Transformed-Based Myoelectric Decoding for Continuous Control of Prosthetic Fingers

Wolf De Wulf

Master of Science by Research

CDT in Biomedical AI

School of Informatics

University of Edinburgh

2023

Abstract

Hand prostheses are evolving to restore people with limb difference with intuitive, independent, continuous control of prosthetic fingers. These types of prostheses are typically controlled by machine learning algorithms that extract intent from electromyographic recordings of muscle activity. Prominent models are successful in extracting simple individual movements from such recordings, but under-perform when decoding multiple simultaneous and more delicate outputs, such as grasps. Neural and deep learning models are potential candidates to improve on this. Their flexibility makes them suitable for multi-output tasks and alleviates the need for handcrafted features. In this dissertation, I evaluate a deep neural model recently popularised for the processing of biomedical time series: the transformer. I compare a number of configurations, representing various approaches to input processing, model training, and architecture. The best-performing model outperforms previous work in a multi-output multi-class prosthesis control paradigm. With a shared core for multiple outputs, each representing a finger, the transformer is able to consider dependencies between its outputs to outperform models that consider them independent. Furthermore, I find that the model can identify muscle groups and that it can transfer across recording sessions. The offline nature of the analysis is a limitation of this research study. Nevertheless, the results suggest that transformer-like architectures can make the simultaneous extraction of multiple degrees of freedom from electromyographic muscle recordings practical. Future research on this topic is recommended.

Declaration

I declare that this document was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Wolf De Wulf*)

Acknowledgements

First and foremost, I want to thank my supervisors for this dissertation: Prof. Kia Nazarpour and Dr. Chenfei Ma. Both of them provided me with perfectly timed guidance and support throughout.

Furthermore, I would like to thank the CDT in Biomedical AI for providing resources and support, as well as the environment in which I was allowed to work on this project. Discussions with such an interdisciplinary group of colleagues boosted my productivity in a way I have not experienced before. In particular, Aryo Pradipta Gema deserves acknowledgement for always being available to answer my questions regarding transformers.

Lastly, I need to thank my partner. Although her influence on my time working on this project could be seen as more negative than positive, I prefer to see it as the latter. She keeps me social and rested, she pulls me away from my work when needed. As a researcher, I have come to appreciate this more than I ever expected I would.

This work was performed using resources provided by the Cambridge Service for Data Driven Discovery (CSD3) operated by the University of Cambridge Research Computing Service, provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/T022159/1), and DiRAC funding from the Science and Technology Facilities Council.

Contents

Chapter 1

Introduction

Limb difference can significantly decrease quality of life (Sinha et al., 2011). Patients speak of depression, anxiety, physical discomfort, phantom pains, difficulties in employment and social activities, and restricted mobility (Gallagher et al., 2011). The goal of a prosthesis is to restore to people with limb difference those aspects in which quality of life is reduced. With recent advancements in robotics and machine learning, modern prostheses have become advanced mechanical devices that come closer and closer to realising this goal.

An ideal prosthesis allows for independent, intuitive, continuous control of a variety of possible actions. The mechanical aspect of such a prosthesis is mostly in place. Robotic hands that can perform fine-grained, continuous movements exist and are constantly being improved (Piazza et al., 2019). In contrast, the control aspect is falling behind because of difficulties faced in the decoding of intent. A controllable prosthesis typically extracts user intent from electromyographic recordings of muscle activity, recorded using electrodes on the surface of the skin (sEMG; Roche et al., 2019). Traditional control algorithms look for simple patterns in such recordings, e.g. activation or deactivation (Vujaklija et al., 2016). Machine learning methods are able to extract more complex patterns from sEMG recordings, allowing for more intuitive control (Simão et al., 2019). Although these machine learning control paradigms have had their successes, finding their way into commercial adoption (Roche et al., 2019), substantial room for improvement remains.

One particular lacking of current machine learning approaches to prosthetic control is that they decode only a single gesture at a time and do so in a discrete manner. Various paradigms have been proposed to improve on this, ranging from multi-output classification to extract multiple gestures at once (Ortiz-Catalan et al., 2014), to using

regression to extract trajectories instead of gestures (Krasoulis et al., 2015). However, due to under-performance, few of these have resulted in effective real-time prosthesis control.

As decoding paradigms become more complex, the question arises whether the decoding models should follow. Most machine learning approaches to sEMG decoding are non-neural and are trained using handcrafted features, encoding the aspects of sEMG thought to be relevant (Simão et al., 2019). Recently, studies have started considering neural models and even deep learning architectures that can learn relevant representations of sEMG signals by themselves (Xiong et al., 2021). The flexibility of neural models makes them innately apt at multi-output tasks, whether classification or regression. Furthermore, with the possibility of sharing weights, they can consider dependencies between outputs in an intuitive way.

In this work, I investigate the effectiveness of a transformer-like model for extracting intention from sEMG recordings. The success of the transformer architecture in the field of natural language processing (Vaswani et al., 2017) has resulted in adaptations to various other research areas (Lin et al., 2022), sEMG decoding included (Montazerin et al., 2023; Rahimian et al., 2021; Zabihi et al., 2022). A comprehensive understanding of how transformer architectures can be leveraged in this field is yet to be established. A number of attempts have been proposed, each implementing a particular input format and certain architecture modifications. Here, I discuss the justifications for each of these design choices and combine the most principled into a transformer-like model that I evaluate in a modern multi-output sEMG decoding paradigm.

The results show significant increases in performance when compared to previous work with traditional machine learning models (Krasoulis & Nazarpour, 2020). Furthermore, they indicate that a transformer-like architecture can identify muscle groups and can learn sEMG representations that effectively generalise over recording sessions and to multiple outputs. A limitation of my work is that it consists of offline analyses, and various aspects of real-time prosthetic control are not considered. Furthermore, I only consider a single dataset, which comprises a limited number of participants and finger movements and was recorded in an isolated environment. Machine learning models are known to suffer from poor generalisation under different limb positions and/or muscle contraction levels and should thus be evaluated in terms of robustness to these factors (Fougner et al., 2011; Khushaba et al., 2016). Nevertheless, my findings show that the transformer architecture has generalisation capabilities that warrant further research addressing these limitations.

1.1 Objectives

My objectives in this dissertation are the following:

- *O.1: Establishing a high-level overview of the effectiveness of transformer-like architectures when employed for extracting intention from sEMG recordings.*
  - *What input formats work best?*
  - *How much data is needed to achieve reasonable performance?*
  - *Is cross-acquisition transfer feasible?*
  - *Which patterns does the model pick up on and can these be related to relevant biophysical phenomena?*
- *O2: Verifying whether a neural model with a shared core for multiple outputs can learn generalised representations of sEMG, comparing to models that unrealistically assume outputs to be independent.*

1.2 Structure

I contextualise my work in Chapter 2, which consists of two parts. The first part briefly discusses traditional and modern sEMG prosthetic control paradigms. The second part summarises transformer models for biomedical time series.

Chapter 3 discusses methodology; data, processing, the transformer model and its variations, as well as details regarding training and evaluation. In-depth mathematical explanations of the modules of the transformer are provided in Appendix A.

Chapter 4 reports the results. I compare a multitude of baselines and variations of the transformer-like architecture in a multi-class multi-output classification paradigm. I then mix training data from two recording sessions separated by a ten minute break to investigate if the transformer-like model can generalise despite the nonstationary nature of sEMG.

In Chapter 5, I frame the capabilities of the transformer in both the movement decoding and digit action decoding paradigms. I further discuss cross-acquisition transfer, certain spatial and temporal patterns the model can identify, and time and memory complexity considerations.

In Chapter 6, I conclude the dissertation with a summary of my work. I also discuss future prospects and possible research routes for transformer-like architectures for sEMG prosthetic control.

Chapter 2

Background

2.1 Electromyographic Prosthetic Control

Traditional clinical solutions for upper-limb prostheses such as prosthetic hands typically implement amplitude-based control algorithms (direct control; Ison & Artemiadis, 2014). These algorithms use the activity of pairs of antagonist muscles, recorded with pairs of electrodes above the amputation. If sufficient activity is detected, a certain function, e.g. opening or closing the hand, is activated. To allow for more degrees of freedom (DOF), the user can switch between functions via a certain trigger signal, e.g. muscle co-contraction or physical buttons (Vujaklija et al., 2016). While such an approach is robust, it limits control and can be cumbersome and non-intuitive to use, eventually leading to prosthesis rejection (Salminger et al., 2022).

2.1.1 Machine Learning Approaches

Recent advancements in robotics and machine learning (ML) bring opportunities to improve on traditional prosthetic control. If ML control algorithms can be designed to capitalise on the mechanical capabilities of modern day prostheses, user satisfaction is expected to improve drastically. A typical approach consists of using a classification model to map recordings extracted from multiple sEMG channels onto a certain function, e.g. a hand grasp. This simple but effective control paradigm has been highly successful and has found its way towards commercial adoption (Roche et al., 2019).

Various models for extracting sEMG features and/or mapping them to functions have been proposed. Our reasonable understanding of what sEMG signals represent has led most approaches to be ML approaches, where features of the sEMG signals

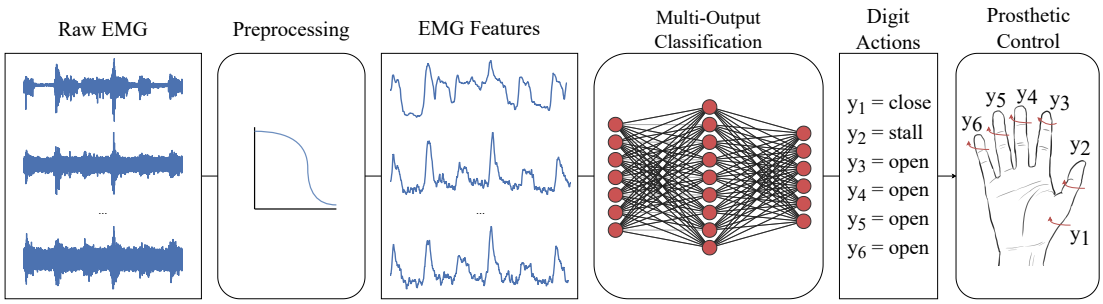


Figure 2.1: Visualisation of the digit action decoding paradigm. Raw sEMG from multiple sensors is processed and the resulting features are used to train a multi-output multi-class classifier. Each output corresponds to one degree of freedom (here finger movements) and can take on three possible values: open, close, or stall. Predictions can be used to control hand prostheses. Figure after Krasoulis and Nazarpour (2020).

are designed by hand (Simão et al., 2019). Recently, however, deep learning (DL; LeCun et al., 2015) approaches that are able to extract high-level abstract features automatically, have become popular as well (Xiong et al., 2021).

2.1.2 Digit Action Decoding

Ultimately, the goal of prostheses is to allow for simultaneous, continuous, and intuitive control of multiple DOFs, restoring people with limb difference with as much functionality as possible. Although pattern finding control algorithms proved a significant improvement compared to direct control, they usually still only allow for a single DOF to be controlled in a discrete manner. Various approaches have been proposed to address this limitation. Multiple functions can be extracted simultaneously, in a multi-output classification paradigm (Ortiz-Catalan et al., 2014; Wurth & Hargrove, 2014). Continuous control can theoretically be achieved via regression-based methods to estimate wrist or finger kinematics (Hahne et al., 2018; Krasoulis et al., 2015). Nevertheless, few of these studies have provided people with limb difference with real-time control of prosthetic fingers (Cipriani et al., 2011; Krasoulis et al., 2019).

Krasoulis and Nazarpour (2020) propose *digit action decoding* to improve continuous, independent control of prosthetic fingers. Figure 2.1 visualises the process. Unlike full-resolution regression on a continuous sEMG recording, digit action decoding aims to extract intent from multiple digit-related DOFs from short windows of the signal. For each DOF, the windows are classified into three classes: open, close, or stall. Krasoulis and Nazarpour reason that replacing the continuous output variables with dis-

crete ones, and thus moving from multi-output regression to multi-output multi-class classification, should simplify the decoding part of the pipeline.

In an initial study, Krasoulis and Nazarpour (2020) evaluate the digit action decoding paradigm in an offline analysis, where they show that it is feasible to extract individual digit actions from sEMG signals. In a first set of evaluations, they unrealistically consider the finger DOFs to be independent and train traditional ML models per DOF. Subsequently, they attempt to leverage the dependencies between finger movements by implementing multi-output classification using classifier chains (CC; Read et al., 2011).

A CC model consists of a classifier for each DOF. Before a CC is trained, an order over the DOFs is established. The first classifier in the chain is trained for the first DOF on the input features only. Subsequently, the second classifier is trained for the second output on the input features and the ground truth labels of the first DOF. This process is repeated, each time including the ground truth labels of all the previous DOFs. Inference with a CC model follows the same procedure, including predictions at each step instead of ground truth labels. A popular variant of CCs ensembles a collection of CCs trained with random orders over the DOFs.

Although CC models can take into account dependencies between DOFs, Krasoulis and Nazarpour observe performance not statistically discernible from that of models that consider the outputs to be independent. Presumably, neural models with a certain proportion of shared parameters in early layers can improve on this. With appropriate output layers, such models are innately able to model multiple dependent output variables.

2.2 Transformers for Biomedical Time Series

The transformer is a DL architecture originally designed for sequence-to-sequence tasks (Vaswani et al., 2017). By replacing recurrence with the attention mechanism, the transformer addresses the long-range dependency problems Recurrent Neural Networks (RNN; Kolen & Kremer, 2001) and Long Short-Term Memory (LSTM; Hochreiter & Schmidhuber, 1997) models suffer from, thereby also allowing for more parallel computation (Kaplan et al., 2020). In-depth explanations of the components that make up the transformer architecture can be found in Appendix A.

Since their introduction, transformer-like architectures have been adapted to various domains distinct from Natural Language Processing (NLP), for which they were

originally intended. Successful examples can be found in computer vision (Dosovitskiy et al., 2021) and protein structure prediction (Jumper et al., 2021). Recently, several attempts have been made at training transformer-like architectures on time series (Wen et al., 2022). The ability to model long-range dependencies and interactions in sequential data makes transformers appealing for time series modelling tasks such as forecasting (Zhou et al., 2022), anomaly detection (Tuli et al., 2022), and classification (Zerveas et al., 2021). Advances for each of these tasks can have meaning in a biomedical context. Transformer-like models have been proposed for predicting seizures from electroencephalographic recordings (EEG; Hussein et al., 2022), detecting arrhythmia from electrocardiographic recordings (ECG; Hu et al., 2022), and classifying sEMG recordings into hand movements to steer prostheses (Rahimian et al., 2021; Zabihi et al., 2022).

Approaches of porting the transformer architecture to the processing of time series range from using it as originally designed, to adapting various aspects of it to the task at hand. Such adaptations can be low-level, i.e. on the module level, or high-level, i.e. on the architecture level. The following few sections discuss considerations that can be made when using a transformer-like architecture for the processing of biomedical time series, focusing on the decoding of sEMG signals.

2.2.1 Input Processing

Since the transformer is a DL architecture, the preprocessing of time series inputs should be kept minimal. The point of having a deeper model is that it can learn on its own which features are most relevant to the task at hand (LeCun et al., 2015). However, oscillatory and highly nonstationary signals such as EEG and sEMG are generally too complex for any model to learn from directly. For such signals, the task becomes simplifying them in such a way that models can learn from them, but not beyond that. Relevant nuances in the signals should remain, in the hope that the DL architecture can learn to use them.

