AUTOMATIC BIRD SOUND IDENTIFICATION USING MACHINE LEARNING

H.D FERNANDO

M.Sc. 2021

AUTOMATIC BIRD SOUND IDENTIFICATION USING MACHINE LEARNING

A PROJECT REPORT PRESENTED BY

H.D FERNANDO
(PGIS/M/DTS/20/09)

to the Board of Study in Statistics and Computer Science of the

POSTGRADUATE INSTITUTE OF SCIENCE

in partial fulfillment of the requirement

for the award of the degree of

MASTER IN DATA SCIENCE

of the

UNIVERSITY OF PERADENIYA
SRI LANKA
2021

DECLARATION

I do hereby declare that the work reported in this project report/thesis was exclusively carried out by me under the supervision of …….. It describes the results of my own independent research except where due reference has been made in the text. No part of this project report/thesis has been submitted earlier or concurrently for the same or any other degree.

Date: ……. ……..

Signature of the Candidate

Certified by:

1. Supervisor (Name): …… Date: …….

   Signature: …….

2. Supervisor (Name): …… Date: …….

   Signature: …….

PGIS Stamp:

AUTOMATIC BIRD SOUND IDENTIFICATION USING MACHINE LEARNING

H.D Fernando

Postgraduate Institute of Science, University of Peradeniya, Peradeniya, Sri Lanka

Department of Statistics and Computer Science, University of Peradeniya, Peradeniya, Sri Lanka

Bird sound identification and recognition aim to detect sound events within an audio clip. This technology is useful in many real-world applications. Research on this topic has received increased attention in recent years. Performance of these identification technologies is increasing rapidly as a result of deep learning methods. Due to climate changes, many bird species are endangered. In order to protect them, we must first identify the species to which it belongs and the special care which needs to be taken care of for their survival. Human ability to recognize the birds through images is much easier as compared to audio classification, so bird species classification through images is in more trend and accurate too. Traditional machine learning and audio-based approaches are not suitable for bird classification. In this project our goal is to classify specific bird sounds using several deep learning models. Transfer learning of a pre-trained model is employed to reduce the necessary training data and time. We have used Mel spectrogram as the audio representation. Pre-trained Resnet models were used. These models were fine-tuned on 62,874 audio recordings representing 397 bird species in the Xano-Canto bird song sharing portal. Which provides a large collection of labeled and categorized recordings. This dataset includes recordings of multiple and potentially overlapping bird vocalizations with background noise. Audio records were transformed into Mel-spectrograms and normalization has been done to images and saved them as precomputed mels. The models' results were evaluated using the pre-computed images and achieved an average precision score of 0.62 for Resnet18 and an average precision score of 0.57 for resnext50_32x4d.

TABLE OF CONTENTS

DECLARATION
ABSTRACT i
TABLE OF CONTENTS ii
LIST OF FIGURES iv
LIST OF TABLES v

CHAPTER 1: INTRODUCTION 1

1.1 Bird Calls 1
1.2 Bird Calls Monitoring with Wireless Sensor Networks
1.3 Machine Learning 3
1.3.1 Classification 3
1.3.2 Training Process 4
1.4 Neural Networks 6
1.4.1 Multi-layer Perceptron 6
1.4.2 Activation Functions 8
1.4.3 Training Neural Networks 9
1.4.4 Convolutional Layers 10
1.4.5 Convolutional Neural Networks 12
1.4.6 Efficient CNNs for image Classification 12
1.4.7 Transfer Learning 13
1.5 Audio Classification 14
1.5.1 Digital Sound 15
1.5.2 Signal processing 16
1.5.3 Spectrogram 17
1.5.4 Mel-Spectrogram 18
1.5.5 Normalization 19
1.5.6 Analysis windows 19
1.5.7 Data augmentation 20
1.5.8 Weak labeling 21

1.6 Microcontrollers 22
1.6.1 Machine Learning on microcontrollers 22
1.6.2 TinyML 23
1.6.3 Audio classifier with edge impulse 24
CHAPTER 2: MATERIALS AND METHODS 26
2.1. Bird Call Classification 26
2.1.1. Dataset 27
2.1.2. Audio waveform models 29
2.1.3. Spectrogram-based models 30
2.1.4. Transfer Learning with ResNet 31
2.2. Pre processing 33
2.3. Training 38
CHAPTER 3: RESULTS AND DISCUSSIONS 39
CHAPTER 4: CONCLUSION 42
CHAPTER 5: FUTURE WORK 43
REFERENCES 44
APPENDIX A 46

LIST OF FIGURES

Figure 1: Different types of calls of the Chaffinch bird. 1

Figure 2: Wireless bird classification project using Arduino Nano 33 BLE Sense. 2

Figure 3: Process of K-Fold cross validation technique. 5

Figure 4: Machine Learning pipeline. 5

Figure 5: A schematic diagram of a Multi-Layer Perceptron (MLP) neural network. 6

Figure 6: Activation Function Representation 7

Figure 7: Widely used activation functions in neural networks. 8

Figure 8: Plot of log-loss for binary classification 9

Figure 9: Convolution for a single channel. Blue outlines show how the filter moves 11

Figure 10: The LeNet-5 architecture illustrated. 12

Figure 11: Conversion of sound into a digital representation 15

Figure 12: Comparison of different filterbank responses 18

Figure 13: Different spectrograms showing a birdsong. Linear spectrogram. 18

Figure 14: Audio stream split into fixed-length analysis windows without overlap 19

Figure 15: Common data augmentations for audio. 21

Figure 16: Number of training samples per species 27

Figure 17: Birds distribution depending on the geo location 28

Figure 18: EnvNet architecture, using raw audio as input 29

Figure 19: The architecture of Piczak CNN, from the original paper 30

Figure 20: Training error with number of layers 31

Figure 21: Neural Network with and without skip connection 32

Figure 22: A comparison of layer depths 33

Figure 23: Audio waveform of a bird call 34

Figure 24: Spectrogram representation of the raw signal 34

Figure 25: Resolution with window length (128, 256, 512, and 1024) 35

Figure 26: 15-seconds, 10-seconds and 5-second mel scale spectrograms 37

Figure 27: Final input to the model (Image size 128*281) 38

Figure 28: Training-Validation F1-scores and Precisions 40

LIST OF TABLES

Table 1: Examples of available STM32 microcontrollers and their characteristics 22

CHAPTER 01

INTRODUCTION

1.1 Bird Calls

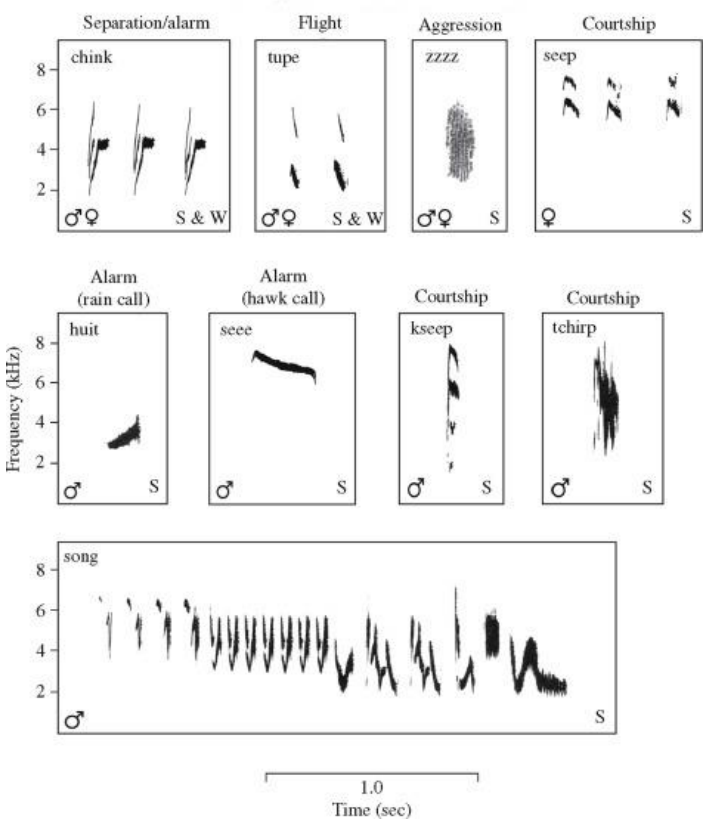


Figure 1: Different types of calls of the Chaffinch bird.

There are over 10,000 bird species around the world today, each with their own unique habitat preferences, diets, ranges and behavior habitats. Identifying the Bewick's wrens or red-winged blackbirds in an area, for example, can provide important information about the habitat. Birds make different types of calls (Figure 1), songs, and sounds with a language as complicated as any spoken words. Each type of sound has a different purpose, and birds use them in different conditions. For example, there are alarm calls, begging calls, contact calls, flight calls and songs. In addition to calls and songs, many bird species also include nonvocal sounds into their

language. Bird species that flock often call back and forth while in flight. That is a good way to detect clouds and predators. Young birds make different noises, often while simultaneously fluttering their wings to get their parents' attention. Because of this complexity bird call identification is laborious task.

Birds are some of the only animals found on every continent of the world. They occupy a wide range of terrestrial environments. When seasons change many bird species often migrate from one habitat to another. Following climates helps for breeding and gathering the most common threat on global threatened birds is ecosystem conversion. Only in the past few decades have scientists understood that this destruction, join with the uniqueness of the species in these areas, presents one of the greatest challenges to conservation. Some of the bird species have small geographical range, these birds face the highest possibility to extinction. It is crucial to monitor these birds' behavior and track their population. When it comes to birds' behavior, we can study their communication, Courtship, Reproduction and Migration. Bird call identification is can be used to track all of these areas. Currently, there are many new inventions to track bird populations and audio classification is in the top of the that list.

1.2 Bird Calls Monitoring with Wireless Sensor Networks



Figure 2: Wireless bird classification project using Arduino Nano 33 BLE Sense.

Wireless Sensor Networks (WSN) based audio monitoring is helpful for scientists and ecologist, for the purpose of monitoring real-time birds' behavior across remotely located areas for considerably large period of time under different climate conditions. Acoustic classification and monitoring have become really popular and with the advancement of modern technology, it is

possible to process and take certain decision about real time data on the onsite devices. There are very advanced and expensive audio detection systems but to monitor large geographical areas, it is required large number of devices. So basically, these devices should be inexpensive and it should be possible to record data for a long time of period. Smartphones have used to remotely monitor birds, but it is not that efficient because they fail to operate for reliably long period of time in a power efficient manner. Microcontrollers can be used to solve all of these practical problems. For example, Raspberry pi, Arduino and ESP devices used to acoustic classification. There are research going on to find low-cost satellite systems that can use with IoT applications. Lacuna Space [1], in contrast to some of its competitors, aims to operate satellites that can directly pick up signals from long range (LoRa) standard IoT sensors (Figure 2) instead of having the signals transferred to a dish and amplified to reach space.

1.3 Machine Learning

Machine Learning is the study of computer algorithms that improves automatically through experience and use of data. It includes computers identifying how they can perform tasks without having to explicitly program the instructions for how to perform this task.

Supervised learning can apply what has been learned in the past to new data using labeled examples to predict future events. It can begin from the analysis of a known training dataset. The learning algorithm construct an inferred function to make predictions about the output values. The trained system is able to provide target values for any new input after abundant training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly. Supervised learning techniques can be used for regression and for classification. In regression where the object is to predict a continuous variable, and for classification a discrete variable.

1.3.1 Classification

The classification predictive modeling is the task of approximating the mapping function from input variables to discrete output variables. The main objective is to identify which

category/class the new data will fall into. For example, predict from a text data whether it is positive or negative in subject matter, determining from an image which breed a cat is.

Most of the classification problems are single label. In single class classification problems, a classifier assigns a single class to each instance and the classes form a flat (non-hierarchical) structure, without superclasses or subclasses. Evaluating machine learning algorithm is an essential part of any project. Metrics are used to evaluate the performance of machine learning models. Most common metrics for classification are Accuracy, Precision and Recall(sensitivity). There will be a tradeoff between Precision and recall for any given model. To fully evaluate the performance of a model, you must examine both precision and recall. Receiver Operating Characteristics Curve (ROC) used to evaluate possible range of tradeoffs in binary classification. Sound classification involves learning to classify sounds and to predict the category of that sound. This type of problem can be applied to many practical scenarios e.g. classifying music clips to identify the genre of the music, or classifying short utterances by a set of speakers to identify the speaker based on the voice.

