Classifying Aggressive and Non-Aggressive Body Gestures – Using Limbs EMG Monitoring

Yoav Sabag	Yuval Luria	Barel Hatuka
205382690	318390275	203765227

Abstract

This research is proposing a new way of classifying an EMG records as an aggressive or non-aggressive actions, with potential applications in healthcare and security. Leveraging the power of the Transformer models which are widely known for their exceptional attention mechanism. The data used in this research is received from UC Irvine Machine Learning Repository website under the name "EMG Physical Action Dataset". Our research involves 2 steps, the first is to build an accurate classifier for distinguishing between aggressive and normal (non-aggressive) actions. And the second is to train a transformer model to distinguish between each action within their category. And to evaluate the performance of our model, we use metrics such as Mean Squared Error (MSE) and Accuracy. Lastly, the code used in this research is written by us and compiled and ran on Macbook M1 Max.

1. Introduction

In recent years, Transformers models have emerged as the leading approach for sequential data analysis due to their exceptional characteristics, including efficient computation time, attention mechanism, long-term memory, and parallel processing capabilities. These models, exemplified by GPT and BERT, have primarily gained recognition in the field of translation. The attention mechanism, a key feature of Transformers, significantly enhances their efficiency during parallel execution.

In this study, we worked to leverage the power of the Transformers model on a time series dataset, composed of data collected from 8 parallel independent EMG sensors. Our objective was to achieve the best possible results in classifying actions based on this dataset. The EMG dataset is a set of 20 actions overall, 10 for aggressive and the other 10 for non-aggressive. To accomplish this, we built a Transformer model and a training process that utilizes the computational power of a M1-Max chip. The training process lasted for approximately 10 minutes, during which the model learned to effectively classify the actions based on the provided data.

2. Background

2.1. Electromyography (EMG) Monitoring

EMG monitoring is a technique that records the electrical activity in our muscles. It involves using small electrodes on our skin

or inserting needle electrodes into the muscles to detect and amplify the signals when the muscles contract or relax. This information is valuable in understanding how our muscles work, identifying activity patterns, and spotting neuromuscular disorders. EMG monitoring has various uses in sports medicine, rehabilitation, and research, helping us study the interactions between nerves and muscles and advancing our knowledge of biomechanics and human movement analysis.

2.2. Data Collected From EMG Monitoring

The EMG Monitor samples the voltage from the muscle at a defined frequency, the samples are collected during the time and creates a data set.

The usual amplitude range of EMG signal is 0-10 mV (+5 to -5) prior to amplification. EMG signals acquire noise while traveling through different tissues, which is accountable in our study.

Figure 1 - EMG Monitoring

2.3. Transformers

Transformers are widely used for artificial understanding of long sequential or hierarchical data, making them the dominant models for sequence based tasks. They have been successfully applied in various applications. In this experiment we used transformers as a classifying model when the data is a time series.

Unlike traditional approaches based on recurrent or convolutional neural networks, Transformers stand out by utilizing a self-attention mechanism. This mechanism allows them to capture dependencies between distant positions in the sequence, enabling better understanding of long-range relationships.

Transformers consist of an encoder for handling input and a decoder for generating output. This architecture enables efficient parallel processing, resulting in faster learning rates as multiple values can be processed simultaneously.

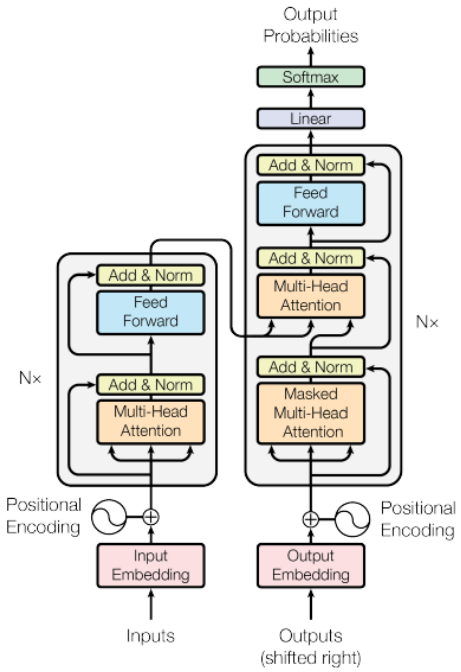


Figure 2 - Transformers Model Architecture

2.4. Time Series

Time series data consists of a sequence of data instances recorded at constant time intervals. In the context of our study, we used data from EMG monitoring, although the specific frequency at which the signals were measured remains unknown. Typically, time series data can be decomposed into components such as trend, seasonality, and noise. However, in our analysis, our main focus was on observing changes in variance over time. It is important to note that due to the unknown frequency, we made assumptions that all actions occurred consecutively. Time series, in a general sense, refers to a sequential data representation that allows us to analyze patterns and dynamics of the data over time.

