Table Of Content:

1. Control Methods

To effectively restore limb functions in amputees, the advanced prostheses that are controlled by electromyography (EMG) signals have been widely investigated and used. Since EMG signals reflect neural activity, they would contain information on the muscle activation related to limb motions. Pattern recognition-based myoelectric control is an important branch of the EMG-based prosthetic control. And the EMG-based prosthetic control theoretically supports multiple degrees of freedom movements that allows amputees to intuitively manipulate the device.

Fundamentally, EMG signals represent electrical manifestation of neuromuscular activation associated with a contracting muscle. Such signals have been proven to contain a rich set of neural information from which different types of upper limb movements could be accurately decoded, thus making them useful control input to prosthetic devices. It has been established that EMG signal patterns for a specific limb movement are repeatable over time and distinct from the patterns of other limb movements. So we can use this phenomenon to decode a range of arm movements from EMG recordings using artificial intelligent algorithms.

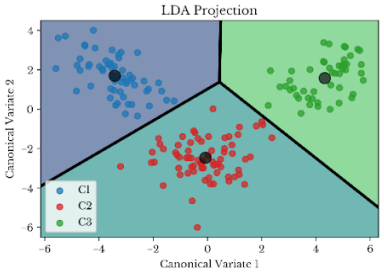
Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

The ideal characteristic of artificial intelligence is its ability to rationalize and take actions that have the best chance of achieving a specific goal. A subset of artificial intelligence is machine learning, which refers to the concept that computer programs can automatically learn from and adapt to new data without being assisted by humans. Deep learning techniques enable this automatic learning through the absorption of huge amounts of unstructured data such as text, images, or video.

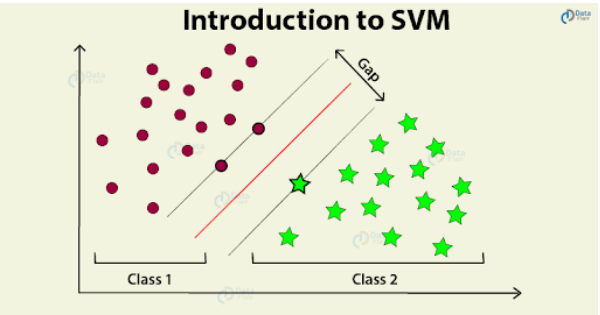
1.1. Machine Learning

Machine learning algorithms are built to “learn” to do things by understanding labelled data, then use it to produce further outputs with more sets of data. However, they need to be retrained through human intervention when the actual output isn’t the desired one.

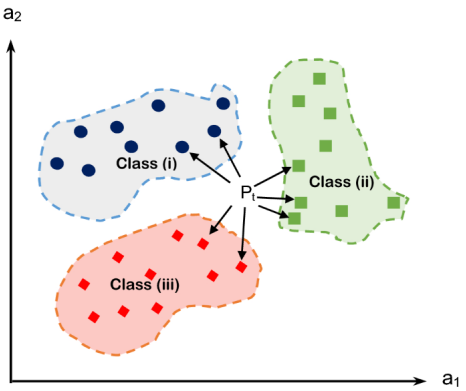
1.1.1. Linear discriminant analysis (LDA), a method used in statistics, pattern recognition, and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.



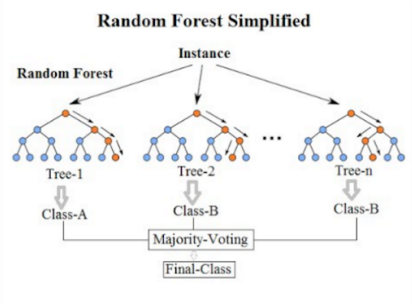
1.1.2. Support-vector machines "SVM", are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. it presents one of the most robust prediction methods, based on the statistical learning framework. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall. When data are unlabeled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The support-vector clustering algorithm, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data, and is one of the most widely used clustering algorithms in industrial applications.



1.1.3. K nearest neighbors "KNN" is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique. a case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. the case is simply assigned to the class of its nearest neighbor.



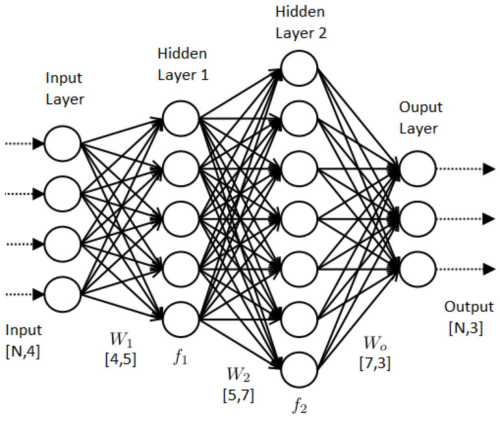
1.1.4. Random forests "RF", are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random forests generally outperform decision trees, Random forests are frequently used as "Blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration in packages such as scikit-learn.



1.2. Deep Learning

Deep learning networks do not require human intervention as the nested layers in the neural networks put data through hierarchies of different concepts, which eventually learn through their own errors. However, even these are subject to flawed outputs if the quality of data isn't good enough.

1.2.1. Neural Networks "NN", are one of main pattern recognition techniques; they comprise a large number of neurons, and these neurons are connected in a layered manner. The training procedure of a neural network can be easily achieved by optimizing the unknown weights to minimize a pre-selected fitness function. Generally, the neuron architecture can be summarized as the following: A neuron (or node) receives inputs, and then respective weights are applied on these inputs. Then, a bias term is added on the linear combination of the weighted input signals. The resulting combination is mapped through an activation function. Usually, the NN consists of input and output layers, as well as hidden layers that permit the neural network to learn more complex features.