2.2.1.1 Envelope

In the context of sEMG, a prevalent method for simplifying the signal is to calculate its envelope. Firstly, we clean up the signal by getting rid of powerline noise using a notch filter and unwanted frequencies using a band-pass filter (Chowdhury et al., 2013). Then, to calculate the envelope, we rectify the signal and pass it through a low-

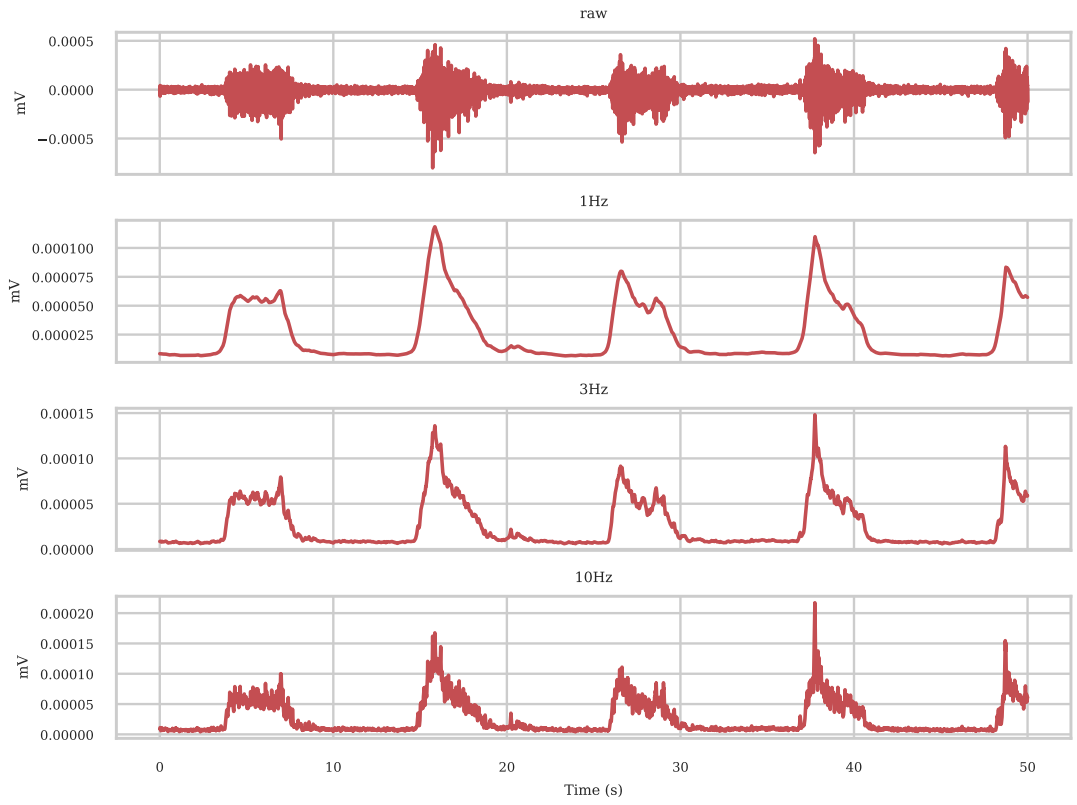


Figure 2.2: A segment of raw sEMG with its envelope at different cut-off frequencies. The envelope is calculated by rectifying the signal (taking the absolute value) and then passing it through a low-pass Butterworth (1930) filter. Choosing a cut-off frequency of 1 Hz results in more smoothed signals. When the cut-off frequency is higher, for example 3 Hz to 10 Hz, the signals retain more oscillatory nuances.

pass filter with an upper bound of 10 Hz or lower. Figure 2.2 depicts an example of the before and after of this process. Other methods for calculating the envelope of a signal exist, e.g. computing the root mean square for a window that slides across the signal (Konrad, 2005). However, such methods tend to downsample the signal to a single value per window, which is undesirable for models such as the transformer.

This way of processing sEMG signals was first tested for training convolutional neural networks (CNN; Atzori et al., 2016), and afterwards adopted for the training of transformer-like models (Rahimian et al., 2021). A consideration for calculating sEMG envelopes using a low-pass filter is the cut-off frequency. If we keep higher frequencies, more oscillatory nuances remain. Attempts at training transformer-like architectures on sEMG signals in literature typically opt for a 1 Hz cut-off, simplifying the envelope as much as possible (Rahimian et al., 2021; Zabihi et al., 2022).

2.2.1.2 Normalisation

When training ML models, it is common practice to normalise the input features, i.e. mapping their values to $[0, 1]$, or transforming them such that they follow a distribution with zero mean and unit variance. For sEMG envelopes, we typically compute the minimum and maximum values of the training data to then map them to $[0, 1]$:

$$x'(t) = \frac{x(t) - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \tag{2.1}$$

We can use the same minimum and maximum values to normalise new sEMG recordings, aligning them with the data the model is trained on. Reducing the distributional shift between recordings is especially important for nonstationary signals such as EMG.

Rahimian et al. (2020) observe that min-max normalisation does not take into account that a high percentage of useful information in the sEMG signals lie close to 0. If we map sEMG envelopes to $[0, 1]$ in a linear manner, a lot of that information can be

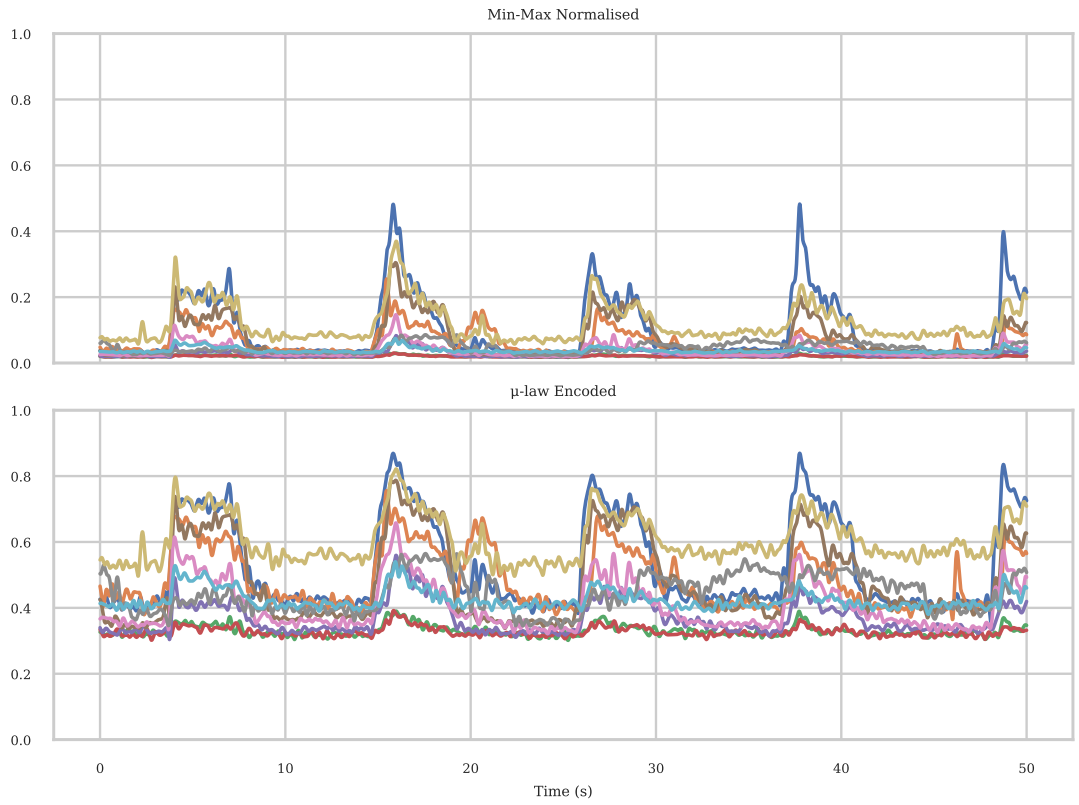


Figure 2.3: The envelopes of 10 sensors recording sEMG signals. On top, when min-max normalised. At the bottom, when μ-law encoded. The μ-law encoding amplifies signals that lie close to zero in a logarithmic manner.

lost to a DL model. Therefore, they propose to use the μ-law encoding as a normalisation technique that normalises envelopes in a logarithmic manner. The μ-law encoding is traditionally used in speech to reduce the range an audio signal spans (TU-T, 1988). The transformation is as follows:

$$x'(t) = \text{sign}(x(t)) \frac{ln(1+\mu|x(t)|)}{ln(1+\mu)}, \tag{2.2}$$

where μ denotes the number of quantisation channels and is often chosen as 255. Figure 2.3 depicts 10 sEMG channels when min-max normalised and when normalised using the μ-law encoding. Those sensors whose values are near zero in the min-max normalised signals are amplified in the μ-law encoded signals. The scale of values that were not close to zero is similar. Multiple studies report increased performance when using the μ-law encoding (Rahimian et al., 2021; Rahimian et al., 2020).

2.2.2 Domain of Attention

Information in biomedical time series is generally encoded in three domains: the time domain, the frequency domain, and the spatial domain. Depending on the signal and the task, it might be beneficial to apply the attention mechanism (see section A.2 in Appendix A) in one or more of these domains.

2.2.2.1 Frequency Attention

Brain signals such as EEG are known to encode important information in the frequency domain (Saby & Marshall, 2012; Tsipouras, 2019). The time series data is typically converted, from the time domain to the frequency domain, using some form of the Fourier transform (FT; Bracewell & Bracewell, 1986). The attention mechanism is then applied to sequences of vectors that contain frequency information for segments in the time series signal, i.e. attention is applied across the time domain, in the frequency domain (Cai et al., 2022; Hussein et al., 2022).

2.2.2.2 Temporal Attention

For sEMG, we typically only consider the frequency domain when trying to quantify muscle fatigue (Viitasalo & Komi, 1977). In contrast, for decoding movements from sEMG recordings the time domain is considered more informative (Tkach et al., 2010). In this context, it is usually the case that multiple sensors record sEMG from different areas, in an attempt to target various muscles. Approaches that implement the attention

mechanism tend to apply it across the spatial domain, in the time domain (Rahimian et al., 2021). In such approaches, the positional encodings of the transformer (see Section A.1) relate to the positions of the electrodes. With fixed positional encodings, the actual electrode positions must be accounted for in the encoding scheme. On the other hand, when the positional encodings are learnt they can indicate if the model is able to separate electrodes and, by extension, muscle groups.

2.2.2.3 Spatio-Temporal Attention

Considering temporal and spatial information independently is a limitation of traditional sEMG feature extraction methods (Jabbari et al., 2021). Prominent DL approaches address this limitation by combining CNNs and LSTMs to capture the spatio-temporal characteristics of sEMG signals (Y. Wu et al., 2018; Xia et al., 2018). More recent work proposes transformer architectures that implement spatio-temporal attention as a solution (Y. Wang et al., 2023). This type of attention segments input sEMG windows into patches which are flattened over the electrode dimension. The attention mechanism is thus applied across the time domain, in the spatio-temporal domain. For spatio-temporal attention (and frequency attention) the positional encodings of the transformer relate to positions in time. When such positional encodings are learnt they can indicate if the model captures the continuous trajectories in its training samples.

2.2.2.4 Combinations

At the cost of more parameters, above types of attention can be combined. For example, Zabihi et al. (2022) implement a two-channel transformer-like architecture that implements temporal and spatio-temporal attention in parallel. Their model consists of two transformers, one applying the former and the other the latter. The outputs of the two are summed or concatenated and passed to a Multi-Layer Perceptron for further processing (MLP; McCulloch & Pitts, 1943). Zabihi et al. observe increased classification accuracy when classifying sEMG into hand gestures using the two-channel model. The catch with this approach is that the number of weights are almost doubled.

A different approach applies temporal and spatio-temporal attention in sequence (Y. Wang et al., 2023). Inputs are firstly embedded using temporal attention, the resulting embeddings subsequently go through spatio-temporal attention. This approach requires only a marginal increase in parameters and allows for simultaneous processing of features that have been contextualised with both types of attention.

Chapter 3

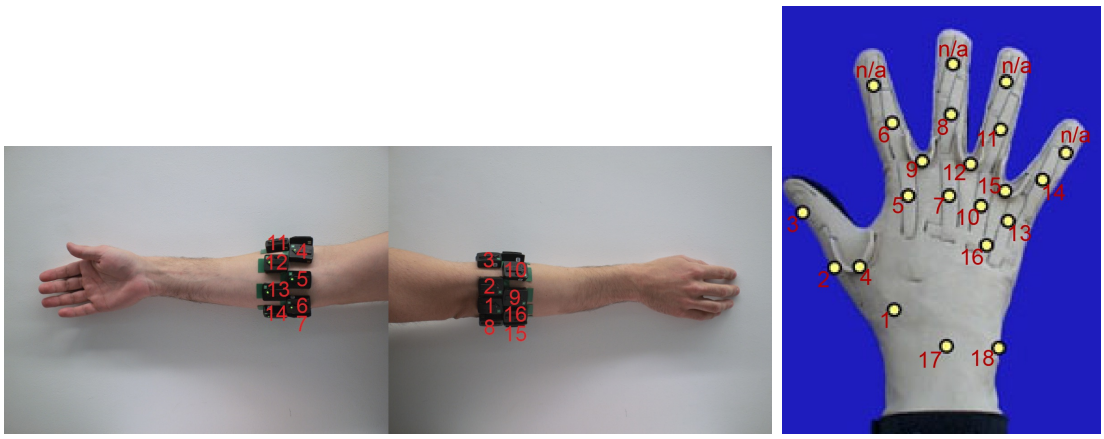
Methodology

3.1 Data

The data in this work comes from the Non Invasive Adaptive Prostheses database (Ninapro; Atzori et al., 2014), a publicly available multimodal database established to foster research on machine learning control systems of prosthetic hands. I use its eighth dataset (DB8; Krasoulis et al., 2019), which is specifically aimed at the estimation of finger movement. A crucial aspect for this focus is that participants wear a 'dataglove' that records finger kinematics. DB8 is the dataset that Krasoulis and Nazarpour (2020) originally used to evaluate their digit action decoding control algorithm, hence this dataset will allow comparisons with the traditional ML models Krasoulis and Nazarpour investigated.

3.1.1 Acquisition Protocol

DB8 consists of sEMG recordings of 12 participants performing nine movements, including single-finger as well as functional movements. Two of the 12 participants are people with limb difference, with right-hand transradial (below elbow) amputations. Participants sit as visual cues on a monitor indicate when they should perform which movement. Participants are equipped with two pieces of recording hardware. Firstly, 16 sEMG electrodes (12 and 13 for the participants with limb difference, respectively) are positioned in two rings of eight around the right forearm, as can be seen in Figure 3.1a. No specific muscles are targeted. Secondly, participants wear a dataglove that records their left (i.e. the other) hand and finger kinematics from 18 DOFs, as can be seen in Figure 3.1b. Further hardware details are given by Krasoulis et al. (2019).



(a) EMG sensor placement (b) dataglove

Figure 3.1: Ninapro DB8 data acquisition hardware and placement. On the left, the sEMG electrodes are positioned in two rings of eight around the right forearm. The rings are shifted with respect to each other. On the right, the dataglove records hand and finger kinematics from 18 DOFs. Pictures by Krasoulis et al. (2019).

During the acquisition, participants repeat nine movements using both hands, as can be seen in Figure 3.2. The duration varies between 6 s to 9 s. Consecutive trials are separated by 3 s of rest. Trials start by participants holding their fingers in the rest state (i.e. movement 0). Mirroring an example on the monitor, they gradually move their fingers to reach the target movement, subsequently returning to the rest state before the end of the trial.

For each participant there are three acquisitions, intended as training, validation, and testing splits. The training and validation splits comprise ten repetitions of each movement, while the testing split contains only 2 repetitions. The acquisitions were all recorded in the same session, with ten minutes of rest in between.

3.1.2 Processing

The sEMG signals are recorded at a 2 kHz sampling rate. We band-pass filter them, preserving the 10 Hz to 500 Hz range using a fourth-order digital band-pass Butterworth (1930) filter. This frequency range is widely accepted to be the range wherein most sEMG activity happens (Komi & Tesch, 1979). Subsequently, we compute the envelopes of the signals via a first-order digital low-pass Butterworth (1930) filter. We use 1 Hz cutoff, simplifying the envelopes as much as possible. We then min-max normalise the envelopes to $[0, 1]$ via formula (2.1), using the minimum and maximum

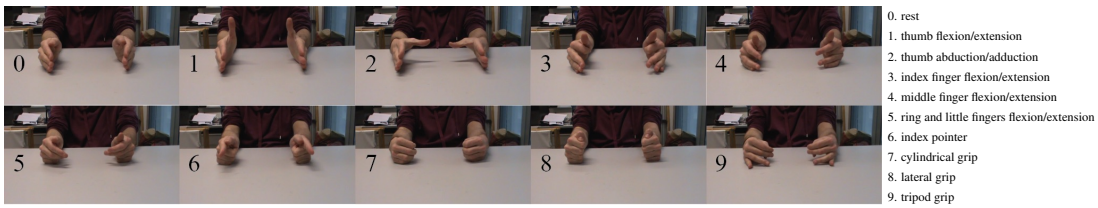


Figure 3.2: Ninapro DB8 movements. Movements 1 to 5 are single-finger movements, for example flexing/extending the index finger. Movements 6 to 9 are functional movements, for example cylindrical grip, mimicking grasping a bottle.

across all sensors of the training dataset (i.e. the first acquisition). Then, we transform the normalised envelopes using the μ-law encoding, amplifying the values close to zero. Since the envelopes are already in $[0,1]$, the sign function and absolute value in formula (2.2) can be dropped, yielding:

$$x'(t) = \frac{ln(1+\mu x(t)}{ln(1+\mu)}, \tag{3.1}$$

with $\mu = 255$. Finally, we segment the signals using a 128 ms sliding window with a 50 ms stride (i.e. around 60 % overlap), conforming to the processing of Krasoulis and Nazarpour (2020). This results in approximately 25×10^3 windows for the training and validation acquisitions and approximately 5×10^3 windows for the testing acquisitions.

