Grollmisch et al. included a novel approach for categorizing augmentation techniques by training a model on a non-augmented training set, and, for each augmentation technique, evaluating the model's accuracy on the augmented training set. Augmentation methods with a resulting averaging accuracy of less than 5% of the non-augmented result, were categorized as weak. Results showed that selection of the right weak augmentation technique is critical to the performance of FixMatch. FixMatch always performed better than Mean Teacher and the baseline CNN trained from scratch. On TUT2017, the most difficult of the datasets, FixMatch was outperformed by the transfer learning method, while, on the other two datasets, FixMatch reached the upper baseline results with less than 5% of the labeled data. For future work, Grollmisch et al. suggests testing raw-audio augmentation techniques, and combining FixMatch with transfer learning, the latter in order to take advantage of more confident pseudo-labels being generated at the beginning of training, and

possibly reducing the confirmation bias of the pseudo labeling process.

On the unsupervised training front, Denton et al. [38] utilized a novel technology to unmix different sources from a sound signal based on the Xeno-Canto dataset . The recently developed method Mixture Invariant Training (MixIT) [39] differs from the supervised counterparts mainly in training on mixture of mixtures (MoMs), and on artificially mixed signals from pure sources. The model estimates a predefined number of different sources, assigns them to one of the original mixture and tries to minimize the loss. The model is also capable of separating less than the predefined numbers of sources, by generating signals that are close to zero. MixIt trains on mixtures of sounds which the Xeno-Canto repository already provides (as weakly-labeled recordings, with always one foreground species and one or multiple background species, if any), and can estimate the original sources from one signal alone. In comparison, Independent Component Analisys (ICA) [40] requires as many signals as separable sources in order to work, a constraint that limits its applicability within bioacustic monitoring, due to signals being at most stereo and recorded from the same microphone. Denton et al. compared the efficiency of MixIT as a separator for bioacustic monitoring when training on domain data (Xeno-Canto dataset) and on general sounds datasets (AudioSet [41]), proving that training on domain data guarantees indeed a performance boost to the system. The method was then tested on three fully-labeled polyphonic datasets from previous BirdClef competitions. Denton et al. compared the classifier accuracy on original (mixed) audio, unmixed sources (a channel for each source) and on both unmixed and original signal (one channel per source + original audio as an extra channel). Employing both unmixed and original sources yielded the best accuracy in almost all the datasets, showing that there is indeed a latent potential in unlabeled data for this problem. Future works include tests on other audio domains and improving classifier accuracy by separation on the classifier training data.

3.3 Structured Literature Review Protocol

The research in its entirety was conducted with the aid of the Google Scholar search motor, while the keywords were evolved and changed during the time of the research. The project description was based on the task description formulated in the Cornell Birdcall Identification task [2], with loose boundaries on what to develop or which part of the task to focus on. The initial research was conducted using the keywords "Cornell Birdcall Identification", "Cornell Kaggle" and "Cornell Challenge". Some of the Kaggle competition deliveries were published in form of code notebook and papers. The "snowball effect" method was initiated from these papers, evaluating the related work, ultimately bringing up the knowledge of a yearly Kaggle competition called LifeClef bird task (later BirdClef), and DCASE, a similar yearly challenge on Detection and Classification of Acoustic Scenes and Events [29].

A search with the keyword "BirdClef" and "LifeClef bird" reported many results regarding delivery publications as well as review papers of the winning approaches, the latter published by the competition organizers. After a thorough reading of those papers it became clear that the shared motivation was to achieve best classification accuracy. A broader search around methods to evaluate was conducted by the participants in the first years, narrowing down to the area around the previous years' winning methods. Deep learning methods governed the scene after showing prominent results in 2016, becoming the base for the majority of the following deliveries. Due to the astonishing number of published work, research and related work covered in papers around BirdClef and DCASE were treated as a time lapse of the bird song classification scene.

In this phase, as new terminology appeared in the reviewed papers, new keywords were added to the search, for instance "Sound event detection", "Polyphonic", "Audio classification" and "Audio Tagging". The stopping criteria followed were limiting the research to bird sound classification or polyphonic event classification, as well as papers not older than 2016, the latter due to the majority of papers published later this time period being based on deep learning methods. An evaluation of the problems found in the works led to a change of motivation for this project, namely discovering the potential of semi-supervised learning on the domain of bird song classification.

Popular Semi-supervised (SL) methods were researched when applied to audio classification and bird songs classification, using the keywords "SSL Semi-supervised methods", "Pseudo-labeling", "Mean Teacher", "audio classification" and "Bird sounds". Research showed fewer papers in comparison to supervised methods on bird songs domain or other audio sound domains, with interesting results regarding the "FixMatch" method published by google in 2020. A research using the keywords "FixMatch" and "bird song", on the other hand, showed no published papers.

3.4 Chapter Summary

The bird songs classification has seen multiple solutions throughout the years, mainly due to yearly competitions since 2014. Deep learning methods dominate the audio classification scene, with CNN, RNN, CRNN or CapsNet architecture designed from scratch, or popular designs initially thought for image classification tasks, like Inception V3, VGG16 and ResNet. Even though images were fed to such networks, different image representations of sounds were tested, such as FFT, mel and MFCC spectrograms, and unsupervised autoencoders vector representations.

The audio domain showed to be more closely related to the image domain, with research findings in the latter being applicable to the former. Some of the works managed to apply transfer learning to an audio classification task, using

weights computed on a bigger image dataset. Another example of domain correlation is the advantage CapsNet holds over CNN in recognising overlapping digits, analogously seen in its superiority regarding classifying overlapping sounds.

FixMatch, SSL method initially developed on the image domain, showed potential when applied to an audio classification task, and was only defeated by transfer learning on the most difficult dataset. The author suggests to combine transfer learning with FixMatch, and to test audio augmentation techniques.

Chapter 4

Method

The technical work done and the design choices taken in regard to designing the system are presented in this chapter. The dataset used is presented in section 4.1. In section 4.2, the used libraries and frameworks are discussed. Section 4.3 conveys the common system pipeline shared by the chosen method FixMatch and baselines, giving an insight into each custom written component and a motivation behind their design choices. FixMatch is discussed in section 4.4, where the algorithm from the original publication is presented, and more in-depth implementational notes and assumptions are provided. Finally, the baselines implementation are discussed in section 4.5.

4.1 Data

4.1.1 Dataset

The dataset is a collection of clips retrieved from the Xeno-Canto website and compiled by Cornell Lab of Ornithology for the "Cornell Birdcall Identification" Kaggle challenge [2]. It contains around 63k recordings divided into 397 folders, each representing a different species. The dataset is imbalanced, with some classes having more than 400 examples while others as little as 30, as shown in Figure A.1. All the sound recordings are digitally sampled at 22.050kHz. Each recording only has one track, meaning that they are mono sound.

Each recording has metadata attached to it, containing the following information:

Primary_label	The main species identified in the recording.
Secondary_labels	Background species with low Signal to Noise Ratio (SNR).
Type	The type of sound, like 'song', 'call' or 'begging call'.
Latitude	Latitude of the site of the recording.
Longitude	Longitude of the site of the recording.
Scientific_name	Scientific name of the main species in the recording.
Common_name	Common name of the main species in the recording.
Author	Full name of the author of the recording.
Date	The recording date of the sample.
Filename	The filename of the recording.
License	The license under which the recording is released.
Rating	The quality of the recording.
Time	The time of recording of the sample.
Url	The unique Xeno Canto url of the recording.

The recordings in this dataset have differing lengths, varying from being a few seconds long to multiple minutes long. Each example has one primary label, i.e., species whose sounds can be heard in the foreground with a high Sound-to-Noise Ratio (SNR), while some may have one or multiple secondary labels, i.e., species whose sounds can be heard in the background with a low SNR. The dataset is weakly-labeled, since the label of the recording lacks temporal reference, i.e., timestamps of where the primary (or secondary) species can be heard in the recordings, similarly to boundary-boxes labels in an object recognition task.

Each sample has a rating from 0.5 to 5, with 5 being the main species vocalization loud and clear, and 0.5 being barely audible. Recordings that do not have a rating are grouped under the rating 0. Figure A.2 shows rating distribution of the dataset. The dataset includes high quality examples with high SNR signals, with 75% of all examples being rated at 3.50 or higher, as seen in Table 4.1.

The primary label is a unique identifier for the species. Ulterior metadata can be recovered in the ebird.org website (https://ebird.org), by typing "/species/" followed by the label name in the url-bar.

Table 4.1: Statistics regarding the ratings of the samples in the dataset.

count	62874.00
mean	3.76
std	1.22
min	0.00
25%	3.50
50%	4.00
75%	4.50
max	5.00

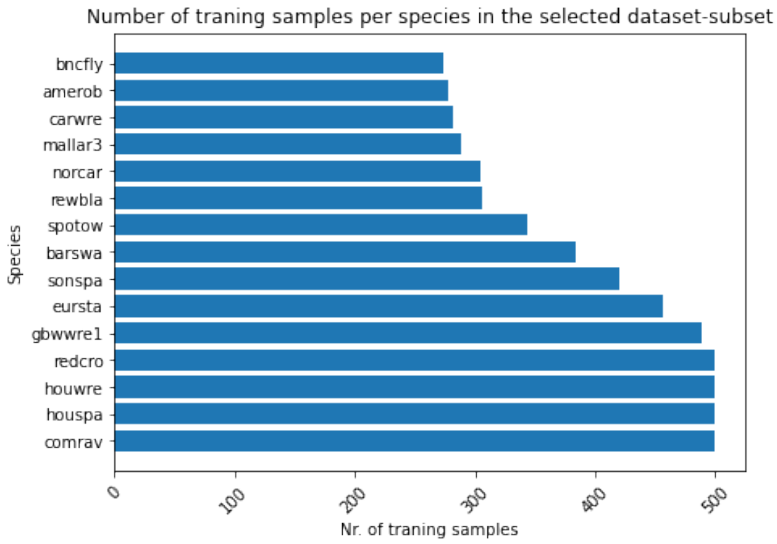


Figure 4.1: Number of recordings per class within the selected subset of the dataset. The figure shows some lingering imbalance between classes.

4.1.2 Dataset Selection

The entire dataset contains a high number of different classes, requiring a noticeable amount of time and computing power in order to run the experiments, therefore only a subset of 15 classes is chosen. The top 15 classes with the highest amount of recordings of high quality, i.e., with a rating of 4 or higher, are selected. Figure 4.1 shows the 15 classes selected for the experiment. The subset is still skewed, with some classes having as much as double the amount of recordings than others.

The metadata "type" of a recording is compiled by the bird watcher upon uploading, whom don't follow any standard regarding on how to fill this info, leading to 254 distinct types with only a few occurring multiple times, like "call" or "song", visible in Figure 4.2. Some of the other types registered are unsure, such as "mimicking?", "not known", "duet?", or multiple types included in the same string, namely "alarm-call and nestlings".

All of the recordings are tagged with positional data. When plotted on a world map, Figure 4.3, it can be noticed that some of the 15 selected species are uniquely found in a specific region, as demonstrated in the case of "carwre" in North America.

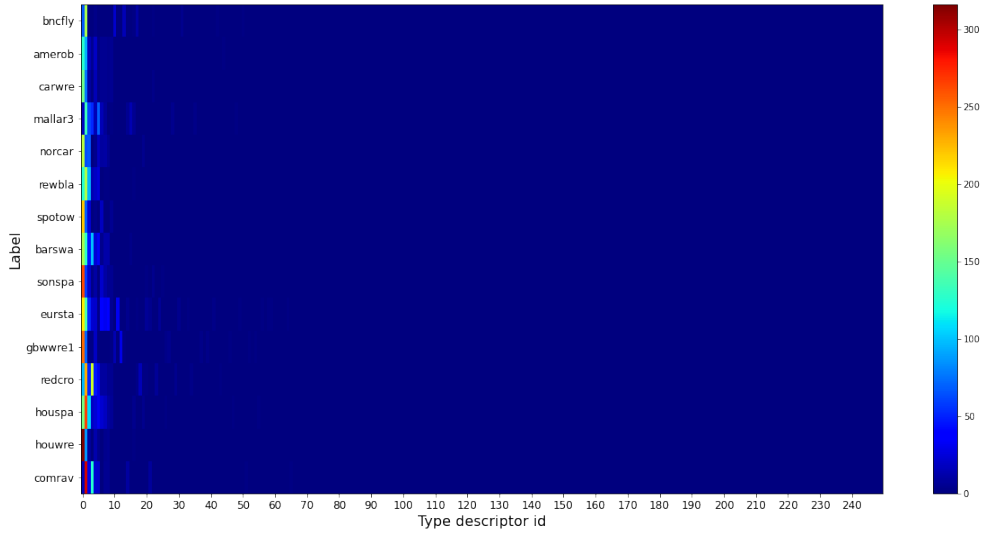


Figure 4.2: All the found instances of metadata "type" for the 15 classes used in this thesis, sorted by number of occurrences. The metadata "type" is assigned by the bird watcher that collects the sample. There are 254 different "type" assigned in the subset, of which only a handful occur multiple times. On the X-axis, the metadata "type" text has been replaced by a number ID. The energies of this plots represent the amount of occurrences, with 315 being the most for a given "type".

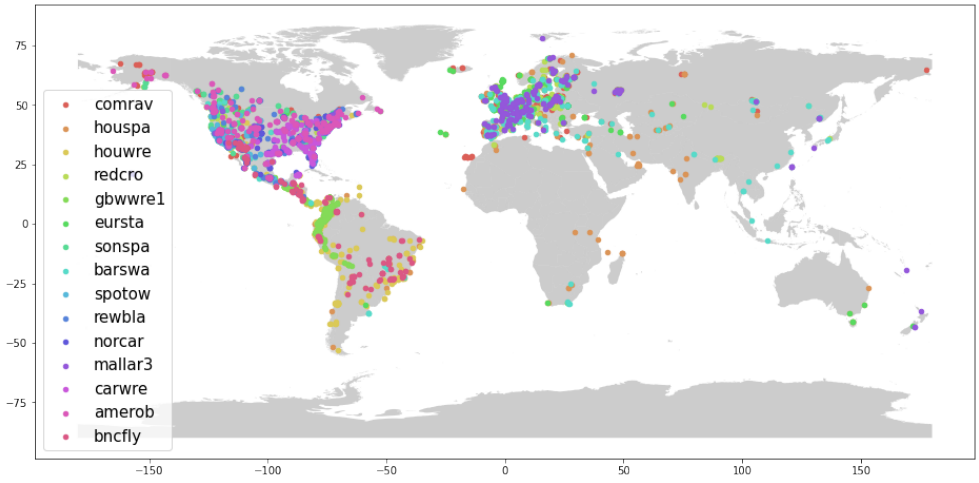


Figure 4.3: Locations included in the dataset for the 15 chosen classes plotted against the world map.

4.2 Frameworks and Libraries

4.2.1 Pytorch

The Pytorch framework allows the user to efficiently set up pipelines for preprocessing either image or sound files. Since image classification tasks are more popular than audio tasks, the included methods and functions in the framework do not allow the user the same degree of freedom in the two cases. For this reason, many of the included functions have been rewritten to handle audio files, and allow a higher degree of manipulation "under the hood" of the system. The downside to this approach is that the loss of the advantages provided by the available built-in functions, thus leading to higher execution time. Some of those advantages being implementational tricks and utilization of workers for parallelizing the code. Trying to include such functionality, developed and optimized by a community of over 2000 contributors, is out of the scope for this thesis. Therefore, some compromises and design choices have been made in order to run the experiments in a reasonable amount of time.

4.2.2 Data Augmentation Libraries

Data augmentation is performed on the audio spectrograms and raw audio data using both the Albumentation and Audiomentation libraries. The former is a fast open source python library with many image augmentation methods, while the latter is a GitHub project inspired by Albumentation that provides a set of audio specific data augmentation methods. The Librosa library is utilized to load and manipulate the audio files. Audioaugmentation methods also employ imports from Librosa. Not every image augmentation method can reasonably be applied onto

a spectrogram, therefore only a small subset of the available transformations are chosen, namely:

Cutout	Removing rectangular regions from the spectrogram, leading to masking some frequencies for a random amount of time.
Shear	Shear transform of the spectrogram, similar to warping of a sound.
Random Brightness	Randomly changing the brightness of the energies in the spectrogram corresponds to increasing/decreasing the volume when seen in the audio domain.
Random Contrast	Randomly changing the contrast of the spectrogram leads to similar results as manipulating the brightness, but the increment is not as linear, meaning that background noise, represented with lower energies, will increment/decrement not as much as the main signal.
Affine-Translate	Translating the spectrogram along the X- and Y-axis corresponds to changing the pitch of the audio signal and moving the audio back and forth on the time domain, respectively.
Grid Distortion	Grid Distortion is a non rigid transformation used in medical imaging problems. The spectrogram is divided into a grid. Time and frequency are distorted differently in each area of the grid.

The audio augmentation methods chosen from the Audiomentation library are the following:

Add Gaussian Noise	This transformation adds Gaussian noise to audio data.
Add Gaussian Noise SNR	This transformation applies Gaussian noise to input with a randomly picked Signal-to-Noise Ratio (SNR) chosen on a logarithmic scale.
Gain	Increasing or decreasing the audio sample volume by multiplying the data-point by a random factor.
Time Mask	Randomly masking all the frequencies of a part of the audio, resulting in muting said part of the audio.
Shift	Translating the audio back and forth along the time domain.
Pitch Shift	Increasing or decreasing the pitch of the audio.
Time Stretch	Stretching the audio along the time domain without changing the pitch.

Some of the selected audio augmentations, like "Add Gaussian Noise SNR" and "Time Stretch", do not have a counterpart present in the selected image transformation for this thesis: although implementations of such augmentations are feasible in the spectrograms domain, the choice of exclusively employing augmentation available in already published libraries has been made in order to meet the deadlines for this project. Other libraries and frameworks used in this project worth mentioning are Numpy, Python Imaging Library (PIL) fork called Pillow, OpenCV, Sklearn, Matplotlib and Optuna.

4.3 System Pipeline

The System Pipeline for both the baselines and the FixMatch methods overlap greatly: Figure 4.4 shows the common components of these pipelines. The "Data-Loader" component is responsible for loading the audio files in memory and serving mini-batches of data when requested. The "Mini-batch Processor" prepares the data as specifically required by each method, making it ready to be ingested by the PyTorch model. The "Method Training Routine" is the specific learning routine for each of the baselines and FixMatch methods.

4.3.1 Dataset Preprocessing

Due to the abundance of data from this dataset, recordings from the 15 chosen categories are split into "train" and "test" with a ratio of 80% and 20% before any run of the experiments, so that the system is tested on the same validation examples throughout all the experiments run. As seen in section 4.1, all the re-

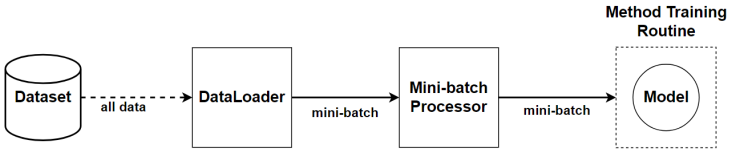


Figure 4.4: A black-box diagram of the system pipeline, showing the shared components by all the methods. The dataset is loaded in memory only once per run, and the "Method Training Routine" is specific for each method, pictured with a segmented box.

cordings are of differing lengths, but they all share the same sampling rate. The audio files have been processed upon submission to the Xeno-canto website by each bird watcher, so that the main species can be identified within the first ten seconds of the recordings. For this reason, each file longer than ten seconds will be trimmed to that length and shorter files will be padded. The padding mode chosen is "wrap", meaning that recordings shorter than ten seconds will simply start again from the beginning of the file, and end when the total length is ten seconds.

The Librosa library is utilized for loading and processing the audio files. It does so by converting the audio files to numpy-arrays of the shape C x S, where C is the numbers of channels (1 for mono and 2 for stereo) and S is the number of sampling-points, the latter being the sampling ratio by the length of the audio file in seconds, as described in subsection 2.1.1.

4.3.2 Dataloader

The custom "Dataloader" component is responsible for loading the audio data and serving mini-batches to the model during training. The dataset is balanced by loading 274 recordings of each label, 274 being the amount of recordings for the class with the less available examples, giving a total of 4110 examples divided equally into 15 classes. The reason behind this choice is that generating new synthetic audio data to balance the classes is not a trivial task, requiring additional advanced machine learning methods like GANs, ultimately resulting in increasing the number of variables in this experiment. Alternatively, more examples could have been retrieved from each audio recording. Nevertheless, since the number of samples seemed to be sufficient in order to test the system, none of those aforementioned methods were explored. Each audio file is then converted to a numpy array of shape C x S and stacked together, making a unique array of shape N x C x S, where N is the total number of examples.