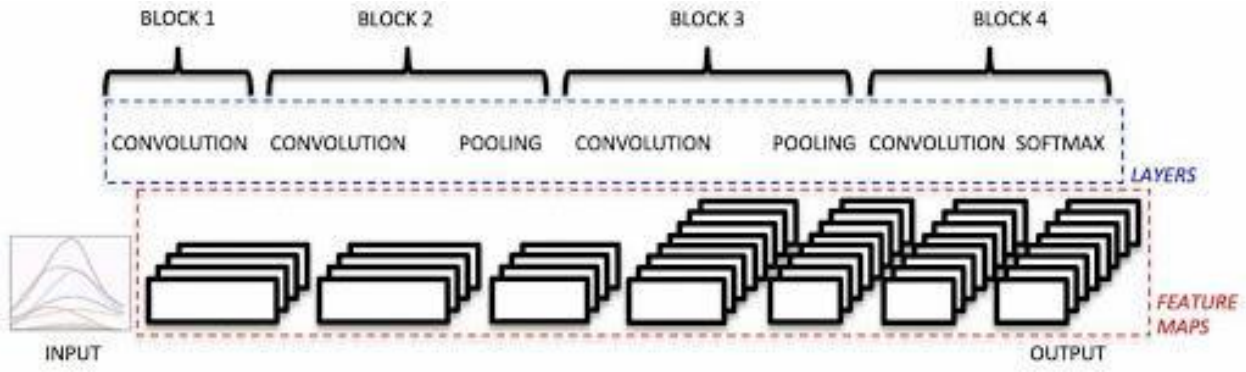


1.2.2. Convolutional Neural Network "CNN", is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

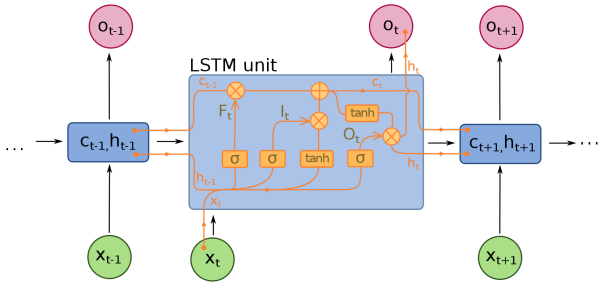
Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.



1.2.3. Recurrent Neural Network

recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs. This makes them applicable to tasks with time series signals



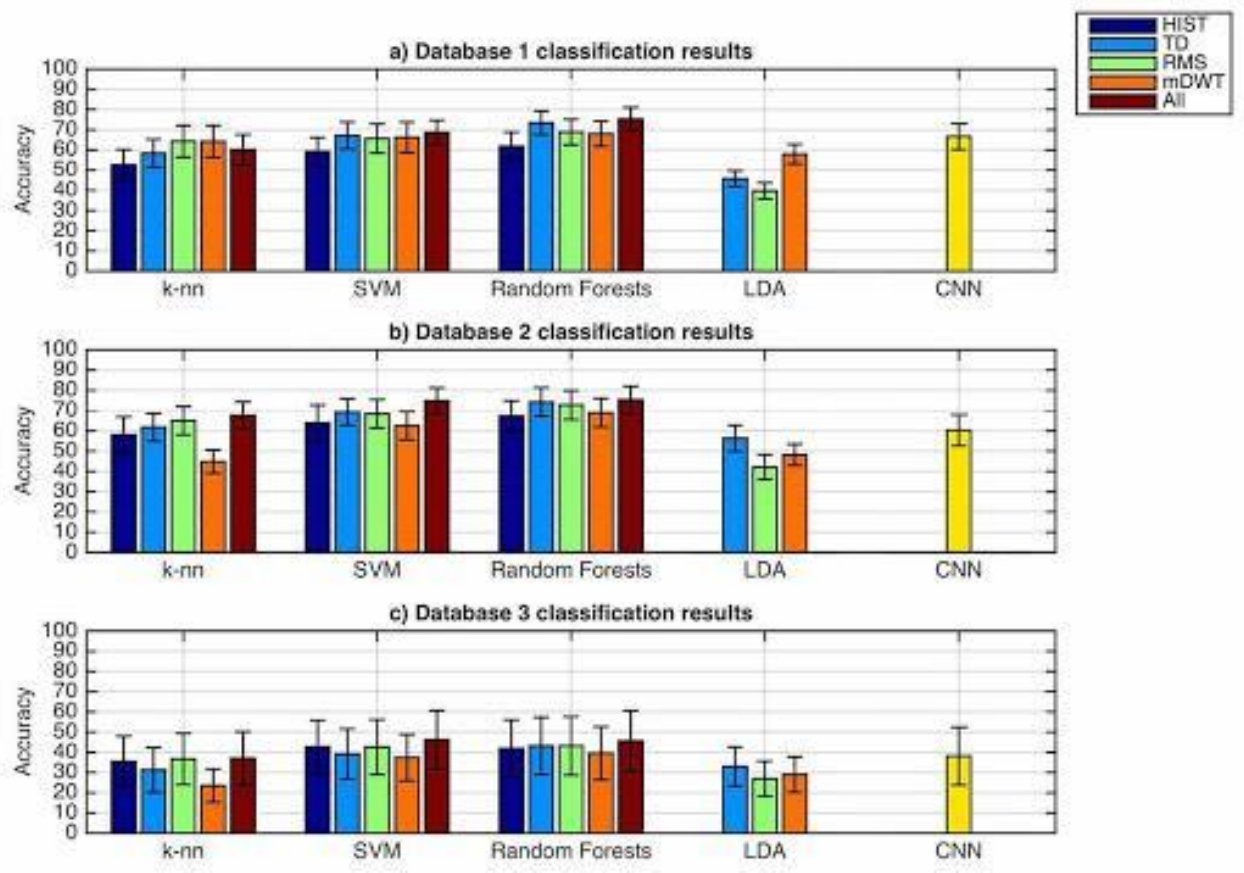
1.3. Algorithm Selection

based on some of research papers, a different machine learning and deep learning algorithms have been implemented with three different datasets with an average of 50 hand movement, and by analysing them, the classification accuracy obtained with convolutional neural networks using the simple architecture proposed is comparable with the average results obtained from classical classification techniques, but lower than the best results obtained with classical classification techniques, results higher than 90% can be easily obtained with similar approaches by reducing the number of classes, even on amputees.

The average classification accuracy obtained using the convolutional neural network on (dataset 1) is 66.59 ± 6.40%. The average classification accuracy obtained using all the classical methods on this dataset is 62.06 ± 6.07%. The best classical classification method (Random Forests with all features) obtained an average classification accuracy of 75.32 ± 5.69%.

The average classification accuracy obtained using the convolutional neural network on (dataset 2) is 60.27 ± 7.7%. The average classification accuracy obtained using all the classical methods on this dataset is 60.28 ± 6.51%. The best classical classification method (Random Forests with all features) obtained an average classification accuracy of 75.27% ± 7.89%.

For amputees (dataset 3), the average classification accuracy obtained using the convolutional neural network is 38.09 ± 14.29%. The average classification accuracy obtained using all the classical methods on this dataset is 38.82 ± 11.99%. The best classical classification method (SVM with all features) obtained an average classification accuracy of 46.27% ± 7.89%.



So, based on these statistics, and taking into account that the hand gestures vary in orientation of fingers and shape of hands. So, non-linearity is one of the characteristics of hand gestures that has to be dealt with. and it can be done using the metadata and content information carried by the images.

The process is the integration of two tasks: feature extraction and classification. Prior to the recognition of any gesture, the features of an image must be extracted. After extracting those features, any classification method should be applied. So, the main problem is how to extract those features and use those features for classification. Machine learning based models for pattern recognition cannot process natural data in raw form. Therefore, huge efforts are

required to extract features from raw data and they are not automated. CNN, a class of deep learning neural networks,can extract features on the fly and fully connected layers can be used for classification. CNN combines these two steps to reduce the memory requirements and computational complexity and gives a better performance. It can also understand the complex and nonlinear relationships among the images. Therefore, a CNN based approach is selected for the project.

