Enhanced sEMG Signals Process with Mario CNN on Gesture Classification

ESE 546 Fall 2021 Final Project

Team 29: Hao Dong, Yuxin Wang, Zhengjia Mao

Abstract: sEMG signals show huge potential in implementing control systems of mechatronics devices, and small muscle movements can generate enough sEMG signals to achieve desired control operations. Traditional methods on the sEMG signals process do not robustly decipher important information to distinguish subtle differences in gesture classification. This paper applied a novel deep learning method, a two-stream CNN architecture called Mario CNN, to process NinaPro DB1 data on gesture classification and achieved higher accuracy than a single stream CNN.

Keywords: gesture recognition; convolution neural network; surface electromyographic

Introduction

Surface Electromyography (sEMG) measures the electrical activity generated by muscle impulses, as shown in Figure 1. The generated potential difference usually ranges from -90mV to 30mV, but the amplitude patch electrodes can only measure about ±5 mV from the skin. sEMG is widely used in implementing control systems of mechatronics devices, and more specifically, small muscle movements can generate enough signals to achieve desired control operations. According to the literature, sEMG is often mixed with very low frequency (near DC) and high-frequency interference signals, while the effective sEMG signal spectrum is distributed between 10-500Hz[1]. Therefore, the signal detected from the patch electrode needs to go through a signal conditioning process such as high-pass filtering (straightening process), high-pass amplification, and low-pass filtering.

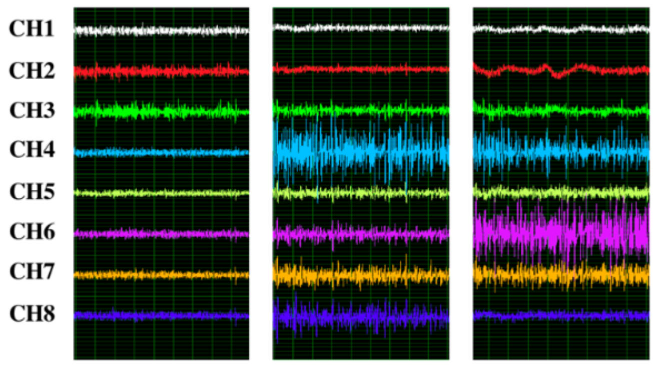


Figure 1: sEMG signals example for three gestures (one for each column)[2]

While sEMG signals show huge potentials, the applications are relatively immature. Current problems include monotonous signal sources, ambiguous classification, and noise under non-ideal conditions. The measurements can be easily affected by many factors like electrode shifts, limb position, muscle fatigue. Meanwhile, inconsistent labels may have similar signals but with different amplitudes, and traditional machine learning classification methods do not robustly decipher important information. Therefore, researchers and scientists in recent years (especially since 2014) began to apply deep learning methods to process sEMG signals on applications, including hand gesture recognition, speech and emotion classification, and sleep stage classification. This paper studied the networks from related papers and applied a novel "Mario Brothers" neural network (Mario CNN) to process sEMG signals on gesture recognition. The name is inspired by Mario and Luigi, who run towards the same goal in a parallel manner, as shown in Figure 2.

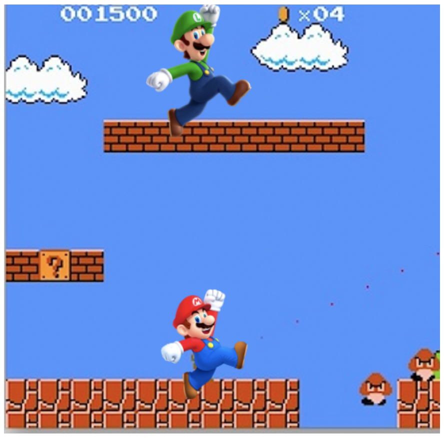


Figure 2: Mario and Luigi run in a parallel manner

Related Works

Five main deep learning architectures are commonly used in sEMG signal recognition: deep feedforward neural networks (DNN), recurrent neural networks (RNN), convolutional neural networks (CNN), autoencoders (AEs), and deep belief networks (DBN)[3]. Some of the related papers we read are summarized below.

