A Machine Learning Approach to Predicting Distal Leg Muscle Activation for Real-time Control

Application Track, 4 Team Members

Abstract

This work explores three machine learning approaches to predict muscle activations in the lower limbs during walking. Our approach is motivated by the potential to integrate this prediction into real-time control for above-knee robotic prostheses, allowing the user to control the prosthesis by activating intact muscles. First, we pre-process the data so that each feature instance includes enveloped muscle activation data over a certain history window, gait cycle phase, and walking speed. Then, we apply Kernel Ridge Regression and Support Vector Regression and evaluate their performance given the computational limits with our very large dataset. Then, we design and fit a Temporal Convolutional Network and again evaluate performance. Due to the TCN's high performance, we also test its accuracy on additional tasks such as ramp ascent. All methods were able to predict the EMG data with somewhat reasonable accuracy ($> 60\%$ VAF minimum), and the TCN model was highest performing with $> 92\%$ VAF across all tasks. Finally, we discuss the different methods, including accuracy trends seen across muscles, and forward propagation complexity in regards to real-time control feasibility.

1 Introduction

1.1 Motivation

Muscles are recruited by the central nervous system through an electrical activation stimulus. This activation stimulus (command) can be measured directly for each muscle using electromyography (EMG), and EMG data has been catalogued in datasets for various activities [1] (See Appendix B Figure 4) . Muscle synergies, which are groups of muscles that fire in consistent patterns, have been shown to exist throughout the lower limbs during walking [2]. These groupings show that inherent patterns and relationships exist between different muscle activations. **This interdependence suggests that machine learning approaches could learn the nonlinear relationships between muscle activations and predict EMG signals in some muscles based on measurements of EMG from others.**

The ability to predict muscle activation in some muscles based on others may have profound impacts in the field of robotic prosthesis control. To enable volitional control, where the user can directly control the artificial joint torques at will, some researchers have implemented real-time measurements of EMG and used it to apply supplemental joint torque [3]. *While this method has shown promise for transtibial (below-knee) prostheses, it cannot be directly applied to transfemoral (above-knee) prostheses because the muscles that would actuate the ankle joint no longer exist for a person with a transfemoral amputation*. Therefore, we propose a similar method where instead of measuring the EMG for the ankle muscles directly, it is predicted from the more proximal leg muscles.

This paper will be organized as follows: the next two subsections will discuss the problem statement and related work. Section 2 will then discuss the methodologies and theoretical foundations for each of the different machine learning techniques used. Section 3 presents an evaluation of each model as

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

well as the performance in predicting EMG signals. Finally, the last section will present a discussion of results and concluding thoughts from this study.

1.2 Problem Statement

Specifically, our goal is to **apply Kernel Ridge Regression (KRR), Support Vector Regression (SVR) and a Temporal Convolutional Network (TCN) to predict the activation of the primary plantarflexor and dorsiflexor muscles, the gastrocnemius (GA), tibialis anterior (TA), and soleus (SOL), based on EMG readings from more proximal leg muscles**. Equation 1 shows the mathematical problem we are trying to solve, where $\widehat{Y}_t^k$ is the predicted EMG signal for $k \in \{\text{GA, TA, SOL}\}$ muscles at a given time instance t based on current and previous muscle activation values contained in X_t. The size of X_t will vary by method. We seek to find a function mapping f_k for each $k \in \{\text{GA, TA, SOL}\}$ such that the mean squared difference between the actual EMG values Y_t and $\hat{Y}_t$ is minimized:

$$\text{Seek } f_k \text{ such that } \widehat{Y}_t^k = f_k(X_t) \text{ and } \frac{1}{n}\sum_{t=1}^{n}(\hat{Y}_t^k - Y_t^k)^2 \text{ is minimized for each } k. \tag{1}$$

To the authors' knowledge, prediction of distal leg EMG based on proximal EMG signals has not been attempted prior. We trained our predictors using the novel, multi-activity dataset presented in [1]. This dataset was released this calendar year, so it is very unlikely that it has been extensively studied using machine learning methods. So far, there are only 7 citations to this dataset on Google Scholar. Only one of these papers is machine learning related, and is about using CNNs to estimate the ground inclination while a user walks in an exoskeleton [4]. Therefore, we believe that our proposed project is sufficiently novel.

1.3 Related Work

Recent work with similar yet distinct goals suggest that our methods will be successful. For example, Gupta et al. used EMG data as inputs to Linear Discriminant Analysis, SVMs, and Neural Network classifiers to predict muscle fatigue in the ankle and foot [5]. Our study will differ from theirs because we are aiming to predict SOL, TA, and GA EMG signals directly instead of fatigue. Another study predicted forearm tremor EMG signals for people with Parkinson's disease using Multilayer Perceptron and Recurrent Neural Network models [6]. While our approach is similar, this study differs because we are studying different limbs and implementing different machine learning approaches.

