Correlating Intended Muscle Movements with Ultrasound Imaging

A Major Qualifying Project

Submitted to the faculty of

WORCESTER POLYTECHNIC INSTITUTE

Worcester, MA

In partial fulfillment of the requirements for the degree of Bachelor of Science

Submitted by:

Cormac Lynch-Collier

Submitted on:

December 13, 2019

Advised by:

Professor Haichong (Kai) Zhang

Table of Contents

Abstract

This project investigates machine learning algorithms in the field of prosthetics. The machine learning model was created to predict finger movements when provided with an ultrasound image of the forearm. Convolutional neural networks, recurrent neural networks, support vector machines and k-nearest neighbor algorithms were developed and tested against one another. Upon completion of research, a convolutional neural network was deemed to be the most effective in this application. The code for the final convolutional network was submitted as an appendix (B) in this paper.

Introduction

Ultrasound technology presents a new and relatively unexplored frontier for the future of wearable prosthetics. Current commercial prosthetics use electromyography (EMG) technology to determine musculoskeletal functions, making use of the EMG signal generated by muscle movement [1]. Although a passable solution, EMG-based prosthetics are hindered by the limitations of said technology. Restraints include the inability to differentiate between deep muscle movements due to crosstalk and attenuation, low signal-to-noise ratio, and the lack of a robust graded signal [2]. Ultrasound using sonomyography (SMG) can overcome these hindrances to create a better more accurate prosthetic.

So far, there has been limited research into the use of ultrasound images and intended muscle movements. In existing research projects, researchers have used machine learning to improve the application's recognition of certain muscle movements. A deeper look at the technology used, shows that neural networks have been the primary source of artificial intelligence in these applications.

Although ultrasound in the field of prosthetics has been relatively unexplored, research in neural networks and the broader field of machine learning has been constantly increasing. Many papers have been written that conduct analysis on different types of networks and informal writing has been produced to guide software engineers through the process of creating their own neural networks.

This MQP developed a neural network for predicting hand movement when provided with ultrasound images. The process undertook an analysis of different types of networks, developing an algorithm for each and comparing, in the context of an amputee's intended muscle movements, to discover the most effective model for this application.

Background

History of Prosthetics

Humans have a long history of overcoming loss of limb; prosthetics can be traced back to B.C.E and have been improving ever since. Early prosthetics were simple, uncomplicated devices. The first of these is believed to have been a prosthetic toe, dated between 950-710 B.C.E. Interestingly, it was not something more important to the body, such as a hand or an arm, this may have been all they were capable of creating at that time [3]. The first prosthetic hand was recorded in 70 A.D. however, it details a story from 200 B.C.E of a general losing his hand battle, only to return with an iron hand prosthetic. Used especially in wartime, prosthetics often aided those who lost limbs in battle. In 1505, a german knight was able to use an iron hand with fingers that could be flexed into different positions in order to hold his reigns and grip weapons [3]. Initial prosthetics were devices made for specific uses and could not be body controlled due to the limitations of the existing technology

The first concept of a body-controlled limb was developed by a German dentist, Peter Baliff, in 1818. Using transmission tension through straps, the device allowed the wearer to trigger prosthetic hand movements from muscle movement in the shoulder and upper arm. Designs marginally improved in the following century from other scientists attempting to find a solution [3].

The first and second world war brought the concept of prosthetic limbs to public attention. During World War One, over 4,400 Americans lost a limb, and in World War Two: 3,475. These resulted in increased awareness throughout the country, and the world, as other participating nations suffered similar losses. In addition to awareness, the US Committee on Prosthetics Research and Development was created in 1945. Demand often prompts innovation and as loss of limb became more prevalent so did prosthetic research [3].

Following the second world war the first myoelectric prosthesis was created, enabling surface electromyography (EMG) potentials to power motorized parts. This innovation was a

major step forward in prosthetic research. In 1960 the first clinically significant myoelectric device was unveiled and by the 1980s they were being used in rehabilitation centers [3]. Moving forward to modern day, innovation with respect to prosthetics now centers around materials and physical creation. Advanced plastic and carbon fiber present opportunities for new prosthetic designs, and technology advances allow for new fully-autonomous functions such as gripping and walking [4]. Prosthetics have come a long way since simple static hands, now they are semi- and in some cases fully-autonomous devices. However, despite these improvements, there are still major opportunities to improve the field of artificial limbs.

Modern Prosthetics

When analyzing modern prosthetics, certain terms have been developed to categorize the growing field. There are four types of prosthetic limbs: transtibial, transfemoral, transradial, and transhumeral. Firstly, transradial prosthesis replaces an arm below the elbow. They can either work by harness cables attached to the shoulder (similar to some of the early prosthetics) or they can be myoelectric, sensing muscle movement in the upper arm. Secondly, transhumeral, replaces an arm missing above the elbow. Unlike transradial, there is no upper arm to utilize when developing a prosthetic, making it considerably harder to create an effective artificial limb. Next, transtibial replaces a leg missing below the knee. Retaining the knee makes it easier for those amputees to regain some movement, as the joint controls a considerable amount of the leg. Lastly, transfemoral replaces a leg missing above the knee, Similarly to transhumeral, it is difficult to replicate leg movement when the key joint is missing (the knee), as well as the upper leg [5].

Myoelectric prosthetics are controlled by electromyographic (EMG) signals generated by the limb. The signals are then picked up by the sockets of the prosthetic and generate some sort of intended movement [6]. EMG is a diagnostic procedure that evaluates the health condition of muscles and the nerve cells that control them. These nerve cells transmit electrical signals that cause muscles to tense and relax [7]. In addition to prosthetics, EMG is used in other biomedical applications such as human exoskeletons or even in computer science research involving human computer interaction [8].

Unfortunately, this technology can present some problems when being used for prosthetics. Firstly, sometimes the muscles that are being sensed monitored for EMG signals do not indicate all necessary movements. Take a hand prosthetic as an example. For many years the best a prosthetic could do was open and close the hand, allowing an amputee to grip objects. Recently, new prosthetics have been developed that allow all fingers to flex. However, these specific prosthetics need to be able to sense the exact regions that generate specific EMG signal, meaning that only some amputees can utilize said prosthetic [9]. Another common issue with EMG based myoelectric prosthetics can be response time [10]. Prosthetics need to operate on the scale of milliseconds or even smaller, anything else is simply not good enough. One of the main reasons that response time becomes an issue is because of the signal-to-noise ratio, in other words, the ratio of the energy in the EMG signals to the energy in the noise signal. Noise can come from any other electrical signals that are not related to EMG [11]. The prosthetic must perform the difficult process of decoding these signals, and isolating the important ones. Despite extremely powerful modern prosthetics there are still some problems that hamper the technology.

Ultrasound Technology

Ultrasound technology is often associated with pregnancy and determining a child's sex or due date, but it can also be used for much more. It works by using high frequency sound waves and their echoes, similar to the way bats and dolphins sense, or even SONAR used by submarines. The waves are reflected off of muscles to display images of the viewed area [12]. There are two types of ultrasound probes and images: A-mode and B-mode. A-mode, often referred to as one-dimensional, returns the depth on the x-axis and amplitude on the y-axis. B-mode on the other hand, returns amplitude across two dimensions and with corresponding variations in brightness relative to depth. B-mode is what is what is often referred to as an "ultrasound image"; baby pictures in the pregnancy stage are B-mode images.

Ultrasound (SMG) can be useful for monitoring muscle movement and can be applied as an alternative to EMG. As was previously mentioned, EMG suffers from issues involving single attenuation and crosstalk. Additionally, they can only indicate whether a muscle directly

beneath it has been contracted. SMG on the other hand, can specify how contracted a muscle is, and does not suffer from signal and crosstalk issues. It can also detect muscles at a deeper depth than EMG can [13].

Using ultrasound technology, information about muscle contraction, thickness and location can be extracted. Images can be processed by machine learning models to produce expected outputs such as joint flexion and torque [14]. The technology could be applied to the current uses of EMG including prosthetics and exoskeletons.

Research has already begun on using ultrasound images of the forearm in the context of prosthetics. Within the forearm there exist four major muscle groups: anterior flexors, the posterior extensors, the lateral extensors-supinators, and medial flexor-pronators. These flexors and extensors are responsible for the control of individual digits and tendon movement. Flexion of the proximal and distal interphalangeal joints are controlled by two muscles, the flexor digitorum profundus (FDP) and flexor digitorum superficialis (FDS). These muscles all contribute to the movement of index, middle, ring and pinky finger. The thumb is controlled by the flexor pollicis longus (FPL), extensor pollicis longus, and abductor pollicis longus [2]. These controlling muscles are all visible through ultrasound imaging of the forearm.

Certain experiments have been conducted in this field of research. The first experiment instructed subjects to perform fifteen distinct hand motions with seven repetitions. The forearm images were analyzed and patterns detected. The second experiment had the subjects perform different hand movements and results were controlled in real-time. The hand motions were classified and processed to see if the images could be accurately associated with the hand movements. The third experiment dealt with evaluating graded control. The flexion of the fingers were tracked through the Vicon 3-D motion capture system. Markers were placed on the fingers and on the probe. The system was able to track the markers and flexion of individual. The fourth experiment was split to focus on a variation of the arm/forearm positions and variations of wrist positions.