"Deep Learning with Convolutional Neural Networks Applied to Electromyography Data: A Resource for the Classification of Movements for Prosthetic Hands "[4]
This paper implemented a modified version of a CNN (LeNet; LeCun et al., 1995) on the NinaPro dataset and achieved an accuracy of 66.59±6.40% on dataset 1. The input data was obtained with 150 ms time-windows. The architecture consisted of 4 blocks, which contained the traditional convolutional layer, rectified linear unit(non-linear activation function), average pooling layer, and the softmax loss. They train the model with SGD and use changed learning rate, normalization, and data augmentation to improve the accuracy. The results show that the classification accuracy obtained with convolutional neural networks is higher than the average results obtained with the classical classification methods.

"Classification of 41 Hand and Wrist Movements via Surface Electromyogram Using Deep Neural Network"[5]
This paper implemented DNN on NinaPro-DB5/DB7 and achieved accuracy of 93.87±1.49 %. The stride between each window was set to be smaller than the window size so as to gain better performance. The noise threshold T was set to 0.01V for DB5. The architecture of Deep Neural Network Classifier was 3 hidden layers followed by ReLU and then softmax, which included Adam optimizer, with a learning rate of 0.005 and decay of 0.00001. The authors adopted batch normalization and 20 % dropout and used a trick called Evaluation metrics in the end. Due to half rest classes in the datasets, the accuracy needed to

be balanced. For multi-class classification, taking the average of recall values can be generalized as the macro recall.

“sEMG-Based Gesture Recognition with Convolutional Neural Networks”[6]

This paper implemented a parallel multiple-scale CNN on NinaPro-DB2 and achieved accuracy better than a single CNN. The sEMG signals from 12 electrodes were converted into the sEMG images of size 12×200 (100ms, 2000 Hz). It selected some appropriate features representing for sEMG which can reduce the input dimension of the classifier. The architecture of the CNN is special, which was composed of two blocks. In Block 1, five convolution layers and two maximum pooling layers were employed. On the other hand, Block 2 was different in the first three convolution layers which adopted the bigger filter kernel size. The pooling layers were not adopted in Block 2. Those two blocks were parallel and did not influence one another when extracting features. The outputs of the two blocks are concatenated and then delivered to the classifier. The results show that a larger kernel filter can cause a slight increase in classification accuracy. Moreover, the combination of different sizes of kernel filters in the parallel blocks can yield better performance than single size.

“A multi-stream convolutional neural network for sEMG-based gesture recognition in muscle-computer interface”[7]

This paper constructed a multi-stream convolutional neural network to train the NinaPro dataset and achieved an accuracy of 85.0%. The authors used the divide-and-conquer classification approach on feature-space, which considered sEMG signals in each channel independently and trained them using the same CNN architecture. In the divide stage: the convolutional neural network was constructed by two convolutional layers (3*3 kernel size) and two locally-connected layers (64 non-overlapping 2D filters) along with batch normalization, Relu, and dropout. The network was trained with SGD and tested with cross-validation. In the conquer stage: all feature maps learned from each CNN were fused into a unified feature map and sent into fusion networks. Relu followed by batch normalization were applied after each fully connected layer. The results show that considering sEMG signals recorded by each channel independently is better than considering sEMG signals recorded by all channels as a whole.

Methodology

The models in the above papers are implemented on different datasets with different experiment objectives and protocols, so it may not be appropriate to compare performances directly. Therefore, we firstly built a simple 5-layer CNN partially based on the ideas and architecture presented in the published paper by Atzori, M et al[4] as a reference. Based on the code by malele4th on GitHub[9], we modified the hyperparameters and layers, and the architecture is shown below in Figure 4.