A number of baseline models I will compare to require sEMG features, instead of time series envelopes. Therefore, parallel to the envelopes, we compute a set of 18 features from the signals that are band-pass filtered and windowed as before. The features are the Wilson amplitude, log variance, Hjorth (activity, mobility, and complexity; 1970), kurtosis, fourth-order autoregressive coefficients, waveform length, and skewness, conforming to the analysis of Krasoulis and Nazarpour (2020). I chose to extend this set with six spatio-temporal features designed by Samuel et al. (2019), which have been found effective in recent DL-based sEMG decoding work (Jabbari et al., 2021).

3.1.3 Labels

Each window is associated with two types of labels. On the one hand, we have a movement label, corresponding to the movements in Figure 3.2. These are recorded within the dataset and do not require any processing. On the other hand, we have the finger (or digit) actions, as defined by Krasoulis and Nazarpour (2020). We extract these labels from the finger kinematics recorded by the dataglove, as follows. Firstly, we linearly transform the 18 DOFs of the dataglove (see Figure 3.1b) into joint angles

Table 3.1: Dataglove to digit DOF transformation matrix. As an example, consider ring flexion, which is defined here as a linear combination of 0.3333 times the tenth dataglove DOF and 0.6666 the eleventh dataglove DOF. See Figure 3.1b for a visualisation of the dataglove DOFs. The matrix was copied from the supplementary material of the work of Krasoulis et al. (2019).

DOF	thumb rotation	thumb flexion	index flexion	middle flexion	ring flexion	little flexion
1	0.639	0	0	0	0	0
2	0.383	0	0	0	0	0
3	0	1	0	0	0	0
4	-0.639	0	0	0	0	0
5	0	0	0.4	0	0	0
6	0	0	0.6	0	0	0
7	0	0	0	0.4	0	0
8	0	0	0	0.6	0	0
9	0	0	0	0	0	0
10	0	0	0	0	0.3333	0
11	0	0	0	0	0.6666	0
12	0	0	0	0	0	0
13	0	0	0	0	0	0.3333
14	0	0	0	0	0	0.6666
15	0	0	0	0	0	0
16	-0.19	0	0	0	0	0
17	0	0	0	0	0	0
18	0	0	0	0	0	0

for the following six DOFs: thumb rotation, flexion/extension of the thumb, index, middle, ring, and little digits. Krasoulis et al. (2019) identified and verified a transformation matrix that achieves this empirically. We use the exact same matrix here, given in Table 3.1. Secondly, we min-max normalise the resulting six DOF angle trajectories to $[0, 1]$, using the minimum and maximum of the training acquisition. Thirdly, we again simplify the normalised trajectories by calculating the envelopes in the same way as for the sEMG signals, using a 1 Hz cutoff. At this point, we have smooth angle trajectories for all six DOFs. We can compute the digit actions from these by, firstly, estimating joint velocities through computing the first-order differences of the trajectories, and then, thresholding these velocities using tolerance $\varepsilon = 0.004$. We assign velocities larger than ε the 'close' label, velocities smaller than $-\varepsilon$ the 'open' label, and velocities in $[-\varepsilon, \varepsilon]$ the 'stall' label, corresponding to no movement. Joint angles that are less than 7.5 % away from either boundary (i.e. 0 or 1, corresponding to fully open or fully closed), we assign the labels corresponding to that boundary, regardless of the velocities.

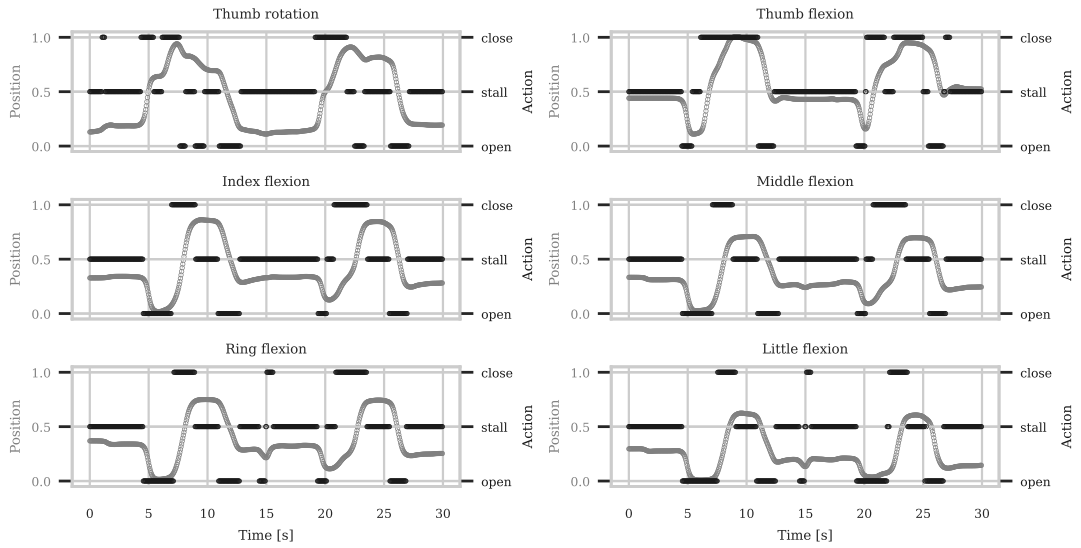


Figure 3.3: An example of how digit action labels are calculated from the six position trajectories. The joint angle trajectories for each DOF (grey traces, left y-axes) are normalised between zero (i.e. fully open) and one (i.e. fully closed). The velocities of the trajectories are estimated and transformed into the digit action labels by thresholding (black traces, right y-axes). This excerpt uses the data from the third acquisition of the first participant of DB8, the participant is performing the seventh movement, i.e. cylindrical grip (see Figure 3.2). Figure after that of Krasoulis and Nazarpour (2020).

The above procedure follows exactly that of Krasoulis and Nazarpour (2020), ensuring comparability. Figure 3.3 visualises an example of the procedure. Both types of labels are highly imbalanced. Approximately 75 % of the digit actions are 'stall', where the remaining 25 % is evenly divided across the 'open' and 'close' labels. This is the case consistently across participants. In contrast, the percentage of rest state labels varies significantly per participant, with a minimum of 47 % and a maximum of 66 %. The respective remaining movement labels are generally evenly divided across the nine non-rest movements.

3.2 Model

Figure 3.4 visualises the transformer-like architecture I evaluate in this work. The model is similar to that of Zabihi et al. (2022), with temporal and spatio-temporal attention in parallel. Data flows through two transformers, denoted 'TNET' and 'FNET', applying temporal and spatio-temporal, respectively.

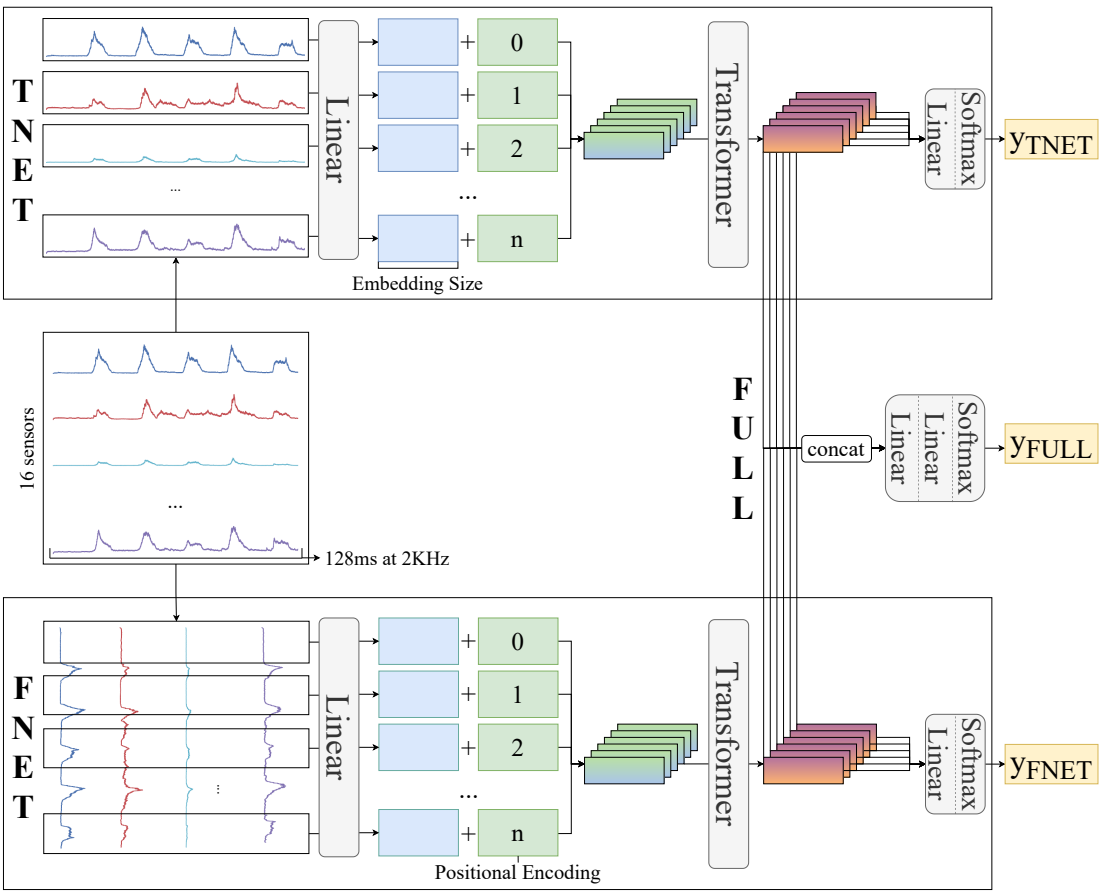


Figure 3.4: Visualisation of the transformer-like architecture. Data from 16 sEMG electrodes flows through the model in two possible ways. The TNET receives one token per electrode and applies temporal attention over the linear envelopes. The FNET receives one token per time patch and applies spatio-temporal attention over segments of the linear envelopes for each electrode. The patches are always of length 16, corresponding to the number of electrodes. With 128 ms at 2 kHz, this results in 16 tokens of 16 values per electrode. In both cases, learnt positional encodings are added to the tokens after they are linearly transformed to the model embedding size. Positionally encoded tokens (green-blue) then go through a number of transformer-encoder blocks, yielding contextualised output tokens (pink-orange). When the TNET or FNET are considered individually these then go through appropriate linear and softmax layers yielding class probabilities. When they are combined into the FULL model, the output tokens are concatenated and passed through two linear layers followed by a softmax layer, yielding class probabilities that account for both types of attention.

We transform input windows into vectors of a general model embedding size via a linear layer. We add learn positional encodings to the resulting tokens (see Sec-

tion A.1). The tokens then go through a traditional transformer-encoder consisting of a sequence of encoder blocks (see Section A.4). Depending on the control paradigm, and the DOFs it implements, we select a certain number of output tokens. In the case of the traditional movement decoding paradigm, we only keep the first output token, for the digit action decoding paradigm, we keep the first six output tokens. The output tokens then go through a linear layer followed by a softmax layer to result in normalised class probabilities. In the digit action decoding paradigm, since the classes are the same for all digits, the linear layer is shared across digits. This forces more computation into the transformers, which is generally desirable.

The two transformers can be seen as separate models. However, Zabihi et al. (2022) reason that a combination of both is more powerful. To combine them, they pair-wise sum the selected output tokens of both channels, yielding a new set of tokens that can be passed through separate linear and softmax layers. The combined model is trained jointly, by backpropagating the sum of the three losses. I opt here for a more recent approach, after the work of Montazerin et al. (2023), who see the two transformers as independent feature extractors and train and optimise them independently. The combined FULL model then freezes both channels, concatenates their output tokens, and only trains the output layers, which are extended with another feed-forward linear layer. While the first approach can be said to encompass the second approach, it requires training twice as much parameters at once. Since training data is limited, this can result in underfitting, yielding performance lower than achieved when the parts of the model are trained individually. I chose to not put the two types of attention in sequence for the same reason. With only around 25×10^3 training samples, we should keep the number of model parameters as small as possible.

3.3 Training

We train the model using mini-batch gradient descent (LeCun et al., 2015) with the sum of cross-entropy losses over all outputs. I opted for L_2-regularisation (Krogh & Hertz, 1991) and dropout to reduce overfitting (Srivastava et al., 2014). We train each configuration on the training acquisition for 40 epochs. Every epoch, we compute performance on the validation acquisition. We evaluate the instance that achieves the highest validation performance during the 40 epochs on the testing acquisition.

I implemented everything in *PyTorch* (Paszke et al., 2019) and made the source code available on GitLab (see Appendix B). Read Appendix C for further details.

3.4 Hyperparameters

The model has a number of hyperparameters. Three of them relate to the training process: batch size, learning rate, and weight decay. The remaining five are inherent to the transformer architecture (see Appendix A): embedding size, number of attention heads, number of encoder blocks, the activation function, and dropout rate. We look for good values for these hyperparameters through 20 hyperparameter optimisation (HPO) trials, guided using tree-structured parzen estimators (TPE; Bergstra et al., 2011). Further details regarding my HPO approach are provided in Appendix D.

3.5 Evaluation

3.5.1 Metrics

Evaluating the single-output movement decoding models can be done using traditional multi-class classification metrics such as classification accuracy and the confusion matrix. In contrast, multiple outputs, each of which can take on multiple classes, introduce considerations such as class imbalance and output dependencies. For the control of the digits of a prosthetic hand, the DOFs are generally not independent and, depending on the movements, certain outputs can be seen as more important. Since this is an offline analysis, the goal is to get a general idea of the capabilities of the model, and comparing it to existing approaches. I thus evaluate the model using performance measures typical of an offline analysis (Krasoulis & Nazarpour, 2020), leaving measures more relevant for an online analysis for future work (Krasoulis & Nazarpour, 2022).

An example of such a measure is the exact match ratio, i.e. the percentage of samples for which all outputs are correctly classified. The exact match ratio is very strict, increasing in strictness with the number of outputs. While relevant for an online analysis, the model in this work is not yet in that stage of the research pipeline. An alternative is the hamming score, i.e. the number of correctly classified labels to the total number of labels, across outputs. This metric is less strict than the exact match ratio, but highly sensitive to imbalance in the data. Since the class labels are significantly skewed towards the ‘stall’ class, the hamming score will generally be high for models that keep all digits from moving and low for models that can correctly predict digit actions but sometimes moves them when it is not supposed to.

Remaining options are metrics typically used for single-output tasks, but averaged

across outputs and classes. Examples are precision, recall, and F1-score, which is the harmonic mean of the former two. These metrics suffer less from imbalance in the data and are less strict than the exact match ratio. They are also the metrics that Krasoulis and Nazarpour (2020) use, with F1-score as main performance measure. To maximise comparability, I follow their approach.

Averaging over outputs and classes provides an overall metric of the model's performance. To get more detailed insights into its strengths and weaknesses I will analyse these metrics per class and per output. This can be done by computing confusion matrices independently for each output. Although such confusion matrices do not show where the shared-core model achieves additive improvements, they can inform us on which outputs are harder to decode than others.

