🤖 Machine Learning | EMG | G29

Introduction & Background

What is an EMG (Electromyogram)?

- Bioelectrical signals generated by muscle cells when they are activated [1].
- Captured using electrodes placed on a skin surface.
    - In regards to our dataset, The Myo Thalmic bracelet captures signals from the forearm. It does this by utilizing 8 EMG sensors and a bluetooth module.

What is EMG Gesture Classification?

- Process in which electric signals gendered by muscle contractions are monitored inorder to identify specific gestures [1].

Literature Review

Article: “Human Hand Movement Classification based on EMG Signal using different Feature Extractor” DOI: https://dx.doi.org/10.13005/bpj/2835

- This paper focuses on applying certain feature selection techniques for classifiers in the medical sector [1].

Article: “Electromyogram-Based Classification of Hand and Finger Gestures Using Artificial Neural Networks” DOI: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8749583/

- This paper attempts to use time-domain features only to reduce the computational complexity instead of including frequency domain features. They also attempted to develop personalized classifiers for each person’s EMG data. They found that overall Artificial Neural Networks did best compared to SVMs, Random Forests, and Logistic Regression [2].

Article: “Gesture Classification in Electromyography Signals for Real-Time Prosthetic Hand Control Using a Convolutional Neural Network-Enhanced Channel Attention Model” DOI: https://www.mdpi.com/2306-5354/10/11/1324

- This paper describes the approach of using a CNN-ECA gesture recognition framework, utilizing a combination of the CNN architecture and ECA module to better enhance the focus and capture ability of important features. It also shows techniques for reducing the influence of noise and improving signal steadiness [3].

Article: “EMG-based online classification of gestures with recurrent neural networks” DOI: https://www.sciencedirect.com/science/article/pii/S0167865519302089?casa_token=S5bUDUK0LzwAAAAA:4ITvJzw3veWQvbAnLJTPRQTyMjddRZSMujJ_Lw-bqfIBOvSjje_uXd69xGUXXIplcwjLH-KNbQ

- This paper involves using RNNs for online hand gesture classification with EMG signals, avoiding motion detection calibration. In the paper they compare FFNN, RNN, LSTM, and GRU models finding that they achieve similar accuracy but that LSTM/GRU are most efficient/faster to train [4].

Dataset Description

URL: https://archive.ics.uci.edu/dataset/481/emg+data+for+gestures

Consists of gesture EMG recordings from a bluetooth MYO Thalmic bracelet. Time based data with raw EMG output, where each “datapoint” is a static hand gesture held for 3 seconds with a 3 second pause between different gestures. We can classify a total of 7 gestures. Each gesture is not uniformly represented in the dataset.

Problem Definition

Problem

Electromyography (EMG) data, captured from muscle movements, is an important source of info which allows us to develop and create systems and technology that can properly interpret hand gestures. With proper interpretation, EMG data is useful in aiding the development of prosthetics and nuanced rehabilitation techniques. However, raw EMG data remains inherently variable and quite complicated, therefore making it quite difficult to accurately read and classify hand gestures.

Motivation

Properly reading and interpreting EMG data has massive implications in physical rehabilitation and prosthetic developments [2], as it would allow us to better mimic natural and intuitive control of limbs, allowing those with amputations and disabilities greater independence and mobility in their lives. In fact, functional prosthetics can improve mental well-being and reduce muscle-atropphy [3]. In this project, we will be exploring the use of ML to enhance the accuracy and reliability of hand gesture recognition, providing a foundation for real-world applications by tackling the inherent variability of EMG data.

