Summary

Motor movement is a fundamental part of every person's life. We use our arms and hands more or less constantly, without even being aware of it for most of the time. But aging and various sicknesses and illnesses can impair these functions. There can still be hope of recovery sometimes, depending on the tardiness of the impairment. One way of rehabilitation, for someone who lost grip strength or control in their fingers, could imply using an external device that would aid with the grip. But such a device would require to know when to activate itself and offer the additional mechanical grip strength in order to grasp an object, and that is where this research plans to aid, on a software level.

Our brains, with their billions of neurons, and trillions of synapses, are constantly sending signals to our bodies to tell them to perform certain actions and activate certain things and processes. When we are grabbing something of the table for example, that action of grabbing something has its own signal, signal commonly named as a movement related cortical potential, or MRCP.

This article will explore if it is possible to conduct transfer learning using MRCPs from one main subject to another secondary one. The motivation is that detecting MRCP currently requires a classifier to be created for each subject, individually. However, this process is time consuming, and can only be done in a lab with proper equipment, since labeled data is needed for training, and that requires even more dedicated equipment as opposed to just simply recording the brainwave signals. We will explore the possibilities of implementing a GAN network to construct a new classifier for new subjects.

We started by conducting experiments on three healthy male subjects. We chose one person as the main subject, and this subject was used to train our network. Our primary focus was to gather as much data as possible for this subject to ensure the best possible training conditions. The experiment consisted of reaching for an object with their primary arm, and we called this a movement. 30 movements were done in a row, then the subject had a small break. The subject would repeat this loop until we had 150 movements. The subjects were equipped with an EEG helmet to record their brain's activity and electromyography (EMG) electrodes to record muscle activity in the arm.

After the experiments, the EMG data was filtered using a high-pass filter with an 80Hz cutoff. Then we used an algorithm to find each signal's peaks. The EEG data was filtered using a 2nd degree Butterworth band-pass filter with a cutoff of 0.05Hz to 5Hz since, in this spectrum, MRCP is most likely to be found. The EEG then had to be transformed into a workable dataset for our classification method. The peaks of EMG were used to find the onset of movement in the EEG, but since MRCP can occur before the movement, we had to adjust the peak by locating the minima of the EMG signal closest to the peak. Then we took the interval 1s before and after this adjusted point and labeled it *active*. Furthermore, we took the interval from 4s to 2s before the adjusted point and labeled it *rest*.

Our first experiment consisted of creating a classifier, than both, could classify MRCP signals, but also could be trained. We need this train-ability in order to create a GAN network later on. We choose a TCN-Encoder with a dense classifier as our classification method for MRCP. The results of MRCP detection on the 5 data sets concluded with an average accuracy of 0.77, and 3 data sets reaching 0.86 accuracy with the lowest score at 0.61. We then did a baseline before doing transfer learning, by taking the TCN model for the main subject as using it on all the other data sets. Looking at the results we scored lower than individual TCN as expected with the average accuracy reaching 0.63, and the lowest score achieved was 0.51. Even the datasets being part of the main subject had lower scores, and the deviation was even bigger for secondary ones. We created a GAN network for the two secondary subjects, to test if we could achieve higher accuracy. Both subjects scored a higher accuracy after the GAN training, with an overall increase of 0.07 accuracy compared to the baseline. It should still be noted that training the TCN with individual data did reach higher a accuracy. In conclusion, we showed that a setup could be made that successfully enables transfer-learning with MRCP for hand movement. We did see a higher accuracy after GAN training compared to the baseline, but still lower than standalone training. We believe this trade off is worth assuming, since it would allow a larger group of people to use the system, as time would be saved by no longer requiring labeling.

Exploring transfer-learning with TCN-encoders using movement related cortical potentials in single-trials

Andrei I. Katona
Aalborg University
akaton20@student.aau.dk

Rasmus S. Krog
Aalborg University
rkrog17@student.aau.dk

Abstract

Classifying movement related cortical potentials (MRCP) from hand related movements is a trivial task if sufficient labeled data is present, but it becomes less trivial when no labeled data is available. In this paper we present an approach of transfer learning, which optimizes an encoder that creates inputs for a classifier. The optimization is done with the aid of a generative adversarial network (GAN), and the result is an improvement in the performance of unlabeled data classification for MRCPs.

The overall idea revolves around training a temporal convolutional network (TCN) with a large amount of data from the main subject. Our TCN is made up of a TCN encoder and a classifier, therefore the TCN encoder will also be trained according to the main subject. Afterwards, we can interface the trained encoder with the GAN, which allows us to train another encoder, for a secondary subject. This secondary encoder allows us to perform classification of MRCPs from the secondary subject, without requiring to label the data.

We trained five TCNs with five different data sets, which yielded an average accuracy of 0.77 with an average loss of 0.46. To test the transfer learning capabilities, we set up a baseline test. This test consisted of using our main subjects' data as the training for a TCN, then using the rest of the data to test on the TCN. The baseline had an average accuracy of 0.54 with a loss of 1.08. Lastly, we tested a GAN network for training the TCNs, with the same data as the baseline. This experiment gave a higher accuracy than the baseline with 0.62 and decreased the loss significantly to 0.64.

1. Introduction

Death caused by stroke is ranked the second in the world of death causes. Strokes are still an increasing problem since factors like obesity, stress, and overall health is causing patients to receive strokes. Some side effects of surviving a stroke are loss of muscle movement, chronic pain, and memory loss. It is estimated that 50% of survivors become chronically disabled. This leads to a higher impact on the world's medicine sector and lowers the life quality of surviving stroke patients [1]. Some side effects can be treated and potentially cured, whereas other effects are currently untreatable.

