Final Report

ECE9603: Data Analytics

November 26, 2020

Names
Names
Names
Names

Methodology - Established (FFNN and SVM)

Data Preparation

The dataset consisted of muscle activity readings for 8 sensors which measured the electrical activity produced by muscle movement. Each line consisted of 8 consecutive readings for all 8 sensors, yielding 64 columns of EMG data plus a target column for the gesture. The data was fairly balanced across all gestures, with each gesture occupying 2900 lines of data. Each 1x64 row was reformatted by stacking corresponding sensor readings on top of each other, transforming each entry into an 8x8 matrix. Doing so allowed each row to represent one time step for all 8 sensors. A target column was appended to each entry for each corresponding gesture. Data was recorded at 200 Hz, meaning that each line represented 40ms of muscle activity readings. Our review of studies in the related work found that data is often divided into segments and analyzed in time windows. These windows can either be contiguous or overlapping [**Figure X**]. The optimal window length for upper-limb myoelectric devices is within the range of 150-250 ms [**REFERENCE NUMBER**]. We chose a window duration of 200 ms with an overlap of 50 ms. This resulted in an average of 777 windows generated between all four gestures.

Figure 1: EMG Signal with overlapping windows

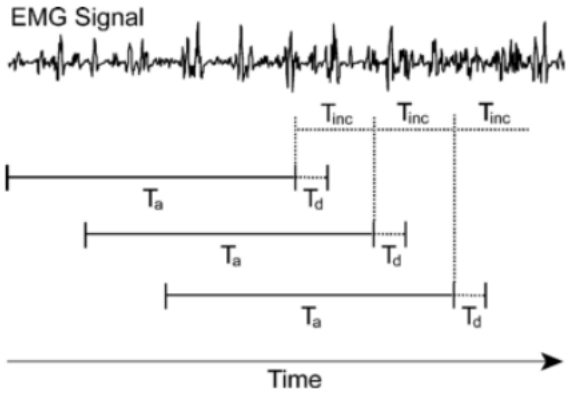

Figure 2.6: EMG signal with overlapping windows. T_a is the duration of the window, T_{inc} is the duration of the increment of new data added to each window, and T_d is the processing time. Reprinted, with permission, from [85] ©2011 IEEE.

For the neural networks, the gestures were encoded in one-hot format such that a softmax activation function could be used for the output layer. The SVM left the target classes in their numerical form 0 through 3. The data was randomly split 60/20/20 into training, validation, and testing data. The networks were tuned using the validation and testing data, and then after the network parameters were finalized, it was tested with the test data for comparison amongst the three types of networks created.

Feature Extraction

After the data is windowed, several features are computed for each windowed section to increase the total number of attributes and obtain additional information from the EMG data. There are many options for the time-domain features that can be obtained, but the four most popular options were selected. Each feature is computed for each of the 8 data sensors, resulting in a total of 56 attributes.

Mean Absolute Value (MAV)

To compute the MAV, the absolute values are taken for each reading, then averaged across each window, as illustrated in Equation 1, where $x_n(i)$ is the reading for channel n at time i, and m is the window size.

Equation 1: MAV Feature Calculation

$$MAV(n) = \frac{1}{m}\sum_{i=1}^{m} |(x_n(i)|$$

Waveform Length (WL)

This feature states the cumulative length of the signal, as per Equation 2, where N is the length of the signal and x(i) is the ith sample of the signal.

Equation 2: WL Feature Calculation

$$WL = \sum_{i=1}^{N-1} |(x(i+1) - x(i)|$$

Zero Crossing (ZC)

This feature counts the number of times the signal crosses zero.

Auto-Regressive Coefficients (AR)

Each channel is represented by a 4^{th} order AR model and the features are the coefficients for the model. The coefficients can be computed using Equation 3, where P is the AR order, ap are the AR coefficients, and w_i is a white noise term.

