Person-Independent sEMG Gesture Recognition Using LSTM Networks for Human-Computer Interaction

Michael Lohr, B.Sc.
Chair for Computer Aided Medical Procedures & Augmented Reality
Technical University of Munich
Boltzmannstraße 3, 85748 Garching
michael.lohr@tum.de

***Abstract*—Hand gestures feel natural to perform, which makes them well-suited to use as Human-Computer Interaction interfaces. But detecting them with high accuracy in real-time is a challenging task. This paper presents an approach based on the Long Short-Term Memory Neural Network architecture to evaluate Surface Electromyography signals and determine the gesture performed. This approach is not new and has limited performance on people for whom it wasn't trained. Therefore, this research evaluates an approach where the Neural Network's existing knowledge is adjusted to a new person using just a few samples from the new person and very little training. This strategy allows getting accurate results with an approach that is usable in a Human-Computer Interface.**

***Index Terms*—Human-computer interaction, hand gesture recognition, sEMG, machine learning, LSTM, transfer learning**

I. Introduction

Hand gestures are a powerful tool that humans use to communicate with each other. They are so deeply rooted in our brain that we often use them without even consciously realizing it. Since using them feels so natural to us, they make a great Human-Computer Interaction (HCI) interface. But detecting gestures to interact with an interface is a challenging task because the data has to be captured, digitalized, and interpreted.

There are various approaches to recognize hand gestures. The most common ones make use of computer vision or Electromyography (EMG). While computer vision techniques have to cope with occlusion and other problems of image recognition, the EMG approach measures the signals directly where they occur. EMG allows the detection of electrical activity produced by skeletal muscles. There are two types of EMGs: intramuscular and surface EMGs. When using intramuscular EMG, a needle is inserted into the muscle [1]. Since this is not practicable for an HCI, this research focuses on the less accurate but more convenient surface Electromyography (sEMG) approach.

In this research, the Myo armband[1] is used to obtain the data, as it is a well-researched and ready-to-use solution. It provides real-time data from eight sEMG sensors and orientational data using a built-in Inertial Measurement Unit (IMU) via Bluetooth. The armband comes with its own software package to detect five gestures, which can be used to control certain applications. Furthermore, a Software Development Kit (SDK) is available, which supports multiple programming languages.

Detecting gestures or, more precisely, the translation of muscle activity data over time with multiple samples per second to specific gestures, is not a trivial task. Since interpreting this data by specifying time-based thresholds for specific sensors is very time-consuming, Neural Networks (NN) are used to learn from and adapt to this data. Their ability to optimize to unknown functions represented by recorded data is well-suited for such a problem. But the NN must be capable of handling time-based data and making connections to previous timestamps. The so-called Long Short-Term Memory (LSTM) NN, which is a type of Recurrent Neural Network (RNN) architecture, can learn long-term dependencies in sequence prediction problems.

The NN has to be trained on labeled pre-recorded data in order to learn to recognize specific gestures. Data will vary significantly from person to person [2], [3]. To make the model robust and able to interpret gestures from any person, it has to learn which features make up the essence of the gesture and ignore person-specific features. The network has to be trained with data from multiple persons in order to generalize on the task. "Generalization" describes how well a model is able to classify data from unseen subjects, which was not used for training.

"Specialization", on the other hand, describes how well a model performs on data more similar to the data it was trained with. An NN trained with only gestures of a specific person will achieve poor accuracy when being used on the data of another person. This is due to the fact that different persons performing the same gesture will result in different EMG data. Not only the random noise, which is in the nature of EMG data, is responsible for this. Also, factors that vary from session to session, like different armband placement, or from person to person, like differences in the human's anatomy, will result in diverse signals.

Models that are able to cope with factors that are different from session to session, like armband placement or other environmental factors, are called "session-independent". Those models are trained with the data from multiple sessions of the same subjects. They will not be able to achieve high accuracy

[1]Developed by North Inc. (formerly Thalmic Inc.) Website: www.bynorth.com

on data from new people but are expected to perform well on data of known subjects from unseen sessions.

Personalized NNs have to be solely trained on data from the person whose EMG measurements will later be used to make predictions. Such networks will not be able to generalize to unseen subjects but will be able to achieve higher accuracy for the person it specialized in. It is also expected to need less training since the data is more uniform. But person-independent NNs were trained with data from different persons. Those networks have developed a generalized model and are better at making predictions from data of new persons whose data was not used during training.

The problem with these approaches is that they require much data and a lot of training time. And even then, the generalized model might still have issues recognizing gestures from unseen subjects. To work around this problem Transfer Learning (TL) can be applied. Before using the trained model on a new user, the subject is asked to perform the gestures a few times. This labeled data is then used to adapt the generalized model of the network to that user. This adaption process will only train a small portion of the network, which is why this approach takes little time and data. After learning how to cope with the differences of this user's data, the model is able to achieve higher accuracy on those specific subjects. Since this approach achieves good prediction results with little additional training data and time, it is well suited for HCI.

This paper proposes and evaluates an LSTM architecture for an sEMG recognition task and further impoves its accuray using the TL approach.

II. Related Work

There is a wide range of different approaches to tackle this task: Linear regression, Neural Networks [4]–[6], support vector machines [2], locally weighted projection regression [7], and hidden markov models [3], and more.

A lot of the research on sEMG-based gesture recognition is based on individualized models, trained on recorded data from only one user [3]. Generalized classifiers are harder to train because of different sensor placement, contact conditions, the anatomy of the human's arm, and inter-individual differences in performing the gesture [2], [3]. Castellini, Fiorilla, and Sandini [2] show that EMG signals vary significantly between different persons, even when controlling precisely for probe placement. Their trained model is able to classify gestures on unseen subjects with little more accuracy than random guesses [2].