While muscle synergy calculation was not part of our process, it is relevant to the problem because it has potential to provide intuitive understanding for why some muscle activations are able to predict others. Machine learning techniques have commonly been used for muscle synergy extraction as well. Non-negative matrix factorization (NMF) is a very common method [7]. Because of the relative success of NMF in processing EMG data, it has become a standard method for obtaining muscle synergies [8, 9] and a performance metric for new EMG processing algorithms [10]. New methods such as Multivariate Curve Resolution-Alternating Least Squares (MCR-ALS) have given better results than NMF [10]. However, NMF is still the standard regarding EMG processing algorithms for extracting muscle synergies.

In recent years, TCNs have become a popular and effective method for modeling sequential data, even outperforming Recurrent Neural Networks (RNN) and Long Short-Term Memory networks (LSTM) on similar tasks [11, 12]. They have been extensively used for gesture prediction using EMG signals [13, 14, 15]. Additionally, they have been used with great success for other sequential data prediction tasks such as weather forecasting [16] and speech recognition [17].

While the studies listed above show that EMG data has been extensively studied in the Machine Learning literature, our proposed application of estimating distal leg muscle activations from proximal ones did not appear to have been studied in our cursory literature review. Further, as mentioned in the introduction, the proposed dataset has not been substantially studied using machine learning methods.

2 Methodologies

The following sections detail the theory and design rationale behind each of the three chosen prediction methods. First, we detail how the raw data was pre-processed, how feature vectors were

constructed, and how non-steady datapoints were handled. Then, we detail our implementation of KRR, focusing on how the formulation was modified to be used with a large dataset. Next, we discuss a similar application of SVR. Finally, we detail our implementation of a TCN. Each method was implemented in `Python` and utilized the packages `scikit-learn` to implement SVR and KRR and `TensorFlow-keras` to implement the TCN.

2.1 Data pre-processing

To prepare the data for the regression tasks, we filtered, enveloped, and segmented the raw EMG signals into strides. Using a script provided with the dataset [1], we extracted the signal envelope from the EMG raw data by rectifying and lowpass filtering (6Hz) the signal. We also identified each gait cycle for each activity, and extracted 101 data points (0 to 100%, called phase points) per cycle. Then, we normalized each muscle's EMG using the following formula:

$$EMG_{ij} = \frac{rEMG_{ij}}{AvgTr_{ij}} \tag{2}$$

where $rEMG_{ij}$ corresponds to the EMG data after extracting the EMG envelope of muscle i on subject j, and $AvgTr_{ij}$ corresponds to the average enveloped EMG of muscle i of subject j while walking on the treadmill at 1.35 m/s. This normalization was the recommended method by [1] and allows for the comparison between activations of the same muscle on different subjects. We extracted the data from 21 different subjects that walked on a treadmill at 28 different speeds, each held for at least 30 seconds. In total, $n \approx 1.8$ million time instances were used as input for the different algorithms. Table 1 shows the descriptive statistics of every predictor (8) and predictand (3) muscle.

	Normalized Muscles										
	VM	VL	RF	BF	ST	GR	GM	REO	**GM**	**TA**	**SOL**
Mean	1.02	0.99	1.05	1.06	1.08	1.05	1.04	1.02	1.01	1.03	1.00
Std Dev	1.06	1.00	0.88	1.18	1.18	1.08	0.84	0.52	1.30	0.82	1.08
Max	20.73	16.85	35.25	17.87	23.84	17.02	21.79	13.49	19.14	12.88	16.62
Min	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

Table 1: Descriptive statistics for normalized muscles across all subjects during treadmill walking. The muscles used as predictors are the following: VM=Vastus Medialis, VL=Vastus Lateralis, RF=Rectus Femoris, BF=Biceps Femoris, ST=Semi Tendinosus, GR=Gracilis, GlM=Gluteus Medius and REO=Right External Oblique. The muscles used as predictands are GA=Gastrocnemius, TA=Tibialis Anterioris and SOL=Soleus.

