JOINT FE-DNN MODELLING FOR TMS-INDUCED EMG ACTIVATIONS

AUTHOR: MARC MARTÍNEZ GOST

DENIZ ERDOĞMUŞ

ADVISORS: DANA BROOKS

XAVIER GIRÓ-I-NIETO

UNIVERSITAT POLÍTÈCNICA DE CATALUNYA

Escola Tècnica Superior d’Enginyeria de Telecomunicació de Barcelona

and

NORTHEASTERN UNIVERSITY

Department of Electrical and Computer Engineering

BOSTON, JUNE 2019
Abstract

Considering the vast amount of degrees of freedom that the Central Nervous System holds, synergies, understood as building blocks led by the linear combination of multiple muscles, a novel hypothesis to understand the generation of movement. This project suggests a Deep Learning approach to build two predictive kinematic model based on muscles and synergies. Both models will be built with Convolutional Neural networks, whose inputs will be Transcranial Magnetic Stimulations and the outputs, the evoked muscle responses monitored through Electromyography. By comparing the predictions from both models, the results evinced that synergies propose a more robust model to understand movement control than direct connections to muscles, even though they are not able to fully characterize the movement by themselves.
Resum

El Sistema Nerviós Central concentra una gran quantitat de graus de llibertat a l’hora de generar moviments. Per això, la inclusió de les sinèrgies, entesos com elements bàsics caracteritzats per la combinació lineal de múltiples músculs, és una hipòtesi innovadora per entendre la concepció del moviment. Aquest projecte suggereix una estratègia basada en Deep Learning per tal d’elaborar dos models predictius que expliquin el moviment basat en músculs i sinèrgies. Ambdós models es construiran mitjançant Xarxes Neuronals Convolucionals, les entrades de les quals seran Estimulacions Magnètiques Transcraniales i les sortides, les respostes evocades als músculs i monitoritzades amb Electromiografia. La comparativa dels resultats obtinguts demostra que les sinèrgies proposen un model més robust per entendre el control del moviment que les connexions directes als músculs, encara que no són capaces de caracteritzar el moviment per elles soles.
Resumen

El Sistema Nervioso Central concentra una gran cantidad de grados de libertad a la hora de generar movimiento. Por eso, la inclusión de las sinergias, entendidas como elementos básicos caracterizados por la combinación lineal de múltiples músculos, es una hipótesis innovadora para entender la concepción del movimiento. Este proyecto sugiere una estrategia basada en Deep Learning con el propósito de elaborar dos modelos predictivos que expliquen el movimiento basados en músculos y sinergias. Ambos modelos se construirán mediante Redes Neuronales Convolucionales, cuyas entradas serán Estimulaciones Magnéticas Transcraneanas y las salidas, las respuestas evocadas a los músculos y monitorizadas con Electromiografía. La comparativa de los resultados obtenidos demuestra que las sinergias proponen un modelo más robusto para entender el control del movimiento que las conexiones directas a los músculos, aunque no son capaces de caracterizar el movimiento por sí mismas.
Acknowledgements

I would like to express my special thanks of gratitude to Deniz Erdoğmuş, Dana Brooks and Xavier Giró-i-Nieto, for offering me this opportunity to join the SPIRAL research team and learning from their experience and passion. I am particularly grateful for the assistance given by Mathew Yarossi in developing this project.

I would also like to extend my thanks to my family: to my mother, my father and my sister for their encouragement throughout my life and being a part of my getting here. Finally, I wish to thank Maryan, whose love and guidance are with me in whatever I pursue.
# Revision history and approval record

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>03/05/2019</td>
<td>Document creation</td>
</tr>
<tr>
<td>1</td>
<td>17/05/2019</td>
<td>Document revision</td>
</tr>
<tr>
<td>2</td>
<td>28/05/2019</td>
<td>Document revision</td>
</tr>
<tr>
<td>3</td>
<td>06/06/2019</td>
<td>Document revision</td>
</tr>
<tr>
<td>4</td>
<td>11/06/2019</td>
<td>Document approbation</td>
</tr>
</tbody>
</table>

## DOCUMENT DISTRIBUTION LIST

<table>
<thead>
<tr>
<th>Name</th>
<th>e-mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marc Martínez Gost</td>
<td><a href="mailto:marctinez97@gmail.com">marctinez97@gmail.com</a></td>
</tr>
<tr>
<td>Deniz Erdoğan</td>
<td><a href="mailto:erdogmus@ece.neu.edu">erdogmus@ece.neu.edu</a></td>
</tr>
<tr>
<td>Dana Brooks</td>
<td><a href="mailto:brooks@ece.neu.edu">brooks@ece.neu.edu</a></td>
</tr>
<tr>
<td>Xavier Giró-i-Nieto</td>
<td><a href="mailto:xavier.giro@upc.edu">xavier.giro@upc.edu</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Written by:</th>
<th>Reviewed and approved by:</th>
<th>Reviewed and approved by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>06/06/2019</td>
<td>Date</td>
</tr>
<tr>
<td>Name</td>
<td>Marc Martínez Gost</td>
<td>Name</td>
</tr>
<tr>
<td>Position</td>
<td>Project Author</td>
<td>Position</td>
</tr>
</tbody>
</table>
Contents