Since the Myo armband also has an IMU built-in, it makes sense to explore the possibilities with angular rates and orientational data. Georgi, Amma, and Schultz [8] show that using the IMU in addition to sEMG sensors greatly improves the classification accuracy, especially on gestures involving a lot of arm motion. It also performs well on new subjects, supposably because the variance in motion data between persons is not as high as in EMG data [8].

Chen, Li, Hu, *et al.* [9] apply TL to reduce the training time and improve the accuracy of a Convolutional Neural Network (CNN) architecture. With only two repetitions of each gesture to fine-tune the model, they were able to reduce the training time significantly and improve the recognition to finally achieve an accuracy higher than 90% [9]. They also show that adjusting a generalized network without TL would take at least six learning repetitions of each gesture to get satisfactory accuracies [9].

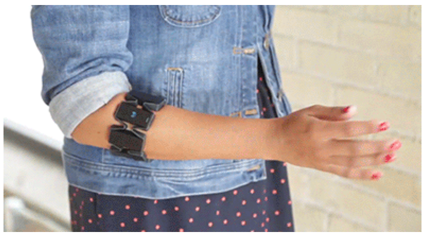


Figure 1: A person wearing the Myo armband correctly. It should be worn on the thickest part of the forearm [10].

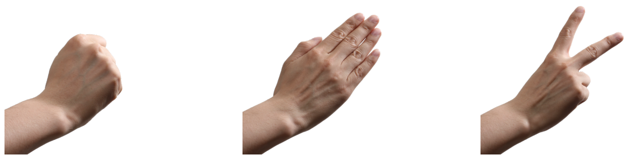
Cedric Fromm [4] compares RNNs and CNNs on a rock paper scissors gesture recognition task. He found that RNNs can achieve higher accuracies for this task [4]. Specifically, his experiments showed that a multilayered LSTM architecture works very well [4]. With that architecture, which is also used as a base for the NN proposed in this work, he was able to achieve up to 94% accuracy [4].

III. Dataset Recording

The discontinued Myo armband is used to collect data. It is a non-invasive armband, which provides live sEMG data from eight sensors with a sample rate of $200\,\text{Hz}$ and orientational data with a sample rate of $50\,\text{Hz}$ using a built-in IMU. The data is transmitted to a Bluetooth dongle and can be accessed using the Myo SDK. The armband wraps around the user's forearm, with each sensor having roughly the same distance to each other, as seen in Figure 1. Therefore the sensors are not assigned to a specific muscle or muscle group. The Myo armband is often used as an interface in HCI applications, as it is small, portable as well as comfortable and requires little initial set-up effort.

Past research on the Myo armband often uses the five gestures (spread fingers, wave right, wave left, make a fist, double-tap) that the Myo classifies using its own recognition software. In this research, the gestures from the Rock paper scissors hand game are used, which are shown in Figure 2. The reasons for this are twofold: The network architecture proposed by Cedric Fromm [4] is used as a base and compared to the results of this paper. To make this comparison more accurate, a similar dataset should be used. Secondly, the three gestures, Rock, Paper, Scissors, are more distinct than the five Myo gestures but require less recording effort, which allows to aggregate more data in the same amount of time.

In order to generate labeled training data, a Python application interfacing with the Myo SDK was implemented. The data is recorded while showing an icon and text label describing the gesture. Each gesture is recorded for two seconds following a 0.5 second break where subjects are supposed to relax their arm. The two-second time-span was chosen because several tests showed that it is the amount of time required for most persons to perform the transition from rest pose to the wanted



(a) Rock **(b)** Paper **(c)** Scissors

Figure 2: The gestures from the Rock paper scissors game, which subjects performed during the dataset recording [11].

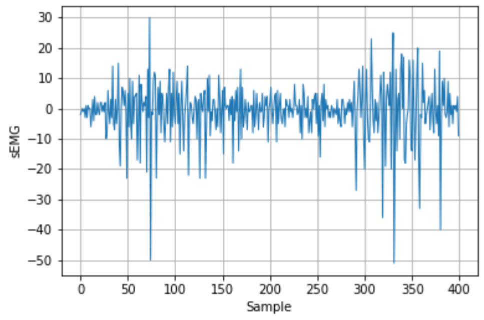
gesture and back. In one recording iteration, each gesture is recorded ten times, which makes a total of 30 gestures per iteration. Therefore, one iteration takes 75 seconds. Those values were chosen after initial testing, which showed that subjects quickly get exhausted performing the gestures. After a short break, without taking the armband off, a second iteration was performed in the same session. This is then repeated once more. Some participants were recorded a second time: After a long break, the armband was put on differently, and the process described above was repeated. In total, each participant contributed three (one session) or six iterations (two sessions). One additional participant recorded 30 datasets in 10 sessions on different days to create a highly specialized dataset.

Thirteen healthy subjects aged between 18 and 58 years participated in the data acquisition process. Four of them were female, and nine candidates male. One participant was left-handed. All subjects agreed to the data recording and analysis by writing. Before starting the recording session, participants filled out the consent form, which states what is being recorded and how the data is used. The Myo armband was then put onto the thickest part of the forearm, just below the elbow[2] on the dominant arm. Then the subject waited until the connection icon on the armband stopped blinking, which indicated that the armband warmed up. After performing the Myo sync gesture, which is needed to detect the arm wearing the armband, the Myo is ready to use. Subjects were told to start from a rest pose, performing the gesture and reaching the highest intensity when the progress bar indicates that half the time passed.

