The Academic College of Tel Aviv-Yaffo School of Computer Science

Bat Echo Classification via Deep Learning

Submitted by Naor David

Under the supervision of Dr. Raid Saabne

Bat Echo Classification via Deep Learning

Abstract

Bats have been well known for classifying plants using echolocation. They emit ultrasonic waves and can recognize a nearby plant according to the echo returning from it, which is vital for the bat to complete tasks. We created a dataset consisting of depicted scenes of bats echolocating around plants using an acoustic simulation, which contains different plant types and their corresponding returned echos to the emitter. We devise an end-to-end solution for classifying a plant type from a single returning echo. We explore different variants of Convolutional Neural Network (CNN) architectures that outperform other baseline models, such as Support Vector Machine (SVM) and Multi-Layered Perceptrons (MLP). Different Machine Learning models are compared one against each other by measuring the model's classification performance on the generated echo dataset. We examine different data representations of a bat's echo, such as time and spectral representations, and observe how adequate is each of the representations to each classifier architecture. Additionally, we present improvements to the model that increases it is robustness, thus improving the overall practicality of the solution. Conclusively, we discuss the evaluation results and sketch an overall idea of incorporating the concept of Optical Flow in the context of echolocation (Acoustic Flow).

Acknowledgements

I would like to express my profound gratitude appreciation to my supervisor, Dr. Raid Saabne, of the Computer Science School at The Academic College of Tel Aviv-Yaffo, for his guidance, and for letting me have a free hand in this research.

I would also like to thank Prof. Yossi Yovel and Dr. Arjan Boonman, of the The George S. Wise Faculty of Life Sciences of Tel Aviv University, for consulting and assisting in this research.

Finally, I would like to thank my dear family for their constant support and unconditional love, and especially my late dear grandmother, who was always an inspiration and a guide to me.

Para minha luz, o campo magnético nunca deixará de existir. eu te amo.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Bat Echolocation

Bats are using echolocation to regularly detect, localize, and classify targets in their surroundings (Griffin (1958); Schnitzler et al. (2003)). Nevertheless, researchers have not yet fully learned how bats classify objects in their surroundings using echolocation Yovel et al. (2011) divided and classified bats into two broad categories: The first is foraging bats that have to recognize their food, like insects and fruits. The second category is orienting bats that have to classify objects in their environment, e.g., water surfaces, meadows, walls, vegetation, and forth. Additionally, they demonstrated that according to their complexity, small food targets often differ from large extended targets in the environment. "Small food items others comprise of a few reflectors and thus reflect rather simple echoes, whereas extended targets like vegetation produce complex echoes that are superpositions of many reflections from the many reflectors a plant contains(i.e., its leaves and branches)." Yovel et al. (2011).

Furthermore, they presented the importance of successfully classifying vegetation to bats survival: "Vegetation can be an indicator where fruit or insects are typically found (Kalko and Condon (1998); Thies et al. (1998); Schmidt et al. (2000); Levin et al. (2009)) and can serve as a landmark to facilitate navigation, roost finding, and habitat selection." Yovel et al. (2011)

In the sense of simple objects, such as pure spherical fruit, most of the previous classification work concluded. Such research focused primarily on measuring and describing bats' ability to discriminate between spectral and temporal variations. These experiments helped to explain the system's limits and expose its mode of operation; however, they generally based on generated artificial deterministic echoes.

Complex echoes of natural textures such as plants were either regarded as disturbing clutter or ignored, although, these echoes contribute to a coherent acoustic image of the bat's environment; The importance of complex echoes to bats is very high. Thus a high capacity model is needed to deal with classifications of these echoes.

Fortunately, recent advancements in the field of Deep Learning (LeCun et al. (2015)) made it possible to achieve state of the art results in numerous fields, mainly Computer Vision, Natural Language Processing, and Signal Processing. In the present paper, we devise an end-to-end solution that includes all the necessary steps to train an Artificial Neural Network (Hubel and Wiesel (1968)), which is capable of classifying complex echoes of different plants, where only a single echo needed for inference.

Before we describe the method, we introduce the Bat Echoes Dataset, which is generated by an acoustical simulation of echoes produced by simulated short frequency-modulated (FM), echolocation calls. The simulation consists of complex 3D objects depicting plants and points in space around the objects that represent emitter locations around the plant. This dataset mimics the real-world scenario of bat classifying a nearby plant by echolocation. Hyper-parameters of the dataset generation scheme control the dataset's number of unique plant types, number of plant specimens per plant type, and echo sample representation; This allowed us to experiment with different datasets of different echo characteristics.

We examine data exploration schemes that give the knowledge of echo data complexity and inter & intraclass variance by employing dimensionality reduction algorithms. As most of all Machine Learning models, the data needs first to be preprocessed before the training phase; thus, we incorporate data specific preprocessing algorithms applied on echo data and describe K-fold validation (Stone (1974)) regime for better generalization analysis.

The generated echo datasets are evaluated by different models using different data representations. The evaluation's metric and loss function values are collected and then compared. We conclude the adequacy between models and data representations, revealing the pros and cons of the models under evaluation against different datasets. Lastly, we discuss future works regarding the incorporation of Deep Learning techniques applied in echolocation domain tasks.

1.2 Former Studies

Previously proposed complex echo classification models incorporated methods from statistical signal processing, feature engineering, and linear machine learning models, such

as the Support Vector Machine (Or: SVM) (Cristianini and Shawe-Taylor (2000)).

Simmons and Vernon (1971) revealed that the absolute intensity of the echoes is a characteristic indicator which can discriminate objects of the same size with different shapes and objects of the same shape with different sizes. Afterward, Simmons and Chen (1989) suggested to compute the intervals between intensity peaks of the echo's impulse response (IR), and concluded that this method could explain he bats' ability to classify mealworms from disks. Yovel et al. (2008) has found that complex echoes created by ensonifying plants with an FM sweep bat-like signal "to contain vast species-specific information that is sufficient for their classification with high accuracy." They prove that from a batting point of view, it can "use a single echo received by one ear, with a surprisingly simple receiver, having a relatively low time resolution and no access to the impulse response, to extract the information required for classification."

Chapter 2

Bat Echoes Dataset

2.1 Echoes Dataset

For the entire scope of our research, a dataset of complex returning echoes is first needed; We present a dataset that depicts the real-world scenario of a bat classifying a plant echo.

For the entire scope of our research, a dataset of complex returning echoes is first needed; We present a dataset that is coherent to the real-world scenario of a bat classifying a plant's echo. It has controllable thus sufficient number of samples for statistical significance and generalization performance of a classifier, retaining the class balance between different plant species for unbiased predictions, and analyze the inter/intra class variance for exploring the effects on the classifier's performance. We describe and discuss different data representations of echoes samples, and what are the pros and cons of each data representation in terms of complexity and performance of the echo classification process.

2.1.1 Real World Scenario

The real-world scenario that the dataset represents is to capture returning echoes of different plants found in ensonifying space. An emitter depicting the bat is placed on different positions around the plants, emitting sound waves, such as FM sweep, towards the plant. An angle and a distance govern the positions of the emitter around the plant; An array of coupled sensors (e.g. two proximate microphones) would then capture the returning echoes from the plant reflectors with respect to the emitter position, mimicking the bat's ears. The plants would differ by their specie, condition (healthy, dry, dying, as such), and bear enough inter/intra class variance between plant specimens

to avoid redundancy. The plant's reflector architecture characterizes plant specie (e.g., the reflectors formation affects the tendency of dispersion of the echo, which affects the resulting echoes), the geometrical shape of the reflector, the reflector length, the average of distances between nearest reflectors, and the number of reflectors.

Such an experiment would highly accurately describe a bat classifying a plant of some specie according to its returning echo. Unfortunately, setting up such an experiment is expensive sensor-wise, and also calibrating the sensors for echo acquisition was found to be challenging on recent tryouts of the experiment. Furthermore, acquiring plant samples with the described properties above is laborious, as well as maintaining their physical conditions. Thus, we decided to simulate the above experiment through an acoustical simulation that portrays the real-world scenario we described now.

2.1.2 Echoes Acoustical Simulation

The following acoustical simulation simulates the real-world scenario that we presented. It consists of the whole necessary steps to acquire the impulse responses of echoes returning from an ensonified 3-D like object (Or: mesh). We will refer to the set of generated returning echoes from the acoustical simulation as the Echo Dataset or simply, the dataset. The following algorithm describes the dataset creation procedure, which is the collection of the outputted echoes of the acoustical simulation.

Figure 2.1 visualize the experimental scene of the acoustic simulation. The mesh consists of randomly oriented in space copies of some reflector mesh. The blue markers in the figure are the unique location of the emitter around the mesh.

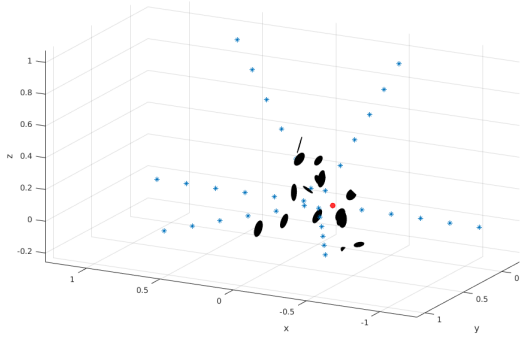


FIGURE 2.1: Scene Plot

Algorithm 1 summarizes the Echo Dataset Creation Algorithm.

Algorithm 1 Dataset Creation Algorithm

Input:
R - Average distance between reflectors
N_l - Number of leaves
F - Frequencies spectrum
A - Rotation angles range
D - Distance range
D_m - Emitters distance coefficient
N_s - Number of plant specimens per plant type

Output:
E_t - Echoes - Temporal Representation
E_s - Echoes - Spectral Representation

$E_t, E_s, E_{sp} \leftarrow \emptyset$
$L \leftarrow GenerateEmitterLocationMatrix(A, D, D_m)$
$P \leftarrow LoadReflectorMesh()$
for i=1 **to** N_s **do**
    $P_i \leftarrow RandomizePlantMesh(P, R, N_l)$
    for location in L **do**
        $frLeft, FRRight \leftarrow ComputeFrequencyResponse(P_i, location, F)$
        $irLeft \leftarrow IFFT(frLeft)$
        $irRight \leftarrow IFFT(frRight)$
        $E_f \leftarrow E_t \cup \{frLeft, frRight\}$
        $E_t \leftarrow E_f \cup \{irLeft, frRight\}$
    end for
end for

The simulation consists of the following hyper-parameters: The average of distances between each two nearest reflectors, number of leaves (Or: reflectors), the spectrum of frequencies to observe, rotation angles and distances ranges of the emitter to the plant (a range is the series of evenly-spaced numbers between an interval), a distance coefficient between the receivers (depicts the bat's ears), and the number of specimens for each plant type.

The simulation begins by computing all the positions of the emitter in space around the plant, given the distance and angle ranges. Afterward, a predefined 3-D mesh is loaded to represent a single leaf (Or: reflector). The mesh is duplicated according to the desired number of leaves to form an array of reflectors. Then, The reflectors are placed in space to have an average distance (denoted R) between them. Each leaf is then randomly rotated, for the reflectors array formation to mimic the structure of a plant. In this research, plant types differ by the average distance between reflector, R. The variation of this hyper-parameter relates to the real differences between different real-life plants. The unique positions of the emitter around the mesh dictate the inter and intra class

variance. In the extreme case of only one position of the emitter towards a random face of a mesh, the inter-class variance would be small, as in this case, potentially all of the returning echoes' distribution of all plant types can be modeled as a random Gaussian process. On the contrary, the intra-class variance would be high, as the chance of two plant specimens of the same type to have similar returning echoes from only one random face is low. Controlling the simulation's hyper-parameters enables us to produce different experimental scenarios with various inter/intra class variances, allowing us to investigate the dataset in greater depth.

Afterward, for each unique location around the plant and each frequency in the frequency range, a simulated FM sweep ensonify the plant, and an approximation of Helmholz equation (Wang and Wu (1997)) is used to compute the frequency response of the echoes. The overall spectra of the returning echoes are used to compute the impulse responses of the echoes; the spectra are transformed with the inverse Fast Fourier Transform (Cooley and Tukey (1965)) to reconstruct the impulse response signals.

2.2 Echo Data Representations

The acoustical simulation output the echoes in two different represnetations. temporal and spectral. The temporal representation corresponds to the impulse response signal, while the spectral representation to the frequency response spectrum. Each representation carries the same echo information in different ways, and the question which fits best to the classification is explored in this research. In the following subsections, the echo representations are defined and discussed.

2.2.1 Temporal Representation

The temporal representation is defined as the impulse response of the returning echo over time. Mathematically, it is a time series of continues sampled values, $X = [x_1, x_2, ..., x_T]$, wherein our case, each time sample is a complex number indicating the phase and amplitude of the sound wave at a discreet time.

The impulse response is approximately a superposition of a series Dirac deltas that may occur at some point in time (e.g. the echo can arrive after 20ms or after 100ms because of the different orientation of the bat towards the plant). The signal containing the impulse response is of fixed length. One advantage of using this representation is that it represented the raw data of an echo; It contains the actual returning echo, before any further transformations. In terms of data complexity, it has the disadvantage of high

number of dimensions (Or: features), as it contains all the time samples of the echo without any aggregating transformation such as FFT. Feature extraction is needed to identify temporal patterns occurring at the signal, and the time samples order needs to be taken into account.

