Gesture Recognition Using EMG Data

J. Guillermo Colli-Alfaro, Claire Lizotte, Tharmiga Loganathan, and Christopher Tam

Abstract—**Cerebral Palsy (CP) is the leading cause of childhood motor disability. It is a group of permanent movement disorders which can result in impaired movement, abnormal reflexes, and muscle stiffness. Gamifying rehabilitation exercises can help children with CP stay engaged through many repetitive physical therapy sessions. An important step to building interactive software for CP therapy is classifying electromyography (EMG) muscle activity data into gestures and movements. This report presents a comparison of three gesture classification techniques applied to an open source EMG dataset with four types of hand gestures. The gesture classification techniques used include Support Vector Machines (SVM), Feed-Forward Neural Networks (FFNN) and Long-Short Term Memory (LSTM) networks. While the standard data preprocessing method using feature engineering was used for the SVM and FFNN models, a more efficient novel preprocessing approach was explored for the LSTM model. The novel preprocessing approach avoids feature engineering and uses sliding windows to feed in data as a time series. The performance of the three models was compared using prediction accuracy, precision, and recall metrics. All models achieved an accuracy ranging 98.7-99.4%. These results suggest that the preprocessing approach suggested in this report can offer a better solution to real-time classification of EMG data.**

I. Introduction

CEREBRAL Palsy (CP) is the leading cause of childhood motor disability [1]. It is a group of permanent movement disorders that affect approximately 2.3-3.6 out of every 1000 children [2]. CP is initially caused by damage to the brain during early development and symptoms typically manifest themselves in early childhood. It primarily affects muscle coordination which can result in symptoms such as impaired movement, abnormal reflexes, and muscle stiffness. While it has no known cure, a combination of physical therapy, medication and surgery can improve quality of life.

Physical therapy is considered to be one of the most important aspects of CP treatment for children as it helps develop coordination, build strength, and improve muscle control. Treatment plans are highly catered toward the needs of an individual child. Each physical therapy session aims to progress the child toward predetermined benchmarks through performing various exercises.

One of the difficulties of regular physical therapy is introducing enough novelty in the sessions to keep children engaged in their therapy. Electromyography (EMG) is being increasingly used to classify gestures and motions. As a result, it offers a promising solution to recognize human movement and create a gamified environment for patients to simulate rehabilitation exercises. EMG is a non-invasive technique that can quantify muscle electrical activity using sensors placed on the surface of the skin. EMG data can be combined with a machine learning model in order to classify human movement. Software can then be developed around this model to offer interactive exercise instruction to the patient. One potential application includes video games where the rehabilitation exercises can control outcome in the game.

The primary objective of this report will be to classify a subset of various hand gestures. Techniques such as Support Vector Machines (SVM), Feed-Forward Neural Networks (FFNN) and Long-Short Term Memory (LSTM) Neural Networks will be used for classification and the results of the models will be compared for accuracy. The content of this report is organized as follows: Section I is this current introduction. Section II presents a description of the three classification methods used in this project along with a review of the previous contributions made in EMG gesture classification. Section III presents the traditional data preparation approach, hyperparameter tuning, and implementation details for the SVM and FFNN models. Section IV presents the novel data preparation approach, hyperparameter tuning, and implementation details for the LSTM network. Section V presents and compares the results of the three classification methods, and Section VI summarizes the results of the project and extracts key implications.

II. Background and Related Work

Support Vector Machines (SVM) are a powerful yet flexible supervised machine learning algorithm that has been found to be very successful in classification problems [3]. SVM models represent the target classes in a hyperplane in multidimensional space and are very effective in high dimensional spaces, even where the number of dimensions is greater than the number of samples [4]. A hyperplane is generated iteratively using data points from the training set and is adjusted to minimize an error. The goal of the SVM is to maximize the marginal hyperplane between target classes, which can be measured by the perpendicular distance from the hyperplane to the support vectors. However, because most classification problems involve separating data that is not linearly separable, SVM uses kernel functions to project data samples into a high dimensionality space by grouping data points that have some sort of similarities between them to solve this issue [3].

The other algorithm used in this work were the Multilayer perceptron (MLP) networks. MLP networks are a type of feedforward network that consists of an input layer, one or multiple hidden layers, and an output layer. The output of each layer is given by an activation function, whose choice depends on the application. Different activation functions exist, however the commonly used one for linear problems is the sigmoid function [5], whereas for multiclass classification problems, the softmax function is employed.

