Classifying Hand Gestures from EMG Data with Multi-layer Perceptron Networks

CP3403 - Data Mining Project

Joshua Gray - 13177877
Harmon Singh - 13182411

Link to Dataset: `https://archive.ics.uci.edu/ml/datasets/EMG+data+for+gestures`
Link to Presentation: `https://youtu.be/_YLLmnAcJ3U`

College of Business, Law and Governance
James Cook University
Australia
02/06/2019

Contents

List of Figures

List of Tables

1 Introduction

Since the turn of the 20th century, a lot of focus has been placed on the incorporation and integration of technology in medical fields. This relationship between medicine and technology has spawned a multitude of life-saving devices such as the Automated External Defibrillator and non-invasive diagnostic tools such as Magnetic Resonance Imaging and Computed Tomography.

With the widespread increase of data mining and analysis in recent years as well as vast improvements in computing and storage power available today, a logical question for the data science community is finding the ways in which this area has the potential of improving the medical industry. In addition to this, in what way can the Internet of Things enable this kind of technology? A field which has attempted to improve the quality of life for it's patients since the beginning is the field of prosthetics.

Prosthetics offer a plethora of opportunities for data to be examined and mined. This potential use has sparked a thoughts and resulted in many studies and articles. This study will be centralised around the classification of raw Electromyography data captured from a EMG Band which is commercially available for at home gesture analysis tool. A myograph is any device used to measure the force produced by a muscle when under contraction. When performing an action, muscles are contracted with different speeds and intensities. An EMG can measure the activation of skeletal muscles by the electrical potential seen at the measurement probe. These voltages can be studied to measure muscle contraction and hence gestures that the subject is performing. This study will attempt to classify EMG data into seven classification categories based on distinct hand gestures recorded by an EMG device.

2 Artificial Neural Networks

The field of classification has been plagued in recent years by the popularity of the Artificial Neural Network (ANN). Constantly evolving, ANN's are machine learning systems that aim to simulate the functionality of biological networks such as the human brain. As they have a constant defined structure, they excel when compared to some other methods as they have no bias from the programmer i.e. they are defined generally, with the data forcing biases in the output of the system.

The basic functionality of ANN's remain the same between different implementations. The building block of all neural networks is the implementation of the "neuron". A neuron is modelled as a weighted function of its inputs. As given by Figure 1, the inputs x_i are weighted by a factor w_i with an added bias b. All of these inputs are then summed and passed through an activation function f_a which filters the output y.

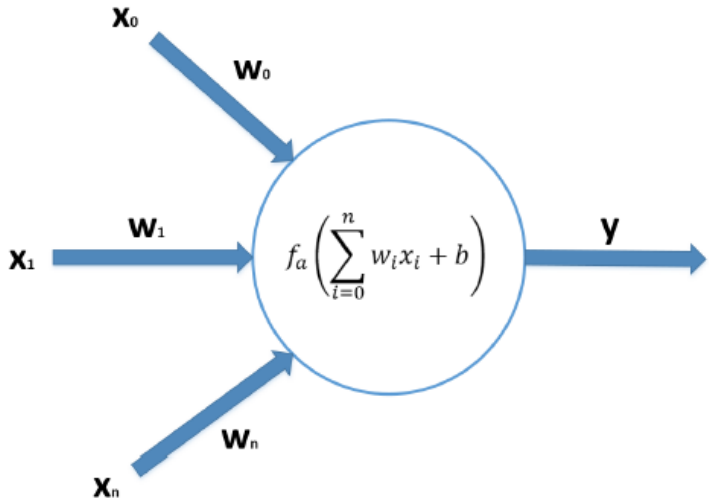


Figure 1: Simple ANN Neuron
[1]

2.1 Multi-layer Perceptron

With this simple neuron functionality, the implementation of the Multi-layer Perceptron (MLP) system can be realises. An MLP system consists of fully connected "layers" of neurons. At their most simple level, they can be realised with a minimum of three layers. The first layer, the input layer is the place in which raw information enters the network. This layer is un-weighted and each neuron fully connects to the next layer, the "hidden" layer. The hidden layer can be a single layer and up to as many layers as the developer chooses. These layers are fully connected i.e. each output connects to every neuron in the next layer. The final layer is called the output layer. This layer is the direct point in which interfacing with the neural network is performed. Depending on the task required, the function of the output layer may vary. The implementation of MLP for this project is explored in Section 2.2.