At the beginning of each system run, the training set is randomly split into labeled and unlabeled sub-sets, and each sample affiliation to either sub-set remains unaltered for the entire duration of the run. During each epoch of the training phase, the model encounters each sample of both sub-sets exactly once; how-

ever, the order of which mini-batch they appear in is randomized. Setting the ratio of labeled-to-unlabeled examples to zero results in only one set being generated.

4.3.3 Mini-Batch Processor

This component is in charge of processing each mini-batch of numpy arrays served by the DataLoader component. The process is shown in Figure 4.5, and run sequentially for each sample of the mini-batch. First, if required by the method, audio augmentation is applied to the one-dimensional numpy array using the Audiomentations library. Successively, the Short Time Fourier Transform (STFT) of each audio recording is computed using a "Hann" window function with a size of 2048 samples, corresponding to a signal length of 93ms when the recording is sampled at 22050Hz, and a hop length of 512 samples, meaning an overlap of 25% between adjacent computed fourier transforms. The resulting two-dimensional numpy array is then converted to an Image object with the Pillow library. The Image object is resized to 384x224 pixels using the nearest neighbor re-sampling method, converted back to a numpy array and then flipped: the flipping is not necessary to train the model correctly, if all the samples are consistently oriented toward the same direction, but it is useful if human eyes are to inspect the spectrograms during the run of the system. At this rate, if required, image augmentation is applied using the Albumentations library. The two-dimensional numpy array is now duplicated and stacked along a new dimension three times, simulating the three color channels in an RGB image. Finally, all the three-dimensional samples are converted to PyTorch tensors, normalized with a mean of (0.485, 0.456, 0.406) and a standard deviation of (0.229, 0.224, 0.225), as required by the PyTorch model. The resulting tensors are thus stacked together along a fourth dimension in order to produce a single mini-batch ready to be processed by the model.

4.3.4 Model Architecture

The chosen network for this experiment is ResNet18, due to its established yielding of good results regarding research, leading it to become a staple in the field of bird song classification. The variation of ResNet with 18 layers is then chosen for implementational reasons: Pytorch framework includes a good range of ResNet architectures, ranging from 18 to 152 layers, and with pretrained weights on the ImageNet dataset. Regarding the choice of depth of the network, the assumptions here are that: ResNet18 provides more than enough weights to learn feature representations of the 15 classes, is less prone to overfitting than the deeper variants, and will save a considerate amount of computational time during all phases of the experiment on the utilized hardware (Intel i5 9600K CPU, Nvidia RTX 2070 and 64GB RAM). Both the baselines and the new methods implementation will share the same architecture and common hyperparameters. The goal is to compare the method against a sensible baseline: minimizing the degrees of freedom is

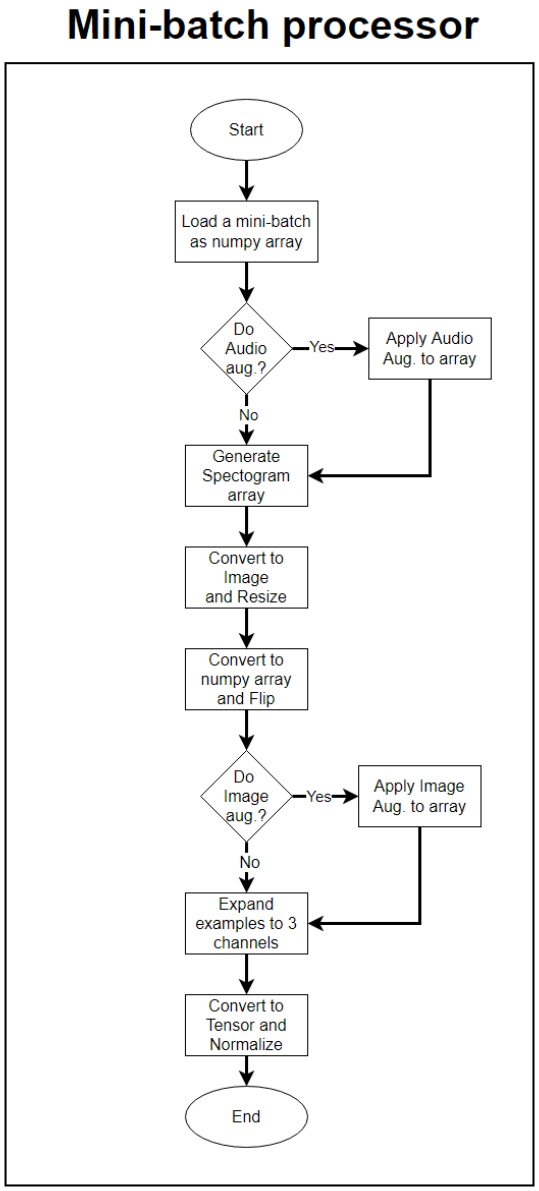


Figure 4.5: Inside the Mini-batch Processor. The mini-batch gets loaded as a numpy array and audio augmentation is applied if needed. The array representing audio datapoints gets converted to a spectrogram array, resized and flipped. Image augmentation is applied on this step if required, then the array is expanded to three identical channels and finally converted to a tensor and normalized, ready to be ingested by the model.

crucial to achieving comparable results, despite the fact that better configuration can exist for each of the methods.

The ResNet18 architecture adapted for this thesis is shown in Figure 4.6, and is composed of an initial convolutional layer with 64 filters of shape 7x7 and a stride of 2, eight convolutional blocks with filters of shape 3x3, and, on the end, a fully connected head with 15 nodes, one for each class. The convolutional blocks are portrayed with four different colors, each representing a different amount of filters at each convolutional layer, doubling every two convolutional blocks. The ResNet architectures allow the model to skip over a convolutional block, a mechanism previously described in subsection 2.3.2.

Pretrained weights tuned on the ImageNet dataset can be loaded in ResNet18: this model includes a fully connected head of 1000 nodes, one for each of the classes in the ImageNet dataset. In this thesis, the last fully connected layer is deleted and substituted with a new head of 15 fully connected nodes with randomly initialized weights. Due to the high domain difference between ImageNet and bird songs, none of the feature extraction layers are frozen.

4.3.5 Hyperparameter Search

A search for effective hyperparameter configurations is conducted using the Optuna library, an optimization framework for Python. The aim is not to maximize the accuracy of the model, but rather finding a good combination of hyperparameters with which the model can perform adequately, leading to a feasible comparison.

All phases of the main experiment were conducted separately for ResNet18 with randomly initialized weights and with pretrained weights. When loading pretrained weights, instead of randomly initialized, the network could prefer a smaller learning rate in order to reach a local optima. For this reason, two separate hyperparameter searches are conducted.

As mentioned in subsection 4.3.4, the methods share the same model architecture, ResNet18, as well as common hyperparameters, therefore the only hyperparameters chosen to tune are the network optimizer and the learning rate. The choice of optimizer is between "Stocastic Gradiant Descent (SDG)" with a fixed momentum of 0.9 and "Adam", with the range for the learning rate set between 1e-6 and 1e-2, using a log-uniform distribution. Each model is trained ten times with a randomly picked optimizer and learning rate. A second search, with ten attempts, was then held around the learning rate of the trained model with the highest score from the previous search, with a fixed choice of optimizer, corresponding to one of the aforementioned models.

The **Adam** optimizer yielded best results for both the randomly initialized and the pretrained model. The best learning rate found were **0.00153** and **0.00032**, for ResNet18 with random and pretrained weights, respectively.

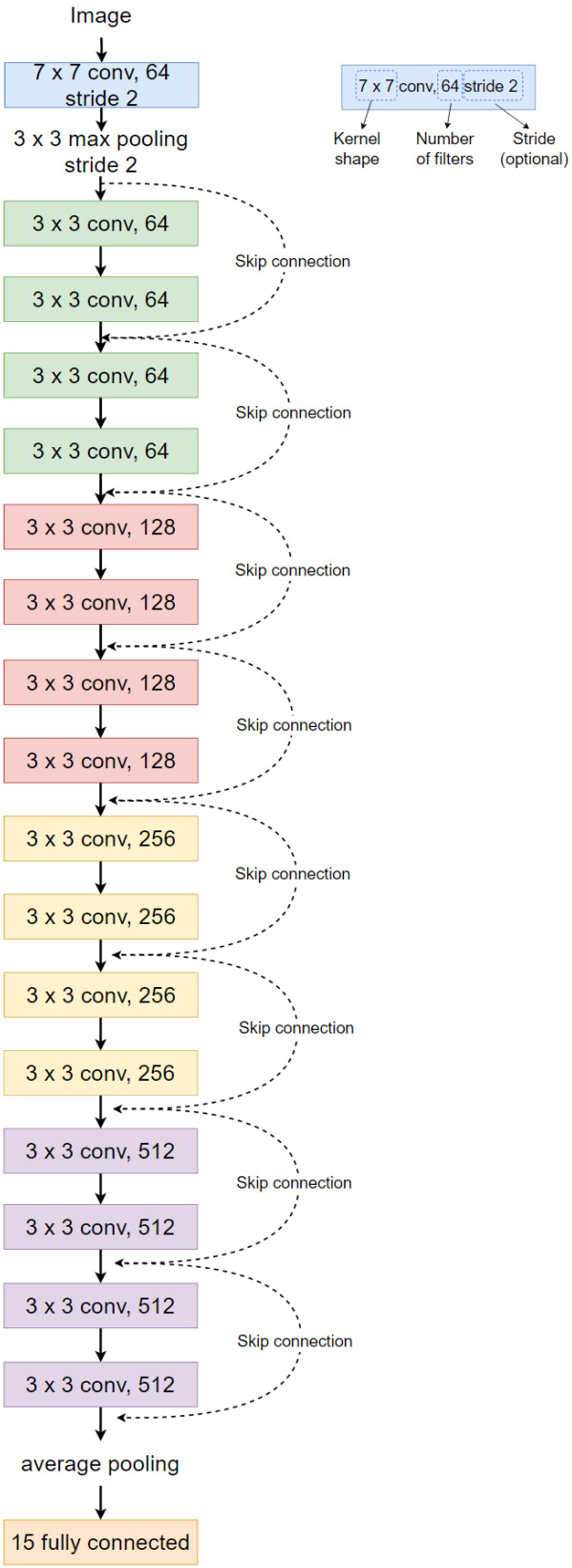


Figure 4.6: ResNet18 architecture used in this experiment. The model consists of four layers, with each having two identical convolution blocks with the possibility of connection skip. Each convolutional block includes two identical convolutional layers. A 15-node fully connected layer sits on top of this architecture.

4.4 FixMatch

4.4.1 Algorithm

FixMatch is the combination of two Semi-Supervised Learning methods, namely Pseudo-Labeling and Consistency Regularization [36]. Consistency Regularization is implemented by image augmentation. The image augmentation techniques are categorized in "weak" and "strong" augmentation, where the results of the former are more closely related to the input than the latter' s are. The model is trained to generate consistent predictions between weak and strong augmentation of the same example. The total FixMatch dataflow can be observed in Figure 4.7.

The loss function used in FixMatch is $L = l_s + \lambda_u l_u$, composed by the cross-entropy loss applied to the labeled batch and a cross-entropy loss on the unlabeled batch multiplied by a scalar λ_u, the latter used to adjust the importance of the unlabeled examples during the training.

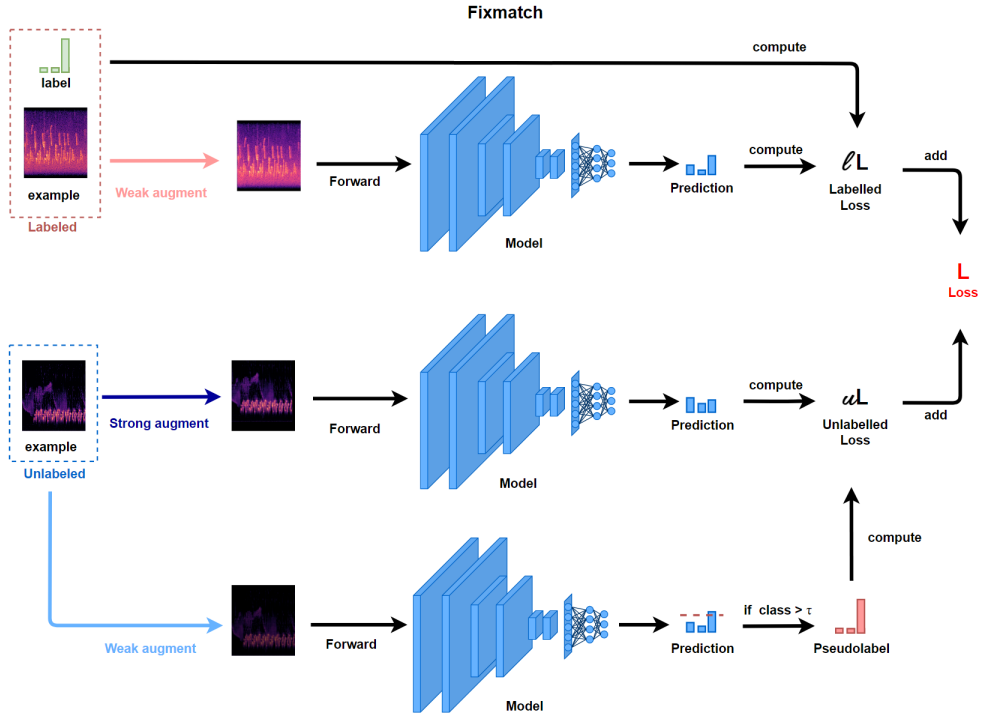


Figure 4.7: FixMatch learning pipeline shown with one labeled and one unlabeled example. The hard pseudo-label is generated from the model prediction of the weakly augmented image, but only if the highest class is predicted with a value higher than the confidence threshold τ. Cross-entropy loss is computed using the aforementioned pseudo-label and the strongly augmented version of the same unlabeled image. The labeled loss is the cross-entropy loss of the weakly augmented labeled image and its label. The labeled and unlabeled loss are combined and minimized by the model.

The labeled loss l_s, showed in Equation 4.1, is the cross-entropy loss calculated

on the model output prediction on the **weakly augmented** examples $p_m(y|\alpha(x_b))$ and their relative labels p_b. The notation utilized in Equation 4.1 includes $\alpha()$, a stochastic weak augmentation function, B, the total amount of examples in the labeled batch, and $H()$, the cross-entropy loss function.

$$l_s = \frac{1}{B}\sum_{b=1}^{B} H(p_b, p_m(y|\alpha(x_b))) \tag{4.1}$$

The unlabeled loss, shown in Equation 4.2, is the cross-entropy loss calculated on the model prediction of the **strongly augmented** unlabeled examples $p_m(y|A(u_b))$ and the model generated pseudo-labels $\hat{q}_b$. The notation utilized in **strongly augmented** includes $A()$, a stochastic strong augmentation function, and μ, the ratio of unlabeled-to-labeled examples. The pseudo-labels are generated by one-hot-encoding the model output probability of the **weakly augmented** unlabeled examples, but only if the dominant class in each output probability distribution has a value greater than the confidence threshold τ. An example of producing a pseudo-label out of the model prediction can be seen in Figure 4.8, while the mechanism of deciding which training example to include in each training epoch is shown in Figure 4.9.

$$l_u = \frac{1}{\mu B}\sum_{b=1}^{\mu B} \begin{cases} H(\hat{q}_b, p_m(y|A(u_b))) & max(q_b) > \tau \\ skip-example & max(q_b) \leq \tau \end{cases} \tag{4.2}$$

4.4.2 Mini-Batch Learning

Some implementational changes to the FixMatch algorithm have to be made due to the limited hardware capability of the available resources. The FixMatch algorithm is based on batch learning, a learning method where, during each epoch, the total loss is achieved by averaging all the training example losses and the model weights are updated exactly once. The total labeled and unlabeled losses are computed separately, by passing all the labeled and unlabeled examples through the model, and then added together to produce the total loss. Since the PyTorch framework keeps computational graphs in memory for each of the passed example tensors, batch learning with these many training examples is not feasible, since it will require to have all the training examples computational graphs loaded in memory. On the other hand, in mini-batch learning each training epoch is divided into different steps. During each step, the loss is calculated on a small amount of examples, also called mini-batch, and the models weights updated.

In this FixMatch implementation with mini-batch learning, each mini-batch will contain both the labeled and unlabeled examples in the desired ratio. The mini-batch size in this system is set to 40, giving four labeled examples and 36 unlabeled examples when a ratio of 90% unlabeled examples is utilized. This setup is in line with the FixMatch method by Sohn et al. [36], where the labeled loss is computed on B labeled examples at each epoch, and the unlabeled loss computed

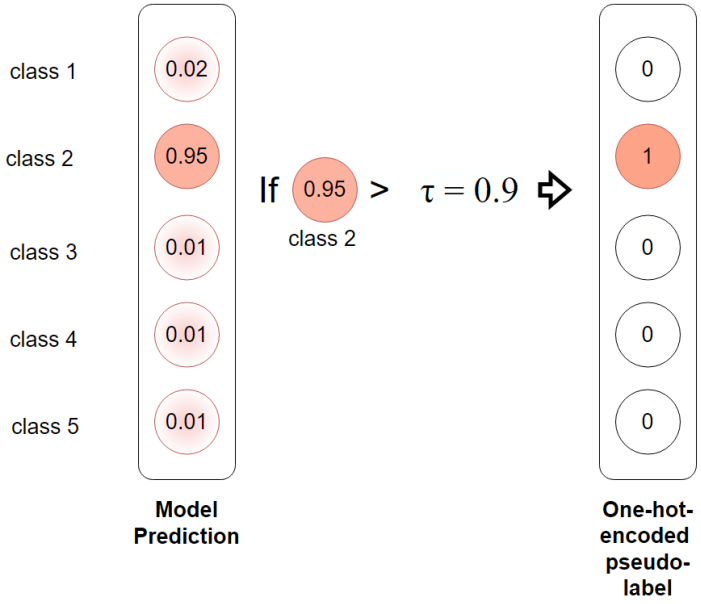


Figure 4.8: Example of pseudo-label generation. On the left the model prediction of a weakly augmented unlabeled example can be observed, where "class 2" is predicted with a confidence of 0.95. Since 0.95 is greater than the confidence threshold τ of 0.90, the one-hot-encoded pseudo-label is produced by setting the predicted "class 2" value to 1 and 0 for the other classes. This is also defined as "hard label". If the predicted class had a lower value than τ, then the unlabeled example would have been discarded during this training epoch.

