CALIFORNIA STATE UNIVERSITY SAN MARCOS

THESIS SIGNATURE PAGE

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

THESIS TITLE: Application of Artificial Neural Network on Speech Signal Features for Parkinson's Disease Classification

AUTHOR: John Wu

DATE OF SUCCESSFUL DEFENSE: April 22, 2019

THE THESIS HAS BEEN ACCEPTED BY THE THESIS COMMITTEE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE.

Dr. Xin Ye	Xin Ye	4/22/19
THESIS COMMITTEE CHAIR	SIGNATURE	DATE
Dr. Rika Yoshii	Rika Yoshii	4/22/19
THESIS COMMITTEE MEMBER	SIGNATURE	DATE

Application of Artificial Neural Network on Speech Signal Features for Parkinson's Disease Classification

John Wu

Department of Computer Science and Information Systems

California State University San Marcos

Thesis Abstract

Parkinson's Disease is a degenerative disorder of the central nervous system. Diagnosis is made based on patient history and physical exam findings. Since speech is affected in up to 90% of Parkinson's Disease patients, it has drawn interest as a target symptom for diagnostic testing.

Using a relatively large dataset of speech sample features compiled from Parkinson's Disease patients and healthy individuals, a multilayer perceptron artificial neural network was trained to accurately classify between these two types of samples. By experimenting with the number of neurons in a single hidden layer network, a model was quickly found that could reach 0.937 accuracy. Using the single hidden layer results to inform a search for viable deep learning models, two hidden layer networks were found that could improve accuracy to 0.952.

Finding high accuracy using a large sample size dataset indicates that multilayer perceptron is a viable method to aid in diagnosis of Parkinson's Disease in the general population. This research demonstrates a potential search strategy for deep learning models that does not require the time and computational power of a comprehensive search of all possible models. The most accurate models found obtained 100% sensitivity. Thus, this method would be very useful as a screening test for Parkinson's Disease as it is also non-invasive, quick, and can be done remotely.

Acknowledgments

I would like to thank my advisor, Dr. Xin Ye, for his teaching, advice, and patience with this research and throughout my studies at CSUSM. I greatly appreciate his help in exploring the possibilities of machine learning and neural networks.

My sincere appreciation extends to my committee member, Dr. Rika Yoshii, for the invaluable suggestions and comments as well as instruction in Artificial Intelligence and other classes.

Thank you to all the faculty of computer science department at CSUSM for their teaching and help in building a solid foundation for this research and my future career.

Special thanks to my family and friends for their support and patience as I pursued my educational goals.

Table of Contents

List of Figures

List of Tables

1 Introduction

In recent years, ever increasing amounts of data are being generated and collected about nearly everything. The high volume and number of features of most of this data make it difficult for humans to extract useful conclusions or information. However, it is possible to have computers do the hard work. Machine learning has been used more and more on a multitude of problems.

1.1 Motivation

Artificial neural networks are computer programs inspired by the way biological neurons are connected. They can be trained to detect patterns and relationships of data given to them as opposed to directly programmed to do so. They can be applied to a wide variety of problems. Stock market prediction is a popular problem with a large variety of data that can potentially be used as input. A recent example is using a recurrent neural network on a time series of stock prices to make short-term stock price predictions (Khare, Darekar, Gupta, & Attar, 2017). Prediction of traffic levels of roads in an area after an accident has been modeled by a long short term memory network using Los Angeles traffic information (Yu, Li, Shahabi, Demiryurek, & Liu, 2017). Knowing the prevalence of influenza is important for control and prevention activities. After training with years of historic data, a neural network with a genetic algorithm setting initial weights was able to predict 16 weeks of influenza levels in advance (Xue, Bai, Hu, & Liang, 2018). As security cameras become more commonplace, making use of the massive amounts of saved video is increasingly difficult. Using a model that fuses using a deep convolutional network on images of

people and another neural network of hand-selected features such as RGB histograms of the image, a person seen on video footage can be re-identified automatically (Wu, et al., 2016).

A very common problem with interpreting data is to classify it into different categories. Neural networks can accomplish classification when given training data consisting of a set of features with their labelled classes. If properly trained, they can accurately return a category label given a previously unseen set of feature data. A movie can be categorized into different levels of potential earning using data features including genre, MPAA rating, star value, special effects usage, and other preproduction knowledge (Ghiassi, Lio, & Moon, 2015). Using headlines from financial news, a deep learning convolutional neural network was able to categorize if stock index or individual stocks would rise or lower in price the next day (Ding, Zhang, Liu, & Duan, 2015). By representing the text from bug reports using Skip-Gram model word embeddings a recurrent neural network can be used to classify which parts of the source code the bug report is relevant to (Ye, Fang, Wu, Bunescu, & Liu, 2018).

Medical diagnosis is a classification problem. For any disease, signs and symptoms need to be evaluated to categorize a patient as suffering from the disease or not and sometimes into different variants or severities of a disease. Use of neural networks has the potential to assist physicians in making more accurate diagnosis or even finding new patterns that can make the diagnosis. For example, a recurrent neural network trained using electronic medical record data called Doctor AI can classify a patient into the conditions or medications they are likely to have out of a large list of possibilities (Choi, Bahadori, Schuetz, Stewart, & Jimeng, 2016). Forecasting depressed mood can help with the mental health of many people. Using a person's self-reported mood via a smartphone app, a long short term memory recurrent neural network can categorize if a patient will

or will not have a severe depressed mood in the upcoming few days (Suhara, Xu, & Pentland, 2017).

Parkinson's Disease is a degenerative disorder of the central nervous system with the most recognizable symptoms resulting from motor system dysfunction (Jankovic, 2008). The disease progresses over a long time period and unfortunately has no cure. Current treatments for Parkinson's Disease are done to manage symptoms and earlier diagnosis can allow for forming more optimal treatment plans. Speech is a part of the motor system that is affected in close to 90% of patients (Logemann, Fisher, Boshes, & Blonsky, 1978) (Ho, Iansek, Marigliani, Bradshaw, & Gates, 1998). Since there is no definitive diagnostic test for Parkinson's Disease (Jankovic, 2008), analysis of speech samples to classify between Parkinson's Disease or healthy individuals is a potential non-invasive test that can be clinically useful to physicians in assessing patients for Parkinson's Disease.

Recently, a dataset of speech sample features from 188 Parkinson's Disease patients and 64 healthy individuals was made publicly available (Sakar C. , et al., 2019). Using their dataset Sakar et al. (2019), explored a variety of classification methods including multilayer perceptron. The accuracy of the methods they explored reached up to 0.86 accuracy, with 0.84 F_1 score and 0.59 MCC[1]. Using previous smaller Parkinson's Speech datasets, studies using different methods eventually reached accuracy of 0.9952 with a complex-valued artificial neural network with k-means clustering-based feature weighting (Gürüler, 2017). As such, I was motivated to explore different configurations or variants of artificial neural networks using the larger sample size available with the new dataset. Obtaining improved classification accuracy with a larger dataset demonstrates the potential to

[1] Accuracy, F_1 score, and MCC are explained in 4.3 Performance Metrics.

expand the use of artificial neural networks on speech for diagnosing or assessing for Parkinson’s Disease to a generalized population.

1.2 Thesis Outline

Chapter 1 covers the motivations that prompted this research and outlines the contents chapter by chapter.

Chapter 2 describes machine learning with artificial neural networks focusing on multilayer perceptron.

Chapter 3 covers the problem domain of Parkinson’s Disease classification and use of speech symptoms to accomplish that diagnostic task.

Chapter 4 describes the data processing, the experimental setups used, and the metrics used to evaluate the results.

Chapter 5 presents the results and analysis of the experiments.

Chapter 6 concludes with a summary of the findings of this research and some ideas for future work

2 Artificial Neural Networks

Artificial neural networks (ANN) can be thought of as a massively parallel processing model composed of numerous densely interconnected simple processing elements (Lippmann, 1987). The individual processing elements and the way they are interconnected were inspired by the way that neurons are connected in the nervous system. Rather than accomplishing a task through a series or sequence of programed instructions, an ANN can improve its performance on the task by adjusting the weights of its connections. In this way, an ANN might be able to accomplish tasks that require simultaneously exploring a multitude of hypotheses in parallel by learning patterns in data through experience rather than using a task specific algorithm.

2.1 Neurons

The basic processing elements of ANNs are neurons or nodes. They are modeled after a biological neuron that generally receive inputs at multiple synapses and then determine if it will generate output by firing neurotransmitters at synaptic connections to other neurons. In 1943, McCulloch and Pitts outlined a representation for artificial neurons consisting of a set of weighted inputs, an adder that sums the weighted inputs, and an activation function that determines if the neuron fires based on the summed input (McCulloch & Pitts, 1943).

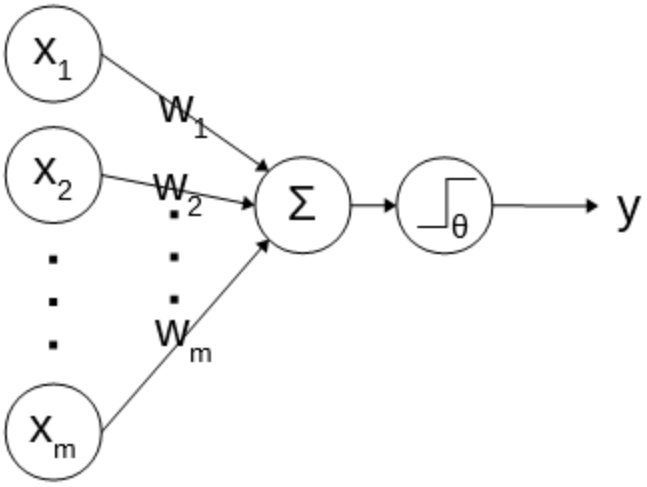

Figure 1 Diagram of a McCulloch and Pitts Neuron.

As Figure 1 shows, the neuron receives a set of input values x_1, x_2... x_m. Each input channel has a weight w_1, w_2,…w_m. This allows the input to be summed by adding the results of multiplying each weight with corresponding input. The summed input is passed through an activation function φ to generate the neurons output y. The overall equation for the neuron is:

$$y = \varphi\left(\sum_{i=1}^{m} w_i x_i\right)$$

Neurons can receive the input data or output of other neurons as input. The weights can be adjusted to generate the desired output which allows the neuron to learn.