The final step in the pre-processing building feature vectors in a way appropriate for every algorithm. Since muscle activation is a dynamic process [18], including history for the algorithms to predict the next activation is essential. Thus, the input to the algorithms included the present time point and some window into the past of size h to generate the prediction. The time points without h consecutive prior time points were excluded. This was the case on the first few phase points of the first gait cycle on each activity, as well as some other uncommon cases where two strides were not consecutive in the data. For KRR and SVR we used $h = 50$ for each of the eight muscles used as predictors and we also included the present speed and gait phase as relevant features. The input to both algorithms was a matrix $X \in \mathbb{R}^{((h+1)\cdot 8+2)\times n}$, where $(h+1)\cdot 8+2$ represents the present and h past time points for each of the 8 muscles, and the two added features that are speed and gait cycle. For our TCN network, we decided to decrease the complexity of the model by reducing the history to $h = 25$. We found that the model was still able to predict muscle activation well using this shorter history. Speed and phase inputs were also excluded, as the model performed well without them. Therefore, the input for the TCN algorithm was a tensor $X \in \mathbb{R}^{n\times(h+1)\times 8}$.

2.2 Kernel Ridge Regression (KRR)

A preliminary predictor using Kernel Ridge Regression (KRR) was developed as a simple, baseline method to compare other results against. Recall that KRR is similar to standard ridge regression in that it minimizes a cost function that includes a mean squared error term and a ridge penalty term. However, a nonlinear feature map function $\Phi(x) : \mathbb{R}^d \to \mathbb{R}^k$ is used to map the feature vectors to a higher dimensional space where they are more easily linearly separated. Our data was not well

linearly separated in its original space, so a radial basis function kernel (rbf) was used. A standard rbf scaling parameter of $\gamma = 1/d$ was used. By allowing for nonlinear transformations of the features prior to linear separation, the KRR optimization problem becomes

$$\underset{w \in \mathbb{R}^k, b \in \mathbb{R}}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^{n}(y_i - w^\top \Phi(x_i) + b)^2 + \lambda||w||^2. \tag{3}$$

We can write a kernel function $k(x, x') =< \Phi(x), \Phi(x') >$ and leverage it to write the closed form solution for a predicted value $\hat{y}$ for a test feature instance x without ever actually calculating the feature map $\Phi(x)$, given by

$$\hat{y} = \bar{y} + \tilde{y}^\top (\tilde{K} + n\lambda I)^{-1}\tilde{k}(x) \tag{4}$$

where $\tilde{y}$ and $\bar{y}$ are the centered and average response vectors of the training data, respectively, and the kernel matrix $\tilde{K}$ and kernel function $\tilde{k}(x)$ are defined in Appendix A.

A standard division of the dataset into 80% training data and 20% testing data would result in $n \approx 1.43$ million training data points. Solving the closed form solution given in (4) requires $O(n^3)$ operations and $O(n^2)$ memory due to the inversion of the $n \times n$ matrix. Therefore, calculating the closed form solution of KRR using the full dataset was not possible, as it would require over a terabyte of RAM. Therefore, two approximate approaches were tested. In the first, the closed form solution (4) was solved using a random subsampling of the dataset. For our computers, we were able to use 5% of the training data before hitting memory limitations. The performance of the resulting solution was evaluated on the remainder of the training data (95% of the dataset). Before fitting the model, the training data was sphered. This process was repeated for a range of regularization parameters for model selection, specifically $\lambda \in \{1.0, 0.1, 0.01, 0.001\}$. The fitting was performed using `sklearn`'s native KRR methods.

In the second method, a Nystroem transform was used to approximate the infinite dimension of the rbf kernel in a finite number of terms. The Nystroem transform is commonly used in large scale learning applications to create a low rank approximation of the kernel matrix [19]. The maximum dimension approximation that could be stored on our computers while still using the full training set was $d = 3750$. The transformed feature matrix $X_\Phi \in \mathbb{R}^{3750 \times 1.43\text{m}}$ is still too large to be used in the closed form solution, as inverting $X_\phi^\top X_\phi$ would still be intractable. Therefore, after sphereing the transformed features, Stochastic Gradient Descent (SGD) was used to solve the standard linear ridge regression problem. This was implemented using the SGD option in `sklearn.Ridge`. The same range of ridge penalties were tested as in Method 1, but scaled by the increased length of the feature vectors, resulting in approximately division by 10.