2. Datasets and data acquisition

Remarkable progress has been made in image recognition, primarily due to the availability of large-scale annotated datasets and deep convolutional neural networks (CNNs). CNNs enable learning data-driven, highly representative, hierarchical image features from sufficient training data. However, obtaining datasets as comprehensively annotated as ImageNet in the medical imaging domain remains a challenge. There are currently three major techniques that successfully employ CNNs to medical image classification: training the CNN from scratch, using off-the-shelf pre-trained CNN features, and conducting unsupervised CNN pre-training with supervised fine-tuning. Another effective method is transfer learning, i.e., fine-tuning CNN models pre-trained from natural image dataset to medical image tasks.

Our approach was searching for the available datasets online to train a model on them from scratch, and then use the trained model for transfer learning to build a model with the data we will collect from us, as the datasets online will be larger and more diverse than what we could collect, so the model will be more reliable

2.1. Online datasets

2.1.1. EMG Signal for gesture recognition | Link

Data Set Information:

For recording patterns, we used a MYO Thalmic bracelet worn on a user's forearm, and a PC with a Bluetooth receiver. The bracelet is equipped with eight sensors equally spaced around the forearm that simultaneously acquire myographic signals. The signals are sent through a Bluetooth interface to a PC.

We present raw EMG data for 36 subjects while they perform a series of static hand gestures.The subject performs two series, each of which consists of six (seven) basic gestures. Each gesture was performed for 3 seconds with a pause of 3 seconds between gestures.

Number of Instances is about 40000-50000 recordings in each column (30000 listed as guaranteed)

Description of raw_data _*** file

Each file consist of 10 columns:

1) Time - time in ms;

2-9) Channel - eight EMG channels of MYO Thalmic bracelet;

10) Class –the label of gestures:

Available Gestures:

1. unmarked data,
2. hand at rest,
3. hand clenched in a fist,
4. wrist flexion,
5. wrist extension,
6. radial deviations,
7. ulnar deviations,
8. extended palm (the gesture was not performed by all subjects).

2.1.2. Electromyography(EMG) Dataset | Link

Data Set Information:

Contain two folders of raw emg data and two folders of extracted features from raw data

1) raw emg data unprocessed folder

   The Electrodes are along the x-axis (columns 1 to 8) and the data points of the EMG-Wave are along the y axis (rows)

2) raw emg data cropped and arranged folder

   The samples are along y axis (rows) and the data points of the EMG-Wave are along x-axis (columns 1 to 150), The data in this folder is derived from the data inside "Raw_data_unprocessed" folder by applying algorithms to detect abrupt changes in the wave and then crop 150 points if a signal is present.

3) extracted features dataset folder

   For each of these features the electrodes are along x-axis (column 1 to 8) and the samples are along y-axis (rows), These features are extracted from the data inside the "Raw_data_cropped_and_arranged" folder. That folder contains separate csv for all electrodes. In those csv's samples are along the y-axis, but the points of the wave are along x-axis (columns 1 to 150) because the wave for each sample was

taken to be 150 points long. These features are extracted based on those 150 points for each electrode.

4) extracted features and labeled dataset folder

Features are along x-axis(columns 1 to 80) and Samples are along the y-axis(rows), The last column(81) consists of labels.

There are 80 columns because there were 8 electrodes and 10 features were extracted for each electrode.

Features are in the order (standard deviation, root mean square, minimum, maximum, zero crossings, average amplitude change, amplitude first burst, mean absolute value, waveform length, willison amplitude).

First 8 columns are standard deviation, the next 8 columns are root mean_square and so on according to the order described above.

Available Gestures:

1. Index_finger
2. Middle_finger
3. Ring_finger
4. Little_finger
5. Thumb
6. Rest
7. Victory_gesture

2.2. Data acquisition

2.2.1. Data with Muscle (EMG) Sensor Kit | Appendex_1

Acquisition protocol:

Using an arduino connected with three channels EMG sensor, a two minutes recording signal with sampling rate 115200 samples per second for each gesture was sampled and sent to the PC through the serial port. Duo to complexity of wiring and uncertainty of the electrodes location each time collecting the data, it was hard to collect from more than one person similar data, so the data was collected from only one person using this kit of sensors

➔ Arduino Code for Collecting data from 3 EMG kits Appendix_2

Data Set Information:

Contain two folders one for the raw output signal from the sensor and the other for the processed data.

1) Raw EMG data folder

   Contain five files, file for each gesture. Each file contain the recorded signal by three EMG channels for two minutes with sampling rate 115200 sample per second , the data format is in one column contain the reading from channel one followed by the reading from channel two and so on

   data format :

1	Channel1
2	Channel2
3	Channel3
4	Channel1
5	Channel2
6	Channel3
7	……..

2) Three Channels EMG data folder

   Contains of a single file with all the collected data cropped, arrangement, and labeled in an easy format to reshape it as images

   data format:

   1025 column and 1702 row, the first 1024 column are contains the recorded signal from the three channels EMG sensor, and the last column contain the signal class, the 1024 recorded signal will be reshaped as a grayscale image with size 32*32, the 1702 rows represent the total number of collected images

	channel1	channel2	channel3	……..	channel1	channel2	channel3	Class
1								
. . .								
1702								

Available Gestures:

1. Index
2. Middle
3. Ring
4. Little
5. Thumb

➔ Python Code for Process the data Appendix_3

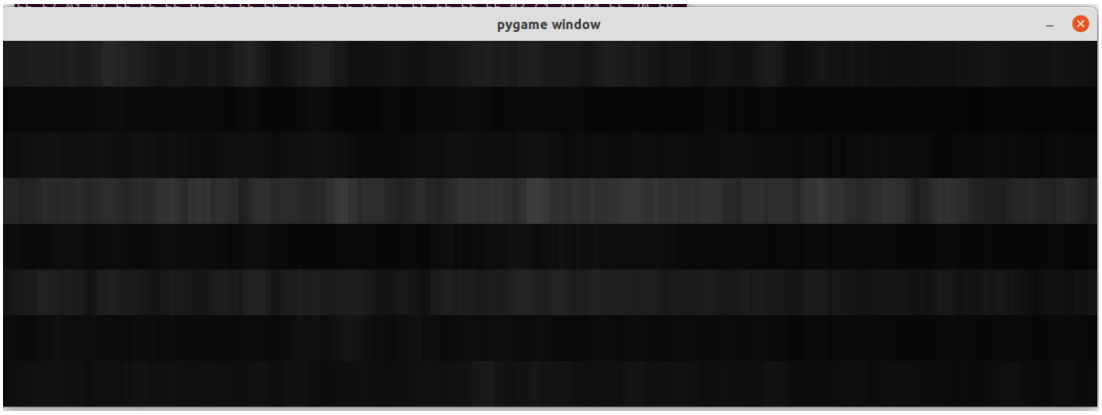
2.2.2. Data with Myo Armband Device | Appendex_4