There are many possible activation functions that change the performance or type of output of a neuron. Many activation functions transform the sum of the weighted inputs into an output value within a set range, such as [-1,1] or [0,1]. They can be continuous or discontinuous.

The McCulloch and Pitts neuron uses a binary step function:

$$f(x) = \begin{cases} 0, for\ x \leq \theta \\ 1, for\ x > \theta \end{cases}$$

It results in firing if the sum of the weighted inputs reaches the threshold, θ. Otherwise, it does not. This is analogous to a biological nerve cell firing once it reaches its action potential.

2.2 ANN Models

Artificial neural networks, as the name implies, are combinations, or networks, of artificial neurons (Lippmann, 1987). While there are many possible topologies, neurons are generally organized in layers including an input layer and an output layer. When organized in a hierarchical manner from input to output, connections between neurons are generally made only to neurons in adjacent layers. Commonly, they are densely interconnected where every neuron in an earlier layer is connected to every neuron in the next layer. Other variations of ANNs may have neurons that connect outside of adjacent layers or use feedback connections to adjacent layers or back to itself or other neurons within the same layer.

Selecting the number of neurons to be used can make a difference in how successful the ANN is and depends on the application. The number must be sufficiently high to be able to form complex enough decision region to accomplish the given problem. However, using too many neurons means that the training data will be insufficient to estimate the weights of at least some of the connections reliably.

For the ANN to improve at accomplishing the given task, an algorithm to adjust weights based on the output must be used. The ANN must be

initialized with certain weights. Usually, they are set randomly and are unlikely to initially be able to solve the given problem well. As such, the ANN must be trained. The generalized idea would be to use inputs to calculate the output of the current model, then use the difference between the actual output and the desired output to adjust the weights of the model. This is repeated to improve the success of the model. Using the full set of training data is considered an epoch. Training often continues over many epochs, running the training data through the model multiple times and continuing to update weights each time.

2.3 Multilayer Perceptron

The simplest network would be to have a single layer of neurons with each taking input and generating an output. This is known as a perceptron network and has some limitations. Adding at least one layer of neurons between the input nodes and the output neurons creates a multilayer perceptron. This network is more complex to train but is also capable of being successfully trained for more tasks.

2.3.1 Perceptron

A perceptron network is a linear classification algorithm that consists of input nodes and a collection of McCulloch and Pitts neurons (Rosenblatt, 1958). As visualized in Figure 2, each neuron is independent and sees every input. There can be *m* inputs that are set based on the data and *n* neurons that are determined by the number of targets for the output. For a given input, this perceptron network will generate a pattern of firing and non-firing neurons

as output. If the fired neuron matches the intended output, then it seems to be working as desired. But if it fired when it should not have or vice versa, then the weights of that neuron need to be adjusted.

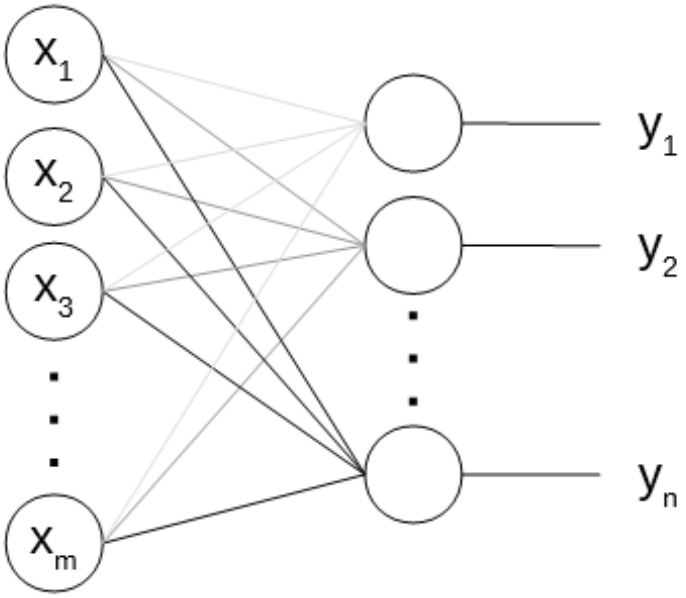

Figure 2 Diagram of a Perceptron network

If neuron *k* was wrong, then it has *m* weights (w_{ik} corresponding to input x_i, where *i* is 1 through m) that could be adjusted to increase or decrease output. If y_k is the actual output of neuron *k* and t_k is the target output of neuron *k*, then an error function can be calculated as $y_k - t_k$. If neuron fires but should not have, then error will be positive and weights should be updated in a negative direction. Conversely, if the neuron didn't fire but should have, then error will have a negative value and weights should be adjusted in a positive direction. Since inputs could be negative, multiplying the error function with the input should move the corresponding weight in the intended direction. Multiplying a learning rate η will specify how much to adjust the weights each time and how fast the network learns. A high

learning rate makes the network unstable due to weights changing by a large value with each adjustment which makes it difficult for the network to settle down. On the other hand, low learning rate results in longer training since the weights change little each time the network sees an input. Overall, this rule for updating the weights is:

$$w_{ik} \leftarrow w_{ik} - \eta(y_k - t_k) \cdot x_i$$

Unfortunately, perceptron is limited as it finds a decision boundary that is linear in 2 dimensions, planar in 3 dimensions, and a hyperplane in greater than 3 dimensions. This allows it to be capable of classifying linearly separable cases. It was shown that this limitation means it could not solve XOR (Minsky & Papert, 1969) and this resulted in little progress in neural network research for almost 2 decades.

2.3.2 Multilayer Perceptron

A multilayer perceptron (MLP) is a feed forward artificial neural network that has an input layer, one or more hidden layers of neurons, and an output layer of neurons. Each layer is fully connected with the next and input moves to output in a feed forward manner through the connections and activations functions of each layer of neurons sequentially. An MLP with a single hidden layer connected in this manner is shown in Figure 3. While training, there is a backward propagation of error through the layers to update the weight of each connection (Rumelhart, Hinton, & Williams, 1986b). This model is able to distinguish data that is not linearly separable and was able to solve XOR (Rumelhart, Hinton, & Williams, 1986a).

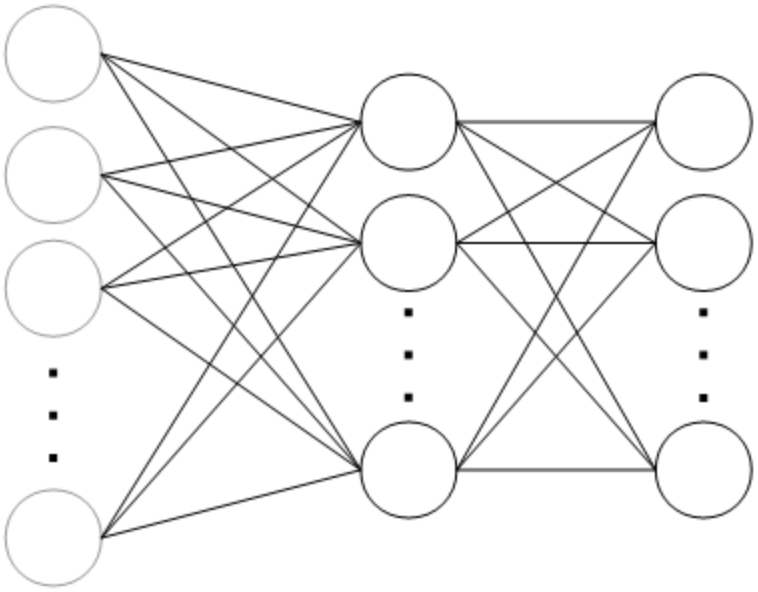

Figure 3 Diagram of an MLP with one hidden layer.

The input nodes are not neurons. They simply take the input values and pass them to each node in the next layer. The neurons utilize a nonlinear, sigmodal function instead of a threshold function as the activation function.

2.3.3 Back Propagation

Similar to training the perceptron, the goal is to minimize the error of the MLP by updating the weights of the neuron connections. However, this is complicated by the layered arrangement meaning there is no longer an independent set of weights affecting the error of each output neuron. MLP will need a different error function because simply summing the error of each output neuron could result in offsetting that would reduce the apparent amount of error (Marsland, 2015). If one neuron was over by the same

amount as another neuron was under their target outputs, then combining those errors directly might result in perception that there is no overall error for the neural network. To prevent this, a sum-of-squares error function can be used. This is done by squaring all the differences between actual output *y* and target output *t* of each neuron before summing for all *n* neurons:

$$E(t, y) = \frac{1}{2} \sum_{i=1}^{n} (y_i - t_i)^2$$

Thus, to figure out how to adjust the weights of the model, differentiating the error function will show the directions where the error increases or decreases the most. Using this gradient of the error, the weights can be modified in the direction that reduces the error.

The output of a neuron is determined by the activation function. So, in order for the error function to be differentiable so that its gradient can be computed, the activation function cannot be discontinuous, such as that used by the McCulloch and Pitts neuron. Commonly used continuous activation functions include sigmoid (logistic) and hyperbolic tangent (TanH). Sigmoid (logistic) produces a scaled output for the neuron in the range (0,1) using this equation:

$$f(x) = \frac{1}{1 + e^{-x}}$$

TanH produces a scaled output for the neuron in the range (-1,1) using this equation:

$$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

These both saturate near the ends of their ranges with a continuous s-shaped curve that resembles a step function but is smooth and differentiable.

Unlike the case with perceptrons, the output neurons cannot see the input values that caused it. Similarly, the hidden layer does not know the target output. If more than one hidden layer, some neurons will see neither the input values or target values. Thus, the weights for these nodes cannot be updated in the same way as with perceptrons since that calculation involved using both the input and target values.

To get around this, the chain rule of differentiation is used. This means that how the error changes as the weights vary can be found by multiplying the way the error changes as the inputs to the weights vary with how those input values change as the weights vary. This allows the use of the activation functions of output nodes to be written using the activations of the hidden nodes and the weights of the connections between the hidden and output nodes. Using this, the error calculations can be sent back to the hidden node and the output target for the hidden neurons can be calculated. This results in two update functions (one for the weights of the neurons in each layer) and allow updating the weights layer by layer backward through the network. This method can be extended for when there are two or more hidden layers (Marsland, 2015).