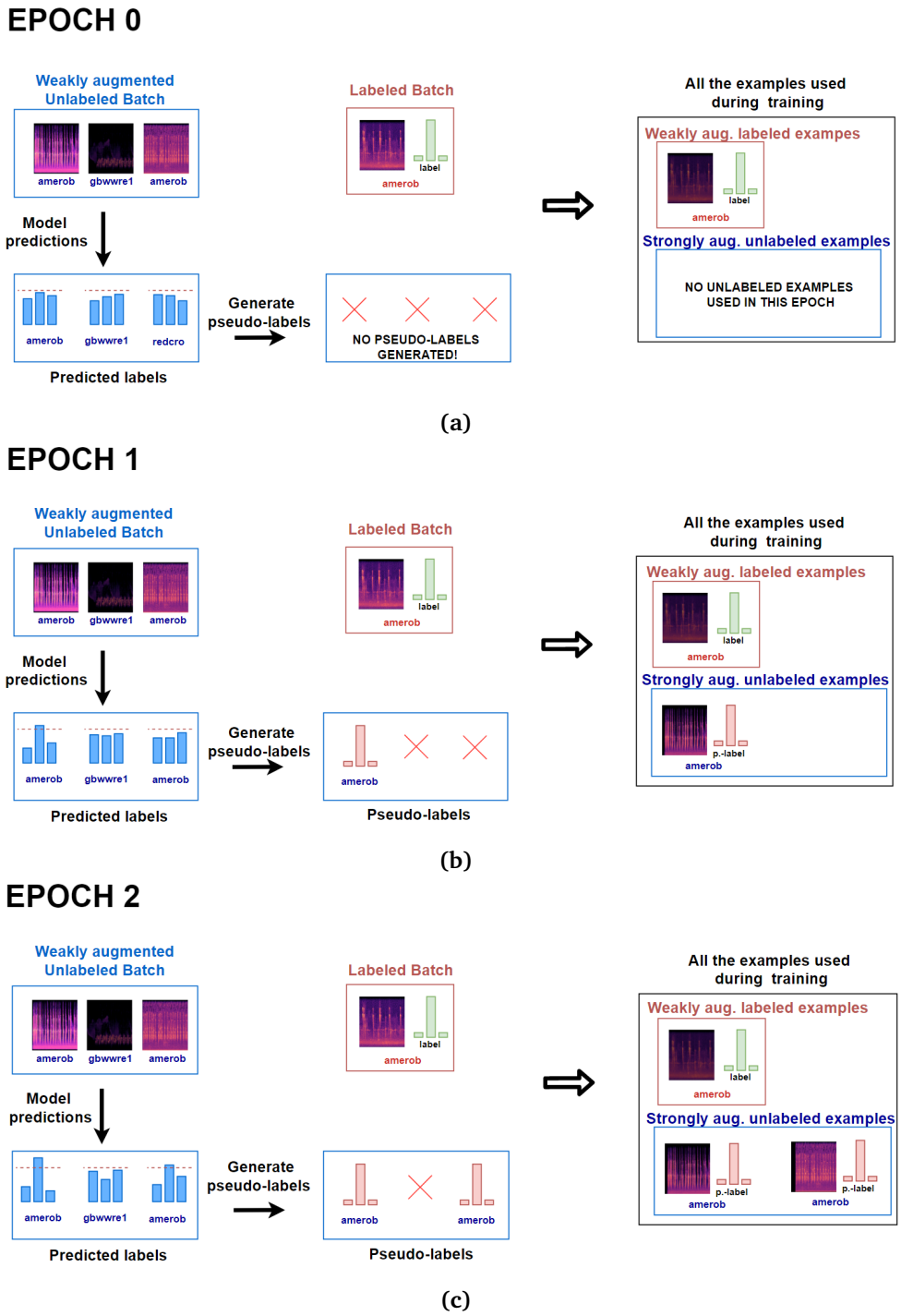


Figure 4.9: How FixMatch chooses the examples to include in a training epoch. The entire labeled batch is always included in the training examples in its entirety. In **epoch 0 (a)**, no predictions on the weakly augmented unlabeled batch were of high enough confidence (the dotted red line being the confidence threshold), hence no pseudo-label were produced, leading to all the unlabeled example being excluded during this training epoch. In **epoch 1 (b)**, the model, previously trained on a labeled example of the "amoreb" class, recognized an unlabeled "amoreb" spectrogram and generated the right pseudo-label, that led to the inclusion of the strongly augmented "amoreb" spectrogram in the training pool. Finally, in **epoch 2 (c)**, the model became more confident about the "amoreb" class, as it recognised and produced two pseudo-labels of the unlabeled "amoreb" examples.

on μB unlabeled examples, respectively, 4 and 36, when the labeled batch size B is 4 and the unlabeled to labeled ration μ is 9. The entire system pipeline is presented in Figure 4.10.

4.4.3 Confidence Decay

One of the components of FixMatch is the generation of pseudo-labels with a high confidence threshold. In the original publication experiments by Sohn et al., this threshold parameter is set to 95% for the entire duration of the experiment. In this system, the confidence threshold is set to 98% and it can be decayed with time, accepting more unlabeled data as training examples with the passing training epochs. This comes with the assumption that the model becomes more precise with the training epochs passing, therefore increasing the trust of the model at every epoch will allow multiple examples to be included as training examples. Since the unlabeled set can contain examples of other classes than the 15 in this experiment, a higher confidence threshold than the work by Sohn et al. is chosen.

4.5 Baselines Implementation

To reduce the number of variables when comparing the FixMatch trained method against the baselines, the same pipeline showed in Figure 4.10 is employed. Each mini-batch is weakly augmented, since data augmentation works as a regularization technique and prevents overfitting. In order to keep a sensible comparison, the mini-batch size is set to be the same as the labeled part when training with FixMatch, meaning that, in the case of 10% labels being used, a mini-batch size of 4 is set for Supervised Learning, 10% of the total mini-batch size used in FixMatch. The metric used to evaluate the baselines and FixMatch is F1-score with micro averaging, as the dataset does not contain any imbalanced classes.

4.6 Chapter Summary

In this chapter the details of the system build for this thesis were described. Custom modules for loading, serving and processing mini-batches were developed, the latter with the possibility of applying either data augmentation on either the raw audio or the processed spectrogram of each recording.

FixMatch was implemented as of the original publication, with the sole exception of exchanging batch learning with mini-batch learning. FixMatch and the SL baseline share the same pipeline, with the latter ignoring the unlabeled examples. Both methods will be tested on the same model architecture and hyperparamenters: pretrained and non-pretrained ResNet18 models differ only in the learning rate value used during training.

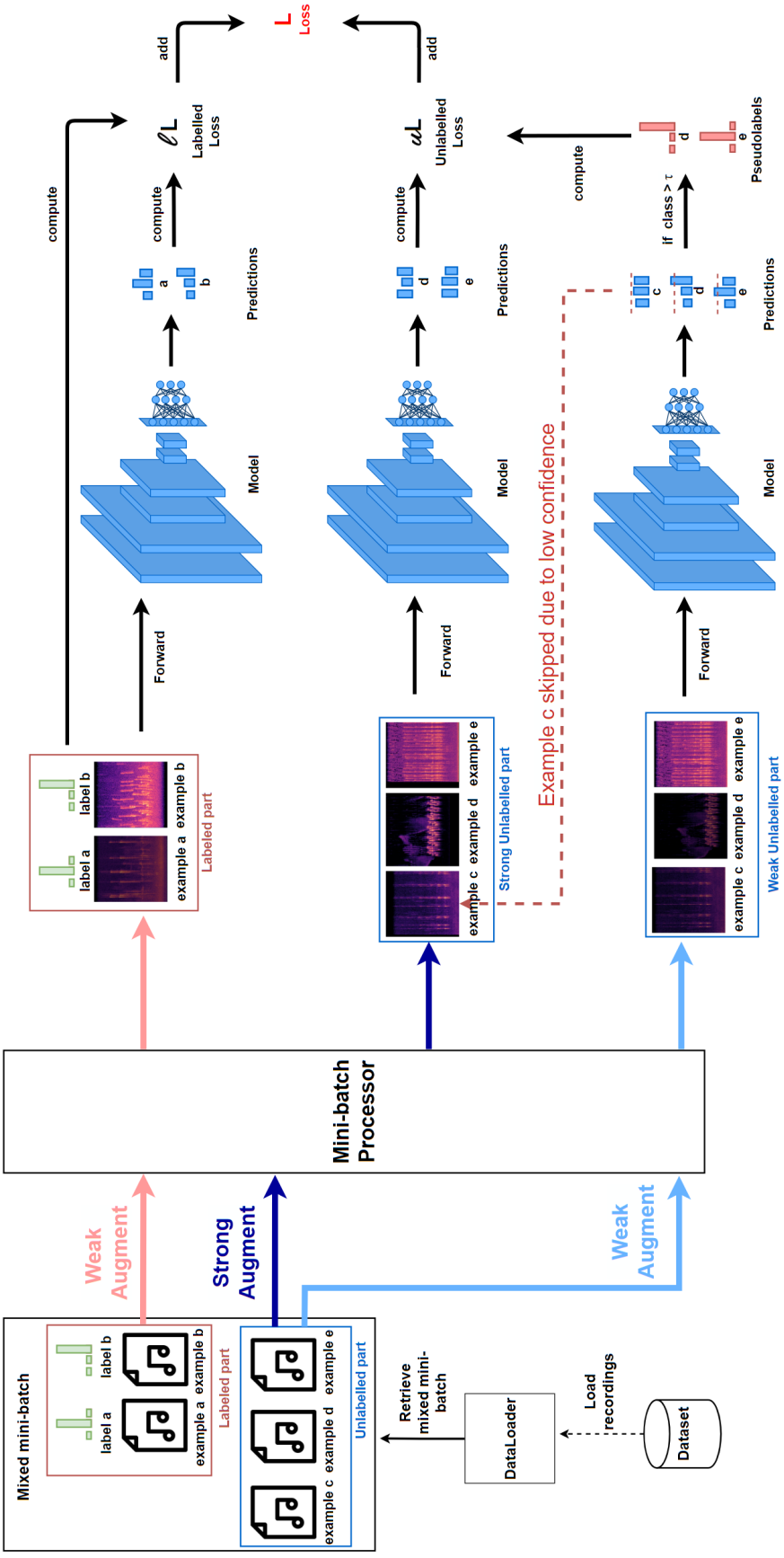


Figure 4.10: The entire system pipeline when training using FixMatch. In SL, the unlabeled part of the mini-batch is discarded and only the labeled loss l_s computed.

Chapter 5

Results

This chapter presents the findings and analysis of the experiments conducted using the system discussed in chapter 4. The experiments plan is presented in section 5.1, while the results of each experiment and immediate remarks in section 5.2. Analysis and discussion follows in section 5.3.

5.1 Experiments Plan

A detailed experiment plan is described in this section. The hyperparameters for learning and the model architectures are commonly shared by all the experiments, in line with subsection 4.3.5.

Augmentation techniques categorization *Categorizing image and audio augmentation techniques as "weak" or "strong".*

The Fixmatch method is based on the application of a "weak" and a "strong" augmentation technique at different stages of the method's pipeline. When applied, a "weak" augmentation technique does not particularly differentiate the ultimate results from the original. On the contrary, a "strong" augmentation technique will produce more distorted and manipulated results. This experiment is based on the work of Grollmisch et al. [14] and consists in training a model on the labeled part of the dataset without the use of augmentation techniques. Then, for each augmentation technique there is to test, the model's predictions on the augmented training set will be evaluated: "weak" augmentation techniques are expected to perform similarly to the case in which no augmentation is utilized, while "strong" augmentation technique to allegedly perform worse.

Main experiment *Comparing the performance of SL and FixMatch on 10% of the labels , with and without the use of pretrained weights.*

All of the FixMatch runs will be performed both with audio and image augmentation techniques. The "weak" and "strong" augmentation techniques for both audio

and spectrogram images are chosen from the results of the "Augmentation techniques categorization" experiment, and shall thus be utilized for this and all succeeding experiments. Each model will be trained for 25 epochs on a 10 % labeled dataset, for 20 runs.

The data augmentation variation type, either image or sound, providing the best performance will be utilized as a base for all upcoming runs.

Long Training *Convergence test by training best models configuration for a longer time.*

To test convergence, the best models will be allowed to train for 100 epochs.

All labels comparison *Comparison of FixMatch performance against a theoretical maximum.*

FixMatch performance will be benchmarked against two theoretical maximums: the case where the amount of generated pseudo-labels during FixMatch training are added as regular labels to the available total in SL, and that in which 100% of the labels are available.

Variations to FixMatch *Testing FixMatch performance with different confidence threshold and loss weighting strategies.*

Confidence threshold is one of the main mechanisms in FixMatch. The tweaking of this parameter leads to the inclusion of different amounts of pseudo-labels in the training batch.

Three strategies will be compared: fixed confidence threshold, continuous decay of the confidence threshold, and increase of the unlabeled loss weight. Respectively, they work as such:

- the confidence threshold remains unchanged during the entire training.
- the confidence threshold will be reduced by 0.0006 after each epoch, reaching a lowest of 92% in the last training epoch.
- the unlabeled weight is linearly increased, starting from 0 and reaching 1 in the last epoch, meaning that, in the latter, the labeled and unlabeled loss will weight the same on the total loss calculation.

All the models are allowed to train for a total of 100 epochs on 10% of the labels. The starting confidence threshold is set to 98% for all the models.

Various unlabeled ratios *FixMatch and baselines performance on different ratios of available labels.*

FixMatch as a Semi-Supervised method shows its potential when fewer labeled examples are available. In this experiment, the best configuration will be benchmarked at 5%, 10% and 20% of the available labels.

Unknown classes in unlabeled set *FixMatch performance benchmarked on an unlabeled set with unknown classes.*

The FixMatch algorithm is tested on a dataset that includes unlabeled samples which don't belong to any of the 15 classes the model is trained to recognize. Only 50% of the unlabeled set will consist of examples of the 15 chosen classes, simulating a scenario where spectrograms are retrieved from soundscape recordings, as the presence of examples of only relevant classes within the unlabeled set cannot always be guaranteed. Since less than 4% of all the classes found in the dataset are utilized in this thesis, building a 50% unknown unlabeled set from the available recordings seemed to be the most sensible approach for this experiment.

Other choice of augmentation techniques *FixMatch performance using different strategies for choosing the augmentation techniques.*

The FixMatch algorithm will be evaluated using a less strong and a less weak augmentation technique.

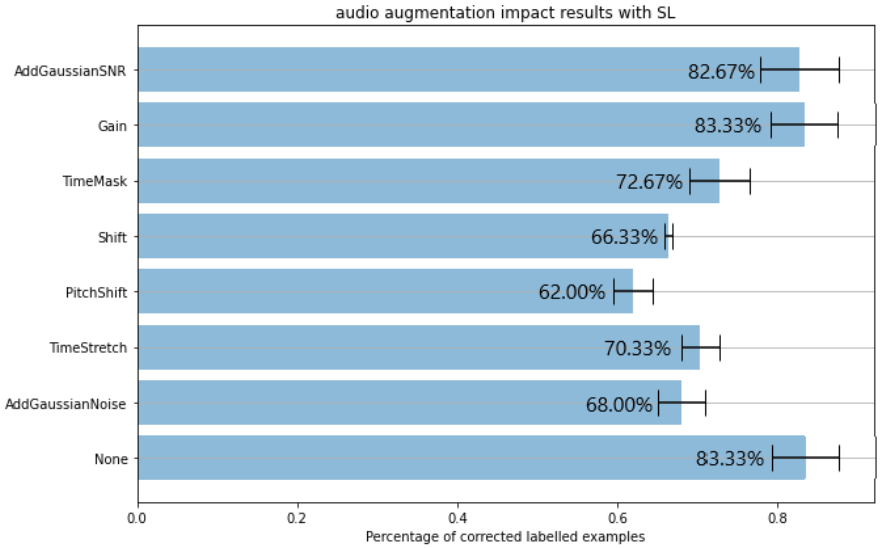
5.2 Experiment Results

5.2.1 Augmentation Techniques Categorization

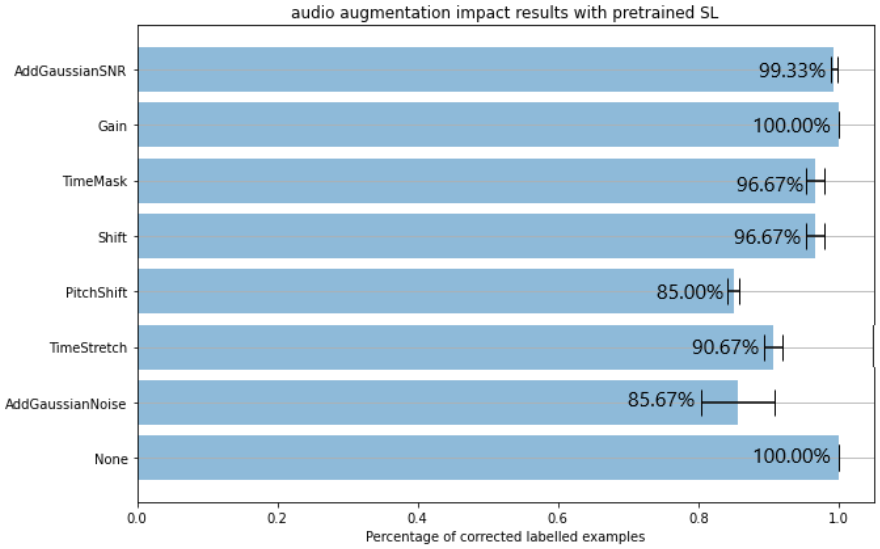
For each of the two data augmentation experiments, ten ResNet18 models are trained for 25 epochs on 10% of the labels using SL. No data augmentation methods are used during training. Of these models, five have random initialization weights, whereas the remaining five have pretrained weights. Figure 5.1 shows the averaged model F1-score when evaluating on the same set used in the training phase, this time applying audio augmentation techniques on each sample. The results show that ResNet18, with random weights, achieves a training F1-score of 83.33% when no augmentation is used. The weakest augmentation technique is "Gain", with an accuracy of 83.33%, performing similarly to not applying any augmentation, while the strongest is "PitchShift", with an F1-score of 62.00%.

Models with pretrained weights achieve a 100.00% F1-score on the training set, recognizing all the training examples when no augmentation techniques are used. The weakest augmentation technique is still "Gain", with 100.00% accuracy. Analogously, the strongest remains "Pitch Shift", with an F1-score of 85.00%.

Figure 5.2 shows the averaged model F1-score when tested on the same set used in the training phase, this time applying image augmentation techniques on all samples. The average F1-score of ResNet18 with random weights on the non-augmented training set is 78.67%: the difference between the latter and the 83.33% score seen in Figure 5.1a is due to different initialization seeds being utilized when training the ten ResNet18 models under the audio augmentation study and the ten ResNet18 models for the image augmentation study.



(a)



(b)

Figure 5.1: Percentage of correctly predicted labels of the audio augmented training dataset using ResNet18 model with randomly initialized weights (a) and with pretrained weights (b).

The weakest image augmentation is "Shear Y" with 77.33%, while the strongest is "Random Brightness" with 31.33%. When loading pretrained weights, the ResNet18 models reaches 100.00% accuracy on the training set. The weakest augmentation is "Shear Y" with 99.67%, and the strongest is "Random Brightness" with 71.00%, consistent with the results found when no pretrained weights are loaded.

Audio augmentation from the Audiomentation library is apparently slower to apply than their image counterpart from the Albumentation library. Table 5.1 shows the time needed to apply each audio augmentation to the entire training set, with "PitchShift" being the slowest at 1 minutes and 43 seconds. The image augmentation techniques have a consistent time utilization of 10 seconds, as showed in Table 5.2.

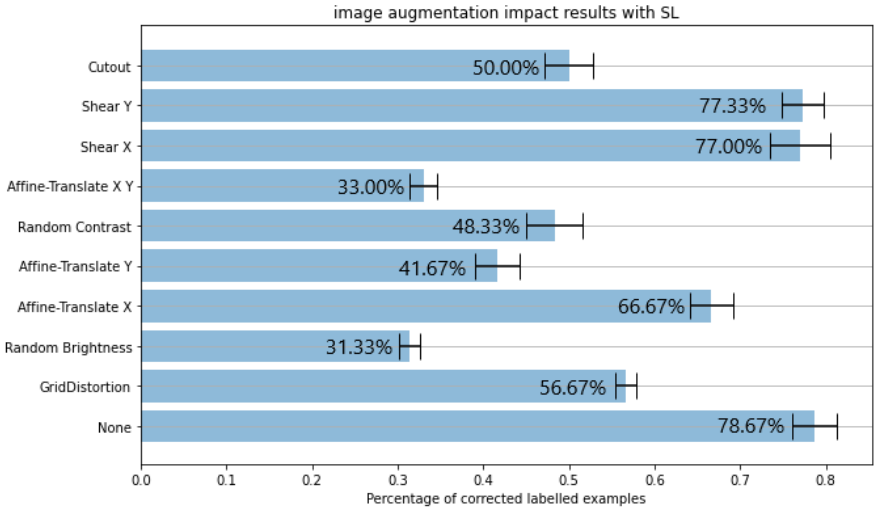
Table 5.1: Time needed for each Audio augmentation technique to be applied to all examples of the training set. The time performance of the different Audiomentations augmentation methods seems to vary, with "TimeStretch" and "PitchShift" being the slowest.

Technique	Time Utilization
AddGaussianNoise	0m 11s
TimeStretch	0m 43s
PitchShift	1m 39s
Shift	0m 10s
TimeMask	0m 10s
Gain	0m 10s
AddGaussianSNR	0m 12s

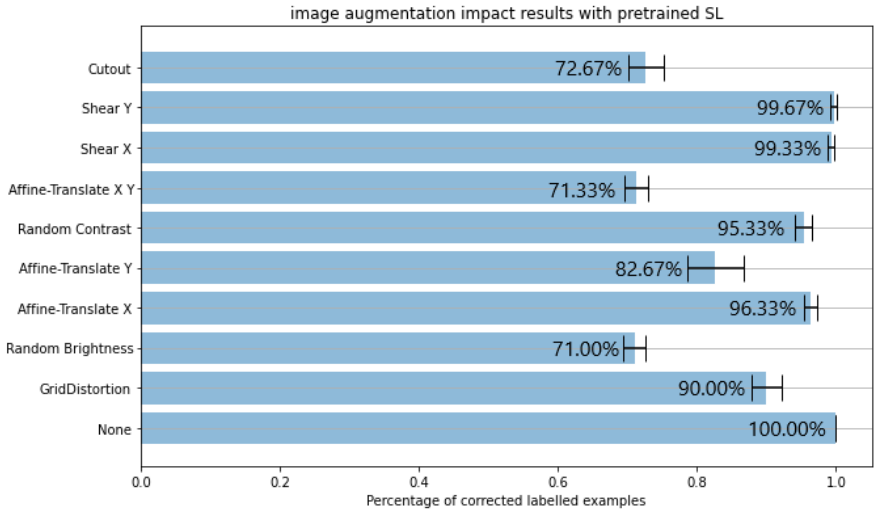
Table 5.2: Time needed for each Image augmentation technique to be applied on every example of the training set. Albumentation augmentation methods reportedly utilize the same amount of time.