The results of the experiments were used to classify individual finger flexion with an expectation to later be used for more complex hand movements. Through these experiments

the classification accuracy was 91% and Real-time imaged-based classification accuracy was 92%. These results prove the feasibility of using ultrasound in a hand prosthetic [2].

Neural Networks

A Neural network is a set of algorithms, modeled loosely on the human brain, that can be used to recognize patterns. The patterns detected are classified as numbers, and can be derived from inputs such as images, text, sound, etc [15]. Within the realm of machine learning and neural networks, one very important concept is the idea of supervised v.s. unsupervised learning. The majority of machine learning models use supervised learning; there is an input x and an output y, and the algorithm learns a mapping function between them. An example of supervised learning is speech recognition; models are trained with a variety of people (x input) to recognize certain words (y output). Unsupervised learning just consumes an input x and is tasked to identify patterns on its own with no instruction on what to look for [16]. An example of this could be constructing sentences; a model is given a large data set of english sentences (x input) and is told to output a reasonably structured sentence based on that input.

Neural networks are a type of supervised machine learning. In essence neural networks cluster and classify. They make connections between data and recognize patterns so that predictions can be made. In training, the model tries to associate inputs with outputs by assigning weights to nodes that the input is processed by. The model then validates using a separate set of data to see how accurately it can predict outputs. With those results it performs a technique known as back-propagation to re-weight aspects of the model in order to better predict in the future. This training process is iterated upon to create a more accurate model [17]. This is a very broad definition of how neural networks work, within the field there are many different types that operate in slightly different ways to improve results.

One extremely common application of neural networks is image recognition. There has been much research surrounding this topic with many different neural network algorithms that can perform image recognition. One such algorithm is a convolutional neural network (CNN). In image recognition CNNs use one (or more) “convolutional” layer(s) to convert a large input

matrix (ie. pixels of an image) into a smaller matrix, extracting the more important features of the input (ie. edges of objects in an image) [18].

Another type of neural networks that is popular for image classification are support vector machines (SVM). An SVM is a supervised learning algorithm, training data is passed in with a label, telling the model exactly what that data is and how to classify it. The goal of using SVM is to lower generalization error by separating differently labeled data with a hyperplane, thus maximizing the margin between sets [19].

K Nearest Neighbor is another type of neural network that can be used for image classification. The underlying principle in nearest neighbor classification is that similar things exist in close proximity. This makes sense for image classification as similar images are naturally close to one another. Nearest neighbor will consume an input and place the image somewhere on a graph or scale. It will then see what images are nearest to it and assume that the output associated with the input image is the same as those around it [20]. In this way, the nearest neighbor algorithm can be applied to image classification.

The previously mentioned neural nets are used to classify images, however in some cases, the model may need to be able to classify video. Recurrent neural networks provide a way to analyze the sequence of frames that make up an action rather than just a single image. Recurrent neural networks work by storing previous analyzed frames in memory. After each frame a new set of predictions are produced, until eventually, all the frames can be associated with one prediction [21].

Objectives and Design

Client Statement

Create a neural network that will predict hand movements when provided with ultrasound images of the forearm.

The following diagram was constructed in order to complete the goal defined by the client statement:

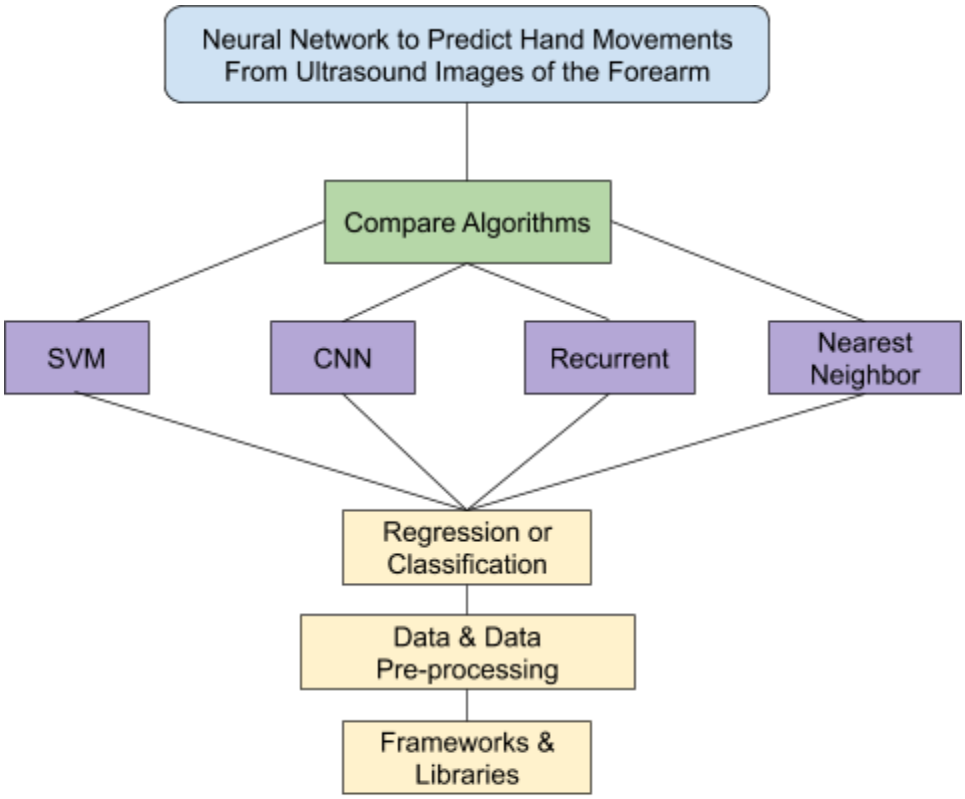


Figure 1: Design Process: From the Bottom to the Top

Goals and Hypothesis

The objective of this MQP was to create a neural network that can be used in a prosthetic limb. The neural network needed to be able to predict intended hand movements when provided with ultrasound images of the muscles associated with said movements. To

achieve the most effective neural network, this MQP compared various classification algorithms to discover which one is the best for this application. The algorithms that were compared are as follows:

- K Nearest neighbor artificial neural network
- Convolutional neural network
- Support vector machine
- Recurrent neural network

A number of decisions had to be made when developing the algorithms including:

- Regression or classification: would the network be able to classify distinct hand movements or would it be able to predict hand (finger) movements on a regression scale?
- Data & Data Pre-processing: what data would be used to train and test these models, how would it need to be processed before feeding into the model?
- Frameworks & Libraries: were there any existing packages that could be used to develop these models and what services do they provide that make them valuable?

These four algorithms were chosen for a variety of reasons. Firstly, nearest neighbor was chosen because of its relatively simple nature and it could provide a good baseline for evaluating other networks. The researcher did not anticipate this algorithm to be the final choice, however it could prove to be the best. Convolutional and support vector machine networks have proven to be efficient when performing image classification. They have also been used in previous ultrasound classification research (hand and abdominal), and for this reason they were implemented and tested [19], [22]. The previous research also helped to evaluate these algorithms as they provide their results which can be compared to this MQP's results. Lastly, a recurrent network was chosen because it has been proven to be most effective for analyzing videos. The researcher chose it because it would help to cluster groups of frames together as one movement that could easily be classified. The four different models were assessed and compared based on prediction accuracy, error from the expected predictions, processing time for training and processing time for a prediction. The researcher expected that a recurrent network would prove to be the most effective for this application because of its

ability to analyze multiple frames at time, however convolutional was also expected to perform well because of its strong ability to analyze single frames. The priority of algorithm development was CNN and SVM → Recurrent → Nearest Neighbor. The reason for this was because CNN and SVM were expected to perform well but would be relatively easier to develop than recurrent. If they were developed first they would help provide a baseline of accuracy and error for recurrent. Nearest neighbor would be developed last because although it may prove to be the easiest, the researcher expected it to perform the worst, and it would benefit considerably from large amounts of training data.

In addition to comparing neural network algorithms, the project investigated smooth finger motion as well as attempting to replicate existing state to state movement. The main difference here is being able to rapidly predict small movements of the finger, rather than predicting an action like open hand to closed hand or vice-versa.

Evaluation

In order to fulfill the client statement the researcher developed a structure to evaluate the final neural network as complete. The algorithm, at the very least, must be able to classify distinct hand movements with an accuracy of 94%. This number was chosen based on previous research into ultrasound image classification, at the minimum this MQP would match previously developed algorithms [19]. Ideally the algorithm would be able to predict movements more efficiently than previous research, however, greater emphasis will be placed on smooth finger movement as an application.

A similarly quantifiable criterion could be applied for said smooth finger movement. When evaluating movements on a regression scale rather than definitive inputs and outputs, classification accuracy does not make sense as a goal. Therefore, the target was to minimize the mean squared error when comparing predictions to actual movements.

$$Mean\ Squared\ Error\ (MSE) = \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{n}$$

The goal of the algorithm was to minimize error to 2%. This number was developed from initial algorithm implementations which produced errors of 6-12%.

In terms of non-numeric qualities the algorithm needed to be able to output information in a way that will be efficient and usable for another team developing the physical prosthetic limb. It is important that the outputs of the neural network could be used in a real application. To achieve such a goal, this MQP will work closely with another MQP working on a prototype in order to produce a usable algorithm.

