Exo-Ada: A Boosting Model for Exoskeleton Angle Prediction

Petersen, Emil Thougaard
epeter15@student.aau.dk

Karlsson, Jonathan
jkarls15@student.aau.dk

Thillemann, Palle
pthill15@student.aau.dk

June 2020

AALBORG UNIVERSITY

STUDENT REPORT

Project Report

Group dt103f20

Aalborg University
Software Engineering

Abstract

Exoskeletons offer many possibilities within the contexts of manual labour as well as rehabilitation. A lot of effort has gone into improving the usability and efficiency of exoskeletons in order to extent their use cases. In order to control an exoskeleton, it is necessary to predict the intended movement of the user. A basic operation within the domain of intention estimation is that of elbow joint angle estimation. We have chosen to work with this prediction problem with measurements from a Force Myography (FMG) sensor armband. This problem is traditionally solved by training a machine learning model on data collected from a single person, used for intention estimation for specifically that same person. We instead wish to optimize this approach such that the solution can quickly be adapted to different users, reducing the amount of required training data for each novel person. Along this line, this paper proposes a novel boosting approach, Exo-Ada, specifically designed for use with exoskeletons in order to efficiently transfer to the domain of a new user. Exo-Ada is built on top of 2-Stage TrAdaBoost and uses a learner component made up by a rectified Convolutional Neural Network (CNN), making use of reference points and dilated convolutions. Exo-Ada outperforms several baselines on a test on FMG sensor measurements from multiple people.

1 Introduction

Powered exoskeletons are wearable machinery with motors that control its joints. They can be used on different limbs to assist with movement, increasing strength and endurance. This has many uses such as controlling prosthetics, rehabilitation from accidents, or lessening strain on the body during heavy-duty work. To utilize the motors on the exoskeleton, it needs to be able to detect the intended movement of the wearer, to know how much to bend its joints. In this work, we focus on detecting the angle of the elbow. To detect the intention of the user, we use two sensor armbands: one using Force Myography (FMG) sensors and another using an accelerometer. The FMG sensors measure the force exerted by the biceps and triceps, while the accelerometer measures the angle of the elbow. Using this data, we can train a neural network to learn the relation between the force exerted and the angle of the elbow. This means it can later estimate the angle of the elbow using only the FMG sensor readings, which is used to control the exoskeletons by flexing upper arm muscles.

The problem we are trying to solve is to make an exoskeleton usable for many people, meaning that it must also be inexpensive to train for new people. As such, the accuracy of predictions should not be dependant on having a large amount of data to train the neural network for a new person, as is the case with a traditional neural network. All existing solutions we could find on this problem, have been in different domains from ours and could not be adapted trivially.

Enter transfer learning, which can exploit data from previously recorded people to better fit a model to a new person, even if the data alone would not traditionally be enough to support this. Transfer learning is particularly useful to reduce such re-calibration efforts [Pan and Yang, 2009].

Our project lies within the area of *inductive transfer learning*, where the source and target domains (the source people and the target person respectively) are all fully labeled and within the same feature space. The difference between the domains lie in their varying data distributions. This is caused both due to the physical differences between people, as well as the sensor armbands being placed in slightly different positions for each use. See Figure 1 for the distribution differences between three people, illustrating the need for a dedicated transfer learning solution.

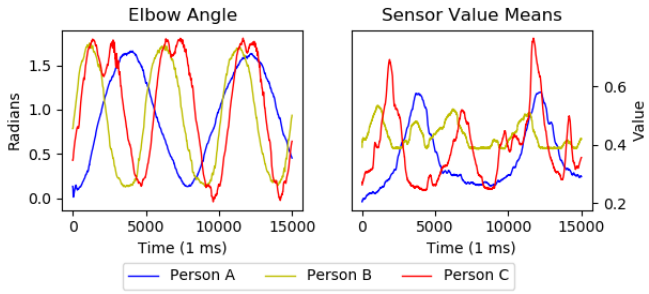


Figure 1: Despite similar angle measurements (left), the sensor measurements (right) are wildly dissimilar

It is important to emphasize that two different measurement sessions on the same person can have distribution differences rivaling the differences caused by measuring across different people. Therefore, session to session differences present a part of this project that equals person to person differences in importance.

This type of data makes transfer learning through AdaBoost [Dai *et al.*, 2007] an obvious fit [Pan and Yang, 2009]. We use AdaBoost as an instance-based transfer solution, which makes use of our source datasets (multiple people) and tries to squeeze as much knowledge out of these for the purpose of learning a target dataset (a novel person).

Our solution contains two main elements, first being exoskeleton angle prediction using regression. The second part is using transfer learning to use measurement data gathered from previously known people, in order to better and more efficiently learn features on a novel

person.

The goal of the project was to create a domain specific implementation of the AdaBoost boosting technique, to be used for transfer learning between people for exoskeleton angle prediction with FMG sensors. For this purpose, we have created Exo-Ada, an evolution on top of the 2-Stage TrAdaBoost algorithm [Pardoe and Stone, 2010], which takes some actions to specifically fit the algorithm to the new domain. We have achieved the following contributions:

- Exo-Ada adapts the 2-Stage TrAdaBoost algorithm to better allow transfer from multiple source domains
- As its base learner, Exo-Ada uses a CNN designed for the particular purpose of FMG exoskeleton prediction
- Our experiments, which show that Exo-Ada outperforms some baselines, but has some drawbacks as compared to 2-Stage TrAdaBoost

We will start by looking at related work and then we introduce notations and definitions for the problem. Next, we present the primary contribution of the paper: Exo-Ada, an algorithm designed to adapt previous solutions to work with our domain while adding some needed functionality. Finally, we present our experiments and results from testing our algorithm against four baselines.

2 Related Work

In this section we will briefly cover some closely related works to this research. We split it up into two distinct categories: exoskeleton angle prediction, with a focus on the problems of each method, and transfer learning methods, where we focus on regression transfer and boosting.