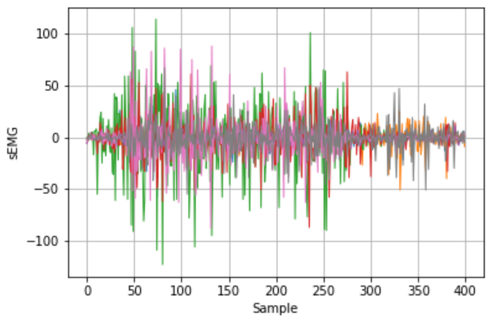
There are multiple reasons for always using the dominant arm during the recording. First, a possible user interface using one Myo armband would be controlled with the dominant arm because it feels more natural to the user. But also not using the dominant arm could introduce irregularities into the dataset because users are not used to performing similar gestures with their non-dominant arm. Also, differences in the data resulting from recording the left and right arm are non-existent since the current arm is detected, and the data mirrored accordingly.

All data provided by the Myo armband is stored in multiple files labeled with a unique but randomized identifier. Each EMG file contains roughly 400 ± 5 entries since the recording took two seconds, and the data was sampled with 200 Hz. Each entry contains an index, a timestamp, and one value for each of the eight sEMG sensors. IMU and other data along with meta-information (label, recording time, left or right arm, battery level, and connection quality) is stored in other files.

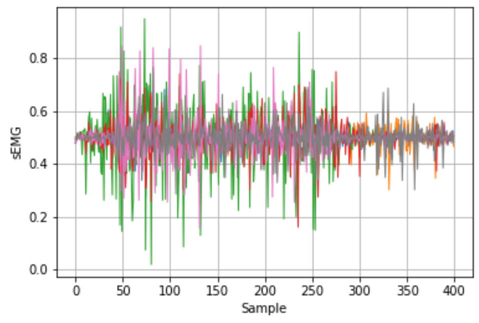
[2]As instructed on the Myo help article "How to wear the Myo armband." Website: support.getmyo.com



(a) The readings of a single sensor of the raw sEMG data.



(b) All sensor readings from the raw data.



(c) Normalized and padded sensor readings.

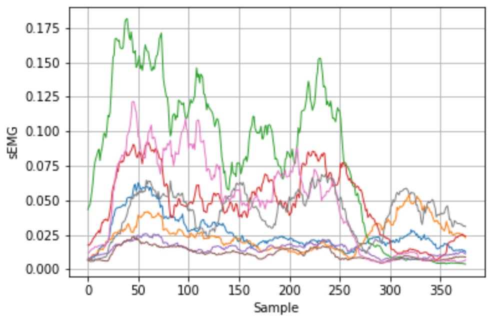
Figure 3: The recorded data of a paper gesture with the X-axis showing the samples from left (second 0) to right (second ~ 2) and the Y-axis showing the sEMG value.

IV. Data Analysis

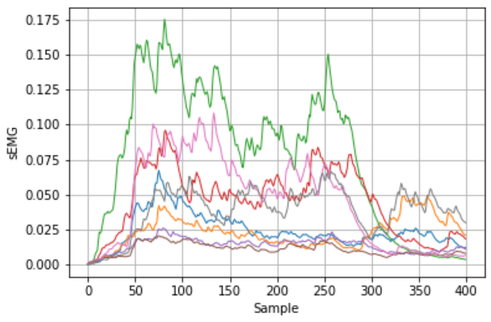
To validate and understand the data recorded as described in the previous section, a manual data analysis was performed first.

To begin with, the data was preprocessed: The raw sEMG data as seen in Figure 3a and Figure 3b was normalized to the range $[-1, 1]$. Since not every recorded gesture's data capture contains exactly 400 samples (2 seconds$*$200 Hz), the data was padded with zeroes at the start of the recording. If a gesture contained more than 400 samples, the data was cut off at the end. The resulting preprocessed data is visualized in Figure 3c.

When looking for similarities in other plots of the same gesture from the same session, patterns were not apparent in the preprocessed data. To visualize the features more prominently,



(a) The rectified data processed with a moving average of $n = 25$.



(b) A first-order digital Butterworth filter applied on the rectified data.

Figure 4: The processed recorded data of a paper gesture with the X-axis showing the samples and the Y-axis showing the sEMG value. Each of the eight sensors is visualized by a different color.

different strategies were used: Since the sEMG data contains a lot of noise, it was rectified (the absolute values were taken) and processed with a moving average over the $n = 25$ past samples. This shows the overall trend of swings, as seen in Figure 4a.

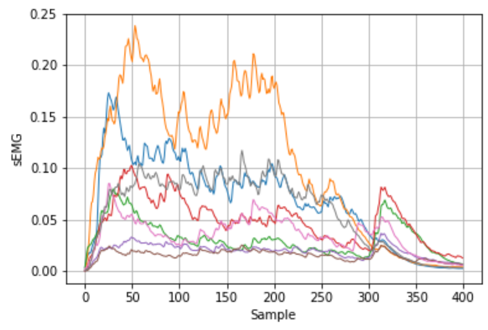
Afterward, another approach with more control was used: A first-order digital Butterworth low-pass filter was applied to the data. The Figure 4b displays the data with less noise.

Looking at the normalized and padded data of different recordings of the same gesture from the same session, no gesture specific pattern can be noticed. But as seen in Figure 5, looking at the filtered results, a pattern emerges: The rock gesture (the sensors visualized with orange and blue color) shows large swings. For the paper gesture, the sensor colored green and red seem to play an important role.

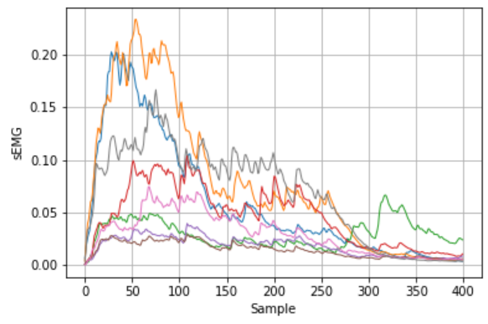
But when comparing the recordings of different persons, those patterns are not that apparent anymore. This is because of the problems mentioned in Section III. An example of such a case is visualized in Figure 6.