2.3.4 Early Stopping

Determining the amount of training to do on an artificial neural network can be difficult. Training for too long can result in overfitting where the network becomes too specific on the training examples by potentially incorporating irrelevant noise from the training data. Insufficient training means that the resulting network will be using weights that may not be as accurate as it potentially could be.

In general, as more training is done, the error with the training set should continue to decrease. Thus, a network can keep being trained until overfitting occurs. In order to check that the network is generalizing and not overfitting, a validation set of data that is not part of the training data can be used (Marsland, 2015). The validation set should also be separate from the data that will eventually be used to test the final network. The network's training can be monitored by periodically passing the validation set through the network and checking the resulting error. This is often done at the end of each epoch. The error with the validation set should initially decrease as the training improves the network. However, the error of the validation set will eventually start to increase, indicating that the network is improving by utilizing input noise within the training set and cannot generalize to data outside of the training set. This is where training can be stopped.

2.4 Deep Learning

Deep learning is a machine learning method that uses multiple levels of nonlinear processing units where each layer uses the previous layers output as input. This includes neural networks with multiple hidden layers between the input and output layers. As each layer transforms the data, it would ideally mean that each layer successively represents a higher level of abstraction about the data. Using enough transformations, the neural network could model very complex functions (LeCun, Bengio, & Hinton, 2015). For classification tasks, upper layers would ideally amplify the useful aspects of the input while filtering out the irrelevant information. The earliest layers could learn a large multitude of smaller basic patterns, such as edges, corners, curves, etc. for an image recognition task. The

subsequent layer might learn to fire for some combinations of these patterns and not for others. Then, getting more complex overall patterns through the higher layers until it learns enough to accomplish the intended task such as identifying a certain object in an image. The power of this type of system is that these features at each level are discovered without explicitly being programmed and would not need to be determined or even recognizable by a human. Since deep neural networks can model more complex functions, there can be drawbacks to deep learning due to potentially longer training and the increased possibility of overfitting.

By potentially forming larger combined patterns at each higher level, there is less need of having enough computational units in a single layer to cover the high overall number of patterns in the data. With higher layers using the output of lower layers as features, there is the possibility of using less total computational units compared to a network with similar performance using fewer layers (Bengio, 2009). Thus, even with more layers, each layer may need fewer nodes. This potentially reduces the number of connections and thus weights in the network that need adjusting to achieve learning. Though, the increased abstraction could need to take much more epochs and thus more overall computational time to fully train the network.

The amount of hyperparameters, parameters that can be used to tune a model, for training a deep neural network is increased compared to single hidden layer networks. These hyperparameters can include learning rate, numbers of nodes at each level, activation functions, etc. As a hyperparameter's effect can be dependent on other hyperparameters, it might not be possible to optimize each individually. Thus, it can take a huge amount of computational power to optimize the hyperparameters by looping through the testing of a multitude of models with a variety of combined settings (Claesen & De Moor, 2015). The number of models to test can quickly get out of hand. Just setting 3 different hyperparameters to 10

different values could result in exhaustively testing 1000 combinations. For example, just varying the numbers of neurons in each of layer of a three hidden layer network could generate huge numbers of model shapes to test.

3 Problem Domain

3.1 Parkinson's Disease

Parkinson's Disease is a degenerative disorder of the central nervous system with the most recognizable symptoms resulting from effects on the motor system. The pathology involves loss of dopamine secreting cells in the substantia nigra in the basal ganglia of the brain (Davie, 2008). The dopamine loss to the connected parts of the brain, including the motor circuit, can explain most of the symptoms associated with Parkinson's Disease.

There are four cardinal motor symptoms in Parkinson's Disease: tremor, bradykinesia, rigidity, and postural instability (Jankovic, 2008).

The tremor in Parkinson's Disease is characterized as a coarse tremor in the hand at rest that goes away with voluntary movement. A notable feature of the tremor is a "pill-rolling" motion. This is a circular movement of the thumb and index finger that resembles the manual assembly of pharmaceutical pills.

Bradykinesia is the slowness of movement. In Parkinson's Disease, this is the disturbance of motor planning, initiation, and execution. Sequential movements are impaired and formal assessment involves having the patient attempt repetitive motions with fingers and feet. Ability to simultaneously complete two independent tasks is also impaired.

Rigidity in limb movement is caused by increased muscle tone. The continuous contraction of muscles causes a resistance in movement that can be either "lead-pipe rigidity" (uniform) or "cogwheel rigidity" (ratcheting

through the range of motion of the limb). The muscle contraction can be enough to result in joint pain.

Postural instability shows up typically more in the later stages of Parkinson's Disease. This results in impaired balance and frequent falls as the disease becomes more severe.

Other motor symptoms include a "Parkinsonian gait" (shuffling steps with forward flex posture and no arm swing), reduced facial expression (described as "mask-like"), changes in handwriting such as micrographia (handwriting becomes smaller and smaller), and speech changes known as hypokinetic dysarthria. Hypokinetic dysarthria manifests in a variety of ways including affected rate of speech (slow with sudden acceleration or deceleration and involuntary pauses), difficulty with initiation of speech, repetition of words or syllables, reduced loudness/intensity, monoloudness, increased nasality, hoarseness, and imprecise consonant articulation (Brabenec, Mekyska, Galaz, & Rektorova, 2017).

Parkinson's Disease is also associated with many non-motor symptoms that affect cognition, mood, behavior, and thought. Executive dysfunction results in poor planning, flexibility, abstract thinking. Patients can have slowed cognitive processing and impaired memory recall. Patients can exhibit inappropriate actions and have poor impulse control. Parkinson's Disease patients are at increased risk of dementia, hallucinations, and delusions. Mood alterations such as depression, apathy, and anxiety are more common than in general population. Parkinson's Disease can result in sleep disorder symptoms of daytime drowsiness and REM sleep disturbance or insomnia. Perception can be affected such that senses of smell, vision and pain sense are impaired and paresthesia can be experienced.

3.2 Problem Overview

Since there is not a definitive test for Parkinson's Disease, diagnosis must be done using a careful history and medical exam. Physicians must make the diagnosis based on findings of the presentation of specific signs and symptoms consistent with Parkinson's Disease and exclusion of certain other findings. As noted above, there are a variety of symptoms in Parkinson's Disease and thus many possible targets for creating diagnostic testing.

Speech changes in Parkinson's Disease, known as hypokinetic dysarthria or Parkinsonian dysarthria, can be a useful target since it has been found in close to 90% of patients (Logemann, Fisher, Boshes, & Blonsky, 1978) (Ho, Iansek, Marigliani, Bradshaw, & Gates, 1998). Further benefits as a target symptom to use for testing include that it is non-invasive and can conveniently be done without a clinical visit using telemedicine (Tsanas, Little, McSharry, & Ramig, 2010).

3.3 Past Research

The Unified Parkinson's Disease Rating Scale (UPDRS) can be used to track Parkinson's Disease progression through levels of severity. It requires a time-consuming physical examination done in clinic. It was shown that score of UPDRS can be closely replicated using self-administered speech tests on 42 patients with Parkinson's Disease. This demonstrated the viability of remote use of speech for monitoring Parkinson's Disease patients (Tsanas, Little, McSharry, & Ramig, 2010). The viability for telediagnosis was shown in a study using speech samples from the milder, earlier Parkinson's Disease severities out of these 42 patients and adding samples

from 8 healthy patients. Using support vector machine with a third-degree polynomial kernel, the accuracy in discriminating between the two groups was as high as 0.964 with an MCC of 0.77 (Sakar, Serbes, & Sakar, 2017).

After gathering speech samples of sustained vowel phonation of 23 Parkinson's Disease patients and 8 healthy subjects, a dataset was created of features extracted by audio processing techniques. Classifying between Parkinson's Disease patients was 0.914 accurate using support vector machine (Little, McSharry, Hunter, Spielman, & Ramig, 2009). This dataset has an average of 6 samples from each subject for a total of 195 samples. It has been the subject of multiple studies and experiments of Parkinson's Disease classification.

One study trains a complex valued artificial neural network with the Little dataset and using mRMR feature selection reached 0.9812 accuracy (Peker, Şen, & Delen, 2015). Another study reached even higher accuracy of 0.9952 using a complex-valued artificial neural network with k-means clustering-based feature weighting (Gürüler, 2017). As individuals in the Little dataset provided multiple samples, there is the potential of individuals appearing in both training and testing data that might result in a benefit to accuracy. Such as, the former of the two previous studies used ten-fold cross-validation. In doing so, the 195 samples were randomly divided into 10 groups with each having a turn as the test set. With any of these ten training/testing splits, there is a high chance that some individuals are represented in both sets. Overall, the smaller population datasets may result in overfitting especially when every item of data is at some point used for training. However, these smaller population studies do establish the viability of certain audio extraction values such as jitter, shimmer, frequency coefficients, etc. as possible features that might be able to classify between Parkinson's Disease patients and healthy individuals in larger, more generalized populations.

A study using a balanced dataset build from 20 Parkinson's Disease and 20 healthy subjects reached 0.85 accuracy and found that sustained vowel speech samples were more useful for Parkinson's Disease classification than other types of samples such as word or sentences (Sakar, et al., 2013).

More recently, some studies have started using larger datasets. Using a dataset with 22 speech features of 147 Parkinson's Disease patients and 48 healthy individuals, multiple machine learning classifiers were trained and tested. These classifiers included support vector machine, k-nearest neighbors, radial bias function neural network, naïve Bayes, regression trees, and linear discriminant analysis. Support vector machine performed at 0.92 accuracy with other methods ranging from 0.63 to 0.75 accuracy (Lahmiri, Dawson, & Shmuel, 2018).

3.4 Parkinson's Disease Classification using Tunable Q-factor Wavelet

A larger dataset of speech samples of Parkinson's Disease patients would allow better evaluation of classification methods. Small datasets may contain certain noise or outlier information such that training classification methods might end up less applicable to the general population. Having enough samples would allow the use of fully independent training, validation, and testing sets without the need to use methods such as leave-one-out cross validation.