2.3 Support Vector Regression (SVR)

In addition to KRR methods, Support Vector Regression is an attractive alternative because the predictor is sparse, i.e. only the support vectors need to be stored in memory. As $\epsilon \to 0$, KRR and SVR solve the same optimization problem, but SVR minimizes the l_1 norm of the prediction error, rather than the l_2 norm used in KRR. This provides the attractive feature of a sparse model. The optimization problem for linear SVR, similar to the optimal soft-margin hyperplane, is

$$\begin{aligned} \underset{w, b, \xi_i^+, \xi_i^-}{\text{minimize}} \quad & \frac{1}{2}||w||_2^2 + \frac{C}{n}\sum_{i=1}^{n}(\xi^+ + \xi^-) \\ \text{subject to} \quad & y_i - w^T x_i - b \leq \epsilon + \xi^+ \quad \forall i, \\ & w^T x_i + b - y_i \leq \epsilon - \xi^- \quad \forall i, \\ & \xi^+ \geq 0 \quad \forall i, \\ & \xi^- \geq 0 \quad \forall i \end{aligned} \tag{5}$$

However, for our purpose we utilized non-linear SVR with an rbf kernel. To find a solution for a predicted value $\hat{Y}_n$ for a test feature instance x_n we first find the dual Lagrangian form of the above objective function. The solution is given by

$$\hat{Y}_n = \sum_{n=1}^{N}(a_n - a_n^*)k(x_n, x) + b \tag{6}$$

where $k(x_n, x)$ is the rbf kernel function, a_n and a_n^* are non-negative multipliers solved for by minimizing the dual form, b is the intercept which can be solved for with the non-negative multipliers after minimization of the dual. The dual was solved using gradient descent as a quadratic program.

Because the cost function of SVR ignores about data points outside of the margin, the number of operations required to form a solution, comparing SVR to KRR, from $O(n^3)$ to $O(dn^2)$, where d denotes the feature vector length. However, similar to KRR, an approximate solution was used due to hardware limitations. We were able to use 5% of the data for training before hitting memory limits. This process was repeated for a range of regularization parameters $C \in 0.1, 1.0, 10$ and performed using `sklearn`'s native SVR methods. It is possible to use a Nystreom tranform to approximate the rbf kernel for our regression, but we chose not to take this approach after seeing little to no benefit with this approach with KRR Regression (see Table 2).

2.4 Temporal Convolutional Network (TCN)

A TCN is a 1D convolutional network that provides strong predictions for sequential data [11, 12]. Each "layer" consists of a residual block, which is essentially a set of deeper layers. In Figure 1 provides a detailed illustration an example TCN architecture and its components. The dilation factor d allows the model to reach far back in time at the expense of skipping some data points. Another key feature of TCN is that the convolution it performs is causal, meaning that the output of one operation only depends on present and past, and never on the future. Since our goal is to develop a method capable of predicting muscle activation in real time, a causal convolution is necessary.

For our network, we used the TCN library developed by Remy [20] which was based in the model presented by Bai et al. [11]. Figure 2 shows our chosen network architecture. We designed it as one TCN layer, one hidden layer and one output layer. The input to the TCN layer corresponds to a matrix $x \in \mathbb{R}^{(h+1)\times 8}$, where the 8 columns represent the different predictor muscles and the $h+1$ rows go from the present to h time points in the past. The output of the TCN layer corresponds to a 128-dimension vector. This vector represents a hidden embedding of the input data. Inspired by the architecture in [21], we added a fully-connected layer with a ReLU activation after the TCN, which helps increase the size of the model and gives it non-linear representation power. Finally, the output layer represents the prediction given for each of the three output muscles. We added a ReLU activation function to the output layer as well, to capture the fact that muscle activation cannot be negative. The parameters chosen for the final model are described in Table 6 in Appendix B, and the decisions were based in multiple trials with smaller subsets of the data and recommendations from [21, 11].

We realized from initial experiments that the TCN network had a better prediction power than KRR and SVR and so we extended the training data to include levelground and ramp ascent walking, together with the original treadmill data (total $n = n_t + n_l + n_r$, where $n_t = 1,801,586, n_l = 254,801, n_r = 248,055$). We randomly divided the joint data into three groups: 70% of the data for training, 10% of the data for validation and 20% of the data for testing. The training set was used for model fitting and the validation set was used to determine whether the model was overfitting to the data or not. We trained the model for 60 epochs using MSE loss and the "adam" optimizer [22]. The training and validation loss during training are shown in Appendix B, Figure 5.