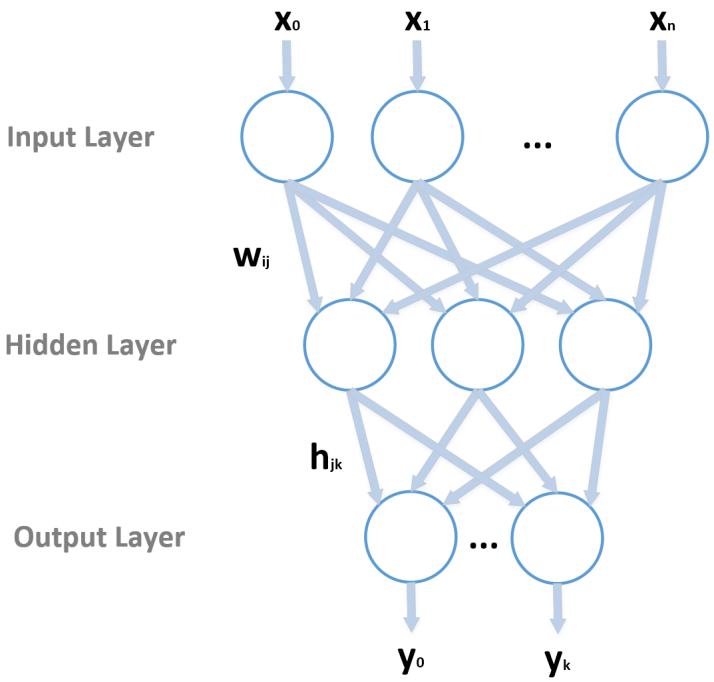


Figure 2: Simple MLP Network
[1]

2.2 MLP's and Classification

As the field of MLP's and ANN's in general are vast in complexities and can differ greatly in their implementations, this section aims to define the general structure of the MLP system used in this paper with some generalisations in terms of hyper-parameter settings as this is the core part of manipulating our system.

The first major consideration required when designing an MLP system is the choice of activation function f_a (See Figure 2). For a long time, the sigmoid or hyperbolic tangent function reigned supreme in this area. However, in recent years, the Rectified Linear Unit (ReLU) has proven to be a better choice in many applications. As the name implies, this function is a rectified linear activation function meaning values below 0 are 0 and the function is linear when values are greater than 0. This can be seen in Figure 3. As back-propagation in neural networks relies on the derivative of the activation function, the ReLU gradient function is a step function about the origin. This normalises weight updates of the loss function and avoids the vanishing gradient problem present with sigmoid and hyperbolic tangent functions.

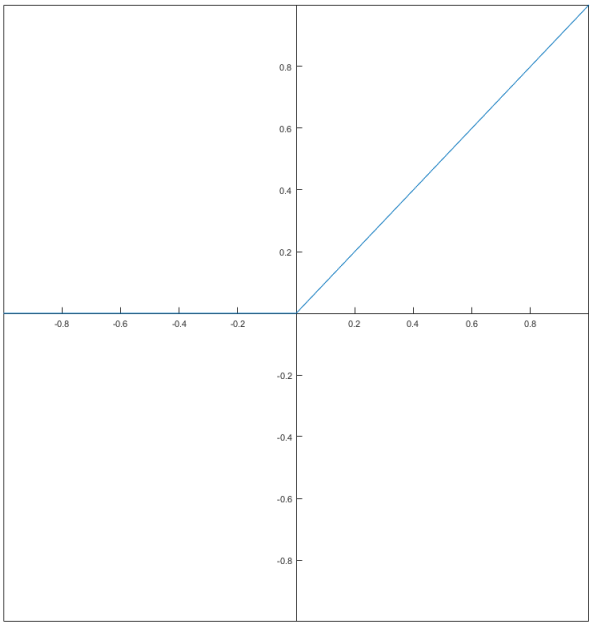


Figure 3: ReLU Activation Curve

When considering activation functions for neurons, you must also consider the output of your neural network system. Depending on the problem you are attempting to solve, this may come in many different forms. For classification, the output of the network should be the probability of the input matching one of the classification classes. The most appropriate activation function to generate this is the softmax. Softmax as a function that accepts K real input arguments and outputs a probability distribution consisting of K probabilities. This essentially enables the output of the neural network to convert the non-normalised output to a probability distribution in which the sum of all class probabilities is 1. The softmax function is given by:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

where z is the input vector

Now that we have a neural network consisting of an arbitrary amount of hidden layers with ReLU activation and softmax output, an appropriate loss function needs to be chosen. For this purpose, the popular RMSProp algorithm was selected. Although never published, the RMSProp loss function designed by famous computer scientist Geoffrey Hinton, is a momentum-based loss function designed to handle some local minima issues present in previous loss functions such as stochastic gradient descent. The RMSProp function was utilised as the loss function for all the models presented in this paper.