1.3.2 Training Process

The main goal of a classification model is to identify the category of an unseen dataset, and these algorithms are mainly used to predict the output for the categorical data. An important consideration in learning the target function from the training data is how well the machine learning model generalizes to new set of data. The data we collect is only a sample, it is incomplete and noisy. So, Generalization is really important. A model that models training data too well fails this generalization criteria, it is often said to be overfitting. A model that can neither model the training data nor generalize to new data is said to be underfitting.

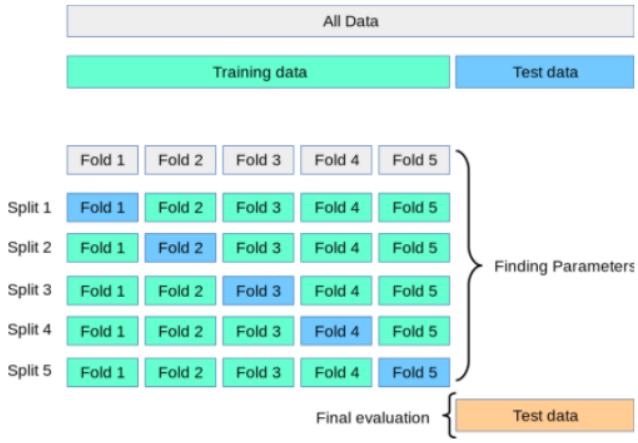


Figure 3: Process of K-Fold cross validation technique.

To solve these problems the dataset is divided into multiple subsets that have different purposes. The training set is data that training algorithms uses to optimize the learning model. This training data usually needs to be enriched or labeled. Might need to collect more of it to power these algorithms. Predictions are made on the validation dataset then we can estimate how well the model generalize to new unseen data. The final performance of the trained machine learning model is evaluated on a test set, which has been hold back from machine learning algorithms until the very end of the project. The most demanding resampling technique is k-fold cross validation. It trains and test your model k-times on different subsets of training data and build up an estimate of the performance of a machine learning model on unseen data. Usually, k is set to value between 5 and 10. The overall process is illustrated in Figure 3.

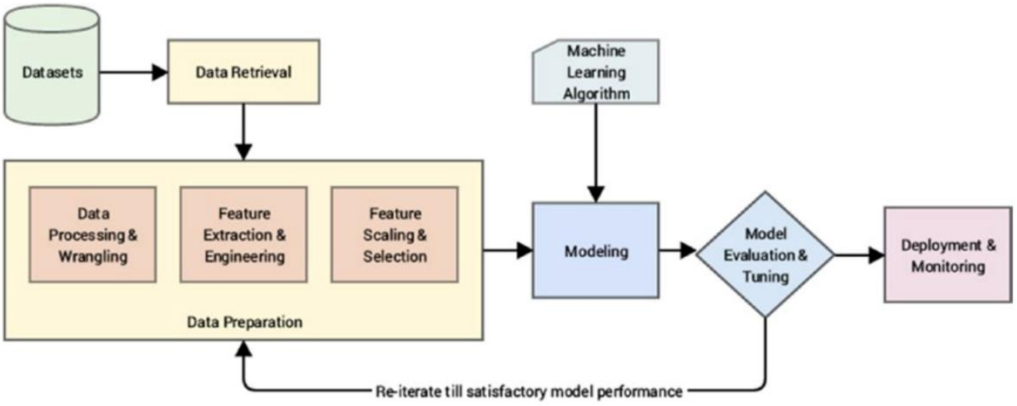


Figure 4: Machine Learning pipeline.

When we starting learning process in supervised learning, it can be start with a base model and initialize its parameters using domain knowledge or randomly, then make prediction using trained model, compare results with the labels to compute an error. Then reduce this error by updating parameters. This process shown in Figure 4

Hyperparameters are the variables which determines the network structure and the variables which determine how the network is trained. Hyperparameters can be chosen using different settings, training model(s) to compare these settings, and evaluating final performance on the test set. The outcome of hyperparameter tuning is the best hyperparameter setting, and the outcome of model training is the best model parameter setting. When training is completed, the predictive model can be used on new unseen data.

1.4 Neural Networks

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Some of the foundations such as the Perceptron [2] dates back to the 1950s, but it was not until around 2010 that neural networks started to become the preferred choice for many machine learning applications.

1.4.1 Multi-layer Perceptron

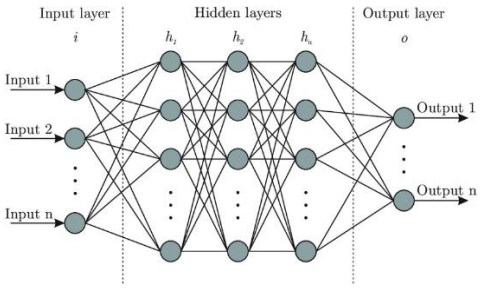


Figure 5: A schematic diagram of a Multi-Layer Perceptron (MLP) neural network.

Multi-layer perceptron (MLP) is a supplement of feed forward neural networks. As shown in the Figure 5. It consists of three layers-the input layer, output layer and hidden layer.

Each layer consists of number of neurons. The input of the network is a 1-dimensional vector. If data is multi-dimensional, it must be flattened to 1 dimensional vector. Fully connected neural networks (FCNNs) are a type of artificial neural network where the architecture is such that all the nodes, or neurons, in one layer are connected to the neurons in the next layer. In MLP each layer act as fully connected layer.

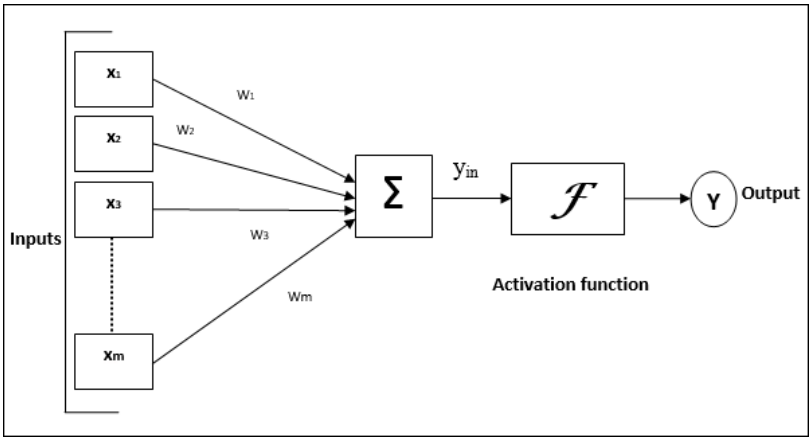


Figure 6: Activation Function Representation

Each neuron has a weighted vector, these weights for each neuron tuned during the training stage such as final output is biased toward some value. Nonlinear behavior in neural network is accomplished by use of an activation function to which the output is passed and modified (Figure 6). This allows neural networks to describe more complicated systems while still combining inputs in a simple fashion.

Forward propagation refers to the calculation and storage of intermediate variables for a neural network in order from the input layer to the output layer. Making predictions in neural networks done by using forward propagation.

1.4.2 Activation Functions

An activation function decides neuron's activation status. This means it will decide whether the neuron's input to the network is important or not in the process of predicting using mathematical operations.

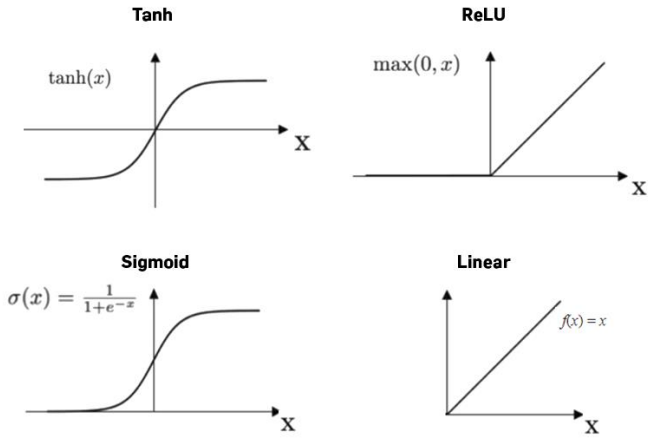


Figure 7: Widely used activation functions in neural networks. Input shown along X-axis, output along Y. Range for Sigmoid is (0,1) and for Tanh (-1,1)

Tanh and Relu[3] are commonly used general purpose activation functions. Sigmoid and softmax are used to convert the input into probability. They are commonly used in output stage of a neural network. Sigmoid is used to binary classification and Softmax is used for multi-class classification. To get a discrete class from continues probability values, a decision function is applied. The simplest decision function for single-label multi-class classification is to take the largest value, using the argmax function. An illustration of the mentioned activation functions can be seen in Figure 7.

When we increase the number of neurons and number of hidden layers in the neural network, it will increase the capacity of the network to learn more complex functions. But this means not necessary, that your classifier is then better on the 'next dataset'. The model gets more affected to over-fitting and so the generalization of you classification model can also decrease.

1.4.3 Training Neural Networks

Neural Networks are trained through numerical optimization of an objective function (loss function). For supervised learning, the standard method is mini-batch Gradient Descent with Backpropagation.

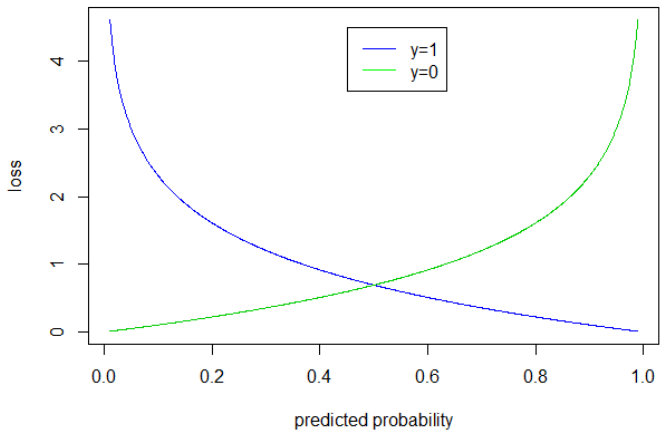


Figure 8: Plot of log-loss for binary classification

For classification tasks, the cross-entropy (log loss) function is often applied. As the predicted probability of the true class gets close to zero, the log-loss goes close to infinity. This penalizes wrong predictions heavily, see Figure 8.

Categorical cross-entropy is an extension of binary cross-entropy to multiple classes. Other loss functions are Mean Squared Error, Mean Absolute Error and Logistic Loss. Making predictions with a forward pass of the neural network and computing the loss function allows to calculate how well or how poorly the current model performs.

In order to find out how the model parameters should be changed in order to perform better; Gradient Descent is applied. The gradient of a function expresses how much and in which direction its output varies with small changes to its inputs. This is computed as the partial derivative of the function.

Backpropagation[4] is the key to calculate the gradients in a multi-layer neural network. This works by propagating the error in the last layers toward the input, using the partial derivative with respect to the layers before it. This makes it possible to compute the gradients for each of the weights in the network. Once the gradients are known, the weights are updated by taking a step in the negative direction of the gradient. The step is kept small by multiplying with the learning rate, a hyperparameter. Too small learning rates can lead to the model getting stuck in bad minima, while too large learning rates can cause training to not converge

In mini-batch Gradient Descent, the training data is usually processed in multiple fixed-size batches of data, and the model parameters and the loss function updates are computed per batch. Because of that all the training data not needed to keep in the memory at the same time, which allows training on very large datasets. The batch size is also a hyperparameter and has to be set high enough for the batch loss to be a reasonable estimate of the loss on the full training set, but small enough for the batch to fit into memory. One iteration through entire training set is called an epoch, training normally consist of many epochs.

1.4.4 Convolutional Layers

Convolution layers are the main building blocks used in convolutional neural networks that does most of the computational heavy lifting.

In convolution layers, in addition to the width and height, the input also has a third dimension, the number of channels. The different channels can contain random kinds of data, but common for the first layer(input) would be 3 channels with an RGB color image or 1 channel for grayscale images or audio spectrogram.