2.1 Exoskeleton Regression Problem

Islam *et al.* [Islam *et al.*, 2018] forms the basis for our understanding and implementation of elbow joint angle estimation. In this paper, the authors make use of FMG sensors and an accelerometer for data collection. They then train a support vector machine (SVM) to map the relation between joint angle and sensor values, meaning that the SVM can later predict the joint angle using only the sensor values. The authors have also previously worked on intention detection [Islam and Bai, 2017] and payload estimation [Islam and Bai, 2019]. Our solution adapts some of their work by using a more complex model with transfer learning and a CNN instead of an SVM to allow the need of less data to train the model.

We would also like to note that our presented research is a continuation of our previous research and observations [Petersen *et al.*, 2019]. In this instance, we were concerned with elbow joint angle estimation for a single individual, making use of different types of neural networks, among which we experimented with the implementation of practices such as reference points, as inspired by Ke *et al.* [Ke *et al.*, 2017], and dilated convolutions from Eddy [Eddy, 2018]. We observed that convolutional neural networks implementing these practices provided the best performance in terms of accuracy for the exoskeleton regression problem. Our previous research only concerned a single person, however, this project will use data from multiple people which is the reason for using transfer learning. We will be using these practices as a starting point for our implementation of regression transfer for elbow joint angle estimation, as well as a subsequent baseline to measure against the performance of our regression transfer approach.

The study by Liu *et al.* [Liu *et al.*, 2019] concerns a similar domain as ours, as it estimates the joint angle of the knee during walking motions to achieve smooth estimation of the human motion intention. It is a surface electromyography (sEMG) based sensor solution, requiring them to perform heavy pre-processing on the raw data before feeding it to their model. The paper proposes a feature-based CNN to learn the complex functions mapping the sensor signals to knee joint motions. Our solution uses FMG sensors instead of EMG, which should allow us to perform less pre-processing and achieve better usability by easing the data collection process.

2.2 Transfer Learning

A problem similar to the one proposed in this research, with regards to transfer learning, is by Wei *et al.* [Wei *et al.*, 2016]. In their research the authors propose a method to overcome the problem of label scarcity and insufficient data when modelling and predicting air quality in different cities. The proposed method involves boosting along with the construction of semantically related dictionaries in order to overcome the problems associated with insufficient labeled data from the target domain. The dictionaries contains related information for different sectors of a city; the information is transferred to sectors of different cities which have similar features. Their solutions uses a complex data representation along with dictionaries to overcome label scarcity and insufficient data, where we do not have any label scarcity but are trying to work with as little data as possible. We use a simpler data representation while trying to use a different form of boosting instead of the AdaBoost they use.

Another paper of particular note is by Yao and Doretto [Yao and Doretto, 2010], which proposes MsTrAdaBoost, an extension of the regular TrAdaBoost algorithm (as designed for classification problems), allowing transfer from multiple source domains. It does this by creating one weaklearner for each source+target

domain combination, but selecting only the strongest of these weaklearners to add to its ensemble per iteration. As such, it iteratively learns relevant features from across either of the source domains. Our solution instead finds relevant features from all source domains at once, and creates an ensemble model from those.

3 Preliminaries

In this section we will first formally introduce the notations and definitions we will be using in the context of regression transfer with a boosting approach. Secondly, we will introduce the formalization of the problem we will be trying to solve, that being elbow joint angle estimation across users. Then we will briefly cover concepts that are used in our approach for the Neural Network component that is not related to the boosting process itself. Finally, we will cover the background knowledge associated with boosting in relation to AdaBoost and its variations, which our boosting approach will be based upon.

3.1 Notations and Definitions

The notations used throughout this paper are described in Table 1. After the introduction of these notations, we move on to presenting definitions that describe the associations between different notations. We will not cover all notations in this section, however, the remainder of the notations not described here will be presented when they become relevant.

The regression problem of this research concerns multivariate time series data. This originates from the fact the exoskeleton which we are developing a solution for employs an armband A with a number of FMG sensors as well as a wristband W_r with an accelerometer, where values are captured for both A and W_r for each timestamp t. These factors, along with an implementation of a machine learning model, is what allows control of the exoskeleton. As stated previously we are interested in regression transfer for predicting the elbow joint angle of a user equipped with only A in order to operate the elbow joint of the exoskeleton, thus circumventing the need of W_r because these values are predicted. The series of timestamps is defined as follows:

Definition 3.1. $T = \{t_1, t_2, \ldots, t_n\}$ where t represents a single timestamp and n denotes the total number of timestamps.

For any given timestamp t we have a collection of values, one for each sensor on the armband A. We define A as follows: $A = \{a_1, a_2, \ldots, a_n\}$, where each a represents a sensor value and n defines the total number of sensors. From this we can now define the collection of all samples X as shown below:

Definition 3.2. $X = \{x_0, x_1, \ldots, x_n\}$ where the samples x follow the form $x_i = (A_i, t_i)$ where $0 \leq i \leq n$, and n in this context denotes the final timestamp in the series.

Table 1: Notations used in the paper

Notation	Description
$P, Session$	The set of persons and the collection of all sessions
D_s, D_t	The source and target datasets respectively
$Learner$	Describes a weaklearner, which, in theory, can be any arbitrary neural network
W, w	The weight vector (containing the weights of all data samples) and a single weight
E, e	A collection of errors for all samples, and a single error value, describing the prediction error for a single sample
$model_s$, $error_s$	An ensemble model that contains a collection of $Learners$, and the error associated with that ensemble model
S, N, F	The maximum number of boosting steps, the maximum number of boosting iterations, and the amount of folds used for cross validation
T, t	A series of timestamps, and a single timestamp
X, x	The collection of all samples (all sensor datapoints) and a single sample
Y, Y'	The collection of all ground truths and the collection of all predicted ground truths

Other than the sensor band A, we also employ an accelerometer in a wristband W_r which returns a value, w_r, for each timestamp. This value represents the elbow joint angle in radians at the current timestamp, and is used as such in our solution. We can define W_r as follows: $W_r = \{w_r\}$. From this we can now define Y as shown below:

Definition 3.3. The collection of all wristband values Y is $Y = \{(W_{r0}, t_0), (W_{r1}, t_1), \ldots, (W_{rn}, t_n)\}$ where t represents a single timestamp and n denotes the total number of timestamps.