Methods

Data Preprocessing

So first, we began by cleaning the dataset by removing unmarked data (class 0) to ensure relevance. Features and labels were separated, and outliers were eliminated using the Z-score method with a threshold of 3 to maintain data integrity. The data was then standardized using `StandardScaler` to normalize it. To capture temporal patterns in the EMG signals, we segmented the data into overlapping windows (window size of 100 with 50% overlap). Then from each window, we extracted statistical features such as standard deviation, maximum, minimum, RMS, absolute mean, and zero crossings. Finally, we split the processed data into training and testing sets which we used for the training/classification.

```
def processAllSubs(root_dir = "EMG_data_for_gestures-master", window_size=100,
    processed_subs = {}
    for sub_id in range(1, 37):
        print(f"Doing subject {sub_id}...")
        print("======================")
        curr = fo.analyze_single_subject(sub_id)

        print("Cleaning data...")
        X, y = cleanDataAndSepFeat(curr)
        X_standardized = X
        #X_standardized = standardize(X)
        #print("Size after cleaning", X_standardized.shape)
        #print("--------------")

        print("Removing outliers...")
        X_no_outliers = removeOutliers(X_standardized, threshold=3)
        #print("Size after outliers", X_no_outliers.shape)
```

```
        #print("--------------")

        #maybe include reduce noise here

        print("Creating windows...")
        windows, window_labels = createWindows(X_no_outliers, y, window_size,
        #print("Size after windows: ", len(windows))
        #print("--------------")


        print("Extracting features...")
        X_features = extractFeaturesForWindows(windows)
        print("Size of features on subject", sub_id, ": ", X_features.shape)
        print("--------------")

        processed_subs[sub_id] = {'features': X_features, 'labels': window_lab


        print("======================")
    return processed_subs
def single_model_data(processed_subs):
    all_feats = []
    all_labels = []
    #feature_names = X_train.columns
    for sub_data in processed_subs.values():
        all_feats.append(sub_data['features'])
        all_labels.append(sub_data['labels'])
    X = pd.concat(all_feats)
    y = np.concatenate(all_labels)

    X_train, X_test, y_train, y_test = splitData(X, y)
    #print(X_train)
    feat_names = X_train.columns
    print(feat_names)

    #scale data (would scale it in processing subjects if we did LOSO)
    #X_train_scaled = StandardScaler().fit_transform(X_train)
    X_train_scaled = pd.DataFrame(StandardScaler().fit_transform(X_train), col
    X_test_scaled = pd.DataFrame(StandardScaler().fit_transform(X_test), colum
    #X_test_scaled = StandardScaler().fit_transform(X_test)
    return X_train_scaled, X_test_scaled, y_train, y_test
#processed_subjects = processAllsubs()
```

ML Methods Implemented

---------- Random Forest ----------

```
def train_rf_classifier(X_train, y_train):

    rf_classifier = RandomForestClassifier(
        n_estimators=1000,
        random_state=42,
        max_depth=None,
        min_samples_split=20,
        min_samples_leaf=10
    )

    rf_classifier.fit(X_train, y_train)
```

```
    return rf_classifier
```

For our first model, we chose to use a **Random Forest** classifier for gesture classification. This is an ensemble method and was (in part) chosen because of its robustness against overfitting, ability to handle high-dimensional feature spaces (like EMG data), and effectiveness in capturing complex patterns within the EMG data. The model was configured with 1000 decision trees, no maximum depth, and specified minimum samples for splits and leaves to enhance generalization. Another thing to note is that a Random Forest also provides feature importance metrics, allowing us to identify and (potentially) prioritize the most influential features in the classification process. We think all of these characteristics make it generally well-suited for classifying gestures from EMG signals.

---------- GMM ----------

```
def train_supervised_gmms(X_train, y_train, n_components_per_class=1):
    classes = np.unique(y_train)
    gmm_models = {}
    priors = {}

    for cls in classes:
        X_cls = X_train[y_train == cls]

        gmm = GaussianMixture(n_components=n_components_per_class, covariance_
        gmm.fit(X_cls)

        gmm_models[cls] = gmm
        priors[cls] = len(X_cls) / len(y_train)

    return gmm_models, priors
```

Our code for GMM trains a separate Gaussian Mixture Model for each class to model its feature distribution. Each GMM uses n_components_per_class mixture components, allowing for flexible modeling of multimodal distributions, with covariance_type='full' enabling detailed covariance fitting. Class priors are calculated from the training data. This approach is well-suited for probabilistic classification tasks like gesture recognition, becuase it is pretty good at capturing class-specific patterns and handles data variance generally well.