Equation 3: AR Equation

$$x_i = \sum_{p=1}^{P} a_p x_{i-p} + w_i$$

SVM

A Support Vector Classifier (SVC) was used to approximate the decision boundaries between the four gesture classes. Four hyperparameters were tuned during training. An L2 regularization parameter C was introduced to prevent the model from overfitting. A kernel parameter specified which kernel to use during computation and offered a variety of paths the model could take towards finding suitable decision boundaries. For instance, the linear kernel would only be effective at finding linearly separable decision boundaries whereas the polynomial and RBF kernels are especially useful when the data points are not linearly separable. When the polynomial kernel function is used, the degree of the polynomial is specified by the 'degree' hyperparameter. Similarly, the 'gamma' parameter specifies the kernel coefficients for the RBF, polynomial and sigmoid functions. The values of the hyperparameters space are given below.

Figure 2: SVM Hyperparameter Tuning Space

Hyperparameter	**Options**
C	1, 10, 100, 1000
Kernel	'Linear', 'Poly', 'RBF', Sigmoid'
Degree	1, 3, 5, 7
Gamma	'Scale', 'Auto'

A baseline SVC model was instantiated with the default parameters {'C': 1, 'kernel': linear, 'gamma': 'scale'} which was then fit on the training data. 10-fold cross validation was used to obtain a baseline train score. GridSearchCV was applied to search over the hyperparameter space given in the table above. GridSearchCV exhausted all possible combinations of parameters looking for the model yielding the best 10-fold cross validation accuracy score. Once obtained, the selected model was evaluated on the test set for final results.

FFNN

A simple fully connected FFNN was created to distinguish the gestures from each other. The input layer had 56 neurons as there are 56 features being input into the network, and the output layer had four neurons to support the softmax activation function for the last layer. L2 regularization was

used in all layers to help prevent overfitting. The number of hidden layers and the number of neurons in the hidden layer were tuned using the default parameters for the all other hyperparameters. The loss functions of these networks were observed to determine the number of epochs that should be used for fine tuning the network. The range of neurons in the hidden layer were computed using Equation 4 where a was varied from 4 to 10 to attempt to prevent overfitting [**REFERENCE - http://hagan.okstate.edu/NNDesign.pdf#page=469**]. This resulted in using 4, 6, and 8 neurons in the hidden layers.

Equation 4: Number of neurons per hidden layer

$$\#\ Neurons\ in\ hidden\ layer = \frac{\#\ Data\ points\ in\ training\ set}{a * (\#\ Neurons\ in\ input\ layer\ +\ \#\ Layers\ in\ output\ layer)}$$

After the network structure was settled, the learning rate, learning function, and activation function were tuned in a grid search. [**FIGURE XX**] summarizes the options investigated for the hyperparameters that were tuned. Default parameters were used for all other options. The models for tuning were evaluated using the accuracy evaluation metric. After completing the hyperparameter tuning, the best model was evaluated on the test data.

Figure 3: FFNN Hyperparameter Tuning Space

Hyperparameter	**Options**
Number of Hidden layers	0, 1, 2
Number of Neurons in hidden layers	4, 6, 8
Learning Function	'Adam', 'SGD'
Learning rate	0.001,0.01,0.05,0.1,0.2,0.3,0.4
Activation function	'ReLU', 'ELU', 'Sigmoid'

Methodology - Novel (LSTM)

Data Preparation

EMG data from each gesture was divided into sliding windows of 200 ms and 50 ms overlap. Then, each window was stacked on top of each other horizontally to form a 3D array that was going to be used to train the LSTM network. Each row of the 3D array represented a training sample, whereas each column represented the EMG reading of one of the eight channels/sensors of the Myo Armband. The third dimension of the 3D array represented the lag values of each EMG sensor/channel. After creating a 3D array for each gesture, the rows of this 3D array were randomly shuffled and then were divided into the training set, the cross-validation set, and the testing set, each one having 60%, 20%, and 20%, respectively, of the whole gesture dataset. This procedure was repeated for each of the remaining gesture datasets. Having divided the data into the three sets, the data labels were separated and then, one-hot encoding was used on the data labels in order for them to be classified using a softmax activation function.