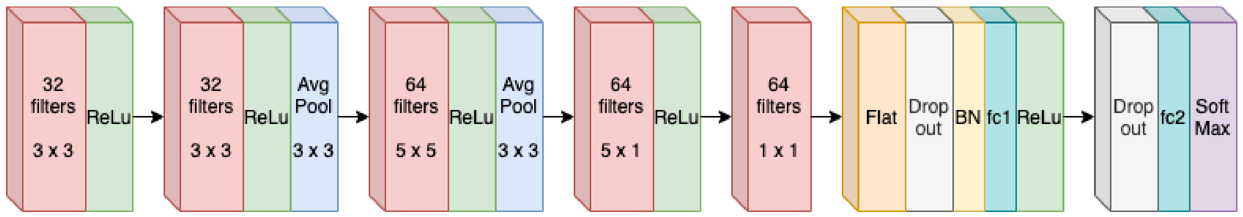


Figure 4: 5-layer CNN Architecture partially based on the tuning and layers from Atzori, M. et al[4]

The architecture consists of 7 blocks: blocks 1-5 are convolutional layers including non-linear activation function and average pooling. Convolutional layers apply a determined number of filters to extract features from the original images resulting in feature maps. Pooling layers build up "local translational invariance" by summarizing the features from the patches of the feature map. Block 6-7 contains two fully connected layers along with Batch Normalization and Dropout. Batch Normalization stabilizes the learning process and standardizes the input distribution so that it reduces overfitting. Dropout also reduces overfitting and improves the model generalization. A softmax classifier in the end uses the cross-entropy loss to calculate the probabilities used for classification.

Then we built a novel two-stream architecture, named Mario CNN, inspired by the idea from Ding, Z. et al[6]. The network designed by Ding, Z. et al has two parallel streams, where the input is splitted, fed into the parallel streams, concatenated after 5 convolutional blocks, and processed together through fully-connected layers in the end. Mario CNN was tuned based on the hyperparameters from the single-stream CNN in Figure 4. After a few iterations of hyperparameters and layers tuning, the optimal Mario CNN architecture is shown below in Figure 5.

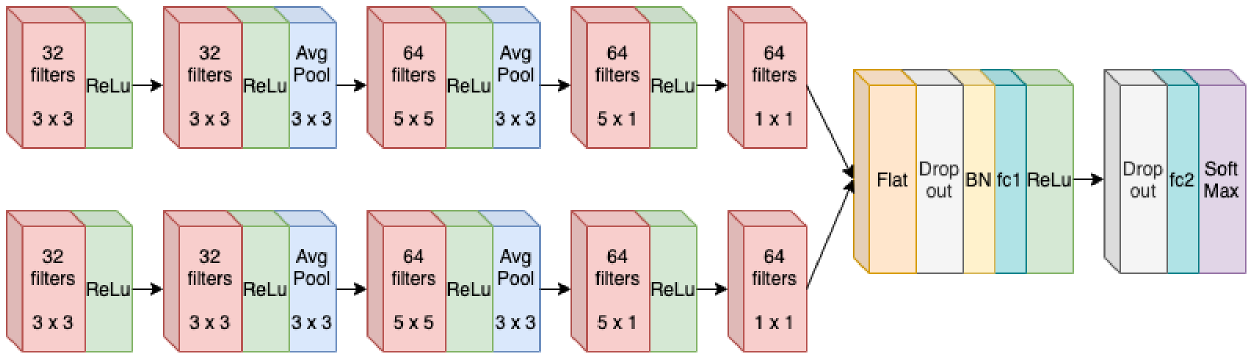


Figure 5: two-stream Mario CNN Architecture

Compared with our first neural network, two-stream Mario CNN architecture has two stages. During the decomposition stage: we first randomly and evenly split our data into two sets and train them using the same convolutional blocks. During the concatenation stage: we concatenated these feature maps into one unified feature map.

Based on the idea presented in the paper[7], as shown in Figure 6, we decided to experiment with architectures with more streams, called multi-stream Mario CNN, and investigate how the number of streams affect the performances. To make the dimensions work, the hyperparameters for the pooling layers are slightly tuned while keeping the other layers the same as they are in the 2-stream Mario CNN architecture.