Brain-Computer Interface (BCI) is the field of study in creating systems using the brain as input and controlling applications. The methods for recording brain activity are invasive methods and non-invasive methods. We will be using non-invasive methods for recording brain activity. The non-invasive method we will be using is Electroencephalography (EEG) because of its easy setup process and portability. Most BCI is commonly split into many paths: medical, entertainment systems, and marketing. The focus of the article will be on the medical field in BCI. Medical BCI branches into many problems such as classifying cognitive disorders, a better understanding of the ADHD spectrum, preserving cognitive performance, and recovery of motor functions [2]. This journal suggests that BCI could help solve the elder burden by reducing cognitive aging and offer better rehabilitation. This article will be a continuation of [3] and, therefore, will focus on the classification of motor functions. In [3] they used three classification methods, AdaBoost, Support vector machine (SVM), and K-Nearest Neighbors (KNN), to classify movement potential. They found that using metrics such as slope coefficient and average voltage could yield an average 80% accuracy across five patients. Furthermore, they also tested a naive "online" method with their best-performing subject. This test achieved 70% accuracy. We will be taking the findings from [3] and we will be testing how transfer learning will perform. Research on motor function recovery has primarily been focused on lab setups, meaning no widespread commercial applications. One of the significant problems that BCI faces is developing classification algorithms that will work without large training trail setups. For a new user to be integrated into this system would require building an individual classifier with data from training sessions. In [4] they proposed a novel framework called Adversarial Discriminative-Temporal Convolutional Networks (AD-TCN) that can train classifiers for new users without having labeled data, meaning no controlled experiments for the user. They trained the model with the DREAMER dataset that contains EEG data and emotions as labels. In this article, we will further research the transfer learning AD-TCN, but with our dataset containing EEG data for motor functions.

2. Methods

This section outlines details about the process, such as test subjects, experiment was setup, and overall experiment flow. In order to facilitate the understanding of the process, details of the connection between brain and muscle will be described as well. Since this project is a continuation of [3], a lot of the processes remained the same, the differences being in the amount of data that was being gathered for the analysis.

2.1. A general overview on the process

In order to perform a classification on a subject, we are extracting 2 different sets of data from each subject. Firstly we place an EEG helmet on the scalp of the subject, this helmet is responsible for capturing the current created by the neurons in the brain [5], this allowed us to track the brains activity as we perform experiments. The EEG helmet we will be using is classified as a lab unit, this means that the recording frequency will be higher than commercial portable EEG helmets. Furthermore will we be applying conductive gel on each electrode to reduce noise in the signal. Secondly will we attach three EMG electrodes on the subjects primary arm, this arm will be used in the experiments.

The EEG data from the subject will be the base of our analysis, since this data reflect the brains activity doing trails. The EMG data will be used mainly to help label the EEG data. Our experimental setup records both EEG and EMG at the same time. If the muscle sends a higher voltage at one point in time, it only means that the muscle was activated and used or contracted harder in that instance, generating more volts, whereas when we get output from the EEG, those are more abstract and do not specifically have from start a direct correlation to what they might indicate. Therefore, to facilitate our training, we are using the EMG as timestamps, in order to know which portions of the EEG to slice and analyze afterwards[3].

2.2. Subjects

As per previous procedures presented in [3], the experiments were conducted on 3 healthy subjects, all male and right handed with the ages between 23 and 27 . Permission was given from each one of them and in order to create a proper environment where an adversarial discriminative network would have higher success chances, one subject was considered the main subject, having more data collected, in order to base the network on him. The main subject had 3 sessions of recording, in contrast to the other 2 subjects who only had 1 session. Since the end goal is to be able to train and classify EEG from "secondary" subjects with respect to the already trained main subject, without the use of EMG for the secondary ones, it would only make sense to have a stronger foundation or a properly trained first subject that by having multiple lab sessions used for training, would be able to accommodate to multiple variations or patterns that the secondary subjects may have.

2.3. Experiment setup

Subjects will all use a gtec cap, which as previous mentioned will be fitted with 11 electrodes and a ground clip. All of electrodes is connected to a g.gammabox which provides power and a driver for system. Then from the g.gammabox the signal goes intro an analog to digital amplifier which provides our digital interface for reading the signals. We will be running this system at 1200Hz. The brain produces an pattern called Movement Related Cortical Potential (MRCP), this signal only occur when the movement is voluntary, meaning the signal do not occur if the movement is a reaction. The MRCP begins about 400ms before the movement via a negative slop and begins to rise again after the movement[6]. This is signal that we intend to classify doing our experiments. The part of the brain responsible for producing MRCP can be found in the motor cortex. Our strategy for capturing the MRCP will be to place electrodes along the primary motor cortex, since this part is responsible for sending the neurological signal to the rest of the body [7]. We placed 9 electrodes running along the primary motor cortex at T7,C5,C3,C1,Cz,C2,C4,C6,T8 as shown in figure 1. Furthermore 2 electrodes are placed at the frontal lobes, Fp1 and Fp2. Note that Cz is used for reference, where the rest will transmit data, giving us 10 data channels for MRCP. A subject with all equipment can be seen in appendix A.

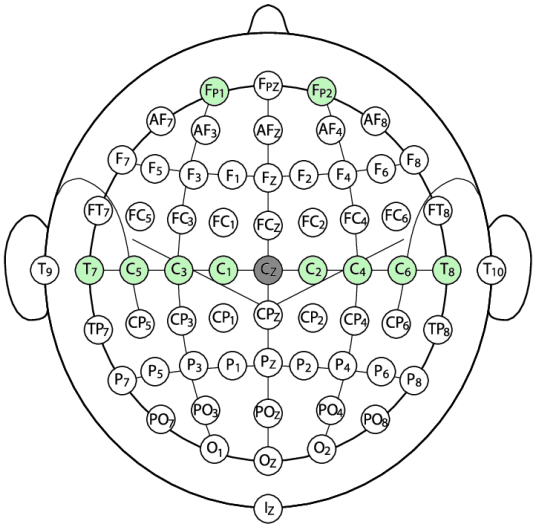

Figure 1. Placement of the 11 electrodes

Since we want to label the EEG for training, we can easily do that by following the logic mentioned earlier. Both, brainwaves(EEG) and muscle activity(EMG), are recorded at the same time. Therefor we can use the data collected from the EMG to help label the EEG, by creating reference points when the EMG signal peaks. The EEG data comes from the helmet and for the EMG, we use 3 electrodes on the subject's arm, as seen in figure 2. Electrode one is places on the carpal wrist bone, then second is placed on muscle going from the elbow to the hand. Lastly the third electrode is placed on olecranon bone. These three electrodes provides us with the EEG signal.