Acquisition protocol:

An open source git-hub project was used to communicate with the bluetooth protocol of the myo armband device, the project is providing an interface to communicate with the Thalmic Myo, providing the ability to scan for and connect to a nearby Myo, and giving access to data from the EMG sensors and the IMU. For Myo firmware v1.0 and up, access to the output of Thalmic's own gesture recognition.

using this project with some editing and additional code ,we were able to extract the emg sensor reading and collecting them.

the bluetooth module was sampling with sample rate 9600 samples per second, and two minutes for each gesture, the data collected from 6 persons



EMG visualisation tool, contains of 8 lines each line represent channel signal as a grayscale pixels, black pixel means no muscle activity appears on its channel, and full white pixel means the muscle is highly active

➔ Python Code to Collecte 8 channels myo armband Appendix_5

Data Set Information:

Contain three folders, the first one for the raw output signal from the myo armband, and the second one for the data processed to work with Deep Neural Network (DNN), and the last one for the data processed as images to work with Convolution Neural Network (CNN).

1) Raw Myo Armband data folder

   Contain 10 folders, folder for each of the available gesture, each folder of the 10 folders contains a 6 files with the recorded data from 6 persons for two minutes, (the last two gesture collected from one person only)

   Data format:

channel1	channel 2	channel3	channel4	channel5	channel6	channel7	channel8
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

2) DNN Armband data folder

   Contains a single file with all the recorded data from the 6 persons, and labeled gestures. The file contains about 436,000 frames of data to train the DNN model.

   Data format:

channel1	channel 2	channel3	channel4	channel5	channel6	channel7	channel8	Class
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

➔ Python Code for Process the DNN data Appendix_6

3) CNN Armband data folder

   Contains a single file with all the recorded data from the 6 persons, and labeled gestures. The file contains about 7,000 frames of data as images to train the CNN model. The file is 513 column, the first 512 column is the recorded data from the eigh EMG channels for 64 sample time, the first 64 column is for the data from the first channel and the second 64 for the second channel and so on, while the last column is for the class name.

   these data will reshaped as images with 8 layers each layer represent a channel and with size of 8*8

Data format:

channel1	channel 1	channel1	……..	channel8	channel8	channel8	Class
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

➔ Python Code for Process the CNN data Appendix_7

Available Gestures:

1. Rest
2. Close
3. Pointing
4. Victory
5. Wrist wave out
6. Wrist wave in
7. Ok
8. Thumbs up
9. Wrist UP
10. Wrist Down

3. Convolution Neural Network Classifier

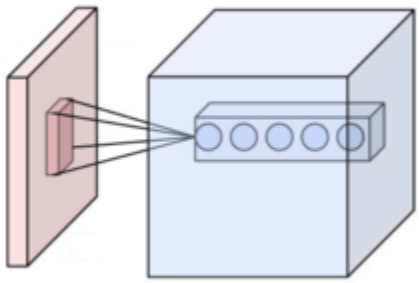
The term Deep Learning or Deep Neural Network refers to Artificial Neural Networks (ANN) with multi layers . Over the last few decades, it has been considered to be one of the most powerful tools, and has become very popular in the literature as it is able to handle a huge amount of data. The interest in having deeper hidden layers has recently begun to surpass classical methods performance in different fields; especially in pattern recognition. One of the most popular deep neural networks is the Convolutional Neural Network (CNN). It takes this name from mathematical linear operation between matrices called convolution. CNN has multiple layers; including convolutional layer, non-linearity layer, pooling layer and fully-connected layer. The convolutional and fully-connected layers have parameters but pooling and non-linearity layers don't have parameters. CNN has an excellent performance in machine learning problems. Especially the applications that deal with image data.

Convolutional neural networks provide an efficient method to constrain the complexity of feedforward neural networks by weight sharing and restriction to local connections. This network topology has been applied in particular to image classification when sophisticated preprocessing is to be avoided and raw images are to be classified directly.

3.1. The Main Layers of Convolutional Neural Network

A CNN architecture is formed by a stack of distinct layers that transform the input volume into an output volume through a differentiable function. A few distinct types of layers are commonly used. These are further discussed below.

1) Convolutional layer



The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the filter entries and the input, producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

2) Pooling layer

Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling, where max pooling is the most common. It partitions the input image into a set of rectangles and, for each such sub-region, outputs the maximum.

Intuitively, the exact location of a feature is less important than its rough location relative to other features. This is the idea behind the use of pooling in convolutional neural networks. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters, memory footprint and amount of computation in the network, and hence to also control overfitting. This is known as down-sampling It is common to periodically insert a pooling layer between successive convolutional layers (each one typically followed by an activation function, such as a ReLU layer) in a CNN architecture.

3) ReLU layer

ReLU is the abbreviation of rectified linear unit, which applies the non-saturating activation function $f(x) = \max(0, x)$.It effectively removes negative values from an activation map by setting them to zero.It introduces nonlinearities to the decision function and in the overall network without affecting the receptive fields of the convolution layers.

Other functions can also be used to increase nonlinearity, for example the saturating hyperbolic tangent $f(x) = \tanh(x)$, and the sigmoid function $\sigma(x) = (1 + e^{-x})^{-1}$. ReLU is often preferred to other functions because it trains the neural network several times faster without a significant penalty to generalization accuracy.

4) Fully connected layer

After several convolutional and max pooling layers, the final classification is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular (non-convolutional) artificial neural networks. Their activations can thus be computed as an affine transformation, with matrix multiplication followed by a bias offset (vector addition of a learned or fixed bias term).

3.2. Models Implementation

for our four datasets we implemented a model to work with each, depending on the data shape, we were implementing a Deep Neural Network (DNN) or Convolutional Neural Network (CNN) as we will discuss below:

3.2.1. DNN with EMG Signal for gesture recognition (online dataset) | Appendex_8

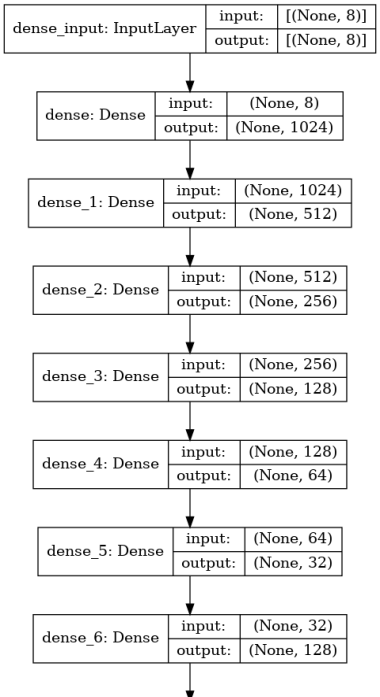
The data shape has 8 columns, each column represents a channel reading from the myo armband, so the suitable architecture to work with is a deep neural network.