V. Neural Networks

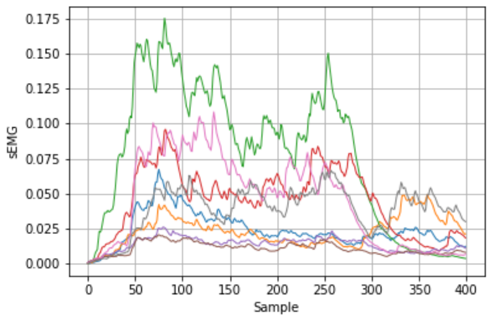
Neural Networks are function approximation algorithms. Using labeled training data, they can “learn” specific patterns found in the data and approximate results for unseen data. A



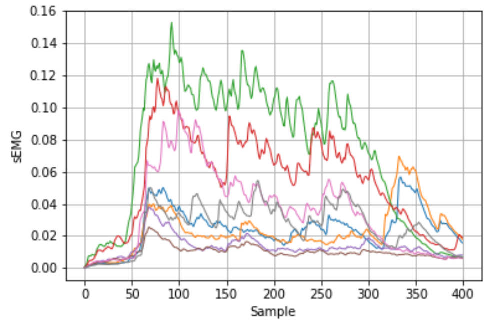
(a) A rock gesture.



(b) A different rock gesture.

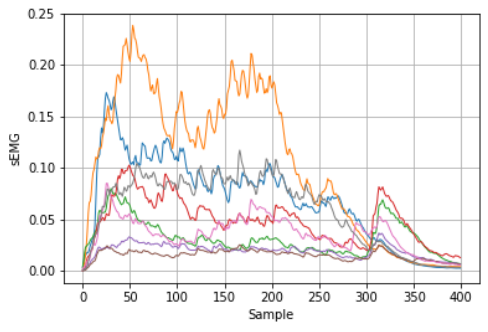


(c) A paper gesture.

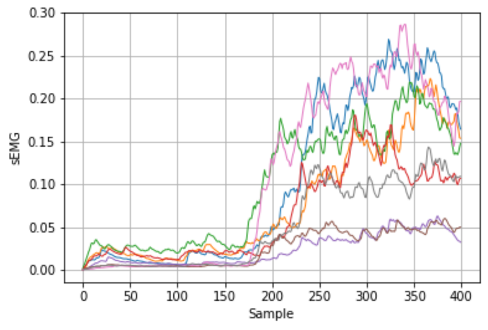


(d) A different paper gesture.

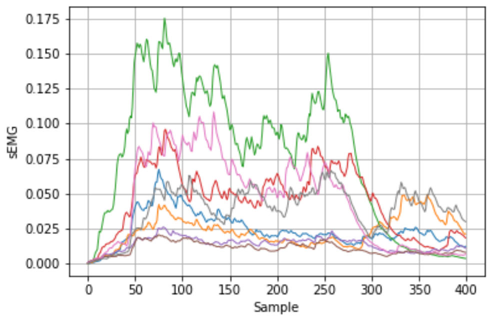
Figure 5: Data plots from the recordings of the same session filtered with the Butterworth filter as described in Section IV.



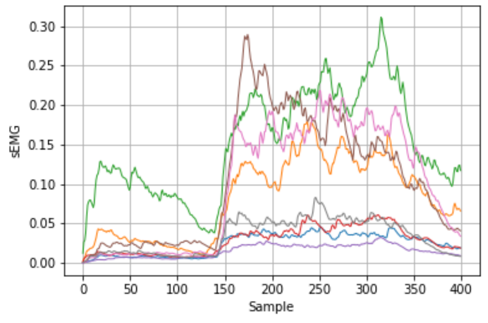
(a) A rock gesture.



(b) A rock gesture from a different person.



(c) A paper gesture.



(d) A paper gesture from a different person.

Figure 6: Data plots from the recordings of different persons filtered with the Butterworth filter as described in Section IV.

NN consists of multiple neurons (or "units"), which are small processing units that are connected to each other and can have numerous inputs as well as one output. During the learning phase, the weights of those connections are adjusted. A so-called "layer" contains multiple neurons that perform different transformations to the input of that layer. A NN typically consists of multiple layers, which are stacked on top of each other.

Conventional feed-forward networks cannot handle sequenced data very well. They are not able to "remember" the data they have seen previously, which is essential in tasks like time series forecasting. Recurrent Neural Networks are a category of NN architectures that try to solve this problem by having an internal temporal memory. The same transformation is performed on every input and depends on the past computations [4].

The Long Short-Term Memory architecture is a type of RNN architecture developed by Hochreiter and Schmidhuber [12]. They introduce a cell that consists of multiple connected neurons [12]. Three gates control the internal memory state: The input gate, which specifies what input is stored in the cell state. The forget gate, which controls which cell state is cleared from the cell [12]. And finally, the output gate, which determines the output of the LSTM cell [12].

The NNs presented in this research were implemented using Keras[3], which can run on top of TensorFlow 2.0[4], in Python.

VI. Base Model Achitecture

The following NN architecture is used to build a model which hereinafter will be referred to as the "base model" since it will be used as the base for the TL process. The architecture used in this research is based on the previous work of Cedric Fromm. He compared different NN architectures and performed a manual hyperparameter optimization. The architecture visualized in Figure 7 was found the most effective on the gesture recognition task [4].