3 Evaluation

3.1 Metrics

The models were evaluated using both mean squared error (MSE) and Variance Accounted For (VAF):

$$\text{VAF} = 1 - \frac{\text{var}(y - \hat{y})}{\text{var}(y)}. \tag{7}$$

VAF is a measure of how well the model explains the variance observed in the training data, and it is commonly used in biomechanics literature [24]. A perfect model would show both VAF = 100% and MSE = 0. Each method was tested using at least 20% of the dataset as test samples (360,000). For the KRR and SVR methods which could not use the full 80% of the data for training, some tests were evaluated using all leftover data (more than 20%). However, it was observed that the metrics for the

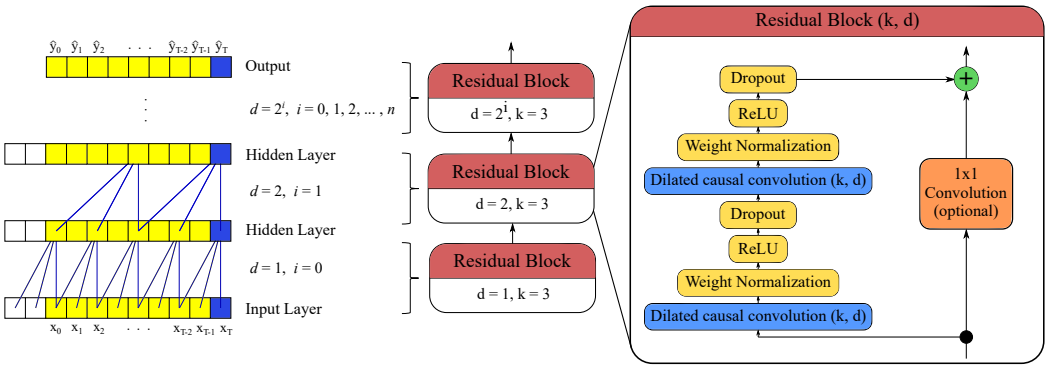

Figure 1: An example of TCN architecture, reproduced from figures created by Bai et al. [11] and Moor et al. [23]. The left-hand side describes the overall architecture of the TCN, with T time points used for the input and n hidden layers. The middle and right-hand sides describe how the residual blocks make up the layers of the TCN. The 1x1 convolution of a residual block is added when the input and output dimesions of the residual blocks have different dimensions [11].

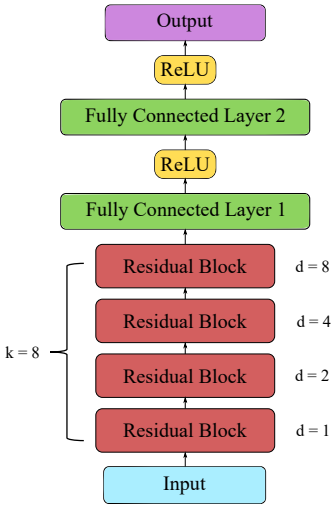

Figure 2: Our TCN model used for EMG prediction. The first FCL has an output of 64. The second FCL has an output of 3 to match the number of muscles for which we are finding EMG predictions.

full remainder of the data were not significantly different than when evaluating on the original 20% test set. Therefore, for time efficiency, the remainder were evaluated using only the original test set.

3.2 Results

Tables 2 and 3 show the testing metrics for the various KRR and SVR models and hyperparameters. We notice that even when training on a very small subset of the data, the closed form KRR solution outperforms the approximated method. This suggests that the high dimension and nonlinear behavior of the rbf kernel is critical to linearly separating the data, as even with a much higher training fraction the approximate method is unable to outperform the exact method. With the SVR model we saw a general trend across all muscles of increased performance with increased regularizer order. Similar to KRR, the SOL muscle had the largest VAF across all regularizers.

Table 4 shows the performance of the TCN model in individual tasks and for all tasks combined. Note that the model had the lowest MSE on the treadmill data, which can be explained by the treadmill data accounting for almost 80% of the total training data. The VAF, on the other hand, showed its best results on the ramp ascent activity and was lowest for the TA muscle, similarly to the previous models.

One interpretation of this result is that ramp ascent EMG has less variance than other activities, or that the patterns are easier to infer.

KRR Type	Regularizer	Train Pct.	Testing MSE			Testing VAF (%)		
			GA	*TA*	*SOL*	*GA*	*TA*	*SOL*
Closed Form	$\lambda = 1 \times 10^{-3}$	5	0.4135	0.1776	0.1911	73.63	71.77	77.84
Closed Form	$\lambda = 1 \times 10^{-2}$	5	**0.2560**	0.1431	0.1609	**75.90**	**75.01**	**84.15**
Closed Form	$\lambda = 1 \times 10^{-1}$	5	0.2641	**0.1378**	**0.1561**	72.47	74.13	83.89
Closed Form	$\lambda = 1 \times 10^{0}$	5	0.3089	0.1567	0.1814	62.97	67.42	80.10
Nystroem Approx.	$\lambda = 1 \times 10^{-4}$	80	0.3405	0.1715	0.1915	73.68	65.08	80.27
Nystroem Approx.	$\lambda = 1 \times 10^{-3}$	80	0.3354	0.1638	0.1955	74.39	65.20	79.29
Nystroem Approx.	$\lambda = 1 \times 10^{-2}$	80	0.3369	0.1750	0.1917	74.39	65.25	79.49
Nystroem Approx.	$\lambda = 1 \times 10^{-1}$	80	0.3442	0.1672	0.1985	74.26	65.48	78.69

Table 2: Predictor performance for both the standard closed form KRR method using 5% of the data for training and the approximated KRR method with a Nystroem transform using 80% of the data for training. Models with the highest performance on the testing data are noted in **bold**.