3.5.2 Baselines

In the movement decoding paradigm, I compare to two baselines. Firstly, a model that always predicts the rest movement, denoted 'All rest'. Secondly, a Linear Discriminant Analysis model (LDA; Izenman, 2008) trained on the sEMG features (see Section 3.1.2).

Similarly, in the digit action decoding paradigm, I compare to a model that predicts the 'stall' class for every input and DOF, denoted 'All stall'. As well as to an LDA model trained on the sEMG features, independently for each DOF. Additionally, I compare to a model that applies attention across the spatial domain, in the domain of the sEMG features, denoted 'FTR'. The FTR configuration represents a small transformer that does not compute its own features from the raw sEMG data, but has to work with the handcrafted features. Thereby, it can yield insight into the benefits of training a transformer model with linear envelopes, i.e. allowing it to calculate its own features.

3.5.3 Comparison

Since results come per participant, I always compare the performance of groups of models using a paired t-test with significance level $\alpha = 0.05$ (Ross & Willson, 2017). I correct for multiple comparisons using the Bonferroni correction (Bland & Altman, 1995).

Chapter 4

Results

4.1 Movement Decoding

Figure 4.1 gives the classification accuracy of the two baselines and the transformer models, in the movement decoding paradigm, i.e. models classify inputs into a single movement (see Figure 3.2). LDA achieves the highest median classification accuracy, but is not statistically discernible from the FULL and FNET configurations. All three of these configurations have outlier participants. The TNET configuration seems to

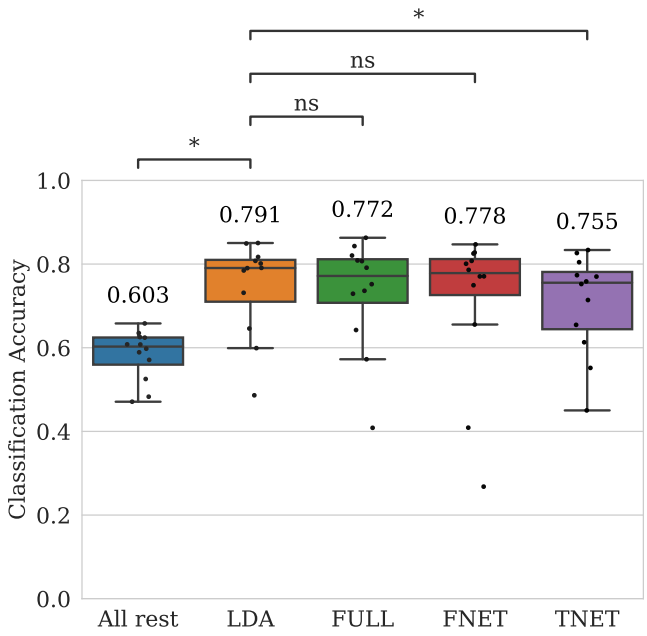


Figure 4.1: Comparison of movement classification accuracy for baselines and transformers. The median is given above each box-plot. Statistical comparisons are performed only against the best-performing algorithm; asterisk: $p < 0.05$, 'ns': $p > 0.05$.

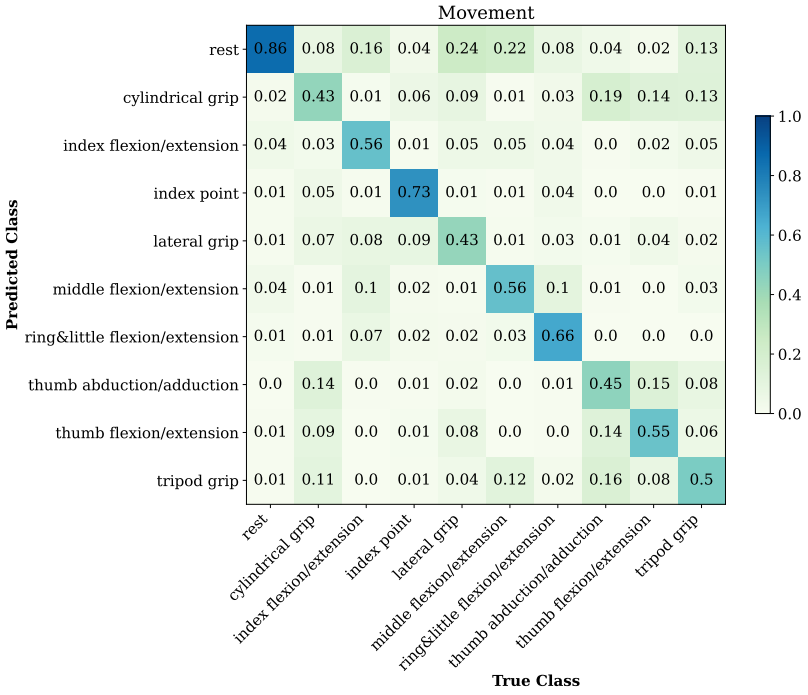


Figure 4.2: Average movement decoding confusion matrix for the FULL configuration. The colour bar and annotated scores indicate normalised prediction rates.

be able to handle the outliers better, although this does result in lower median performance. All models perform better than the 'All rest' baseline.

Figure 4.2 depicts the average confusion matrix for the FULL configuration. The imbalance in the data shows in the disproportionate accuracy for the rest movement. Individual class accuracy scores differ for the non-rest movements, with a minimum of 0.43 for cylindrical and lateral grip, and a maximum of 0.73 for index point.

4.2 Digit Action Decoding

Figure 4.3 gives the macro-averaged F1-score of the three baselines and the transformer models, in the digit action decoding paradigm, i.e. the six digit DOFs are classified into 'open', 'stall', or 'close'. The FULL configuration achieves the highest median F1-score, although not statistically discernible from the other transformer models. There is statistical significance between LDA and the FTR configuration. The FNET configuration again has an outlier participant, whereas the FULL and TNET configurations do not. All models perform better than the 'All stall' baseline.

Figure 4.4 depicts the average confusion matrices per DOF for the FULL configuration. The imbalance in the data shows in the disproportionate accuracy for the

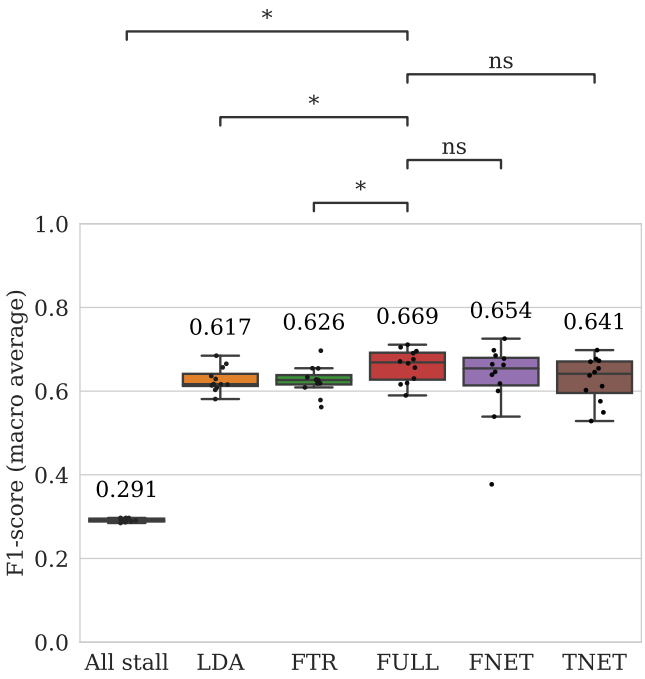


Figure 4.3: Comparison of digit action decoding performance using F1-score of baselines and transformers. The median is given above each box-plot. Statistical comparisons are performed only against the best-performing algorithm; asterisk: $p < 0.05$, 'ns': $p > 0.05$.

stall class, across DOFs. Confusion generally happens because the model predicts the 'stall' class when it should predict 'open' or 'close', less so the other way around. The individual accuracy scores for the 'open' and 'close' classes are visibly lower for the two thumb DOFs.

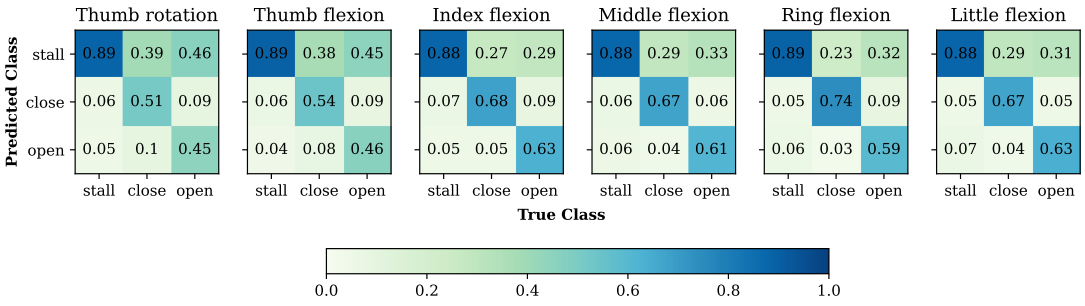


Figure 4.4: Average confusion matrix for each DOF for the FULL configuration. The colour bar and annotated scores indicate normalised prediction rates.

4.2.1 Input & Model Variations

In the interest of exploring possibilities for the transformer models, I evaluate a number of derivations. Each of the following configurations' hyperparameters we optimised independently:

- FULL*: the FULL configuration but its TNET and FNET are jointly trained and optimised, i.e. the approach of Zabihi et al. (2022).
- FULL MM: the FULL configuration but the linear envelopes are only Min-Max normalised and not μ-law encoded, i.e. values near zero are not enlarged (see Figure 2.3).
- FULL<3 Hz: the FULL configuration but the linear envelopes are computed using a 3 Hz cut-off instead of a 1 Hz cut-off, i.e. more oscillatory nuances are retained in the envelopes (see Figure 2.2).
- FULL WCE: the FULL configuration but trained using weighted cross-entropy loss to address the class imbalance (Cui et al., 2019).
- FULL 6 heads: the FULL configuration but with a separate output head for each DOF, i.e. allowing more DOF-specific computation outside of the transformer-encoder.

Figure 4.5 gives the macro-averaged F1-score of these variations. The FULL WCE configuration achieves the highest median F1-score, although not statistically discernible from most other configurations. Only the 3 Hz cut-off variation shows a statistically significant decrease in performance.

The difference in performance between these variations is marginal. This suggests that performance bottlenecks most likely lie elsewhere. Potential culprits are inaccurate finger kinematics measurements (see Section 3.1.1), inaccurate label extraction (see Section 3.1.3), insufficient data (see Section 3.1.2), and the distribution shift between the training and testing splits. Recording more accurate finger kinematics requires recording a new dataset with potentially new equipment. This is beyond the scope of this thesis and I therefore do not consider it. Fine-tuning Krasoulis and Nazarpour's (2020) method for extracting finger actions from the recorded finger kinematics is a possible route to take. However, the outcome of this would still heavily depend on the accuracy of the kinematics measurements, which Krasoulis and Nazarpour think to be the more influential factor. Therefore, for this dissertation, I focus on

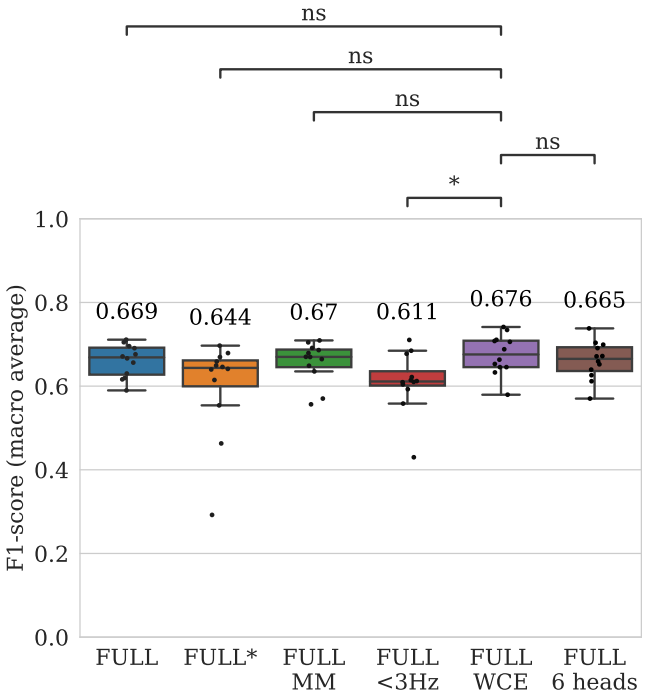


Figure 4.5: Comparison of digit action decoding performance using F1-score of FULL variations. The median is given above each boxplot. Statistical comparisons are performed only against the best-performing algorithm; asterisk: $p < 0.05$, 'ns': $p > 0.05$.

investigating the effects of the distribution shift between acquisitions and an increase in training data.

4.2.2 Cross-Acquisition Training Data

A big part of the complexity of decoding EMG signals stems from their nonstationary nature. Recordings can differ significantly across days, recording sessions, and even within sessions. The pre-determined training/validation/testing splits in DB8 each correspond to a recording acquisition where participants had a ten-minute break in between. A break as short as ten minutes can already result in participants performing gestures differently, because of fatigue or forgetting (Kyranou et al., 2018). Generalising across these differences is hard when training data comes from a single acquisition.

To address this, we evaluate a final set of models that is trained on data from two separate acquisitions, denoted using '-MIX'. A new training split comprises all odd repetitions from the original training acquisition and all even repetitions from the original validation acquisition. A new validation split then comprises the leftover repetitions. The testing split remains the third acquisition. These new training and validation

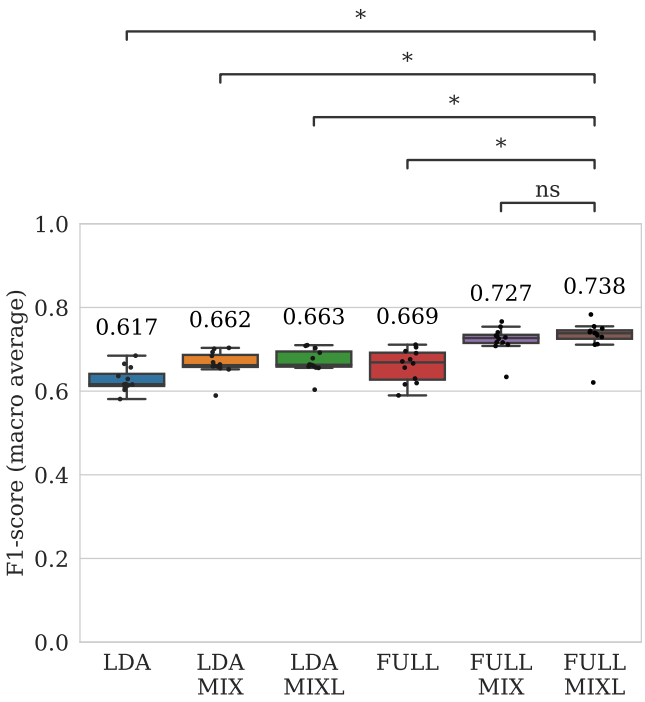


Figure 4.6: Comparison of digit action decoding performance using F1-score of LDA and the FULL configuration when trained with mixed acquisition data. The median is given above each box-plot. Statistical comparisons are performed only against the best-performing algorithm; asterisk: $p < 0.05$, 'ns': $p > 0.05$.

splits each consist of ten repetitions of movements, just as the original splits. They should yield insight into the capability of the models to generalise over acquisitions.

To also get insight into the benefit of having more training data, we establish a third set of splits with a larger training split, denoted using '-MIXL'. Specifically, the validation split comprises only the fifth repetition of the original training and validation splits. The remaining 18 repetitions then form a large training split.