Some adjustments had to be taken in order to use his network on the data recorded for this research. First, the input data sequence does not have a shape of 40×8, but rather 200×8 because each gesture consists of 200 instead of 40 samples. Each sample contains eight sensor readings, as explained in Section III. A masking layer is introduced to mask values that were added in the preprocessing stage by padding the data. Timesteps containing masked values will be ignored by all downstream layers. Right after the LSTM cells, a dropout layer is added, which, with a given rate (the "dropout rate"), randomly drops unit input values during training. This prevents all neurons from synchronously performing their weight updates, which increases the robustness of the resulting model. Also, some fully connected layers (also called "dense layers") are added to process the values from the LSTM. Finally, the output layer or "activation function" only has three final neurons instead of four. Cedric Fromm [4] introduced an additional

[3]Keras is a deep learning API that runs on top of TensorFlow. Website: www.keras.io

[4]TensorFlow is a software library for machine learning tasks. Website: www.tensorflow.org

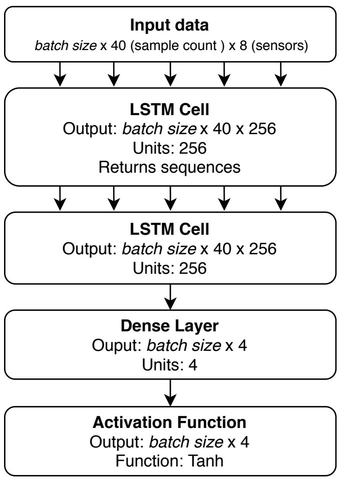


Figure 7: The proposed LSTM architecture from Cedric Fromm [4]. The network consists of one input layer, two LSTM cells and finally the output layer.

Hyperparameter	Type	Values	Chosen Value
Batch size	Integer	$[10-100]$	55
LSTM units	Integer	$[1-600]$	250
LSTM recurrent dropout	Float	$[0-0.5]$	0.55
LSTM dropout	Float	$[0-0.5]$	0.45
Additional LSTM layers	Integer	$[0-2]$	1
Hidden units	Integer	$[50-300]$	80
Hidden activation function	Categorial	Sigmoid, Relu, Tanh	Sigmoid
Dropout	Float	$[0-0.5]$	0.60
Optimizer	Categorial	Adam, NAdam, RMSprop, SGD	NAdam

Table I: The hyperparameters that were tuned together with their ranges and chosen values.

class to the rock, paper, and scissors classes, which should trigger when no gesture is detected. It was found that this extra neuron is not required since the confidence values can be used to detect if no gesture is performed. The final architecture, without concrete values for the neuron count, can be seen in Figure 8.

VII. Hyperparameter Optimization

Hyperparameters are parameters that influence the learning process itself or the NN structure. They are not learned during the learning phase but are defined manually by the architect. Cedric Fromm [4] tuned the NN presented in Section VI by hand. His architecture is used as a basis for the hyperparameter tuning in this research. Important parameters from the base architecture that made sense to tune were converted to ranges around the old value. The parameters, along with their ranges and chosen values, are shown in Table I.

The parameters were tuned using a bayesian hyperparameter tuning algorithm in Comet[5]. Comet uses an algorithm called "Sequential Model-based Algorithm Configuration (SMAC)" based on the work of Hutter, Hoos, and Leyton-Brown [13]. This algorithm uses Bayesian Optimization to compare two configurations and decide which parameter performs better [13].

[5]Comet is a cloud-based meta machine learning platform. Website: www.coment.ml

To tune the parameters, new values are set every run, and the network trains the new architecture. The performance of the resulting model is then evaluated, and the process is repeated. Comet was configured to tune the parameters of Table I in a way that maximizes the performance of the network on an additional dataset of gestures (the "validation" set). After around 300 runs with different parameters, the results were evaluated: 12% of the runs achieved a validation set accuracy higher than 80%. The final parameters where chosen from runs that achieved more than 95% accuracy and with stability in the learning process.

The resulting NN architecture of the base model is shown in Figure 8. As described in Section V, the input data is first passed into a masking layer. Afterward, the masked values are fed into the first LSTM cell, which is set to return all internal cell outputs as a sequence so that these values can be fed into another cell. The second cell is set to return only the last hidden state output. Both LSTM cells use the "Tanh" activation function, contain 250 neurons, have a dropout probability (fraction of the units to drop for the transformation of the input) of 45% and a recurrent dropout (fraction of the units to drop for the transformation of the recurrent state) of 55%. Before values are passed to the fully connected layers, they are dropped with a probability of 60% in the dropout layer. The first dense layer contains 80 neurons and uses the "Sigmoid" activation function. Finally the output layer consists of three neurons, as mentioned in Section V, and passes its values to the "Softmax" activation function. The network is trained with the "Nadam" (short for "Nesterov-accelerated Adaptive Moment Estimation") optimizer, which incorporates Nesterov Momentum as a better momentum component into the Adam optimizer [14].

VIII. Training

Three different datasets were used to train the three base models with the previously mentioned NN architecture:

- **10sub**: A generalized model, trained with three recording iterations from ten subjects
- **5sub**: A generalized model, trained with six recording iterations from five subjects
- **1sub**: A personalized model, trained with 30 recording iterations from one subject

Each dataset contains exactly 30 iterations or 300 recorded gestures per gesture type (see Section III). These sets are split into the training set (66% of the dataset) and the test set (33%) before training.