---------- Neural Network ----------

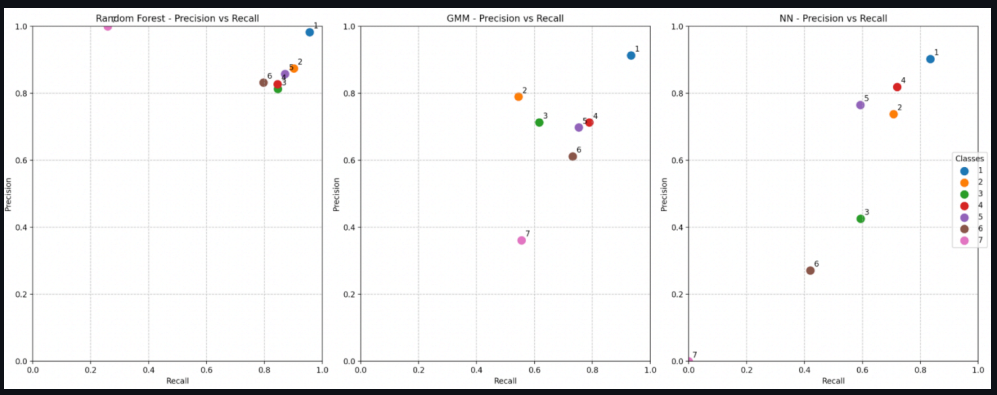
```
def train_nn_classifier(X_train, y_train, input_dim, num_classes, my_epochs=20
    model = Sequential([
        Dense(128, input_dim=input_dim, activation='relu'),
        Dropout(0.5),
        Dense(64, activation='relu'),
        Dropout(0.3),
        Dense(num_classes, activation='softmax')
    ])

    optimizer = Adam(learning_rate=0.001)  # Lower learning rate
    model.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy',


    history = model.fit(X_train, y_train, epochs=my_epochs, batch_size=32, ver
    return model, history
```

Our code for NN defines a feedforward architecture with three fully connected layers. The first layer has 128 neurons with ReLU activation and a 50% dropout rate, while the second layer reduces complexity with

64 neurons and a 30% dropout rate. The final layer employs a softmax activation for multi-class classification. The model is optimized using the Adam optimizer with a 0.001 learning rate and trained with a sparse_categorical_crossentropy loss function for 20 epochs with a batch size of 32. This neural network model was selected because neural nets are good at handling non-linear or complicated data, like EMG signals.

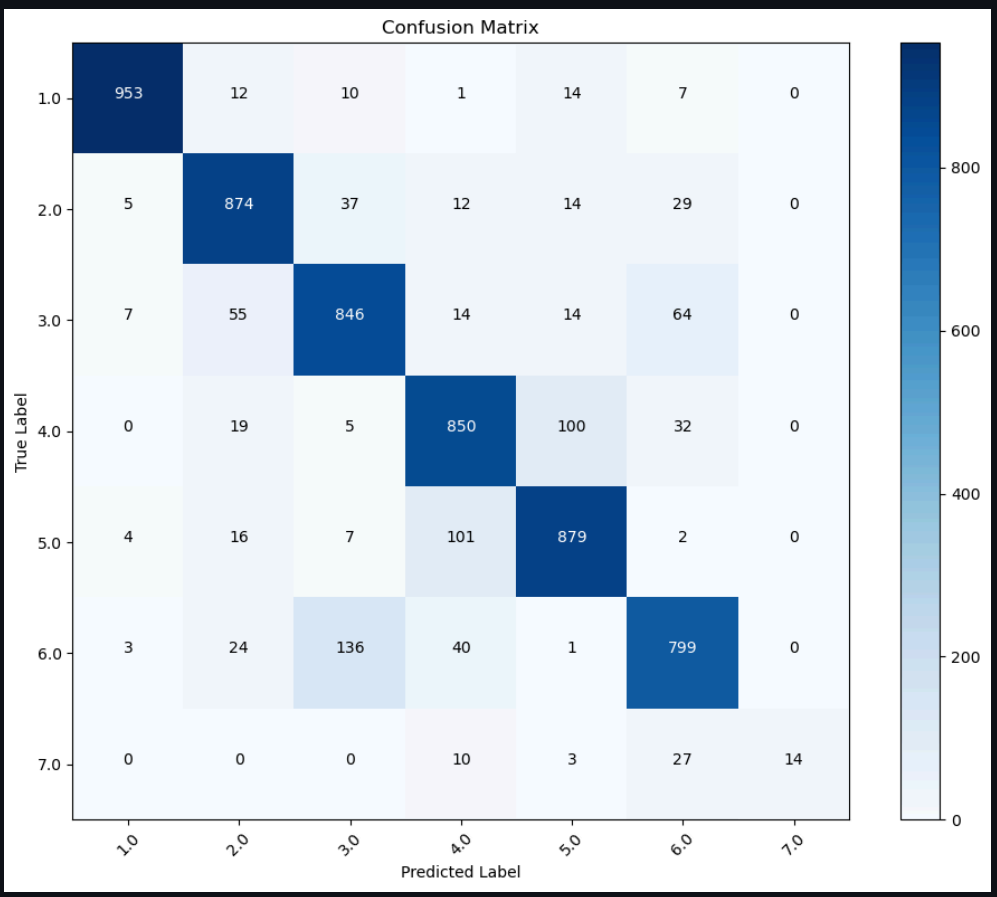
Results & Discussion

Visualizations (Plot & Confusion Matrices)



Above is the plotted Precision vs. Recall for the three models we employ. It gives a sense of how the classes (1 - 7) compare in Precision vs. Recall for Random Forest, GMM, and NN.