Figure 4.6 gives the macro-averaged F1-score of LDA and the FULL configuration when trained on the three sets of splits. The FULL-MIXL configuration achieves the highest median F1-score, although not statistically discernible with the FULL-MIX configuration. There is statistical significance with all LDA models and with the FULL configuration when trained using the original splits. Performance for both models visibly increases when training data is mixed. There is no statistical basis to claim that a larger training split results in increased performance. The MIX and MIXL versions of both LDA and the FULL configuration all have an outlier. This outlier is always the performance for the 12th participant, indicating that there might have been issues in

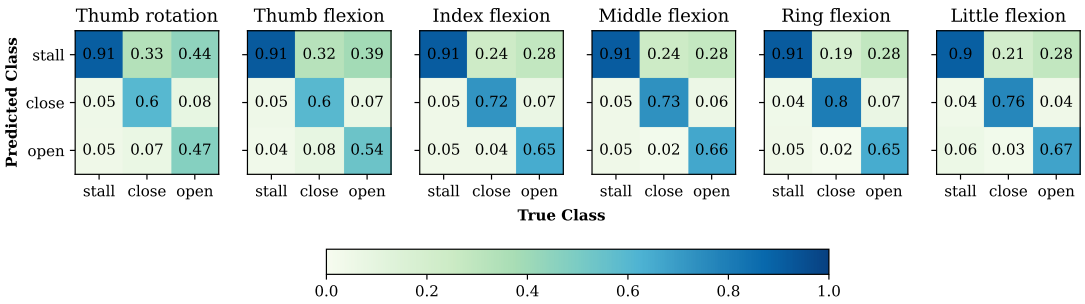


Figure 4.7: Average confusion matrix for each DOF for the FULL-MIX configuration. The colour bar and annotated scores indicate normalised prediction rates.

the recording of the validation acquisition of this participant.

Figure 4.7 depicts the average confusion matrices per DOF for the FULL-MIX configuration. Similar patterns are visible as for the FULL configuration. The increases are generally in the 'open' and 'close' classes, for which the FULL-MIX configuration predicts 'stall' less than the FULL configuration.

4.2.3 Independent Output Models

An important comparison for this work is that with models that assume the six output DOFs to be independent, such as those of Krasoulis and Nazarpour (2020). Table 4.1 presents our results side-by-side.

The performance of the main and extended transformer variations is similar to that of the independent output models. The FULL configuration achieves a higher median F1-score than the best-performing independent output model RDA. The median exact match ratio of the FULL configuration is similar to that of RDA. The best-performing models in terms of median exact match ratio are Krasoulis and Nazarpour's independent output random forests (RF), which also perform well in terms of precision. These are the only metrics for which the independent output models outperform the transformer models, albeit marginally.

When compared to Krasoulis and Nazarpour's classifier chain approach (CC, see Section 2.1.2), the transformer models provide significant improvements in all metrics. Overall, the CC models perform worst. A potential cause is that these models are still trained with handcrafted features. An interesting comparison in the future could compare a CC consisting of transformer models trained on linear envelopes. The results of such a comparison should shed light on whether the handcrafted features are the bottleneck that limit a CC in learning the dependencies between the outputs.

Table 4.1: Median scores and overall ranges for each comparison set of models. Bold values indicate the highest median performance for each set.

	Model	F1-score (macro-average)	Exact Match Ratio	Hamming Score	Recall (macro-average)	Precision (macro-average)
independent output models (Krasoulis & Nazarpour, 2020)	RDA	**0.64 (0.56, 0.69)**	0.58 (0.52, 0.62)	0.82 (0.79, 0.84)	0.63 (0.54, 0.70)	0.67 (0.59, 0.70)
	QDA	0.63 (0.39, 0.67)	0.53 (0.03, 0.60)	0.78 (0.40, 0.81)	**0.66 (0.58, 0.73)**	0.61 (0.51, 0.65)
	RF	0.61 (0.50, 0.68)	**0.62 (0.57, 0.64)**	**0.84 (0.81, 0.85)**	0.57 (0.47, 0.66)	**0.77 (0.65, 0.81)**
	GNB	0.57 (0.49, 0.63)	0.54 (0.10, 0.57)	0.76 (0.61, 0.78)	0.59 (0.54, 0.69)	0.58 (0.47, 0.61)
	KNN	0.56 (0.46, 0.63)	0.58 (0.51, 0.62)	0.82 (0.77, 0.83)	0.52 (0.44, 0.61)	0.65 (0.50, 0.70)
	LR	0.56 (0.45, 0.66)	0.58 (0.55, 0.60)	0.82 (0.80, 0.84)	0.53 (0.43, 0.65)	0.69 (0.62, 0.72)
	ET	0.51 (0.43, 0.62)	0.59 (0.56, 0.62)	0.83 (0.80, 0.84)	0.47 (0.41, 0.60)	0.76 (0.68, 0.83)
	All stall	0.33 (0.33, 0.33)	0.10 (0.08, 0.15)	0.61 (0.58, 0.66)	0.33 (0.33, 0.33)	0.33 (0.33, 0.33)
	LDA	0.64 (0.55, 0.70)	0.59 (0.53, 0.62)	0.83 (0.79, 0.85)	0.62 (0.54, 0.69)	0.70 (0.58, 0.73)
classifier chains (Krasoulis & Nazarpour, 2020)	CC (best)	**0.63 (0.53, 0.66)**	0.53 (0.44, 0.60)	0.80 (0.70, 0.83)	**0.61 (0.54, 0.67)**	**0.66 (0.51, 0.71)**
	CC (worst)	0.61 (0.53, 0.66)	0.53 (0.44, 0.60)	0.80 (0.70, 0.83)	0.60 (0.54, 0.67)	0.65 (0.52, 0.71)
	CC ensemble	**0.63 (0.55, 0.67)**	0.53 (0.44, 0.60)	0.80 (0.71, 0.83)	**0.61 (0.55, 0.68)**	**0.66 (0.53, 0.72)**
baselines & transformers	All stall	0.29 (0.28, 0.30)	0.56 (0.55, 0.59)	0.77 (0.75, 0.80)	0.33 (0.33, 0.33)	0.26 (0.25, 0.27)
	LDA	0.62 (0.58, 0.68)	**0.58 (0.09, 0.62)**	0.82 (0.73, 0.85)	0.61 (0.57, 0.67)	0.67 (0.57, 0.73)
	FTR	0.63 (0.56, 0.70)	**0.58 (0.43, 0.63)**	0.82 (0.78, 0.86)	0.61 (0.54, 0.68)	0.68 (0.57, 0.74)
	FNET	0.65 (0.38, 0.73)	0.57 (0.49, 0.62)	0.83 (0.75, 0.86)	0.65 (0.39, 0.73)	0.71 (0.53, 0.82)
	TNET	0.64 (0.53, 0.70)	0.57 (0.49, 0.65)	0.83 (0.75, 0.85)	0.61 (0.57, 0.70)	0.69 (0.57, 0.76)
	FULL	**0.67 (0.59, 0.71)**	**0.58 (0.41, 0.64)**	**0.84 (0.77, 0.86)**	**0.66 (0.58, 0.69)**	**0.72 (0.60, 0.75)**
FULL variations	FULL*	0.64 (0.29, 0.70)	**0.58 (0.39, 0.64)**	0.82 (0.78, 0.86)	0.61 (0.33, 0.68)	0.70 (0.26, 0.80)
	FULL MM	**0.67 (0.56, 0.71)**	**0.58 (0.17, 0.64)**	**0.84 (0.73, 0.86)**	0.66 (0.55, 0.71)	**0.71 (0.58, 0.79)**
	FULL<3Hz	0.65 (0.53, 0.69)	0.56 (0.48, 0.62)	0.82 (0.79, 0.86)	0.62 (0.49, 0.70)	0.68 (0.57, 0.77)
	FULL WCE	0.66 (0.53, 0.72)	0.50 (0.36, 0.61)	0.81 (0.68, 0.85)	**0.72 (0.62, 0.77)**	0.65 (0.51, 0.73)
	FULL 6 heads	0.66 (0.55, 0.72)	0.56 (0.48, 0.63)	**0.84 (0.79, 0.86)**	0.65 (0.54, 0.73)	**0.71 (0.59, 0.76)**
mixed training data	LDA MIX	0.66 (0.59, 0.70)	0.60 (0.53, 0.63)	0.84 (0.80, 0.86)	0.64 (0.59, 0.70)	0.72 (0.60, 0.75)
	LDA MIXL	0.66 (0.60, 0.71)	0.60 (0.53, 0.63)	0.84 (0.80, 0.85)	0.65 (0.60, 0.71)	0.72 (0.62, 0.75)
	FULL MIX	0.73 (0.63, 0.77)	0.61 (0.54, 0.66)	0.86 (0.82, 0.89)	0.72 (0.63, 0.76)	0.76 (0.65, 0.80)
	FULL MIXL	**0.74 (0.62, 0.78)**	**0.63 (0.57, 0.68)**	**0.87 (0.83, 0.89)**	**0.73 (0.59, 0.78)**	**0.76 (0.68, 0.80)**

Although comparing the mixed training data set to the independent output models is unfair, it does provide added insight into the benefit of training on cross-acquisition data. Here, doing so yields significant increases in median F1-score, hamming score, and recall. Increasing training data has a diminished effect, with a 2 % increase for the median exact match ratio and a 1 % increase for all the other metrics.

Chapter 5

Discussion

5.1 Transformers for Movement decoding

In terms of extracting individual movements from sEMG, the transformer architecture does not provide a significant performance increase compared to a traditional ML approach such as LDA trained on handcrafted features. In this work, this can be attributed to the dataset's acquisition protocol; Ninapro DB8 was not designed for such a control paradigm (Krasoulis et al., 2019). When participants gradually move between rest and a specific movement, very dissimilar window samples are assigned the same labels. The transformer has to find the similarities between these windows, which will result in computing higher-level features that resemble established sEMG window features such as the average amplitude of the window.

I evaluate the transformer architecture in this paradigm as a safety check, to see if I implemented everything correctly and to verify if the models learn correctly. The conclusion there is that they do, and the results indicate that even when the transformer can not exercise its temporal advantage (since it sees window samples as a whole, instead of a feature calculated across them), it can still learn more global representations that generalise over the dissimilar movement windows.

When data matches the movement decoding paradigm (e.g. Ninapro DB2), Zabihi et al. (2022) show that the transformer architecture can exploit its temporal advantage to outperform handcrafted features. They achieve significant performance increases when compared to traditional ML approaches, indicating that there are better features to be learned. However, this does not necessarily mean that such representations also exist when the control paradigm implements multiple DOFs and there is only one transformer to generalise over them.

5.2 Transformers for Digit Action Decoding

The transformer architecture provides a statistically significant increase in the main performance measure when used to extract digit actions. I have two hypotheses concerning what causes this. On the one hand, the transformer can be learning representations that capture dynamics in the linear envelope windows that are not captured by traditional handcrafted features. On the other, it can exploit its shared core to learn the dependencies between the digits. The comparison with the FTR configuration, which simply applies the attention mechanism to the handcrafted features but performs similarly to LDA, strengthens the former hypothesis. Furthermore, the performance of the FTR configuration is similar to that of Krasoulis and Nazarpour's (2020) independent output models, suggesting that either the model is not learning the dependencies between the outputs, or the handcrafted features do not capture sufficient information to do so. From this, we can only conclude that both hypotheses hold. Although not substantial, a 3 % increase in median F1-score when compared to the best-performing independent output model of Krasoulis and Nazarpour suggests that the transformer architecture is finding relevant features in the linear envelopes. In addition, the fact that it does this with a shared core shows that it might not exploit the dependencies between the digits optimally, but it is able to at least consider them all at once.

The TNET and FNET configurations seem to have their own strengths and weaknesses. The FNET configuration has outlier performance for the ninth participant, indicating a possibly less stable model. The TNET configuration does not have outliers but has a lower median performance. Nevertheless, both of these configurations are not statistically discernible, so these differences could come down to hyperparameter values. Combining the two transformer channels into the FULL configuration yields slightly higher performance, which is expected as the FULL model can simulate its parts. Training both the TNET and FNET simultaneously, i.e. the FULL* configuration, does not result in increased performance. The most likely explanation for this is that there is insufficient data. The FULL* configuration is definitely the most powerful model, and thereby also the most likely model to be able to exploit the dependencies between outputs. However, we need more data to be able to test such a hypothesis.

Other variations of the architecture and data processing do not impact performance much. Choosing a higher frequency cut-off for the envelope calculation seems to decrease performance. An interesting observation is that while the F1-score of the weighted loss configuration is similar to that of the non-weighted version, its exact

match ratio and precision lie much lower and its recall lies much higher. Naturally, this is because the class proportions in the testing acquisition are the same as in the training acquisition, i.e. severely skewed towards the 'stall' class, which should also be the case in real-life usage. This raises the question of whether we can design a model that can take this imbalance into account without compromising performance for the most represented classes. A potential approach could be a two-phased model that first decides whether or not the digit is stalling, and if so, only then decides whether it is opening or closing.

5.2.1 Cross-Acquisition Transfer

The most evident way to increase performance after observing that various architecture and processing methods do not change much, is introducing more and/or more variable data. The results here show that the latter on its own can already increase performance significantly. This holds for both LDA and the transformer model. Increasing training data yields a larger performance increase, although not linearly nor statistically discernible.

There is no basis to claim that the transformer architecture is able to make better use of cross-acquisition data than the ML models with independent outputs, as LDA achieves similar increases in performance. However, we can claim that the transformer can generalise across acquisitions. With only ten repetitions of cross-acquisition training data, it is possible to achieve a median macro-averaged F1-score of 72.7 % on an unseen acquisition. Such a level of performance goes beyond Krasoulis and Nazarpour (2020, 2022) proving feasibility and indicates that the digit action decoding paradigm can achieve high accuracy with a reasonable amount of training data.

5.2.2 Muscle Groups

By analysing the model and its performance in a bit more detail, I find that it has some understanding of the muscle groups involved in finger movements. The model performs worst for the thumb DOFs (see Figures 4.4 and 4.7). This is expected from a physiological perspective as the thumb is controlled by intrinsic and deep extrinsic muscles not easily accessible from the surface of the forearm. Furthermore, it matches previous work on extraction of digit actions/trajectories (Ngeo et al., 2014).

The positional encodings of the TNET configuration, i.e. the model that applies temporal attention, can yield further insight into the transformer's understanding of

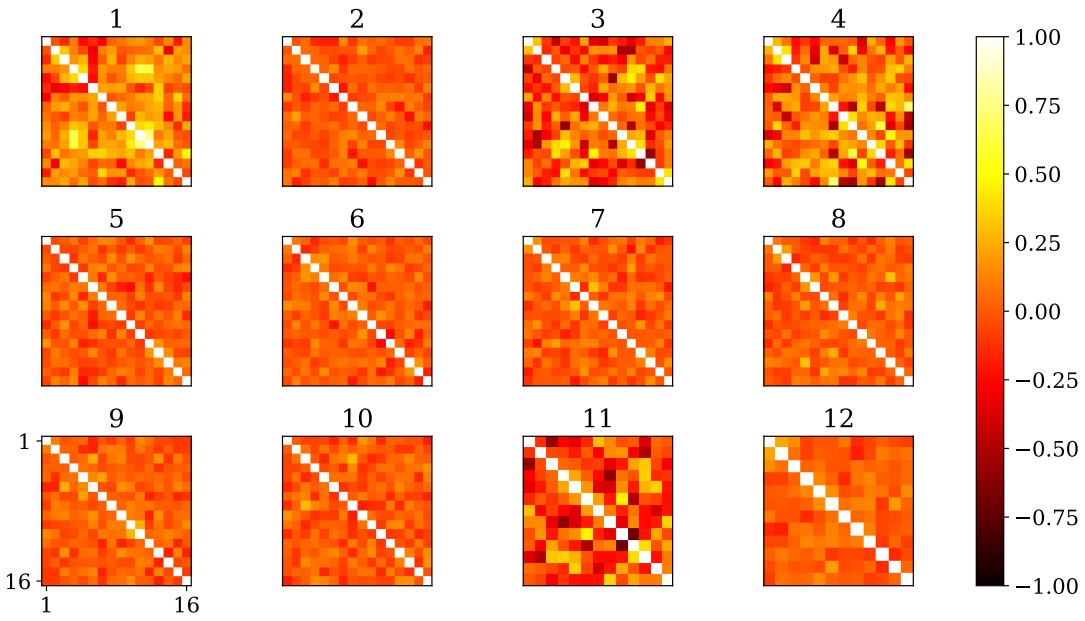


Figure 5.1: Positional encoding similarities for the TNET configuration for all participants. Each row in each figure represents the cosine similarity between one position and all the other positions. Participants 11 and 12 were people with limb difference for which there were only 13 and 12 electrodes, respectively.

the muscles. Figure 5.1 depicts cross-similarities for the positional encodings of the TNET models of all participants. For the majority of the participants the similarity maps only depict similarity on the diagonal. Intuitively, this might make it seem as if the model is not using positional information at all. However, training the model without positional encodings results in substantial performance decreases (up to 20 % in median F1-score), indicating that the model does in fact learn and use positional information. The diagonal similarity maps rather indicate that the models do not capture electrode similarities for these participants.