2.2.2 Spectral Representation

The spectral representation is well known and used in the scientific community as a representation for observing frequent patterns occurring in a complex signal in time. The most common use case is to compute the spectrum of an acoustic wave and reveal the intensity as a function of the frequency. The spectrum can indicate modes and trends in the echo that is not visible on the impulse response. The resulting spectral intensities $F = [f_1, f_2, ..., f_k]$ can are used as features for later classification. Commonly, it has fewer features than the temporal representation due to the aggregation of time samples. The resolution bandwidth (Or: RBW) controls the number of time samples are aggregated into bins. The smaller the RBW, the higher the spectral resolution, which implies more frequencies are presented and are more distinguishable. If the RBW is too large, two frequencies can easily be combined into one, and it can be difficult to tell them apart. Thus, when using the spectral representation, the tuning of the RBW needs to be taken into account.

Chapter 3

Theoretical Background

In this chapter, we will cover all the necessary theoretical background needed for the understanding of the models tested. The First section describes the baseline models, which are the Linear Discriminant Analysis (LDA) (Ye (2007)) and Support Vector Machine (SVM) (Cristianini and Shawe-Taylor (2000)). The second section covers the theory of neural networks, today's state of the art tool of machine learning. The relationship between echo data representations and the performance of different variants of neural networks is discussed. Feature characteristics extracted by the neural network is also discussed as a mean for better understanding the adequacy between neural networks and echoes.

3.1 Baseline Models

The following classifiers described in this section were considered state of the art tool for machine learning applications before the rise of neural networks in the last decade. These models are linear classifiers that accept their input without a temporal context; The baseline models do not take into account their input's temporal dependencies between time samples (Or: features), as the models assume features to be independent and identically distributed.

Moreover, these models suffer from the curse of dimensionality (Bellman (1957)), which means that as the number of dimensions of the feature vector increases, the generalization performance decreases. Furthermore, these models are not salable to larger enough datasets, since their training complexity is highly dependent on the size of the dataset. For example, SVM's training algorithm complexity is known to be $O(\max(n,d)\min(n,d)^2)$, where n is the number of data samples, and d is the number of dimensions (Chapelle (2007)).

3.1.1 LDA

In 1936, Fisher made the first introduction of linear discriminant analysis (LDA) (Huberty and Olejnik (2006)).

The goal of LDA is to project a dataset into a lower-dimensional space while maintaining good class-separability. First, the mean vectors of each class are computed; Second, the scatter matrices are computed from in-between class and within-class covariance matrices; Then, the scatter matrices are decomposed to compute their eigenvalues and their corresponding eigenvectors. After, the eigenvectors with the top-k highest eigenvalues are selected to form a transformation matrix W, of dimensions $d \times k$, where d is the data's dimension; Finally, the data matrix X is transformed to a new subspace by applying the transformation $Y = X \times W$. The subspace is used to model a probability distribution of $P(y = k|X = x)$, meaning the probability of a sample x to belong to class k. We describe The model optimization process at greater depth in section 4.3.4.

3.1.2 SVM

According to Chauhan et al. (2018), Support Vector Machine (Or: SVM) (Boser et al. (1992); Cortes and Vapnik (1995)), also known as optimal margin classifier, is a binary linear classification technique in Machine Learning, that classify data samples by finding a separating hyper-plane between classes with optimal margin.

SVM has extended to multi-class problems for almost three decades. Knerr et al. (1990) were the first ones to describe techniques like One-versus-One, later followed by Kressel (1999); One-versus-Rest scheme was formulated by Vapnik (1998). Due to its limitation regarding the non-linearly separable data, SVM expanded the usage of kernels, which are mathematical functions able to efficiently convert the data from an input space to a feature space, where data can be linearly separated with a hyper-plane.

The kernel trick is formulated as:

$$K(x_1, x_2) = \phi(x_1)^T \phi(x_2)$$

Where ϕ is a non-linear kernel function, e.g., a radial basis function and x_1, x_2 are two feature vectors of two samples in the input space. Each variant of SVM receives a different terminology; Linear SVM, and non-linear (Or: Kernel) SVM.

SVM finds the best class separating hyperplane by optimizing the term:

$$\begin{aligned} \min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \\ \text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i, \\ \zeta_i \geq 0, i = 1, ..., n \end{aligned} \tag{3.1}$$

Where ζ_i are regulating slack variables added to the loss term, allowing borderline samples to satisfy the classification criterion, that would not satisfy the unregulated criterion, C is a regulating coefficient that controls the influence of the slack variables on the classification criterion. The model optimization process is later described in greater depth in chapter 4.3.3.

3.2 Neural Networks

In recent years, Artificial Neural Networks (Hubel and Wiesel (1968)), a machine learning model first proposed in the 1960s, came to high dominance in numerous scientific fields. Deep Learning is a sub-field of Machine Learning that mostly consists of variants of the Artificial Neural Network. Today, Deep Learning is the state-of-the-art tool of machine learning-related tasks: "Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art speech recognition, visual object recognition, object detection, and many other domains such as drug discovery and genomics." (LeCun et al. (2015)) Today's feed-forward neural networks consists of million of parameters (Simonyan and Zisserman (2014); He et al. (2016); Aaron van den Oord (2016)). The model capacity, which measures its capacity to handle complex dependencies between features and samples present in a dataset, is higher in orders of magnitude than of the baseline models explored. For comparison, SVM's computed decision hyperplane is defined by a vector of dimension $\mathbf{d} - \mathbf{1}$, where $\mathbf{d}$ is the number of dimensions of the feature vector, thus limiting SVM's ability to find patterns present in the data that are more complex and fine-grained than a linear combination of the input. As discussed in 2.2, echo data complexity is high, e.g., temporal dependencies between time samples in the impulse response may lead to a more challenging feature extraction process. Inter and Intra class variances also need to be taken into account; Plants of different types may have similar impulse responses, while plants of the same type may differ, possibly due to the angle of the bat to the plant's reflectors. The trade-off between achieving good training accuracy and generalization performance always exists when training a neural network. From one side, neural networks have

an excellent ability to memorize complex dependencies of the data it trains on. It can "remember" all returning echoes from all faces of a plant specimen, and then predict its type with high probability. On the other hand, when the network achieves perfect training accuracy, generalization performance usually decreases, which comes in the form of a reduced model's accuracy when applied on a validation set. This phenomenon, called overfitting, is one the most significant challenges in the Machine Learning field and is well researched and studied until today.

3.2.1 Perceptron

The first neural network devised and tested was the Perceptron model (Rosenblatt (1958)). The model consists of a single neuron unit, which is a biologically-inspired mathematical model. The neuron accepts feature vector $X = (x_1, x_2, x_3, ..., x_d)$ as an input, and carries learned parameters $W = (w_1, w_2, w_3, ..., w_d)$, and a bias term b. W is referred to as the neuron's weights, which directly affects the influence of features on the sample's classification process. Each data sample X_i is paired with a label y, in the binary classification case, can be $+1$ or -1. The Perceptron response to a feature vector is formulated as follows:

$$f(x) = W^T X + b. \tag{3.2}$$

The classification rule is is determined by sign(f(x)). To optimize the model when it miss-classifies a sample, the weights and bias term are updated as follows: Let $y = f(x)$

$$W^{t+1} = W^t + yx, \tag{3.3}$$

$$b^{t+1} = b^t + y \tag{3.4}$$

These updates correspond to a gradient descent (Ruder (2016)) step on the following loss function:

$$L(\hat{y}, y) = \begin{cases} -y\hat{y} & \text{if } \hat{y}y < 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

Where W^t is the weight vector at time step t of the training algorithm, $\hat{y} = \text{sign}(f(x))$, and y is the ground truth label of the sample. The loss function penalizes miss-classified samples, and when a sample is correctly classified, its values are 0. Note that the Perceptron model is a linear classifier, which in today's standards, is considered to be a weak classifier compared to non-linear models such as kernel SVM.

Figure 3.1 illustrates a Perceptron.

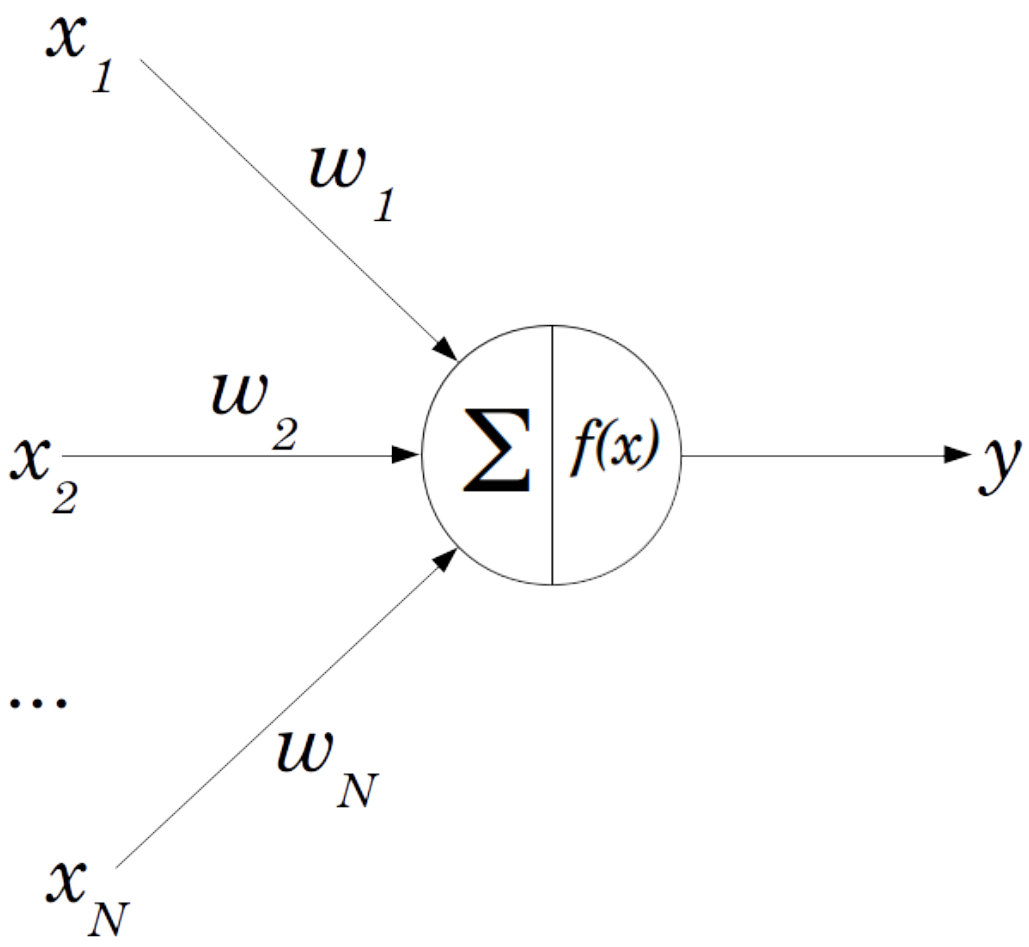


FIGURE 3.1: MLP Example

3.2.2 Multi Layer Perceptron

The next advancement of the Perceptron that came is the Multi-Layer Perceptron (MLP). First devised by Ivakhnenko in the mid-1960s (Farlow (1981)), A multi-layer Perceptron (MLP) is a deep, artificial neural network, which is composed of more than one Perceptron. It consists of an input layer that received an input, an output layer that predicts the input's class, and in between them, an arbitrary number of hidden layers exists.

The MLP model is also known as the Fully Connected Neural Network. A hidden layer is an array of Perceptrons units; Each layer accepts the outputs of its previous layer and outputs its computation to the following layer. Except for the input and output layers, each layer's neuron is connected to all neurons in the following layer; thus, the

name Fully Connected Neural Network. An MLP which consists of one hidden layer is capable of approximating any continuous function (Scarselli and Tsoi (1998)).

Figure 3.2 illustrates an MLP with a single hidden layer.