1. Introduction ...........................................................................................................9
   1.1. Background of the Problem .........................................................................9
   1.2. Objectives and Methods ...........................................................................9
   1.3. Previous Work ...........................................................................................10
   1.4. Requirements and Specifications .............................................................11
       1.4.1. Framework used ..............................................................................11
       1.4.2. GPU Power ......................................................................................12
   1.5. Work Plan ....................................................................................................12
       1.5.1. Work Packages ..............................................................................12
       1.5.2. Gantt Diagram ...............................................................................12
       1.5.3. Deviations and Incidents .................................................................13
2. State of the art ......................................................................................................14
   2.1. Convolutional Neural Network .................................................................14
   2.2. Autoencoder ...............................................................................................15
   2.3. Non-negative Matrix Factorization ............................................................16
3. Methodology .........................................................................................................18
   3.1. Objective .....................................................................................................18
   3.2. Dataset .........................................................................................................18
   3.3. Autoencoder ................................................................................................19
       3.3.1. Architecture .....................................................................................20
       3.3.2. Pre-processing ..................................................................................22
       3.3.3. Training .........................................................................................22
   3.4. DNN-MEP ..................................................................................................23
       3.4.1. Architecture .....................................................................................24
       3.4.2. Pre-processing ..................................................................................24
       3.4.3. Training .........................................................................................24
   3.5. DNN-SYN ..................................................................................................25
3.5.1. Architecture........................................................................................................25
3.5.2. Pre-processing........................................................................................................25
3.5.3. Training..................................................................................................................26

4. Results ..........................................................................................................................29
  4.1. Experimental setup....................................................................................................29
  4.2. Metrics......................................................................................................................29
  4.3. Results.....................................................................................................................30
    4.3.1. Autoencoder.......................................................................................................30
    4.3.2. DNN-MEP..........................................................................................................32
    4.3.3. DNN-SYN..........................................................................................................33

5. Budget ..........................................................................................................................34

6. Conclusions and future development...........................................................................35

Bibliography.....................................................................................................................36

Glossary.............................................................................................................................38
List of Figures

1.1 The theoretical and methodological scope of the whole team project...............11
1.2 Gantt diagram of the project..............................................................................14
2.1 Example of an Artificial Neural Network........................................................15
2.2 Example of a Convolutional Neural Network.....................................................16
2.3 Example of an Autoencoder.................................................................................16
2.4 Non-negative Matrix Factorization structure.......................................................17
3.1 Scheme of the project representing the mapping to MEPs (top) and to SYN... (bottom)..............................................................................................................19
3.2 Top view of a raw input sample (left) and after masking it (right)......................20
3.3 30 different MEP resulting from an EMG..........................................................20
3.4 One sample producing activations (left) and a similar sample with no activation (right) .................................................................................................................21
3.5 Architecture of the autoencoder..........................................................................22
3.6 AE reconstruction before applying the M1 mask (left) and after applying it (right) .........................................................................................................................23
3.7 Autoencoder model order comparison..................................................................24
3.8 Ground truth MEPs (top) and estimation from NMF results (bottom)..............27
3.9 Basis (left) and Activation (right) matrices from NMF over the MEP matrix.....27
3.10 Architecture of DNN-MEP...................................................................................29
3.11 Architecture of DNN-SYN...................................................................................29
4.1 Original sample (left), latent representation (center) and its reconstruction (right) .........................................................................................................................31
4.2 Histogram of the reconstruction error of the autoencoder.................................32
4.3 Original sample (left), latent representation (center) and spatial downsampling (right) ..................................................................................................................32
4.4 Ground truth MEP activations (top) and predictions (bottom) for DNN-MEP...33
4.5 Ground truth MEP activations (top) and predictions (bottom) for DNN-SYN....34
List of Tables

3.1 Hyperparameter tuning scheme for DNN-MEP ..................................................25
3.2 Hyperparameter tuning scheme for DNN-SYN ..................................................27
5.1 Budget of the project .........................................................................................34
1. Introduction

1.1. Background of the Problem

When the Central Nervous System (CNS) generates a movement, the involved muscles are activated by hundreds of motor units that need to be coordinated. Although the brain is capable of controlling muscles individually, empirical evidence [1-4] shows that there are many degrees of freedom: in an experiment conducted by Bizzi et al. [1], the spinal cord of a frog was excited with microstimulations. As distinct areas were stimulated, its legs did not produce the same range of diverse movements. Additionally, the rest of the kinematics were achieved when several areas were coactivated. Therefore, how does the primary motor cortex (M1) coordinates the muscles in order to generate movements?

One reasonable hypothesis [5] to explain that dimensionality reduction resides in this complexity being lessened by controlling discrete muscle groups known as synergies. These physiological structures represent basic building blocks whose combination may explain the existence of such vast repertoire of movements.

However, in little literature [6] this hypothesis has been contrasted predictively, only descriptively. Hence, this project aims to define a predictive framework by modelling a Deep Neural Network (DNN), which, from a stimulation, will be able to forecast which muscles and synergies will be activated. Our hypothesis is that synergies will generalize better than direct connections to muscles, whose error in predicting which muscles are expected to be activated will be smaller.

Consequently, understanding the relationship between the spatial organization of the M1 and the activation of muscles could mean a significant progress in understanding medical conditions such as how the brain is affected after suffering from a stroke and how it recovers [7, 8].

1.2. Objectives and Methods

This project belongs to a wider line of research composed of three different aims:

- **Aim 1**: the most suitable way to excite the brain through non-invasive methods is Transcranial Magnetic Stimulation (TMS). Nevertheless, current techniques are outdated and consume a lot of resources, meaning that a session to obtain samples for the dataset may imply high cost and time. So, the first approach of the research is trying to develop novel ways of sampling the M1 with the help of Active Learning.

- **Aim 2**: this aim deals with how the evoked responses in the muscles through TMS and voluntary movements are alike.
**Aim 3**: using a forward modelling of the M1, evoked muscle responses by TMS will be monitored with Electromyography (EMG) and trained with a DNN in order to build a predictor. The objective is to compare a model predicting the muscles activations to synergy activations, both generated from a TMS stimulation. Secondly, an inverse modelling is proposed in order to map a distribution of the synergies and individual muscles over the brain.