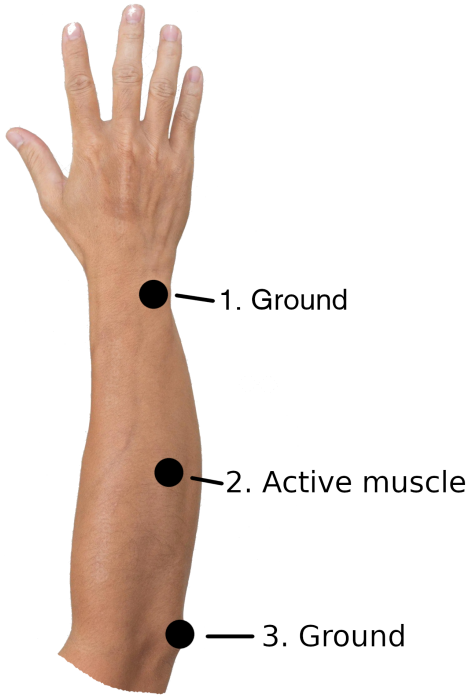

Figure 2. Positioning of the EMG electrodes on a arm

2.4. Experiment preparation

Before the experiment can begin, we showcase the movement so that the subject can perform this movement correctly. Then we fit the three electrodes for the EMG and the helmet for EEG. In order to prime the digital converter and amplifier, we have it running for a short period before actually recording and saving data. This helps settle the frequencies and somewhat normalize the data, since there is quite a lot of noise as soon as you turn on these devices. Afterwards, the participant then uses the muscle that we attached the EMG electrodes to confirm reading are correct, and then the participant is asked to perform a movement, so that we can check the EEG data is working. We are doing one last visual inspection over all brain channels, and if everything looks to be right, the experiment begins [3].

2.5. Experiment procedure

In order to complete the experiment, a subject has to perform 5 batches of 30 movements in succession. After being seated down, the motion consists of energetically grabbing a bottle with their primary hand, squeezing it hard in mid air, followed by placing it back on the table and returning the arm to the starting position. In an ideal training scenario, there would be around 6 seconds between each movement, time split between focusing before performing the movement and also relaxing after a previous one. The procedure of the movement is split into 5 phases and can be seen in figure 3. Most notably of these phases is the rest period, this phase ensures the subject can be focuses on only moving the arm. The rest period should normally only last around three seconds, but if the subject can take longer if they need to move body parts not related to the experiment. Without this rest phase the subject could eg. move their foot during the contraction phase which would have a big negative impact on the MRCP signal, since these MRCPs would overlap.

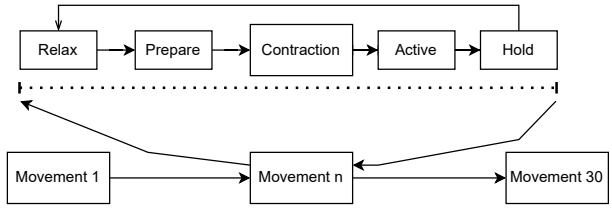

Figure 3. Overview over the experiment procedure with 30 movements and each movement consists of 5 phases

This repetitive task of grabbing a bottle, while easy to perform, can become a tedious task as the experiment goes on. Therefore, in order to stimulate and keep the subject up to date with how the recording goes, the observer or the person in charge of the recording will update the subject with the number of movements he did every 5 movements (e.g. "You have now done 5 repetitions", "15", "And those were the last 5"). The time between 2 batches of movements can be as long as the subject feels like they need in order to feel ready to start again.

The reason we chose 5 batches over any other number is that while a higher number of batches would give us more data, that would be mentally exhausting for the subject, which aside from fatigue, could decrease the quality of the data. Had we chosen a lower number like 2 or 3, while that could probably increase the quality by a small margin, it would greatly limit the amount of data we have, especially since due to hygienic reasons, after a recording session is done, the EEG helmet has to be washed, which renders it useless for at least a day, or until it properly dries out.

3. Analysis

In the following section will we describe how we manipulate the data in order to filter it. Furthermore, will we introduce our labeling strategy for producing training and test data. We will give details of what models we will use for classification and transfer learning. Lastly will we discuss which evaluation metrics we deem useful in our experiments.

3.1. Data overview

As per table 1, one dataset contains 16 channels out of which we use 11. The reason for being 16 is that the amplifier have 16 channels. Channel 1-10 contains the EEG signal from the brain, and channel 13 and 14 are EMG data. Channel 13 and 14 are polar opposite, therefor will we only continue with one of the channels. Channel 10 used primarily used for detecting blinking, therefor will this channel be excluded.

Channel(s) number/-name	Content
1 to 9, T7,C5,C3, C1,C4,C6, T8,Fp2, Fp1	EEG brain waves, used further for analysis.
10, C2	Special EEG channel which detects eye lid movements, used only visually to assure recording device works.
13 and 14	EMG data from the forearm.
11, 12, 15, 16	Leftover data used for ground

Table 1. Channels overview from a dataset.

3.2. Cleaning and labeling of EEG data

Before we can use the EEG data as input for classification, we need to both clean it of noise, as well as label it. In this section we will describe how we cleaned both EEG and EMG using different methods, and how we used EMG as a labeling strategy. For both signals, will we be removing the first 2s of each trial, since the recording device needs some time to calibrate and settle down in order to avoid artifacts on the first movement.

Cleaning the EMG

After recording the EMG, we are left with a noisy signal. As seen in figure 4, it is clear that the peaks of the EMG can be detected with either an algorithm or the naked eye, but this approach is not every effective on raw, unfiltered data, since the average fluctuations of a wave are not spread on multiple orders of magnitude, which could cause issues due to the noise on the outbursts that were bellow the average, and their peaks might be confused with neighbouring noise.

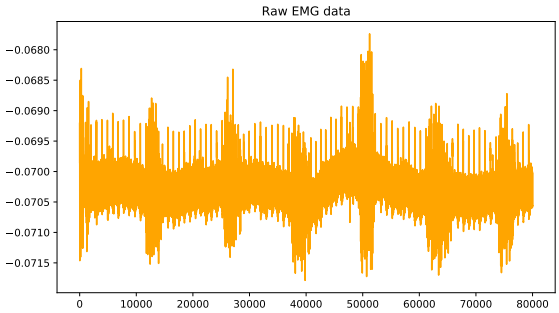

Figure 4. Raw EMG after data collection

In [8] they primarily saw the noise came from the 50hz power line. For removal of this noise, we will use a High-pass filter, which, in essence, removes all frequencies under a certain threshold. The threshold we choose was 80hz, and in figure 5 can we observe how the High-pass filter removed the issue with the noise and made the peaks more easily detectable.