Model architecture:

The first layer of the model takes data of shape 8, which is the 8 channels emg reading, and passes it to the second layer.

The second layer takes the emg readings represented in the 8 neurons and passes it to 1024 neurons, and so on as shown in the figure.

The last layer output is 8 neurons, which represents the 8 classes of this dataset, and by using softmax function the network decides the class of the input data.

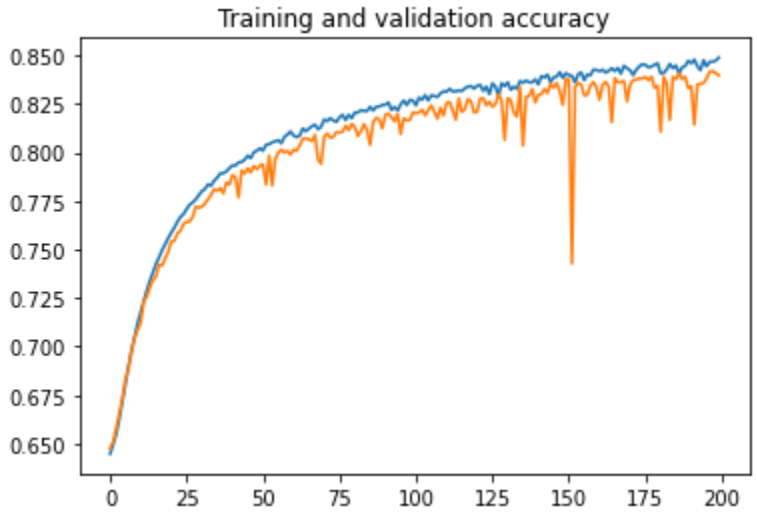


Model results:

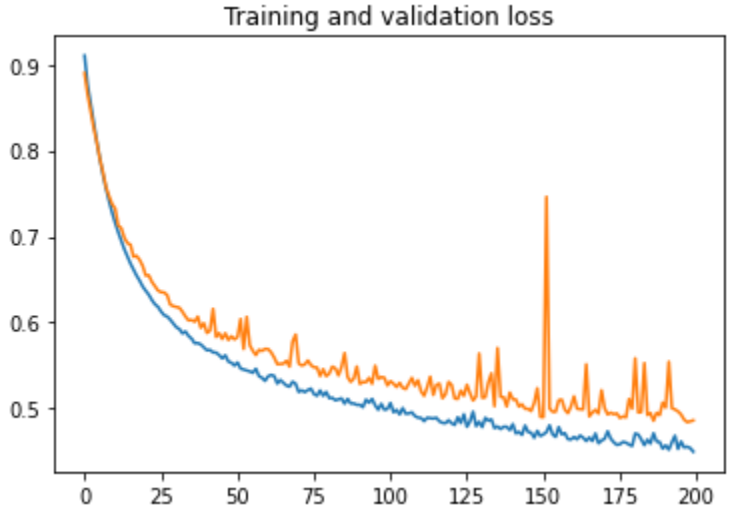
The model is optimized with NADAM optimizer with learning rate of 1×10^{-3}.

The loss is measured with the categorical_crossentropy function.

The model takes about 41 second to make a prediction



Accuracy on training : 84.9% Accuracy on validation: 83.9%

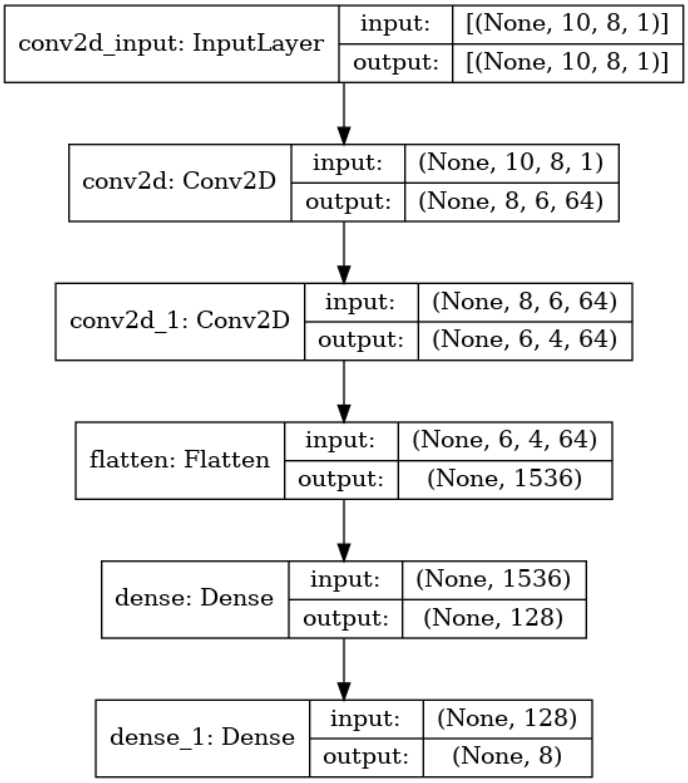


Loss on training : 0.45 Loss on validation: 0.49

3.2.2. CNN with Electromyography(EMG) Dataset (online dataset) | Appendex_9

We used the extracted features file to build our network, the data shape of the features has 80 columns as described above, we reshaped the data to make images that has a single layer and size of 10*8, so the suitable architecture to work with is a Convolution neural network.

Model architecture:



The input layer of the model takes data of shape 10*8*1, which is the 10 features of the 8 channels emg reading, and passes it to the second layer.

The second layer is a convolution 2d layer, takes the emg readings represented as an image and makes the convolution operation with 64 layers of the 2d conv layer, and so on as shown in the figure.