For some participants (1, 3, 4, and 11), the models do identify similarities and inverses in the electrodes. For these participants, the similarity maps roughly match the electrodes' positions depicted by Figure 3.1a. Lines parallel to the diagonal (most clear for participant 11) indicate similarity with electrodes opposed to each other in the double ring formation. Square patterns on the diagonal indicate similarity with electrodes adjacent in the rings.

It is difficult to pinpoint the cause of why the model finds similarities for some participants and not for others. It could be that the electrodes of the participants for which there are clear similarity patterns are positioned closer to each other, as a result of less

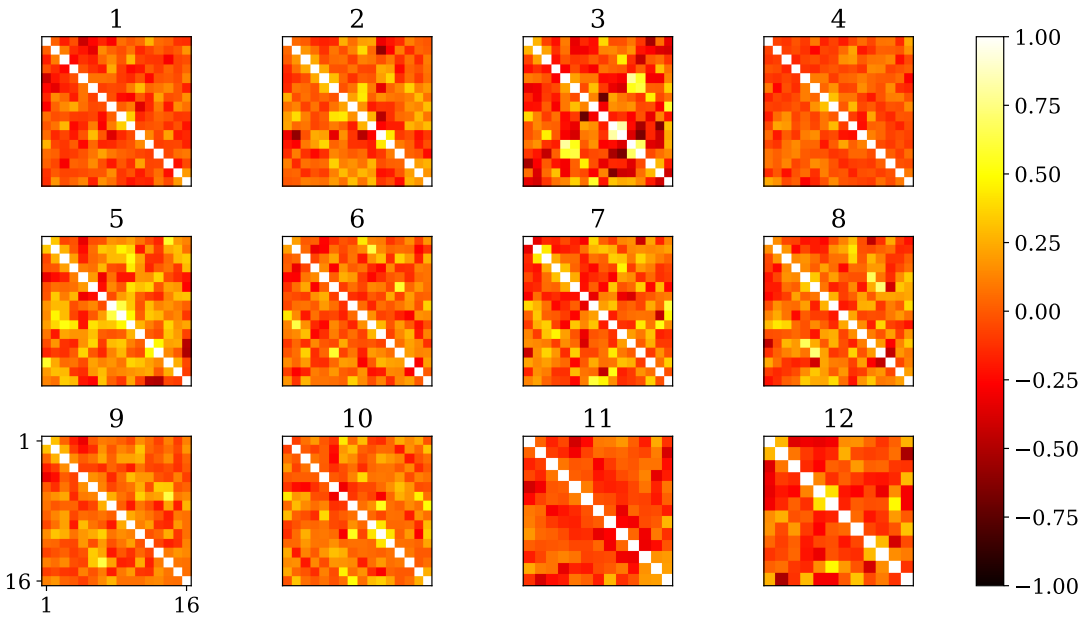


Figure 5.2: Positional encoding similarities for the TNET-MIX configuration for all participants. Each row in each figure represents the cosine similarity between one position and all the other positions. Participants 11 and 12 were people with limb difference for which there were only 13 and 12 electrodes, respectively.

surface areas because of thinner arms. However, this is disproved by the similarity maps of the TNET-MIX configuration in Figure 5.2. Remember that the training data for this configuration comes from two acquisitions, separated by a ten minute break during which the electrodes were not removed from the participants' arms (see Section 3.1.1). Comparing to the similarity maps of the TNET models, we can see that there are now participants with clear similarity patterns, where there were not before. This suggests that when the TNET configuration was trained within one acquisition, it was less inclined to focus on muscle similarity and was in some way overfitting on characteristics specific to that acquisition. In contrast, when training samples come from two acquisitions, it is forced to focus on more transferable characteristics. Since the electrodes were not moved between acquisitions, their position and similarities between them seem to be part of such characteristics. This is also reflected by the increased performance of the TNET-MIX configuration.

Interestingly, when training data still comes from two acquisitions but there is almost twice as much, Figure 5.3 depicting the similarity maps for the TNET-MIXL configuration suggests that the models seem to once more disregard muscle similarity. It is not clear why this happens. An unlikely explanation is that the additional sam-

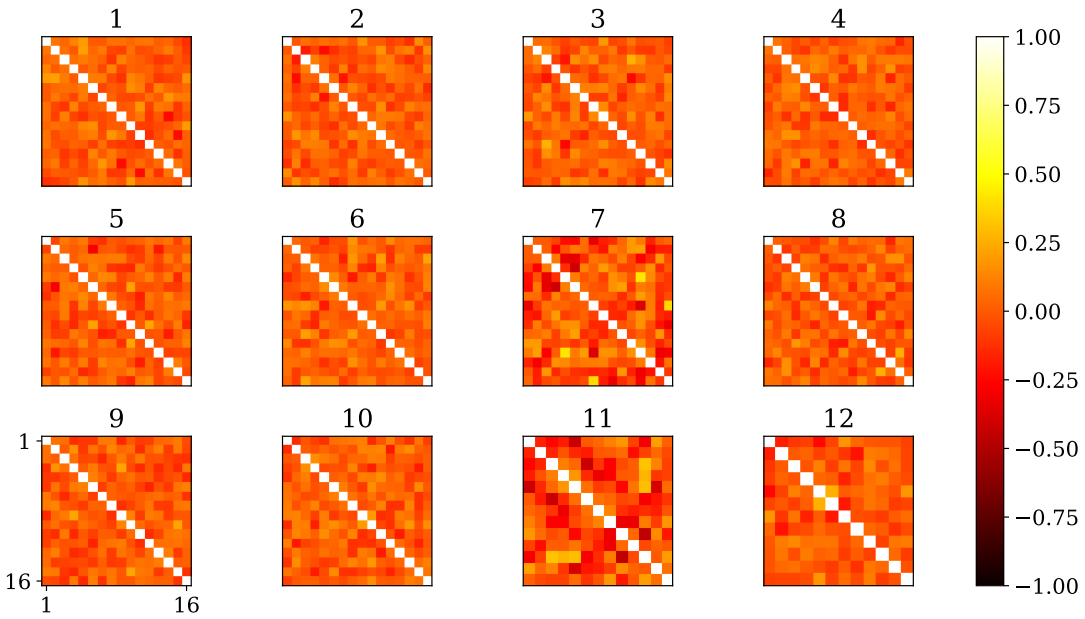


Figure 5.3: Positional encoding similarities for the TNET-MIXL configuration for all participants. Each row in each figure represents the cosine similarity between one position and all the other positions. Participants 11 and 12 were people with limb difference for which there were only 12 and 13 electrodes, respectively.

ples in the training split of the MIXL configurations are particularly dissimilar from all other samples. More likely is that the extraction of muscle similarity happens in a deeper layer and is therefore not reflected in the positional encodings. The black-box nature of the transformer makes proving this hypothesis difficult. An interesting experiment for the future would be to train the same models with fixed positional encodings (see Section section A.1). Two phase-shifted sinusoids could be used to establish an encoding scheme representing the similarity between the electrodes in the double ring formation. Doing so would force the models to consider muscle similarity in the positional encodings as well as eliminate the effect of limited training data on its spatial understanding.

5.2.3 Correlated Samples

A similar analysis of the positional encodings of the FNET configuration yields insight into how the model perceives temporal changes in spatio-temporal space. Figure 5.4 depicts the cross-similarities for the positional encodings of the FNET models of all participants. Here as well, the models sometimes disregard positional similarities. In

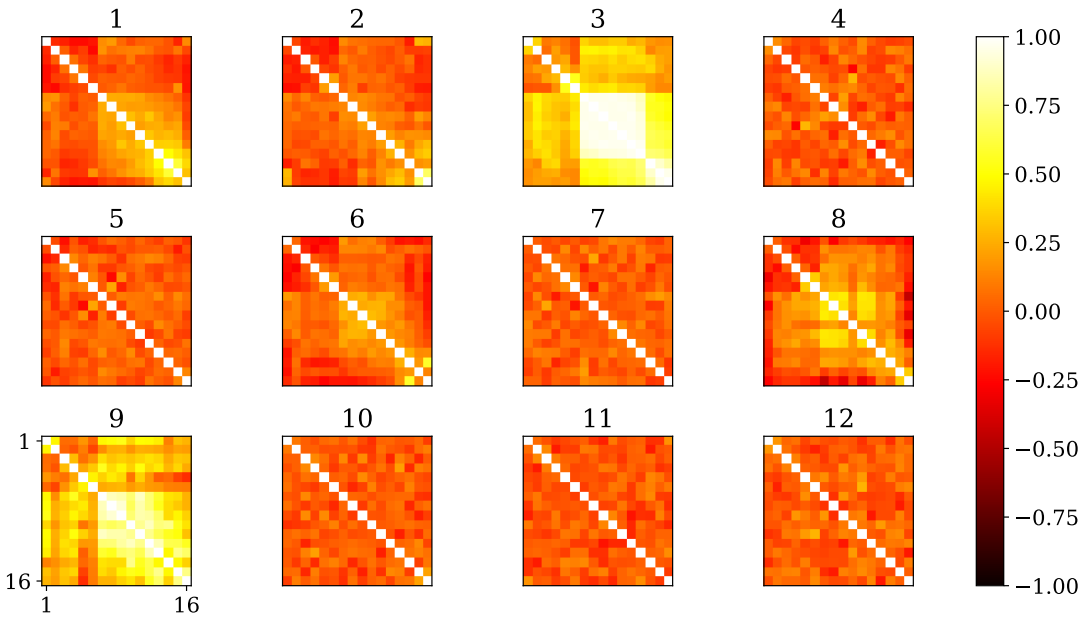


Figure 5.4: Positional encoding similarities for the FNET configuration for all participants. Each row in each figure represents the cosine similarity between one position and all the other positions.

the cases that it does consider them, the similarity maps seem to always be separated into two parts, corresponding to a 40/60 % split. My hypothesis for this is that the models are picking up on the correlation between samples, as the sEMG recordings are segmented into windows with approximately 60 % overlap (see Section 3.1.2).

Similar patterns are visible in the similarity maps of the MIX and MIXL variants (see Figures E.1 and E.2, in Appendix E). The fact that models are able to pick up on these correlations indicates that reducing the overlap should not yield different results if it would not also significantly reduce the number of windows the sEMG recordings can be split into. In contrast, increasing overlap comes with a substantial increase in training samples, even though they are highly correlated. Other studies evaluate spatio-temporal attention with a range of overlaps (Zabihi et al., 2022). Although I mirrored Krasoulis and Nazarpour's (2020) approach in this work, it should be interesting to explicitly look for the optimal overlap for the digit ation action decoding paradigm, as this is a task-specific setting. In addition, it should again be interesting to implement fixed positional encodings for spatio-temporal attention. Positions are one-dimensional in this case so traditional encoding schemes would suffice.

5.2.4 Time & Space Complexity

Prediction time and memory requirements are important considerations for sEMG decoding models. After all, these algorithms are usually intended to run on low-memory embedded systems and predict intention in real-time. Transformer models are universally seen as large models (e.g. large language models) and even with their depth and width restricted (here maximally 10 layers of size 512, see Appendix D), they do not belong to the category of models that are typically implemented in prosthesis controllers.

The transformer uses the self-attention mechanism, which issues a query for every position in an input sequence. Hence, its overall time and space complexity is $O(n^2)$, with n the length of the input sequence (Vaswani et al., 2017). As such, various attempts have been made at reducing these complexities, with successes able to get it down to $O(nlog(n))$ using hashing (Reformer; Kitaev et al., 2020) or even $O(n)$ with certain approximations (Linformer; S. Wang et al., 2020). Nevertheless, even with such reductions, the transformer architecture would still only qualify for certain cases where prostheses come with costly portable Graphics Processing Units (GPU; Nguyen et al., 2021) or are connected to static external processing units (H. Wu et al., 2021).

In general, the research I present in this dissertation is not intended to run on a real-time prosthesis. With model sizes ranging between 10 MB to 70 MB, such prostheses would not be feasible to design in terms of cost and size. The goal is rather to, on the one hand, get insight into the capability of the architecture to learn from biomedical time series, and on the other, push the boundaries in terms of accuracy. Moreover, by investigating what the model learns to attend to, we can possibly discover new features that can ultimately be used with models that do match current-day prosthesis specifications.

Chapter 6

Conclusion

I evaluate the transformer architecture when implemented to extract finger movements from sEMG recordings. I compare and weigh a number of configurations representing approaches to input processing, model training, and architecture. The best-performing model achieves a statistically significant increase in the main performance measure when compared to previous work (Krasoulis & Nazarpour, 2020). It does this through a combination of computing its own features of the complete sEMG windows it gets as input and exploiting its shared core to learn dependencies between its multiple outputs. Through a lower-level analysis of the model, I discover that it can identify muscle similarities and learns to use these in its computations. Furthermore, by training the model with cross-acquisition data, I find that it can generalise across acquisitions and that this comes with substantial performance improvements. Combined, these findings show that the digit action decoding paradigm is not only feasible (Krasoulis & Nazarpour, 2020, 2022), but it is also possible to implement it with non-trivial accuracy.

There are two main limitations to my work. Firstly, I only perform offline analyses. Real-time control is known to come with its own difficulties for which offline performance is often not a good proxy (Ortiz-Catalan et al., 2015; Vujaklija et al., 2017). Secondly, the scale of my evaluations is relatively small. I only consider Ninapro's DB8, which comprises a limited number of participants and finger movements and was recorded in an isolated environment. The transformer architecture should be evaluated in terms of robustness under non-stationary conditions and generalisation to unseen finger movements. Machine learning models are known to suffer from poor generalisation under different limb positions and/or muscle contraction levels (Fougner et al., 2011; Khushaba et al., 2016). The transformer architecture should be a good candidate to address these issues.

6.1 Contribution

In terms of the research objectives set out in Section 1.1, my contributions are the following:

- *O.1: Establishing a high-level overview of the efficacy of transformer-like architectures when employed for extracting intention from sEMG recordings.*
  Transformers can learn to extract intention from sEMG recordings in both the movement and digit action decoding paradigms. They achieve slightly better performance than the current state-of-the-art.
  - *What input formats work best?*
    Although a small transformer can learn from a collection of sEMG features, a larger model trained on linear envelopes achieves higher performance. A cutoff of 1 Hz performed best in this work.
  - *How much data is needed to achieve reasonable performance?*
    State-of-the-art performance is achievable with ten 6 s repetitions of training data. Using almost twice as much training data yields marginally increased performance.
  - *Is cross-acquisition transfer feasible?*
    The transformer can generalise across acquisitions and achieves significant performance boosts when doing so.
  - *Which patterns does the model pick up on and can these be related to biophysical phenomena?*
    The model learns positional encodings that capture similarities between muscles (temporal attention) and correlations between samples (spatio-temporal attention).
- *O2: Verifying whether a neural model with a shared core for multiple outputs can learn generalised representations of sEMG, comparing to models that unrealistically assume outputs to be independent.*
  The transformer architecture can learn representations of linear sEMG envelopes that generalise to multiple outputs. As a single model, it outperforms previous evaluations of ML models that are independently trained for each output.

6.2 Prospects & Future work

The transformer architecture is relatively new, especially in biomedical contexts. As such, a lot of research remains to be done before it will find its way into commercial and clinical use. It is a powerful model that, as shown here, can find transferable patterns in nonstationary signals. Furthermore, its popularity comes paired with a vast interest in improving the architecture. There are complete research fields dedicated to improving aspects of the attention mechanism such as computational complexity (S. Wang et al., 2020) and interpretability (Chefer et al., 2021). Successes in these fields are particularly relevant for my work as they would allow me to either, run an efficient transformer on an embedded prosthesis controller, or extract the transformer's understanding of the sEMG recordings and use it to design more compact models.