![Diagram of Aim 1, 2, and 3](image)

*Figure 1.1. The theoretical and methodological scope of the whole team project.*

This project is only focusing in the forward model of the Aim 3: TMS generates a magnetic field that induces an electrical current in the specific region of the M1 associated to hand control. This evokes a multi-muscle response which is recorded using Electromyography (EMG) [9]. Therefore, the dataset contains 3D maps of the induced Electrical fields generated with a Finite Element (FE) modelling (inputs) and its corresponding EMG responses (labels).

The main question we will try to answer is: are synergies prevailing features of motor control? Firstly, one DNN will be trained to map those 3D images directly to the muscle responses. On the other hand, the synergies will be extracted from the EMG responses using Non-negative Matrix Factorization and the inputs will be mapped to these new labels. Finally, the generalization errors from both predictors will be compared in order to contrast the results and confirm or reject the initial hypothesis stated above.

The whole dataset was previously obtained by other members of the team and given to me to train the neural networks.

### 1.3. Previous Work

In 1967 Bernstein was a pioneer in neurophysiology, and all his understanding in the field of motor control supported the existence of a high number of degrees of freedom that the CNS manages [10]. However, not until the early 90’s the first experiments testing that hypothesis
started to appear. Bizzi is a renowned neuroscientist for making extensive progress in the understanding of the CNS signalling and how synergies are fundamental blocks of movement [1-5, 7]. He is the responsible of the first experiment involving invasive microstimulations in a frog, where the inclusion of synergies evinced a reduction in the degrees of freedom [1]. After that, more experimental results proven in macaques and cats only reaffirmed the existence of these building blocks.

From that, many mathematical methods were used in order to extract the synergies resulting from the muscle activation. However, these algorithms produced dissimilar results, indicating that these synergies had to be interpreted physiologically different. This is because the EMG signal has to be modelled and the results may change depending on the assumptions [11]. In order to understand the behaviour of all the factorization methods (PCA, ICA, NMF, FA, etc.) d’Avella et al. [12] developed a rigorous analysis where they compared all these manners of extracting synergies in different datasets. They detected that PCA provided less reliable results in comparison with the other methodologies, whereas ICA and NMF performed similarly in datasets with the presence of additive gaussian noise. However, the fact is that the noise in an EMG signal is signal dependant and, in that scenario, NMF provided the most robust results [13].

The increasing interest in this topic has influenced the development of prosthetic implants like Freehand [14, 15] which takes advantage of the synergy modelling of the kinematics.

1.4. Requirements and Specifications

The project Joint Finite Element and Deep Network Modelling for TMS-induced EMG Activations is carried out at the Electrical and Computer Engineering Department of Northeastern University, in the Signal Processing, Imaging, Reasoning, and Learning (SPIRAL) researching group [16].

The requirement of this project is to provide a system approaching the mapping of the M1 to the activations of the muscles. To do so, two forward Deep Neural Network (DNN) will be trained to predict which muscles and synergies are activated when a TMS stimulation is induced.

1.4.1. Framework used

The main programming language used to develop the entire project is Python [17], and Keras [18] has been the chosen platform to implement the Deep Learning architectures and algorithms. Keras is a high-level API written in Python and capable of running on top of TensorFlow [19]. Additionally, MATLAB [20] has been also used to pre-process the data and display results.
1.4.2. GPU Power

CUDA libraries [21] were also required to run the code on a GPU: since the samples used are large, a GPU is almost indispensable to obtain results faster. A laptop with a built-on GPU was used (Nvidia GTX 850M [22]) in parallel with the Discovery Cluster [23] of the Northeastern University.

1.5. Work Plan

In this section, the project organization is discussed. The tasks have been grouped in packages and organized in a time diagram. Finally, the workflow is assessed in order to examine the delays and setbacks encountered during the development.

1.5.1. Work Packages

- **WP 1**: Project Documentation
  - State-of-the-art research
  - Project proposal

- **WP 2**: Deep Learning Techniques
  - Python and Keras
  - State-of-the-art research
  - Deep Neural Networks
  - Autoencoders and CNN

- **WP 3**: Dataset Analysis
  - Working with the dataset

- **WP 4**: Experimentation
  - Architectures
  - Critical Review

- **WP 5**: Final Review and Presentation
  - Final Review
  - Presentation

1.5.2. Gantt Diagram

Figure 1.2 represents the tasks breakdown in a timeline:
1.5.3. Deviations and Incidents

The only deviation in the project was found in the WP 3: working with the dataset was conceived a previous process to building the architectures. However, the results obtained from the DNNs gave us an intuition about the nature of the dataset. This means that the dataset analysis is an iterative process where the obtained results may give you new insights.

On the contrary, even though the dataset inputs have four dimensions (one indicating the sample, and three for the E-field), some Python packages were not implemented to use tensors [24], meaning that I had to implement those functions by myself. However, the Gantt diagram was conceived with time buffers in order to assess these setbacks. In conclusion, due to an appropriate planning, there were no delays.
2. State of the art

In order to understand the methods and techniques that have been implemented, the following section provides a description of the main building blocks used to tackle the problem. The reader is referred to CS231n: Convolutional Neural Networks for Visual Recognition course [25] by Stanford University and to chapter 14 of Deep Learning [26] by Goodfellow et al. for a deeper explanation.