The base models were compared against each other in Table II, in terms of the highest achieved Test Set Accuracy (TSA), Session-Independent Accuracy (SIA), and Person-Independent Accuracy (PIA). The Test Set Accuracy (TSA) shows how the NN performs on data similar to the one it was trained with. The Session-Independent Accuracy (SIA) indicates how well the model generalizes to also detect gestures in new sessions from subjects whose data it used to train with. The Person-Independent Accuracy (PIA) shows how well the

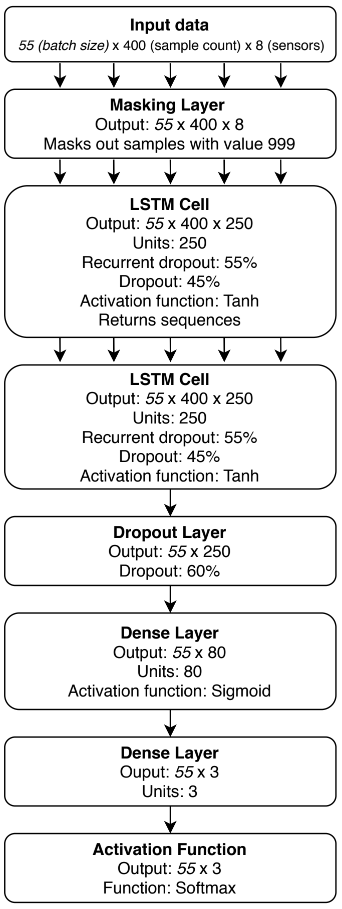


Figure 8: The proposed base model architecture, as it is implemented in Keras. Below the type of the layer, the output shape and other important settings are specified. Multiple arrows indicate that sequenced data (three dimensions) is passed to the next layer.

Dataset	TSA	SIA	PIA
10sub	0.852	0.678	**0.656**
5sub	0.852	**0.700**	0.645
1sub	**0.974**	0.234	0.333

Table II: The models that were trained with different datasets and their highest accuracies: Test Set Accuracy (TSA), Session-Independent Accuracy (SIA), and Person-Independent Accuracy (PIA).

network is able to recognize gestures from subjects it has never seen before.

As seen in Table II, the 1sub model is able to achieve the highest TSA with 97%. This can be explained by the fact that the data is more uniform since it does not contain differences in anatomy and ways to perform a gesture. But this NN is not able to generalize very well. On the other hand, model 10sub, and 5sub, which were trained on multiple persons, are able to generalize to different sessions and even unseen subjects with an accuracy in the range of 64% to 70%.

Since the intial NN structure, as well as the type of gestures, is based on the work by Cedric Fromm [4], it makes sense to compare the performance: The highest TSA he was able to achieve using an LSTM architecture on the five best datasets (containing the data of five subjects) is 93.71%. This accuracy is more than eight percent better than the highest TSA of the 5sub set, which is probably due to the fact that Cedric Fromm [4] recorded more datasets and evaluated them, to only use the datasets that performed best.

Cedric Fromm [4] also evaluated his network in terms of generalization and specialization on different subjects: The best PIA the network, trained on five subjects, achieved on three persons outside the training set was 67.91% [4]. Also, in terms of specialization, the results are similar to the ones in this work: The TSA on a model trained on data only from one person (like the 1sub dataset) is 97.64% [4].

IX. Transfer Learning

Transfer Learning describes a particular process of adapting a trained NN to a different but related problem by leveraging the existing knowledge. While a traditional NN is also able to adapt to some extents, it requires a lot of training data and time. TL is fast, since only some parts of the network need to be trained, and requires little extra data as the existing NN already knows how to handle a similar problem.

Typically, transfer learning is defined using:

- The training dataset that consists of samples $\{x_i, y_i\}$ where $x_i \in \mathcal{X}$ with $\mathcal{X}$ being the feature space and $y_i \in \mathcal{Y}$ with $\mathcal{Y}$ being the label space
- The learning task $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$, with an objective predictive function $f(\cdot)$ which is learned and can be used to predict the corresponding label $f(x)$, of a new instance x.
- The domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ of the task, with a marginal probability distribution $P(X)$

And the following definition from Pan and Yang [15]: *Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$.*

To apply the TL approach to the existing network from Section VII (the base model), an additional NN architecture is put on top of the base model architecture: Fully connected layers enable to learn the difference between the learned knowledge and the new input data. To reshape the data into a matrix that the base model expects (dimensions of 200×8), a final dense layer with eight neurons (one per sensor) is required. These new layers ("transfer layers") are then trained with recorded gesture data from subjects that were not used to train the base model. The layers of the base model will not receive weight updates anymore. Since the added layers have not many neurons, short training time is expected.

The number of layers and neuron count, as well as batch size and optimizer, were determined using a hyperparameter optimization approach, in a similar fashion to the one presented in Section VII. The parameters that were tuned together with the chosen value are displayed in Table III. Since the dataset for the TL process is small, the tuning algorithm was able to find multiple architectures resulting in a high test set accuracy. To decide on the parameters, also the training time and loss

Hyperparameter	Type	Values	Chosen Value
Batch size	Integer	[5 – 100]	55
Dense layer count	Integer	[1 – 3]	3
Dense units	Integer	[10 – 500]	350
Optimizer	Categorial	Adam, NAdam, RM-Sprop, SGD	NAdam

Table III: The hyperparameters of the transfer layers together with the chosen value.

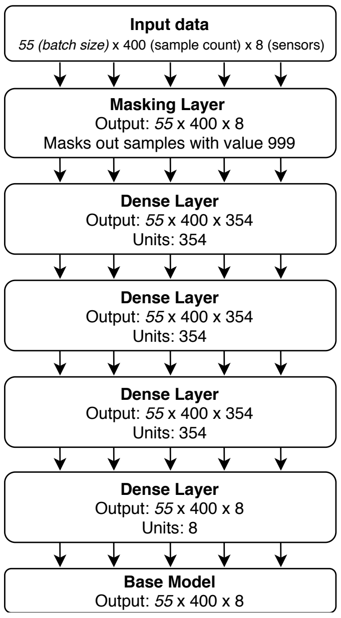


Figure 9: The proposed TL architecture, as it is implemented in Keras. Below the type of the layer, the output shape and other important settings are specified. Multiple arrows indicate that sequenced data (three dimensions) is passed to the next layer. The last layer is the base model, presented in Figure 8.

over time from different runs of the hyperparameter tuning were considered. The resulting architecture can be seen in Figure 9.