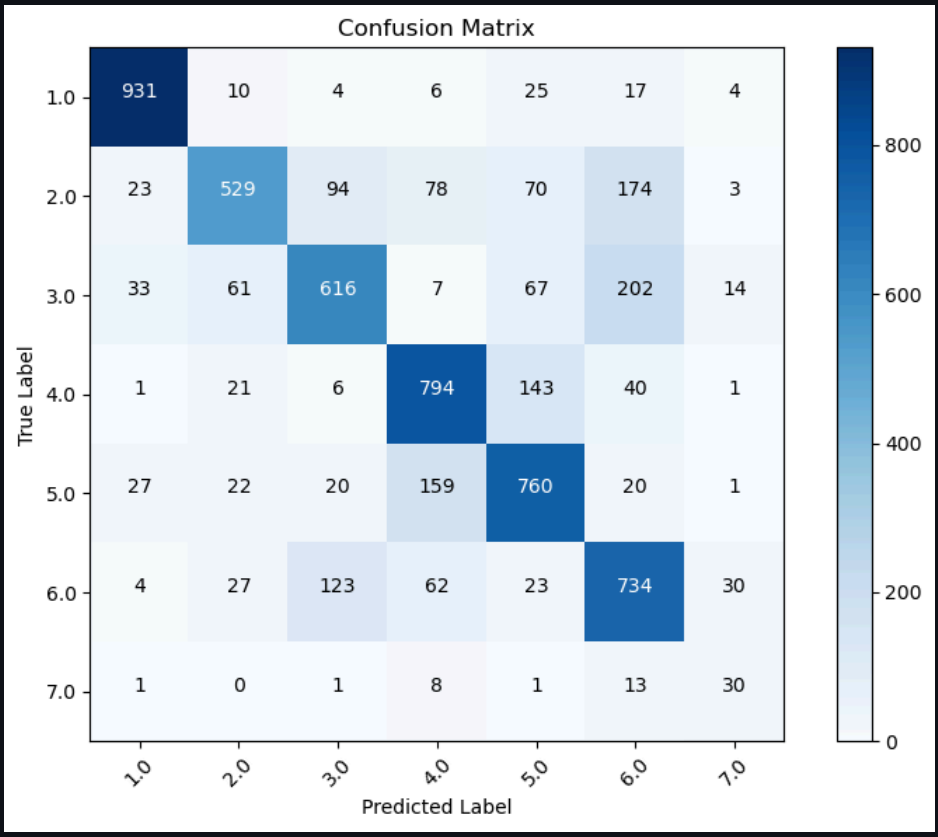
---------- Random Forest ----------



We can see that the diagonal dominance in the confusion matrix above indicates that the model was able to correctly classify most samples in each seperate class. We are also able to see how the model doesn't