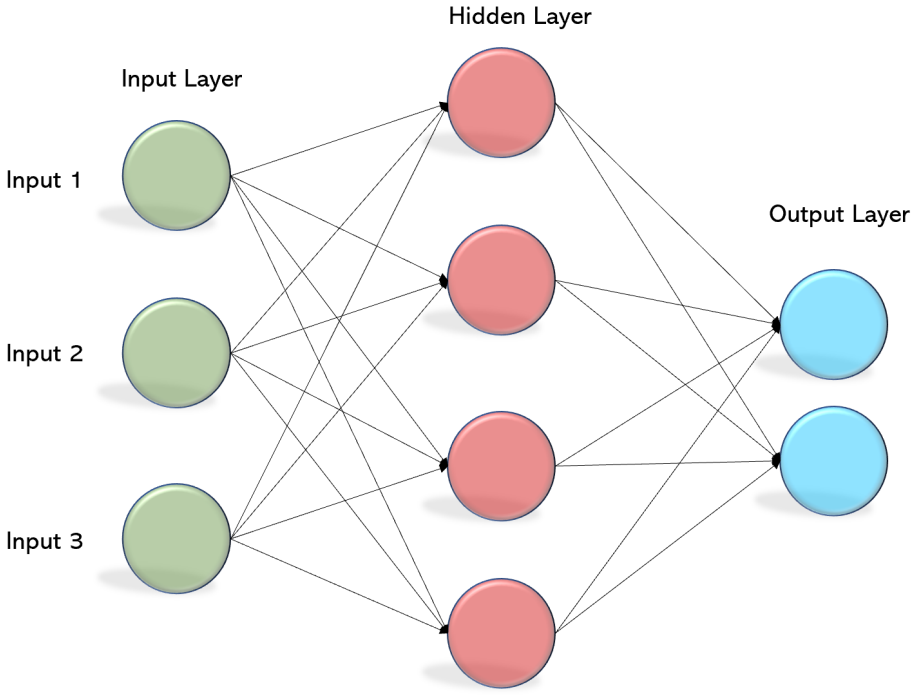


FIGURE 3.2: MLP Example

For a neuron j in layer i, its output, or activation, is defined as:

$$o_{ij}^{(l)} = \phi(\sum_{k=1} W_{kj} \cdot o_{i-1j})$$

Where W_{ki} is the weight between neuron k of the previous layer to neuron j of the current layer i, l is the number of neurons in layer $i-1$, and ϕ is a non-linear differentiable function, such as $\phi(x) = \tanh(x)$ (hyperbolic tangent), or $\phi(x) = \max(0, x)$ (also known as rectified linear unit, ReLU). ϕ is also named as the neuron's activation function. The computation done in the output layer of the network is regarded as the network's prediction on a sample. An activation function enables the neuron to map information from a previous layer to the next one; a linear activation function is easily computed and differentiated but is limited to a linear response. Besides, stacking multiple linear layers stem an overall linear model, as the compositions of linear operators on top of each other outputs a linear response.

On the contrary, non-linear activation functions map the input through a complex transformation, allowing the network to learn much more robust features from the data, potentially improving the network's generalization performance.

From a high point of view, the training procedure of a Feed-Forward Neural Networks is as follows:

1. Samples are fed into the network and propagated through the network.
2. Once the output layer yields predictions, the objective function loss term is computed to obtain the prediction errors of the output layer.
3. The resulting errors are transformed back through the network layers, to compute each neuron's corresponding error term, as a function of the weights. The procedure, called the backpropagation algorithm (Hecht-Nielsen (1992)), is the most common heuristic for optimizing neural networks.
4. The error terms are used to update weights and biases to minimize the network's loss term
5. The process is repeated until a maximum number of data epochs reached, or the desired metric reaches a certain value.

The model training and inference process are later described in greater depth in chapter 4.

3.2.3 Convolutional Neural Network

The Convolutional Neural Network (CNN), first devised in the 1980s (LeCun et al. (1999)), has become a predominant tool in machine learning-related tasks. Convolutional-based architectures as ResNet (He et al. (2016)) and WaveNet (Aaron van den Oord (2016)) have proven to show excellent performance at the fields of Computer Vision and Signal Processing, never achieved by previous MLP models. The complex arraignments inspired CNN architecture design in the visual cortex of the mammalian brain.

A convolutional layer, parameterized by kernel (Or: filter) size, and the number of kernels in that layer, the input of a convolutional layer can be an N-dimensional (aliased: N-D) vector. The most common cases are the 1-Dimensional and 2-Dimensional CNNs. In the case of 1-D CNN, the input is a 2-dimensional vector of dimensions (D, C), where D represents a 1-dimensional signal of length D, and C is the number of possible signal channels. For example, an impulse response signal (after transformed from the complex

to real numbers domain) is represented by a vector of dimensions $(D, 1)$, where D is the number of time samples of the impulse response, and a single feature (the amplitude over time) exists.

Each convolution layer consists of filters, which are spatially arranged vectors of relatively small size. When a convolution layer accepts an input, each filter is convolved with the signal in the manner of a sliding window; Each convolution output corresponds to a neuron of the next layer, forming a transformed signal that is dictated by the convolved filter. The number of channels in a convolutional layer output is determined by the number of filters applied in that layer, creating a feature map of the input.

Figure 3.3 illustrates a 1-D CNN with input dimensions of $(128, 1)$, and 97 filters of length 32.

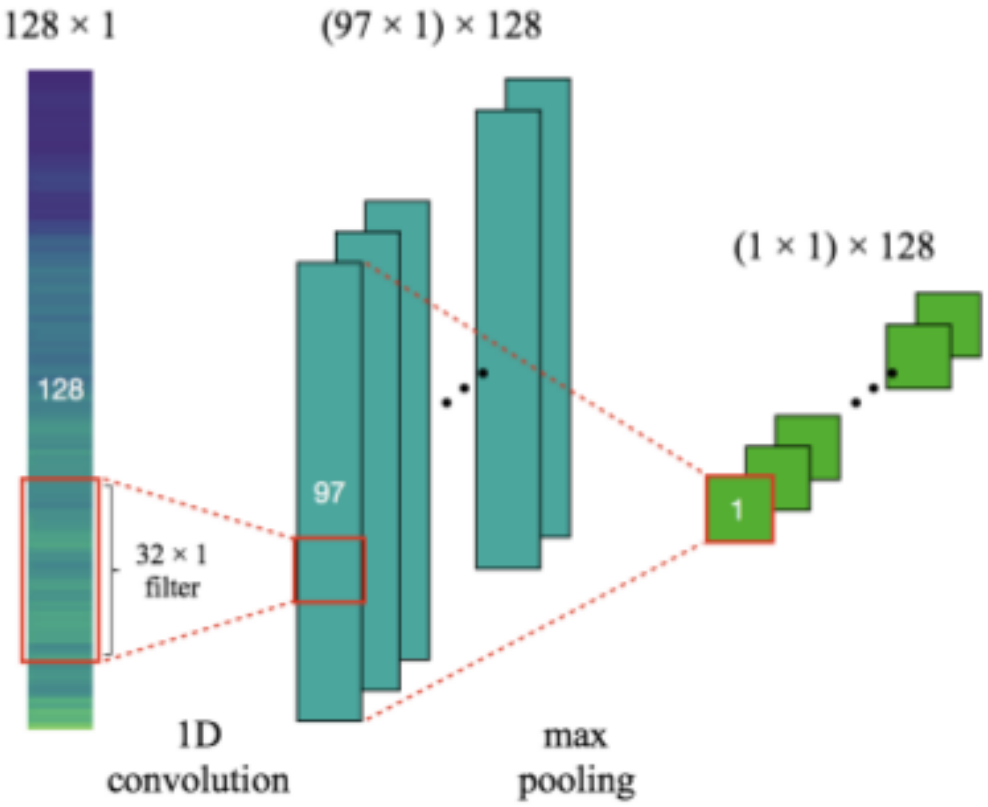


FIGURE 3.3: 1-D CNN Example

The convolution of a filter k on a signal f, as a function of t, is formulated as:

$$(k * f)_t = \sum_{\tau=-\infty}^{\infty} k_\tau \cdot f_{t-\tau}$$

The output of a convolutional layer at index t, with filter k applied on an input x, is given by:

$$o_{t,k}(x) = \phi((k * x)_t + b)$$

Where ϕ is a non-linear activation function, and b is a bias term.

Potentially, each filter recognizes a different occurring pattern in the signal, concerning its size. Furthermore, The learned features of a CNN form feature hierarchy, allowing the models to detect patterns at different scales and resolutions. Because the filter is convolved across the signal, the recognized patterns are invariant to their locations in the signal; This may be beneficial when processing a time series signal, where, for some reason, similar patterns occur randomly across a signal. Also, a learned filter is capable of capturing dependencies between proximate features that are convolved together with a given filter.

Training a CNN is similar to the training of an MLP, as described in the former subsection. Instead of updating the weights of a fully connected layer, the filter values are updated via backpropagation.

3.2.4 Fully Convolutional Neural Network

In a conventional CNN, the last layers are fully-connected layers, due to the nature of a Softmax classification layer, that requires a flat vector as an input. This means that the fed forward sample is first transformed from a feature map into a 1D vector of fixed length. This adds the constraint of a fixed input dimension; otherwise, the number of weights of the fully connected layer would not match the feature vector's dimensions. The constraints lower the practicality of using a "vanilla" CNN to classify echoes; Impulse responses of returning echoes may vary in length and time of appearance, as discussed in section 2.2.1. Thus, A robust model should cope with inputs varying in length.

In order to achieve this, we will apply the concept of Fully Convolutional Neural Networks (Or: FC-NN) (Long et al. (2015)). The difference between a regular CNN and FC-NN is that the latter does not contain any fully connected layers. The only fixed parameter in a convolutional layer is the number of filters (or channels) in it; This allows the layer's input to vary in feature vector dimensions, enabling the model to accept echoes of varying lengths, thus improving the model's practicality.

Usually, instead of using a fully connected layer, a global average pooling layer is used. For example, in a 1-D CNN model, given an input's feature map of dimensions $h \times c$, where h is the length of a signal, and c is the number of the input's channels. The feature map is reduced to a flat vector of length c by computing the average of each signal in each channel separately. Then, it can be fed into a Softmax classification layer, like in a regular CNN. This variant of the CNN makes the model more robust to real-world scenario data, where the echo signals may vary in their lengths.

3.2.5 Dilated Convolutional Neural Network

Another variant of the CNN is the use of dilated convolution instead of a regular convolution. A l-dilated convolution of filter k on a signal f is formulated as:

$$(k *_l f)_t = \sum_{\tau=-\infty}^{\infty} k_\tau \cdot f_{t-l\tau}$$

The filter is convolved with the input only at every l-the entry of the input, that enables the filter to have an exponentially bigger receptive field, while the size of the filter i.e., the number of learned parameters grows linearly.

Stacking dilated convolution layers provides the model with multi-scale context aggregation abilities, as the dilation rate potentially controls the granularity and resolution of the learned filters.

In their review, Yovel et al. (2011) discussed the effects of the distance on the returning echo. They concluded that the distance of the emitter to a complex object (such as a plant) affects the returning echo; They tied it with the widening effect of the projected beam, resulting in more reflection in the echo.

A dilated convolution may perform better on echoes from various distances and scales, as the learned filters detect multi-scale patterns in the signals.

Chapter 4

Method

This chapter thoroughly describes our proposed solution. First, we demonstrate the use of dimensionality reduction algorithms to visually explore echo data to get more intuition about the statistical nature of the echoes. Afterward, we describe the data preprocessing pipeline in detail, which includes feature scaling and a signal segmentation algorithm for extracting the impulse response.

The third section describes the learning procedure of the model - metric definition, loss function to optimize, training and validation arrangement scheme, and the model's training and inference procedures.

4.1 Data Exploration

Data exploration is the process of obtaining insights about processed data, usually with the help of dimensionality reduction algorithms, data visualizations, statistical properties, and others. By randomly sampling echoes samples, applications of the above can provide insights about the data's statistical properties.

4.1.1 Dimensionality Reduction

Principal Component Analysis Smith (2002) (Or: PCA) and t-SNE Maaten and Hinton (2008) algorithms are well-known dimensionality reduction algorithm. We apply them on the dataset to project the samples from their original feature space into a 2-dimensional space. PCA computes a linear projection, while t-SNE provides a non-linear one.

Figure 4.1 visualize an echo dataset of temporal representation and 3 plant types. The data is projected into 2-dimensional space with PCA.

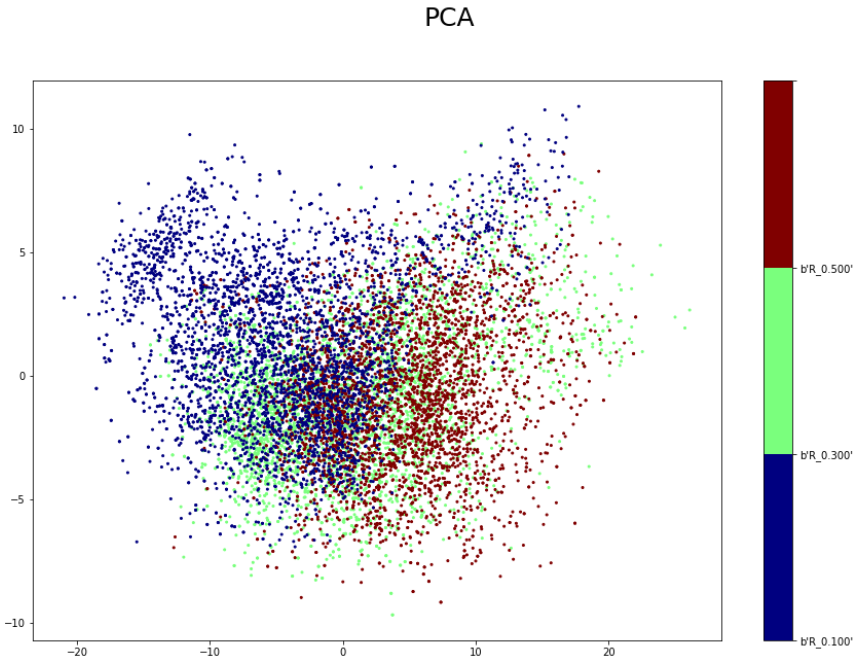


FIGURE 4.1: PCA Projection - 3 Classes, Temporal Representation

Figure 4.2 visualize an echo dataset of spectral representation and 3 plant types. The data is projected into 2-dimensional space with t-SNE.