A dataset was created from speech samples of 188 Parkinson's Disease patients and 64 healthy individuals (Sakar C. , et al., 2019). The speech samples were processed with many methods previously used in Parkinson's Disease speech research and added a novel method using tunable Q-wavelet

transform (TQWT). This new dataset of extracted features was used to evaluate classification between Parkinson's Disease patients and healthy individuals using naïve Bayes, logistic regression, k-nearest neighbors, multilayer perceptron, random forest, support vector machine with linear kernel, and support vector machine with radial bias function kernel.

These classification methods were first evaluated on performance with features extracted using previous speech processing methods. Then, performance with features extracted with TQWT was found. Finally, performance was evaluated on the top 50 features from all possible features. In general, using the optimal TQWT settings, classification methods appeared to perform better with those speech features as input compared to using previous speech feature extraction methods. However, using SVM-RBF with a previous speech feature extraction method, MFCC, was able to achieve 0.84 accuracy with an F_1 score of 0.83 and MCC of 0.54. The best performing classification method using TQWT speech features was multilayer perceptron with 0.85 accuracy, F_1 score of 0.84, and MCC of 0.57. Finally, using the top 50 features of all speech feature extraction methods including TQWT was achieved by SVM-RBF with 0.86 accuracy, 0.84 F_1 score, and 0.59 MCC.

4 Data Analysis and Processing

For this research, an existing dataset of speech sample features will be used. The dataset will be used to train and test both single hidden layer and two hidden layer multilayer perceptron models.

The overall steps of the research plan described in this chapter can be visualized in Figure 4.

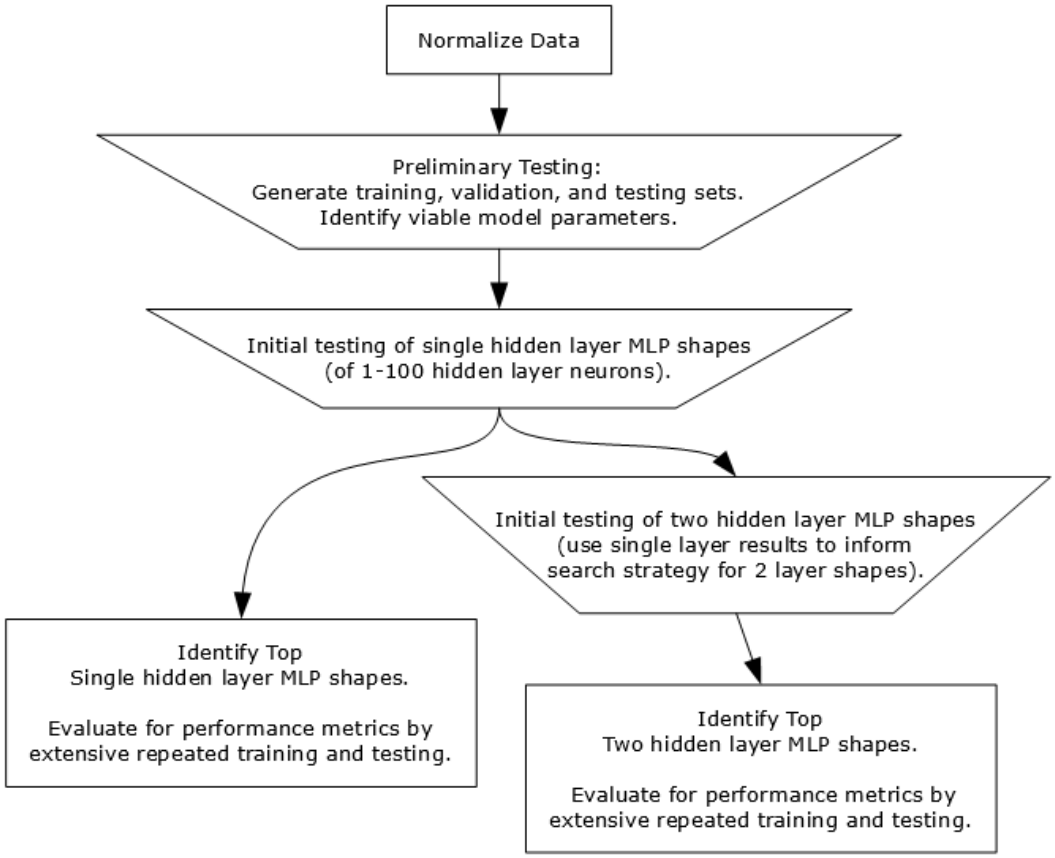

Figure 4 Research Plan Overview

4.1 Dataset

A dataset of features from speech samples of Parkinson's Disease patients was made available via the UCI Machine Learning Repository on November 5, 2018 [2]. The data was generated from vocal samples of 252 individuals. Each individual provided 3 samples of sustained phonation of vowel /a/ sound recorded through a microphone set at 44.1 KHz for a total of 756 samples (Sakar C. , et al., 2019).

The 252 individuals consisted of 188 Parkinson's Disease patients and 64 healthy individuals. The Parkinson's Disease group had ages ranging from 33 to 87 with an average of 65.1 and a standard deviation of 10.9. It contained 107 men and 81 women. The control group had ages ranging from 41 to 82 with an average of 61.1 and standard deviation of 8.9. Healthy individuals consisted of 23 men and 41 women.

The provided dataset contains data features already processed from the speech samples. It also provides gender, an ID number that can be used to identify which 3 samples are from one user, and indicates whether the sample is from a Parkinson's Disease patient or not. The speech features are in sections based on the technique used to extract them from the speech sample.

The first section is 21 baseline features made up of 5 jitter, 6 shimmer, 5 fundamental frequency parameters, 2 harmonicity parameters, 1 Recurrence Period Density Entropy, 1 Detrended Fluctuation Analysis, and 1 Pitch Period Entropy features. Time frequency features are made up of 3 intensity, 4 formant frequency, and 4 bandwidth features. Vocal fold features consist of 3 glottis quotient, 6 glottal to noise excitation, 7 vocal fold excitation ratio,

[2] https://archive.ics.uci.edu/ml/datasets/Parkinson's+Disease+Classification

and 6 empirical mode decomposition features. There are 84 Mel Frequency Cepstral Coefficients based features. Next, there are 182 wavelet transform based features. Finally, using a tunable Q-factor wavelet transform (TQWT) method that has not previously been applied to Parkinson's Disease speech analysis, 432 features are generated.

4.2 Preprocessing

All features in the available dataset already consisted of numeric values. All features except for ID number and Parkinson's Disease status were used as possible input for neural network models. After reading in these features for all samples, each feature was normalized. First, for every feature, its mean was subtracted from each sample's value for that feature and then divided by the variance of that feature. Next, each feature was scaled by subtracting the minimum value of each feature and then dividing by the difference between the maximum and minimum value. This resulted in all feature values being in the range of 0 to 1 based on their original value.

As a classification problem, output was formatted with one output for each category. In this case, there were only two categories of Parkinson's Disease or healthy individual and the Parkinson's disease status from the dataset was used to label output. So, a healthy individual had output label of [1,0] and a Parkinson's Disease patient would be [0,1].

4.3 Performance Metrics

Models will primarily be compared using accuracy. To this end, every sample in the test data will be run through the model. The number of true

positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) will be totaled. Then, accuracy of the model will be calculated using:

$$acc = \frac{TP + TN}{TP + TN + FP + FN}$$

F_1 score and Matthews Correlation Coefficient (MCC) were previously used to evaluate classification methods using this dataset and will also be calculated. F_1 score is the harmonic average of precision and recall. It calculates to a value in the range of 0 to 1 with 1 resulting from perfect precision and recall (no false positives or false negatives). It is calculated with this formula:

$$F_1\ score = 2\left(\frac{precision\ \times recall}{precision + recall}\right)$$

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

Matthews Correlation Coefficient is a useful measure of accuracy for unbalanced datasets that also ranges from 0 to 1 with 1 resulting in complete correlation. This dataset is unbalanced with 188 Parkinson's Disease patients and 64 healthy individuals. MCC can be calculated as:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

If any of the additions in the denominator result in 0, then the entire denominator will be set as 1.

Sensitivity (Sen), specificity (Spe), positive predictive value (PPV), and negative predictive value (NPV) are important performance measures for diagnostic tests. Sensitivity is a measure of how well the test detects the disease in individuals with the disease. Specificity is a measure of how well the test correctly identifies no disease in individuals that do not have the

disease. PPV and NPV are measures of how likely the positive or negative test results indicate the actual disease status. As in, if the test result is positive, then how likely is the individual actually positive for the disease. The PPV and NPV measurements are affected by prevalence of the disease and the PPV and NPV calculated from the experimental population are only reliable if the test is being used on a similar population. These performance metrics can be calculated using these formulas:

$$sensitivity = \frac{TP}{TP + FN}$$

$$specificity = \frac{TN}{TN + FP}$$

$$PPV = \frac{TP}{TP + FP}$$

$$NPV = \frac{TN}{TN + FN}$$

The number of epochs and amount of time needed until early stopping was also recorded in some instances to use for evaluating computational power needed to train certain models.

4.4 Preliminary Studies

In the Sakar et al. study, for multilayer perceptron, they reported accuracies ranging from 0.75 to 0.78 when using individual non-TQWT feature subsets. They reported an accuracy of 0.85 using certain TQWT settings and 0.84 when using the top 50 features out of all feature subsets as selected by mRMR filter. The use of the top 50 features was reported to be done to avoid the curse of dimensionality (Sakar C. , et al., 2019).

To establish a baseline approximately in the same level of accuracy, initial experiments with a single hidden layer MLP were conducted using educated guesses at possible settings for network shape, activation functions, etc.

Keras (Chollet & Others, 2015) is a framework written in Python that allows quick and simple implementation of neural networks. It can utilize multiple deep learning frameworks as its backend including the popular TensorFlow (Abadi, et al., 2015). Because it allows neural network models to quickly be implemented or modified, it was ideal to conduct this research using Python 3.6 using Keras with TensorFlow backend. Numpy (Oliphant, 2006) and Scikit-Learn (Pedregosa, et al., 2011) Python libraries were also utilized.