The flatten layer is used to make the represented data in 6 * 4 with 64 layer represented only in single layer to connect it with a fully connected layer

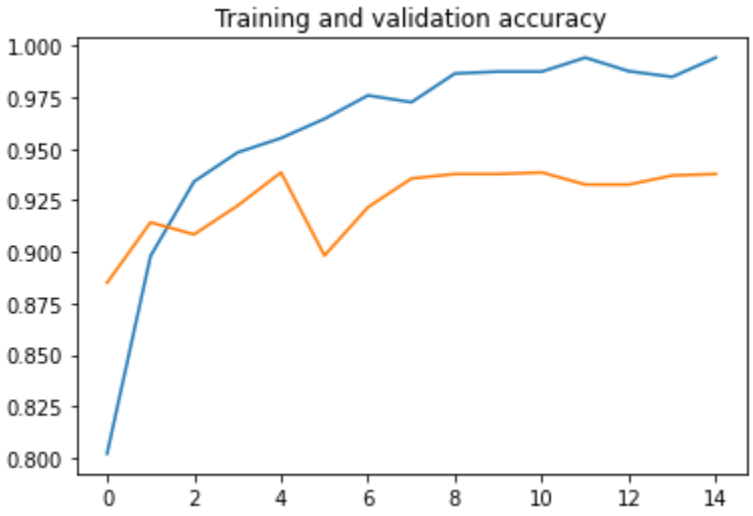
The last layer output is 8 neurons, which represents the 8 classes of this dataset (there is a redundant class, as this dataset contains only seven classes, the class 0 is redundant), and by using softmax function the network decides the class of the input data.

Model results:

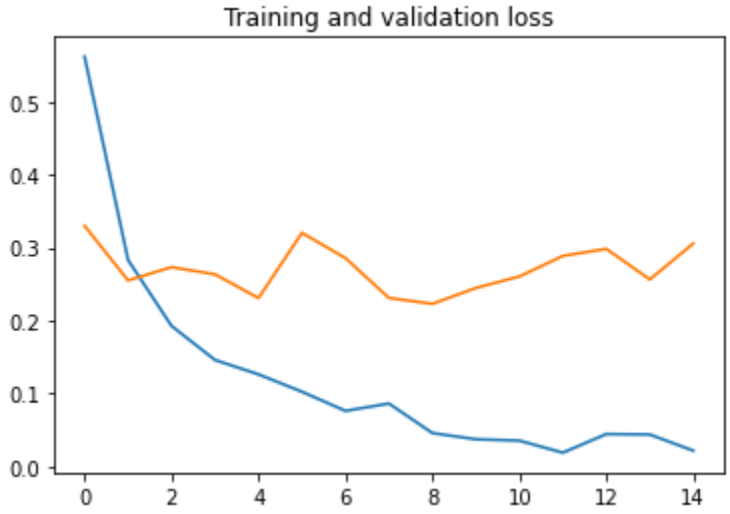
The model is optimized with ADAM optimizer.

The loss is measured with the categorical_crossentropy function.

The model takes about 0.17 second to make a prediction



Accuracy on training : 99.5% Accuracy on validation: 93.8%



Loss on training : 0.02 Loss on validation: 0.30

- it appears that the model start to overfit on the training data

3.2.3. CNN with Muscle (EMG) Sensor Kit Dataset (collected dataset) | Appendex_10

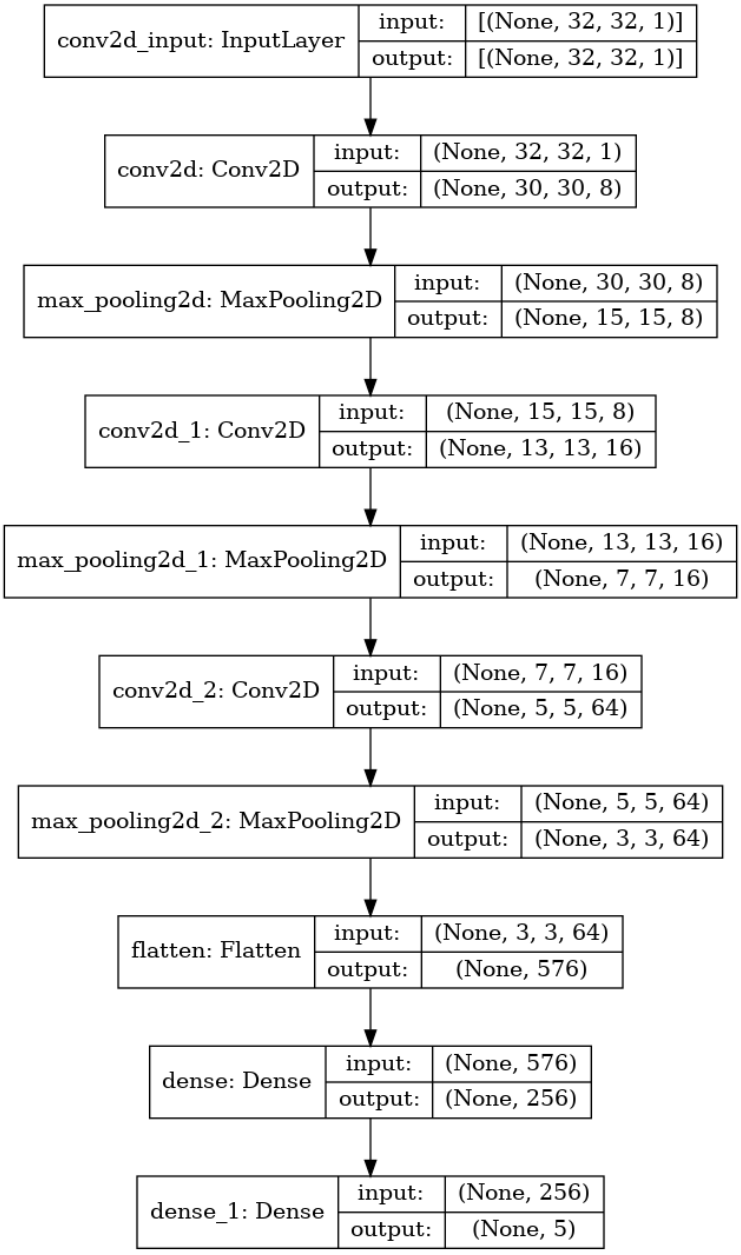
our collected data is saved as 1024 column to reshape them as images with a single layer and size of 32*32, to classify them using convolution neural network CNN

Model architecture:

The input layer of the model takes data of shape 32*32*1, which is the 1024 sample of the 3 channels emg reading, and passes it to the second layer.

A max pooling layer is used after each conv 2d layer as nonlinear operation to progressively reduce the spatial size of the representation, to reduce the computational power

The last layer output is 5 neurons, which represents the 5 classes of this dataset, and by using softmax function the network decides the class of the input data.