2.1. Convolutional Neural Network

Artificial Neural Networks (ANN) are a class of Machine Learning techniques whose basis is the neuron. The mechanism resides in an input variable being weighted, a bias term added to the result and an activation (a non-linear function) being applied. Finally, this result is fed to forward neurons. Consequently, neurons are distributed in layers: the input layer, the hidden layers and the output layers. Neural Networks learn because the weights and biases are trainable parameters that the network tunes through a learning process known as backpropagation, which basically computes the gradients of the learnable parameters in order to minimize a cost function.

![Example of an Artificial Neural Network](figure.png)

*Figure 2.1. Example of an Artificial Neural Network. Source: Good Audience.*

Convolution Neural Networks (CNN) were conceived by LeCun [27]. They belong to the family of ANNs and take advantage of the correlation between pixels existing in images. CNNs can be understood as a regularized version of ANNs, since weights are encapsulated in filters that will be applied to the whole image. There are several types of layers in a CNN:

- **Input layer**: containing the input image.
- **Convolutional layer**: it applies a spatial filtering in a local region of the input volume with the objective of extracting features.
- **Pooling layer**: spatial downsampling that also reduces the dimensions.
- **Fully Connected (FC) layer**: as in ANNs, usually being the last layers.
Figure 2.2. Example of a Convolutional Neural Network. Source: Research Gate.

In this project the E-fields can be understood as 3D images and, hence, CNNs are the most appropriate architectures to work with the dataset.

2.2. Autoencoder

An autoencoder (AE) is a neural network that replicates the input in its output, i.e., it computes the identity function. The AE is composed of two elements: an encoder function $h = f(x)$ and a decoder that reconstructs the original input $r = g(h)$. Hence, $g(f(x)) = x$ [26].

Figure 2.3. Example of an Autoencoder. Source: Toward Data Science.

As it can be seen in Figure 2.3, since the code produced by the encoder, $h$, has less dimensions than the input, $x$, the decoder will not be able to reconstruct the input identically. Therefore, the AE will need to learn to extract the most important features of the input in order to compress the information and obtain the best reconstruction. This is known as an undercomplete autoencoder because the constraint is directly imposed by the dimension of the latent representation $h$. Otherwise, there are alternative ways of constraining the problem, such as using regularization [28].
There is one intuition about why it would be rational to apply an AE to the E-fields: the E-fields generated by TMS are 64-cubed matrices that depend on the amplitude of the signal, the location of the coil, and its orientation, seven parameters in total. Hence, it makes sense that these 64-cubed E-field matrices could be compressed to only those seven parameters.

In addition to that, taking advantage of an AE is necessary for the dataset: novel architectures, like Inception [29] or ResNet [30], that are usually presented in the Conference on Computer Vision and Pattern Recognition to exceed existing CNNs, are trained with 28x28 images in a dataset of one million samples. This is to obtain a good representation of the classes and prevent overfitting. However, this dataset only contains 300 images and, moreover, with a size of 64x64x64. Since the dataset cannot be increased, the remaining alternative is trying to reduce the input size. In conclusion, an autoencoder will be implemented to encode a latent representation of the E-field inputs with a smaller input size. Additionally, the chosen architecture will be an undercomplete Convolutional-Autoencoder: a CNN, as explained before, is the most appropriate architecture to work with images and extract useful features, and an undercomplete AE enables us to control the size of the latent encoding.

2.3. Non-negative Matrix Factorization

Non-negative Matrix Factorization (NMF) is an algorithm used to approximate one matrix as a product of two, $M \approx BA$, being all the elements non-negative.

![Non-negative Matrix Factorization structure.](image)

The algorithm tries to minimize the distance between the original matrix and its approximation. The distance is understood as the Frobenius Norm. Among all the optimization methods, the one used in this project is the Multiplicative Update: optimizing for both matrices comes with the difficulty of solving a non-convex problem. However, the problem is convex when solving for only one of the variables: the multiplicative update method optimizes with respect to one variable while keeping the other one fixed, and vice versa.

$$B,A = \arg\min_{B,A} \|M - BA\|_F^2, \quad s.t. \quad B,A \geq 0$$
In this context, the matrix $M$ represents the EMG responses of the hand muscles, $B$ represents the synergy basis and $A$, the activation of those synergies. Therefore, applying NMF to the EMG responses means that a muscle activation can be understood as the linear combination of the activations of several synergies. The NMF modelling is accurate in terms of describing variables that have a physiological sense, but it also permits the desired interpretation of describing a linear combination, since each element of $M$ is a superposition of the elements of $B$ weighted by the elements of $A$ [31]. This method allows us to obtain the labels to train the second neural network, which will map the 3D E-fields to those synergies.

The indexation used in this project is $m$ for muscles (15), $t$ for trials (300) and $s$ for synergies (9). There are 9 synergies because is the minimum number of muscle clusters that explain the EMG responses reconstructions with a 90% of the variance. This is a habitual measure to set the number of synergies in most of the literature [6, 32]. Additionally, NMF is a robust method when the number of EMG channels, i.e., muscles recorded is larger than the number of synergies [12], which here it is the case.
3. Methodology

3.1. Objective

This project aims to study the impacts and trade-offs of understanding synergies as elemental building blocks for the M1 to generate movement. In order to do that, Deep Neural Networks will be used to model TMS activations and link them to EMG responses in two different ways: by connecting them directly to the muscles, or by connecting them to the synergies corresponding to that muscles. Explicitly, this analysis aims to gain deeper understanding on how the CNS works and how synergies could justify different phenomena in medicine, in order to elaborate new insights in the field of physiology.

![Diagram](image.png)

*Figure 3.1. Scheme of the project representing the mapping to MEPs (top) and to SYN (bottom).*

3.2. Dataset

This project is focused on the M1 sector controlling hand movements. As a supervised machine learning problem, the dataset contains input-output pairs that will be used to train the networks.