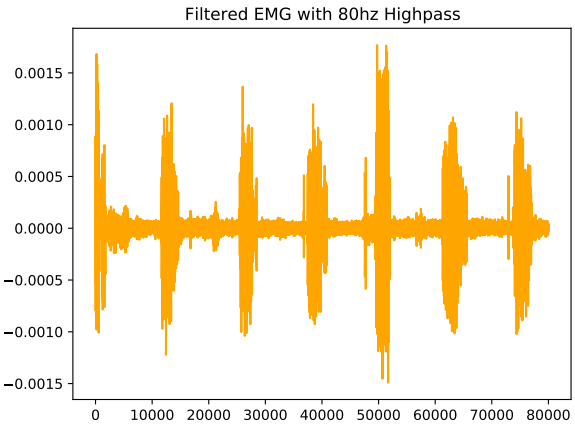

Figure 5. Filtered EMG after using 80Hz highpass filter

Cleaning the EEG

According to [9], the spectrum for MRCP can be found between 0.05Hz to 5Hz in the EEG signal. To separate this range from the signal, will we be using a butterworth band-pass filter. The reasoning is that a low-pass filter is not sufficient, as we want need to remove as much as we can from the noise generated by the equipment. A comparison of the signals can be seen in appendix B. After the signal has been cleaned, it is ready to be labeled.

Labeling EEG

Our labeling strategy will be be using the filtered EMG to find the movement, and then adjust to the negative peak of the EEG. Our approach to finding the peak of the EMG signal was to set a threshold, and if the signal amplitude broke this threshold, then that spot is considered to be a peak. Since noise would often hit this threshold, we set in a rule of how many points in a row should break the threshold for it to be a peak. As we knew that in each temporal sequence of EMG there should be 30 peaks, we would then adjust the threshold until we found 30 peaks. A figure of this can be found in appendix C. After finding the peaks we could then match these timestamps to the EEG data. Because the onset of MRCP happens before the EMG peak, we would then have to adjust the timestamps. This is done by locating the minima of the MRCP signal before the peak. We choose to use 1s before and after the minima of the MRCP, and we labeled this data as *active*. Since we knew that before the *active* period there was a rest phase, we labeled the 2-4s before the minima as *rest*.

3.3. Selection of Models

In order to perform benchmarks, we created temporal convolutional network (TCN) model, which is a specialized version of a convolutional neural network (CNN). Another model created was a more complex model for a GAN network. Both of these models make use of a TCN-encoder.

TCN-encoder, CNN and TCN

In most of the use cases, convolutional neural networks are aimed at tasks mainly involving features present in

the 2-dimensional or 3-dimensional space, where the relationships between individual data points matters collectively. A very common domain where this approach fits well is visual imagery, specifically anything related to image recognition[10].

If we take a brainwave, in comparison to an image, a brainwave has only 1 dimensional dependency for its data points, as opposed to an image which has 2. The brainwave is a 1 dimensional space of voltages, each at a different point in time. A single data point from a brainwave is not as expressive on its own, as compared to having the whole brainwave or a segment of it, while also taking into account the temporal relationships between points.

An image, on the other hand, has 2 dimensional spaces since the pixels making the image up are placed in a 2 dimensional grid, which has both a width and a height. In a similar manner to the brainwave's datapoint, an individual pixel is much less expressive on its own, compared to the whole image or even just a group of neighbouring pixels which make up a small section of the image.

If we have a situation where the temporal relationships between features matter, and we would like to use a CNN, but the features do not fit in a 2 dimensional space, we can instead use a special version of the CNN, known as a TCN, or temporal convolutional network, network which performs convolutions in a 1 dimensional space, and it was first proposed by Bai et all[11].

The way in which a TCN works is by applying filters based of a kernel size and a dilation factor, which encodes the input before it is being classified.

Overall, for a classification task, our TCN takes an input of the shape `[movements, 2400, 9]`, where `movements` are the amount of movements we want to classify and the latter 2 numbers represent the fact that we are feeding 9 EEG channels, each containing 2400 different data points.

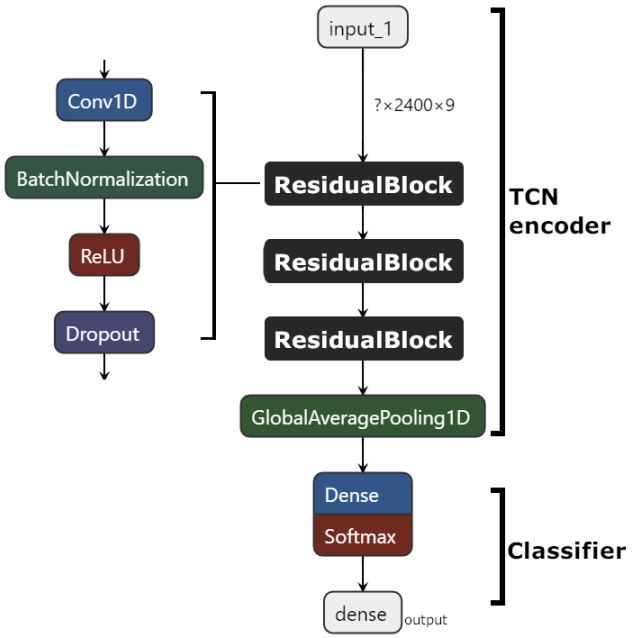

Figure 6. TCN structure overview. Input is passed through 3 residual blocks and a global average pooling, which is considered the encoder, before it is being passed to the classifier.

The layer structure of our TCN can be seen in figure 6. The network takes sequences of brainwave segments as inputs, passes them through 3 convolutional or residual blocks, afterwards it passes it through a fully connected global average pooling layer, and finally classifies the output with the help of dense layer having the softmax activation function. A residual block is responsible for a 1 dimensional convolution, and it is made up of 4 different layers, a temporal convolutional layer as an input, followed by a batch normalization, a ReLu, and finally a dropout layer in order to prevent overfitting and the posibility of the "dying ReLu" phenomena occuring, where due to the behaviour of ReLu, some neurons in the network might "die", and they end up outputting the same result, no matter the input.

The encoding part of the TCN is finalized when we get the output from the global average pooling layer. At that point, an EEG signal that started with a shape of `[1, 2400, 9]` has now been reshaped and reduced down to a much smaller array containing just `[32]` elements, which can be considered a downscale with a factor of 675 from the original footprint.