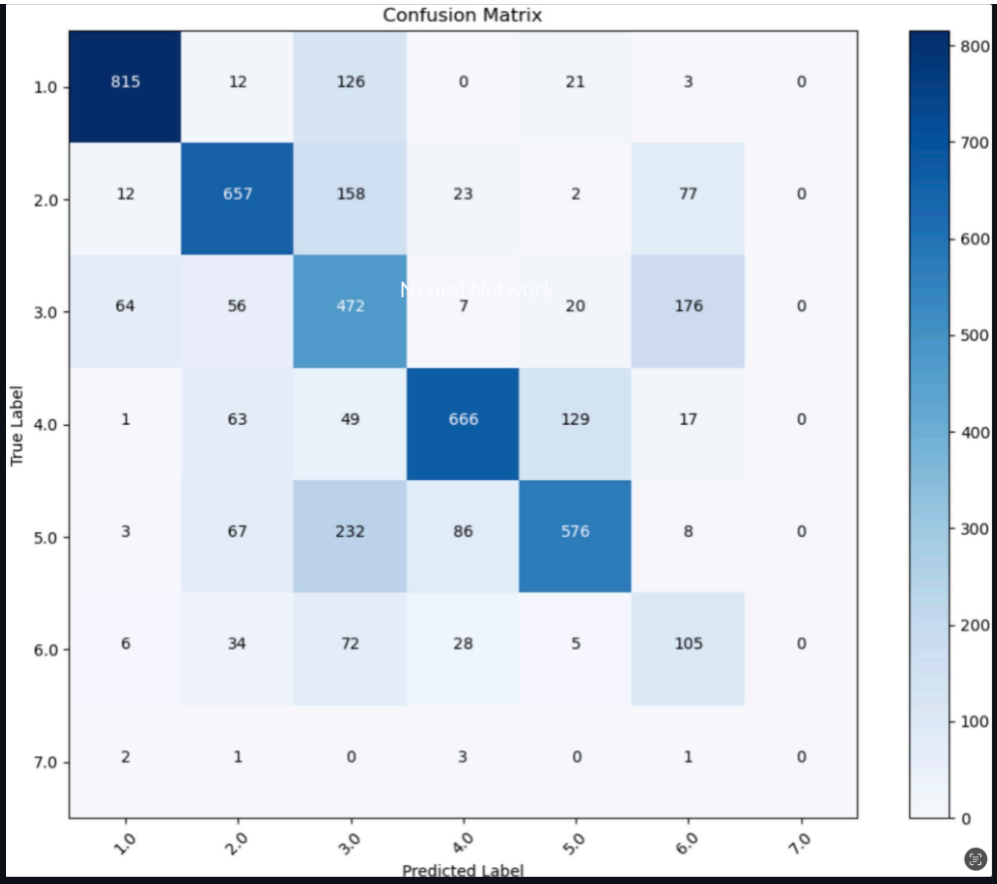
perform as well with classes 3 and 6 (high values not on the diagonal for those class intersections), which could suggest some class imbalance. It is also more apparent because it seems like class 7 did not have nearly as many samples as the other classes, further signifying a potential class imbalance. To fix this, one could resample the data or use synthetic data generation, which would improve the model's performance on the classes with less samples.

---------- GMM ----------



In this confusion matrix, we can notice substantially less diagonal dominance than in the one above for Random Forest, indicating that the model struggled slightly more in classification. Similarly, there is a pervasive struggle with class 7 (likely due to a lack of data).

---------- Neural Network ----------



This confusion matrix is the least diagonal, although it is still clearly diagonal and demonstrates a general ability at classification. This is related to how, despite our expectations, the Neural Network model struggled more with classification than the other techniques. Again there is a struggle with class 7.

Quantitative Metrics (Accuracy, Precision, Recall)

---------- Random Forest ----------

- **Accuracy:** With the model achieving an accuracy of ~86%, we can see that it is able to successfully classify a significant majority of our data, supporting the fact that the random forest classifier performed relatively well overall. However, it is important to note that this accuracy, while good, doesn't account for the misclassification jumps in specific classes that could be significant in specific use cases.
- **Precision:** Our precision gives us an indicator of the proportion of correctly predicted gestures out of all gestures that the model predicted for each class. Overall, we can see that we obtained a weighted average precision of 0.87, which is fairly good. This precision does vary by class however, with a minimum of 0.81 and a maximum of 1.0.
- **Recall:** The recall score helps us understand the proportion of actual gestures that were correctly predicted by the model. We were able to obtain a weighted average recall of 0.86, which is also fairly good. Unfortunately, this varies all the way down to a recall of 0.26 for class 7.0.

With an F1 score of ~0.86, we are able to see that the model is able to act precisely with decent recall. The high precision shown in most classes compared to a slightly lower recall signify that the model is good at picking up negatives (e.g doesn't classify many false positives), but it is missing some true positives at times as well.

---------- GMM ----------

- **Accuracy:** This model achieved an accuracy of ~73%.
- **Precision:** This model achieved a weighted average precision of 0.74.

- **Recall:** This model achieved a weighted average recall of 0.73.

With an F1 score of ~0.73, we are able to see that the model is able to act precisely with decent recall. The balance between precision and recall indicates moderate performance but struggles with finer class distinctions.

---------- Neural Network ----------

- **Accuracy:** This model achieved an accuracy of ~66%.
- **Precision:** This model achieved a weighted average precision of 0.67.
- **Recall:** This model achieved a weighted average recall of 0.66.