3 EMG Data for Gestures Data Set

The EMG Data for Gestures Data Set (Available from: `https://archive.ics.uci.edu/ml/datasets/EMG+data+for+gestures`) was a data set created for determining latent factors in the performance of

sEMG interfaces [2]. The overall achievement of this paper was identifying some potential medical influences on the use of EMG interfaces to measure gesture information.

This particular dataset was obtained through the use of the Myo Thalmic bracelet (See Figure 4). The EMG channel recordings were obtained through a Bluetooth interface to a computer. A total of eight channels are recorded with each of the sensors equally spaced around the forearm of the subjects recorded.

Table 1: Attributes of the Raw Data Set

Attribute	Description
Time	Time Step of Recording (Numeric)
Channel 1-8	Raw EMG Measurement (Double Precision Float)
Class	Gesture Performed (Numeric 0-7)

Table 2: Descriptions of Attribute Class

Class	Description
0	Unassigned
1	Hand at rest
2	Hand clenched in a fist
3	Wrist Flexion
4	Wrist Extension
5	Radial Deviations
6	Ulnar Deviations
7	Extended Palm

A total of 36 subjects were used for this experiment. Each subject performed a series of static hand gestures (a total of 7 unique gestures) with a 3 second gap between the recording of each gesture. The series were also repeated twice for each subject.

The values recorded by each channel are represented in scientific format corresponding to a double precision floating point number. The magnitude corresponds to the potential measured by the device at the skin level but units were withheld from the data. Further information about the Myo armband revealed that raw EMG data should be transferred as uint8 values corresponding to the amount of "activation" of the muscle [3] but this is not reflected in the data. Based on preliminary analysis, the data should be sufficient as the final system should adapt to incoming signals without the need of specific input data (As long as the format is known).

An important thing to note for this data set is that due to the nature of the data (different subjects), the general classification task will prove more difficult as there are physiological aspects that affect the magnitude of the readings. As explored in [2], people who have greater muscle development in their forearms generally show higher magnitudes on the EMG readings and people with higher body fat content generally have smaller magnitudes.

Figure 4: Thalmic Labs Myo Bracelet

Due to the discontinuation of the Myo armband in 2018, this dataset is unlikely to be able to be replicated or in future. However, the design of the final analysis system should be independent of the recording device used (as preprocessing should solve this problem). As is the design of many systems in use today, modularity is key for these devices to remain cost effective for the end users.

4 Preprocessing

Preprocessing is a crucial step of any data analysis task. Much like any other analysis, the quality of the outputs of the system is directly dependent on the quality and accuracy of the inputs. In addition to this, deeper knowledge of the system is also useful in mapping input data values into system values.

For this system, the following is known about the nature of the data:

- **Values -** The values recorded in the data can generally be represented in scientific notation with up to 5 decimal places.
- **Data Nature -** According to the data sets provided information[2], the data recorded are raw EMG values measured by the Myo armband. As such, special treatment of these values is required to generate reasonable predictions.
- **Data Bias -** As 36 subjects were used to conduct this experiment and the overall goal of this system is to generalise a system for classification, special consideration was required to obtain an accurate representation of the whole data set.

With these goals in mind, a system was designed to both format and clean data as well as prepare it for an MLP system. To design the system the most efficiently and allowing the ability for further experimentation, the system was to be designed in two steps. The first step was the formatting and cleaning of the data files

into a format that could be directly imported into a Relational Database Management System (RDBMS). The second section was then the direct mapping of values into a format conducive to making classifications by the MLP system.

4.1 Data Format Tool

The first section of the overall preprocessing system was to develop a system to interface between raw data values and the RDBMS system storing all the training data for this project. For the sake of reproducibility, this section was designed to interface directly with the EMG Data for Gestures Data Set. Although a reasonably trivial process, it was important to structure the output data in such a way that all systems are able to interpret. The following flow chart describes the functionality of this section.