The inputs consist of 300 E-fields samples produced by stimulating the brain with TMS. The result is 64-cubed matrices representing the intensity of the electrical field. However, this data will be masked in order to only consider the M1 region. Figure 3.2 represents the top view of one sample and its masking.
The outputs consist of the 300 corresponding EMG activations, usually referred as Motor Evoked Potentials (MEP) understood as the movement generated by the TMS stimulation. The result is a 300x15 matrix, since 15 muscles were monitored and whose values represent the intensity of activation of each muscle. Figure 3.3 illustrates 30 different activations.

3.3. Autoencoder

As explained before, this data defines a small dataset which will probably hinder the learning of the neural networks. As seen in Figure 3.4, apparently similar inputs can produce distinct outputs. Hence, the AE can facilitate and accelerate the learning by compressing the inputs and representing what makes each sample unique from the others.
Figure 3.4. One sample producing activations (left) and a similar sample with no activation (right).

3.3.1. Architecture

A CNN has been chosen to model the autoencoder. We impose the output to be a 16-cubed compressed version of the 64-cubed input, meaning that the resulting data will have 64 times less information. This would be equivalent to reducing the 64-sided cube to a 64-sided square, i.e., supressing one dimension. Notice that the output of the autoencoder is still the input of the dataset, that is, the input of the following CNNs.

The encoder contains three convolutional blocks, as it can be seen in figure 3.5:

- The first layer contains 32 3x3x3 filters to perform convolutions and a posterior Exponential Linear Unit (ELU) activation, followed by a max pooling layer, which downsamples the input by a factor of 2.
- The second layer contains 16 3x3x3 filters to perform convolutions and a posterior Exponential Linear Unit (ELU) activation, followed by a max pooling layer which downsamples the input by a factor of 2.
- The third layer contains only one 1x1x1 filter to perform convolutions and a posterior Exponential Linear Unit (ELU) activation.

The decoder contains the same conceptual layers but producing the reverse process:

- The first layer contains 16 3x3x3 filters to perform convolutions and a posterior Exponential Linear Unit (ELU) activation, followed by a upsampling layer with a factor of 2.
- The second layer contains 32 3x3x3 filters to perform convolutions and a posterior Exponential Linear Unit (ELU) activation, followed by a upsampling layer with a factor of 2.
- The third layer contains only one 1x1x1 filter to perform convolutions and a posterior linear activation, which is typically used for regression tasks.
- The fourth layer contains the mask, which is a deterministic operation.
As studied by Clevert et al. [34] the ELU activation has been proved to work in autoencoders and outperforms the ReLU function for several reasons: firstly, the positive section is identical to a ReLU function, meaning that is prevents the gradients from vanishing. However, its negative part pushes the activation to zero, decreasing the bias shift, which is what batch normalization also does. Besides, it is robust to noise. So, in conclusion the ELU function boosts learning and avoids the inclusion of a batch normalization layer, since it prevents the data from having a bias.

The 1x1x1 convolutions are usually used for dimension reduction [29]. This is because, spatially, they do not alter the data, they only affect in the depth dimension, that is, varying the number of filters. Here it is used to obtain a latent representation of the data with only one filter.

**Figure 3.5. Architecture of the autoencoder.**
Finally, the same mask used to pre-process the data (only focusing the M1 zone) is used at the end of the reconstruction. This is because convolutions will inevitably create tails and, by doing so, all the generated data outside the region of interest will be suppressed. However, this layer affects all the learning process by accelerating it, as the AE now does not need to reconstruct the shape of the input data, which is done by the mask, only the amplitude of the reconstruction. As it can be seen in figure 3.6, this is what the AE reconstructs with the inclusion of the mask layer. Also, see that the amplitude reconstruction is smooth, which has a physical interpretation of an Electrical field attenuating. It boosts learning too, because of the smoothness.

![Unmasked reconstruction](image1) ![Masked reconstruction](image2)

*Figure 3.6. AE reconstruction before applying the M1 mask (left) and after applying it (right).*

### 3.3.2. Pre-processing

Since the M1 section is the only region that matters containing information, the inputs will be processed by applying a mask that will zero all the pixels outside the zone of interest. Additionally, these inputs will be normalized to speed up learning. Note that the mask was also given to me by other members of the team.

### 3.3.3. Training

The data was split in a train test (80%) and a test set (20%), whose inputs-output pairs are the E-fields. The training was divided in two steps: the first one is a model order selection, involving selecting the number and size of the filters. The second stage takes into consideration the training hyperparameters: optimizer, learning rate, momentum, number of epochs and batch size. Splitting the optimization in two steps may finish in a suboptimal result. Nevertheless, it simplifies drastically the time required to train the CNN.

For the first stage, a 5-fold Cross-Validation (CV) method was used, since the dataset is not big enough to generate a development set. Therefore, the training set is divided into five segments from which, four of them are used as training and the left one, as test. This
processed is repeated five times in order to use each of the segmentations as a test set. The optimizer used is Adam, which is insensitive to hyperparameter tuning [33] and during 50 epochs. Figure 3.7 shows how 16 models were compare, whose reconstruction error were compared. The models were tuned with grid-search and had varying number of filters (16 and 32) and varying filter sizes (3 and 5) for the two encoding layers (see that the decoder has the same parameters but in reverse order). By inspecting the plots in colors, note that they all reflect the same quadratic shape: a small and shallow architecture produces underfitting, whereas a large and deep architecture, overfitting. Therefore, there is an overall optimal architecture, which has the lowest mean and variance reconstruction error, which is formed by only 3x3x3 filters and 32 and 16 filters, respectively.