Results

Development Decisions

The project encountered some blocking factors throughout development and this, along with new knowledge gained, and initial results, created in a slight deviation from the previously defined goals and design process.

The biggest hindrance in developing an effective algorithm was data collection. Throughout the months working on this project, the group was unable to collect more data than was originally available. This was a result of inexperience with the required tools, and coordinating time with many different people involved. Firstly, the researcher used a Verasonics machine to capture ultrasound images. However, this research tool has a large learning curve associated with it. The researcher worked with graduate students who had the most knowledge about the machine, however, were only recently able to use it to a competent level. The other important piece of equipment was the Vicon motion capture machine. This would be used to record what hand movements were being performed and could be associated with the ultrasound images. It was extremely difficult to learn how to synchronize data collection with the Vicon device and the Verasonics device, and unfortunately the researcher learned how to do this too late to collect usable data for their algorithms.

Despite these issues with data collection, the researcher was able to make full and extensive use of existing data, collected through a graduate research project over the summer. The available data was a thirty second segment of a participant opening and closing their hand repeatedly. This resulted in around 3,500 frames of ultrasound images and individual finger movements associated with them. In addition to this dataset, there was another set of images and labels (around a similar number of frames) of the same participant pinching all five fingers together and then relaxing repeatedly. Using both sets of data, the researcher was able to train and test the developed neural network algorithms.

The researcher also choose to make a conscious pivot in developing an effective model for a prosthetic application. The existing data could be used to calculate the angle of four fingers (not including the thumb) at the carpometacarpal joint (as seen below).

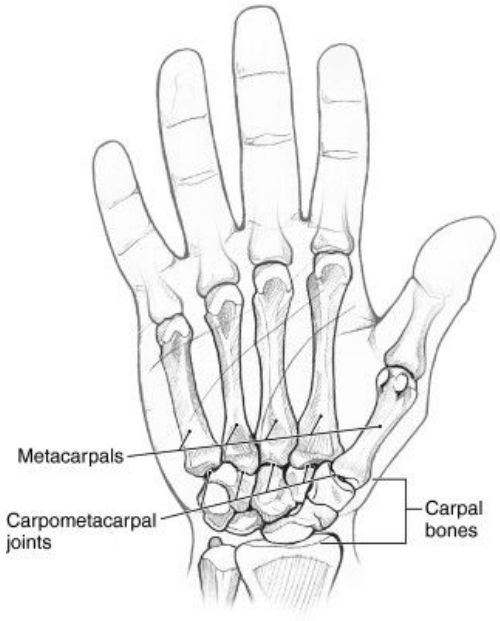


Figure 2: Carpometacarpal Joints [23]

This angle was calculated by using the carpal bones as a base orientation, and finding the angle of the metacarpals in respect to said bones. This data was extremely effective in producing a multi-finger movement algorithm. This new knowledge, along with the results of the recurrent network in predicting state to state movement, encouraged the researcher to focus more on smooth movement of fingers, which is ultimately a less explored field of research and a more applicable approach to creating an effective prosthetic.

The other main decision that needed to be made revolved around packages, frameworks and other tools to use when developing, training, and testing these algorithms. In terms of frameworks and packages, the algorithms were mainly developed using Keras as a “frontend” neural network library and Tensorflow as the accompanying “backend” library. Some of the more simple algorithms were developed using Sklearn packages. The last tool that was used extensively in this MQP was the WPI ACE Linux clusters. Some of the algorithms required intensive resources to train, and that computing power was outsourced to the clusters provided

by the university. Further information on them can be found here: https://arc.wpi.edu/cluster-documentation/build/html/clusters.html

Data Pre-Processing

In order to train and test the developed algorithms, there was a certain amount of pre-processing that had to be performed on the data before it could be used by the neural networks. Firstly, as previously mentioned, the angle of the carpometacarpal joint was used as a label for each image of ultrasound data. All data was split into seventy percent for training a model and thirty percent for testing the model. If the model was trained and tested with the same data then the researcher would not know if predictions were based on what it had learned or was simply matching what the images to what it already knew as fact.The input data were 128x310 grayscale images such as the following:

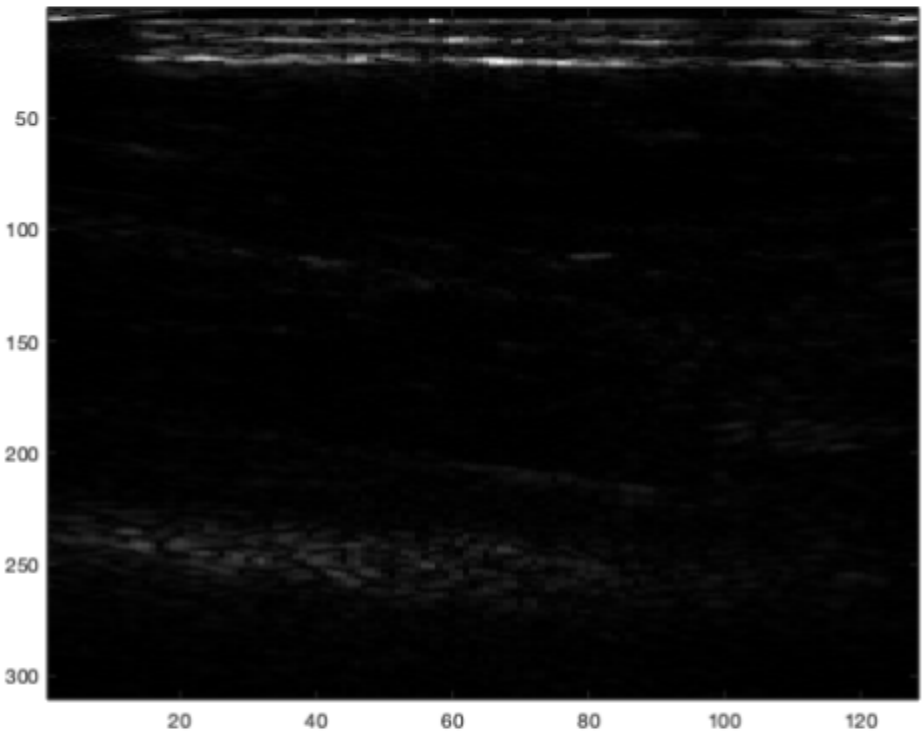


Figure 3: Ultrasound Image of the Forearm

This is just a single frame of the ultrasound data. The muscle can be slightly seen in the bottom left of the image, the algorithm is tasked to notice shifts in the muscle and correlate those

movements with finger movements. The algorithms consume these images as pixels in an array. The value of the pixels are from zero to 255 (black to white). Before inputting to CNN, the array is reshaped to a matrix of 128x310 because of the nature of convolutional layers, whereas KNN and SVM simply consume the image as a flattened array of pixels. This processed data could be fed into the networks to predict the angle of the carpometacarpal joint.

The Recurrent Neural Network needed further data processing because that specific network was being used for state to state motion classification rather than predicting angles. The first step of this process was to analyze the angle data and segment it into specific movements. Below is the training for the angle of the index finger (in radians):

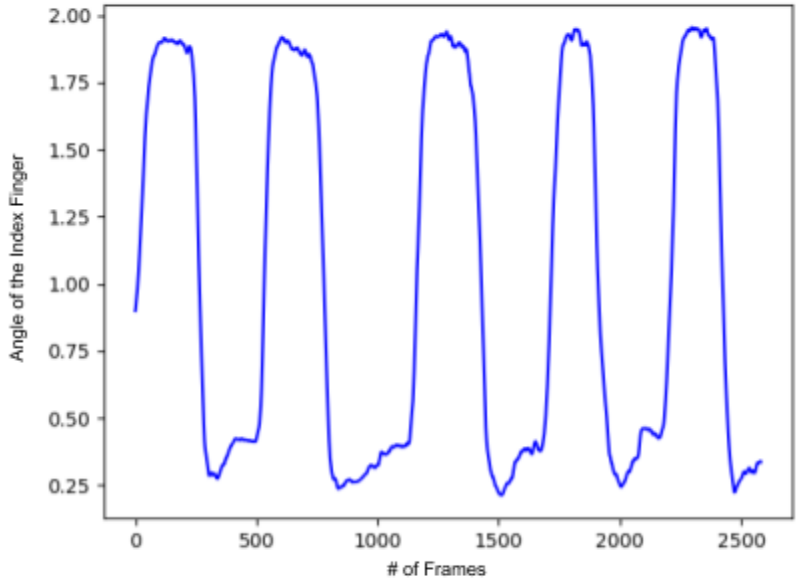


Figure 4: Fist and Relax Training Data

At the maximums the fist is open, at the minimums the fist is closed. When the angle is increasing, the fist is opening and when the angle is decreasing, the fist is closing. The data was segmented so that the frames at each of these four intervals were compiled together and labeled as one of the four states. This resulting dataset could then be fed into the recurrent neural network.

State to State Motion

The Recurrent Neural Network (RNN) that was developed, presented an opportunity to analyze multiple frames of video continuously, as if looking at a video. The idea was for this algorithm to be able to recognize the patterns of muscle movement over multiple frames of data, and then classify those images as a distinct state of hand movement.