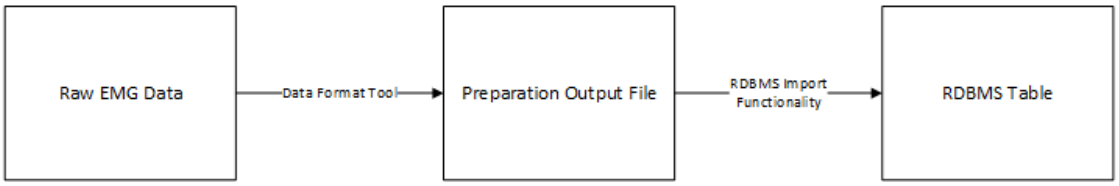
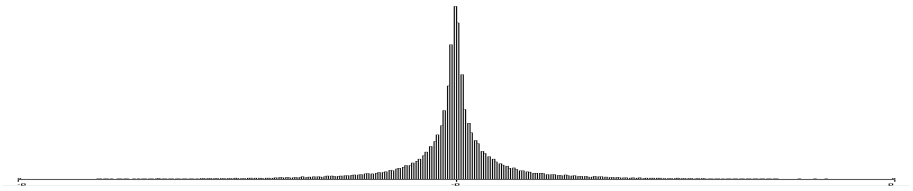
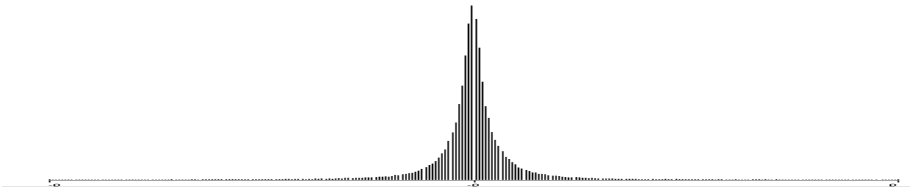
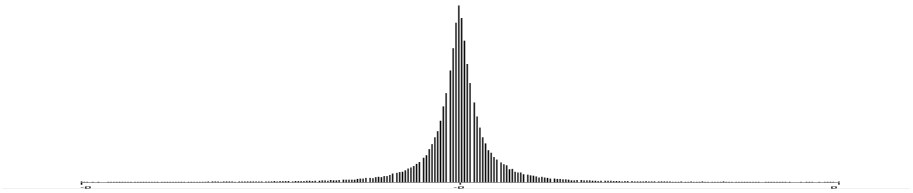
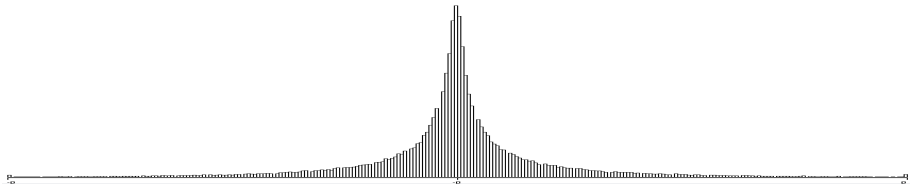
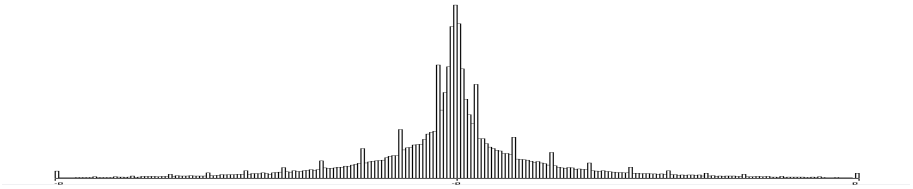
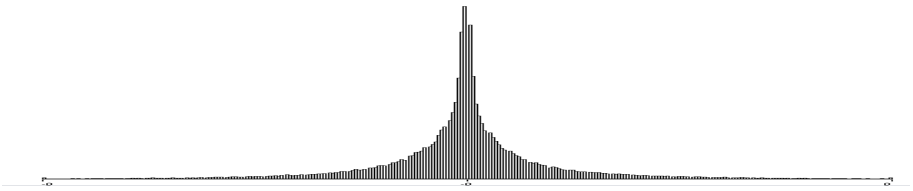
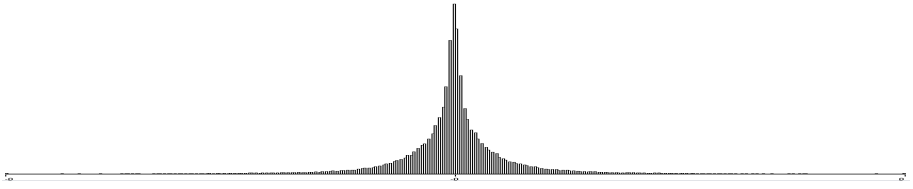
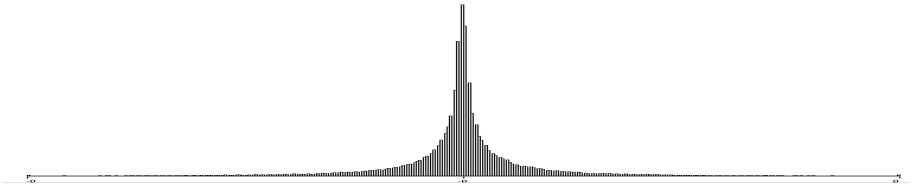