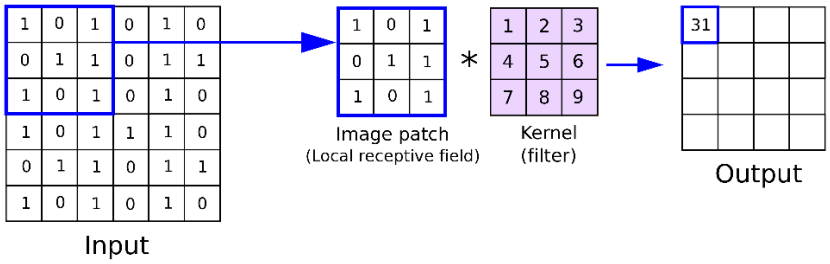


Figure 9: Convolution for a single channel. Blue outlines show how the filter moves

In the context of a convolutional neural network, a convolution is a linear operation that involves the multiplication of a set of weights with the input, much like a traditional neural network. Given that the technique was designed for two-dimensional input, the multiplication is performed between an array of input data and a two-dimensional array of weights, called a filter or a kernel. (Figure 9)

This convolution process allows expressing many useful transformations on two-dimentional data, by simply selecting different weights in the kernel. Simple examples are edge detection or smoothening filters (mean/Gaussian). And since the kernels weights in a convolutional layer are trained, they learn to detect local features specific to the training data.

When there are multiple input channels, the same kernel is applied over all the input channels, and all the results at one location., and the bias are computed together to become the output. Multiple convolution filters are normally used per layer, to produce M new output channels with different features from the N input channels.

In Figure 9 each location of the kernel has the entire kernel inside the input area. This is called convolution with “valid” padding, and the resulting output will be smaller by k/2 on each side, where k is the kernel size. If the input is instead padded by this amount when moving the kernel, the output will be the same size as the input. This is called convolution with “full” padding.

1.4.5 Convolutional Neural Networks

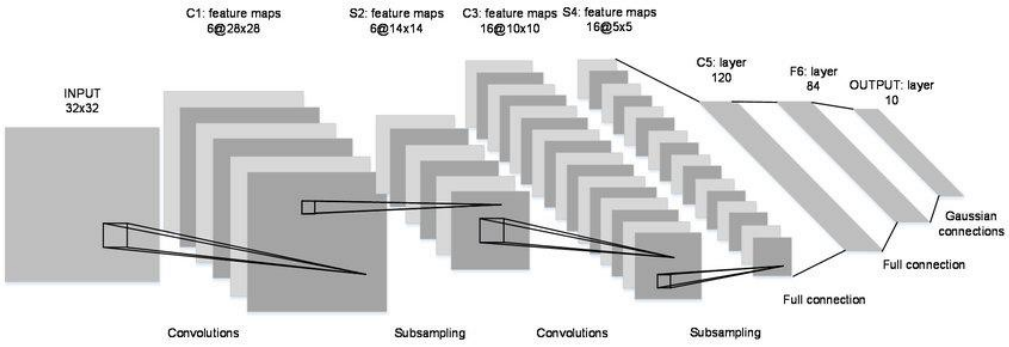


Figure 10: The LeNet-5 architecture illustrated.

The innovation of convolutional neural networks is the ability to automatically learn a large number of filters in parallel specific to a training dataset under the constraints of a specific predictive modeling problem, such as image classification. One of the early examples of a CNN model was LeNet5 [5], which was successfully applied to the recognition of handwritten digits. As seen in Figure 10, the architecture uses two convolutional layers (with subsampling after each), followed by two fully-connected layers and then the output Architectures with more layers based on very similar structures have been shown to work well also on more complex tasks, like VGGNet [6] on the 1000-class image recognition task ImageNet [7].

1.4.6 Efficient CNNs for image Classification

The development of more efficient Convolutional Neural Networks for image classification has received a lot of attention over the last few years. This is especially drive by the ability to run models that give close to state-of-the-art performance on mobile phones and tablets. Since spectrograms are 2D inputs that are similar to images, it is possible that some of these techniques can be use in Environmental Sound Classification.

SqueezeNet[8] (2015) focused on reducing the size of model parameters. It demonstrated AlexNet[9]-level accuracy on ImageNet challenge using 50x fewer parameters, and the parameters can be compressed to under 0.5MB in size compared to 240MB for AlexNet. It

replaced most 3x3 convolutions in a convolution block with 1x1 convolutions, and reduce the number of channels using “Squeeze” layers consisting only of 1x1 convolutions. The paper also found that a residual connection between blocks increased model performance by 2.9% without adding parameters.

Mobilenets[10] (2017) focused on reducing inference computations by using Depthwise separable convolutions. A family of models with different complexity was created using two hyperparameters: a width multiplier α (0.0-1.0) which adjusts the number of filters in each convolutional layer, and the input image size. On ImageNet, MobileNet-160 $\alpha = 0.5$ with 76M MACC performs better than SqueezeNet with 1700M MACC, a 22x reduction. The smallest tested model was 0.25 MobileNet-128, with 15M MACC and 200k parameters

Effnet[11] (2018) uses spatial separable convolutions, but additionally performs the downsampling in a separable fashion: first a 1x2 max pooling after the 1x3 kernel, followed by 2x1 striding in the 3x1 kernel.

1.4.6 Transfer Learning

The main aim of transfer learning (TL) is to implement a model quickly. To solve the current problem, instead of creating a Dense Neural Network (DNN) from scratch, the model will transfer the features it has learned from the different dataset that has performed the same task. This transaction is also known as knowledge transfer.

Transfer learning is a method where a model developed for one task is reused/repurposed for a second related task. The first model is used as the starting point for the second task. Transfer learning is useful and important in deep learning given a large amount of data required to train a CNN model from scratch. In transfer learning, a source model is selected firstly. The source model is a pre-trained model that is trained on large and challenging datasets. The source model is then used as the starting point for the task of interest. In this, it may involve only using parts of the model or the whole model depending on the task of interest. The source model is fine adapted for the task at hand by fine-tuning the source model based on input-output pairs of the task of interests.

In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a ConvNet on a very large dataset (e.g., ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest.

With this technique learning process can be faster, more accurate and need less training data, in fact, the size of the dataset and the similarity with the original dataset (the one in which the network was initially trained) are the two keys to consider before applying transfer learning. In a network, the earlier layers capture the simplest features of the images (edges, lines…) whereas the deep layers capture more complex features in a combination of the earlier layers (for example eyes or mouth in a face recognition problem). In order to fine-tune a model, we need to retrain the final layers because the earlier layers have knowledge useful for us.

PyTorch offers several trained networks ready to download to your computer. Each model has its own benefits to solve a particular type of problem. Ranging from image classification to semantic segmentation. Some are faster than others and required less/more computation power to run. For example, choosing SqueezeNet requires 50x fewer parameters than AlexNet while achieving the same accuracy in ImageNet dataset, so it is a fast, smaller and high precision network architecture (suitable for embedded devices with low power) while VGG network architecture have better precision than AlexNet or SqueezeNet but is heavier to train and run-in inference process.

1.5 Audio Classification

In Audio Classification, the relevant predictive models operate on audio data (digital sound). Music genre or artist classification in Music Information Retrieval, speech command identification in Speech Recognition are some example tasks.

Machine learning has shown exemplary results when evaluating the environment using pictures. However, this field has not been fully exploited in audio classification. This is because sound can give us a nondirectional perspective, unlike a camera. Sound does not depend on illumination. This means that you can hear the sound the same way no matter if it’s day or night.

Nevertheless, converting sound waves into audio and spectrograms (visual representation of frequencies) can allow us to use machine learning capabilities. Machine learning for audio can be used in pitch detection and music generation. For our case, we'll use it for classification

1.5.1 Digital Sound

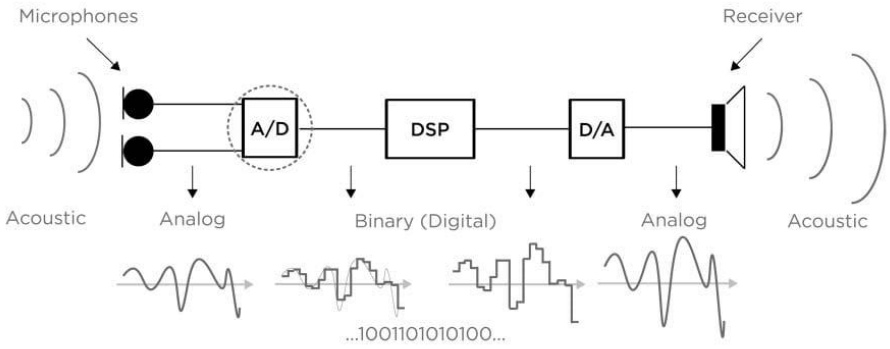


Figure 11: Conversion of sound into a digital representation

Sound is a mechanical wave that results from the back-and-forth vibration of the particles of the medium through which the sound wave is moving. If a sound wave is moving from left to right through air, then particles of air will be displaced both rightward and leftward as the energy of the sound wave passes through it. The motion of the particles is parallel (and anti-parallel) to the direction of the energy transport. This physical sound must be converted to digital format to process the sound with machine learning. The acoustic data is converted to analog electric signal by microphone and then digitized using an Analog-to-Digital Converter, as illustrated in the Figure 11.

In the digitization process, the signal is quantized in time at a certain sampling frequency, and the amplitude quantized at a certain bit-depth. A typical sampling frequency is 44100 Hz and a bit-depth 16 bit, as used in the Audio CD format [12]. Using these parameters, the most of human-perceivable information in acoustic sound is captured. In waveform representation sound is a one-dimensional sequence of numbers.

Recordings can have multiple channels of audio but for machine learning on audio single-channel data (mono) is still common in the current world.

1.5.2 Signal processing

In order to create a bird call identification device, the we needed to correctly utilize several signal processing techniques. Some of these techniques included filters, discrete Fourier transforms and cross-correlation. Each technique manipulated the bird call in a different way to help identify which species the call originated from. The filters were used to filter the incoming bird call and remove the noise that would disturb the results. Discrete Fourier transforms transformed the signals into the frequency domain so we were able to examine the different frequencies within each signal. Cross-correlation let the group compare the incoming bird call with their database of bird calls, both in the time and frequency domains, and match the signals with the highest correlations to determine the most likely matches.

The first signal processing technique necessary to identify a bird call is filtering. Filtering is necessary because it is likely that the bird call will contain a certain amount of noise. Additive noise could adversely affect the results. There are several types of filters that could have used to filter noise. Some basic filters are low-pass filters, high-pass filters, band-pass filters and band-stop filters. Low-pass filters function by passing low-frequency signals and attenuating or reducing the amplitude of high-frequency signals that are higher than the filters designed cutoff frequency. High-pass filters function the opposite way by passing high-frequency signals and attenuating frequencies lower than the cutoff frequency. Band-pass filters pass frequencies within a certain range and reject all frequencies outside that range. These filters can be created by combining a low-pass filter with a high-pass filter. The opposite of a band-pass filter is a band-stop filter. This filter blocks all frequencies within a certain range and passes all others. Since bird calls are characterized by a wide range of frequencies, it is important to be very particular in their choice of filters in order to prevent any information loss and to ensure that the filters are only removing unwanted noise.

The second signal processing technique used was discrete Fourier transforms (DFTs). Since, we were sampling analog signals from a microphone, all of the bird calls were digital signals. This means that some signal processing was done digitally using discrete signals. A DFT transforms one function in the time domain into another function in the frequency domain.

1.5.3 Spectrogram

Sounds of interest often have characteristic patterns not just in time (temporal signature) but also in frequency content (spectral signature). Therefore, it is common to analyze audio in a time-frequency representation (a spectrogram).

The Shot-Term Fourier Transform (STFT)[13] is the most common way to compute spectrogram from an audio waveform. The STFT operates by splitting the audio up in short consecutive chunks and computing the Fast Fourier Transform (FFT) to estimate the frequency content for each chunk. To reduce artifacts at the boundary of chunks, they are overlapped and a window function (such as the Hann window) is applied before computing the FFT. With the appropriate choice of window function and overlap, the STFT is invertible[14].

There is a trade-off between frequency (spectral) resolution and time resolution with the STFT. The longer the FFT window the better the frequency resolution, but the poorer the temporal resolution. For speech, a typical choice of window length is 20 ms. Similar frame lengths are often adopted for acoustic events. The STFT returns complex numbers describing the phase and magnitude of each frequency bin. A spectrogram is computed by squaring the absolute of the magnitude and discarding the phase information. This is called a linear spectrogram or sometimes just spectrogram. The lack of phase information means that the spectrogram is not strictly invertible, though estimations exist[15].