For a quick initial test, a multilayer perceptron model using all 753 features as input, one hidden layer of 11 nodes, and 2 output nodes was used. All features were used for the simplicity of not needing to make any guesses regarding feature selection. Some method of feature selection could be used if it was found that using all 753 features resulted in problems with the curse of dimensionality at any point. This was never found to be the case and all features continued to be used as input for all experiments. Two output nodes were used to account for the two possible labels of Parkinson's Disease or no Parkinson's Disease. The number of neurons for the hidden layer was a fortunate choice of a random prime number. Initial weights for all connections were generate randomly.

Dataset was divided into 50% for training, 25% for validation, and 25% for testing. The initial testing accomplished by simply using the first half as training set, the third quarter as validation set, and the fourth quarter as testing set. This produced an average accuracy of 0.660 over 10 tests with the best test having an accuracy of 0.772. However, since the dataset was not evenly distributed, this split results in each set having disparate percentages of Parkinson's Disease patients (78.6% in training, 77.8% in

validation, and 63.5% in testing). This split means that fewer than average samples from healthy individuals was used in training. This might have resulted in formation of less accurate decision regions. The best trial in this initial setup is encouraging as it is around the accuracy of some of the previously reported accuracies with this dataset. The randomness of the initial weights and the possibility of multiple local minimas might explain the variation in final accuracy from training the same model.

Randomizing the samples into sets should hopefully produce more comparably distributed sets. Due to the dataset having 3 samples from each Parkinson's patient or healthy control, simply randomly selecting the training, validation, and testing sets could result in some individuals having samples in 2 or all 3 of the sets. Assuming that each person produces 3 relatively similar samples, then having the model trained to recognize someone that is also in the test set could result in a false increase in accuracy. Using the preliminary model with a fully randomized splitting results in 0.851 average accuracy over 10 different trainings. While randomizing but ensuring all 3 samples of each person were only in 1 set did show a lower average accuracy of 0.831 over 10 different trainings.

Next, testing with one sample from each person showed an average accuracy of 0.861 over 10 different trainings. Since the results appeared to be comparable to using all samples and to previous work, a trimmed dataset using only the first sample of each individual in the full dataset was used for all further testing. This simplifies randomizing to training, validation, and testing sets as there will never be a possibility of an individual's data being in more than one set. Furthermore, using 3 samples could tend toward overfitting as similar noise in the data could show up multiple times due to three similar samples being present. A further benefit might result from each epoch of training being faster by only having to go through a third of the samples compared to before.

With labels being 0 and 1, an activation of "hard sigmoid" was used at the end of the model to push the output further toward one category or the other. At the very beginning, using the sigmoid activation function for neurons resulted in final output values in the range of 0.5 to 1. This was less than optimal since the threshold value for deciding between labels was 0.5. Thus, it was quickly switched to using TanH as the activation function for neurons. This allowed for the output values to below 0.5.

At this point, the dataset has 252 samples overall. Before doing any further experiments, training, validation, and testing sets with relatively similar ratios of between Parkinson's Disease and healthy individual samples was obtained by splitting using a variety of random seeds and checking the percentage with Parkinson's Disease in each set. This eventually generated a training set of 126 samples with 73.8% Parkinson's Disease, validation set of 63 samples with 73.0% Parkinson's Disease, and test set of 63 samples with 77.8% Parkinson's Disease. As these sets appeared relatively evenly distributed, all further experiments with different neural network models would be trained on this training set with evaluation for early stopping using this validation set and testing would be done by having the model predict against this test set.

4.4 Single Hidden Layer MLP

With many settings and model hyperparameters decided during preliminary testing, the major remaining parameter for finding an accurate MLP for classifying Parkinson's Disease would be the shape of the MLP. It is difficult to know now many neurons will produce the best model shape. Thus, optimizing this parameter will be done through experimentation. With a

single hidden layer, it should be relatively easy because the number of neurons in only one layer needs to be changed.

Based on preliminary testing success with 11 internal neurons, MLP models starting with 1 hidden layer neuron and up to 100 hidden layer neurons were generated. As with the preliminary testing, each model was run 10 times to find their average accuracy. If it had appeared that accuracy was still trending higher near the end of this range, then testing would have been continue using models with over 100 hidden layer neurons.

Subsequently, the best few models would be trained and tested 100 times while recording TP, FP, TN, FN, and epochs and clock time until early stopping. This would be used to further evaluate accuracy and the other performance metrics. The best performing instance of each model shape could then be identified.

4.5 Two Hidden Layer MLP

With a two hidden layer MLP, there are naturally two parameters to adjust that can change the shape of the MLP. Using the same search strategy as with the single hidden layer would result in a huge number of models to test. Such a strategy would involve a grid search of all combinations of 1 to 100 neurons in each hidden layer. Thus, a comprehensive search in this manner would require testing of 10000 models. It could be a feasible way with sufficient time and computational power.

It might be possible to use the single hidden layer MLP results to make some decisions about how many neurons to use in each layer for models with multiple hidden layers. The number of neurons needed to be accurate in single layer might be an indication of the number of different patterns in the

data that were useful for the task. Thus, once the single layer model testing produces an upper value after which adding neurons either does not improve accuracy or shows decreased accuracy, then that could be the basis for the shapes of the two hidden layer models that might be viable to try.

With deep learning, the later layers use the previous layers output as their features. Conservatively, this might mean the first hidden layer should also be allowed to find up to the same number of patterns from the input as the single layer models. For simplicity, the range of the second layer could also go up to the upper value of the single layer models. Even using this reasoning to limit the maximum neurons in each layer could still result in a very inefficient comprehensive search. It is probably unnecessary to go up to the upper value in each layer since deep learning has the possibility of using less computational units compared to a network with similar performance using fewer layers (Bengio, 2009).

Furthermore, it might not be necessary or efficient to initially cover every single value in the range. The performance of using 20 neurons verses 21 neurons should likely be relatively similar. Thus, it might be possible to initially cover the search area by testing models spread out over the search area. For this study, an initial guess would be made of each of the two hidden layers of the deeper learning model using up to half the neurons needed in the single layer MLP. In this way, the two layer model with the most neurons would not have more than neurons needed in single layer models. To quickly cover the range, all the prime numbers in the range will be used rather than every value in the range. This should be able to cover the range well without using a specific set interval between test points.

With the second layer acting upon the output of the first layer, the model could be finding combinations of the first layer's outputs that can solve the task. With this reasoning, it could be possible to multiply the number of

neurons in each hidden layer together to get an estimation of the number of overall patterns that the model can handle. Therefore, concentrating on combinations of amounts of neurons that might approximately be able to find the same number of patterns as the upper value of the single layer models might be a viable strategy. Such as, if the upper value for single layer models was 30 hidden layer neurons, then models using 6 and 5, 7 and 4, 8 and 4, and so on could be tested as possibilities for highly accurate two layer models.

Just as with the single layer models, the best few models found using the outlined search strategies would be further evaluated by being trained and tested 100 times while recording TP, FP, TN, FN, and epochs and clock time until early stopping. This would be used to further evaluate accuracy and the other performance metrics. The best performing instance of each model shape could then be identified.

5 Results and Analysis

5.1 Single Hidden Layer MLP

After running training on every shape single hidden layer MLP up to 100 neurons ten times each, average accuracies for each model were calculated. The best model in this initial test had an average accuracy of 0.908 and used 64 neurons in the hidden layer. The top models by average accuracy are shown in Table 1. These top models underwent further testing and evaluation with more performance metrics.

Table 1 Top Initial Single Hidden Layer Models

Neurons	**Average Accuracy**
64	0.9079
53	0.9048
14	0.9032
24	0.9032
31	0.9032
36	0.9032
47	0.9032

The average accuracy of every model tested is shown in Figure 5. This shows that many models exceeded or were close to 0.90 in average accuracy. Over the range of approximately 10 to 65 neurons, most models were relatively close to the best accuracy. Above this range, accuracy declines and does not appear likely to improve again with models with even more neurons.

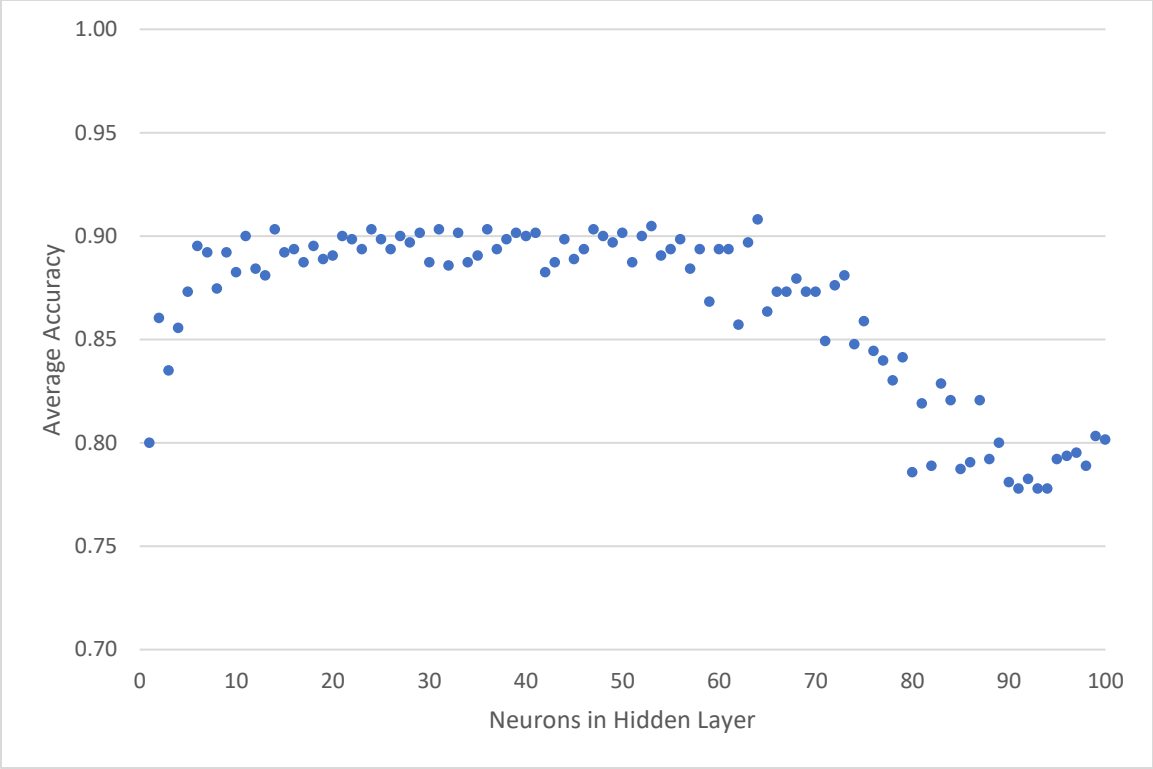

Figure 5 Average Accuracy of Single Hidden Layer MLP Models

The top models listed in Table 1 were each trained 100 times starting with random weights while recording epochs and clock time until early stopping. For every instance trained, testing to determine TP, FP, TN, and FN with the testing set was done and then used to calculate the accuracy, F_1 score, MCC, sensitivity, specificity, PPV, and NPV of that instance.