The set of persons P is given as $P = \{p_0, p_1, \ldots p_n\}$. We make a distinction between labeled source and target data by having distinct datasets for these. The target dataset D_t is made up of labeled data for a single person, $p_0 \in P$, for whom we wish to adapt our regression transfer approach, such that $p_0 \cap P = D_t$. The source dataset D_s also used in our transfer regression approach is made up of labeled data from the remaining persons who are not represented in the target dataset $P - p_0 = D_s$. The datasets can be combined to form the collection of all samples X where the datasets are given in good order. How the datasets are combined is given in the following definition:

Definition 3.4. The collection of all samples X is the

combination of dataset D_s and D_t which represent the source and target domain respectively. X is constructed such that D_s represents the entries $\{x_0 \dots x_n\}$ and D_t represents the entries $\{x_{n+1} \dots x_m\}$. From this it follows that $X = \{x_0 \dots x_m\}$ and $X = D_s \cup D_t$ also holds. Furthermore, since X is the combination of D_s and D_t, we can define $X^{D_s} = D_s$ and $X^{D_t} = D_t$ for the source and target dataset.

Moreover, all labeled data obtained from persons in P have not been collected in one take to make up the collection of all samples X. Rather, they have been collected through different sessions s which are each associated with a unique person p belonging to either D_s or D_t. The combination of all sessions s is what makes up the collection of all samples X.

Definition 3.5. A session s is a collection of samples $\{x_0^{p_i}, x_1^{p_i}, \dots, x_{n'}^{p_i}\}$ associated with a unique person p_i where n' denotes the total number of samples for a session. The samples of a session s is given in good order, thus n' can also be taken to denote the number of timestamps t for respective sessions. A session is of the form $s_j^{p_i} \in \mathit{Session}$ where $0 \leq i \leq n$ and $0 \leq j \leq m$. In this context n denotes the total number of people $p \in P$ and m denotes the total number of sessions captured for each person. Because of the structure of a session s, it follows that $\mathit{Session} = X$ which also implies $\mathit{Session} = X^{D_s} \cup X^{D_t}$.

3.2 Problem Formalization

We now formalize the problem that we have attempted to solve with our research. To begin with, we will introduce what we consider as the input and output of our solution. As data from A, represented in X, and W_r, represented in Y, are related through timestamps t, we can create two series of pairs, one for the input and one for the output (note that we have substituted x with its components (A, t) in X for clarification purposes):

$$\text{Input: } X = \{(A_0, t_0), (A_1, t_1), \dots, (A_n, t_n)\}$$

$$\text{Output: } Y = \{(W_{r0}, t_0), (W_{r1}, t_1), \dots, (W_{rn}, t_n)\}$$

The problem has now taken shape, as the goal becomes to estimate the value of W_r with the values of A at each timestamp t. Since we will be performing regression transfer we will need to distinguish between our source domain and target domain. For our purposes we will use the notation for source and target datasets, D_s and D_t, in order to make this distinction. The act of regression transfer itself will be performed by Exo-Ada, denoted as $\mathcal{F}$, which we will describe in section 4. It will take as input labeled source data from D_s, as well as labeled target data from D_t, available in order to transfer to the target domain. The labeled data will be in the form of sessions $s \in \mathit{Session}$ such that:

$$X^{D_t} = \{s_0^{p_0}, s_1^{p_0}, \dots, s_j^{p_0}\}$$

$$X^{D_s} = \{s_0^{p_1}, s_1^{p_1}, \dots, s_j^{p_1}, \dots, s_0^{p_n}, s_1^{p_n}, \dots, s_j^{p_n}\}$$

Where person p_0 represents the target domain, and person p_1 through p_n represents the source domain. Recall from definition 3.5 that each session s contains a collection of samples x which can be associated to respective timestamps t for each session. Thus the relation between X^{D_t}, X^{D_s} and Y is kept through t when we make a prediction. We can define our prediction goal for regression transfer as:

$$Y'^{p_0}_{t+1} = \mathcal{F}(X^{D_s}_{t-h}, \dots, X^{D_s}_{t-1}, X^{D_s}_t, X^{D_t}_{t-h}, \dots, X^{D_t}_{t-1}, X^{D_t}_t)$$

Here h represents the past history of timestamps and associated values of A in both D_s and D_t respectively, which we take into consideration when we make a prediction of Y'^{p_0} at timestamp $t+1$.

3.3 Learner Concepts

In this subsection we will briefly cover the concepts that we make use of in the *Learner* component introduced in table 1. Based on previous work in [Petersen *et al.*, 2019] we have chosen that the *Learner* component will be made up by a rectified CNN that makes use of both reference points and dilated convolutions.

Reference Points The use of reference points [Ke *et al.*, 2017] in this context is a way to represent relationships between the different sensor values a, as if they were spatial relationships. With this, sensor values in A are chosen as anchor points a^{anc} to which the remaining sensor values $\{a^1 \dots a^n\}$ are changed in relation to, for each timestamp t. The process of creating this spatial representation Z between the sensor values can be described with the following function:

$$Z_t = \mathcal{G}(a_t^{anc}, \{a_t^1, \dots, a_t^n\})$$

The function $\mathcal{G}$ calculates a new value for each sensor value $a \in \{a^1 \dots a^n\}$, except for the sensor that was chosen as the anchor point, at each timestamp t:

$$\mathcal{G}(a_t^{anc}, a_t) = a_t - a_t^{anc}$$

If we are using multiple anchor points, the resulting $\{1 \dots n\}$ reference values are concatenated at each timestamp.