LSTM

LSTM networks have been widely used in sequence problems where there exists a time dependency between data points. They have become popularized for being able to retain information over longer periods of time, a problem that traditional RNNs have struggled with. A baseline LSTM model was constructed with a 50-unit LSTM layer which then passed through a dropout layer, dropping 20% of the neurons. The remaining neurons were fed into a 100-unit dense layer with the ReLU activation function. In order to output classes, the dense layer was connected to a 4-unit softmax layer which outputted gesture classifications. Tuning took place on the number of hidden dense layers, the number of neurons per hidden layer, the activation function and the dropout rate. The values of the hyperparameter space are given below.

A baseline LSTM model was instantiated with the parameters {'# Hidden layers': 1, '# Neurons per layer': 100, 'Activation function': ReLU, 'Dropout rate': } which was then fit on the training data. Hold

Figure 4: LSTM Hyperparameter Tuning Space

Hyperparameter	Options
Number of Hidden layers	0, 1, 2
Number of Neurons in hidden layers	4, 6, 8
Activation function	'ReLU', 'ELU', 'Sigmoid'
Dropout rate	0.15, 0.18, 0.21, 0.24, 0.27, 0.30

out cross validation was used to obtain a baseline train score. During tuning, RandomSearch randomly sampled a set of parameters from the available parameters and returned the model yielding the lowest hold out validation score. The selected model was then evaluated on the test set for final results.

Results

Evaluation Metrics

Three metrics were used to evaluate the performance of our models: accuracy, precision and recall. Accuracy was the most important metric since it represents the proportion of true results among the total number of cases examined. Precision was used to examine the proportion of predicted gestures that were actually correct, for each class. Recall was used to evaluate the proportion of actual gestures that were correctly identified by the models. The equations for how we calculated these metrics are given below.

Evaluation Metrics

$$\text{Accuracy} = \frac{\text{\# Correct predictions}}{\text{Total \# predictions}}$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}, \text{ per gesture}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}, \text{ per gesture}$$

SVM Results

Results after iterating through the hyperparameter space showed that linear and low-order polynomial models performed the best. In general, linear kernels consistently outperformed all other kernels. Almost every single linear kernel, regardless of the state of other hyperparameters, scored an impressive 99.48% accuracy on the validation set. This was only 0.4% lower than the highest score, which was found by a 1st-order polynomial kernel with minimal smoothing. Kernels that performed notably worse were the sigmoid and RBF kernels, scoring up to 75% worse than their peers when all other hyperparameter values were left equal. Regularization appeared to have minimal effect over the polynomial kernels and no effect on linear kernels. The validation score decreased as the degree of the polynomial kernel increased, signalling the data had a strong preference for linear models. The best performing models and their hyperparameters are summarized in [**Figure ZZ**]. The scores reported are their validation scores. Based on these results, Network 1 was selected to be evaluated on the test set. It scored a test accuracy of 99.36%. [**FIGURE XX**] shows the confusion matrix generated the predictions made on the test set by Network 1, and [**FIGURE XX**] shows the corresponding precision and recall scores.

FFNN Results

Upon investigating which network architecture performed best, the 2-layer network consistently resulted in higher accuracies than the 3- and 4-layer networks, as overfitting was beginning to occur. It was found that the loss function stabilized after approximately 30 epochs, thus 120 epochs were used for all further tuning. The 4-layer networks got "stuck" and would not pass accuracies of 25-35%, despite trying multiple learning rates and 200+ epochs. The grid search was performed with the 2 layer network and the best performing 3 layer network (4 neurons in the hidden layer) to determine if tuning would

Figure 5: SVM Hyptertuning Results

Hyperparameter	**Network 1**	Network 2	Network 3	Network 4	Network 5
C	10	1	1	1	10
Kernel	Polynomial	Linear	Linear	Linear	Linear
Degree	1	1	3	3	1
Gamma	Scale	Scale	Scale	Auto	Scale
Validation Accuracy	***99.52%***	***99.48%***	***99.48%***	***99.48%***	***99.48%***