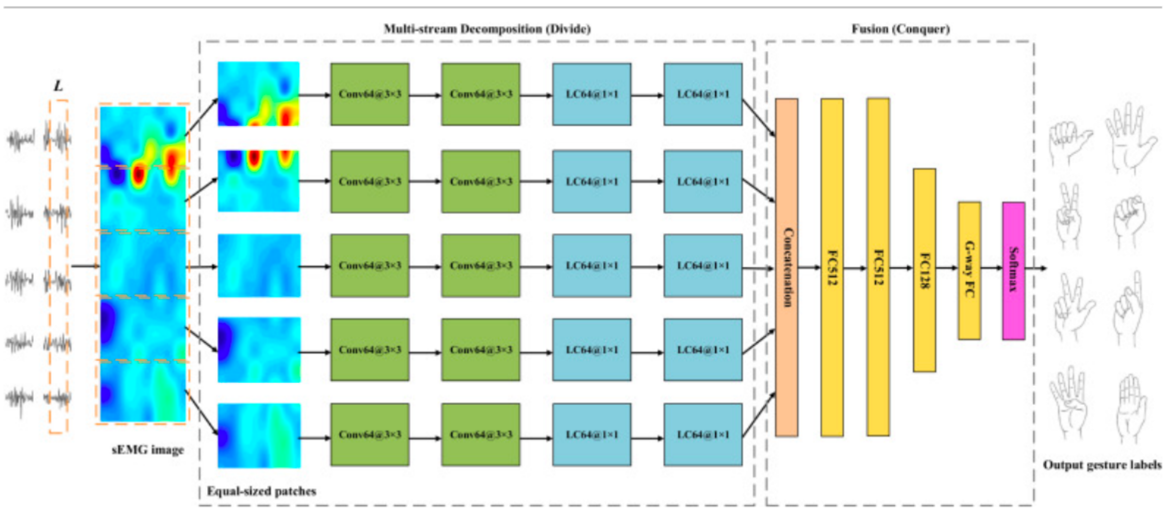


Figure 6: Multi-stream CNN Architecture[7]

In this project, the designed architectures mentioned are implemented on NinaPro-DB1-S1 dataset[8]. NinaPro is a public database for research on hand gesture recognition, and the preprocessed datasets provided, DB1-DB9, have different acquisition protocols and settings. For DB1, 27 experimenters were sampled, and each experimenter repeated 10 trials of 52 gestures. The 52 gestures included three different gesture groups: Basic finger movements(E1), hand and wrist movements(E2), functional grasping movements(E3). The sEMG data was collected at a sampling rate of 100 Hz with 10 located electrodes placed on upper forearms. The hand poses were recorded by 22 sensors of CyberGlove II.

Taking a careful look at our DB1-S1 data folder, there are two important datasets in each hands movement folder: 'emg' and 'stimulus'. We take 'emg' as our training image, for 'emg' in E1, it has a shape of (101014, 10), where each of the 10 channels generates 101014 data during the processing of collecting the hand gesture data. And we take 'stimulus' as our training label, for 'stimulus' in E1, it has a shape of (101014,1), which contains 18 labels representing a rest stage and 12 basic finger movements. Each movement is around 5 seconds (500 data points).The details of our DB1-S1 data are shown in the following Table.

Table #1 The details of DB1-S1 data

	E1(12 basic finger movements)	E2(17 hand and wrist movements)	E3(functional grasping movements)
emg.shape	`(101014, 10)`	`(142976, 10)`	`(227493, 10)`
stimulus.shape	`(101014, 1)`	`(142976, 1)`	`(227493, 1)`
labels and corresponding data numbers	`rest stage: 0: 39063 12 movements: 1: 5149`	`rest stage: 0: 55113 17 movements: 1: 5165`	`rest stage: 0: 108449 23 movements: 1: 5162`

```
2: 5174          2: 5176          2: 5188
3: 5158          3: 5178          3: 5133
4: 5173          4: 5169          4: 5174
5: 5173          5: 5166          5: 5153
6: 5170          6: 5167          6: 5132
7: 5171          7: 5170          7: 5177
8: 5172          8: 5177          8: 5182
9: 5135          9: 5167          9: 5190
10: 5137         10: 5158         10: 5182
11: 5166         11: 5166         11: 5182
12: 5173         12: 5170         12: 5189
                 13: 5174         13: 5186
                 14: 5170         14: 5191
                 15: 5173         15: 5161
                 16: 5155         16: 5165
                 17: 5162         17: 5184
                                  18: 5210
                                  19: 5189
                                  20: 5202
                                  21: 5185
                                  22: 5161
                                  23: 5166
```

DB1-S1 data is a totally row time-series data set, so it's important to preprocess it before training. First, we need to drop the data of the rest stage (label 0). Considering the error at the beginning and end of each movement, we select 70% of the data in the middle of each set of movements as training data. And take a 120ms (12 data points) windows to convert emg data into a 12*10 size of image. At the same time, we use a 52-dimension one-hot matrix to represent our labels. The preprocessed dataset is found from malele4th on GitHub [9], with the data shape shown below in Figure 3.