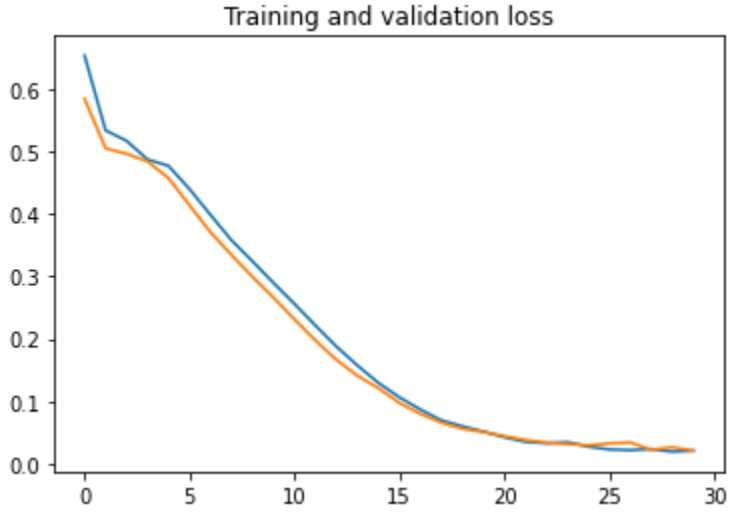
Model results:

The model is optimized with ADAM optimizer with learning rate of 1×10^{-3}.

The loss is measured with the categorical_crossentropy function.

The model takes about 0.14 second to make a prediction

Accuracy on training : 98.9% Accuracy on validation: 98.8%



Loss on training : 0.023 Loss on validation: 0.021

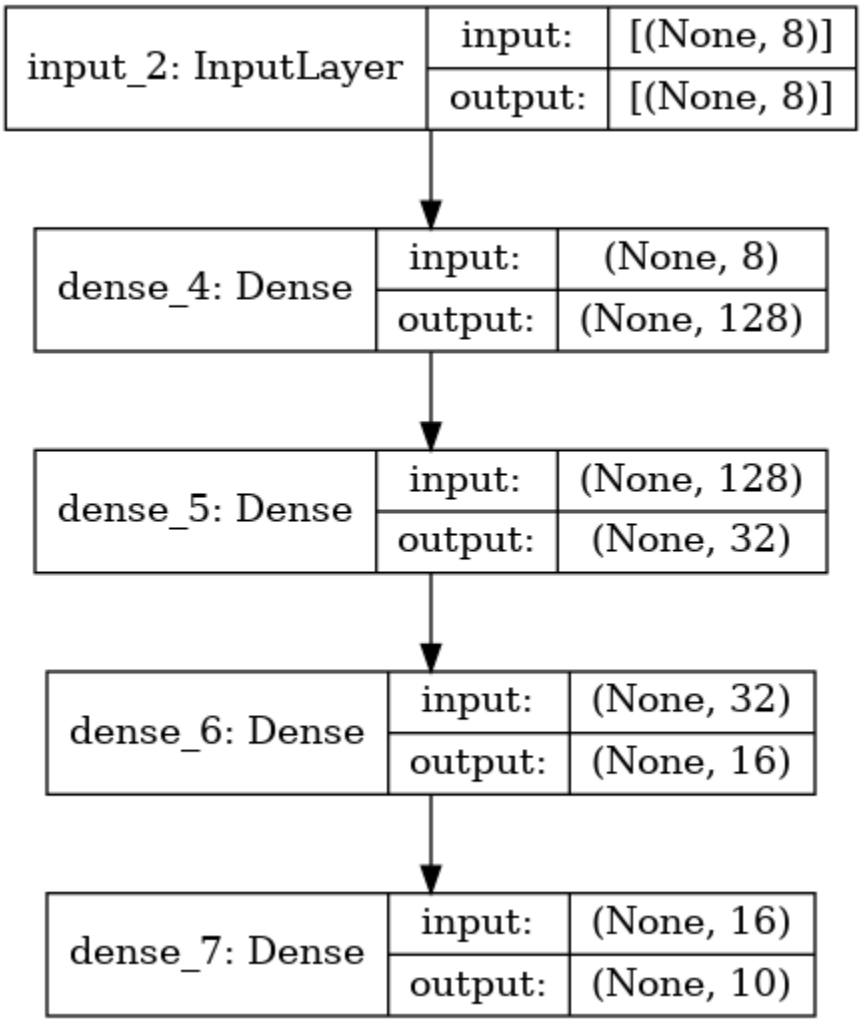
- This very high accuracy result is due to the fact that all data used in the training and testing of the model was acquired from one person only.

3.2.4. DNN with Myo Armband Dataset (collected dataset) | Appendex_11

The data from myo armband is an array of the readings from the eight channels emg sensor. and to make the time delay as low as possible, a single data frame is used to predict to reduce the time delay from acquisitioning of the data. so, to work with this shape of data directly the Deep Neural Network model is built.

Model architecture:

The first layer of the model takes data of shape (8,) which is the 8 channels emg reading, and the output layer is a softmax function to classify the input to the 10 available classes

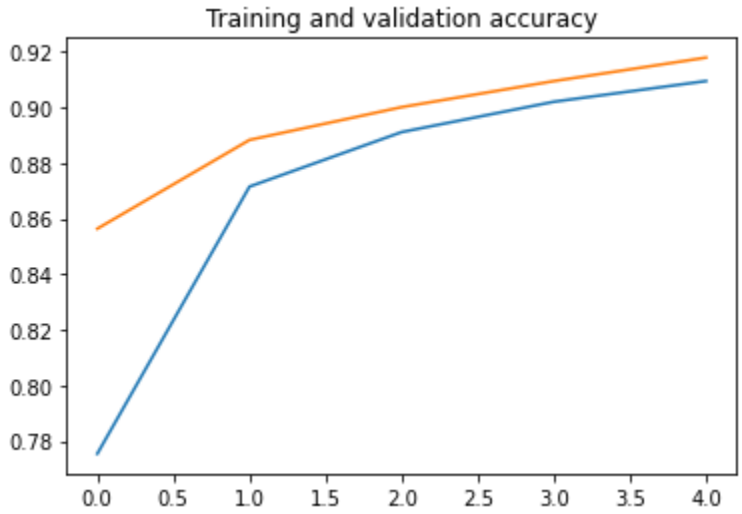


Model results:

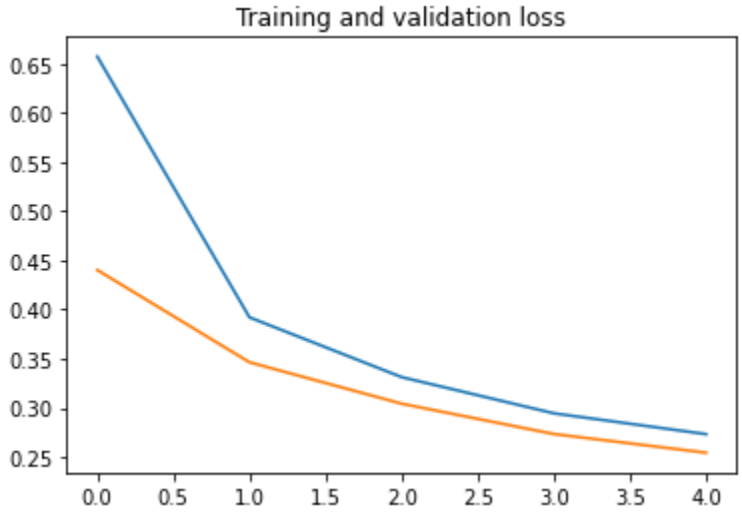
The model is optimized with ADAM optimizer.