Figure 6: SVM Confusion Matrix

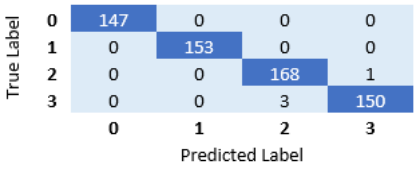

Figure 7: SVM Precision and Recall Scores

	precision	recall
0	1.00	1.00
1	1.00	1.00
2	0.98	0.99
3	0.99	0.98

improve the performance of the 3 layer network. The best performing models and their hyperparameters are summarized in [**Figure ZZ**]. Again, it was found that the 2-layer network consistently performed better than the 3-layer one. When the SGD learning function was used, there was no dependency for the model's accuracy on the learning rate once it was larger than 0.05. On the other hand, with the Adam learning function, the accuracy decreased as the learning rate increased. Additionally, the Adam learning function frequently resulted in a loss function that oscillated about an asymptote, as opposed to converging to the asymptote like the SGD function did. For both learning functions, the number of epoch required for the loss function to reach the asymptote decreased as the learning rate increased. The sigmoid activation function resulted in accuracies around 97-98%, whereas 99% accuracy was achieved with both ReLU and ELU activation functions. Overall, Network 1 showed the most promise, scoring an accuracy of 99.36% on the validation set and an accuracy of 98.71% on the test set. [**Figure XX**] displays the confusion matrix for Network 1 on the test data, while [**Figure XX**] shows the corresponding precision and recall values for each gesture. The rows are what the gesture is, whereas the columns are what the FFNN predicted. The network was capable of identifying gesture 1 with no problems. It struggled more with gestures 3 and 4. Gesture 3 was sometimes mistaken for 2 or 4, whereas gesture 4 was mistaken for all the other options in 1-2 instances.

Figure 8: FFNN Hypertuning Results

Hyperparameter	**Network 1**	Network 2	Network 3	Network 4	Network 5
Number of Hidden layers	2	2	2	2	3
Number of Neurons in hidden layers	-	-	-	-	4
Learning Function	Sgd	Adam	Sgd	Sgd	Sgd
Learning rate	0.05-0.4	0.001	0.05-0.4	0.05-0.4	0.05-0.4
Activation function	ReLU	ELU	ELU	Sigmoid	Sigmoid
Validation Accuracy	***99.36***	***99.36***	***99.20***	***98.71***	***98.87***

Figure 9: FFNN Confusion Matrix

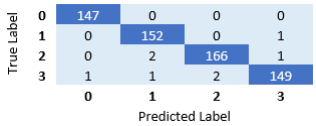

Figure 10: FFNN Precision and Recall Scores

	precision	recall
0	0.99	1.00
1	0.98	0.99
2	0.99	0.98
3	0.99	0.97

LSTM Results

After reviewing the results from the RandomSearch hyperparameter tuning process, we found that networks with 3 to 4 hidden layers were consistently preferred, although one shallow network with a single hidden layer performed substantially well. We observed no strong preference for the number of neurons per hidden layer, though it's worth noting that the highest scoring models tended to have a couple dense layers of 100+ neurons. We also found that having too harsh of a dropout policy negatively affected the models performance. Results show that a dropout rate of 18%-24% generated the highest validation scores, with scores dipping drastically after the 27% mark. Finally, we found that the ELU activation function consistently outperformed the ReLU function and was the activation function of choice for the first 8 out of 10 top models returned by RandomSearch. The best performing models and their hyperparameters are summarized in [**Figure ZZ**]. The scores reported are their validation scores. Network 1 was evaluated on the test set and scored a test accuracy of 99.36%. [**Figure XX**] shows the training accuracy and loss graphs for Network 1, which was selected from RandomSearch. [**Figure XX**] shows the confusion matrix for Network 1 on the test set.

Figure 11: LSTM Hypertuning Results

Hyperparameter	**Network 1**	Network 2	Network 3	Network 4	Network 5
Number of Hidden layers	3	1	3	4	3
Number of Neurons in hidden layers	50, 200, 200	50	100, 50, 100	100, 50, 150, 150	150, 150, 200
Activation function	ELU	ELU	ELU	ELU	ELU
Dropout rate	21%	24%	15%	21%	18%
Validation Accuracy	***99.68***	***99.67***	***99.64***	***99.61***	***99.56***

Model Comparison

[**Table XX**] summarizes the test accuracy for the three methods. It is clear that all three methods are very successful. Based on the tuning that was performed, the SVM and LSTM networks performed better than the FFNN. While care was taken to tune each network approximately the same amount, it is possible that with further tuning the FFNN could achieve the accuracy obtained by the other two methods. From observing the precision and recall tables, it can be seen that each architecture had issues with different gestures. The novel LSTM neural network mis-identified gesture 2 more than others. It had no issues with gestures 0 or 3. Conversely, the SVM and FFNN had more issues with gesture 3 than the others. This difference could be due to the feature extraction that was performed for the FFNN and the SVM. It is fortunate to know that the LSTM network, once trained, performs comparably to the standard methods. This is advantageous for real-time systems (such as video games for rehabilitation), as they would not have to take the time or computational power to compute all the features while the system is running, which could lead to faster response times.