Figure 5: Data Flow of the Formatting System

Table 3 show the distribution of values across each channel. This table was created using the popular data-mining tool WEKA. Due to the size of the input values and the implementation of the WEKA interface, little data insights are directly available. However, the distribution of values for the raw data indicate that there is a fair chance that the data is relatively clean with small quantities of outliers and a very defined bell curve shape about the resting value (channel reading 0).

Table 3: Histogram of values across the EMG Data for Gestures Data Set

Channel	Histogram
1	
2	
3	
4	
5	
6	
7	
8	

The original data file structure consists of multiple folders relating to each subject with multiple data files for each test performed. The data format module was designed such that the entire file-system will be traversed and each file available on the system is read and converted into the desired interim format. The full exploration of the data format module can be seen in Appendix A. Beyond this, the data format tool has the following features

- Unmarked data is completely removed
- Incomplete data is ignored
- Records are randomised at the conclusion of the import (Generalising the data)
- Records are outputted to a single csv file that is directly importable into SQL Server

4.2 Data Storage

Data storage for this task is an important one, With hundreds of thousands of records available, it is important to develop a system for access and storage of this information. Being a staple of the business world for the last few decades, SQL is an appropriate choice for this task. For this project, Microsoft SQL Server Professional 2017 was chosen as the RDBMS for the gestures data. The output of the Data format tool explained in Section 4.1 can be directly imported into SQL Server through the use of Microsoft's SQL Server Import and Export Wizard.

Table 4: Gestures Table Structure in SQL Server

Field	Data Type	Data Length	Precision	Primary Key
record	int	4	10	1
channel1	float	8	53	0
channel2	float	8	53	0
channel3	float	8	53	0
channel4	float	8	53	0
channel5	float	8	53	0
channel6	float	8	53	0
channel7	float	8	53	0
channel8	float	8	53	0
class	int	4	10	0

The last aspect of the data storage system is accessing the data within the MLP software. This is done using an ODBC driver written in python (pyodbc). Being a generic driver for database connectivity, this interface should be accessible for different RDBMS software (Oracle or MySQL) as well as different programming languages as the ODBC driver is implemented in all major programming languages.

4.3 Data Processing

Much unlike many problems in Data Mining, the preprocessing for this system lies primarily in the ability to use insights of the raw data in order to process them for a neural network. As it is inherently a signal processing problem[4], the following is assumed of the Gestures data based on the range of values:

- DC offset is removed from the signal

- The signal has been applied with low-pass filtering before recording
- Presence of negative values eludes to rectification and time-averaging not being performed on the data. This may be a serious issue as theoretically, the values recorded might not be time-representative of the gesture.

Based on this, the first step of data processing is to "rectify" the incoming values. As the EMG data is an electrical signal based on activity produced by skeletal muscles, the value of such signal are sinusoidal in nature. Given this, the sign of record values is irrelevant to the classification task and in order to minimise confusion of the MLP system, were removed.

The next step is to prepare the values to be used with the chosen MLP system. In a neural network system, there are multiple variables that come into play when preprocessing data. With the increase in computation time, it is important that only necessary functions are applied for real-time systems, as well as designing input functions to fully utilise the activation functions of MLP neurons and minimise errors due to floating point quantisation error. For this aspect, MinMax Scaling was used to ensure that values were within the gradient points of the activation function (ReLU) as well as achieving a zero mean and unit variance.

5 Result and Discussion

The aim of the study was to train the algorithm to categorise eight channels of raw EMG data into one of seven classes. Looking at the accuracy of four models tested we can observe high levels of loss in Figure 6.