Dilated Convolutions The use of dilated convolutions in CNNs [Eddy, 2018] allows us to substantially increase the receptive field of the output neuron without adding additional hidden layers to the CNNs, which are used as *Learners* for our boosting approach. This is because the receptive field R_f increases exponentially

as a function of convolution layer depth L_d when using dilation:

$$R_f = L_d^2$$

The receptive field increase is achieved by skipping certain neurons in the hidden layers according to a dilation rate. This rate increases multiplicatively (e.g. 1, 2, 4, 8 etc.) for each hidden layer and is what allows the exponential relationship between layer depth and receptive field size.

3.4 Transferring Knowledge with AdaBoost

In this section we explain essential background knowledge of boosting techniques, and how this applies to transfer learning with TrAdaBoost, which is necessary prerequisite to understand the rest of this paper.

AdaBoost [Freund and Schapire, 1997] is a technique designed for classification problems, which combines multiple less powerful learners (called *weaklearners*), that are iteratively being trained on the same dataset, into one powerful ensemble learner (called a *stronglearner*). All throughout this process, AdaBoost changes the importance of the individual samples from the dataset, where it prioritizes the "hard-to-learn"-features, performing the so-called "boosting", to ensure that all important features are eventually being learned by a weaklearner. As such, each weaklearner trains to predict different kinds of features. Lastly, when the stronglearner is used for prediction, each of its constituent weaklearners votes on the result, the importance of each vote depending on the prediction strength of the weaklearner.

TrAdaBoost [Dai *et al.*, 2007] redesigns AdaBoost for the purpose of transfer learning from one source domain to a different target domain. It is a technique for classification problems within the area of *inductive transfer learning* [Pan and Yang, 2009]. It works by iteratively increasing the weights of the target samples, creating one AdaBoost ensemble learner per boosting step. As such, it elegantly attempts to find the "sweet spot" between the source and the target domains, to identify and prioritize the most transferable features from source dataset, whilst concurrently learning the most important features of the target dataset. As such, it attempts to transfer as many relevant features as possible, whilst ignoring the features that are not transferable, and avoiding the common transfer learning problem of negative transfer (where a model loses accuracy through the addition of less relevant data).

Lastly, 2-Stage TrAdaBoost [Pardoe and Stone, 2010] modifies AdaBoost for use in regression problems. For this purpose, it splits the algorithm into two main steps: one where it is only allowed to modify the weights of the target domain, and one where it is only allowed to modify source.

In the context of our project, a number of modifications are necessary. Most importantly, we transfer to a novel person from multiple source domains (multiple source people), rather than from just one source domain.

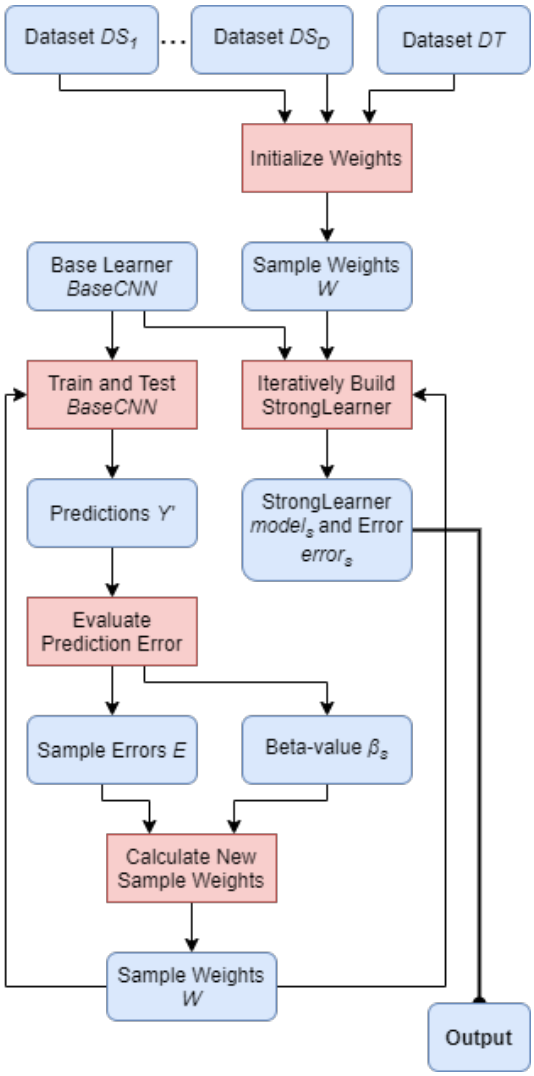


Figure 2: Visual overview of Exo-Ada. The red squares represent actions which take some input and outputs something. The blue squares with rounded edges represents stored information, that is used as input for some action.

4 Exo-Ada

In this section we will present our proposed method in detail. We will first introduce an overview of the general framework in figure 2 which includes all major parts of the method, from data pre-processing to final prediction. This introduction will be followed up by a detailed look at the Exo-Ada algorithm, which is a modified version of Two-Stage TrAdaBoost [Pardoe and Stone, 2010].

4.1 Overview

AdaBoost attempts to avoid common transfer learning problems such as negative transfer, by determining when it is even useful to transfer a sample through its sample weight values [Pan and Yang, 2009]. To the best of our knowledge, a version of regression-AdaBoost that is designed for multiple sources does not exist. So we have to make our own improvements to allow for this, as it is necessary for our domain.

The complete Exo-Ada algorithm is given in algorithm 1 showing the steps that are taken from input to output. A more simple overview of the algorithm is visualised in Figure 2. We will go into details with its most important parts later.

Algorithm 1: Exo-Ada

Input: A number of labeled source datasets D_{s1} to D_{sd} of sizes ss_1 to ss_d respectively, belonging to X^{D_s}, one labeled target dataset D_t of size st belonging to X^{D_t}, the amount of boosting steps S, the amount of boosting iterations N and the amount of folds for cross validation F.

Initialize: Assign the total sample weight of each dataset to $TW = \frac{1}{d+1}$. Initialize the weight vector of the target dataset, W_t, such that each sample, w_1 to w_st has a value of $w = \frac{1}{TW}$. The source datasets are initialized in a similar manner.