![Model Comparison](image)

*Figure 3.7. Autoencoder model order comparison.*

This particular model was trained with Stochastic Gradient Descent (SGD), with an initial learning rate of 0.05, a momentum equal to 0.9 and for 200 epochs. The learning rate was scheduled to 0.005 for the last 100 epochs.

Since this is a regression task, the loss function used for the autoencoder is Mean Squared Error (MSE), which focuses on the reconstruction error, the squared distance between the ground truth and the prediction. On the other hand, as tested in several Deep Neural Networks [34], a batch size of 4 has produced the best generalization error and stability while training.

### 3.4. DNN-MEP

The first DNN will be used to map TMS-induced activations to individual muscles directly. By doing so, we are building a predictor based on a regression model that will learn which muscles will be activated from a given stimulation.
3.4.1. Architecture

In order to extract features, a CNN architecture has been implemented. This is deeper than wider for two reasons: the number of filters per layer has been maintained small since the network overfits easily, though it is deep in order to extract as many features as possible. As seen in Figure 3.10, the architecture is as follows:

- The first convolutional block contains 2 identical layers: 8 3x3x3 filters, a batch normalization layer and a ReLU function. It is followed by a max pooling layer, which downsamples the input by a factor of 2.
- The second convolutional block contains 2 identical layers: 16 3x3x3 filters, a batch normalization layer and a ReLU function. It is followed by a max pooling layer, which downsamples the input by a factor of 2.
- The third convolutional block contains 2 identical layers: 32 3x3x3 filters, a batch normalization layer and a ReLU function. It is followed by a max pooling layer, which downsamples the input by a factor of 2.
- The output is flattened towards two FC layers the architecture: the first one contains 1000 neurons and the second one, 100. These also contain dropout layers to regularize the CNN and whose rates are 0.5 and 0.2 respectively.
- The last FC layer contains 15 neurons, as a 15-feature output is expected, and a sigmoid function to perform regression.

3.4.2. Pre-processing

The only pre-processing mechanism used is the same as the implemented in the AE: the input data, now the compressed E-fields, will be normalized in the 0-1 scale to boost learning.

3.4.3. Training

The dataset was split in a train test (80%) and a test set (20%), whose input-output pairs were the compressed E-fields resulting from the MEP matrix, respectively. The training focused on optimizing the given architecture to obtain the best performance. A 5-fold CV was also implemented due to the size of the dataset.

Since it is a regression task, the loss function is still MSE, however the optimizer is Adagrad [35]. In order to tune the hyperparameters, a random-search method was implemented. This algorithm has proven to be more efficient and avoid repeating experiments in which there are dependencies [36]. It is important to consider that most of these hyperparameters were randomly selected in a uniform probability distribution. However, the learning rate has a largest scale with two orders of magnitude. This means that it is better to perform random search over a logarithmic scale. Table 3.1 shows the hyperparameter tuning scheme. Finally, the batch size was 48.
### Table 3.1. Hyperparameter tuning scheme for DNN-MEP.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Range</th>
<th>Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>$1 \cdot 10^6$ - $1 \cdot 10^2$</td>
<td>$4 \cdot 10^4$</td>
</tr>
<tr>
<td>Epochs</td>
<td>100 - 1000</td>
<td>200</td>
</tr>
<tr>
<td>Dropout 1</td>
<td>0.2 - 0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Dropout 2</td>
<td>0.0 - 0.3</td>
<td>0.2</td>
</tr>
</tbody>
</table>

#### 3.5. DNN-SYN

The second DNN will be used to map TMS-induced stimulations to muscle synergies. These synergies will be extracted from the MEP activations via NMF. By doing so, we are building a predictor based on a regression model that will learn which synergies will be activated from a given stimulation.

##### 3.5.1. Architecture

A CNN architecture has also been chosen to model the DNN-SYN. The implemented structure is similar in composition to the DNN-MEP, which is reasonable since we want to extract features from the inputs. Alternatively, the FC layers are different because the function to be found from the extracted features is different. As seen in Figure 3.11, the architecture is:

- The first convolutional block contains 2 identical layers: 8 3x3x3 filters, a batch normalization layer and a ReLU function. It is followed by a max pooling layer, which downsamples the input by a factor of 2.
- The second convolutional block contains 2 identical layers: 16 3x3x3 filters, a batch normalization layer and a ReLU function. It is followed by a max pooling layer, which downsamples the input by a factor of 2.
- The third convolutional block contains 2 identical layers: 32 3x3x3 filters, a batch normalization layer and a ReLU function. It is followed by a max pooling layer, which downsamples the input by a factor of 2.
- The output is flattened towards two FC layers the architecture: the first one contains 1000 neurons and the second one, 100. These also contain dropout layers to prevent the CNN from overfitting and whose rates are 0.3 and 0.1 respectively.
- The last FC layer contains 9 neurons and a sigmoid function to perform regression.

#### 3.5.2. Pre-processing

Since the inputs for this CNN are the same as in DNN-MEP, the compressed images will be also normalized. Additionally, the labels need to be processed in order to extract the synergy basis from the MEPs. To do so, the NMF function from MATLAB [37] is used to extract 9 synergies from the MEP matrix, $M$. This results in two matrices: $B$ represents the basis, which is constant for all the stimulations, and $A$ symbolises the activations, understood as
the linear combination of the basis in $B$ to generate an approximate of the stimulations in $M$. This time, the matrix $A$ will be used as the outputs of the DNN used for prediction, instead of the MEPs matrix. Figures 3.8 and 3.9 show the decomposition of 30 different stimulations.