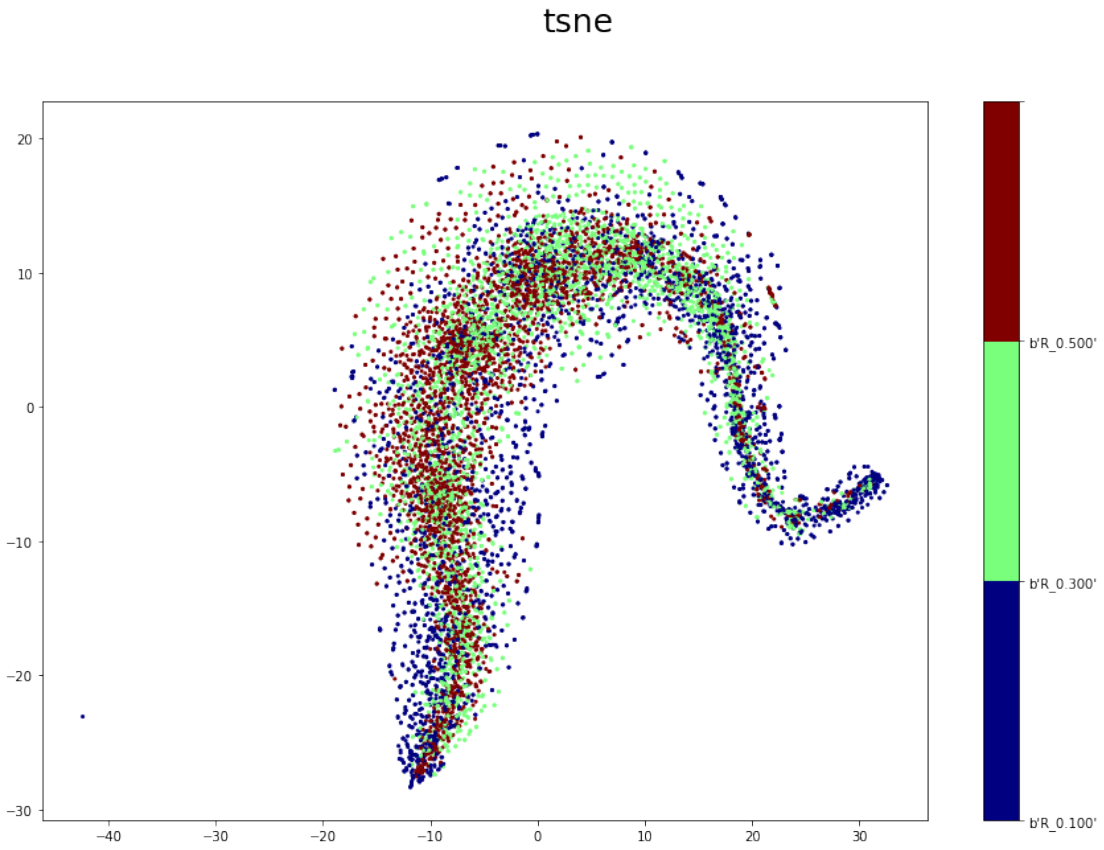


FIGURE 4.2: t-SNE Projection, 3 Classes, Spectral Representation

Figure 4.3 visualize an echo dataset of temporal representation and 2 plant types. The data is projected into 2-dimensional space with t-SNE.

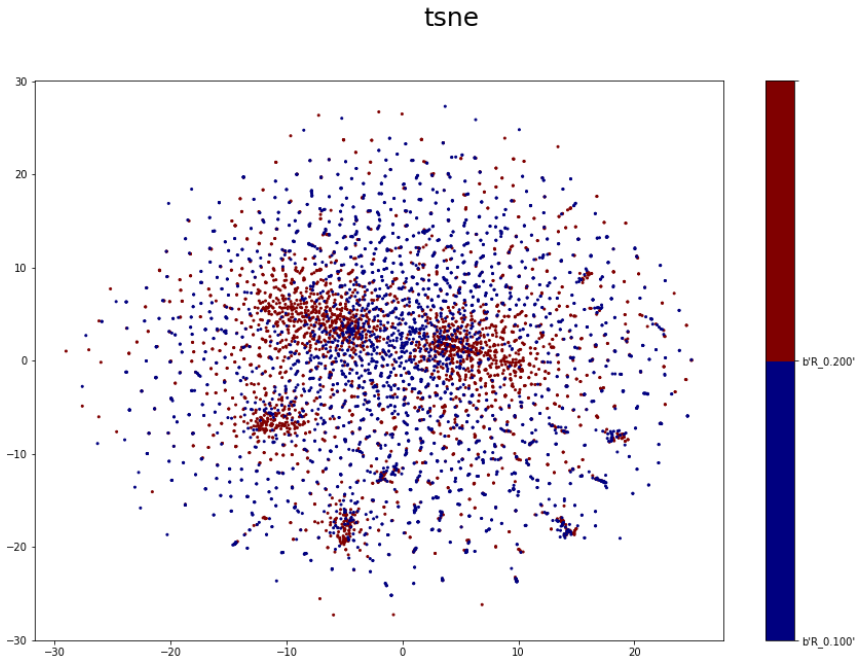


FIGURE 4.3: t-SNE Projection - 2 Classes, Temporal Representation

As seen from the figures, the data can not be trivially separated, in both linear and non-linear projections. This observation emphasizes the echo's natural complexity, low inter-class variance, and high intra-class variance.

4.1.2 Data Representations

4.1.2.1 Temporal Representation Plots

Figures 4.4 and 4.5 visualize randomly selected echoes, using the temporal representation.

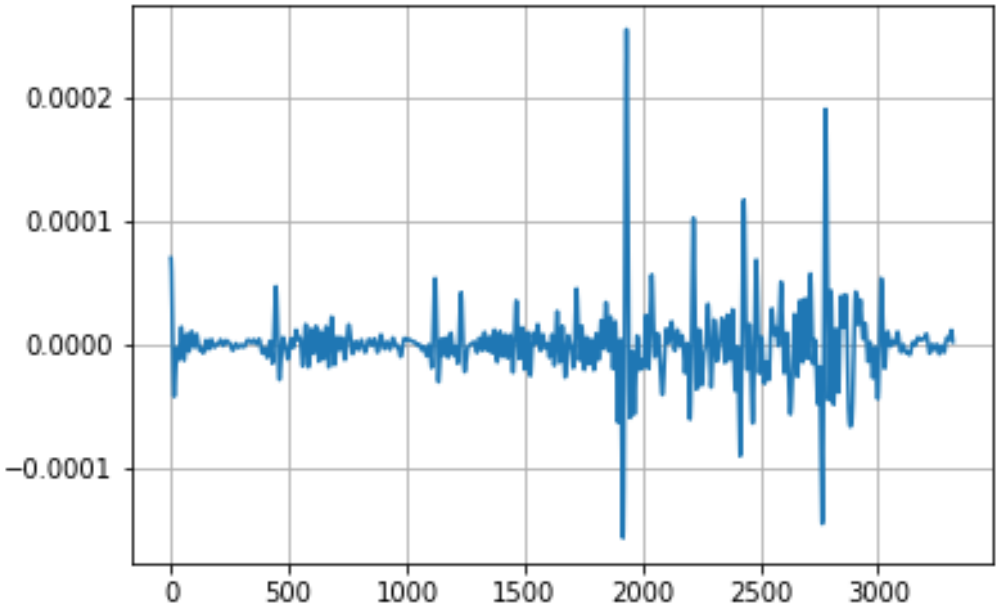


FIGURE 4.4: Temporal Representation Example 1

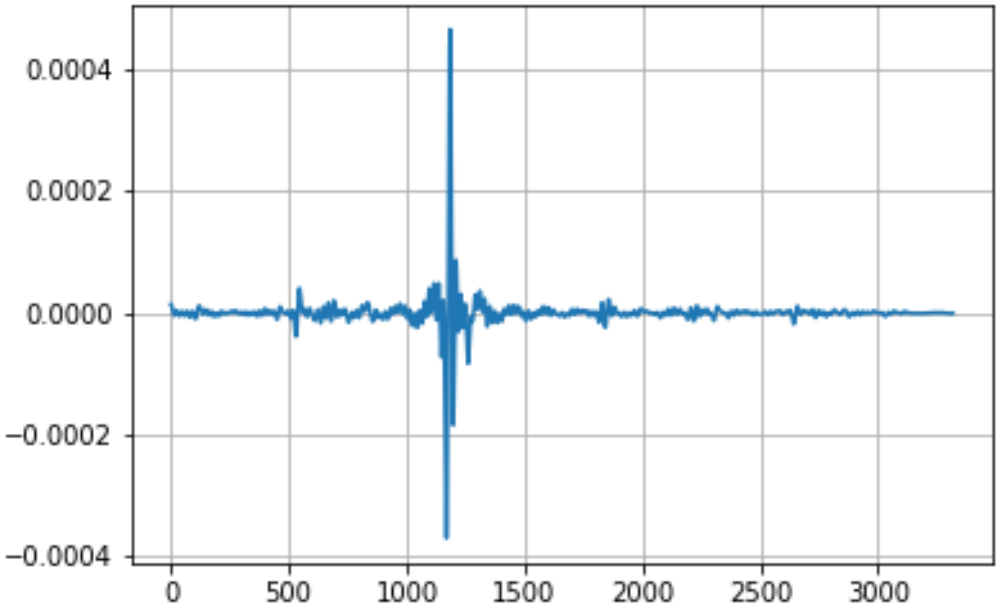


FIGURE 4.5: Temporal Representation Example 2

Figure 4.6 illustrates the difference between normal and log-scale of an impulse response.

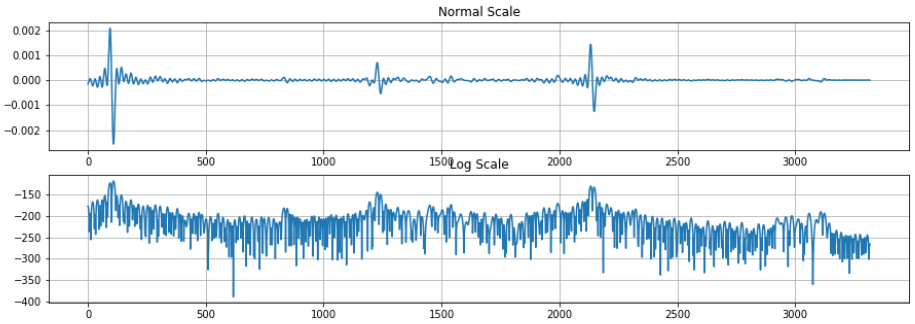


FIGURE 4.6: Temporal Representation Scale Difference

4.1.2.2 Spectral Representation Plots

Figures 4.7 and 4.7 plot the power spectrum of randomly selected returning echoes

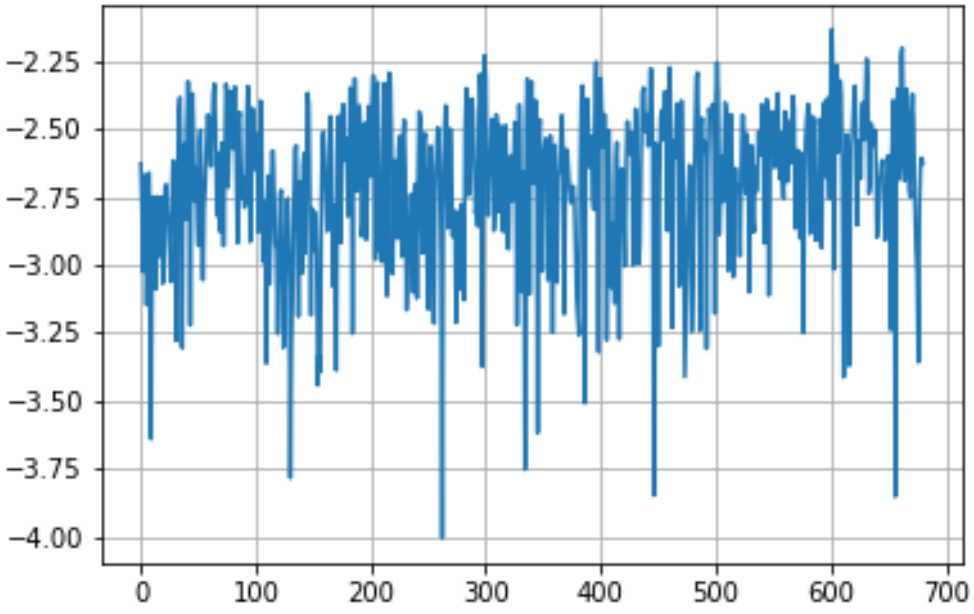


FIGURE 4.7: Spectral Representation Example 1

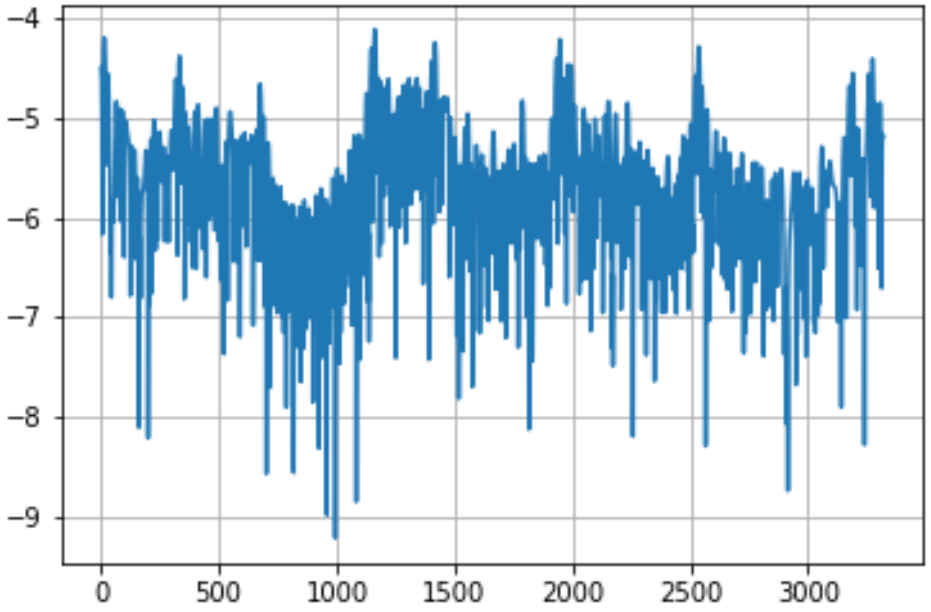


FIGURE 4.8: Spectral Representation Example 2

4.1.3 Impulse Response Delay

One noticeable phenomenon is that the impulse response can appear at any part of the signal in a delay. The echo signals differ by the time of appearance of the impulse response. One may also conclude that the signal is non-stationary. The temporal representation figures visually demonstrate the claim.