As foremost extension to my work, I propose to further investigate the temporal positional encodings of the model. If we can train a model on a single acquisition but still have it learn muscle similarities as it did when trained on cross-acquisition data, I expect performance to increase significantly. This is a difficult task because the large model is likely to overfit with the amounts of data typical of sEMG datasets. A good start might be to implement more advanced training approaches that use periodically decreasing learning rates and early stopping. In parallel with this research route, it should also be of interest to investigate fixed positional encodings. These are not affected by the amount of training data and should provide a relevant baseline for their learnt counterparts.

Another interesting route to take would be to investigate the effect of window overlap in the context of digit action decoding. The spatio-temporal positional encodings of the model reflect the value of this hyperparameter, raising the question of whether changing it will drastically change performance. Previous work uses much smaller overlap (down to 3 %) in a movement decoding paradigm (Zabihi et al., 2022), thereby increasing the number of training samples substantially. Even though these samples are highly correlated, the transformer still seems to benefit from having more. It should be interesting to verify this for the digit action paradigm. Fixed positional encodings should provide a relevant baseline in this case as well.

Lastly, it should also be interesting to investigate hardware related questions. For example, there is the matter of extracting digit actions from the dataglove recordings. Datagloves are known to be designed to fit the average human finger and hand length and circumference, often unrealistic in terms of real-life variance, as well as not tak-

ing into account ageing and gender differences (Kahlesz et al., 2004). Furthermore, the calibration process that is intended to reduce the effects of these mismatches has been reported to be time-consuming and inadequate (Huenerfauth & Lu, 2010). In the digit action decoding paradigm as implemented by Krasoulis and Nazarpour (2020), this has a direct effect on model performance, as labels are computed directly from the dataglove recordings. A promising approach addressing this dependency consists of using neural networks to correct the dataglove recordings, alleviating the need for calibration (Connolly et al., 2022). On the side of the electrodes, temporal attention can be capitalised upon through high-density sEMG recordings (HD-sEMG; Celadon et al., 2016). Such recordings are made using substantially more electrodes, potentially increasing the transformer's spatial field of view.

Bibliography

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631.

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.

Atzori, M., Gijsberts, A., Castellini, C., Caputo, B., Hager, A.-G. M., Elsig, S., Giatsidis, G., Bassetto, F., & Müller, H. (2014). Electromyography data for non-invasive naturally-controlled robotic hand prostheses. *Scientific Data*, *1*(1), 140053.

Ba, L. J., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *CoRR*, *abs/1607.06450*.

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, *24*.

Bland, J. M., & Altman, D. G. (1995). Multiple significance tests: The bonferroni method. *BMJ*, *310*(6973), 170.

Bracewell, R. N., & Bracewell, R. N. (1986). *The fourier transform and its applications* (Vol. 31999). McGraw-Hill New York.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (pp. 1877–1901). Curran Associates, Inc.

Butterworth, S. (1930). On the theory of filter amplifiers. *Experimental Wireless & the Wireless Engineer*, *7*, 536–541.

Cai, S., Su, E., Xie, L., & Li, H. (2022). EEG-based auditory attention detection via frequency and channel neural attention. *IEEE Transactions on Human-Machine Systems*, *52*(2), 256–266.

Celadon, N., Došen, S., Binder, I., Ariano, P., & Farina, D. (2016). Proportional estimation of finger movements from high-density surface electromyography. *Journal of NeuroEngineering and Rehabilitation*, *13*, 73.

Chefer, H., Gur, S., & Wolf, L. (2021). Transformer interpretability beyond attention visualization, 782–791.

Chowdhury, R. H., Reaz, M. B. I., Ali, M. A. B. M., Bakar, A. A. A., Chellappan, K., & Chang, T. G. (2013). Surface electromyography signal processing and classification techniques. *Sensors (Basel, Switzerland)*, *13*(9), 12431–12466.

Cipriani, C., Antfolk, C., Controzzi, M., Lundborg, G., Rosen, B., Carrozza, M. C., & Sebelius, F. (2011). Online myoelectric control of a dexterous hand prosthesis by transradial amputees. *IEEE transactions on neural systems and rehabilitation engineering: a publication of the IEEE Engineering in Medicine and Biology Society*, *19*(3), 260–270.

Connolly, J., Condell, J., Curran, K., & Gardiner, P. (2022). Improving data glove accuracy and usability using a neural network when measuring finger joint range of motion. *Sensors (Basel, Switzerland)*, *22*(6), 2228.

Cui, Y., Jia, M., Lin, T.-Y., Song, Y., & Belongie, S. (2019). Class-balanced loss based on effective number of samples, 9268–9277.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies, NAACL-HLT 2019, minneapolis, MN, USA, june 2-7, 2019, volume 1 (long and short papers)* (pp. 4171–4186). Association for Computational Linguistics.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale.

Fougner, A., Scheme, E., Chan, A. D. C., Englehart, K., & Stavdahl, O. (2011). Resolving the limb position effect in myoelectric pattern recognition. *IEEE trans-*

actions on neural systems and rehabilitation engineering: a publication of the IEEE Engineering in Medicine and Biology Society, *19*(6), 644–651.

Gallagher, P., O'Donovan, M.-A., Doyle, A., & Desmond, D. (2011). Environmental barriers, activity limitations and participation restrictions experienced by people with major limb amputation. *Prosthetics and Orthotics International*, *35*(3), 278–284.

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 1243–1252.

Hahne, J. M., Schweisfurth, M. A., Koppe, M., & Farina, D. (2018). Simultaneous control of multiple functions of bionic hand prostheses: Performance and robustness in end users. *Science Robotics*, *3*(19), eaat3630.

Hjorth, B. (1970). EEG analysis based on time domain properties. *Electroencephalography and Clinical Neurophysiology*, *29*(3), 306–310.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hu, R., Chen, J., & Zhou, L. (2022). A transformer-based deep neural network for arrhythmia detection using continuous ECG signals. *Computers in Biology and Medicine*, *144*, 105325.

Huenerfauth, M., & Lu, P. (2010). Accurate and accessible motion-capture glove calibration for sign language data collection. *ACM Transactions on Accessible Computing*, *3*(1), 2:1–2:32.

Hussein, R., Lee, S., & Ward, R. (2022). Multi-channel vision transformer for epileptic seizure prediction. *Biomedicines*, *10*(7), 1551.

Ison, M., & Artemiadis, P. (2014). The role of muscle synergies in myoelectric control: Trends and challenges for simultaneous multifunction control. *Journal of Neural Engineering*, *11*(5), 051001.

Izenman, A. J. (2008). Linear discriminant analysis. In A. J. Izenman (Ed.), *Modern multivariate statistical techniques: Regression, classification, and manifold learning* (pp. 237–280). Springer.

Jabbari, M., Khushaba, R., & Nazarpour, K. (2021). Spatio-temporal warping for myoelectric control: An offline, feasibility study. *Journal of Neural Engineering*, *18*(6), 066028.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C.,

Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., … Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, *596*(7873), 583–589.

Kahlesz, F., Zachmann, G., & Klein, R. (2004). 'visual-fidelity' dataglove calibration. *Proceedings Computer Graphics International, 2004.*, 403–410.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020, January 22). Scaling laws for neural language models.

Khushaba, R. N., Al-Timemy, A., Kodagoda, S., & Nazarpour, K. (2016). Combined influence of forearm orientation and muscular contraction on EMG pattern recognition. *Expert Systems with Applications*, *61*, 154–161.

Kim, Y., Denton, C., Hoang, L., & Rush, A. M. (2022). Structured attention networks.

Kitaev, N., Kaiser, L., & Levskaya, A. (2020). Reformer: The efficient transformer. *8th international conference on learning representations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*.

Kolen, J. F., & Kremer, S. C. (2001). Gradient flow in recurrent nets: The difficulty of learning LongTerm dependencies. In *A field guide to dynamical recurrent networks* (pp. 237–243).

Komi, P. V., & Tesch, P. (1979). EMG frequency spectrum, muscle structure, and fatigue during dynamic contractions in man. *European Journal of Applied Physiology and Occupational Physiology*, *42*(1), 41–50.

Konrad, P. (2005). The abc of emg. *A practical introduction to kinesiological electromyography*, *1*(2005), 30–5.

Krasoulis, A., & Nazarpour, K. (2020). Myoelectric digit action decoding with multi-output, multi-class classification: An offline analysis. *Scientific Reports*, *10*(1), 16872.

Krasoulis, A., & Nazarpour, K. (2022). Discrete action control for prosthetic digits. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *30*, 610–620.

Krasoulis, A., Vijayakumar, S., & Nazarpour, K. (2015). Evaluation of regression methods for the continuous decoding of finger movement from surface EMG and accelerometry. *2015 7th International IEEE/EMBS Conference on Neural Engineering (NER)*, 631–634.

Krasoulis, A., Vijayakumar, S., & Nazarpour, K. (2019). Effect of user practice on prosthetic finger control with an intuitive myoelectric decoder. *Frontiers in Neuroscience*, *13*.

Krogh, A., & Hertz, J. (1991). A simple weight decay can improve generalization. In J. Moody, S. Hanson, & R. Lippmann (Eds.), *Advances in neural information processing systems*. Morgan-Kaufmann.

Kyranou, I., Vijayakumar, S., & Erden, M. S. (2018). Causes of performance degradation in non-invasive electromyographic pattern recognition in upper limb prostheses. *Frontiers in Neurorobotics*, *12*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J. E., & Stoica, I. (2018, July 13). Tune: A research platform for distributed model selection and training.

Lin, T., Wang, Y., Liu, X., & Qiu, X. (2022). A survey of transformers. *AI Open*, *3*, 111–132.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

Montazerin, M., Rahimian, E., Naderkhani, F., Atashzar, S. F., Yanushkevich, S., & Mohammadi, A. (2023). Transformer-based hand gesture recognition from instantaneous to fused neural decomposition of high-density EMG signals. *Scientific Reports*, *13*(1), 11000.

Ngeo, J. G., Tamei, T., & Shibata, T. (2014). Continuous and simultaneous estimation of finger kinematics using inputs from an EMG-to-muscle activation model. *Journal of NeuroEngineering and Rehabilitation*, *11*(1), 122.

Nguyen, A. T., Drealan, M. W., Khue Luu, D., Jiang, M., Xu, J., Cheng, J., Zhao, Q., Keefer, E. W., & Yang, Z. (2021). A portable, self-contained neuroprosthetic hand with deep learning-based finger control. *Journal of Neural Engineering*, *18*(5).

Ortiz-Catalan, M., Håkansson, B., & Brånemark, R. (2014). Real-time and simultaneous control of artificial limbs based on pattern recognition algorithms. *IEEE transactions on neural systems and rehabilitation engineering: a publication of the IEEE Engineering in Medicine and Biology Society*, *22*(4), 756–764.

Ortiz-Catalan, M., Rouhani, F., Branemark, R., & Hakansson, B. (2015). Offline accuracy: A potentially misleading metric in myoelectric pattern recognition for prosthetic control. *Annual International Conference of the IEEE Engineering*

in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual International Conference, *2015*, 1140–1143.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc.

Piazza, C., Grioli, G., Catalano, M., & Bicchi, A. (2019). A century of robotic hands. *Annual Review of Control, Robotics, and Autonomous Systems*, *2*(1), 1–32.

Rahimian, E., Zabihi, S., Asif, A., Farina, D., Atashzar, S. F., & Mohammadi, A. (2021, September 25). TEMGNet: Deep transformer-based decoding of upper-limb sEMG for hand gestures recognition.

Rahimian, E., Zabihi, S., Atashzar, S. F., Asif, A., & Mohammadi, A. (2020). XceptionTime: Independent time-window xceptiontime architecture for hand gesture classification. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1304–1308.

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, *85*(3), 333–359.

Roche, A. D., Lakey, B., Mendez, I., Vujaklija, I., Farina, D., & Aszmann, O. C. (2019). Clinical perspectives in upper limb prostheses: An update. *Current Surgery Reports*, *7*(3), 5.

Ross, A., & Willson, V. L. (2017). Paired samples t-test. In A. Ross & V. L. Willson (Eds.), *Basic and advanced statistical tests: Writing results sections and creating tables and figures* (pp. 17–19). SensePublishers.

Saby, J. N., & Marshall, P. J. (2012). The utility of EEG band power analysis in the study of infancy and early childhood. *Developmental Neuropsychology*, *37*(3), 253–273.

Salminger, S., Stino, H., Pichler, L. H., Gstoettner, C., Sturma, A., Mayer, J. A., Szivak, M., & Aszmann, O. C. (2022). Current rates of prosthetic usage in upper-limb amputees - have innovations had an impact on device acceptance? *Disability and Rehabilitation*, *44*(14), 3708–3713.

Samuel, O. W., Grace Asogbon, M., Geng, Y., Li, X., Pirbhulal, S., Chen, S., Ganesh, N., Feng, P., & Li, G. (2019). Spatio-temporal based descriptor for limb movement-

intent characterization in EMG-pattern recognition system. *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2637–2640.

Simão, M., Mendes, N., Gibaru, O., & Neto, P. (2019). A review on electromyography decoding and pattern recognition for human-machine interaction. *IEEE Access*, *7*, 39564–39582.

Sinha, R., van den Heuvel, W. J., & Arokiasamy, P. (2011). Factors affecting quality of life in lower limb amputees. *Prosthetics and Orthotics International*, *35*(1), 90.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958.

Tkach, D., Huang, H., & Kuiken, T. A. (2010). Study of stability of time-domain features for electromyographic pattern recognition. *Journal of NeuroEngineering and Rehabilitation*, *7*(1), 21.

Tsipouras, M. G. (2019). Spectral information of EEG signals with respect to epilepsy classification. *EURASIP Journal on Advances in Signal Processing*, *2019*(1), 10.

Tuli, S., Casale, G., & Jennings, N. R. (2022). TranAD: Deep transformer networks for anomaly detection in multivariate time series data. *Proc. VLDB Endow.*, *15*(6), 1201–1214.

TU-T. (1988). Pulse code modulation (PCM) of voice frequencies.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc.

Veit, A., Wilber, M., & Belongie, S. (2016). Residual networks behave like ensembles of relatively shallow networks. *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 550–558.

Viitasalo, J. H., & Komi, P. V. (1977). Signal characteristics of EMG during fatigue. *European Journal of Applied Physiology and Occupational Physiology*, *37*(2), 111–121.

Vujaklija, I., Farina, D., & Aszmann, O. C. (2016). New developments in prosthetic arm systems. *Orthopedic Research and Reviews*, *8*, 31–39.

Vujaklija, I., Roche, A. D., Hasenoehrl, T., Sturma, A., Amsuess, S., Farina, D., & Aszmann, O. C. (2017). Translating research on myoelectric control into clinics—are the performance assessment methods adequate? *Frontiers in Neurorobotics*, *11*.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., & Ma, H. (2020, June 14). Linformer: Self-attention with linear complexity.

Wang, Y., Zhao, P., & Zhang, Z. (2023). A deep learning approach using attention mechanism and transfer learning for electromyographic hand gesture estimation. *Expert Systems with Applications*, *234*, 121055.

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., & Sun, L. (2022). Transformers in time series: A survey. *CoRR*, *abs/2202.07125*.

Wu, H., Dyson, M., & Nazarpour, K. (2021). Arduino-based myoelectric control: Towards longitudinal study of prosthesis use. *Sensors*, *21*(3), 763.

Wu, Y., Zheng, B., & Zhao, Y. (2018). Dynamic gesture recognition based on LSTM-CNN. *2018 Chinese Automation Congress (CAC)*, 2446–2450.

Wurth, S. M., & Hargrove, L. J. (2014). A real-time comparison between direct control, sequential pattern recognition control and simultaneous pattern recognition control using a fitts' law style assessment procedure. *Journal of Neuro-Engineering and Rehabilitation*, *11*(1), 91.

Xia, P., Hu, J., & Peng, Y. (2018). EMG-based estimation of limb movement using deep learning with recurrent convolutional neural networks. *Artificial Organs*, *42*(5), E67–E77.

Xiong, D., Zhang, D., Zhao, X., & Zhao, Y. (2021). Deep learning for EMG-based human-machine interaction: A review. *IEEE/CAA Journal of Automatica Sinica*, *8*(3), 512–533.

Zabihi, S., Rahimian, E., Asif, A., & Mohammadi, A. (2022, March 30). TraHGR: Transformer for hand gesture recognition via ElectroMyography.

Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., & Eickhoff, C. (2021). A transformer-based framework for multivariate time series representation learning. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2114–2124.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., & Jin, R. (2022). FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. *Proceedings of the 39th International Conference on Machine Learning*, 27268–27286.

Appendix A

Transformers

Transformers are deep ML sequence-to-sequence models that implement the attention mechanism. In the following few sections, the components that make up the transformer are individually explained and, in a final section, combined into a present-day transformer architecture.

A.1 Positional Encoding

In RNNs, the position of an input in a sequence is captured in the way that sequence is fed to the model, i.e., input by input. In contrast, transformers are designed to process complete sequences in parallel. This is beneficial from a computational point of view, but it requires that positional information is explicitly encoded in the input. There are two approaches to so-called positional encodings. Either a specific scheme is used to generate a fixed vector that encodes positional information for each position in the sequence, or such vectors are learnt as model weights (Gehring et al., 2017).

The original transformer implements fixed positional encodings generated through sine and cosine functions of different frequencies (Vaswani et al., 2017). For every position p in an input sequence of inputs of dimension n, a positional encoding PE is generated using:

$$\mathrm{PE}_{p,2i} = sin\left(\frac{p}{10000^{\frac{2i}{d}}}\right) \tag{A.1}$$

$$\mathrm{PE}_{p,2i+1} = cos\left(\frac{p}{10000^{\frac{2i}{d}}}\right) \tag{A.2}$$

where i indicates the dimension of the encoding. Thus, each dimension of the encoding is a sinusoid and for any offset k, PE_{p+k} can be represented as a linear function of PE_p.

An input vector is enriched with positional information by adding its PE vector to it. Many other possible positional encoding generation schemes exist (Gehring et al., 2017). Vaswani et al. (2017) reason that these kinds of encodings should allow models to learn to relate positions to each other.

While fixed positional encodings are effective, it is usually possible to achieve similar performance using learnt positional encodings. In this case, an $L \times d$ matrix of model weights is learnt with the model, where L denotes the maximum allowed sequence length. Each row of said matrix corresponds to a PE vector.

A.2 Attention

The attention mechanism was designed as a technique that would allow a model to learn the importance of certain features for the task at hand (Kim et al., 2022). In its most simple form, attention is a function that takes as input a sequence of vectors $\boldsymbol{x}_i, \ldots, \boldsymbol{x}_n$ of dimension d and returns a sequence of vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$ of the same length and same dimension:

$$\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n = \text{attention}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \tag{A.3}$$

The vectors $\boldsymbol{y}_i$ are weighted averages of the the vectors $\boldsymbol{x}_i$, where the weights $\boldsymbol{w}_i$ represent the 'attention' the model is assigning to a specific vector $\boldsymbol{x}_i$, i.e., the attention weights:

$$\boldsymbol{y}_i = \boldsymbol{w}_i^\top \boldsymbol{x}_i, \tag{A.4}$$

where the weights $w_i^{(d)}$ in $\boldsymbol{w}_i$ are normalised such that they sum to 1.

The power of the attention mechanism lies in the way it models the attention weights, which it does using a compatibility measure. Each input vector $\boldsymbol{x}_i$ is assigned two vectors, a 'query' vector $\boldsymbol{q}_i$ and a 'key' vector $\boldsymbol{k}_i$, both also of dimension d. The compatibility score s_{ij} between two input vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is then calculated by taking the dot product of the query vector of one and the key vector of the other:

$$s_{ij} = \boldsymbol{q}_i^\top \boldsymbol{k}_j, \tag{A.5}$$

such that the 'score' vector $\boldsymbol{s}_i$ of input vector $\boldsymbol{x}_i$ contains its compatibility scores w.r.t. to all input vectors (including itself):

$$\boldsymbol{s}_i = [s_{i1}, \ldots, s_{in}] \tag{A.6}$$

The attention weights are computed by normalising the score vectors $\boldsymbol{s}_i$ using the softmax function:

$$\boldsymbol{w_i} = \text{softmax}(\boldsymbol{s}_i), \tag{A.7}$$

with:

$$\text{softmax}(\boldsymbol{x})_i = \frac{e^{x^{(i)}}}{\sum_{j=1}^{d} e^{x^{(j)}}} \tag{A.8}$$

The calculation of attention relies thus on the query and key vectors. These are usually stacked into matrices Q and K, respectively. Depending on the maximum allowed length L of the input sequence, which is a hyperparameter, these matrices are of dimension $L \times d$. The values in these matrices, and by further computation thus also the attention weights, are learned from data using optimisation methods. Most often, the attention mechanism is followed by a fully-connected feed-forward neural network, aimed at further processing the output vectors that have been enriched with the attention information.

A.3 Layer Normalisation & Residual Connections

When training ML models, it is common practice to normalise the input features, i.e. mapping their values to $[0, 1]$, or transforming them such that they follow a distribution with zero mean and unit variance. Doing so reduces training time, as it will be easier for the gradient descent algorithm to converge. Furthermore, normalising reduces the risk of accumulating large gradients, which could lead to an unstable training process. An extension of normalisation for NNs stems from the observation that the input layer of a NN is not the only layer that receives input. Every layer of a NN receives as input the outputs of the previous layer. If those outputs' scales differ significantly, the same issues from before can occur. Hence, Ba et al. (2016) devised *layer normalisation*, where the inputs a_i of every layer l of dimension d are normalised into:

$$\hat{a}_i = \frac{a_i - \mu_l}{\sqrt{\sigma_l^2}} \tag{A.9}$$

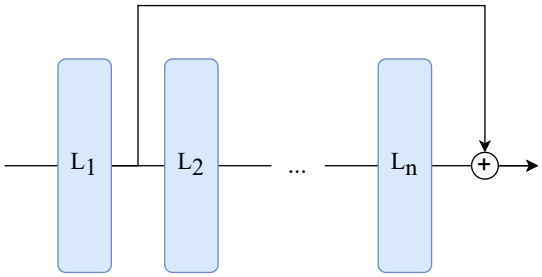


Figure A.1: Visualisation of a residual connection.

with:

$$\mu_l = \frac{1}{d}\sum_{i=1}^{d} a_i \tag{A.10}$$

$$\sigma_l^2 = \frac{1}{d}\sum_{i=1}^{d} (a_i - \mu_l)^2 \tag{A.11}$$

This normalisation step is part of the model and is performed every time input flows through the model.

In the transformer architecture, layer normalisation is usually preceded by a *residual connection*. Such a connection allows the data flow to skip certain layers, as depicted by Figure A.1. The outputs of the layers that are skipped are summed with the outputs that did the skip. Veit et al. (2016) explain that the different paths that residual connections introduce in NNs can be unraveled into ensembles of smaller NNs that do not strongly depend on each other. Since a big part of the gradients in gradient descent originate from short paths, residual connections do not resolve the vanishing/exploding gradients problem, but they avoid it by introducing shallow subnetwork ensembles.

A.4 Architecture

Figure A.2 depicts a present-day transformer-like architecture that implements positional encodings, the attention mechanism, layer normalisation, and residual connections. A hyperparameter of the architecture is the depth of the model, which denotes the number of 'encoder' blocks that follow each other in sequence. This type of transformer architecture is often called an 'encoder-transformer', because it only consists of encoder blocks (Devlin et al., 2019). The original transformer architecture also implements 'decoder' blocks, which are intended to decode the encoded outputs of the encoder blocks (Vaswani et al., 2017). Architectures that are seen as 'decoder-transformers' exist as well (Brown et al., 2020). However, whether or not the architec-

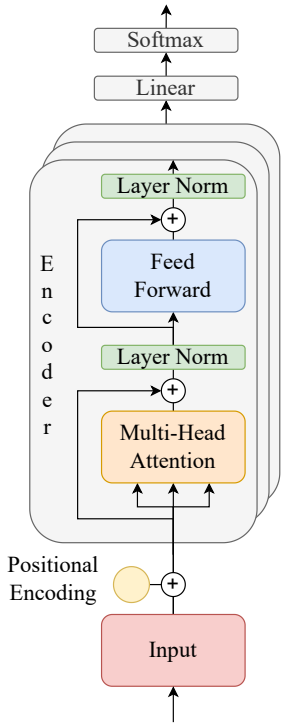


Figure A.2: Architecture diagram of a traditional transformer-encoder. Figure after those of Vaswani et al. (2017).

ture is an encoder or a decoder is a matter of what it is trained for, rather than how it works. When using the encoder or decoder blocks on their own, architecturally, there is no difference between them.

The transformer architecture comes with some considerations. For example, it is often the case that the encoder blocks implement multi-head attention, an extension of the attention mechanism that comes down to modelling multiple attention mechanisms in parallel. Multi-head attention allows for more than 1 set of attention weights, each of which can learn to attend to different aspects in the data. Another consideration is the location of the residual connections and layer normalisation. In the original transformer, these two are positioned after the attention mechanism and feed-forward neural network. More recent models often place them before (Brown et al., 2020), however, there is no real consensus as to which is better. Lastly, depending on the task the architecture is intended for, the output layers can differ. If the task is a regression task, the outputs of the last encoder block go through a linear layer that has the required output dimension. In the case of classification, the linear layer is followed by a softmax layer to get normalised class probabilities.

Appendix B

Resources

The processing and model training code is available on GitLab[1], the README gives an overview of how to use it. The data can be downloaded from the Ninapro website[2], or using the download script from the repository. The repository further contains seeded configuration files that allow for reproducing all my evaluations. I can provide the models' weights and metadata upon request. Lastly, an interactive dashboard that presents all the models' training/validation curves and testing results is available on Weights&Biases[3].

[1] https://git.ecdf.ed.ac.uk/s2408107/EMG-DAD-transformer
[2] http://ninapro.hevs.ch/DB8_Instructions
[3] https://api.wandb.ai/links/wulfdewolf/u15bse2e

Appendix C

Model Training

Every participant has their own model. Due to the relatively small scale of the models I opted for a fixed number of epochs, instead of an early-stopping approach. I ran preliminary experiments to determine a number of epochs sufficiently large for all configurations to converge and decided on 40 as a good value. I train using mini-batch gradient descent through *PyTorch*'s Adam optimiser with the weight decay parameter for L_2-regularisation (Paszke et al., 2019). All training happened deterministically (and with set seeds for reproducibility), in full precision, and optimised for performance using *PyTorch*'s compile method.

Appendix D

Hyperparameter Optimisation

I optimise hyperparameters per participant, considering the follow hyperparameters and respective possible values:

- **Mini-batch size:** uniform choice from $\{32, 64, 128, 256, 512\}$.
- **Learning rate:** log-uniform choice from [0.0001, 0.1].
- **Weight decay:** log-uniform choice from [0.0001, 0.1].
- **Activation function:** uniform choice from $\{`relu', `gelu'\}$.
- **Dropout rate:** log-uniform choice from [0.0001, 0.1].
- **Embedding size:** uniform choice from $\{64, 128, 256, 512\}$.
- **Number of attention heads:** uniform choice from $\{2, 4, 8, 16\}$.
- **Number of encoder blocks:** uniform choice from $[1, 10]$.

For the FULL configuration, I only optimise the first three hyperparameters, the other hyperparameter values are taken from the individually optimised submodules.

I run 20 HPO trials per model, where 5 trials are always running concurrently. Choices for new trials are guided using tree-structured parzen estimators (TPE; Bergstra et al., 2011).

I use the *Ray-Tune* framework (Liaw et al., 2018) with the *Optuna* package's implementation of TPEs (Akiba et al., 2019). I performed all HPO on the Cambridge Service for Data Driven Discovery (CSD3) high performance computing cluster.

Appendix E

Positional Encodings

The positional encoding similarity figures on the following page accompany those in Section 5.2.3.

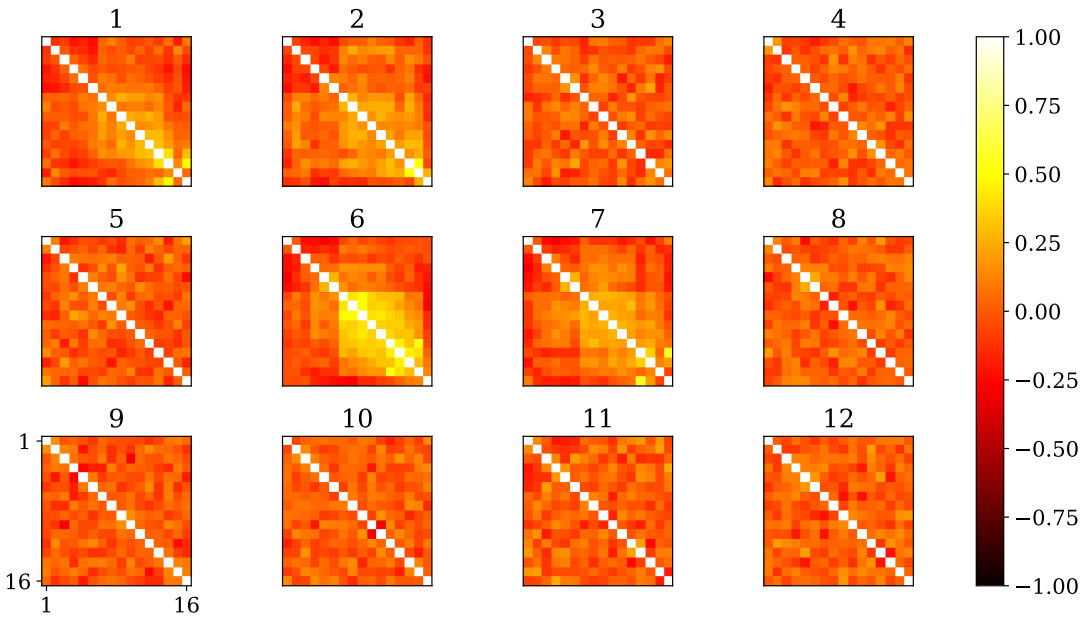


Figure E.1: Positional encoding similarities for the FNET-MIX configuration for all participants. Each row in each figure represents the cosine similarity between one position and all the other positions.

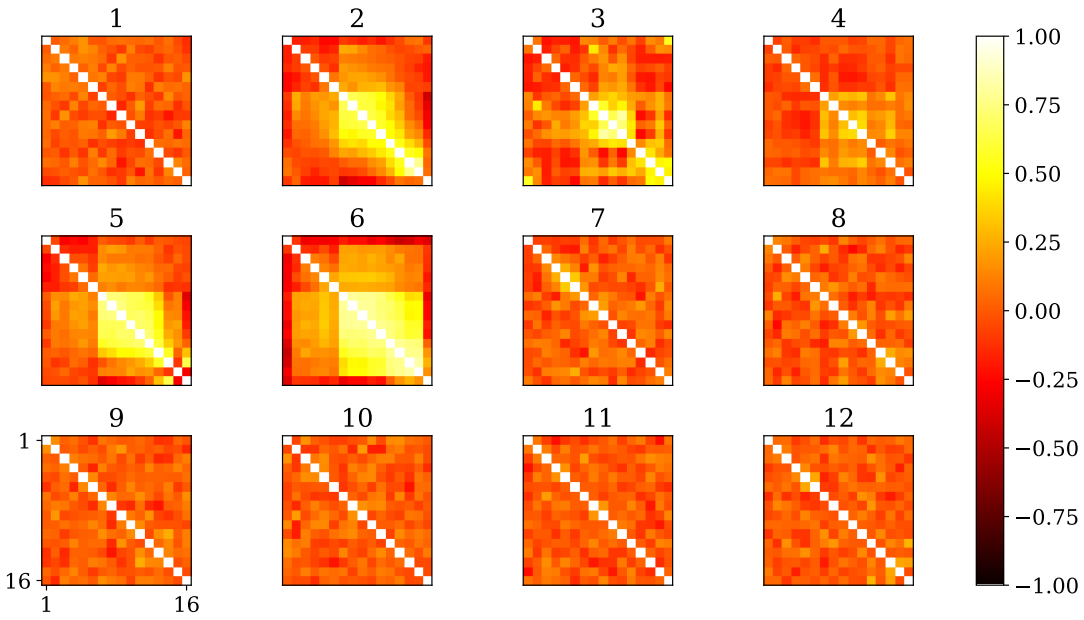


Figure E.2: Positional encoding similarities for the FNET-MIXL configuration for all participants. Each row in each figure represents the cosine similarity between one position and all the other positions.