1.5.4 Mel-Spectrogram

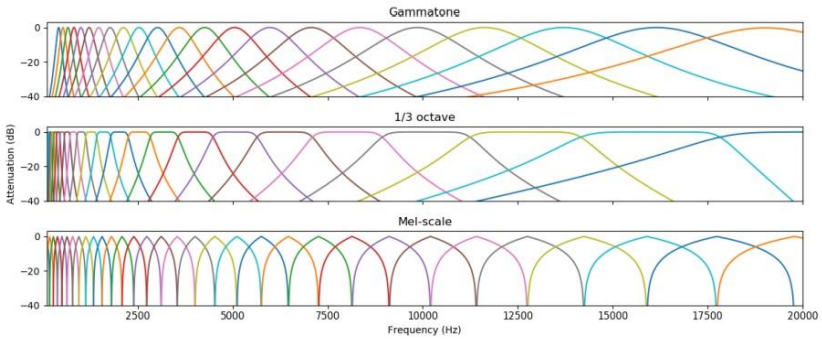


Figure 12: Comparison of different filterbank responses

When the input to the machine learning algorithm gets more complex, more processing power is needed both for training and for inference. Therefore, it is better to reduce the dimensions of the input as much as possible. A linear spectrogram often has considerable correlation (redundant information) between adjacent frequency bins and is often reduced to 30-128 frequency bands using a filter-bank. Several different filter-bank alternatives have been investigated for audio classification tasks, such as 1/3 octave bands, the Bark scale, Gammatone, and the Mel scale. All these have filters spacing that increases with frequency, mimicking the human auditory system. See Figure 12.

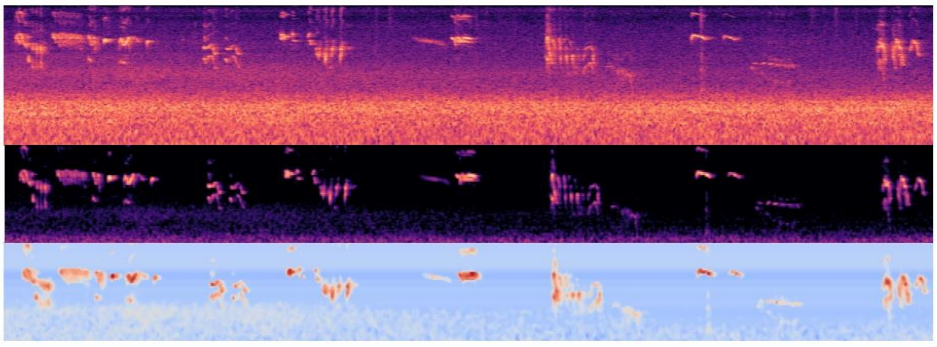


Figure 13: Different spectrograms showing a birdsong. Linear spectrogram. Mel-spectrogram, Normalized mel-spectrogram after mean-subtraction and standard scaling

The Mel scaled filters are generally used for audio classification[16]. The spectrogram that results for applying a Mel-scale filter-bank is often called a Mel-spectrogram. Mel-Filter Cepstral Coefficients (MFCC) is a feature representation computed by per- forming a Discrete Cosine Transform (DCT) on a Mel-spectrogram. This further reduces dimensionality to just 13-20 bands with low correlation between each band. MFCC features have been very popular in speech recognition tasks[16], however in general sound classification tasks Mel-spectrograms tend to perform better[17]. Figure 13 illustrates different spectrograms showing a birdsong.

1.5.5 Normalization

Audio has a very large dynamic range. The human hearing has a lower threshold of hearing down to 20µPa (0 dB Sound pressure level) and a pain threshold of over 20 Pa (120 dB SPL), a difference of 6 orders of magnitude [18] A normal conversation might be 60 dB SPL and a pneumatic drill 110 dB SPL, 4 orders of magnitude difference. It is common to compress the range of values in spectrograms by applying a log transform.

In order to center the values, the mean (or median) of the spectrogram can be removed. Scaling the output to a range of 0-1 or -1,1 is also sometimes done. These changes have the effect of removing amplitude variations, forcing the model to focus on the patterns of the sound regardless of amplitude.

1.5.6 Analysis windows

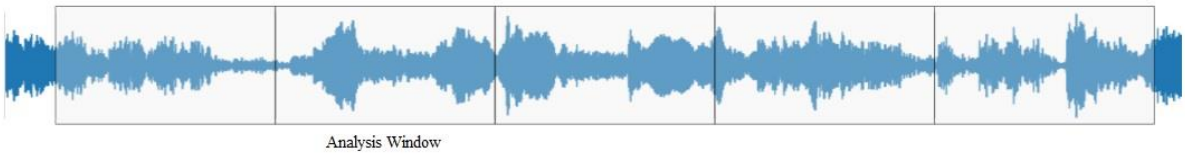


Figure 14: Audio stream split into fixed-length analysis windows without overlap

When recording audio, it forms a continuous, never-ending stream of data. The machine learning classifier however generally needs a fixed-size feature vector. Also, when playing back a finite-length recording, the file may be many times longer than the sounds that are of interest. To solve these issues, the audio stream is split up into analysis windows of fixed length (Figure 14), which are classified separately. Most of the cases, the length of the window is typically a little bit longer than the longest target sound. The windows can follow each-other with no overlap, or move forward by a number less than the window length (overlap). With overlap, a target sound will be classified a couple of times, at a slightly different position inside the analysis window each time. This can help to improve classification accuracy.

A short analysis window has the benefit of reducing the feature size of the classifier, which uses less memory and possibly allows to reduce the model complexity, and in turn allow to make better use of a limited dataset. When the length of audio clips is not evenly divisible by length of analysis windows, the last window is zero-padded.

1.5.7 Data augmentation

Data augmentation is an approach that is used to expand the dataset without collecting new data, which helps to reduce overfitting and gives better generalization. This is achieved by applying slight modifications to the existing data, resulting in an increase of the diversity in the dataset.

Access to labeled samples is often limited because they are expensive to acquire. This can be a limiting factor for reaching good performance using supervised machine learning. Data Augmentation is a way to synthetically generate new labeled samples from existing ones, in order to expand the effective training set. A simple form of data augmentation can be done by modifying the sample data slightly. Common data augmentation techniques for audio include Time-shift, Pitch-shift, and Time-stretch. These are demonstrated in Figure 15. Data augmentation can be applied either to the raw audio waveform or to preprocessed spectrograms.

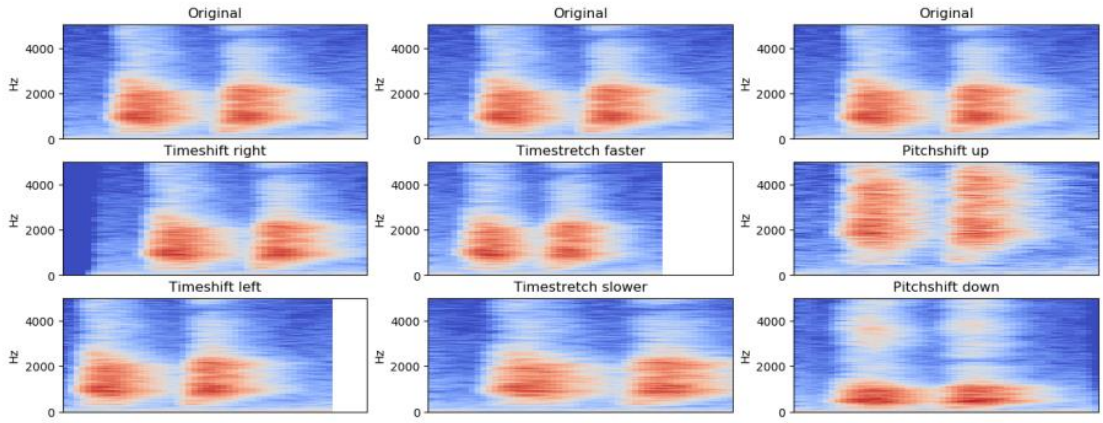


Figure 15: Common data augmentations for audio. Figure shows log-scaled linear spectrograms before and after applying the augmentation.

1.5.8 Weak labeling

Sometimes there could be a mismatch between the desired length of the analysis window, and the labeled clips available in the training data. For example, a dataset may consist of labeled audio clips with a length of 10 seconds, while the desired output is every second. When a dataset is labeled only with the presence of a sound at a coarse timescale, without information about where exactly the relevant sound(s) appears it is referred to as weakly annotated or weakly labeled data.

If one assumes that the sound of interest occurs throughout the entire audio clip, a simple solution is to let each analysis window inherit the label of the audio clip without modification, and to train on individual analysis windows. If this assumption is problematic, the task can be approached as a Multiple Instance Learning (MIL) problem. Under MIL each training sample is a bag of instances (in this case, all analysis windows in an audio clip), and the label is associated with this bag. The model is then supposed to learn the relationship between individual instances and the label.

1.6 Microcontrollers

A microcontroller is an integrated circuit (IC) device used for controlling other portions of an electronic system, usually via a microprocessor unit (MPU), memory, and some peripherals. These devices are optimized for embedded applications that require both processing functionality and agile, responsive interaction with digital, analog, or electromechanical components.

More recently, new generations of microcontrollers such as the ARM Cortex-M family or the ESP32 have been offering extended computational power (e.g., 600MHz for the CortexM7), suitable for real-time audio processing applications. Examples of microcontrollers (from ST Microelectronics) that could be used for audio processing are shown in Table 1 Similar offerings are available from other manufacturers such as Texas Instruments, Freescale, Atmel, Nordic Semiconductors, NXP.

Table 1: Examples of available STM32 microcontrollers and their characteristics.

Name	Architecture	Flash (KB)	RAM (KB)	CPU (MHz)
STM32F030CC	Cortex-M0	256	32	48
STM32L476	Cortex-M4	1024	128	80
STM32F746	Cortex-M7	1024	1024	216
STM32H743ZI	Cortex-H7	2048	1024	400
STM32H735G	Cortex-M7	1024	564	550

1.6.1 Machine Learning on microcontrollers

Machine learning enables software developers and engineers to unlock new capabilities in their applications. Instead of explicitly defining instructions and rules for a computer to execute, we can collect large amounts of data for a classification task that our application requires, and train an ML model to learn from the important patterns in the data.

It is a well-known fact that deep learning requires a lot of hardware to solve complex computational processing. The data scientists have swiftly moved from CPUs to GPUs. They devote proportionally more transistors to arithmetic logic units and fewer to caches and flow control as compared to CPUs. Training can be done is a high performing CPU/GPU or in the cloud on computers equipped with one or more GPUs. Once a model has been trained, depending on its size, it can be deployed for inference on a wide range of devices. These devices range from large computers in the cloud with gigabytes of memory, to small microcontrollers, which typically have just kilobytes of memory.

Microcontrollers are low-power, self-contained, cost-effective computer systems that are embedded in devices that you use every day, such as your smart devices, electric toothbrush, or smart door lock. Microcontroller based systems typically interact with their surrounding environment via one or more sensors (microphones, motion sensors, temperature sensors) and perform an action using one or more actuators (LEDs, speakers, motors). Microcontrollers also offer privacy advantages, and can perform inference locally on the device, without needing to send any data to the cloud. This can have power advantages too for devices running off batteries.

1.6.2 TinyML

We can conclude that TinyML is comparatively a new field, and is all about creating tiny machine learning models that can run on microcontrollers and low powered devices. These models are really tiny - in the order of kilobytes instead of the usual megabytes or gigabytes. They need to be this tiny to run on microcontrollers that typically have killobytes of RAM. These models also draw little power, typically in the single-digit milliwatts or lower.

There are a lot of use cases of TinyML. If you want to run ML models offline with minimal power draw, you can use TinyML. For example, smart voice-controlled devices listen for a wake word, and this needs to be offline and draw minimal power - perfect for a TinyML model. Another use case is in healthcare with devices that can monitor your health that run for years on tiny batteries. It's also being used in animal smart collars and trackers, using audio to monitor the health of elephants in the wild.

To train TinyML models you not only need the model to by tiny, but you also need small inputs - the more data that goes into training the model or inference (that is running the model), the larger it is. Audio data can be quite large - for example CD quality audio is 44.1KHz/16-bit which means it captures 2 bytes of data 44,100 times per second, or 176KB per second! That's a lot of data - if we wanted to use all of it and train our model with 2 seconds worth of data it wouldn't be TinyML anymore.