4.2 Preprocessing

Like every machine learning problem, data needs first to be preprocessed before presented to the model. Preprocessing affects the ability of models to learn. In our case, it involves feature scaling and signal segmentation.

4.2.1 Signal Scaling

The echo's time represented signal consists of the actual impulse response, arriving at some point in time in the signal, which will we refer to as the "impulse" part. The other part of the signal (which we will refer to as the "silent" part) is every interval in which the impulse does not occur.

We observe that the intensity of the impulse part differs in orders of magnitude from the silent part. Such a phenomenon led us to numerical instabilities in the middle of the model's learning process. In the context of optimizing neural networks, such instabilities may produce the vanishing gradient problem (Bengio et al. (1994)). To avoid the above, the signal is first moved from the complex plain to real values by computing its absolute value. Then, transformed to a logarithmic scale, and finally its time samples are scaled to $[-1, 1]$ range. The scale transformation, X_{scaled}, for a given range $[a, b]$ is given by:

$$X_{std} = \frac{(X - \min X)}{\max X} - \min X$$
$$X_{scaled} = X_{std} * (b - a) + a$$

We noticed that at the model's evaluation phase, the logarithmic scale preprocessing reduced numerical instabilities, which enabled a more stable learning process. Since the logarithm is a monotonic and non-negative function, there is no need to worry about the data distribution to change after transformed.

4.2.2 Signal Segmentation

As discussed in the last subsection, the signal consists of two main parts: the impulse response itself, which is a series of closed-interval Dirac deltas. The other part is a white noise signal which consists before and after the echo returned to the emitter. It happens due to the different distances and angles of the emitter to the plant. Researchers have already proved that bats can use a single echo, received by even one ear, to extract the information required for classification (Yovel et al. (2008)). This means that the bat's processing of an echo is invariant to the bat's current location of ensonification; thus, it is also invariant to when the impulse response occurs. To treat this, the model either has to learn time-invariant features, or the impulse response needs to be extracted from the overall signal, so the model will only accept the relevant impulse response. Algorithm 2 summarizes the impulse response extraction via signal segmentation

Our proposed algorithm extracts the impulse response by segmenting the signal into active and non-active intervals. Active intervals defined as time intervals, which starts with a peak (which will be defined afterward); it continues with some non-stationary signal (which is the duration of the impulse response). The interval ends at the start of a non-active interval, which can be seen as a white noise signal. Physically it represents the time where the echo is still traversing from/to the emitter. The algorithms start by finding the local maxima of the signal. We used MATLAB's `findpeaks` function to locate the local maxima. It receives as input two parameters - The minimum (absolute)

Algorithm 2 Segmentation Algorithm

Input:
D - Signal Dataset
ϵ - Error margin term
N_p - Chosen percentile of activity lengths
Output:
D_{seg} - Dataset of segmented signals

$D_{seg} \leftarrow \emptyset$
$activityLengths \leftarrow \emptyset$
$activityStarts \leftarrow \emptyset$
$D_{log} \leftarrow 20 \cdot \log_{10} ||D||$
for x **in** D_{log} **do**
    $peaksIndices \leftarrow findpeaks(x)$
    $peaksIndices \leftarrow findpeaks(x)$
    $firstPeak \leftarrow peaksIndices[0]$
    $lastPeak \leftarrow peaksIndices[-1]$ // last element
    $activityLength \leftarrow lastPeak - firstPeak$
    $activityStarts[i] \leftarrow firstPeak$
    $activityLengths[i] \leftarrow activityLength$
end for
$T \leftarrow percentile(activityLengths, 20) + \epsilon$
for x **in** D **do**
    $x_{seg} \leftarrow trimSignal(x, activityStarts[i], T)$
    $D_{seg} \leftarrow D_{seg} \cup \{x_{seg}\}$
end for

height a peak has to have to be recognized as such, and the minimum distance (in indices) peaks have to have to be counted. Due to the substantial difference in magnitudes of the impulse response and the white noise signal, configuring these parameters was trivial. The peaks match the starting time indices of active intervals accurately. The first and last peak indices are retrieved to form the interval in which the impulse response is found. Before the extraction can begin, all must agree on the same impulse response length; This is due to the requirement of the baseline models to have the same input size (Described in section 3.1). Since impulse responses vary in length and may occur near the end of the overall signal, we heuristically calculate it by choosing the 20th-percentile of the sorted impulse response lengths. Then, we trim each signal to the interval $[t_s, t_s + \tau + \epsilon]$, where τ denotes the computed trim interval and ϵ is a margin for error. Properly tuned ϵ value will compensate edge cases where the value of τ resulted in the incomplete impulse response extracted. The percentile value for computing τ was configured by trial and error by visually observing the resulted trimmed signal.

4.2.3 One Hot Encoding

Each echo in the dataset is associated with a plant type. Mathematically, it is represented by a categorical random variable that its outcome is one of the plant types. The outcome in our case is transformed to a indicator vector, representing the sample's class; For the set of possible classes $C = \{c_1, c_2, ..., c_i, ..., c_n\}$, suppose a sample belongs to class c_i. Then, its one-hot encoding is set to the vector $[0, 0, ..., 1, 0, .., 0]$, where the value of the i-th index is 1, and the vector's length is n. Using one hot encoding enables a Softmax loss function to be analytically computed, and it is more suitable to describe a plant type than to use ordered integers to encode different plant types.

4.3 Learning Phase

In this section, the learning phase of the classifier is described in detail. The measured metric is defined for the later Experimental Results chapter, and the loss (Or: Objective) functions used to optimize the models are defined and discussed. The training regime consists of splitting the data into training, validation, and testing sets. The model is first trained on the training set, and after each training epoch, the model evaluates the validation set. In that way, we can monitor the generalization performance of the model. Finally, the model evaluates the held-out test set.

4.3.1 Metric

Accuracy is a metric for evaluating classification models. Informally, accuracy is the fraction of predictions our classifier correctly classified. Formally, accuracy has the following definition: $Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$. In the multi-class classification, where more than two class labels exist, the accuracy metric is reliable when the class balance is maintained. In our case, we have control of the perfect class balance of class specimens per plant type. The different models were evaluated using this metric.

4.3.2 Loss Functions

Loss functions are differentiable functions of the model's learned parameters. They express how much a model is wrong in the context of classification.

4.3.3 SVM

As discussed formerly, SVM's goal is to maximize this distance between a decision hyper-plane $\mathbf{x}^T\mathbf{w} + b = 0$ to the closet samples to him in space, or, equivalently, to minimize the norm $||\mathbf{w}||$. The hyper-plane is parameterized by w, and b is a bias term. Regarding the problem of finding the optimal decision plane in terms of $\mathbf{w}$ and b can be formulated as:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}||\mathbf{w}||^2 \qquad \text{(objective function)} \\ &\text{subject to} && y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq 1, \;\; \text{or} \;\; 1 - y_i(\mathbf{x}_i^T\mathbf{w} + b) \leq 0, \quad (i = 1, \cdots, m) \end{aligned}$$

Because the objective function is quadratic, this constrained optimization problem is called a quadratic program (QP) problem.

4.3.4 LDA

In Least Discriminant Analysis, first we compute the within-class scatter matrix S_W and The between-class scatter matrix S_B of the data as follows:

$$S_W = \sum_{i=1}^{\mathbf{K}} S_i$$

where

$$S_i = \sum_{x \in D_i}^{\mathbf{n}} (x - m_i)(x - m_i)^T$$

where m_i are the class means, $\mathbf{K}$ is number of classes, and D_i is are samples of class i.

$$S_B = \sum_{i=1}^{\mathbf{K}} N_i(m_i - m)(m - m_i)^T$$

where m are m_i are the class means, $\mathbf{K}$ is number of classes, N_i is the the sample number of class $\mathbf{i}$. Next, we solve the generalized eigenvalue problem for the matrix $\frac{\mathbf{S_W}}{\mathbf{S_B}}$ to obtain the linear discriminants. The eigenvalues of the matrix above represents a new feature sub-space where data samples are linearly separable. A multivariate normal distributions with a common covariance matrix Σ and different mean vectors μ_k for $1 < k \leq \mathbf{K}$ classes.

One can compute the probability of a sample X, given a class k as a multivariate normal distribution: $P(X|y = k) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(X - \mu_k)^t\Sigma^{-1}(X - \mu_k)\right)$ Where d is the input's number of features.

We then use Bayes rule to compute the probability of a sample x to belong to class k:

$$P(Y = k|X = x) = \frac{P(X=x|Y=k)P(Y=k)}{P(X=x)}$$

$P(X = x)$ does not affect the above probability (not a function of k); thus, it is ignored from the computation. Class prior probability $P(Y = k)$ is estimated by the fraction of training samples of class k. Practically, the log probability $\log P(Y = k|X = x)$ is computed. To classify a sample, the log probabilities ratios of different classes are compared to estimate to most likely class of a sample: $\log \frac{P(c_1|x)}{P(c_2|x)} = \log P(c_1|x) - \log P(c_2|x)$ for classes c_1, c_2.

4.3.5 Neural Networks

In the case of neural networks, the optimization objective function is the categorical cross-entropy defined as:

$$-\sum_{i=1}^{C} y_{o,c} log(p_{o,c})$$

Where $y_{o,c}$ is a ground truth binary indicator if class label **c** is the correct classification for observation **o**. The loss function is optimized towards a local minimum via back-propagation, as discussed in section 3.2.2. The output layer is set to use the Softmax activation, which approximates the probability distribution of a sample to belong to each class, The Softmax's output vector sums to 1. The output of neuron i in a Softmax layer represents the probability of a sample x to belongs to class i, and is given as:

$$p_i = \text{Softmax}_i(x) = \frac{\exp(x_i)}{\sum_{j=1}^{K} \exp(x_j)}$$

For $i = 1, 2, 3, ..., K$, x_i is the i-th entry of the vector x, and K is the number of classes. The network's final prediction is the index of the highest probability of the Softmax output vector:

$$y_{pred} = \arg\max (p_1, p_2, ..., p_k)$$

4.3.6 Training and Validation Sets

After preprocessing the dataset, data is split into training, validation, and testing sets. Training data is used by the model to optimize its parameters in a supervised manner; the model computes its loss function by the ground truth labels and optimizes accordingly. Validation sets are used to monitor the generalization performance while training the

model strictly; For example, After each training epoch (Iteration through the entire dataset) of the model, the model's accuracy is computed on the training and validation sets. A well generalizing model is associated with high validation accuracy in addition to high accuracy on the training set. Such a model is more probable to classify unobserved samples correctly. The test set is evaluated once model training has completed.

4.3.7 Training Procedure

In order to gain the statistical significance of the classifier performance, we apply the concept of cross-validation with the use of K-fold validation (Stone (1974)). The approach involves dividing the set of observations randomly into K groups, or folds of approximately equal size. The first fold is treated as a held-out validation set, and the model trains on the remaining $K - 1$ folds (James et al. (2013)). Training includes optimizing the model's loss function as defined in 4.3.2 when fed with the echoes of the training set.

For each fold out of K folds, we train on the $K - 1$ folds not held out for validation and evaluate at the validation fold. The accuracy of training and validation groups is collected throughout the cross-validation. Finally, the model evaluates the held-out test group.