Technique	Time Utilization
GridDistortion	0m 10s
Random Brightness	0m 10s
Affine-Translate X	0m 10s
Affine-Translate Y	0m 10s
Random Contrast	0m 10s
Affine-Translate X Y	0m 10s
Shear X	0m 10s
Shear Y	0m 10s
Cutout	0m 10s
Flip	0m 10s

The data augmentation techniques chosen as basis for training with FixMatch and SL in the upcoming experiments, are the weakest and strongest of each data



(a)



(b)

Figure 5.2: Percentage of correctly predicted labels of image augmented training dataset using ResNet18 model with randomly initialized weights (a) and with pretrained weights (b).

augmentation type, as shown in Table 5.3.

Table 5.3: The chosen audio and image augmentation techniques when training with FixMatch (both weak and strong) and SL (only weak)

Type	Weak	Strong
Audio	Gain	Pitch Shift
Image	Shear Y	Random Brightness

5.2.2 Main Experiment

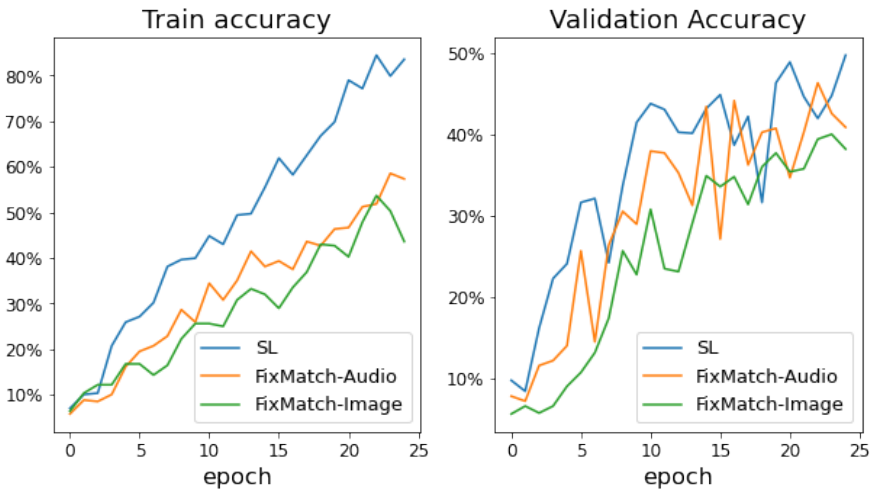
Table 5.4: Averaged F1-score of the model after 20 runs with different initialization seeds. Either Audio or Image augmentation techniques are utilized in FixMatch. Each model trained on a dataset with 10% of labels.

Model	Accuracy	P-value
SL	**48.53%**	-
FixMatch-Audio	45.37%	1.04e-2%
FixMatch-Image	41.22%	1.97e-4%
SL-Pretrained	62.83%	-
FixMatch-Audio-Pretrained	65.63%	2.79e-3%
FixMatch-Image-Pretrained	**67.07%**	1.72e-5%

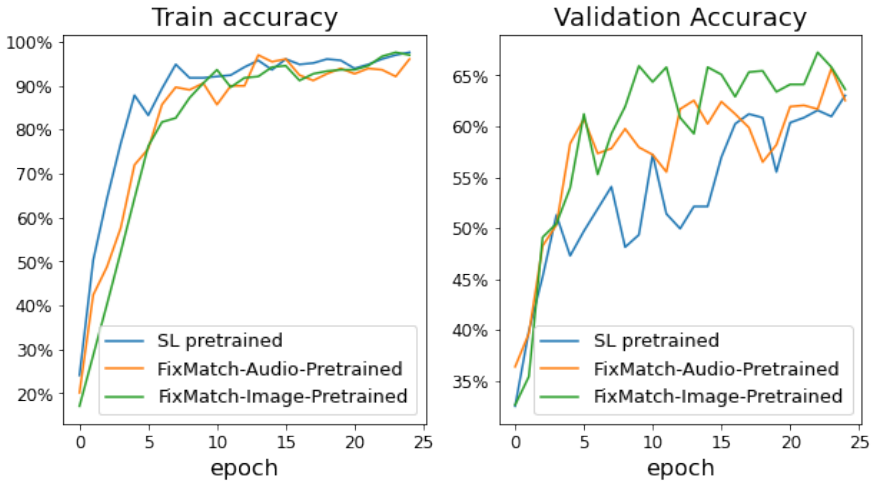
Each of the models is trained for 25 epochs on a dataset composed of 10% of the labeled examples. The results shown in Table 5.4 are the averaged F1-score obtained from 20 runs using different initialization seeds. When training with Supervised Learning (SL), the model achieves an F1-score of 48.53% with random initialization weights and 62.83% when loading pretrained weights. Training with FixMatch, using only the audio augmentation techniques "Gain" as weak and "Pitch Shift" as strong, yielded an accuracy of 45.37%, as opposed to 65.63% when loading pretrained weights.

Applying the two-tailed unequal variance Student-T test on the population of 20 SL samples and 20 FixMatch-Audio samples resulted in a P-value of 1.04e-2%. Student-T test applied on 20 samples of SL-Pretrained and pretrained FixMatch-Audio-Pretrained produced a P-value of 2.79e-3%. Applying the same test on the 20 samples of FixMatch-Image and FixMatch-Image-Pretrained showed similar results, with P-values of 1.97e-4% and 1.72e-5%, respectively. Because the resulting P-values are lower then 5%, the null-hypothesis can be rejected and the results deemed statistical relevant.

A higher sample population is preferable, since a study with 20 samples for each population has a low statistical power. Nevertheless, the number of samples is limited by the hardware resources available as well as the lower performing custom components written for this experiment and used in lieu of higher performing modules included in Pytorch. Since the total computational time needed



(a) Methods comparison with randomly initialized weights



(b) Methods comparison with pretrained weights

Figure 5.3: Main experiment. Results of training with FixMatch using either audio or image augmentation and SL are shown in this picture. The plots show train and validation F1-score for each method when training a ResNet18 model with randomly initialized weights (a) and with pretrained weights (b). For each method, the model with the medial F1-score is picked to be plotted.

to retrieve one sample for each model is about six hours, a trade-off has been made where only 20 samples per model are generated.

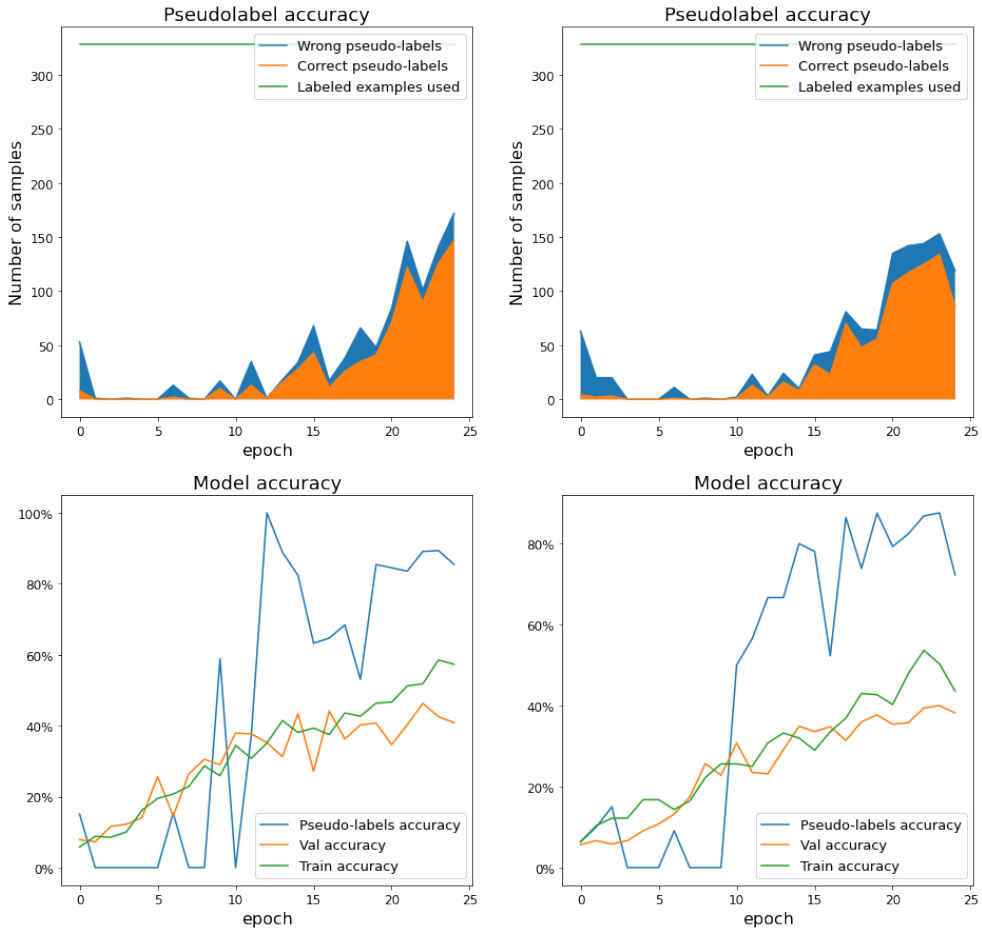
The results shown in Table 5.4 picture two different situations perfectly split according to whether pretrained weights are employed or not. The performance the FixMatch trained models with randomly initialized weights are worse than using Supervised Learning (SL) in both the case where audio and image augmentation methods are used, as seen in Figure 5.3a. FixMatch-Audio models have a lower average F1-score of 3.16% compared to SL, while FixMatch-Image models performed on average 7.31% lower than SL.

Only few pseudo-labels are produced and included in the training routine during FixMatch, shown in Figure 5.4, with an approximate peak of about half the number of labeled examples. The number of pseudo-labels produced during each training epoch is consistent when utilizing either audio augmentation or image augmentation.

A similar amount of correct pseudo-labels is produced by both FixMatch-Audio and FixMatch-Image, explainable by the fact that weak augmentation performs similarly to no augmentation being used. The ability of the model to handle spectrograms augmented with the strong augmentation technique is the major factor in explaining the worse performance of FixMatch-Image compared to FixMatch-Audio: ResNet18 model trained only on labeled examples appears to correctly classify spectrograms augmented using "Random Brightness" with 31.33% accuracy, as shown in Figure 5.2a, against the 62.00% of "Pitch Shift", shown in Figure 5.1a.

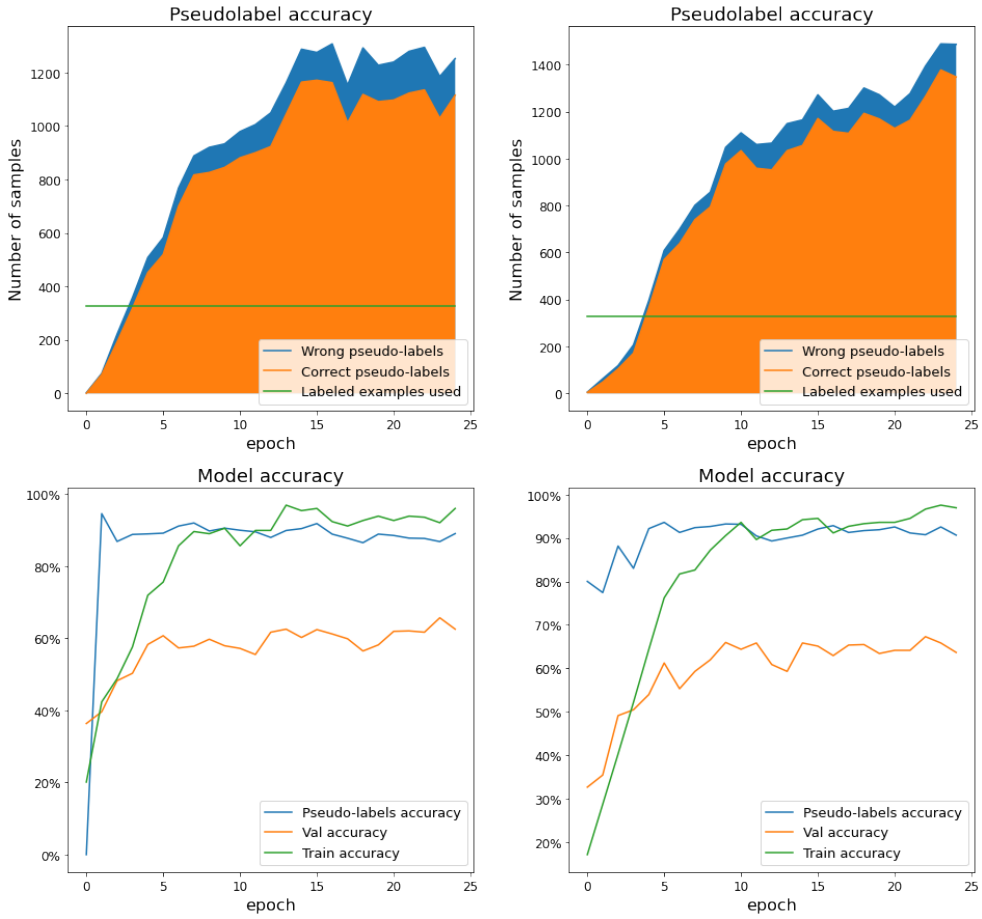
When pretrained weights are loaded, the FixMatch trained models perform better than SL trained ones, both when using audio or image augmentation, as seen in Figure 5.3. FixMatch-Audio-Pretrained and FixMatch-Image-Pretrained seem to be simultaneously better at generalizing and less prone to overfitting. The ability of pretrained model to be more confident about its prediction from the start translates in more pseudo-labels being produced and thus included in the training phase, as shown in Figure 5.5. FixMatch-Audio-Pretrained manages to produce over 1250 pseudo-labels in the later training epochs, while FixMatch-Image-Pretrained manages to produce almost 1500 pseudo-labels, approximately four times the amount of labeled examples used. The correctness ratio of the generated pseudo-labels converges in both models, laying just below the training accuracy curve.

Results in Table 5.4 show that training ResNet18 using FixMatch with either augmentation type provides a worse performance than SL when no pretrained weights are loaded, and better than SL when they are. Due to similar patterns seen in pseudo-labels generation and accuracy performance, only image augmentation will be used in the upcoming experiments, as it is the less time-demanding data augmentation method, as seen in Table 5.1 and Table 5.2.



(a) FixMatch-Audio **(b)** FixMatch-Image

Figure 5.4: Main Experiment. Pseudo-labels produced during training with FixMatch with either Image or Audio augmentation. Plots with FixMatch-Audio are shown in figure A, while plots using FixMatch-Image are showed in figure B. In each figure, the upper graph shows the amount of wrong and correct pseudo-labels. Train and validation accuracy are plotted in the lower graphs, along with the accuracy of the pseudo-labels used for training. For each method, the model with the medial F1-score is picked to be plotted.



(a) FixMatch-Audio-Pretrained **(b)** FixMatch-Image-Pretrained

Figure 5.5: Main experiment. Pseudo-labels produced during training with FixMatch with either Image or Audio augmentation. Plots regarding FixMatch-Audio-Pretrained are shown in figure A, while for FixMatch-Image-Pretrained in figure B. In each figure, the upper graph shows the amount of wrong and correct pseudo-labels produced. Train and validation accuracy are plotted in the lower graphs along with the accuracy of the pseudo-labels used for training. For each method, the model with the medial F1-score is picked to be plotted.

5.2.3 Long Training

Training for 25 epochs does not show convergence, as it can be observed in the graphs from Figure 5.3. Results of training the model for 100 epochs are presented in Table 5.5, while plots accounting for the development of the models' accuracy and error rate are shown in Figure 5.6. All the models show convergence on both training and validation accuracy. All models except FixMatch-Image manage to learn the training set. SL and FixMatch models show signs of overfitting from around the 30th epoch, as shown in Figure 5.6b, where the validation error starts to increase as the training error converges close to 0, meaning that the model is memorizing the training set instead of learning general knowledge from it.

FixMatch-Image-Pretrained still performs better than SL-Pretrained, with an F1-score curve laying almost always over the SL-Pretrained curve, with a P-Value of 1.24%. This result can be deemed statistically relevant. Regarding FixMatch-Image, the average of ten runs showed an F1-score of 49.11%, lower than SL and its 49.84%, but with a P-Value of 17.64%. This result has a high chance of being achieved due to randomness, and therefore to not be considered statistically relevant.

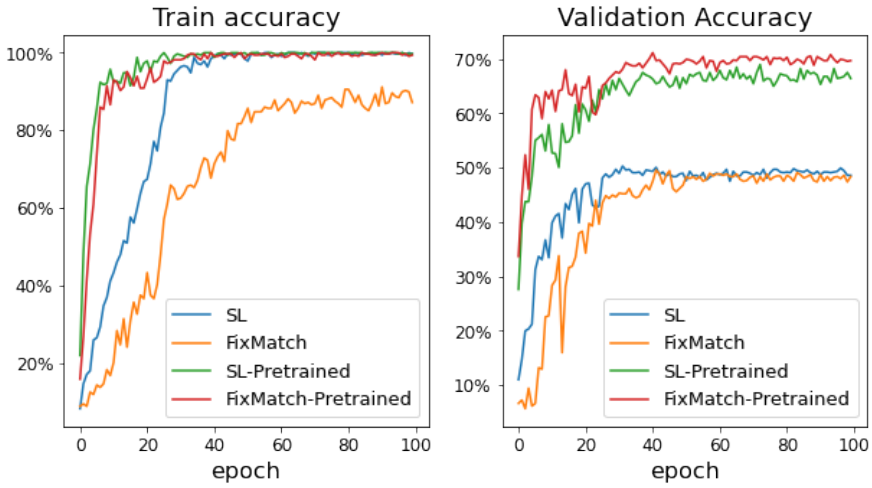
The number of pseudo-labels generated and model accuracy are plotted in Figure 5.7. FixMatch-Image sees an increase of pseudo-labels generated after 25 epochs of training, rising to just above 400 in the later training epochs. FixMatch-Image-Pretrained manages to generate more than 1750 pseudo-labels in the later training epochs. Pseudo-labels accuracy converges in both configurations.

Table 5.5: Averaged F1-score of the model after 10 runs with different initialization seeds. All the models are trained for 100 epochs on 10% labeled dataset. During each training, the weights of the best model are saved and utilized in the calculation.

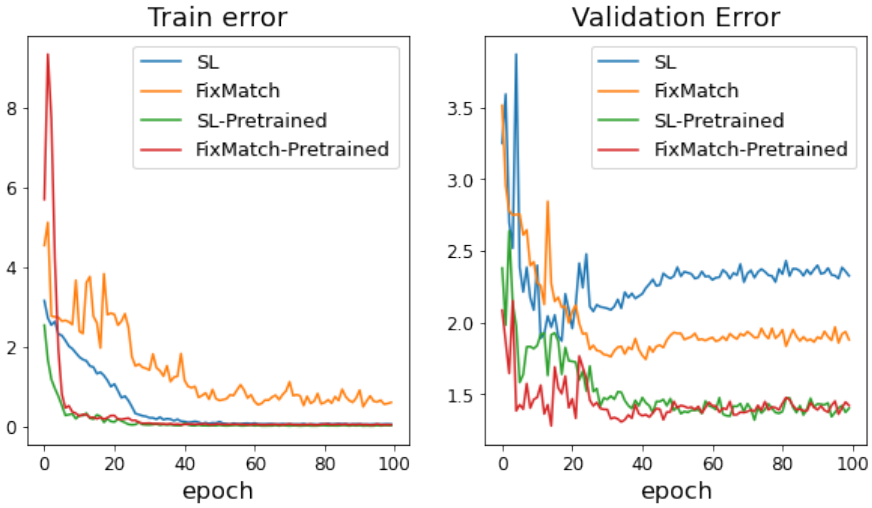
Model	Accuracy	P-Value
SL	**49.84%**	-
FixMatch-Image	49.11%	17.64%
SL-Pretrained	68.44%	-
FixMatch-Image-Pretrained	**71.22%**	1.24%

5.2.4 All Labels Comparison

The performance of FixMatch-Image and FixMatch-Image-Pretrained is compared to training with Supervised Learning (SL) on a labeled set corresponding to the sum of labels and pseudo-labels used by FixMatch in the later epochs of training. As shown in Figure 5.7, FixMatch-Image manages to produce around 400 pseudo-labels when trained for 100 epochs, while FixMatch-Image-Pretrained produces around 1750 pseudo-labels. Adding these amounts of pseudo-labels to the 328 labeled examples brings the total ratio of labeled examples to 22% and 63%, respectively.