![Ground truth](image1)

**Figure 3.8.** Ground truth MEPs (top) and estimation from NMF results (bottom).

![Estimation](image2)

**Figure 3.9.** Basis (left) and Activation (right) matrices from NMF over the MEP matrix.

### 3.5.3. Training

The dataset was split in a train test (80%) and a test set (20%), whose input-output pairs were the compressed E-fields resulting from the AE and the $A$ matrix obtained from the
MEPs NMF, respectively. Similarly, a 5-fold CV was implemented to obtain a better performance estimation. The loss function and optimizer are constant with respect to DNN-MEP and the random-search scheme for hyperparameter tuning is displayed in Table 3.2.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Range</th>
<th>Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning rate</td>
<td>$1 \cdot 10^6 - 1 \cdot 10^2$</td>
<td>$1.5 \cdot 10^4$</td>
</tr>
<tr>
<td>Epochs</td>
<td>100 - 1000</td>
<td>450</td>
</tr>
<tr>
<td>Dropout 1</td>
<td>0.2 - 0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>Dropout 2</td>
<td>0.0 - 0.3</td>
<td>0.1</td>
</tr>
</tbody>
</table>

*Table 3.2. Hyperparameter tuning scheme for DNN-SYN.*

It is important to notice that, even though the DNN performs regression over the synergies, for a better understanding and comparison of the results, the NRMSE was computed over the reconstructed MEPs. To do so, the predicted synergies were multiplied by the $B$ matrix, giving a prediction over the MEPs, as Figure 3.8 shows. This enhances the ability to compare DNN-MEP and DNN-SYN, as now both of them show a reconstruction over the MEPs.
Figure 3.10. Architecture of DNN-MEP.

Figure 3.11. Architecture of DNN-SYN.
4. Results

This section summarizes the different experiments that were covered and the results obtained for the architectures proposed in the previous chapter.

4.1. Experimental setup

The following experiments were developed:

- **Experiment 1**: train the autoencoder to generate compressed representations of the input data.

- **Experiment 2**: train the regression DNN (DNN-MEP) that maps the latent representations to the MEP obtained from the EMG responses.

- **Experiment 3**: train the regression DNN (DNN-SYN) that maps the latent representations to the SYN extracted of the EMG responses through NMF.

4.2. Metrics

The loss function used to train the three neural networks has been MSE, which is appropriate for regression tasks. However, the overall objective is to compare the results obtained in Experiment 1 and 2, which is not possible through MSE because they have different sizes. Additionally, MSE is not an enclosed measure, which does not provide an intuitive understanding of the reconstruction and depends completely on the nature of the dataset.

To overcome this situation, the metric used to evaluate the performance of the different CNNs is the Normalized Root Mean Squared Error (NRMSE). Normalizing by the mean of the observed data makes the metric enclosed and with a better interpretability. Along with that, taking the root of the result makes the metric have the same dimensions as the data.

\[
NRMSE = \sqrt{\frac{E[\|\hat{y} - y\|^2]}{E[\|y\|^2]}}
\] (4.1)

The metric is used in a CV scenario, meaning that to fully understand the behavior of a particular architecture, the mean and variance (or standard deviation) of the NRMSE have to be computed. The mean is computed directly over the different folds, though the variance is computed through the jackknife resampling method [38]: for \(k\) different estimations that belong to \(k\) folds \((\bar{\theta}_i, i = 1 \ldots k)\), \(k\) means are computed \((\bar{\theta}_i, i = 1 \ldots k)\), each time leaving one fold out. Then, the overall mean \(\bar{\theta}\) is computed as an average of the previous computed means. The variance jackknife estimation of that estimator is given by the formula
\[
\hat{\theta}_{jack} = \text{var}(\bar{\theta}) = \frac{k - 1}{k} \sum_{i=1}^{k} (\bar{\theta}_i - \bar{\theta})^2
\] (4.2)

4.3. Results

4.3.1. Autoencoder

The AE obtained its best performance with an NRMSE of 0.097±0.004. Figure 4.1 shows an original sample, its latent representation and its reconstruction, respectively.

![Figure 4.1](image)

*Figure 4.1* Original sample (left), latent representation (center) and its reconstruction (right).

The reconstruction looks almost indistinguishable from the original sample, meaning that the AE has achieved its objective. In order to gain a deeper understanding of the reconstruction, the reconstruction error can be obtained by subtracting the estimation from the original E-field and, then, plot its histogram. The result is observed in Figure 4.2, which reveals a gaussian-shaped probability distribution of the error because the MSE penalizes more larger errors due to the squared term, i.e. a distance of 10 means 4 times larger MSE than a distance of 5., which results in a small range of error values. This evinces the MSE as an appropriate choice as a loss function.
Additionally, the encoded representation shows how the network decided to maintain spatial information, since the shape of the original data can be intuited. This makes sense considering that the decoder will also compute convolutions and will be able to extract spatial information. In fact, the latent representation resembles a basic spatial downsampling, so Figure 4.3 illustrates this comparison: applying a downsampling factor of 4 in two of the three dimensions results in a similar result as the encoded representation. However, due to the sparsity of the data, i.e. most of the 64-cubed matrix is zeros, when applying a downsampling in the three dimensions the results is almost a blank matrix. Along with that, a direct upsampling would not be able to recover all the information of the original data. In conclusion, the encoder function of the AE outperforms the downsampler for compressing data while retaining more information.

Figure 4.2. Histogram of the reconstruction error of the autoencoder.

Figure 4.3. Original sample (left), latent representation (center) and spatial downsampling (right).
4.3.2. DNN-MEP

The DNN mapping the stimulations directly to the muscles obtained an NRMSE of 0.75±0.03. Figure 4.4 illustrates the comparison between the ground truth MEP activations and the predicted from the network.