Figure 12: LSTM Training Loss

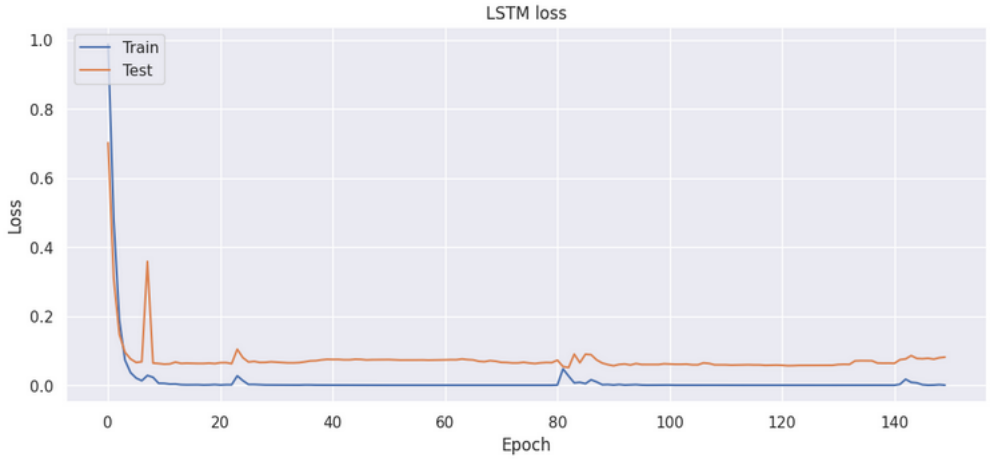

Figure 13: LSTM Training Accuracy

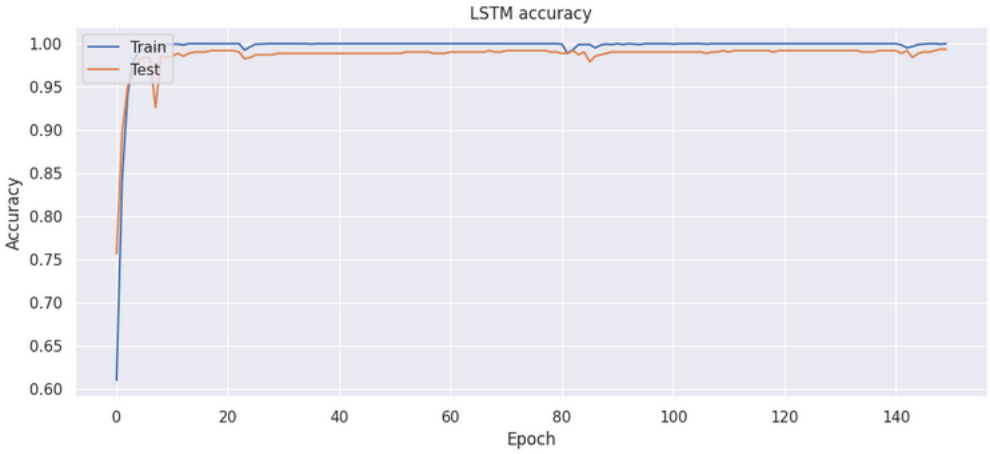

Figure 14: LSTM Confusion Matrix

Figure 15: LSTM Precision and Recall Scores

	precision	recall
0	1.00	1.00
1	1.00	0.99
2	1.00	0.98
3	0.97	1.00

Figure 16: Final Model Comparison

Method	SVM	FFNN	LSTM
Test Accuracy	***99.36***	***99.71***	***99.68***