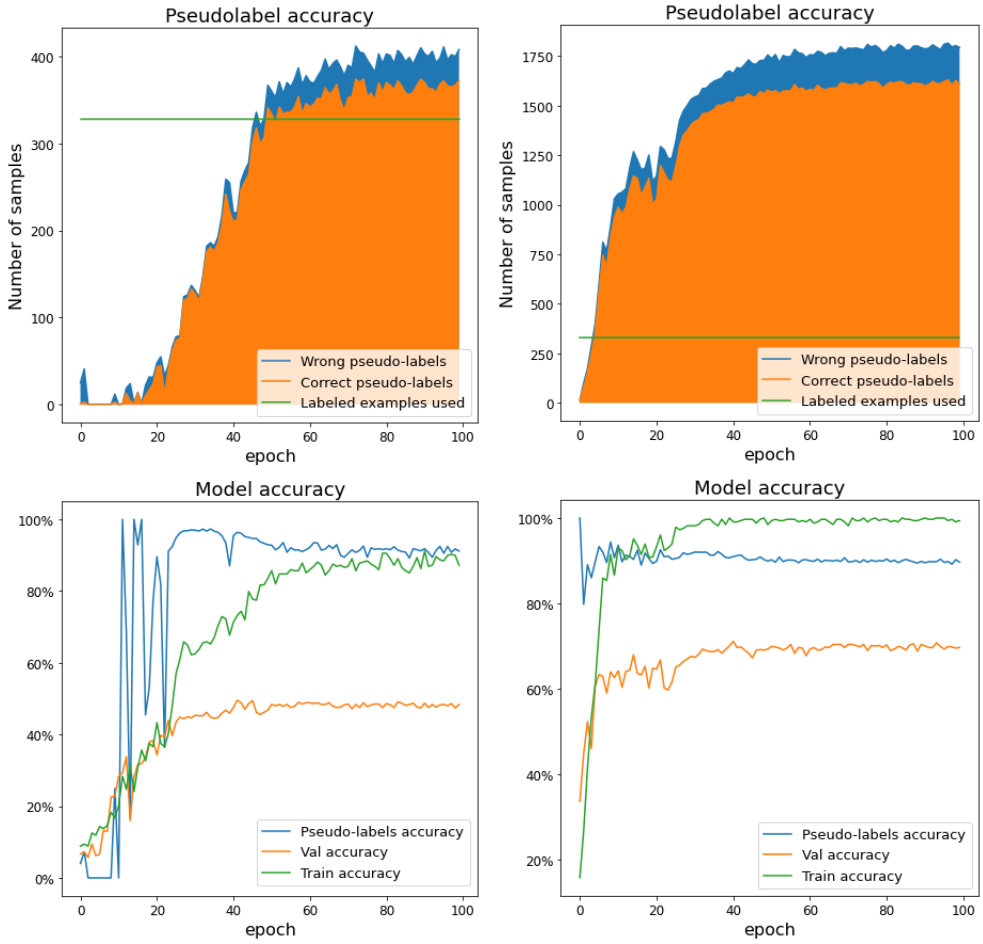


(a) Models train and validation accuracy during training.



(b) Models train and validation error during training.

Figure 5.6: Long training. Performance comparison of FixMatch-Image against Supervised Learning with and without the use of pretrained weights.Each model is trained for 100 epochs on a train set composed of 10% labeled examples. All but the FixMatch model manage to learn the training set, with SL showing signs of overfitting after around the 30th epoch.



(a) FixMatch-Image **(b)** FixMatch-Image-Pretrained

Figure 5.7: Long training. Pseudo-labels produced during training with FixMatch-Image. Figure A shows plots of FixMatch-Image while figure B shows plots of FixMatch-Image using pretrained weights. In each figure, the upper graph shows the amount of wrong and correct pseudo-labels produced. Train and validation accuracy are plotted in the lower graphs along with the accuracy of the pseudo-labels used for training.

The results presented in Table 5.6 show that the pseudo-labels generated by FixMatch-Image-Pretrained do increase the accuracy of the model, with it going from 68.44% to 71.22%, but their impact on the training is not equivalent to including the same amount of labeled examples in SL-Pretrained, the latter reaching an accuracy of 84.36% and using 68% of labels. The SL-Pretrained model accuracy converges to 84.66% when using the entire labeled dataset, making it the theoretical maximum model accuracy for ResNet18 using pretrained weights and training for 100 epochs with this choice of hyperparameters. Figure 5.8b shows validation F1-score plots for all the models. FixMatch-Image-Pretrained validation F1-score is plotted just above SL-Pretrained, leaving room for possible future improvements.

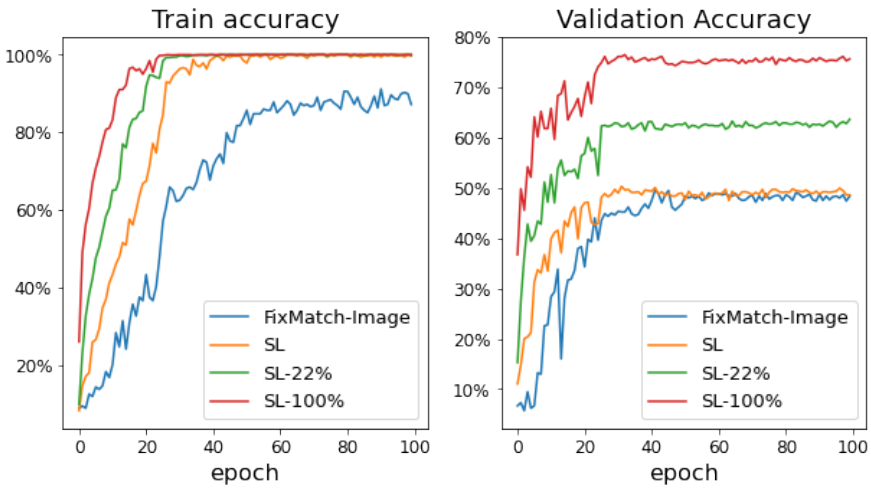
Figure 5.8a shows that every SL model reaches an F1-score of around 100% on the train set except for FixMatch-Image. Validation accuracy of FixMatch-Image is inferior to SL with the chosen hyperparameters, when it is supposed to lay between SL and SL-22%, similarly to the pretrained counterpart shown in Figure 5.8b. The theoretical maximum of FixMatch-Image is 76.12%, achieved in this case as well by using 100% of the dataset as labels.

Table 5.6: Averaged F1-score of the models after ten runs with different initialization seeds. Fixmatch-Image performance on 10% of labels is compared against SL when 10%, 22% and 100% of the labels are included in the training phase. Fixmatch-Image-Pretrained performance on 10% of labels is compared against SL-Pretrained when 10%, 63% and 100% of the labels are included in the training phase. Performance of FixMatch-Image is inferior to all the SL trained models, while performance of FixMatch-Image-Pretrained is inbetween SL-Pretrained with 10% labels and SL-Pretrained with 63% labels.

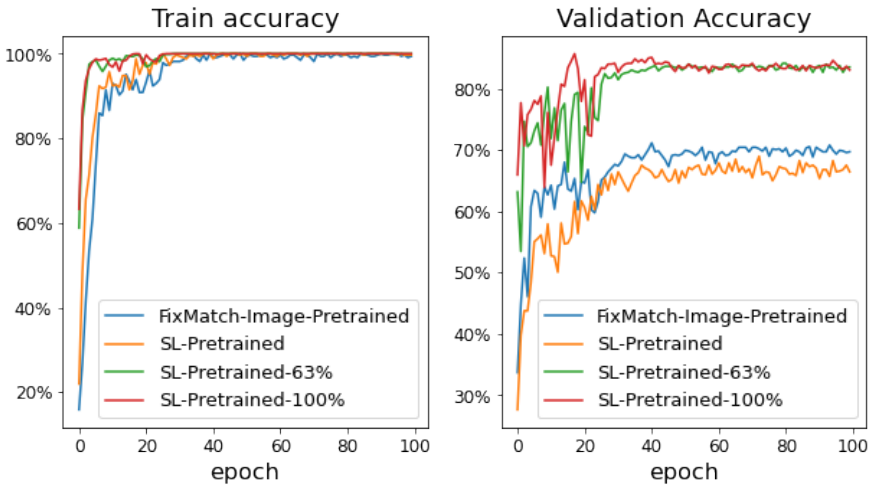
Model	Labels used	Accuracy
FixMatch-Image	10%	49.12%
SL	10%	49.84%
	22%	57.07%
	100%	**76.12%**
FixMatch-Image-Pretrained	10%	71.22%
SL-Pretrained	10%	68.44%
	63%	84.36%
	100%	**84.66%**

5.2.5 Variations to FixMatch

Three variations of FixMatch-Image are benchmarked in this experiment. In FixMatch-Image-uLoss, the unlabeled loss is multiplied by a scalar weight that increases linearly for each passing epoch by 1%. In FixMatch-Image-Decay, the confidence threshold decays in a linear manner from 98% to 92% by the end of 100 epochs. The same variation are tested on FixMatch-Image-Pretrained.



(a) Methods comparison with randomly initialized weights



(b) Methods comparison with pretrained weights

Figure 5.8: All labels comparison. Train and Validation F1-score comparisons of FixMatch-Image against SL when including 10%, 22% and 100% of the labels during training are shown in figure A, while F1-score comparisons of FixMatch-Image-Pretrained against SL with 10%, 63% and 100% of the labels are plotted in figure B.

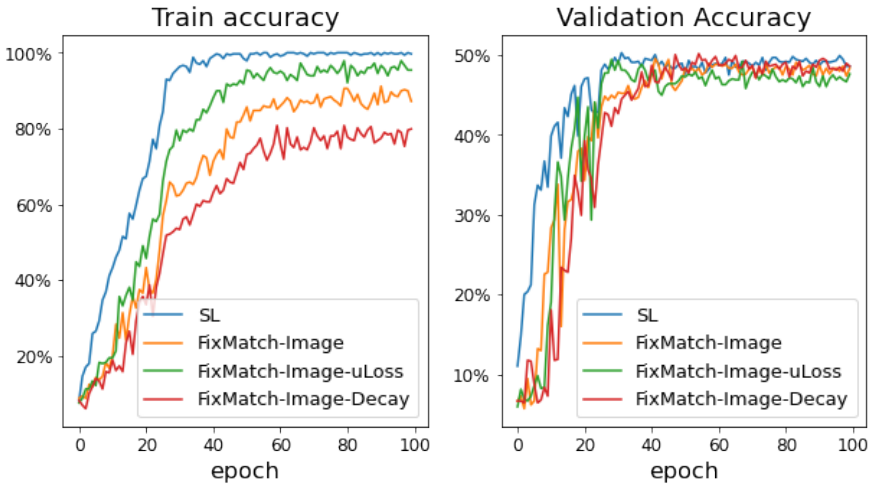
Results in Table 5.7 show that the two new variations of FixMatch-Image perform better on the test set than the SL models. When averaging the ten runs of each model, FixMatch-Image-uLoss and FixMatch-Image-Decay outperform the FixMatch-Image on the test set, proved by the increased number of pseudo-labels produced, shown in Figure 5.10. The FixMatch-Image-uLoss model produces more incorrect pseudo-labels, but the lower weight of the unlabeled loss pushes the model to focus the training, in the beginning, more towards the labeled set, achieving a higher train accuracy, as seen in Figure 5.9a. On the contrary, FixMatch-Image-Decay pushes the training of the model towards the unlabeled part, as more pseudo-labels are produced, scoring lower accuracy on the train set. Nevertheless, a P-Value study indicates that improvement of FixMatch-Image-uLoss and FixMatch-Image-Decay over SL is not statistically relevant and can be due to randomness.

The study regarding the pretrained models pictures a different situation. Each FixMatch-Pretrained configuration achieves a higher F1-score than SL-Pretrained, as shown in Table 5.7, and with lower P-Values than 5%, those results are to be deemed statistically relevant. All the pretrained models manage to learn the train set, as seen in Figure 5.9b. FixMatch-Image-Pretrained-uLoss scores a lower accuracy than regular FixMatch-Image-Pretrained; this is proved by the model's loss function weighting the unlabeled loss less, even though the model is able to generate many high quality pseudo-labels from the beginning of the training. The lower accuracy reached by the models translates to a lower number of pseudo-labels generated, which also contributes to a lower accuracy, as seen in Figure 5.11a.

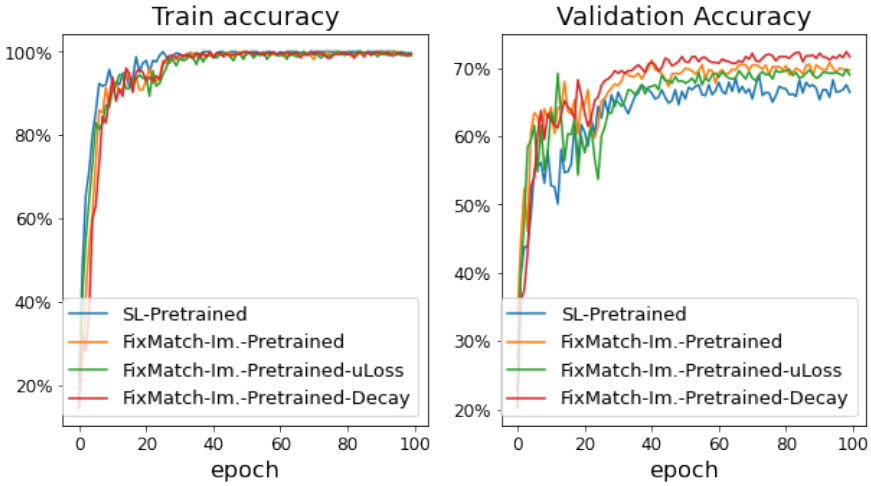
FixMatch-Image-Pretrained-Decay performs better than all of the other trained models, with an average F1-score of 71.95%. The model shows stability when the confidence threshold is lowered over time: in the beginning of the training, few pseudo-labels of high-quality are produced, thus contributing to reaching a high accuracy. As the confidence threshold is lowered, more pseudo-labels are included in the training, as seen in Figure 5.11b, but since the model performance is already high, the new examples can be trusted to be included in the training set without risking the deterioration of the system.

5.2.6 Various Unlabeled Ratios

The results presented in Table 5.8 show the impact of using the FixMatch algorithm when the ratio of labeled examples varies between 5%, 10% and 20%. Results show that in each different ratio configuration, FixMatch-Image fails to produce a higher validation F1-score than SL. All the calculated P-Values, retrieved from analyzing the results distributions, are higher than 5%, thus consistent with the previous experiments. FixMatch-Image-Pretrained, instead, manages to reach a higher performance than SL-Pretrained on all the different ratio configurations. It is worth noticing that the performance increase seen using FixMatch varies with the different labels' ratios, with an F1-score increase of 9.07%, 2.78% and 5.53% when using 5%, 10% and 20% of the labels, respectively. Those results show that

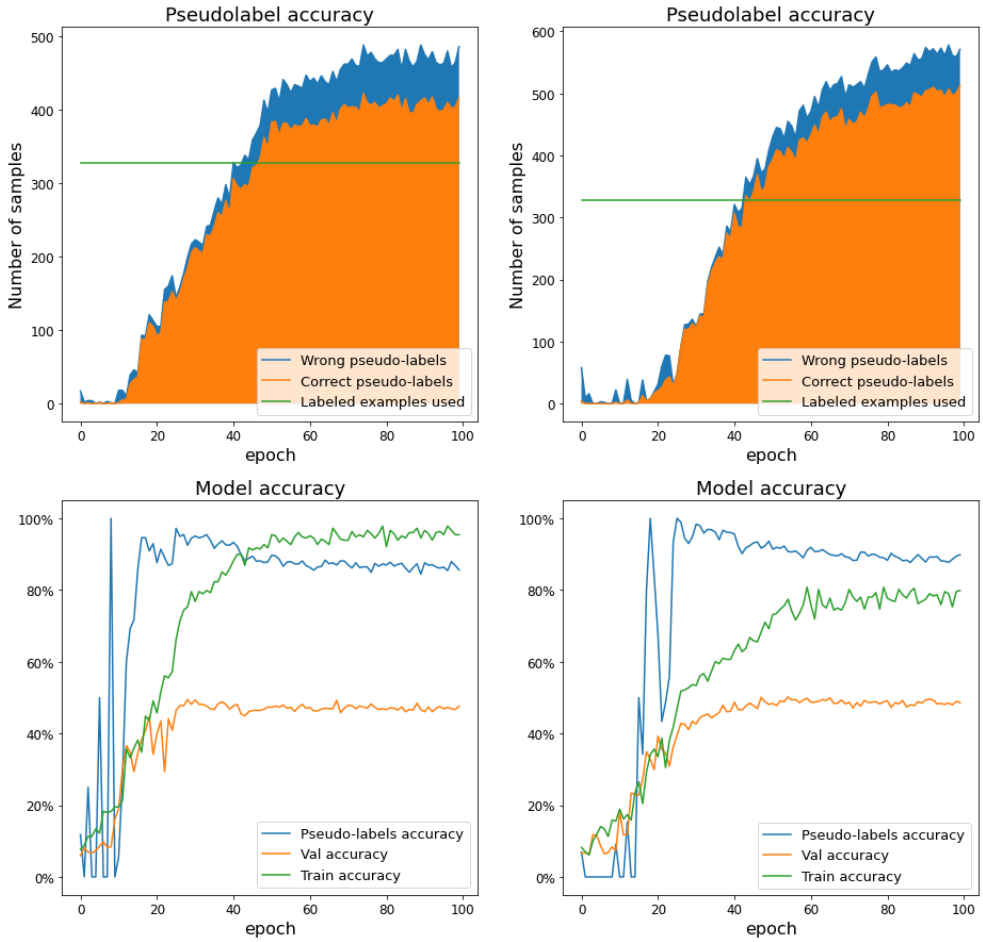


(a) FixMatch-Image Variations against SL



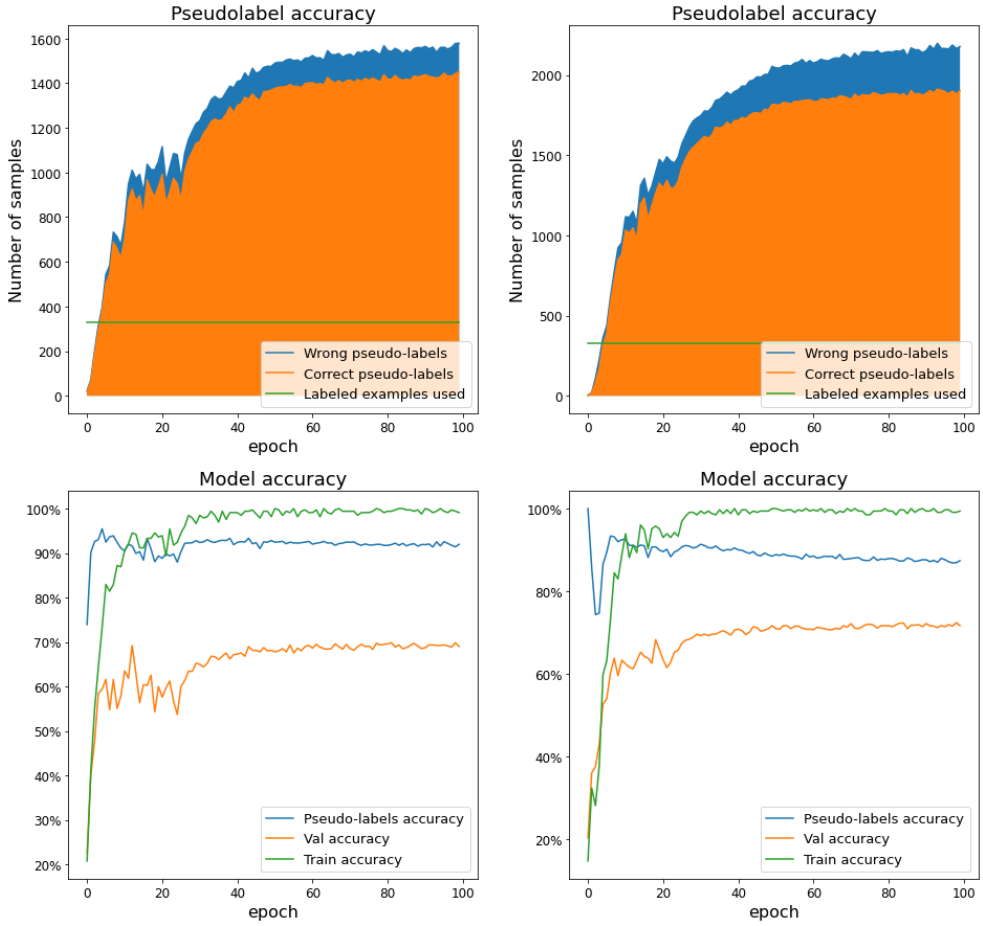
(b) FixMatch-Image-Pretrained Variations against SL

Figure 5.9: Variations to FixMatch. Training and validation accuracy performance of three FixMatch-Image variations against SL. The plots show train and validation F1-score for each FixMatch-Image variation when using randomly initialized weights (a) and with pretrained weights (b).