The average of the performance metrics of each model from this testing is shown in Table 2 while the best performing instance of each model is shown in Table 3. Overall, each of these models continued to have high average performance. On average, the model using 47 hidden layer neurons was the best performing with just over 0.901 accuracy, F_1 score of 0.940, and MCC of 0.699. The other models were not far behind. All models had high average sensitivity, PPV, and NPV. As can be seen in Table 3, nearly every

model tested had at least one instance where accuracy was 0.937, F_1 score of 0.961, and MCC of 0.813. These instances all had sensitivity and NPV of 1.00 as there were no false negatives. This indicates the test is very good at finding all individuals with Parkinson's Disease and that a negative result is very reliable to mean the individual does not have Parkinson's Disease. PPV of 0.925 means that a positive test result is highly likely to mean the tested individual has Parkinson's disease.

Table 2 Average Metrics of Top Single Layer MLPs

Neurons	Accuracy	F_1 score	MCC	Sen	Spe	PPV	NPV
14	0.8949	0.9372	0.6762	0.9951	0.5480	0.8861	0.9561
24	0.8910	0.9350	0.6678	0.9961	0.5259	0.8814	0.9829
31	0.8976	0.9387	0.6899	0.9961	0.5552	0.8878	0.9822
36	0.8968	0.9382	0.6877	0.9978	0.5459	0.8856	0.9897
47	0.9006	0.9403	0.6993	0.9959	0.5689	0.8909	0.9810
53	0.8914	0.9354	0.6525	0.9941	0.5364	0.8843	0.9141
64	0.8724	0.9254	0.5444	0.9937	0.4514	0.8678	0.7622

Table 3 Best Performing Instance of Each Top Single Layer MLP

Neurons	Accuracy	F_1 score	MCC	Sen	Spe	PPV	NPV
14	0.9365	0.9608	0.8126	1.0000	0.7143	0.9245	1.0000
24	0.9206	0.9515	0.7638	1.0000	0.6429	0.9074	1.0000
31	0.9365	0.9608	0.8126	1.0000	0.7143	0.9245	1.0000
36	0.9365	0.9608	0.8126	1.0000	0.7143	0.9245	1.0000
47	0.9365	0.9608	0.8126	1.0000	0.7143	0.9245	1.0000
53	0.9365	0.9608	0.8126	1.0000	0.7143	0.9245	1.0000
64	0.9365	0.9608	0.8126	1.0000	0.7143	0.9245	1.0000

On average, specificity was in the mid-0.500s but rose to 0.714 in the best performing instances. With the unbalanced dataset, Parkinson's Disease is

the more common label. Thus, the trained network might err on the side of labeling samples as Parkinson's Disease since it is more likely to be correct and result in some false positives. However, the overall low number of false positives means the PPV remains high. So, this level of specificity along with the very high sensitivity would be consistent with the likely use of this classification method as a quick and easy screening test that could prompt further evaluation for positive results.

The 64 neuron model had the highest average accuracy in the initial testing survey but slightly dropped in accuracy in the more extensive testing. The 64 model neuron was right at the level where increasing neurons in the hidden layer started having decreasing average accuracy. As its best performing instance matched the best performance of the other models, the initial overperformance might have been due to random chance of having high accuracy results in the initial ten trials. As there is some variance in performance, training a model multiple times and then recording the weights of the best performing instance appears to be necessary to achieve the most accurate model possible.

5.2 Two Hidden Layer MLP

Since accuracy in the single hidden layer MLP appears to start decreasing after approximately 65 neurons as seen in Figure 5, initial testing with two hidden layer MLP models will use combinations of all prime numbers up to 31 for the neurons in each layer. Since initial survey of single layer models had highest average accuracy with 64 neurons, testing of two hidden layer MLP models where the number of neurons in the hidden layers multiplied to a value near 64 (range of 56 to 72) was also used as a search strategy for potentially accurate models. The prime number grid search generates 121

models to evaluate while the "64 combination" search generated 61 models to evaluate.

The average accuracies of an initial survey of 10 different trainings of each model in these search strategies are visualized in Figure 6 and Figure 7. The prime number grid search shows many peaks and valleys in average accuracy. There is not as clear of a plateau of high accuracy as seen in Figure 5 with single hidden layer models. However, there appears to be an overall trend with better accuracy in higher numbers of neurons in the first hidden layer and lower numbers of neurons in the second hidden layer. The "64 combination" search method generated models that formed a curve through the prime number grid area. These form a continuum of low first layer-high second layer neuron count to low-moderate neuron count in both layers to high first layer-low second layer neuron count. Similar to the grid search, there appears to be higher accuracy in the high first layer-low second layer neuron models.

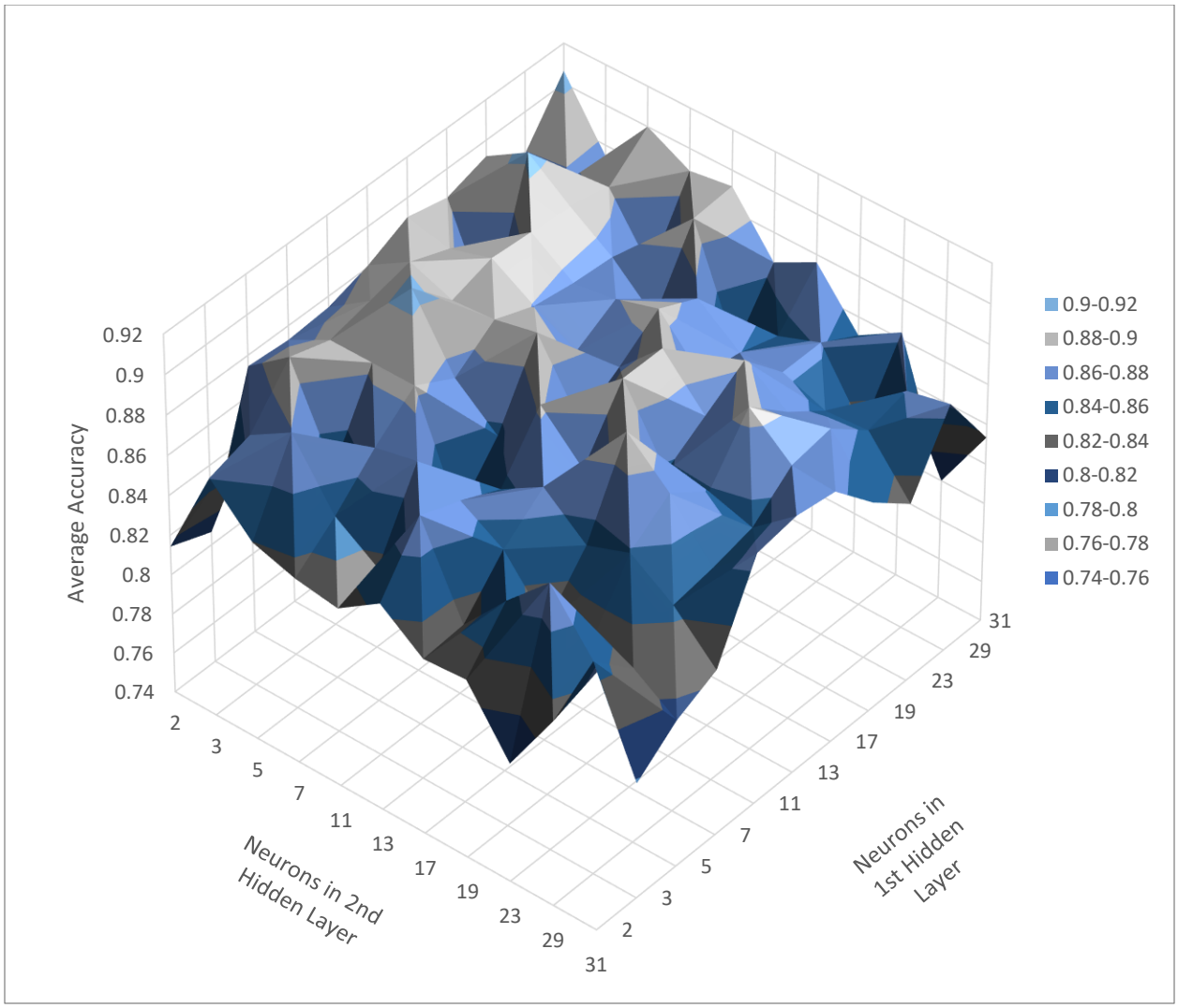

Figure 6 Average Accuracy of Two Hidden Layer Models (Prime Number Grid)