TCN-encoder and GAN

A generative adversarial network or a GAN is a class of machine learning techniques which run similar to a zero-sum game. It usually has 2 main components, a discriminator and a generator. The discriminator receives as inputs a real sample from the data set labeled as real, and a fake one generated from the generator, which is labeled as fake. The purpose of this zero-sum game is to train the generator to create fakes to such a fine degree that the discriminator is no longer able to tell them apart from the real inputs of the data set. Since the discriminator and the generator are in a continuous contest of improvement, the concept is oftenly reffered to as a zero-sum game, and one of the earliest remarks about this was in [12]. A visual representation of this concept can be seen in figure 7

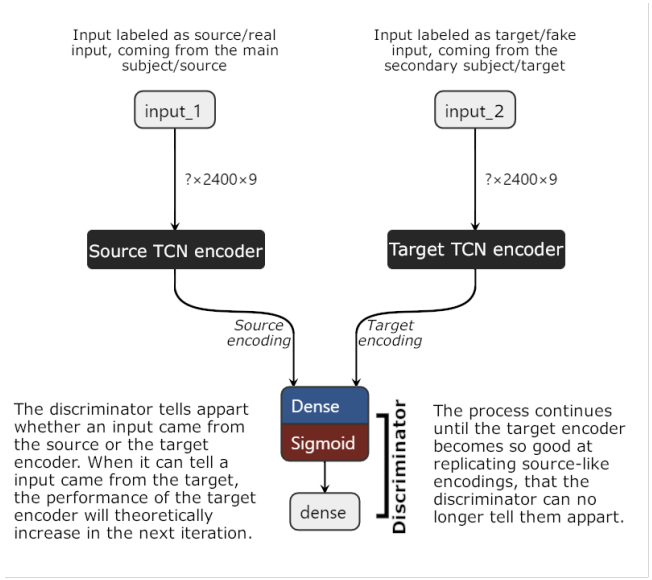

Figure 7. GAN overview. A discriminator tells apart an actual encoding made through the main subject, from an encoding made from a secondary subject.

While the data used during the training of a GAN is labeled, it is not considered labelled because the label is not given on a feature level, to separate the inputs based on their different features, but it is used instead to label

the domain of the input, in order for the discriminator to be able to verify through training whether the guess it made on the input's origin, being from either source or generator, was correct.

For our case, the discriminator is going to discern whether an input it received was from a source subject's signal or a target subject's one. At this point we can extract the TCN-encoder from our already trained TCN, and duplicate it into 2 copies, one for a source encoder, and one for a target encoder. Both of these encoders will receive an EEG signal and both of them will output its encoding. The source encoder is going to have its weights frozen, as we are only interested in training the target encoder to generate encodings of a more similar nature to the source. The training process afterwards consists of alternating training rounds for the discriminator, where it has to try to tell apart whether an encoding is from the source or the target. The key training segment happens afterwards, when we subsequently train the target encoder, and its weights get changed in order to generate an encoding that resembles more to the encodings from the source.

After a GAN training is complete, we can disregard the discriminator, extract the target encoder, and extend it with the pre-trained classifier from the TCN, process which should yield us a new classifier which did not require feature labels from the target for training. An overview of this process can be found in figure 8.

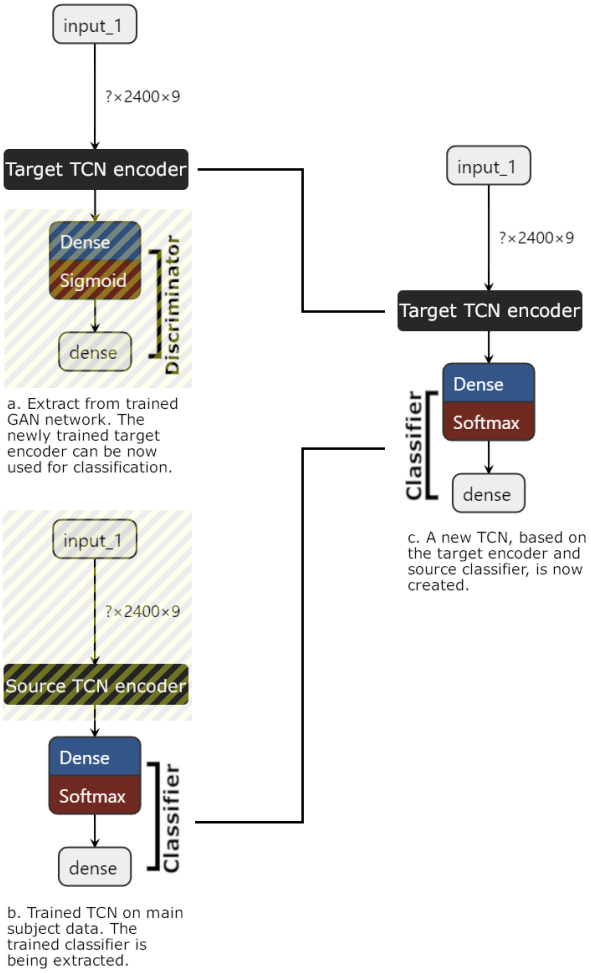

Figure 8. Overview on the process of creating a classifier from the newly trained encoder of a GAN and the pretrained classifier of the TCN.

3.4. Implementation details

All the data cleaning and all the models were created using python. The deep learning libraries `tensorflow` and `keras` were used to create, train and validate models, and `scikitlearn` was used to facilitate splitting the data between train and test. In order to boost the execution time, the `cuDDN` library from Nvidia was used as well.

The hyperparameters used were obtained through a process of trial and error, and our final selection for them, including the training part is as it follows:

TCN training, validating and hyperparameters

In order to train the main TCN, and concomitantly the TCN-encoder, our model has a filter size of 32, kernel size of 4, momentum of 0.65, and a batch consists of 32 randomly distributed samples. All the samples came from the main subject, on whom we performed data acquisition in 3 different sessions. A session contained 5 trials, each of them being made of 30 rest states and 30 active states, bringing the total to 900 samples. 2 thirds or 600 samples chosen at random were used to train, with a further validation split of 0.2. Finally, the last 300 samples were used purely for validation.

The model used the `Adam` optimizer, the loss was the main metric, and 2 callbacks from keras were used to prevent over training. The learning rate was 0.001 from start, but in case the loss reached a plateau for 20 consecutive iterations, the learning rate was reduced by a factor of 0.5, down to a minimum of 0.0001. The other callback used was early stopping, where if the loss did not improve after 50 iterations, the training would stop after that epoch.

In regards to validation on the other data sets except the main one, both of the other 2 subjects had 1 recording session consisting of 5 trials and a total of 300 labeled samples. The model containing the pretrained TCN was loaded, containing all the hyperparameters and weights computed for the main data set, and all 300 labeled samples from both subjects were fed in as testing samples for the network.