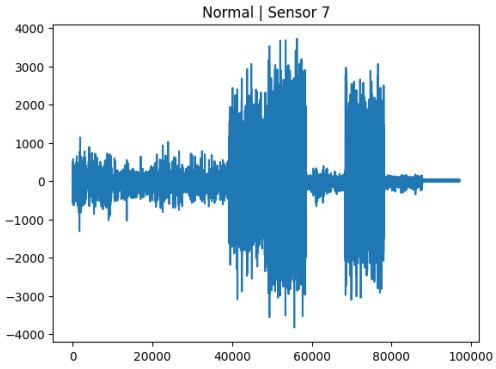


Figure 3 - Normal 7th sensor plot

Figure 3 shows the plot of the 7th sensor connected to the examinee, during the test the sensor frequently sampled the muscle electricity pulses over time, when the examinee performed the activity there was an obvious change in the variance of the data collected series.

2.5. Why attention and TS

The Transformers model, particularly its attention mechanism, is highly beneficial for analyzing time series data, especially when dealing with multiple time series in parallel. The attention mechanism allows the model to capture long-range dependencies and establish connections between different time steps within the series. This is particularly advantageous when working with time series data, as it enables the model to have long-range memory and consider relationships across a wider temporal context. By using attention, the model can effectively incorporate information from various time steps in parallel, enhancing its understanding and capturing complex patterns within the time series data.

3. Experiment Methodology

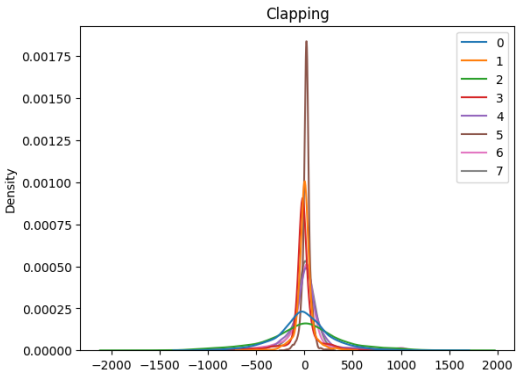
3.1. Data Set

The data used in this study is derived from the School of Computer Science and Electronic Engineering. The experiments involved 4 subjects, 3 male and 1 female, aged between 25 and 30, who have experienced aggression at some point in their lives. Each subject was asked to perform a series of activities, both normal and aggressive, totaling 10 actions each. The normal activities included bowing, clapping, handshaking, hugging, jumping, sitting, standing, running, walking, and waving. The aggressive activities included elbowing, front kicking, hammering, heading, kneeing, pulling, punching, pushing, sidekicking, and slapping.

The information given in Each column (Each variable) is a time series that correspond for each segment of a muscle:

- R-Bic: right bicep (C1)
- R-Tri: right tricep (C2)
- L-Bic: left bicep (C3)
- L-Tri: left tricep (C4)
- R-Thi: right thigh (C5)
- R-Ham: right hamstring (C6)
- L-Thi: left thigh (C7)
- L-Ham: left hamstring (C8)

The dataset files are organized into separate folders for each subject, with two subfolders labeled "aggressive" and "normal." Each subfolder contains datasets for 10 activities. For each activity, there are 8 features representing 8 channels in EMG (electromyography), along with the labeled activity. The dataset comprises approximately 10,000 samples, representing 15 actions in each experimental session. However, there is a lack of information regarding the sampling frequency used in the data collection process.



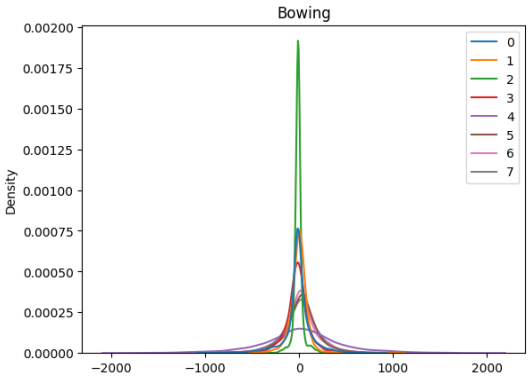


Figure 4 - Normal Class, All 7 Sensors Plot

As shown in the figure, the sensor values standard deviation of the normal activities is 2060.8 on average and expectancy of 0.