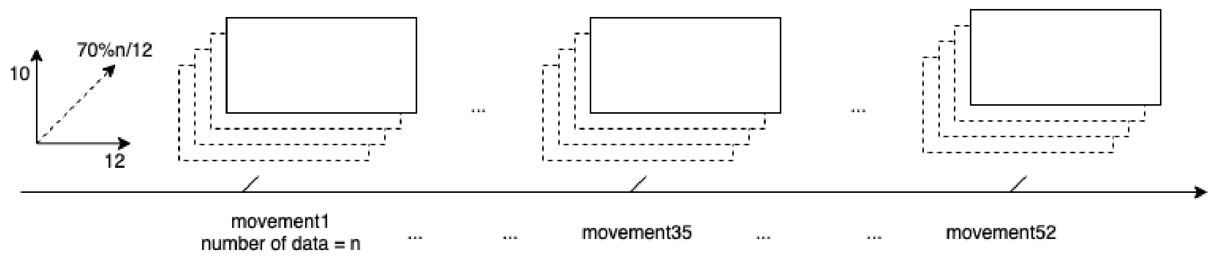
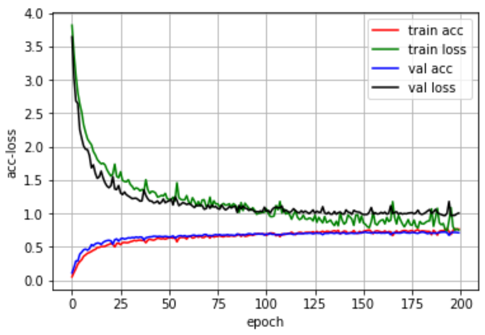
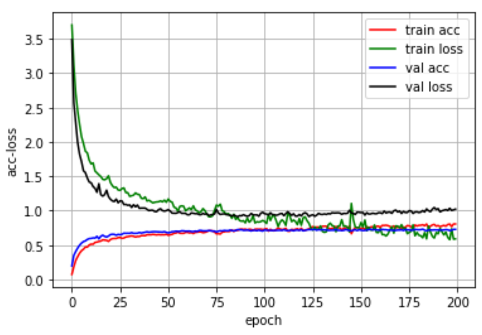


Figure 3: Preprocessed dataset dimensions

Result

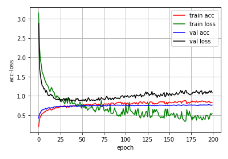
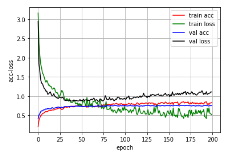
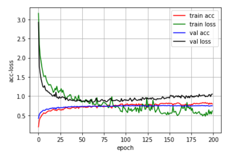
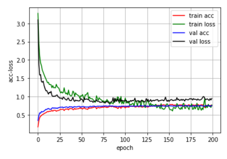
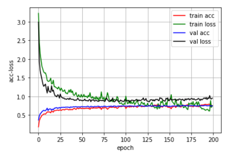
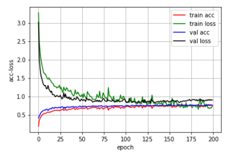
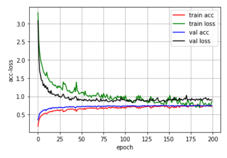
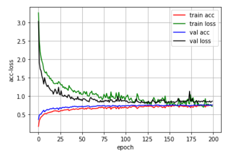
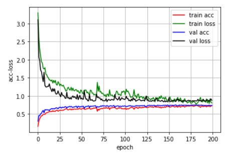
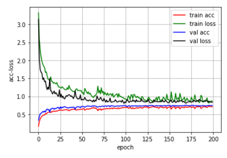
Because the related works we referenced are not tested under the same conditions and on the same dataset, their accuracies are not eligible for direct comparisons. Therefore, we duplicated the ideas from the referenced architectures and compared them on the same dataset with the rest conditions consistent. We ran 200 Epochs for each model and have concluded the results in the tables shown below.

