**README for Improved Classification Control for Multi-DOF EMG-Controlled Exoskeleton**

**Introduction**

This project aimed to improve classification control over the MyoPro exoskeleton movements in multiple degrees of freedom facilitated by a wrist add-on extension. The facilitated exoskeleton movements or states currently implemented are defined as follows:

* 0 – Rest
* 1 – Grasp
* 2 – Open
* 3 – Pronate
* 4 – Supinate

**General Procedure**

Prerecorded training datasets for hand gestures were used. The data was first preprocessed, and features were categorized based on the corresponding Kinematic position. Then the preprocessed features (X inputs) and states (Y predictions) were then used to train two models – a convolutional neural network (CNN) and a K-nearest neighbor classification model. The state predictions are to be used to control the MyoPro exoskeleton and Wrist add-on kinematic positions in real time.

**Datasets**

All offline analyses were conducted on the following datasets on the server:

* D:\SmartHome\PvNP\_Wrist\_Forearm\S1\_P\TaskData\_20230308-174844.kdf
* D:\SmartHome\PvNP\_Wrist\_Forearm\S1\_NP\TaskData\_20230308-171058.kdf
* D:\SmartHome\PvNP\_Wrist\_Forearm\S2\_P\TaskData\_20230310-160842.kdf
* D:\SmartHome\PvNP\_Wrist\_Forearm\S2\_NP\TaskData\_20230310-152716.kdf
* D:\SmartHome\PvNP\_Wrist\_Forearm\S3\_P\TaskData\_20230313-112317.kdf
* D:\SmartHome\PvNP\_Wrist\_Forearm\S3\_NP\TaskData\_20230313-105333.kdf

**File Descriptions**

**Model Training:**

* **preprocessData.m**: Function – Aligns features and kinematics, defines states, filters unclassified data points, selects relevant feature channels, and returns features and states ready for model training.
* **trainAgnosticContinuousCNN.m**: Function – Adjusts Connor’s CNN model with padding for continuous state prediction.
* **knnClassifier.m**: Function – Trains a KNN model based on input preprocessed features and states.

**Offline Analysis:**

* **modelComparison.m**: Performs a statistical comparison of KNN vs. CNN across datasets, evaluating the accuracies, prediction times, ROC curves, and aggregate confusion matrices.
* **visualizeKinematics.m**: Visualizes the features, states, and corresponding exoskeleton hand and wrist kinematic positions.
* **confidenceTesting.m**: Visualizes and evaluates the impact of confidence level-based state transitions.
* **cnnVisualization.m**: Visualizes CNN convolutional layers through feature maps.
* **knnVisualization.m**: Visualizes the KNN classification decision boundaries.

**Feedback Decode:**

The following functions are located in FeedbackDecode’s dependencies folder and currently function to be able to run the classification models in real time.

* **trainKNN.m**: Function – KNN model after preprocessing input kinematics and features.
* **trainCNN.m**: Function – Trains a CNN model after preprocessing input kinematics and features.
* **testKNN.m**: Function – Returns predicted kinematics using input features and trained KNN model.

**Findings**

The KNN model had statistically higher accuracies and lower prediction times across the 6 datasets.

**Next Steps**

* Implement control in additional DOFs, e.g., wrist flexion
* Implement Add-On Class in Feedback Decode
* Evaluate wrist add-on and control algorithm through online clothespin relocation test
  + Cognitive load
  + Compensation angles