Regularizer	Train Pct.	Testing MSE			Testing VAF (%)		
		GA	*TA*	*SOL*	*GA*	*TA*	*SOL*
$C = 1 \times 10^{-1}$	5	0.6561	0.1899	0.1417	15.22	56.51	71.53
$C = 1 \times 10^{0}$	5	0.4689	**0.1519**	**0.1678**	54.00	**70.08**	**81.49**
$C = 1 \times 10^{1}$	5	**0.3509**	0.1789	0.2067	**72.76**	66.15	77.86

Table 3: Predictor performance for both the standard non-linear SVR method using 5% of the data for training. Models with the highest performance on the testing data are noted in **bold**.

		Activity			
		Treadmill	Levelground	Ramp Ascent	**All**
MSE	GA	**0.105**	0.120	0.161	0.112
	TA	**0.087**	0.098	0.163	0.097
	SOL	**0.081**	0.087	0.161	0.090
VAF	GA	94.5%	94.1%	**97.0%**	95.2%
	TA	**88.0%**	83.9%	84.4%	87.2%
	SOL	94.5%	93.3%	**96.0%**	94.4%

Table 4: TCN performance metrics on each of the different activities as well as aggregate. Activities and muscles with the best performance are shown in **bold**.

3.3 Example Real-time Test

Figure 3 shows example time series predictions for consecutive example strides from subject AB10 using each of the three methods. For methods where multiple hyperparameters were tested, the models with the best average testing metrics were used (KRR: closed form, $\lambda = 0.01$, SVR: $C = 10$). The MSE and VAF for these specific trials are shown in Table 5. Note that VAF is not meaningful when the time series is short, so MSE in this case provides a more accurate comparison. Further, the average prediction time for each model was estimated. On average, KRR required 2.3 ms per prediction, SVR required VAL ms, and the TCN required 2.6 ms. Note, though, that the TCN predicts all three muscle activations at the same time and so this model is much faster than the other two.

4 Discussion and Conclusions

In this report, we presented three methods to predict three distal leg muscles' activation from eight proximal leg muscles. We used Kernel Ridge Regression, Support Vector Regression, and a neural network based on Temporal Convolutional Networks. The best results came from the TCN model, achieving an average VAF of over 92%. All models were able to predict muscle activations with

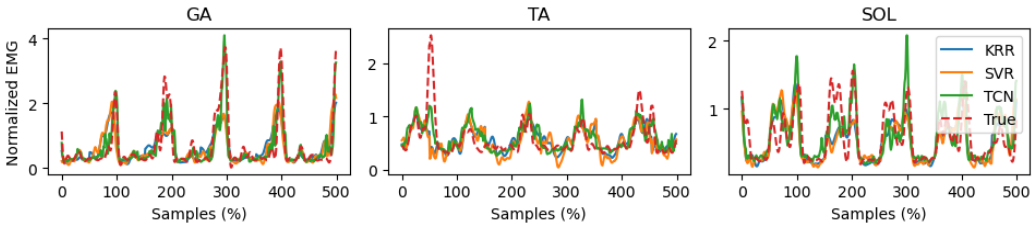

Figure 3: Plots of the predicted values for the *GA* (left), *TA* (middle) and *SOL* (right) muscles using each of the three methods tested for the first 5 strides of treadmill data for subject AB10. Note that the TCN model more closely resembles the true measured values in all 3 muscles.

Method	Prediction Time (ms)	**MSE**			**VAF (%)**		
		GA	*TA*	*SOL*	*GA*	*TA*	*SOL*
KRR	$\sim$ 2 ms	0.2635	0.0997	0.0836	4.75	-95.12	15.41
SVR	$\sim$ 3 ms	0.3263	0.1262	0.1012	-33.35	-82.09	-2.44
TCN	$\sim$ 3 ms (all muscles)	0.0977	0.0600	0.0738	84.19	49.40	43.47

Table 5: Performance metrics for each prediction method when applied sequential prediction of the first 5 consecutive strides of subject AB10 during treadmill walking. Note that MSE is more meaningful than VAF for short time series such as these, as VAF can produce non-descriptive (negative) values when the model predicts more variance than the actual signal. All prediction times are reasonable for use in real-time control.

reasonable accuracy (mean MSE well below 30% of the average signal) across different subjects. We believe that a key factor that helped improve this prediction was the normalization step during pre-processing, which has been shown to allow a better comparison of muscle activation between subjects [25]. Further, the TCN model was surprisingly effective at predicting activation across different different activities. We believe that the time dependent nature of the TCN was able to better learn the dynamics of muscle activations and generalize it across tasks.