4.3.8 Inference Procedure

To classify an unobserved example, in our case, a bat classifying a returning echo of a plant it encounters. The model outputs is a vector $P = (p_1, p_2, ..., p_K)$, where $\mathbf{K}$ is the number of class labels. The model assigns probabilities to samples, by outputting a probability vector $P = (p_1, p_2, p_3, ..., p_K)$, where K is the number of plant types, and p_i is the probability of an echo to belong to plant type i. he model final prediction is the most probable class:

$$y_{pred} = \arg\max (p_1, p_2, ..., p_k)$$

Chapter 5

Experimental Results

This chapter covers the evaluation of all models described in chapter 3 on the Echo Data Set 2. The experiment's setup, which includes the evaluated datasets' generation process, is described in the first section. Afterward, the training procedure of both baseline models, described in section 3.1, and the different neural networks described in section 3.2, are applied to the generated datasets.

5.1 Experimental Setup

5.1.1 Hyper-Parameters

In order to evaluate the various models discussed, we first generate datasets of returned echoes, as described in 2.1.2. As a reminder, the acoustical simulation has the following hyper-parameters: average distance between reflectors (denoted R), number of leaves (denoted N_l), desired frequencies spectrum denoted F, rotation angles range (denoted A), distance range (denoted D), emitters distance coefficient (denoted D_m), and number of plant specimens per plant type (denoted N_s).

We generated two datasets, varying by the hyper-parameters of the acoustical simulation, to evaluate the different models. The outputs of the acoustical simulation is a multi-dimensional vector of dimensions (N_c, N_s, N_p, N_d). N_c denotes the number of distinct plant types (Or: classes), N_s is the number of plant specimens per plant type, N_p is the number of unique positions of the emitter around the plant, which is a function of the number of unique angles and distances from the emitter to the plant, and N_d is the dimension of the echo's representative vector, and is set according to the dimension of the selected echo data representation (as described in section 2.2).

After the simulation completes, the preferred data representation is set for the rest of the experiment. For example, if the simulation consists of 3 plant types, five plant specimens for each plant type, 30 unique emitter positions around the plant, and where the echoes represented with the temporal representation, the echoes dataset's dimensions would be $(3, 5, 30, 16385)$.

5.1.2 Datasets Generation

In this research, plant types (Or: classes) differ by the average distance between reflectors, R, and the chosen frequency range of the returning echo's spectrum is from 12 Khz to 80 Khz, where the frequency resolution bandwidth is set to 100Hz, defining 681 unique frequencies. N_d, the length of the input vector of the network is set to 681 for the spectral representation, while for the temporal representation, it is set to the length of the extracted impulse response (after being preprocessed by the signal segmentation algorithm described in 4.2.2).

Two datasets were created for model evaluation. The first dataset denoted for further references as D_{simple}, consists of two different plant types. The first plant type's R is set to 0.1, while the second plant type's R it is set to 0.2. For both groups, there are ten plant specimens. Two hundred different unique emitter positions are generated according to ten evenly-spaced angles and distances. The corresponding dimension of the dataset is $(2, 10, 200, N_d)$.

This dataset represents a simple case of two plant types, a relatively low number of specimens, and a relatively high number of emitter's positions. We evaluate the baseline models and the neural networks to measure the performance difference between the groups of models.

The seconds dataset, denoted for further references as $D_{complex}$, consists of three plant types, generated from three different R values: 0.1 , 0.3, and 0.5. In each group, there are 100 specimen and 80 different unique emitter positions for each specimen. Four evenly-spaced angles and ten evenly-spaced distances dictate the positions from the emitter to the plant. The corresponding dimension of the dataset is $(3, 100, 80, N_d)$.

This dataset represents a more complex scenario to classify; Compared to D_{simple}, there are more specimens per plant type and fewer emitter positions around the plant, which makes the classification task more complicated. Overall, D_{simple} consists of 8,000 echo samples, whereas $D_{complex}$ consists of 48,000 echo samples.

Due to computational limitations and algorithmic complexity reasons, only the neural network models were evaluated on the dataset. Baseline models' complexity, as discussed in 3.1, is greatly affected by the dataset's sample count and sample's feature vector dimension; thus, it can not train on this dataset. On the contrary, neural networks' current practical implementations (**?**; **?**) enables optimization of complex neural networks on large datasets. It is possible mainly due to the recent advancements of hardware acceleration technologies, such as NVIDIA's CUDA toolkit.

5.1.3 Training and Validation Sets

After the dataset is generated, the echo dataset split into train, validation, and test sets. First, 25% of specimens are randomly selected and held out and used as a test set. The remaining samples are split by K-Fold validation, as described in 4.3.7. As mentioned in 4.3.6, the train set is randomly split into K validation folds, where each validation fold consists of a training set and a held-out validation set. The model repetitively trains on each validation fold's training set and evaluates its validation set. Finally, the model evaluates the held-out test set. All metric results of the model evaluation on the validation folds are saved for further analysis.

It is worthwhile to mention that the splitting procedure of training and testing sets is done in a way such there are no plant specimens of the same plant type that occur in both the training and validation sets. That is, echoes are divided into training and validation sets by randomly selecting a subset of plant specimens, retrieve all of their echoes samples for training, while the validation set's specimen's echoes are held out for testing the model. This behavior depicts the scenario where a bat classifies an unseen plant specimen, potentially from one returning echo from it. The fractions of samples belong to each training, validation, and testing sets are configured as hyper-parameters.

5.2 Training Phase

The whole method described in 4 is applied to the generated datasets described earlier. Each model under test optimizes differently, using its objective function. The next following subsections will describe the hyper-parameters, optimization methods, and objective functions used in each model for the training procedure.

5.2.1 Baseline Models

5.2.1.1 SVM

A non-linear SVM with the radial basis function kernel was tested. The following hyper-parameters that are required for an SVM model (3.1.2) are described in table 5.1.N denotes the length of the input's feature vector, and Var(D) denotes the variance of the samples.

TABLE 5.1: SVM Hyper-Parameters

Parameter	Value	Description
C	1	Penalty parameter of the error term
Gamma	$\frac{1}{N*Var(D)}$	Kernel coefficient

5.2.1.2 LDA

The LDA model is set with the following hyper-parameters required for the model (3.1.1 are described in table 5.2.

TABLE 5.2: LDA Hyper-Parameters

Parameter	Value	Description
Solver	SVD	Eigenvalue decomposition algorithm
Tolerance	1×10^{-4}	Threshold used for rank estimation in SVD solver.

5.2.2 Neural Networks

All Neural Networks described in section 3.2 were evaluated using the hyper-parameters described in table 5.3. Optimization is carried via the backpropagation Hecht-Nielsen (1992) algorithm. The Fully Connected Neural Network that shall be referred to as FC-NN consists of 4 densely connected layers, followed by a Softmax classification layer, and has $\sim 258K$ learned parameters. Its architecture is described in figure 5.1, where each block in the figure describes a layer of the network, and its input and output sample dimensions are provided.

The 1-D Convolutional Neural Network, which shall be referred to as 1D-CNN, consists of 4 blocks, where each block consists of a convolution layer, followed by a max-pooling layer. The convolution layer uses the causal padding scheme, which means that a learned filter at time step t can only see inputs that are no later than t, and is achieved by padding the input's beginning with $k-1$ zeros, where k is the filter's size. This property implies that there is no information leakage from future signal samples to previous ones when optimizing the network, and is most useful when the data representation is temporal. The network's architecture is described in figure 5.2, which includes The filter size and the number of channels in each convolution layer. The network has $\sim 66K$ learned parameters.

The Dilated 1-D Convolutional Neural Network that shall be referred to as Dilated-CNN shares the same architecture and number of parameters of the 1D-CNN model, except that each convolution layer has a dilation factor of 2.

TABLE 5.3: Neural Networks Hyper-Parameters

Parameter	**Value**	**Description**
Number of epochs	40	Amount of times the whole dataset is shown to the network
Batch size	32	Number of samples to be propagated through the network
Activation	{ tanh, ReLU }	Activation function of layers
Optimizer	Adam Kingma and Ba (2014)	Optimization algorithm for gradient descent
Learning Rate	0.001	Learning rate of the Neural Network
β_1	0.9	Adam hyper-parameter
β_2	0.999	Adam hyper-parameter

5.3 Evaluation

The following tables describe the different model's evaluations on each of the generated datasets. The figures also show the loss and accuracy values of each neural network as a function of the training epochs. On each table, the highest test accuracy of each validation fold is bolded for comparison.

5.3.1 D_{simple} Dataset

Table 5.4 hold the model's evaluation metrics on the D_{simple} dataset, using temporal data representation.

TABLE 5.4: D_{simple} Evaluation - Temporal Representation

Model	**Fold Index**	**Train Loss**	**Train Accuracy**	**Val. Loss**	**Val. Accuracy**	**Test Accuracy**
LDA	1	-	0.9920	-	0.6183	0.605
	2	-	0.9933	-	0.6116	0.629
	3	-	0.9931	-	0.5883	0.64
SVM	1	-	0.9577	-	0.8291	0.8605
	2	-	0.9602	-	0.845	0.8705
	3	-	0.9606	-	0.795	0.8465
FC-NN	1	0.0612	0.9777	0.8506	0.8083	0.8295
	2	0.03911	0.9889	0.7050	0.8216	0.845
	3	0.0689	0.9787	1.0077	0.7741	0.8365
1D-CNN	1	0.0973	0.9645	0.6282	0.8316	0.835
	2	0.0606	0.9775	0.5514	0.8408	0.8415
	3	0.1076	0.9622	0.5175	0.8366	**0.8665**
Dilated-CNN	1	0.0803	0.9687	0.4027	0.8641	**0.882**
	2	0.0542	0.9835	0.4447	0.86	**0.8945**
	3	0.0385	0.9875	0.6742	0.8341	0.8625

Table 5.5 hold the model's evaluation metrics on the D_{simple} dataset, using spectral data representation.

TABLE 5.5: D_{simple} Evaluation - Spectral Representation

Model	**Fold Index**	**Train Loss**	**Train Accuracy**	**Val. Loss**	**Val. Accuracy**	**Test Accuracy**
LDA	1	-	0.6814	-	0.54	0.557
	2	-	0.6852	-	0.5075	0.548
	3	-	0.6697	-	0.530	0.552
SVM	1	-	0.7766	-	0.5816	0.6165
	2	-	0.7770	-	0.585	0.615
	3	-	0.7716	-	0.620	0.639
FC-NN	1	0.1396	0.9289	1.8852	0.5866	0.626
	2	0.1377	0.9266	2.1162	0.5783	0.627
	3	0.1398	0.9272	1.602	0.6258	0.6445
1D-CNN	1	0.0638	0.9710	0.4354	0.8791	**0.906**
	2	0.0497	0.9797	0.3801	0.8666	**0.9035**
	3	0.0587	0.9733	0.353	0.8791	**0.912**
Dilated-CNN	1	0.0370	0.9852	0.6258	0.8641	0.8885
	2	0.0341	0.9870	0.5660	0.8666	0.8765
	3	0.0650	0.9764	0.4372	0.8725	0.8915

Figures 5.3 and 5.4 plots the network's loss function and accuracy metric values of the temporal and spectral data representations, respectively.

5.3.2 $D_{complex}$ Dataset

Table 5.6 hold the model's evaluation metrics on the $D_{complex}$ dataset, using temporal data representation.

TABLE 5.6: $D_{complex}$ Evaluation - Temporal Representation

Model	Fold Index	Train Loss	Train Accuracy	Val. Loss	Val. Accuracy	Test Accuracy
FC-NN	1	0.1263	0.9548	1.3273	0.6981	0.687
	2	0.1059	0.9629	1.3290	0.6905	0.6905
	3	0.1051	0.9638	1.2783	0.6904	0.6990
CNN	1	0.0571	0.9797	1.1140	0.7730	0.7631
	2	0.0663	0.9754	1.1061	0.7595	0.7506
	3	0.0641	0.9769	1.1457	0.7468	0.7602
Dilated-CNN	1	0.0739	0.9736	0.9322	0.7812	**0.7775**
	2	0.0527	0.9815	1.0472	0.7766	**0.7749**
	3	0.0506	0.9825	1.1661	0.7573	**0.7658**

Table 5.7 hold the model's evaluation metrics on the $D_complex$ dataset, using spectral data representation.

TABLE 5.7: $D_{complex}$ Evaluation - Spectral Representation

Model	Fold Index	Train Loss	Train Accuracy	Val. Loss	Val. Accuracy	Test Accuracy
FC-NN	1	0.2213	0.9056	2.1975	0.5412	0.5565
	2	0.2275	0.9015	2.0364	0.5795	0.5714
	3	0.2185	0.9053	2.4275	0.5494	0.5200
CNN	1	0.0810	0.9671	1.009	0.7793	**0.7855**
	2	0.0708	0.9718	1.0730	0.7877	**0.7841**
	3	0.0828	0.9663	1.1214	0.7987	**0.7899**
Dilated-CNN	1	0.1117	0.9551	1.2578	0.7388	0.738
	2	0.1036	0.9594	1.2313	0.7377	0.7425
	3	0.0979	0.9605	1.4924	0.7212	0.7275

Figures 5.5 and 5.6 plots the network's loss function and accuracy metric values of the temporal and spectral data representations, respectively.