Table 2: Performance Comparison between single-stream CNN and Mario CNN for 200 Epochs

	CNN	Mario CNN (two-stream)
Batch Size	`256`	`256`
Input Shape	`(12, 10, 1)`	`(12, 10, 1)`
Train Accuracy	`0.805`	`0.877`
Test Accuracy	`0.817`	`0.884`
Loss & Acc Plot		

By increasing the number of streams from 1 to 10, given we only have 10 channels, we have observed the test accuracy increased but overfitting occurs. The loss and accuracy plots are shown below in Table 3. Train/test accuracy as a function of the number of streams is plotted as shown in Figure 7.

Table 3: Loss and Accuracy Plots for Different Number of Streams

1-stream	2-stream	3-stream	4-stream	5-stream
				
6-stream	**7-stream**	**8-stream**	**9-stream**	**10-stream**
				

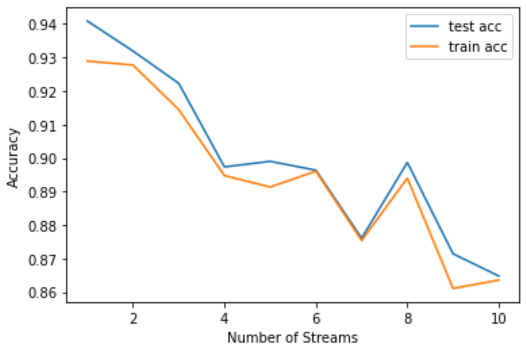


Figure 7: Train/test accuracy as a function of the number of streams

Discussion and Conclusions

Based on the results, it is obvious that two-stream architecture, which we called Mario CNN, did improve the performances. Given the same hyperparameters and layers, we applied a single-stream CNN and Mario CNN on the same preprocessed dataset. Mario CNN gets higher accuracies on both training set and test set around 88%, while the single-stream CNN only gets accuracies around 81%. One possible reason is that 10 channels have different feature concentrations, if we train these data using a one-stream neural network, we could only get some "average information" from them and lose some important information. Instead, training them using two streams and combining them into one classifier could help us extract more information from sEMG data.

We have also noticed that purely increasing the number of streams is not a deliberate method, although we may see accuracies increase with appropriate tuning. Given the same hyperparameters and layers, from Figure 7, as the number of streams increase, the training and test accuracies overall decrease. Some fluctuations (from 7 to 8) occur for statistical stochastic reasons. By increasing the number of streams, the fed-in data in each stream becomes smaller in dimension, so pooling layers are highly restricted in size and therefore capture too much insignificant information, leading to overfitting. Choosing appropriate layer size, parameters, and the number of streams becomes a challenge, and it is highly dependent on the problem content and data size. Once finely tuned, multi-stream architecture is proven to be effective.

Even so, software improvement is effective until a certain point and then becomes saturated, and further improvements can be achieved by hardware improvements. Some papers point out that the use of highly accurate sensors (i.e. dry type sensors to wrist [2]) is effective in improving the correct recognition rate, which can obtain 95% accuracy easily. Moreover, changing electrodes positions also can improve the accuracy of recognition. The results show that electrodes applied near the wrist are more likely to discriminate gestures. Indeed, in a recent study[10] on able-bodied subjects, the arm position was shown to significantly influence the performance of pattern classification-based myoelectric control algorithms. The results of the current study confirmed that such an effect is also relevant for algorithms based on the preprocessing and proportional control of NinaPro.