One of the goals of this project was to be able to develop an algorithm that could be used for real-time prediction of the distal leg muscles. In Table 5, we show the average prediction time for each algorithm when run on a desktop computer. More extensive tests are necessary to determine the exact time it would take on a standard embedded system for prosthesis control, but this gives a rough estimate of the order of magnitude. The millisecond predictions we found suggest that the predictions could be made at a minimum of 100 Hz, which is around the frequency at which most prosthesis control systems run. Therefore, these results show that the proposed methods could certainly be used for real-time control.

It is also interesting to note that across all methods, the *TA* prediction was worse by approximately 10% VAF than the other two predictor muscles. We hypothesize that this is because the TA is mainly active at two points in the gait cycle, right at heelstrike and during midswing to initiate ankle dorsiflexion [26]. At these points in the gait cycle, many of the other muscles that we use in the feature vector have low activation. Particularly during midswing, the only other muscle with significant activation is the rectus femoris, and it is still at a much lower activation. Therefore, we attribute this decreased prediction ability a lower signal to noise ratio for between the *TA* activation and the features. Perhaps in future work, it would be beneficial to identify another muscle that correlates more with the *TA*.

Another takeaway from this project was that the KRR and SVR methods scale poorly to large amounts of data both in training time and memory requirements. Both generate huge matrices of data during the training process, which require a lot of memory to run successfully. It proved impossible with our resources to run those algorithms with a significant part of the training set, and instead we had to use a very small part of it, or resort to kernel approximation methods. It was surprising that even with that low amount of data, we still achieved over 60% VAF for all muscles in the testing set.

5 Group Member Contributions

The group member contributions roughly followed the distribution outlined in the project proposal. Author 1 wrote the primary code for normalizing and arranging the data into suitable feature vectors. Author 2 wrote the code to implement KRR and Author 3 wrote the code to implement SVR. Author 1 and Author 4 worked jointly to write the code for the TCN. All group members participated in calls and discussions regarding model design and debugging for all methods, which took place either on Zoom, in person, or on Slack. The specific sections in the report regarding each method were written by the code authors of the respective methods. All authors contributed to revising, editing, and verifying these sections. All authors also worked jointly on the abstract, introduction, discussion, and conclusion sections.

References

[1] J. Camargo, A. Ramanathan, W. Flanagan, and A. Young, "A comprehensive, open-source dataset of lower limb biomechanics in multiple conditions of stairs, ramps, and level-ground ambulation and transitions," *Journal of Biomechanics*, vol. 119, p. 110320, 2021. [Online]. Available: https://doi.org/10.1016/j.jbiomech.2021.110320

[2] S. A. Chvatal and L. H. Ting, "Common muscle synergies for balance and walking," *Frontiers in Computational Neuroscience*, 2013.

[3] B. Chen and Q. Wang, "Combining human volitional control with intrinsic controller on robotic prosthesis: A case study on adaptive slope walking," vol. 2015-November, 2015.

[4] D. Lee, I. Kang, D. D. Molinaro, A. Yu, and A. J. Young, "Real-time user-independent slope prediction using deep learning for modulation of robotic knee exoskeleton assistance," *IEEE Robotics and Automation Letters*, vol. 6, pp. 3995–4000, 2021.

[5] R. Gupta and R. Agarwal, "Lower-limb muscle emg analysis to predict ankle-foot activities for prosthesis control," in *Smart Computing: Proceedings of the 1st International Conference on Smart Machine Intelligence and Real-Time Computing (SmartCom 2020), 26-27 June 2020, Pauri, Garhwal, Uttarakhand, India*. CRC Press, 2021, p. 404.

[6] R. A. Zanini, E. L. Colombini, and M. C. F. D. Castro, "Parkinson's disease emg signal prediction using neural networks," vol. 2019-October, 2019.

[7] M. F. Rabbi, C. Pizzolato, D. G. Lloyd, C. P. Carty, D. Devaprakash, and L. E. Diamond, "Non-negative matrix factorisation is the most appropriate method for extraction of muscle synergies in walking and running," *Scientific Reports*, vol. 10, 2020.