To evaluate the performance of this TL approach, the following experiment was conducted: Each base model, as presented in Section VIII, was modified as described above to enable transfer learning. Then, the transfer layers were trained with the data of one recording session (90 gestures) of one person whose data was not used to train the base model. The network was trained with a test set split of 33% (meaning 33% of the data was used as a test set) and 200 epochs. This was repeated two more times with different persons. After the transfer layers were trained, the accuracy on the test set (TSA), the accuracy of the data of another session from the person the transfer layers were trained with (SIA), and the accuracy of the data from an unseen person, whose data was not used to train the base model nor the transfer layers with (PIA) were determined. This process was repeated with different persons.

In every run, the network was able to achieve 100% accuracy on the training data as well as the test set (TSA). The highest accuracies of this experiment are shown in Table IV. These values indicate how good the TL approach can be. With some

Dataset	TSA	SIA	PIA
10sub	1.000	**0.983**	0.500
5sub	1.000	0.950	0.656
1sub	1.000	0.967	0.400

Table IV: The transfer learned models that were trained with different datasets and their highest accuracies: Test Set Accuracy (TSA)), Session-Independent Accuracy (SIA)), and Person-Independent Accuracy (PIA)).

subjects, the SIA and PIA accuracies were up to three percent lower, but still higher then what the base model would score. To get a good overview of what accuracies are reached on average, this experiment has to be repeated with more subjects and gesture data. The transfer learning approach with base model 10sub resulted in the highest accuracy (98%). This was expected since that model already learned to generalize and to cope with differences. But even the 1sub model, whose base model was only trained with the data of one subject, performed well (97% accuracy). The PIA is not very high, which was expected, too, since the transfer learned model is now specialized to that one user and not expected to work on other subjects as well.

X. Usage as HCI Interface

The results of Section IX showed that it is possible to achieve high accuracy using the TL approach presented in this research. But to make it a feasible process for HCI, it is essential that few training data and short training time is sufficient. Also, predicting the data as it comes in from the Myo armband ("inference") has to be done in real-time.

In the presented experiments, the data of three recording iterations of one session are used to train the transfer layers. It would not be possible to train the base model with that small amount of data. However, it is enough for the small transfer layers. With the dataset recorder from Section III, recording those three iterations would take around four minutes (225 seconds), given no break is made in between iterations.

The TL network always reached a test set accuracy of 100% after 30 to 90 epochs of training. Training an epoch takes around five seconds on an NVIDIA GeForce GTX 1080 Graphics Processing Unit (GPU), which is consumer-grade hardware. Therefore a training time of three to eight minutes is expected.

When feeding live sEMG data from the Myo armband into the NN, an approach similar to the sliding window algorithm can be used. Every second, the last 400 timestamps are fed into the network to make predictions on the fly. This only works because the inference time is about 0.2 seconds on the aforementioned GPU. Gestures are then detected when the prediction confidence (the result of the activation function for the corresponding neuron) for one gesture is higher than 90%. The name of the gesture with the highest confidence is then displayed to the user. In real HCIs, the corresponding action would be executed instead.

Assuming that high accuracies, as presented in Section IX, can be achieved consistently, this approach can be a viable process for improving gesture detection results. But due to the recording and training time of about 10 to 15 minutes, it will not be suitable for applications that are only used for a short time. The time needed to train and record the data will probably be higher when detecting more complex gestures, too.

More research has to be performed on this topic, to figure out how to decrease those times further. Depending on the complexity of the gestures, there is certainly a limit on how few recorded gestures are required to still perform well. Also, the training time can probably be reduced even further by fine-tuning the network.

HCIs that are made for long-term usages, like new input devices for personal computers, are excellent use cases for the TL approach. A pre-trained generalized model should come with the device. To increase detection accuracy, the user can then perform the TL process and profit from the high SIA. If even better accuracy is required, the user could perform the TL training before each session because of the high TSA. To enable a multi-user scenario, different adapted models have to be stored as user profiles.

A small application was implemented to test the capability of the model to detect gestures in real-time. The EMG data is put into a queue, and every two seconds, a prediction is made using the last 400 timestamps. A gesture is displayed when the value of the corresponding output neuron (the confidence value) of one gesture is higher than 80%. The gesture with the highest confidence is then displayed. This proof-of-concept achieved good accuracy with the base model already. After applying TL, the accuracy for the rock and scissors gesture improved, but every other gesture that was not trained (rest pose and random other poses), were detected as the paper gesture. Training a rest gesture like Cedric Fromm [4] might fix this problem. To test whether this approach can cope with different armband placement, the armband was rotated, and then TL applied again. The model was then able to detect rock and scissors gestures correctly.

XI. Future Work

The TL experiment, as presented in Section IX was only tested on three subjects due to limited training data. To better compare this method to others, more data is required to run the experiment multiple times, which will allow calculating averaged accuracies. The base model's accuracy can be further improved with more training data. By testing and ranking datasets beforehand like Cedric Fromm [4] suggests, high accuracies as 93.71% for a generalized model can be achieved. To improve the results even further, hyperparameter tuning can be performed on each base model with a different dataset individually.

For reasons described in Section III, only three different gestures were used. But this approach should scale well with a lot more gestures, which would require more training data. A similar approach to this research showed that the TL method works well with as much as 30 gestures when using a CNN network architecture and more sensors [9]. Additionally, other research showed that in terms of accuracy, an LSTM architecture would achieve even better results in gesture dectection [4].