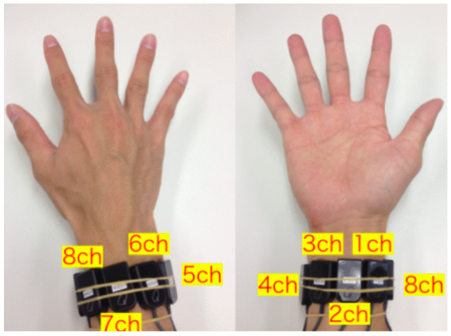


Figure 8: Changed Sensor Positions for Hardware Improvement

We can also improve our work by changing our input data. (eg: try to construct a new data feature map to improve the accuracy of sEMG classifier[11]). The existing DL studies on sEMG using data extracted from either time domain(TD) or frequency domain(FD). However both of these methods can only obtain partial information from sEMG. In order to solve this inappropriate data processing problem, this essay proposes a fusion framework: choose a combination of TD and FD features and use it as the input data of cnn. The accuracy of the proposed method is 8.58% higher than SVM.

In conclusion, we successfully built a novel two-stream Mario CNN and it is proven to perform better than a single-stream CNN on the dataset NinaPro-DB1-S1. In the meantime, a few reflections came to our minds. Firstly, while the kernel size plays a pivotal role in information extraction, the casual relationship between hyperparameters and accuracy in this black-box procedure is still a mystery. Second, given that the signals from different channels reflect different information, tuning the streams independently based on signal characteristics rather than having all the streams the same may grant more flexibility and practicalities. Finally, and most importantly, concluding a model is "better" than another merely based on accuracies on one dataset is ambiguous and dogmatical. As an application project, sEMG signal processing is affected by many other factors, like biological conditions, scenarios, data sources, etc. To further expand this project, it is more valuable to consider this Mario CNN as a tool and apply it on more diverse datasets and study sEMG signal processing as a broader problem from a statistical perspective.

References

[1] Tankisi, H., Burke, D., Cui, L., de Carvalho, M., Kuwabara, S., Nandedkar, S. D., Rutkove, S., Stålberg, E., van Putten, M., & Fuglsang-Frederiksen, A. (2020). Standards of instrumentation of EMG. Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology, 131(1), 243–258. https://doi.org/10.1016/j.clinph.2019.07.025

[2] R. Shioji, S. Ito, M. Ito and M. Fukumi, "Personal Authentication and Hand Motion Recognition based on Wrist EMG Analysis by a Convolutional Neural Network," 2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS), 2018, pp. 184-188, doi: 10.1109/IOTAIS.2018.8600826.

[3] Buongiorno, D., Cascarano, G. D., De Feudis, I., Brunetti, A., Carnimeo, L., Dimauro, G., & Bevilacqua, V. (2021). Deep learning for processing electromyographic signals: A taxonomy-based survey. Neurocomputing, 452, 549-565.

[4] Atzori, M., Cognolato, M., & Müller, H. (2016). Deep learning with convolutional neural networks applied to electromyography data: A resource for the classification of movements for prosthetic hands. *Frontiers in neurorobotics*, 10, 9.

[5] Sri-iesaranusorn, P., Chaiyaroj, A., Buekban, C., Dumnin, S., Pongthornseri, R., Thanawattano, C., & Surangsrirat, D. (2021). Classification of 41 Hand and Wrist Movements via Surface Electromyogram Using Deep Neural Network. Frontiers in Bioengineering and Biotechnology, 9, 394.

[6] Ding, Z.; Yang, C.; Tian, Z.; Yi, C.; Fu, Y.; Jiang, F. sEMG-Based Gesture Recognition with Convolution Neural Networks. Sustainability 2018, 10, 1865.

[7] Wei, W., Wong, Y., Du, Y., Hu, Y., Kankanhalli, M., & Geng, W. (2019). A multi-stream convolutional neural network for sEMG-based gesture recognition in muscle-computer interface. Pattern Recognition Letters, 119, 131-138.

[8] http://ninapro.hevs.ch/node/2

[9] Malele, sEMG_DeepLearning, (2018), GitHub repository

[10] Jiang, N., Muceli, S., Graimann, B., & Farina, D. (2013). Effect of arm position on the prediction of kinematics from EMG in amputees. Medical & biological engineering & computing, 51(1), 143-151.

[11] Luo, Y., Luo, T., Xia, Q., Yan, H., Xie, L., Yan, Y., & Yin, E. (2021, December). A Fusion Framework to Enhance sEMG-Based Gesture Recognition Using TD and FD Features. In International Conference on Neural Information Processing (pp. 168-175). Springer, Cham.