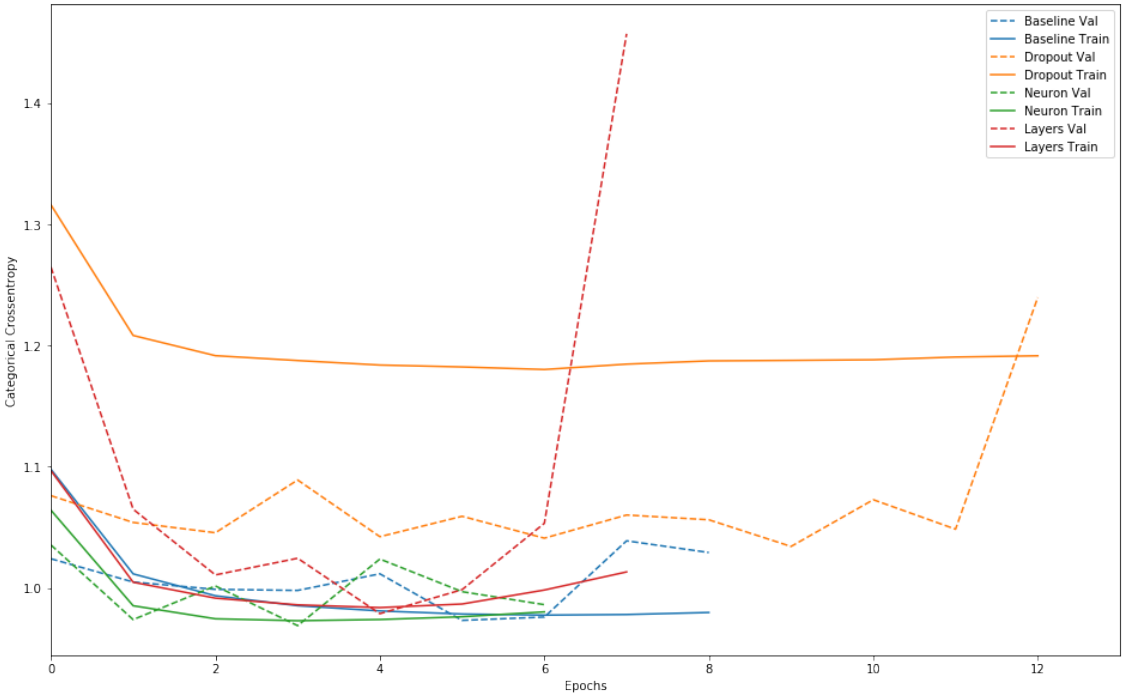


Figure 6: Model Training Data - Model Loss per Epoch

Upon further investigation it can be seen that the lowest loss values can be generated after minimal epochs which shows how generalisation of data is not a viable solution for classifying EMG data. This method of batching the data was used for all four models. A better solution in future would be to preprocess data per

subject rather than as a bulk procedure.

For extended insights into the models, metrics were generated in the form of classification reports. These reports, generated for each of the main models, show some valuable metrics for classification tasks.

Precision is a measure of how exact the classifier is with respect to each class. A low precision indicates a high number of false positives. On the other hand recall is a measure of sensitivity and shows how often a classifier regularly misclassified records of each class. F1-Score is defined as the balance between precision and recall. This method can be used to determine which model is superior.Support can be seen as the number of instances each class has a record in the trained model.The tables below are generated using a random selection data. From the SQL database.

Table 5: Classification Metrics for the Baseline Prediction

Class	Precision	Recall	F1-Score	Support
1	0.67	0.99	0.8	146
2	0.68	0.57	0.62	160
3	0.62	0.61	0.61	175
4	0.67	0.63	0.65	191
5	0.58	0.54	0.56	166
6	0.57	0.54	0.56	179
7	0	0	0	8
Avg/Total	0.63	0.63	0.62	1025

Table 6: Classification Metrics for the Dropout Model Prediction

Class	Precision	Recall	F1-Score	Support
1	0.92	0.84	0.88	146
2	0.35	0.76	0.48	160
3	0.53	0.63	0.58	175
4	0.64	0.52	0.58	191
5	0.6	0.29	0.39	166
6	0.63	0.37	0.46	179
7	0	0	0	8
Avg/Total	0.6	0.56	0.55	1025