A third algorithm explored consisted in Long-Short Term Memory (LSTM) networks. These networks are a type of Recurrent Neural Networks (RNN) designed to overcome the vanishing gradient inherent to RNN. LSTM accomplishes this task by incorporating memory cells in the recurrent hidden layer. These memory cells store temporal information from previous states of the network and then, by using gates known as the input gate and output gate, the flow of inputs and outputs of these memory cells is controlled [6].

Moving on, one way to assess the performance of classification problems is to use metrics such as accuracy. However the main issue with just using accuracy is that this metric is susceptible to class imbalances, i.e, it is possible to achieve high accuracies when the majority of the time the model is predicting negative values most of the time [7]. Therefore, for this project it was decided to explore other metrics such as the precision and recall scores. The former evaluates the performance of the model when evaluating the positive class, or in other words, it evaluates the model based on the number of total predicted positives. On the other hand, the recall value represents the rate of the true positives (the correctly classified samples) when compared against the samples that were incorrectly classified as a different class (true negatives).

Previous work in the field of EMG gesture recognition has tried to classify gestures using different algorithms. For example, in a study conducted by Amirabdollahian, *et al.* [8], SVM were used to detect four hand grasp gestures using the Myo Armband. They compared the efficacy of the algorithms using three different kernels and achieving accuracies of up to 94.9%. However, their presented results were based on a limited amount of training data, which may indicate that their model is not suitable for generalizing to unseen data.

In another study [9], six hand gestures were classified using MLP networks. What is interesting about this work is that each gesture was classified using three different MLP models, as each gesture was performed in three different orientations. The authors of this study showed that it was possible to achieve accuracies of up to 98.3%. Unfortunately, the time to train each network was significantly high as each network had a complex architecture. Moving on, recognition accuracies of 74.3% were reported in [10] when classifying 12 gestures using Hidden Markov Models (HMM).

Similarly, in a study performed by Zhang, *et al.* [11], recognition accuracies of up to 90.2% were reported when classifying 18 gestures used to solve a virtual Rubik's cube. The algorithm employed by Zhang, *et al.* was also based on HMM and decision trees. However, the main issue with these two studies is that in both cases, the number of testing data was low, which prevented the algorithm to generalize better to a larger population.

Other EMG gesture classification studies have tried to implement deep learning algorithms such as LSTM networks or Convolutional Neural Networks (CNN). For example, in a study conducted by Allard, *et al.* [12], CNN were used to classify seven EMG signals associated with seven gestures. The information obtained from this classification was used to control the motions of a serial link manipulator. The methods proposed by these authors suffered from a performance degradation, which caused the models to be constantly recalibrated.

Finally, in [13], an EMG-based hand gesture classification was proposed using LSTM networks. This method was proposed as an alternative to conventional EMG gesture classification algorithms such as SVM. The authors of this study showed that by using LSTM networks, it was possible to achieve low classification error rates (less than 10%) when classifying six gestures. However, the authors mentioned that the performance of their proposed model was no better than traditional classification algorithms.

It can be seen that different methods have been proposed to classify EMG signals obtained after performing different gestures. However, these methods follow a traditional pattern recognition flow. Traditional EMG pattern recognition consists of extracting data from specific muscles, then processing these data (filtering, rectification, conditioning, etc.) and then, extracting specific features that are later used to train the classification models. Usually, these features are extracted from sliding windows of less than 300 ms so that the pattern recognition can be used in real time [14].

The main inconvenience of this pattern recognition flow is that the system is required to detect when a muscle contraction is happening, which makes the system more complex. Therefore, in this work a different approach to EMG pattern recognition is proposed. First, SVM and MLP networks are used to classify EMG hand gestures. The pattern recognition flow for these two algorithms will be used as is. Then, an LSTM network will be used to classify the same hand gestures. However, when using the LSTM network, EMG features will not be extracted traditionally. Instead, the EMG signal will be segmented into sliding windows, which are then going to be fed to the LSTM networks as a time series data. The results from this approach will be compared to those obtained from the SVM and MLP networks.

III. Methodology - Established (FFNN and SVM)

A. Data Preparation

The dataset used for this project was obtained from a publicly available EMG data [15] and consisted of muscle activity readings for 8 sensors which measured the electrical activity produced by muscle movement. Each row in the dataset 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 (Fig. 1). The optimal window length for upper-limb myoelectric devices is within the range of 150-250 ms [14]. 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.