GAN training, validating and hyperparameters

The GAN had very similar setup of parameters compared to the TCN. The batch size was reduced to 10, the epoch count was set to 50, saving a model after each epoch, and the number of steps in an epoch was also reduced to 20.

Traditionally, the output from a generator would be a random noise that gradually becomes less random over time, but since our generator is actually the target encoder, our "random noise" is the encoding created by itself when a sample from a secondary data set is fed into it. This encoding already receives a head start by being pretrained on the main subject. The secondary data sets contain 300 samples and the discriminator ends up being trained 2,000 times with 10 batches randomly picked from the samples. Half of those times it is trained with the "real" encodings from the source encoder, and the other half it is trained with the "fakes" that the target encoder is trying to produce. The target encoder gets trained 1,000 times, also with 10 batches.

3.5. Benchmarks

This section will describe the benchmarking approaches we used to test our system. All the benchmarks ran on a

machine with an RTX2060, in order to make use of CUDA and tensor cores for faster tensor based mathematical computations. All the benchmarks were run 10 times and an average was made, but thanks to the vast amount of training data, the differences between the highest and the lowest values in a batch of 10 benchmarks were on average around the 3% mark.

Evaluation metrics

To evaluate our classification of MRCP during the offline experiments we are using accuracy. It provides us with a simple and reliable metric. Furthermore by using the accuracy as metric, it will allow us to compare our results to [4], as they used it for their experiments. In our experiments we will also use the confusion matrix to further evaluate our models and gain additional insights. The reasoning for this is that we strive to minimize false positives, since it will trigger an external device unnecessarily.

Individual classification of MRCP

The first benchmark that we conduct is classification of the MRCP signal. We will be using the aforementioned TCN model , made of the TCN-encoder and the Dense classifier. For each of the 5 subjects, we will train and validate the network w.r.t. the hyperparameters mentioned. An overview for the classification can be observed in Figure 9.

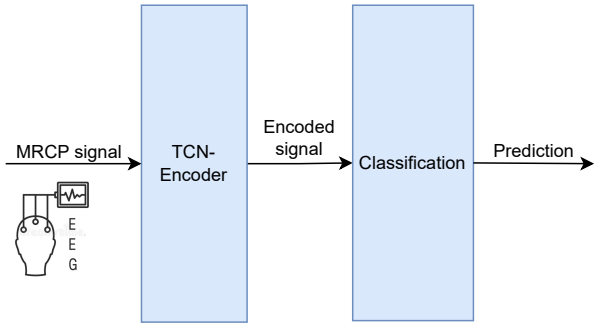

Figure 9. Overview of our classification method

Transfer learning via TCN

It is known that each individual has different unique patterns in their brainwaves, which makes it a challenge to have one pretrained model universally viable for use on multiple subjects without supervised training beforehand. But since all the inputs get normalized and even before normalization the data does not seem to be very far from a normal distribution, we also test validating the secondary subjects on the trained TCN, where the encoder and classifier were trained on the 15 trials from the main subject. Aside from the distribution and normalization, we also assume beforehand that even if we use data from another subject on the pretrained TCN, since both subjects did the same types of movement, the TCN might have captured some specific patterns for that action that are present in the brainwaves, but we are not aware of at a first sight. Aside from that, they will also serve as a good baseline to evaluate the GAN model.

These results will be compared both with the standalone single subject testing and classification, and also with the transfer learning results from GAN benchmarks.

Transfer learning via GAN

In this part we want to see if the GAN helps us create a new classifier for a "new" subject, based on the data from the main subject. By using the discriminator, the TCN-encoder will be refined, and after the training is done, the encoder will be extended with the pretrained classifier from the TCN trained on the main subject, thus creating a new classifier that did not need features-labeled data for training. In this case we are mainly interested to find out whether or not refining the TCN-encoder improves the classification.

4. Results and Discussion

In this section will we present the results from the benchmarks detailed in the previous section.

Individual classification of MRCP

Table 2 shows the standalone performance of our TCN, made from the TCN-encoder and the dense classifier. Datasets 1, 2 and 3 come from the main subject and datasets 4 and 5 from the secondary ones, but in this case it is irrelevant, as the TCN was initialized from scratch, fit with training data, and validated with test data. This process was repeated 10 times for each data set and an average was made.

Dataset	Accuracy	Loss
Dataset 1	0.61	0.67
Dataset 2	0.86	0.41
Dataset 3	0.86	0.36
Dataset 4	0.68	0.54
Dataset 5	0.86	0.35
Average	0.77	0.46

Table 2. Results from training and validating a new TCN with labeled data

Transfer learning

Table 3 shows the difference between the metrics of a model that was trained and validated with its own dataset, compared to only validating the dataset using a pretrained TCN from the main subject. The main subject consists of datasets 1, 2 and 3, and the first line in the table shows its standalone performance. Despite those 3 datasets were the training data for the model used, the results show how the system performs worse if it was not specifically trained for that dataset. This goes even further when we look at datasets 4 and 5 from the secondary subjects, where both accuracy and loss take an even bigger hit.

Despite that, these results serve as a good baseline for evaluating the performance of the GAN.

Dataset	Acc. Ind.	Acc. on TCN	Loss Ind.	Loss on TCN
Main subject	0.74	-	0.58	-
Dataset 1	0.61	0.51	0.61	0.81
Dataset 2	0.86	0.73	0.41	0.49
Dataset 3	0.86	0.84	0.36	0.38
Dataset 4	0.68	0.58	0.54	1.04
Dataset 5	0.86	0.51	0.35	1.13
Average	0.77	0.63	0.46	0.77

Table 3. Results from validating data sets on TCN without individual training (TCN) compared to individual training (Ind.)

Table 4 shows the performance of the refined TCN encoder from a GAN interfaced with the pretrained classifier, compared to just using the pretrained classifier on a new data set. We can see the loss was almost halved, and the accuracy was also improved by 5% for dataset 4 and 11% for dataset 5 respectively.

Dataset	Acc. TCN	Acc. GAN	Loss TCN	Loss GAN
Dataset 4	0.58	0.63	1.04	0.65
Dataset 5	0.51	0.62	1.13	0.63
Average	0.54	0.62	1.08	0.64

Table 4. Results from validating data sets through GAN-trained encoder compared to TCN without individual training

Finally, table 5 compares the earlier results from training and validating a TCN for one specific subject, as opposed to doing the same process using the GAN, without requiring labeled data. While the accuracy dropped with an average of 0.15 and the loss grew by 0.18, it is to be kept in mind that this latter process does not require labeling the data, and depending on the use case, there can be situations when the tradeoff is worth for all intents and purposes, as it is possible that some future data set will lack the EMG data, making labeling next to impossible.