With an F1 score of ~0.66, we are able to see that the model is able to act with some precision and recall, although it struggles more than the other models with generally lower scores. The similar precision and recall suggest consistent but limited effectiveness, likely due to feature or model limitations.

Analysis of Algorithm: Random Forest

For making classifications, we use the **random forest** method with scikit-learn, which is an ensemble method of multiple decision trees, combining their results to make more accurate classifications and control overfitting.

A **decision tree** is a structure where classification flows from the root to a leaf node, where each intermediary node represents a decision based on a feature, each branch represents the subsequent outcome and choices based on the parent decision made, and each leaf node represents a final classification. When splitting the data based on a feature, the decision tree uses the metric of **Gini Impurity** to make each subset of data as pure as possible and improve classification accuracy (by reducing uncertainty about what a classification should be based on feature information).

When D is the dataset, C is the number of classes, and p_i is the probability that a randomly selected point in D belongs to class i, Gini Impurity can be calculated as follows:

$$Gini(D) = 1 - \sum_{i=1}^{C} (p_i)^2$$

When using the random forest method, we randomly select subsets of the training data to create diverse datasets, and for each of these subsets we build a decision tree by finding the best feature to split the data at each node. For the actual broader classification, each tree votes for a class assignment and the majority becomes the final prediction. T

When $\hat{y}$ is the prediction in a Random Forest, and DT_i is the classification made by the i-th Decision Tree, this voting can be formalized with n many trees as follows:

$$\hat{y} = mode\{DT_1(x), DT_2(x), \ldots, DT_n(x)\}$$

This technique is a solid fit for our problem space, because it is relatively good at robustly handling datasets that are large with high dimensionality (such as our EMG dataset) and the use of multiple decision trees "voting" can help cancel out imperfections caused by signal noise, which is difficult to fully remove from sensor data datasets, such as the dataset we use. Random forests are also convenient in that they lend themselves to being more interpretable than some other methods, since you can see how a classification was made in the decision trees that voted and can directly observe and visualize what features were important for that classification.

For our implementation, we use a Random Forest of 1000 decision trees, with a starting seed of 42, no max depth (allowing for more purity), a minimum samples required to split of 20 (reducing overfitting for very small sample numbers), and a minimum number of samples in a leaf node of 10. Our model has decently solid performance, with an accuracy score of about 0.8634. Additionally, there is a decent balance between precision and recall, with an f1 score of about 0.8621. However as discussed in the

"Quantitative Metrics" section, there is still room to improve. The most important features were derived from channel 7 of the signal data.

Analysis of Algorithm: GMM

GMMs can be well-suited for the gesture recognition problem, because their probabilistic nature helps for handling noisy, overlapping, and multimodal data, as is the case with our EMG signal data. Our implementation of the GMM is essentially a supervised GMM model, in which seperate GMMs are trained for each class. This enables the model to learn the unique charecteristics and distributions of each respective class, ensuring that class labels guide the training process rather than unsupervised clustering.

We track the model's performance using accuracy, f1 score, and a confusion matrix. The accuracy score is 73%, which indicates pretty decent model performance. The weighted average f1 score of .73 aligns with the accuracy, representing decent consistency across classes.

We can analyze the precision and recall numbers to see some aspects of the model that could be improved. For example, class 1 achieves high precision and recall, but class 7 (which is smaller) achieves a much lower score. This signifies that the model is likely experiencing some class imbalance. The ECG data is also likely too complex for a GMM implementation that assumes a gaussian distribution.

Analysis of Algorithm: Neural Network

Neural Networks are good for gesture recognition as a problem because they are good (and decently robust) at handling complex, nonlinear, and high-dimensional data. Our neural network is a simple model for classification with two hidden layers using ReLU activation. It includes 50% and 30% dropout to prevent overfitting and trains with a learning rate of 0.001 using the Adam optimizer. We also track its performance with accuracy, F1-score, and a confusion matrix, plus plot its loss and accuracy over the training process.

The accuracy score is 65.8%, which indicates moderate performance on the test data. The weighted average F1-score of 0.66 aligns with the accuracy, showing consistency accross classes.

We can see that the loss goes down in the first couple epochs then starts increasing, this issue might be that our model is too complex and that potential issues like learning rate, overfitting, or exploding gradients might exist.