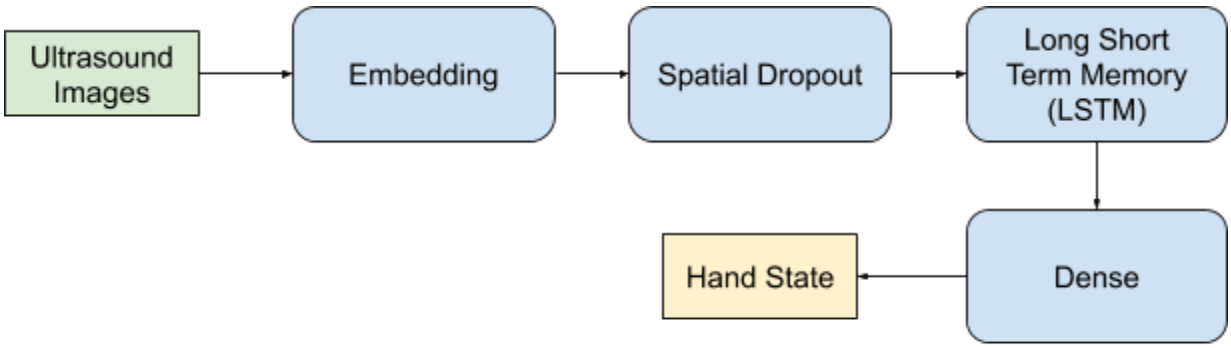


Figure 5: RNN Layers

The RNN was developed with four layers for data to pass through. A series of 128x310 px images first pass through an embedding layer, where it breaks it into 4 possible outputs (opening, closing, closed and open). Next there is a 20% dropout layer to prevent overfitting. The next layer is what makes this algorithm recurrent. The LSTM layer will cycle through the image frames until it can classify them into a prediction. The dense layer separates the output into four possible options (the four states), and lastly returns whichever is the most likely. This is a very simple RNN for classification and the plan was to improve it to make it more accurate depending on the results of the first test.

There were two major problems when testing this neural network. As previously stated, the data was specially processed for this network in order to classify state to state motion. Because of this, there was very little training data to work with to create an accurate model. There were 2,700 frames of training data, however, when processed into segments for the RNN, there ended up being only twenty sets of frames. With this amount of data it was nearly impossible to train a neural network to a usable level of accuracy. Below are the results of

testing the data with the 30% of total fist and relax data (0: opening, 1: closing, 2: open, 3: closed):

Table 1: Actual vs. Predicted of RNN

Actual	3	0	2	1	3	0	2	1	3
Predicted	2	2	2	2	1	2	2	2	2

The network was not even close to a useable level of accuracy. It seems as if the network simply predicted 2 no matter what. These results make sense considering the amount of training data provided, the model simply didn't know enough to make accurate predictions. Unfortunately, the researcher was unable to retrieve more data in order to try to improve the RNN model.

The other major issue with using RNN is how computationally intensive it is to train a model. First of all, when the LSTM layer is recurrently looking through all the frames, the data is stored in memory, so if the images are too large, the computer can run out of memory when analyzing the inputs. For example, in the previous test, the researcher had to downscale the images to take every other pixel, (converting to a 64x155px image) in order to run on a 32GB RAM machine. Secondly, the training period simply takes an extraordinary amount of time to complete. In the previous example, it took fourteen hours to train a model, even with that small amount of data. If a larger amount of data was used to create a more accurate model, it would only increase the time it took to train.

Due to the inaccuracy and vast computational requirements of the RNN, the researcher decided to pursue other algorithms after this first test and development. Additionally, research had already been conducted on state to state motion, smooth finger movement could be implemented by the rest of the algorithms and presented a new and more applicable area of research for this MQP.

Smooth Finger Movement

Support vector machines (SVM), convolutional neural networks (CNN), and K Nearest Neighbors (KNN) were used to create models that could predict the angle of all four fingers

(not including the thumb). SVM and KNN were developed using sklearn (a data analysis library for pythons). The CNN was developed using Keras and Tensorflow.

The researcher chose to develop these algorithms by predicting the angle of the carpometacarpal on the index finger, and comparing the mean squared error of the predictions with respect to the actual angle. The results were as follows:

Table 2: Initial MSE of SVM, KNN, and CNN

Network	SVM	KNN	CNN
Mean Squared Error (MSE %)	12.6	7.6	8.1

Additionally, the predictions were plotted against the actual for each individual network:

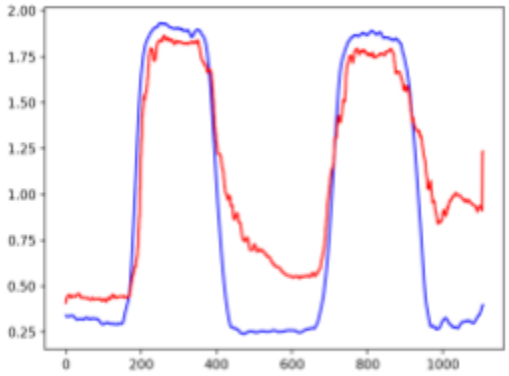


Figure 6: Initial SVM Performance

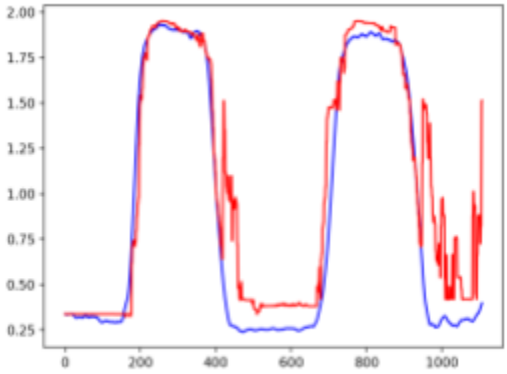


Figure 7: Initial KNN Performance

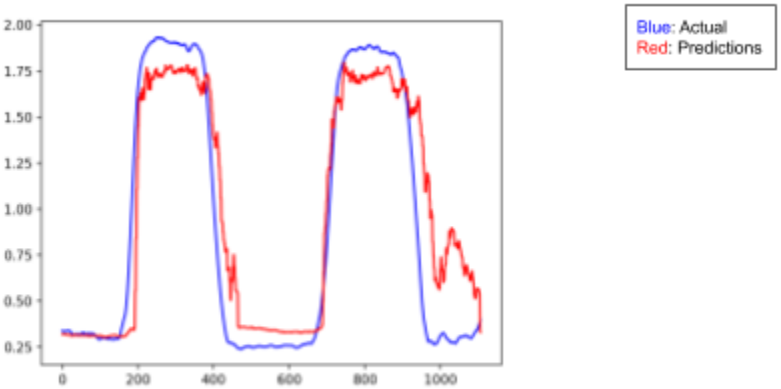


Figure 8: Initial CNN Performance

The error figures clearly show that CNN and KNN performed better than SVM. The plots also help to prove that assumption. Analyzing the mean squared error was important for deciding that SVM was not the best algorithm for this use case, however the raw numbers can not be taken as the only useful metric for decisions. The plots were extremely important for deciding between CNN and KNN.

At first glance it appears as if KNN is the better network. It matches the peaks almost perfectly, whereas CNN doesn't quite reach the maximums. However, when thinking about the network in terms of its application, mapping the maximum exactly is not that important to the prosthetic. As long as the algorithm can identify the finger as being fully extended (something CNN and KNN both do) then the algorithm is accurate. In that respect they are both just as valuable. The worrying thing when looking at the plots is the erratic predictions of KNN. Around 400 frames, there is a spike that cannot be explained. Similarly, in the 900 to 1,100 frames range, there are continually spikes in the predicted values. CNN does not seem to suffer from these occasional, extremely inaccurate predictions, it is always within the realm of the actual angle. To quantitatively analyze this issue, the researcher calculated the standard deviation of the squared error for both the CNN and KNN predictions:

Table 3: Standard Deviation of Error for CNN and KNN

Network	CNN	KNN
Standard Dev of Error	0.1599	0.1958

With this data we can quantifiably say that CNN has less deviation in error than KNN. Therefore the network may prove to be more reliable in a practical application. KNN could be trained to become more accurate and avoid these inaccurate predictions with more data however, the researcher did not have access to such data, and ultimately, CNN would also be improved with more data. Lastly, KNN is far less adjustable than CNN is. For instance, CNN can be modified by adding/removing different layers that parameters pass through when the model is making a prediction. KNN does not benefit from a similar process. For these reasons, the researcher decided to pursue CNN as the network to use for the final model.