Dataset	Acc. Ind.	Acc. GAN	Loss Ind.	Loss GAN
Dataset 4	0.68	0.63	0.54	0.65
Dataset 5	0.86	0.62	0.35	0.63
Average	0.77	0.62	0.46	0.64

Table 5. Comparison between GAN and TCN standalone individual training (Ind.)

Confusion matrices

When we are training a new TCN with its own labeled dataset, our testing showed that the distribution of false negatives and false positives is similar in most of the benchmarking runs. If its a model where the accuracy is lower, it can happen in some cases that the system learned to classify and tell appart better only 1 of the 2 movement types with a false positive/false negative or vice-versa ratio of up to 1 to 8.

An interesting trend occurs when we run a secondary dataset on the pretrained classifier. There seems to be a tendency to maximize the number of false positive, up to a 11 to 1 ration when compared with the false negatives.

5. Further Discussion

We can see from the results that our accuracy increases and the loss decrease when we create a classifier inferred from GAN, but as expected, its performance is still lower compared to having a standalone classifier trained for a specific subject, which goes to show once again just how different and unique our brains are. Despite that, we still believe the decrease in performance is a worth tradeoff, especially when considering how tedious labelling can become, or in some cases, depending on the recording setup, even impossible.

Regarding the tendency of maximizing the number of false positives in the case of validating a secondary dataset on the pretrained TCN, we believe it does make sense and we could even say it was partially expected. From our visual observations, if you take all the positives or the active movement phases from a dataset, they have a tendency to follow a pattern in their shape, collectively as a group, more often than their negative or rest phases. The active phases have a valley around their signal's minima point, meanwhile the rest phases have more of a stochastic data points distribution, and they differ one from the other to a higher degree than the active phases do. Because of this, there is a higher chance for a rest phase to be wrongfully classified as an active phase, increasing the false positives count, rather than the opposite happening, where an active phase would be classified as rest.

6. Future Work

While our results showed a good and consistent improvement when using the TCN encoder in combination with the GAN, there are still things that could be done to improve this cross-domain, transfer learning classifier creation even more. Since our individual training and validation of a TCN shows consistent differences of up to 25% between just 5 different datasets, a good first step would be to survey multiple individuals and perform recording sessions, in order to find someone with a great performance in a single training and validation using just the TCN. After such a person is found, multiple recording sessions may be made, and regarding the transfer learning part, the number of epochs and steps should be increased, and the learning rate decreased.

Acknowledgements

We thank the supervisors Dr. Dalin Zhang and Dr. Yan Zhao for the support and comments that aided this research, we thank Dr. Susan Aliakbary Hosseinabadi and Dr. Strahinja Dosen for offering access to the laboratory, equipment and domain knowledge, and we thank the participants who volunteered to do data acquisition sessions.

References

[1] Eric S. Donkor. Stroke in the 21(st) century: A snapshot of the burden, epidemiology, and quality of life. *Stroke research and treatment*, 2018:3238165–3238165, Nov 2018. 30598741[pmid].

[2] Christoph Guger, Vivek Prabhakaran, Rossella Spataro, Dean J. Krusienski, and Adam O. Hebb. Editorial: Breakthrough bci applications in medicine. *Frontiers in Neuroscience*, 14, 2020.

[3] Andrei I. Katona and Rasmus S. Krog. Exploring intention of movement detection in single-trials using movement related cortical potentials. 2022.

[4] Zhipeng He, Yongshi Zhong, and Jiahui Pan. An adversarial discriminative temporal convolutional network for eeg-based cross-domain emotion recognition. *Computers in Biology and Medicine*, 141:105048, 2022.

[5] Alexander J Casson. Wearable EEG and beyond. *Biomed Eng Lett*, 9(1):53–71, January 2019.

[6] M Hallett. Movement-related cortical potentials. *Electromyogr Clin Neurophysiol*, 34(1):5–13, January 1994.

[7] Jeffrey D. Meier, Tyson N. Aflalo, Sabine Kastner, and Michael S. A. Graziano. Complex organization of human primary motor cortex: A high-resolution fmri study. *Journal of Neurophysiology*, 100(4):1800–1812, 2008. PMID: 18684903.

[8] Krzysztof Strzecha, Marek Krakós, Bogusław Wiecek, Piotr Chudzik, Karol Tatar, Grzegorz Lisowski, Volodymyr Mosorov, and Dominik Sankowski. Processing of emg signals with high impact of power line and cardiac interferences. *Applied Sciences*, 11:4625, 05 2021.

[9] Fatemeh Karimi, Jonathan Kofman, Natalie Mrachacz-Kersting, Dario Farina, and Ning Jiang. Detection of movement related cortical potentials from EEG using constrained ICA for Brain-Computer interface applications. *Front Neurosci*, 11:356, June 2017.

[10] M.V. Valueva, N.N. Nagornov, P.A. Lyakhov, G.V. Valuev, and N.I. Chervyakov. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, 177:232–243, 2020.

[11] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. 2018.

[12] Murad Abu-Khalaf, Frank L. Lewis, and Jie Huang. Neurodynamic programming and zero-sum games for constrained control systems. *IEEE Transactions on Neural Networks*, 19(7):1243–1252, 2008.