While working with the data recorded as described in Section III, it became apparent that users are performing the gesture very differently in terms of timing. Signaling the user when to start the gesture, when to perform it with it's highest intensity, and when to let go, to get a more consistent dataset could improve gesture recognition accuracy.

To really show that this approach can be used in practice, an HCI application could be implemented. A useability study would then show how users think of the interface and its performance. To improve the user experience, it should be explored whether the training time and required data for the TL process can be further reduced.

The limiting factor might be the Myo armband at some point. To improve accuracy and be able to detect more complex gestures, previous research had success incorporating two Myo armbands [16]. Other methods of applying sEMG could also be explored.

Since the Myo has an IMU built-in, this data could be feed into the NN along with the EMG data. Though problems might arise because the IMU is sampled with a different rate than the sEMG data. Georgi, Amma, and Schultz [8] discusses the benefits that arise when combining the sEMG and IMU data.

XII. Conclusion

Surface Electromyography is a convenient way to gather EMG data for HCI interfaces. But because of the nonuniform and noisy nature of sEMG data, it is still a challenging task. To detect hand gestures from it, NNs can be used. Since the gesture data will vary from session to session as well as from person to person, the model has to generalize very well or be able to specialize in a short period of time. Huge datasets and long training time is required to train a conventional NN on this kind of data.

Hyperparameter tuning is used to refine the model by automatically optimizing specific parameters that define the architecture. The final base model architecture uses two LSTM cells and a fully connected layer.

Datasets with different amounts of subjects were used to determine the learned model's generalization and specialization accuracy. NNs trained on data of many subjects were able to generalize better.

TL can be used to translate a pre-trained model to a different session or a new person with very little additional data and training time. It is implemented by adding a few fully connected layers on top of the base architecture. The resulting network is then re-trained with little data of the new subject while locking the weights of the already trained layers of the base network. These layers learn how to translate the new data to the data expected by the generalized model.

To use this approach in a real-time HCI scenario, short training and inference periods are expected. Since the TL setup requires acquiring data and training the transfer layers, which takes about 15 minutes, this approach is meant for long-term usage of such an application. Inference takes about 0.2 seconds, which is little enough to perform the prediction in real-time.

The resulting NN was able to detect gestures with high accuracy. These results show that this approach is a promising one for use in HCI.

Acknowledgements

Foremost, I would like to thank my supervisor, Prof. Gudrun Klinker, for allowing me to conduct this research at her research group Forschungsgruppe Augmented Reality (FAR) at the Chair for Computer Aided Medical Procedures & Augmented Reality.

I would also like to express my sincere gratitude to my advisor, Sandro Weber, for his close supervision and helpful advice.

A special thank goes to all my friends and family members who took the time to participate in the data recording sessions.

References

[1] M. Schwartz, Ed., *Emg methods for evaluating muscle and nerve function*. InTech, 2012, ISBN: 978-953-307793-2.

[2] C. Castellini, A. E. Fiorilla, and G. Sandini, "Multi-subject/daily-life activity emg-based control of mechanical hands," *Journal of NeuroEngineering and Rehabilitation*, vol. 6, no. 1, p. 41, 2009.

[3] A.-A. Samadani and D. Kulic, "Hand gesture recognition based on surface electromyography," *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference*, vol. 2014, pp. 4196–4199, 2014.

[4] Cedric Fromm, "Hand-gesture classification via neural networks: Improving accuracy with continuous use," 2018.

[5] U. Cote-Allard, C. L. Fall, A. Campeau-Lecours, C. Gosselin, F. Laviolette, and B. Gosselin, "Transfer learning for semg hand gestures recognition using convolutional neural networks," in *2017 IEEE International Conference on Systems, Man and Cybernetics (SMC), 05 Oct - 08 Oct 2017, Banff, AB, Canada*, IEEE - Advancing Technology for Humanity, 2017, pp. 1663–1668, ISBN: 978-1-5386-1645-1.

[6] U. Cote-Allard, C. L. Fall, A. Drouin, A. Campeau-Lecours, C. Gosselin, K. Glette, F. Laviolette, and B. Gosselin, "Deep learning for electromyographic hand gesture signal classification using transfer learning," *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 27, no. 4, pp. 760–771, 2019.

[7] S. Vijayakumar, A. D'Souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[8] M. Georgi, C. Amma, and T. Schultz, "Recognizing hand and finger gestures with imu based motion and emg based muscle activity sensing," in *BIOSIGNALS 2015*, H. Loose, Ed., SCITEPRESS, 2015, pp. 99–108, ISBN: 978-989-758-069-7.

[9] X. Chen, Y. Li, R. Hu, X. Zhang, and X. Chen, "Hand gesture recognition based on surface electromyography using convolutional neural network with transfer learning method," *IEEE Journal of Biomedical and Health Informatics*, vol. PP, p. 1, 2020.

[10] Thalmic Inc. (2015). How to wear the myo armband? [Online]. Available: https://support.getmyo.com/hc/en-us/articles/201169525-How-to-wear-the-Myo-armband (visited on 08/11/2020).

[11] Fluff, Sertion, and Ayane m. (2017). Rock, paper, scissors, [Online]. Available: https://commons.wikimedia.org/wiki/File:Rock-paper-scissors_(rock).png (visited on 09/29/2020).

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Learning and Intelligent Optimization*, C. A. C. Coello, Ed., ser. Lecture notes in computer science, Springer Berlin Heidelberg, 2011, pp. 507–523, ISBN: 978-3-642-25566-3.

[14] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[15] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[16] S. Tortora, M. Moro, and E. Menegatti, "Dual-myo real-time control of a humanoid arm for teleoperation," in *Proceedings of the 14th ACM/IEEE International Conference on Human-Robot Interaction*, IEEE Press, 2019, pp. 624–625, ISBN: 978-1-5386-8555-6.