Adjusting the Model

The next step of the process was to adjust the initial CNN model for predicting just the index finger, to predicting all four fingers (not including the thumb), and adjusting its parameters and layers to achieve smaller amounts of error. The original CNN was constructed with the following layers:

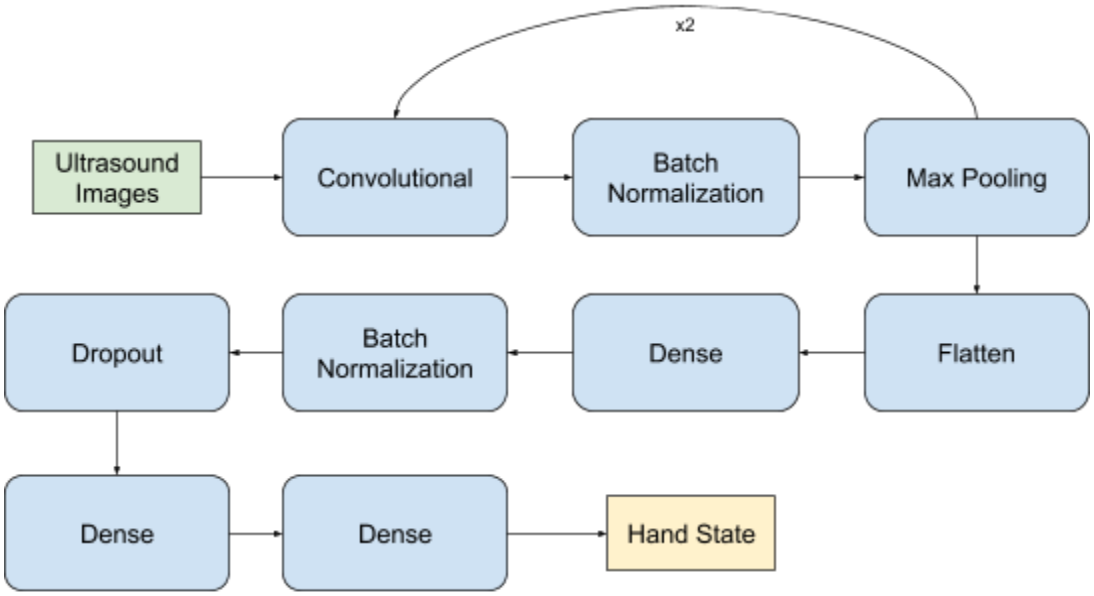


Figure 9: CNN Layers

The CNN was a complicated network with many different layers to perform convolution. Additionally, outputting values of each finger on a regression (the angles) rather than simple classification also resulted in a complicated network and accounts for the dense layers at the end. The first step was for the image to pass through convolutional, batch normalization and max pooling layers repeatedly (three times). The reason there was repetition was to perform convolution on smaller images with different dimensions of the output space.

Table 4: Convolutional Layers of CNN

Layer	Parameters
First Convolutional	`filter=16, kernel_size=(3,3), strides=(1,1), padding="same", activation="relu"`

Second Convolutional	`filter=21, kernel_size=(3,3), strides=(1,1), padding="same", activation="relu"`
Third Convolutional	`filter=64, kernel_size=(3,3), strides=(1,1), padding="same", activation="relu"`
All Batch Normalization	`axis=-1`
All Max Pooling	`pool_size=(2,2)`

The convolutional layers were responsible for the different dimensions of the space (different filter size) and the max pooling layer was responsible for downscaling the images by a factor of two each iteration. Batch normalization helped to "normalize" the output of the convolution using a "relu" activation function. It kept the standard deviation low and the mean around zero.

After the network performed convolution on the image, it went through a flattening layer to prepare the data to pass through dense layers. The first dense layer was added in order to increase the number of units and apply more weighted values to the prediction. A batch normalization (same configuration as the previous ones) and a dropout were added to prevent overfitting with the available data. Lastly, two more dense layers were incorporated, one to account for four fingers, the other to account for predicting on a regression.

Table 5: Dense Layers of CNN

Layer	Parameters
Flatten	`n/a`
First Dense	`units=16, activation="relu"`
Dropout	`rate=0.5`
Second Dense	`units=4, activation="relu"`
Third Dense	`units=4, activation="linear"`

The important distinctions in this section of the network are firstly, the switch from 16 units in a dense layer to four units. This was done to take into account the four outputs (one for each finger), originally there was only one unit when just predicting a single finger. Secondly,

between the second and last dense layer, the activation function changed to linear in order to make a prediction on a regression (angle of the carpometacarpal joint).

This model was initially only tested on the index finger of the fist and relax data set. The next step was to test the model when predicting all four fingers, as well as utilize the other dataset of pinch and relax. The results are as follows:

Table 6: Initial CNN Error on Both Datasets

Finger	Fist and Relax Error (%)	Pinch and Relax Error (&)
Index	6.6	~40
Middle	6.8	~40
Ring	3.4	~50
Pinky	7.2	~50

This data told the researcher that although though the configuration they had started with worked fairly well for the fist and relax data set. It was not even close to as accurate as it needed to be for the pinch and relax data set. The researcher hypothesized that because there was less movement for that motion than fist and relax the muscle movement was subtler in the forearm. To remedy this, parameters were tuned while training the model. Originally, the parameters were: 10 epochs, batch size of 8 and a learning rate of 1e-3. The optimal configuration for pinch and relax was found to be: 40 epochs, batch size of 32 and a learning rate of 1e-3. The increase in epochs and batch size proved to be a better fit and the researcher tested this configuration on both data sets:

Table 7: Modified CNN Error on Both Datasets

Finger	Fist and Relax Error (%)	Pinch and Relax Error (&)
Index	10.3	6.9
Middle	7.8	5.1
Ring	6.9	30.0
Pinky	12	14.5

This was a massively better fit for the pinch and relax data set. However, the error when applied to the fist and relax set actually deteriorated from the original configuration. This led the researcher to test a series of configurations (only on the fist and relax training set to find the optimal one). The following is a summary of those tests:

Table 8: Summary of CNN Errors on Fist and Relax Dataset

Changes	**Index Error (%)**	**Middle Error (%)**	**Ring Error (%)**	**Pinky Error (%)**
Original	6.6	6.8	3.4	7.2
Optimal for Pinch and Relax	10.3	7.8	6.9	12
Convolutional Filters: (16, 32, 64)	4.6	4.3	4.2	8.2
Convolutional Filters: (16, 21)	6.1	5.3	4.2	10.3
Learn Rate: 1e-2, Batch Size: 32, Epochs: 20	4.5	3.2	2.6	6.0
Remove MaxPooling (Only 1 at the end of Conv layers)	4.2	3.5	3.6	6.5
Learn Rate: 1e-2, Batch Size: 32, Epochs: 30	18.4	15.3	14.4	22.1
Learn Rate: 1e-2, Batch Size: 64, Epochs: 20	9.7	7.9	7.3	12.2
Learn Rate: 1e-2, Batch Size: 64, Epochs: 30	19.6	14.9	10.5	16.3

The optimal configuration (highlighted in green) was found to keep the layers the same as they were originally, the difference was increasing the learning rate by a factor of 10, using a batch size of 32 and training for 20 epochs. This yielded results superior to what the model had previously been achieving (highlighted in grey).

Using this configuration, the researcher trained a model using both datasets and tested the networks predictions for each data set. The results were as follows:

Table 9: Final CNN Errors on Both Datasets

Finger	Fist and Relax Error (%)	Pinch and Relax Error (%)
Index	5.7	3.9
Middle	4.5	2.1
Ring	4.4	18.9
Pinky	7.9	8.2

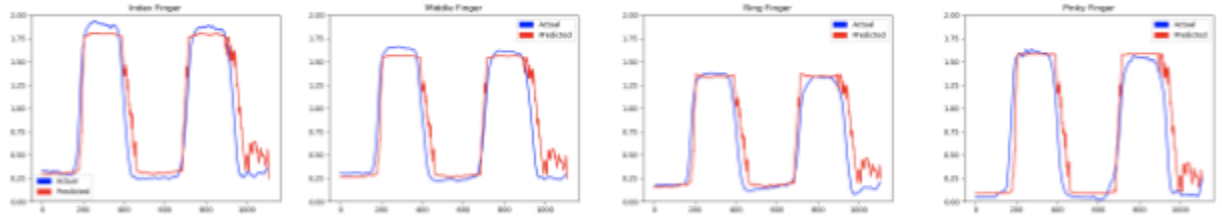


Figure 10: Fist and Relax Final Plots

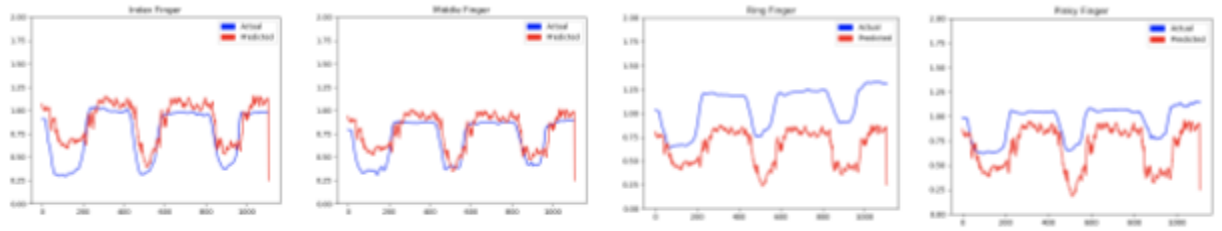


Figure 11: Pinch and Relax Final Plots

To see these images in more detail, see Appendix A. Overall, these results were pretty good. The fist and relax results were nearly as good as they were on a model that was trained with only that data set, and the pinch and relax data was far better than it had ever been before. The only worrying numbers were the ring finger of pinch and relax and pinky finger of both. The researcher believes that this is a result of those the probes placement on the forearm. It was difficult to see the muscle movement with respect for those specific fingers in the ultrasound images, therefore it was harder for the algorithm to make assumptions and predictions about said fingers. This could be remedied with different probe placement/size when collecting data.

The researcher decided that this was the optimal network for predicting finger movement when given forearm ultrasound data. See Appendix C for a plot of all algorithms together. The results would be even better when more data is used to train the model,

something that would need to be done in a practical application. The final CNN python code can be found in Appendix B.