(a) FixMatch-Image-uLoss **(b)** FixMatch-Image-Decay

Figure 5.10: Variations to FixMatch. Pseudo-labels produced during training using FixMatch-Image with linearly increasing unlabeled loss weight (a), and with a continuous decay of the confidence threshold (b). In each figure, the upper graph shows the amount of wrong and correct pseudo-labels produced. Train and validation accuracy are plotted in the lower graphs along with the accuracy of the pseudo-labels used for training.



(a) FixMatch-Image-Pretrained-uLoss **(b)** FixMatch-Image-Pretrained-Decay

Figure 5.11: Variations to FixMatch. Pseudo-labels produced during training using FixMatch-Image-Pretrained with linearly increasing unlabeled loss weight (a), and with a continuous decay of the confidence threshold (b). In each figure, the upper graph shows the amount of wrong and correct pseudo-labels produced. Train and validation accuracy are plotted in the lower graphs along with the accuracy of the pseudo-labels used for training.

Table 5.7: Averaged F1-score after ten runs with different initialization seeds. Every model is trained for 100 epochs on 10% of the labels. The P-Value study indicates that the results regarding the FixMatch-Image variations have a high chance of being such due to randomness. Decaying the confidence threshold of each epoch proves to be a better strategy for training using FixMatch on a pre-trained model.

Model	Validation	P-Value
SL	49.84%	-
FixMatch-Image	49.11%	17.64%
FixMatch-Image-uLoss	**50.16%**	77.67%
FixMatch-Image-Decay	49.89%	95.28%
SL-Pretrained	68.44%	-
FixMatch-Image-Pretrained	71.22%	1.24%
FixMatch-Image-Pretrained-uLoss	69.65%	4.81%
FixMatch-Image-Pretrained-Decay	**71.95%**	0.44%

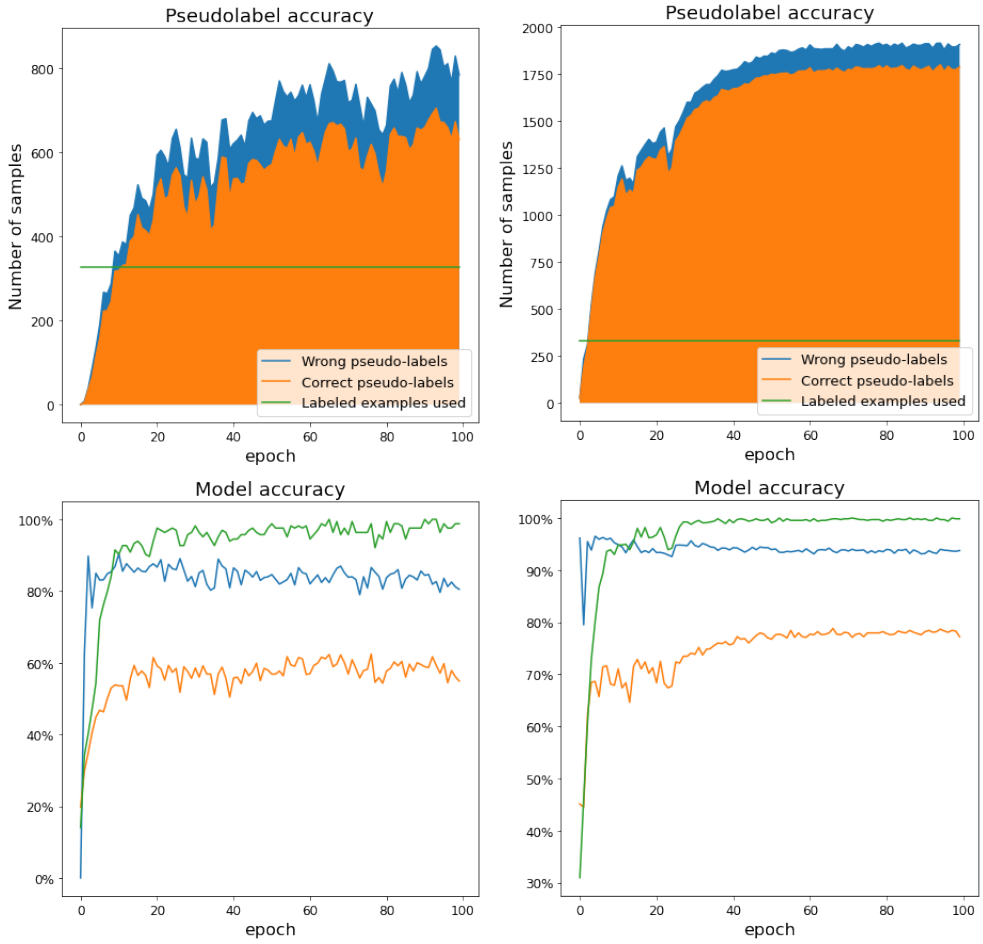
there is much to gain by using FixMatch on a small labeled dataset when a high number of unlabeled examples are available. At the same time, the performance boost given by FixMatch is not constant, but rather increases with the model's accuracy, as it produces more pseudo-labels with a higher accuracy: in Figure 5.12b it is evident that FixMatch-Image-Pretrained produces more pseudo-labels with 20% labels than with 10%, as seen in Figure 5.7b.

Table 5.8

Labels used	Model	Accuracy	P-Value
5%	SL	**44.17%**	-
	FixMatch-Image	36.34%	4.38%
	SL-Pretrained	53.27%	-
	FixMatch-Image-Pretrained	**62.34%**	0.03%
10%	SL	**49.84%**	-
	FixMatch-Image	49.11%	17.64%
	SL-Pretrained	68.44%	-
	FixMatch-Image-Pretrained	**71.22%**	1.24%
20%	SL	**59.59%**	-
	FixMatch-Image	59.30%	74.45%
	SL-Pretrained	73.43%	-
	FixMatch-Image-Pretrained	**78.96%**	0.003%

5.2.7 Unknown Classes in the Unlabeled Set

In this experiment, 50% of the unlabeled set is exchanged with high quality examples from the other 382 classes available in the original dataset, as seen in



(a) FixMatch-Image-Pretrained 5% Labels **(b)** FixMatch-Image-Pretrained 20% Labels

Figure 5.12: Various unlabeled ratios. Pseudo-labels produced during training using FixMatch-Image-Pretrained with 5% labels (a), and with 20% labels (b). In each figure, the upper graph shows the amount of wrong and correct pseudo-labels produced. Train and validation accuracy are plotted in the lower graphs along with the accuracy of the pseudo-labels used for training.

section 4.1, meaning that the dataset used for training contains 10% labels of the chosen 15 classes, 45% unlabeled examples of the 15 chosen classes and 45% unlabeled examples from unknown classes. The premise behind this experiment is to simulate an unlabeled set retrieved from soundscape recordings, where the unlabeled examples affiliation to the classes the model tries to classify cannot be guaranteed.

The findings presented in Table 5.9 show that including unknown unlabeled examples results in a small degradation of the system, when compared to using the unlabeled set composed exclusively of examples from the 15 classes. FixMatch-Image-Pretrained-External50% reaches a validation F1-score of 70.04%, lower than FixMatch-Image-Pretrained but still higher than SL-Pretrained. Figure 5.13b shows that FixMatch-Image-Pretrained-External50% produces less pseudo-labels than FixMatch-Image (showed in Figure 5.7b), but the pseudo-labels generated are from a majority of unlabeled examples originally taken from the chosen 15 classes, since the pseudo-labels accuracy lays around 60%. The rest are either wrongly labeled examples or examples that do not fall into any of the 15 classes.

Table 5.9: Averaged F1-score of the model after ten runs with different initialization seeds. All models are trained for 100 epochs and show convergence. Decaying the confidence threshold by 0.0006 per epoch proves to worsen the performance on the model.

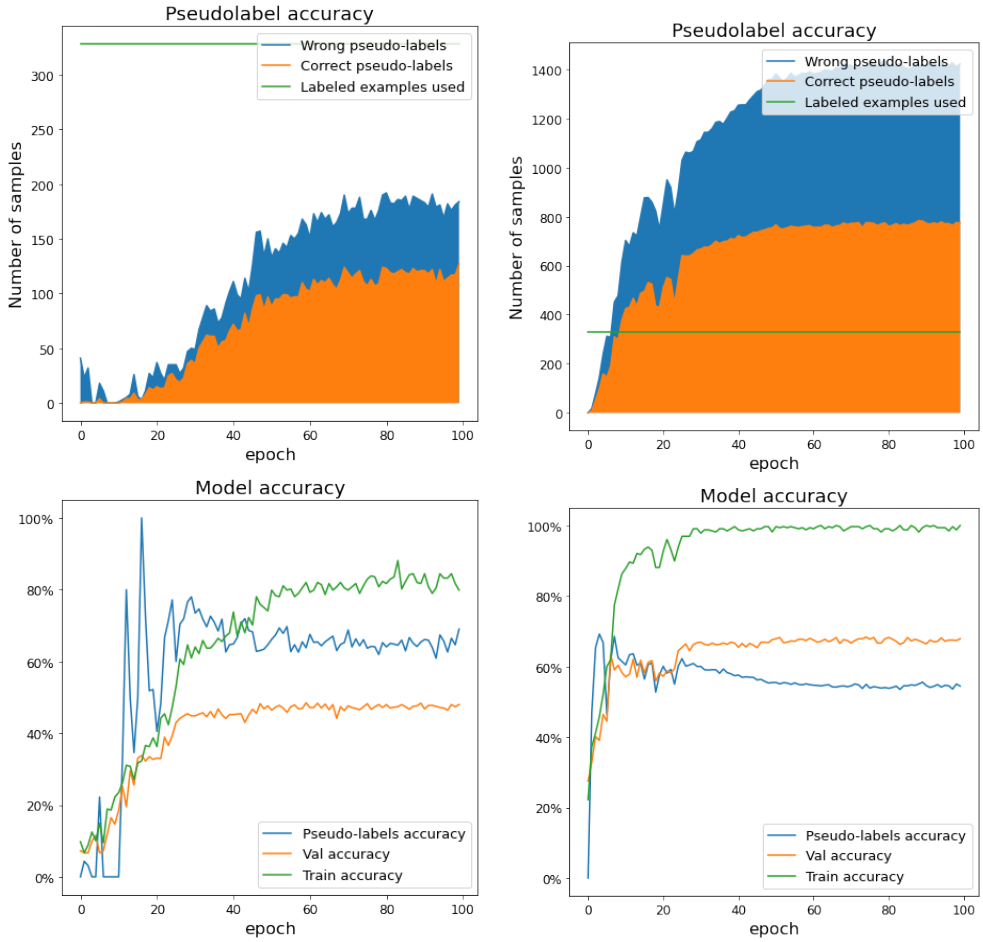
Model	Accuracy	P-value
SL	**49.84%**	-
FixMatch-Image	49.11%	17.64%
FixMatch-Image-External50%	49.26%	46.42%
SL	68.44%	-
FixMatch-Image-Pretrained	**71.22%**	1.24%
FixMatch-Image-Pretrained-External50%	70.04%	4.06%

5.2.8 Other Choice of Augmentation Techniques

In this experiment, FixMatch is tested in pairing with a less weak/strong augmentation technique, chosen out of the results of the study conducted in subsection 5.2.1. The augmentation techniques previously utilized in FixMatch-Image and FixMatch-Image-Pretrained are shown in Table 5.10, along with the techniques used in the "AlternativeAug" variants.

Results, presented in Table 5.11, show an increment in the F1-score when the alternative augmentation techniques are utilized. FixMatch-Image-AlterativeAug perform better than FixMatch-Image, reaching the same performance of SL. The P-Value calculated is 100%, meaning that this result is totally due to randomness and there is absolutely no difference in utilizing FixMatch or SL when training with the chosen parameters on a non-pretrained ResNet18 model.

Regarding pretrained models, there is a performance boost, with FixMatch-



(a) FixMatch-Image-External50% **(b)** FixMatch-Im.-Pretrained-External50%

Figure 5.13: Unknown classes in unlabeled set. Pseudo-labels produced during training FixMatch-Image with 50% of the unlabeled set consisting of examples from external classes, on a randomly initialized model (a), and pretrained model (b).

Image-Pretrained-AlternativeAug reaching an average F1-score of 73.60%. The calculated P-Value is 0.02%, meaning that this result has high statistical relevance.

Table 5.10: Weak and strong augmentation chosen for each model. A less weak-/strong augmentation is chosen for each model, based on results in Figure 5.2.

Model	Weak	Strong
FixMatch-Image	Shear Y	Random Brightness
FixMatch-Image-Pretrained	Shear Y	Random Brightness
FixMatch-Image-AlternativeAug	Affine-Translate X	Random Contrast
FixMatch-Image-Pretrained-AlternativeAug	Random Contrast	Affine-Translate Y

Table 5.11: Averaged F1-score of the models after ten runs with different initialization seeds. Each model is trained for 100 epochs on 10% labels.

Model	Accuracy	P-value
SL	**49.84%**	-
FixMatch-Image	49.11%	17.64%
FixMatch-Image-AlternativeAug	**49.84%**	100.0%
SL	68.44%	-
FixMatch-Image-Pretrained	71.22%	1.24%
FixMatch-Image-Pretrained-AlternativeAug	**73.60%**	0.02%

5.2.9 Best Models

The final models are trained by collecting the findings in the previous experiments. Since the performance of FixMatch-Image is not satisfactory, with every experiment leading to non statistically relevant results, only the pretrained variant are trained.

FixMatch models trained using the alternative augmentation techniques (less strong and weak), did perform better and became therefore consolidated as the new augmentation choice. The unlabeled set provided to FixMatch contains only 50% of relevant classes, as assuring the purity of an automatically recorded unlabeled set is highly improbable in a realistic scenario, so this design choice will therefore increase the credibility of the results. Confidence threshold decay is tested against a fixed threshold, as it is unsure how the combination of the former with a 50% unlabeled set performs.

The results, available in Table 5.12, shown that the FixMatch trained models performed better on a dataset with 5%, 10% and 20% labeled examples, respectively.

Lowering the confidence threshold during training has shown to be a good strategy, as it includes more examples from the unlabeled set on the right time, without disturbing the stability of the system. The combination of 50% external unlabeled data and decay of the confidence threshold does not produce the same

results as when each component is tested by itself: as the threshold is lowered, the model includes more mislabeled unlabeled examples into training, worsening the performance. In Figure 5.14, it can be observed that linearly lowering the confidence threshold results almost entirely in the inclusion of incorrect pseudo-labels, while the amount of correct pseudo-labels remains almost unaltered.

Table 5.12: Averaged results of ten runs with different initialization seeds. All models are trained for 100 epochs and the FixMatch models are trained both with a fixed confidence threshold decay and with a decay of 0.0006 at each epoch, starting from 98%. FixMatch is paired with the alternative augmentation, and 50% of the unlabeled set has been swapped with examples from unknown classes.

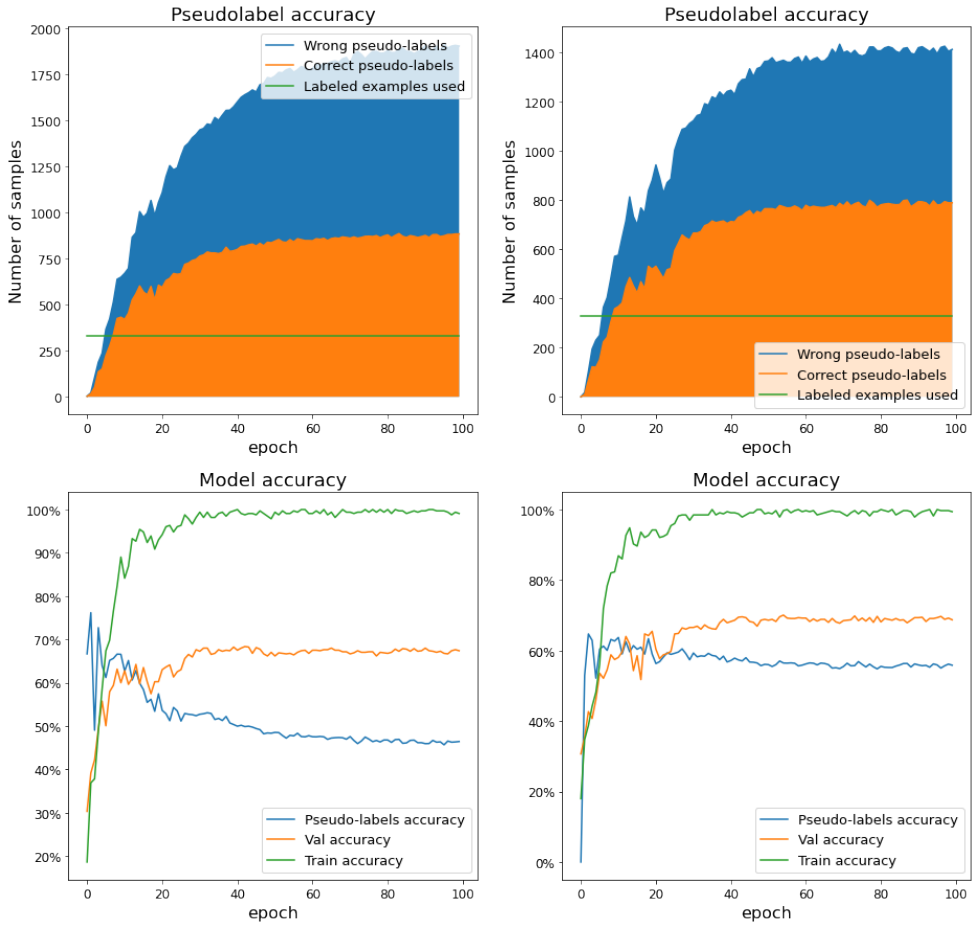
Labels used	Model	Conf. Decay	Accuracy	P-Value
5%	SL-Pretrained-Final	-	53.27%	-
	FixMatch-Image-Pretrained-Final	No	**60.80%**	0.11%
	FixMatch-Image-Pretrained-Final	Yes	59.52%	0.25%
10%	SL-Pretrained-Final	-	68.44%	-
	FixMatch-Image-Pretrained-Final	No	**70.06%**	3.96%
	FixMatch-Image-Pretrained-Final	Yes	68.85%	9.28%
20%	SL-Pretrained-Final	-	73.43%	-
	FixMatch-Image-Pretrained-Final	No	**78.11%**	0.03%
	FixMatch-Image-Pretrained-Final	Yes	77.38%	0.03%

5.3 Results Analysis

5.3.1 Use of Pretrained Weights

A common denominator in every experiment is the failure of training a model without the use of pretrained weights. It has become clear that this failure is due to some other factor that has been left unchanged in all the experiments performed. In Figure 5.15 the validation F1-score of the best strategy for both non-pretrained models and pretrained models when trained on a dataset with 5%, 10% and 20% of labeled examples is plotted. When pretrained weights are not loaded in the model, the performance of FixMatch is at most as good as SL. On the other hand, the pretrained version of the model always performed better with FixMatch, even when the unlabeled set contained many "impure" examples from other 382 bird classes.

There is no difference between the pretrained model and the randomly initialized model other than the pretrained weights computed on the ImageNet dataset and the Learning Rate (lr), the latter being selected as result of the hyperparameter search in subsection 4.3.5. Training an SL model using the same lr used in training the pretrained models ($lr = 0.00032$), resulted only in a worsening of the performance, as seen in Table 5.13.