The loss is measured with the sparse_categorical_crossentropy function.

The model takes about 0.5 second to make a prediction



Accuracy on training : 90.8% Accuracy on validation: 91.7%

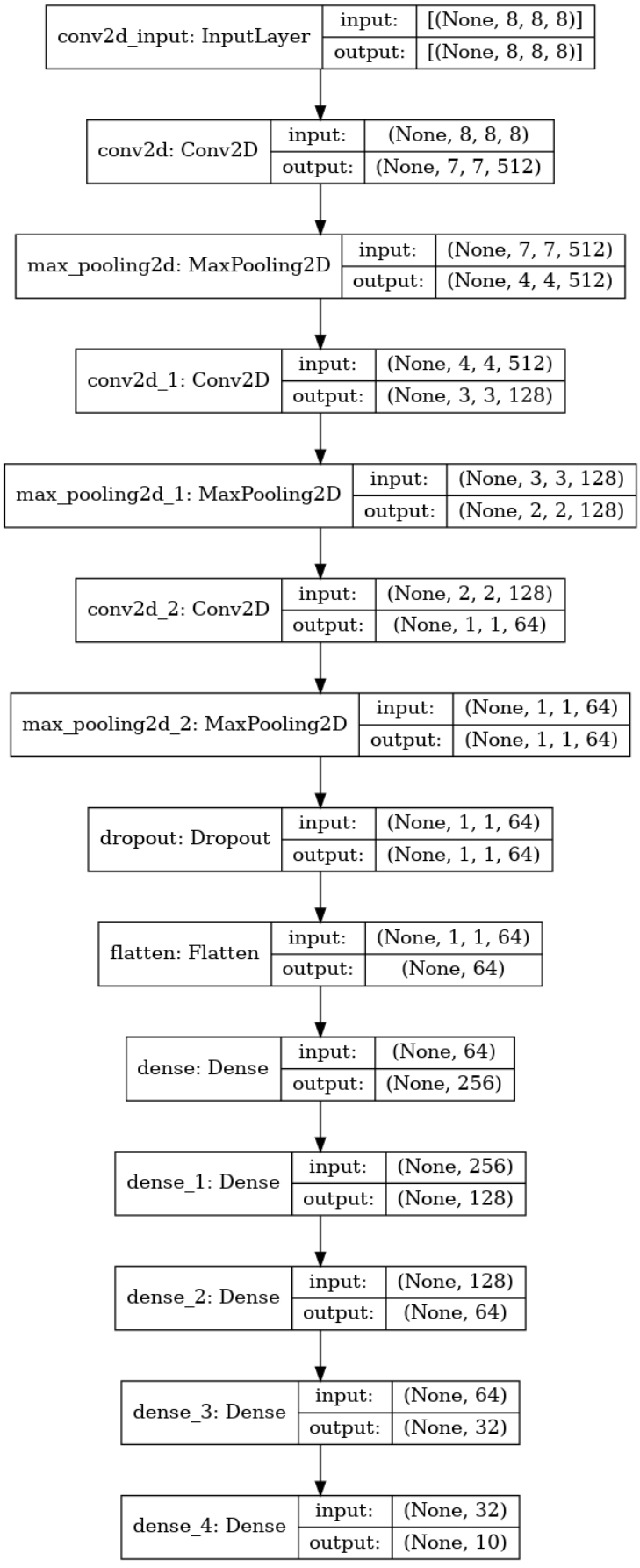


Loss on training : 0.27 Loss on validation: 0.25

3.2.5. CNN with Myo Armband Dataset (collected dataset) | Appendex_12

to minimize the error of the noise, more than one sample needed to collect and take the average of them to make a prediction. so, our CNN collected dataset was formatted to make an images with 64 sample from the eight emg channels

Model architecture:



Model results:

The model is optimized with ADAM optimizer with learning rate of 1×10^{-3}.

The loss is measured with the categorical_crossentropy function.

The model takes about 0.25 second to make a prediction

Accuracy on training : 96.5% Accuracy on validation: 94.0%

Loss on training : 0.10 Loss on validation: 0.20

4. Algorithm Results in the Real Time

The CNN model with Muscle (EMG) Sensor Kit wasn't working well in the real time due to the very noisy signal from he sensor, and it was hard to place it at the exact location as the data collected from, as well as we couldn't acquire enough data using this sensor because it can't be fixed well on the arm. Also the sensor signal is rectified which makes the signal lose a lot of its unique features and leads to poor performance.

using the myo armband helped us get raw emg signals without noise, and collecting more and enough data from more people to train the CNN model. the model in the real time was able to make almost all the 10 classes predictions except the pointing class the model was highly confused about its prediction

The delay time of the model response was barely noticeable

➔ Code to Run the Myo Armband CNN Model in Real Time Appendex_13

5. Future Work

5.1. Implementing RNN model

The RNN architecture is efficient with data with time series, but it also needs more computational power than the CNN architecture. so, we need to implement a RNN model to see its result and compare it with the implemented CNN model.

5.2. Implementing an Embedded Linux system to make the model run on raspberry pi board instead of the PC, so the arm would be movable

6. Appendix | Drive Link

| GP_High Level Control
|
|
|_____ | Collected Datasets
| |
| |
| |_____ | Data with Muscle (EMG) Sensor Kit | Appendix_1
| | |
| | |_____ Arduino Code for Collecting data from 3 EMG kits | Appendix_2
| | |
| | |_____ Python Code for Process the data | Appendix_3
| |
| |_____ | Data with Myo Armband Device | Appendix_4
| |
| |_____ Python Code to Collecte 8 channels myo armband | Appendix_5
| |
| |_____ Python Code for Process the DNN data | Appendix_6
| |
| |_____ Python Code for Process the CNN data | Appendix_7
|
|
|_____| DNN with EMG Signal for gesture recognition (online dataset) | Appendix_8
|
|
|
|
|_____| CNN Model with Electromyography(EMG) Dataset (online dataset) | Appendix_9
|
|
|
|
|_____| CNN Model with Muscle (EMG) Sensor Kit Dataset (collected dataset) | Appendix_10
|
|
|
|
|_____| DNN Model with Myo Armband Dataset (collected dataset) | Appendix_11
|
|
|
|
|_____| CNN Model with Myo Armband Dataset (collected dataset) | Appendix_12
|
|
|
|
|_____| Python Code to Run the Myo Armband CNN Model in Real Time | Appendix_13