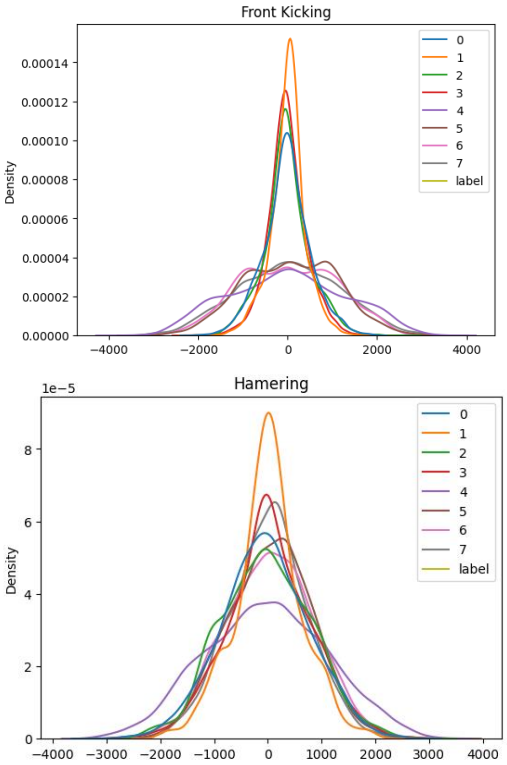


Figure 5 - Aggressive Class, All 7 Sensors Plot

As shown in the figure, the sensors' values standard deviation of the aggressive activities is 17097.4 in average and expectancy of 0, therefore the first classification condition is determined by threshold of std=8000.

3.2. Model Methodology

3.2.1. *Data Preparation*

Data Preparation is a long iterative process that involves multiple steps such as data cleaning, feature engineering, normalization etc.

The first step is to load the EMG data for different physical actions. Usually there's nothing special when loading the data, but in our project we're performing a couple of steps doing so, the first is to differentiate between Aggressive actions and Normal actions. After studying our data we came to the conclusion that calculating the standard deviation of each action can make that difference. So, the first step is to create 2 datasets for each action type, normal and aggressive, the threshold is standard deviation of 8000, each action below that is classified as normal and each above is classified as aggressive, and the number of classes of classification is based on the number of actions in each dataset.

The second step is to split the data into train and test datasets, we're doing it early in the process due to organization of the files, as mentioned before each action has approximately 10,000 rows and about 15 repetitions of action, so the split is 14/15 of each file is concat with the train dataset and the remain of 1/15 of each file is go to test dataset.

Finally each action gets a numbered label assignment on loading.

3.2.2. *Wavelet Denoising*

Wavelet Denoising is a method to enhance the quality of the EMG data, the function utilizes the PyWavelets library to perform denoising on each channel of the input data, in our data each channel is a column in the dataset. It decomposes the data using a wavelet transform, applies a threshold to remove noise, and reconstructs the denoised data. This process helps to enhance the important parts of the data and the output of the denoising is then being fed into the model.

3.2.3. *Model Architecture*

We defined a custom Transformer model using PyTorch library. The model class consists of several components. The input data is passed through an embedding layer, which maps the input dimensions to a higher-dimensional space. A Transformer encoder layer is then applied to capture the sequential dependencies in the data. The output of the encoder is flattened, followed by a dropout layer for regularization.

Finally, the output layer produces the predictions for each class.

3.2.4. Training

The training process begins by converting the training data and labels into PyTorch tensors. These tensors are then used to create a DataLoader, which allows for efficient batch processing during training. The model's parameters are optimized using the Adam optimizer, and the learning rate is adjusted using a learning rate scheduler, and for the loss function we went with cross-entropy. The model is trained in a loop for a specified number of epochs. And for each epoch the model iterates over the training data in batches the DataLoader created. It performs forward and backward passes, calculates the loss, updates the weights, and tracks the training accuracy. After each epoch, the model is evaluated on the validation set, also converted into tensors, and the model's outputs are computed. The validation loss and accuracy are calculated using the cross-entropy function and the results are the predicted labels. If the validation accuracy improves from the last epoch, the model's state is saved, if the validation doesn't improve for a patience parameter that was defined beforehand, an early stopping mechanism is triggered to prevent overfitting.
Finally, the best model is loaded, and the model is evaluated on the test data, which is also converted into tensors, and the model's outputs are computed. Then the predicted labels are computed to the ground truth labels to calculate the test accuracy, precision, recall, and F1-score.