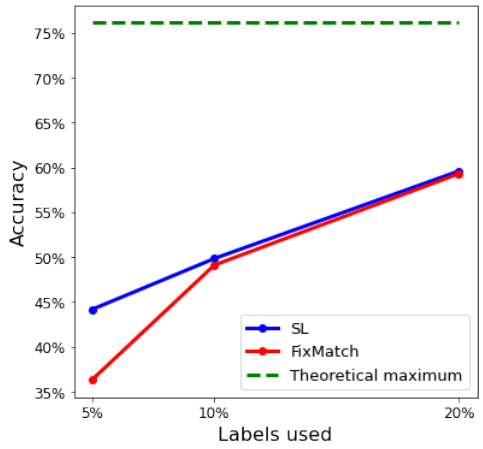
(a) FixMatch-Image-Pretrained-Final **with** Confidence Threshold Decay

(b) FixMatch-Image-Pretrained-Final **without** Confidence Threshold Decay

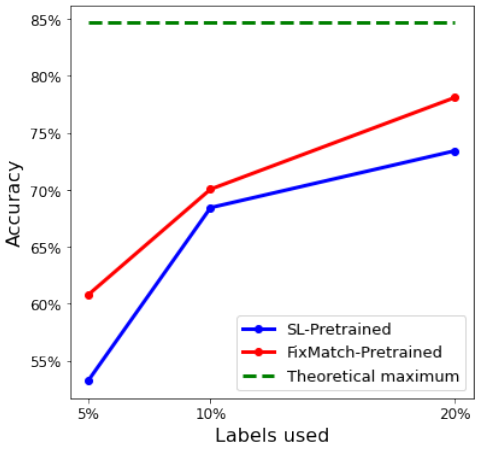
Figure 5.14: Best models. Pseudo-labels produced during training using the best FixMatch-Image-Pretrained configuration with (a) and without (b) continuous decay of the confidence threshold. When decaying the confidence threshold, a higher amount of examples from external classes or wrongly labeled are introduced, reducing the pseudo-label accuracy. The models are trained on 10% of labels.

Table 5.13: Averaged F1-score of ten runs with different initialization seeds. All models are trained for 100 epochs on 10% of the labeled set.

Model	Learning Rate	Accuracy
SL	0.00153	49.84%
SL	0.00032	44.77%



(a) Non-pretrained models trained on 100% of the unlabeled set composed of examples of the 15 known classes and no confidence threshold decay.



(b) Pretrained models trained on 50% of the unlabeled set composed of examples of other 382 classes, alternative augmentation techniques, and no confidence threshold decay.

Figure 5.15: average validation F1-score of the models configurations plotted against the amount of labels used during training. The plots show F1-score of the best discovered training strategy, when pretrained weights are not loaded (a) and when they are loaded in the model (b). In each of the plots, the discovered theoretical maximum, i.e., training the same model on 100% of labels using SL, is plotted in green.

5.3.2 FixMatch Effect on Bird Sounds Dataset

Bird sounds classification has proved to be a non-trivial task to solve. Using only high quality recordings, a modern and robust network architecture, pretrained weights, and limiting the dataset to 15 classes, the highest validation F1-score reached is 84.66%, training on the entire dataset, composed of 219 examples per class.

When reducing the amount of labeled examples, the accuracy of the model reduces drastically. One of the reasons of this being that bird sounds differ highly intraspecies, with dialects, male/female differentiation and sounds produced during special events such as fights, mating calls or duets. For a model to learn feature representation for all those cases, the train set should include enough examples for each case, increasing the need of domain knowledge required.

Analyzing the dataset split shows that the validation set is balanced regarding the amount of training examples per class, but not regarding the metadata "type" included in each class. Figure 5.16 shows that the training dataset contains many fewer examples of metadata types, like "sex uncertain" and "life stage uncertain", than the validation dataset, even though they were split 80-20% from the original dataset, respectively. For the "gbwwre1" class, the validation set includes many examples without any metadata type, plotted in the rightmost column of Figure 5.16b, while the training set includes almost no such examples.

The 10 most used metadata types in a randomly picked 10% and 5% subset of the training dataset is plotted in Figure 5.17. A severe "thinning" of the number of examples per metadata type can be observed, with some having no representative examples for some of the classes. In these cases, using SSL methods like FixMatch, increases the performance by giving the model a chance of seeing more examples of each class.

When taken as example, an SL-pretrained model and a FixMatch-Image-Pretrained model, both trained on 5% labels and using the same initialization seed, the class "mallar3" gets correctly predicted 16 out of 55 times by the SL-Pretrained model, shown in the confusion matrix in Figure 5.18a: of these wrong predictions, the model outputs the class "amoreb" ten times. A closer look at the spectrograms of the "amerob" and "mallar3" classes, visible in Figure 5.19, shows some similarities between the two metadata types "alarm-call", while spectrograms representing the "call" of the two shows clearly distinctive features.

In Figure 5.16a, it can be observed that the training dataset includes 15 examples of "alarm-call" for "mallar3" and 19 of the "amerob" class. When randomly retrieving the 5% of the labels (same sub-set used when training both models, as the initialization seed is the same), only one "alarm-call" example of each of the two classes is included in the labeled training set, shown in Figure 5.17b, leaving the model confused about how to separate the two similar types.

Training with FixMatch, the model managed to include enough unlabeled examples of "mallar3" and "amerob", ultimately learning to separate the two classes. The model predicted the class "mallar3" correctly 41 out of 55 times, of which

none of the wrongly predicted class includes the class "amerob", as visible in the confusion matrix in Figure 5.18b.

In the case where no labeled "alarm-call" spectrogram of the "mallar3" class were included in the training set, the FixMatch training routine would have assigned all of the "mallar3" unlabeled examples, of the type "alarm-call", the pseudo-label "amerob", due to them being the closest labeled examples it has seen, propagating the error in the successive training epochs. Similarly, using FixMatch has yielded a minimal improvement for the class "carwre", going from 14 to 15 correct predictions out of 55. The FixMatch trained model began mismatching "carwre" for "eursta", even though it did not when training with SL (Figure 5.18). The model receives labeled examples of only two types of spectrograms for "carwre", while seeing six different types of "eursta" (Figure 5.17b), leading to the mislabeling of the unlabeled examples.

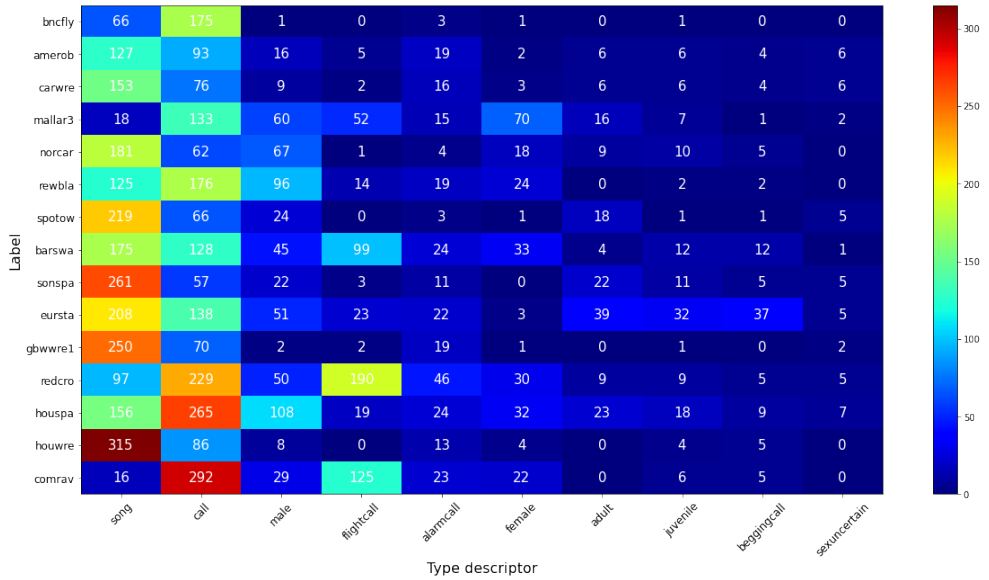
5.4 Chapter Summary

This chapter presented the experiments plan, experiments results with initial remarks, as well as the analysis of the results. The data augmentation experiment resulted in the classification of each audio and image augmentation technique into weak and strong. Opting to pair a less weak/strong augmentation technique to FixMatch yielded better results rather than the weakest/strongest one. Results showed that audio augmentation techniques yield comparable results to image augmentation, but costing a higher computational time.

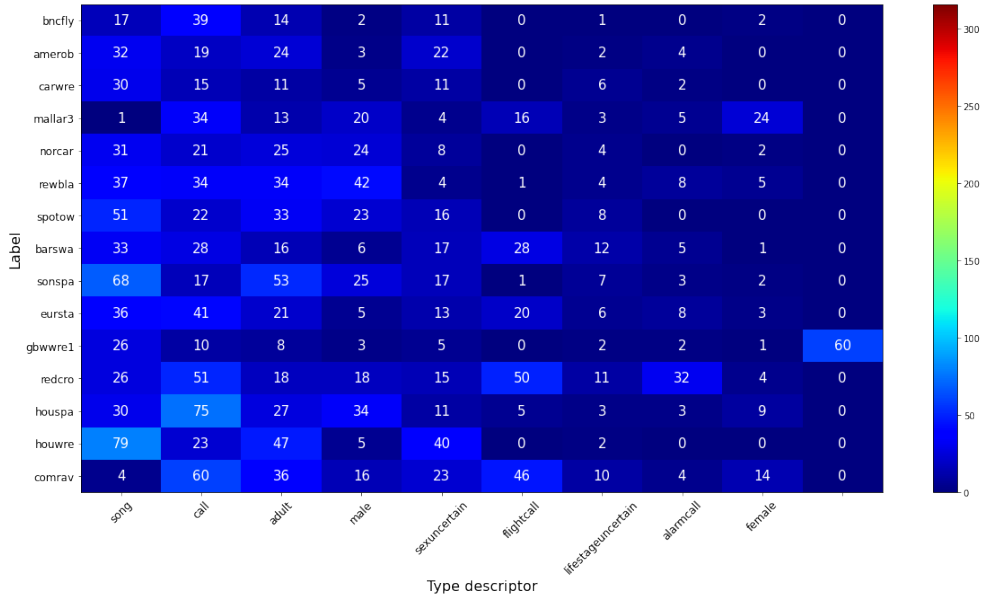
All the FixMatch models without the use of pretrained weights failed to surpass the F1-score achieved by the SL baseline. Additionally, all results regarding non-pretrained FixMatch models are to be deemed non statistically relevant, due to the resulting P-values being higher than 5%.

On the other hand, pretrained FixMatch scored always better than the pretrained SL baselines. Decaying the confidence threshold with time proved to be a good strategy, if the unlabeled set is guaranteed to exclusively include relevant examples. When 50% of the unlabeled spectrograms were exchanged with spectrograms of unknown classes, the performance of pretrained FixMatch dropped as the confidence threshold was lowered, due to more unknown examples being accepted as training examples.

The performance boost of FixMatch showed to be non-linear, with a higher increase when providing 5% and 20% of the labels, the former due to more examples being introduced in an otherwise too little labeled set, while the latter due to the increased model accuracy, reached by training on the bigger labeled set, led to more precise pseudo-labels being produced. The best FixMatch configuration reached F1-scores of 60.80%, 70.06% and 78.11%, whereas SL scored 53.27%, 68.44% and 73.43%; in both cases, pretrained weights were used, and training was performed on 5%, 10%, and 20% of the labels.

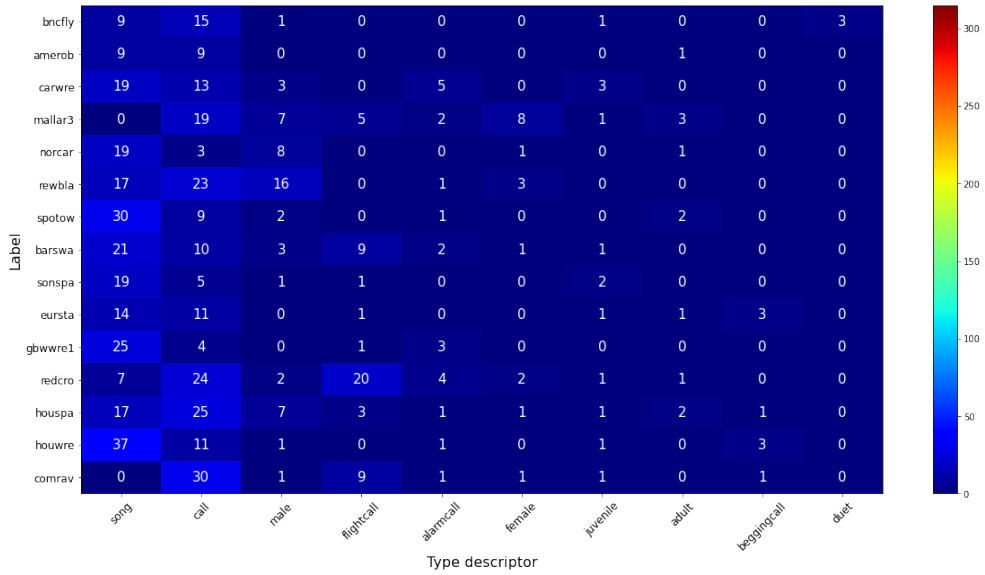


(a) Training set

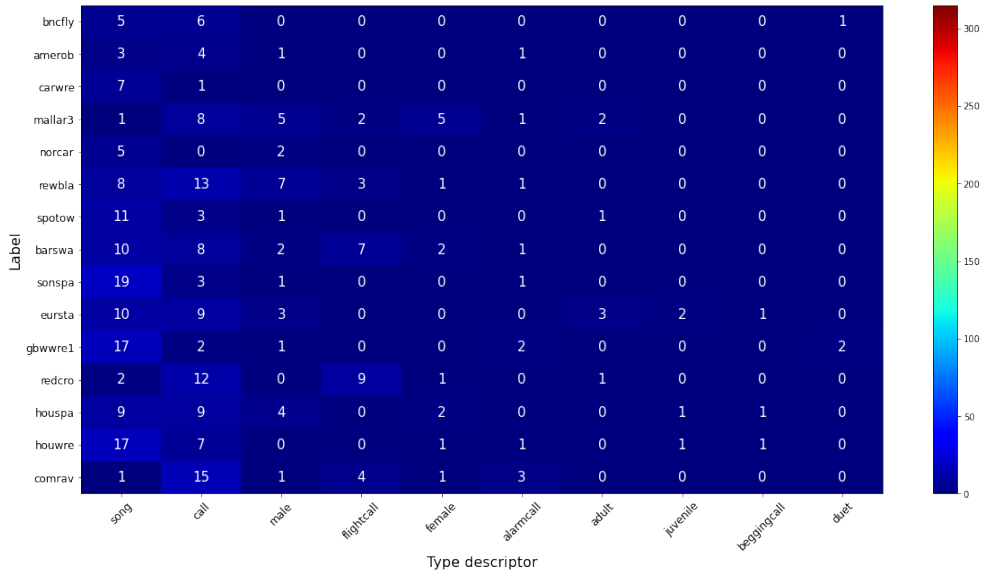


(b) Validation set

Figure 5.16: Occurrence of each metadata "type", distinct by labels, for the **entire Training and Validation set**, in (a) and (b), respectively. On the y-axis the labels of the examples is plotted, and on the x-axis lay the ten more common type descriptors available in the dataset. The "energies" show the amount of examples of that type for a specific label and are normalized between 0 and 315, the latter being the highest amount of examples of all the type/class combinations. It can be observed that the validation set is not representative of the training set, with some metadata "type" present in the Validation set being absent in the Training set.

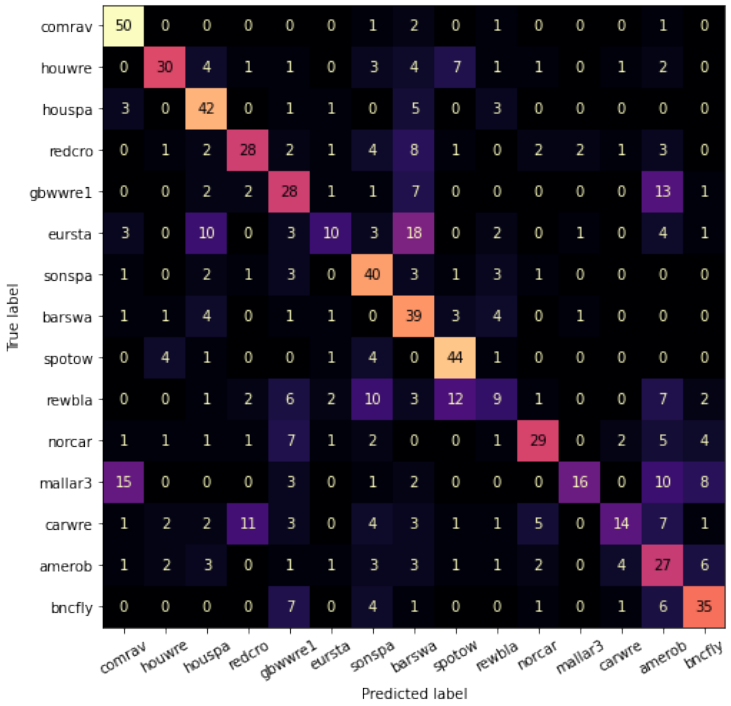


(a) Training set: 10% labels

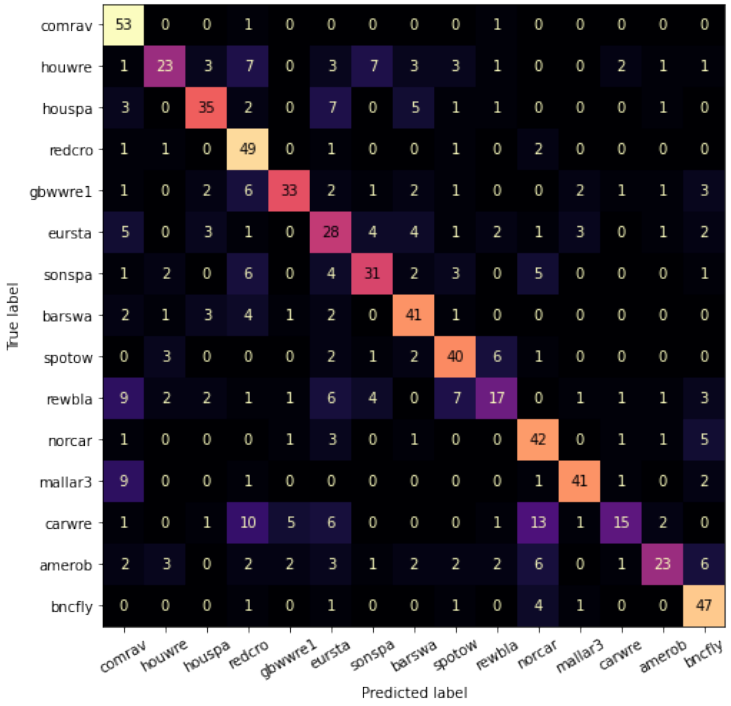


(b) Training set: 5% labels

Figure 5.17: Occurrence of each metadata "type", distinct by label, for **a random subset of 10% and 5% of the training set**, in (a) and (b), respectively. On the y-axis the labels of the examples is plotted, while, on the x-axis, the ten more common type descriptors available in the dataset are shown. The "energies" show the amount of examples of that type for a specific label and are normalized between 0 and 315, the latter being the highest amount of examples of all the type/class combination. A "thinning" of the metadata "type" energies can be observed here, with some "type" disappearing in the 5% subset.

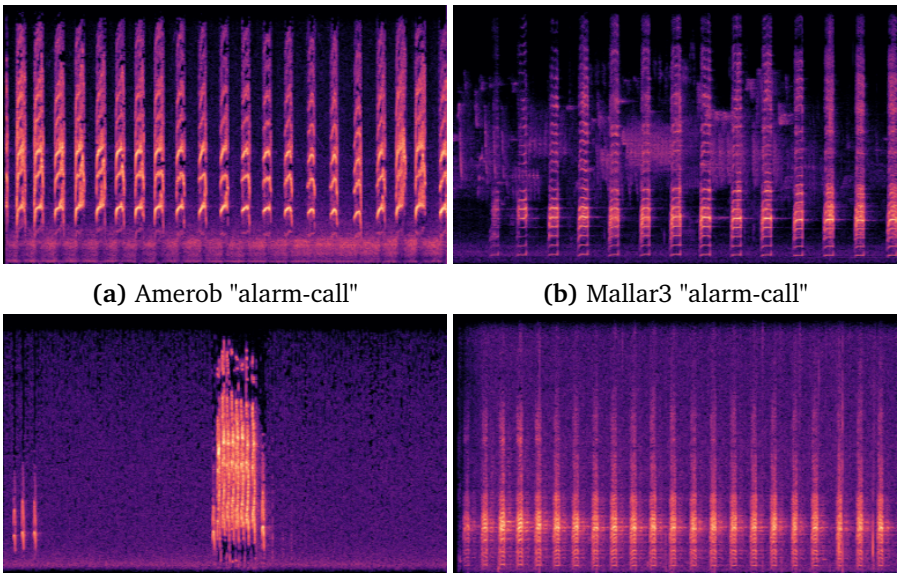


(a) Heatmap SL-Pretrained model with 5% labels



(b) Heatmap FixMatch-Pretrained model with 5% labels

Figure 5.18



(a) Amerob "alarm-call" **(b)** Mallar3 "alarm-call"

(c) Amerob "call" **(d)** Mallar3 "call"

Figure 5.19: Spectrograms examples from the Amerob and Mallar3 classes. A strong resemblance between the "alarm-call" of Amerob and Mallar3 can be observed here. On the other hand, those similarities do not apply when inspecting the "call" of the two species.