A great advantage with audio data is that you don't need all of it to classify particular sounds. Instead, you can get an average value that represents many samples and use that as the data. In the case of the Arduino, the library that captures audio, Pulse-density modulation (PDM), captures audio at 16KHz in 512-byte buffers, containing 256 2-byte samples. This means each buffer has 1/64th of a second of audio data in it. We can then calculate a root mean square (RMS) of all this data to get a single 4-byte floating point value. If we do this for every buffer, we end up with 64 4-byte floats per second, or 256 bytes per second. Much smaller than raw audio at the PDM sample rate of 16KHz giving 32,000 bytes per second!

1.6.3 Audio classifier with edge impulse

We can use machine learning to build a system that can recognize when a particular sound is happening, a task known as audio classification. The system you create will be able to recognize the sound of water running from a faucet, even in the presence of other background noise.

Edge Impulse provides a user-friendly way to train machine learning models without any problems. It has made most of the data processing part automated. Using edge Impulse, you can collect audio data from microphones, use signal processing to extract the most important information, and train a deep neural network that can tell you whether the sound of running water can be heard in a given clip of audio. Finally, you will be able deploy the system to an embedded device and evaluate how well it works.

Edge Impulse slices up the raw samples into windows that are fed into the machine learning model during training. The Window size field controls how long, in milliseconds, each window of data should be. A one second audio sample will be enough to determine whether a faucet is

running or not, so you should make sure Window size is set to 1000 ms. You can either drag the slider or type a new value directly.

Each raw sample is sliced into multiple windows, and the Window increase field controls the offset of each subsequent window from the first. For example, a Window increase value of 1000 ms would result in each window starting 1 second after the start of the previous one. By setting a Window increase that is smaller than the Window size, we can create windows that overlap. This is actually a great idea. Although they may contain similar data, each overlapping window is still a unique example of audio that represents the sample's label. By using overlapping windows, we can make the most of our training data. For example, with a Window size of 1000 ms and a Window increase of 100 ms, we can extract 10 unique windows from only 2 seconds of data.

In the Edge Impulse, at the start of training, 20% of the training data is set aside for validation. This means that instead of being used to train the model, it is used to evaluate how the model is performing. The Last training performance panel displays the results of this validation, providing some vital information about your model and how well it is working

The On-device performance region in the Edge Impulse shows statistics about how the model is likely to run on-device. Inferencing time is an estimate of how long the model will take to analyze one second of data on a typical microcontroller (ex., an Arm Cortex-M4F running at 80MHz). Peak memory usage gives an idea of how much RAM will be required to run the model on-device.

CHAPTER 02

MATERIALS AND METHODS

2.1 Bird Call Classification

Bird call Classification or Bird Song Classification, is the task of classifying audio of birds. It has been researched through many years. This helps researchers properly detect and classify bird sounds, remarkably improving their ability to monitor the trends and status of biodiversity in remote ecosystems.

Automatic species classification of birds from their sounds has many potential applications in conservation, ecology and archival. However, to be useful it must work with high accuracy across large numbers of possible species, on noisy outdoor recordings and at big data scales. The ability to scale to big data is crucial: remote monitoring stations can generate huge volumes of audio recordings, and audio archives contain large volumes of audio, much of it without detailed labelling. Big data scales also imply that methods must work without manual intervention, in particular without manual segmentation of recordings into song syllables, or into vocal/silent sections. The lack of segmentation is a pertinent issue for both remote monitoring and archive collections, since many species of bird may be audible for only a minority of the recorded time, and therefore much of the audio will contain irrelevant information.

Bioacoustics is very useful in studying the environment. It has been used for a long time to track submarines and whales. Birds help a lot to shape the plant life we see around us. Recognizing bird songs is very important for automatic monitoring of wildlife and studying the behavior of birds. It can help in tracking birds without disturbing them. We can also tell which birds species exist at a particular place. It can give us some information about their migration pattern. Every species of birds has their unique sounds. They use songs of varying length and complexity to attract mates, warn other birds of nearby danger, and mark their territory. Songbirds can have different dialects based on their geographical location. But sounds from non-songbirds do not change much based on geography. Using Deep learning methods, we can easily classify birds

based on their calls and songs. We can use any of the neural network frameworks like CNNs, Siamese Networks, WaveNets, etc. for this task.

2.1.1 Dataset

Xeno-canto[19] is a website dedicated to sharing bird sounds from all over the world. Anyone can download, listen and explore the bird sound recordings in the collection. The training data for this project consists of a collection of so-called “focal recordings”. These recordings were made using semi-professional equipment (often using highly directional microphones) and primarily focus on one single species. All recordings were contributed by Xeno-canto[19], one of the largest digital archives for bird sounds. Each recording comes with metadata specifying things like recording date, recording location, and the bird species that was recorded.

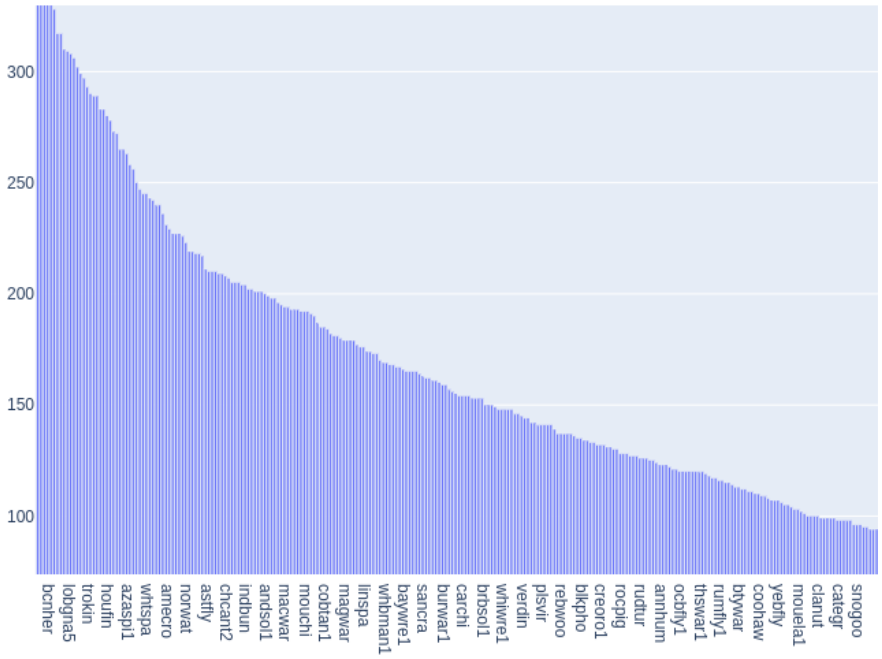


Figure 16: Number of training samples per species

This specific dataset has a total of 62874 samples, in 397 classes. We can clearly see that in this project we are facing a “long-tail” classification problem with highly imbalanced training

data(Figure 16). We have limited number of recordings to 500 for each species, but that only affected few species. It can be noted that the training data only contains species that are likely to occur at the test data recordings sites.

The metadata for each recording lists the number of audible background species. Secondary label contains list of eBird codes (i.e., primary labels) that recordists annotated. Sometimes we might be able to hear background noises of species, although none are specified in the metadata. Therefore, lists of secondary labels are not very reliable, but they might still be useful for multi-label training (e.g., through loss masking for background species).

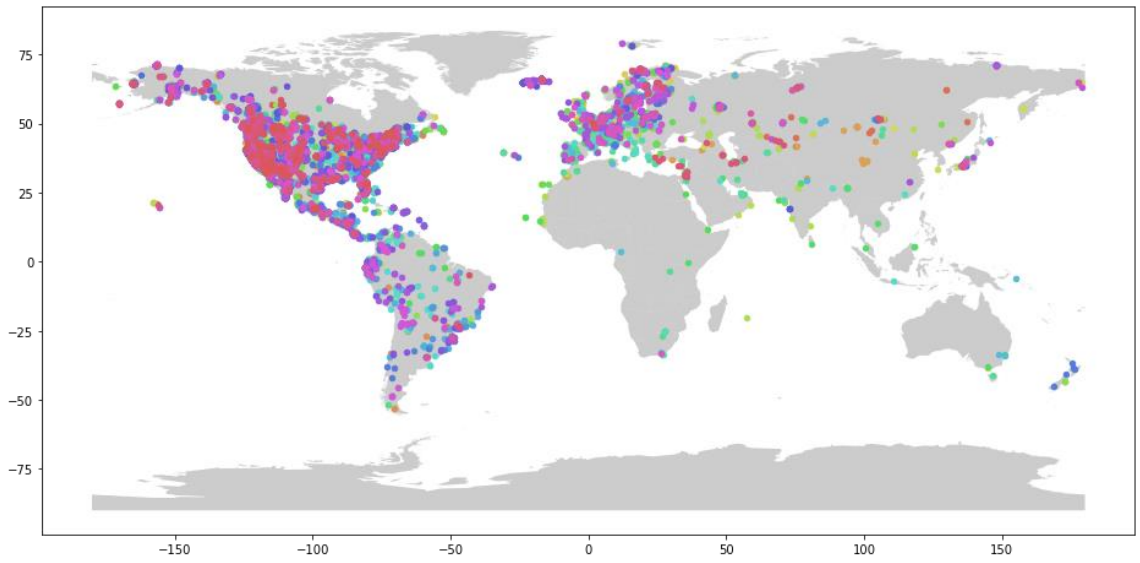


Figure 17: Birds distribution depending on the geo location

Each recording comes with a recording location specified in the metadata. Data fields “latitude” and “longitude” contain GPS coordinates as provided by the recordist. This information can be very useful to map distribution(Figure 17) and migration patterns because not all birds occur at all locations at all times.

Xeno-canto has a rating system for the quality of each recording. Ratings are assigned by users. In this dataset, ratings range from 0.5 to 5.0 (the latter being the best possible rating) and reflect the overall quality assigned by users and the number of background species. A value of “0” means that this particular recording does not have a rating, and it is by that the fallback value.

Sub-sampling training data based on user rating might help to extract high-quality training samples.

2.1.2 Audio waveform models

Some of the recent approaches that use the raw audio waveform as input have been documented.

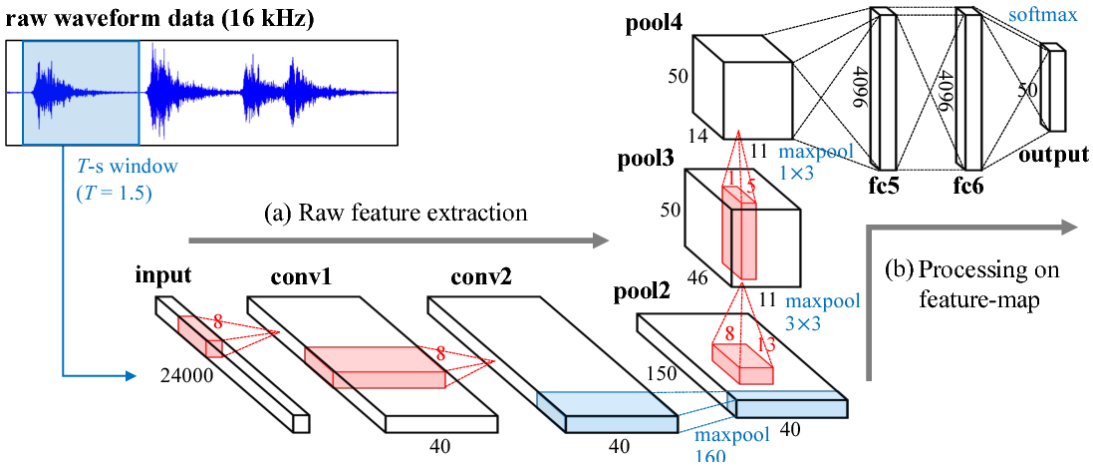


Figure 18: EnvNet architecture, using raw audio as input

EnvNet[20] (2017) used 1D convolutions in order to learn a 2D spectrogram like representation which is then classified using standard 2D convolutional layers. The model architecture is shown in Figure 18. They show that the resulting spectrograms have frequency responses with a shape similar to mel spectrograms. The model achieves a 66.3% accuracy score on Urbansound8k[21] with raw audio input.

In [22], the authors evaluated a number of deep CNNs using only 1D convolutions. Raw audio with 8 kHz sample rate was used as the input. Their 18-layer model (M18) got a 71% accuracy on Urbansound8k, and the 11-layer version (M11) got 69%. EnvNet2(2018) is like EnvNet but with 13 layers total instead of 7, and using 44.1 kHz input sample-rate instead of 16 kHz. Without data augmentation, it achieves 69.1% accuracy on Urbansound8k. When combining data augmentation with Between-Class examples, a technique similar to Mixup, the model is able to reach 78.3% on Urbansound8k.