For $s = 1, \ldots, S$:

1. Call AdaBoost.R2' with the base learner *BaseCNN*, iterations N and weight vector W to obtain the ensemble $model_s$.
2. Use F-fold cross validation and call AdaBoost.R2' with the *BaseCNN*, iterations N and weight vector W, to obtain the mean estimate $error_s$ over $model_s$.
3. Call *BaseCNN* on samples X and weight vector W to obtain predicted ground truth Y'.
4. Calculate the error vector E corresponding to the predicted ground truths in Y' as in Two-stage TrAdaBoost [Pardoe and Stone, 2010].
5. Calculate β_s such that the resulting weight of the target samples (the final st weights in W) is $TW + \frac{s}{S-1} * (1 - TW)$
6. Update the weight vector as in Two-stage TrAdaBoost [Pardoe and Stone, 2010] using β_s.

Output: The $model_s$ with the lowest $error_s$

4.2 Main Concepts

Here we define the main concepts that Exo-Ada makes use of. Since our approach is based on boosting, it requires a collection of base learners in order for boosting to be performed. For our purposes this will be fulfilled by the *Learner* component, which along with the input dataset X provides the foundation for our solution. Specifically, we use the *BaseCNN*, which is a rectified CNN that uses reference points and dilated convolutions to predict the elbow angle.

Definition 4.1. A *Learner* is a base component for our boosting solution. Within boosting it is commonly referred to as a weaklearner, that is constructed from a base learner throughout the boosting process. In our case, we use the *BaseCNN* as our base learner. Each *Learner* is associated with a specific estimation weight, W_E, which is calculated as in 2-Stage TrAdaBoost, using the weights W and errors E of all samples X.

The weight W_E is used as a measure of the learner's accuracy belonging to the target domain D_t at the current boosting iteration N. As we progress through the algorithm, we create new learners that improve upon the accuracy of the previous learner, for the samples x that were hardest to estimate.

Likewise a $model_s$ represents an ensemble of multiple *Learner* components, which is also commonly referred to as a stronglearner. Because of this, each $model_s$ is associated with a collection of weights, one W_E for each *Learner*. As such we can define a $model_s$ as:

Definition 4.2. An ensemble model, $model_s$ is a collection of learners: $model_s = \{Learner_1, Learner_2, \ldots, Learner_n\}$ where n represents the number of learners in the ensemble. Each ensemble model has a collection of weights, one W_E for each constituent *Learner*, and an $error_s$ which is the error of the ensemble model measured on target domain samples x from D_t.

4.3 Initialization

After receiving the necessary input, the algorithm initializes sample weights for all datasets W, such that each dataset has the same amount of total weight regardless of how much data it contains. Since, at the moment of initialization, we know nothing about the samples contained within each dataset, and therefore cannot pass judgement on the applicability of the different source datasets compared to the target person, our start assumption is that each dataset is precisely as important as the others. This is regardless of the individual lengths of the different datasets that make up our total samples X - otherwise we would unfairly bias longer measurement sessions, even though they might contain very little transferable knowledge. Recall that the major challenge of this research is to predict intentions across multiple users, meaning that it is prudent to focus our efforts on learning features that are applicable across multiple users.

This initialization is essential, since the algorithm immediately starts adjusting source vs target weight distributions, meaning that we need an acceptable start

point, and that bad start sample weights would not just fix themselves over boosting steps S.

4.4 Ensemble Model

In step (1) of algorithm 1 we call AdaBoost.R2' with the base learner and the weight vector W of the current step. AdaBoost.R2' iteratively creates N different weaklearners, *Learner*, combining them into a single ensemble model $model_s$, the stronglearner. Whilst building $model_s$, each weaklearner learns a particular set of features, and is then evaluated. The weights of the target domain samples are then adjusted to make certain that we learn even the hardest features throughout the process of stacking weaklearners on top of each other.

To calculate the error of each ensemble learner, we recall that only the performance on the target domain matters. Unfortunately, since target data is scarce in our problem, we cannot be confident in the results. As such, step (2) uses cross validation to run multiple versions of the current $model_s$, each adapted to an F-fold part of the target domain, and average the results to gain the most accurate possible estimation error $error_s$ of stronglearner $model_s$. At the end of the algorithm, the $model_s$ with the lowest $error_s$ is used as output. The final predictions of this ensemble model is the weighted median of each of its constituent weaklearners' predictions.

4.5 Weight Update

The last parts of the Exo-Ada algorithm are where we update the weights of all samples. In step (3) we create and fit a new *BaseCNN* model on all samples, using the same weight vector we used in prior steps W. We use this new model to get a prediction for each sample. In step (4) we use the predictions to calculate the error e of each sample. The error for a specific sample e_t^s is calculated by: $e_t^s = |y_t - y'_t|/ME_s$ where ME_s is the highest error obtained, which is then used for normalization between 0 and 1. Here y and y' are the individual ground truths and predictions taken from $Y = \{y_0 \dots y_n\}$ and $Y' = \{y'_0 \dots y'_n\}$. This error vector allows us to judge, for our current iteration of sample weights, which source samples were the best transferable to the target domain.

Then in step (5) we approximate a value β_s using binary search, one which will ensure that the importance of the target samples increases for each boosting step. In step (6) we use the error vector E and β_s to update the weights of each source sample, ready for later boosting steps. Updating the weights is done using the formula:

$$w_i^{s+1} = \begin{cases} w_i^s \beta_s^{e_i^s} / ZN_s, & 1 \le i \le n \\ w_i^s / ZN_s, & n+1 \le i \le n+st \end{cases}$$

Where $n = ss_1 + ss_2 + \dots + ss_d$, and ZN_s is a normalizing constant such that the sum of all weights W_s stays at 1.

This ensures that we, throughout our boosting steps S, increase our focus on only the most transferable features within source. This also means that the distribution of weights between the different source datasets is bound to change depending on which dataset is the most similar to our target dataset.