Chapter 6

Conclusion

This chapter concludes the work done in this thesis. The research questions and goal are discussed in section 6.1, and some ideas of possible future work are presented in section 6.2.

6.1 Discussing Around the Research Questions

To answer the research questions posed in this thesis, a balanced subset of 15 classes has been retrieved from the data available in the Xeno-canto collection. Then, a custom pipeline has been built in order to preprocess the examples with either audio or image augmentation.

Different FixMatch configurations have been tested against SL on different label-ratios, by loading pretrained weights or starting training from scratch.

Research question 1 *How do SSL methods applied in other domains perform on bird songs in comparison to a sensible baseline?*

FixMatch, an SSL method developed around image classification, has shown to be applicable on an audio classification task with the right adjustments. The choice of data augmentation to use with FixMatch has to be carefully picked out of a relevant selection of transformations, since not every image augmentation technique is applicable to an audio spectrogram. Audio augmentation is also compatible with FixMatch, with comparable performance to image augmentation, but requires more time to compute.

FixMatch has proved to be able to take advantage of pretrained weights retrieved on an image dataset, and fine-tune a model for an audio classification task. By not freezing any of the layers, the model manages to optimize the whole feature space, finding a better optimum.

The best FixMatch configuration reached F1-scores of 60.80%, 70.06% and 78.11%, whereas SL scored 53.27%, 68.44% and 73.43%; in both cases, pretrained weights were used, and training was performed on 5%, 10%, and 20% of the labels.

These findings are partially in line with the results presented in the work of Grollmisch et al. [14], where it was found that FixMatch always improved the accuracy compared to SL, and was outperformed by SL-pretrained only on the most difficult dataset. Results from the experiments in this thesis showed that only by loading pretrained weights FixMatch managed to outperform the SL baselines. This is due to the more complex nature of the dataset used in this thesis.

Research question 2 *To which extent can the labeled-to-unlabeled examples ratio be reduced before degradation of the system?*

Reducing the amount of labels used to 5%, 11 examples per class, the best FixMatch configuration, using pretrained weights, reaches an average accuracy of 60.80%, a 7.53% increase over using SL with the same amount of labels. This thesis has shown that FixMatch is still able to generate enough good quality pseudo-labels from an impure unlabeled set using a limited amount of labels, leading the model to find a better optimum. The performance worsened when some of the bird song "types" were neglected in the labeled set, as the model mislabeled more unlabeled examples and increased the confirmation bias.

Goal *Reducing the labeled data needed to reach baseline performance on classification of bird sounds.*

Training a ResNet18 model using FixMatch yielded better results than using supervised learning on all the labeled to unlabeled ratios tested. Nevertheless, none of the models managed to reach the same performance of using the entire labeled dataset during SL training. The results showed that including pseudo-labels during FixMatch training does not yield the same F1-score as if they were the real labels, leading to believe that a new strategy for selection and optimization of the strong augmentation technique ought to be developed.

A non-pure unlabeled set is a reasonable scenario for SSL within audio classification, when the unlabeled dataset is a collection of audio retrieved from a not human-supervised recorder unit. Lowering the confidence threshold level as the model trains proves to be a good strategy when the unlabeled set exclusively includes examples of the selected classes, but the performance drops when half of the unlabeled set is composed of unknown examples. Therefore, a better strategy in this case is to keep a fixed high threshold, such that only pseudo-labels of high enough confidence are included. This general approach can be applied to other audio domain classification tasks, but also to an image classification task, where the unlabeled set is retrieved using an automatic pipeline, like a camera and motion sensor setup.

6.2 Future Work

The results presented in this thesis are promising and represent a step further into building good classifiers using less domain knowledge. Nevertheless, some

improvements to the system can be made, and new areas can be explored. Some are presented in this section.

6.2.1 Increasing the Dataset Balance

The pseudo-labels accuracy can be increased by balancing the supervised set both class-wise and type-wise, and including as many different "type" spectrograms for each class as possible. For a more accurate validation of the system, in the case where the validation set is not balanced type-wise, cross validation can be implemented instead of a fixed split of the dataset. NLP methods for analyzing, clustering and cleaning the manually-assigned "type" in the dataset could be explored.

6.2.2 Changes in the FixMatch Implementation

The implementation of FixMatch presented in this thesis is based on minibatch-learning, and each minibatch is divided into a labeled and an unlabeled part: in the case of 5% of the labels being used, each minibatch of size 40 contains only two labeled examples, with the rest being unlabeled. This implementation makes it difficult to try different ratios of labels and optimize the minibatch-size parameter.

Alternatively, at the beginning of each training epoch, the model can generate the hard pseudo-labels of all of the unlabeled examples predicted with a higher confidence than τ. Then, the weakly augmented supervised examples and the strongly augmented unlabeled examples, of higher confidence than τ, can be combined into the same batch. For each step of the training epoch, a random minibatch can be retrieved from this new batch and continue with regular minibatch training. The results of this thesis showed that weighting the labeled and unlabeled loss the same yielded best results, therefore there is no need to separate the examples when calculating the total loss. The minibatch-size parameter can now be chosen indifferently from the labels ratio.

6.2.3 Accuracy Improvements

The model can produce more accurate pseudo-labels by including positional data, as it is part of the metadata of the utilized dataset. Positional data is easy to retrieve for an unlabeled recording, as it corresponds to the longitude and latitude position of the recording unit utilized. By including this information during training, the model should be able to distinguish similar spectrograms of different species that do not appear in the same geographic location. The same applies to the time of the recording, as some of the species are, for instance, nocturnal or produce different types of sounds at specific times during the day.

Allowing the model to experience multiple weak and strong augmentation technique pairs should also be explored, as it shows more scenarios of possible sounds.

6.2.4 Testing on Real Landscape Recordings

A natural next step is to test the efficiency of FixMatch by training using unlabeled landscape recordings. A new model capable of recognizing more species than the 15 mentioned in this thesis needs to be set up, in order to ensure a good representation of the classifiable species in the unlabeled landscape recordings.

For filtering exclusively bird sound spectrograms generated from the unlabeled soundscape recordings, a simpler classifier could be built to recognize the "bird" class, using bigger datasets like AudioSet.

In this thesis, experiments with impure unlabeled sets were performed by exchanging 50% of the aforementioned set with audio recordings of unknown classes, namely the other 382 classes available in the dataset. Only examples with a high SNR ratio have been utilized in this thesis, both in the labeled set and the unlabeled set with unknown examples. Soundscape recordings will probably include species sounds with lower SNR ratios, in comparison to ones manually retrieved by bird watchers. As the model is trained only on high quality examples, it can fail to recognize low quality unlabeled examples, assigning the wrong pseudo-label, if any at all. Lower quality examples could be introduced in the labeled set to mitigate this problem.

Bibliography

[1] B. D. Sparrow, W. Edwards, S. E. Munroe, G. M. Wardle, G. R. Guerin, J.-F. Bastin, B. Morris, R. Christensen, S. Phinn and A. J. Lowe, 'Effective ecosystem monitoring requires a multi-scaled approach,' *Biological Reviews*, vol. 95, no. 6, pp. 1706–1719, 2020. DOI: https://doi.org/10.1111/brv.12636. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/brv.12636. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/brv.12636.

[2] C. L. of Ornithology, *Kaggle competition: Cornell Birdcall Identification*, https://www.kaggle.com/c/birdsong-recognition/overview, [Online; accessed 5-December-2021], 2020.

[3] S. Mekonen, 'Birds as biodiversity and environmental indicator,' *Indicator*, vol. 7, no. 21, 2017.

[4] S. Kahl, C. M. Wood, M. Eibl and H. Klinck, 'Birdnet: A deep learning solution for avian diversity monitoring,' *Ecological Informatics*, vol. 61, p. 101 236, 2021, ISSN: 1574-9541. DOI: https://doi.org/10.1016/j.ecoinf.2021.101236. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954121000273.

[5] E. Sprengel, M. Jaggi, Y. Kilcher and T. Hofmann, 'Audio based bird species identification using deep learning techniques,' in *CLEF*, 2016.

[6] V. Boddapati, A. Petef, J. Rasmusson and L. Lundberg, 'Classifying environmental sounds using image recognition networks,' *Procedia Computer Science*, vol. 112, pp. 2048–2056, 2017, Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 21st International Conference, KES-20176-8 September 2017, Marseille, France, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.08.250. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050917316599.

[7] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen and T. Virtanen, 'Convolutional recurrent neural networks for polyphonic sound event detection,' *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, Jun. 2017, ISSN: 2329-9304. DOI: 10.1109/taslp.2017.2690575. [Online]. Available: http://dx.doi.org/10.1109/TASLP.2017.2690575.

[8] G. Gupta, M. Kshirsagar, M. Zhong, S. Gholami and J. Lavista Ferres, 'Comparing recurrent convolutional neural networks for large scale bird species classification,' *Scientific Reports*, vol. 11, Aug. 2021. DOI: 10.1038/s41598-021-96446-w.

[9] F. Vesperini, L. Gabrielli, E. Principi and S. Squartini, 'Polyphonic sound event detection by using capsule neural networks,' *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 310–322, May 2019, ISSN: 1941-0484. DOI: 10.1109/jstsp.2019.2902305. [Online]. Available: http://dx.doi.org/10.1109/JSTSP.2019.2902305.

[10] E. J. Henri and Z. Mungloo–Dilmohamud, 'A deep transfer learning model for the identification of bird songs: A case study for mauritius,' in *2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, 2021, pp. 01–06. DOI: 10.1109/ICECCME52200.2021.9590917.

[11] I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri and D. Mahajan, *Billion-scale semi-supervised learning for image classification*, 2019. arXiv: 1905.00546 [cs.CV].

[12] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor and K. McGuinness, 'Pseudo-labeling and confirmation bias in deep semi-supervised learning,' in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207304.

[13] M. Zhong, J. LeBien, M. Campos-Cerqueira, R. Dodhia, J. Lavista Ferres, J. P. Velev and T. M. Aide, 'Multispecies bioacoustic classification using transfer learning of deep convolutional neural networks with pseudo-labeling,' *Applied Acoustics*, vol. 166, p. 107 375, 2020, ISSN: 0003-682X. DOI: https://doi.org/10.1016/j.apacoust.2020.107375. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003682X20304795.

[14] S. Grollmisch and E. Cano, 'Improving semi-supervised learning for audio classification with fixmatch,' *Electronics*, vol. 10, no. 15, 2021, ISSN: 2079-9292. DOI: 10.3390/electronics10151807. [Online]. Available: https://www.mdpi.com/2079-9292/10/15/1807.

[15] K. Lu, C.-S. Foo, K. Teh, T. Dat and V. Chandrasekhar, 'Semi-supervised audio classification with consistency-based regularization,' Sep. 2019, pp. 3654–3658. DOI: 10.21437/Interspeech.2019-1231.

[16] Xeno-canto, *xeno-canto. Sharing bird sounds from around the world*, https://xeno-canto.org/, [Online; accessed 12-December-2021].

[17] T. Giannakopoulos and A. Pikrakis, 'Chapter 2 - getting familiar with audio signals,' in *Introduction to Audio Analysis*, T. Giannakopoulos and A. Pikrakis, Eds., Oxford: Academic Press, 2014, pp. 9–31, ISBN: 978-0-08-099388-1. DOI: https://doi.org/10.1016/B978-0-08-099388-1.00002-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080993881000029.

[18] Phonical, *FFT-Time-Frequency-View.png*, Nov. 2017. [Online]. Available: https://commons.wikimedia.org/wiki/File:FFT-Time-Frequency-View.png.

[19] T. Giannakopoulos and A. Pikrakis, 'Chapter 3 - signal transforms and filtering essentials,' in *Introduction to Audio Analysis*, T. Giannakopoulos and A. Pikrakis, Eds., Oxford: Academic Press, 2014, pp. 33–57, ISBN: 978-0-08-099388-1. DOI: https://doi.org/10.1016/B978-0-08-099388-1.00003-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080993881000030.

[20] *Mel scale*, https://en.wikipedia.org/wiki/Mel_scale, [Online; accessed 12-December-2021].

[21] R. Yamashita, M. Nishio, R. Do and K. Togashi, 'Convolutional neural networks: An overview and application in radiology,' *Insights into Imaging*, vol. 9, Jun. 2018. DOI: 10.1007/s13244-018-0639-9.

[22] K. He, X. Zhang, S. Ren and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[23] S. J. Pan and Q. Yang, 'A survey on transfer learning,' *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010. DOI: 10.1109/TKDE.2009.191.

[24] O. Chapelle, B. Schölkopf and A. Zien, Eds., *Semi-Supervised Learning*. The MIT Press, 2006, ISBN: 9780262033589. [Online]. Available: http://dblp.uni-trier.de/db/books/collections/CSZ2006.html.

[25] D.-H. Lee, 'Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks,' *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*, Jul. 2013.

[26] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor and K. McGuinness, *Pseudo-labeling and confirmation bias in deep semi-supervised learning*, 2020. arXiv: 1908.02983 [cs.CV].

[27] K.-H. Tsai and H.-T. Lin, *Learning from label proportions with consistency regularization*, 2019. arXiv: 1910.13188 [cs.LG].

[28] ImageCLEF, *ImageCLEF - The CLEF Cross Language Image Retrieval Track*, https://www.imageclef.org/, [Online; accessed 12-December-2021].

[29] DCASE, *Detection and Classification of Acoustic Scenes and Events*, http://dcase.community/, [Online; accessed 5-December-2021].

[30] A. Joly, H. Goëau, H. Glotin, C. Spampinato, P. Bonnet, W.-P. Vellinga, J. Champ, R. Planqué, S. Palazzo and H. Müller, 'Lifeclef 2016: Multimedia life species identification challenges,' in *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, N. Fuhr, P. Quaresma, T. Gonçalves, B. Larsen, K. Balog, C. Macdonald, L. Cappellato and N. Ferro, Eds., Cham: Springer International Publishing, 2016, pp. 286–310, ISBN: 978-3-319-44564-9.

[31] C. K. Catchpole and P. J. Slater, *Bird song: biological themes and variations*. Cambridge university press, 2003.

[32] H. Zhang, M. Cisse, Y. N. Dauphin and D. Lopez-Paz, *Mixup: Beyond empirical risk minimization*, 2018. arXiv: 1710.09412 [cs.LG].

[33] T. Iqbal, Y. Xu, Q. Kong and W. Wang, *Capsule routing for sound event detection*, 2018. arXiv: 1806.04699 [cs.SD].

[34] S. Sabour, N. Frosst and G. E. Hinton, *Dynamic routing between capsules*, 2017. arXiv: 1710.09829 [cs.CV].

[35] B. Rowe, P. Eichinski, J. Zhang and P. Roe, 'Acoustic auto-encoders for biodiversity assessment,' *Ecological Informatics*, vol. 62, p. 101 237, 2021, ISSN: 1574-9541. DOI: https://doi.org/10.1016/j.ecoinf.2021.101237. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954121000285.

[36] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang and C. Raffel, *Fixmatch: Simplifying semi-supervised learning with consistency and confidence*, 2020. arXiv: 2001.07685 [cs.LG].

[37] A. Tarvainen and H. Valpola, *Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results*, 2018. arXiv: 1703.01780 [cs.NE].

[38] T. Denton, S. Wisdom and J. R. Hershey, *Improving bird classification with unsupervised sound separation*, 2021. arXiv: 2110.03209 [eess.AS].

[39] S. Wisdom, E. Tzinis, H. Erdogan, R. J. Weiss, K. Wilson and J. R. Hershey, *Unsupervised sound separation using mixture invariant training*, 2020. arXiv: 2006.12701 [eess.AS].

[40] T.-W. Lee, 'Independent component analysis,' in *Independent Component Analysis: Theory and Applications*. Boston, MA: Springer US, 1998, pp. 27–66, ISBN: 978-1-4757-2851-4. DOI: 10.1007/978-1-4757-2851-4_2. [Online]. Available: https://doi.org/10.1007/978-1-4757-2851-4_2.

[41] Google, *AudioSet: A large-scale dataset of manually annotated audio events*, https://research.google.com/audioset/, [Online; accessed 4-December-2021].

Appendix A

Data

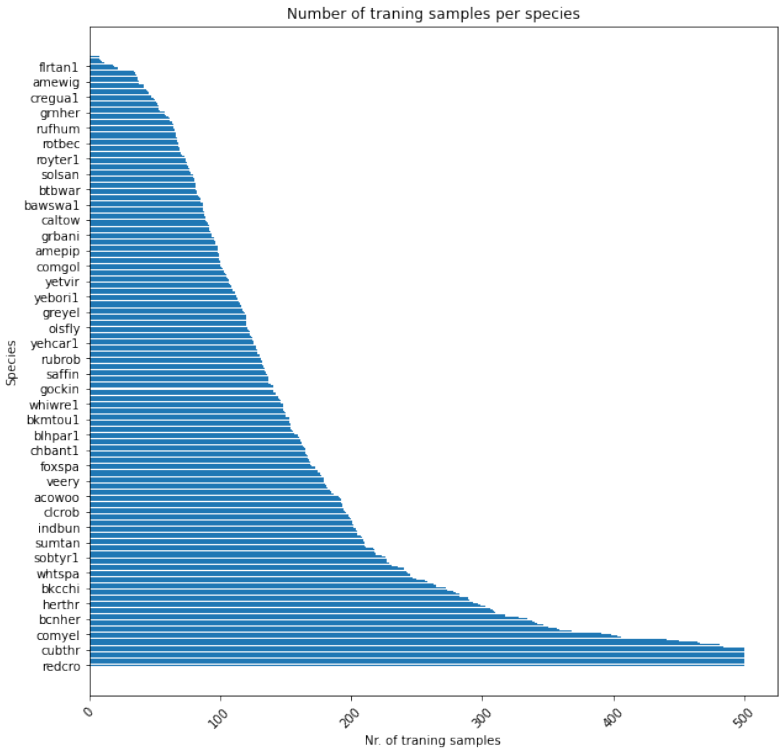


Figure A.1: Number of recordings per class. The figure shows high imbalance between classes. Around 35 of the 397 labels are shown along the y-axis.

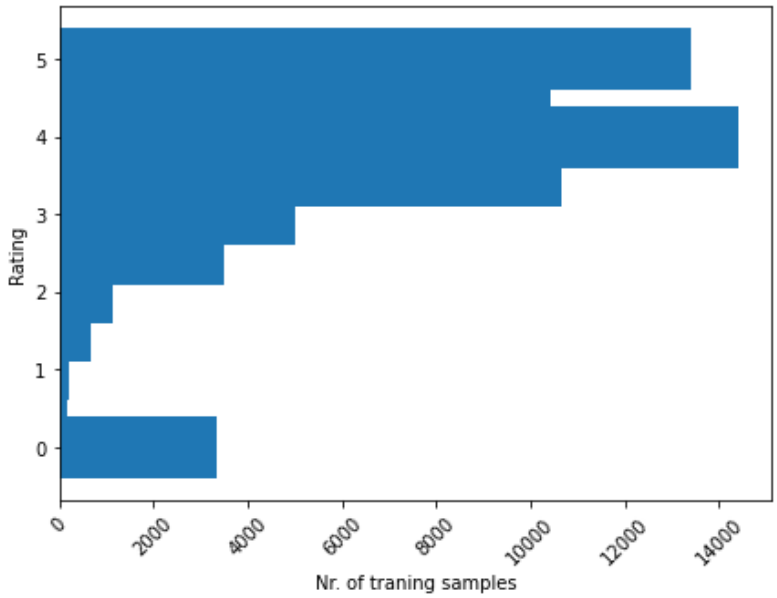


Figure A.2: Rating distribution: number of example per rating.

Emanuele Caprioli

A semi-supervised approach to bird song classification

NTNU
Norwegian University of
Science and Technology