Table 7: Classification Metrics for the Extra Neurons Prediction

Class	Precision	Recall	F1-Score	Support
1	0.83	0.95	0.88	146
2	0.65	0.6	0.62	160
3	0.71	0.49	0.58	175
4	0.61	0.67	0.64	191
5	0.46	0.63	0.53	166
6	0.59	0.51	0.55	179
7	0	0	0	8
Avg/Total	0.63	0.63	0.62	1025

Table 8: Classification Metrics for the Extra Layers Prediction

Class	Precision	Recall	F1-Score	Support
1	0.95	0.14	0.24	146
2	0.39	0.73	0.51	160
3	0.71	0.32	0.44	175
4	0.53	0.65	0.58	191
5	0.32	0.61	0.42	166
6	0.64	0.25	0.35	179
7	0	0	0	8
Avg/Total	0.58	0.45	0.43	1025

5.1 Artificial Neural Networks

The four models tested displayed four different methods tried to improve output accuracy. The baseline model consisted of two layers of 16 nodes and this proved to produce the best results with an exponential like approach to our minimum loss value. This can be compared to both models where the neurons were doubled and layers were doubled in the neuron and layers curves respectively. The drop out model tested showed much higher values of loss, this could be due to the number of epochs tested as a result of the early stop function implemented.

On further analysis of the curves, observation shows how the validation curves for each model rubber band around its minimum loss and this could be a multitude of reasons, however this is usually correlated with over training. From another view this could be related to how the data was randomised. With different peoples data being fed into the model this could result in a learning curve for each epoch.

5.2 Performance

One caveat realised towards the end of the data preprocessing was the use of MinMax scaling at the batch level potentially skewing results obtained. A more appropriate solution is the use of MinMax at the subject level such that the maximum of a record is relative to the maximum value recorded by that subject. With this, the generalisation process may have been considerably more accurate.

Another realisation was the way in which the model was trained and data was split. With persons data being split via class and randomised this would prove for the model training to be very generalised skewing result back and fourth amongst an average accuracy.

The data found for this study was conducted in Russia with no data on the ages and types of people used for gathering results. Although the algorithm would be aiming for a generalisation. It is inevitable that each person would develop and control their muscles differently. This would differ from male to female, a person with injuries or nerve damage or different nerve endings and their ability to flex their muscles to perform the gesture classes classified.

6 Conclusion

In summary the realisations of this study can show many different perspectives; the most notable is data accuracy. Whether the data is inaccurate because of the use of surface EMGs or human error in placement, it is safe to say that there are errors in the processing before being released. This would have to be examined

further by the authors or in future works. All in all the results correlated well with initial hypothesis of the data being too generalised. This provided many insights in the world of data mining and analysis.

7 Future Work

Looking further into improving the performance of the model designed there a few methods devised for future works. With data not in being able to be changed the recommendations are primarily focused on algorithm selection, tuning and use of ensembles.

References

[1] G. Bonaccorso, *Machine Learning Algorithms: A reference guide to popular algorithms for data science and machine learning*. Packt Publishing, 2017.

[2] S. Lobov, N. Krilova, I. Kastalskiy, V. Kazantsev, and V. Makarov, "Latent factors limiting the performance of sEMG-interfaces," *Sensors*, vol. 18, no. 4, p. 1122, apr 2018.

[3] Thalmic. (2019, Jun.) How do i access the raw emg data from the myo armband? Online. [Online]. Available: https://support.getmyo.com/hc/en-us/articles/202536726

[4] W. Rose, "Electromyogram analysis," *Mathematics and Signal Processing for Biomechanics*, 2014.

[5] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[6] V. Sharma, S. Rai, and A. Dev, "A comprehensive study of artificial neural networks," *International Journal of Advanced research in computer science and software engineering*, vol. 2, no. 10, 2012.

[7] R. Bonnin, *Machine Learning for Developers: Uplift your regular applications with the power of statistics, analytics, and machine learning*. Packt Publishing - ebooks Account, 2017.

[8] J. Brownlee. (2018, Sep.) Machine learning algorithms in python. Machine Learning Mastery. [Online]. Available: https://machinelearningmastery.com/

[9] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

Appendices

A Data Format Module

PreProcessing

May 31, 2019

1 Proof of Concept - Pre-Processing

This brief report summarises the method used to preprocess the raw gestures data into manageable, database ready data. It is important that the data is free from errors at this stage as well as have values in a form that is also manageable for direct import into the database management system (MSSQL in this case).