3.2.5. Optimizer: Adam

The optimizer has a very important role in the training process, it determines how the model's parameters, such as weights and biases are updated during training to minimize the loss and improve the model's performance. We used the Adam optimizer, which is short for Adaptive Moment Estimation. Adam combines the benefits of two other algorithms, namely Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp).It dynamically adjusts the learning rate for each parameter based on the magnitude of the gradients and the previous update information.

3.2.6. Loss Function: Cross-Entropy

The loss function, similar to the optimizer, has a crucial role in the training process, quantifying the dissimilarity between the predicted outputs and the true labels. Cross-entropy loss is a commonly used loss function when it comes to multi-class classification tasks. It calculates the loss by measuring the difference between the predicted class probabilities and the true one-hot encoded labels. The function penalizes larger deviations between the predicted and the true, encouraging the model to learn accurate class predictions. Let's assume we have C classes and the predicted probabilities for each class are represented by a vector $\hat{y} = [\hat{y_1}, \hat{y_2}, ..., \hat{yC}]$. The true labels for each class, represented as a one-hot encoded vector, are denoted by $y = [y_1, y_2, ..., y_i, ..., y\cdots, y\cdot]$. The cross-entropy loss is then calculated as:

$$L(\hat{y},y)=-\sum y_i \log(\hat{y_i})$$

This equation computes the negative log-likelihood of the true class label based on the predicted probabilities. It sums over all classes (from 1 to C) and takes the logarithm of the predicted probability for the true class. The negative sign is applied to make the loss a positive value.

3.2.7. Regularization: Dropout

Dropout is a regularization technique commonly used in neural networks to prevent overfitting. It's used during model training to improve generalization. Dropout randomly deactivates a fraction of neurons during each training iteration, introducing noise and preventing the model from relying too heavily on specific connections. This technique encourages the model to learn more robust and generalized features and connections, making it less sensitive to small variations in the input data.

3.2.8. Hardware: Macbook M1 Max

The code was executed on a Macbook M1 Max, which incorporated a powerful ARM-based CPU with a large number of cores,

enabling efficient parallel processing. Additionally, the computer features neural cores. The neural engine is a specialized component within the M1 chip that is specially designed for machine learning tasks.

4. Results and discussion

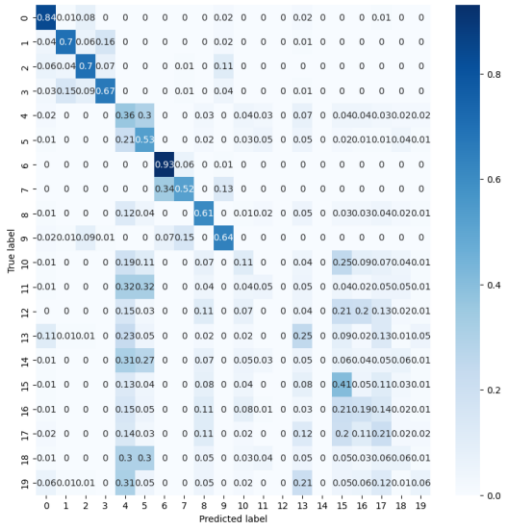
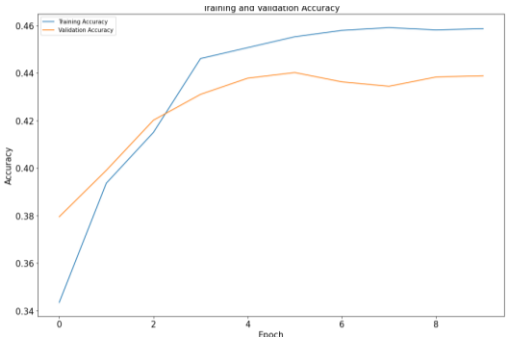


Figure 6 - Confusion Matrix
The confusion matrix presented in Figure 6 illustrates the model's confusion between different classes. This confusion can be attributed to the similarity between certain movements. For instance, the model displayed a 30% confusion rate between the "kneeing" and "jumping" actions. This confusion is expected since both actions may involve similar sensor readings and exhibit comparable patterns.