We use Mean Absolute Error (MAE) for our loss function for the algorithm, which is given by:

$$MAE = \frac{\sum_{i=1}^{n} y'_i - y_i}{n}$$

Here n is the length of the given data in the specific situation.

5 Experiments

This section describes the dataset and baselines, and compares their results to Exo-Ada. We will then go into detail with some noteworthy Exo-Ada results to explain some of its qualities and shortcomings. All our tests evaluate model performance based on the MAE values obtained while attempting to estimate a single session, that is purely used as the test set.

5.1 Dataset

Our data is collected using our two sensorbands with a MATLAB application that allows us to calibrate the FMG sensors, perform data-collection and choose the data we want to save. The data is recorded in sessions s of roughly 60 seconds each, where the user performs lifting sequences, as in Figure 1. The data is recorded in 1000 Hz, meaning that 60 seconds of data will generate 60,000 samples of data. In our tests we use a granularity of 30, which means that we average every 30 samples, to get rid of unnecessary and repeat data for the purposes of improving training time. Granularity of 30 was chosen, as the samples often stay the same for 10-30 samples in a row, meaning it would not change much to combine them. We chose 30 over something like 10 or 20 as it would allow us to test with more steps and iterations. We perform five measurement sessions for each person and have data from three different people. We refer to these people as person A, B and C as well. The first two people are used as source D_s, and the third person is used as target D_t. For the target person, we use the first four sessions as training data, and the last session is used purely as test data. We refer to this last session, s_5^{p3}, as the test session.

5.2 Baselines

This section describes the baselines used for our experiments. The common thread for all baselines is that they concern themselves with only their performance on the last session of person C, though some are able to make use of sessions from person A and B to help for this purpose.

CNN Small This baseline uses our *baseCNN* estimator, but is only trained on person C. It shows the accuracy if we were to forego any kind of cross-person transfer learning.

CNN Big This model also uses the *baseCNN* estimator, and shows the performance on person C, if we naively train on person A, B and C, whilst testing on person C, while the model has no intrinsic knowledge of transfer learning.

Ensemble CNN This model consists of one *baseCNN* estimator trained on each person individually. Its predictions are the means of the constituent models' predictions (ergo naive voting, not based on weight). This model shows how voting might be an important way to squeeze additional knowledge out of transfer learning solutions.

2-Stage TrAdaBoost This baseline utilizes 2-stage TrAdaBoost R2 [Ren, 2018][Pardoe and Stone, 2010] with out-of-the-box configurations, utilizing decision trees as its base learner. This model has an intrinsic knowledge of the difference between source and target samples and introduces boosting as a way to evaluate and highlight the relevancy of individual samples to the base learner. This baselines also measures the strength of weight based voting. Since the decision trees are not designed for multivariate data, we flatten the inner layer of the data array.

Exo-Ada This model is based upon 2-Stage TrAdaBoost, its main difference being that it utilizes our domain specific *baseCNN* estimator as its base learner. Additionally, its adds functionality allowing it to distinguish between the multiple source domains (person A and person B).

The *baseCNN* has had its hyperparameters tuned through 500 iterations, and includes the domain specific initiatives of reference points and dilated convolutions, as described in subsection 3.3. See all hyperparameters below:

Keras Epochs 10, batch size 500, optimizer "adam"

CNN Filters 94, kernel size 2, past history 267, layers 5, padding "valid", kernel initializer "uniform", activation "relu", dilation rate 2, reference points "one at biceps, one at triceps"

Boosting Steps 10, folds 3, learning rate 1, decision tree max depth 4, loss "linear"

Additional to these parameters, 2-Stage TrAdaBoost uses 25 iterations N, which cannot be set much higher due to training time constraints. Exo-Ada instead uses 5 iterations N, as initial tests show that its performance does not meaningfully increase and might even decrease with more weaklearners, because its individual weaklearners are much more powerful. Further tuning of the Exo-Ada and 2-Stage TrAdaBoost algorithms is a domain for future expansion of the tests.

5.3 Results

Accuracy Comparisons

One of our main goals with this research was to increase the quality of the regressor in the cases where little target data is available. This test shows the performance of our model and baselines, as we we decrease the amount of target data available for training, see Figure 3.

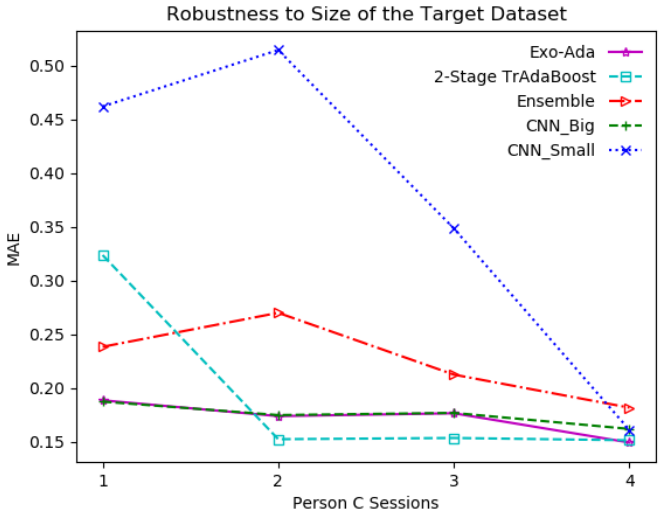


Figure 3: Comparison of the error of Exo-Ada and the baselines, as we increase the number of sessions available for training person C

Looking at *CNN_Small*, the results show that, when using the full 4 target sessions for training, it is able to learn sufficient knowledge for good performance compared to the other baselines, almost rivaling the performance of the boosting methods without employing any kind of transfer learning. As we reduce the available data, however, *CNN_Small* quickly falls behind, proving the importance of transfer learning solutions in cases with less data. Comparatively, with 4 available sessions, *CNN_Big* is affected by negative transfer, making its performance worse than *CNN_Small* in this case, since it learns too many unrelated features from other people and has no procedures to distinguish between different people. Notably, as we reduce available training data, *CNN_Big* loses the least accuracy along with *Exo-Ada*, because it focuses on learning features that are shared across sessions and people.