Conclusion

Initial Objectives

This MQP found a suitable algorithm for predicting finger movements when provided with ultrasound images of the forearm. Although the algorithm performed to a satisfactory level in the context of the project, some of the initial objectives were not achieved. For instance, the researcher had initially been hoping to achieve a 94% accuracy when classifying distinct muscle movements. The project quickly took a different course however, classifying state to state motion was left behind to focus on smooth finger movement. This decision was made because of the relatively poor performance of the recurrent neural network as well as the researcher deciding that smooth finger movement was a more impactful aspect of the research in the context of creating a prosthetic arm. Secondly, in regards to smooth finger movement, the goal was to minimize error to 2%. This was an extremely lofty ambition, and the researcher was ultimately unable to achieve that goal. Despite this, the researcher was still pleased with the level of error they did achieve, ~5% for each finger. The researcher is also confident that the algorithm developed could reach that level of error if provided with enough training data. Overall, the main goal of developing an algorithm than can effectively predict hand movements was achieved and could be used in a prosthetic arm.

Future Work

The future steps that should be taken to continue this project are as follows:

1. Collect more data to train the model with;
2. Re-develop the RNN
3. Integrate with a mechanical prosthetic (or simulation);

Collecting more data is the obvious next step for this project. Larger amounts of data would only serve to improve the network and reduce the error of the predictions. Additionally, this MQP only had access to data of the four fingers (not the thumb), so collecting data on all five

fingers would only increase its applicability. Another aspect of collecting data would be to find the optimal position for placement of the ultrasound probe. A second MQP team is currently working on that field of study, so integrating this algorithm with the results of their research would reduce error as well.

Next, with new data collected, a team/researcher should attempt to re-develop the recurrent neural network. A lack of data was the main problem when implementing this algorithm. More data would provide a better conclusion as to whether the algorithm is viable to continue with. Of course, there would have to be considerable data pre-processing done (see "Data Pre-Processing" section above) before the RNN could accept inputs. Additionally, if a team was truly attempting to develop the RNN to achieve a reasonable level of accuracy, an extremely powerful machine would be needed to train such a model. This researcher tested the algorithm on a 4 CPU machine with 32GB RAM and still encountered computational issues; the model simply took too long to train. The RNN presents a fascinating field of research that could prove to be successful in this context if the previously stated objectives were achieved.

The last aspect of future work would be to integrate with a physical prosthetic limb or simulation package to demonstrate the value of the algorithm. The researcher worked on creating web-based api that could consume an ultrasound image and respond with the angle of all four fingers. This can be easily integrated into a demonstration package. The second MQP team is working on the physical prosthetic and will be able to implement this algorithm into their work to demonstrate the applicability of the product. In summary, the researcher is confident that the algorithm developed could be used in practical application, however, training data needs to be collected to improve the model before implementing the network in a prosthetic arm.

Bibliography

[1] Y. P. Zheng, M. M. F. Chan, J. Shi, X. Chen, and Q. H. Huang, "Sonomyography: monitoring morphological changes of forearm muscles in actions with the feasibility for the control of powered prosthesis.," *Med. Eng. Phys.*, vol. 28, no. 5, pp. 405–415, Jun. 2006.

[2] N. Akhlaghi, C. A. Baker, M. Lahlou, H. Zafar, K. G. Murthy, H. S. Rangwala, J. Kosecka, W. M. Joiner, J. J. Pancrazio, and S. Sikdar, "Real-Time Classification of Hand Motions Using Ultrasound Imaging of Forearm Muscles.," *IEEE Trans Biomed Eng*, vol. 63, no. 8, pp. 1687–1698, 2016.

[3] K. J. Zuo and J. L. Olson, "The evolution of functional hand replacement: From iron prostheses to hand transplantation.," *Plast Surg (Oakv)*, vol. 22, no. 1, pp. 44–51, 2014.

[4] I. P. Clements, "How Prosthetics Work," *How Stuff Works*. [Online]. Available: https://science.howstuffworks.com/prosthetic-limb2.htm. [Accessed: 11-Oct-2019].

[5] "Prostheses - Prosthetics: Artificial Limb Information," *Disabled World*, 18-Mar-2018. [Online]. Available: https://www.disabled-world.com/assistivedevices/prostheses/#targetText=There%20are%20%20Four%20Main%20Types%20of%20Artificial%20Limbs&targetText=These%20include%20the%20transtibial%2C%20transfemoral,arm%20missing%20below%20the%20elbow. [Accessed: 11-Oct-2019].

[6] A. Chadwell, L. Kenney, S. Thies, A. Galpin, and J. Head, "The reality of myoelectric prostheses: understanding what makes these devices difficult for some users to control.," *Front. Neurorobotics*, vol. 10, p. 7, Aug. 2016.

[7] "Electromyography (EMG): Purpose, Procedure, and Results," *Healthline*. [Online]. Available: https://www.healthline.com/health/electromyography. [Accessed: 11-Oct-2019].

[8] M. Wehner, "Man to machine, applications in electromyography," in *EMG methods for evaluating muscle and nerve function*, M. Schwartz, Ed. InTech, 2012.

[9] R. M. Bongers, P. J. Kyberd, H. Bouwsema, L. P. J. Kenney, D. H. Plettenburg, and C. K. Van der Sluis, "Bernstein's levels of construction of movements applied to upper limb prosthetics," *JPO Journal of Prosthetics and Orthotics*, vol. 24, no. 2, pp. 67–76, Apr. 2012.

[10] T. A. Kuiken, G. A. Dumanian, R. D. Lipschutz, L. A. Miller, and K. A. Stubblefield, "The use of targeted muscle reinnervation for improved myoelectric prosthesis control in a bilateral shoulder disarticulation amputee.," *Prosthet. Orthot. Int.*, vol. 28, no. 3, pp. 245–253, Dec. 2004.

[11] M. B. I. Raez, M. S. Hussain, and F. Mohd-Yasin, "Techniques of EMG signal analysis:

detection, processing, classification and applications.," *Biol. Proced. Online*, vol. 8, pp. 11–35, Mar. 2006.

[12] C. Freudenrich, "How Ultrasound Works," *HowStuffWorks*. [Online]. Available: https://science.howstuffworks.com/ultrasound.htm. [Accessed: 11-Oct-2019].

[13] N. Hettiarachchi, Z. Ju, and H. Liu, "A new wearable ultrasound muscle activity sensing system for dexterous prosthetic control," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, 2015, pp. 1415–1420.

[14] J.-Y. Guo, "Dynamic monitoring of forearm muscles using one-dimensional sonomyography system," *JRRD*, vol. 45, no. 1, pp. 187–196, Dec. 2008.

[15] "A Beginner's Guide to Neural Networks and Deep Learning," *Skymind*. [Online]. Available: https://skymind.ai/wiki/neural-network. [Accessed: 11-Oct-2019].

[16] J. Brownlee, "Supervised and Unsupervised Machine Learning Algorithms," *Machine Learning Mastery*, 12-Aug-2019. [Online]. Available: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/. [Accessed: 11-Oct-2019].

[17] M. Haldar, "How Do Neural Networks Work?," *Machine Intelligence Report*, 21-Dec-2017. [Online]. Available: https://medium.com/machine-intelligence-report/how-do-neural-networks-work-57d1ab5337ce. [Accessed: 11-Oct-2019].

[18] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way," *Towards Data Science*, 15-Dec-2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed: 11-Oct-2019].

[19] J. Shi, J.-Y. Guo, S.-X. Hu, and Y.-P. Zheng, "Recognition of finger flexion motion from ultrasound image: a feasibility study.," *Ultrasound Med. Biol.*, vol. 38, no. 10, pp. 1695–1704, Oct. 2012.

[20] O. Harrison, "Machine Learning Basics with the K-Nearest Neighbors Algorithm," *Towards Data Science*, 10-Sep-2018. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. [Accessed: 11-Oct-2019].

[21] W. Koehrsen, "Recurrent Neural Networks by Example in Python," *Towards Data Science*, 04-Nov-2018. [Online]. Available: https://towardsdatascience.com/recurrent-neural-networks-by-example-in-python-ffd204f99470. [Accessed: 11-Oct-2019].

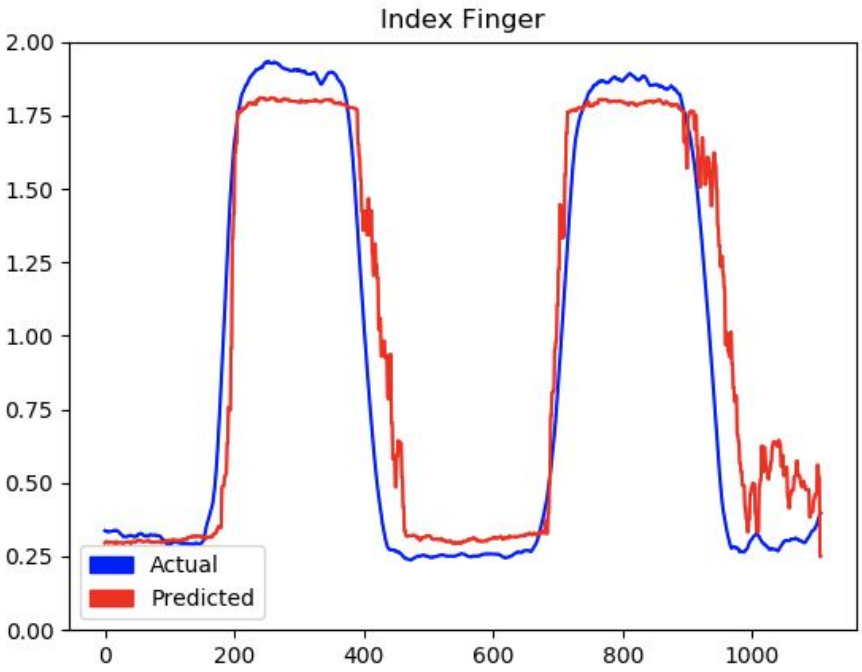
[22] P. M. Cheng and H. S. Malhi, "Transfer Learning with Convolutional Neural Networks for Classification of Abdominal Ultrasound Images.," *J. Digit. Imaging*, vol. 30, no. 2, pp. 234–243, 2017.