Comparison

Each model comes with its own respective strengths and limtations, and the tradeoff between these and what you decide to prioritize can depend problem space. Accordingly, we tried to pick models that might be beneficial for this sort of problem. Comparing:

- Random Forests are generally simple and effective for small datasets but might struggle with modeling more complex patterns compared to some other models. They also seem to be relatively scalable, and the least computationally expensive, (even if they might not be as good as certain models with certain kinds of complexity). Overall relatively strong model. This was a good fit for us, also because the use of multiple decision trees "voting" can help cancel out imperfections caused by signal noise.
- GMMs are intuitive and probabilistic but rely on assumptions (Gaussian) and struggle with high-dimensionality. Depending on how they are applied, GMMs are probably in between NNs and Random Forests in terms of computational complexity (due to reliance on probabilistic modeling and parameter initialization).
- Neural Networks offer really strong flexibility and power but require significant data, tuning, and computational resources. Neural Networks are used in a ton of widely successful applications, but they seem to require more scale to be done correctly. They are the most, generally, computationally

expensive method we tried. Because our dataset was relatively not that large, our model quickly overfit to the training set and did poorly on our test set.

Next Steps

Moving forward, we aim to both refine our methods for data preprocessing and attempt to improve our classification accuracy by attempting a variety of strategies. For instance, we are considering:

- Hyperparameter tuning
- Data augmentation (adding noise or time-warping for robustness)
- Feature engineering
- Try one vs rest classification

Finally, we would like to test all three of our models with variable real-world data (if we get the chance), by using our own sensors to test how well the models respond to the change that comes from a new data source.

References

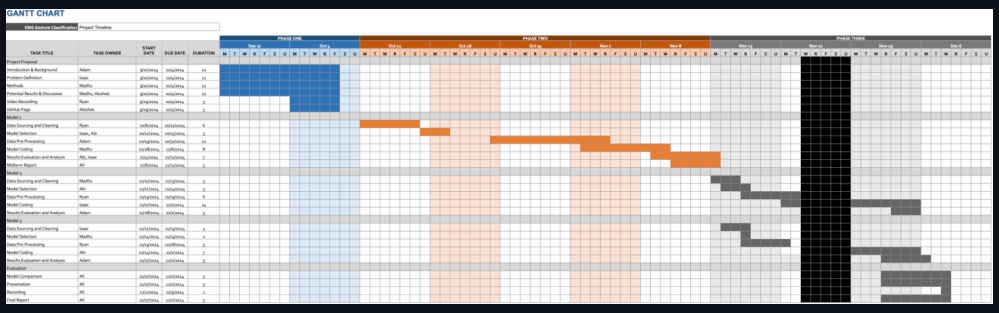
[1]Swati Shilaskar, Shripad Bhatlawande, Ranveer Chavare, Aditya Ingale, R. Joshi, and Aditya Vaishale, “Human Hand Movement Classification based on EMG Signal using different Feature Extractor,” Biomedical and Pharmacology Journal, vol. 17, no. 1, pp. 71–82, Mar. 2024, Available: https://biomedpharmajournal.org/vol17no1/human-hand-movement-classification-based-on-emg-signal-using-different-feature-extractor/

[2]K. H. Lee, J. Y. Min, and S. Byun, “Electromyogram-Based Classification of Hand and Finger Gestures Using Artificial Neural Networks,” Sensors, vol. 22, no. 1, p. 225, Dec. 2021, doi: https://doi.org/10.3390/s22010225.

[3]G. Yu, Z. Deng, Z. Bao, Y. Zhang, and B. He, “Gesture Classification in Electromyography Signals for Real-Time Prosthetic Hand Control Using a Convolutional Neural Network-Enhanced Channel Attention Model,” Bioengineering, vol. 10, no. 11, pp. 1324–1324, Nov. 2023, doi: https://doi.org/10.3390/bioengineering10111324.

[4]M. Simão, P. Neto, and O. Gibaru, “EMG-based online classification of gestures with recurrent neural networks,” Pattern Recognition Letters, vol. 128, pp. 45–51, Dec. 2019, doi: https://doi.org/10.1016/j.patrec.2019.07.021.

Gantt Chart



Contributions

Person	Contributions
Madhu	Contribution Table, Final Video, Slides
Abi	NN implementation, Model eval
Ryan	Updating slides, Streamlit website
Isaac	GMM implementation, Model eval
Adam	Gantt Chart, Slides