The *Ensemble CNN* has some similarly conservative results, the accuracy of which are not particularly affected by the size of the target dataset. It falls behind the boosting strategies because it lacks weight based voting, opting for a naive mean of the constituent results instead. Moreover, it performs worse than *CNN_Big* because its constituent models are not able to learn features shared across multiple people.

Exo-Ada performs noticeably better than *CNN_Big*

in the case of 4 available target sessions, owing to its boosting strategy that elegantly puts more emphasis on hard to estimate features across all people and sessions, squeezing more knowledge out of the available data, so long as there are enough sessions for it to actually draw deeper meaning from.

Exo-Ada prominently outperforms *2-Stage TrAdaBoost* in the case of only 1 available training session, managing to keep the good performance of its base learner. This is achieved by avoiding negative transfer in conditions where the target data is not representative of the test set because of notable session to session differences. Recall that the boosting approaches are designed to focus on learning only the features from its source domains, which are relevant for its target domain.

The major difference here is that *Exo-Ada* uses the domain specific *baseCNN* whilst *2-Stage TrAdaBoost* uses a general purpose regression decision tree, and the results represents a fundamental difference in how the two base learners operate. While the *baseCNN* focuses on learning features that concern the relationships between sensor values over time, the decision tree has no such notions. As such, it appears that the *baseCNN* learns features that span across sessions, learnt from source, before it even has more than one target session to train for, allowing *Exo-Ada* and *CNN_Big* to get a head start on *2-Stage TrAdaBoost*.

However, this approach has clear drawbacks in the case of 2 and 3 available sessions, where *Exo-Ada* does not improve noticeably. It clearly is great at capturing the most obvious cross-session features immediately, while it has trouble grasping the inner meanings until sufficient data has been provided. The weaker decision trees used by *2-Stage TrAdaBoost* have the advantage of focusing more narrowly on the features that are the most relevant to the available target domain, rather than having an inborn general intuition of the FMG sensor feature space. As such, it is our notion that *2-Stage TrAdaBoost* inherently is better at transferring data from other people, while *Exo-Ada* has more inherent knowledge about the problem domain, meaning that its performance is more consistent, and rarely swings wildly dependant on the quality of the available target datasets.

The experiments show that *Exo-Ada* is on par with, or better than, all baselines in all tested situations, other than *2-Stage TrAdaBoost* in the case of 2 and 3 available target sessions. As the MAE values for the 4 session measurements are hard to distinguish on the graph, we repeat them here: *Exo-Ada* has an MAE of 0.149, *2-Stage TrAdaBoost* 0.152, *Ensemble CNN* 0.182, *CNN_Big* 0.162 and *CNN_Small* has 0.161. As we can see, *Exo-Ada* performs better than even *2-Stage TrAdaBoost* in this situation, however, we are vary of jumping to conclusions based on only such marginal differences.

5.4 Exo-Ada Deep Dive

In this section, we will look into detail at some of the qualities of Exo-Ada.

Predictions

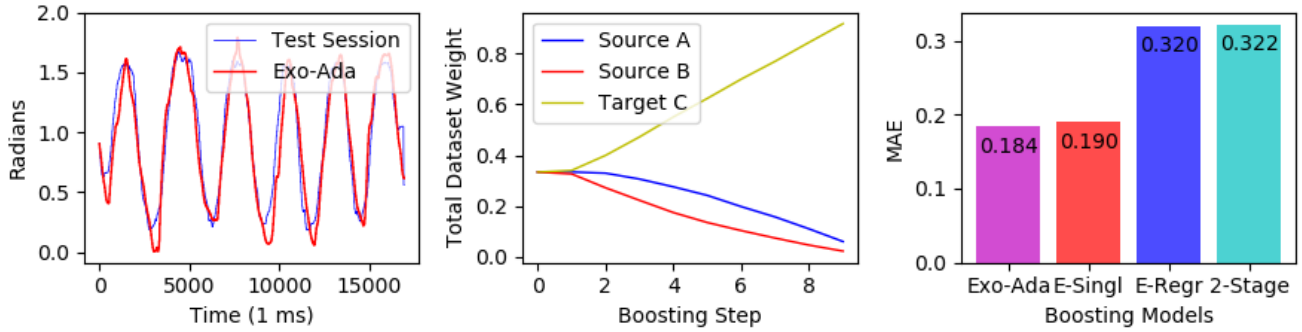
See Figure 4 (a) for the plotted predictions on a representative subset of the test session. Its main difficulties lie in correctly predicting the peaks and, most notably, the valleys of individual lifting sequences. This appears to be a consistent shortcoming of Exo-Ada regardless of how many iterations N it uses to create its ensemble learners. This deficiency might be caused by the way that the *baseCNN* convolutes a long history of measurement samples to make its predictions, making it performant on the majority of the dataset, but not for the peaks and valleys, that have a high rate of change with regards to the elbow angle.

Source vs Target Similarities

Going further in-depth with the inner workings of Exo-Ada, we see in Figure 4 (b) the sample weight distributions between the datasets of each person used for training, over the boosting steps S. It uniformly increases the value of the target dataset over its steps, building one stronglearner per step, and later selecting the one with lowest error to perform its final predictions. The multi-domain adaptations of Exo-Ada cause the weight distributions of each of the three datasets to start at 0.33, regardless of differences in the amount of samples within each dataset.

Strikingly, and this result is substantiated by almost every other performed test, we see that the algorithm prefers to prioritize samples from person A higher than person B, as it finds more transferable features within person A's dataset. This difference likely has to do with multiple factors affecting the feature similarities, therein the quality of the dataset, and the physical build of the people involved. In this case, the build of the two people it likes to associate is similar. This leads to future work of note, as it may be worth investigating additional actions that can be taken based upon "expert" knowledge of the similarities between the datasets.