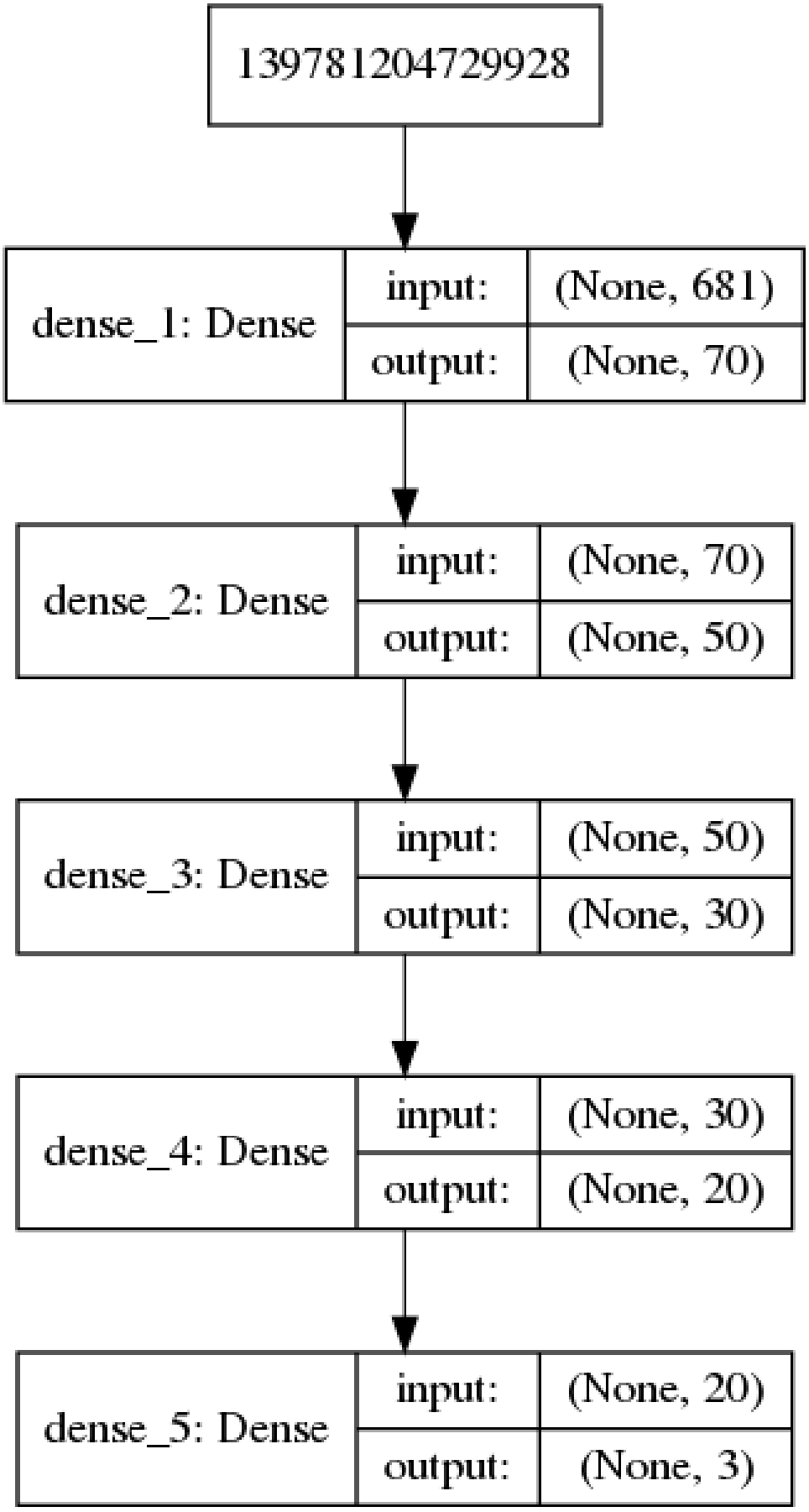


Figure 5.1: FC-NN Architecture

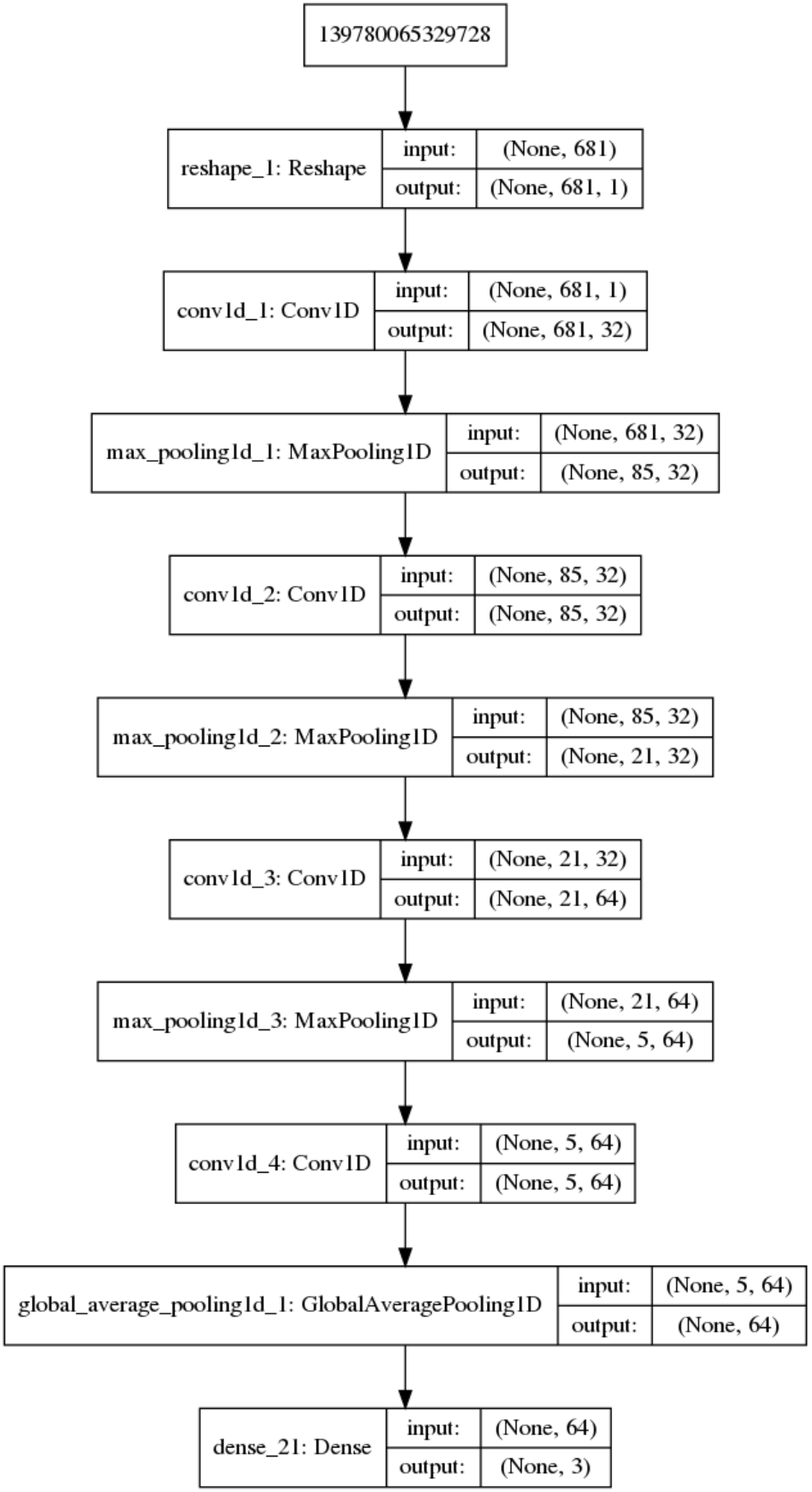


FIGURE 5.2: 1D-CNN Architecture

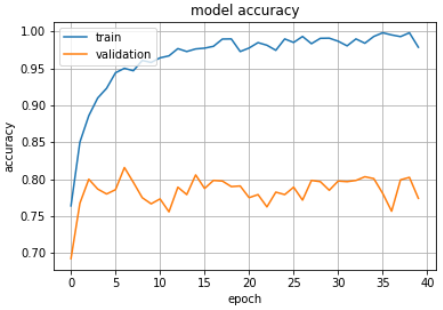
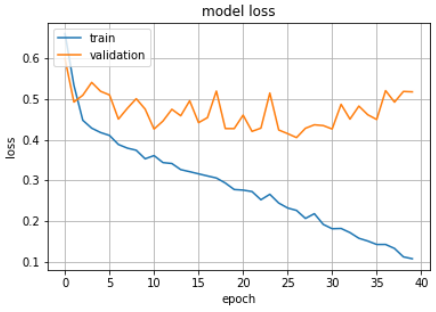
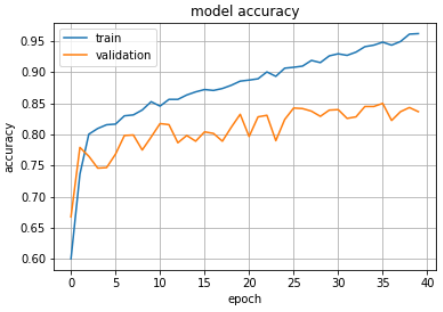
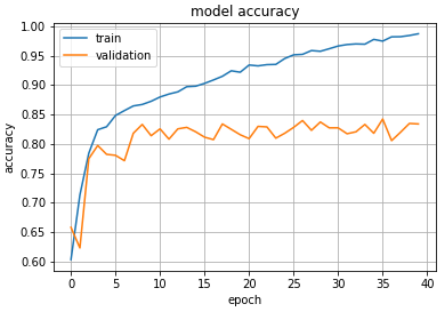
Model	Loss	Accuracy
FC-NN		
1D-CNN		
Dilated-CNN		

FIGURE 5.3: D_{simple} Learning Plots - Temporal Representation

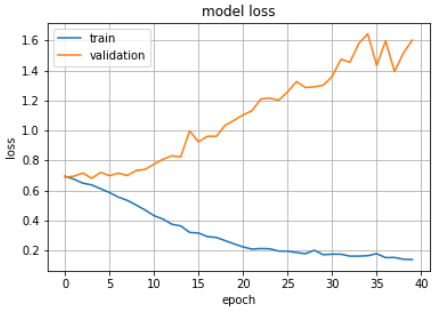
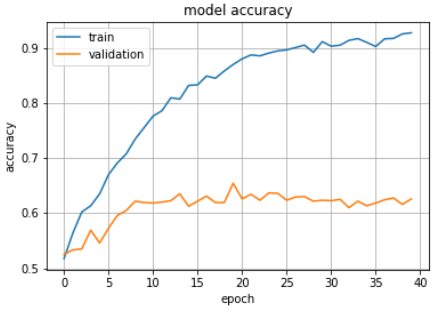
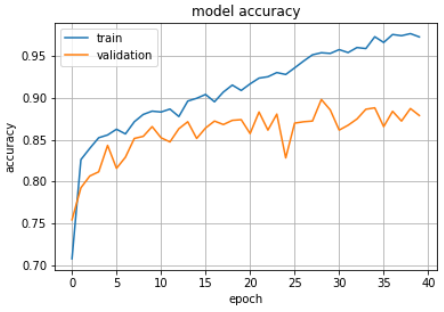
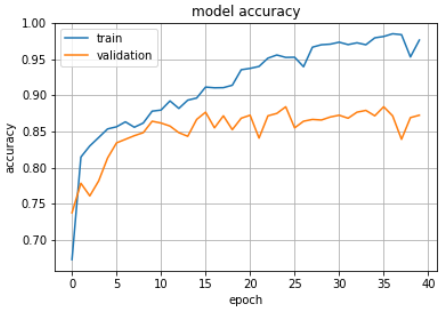
Model	Loss	Accuracy
FC-NN		
1D-CNN		
Dilated-CNN		

FIGURE 5.4: D_{simple} Learning Plots - Spectral Representation

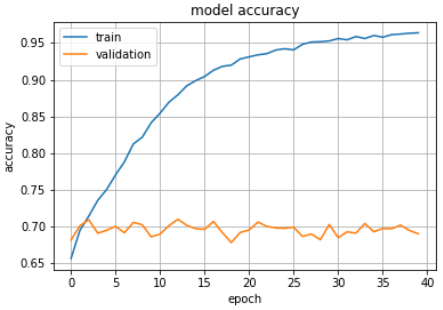
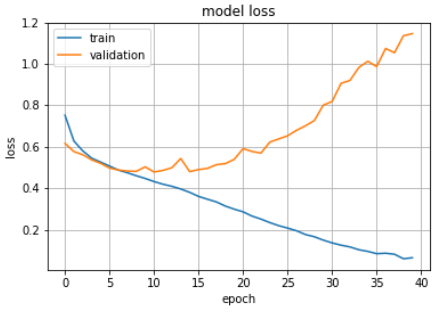
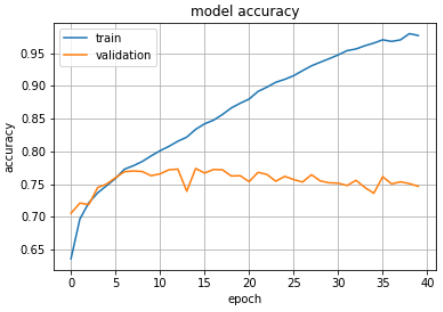
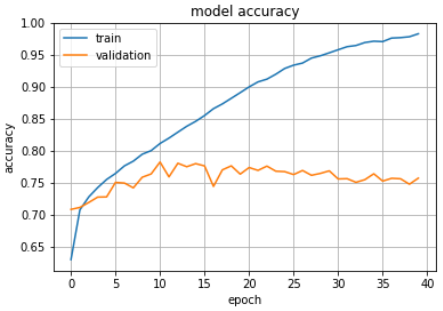
Model	Loss	Accuracy
FC-NN		
1D-CNN		
Dilated-CNN		