Aggressive Accuracy



Normal Accuracy

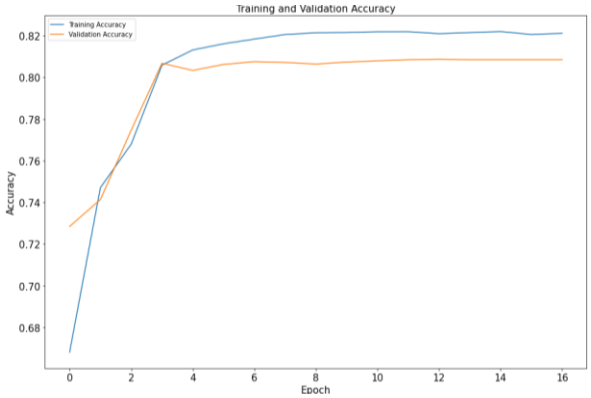
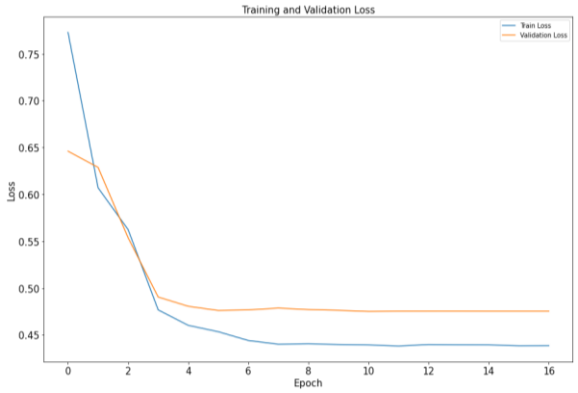


Figure 7 –Training vs Validation Accuracy
The observations from Figure 7 indicate a progressive increase in model accuracy as the epochs progress and as expected eventually the validation is lower than train.

Normal Loss



Aggressive Loss

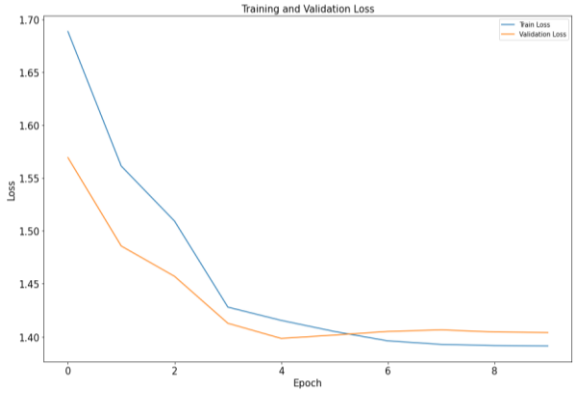


Figure 8 – Training vs Validation Loss
Figure 8 provides a graphical representation of the training and validation losses as a function of the number of epochs.

Classification Report:

	precision	recall	f1-score	support
0	0.88	0.77	0.82	657
1	0.92	0.90	0.91	667
2	0.79	0.58	0.67	641
3	0.93	0.94	0.93	651
4	0.90	0.69	0.78	667
5	0.67	0.90	0.77	649
6	0.82	0.91	0.86	667
7	0.66	0.79	0.72	667
accuracy			0.81	5266
macro avg	0.82	0.81	0.81	5266
weighted avg	0.82	0.81	0.81	5266

Figure 9 - Classification Report of normal actions.

Figure 9 provides information on the classification prediction accuracy for each of the 8 individual actions, as well as the overall accuracy of the "Normal" class. There's only 8 out of 10 due to the filtering of the first phase.