For the neural networks, the gestures were encoded in one-hot format such that a softmax activation function could be

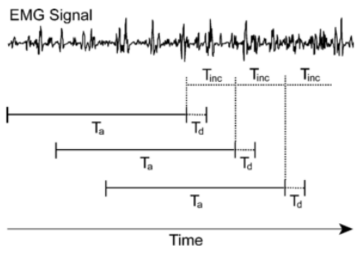


Fig. 1: EMG Signal with overlapping windows [14].

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 training and validation 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.

B. 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.

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

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

b) Waveform Length (WL): This feature states the cumulative length of the signal, as per Eq. (2), where N is the length of the signal and x(i) is the ith sample of the signal.

$$WL = \sum_{i=1}^{N-1} |(x(i+1) - x(i)| \,. \tag{2}$$

c) Zero Crossing (ZC): This feature counts the number of times the signal crosses a predefined threshold th as follows:

$$ZC = \sum_{i=1}^{N-1} f\left[x\left(i\right), x\left(i+1\right)\right], \tag{3}$$

where $f(x, y)$ is defined as:

$$f(x,y) = \begin{cases} 1 & \text{if}(x \cdot y) < 0 \cap |x - y| \geq th \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

d) 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 Eq. (5), where P is the AR order, a_p are the AR coefficients, and w_i is a white noise term.

$$x_i = \sum_{p=1}^{P} a_p x_{i-p} + w_i. \tag{5}$$

C. 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 in Table I.

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.

D. 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 (N_{HL}) were computed using Eq. (6), where D_{TS} is the number of data points in the training set, N_{IL} and N_{OL} are the number of neurons in the input and output layer, respectively, and a is a constant that was varied from 4 to 10

TABLE I: SVM Hyperparameter Tuning Space.

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

TABLE II: 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'

TABLE III: LSTM Hyperparameter Tuning Space.

Hyperparameter	Options
Number of Hidden layers	1, 2, 3, 4,
Number of Neurons in hidden layers	50, 100, 150, 200, 250
Activation function	'ReLU', 'ELU'
Dropout rate	0.15, 0.18, 0.21, 0.24, 0.27, 0.30

to prevent overfitting [16]. This resulted in using 4, 6, and 8 neurons in the hidden layers.

$$N_{HL} = \frac{D_{TS}}{a * (N_{IL} + N_{OL})}. \tag{6}$$

After the network structure was settled, the learning rate, learning function, and activation function were tuned in a grid search. Table II 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.

IV. Methodology - Novel (LSTM)

A. 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.

B. 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 in Table III.

A baseline LSTM model was instantiated with the parameters {'# Hidden layers': 1, '# Neurons per layer': 100, 'Activation function': ReLU, 'Dropout rate': 20%} which was then fit on the training data. Hold 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.

V. Results

A. 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.

$$\text{Accuracy} = \frac{\text{\# Correct predictions}}{\text{Total \# predictions}}, \tag{7}$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}, \text{ per gesture,} \tag{8}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}, \text{ per gesture.} \tag{9}$$

B. SVM Results

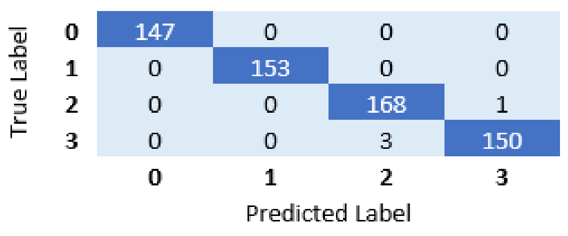
Results after iterating through the hyperparameter space showed that linear and low-order polynomial models performed the best. In general, linear kernels consistently outperformed all other kernels. Almost every single linear kernel, regardless of the state of other hyperparameters, scored an impressive 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

TABLE IV: 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%***

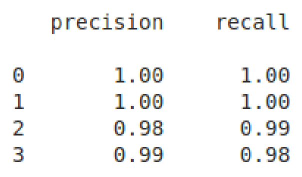
TABLE V: FNN Hyptertuning Results.