Appendix

A. Figure of subject

Figure 10. One of the subjects during a trial

B. Comparison of raw and filtered EEG

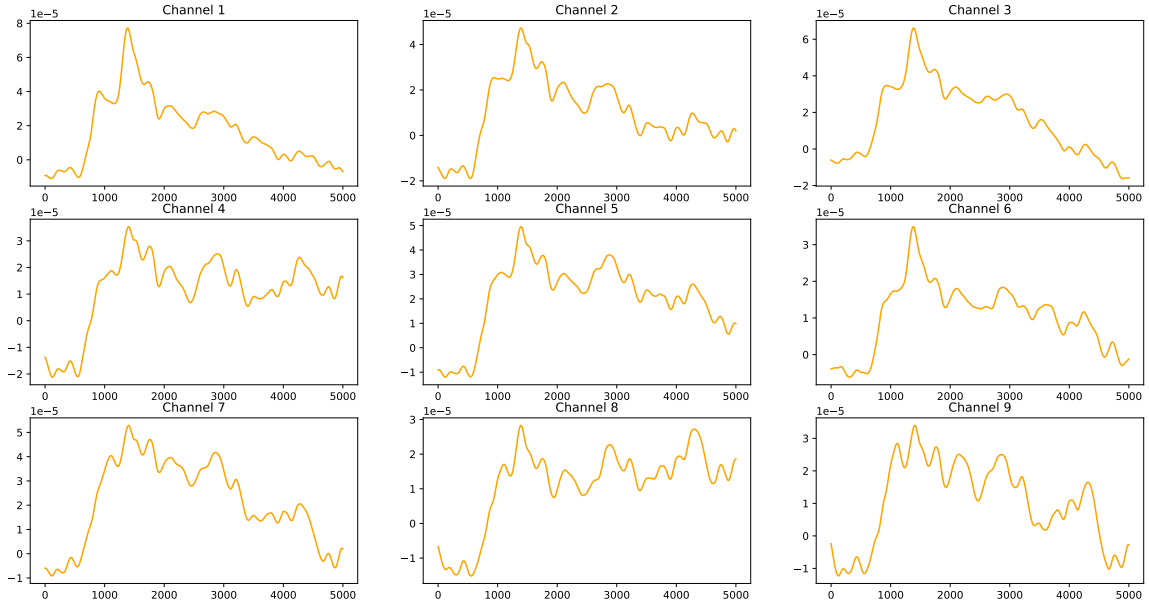

Figure 11. **Snippet of nine channels filtered EEG with band-pass filter using 0.05hz and 5hz**

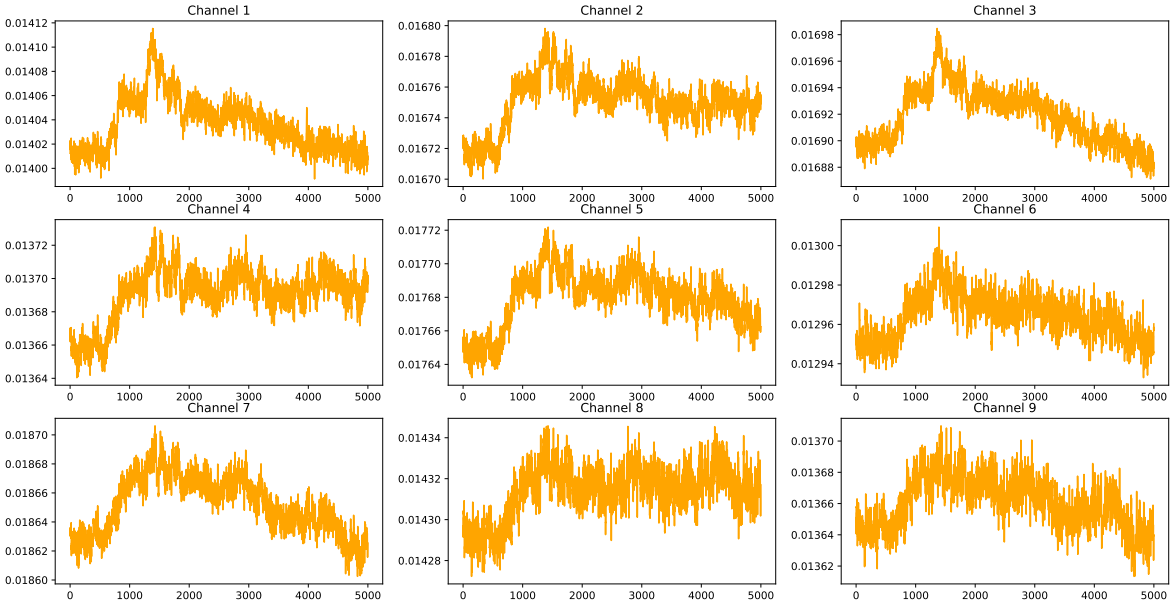

Figure 12. **Snippet of nine channels raw EEG**

C. Filtered EMG signal with detected peaks

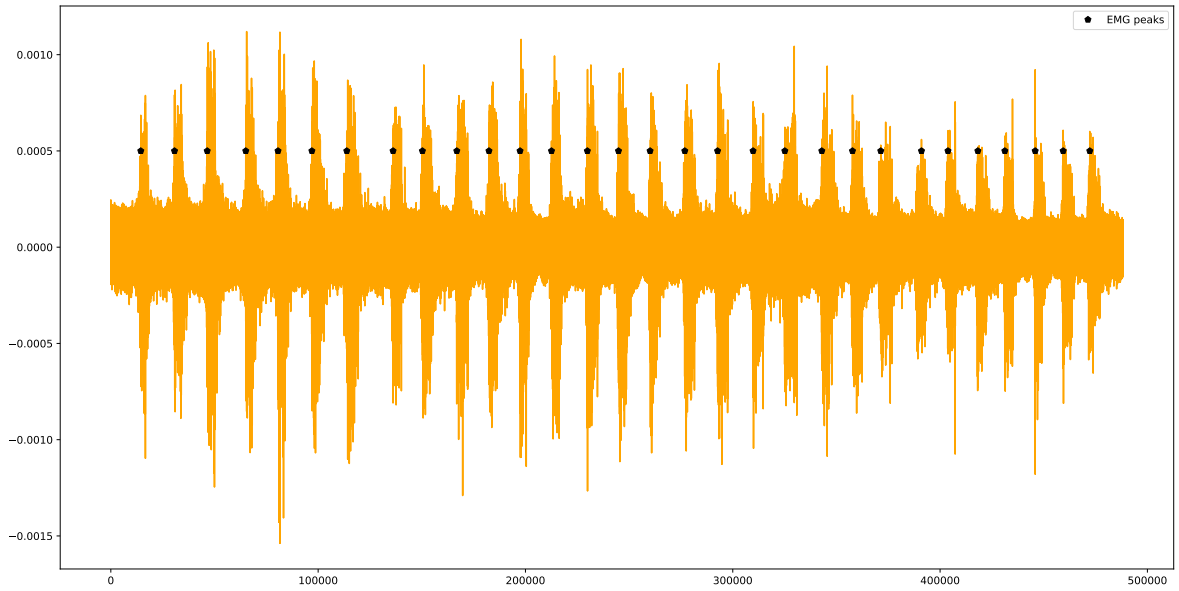

Figure 13. **Filtered EMG on a subject with their peaks marked by our algorithm**