2.1.3 Spectrogram-based models

Most of the recent papers have used Convolutional Neural Networks (CNN) for Environmental Sound Classification. Approaches based on spectrograms and in particular log-scaled Mel-spectrogram being the most common.

PiczakCNN[23] in 2015 was one of the first applications of CNNs to the Urbansound8k dataset. It uses 2 channels of log-Mel-spectrograms, both the plain spectrogram values and the first-order difference (delta spectrogram). The model(Figure 19) uses 2 convolutional layers, first with size 57x6 (frequency x time) and then 1x3, followed by two fully connected layers with 5000 neurons each. The paper evaluates short (950 ms) versus long (2.3 seconds) analysis windows and majority voting versus probability voting. Performance on Urbansound8k ranged from 69% to 73%. It was found that probability voting and long windows perform slightly better[23].

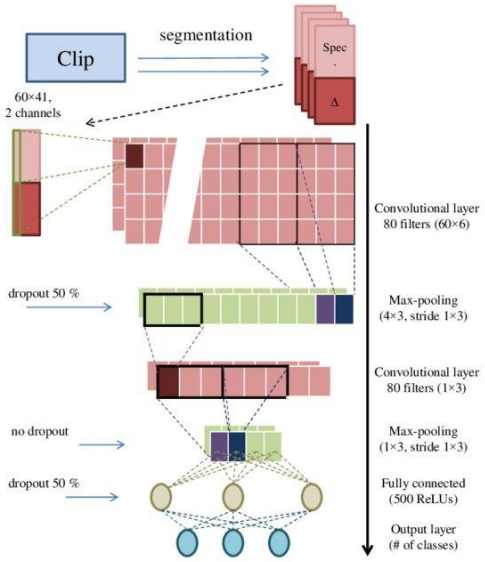


Figure 19: The architecture of Piczak CNN, from the original paper

SB-CNN[24] (2016) is a 3-layer convolutional with uniform 5x5 kernels and 4x2 max pooling. The paper also analyzes the effects of several types of data augmentation on Urbansound8k. including Time Shift, Pitch Shift, Dynamic Range Compression and Background Noise. With

all augmentations, performance on their model raised from 72% to 79% classification accuracy. However, time-stretching and pitch-shifting were the only techniques that consistently gave a performance boost across all classes. D-CNN[25] (2017) uses feature representation and model architecture that largely follows that of PiczakCNN. With additional data augmentation of time-stretching and noise addition, this gave a performance of up to 81.9% accuracy on Urbansound8k. LeakyRelu was found to perform slightly better than ReLu which scored 81.2%.

2.1.4 Transfer Learning with ResNet

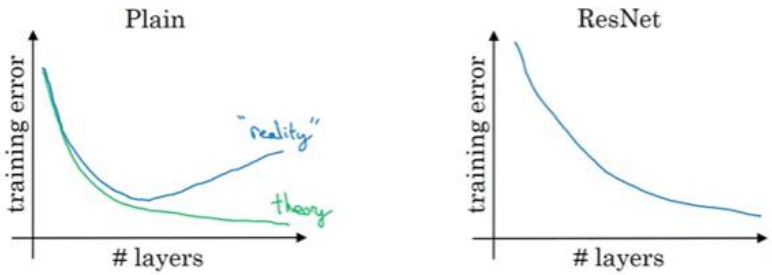


Figure 20: Training error with number of layers

A multi-layer deep neural network can produce unexpected results. In this case, the training accuracy dropped as the layers increased, technically known as vanishing gradients. While training, the vanishing gradient effect on network output with regard to parameters in the initial layer becomes extremely small. The gradient becomes further smaller as it reaches the minima. As a result, weights in initial layers update very slowly or remain unchanged, resulting in an increase in error.

A residual network, or ResNet for short, is an artificial neural network that helps to build deeper neural network by utilizing skip connections or shortcuts to jump over some layers. Skipping helps build deeper network layers without falling into the problem of vanishing gradients(Figure 20).

ResNet owes its name to its residual blocks with skip connections that enable the model to be extremely deep. Even though including skip connections is a common idea in the community

now, it was a revolutionary architectural choice and allowed ResNet to reach up to 152 layers with no vanishing or exploding gradient problems during training. There are different versions of ResNet, including ResNet-18, ResNet-34, ResNet-50, and so on. The numbers denote layers, although the architecture is the same.

To create a residual block, add a shortcut to the main path in the plain neural network, as shown in the Figure 21.

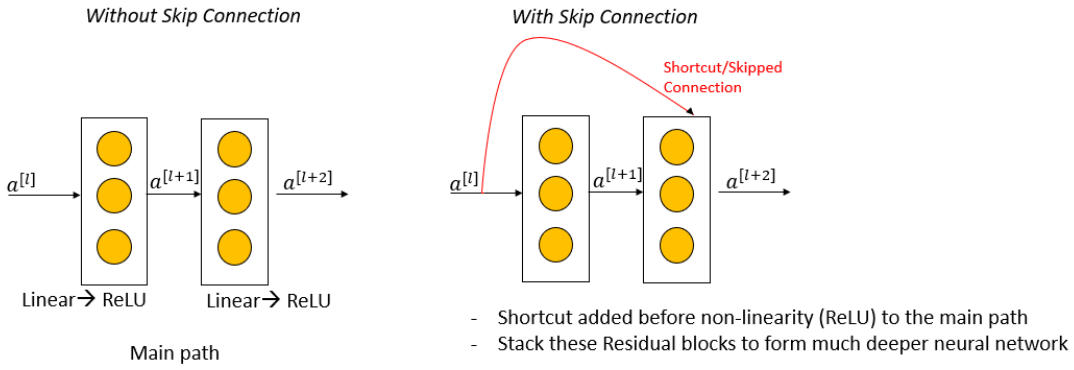


Figure 21: Neural Network with and without skip connection

ResNet introduces bypass connections in the network and allows the gradient to flow without getting multiplied with weight matrices several times[26].

In a branched network structure, if a layer leads to multiple modules, the gradients coming from all the modules are summed up and backpropagation continues with the chain rule. If the weight layers tend to have very small numbers (in the order of 10^{-5} or smaller) at least some of the gradients would be decreased to a millionth of the gradients in the upper layer. But the design of ResNet is providing identity connection by skipping some of the layers, thus making the gradients flow without being subject to any multiplications. That method is a very common architectural choice in recent networks having hundreds of layers.

The ResNet model consists of lots and lots of convolutional layers each having 3x3 masks (except the first layer with has 7x7 masks). There are a few variations of the model but ResNet-152 was the model that won ILSVRC in 2015.152 is the number of layers the model has. The network is over eight times larger than VGGNet while still having lower complexity.

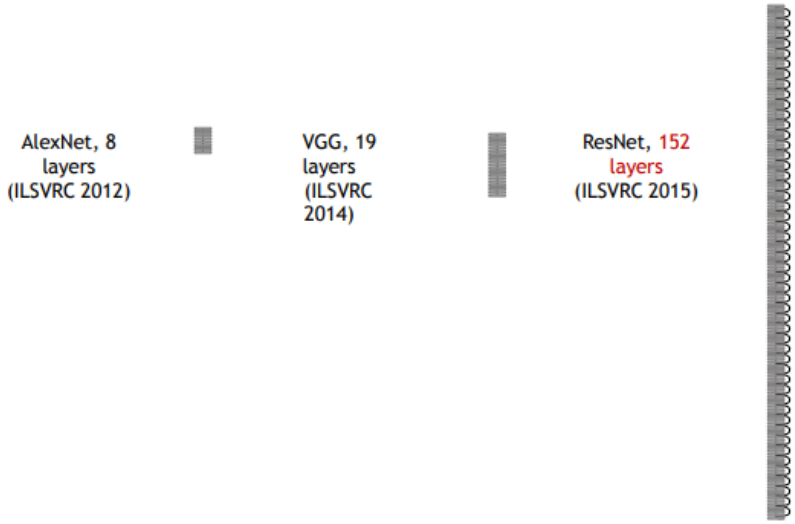


Figure 22: A comparison of layer depths

As you can see in the visuals above, ResNet-152 is absurdly deep and it is usually a good idea to load the model using deep learning library.

2.2 Pre processing

Data for this project consists of audio files. There are many ways to build machine learning systems that can analyze audio data. Yet, past researches shown that some methods appear to be more practical than others. In this project, we will try to explore a few ways to how to extract meaningful samples for training. Few things about audio files,

• all audio files sampled at 32 kHz, contains only mono(one) channel, and are compressed using the open OGG vorbis encoding.

• Training audio files are consisted of different length, ranging between a few seconds and several minutes.

The different lengths of training samples are one of the challenges in this dataset. On top of that, we also have to deal with weak labels, which means that we don't exactly know the precise timestamp of each bird call. If we look at one audio file, Figure 23 Include an audio waveform.

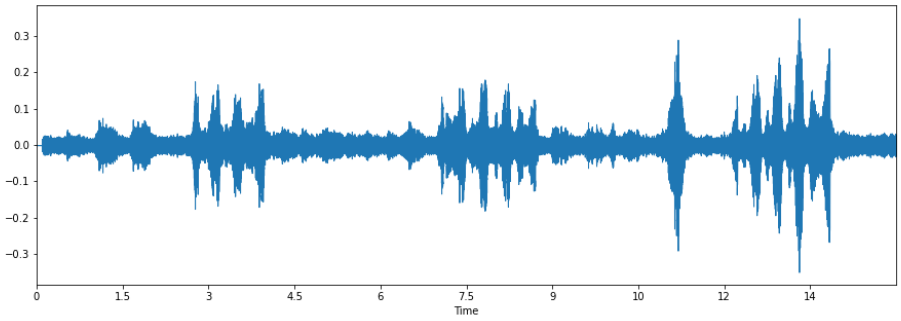


Figure 23: Audio waveform of a bird call

We could think to use raw audio signal for training (there are models using raw audio files), for example if we consider 5 seconds as input segment. It contains 160,000 samples (5 seconds*32,000 Hz). That would be mostly equal to 400 x 400-pixel image, which would be bit unusually large input for a neural network.

In the past researches we have seen that convolution neural networks (CNN) perform quite well for audio classification. But CNN needs 2D inputs. Because of that we can transform a particular audio signal into 2D representation using spectrogam. It is a visual representation of the audio signal. We have used Librosa library to extract spectrogram from raw audio.

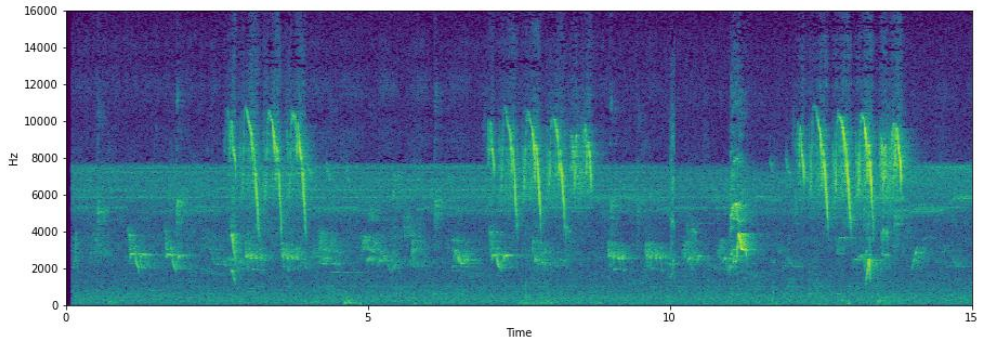


Figure 24: Spectrogram representation of the raw signal

In the Figure 24, we can clearly see foreground vocalization of the bird, each 5 second audio chunk contains one of them (this is not always the case). Since our initial sampling rate is 32,000

Hz, the maximum frequency that we can visualize in a spectrogram is 16,000 Hz (y-axis). With the initial parameters output shape of a 5 second chunk is 1025* 938. However, this representation still a very large input for a CNN.

We need to consider some parameters which we can consider during the STFT: “window length” (and with that implicitly also the number of frequency bins, "n_fft") and “hop length”. We tried different values. Say, we want to have an overlap between frames of 50%, so our “hop length” should be half of our “win length”.