FIGURE 5.5: $D_{complex}$ Learning Plots - Temporal Representation

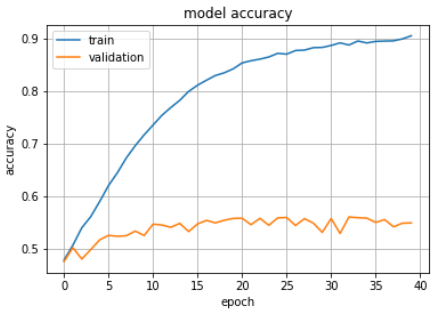
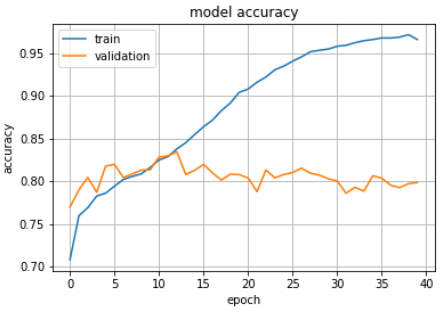
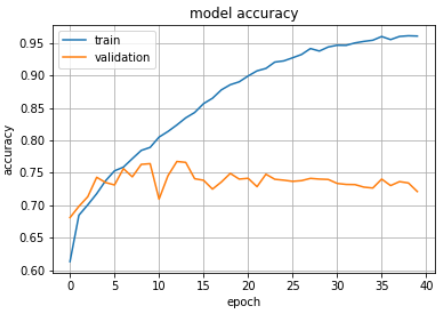
Model	Loss	Accuracy
FC-NN		
1D-CNN		
Dilated-CNN		

FIGURE 5.6: $D_{complex}$ Learning Plots - Spectral Representation

Chapter 6

Conclusions and Future Works

6.1 Evaluation Discussion

Evaluations on both datasets show the superiority of the neural networks over the baseline models. On both data representations, the LDA model had a very high training accuracy but generalized poorly. On the other hand, the SVM model showed competitive testing accuracy against the FC-NN, but only when using the temporal representation; It had low training and validation accuracy when using the spectral representations.

Between the neural networks, the FC-NN model had lower testing accuracy than the convolutional models on both data representations. It may be accounted to the nature of the data; Temporal representation requires time-invariant features to generalize better, where spectral representation requires a model to recognize correlations between adjacent frequencies, and potentially detect occurring patterns throughout the spectrum. FC-NN treats each frequency as an independent feature, where a learned filter of a CNN detects patterns that co-occurs between frequencies. Furthermore, the convolutional models have ~ 10 times less learned parameters than the fully-connected neural network, and at the same time, have better generalization performance.

The 1D-CNN model had the highest test accuracy when using the spectral representation, while the Dilated-CNN model had the highest test accuracy when using the temporal representation.

These results may indicate that spectral representation has the property that all given frequencies of the returning echo are crucial for classification. Evidence for this is the reduction in performance when using dilated convolutions, as the receptive field of a learned filter does not include all features of a sample, but rather "skips" some of them

concerning the dilation rate. On the other hand, when using the temporal representation, each sample is represented by a much longer feature vector, ordered by time, that is potentially sparse. That is, information is not nearly uniformly distributed as the spectral representation. Furthermore, temporal patterns may occur on different scales. Using dilated convolution may reveal more fine-grained information, and "systematically aggregate multi-scale contextual information without losing resolution." (Yu and Vladlen (2015)) Besides, the use of dilated convolutions has been proven to be state of the art in audio processing tasks (Aaron van den Oord (2016)), which is coherent with our results.

6.2 Conclusions

We present datasets that depict echolocating bats classifying plants around them, which were generated by an acoustical simulation. The datasets are controlled by a series of hyper-parameters, each one affecting the nature and complexity of the dataset. Specific preprocessing algorithms were employed to ensure the robustness of our solution, and carefully designed cross-validation schemes were employed to verify the real-world scenario of a bat classifying an unseen plant specimen.

We showed that the use of Deep Learning models, especially variants of the Convolutional Neural Networks, outperforms previously considered state-of-the-art classification models, such as Support Vector Machine. Furthermore, these kinds of models are capable of classifying an unseen plant specimen using only one echo, with relatively high accuracy, mimicking how a bat classifies objects in its surroundings. We observed the adequacy between data representations and models; From the evaluation results, a "vanilla" 1-dimensional convolution neural network outperforms other models when using the spectral representation, while models integrating dilated convolutions outperform other models when using the temporal representation.

6.3 Future Work

For a start, one may always seek to optimize the hyper-parameters of the evaluated models to improve model performance further. For example, increasing the number of training epochs, change the learned filter size and number of channels of a convolutional layer, or altering the learning rate of the gradient descent algorithm used, all may increase the model's performance.

A realistic echo dataset, acquired by physical sensors we lack on this research, should be devised and evaluated using the models mentioned in our research. Then, we will have a better understanding of how well the simulation depicts the physical world and may lead us for more insights about echo data patterns and attributes.

Unsupervised Learning (Barlow (1989)) methods can also be applied to echo data. It may aid in further classification tasks, e.g., enhanced feature extraction via an Autoencoder Rumelhart and William (1985)), or to train a generative model such as the Generative Adversarial Network (Goodfellow et al. (2014)) on echo data, in order to generate additional synthetic echo samples for data augmentation Tanner and Wong (1987) and enrichment.

6.3.1 Acoustical Flow

Optical Flow Horn and Schunck (1981) is a well-known problem in the field of computer vision, that computes the vector field of motion between a sequence of images (Or: a video). That is, given two adjacent images, the optical flow is a matrix of the same dimension of the images, where each entry on the matrix indicates the direction and amplitude of the motion between these frames, Optical Flow, used in many scientific fields, best known for motion estimation of a physical object (Brox and Malik (2010)), or reconstructing a live 3-dimensional video scene.(Suwajanakorn et al. (2014))

We can re-apply the concept of this method onto the acoustical flow; A 3-D scene of a plant, with its corresponding returning echoes, can be ordered concerning a predefined emitter locations order, as a sequence of signals. For every two adjacent echoes in the sequence, one may re-formulate the Acoustical Flow problem as a reduction of Optical Flow; Find the motion between frequencies or impulse responses, with respect to Optical Flow assumptions and restrictions. One possible usage of the computed flow field is to interpolate intermediate echoes between two adjacent echoes, allowing us to incorporate more information about the ensonified object for further tasks and analysis.

Bibliography

H. Z. K. S. O. V. A. G. N. K. A. S. K. K. Aaron van den Oord, Sander Dieleman. Wavenet: A generative model for raw audio, 2016. URL https://arxiv.org/pdf/1609.03499.pdf.

Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

Bellman. Dynamic programming. *Princeton University Press, Princeton, NJ, USA*, 1957.

Bengio, Simard, and Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, 1994.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):500–513, 2010.

Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19 (5):1155–1178, 2007.

Chauhan, Dahiya, and Sharma. Problem formulations and solvers in linear svm: a review. *Artificial Intelligence Review*, pages 1–53, 2018.

Cooley and Tukey. An algorithm for the machine calculation of complex fourier series. *Math. Comput. 19 (90)*, pages 297–301, 1965.

Cortes and Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995. doi: 10.1023/A:1022627411411. URL https://doi.org/10.1023/A:1022627411411.

Cristianini and Shawe-Taylor. An introduction to support vector machines and other kernel based learning methods. *Cambridge University Press*, pages 93–120, 2000.

S. J. Farlow. The gmdh algorithm of ivakhnenko. *The American Statistician*, 35(4): 210–215, 1981.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. pages 2672–2680, 2014.

D. R. Griffin. *Listening in the dark*. New Haven, Yale University Press, 1958.

He, Zhang, Ren, and Sun. Deep residual learning for image recognition. pages 770–778, 06 2016. doi: 10.1109/CVPR.2016.90.

R. Hecht-Nielsen. Theory of the backpropagation neural network. pages 65–93, 1992.

B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial intelligence*, 17 (1–3):185–203, 1981.

Hubel and Wiesel. Receptive fields and functional architecture of monkey striatecortex. *Journal of Physiology (London), vol. 195*, pages 215—243, 1968.

Huberty and Olejnik. *Applied MANOVA and Discriminant Analysis*. Wiley, 2006.

James, Witten, Hastie, and Tibshirani. An introduction to statistical learning: with applications in r. *Springer Texts in Statistics*, pages 180–181, 2013.

Kalko and Condon. Olfaction and fruit display: how bats find fruit of flag ellichorous cu curbits. *Funct Ecol 12*, pages 364—372, 1998.

Kingma and Ba. Adam: A method for stochastic optimization, 2014. URL https://arxiv.org/abs/1412.6980.

Knerr, Personnaz, and Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. *Neurocomputing*, pages 41–50, 1990.

Kressel. Pairwise classification and support vector machines. *Advances in Kernel Methods – Support Vector Learning*, 1999.

LeCun, Haffner, Bottou, and Bengio. Object recognition with gradient-based learning. page 319, 1999. URL http://dl.acm.org/citation.cfm?id=646469.691875.

LeCun, Bengio, and Hinton. Deep learning. *Nature 521*, pages 436—444, 2015.

Levin, Yom-Tov, and Barnea. Frequent summer nuptial flights of ants provide a primary food source for bats. *Naturwissenschaften 96*, pages 477—483, 2009.

Long, Shelhamer, and Darrell. Fully convolutional networks for semantic segmentation. pages 3431–3440, 2015.

Maaten and Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

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

S. Ruder. An overview of gradient descent optimization algorithms, 2016. URL https://arxiv.org/abs/1609.04747.

Rumelhart and H. William. Learning internal representations by error propagation. 1985.

Scarselli and Tsoi. Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results. *Neural networks*, 11(1): 15–37, 1998.

Schmidt, Hanke, and Pillat. The role of echolocation in the hunting of terrestrial prey—new evidence for an underestimated strategy in the gleaning bat megaderma lyra. *J Comp Physiol A186*, pages 975—488, 2000.

H.-U. Schnitzler, C. F. Moss, and A. Denzinger. From spatial orientation to food acquisition in echolocating bats. *Trends in Ecology and Evolution 18(8)*, pages 386–394, 2003.

Simmons and Chen. The acoustic basis for target discrimination by fm echolocating bats. *J Acoust Soc Am 86*, pages 1333–1350, 1989.

Simmons and Vernon. Echolocation: discrimination of targets by the bat, eptesicus fuscus. *J Exp Zool 176*, pages 315–328, 1971.

Simonyan and Zisserman. Very deep convolutional networks for large-scale image recognition, 2014. URL https://arxiv.org/abs/1409.1556.

Smith. A tutorial on principal components analysis. 2002.

Stone. Cross-validatory choice and assessment of statistical predictions. *J. Royal Stat. Soc., 36(2)*, pages 111—147, 1974.

S. Suwajanakorn, I. Kemelmacher-Shlizerman, and S. M. Seitz. Total moving face reconstruction. In *European Conference on Computer Vision*, pages 796–812. Springer, 2014.

M. A. Tanner and W. H. Wong. The calculation of posterior distributions by data augmentation. *Journal of the American statistical Association*, 82(398):528–540, 1987.

Thies, K. E. W, and S. H-U. The roles of echolo-cation and olfaction in two neotropical fruit-eating bats. *Ecol Sociobiol 42*, pages 397—409, 1998.

V. Vapnik. *The Support Vector Method of Function Estimation*. Springer US, Boston, MA, 1998. ISBN 978-1-4615-5703-6. doi: 10.1007/978-1-4615-5703-6_3. URL https://doi.org/10.1007/978-1-4615-5703-6_3.

Wang and Wu. Helmholtz equation - least-squares method for reconstructing the acoustic pressure field. *The Journal of the Acoustical Society of America*, 102(4): 2020–2032, 1997.

J. Ye. *Least Squares Linear Discriminant Analysis*. Association for Computing Machinery, 2007. doi: 10.1145/1273496.1273633. URL https://doi.org/10.1145/1273496.1273633.

Yovel, Franz, Stilz, and Schnitzler. Plant classification from bat-like echolocation signals. *PLoS Comput Biol 4(3)*, 2008.

Yovel, Franz, Matthias, Stilz, and Schnitzler. Complex echo classification by echo-locating bats: A review. *Journal of comparative physiology. A, Neuroethology, sensory, neural, and behavioral physiology*, pages 475—490, 2011.

Yu and Vladlen. Multi-scale context aggregation by dilated convolutions, 2015. URL https://arxiv.org/abs/1511.07122.