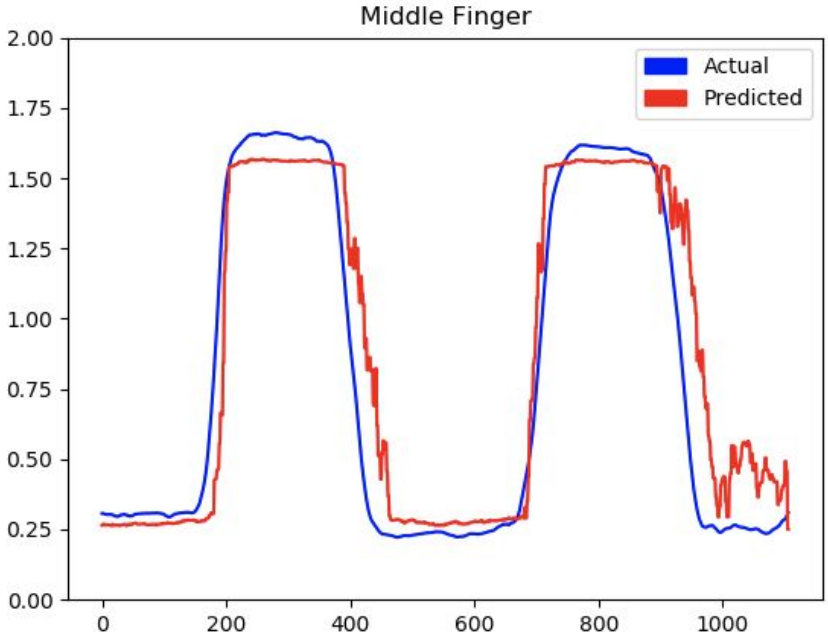
[23] "Carpometacarpal Joint - an overview | ScienceDirect Topics." [Online]. Available: https://www.sciencedirect.com/topics/medicine-and-dentistry/carpometacarpal-joint.

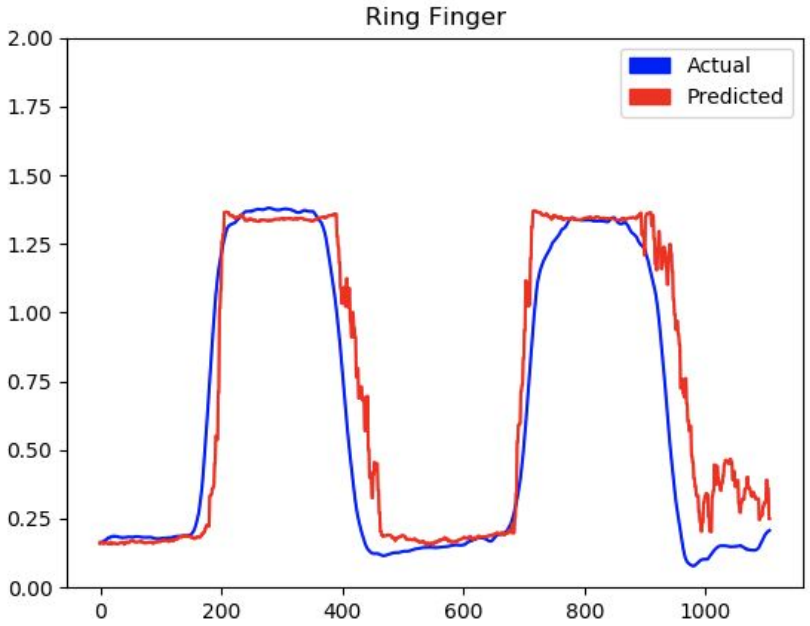
[Accessed: 08-Dec-2019].

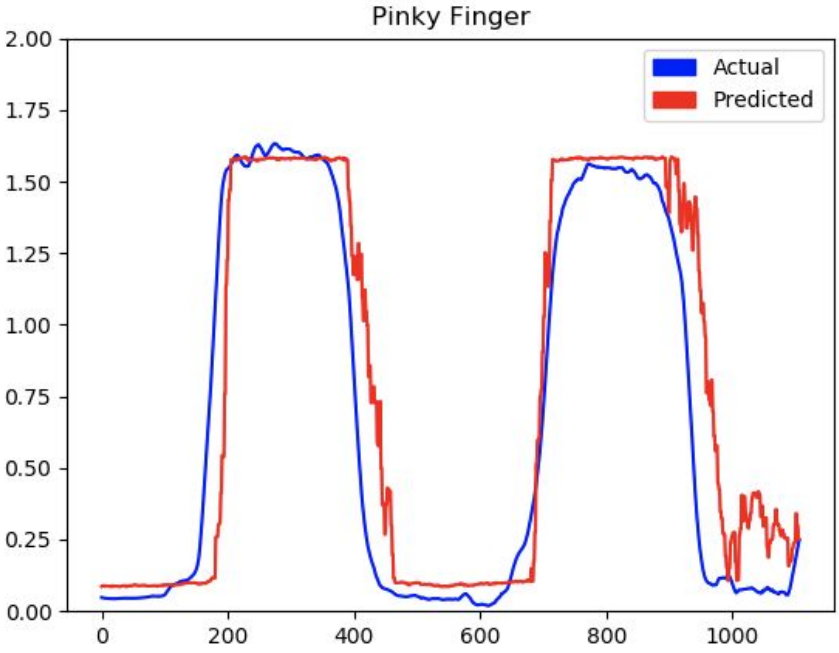
Appendix A: Final Plots

Fist and Relax Plots:

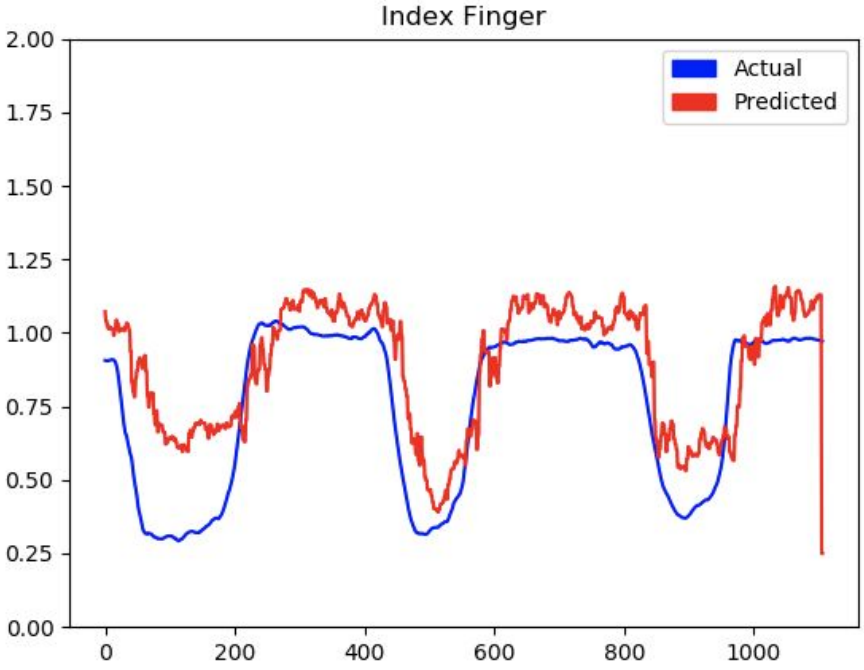


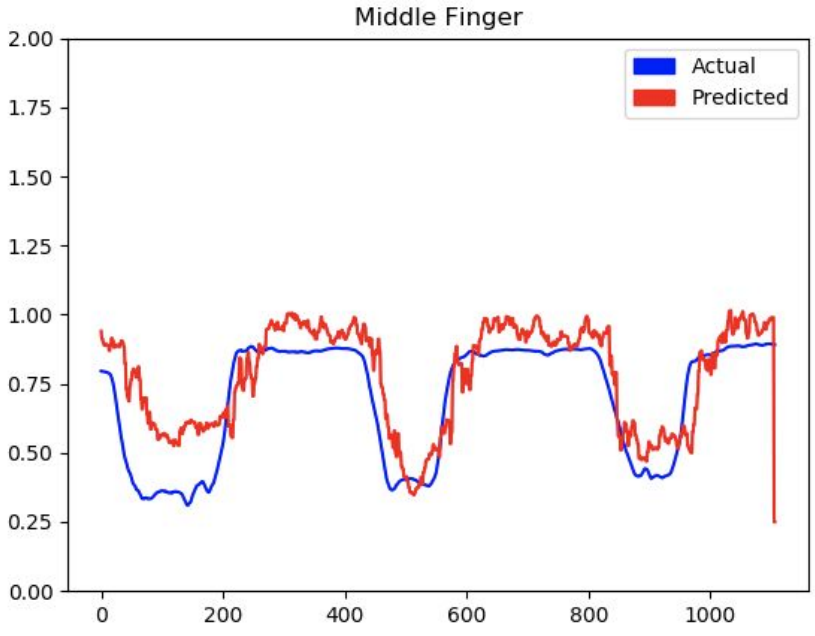


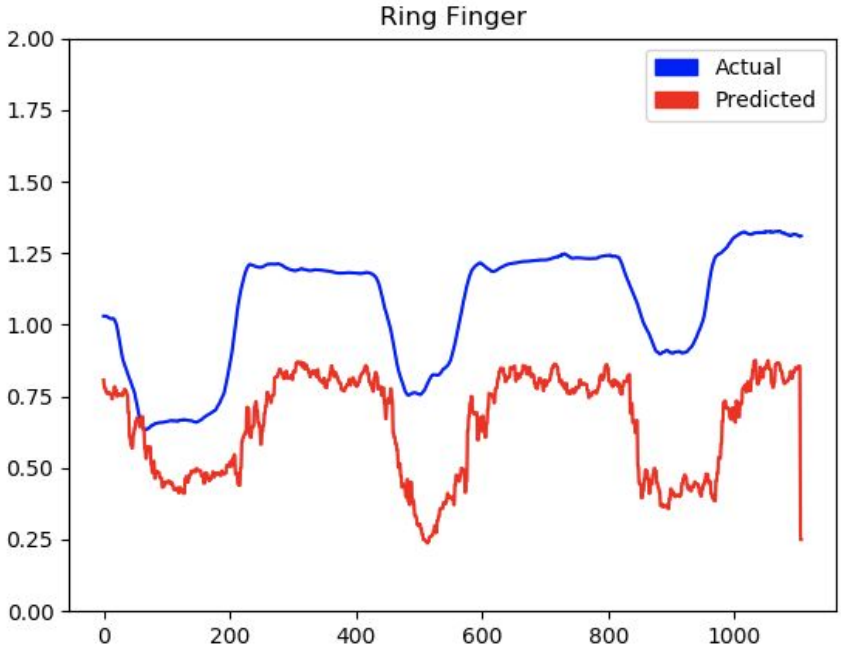

Ring Finger
Actual
Predicted
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000

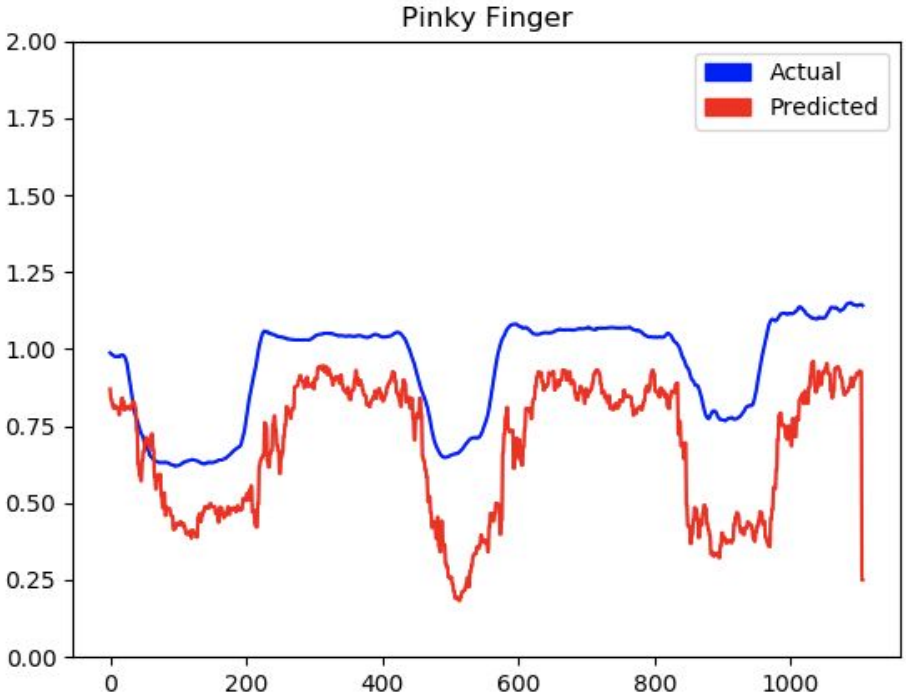

Pinky Finger
Actual
Predicted
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000

Pinch and Relax:






Ring Finger
Actual
Predicted
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000

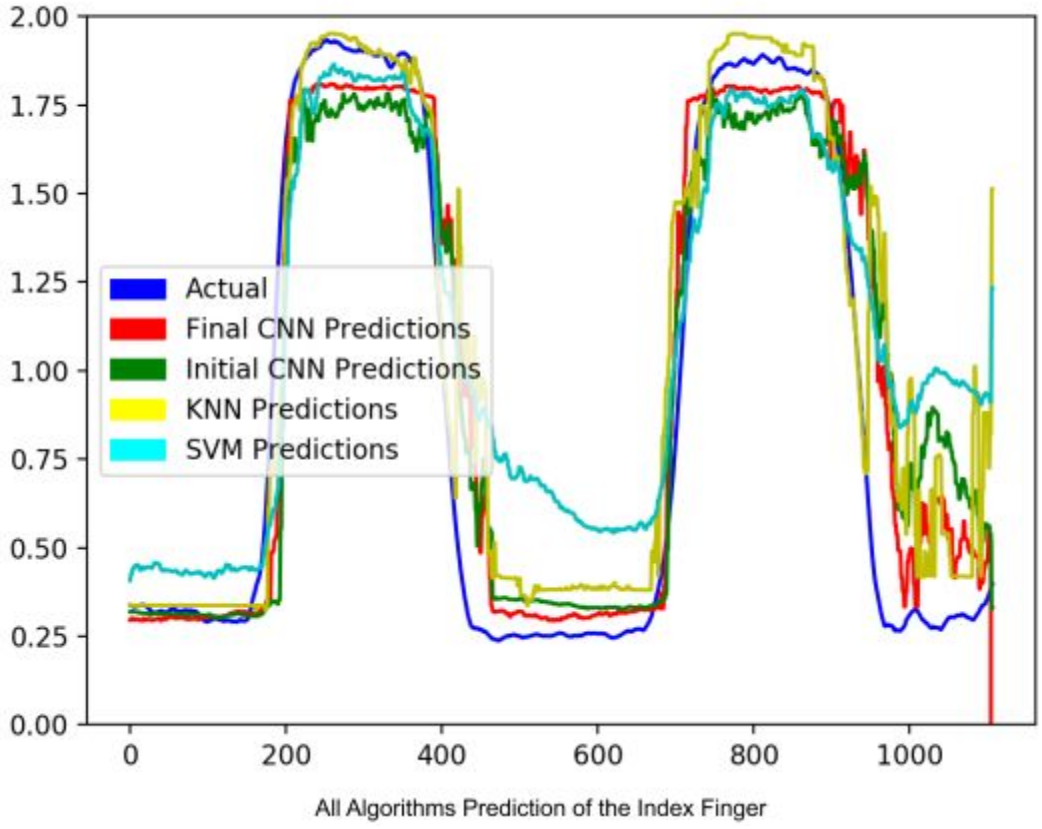

Pinky Finger
Actual
Predicted
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000

Appendix B: Python CNN

```
import numpy as np
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten,
Input, Activation, BatchNormalization, Dropout
from tensorflow.keras.optimizers import Adam


def create_cnn(width, height, depth, filters=(16, 21, 64)):
    input_shape = (height, width, depth)
    chan_dim = -1

    inputs = Input(shape=input_shape)
    X = inputs

    for filter in filters:
        x = Conv2D(filter, (3, 3), padding="same")(x)
        x = Activation("relu")(x)
        x = BatchNormalization(axis=chan_dim)(x)
        x = MaxPooling2D(pool_size=(2, 2))(x)

    x = Flatten()(x)
    x = Dense(16)(x)
    x = Activation("relu")(x)
    x = BatchNormalization(axis=chan_dim)(x)
    x = Dropout(0.5)(x)

    x = Dense(4)(x)
    x = Activation("relu")(x)

    x = Dense(4, activation="linear")(x)

    model = Model(inputs, x)
    return model

### LOAD DATA HERE ###
### reshape x_train to [-1, 4], reshape y_train to [-1, 310, 128, 1] ###

print('creating model...')
```

```
model = create_cnn(128, 310, 1)

opt = Adam(lr=1e-2, decay=1e-3/200)
model.compile(loss="mean_absolute_percentage_error", optimizer=opt)

print('training model...')
model.fit(x_train, y_train, validation_split=0.3, epochs=20,
batch_size=32, verbose=2)
```

Appendix C


2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
200
400
600
800
1000
Actual
Final CNN Predictions
Initial CNN Predictions
KNN Predictions
SVM Predictions
All Algorithms Prediction of the Index Finger