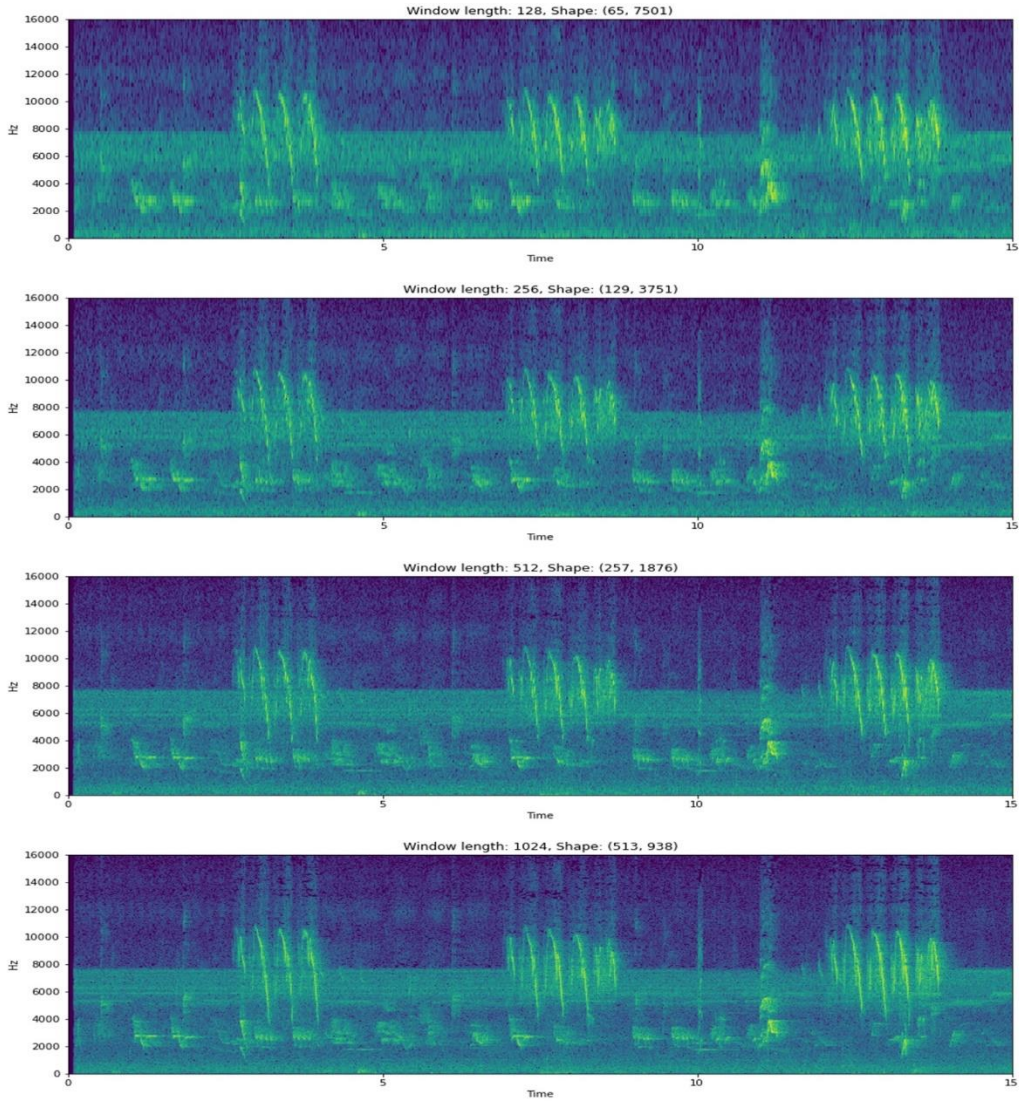


Figure 25: Resolution with window length (128, 256, 512, and 1024)

From the Figures 25, we can clearly see that there is a trade-off between vertical resolution (frequency resolution) and horizontal resolution (number of time steps). We can get a reasonable size for input using mel- scale to scale the frequency axis of our spectrogram. It is initially designed for human speech detection but it is also worked well for bird sound classification as well. For this transformation we have used Librosa library. We can set the number of mel bins we want to use and that number would eventually be our vertical resolution of the spectrogram. We also know that the hop length we choose is key for the width of the spectrogram, so we have to settle on a certain value. On top of that, we should probably process 7-second chunks of audio.

It is also important to consider the vocal and auditory range of birds. Most of the songbirds vocalize in between 1 kHz and 4 kHz. Sometimes, some bird species vocalize below that, and some significantly above. If we look at above birdsong audio, we can see that it is categorize into "high frequency" species, vocalizing up to 10 kHz. In general, we can probably limit the frequency range we want to include in a spectrogram between 500 Hz and 12 kHz. There could be few birds those who vocalize outside this range. But many birds will vocalize inside this range. In this project we are going to use default sample rate as 32 kHz. Figure 26 demonstrates what it would look like to extract 15-5 seconds, mel scale spectrograms with a target resolution of 64 x 256 pixels with Librosa.

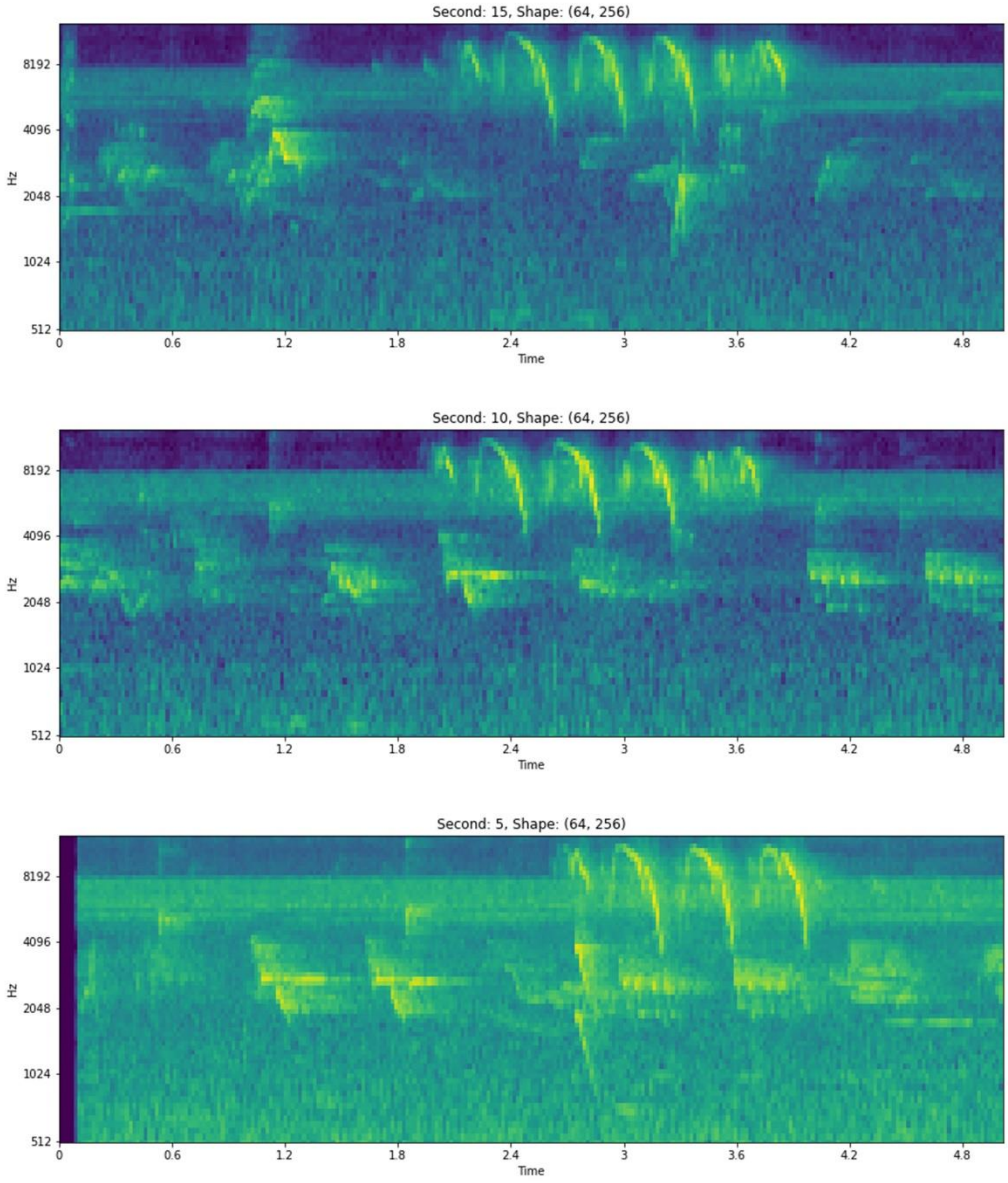


Figure 26: 15-seconds, 10-seconds and 5-second mel scale spectrograms with a target resolution of 64 x 256 pixels with Librosa.

We could use this processed data to train the network. However, many 5-second chunks might not contain the target species. This problem arises because of the weak labeling of the audio. There are many ways to solve weak labeling but in this we are not going to focus on that. For shorter audios we include padding and for the longer ones we have taken random starting point. Normalization has been done to images and saved them as precomputed mels. Below Figure demonstrates example of final input the model and this image size is 128*281.

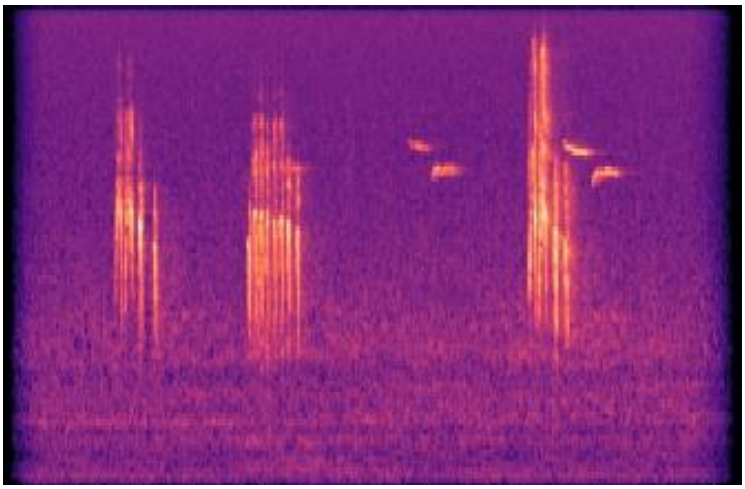


Figure 27: Final input to the model (Image size 128*281)

2.3 Training

Pytorch provides us tools to build various deep learning models in object-oriented fashion thus providing a lot of flexibility. A lot of the difficult architectures are being implemented and it is possible to build a state of art of classifier with a very small dataset using Pytorch.

Instead of a feed-forward Neural Net, we used a pre-trained ResNet model (Transfer Learning) to gain better accuracy. ResNet is a model called ResNet from Microsoft which won the ImageNet competition in 2015. It showed how deep networks can be made possible. It has a very complex architecture. We have downloaded the model and most of the modern deep learning frameworks makes loading a model easier. The ResNet model compromises of a bunch of ResNet blocks (Combination of convolution and identity block) and a fully connected layer. The model is trained on Imagenet dataset on 1000 categories, we have removed the last fully connected layer and added a new fully connected layer which outputs 397 categories which tells the probability of the image. Loss function helps in calculating how good is our model. We have used categorical cross entropy here. We have used Adam to optimize our weights with the gradients. In our case we update the weights of only the last layer.

CHAPTER 03

RESULTS AND DISCUSSIONS

Various species of birds have unique bird calls. These bird calls are distinct based on inflection, length, and context, meaning the same bird may have more than one call. A device that would analyze the signal and identify the bird based on the bird call could be of tremendous help to an ornithologist. This project proposed the development of machine learning model. The first task was to find or create a database of high-quality bird calls to use for identification. Using Xeno-Canto database, using various features of the bird calls of a certain species, a recorded bird call was identifiable as a species of bird.

This project is an important effort because ornithology is not always an exact science in the field; it is based on the interpretation of the scientist hearing the bird's song. A device that could quantitatively match signal waveforms would make the science more exact. Furthermore, bird watching is a hobby that many people enjoy. The ability to identify birds could increase the enjoyment of bird watching enthusiasts everywhere.

One challenge that was faced during the attempts to identify actual bird songs was recognizing all calls made by a particular species of bird. A bird's song can contain a significant amount of information including the bird's species, sex, individual identity, his territorial and reproductive status, and his probability of responding aggressively or sexually to a potential recipient. There is so much information that call variation is inevitable among birds and any one bird can have four or more songs in its repertoire. This was one of the greatest challenges because understanding how birds vary their songs was a significant roadblock to understanding how to differentiate species.