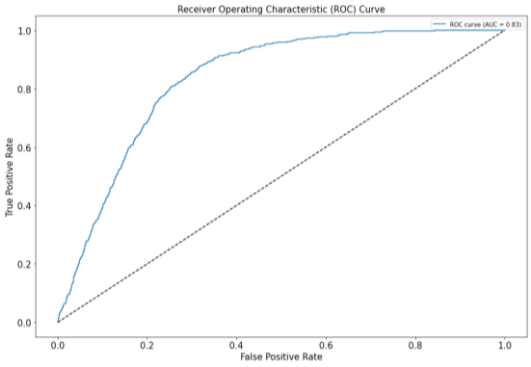
Classification Report:

	precision	recall	f1-score	support
0	0.49	0.39	0.43	653
1	0.28	0.35	0.31	655
2	0.43	0.41	0.42	667
3	0.26	0.31	0.28	667
4	0.60	0.40	0.48	644
5	0.40	0.28	0.33	643
6	0.51	0.58	0.54	646
7	0.27	0.20	0.23	656
8	0.64	1.00	0.78	653
accuracy			0.43	5884
macro avg	0.43	0.43	0.42	5884
weighted avg	0.43	0.43	0.42	5884

Figure 10 - Classification Report of aggressive actions.

Figure 10 presents a classification for 9 classes. This classification is derived from a binary classification process during the first phase.

Aggressive ROC



Normal ROC

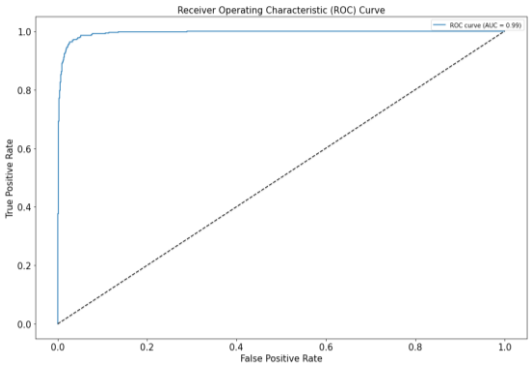


Figure 11 - ROC Curves

In the figure above, the model exhibits excellent performance as evidenced by an approximate true positive rate of 1, indicating a high level of accuracy in correctly identifying positive instances.

5. Conclusion

The exploration of using Transformers as a classification model has been both an educational and captivating journey for us. The potential that Transformers hold to revolutionize machine learning and classification is truly remarkable. Throughout our research, we have found this approach to be rewarding and promising.

However, it is worth noting that in our experiment, the performance of the Transformers model fell short compared to that of a random forest model. This discrepancy could potentially be attributed to the accuracy of the time series data utilized. Time series data, being inherently sequential and dependent on temporal patterns, can pose challenges in terms of accuracy and representation.

Nonetheless, this experience served as a valuable learning opportunity for us. We were able to explore and compare different classification methods, including the Transformers model and random forest, gaining insights into their strengths and weaknesses. Each method offers unique advantages and considerations, and this exploration broadened our understanding of the diverse approaches available for classification tasks.

In conclusion, while our specific application of the Transformers model may not have outperformed a random forest, the investigation has deepened our knowledge and expertise in utilizing alternative classification techniques. The journey has been rewarding, providing us with valuable insights and paving the way for further advancements in machine learning and classification research.

6. References

6.1. Analysis of EMG Physical Data: Aggressive & Normal Activities by Shraddha Anala | May 5, 2020:https://archive.ics.uci.edu/dataset/213/emg+physical+action+data+set

6.2. Chada, S., Taran, S., & Bajaj, V. (2019). An efficient approach for physical actions classification using surface EMG signals. *Health Information Science and Systems*, *8*(1), 3.

6.3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

6.4. Xiong, C., Zhong, V., & Socher, R. (2016). Dynamic co-attention networks for question answering. *arXiv preprint arXiv:1611.01604*.

6.5. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016, June). Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies* (pp. 1480-1489).

6.6. Ahmed, S., Nielsen, I. E., Tripathi, A., Siddiqui, S., Rasool, G., & Ramachandran, R. P. (2022). Transformers in time-series analysis: A tutorial. *arXiv preprint arXiv:2205.01138*.

7. Additional information

7.1. Long term Forecasting with Transformers Https://arxiv.org/abs/2211.14730

7.2. Analysis of the dataset - https://medium.com/analytics-vidhya/analysis-of-emg-physical-data-aggressive-normal-activities-4d5a696730b4

7.3. Data set analysis - https://dataportal.asia/dataset/212571409_emg-physical-action-data-set

7.4. Code for the dataset on kaggle - https://www.kaggle.com/code/durgancegaur/emg-dataset#Combining-the-two-models

7.5. https://github.com/ranakroychowdhury/TARNet - Task-Aware Reconstruction for Time-Series Transformer

7.6. https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html