Hyperparameter	Network 1	Network 2	Network 3	Network 4	Network 5
# Hidden layers	2	2	2	2	3
# 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***



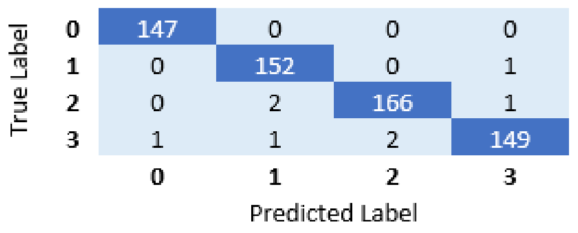
	0	1	2	3
0	147	0	0	0
1	0	153	0	0
2	0	0	168	1
3	0	0	3	150

Fig. 2: SVM Confusion Matrix



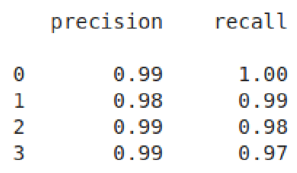
```
        precision    recall

0         1.00        1.00
1         1.00        1.00
2         0.98        0.99
3         0.99        0.98
```

Fig. 3: SVM Precision and Recall Scores



	0	1	2	3
0	147	0	0	0
1	0	152	0	1
2	0	2	166	1
3	1	1	2	149

Fig. 4: FNN Confusion Matrix



```
        precision    recall

0         0.99        1.00
1         0.98        0.99
2         0.99        0.98
3         0.99        0.97
```

Fig. 5: FNN Precision and Recall Scores

performing models and their hyperparameters are summarized in Table IV. 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%. Fig. 2 shows the confusion matrix generated the predictions made on the test set by Network 1, and Fig. 3 shows the corresponding precision and recall scores.

C. *FFNN Results*

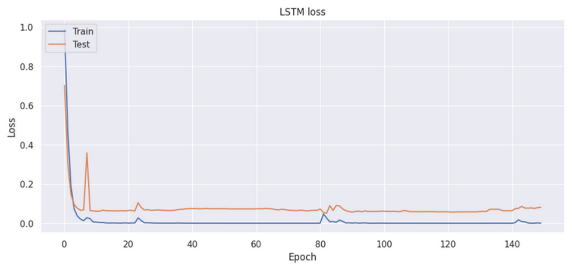
Upon investigating which network architecture performed best, the 2-layer network consistently resulted in higher accuracies than the 3- and 4-layer networks, as overfitting was beginning to occur. It was found that the loss function stabilized after approximately 30 epochs, thus 120 epochs were used 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 improve the performance of the 3 layer network. The best performing models and their hyperparameters are summarized in Table V. 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. Fig. 4 displays the confusion matrix for Network 1 on the test data, while Fig. 5 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.

D. *LSTM Results*

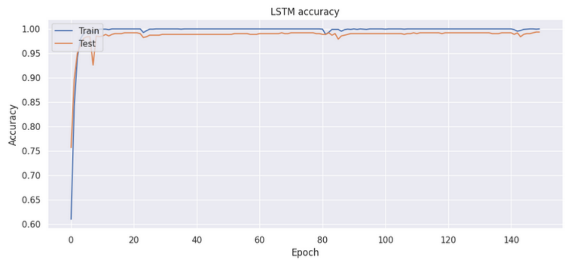
After reviewing the results from the RandomSearch hyperparameter tuning process, we found that networks with 3 to 4 hidden layers were consistently preferred, although one shallow network with a single hidden layer performed substantially well. We observed no strong preference for the number of neurons 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

TABLE VI: LSTM Hyptertuning Results.

Hyperparameter	**Network 1**	Network 2	Network 3	Network 4	Network 5
# Hidden layers	3	1	3	4	3
# 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***



(a)



(b)

Fig. 6: LSTM accuracy and training loss for Network 1. a) LSTM Training Loss. b) LSTM Training Accuracy.

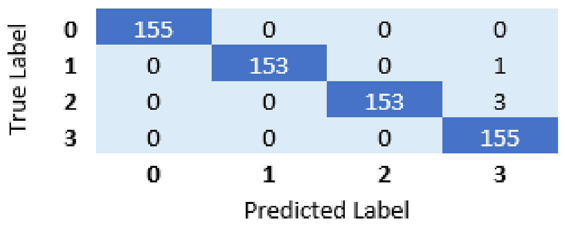
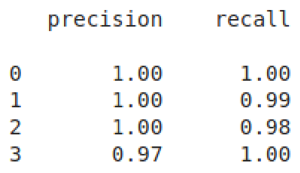


Fig. 7: LSTM Confusion Matrix



```
     precision    recall

0       1.00       1.00
1       1.00       0.99
2       1.00       0.98
3       0.97       1.00
```

Fig. 8: LSTM Precision and Recall Scores

top models returned by RandomSearch. The best performing models and their hyperparameters are summarized in Table VI. The scores reported are their validation scores. Network 1 was evaluated on the test set and scored a test accuracy of 99.36%. Fig. 6 shows the training accuracy and loss graphs for Network 1, which was selected from RandomSearch. Fig. 7 shows the confusion matrix for Network 1 on the test set, while Fig. 8 shows the corresponding precision and recall values for each gesture.