DNN-MEP differentiates precisely when a sample is activated and when it is not, since non-active muscles are predicted perfectly (sample 2). However, it finds difficulties in identifying samples with little activation (sample 16), which are predicted as non-active. Additionally, it does not predict accurately more complex samples where most of the muscles are co-activated (samples 6 and 7). The main issue of the DNN is finding solutions with more variance, i.e., having the ability to activate and deactivate muscles simultaneously.

![Figure 4.4](image-url)  
*Figure 4.4. Ground truth MEP activations (top) and predictions (bottom) for DNN-MEP.*
4.3.3. DNN-SYN

The network mapping the stimulations to the synergies resulted in an NRMSE of 0.68±0.03. Figure 4.5 illustrates the comparison between the ground truth MEP activations and the predicted from the network.

DNN-SYN has also the ability to differentiate complete non-activations (sample 2). Moreover, contrary to DNN-MEP, this one is able to identify little activation samples (sample 12). Even though this network inherits the same obstacles as differentiating active and non-active muscles in the same sample, it performs a better prediction by locating more precisely the muscles with maximum and minimum activation (sample 6).

![Figure 4.5. Ground truth MEP activations (top) and predictions (bottom) for DNN-SYN.](image-url)
5. **Budget**

The project has been developed using the resources provided by the SPIRAL research group in Northeastern University. These costs, therefore, will not be reflected in the budget assessment.

Nevertheless, the wages of the researchers and engineers involved in the development represent most of the balance: I considered myself as a full-time junior engineer, and the two supervisors as senior engineers. The work took 20 weeks, as it can be seen through the Gantt Diagram.

<table>
<thead>
<tr>
<th>Position</th>
<th>Quantity</th>
<th>Wage (€/h)</th>
<th>Dedication (h/week)</th>
<th>Total (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-time junior engineer</td>
<td>1</td>
<td>10.00</td>
<td>35</td>
<td>7000</td>
</tr>
<tr>
<td>Senior engineer</td>
<td>2</td>
<td>40.00</td>
<td>1</td>
<td>1600</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>8600</strong></td>
</tr>
</tbody>
</table>

*Table 5.1. Budget of the project.*
6. Conclusions and future development

This project proposed a deep learning-based solution to contrast a hypothesis in the field of physiology, which resides in testing the existence of synergies as basic building blocks of the Central Nervous System to manage its degrees of freedom and generate movement.

The suggested approach takes advantage of the Convolutional Neural Networks architectures in order to extract features from the inputs and map them to labels. Firstly, as a pre-processing stage, an autoencoder was implemented to compress the information of the input E-fields. Then, these processed inputs were mapped to muscles directly or with an intermediate step of synergies, resulting in two different architectures and whose results were compared to test the initial hypothesis.

The results from the first neural network, DNN-MEP, demonstrated that connecting the muscles do not directly explain the stimulations occurred in the motor cortex. Even though the predictions demonstrated an intuition on how the movement was induced in the muscles, they do not fully characterize it. On the other hand, DNN-RAND evinced that synergies contribute in a better understanding on how the muscles were activated. Predicting Motor Evoked Potentials with an intermediate connection to synergies contribute in a more robust characterization of movement, led by a 0.07 improvement in the NRMSE metric.

The obtained results allow the acceptance of the initial hypothesis and confirm that synergies model appropriately the degrees of freedom that the Central Nervous System addresses. Along with that, this project conforms a rigorous analysis in these building blocks in a predictive environment, not in a descriptive manner as in most of the literature.

This project evinces that synergies are an essential element for understanding how movement is generated, though not sufficient. Similarly, this study exhibits that movement may be more complex than just introducing a new building block. For instance, it is reasonable to assume that the MI motor cortex is able to control synergies and individual muscles simultaneously. Therefore, a proposal for a following development of this project would be to build a joint model that unifies the contribution of both structures, muscles and synergies in order to characterize movement.
Bibliography


## Glossary

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>Autoencoder</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>CNS</td>
<td>Central Nervous System</td>
</tr>
<tr>
<td>CV</td>
<td>Cross-Validation</td>
</tr>
<tr>
<td>DNN</td>
<td>Deep Neural Network</td>
</tr>
<tr>
<td>E-field</td>
<td>Electrical field</td>
</tr>
<tr>
<td>ELU</td>
<td>Exponential Linear Unit</td>
</tr>
<tr>
<td>EMG</td>
<td>Electromyography</td>
</tr>
<tr>
<td>FC</td>
<td>Fully Connected</td>
</tr>
<tr>
<td>FE</td>
<td>Finite Element</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>FA</td>
<td>Factor Analysis</td>
</tr>
<tr>
<td>ICA</td>
<td>Independent Component Analysis</td>
</tr>
<tr>
<td>M1</td>
<td>Primary Motor Cortex</td>
</tr>
<tr>
<td>MEP</td>
<td>Motor evoked Potentials</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
</tr>
<tr>
<td>NMF</td>
<td>Non-Negative Matrix Factorization</td>
</tr>
<tr>
<td>NRMSE</td>
<td>Normalized Root Mean Squared Error</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>ReLU</td>
<td>Rectified Linear Unit</td>
</tr>
<tr>
<td>SGD</td>
<td>Stochastic Gradient Descent</td>
</tr>
<tr>
<td>SYN</td>
<td>Synergy</td>
</tr>
<tr>
<td>TMS</td>
<td>Transcranial Magnetic Stimulation</td>
</tr>
<tr>
<td>WP</td>
<td>Work Package</td>
</tr>
</tbody>
</table>