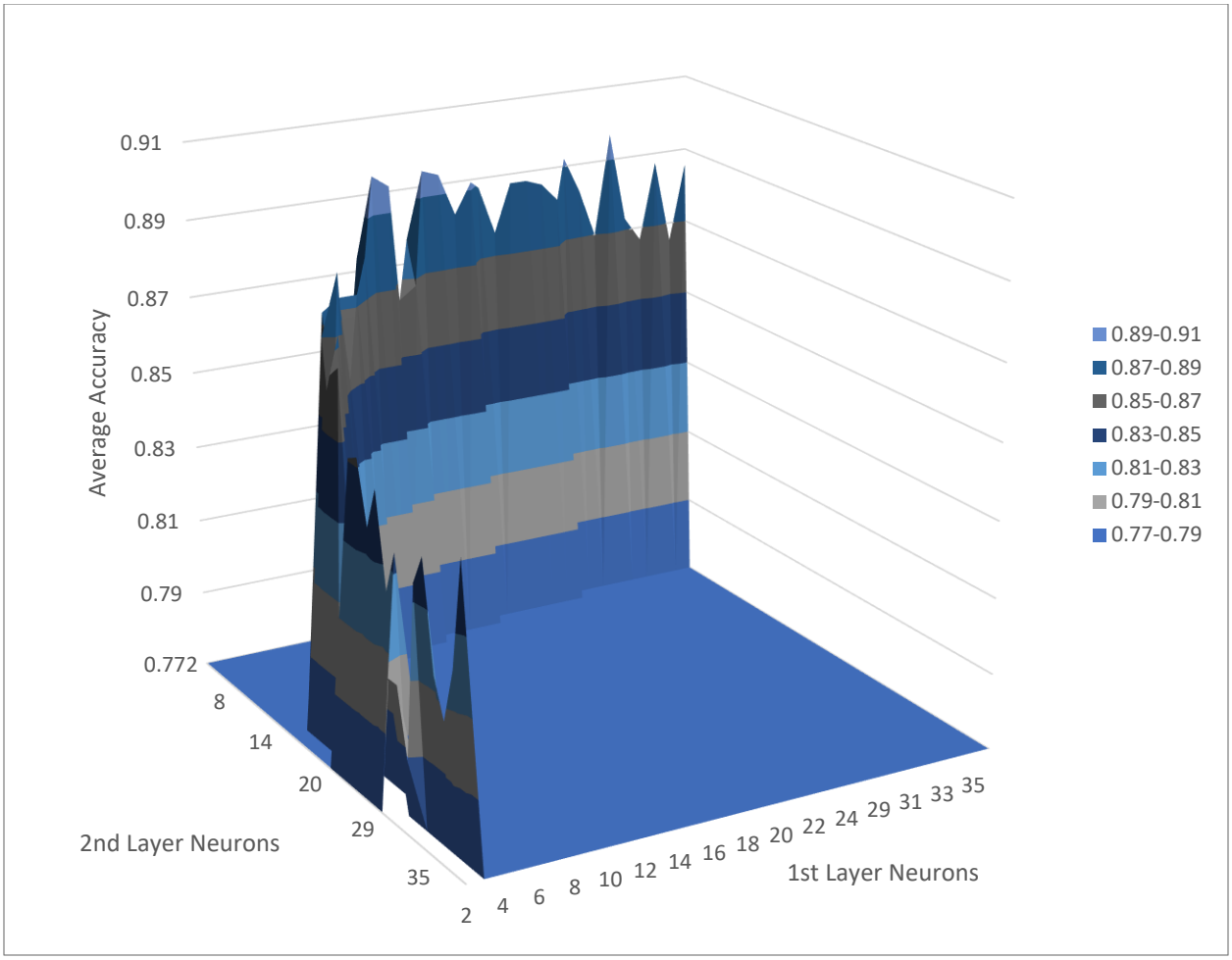

Figure 7 Average Accuracy of Two Hidden Layer Models (Combination Search)

The best models with over 0.9 average accuracy from either of these search methods are shown in Table 4. The first 4 models are from the prime number grid method and the last model is from the "64 combination" method. As with the single layer evaluation, these top models were each trained 100 times starting with random weights while recording epochs and clock time until early stopping. For every instance trained, testing to determine TP, FP, TN, and FN with the testing set was done and then used to

calculate the accuracy, F_1 score, MCC, sensitivity, specificity, PPV, and NPV of that instance.

Table 4 Top Initial Two Hidden Layer Models

1st Layer Neurons	2nd Layer Neurons	Average Accuracy
11	5	0.9063
31	2	0.9063
23	3	0.9048
31	5	0.9000
11	6	0.9000

The average performance metric values for the best two layer models from the initial survey are shown in Table 5. The 31-5 neuron model was the best average performer with 0.886 accuracy, 0.931 F_1 score, and MCC of 0.641. All the models tested averaged very similarly in each performance metric. These results are similar to that of the single layer models seen in Table 2 with the single layer models averaging slightly higher.

Table 5 Average Metrics of Top Two Layer MLPs

Neurons								
Layer 1	Layer 2	Accuracy	F_1 score	MCC	Sen	Spe	PPV	NPV
11	5	0.8705	0.9220	0.5582	0.9716	0.5171	0.8806	0.7514
23	3	0.8817	0.9274	0.6218	0.9627	0.5986	0.8978	0.8065
31	2	0.8721	0.9232	0.5634	0.9706	0.5285	0.8838	0.7527
31	5	0.8859	0.9306	0.6405	0.9733	0.5813	0.8940	0.8704
11	6	0.8797	0.9274	0.6076	0.9790	0.5321	0.8835	0.8601

However, when examining the best performing instances of each two layer model shown in Table 6, two models had instances that were able to train to

a better accuracy than any instance of any single layer model. Both 23-3 and 31-2 models had instances reaching 0.952 accuracy with F_1 score of 0.970 and MCC of 0.861. This is rather significant as the best performing single layer models only had 4 false positives with 0 false negatives. This means these instances of the two layer models were able to correctly label one of those 4 false positives as a true negative.

Table 6 Best Performing Instance of Each Top Two Layer MLP

Neurons								
Layer 1	**Layer 2**	**Accuracy**	**F_1 score**	**MCC**	**Sen**	**Spe**	**PPV**	**NPV**
11	5	0.9365	0.9608	0.8126	1.0000	0.7143	0.9245	1.0000
23	3	0.9524	0.9703	0.8605	1.0000	0.7857	0.9423	1.0000
31	2	0.9524	0.9703	0.8605	1.0000	0.7857	0.9423	1.0000
31	5	0.9365	0.9608	0.8126	1.0000	0.7143	0.9245	1.0000
11	6	0.9365	0.9608	0.8126	1.0000	0.7143	0.9245	1.0000

Specificity of the two hidden layer models was similar to the single hidden layer models but with the best instances reaching a slightly higher to 0.786. Again, the PPV remains high and using these models would be viable as a screening test that could prompt further evaluation for Parkinson's Disease.

5.3 Comparison of Training Metrics

To evaluate the training of these models, epochs and clock time (in seconds) until early stopping for each training was recorded. This was only done while training the top models after the initial single layer and two layer model surveys. This means the average values are calculated from 100 trainings.

Since the number of neurons in the hidden layer affects the number of weights that need to be updated, that should affect the computational power needed to complete the training. The number of weights in each single layer model was calculated by multiplying the number of inputs with hidden layer neurons and adding that to hidden layer neurons multiplied by 2 output neurons. The weights in each two layer model were calculated by adding the results of multiplying the number of inputs with first hidden layer neurons, the first hidden layer with the second hidden layer, and the second hidden layer neurons with 2 output neurons. Thus, for the model with 11 neurons in the first layer and 5 neurons in the second layer, this would be 753 * 11 = 8283 connections between inputs and first layer plus 11 * 5 = 55 connections between two hidden layers plus 5 * 2 = 10 connections between second layer and output neurons for a total of 8348 connections.

Since early stopping was used, the number of epochs needed to train each instance of every model could vary significantly. An instance that needed more epochs to train would naturally need more computing power. Thus, to be able to make comparisons between models, a metric of "epoch-weights" (similar to kilowatt-hours) was used. It is simply the multiplication of epochs needed for training with weights in the model.

As Table 7 indicates, training each model used just under 30 epochs on average regardless of the number of hidden neurons. Training of the instance with the best accuracy appeared to usually take approximately the average amount of epochs as other instances of the same model shape. The best 47 neuron and 64 neuron instances did take about 50% more epochs and clock time to be trained than average for their model shape. The expectation is that more neurons should require more computing power. The "epoch-weights" metric follows this trend. The recorded clock times do initially appear to follow this trend, but then the 53 and 64 neuron models have a drastic drop in clock times. It is possible that this measurement of

training performance could be reliable if available computing power on the machine running the neural network is consistent.

Table 7 Single Layer MLP Training Metrics

	Average			Best Instance		
Neurons	**Epochs**	**Clock Time**	**Epoch-Weights**	**Epochs**	**Clock Time**	**Epoch-Weights**
14	29.47	35.91	311498	36	38.22	380520
24	28.26	47.14	512071	33	47.56	597960
31	28.11	66.87	657915	29	64.69	678745
36	27.23	81.70	740111	27	73.84	733860
47	29.69	107.60	1053550	49	149.42	1738765
53	29.91	25.10	1196849	35	25.92	1400525
64	28.74	13.64	1388717	45	20.92	2174400

Table 8 shows the training metrics of the tested two layer models. Once again, the clock times may not be reliable as a metric. The 11-5 and 11-6 models are very similar in shape but recorded vastly different clock times while using comparable amounts of epochs until early stopping.

Table 8 Two Layer MLP Training Metrics

Neurons		**Average**			**Best Instance**		
Layer 1	**Layer 2**	**Epochs**	**Clock Time**	**Epoch-Weights**	**Epochs**	**Clock Time**	**Epoch-Weights**
11	5	40.06	20.16	334421	31	22.65	258788
23	3	55.27	77.74	961366	32	44.47	556608
31	2	74.80	70.79	1750993	34	28.86	795906
31	5	40.57	86.62	953720	40	74.27	940320
11	6	36.08	106.48	301665	27	72.86	225747

In terms of epochs of training, all the two layer models needed more epochs on average than any of the single layer models. This indicates that while there might be fewer weights to update, it seems to take more epochs to train those weights for this task. The 14 neuron single layer model has 10570 weights while the 11-5 neuron model with more neurons only has 8348 weights. However, by the "epoch-weights" metric, the 11-5 model required slightly more computational power for training than the 14 neuron model. Aside from the 31-2 model, two layer models had relatively similar "epoch-weight" measurements to single layer models with similar amounts of total neurons. Overall, it appears that for this task and implementation, two layer deep learning likely uses slightly more computational power for training. Interestingly, training the best instance of each two layer model performed much better than the average number of epochs need for the same model.

6 Conclusion

6.2 Summary

Multilayer perceptron is potentially a very useful classification method for diagnosing Parkinson's Disease using extracted speech features. After systematically surveying many network shapes, some two hidden layer deep learning MLP was found to be able to reach 0.952 accuracy with F_1 score of 0.970 and MCC of 0.861. This is an improvement compared to previous classification methods using this dataset. Even though multilayer perceptron was already previously successfully utilized with this dataset, it was still possible to further improve accuracy using multilayer perceptron by optimizing hyperparameters and adding more layers for deep learning. By increasing the accuracy to this level with this larger dataset of Parkinson's Disease speech features, there is an increased likelihood that use of speech to classify Parkinson's Disease with this method will be applicable to the general population.