TABLE VII: Final Model Comparison.

Method	SVM	FFNN	LSTM
Test Accuracy	***99.36***	***98.71***	***99.68***

E. Model Comparison

Table VII 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.

VI. Conclusion

Gamifying rehabilitation exercises can help children with CP stay engaged through many repetitive physical therapy sessions. An important step to building interactive software for physical therapy is classifying electromyography (EMG) muscle activity data efficiently and accurately. Three different gesture classification techniques were applied to an EMG dataset and compared. While the traditional approach to preprocessing for EMG classification is computationally intensive, a faster, novel approach was also tried in this report with equivalent results.

The established method of data preparation resulted in an accuracy of 99.36% for the SVM network and 98.71% for the FFNN. The SVM model performed best with a linear kernel or a low order polynomial kernel and regularization had minimal effect. The best hyper parameters for the FFNN model include a two layer network with an sgd learning function, a Relu activation function, and a learning rate of 0.05-0.4.

When combining the novel method of data preparation with an LSTM network, an accuracy of 99.68% was achieved - matching the SVM network and performing better than the FFNN. The LSTM network tuning resulted in 3-4 hidden layers with multiple 100+ neuron layers and an elu activation function.

Thus, the novel LSTM preprocessing approach can perform comparably to the other traditional approaches. These findings suggest that an application that requires real-time classification of EMG data can benefit from using the novel LSTM approach as it is computationally more efficient

References

[1] National Center on Birth Defects and Developmental Disabilities, Centers for Disease Control and Prevention, "Data and statistics for cerebral palsy." [Online]. Available: https://www.cdc.gov/ncbddd/cp/data.html

[2] CerebralPalsy.org, "Prevalence of cerebral palsy." [Online]. Available: https://www.cerebralpalsy.org/about-cerebral-palsy/prevalence-and-incidence

[3] M. A. Hearst, S. T. Dumais, *et al.*, "Support vector machines," *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[4] F. Pedregosa, G. Varoquaux, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] B. Juba and H. S. Le, "Precision-recall versus accuracy and the role of large data sets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4039–4048.

[8] F. Amirabdollahian and M. L. Walters, "Application of support vector machines in detecting hand grasp gestures using a commercially off the shelf wireless myoelectric armband," in *IEEE International Conference on Rehabilitation Robotics*, London, UK, July 17–20, 2017, pp. 111–115.

[9] A. A. M. Lima, R. M. Araujo, *et al.*, "Classification of hand movements from EMG signals using optimized MLP," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro, Brazil, July 8–13, 2018, pp. 1–7.

[10] M. Georgi, C. Amma, and T. Schultz, "Fusion and comparison of IMU and EMG signals for wearable gesture recognition," in *International Joint Conference on Biomedical Engineering Systems and Technologies*. Lisbon, Portugal: Springer, January 12–15, 2015, pp. 308–323.

[11] X. Zhang, X. Chen, *et al.*, "A framework for hand gesture recognition based on accelerometer and EMG sensors," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 6, pp. 1064–1076, 2011.

[12] U. C. Allard, F. Nougarou, *et al.*, "A convolutional neural network for robotic arm guidance using sEMG based frequency-features," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, South Korea, October 9–14, 2016, pp. 2464–2470.

[13] M. Jabbari, R. N. Khushaba, and K. Nazarpour, "EMG-based hand gesture classification with long short-term memory deep recurrent neural networks," in *42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, Montreal, QC, Canada, July 20–24, 2020, pp. 3302–3305.

[14] K. Englehart, B. Hudgins, *et al.*, "A robust, real-time control scheme for multifunction myoelectric control," *IEEE Transactions on Biomedical Engineering*, vol. 50, no. 7, pp. 848–854, 2003.

[15] K. Yashuk, "Classify gestures by reading muscle activity." [Online]. Available: https://www.kaggle.com/kyr7plus/emg-4?select=1.csv

[16] H. B. Demuth, M. H. Beale, *et al.*, *Neural network design*. Martin Hagan, 2014.