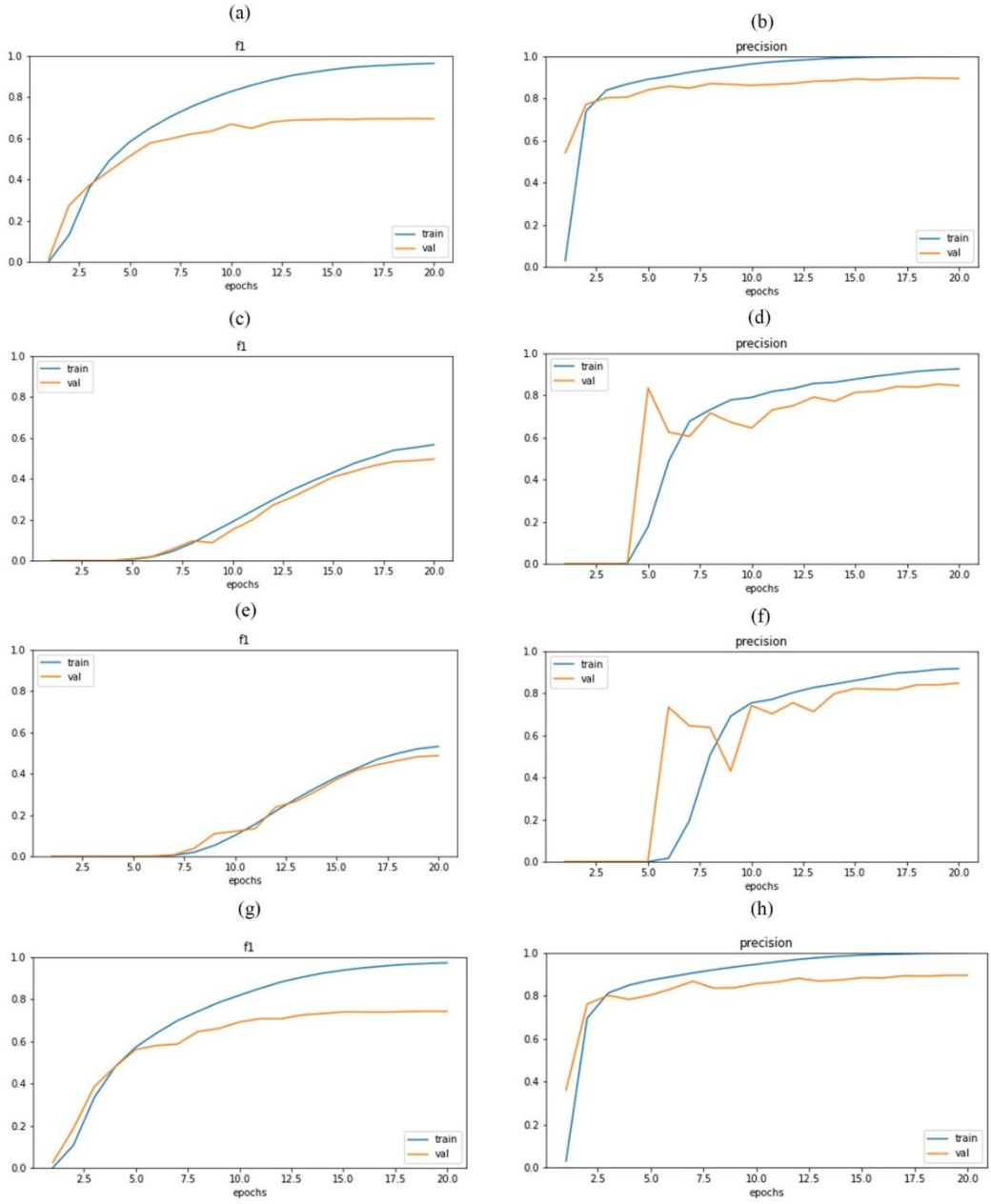


Figure 28: Training-Validation F1-score and Precision of the Transfer Learning Models. (a) Resnet18 f1-score. (b) Resnet18 precision. (c) Resnet101 f1-score. (d) Resnet101 precision. (e) Resnet152 f1-score. (f) Resnet152 precision. (g) resnext50_32x4d f1-score. (h) resnext50_32x4d f1-precision.

The main challenge of this project was to identify which birds are calling in long recordings, given training data generated in meaningfully different contexts. This is the exact problem facing scientists trying to automate the remote monitoring of bird populations.

In this project, we have studied the task of bird sound classification and identification using different deep learning-based models. The feature of bird sound that we mainly used was the Mel-spectrogram as it aligns with human's auditory system. The Mel-scaled spectrogram highlighted variation in lower frequencies. Several convolutional neural network models were implemented using transfer learning to achieve classification of bird sounds based on the audio representation. The models involved in this transfer learning task were Resnet18, Resnet101, Resnet152 and resnext50_32x4d. To have knowledge on the performance of different models we used evaluation matrix to evaluate the results with respect to classification. The classification was evaluated by accuracy in the beginning and the classification report consist of precision and F1-score (Figure 28).

In our experiments, we first conducted pre-processing on audio samples. The audio samples were pre-processed into Mel-spectrograms in a consistent dimension. We considered one spectrum size (128*281). We have selected specific number of mel bins and specific hop length to keep the dimensions of the preprocessed images equally. The lengths of sub-segment extracted from original audio were 7 seconds. Preprocessed images are saved as "npy" files to reduce the size. Then those images are loaded to each model. We have trained our dataset using transfer learning models. We have trained Resnet18, Resnet101, Resnet152 and resnext50_32x4d using a constant learning rate (0.005). These models are trained for 20 epochs and it took average 5/6 hours to fully train each model. These models we tested a set of hyper-parameters to investigate how the performance might be affected. Our results showed that the best classification performance was achieved by Resnet18, where average precision reached 0.62. Average precision of 0.57 was achieved by resnext50_32x4d, which is close to the best result. From the results we can conclude that among our experimental implementations, the most promising model to perform classification was Resnet18, with the input of Mel-spectrograms.

CHAPTER 04

CONCLUSION

Research directions that are worthy of further study and have the potential to improve performance mainly focus on two aspects; one is to investigate the signal processing and audio features, the other is to explore the neural network models. The experimental results showed that Resnet18 using combination of Mel spectrogram and its feature as input have achieved the best accuracy. This accuracy was higher than the accuracy of using only raw audio files. This is likely because the dynamical information of the spectrogram provided extra features about an audio sample. The results show that there is still a lot of room for improvements especially for the soundscape domain, which likely is the most important real-world application.

We can conclude that it is computationally feasible to classify bird sound using machine learning. Further investigations into the power consumption and practical considerations for applying on-edge Bird Sound Classification using Machine Learning seems to be worth pursuing.

CHAPTER 05

FUTURE WORK

The transfer learning models have the potential to be further improved. When we selecting audio chunk from each audio clip, there is a possibility to not to include each bird's sound in the selected audio chunk. For this problem it is possible to train model using different dataset and identify "no sound" audio chunks and "sound" audio chunks. Then we can improve the quality of the processed dataset and it will improve the overall accuracy of each model.

We make the models infer on 7 seconds chunks. Basically, the longer the input chunk, the better. The score saturates when it gets longer, so we chose 7 seconds which was in a good balance between memory consumption and performance. If there are enough computational power, it is possible to test different chunks of audio depending on the length of the average bird call.

The hardware design part of this project was largely abandoned due to time constraints, but there is a considerable amount of work to be done in that area. Trained model can be used to build a bird call identification device using TinyML and low powered microcontrollers. There are a lot of ongoing research going on this area and it is possible to further extend this research to develop an audio detection device.

Based on the need for wireless sensor systems that can monitor and classify enviro bird calls, it is possible to investigate sound classification directly on microcontroller-based sensor hardware. This on-sensor classification makes it possible to reduce power-consumption and privacy issues associated with transmitting raw audio or detailed audio fingerprints to a cloud system for classification.In a practical deployment of on-sensor classification, it is still desirable to collect some data for evaluation of performance and further training. This could be sampled at random, but an on-sensor implementation of Active Learning could be able to make this process more power-efficient. It is critical for overall power consumption to reduce how often on-sensor classification is performed. This should also benefit from an adaptive sampling strategy. For example, to primarily do classification for time-periods that exceed a sound level threshold, or to sample less often when the sound source changes slowly.

REFERENCES

(1) Henry, C., Lacuna Space aims to ride IoT wave with a 32-CubeSat constellation. *SpaceNews*, Available at: https://directory.eoportal.org/web/eoportal/satellite-missions/content/-/article/lacuna-constellation

(2) Rosenblatt, F., The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 1958, 65, 386.

(3) Nair, V., Hinton, G.E., Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th international conference on machine learning (icml-10)*, 2010, 807–814.

(4) Rumelhart, D. E., Hinton, G. E., Williams, R. J., Learning representations by back-propagating errors. *Cognitive modeling*, 1988, 5.

(5) LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998, 86, 2278–2324,

(6) Simonyan, K., Zisserman, A., Very deep convolutional networks for large-scale image recognition. In ICLR, 2014.

(7) Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., Imagenet: A large-scale hierarchical image database. *in IEEE conference on computer vision and pattern recognition*, 2009, 248–255

(8) Iandola, F.N., Han, S., Moskewicz, M.W, Ashraf, K., Dally, W.J., Keutzer, K., SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, *arXiv preprint arXiv:1602.07360*, 2016.

(9) Krizhevsky, A., Sutskever, I., Hinton, G.E., Imagenet classification with deep convolutional neural networks. *in Advances in neural information processing systems*, 2012, 1097–1105.

(10) Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint*, 2017, 1704.04861.

(11) Freeman, I., Roese-Koerner, L., Kummert, A., Effnet: An efficient structure for convolutional neural networks. *in 2018 25th ieee international conference on image processing (icip)*, 2018, 6–10.

(12) International Electrotechnical Commission, Audio recording - compact disc digital audio system. *Geneva: IEC*, 1999.

(13) Smith, J. O., Spectral audio signal processing. *Accessed Online Book*, 2011, Available at: https://ccrma.stanford.edu/~jos/sasp/.

(14) Allen, J., Short term spectral analysis, synthesis, and modification by discrete fourier Transform, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1997, 25, 235–238.

(15) Griffin, D., Lim, J., Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1984, 32, no. 2, 236–243.

(16) Anusuya, M., Katti, S., Front end analysis of speech recognition: A review. *International Journal of Speech Technology*, 2011, 14, 99–145.

(17) Huzaifah, M., Comparison of time-frequency representations for environmental sound classification using convolutional neural networks. *arXiv preprint arXiv:1706.07156*, 2017.

(18) Smith, S. W., The scientist and engineer's guide to digital signal processing. *California Technical Publishing*, 1997.

(19) Planque, B., Vellinga, W. P., Xeno-canto: Sharing bird sounds from around the world [online]. 2017, Available at: http://www.xeno-canto.org/.

(20) Tokozume, Y., Harada, T., Learning environmental sounds with end-to-end convolutional neural network. *Acoustics Speech and Signal Processing (ICASSP) 2017 IEEE International Conferenc*, 2017, 2721–2725.

(21) Tokozume, Y., Ushiku, Y., Harada, T., Learning from between-class examples for deep sound recognition. *arXiv preprint arXiv:1711.10282*, 2017.

(22) Dai, W., Dai, C., Qu, S., Li, J., Das, S., Very deep convolutional neural networks for raw waveforms. *in 2017 ieee international conference on acoustics, speech and signal processing (icassp)*, 2017, 421–425.

(23) Piczak, K.J, Environmental sound classification with convolutional neural networks. *in 2015 ieee 25th international workshop on machine learning for signal processing (mlsp)*, 2015, 1–6.

(24) Salamon, J., Bello, J.P., Deep convolutional neural networks and data augmentation for environmental sound classification. *CoRR*, 2016, Abs/1608.04363.

(25) Zhang, X., Zou, Y., Shi, W., Dilated convolution neural network with leakyrelu for environmental sound classification, *Proceedings of the 2017 22nd International Conference on Digital Signal Processing (DSP)*, 2017,1–5.

(26) He, K., Zhang, X., Ren, S., Sun, J., Deep Residual Learning for Image Recognition. *In Proceedings of CVPR*, 2016.

APPENDIX A