Another element of note is that Exo-Ada commonly selects the stronglearner created at boosting steps 0-3 as the one with the lowest $error_s$, while 2-Stage TrAdaBoost commonly goes for indexes 7-9. This represents a fundamental difference in how the two models learn features from across sessions and people. This divergence is likely caused by the difference in relative predictive strength between using a CNN as the base learner, rather than using a weaker regression tree, and letting the stacking of many weaklearners do the job. Whereas the CNN's can achieve good performance across sessions and people to begin with, the regression trees may require the later boosting steps to have pre-tuned the weights of the target dataset, in order to focus on the most important features first.



(a) Elbow Angle Estimation (b) Sample Weight Distributions (c) Ablation Study

Figure 4: (a) shows predicted elbow angle Y' compared to the ground truths from a subset of the test session $s_5^{p_3}$, (b) shows the distributions of sample weights W between the datasets (all training sessions) by person p_1, p_2 and p_3 (c) shows the error of ablated models

5.5 Ablation Study

Figure 4 (c) presents the ablation study to display the impact of individual components we have added, and see how they contribute to the performance of Exo-Ada. We test the four ablated models: *Exo-Ada*, which uses the full Exo-Ada algorithm, *E-Singl*, which removes Exo-Ada's multi-domain adaptations, *E-Regr*, that uses regression decision trees as its base learner, and *2-Stage*, which is the original 2-stage algorithm. The tests are run in the condition where Exo-Ada has achieved the most improvement compared to the baselines, where only one target session is available for training.

Clearly, the most important performance impact is caused by the removal of the *baseCNN*, which is what allows us to learn cross-session differences so well. The multi-domain adaptations show performance increases as well, that may be explained by the model not missing its source vs target weight balancing "sweet spot" because it has to spend the first few steps S tuning the target weights to emphasize its most important samples. The ablation also shows that both of our adaptations are required to achieve the qualities of Exo-Ada.

6 Conclusion

This paper proposes a domain specific method called Exo-Ada for the use with FMG-based exoskeletons to predict a user's elbow angle. The method uses knowledge gained from other users through transfer learning to improve its performance on the novel person. Exo-Ada adapts the algorithm 2-Stage TrAdaBoost to better transfer from multiple source domains and uses CNNs designed for the particular domain of exoskeleton elbow estimation. Our experiments show that Exo-Ada outperforms several baselines in tests, but is still only on par with some of the others at its best. Results did, however, show a clear advantage of using Exo-Ada when too few target data sessions are available.

7 Future Work

We believe that Exo-Ada can perform better than shown in this paper and we have several areas we would like to experiment with in the future. It would be interesting to experiment with some other *Learners* to see what kind of effect this could have, and possibly even using multiple, different base learners, to harness the powers of each.

Our dataset in this paper only consisted of three people, which could be expanded to facilitate more tests about the amount of data, and to see if our hypotheses about Exo-Ada preferring to boost samples from people with a similar build hold true, and if so, how this could be exploited.

We would also like to do some general optimization of the algorithm as we started encountering memory problems when running with too much data. This, along with more powerful hardware, would allow us to tune the models, particularly Exo-Ada and 2-Stage TrAdaBoost, much more precisely, and be able to better gauge their true performance limits.

Lastly, we had some additional features in mind, such as adding *StartSteps*, which would be a specifiable amount of steps the algorithm would take without changing the weight distributions, to allow more time to emphasize the most relevant features of the target dataset. Intuitively this would allow the algorithm to not miss the source vs target "sweet spot", simply because it had not highlighted the most essential target features yet.

Acknowledgments

We would like to thank our project supervisor Chenjuan Guo. Additionally, we would like to thank Muhammad Raza Ul Islam, research assistant at Aalborg University, for providing the data collection software and the sensor armband utilized in this project.

References

[Dai *et al.*, 2007] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200, 2007.

[Eddy, 2018] Joseph Eddy. Time series forecasting with convolutional neural networks - a look at wavenet, 2018.

[Freund and Schapire, 1997] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting, 1997.

[Islam and Bai, 2017] Muhammad Raza Ul Islam and Shaoping Bai. Intention detection for dexterous human arm motion with fsr sensor bands. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 139–140. ACM, 2017.

[Islam and Bai, 2019] Muhammad RU Islam and Shaoping Bai. Payload estimation using forcemyography sensors for control of upper-body exoskeleton in load carrying assistance. 2019.

[Islam *et al.*, 2018] Muhammad RU Islam, Kun Xu, and Shaoping Bai. Position sensing and control with fmg sensors for exoskeleton physical assistance. In *International Symposium on Wearable Robotics*, pages 3–7. Springer, 2018.

[Ke *et al.*, 2017] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Ahmed Sohel, and Farid Boussaïd. A new representation of skeleton sequences for 3d action recognition. *CoRR*, abs/1703.03492, 2017.

[Liu *et al.*, 2019] Geng Liu, Li Zhang, Bing Han, Tong Zhang, Zhe Wang, and Pingping Wei. semg-based continuous estimation of knee joint angle using deep learning with convolutional neural network. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 140–145. IEEE, 2019.

[Pan and Yang, 2009] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[Pardoe and Stone, 2010] David Pardoe and Peter Stone. Boosting for regression transfer. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 863–870, 2010.

[Petersen *et al.*, 2019] Emil Thougaard Petersen, Frederik Østerby Hansen, Jonathan Karlsson, Matias Dahlin Holst, Mikkel Vestergaard Hem, and Palle Thillemann. Exploring neural network models for estimating accelerometer data with multivariate time-series prediction, 2019.

[Ren, 2018] Jie Ren. Two-stage tradaboost github. https://github.com/jay15summer/Two-stage-TrAdaboost.R2, 2018.

[Wei *et al.*, 2016] Ying Wei, Yu Zheng, and Qiang Yang. Transfer knowledge between cities. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1905–1914, 2016.

[Yao and Doretto, 2010] Yi Yao and Gianfranco Doretto. Boosting for transfer learning with multiple sources. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1855–1862. IEEE, 2010.