1.1 Imports

The python libraries used for this pre-processing are well established in the data science field. We are creating and managing data with the use of the pandas DataFrame functionality and are also using the os library built-in to python to manage access to the local filesystem.

```
In [1]: import pandas as pd
        from os import listdir
        from os.path import isfile, join, isdir
```

1.2 Process

The first step in developing the pre-processing system is to manage access to the raw data. For the Gestures data, each candidate's records were placed in a folder associated with a number. The files themselves are timestamped and associated with their parent folder. In order to obtain a database of individual records to access, a small helper function is required to traverse and collate each file.

The code below establishes the location of the raw data and the output path in respect to the execution directory of the python code. The loop below simply prints, the file directory for these files which aligns to the filestructure of the raw data as explained above.

```
In [2]: path_to_data = "./Raw Data/gestures/"
        output_path = "./Raw Data/Formatted_Files/"

        for dir in listdir(path_to_data):
            print(dir, end=',')
```

```
01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,
```

The following code now produces a list of directories for the system to traverse to retreive data from. The difference between the direct listdir above and this code is that only subdirectories of the parent directory are listed. You can see this as the raw data's README.txt is not listed.

This step was added as a simple way of ensuring that the raw data can be directly obtained and pre-processed before use without any need to manually modify any information

```
In [3]: folders = [dir for dir in listdir(path_to_data) if isdir(join(path_to_data, dir))]

        for dir in folders:
            print(dir, end=',')
```

```
01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,
```

Below is the code that was used to prepare data for importing to the MSSQL database. The only difference being that the file was exported as a single csv file before importing. The basic process of this function is to obtain the list of files present in each folder as defined above. Once obtained, the system then reads the information directly as a tab seperated csv file (As the raw data is expressed). Also note that the usecols function strips the time information from the data. As this final system was a classification task, the requirement of time space information was deemed not necessary and has hence been excluded from the system.

Once the system has completed compiling all of the data into a single pandas DataFrame, some processing was performed on the data. Firstly, this involved removing the subset of the data with the class value equal to 0 as this corresponds to unlabeled data (as stated in the README). As these classes are not useful for training/validation of the chosen neural network, they are cleanly removed from every file. After this set of processing, the result was then shuffled to ensure that a random distribution of each subjects data was obtained. For the final neural network, this ensures that specific epochs are not biased towards the gestures of a single individual, rather a random subset of gestures from the entire pool.

```
In [4]: output_df = pd.DataFrame()


        for folder in folders:
            files = [f for f in listdir(path_to_data + folder + '/') if isfile(join(path_to_data

            for file in files:
                df = pd.read_csv(path_to_data + folder + '/' + file, sep='\t', header=0, usecols
                df.drop(df.loc[df['class']==0].index, inplace=True)
                output_df = output_df.append(df)

        output_df = output_df.sample(frac=1).reset_index(drop=True)
        output_df.head()
```

Out[4]:

	channel1	channel2	channel3	channel4	channel5	channel6	channel7	\
0	-0.00001	-0.00003	-0.00001	0.00001	0.00001	0.00000	-0.00002	
1	-0.00005	-0.00005	-0.00005	-0.00005	-0.00003	-0.00003	-0.00001	
2	0.00005	-0.00002	0.00001	0.00004	0.00008	0.00009	0.00007	
3	0.00056	0.00006	0.00002	0.00000	-0.00001	-0.00001	0.00006	
4	-0.00001	0.00001	0.00018	-0.00033	0.00041	0.00031	0.00025	

	channel8	class
0	0.00000	1.0

```
1  -0.00004    1.0
2   0.00011    3.0
3   0.00024    3.0
4   0.00013    5.0
```

The final task is to analyse the final dataset. With the following info function available in pandas.DataFrame, we are able to see some high level information about the data. The total amount of records amounted to over 1.5 million with all values in every channel and class non-null. Although the values are expressed here as float64, the class variable will be modified to be an ungsigned integer in the database as a full 64 bit float value is considerably more storage than required for this variable.

```
In [5]: output_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1512751 entries, 0 to 1512750
Data columns (total 9 columns):
channel1    1512751 non-null float64
channel2    1512751 non-null float64
channel3    1512751 non-null float64
channel4    1512751 non-null float64
channel5    1512751 non-null float64
channel6    1512751 non-null float64
channel7    1512751 non-null float64
channel8    1512751 non-null float64
class       1512750 non-null float64
dtypes: float64(9)
memory usage: 103.9 MB
```