As demonstrated, by optimizing the number of neurons in the single hidden layer MLP, the models in the optimal range of hidden layer neurons were on average 10 to 15 percentage points more accurate. While it was relatively quick to run through 100 different models, a broad plateau of higher accuracy models was found for this data. This suggests that it would be efficient to survey network shapes at higher increments initially and then subsequently do more detailed evaluation of network shapes around the better performing initial shapes to potentially find higher accuracy shapes. However, a comprehensive search could still be used if allowed by having sufficient time or computational power available.

While the single hidden layer models already reached high accuracy, in this case, using deep learning with a two hidden layer model did improve accuracy slightly. Having the first layer generate features for the second layer may have allowed for a more complex decision area to be generated. Thus, with proper training, a deeper model could be more able to correctly map a decision between Parkinson's Disease samples and healthy individual samples that are relatively close in the data space.

In this case, using results from single hidden layer models was able to guide the search for two hidden layer models. Two methods were used. In the first, the range of the number of neurons in each of the two layers was restricted using the highest neuron single layer models that had high average accuracy. By seeing that a broad range of models all had similarly high accuracy in the single layer experiments, values to use in the range were spaced out to cover the range without needing to comprehensively try every value. In the second method, models where the first and second layer neuron counts multiplied together to be close to the highest single layer model neuron count were included. This would allow the two layer models to have approximately the same amount of combinations of patterns as the patterns the single hidden layer could recognize. Both of these search methods were found to be able to discover viable two layer models that were approximately as accurate as the single layer models. Using these search methods required examining significantly fewer network shapes than a comprehensive search would have required. Furthermore, a couple of the two layer models found this way demonstrated instances with the highest accuracy at Parkinson's Disease classification out of all models evaluated.

While it does seem that 753 features for 756 samples might result in difficulties with the curse of dimensionality, it did not arise as a problem in this case. Having all features available to train on may have allowed the MLP to find a better decision area than it would if restricted to specific

subsets of the data or with algorithmically-decided top features. In this implementation, using only 1 sample instead of 3 from each individual in the dataset meant only having 252 total samples with just half being used for training. The curse of dimensionality still did not present as a problem. In general, the apparent best practice would be to try initially using as much data as is available for input and then subsequently use some method to cut down the number of features only if necessary.

After optimizing the shape of single and two hidden layer multilayer perceptrons for this research, the best models were almost always able to identify every or nearly every Parkinson's Disease patient. With the best instances finding all Parkinson's Disease patients, this method was able to achieve 100% sensitivity. With zero false negatives, the negative predictive value was 1.00 and receiving a negative test result would indicate very high confidence to exclude a person from having Parkinson's Disease. Specificity was quite high at 78.6% in the best instances since a few healthy individuals still ended up as false positives. However, positive predictive value was very reliable at 0.942. Use of multilayer perceptron for classification using speech samples has the potential to be a very good screening test for Parkinson's Disease. In addition to the high sensitivity of this method, it would be quick, non-invasive, and would not require a clinical or laboratory visit.

6.2 Future Work

While the multilayer perceptron models identified in this research are already very accurate, there can still be improvement. This might be accomplished by using deep learning with even more layers or using another neural network architecture.

Further expanding the dataset by adding speech samples from more individuals could allow stronger demonstration that this method would be applicable for testing the general population. While neural network models found using the methods in this research show high correlation when evaluated using Matthew's Correlation Coefficient, making the dataset more balanced by adding more samples from healthy individuals would be relatively easy and could allow improved evaluation of Parkinson's Disease classification methods.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., . . . Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Retrieved from https://www.tensorflow.org/

Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning, 2*(1), 1-127.

Brabenec, L., Mekyska, J., Galaz, Z., & Rektorova, I. (2017, March). Speech disorders in Parkinson's disease: early diagnostics and effects of medication and brain stimulation. *Journal of Neural Transmission, 124*(3), 303–334.

Choi, E., Bahadori, M. T., Schuetz, A., Stewart, W. F., & Jimeng, S. (2016, August). Doctor AI: Predicting Clinical Events via Recurrent Neural. *JMLR Workshop Conf Proc, 56*, 301-318.

Chollet, F., & Others. (2015). Keras. Retrieved from https://keras.io

Claesen, M., & De Moor, B. (2015). Hyperparameter Search in Machine Learning. *Computing Research Repository*, 14-1 - 14-5.

Davie, C. A. (2008, June). A review of Parkinson's disease. *British Medical Bulletin, 86*(1), 109–127.

Ding, X., Zhang, Y., Liu, T., & Duan, J. (2015). Deep Learning for Event-Driven Stock Prediction. *Proceedings of the 24th International Conference on Artificial Intelligence*, 2327-2333.

Ghiassi, M., Lio, D., & Moon, B. (2015). Pre-production forecasting of movie revenues with a dynamic artificial neural network. *Expert Syst. Appl*, 3176-3193.

Gürüler, H. (2017, July). A novel diagnosis system for Parkinson's disease using complex-valued artificial neural network with k-means clustering

feature weighting method. *Neural Computing and Applications, 28*(7), 1657–1666.

Ho, A. K., Iansek, R., Marigliani, C., Bradshaw, J. L., & Gates, S. (1998). Speech Impairment in a Large Sample of Patients with Parkinson's Disease. *Behavioural Neurology, 11*(3), 131-137.

Jankovic, J. (2008). Parkinson's disease: clinical features and diagnosis. *Journal of Neurology, Neurosurgery & Psychiatry, 79*, 368-376.

Khare, K., Darekar, O., Gupta, P., & Attar, V. (2017). Short term stock price prediction using deep learning. *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, (pp. 482-486). Bangalore.

Lahmiri, S., Dawson, D. A., & Shmuel, A. (2018, Feb). Performance of machine learning methods in diagnosing Parkinson's disease based on dysphonia measures. *Biomed Eng Lett, 8*(1), 29–39.

LeCun, Y., Bengio, Y., & Hinton, G. E. (2015). Deep Learning. *Nature, 521*, 436-444.

Lippmann, R. P. (1987, April). An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine, 4*(2), 4 - 22.

Little, M. A., McSharry, P. E., Hunter, E. J., Spielman, J., & Ramig, L. O. (2009). Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *IEEE Transactions on Bio-medical Engineering, 56*(4), 1015 - 1022.

Logemann, J., Fisher, H. B., Boshes, B., & Blonsky, E. R. (1978). Frequency and Cooccurrence of Vocal Tract Dysfunctions in the Speech of a Large Sample of Parkinson Patients. *Journal of Speech and Hearing Disorders, 43*, 47-57.

Marsland, S. (2015). *Machine Learning: An Algorithmic Perspective* (2 ed.). Boca Raton, FL: CRC Press.

McCulloch, W. S., & Pitts, W. (1943, December). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics, 5*(4), 115-133.

Minsky, M., & Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry.* Cambridge, MA: MIT Press.

Oliphant, T. E. (2006). *A Guide to NumPy.* USA: Trelgol Publishing.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Bertrand, T., Grisel, O., . . . Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830.

Peker, M., Şen, B., & Delen, D. (2015). Computer-aided diagnosis of Parkinson's disease using complex-valued neural networks and mRMR feature selection algorithm. *Journal of Healthcare Engineering, 6*(3), 281–302.

Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review, 65*(6), 386-408.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986a). Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, & PDP Research Group, CORPORATE, *Parallel distributed processing: explorations in the microstructure of cognition* (Vol. 1, pp. 318-362). Cambridge, MA: MIT Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986b). Learning representations by back-propagating errors. *Nature, 323*, 533–536.

Sakar, B. E., Isenkul, M. E., Sakar, C. O., Sertbas, A., Fikret, G., Delil, S., . . . Kursun, O. (2013, July). Collection and Analysis of a Parkinson Speech Dataset With Multiple Types of Sound Recordings. *IEEE Journal of Biomedical and Health Informatics, 17*(4), 828-834.

Sakar, B. E., Serbes, G., & Sakar, C. O. (2017). Analyzing the effectiveness of vocal features in early telediagnosis of Parkinson's disease. *PLOS ONE, 12*(8), e0182428.

Sakar, C., Serbes, G., Gunduz, A., Tunc, H., Nizam, H., Sakar, B., . . . Apaydin, H. (2019, January). A comparative analysis of speech signal processing algorithms for Parkinson’s disease classification and the use of the tunable Q-factor wavelet transform. *Applied Soft Computing, Volume 74*, 255-263. doi:https://doi.org/10.1016/j.asoc.2018.10.022

Suhara, Y., Xu, Y., & Pentland, A. (2017). DeepMood: Forecasting Depressed Mood Based on Self-Reported Histories via Recurrent Neural Networks. *In Proceedings of the 26th International Conference on World Wide Web (WWW '17)*, 715-724.

Tsanas, A., Little, M., McSharry, P., & Ramig, L. (2010, April). Accurate telemonitoring of parkinsons disease progression by noninvasive speech tests. *IEEE Transactions on Biomedical Engineering, 57*(4), 884-893.

Wu, S., Chen, Y.-C., Li, X., Wu, A.-C., You, J.-J., & Zheng, W.-S. (2016). An enhanced deep feature representation for person re-identification. *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1-8. doi:10.1109/WACV.2016.7477681

Xue, H., Bai, Y., Hu, H., & Liang, H. (2018). Influenza Activity Surveillance Based on Multiple Regression Model and Artificial Neural Network. *IEEE Access, 6*, 563-575.

Ye, X., Fang, F., Wu, J., Bunescu, R., & Liu, C. (2018). Bug Report Classification Using LSTM Architecture for More Accurate Software Defect Locating. *2018 17th IEEE International Conference on Machine Learning and Applications*, 1438-1445.

Yu, R., Li, Y., Shahabi, C., Demiryurek, U., & Liu, Y. (2017). Deep Learning: A Generic Approach for Extreme Condition Traffic Forecasting. *Proceedings of the 2017 SIAM International Conference on Data Mining*, 777-785.