[8] A. Saito, A. Tomita, R. Ando, K. Watanabe, and H. Akima, "Muscle synergies are consistent across level and uphill treadmill running," *Scientific Reports*, vol. 8, 2018.

[9] Y. Kim, S. Stapornchaisit, M. Miyakoshi, N. Yoshimura, and Y. Koike, "The effect of ica and non-negative matrix factorization analysis for emg signals recorded from multi-channel emg sensors," *Frontiers in Neuroscience*, vol. 14, p. 1254, 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2020.600804

[10] Y. Ma, C. Shi, J. Xu, S. Ye, H. Zhou, and G. Zuo, "A Novel Muscle Synergy Extraction Method Used for Motor Function Evaluation of Stroke Patients: A Pilot Study," *Sensors*, vol. 21, no. 11, p. 3833, Jun. 2021.

[11] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[12] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 156–165.

[13] P. Tsinganos, B. Cornelis, J. Cornelis, B. Jansen, and A. Skodras, "Improved gesture recognition based on semg signals and tcn," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 1169–1173.

[14] M. Zanghieri, S. Benatti, A. Burrello, V. Kartsch, F. Conti, and L. Benini, "Robust real-time embedded emg recognition framework using temporal convolutional networks on a multicore iot processor," *IEEE transactions on biomedical circuits and systems*, vol. 14, no. 2, pp. 244–256, 2019.

[15] J. L. Betthauser, J. T. Krall, S. G. Bannowsky, G. Lévay, R. R. Kaliki, M. S. Fifer, and N. V. Thakor, "Stable responsive emg sequence prediction and adaptive reinforcement with temporal convolutional networks," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 6, pp. 1707–1717, 2019.

[16] P. Hewage, A. Behera, M. Trovati, E. Pereira, M. Ghahremani, F. Palmieri, and Y. Liu, "Temporal convolutional neural (tcn) network for an effective weather forecasting using time-series data from the local weather station," *Soft Computing*, vol. 24, no. 21, pp. 16 453–16 482, 2020.

[17] B. Xu, C. Lu, Y. Guo, and J. Wang, "Discriminative multi-modality speech recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 433–14 442.

[18] F. E. Zajac, "Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control." *Critical reviews in biomedical engineering*, vol. 17, 1989.

[19] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling techniques for the nystrom method," in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, D. van Dyk and M. Welling, Eds., vol. 5. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 16–18 Apr 2009, pp. 304–311. [Online]. Available: https://proceedings.mlr.press/v5/kumar09a.html

[20] P. Remy, "Temporal convolutional networks for keras," https://github.com/philipperemy/keras-tcn, 2020.

[21] M. S. Willsey, S. R. Nason, S. R. Ensel, H. Temmar, M. J. Mender, J. T. Costello, P. G. Patil, and C. A. Chestek, "Real-time brain-machine interface achieves high-velocity prosthetic finger movements using a biologically-inspired neural network decoder," *bioRxiv*, 2021.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] M. Moor, M. Horn, B. Rieck, D. Roqueiro, and K. Borgwardt, "Early recognition of sepsis with gaussian process temporal convolutional networks and dynamic time warping," in *Machine Learning for Healthcare Conference*. PMLR, 2019, pp. 2–26.

[24] R. E. Kearney, "System Identification of Human Joint Dynamics_Kearney 1990.pdf," 2014.

[25] M. Halaki and K. Ginn, "Normalization of emg signals: to normalize or not to normalize and what to normalize to," *Computational intelligence in electromyography analysis-a perspective on current applications and future challenges*, pp. 175–194, 2012.

[26] G. Lichtwark, "The role of the tibialis anterior muscle and tendon in absorbing energy during walking," *Journal of Science and Medicine in Sport*, vol. 18, 2014.

A Additional Equations

Using the notation from the lecture notes, the kernel matrix and kernel vector are defined as:

$$[\tilde{K}]_{ij} = k(x_i, x_j) - \frac{1}{n}\left(\sum_{r=1}^{n} k(x_i, x_r) + \sum_{s=1}^{n} k(x_s, x_j)\right) + \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} k(x_r, x_s), \tag{8}$$

$$[\tilde{k}(x)]_i = k(x_i, x) - \frac{1}{n}\left(\sum_{r=1}^{n} k(x_i, x_r) + \sum_{s=1}^{n} k(x, x_s)\right) + \frac{1}{n^2}\sum_{r=1}^{n}\sum_{s=1}^{n} k(x_r, x_s). \tag{9}$$

B Supplemental Figures and Tables

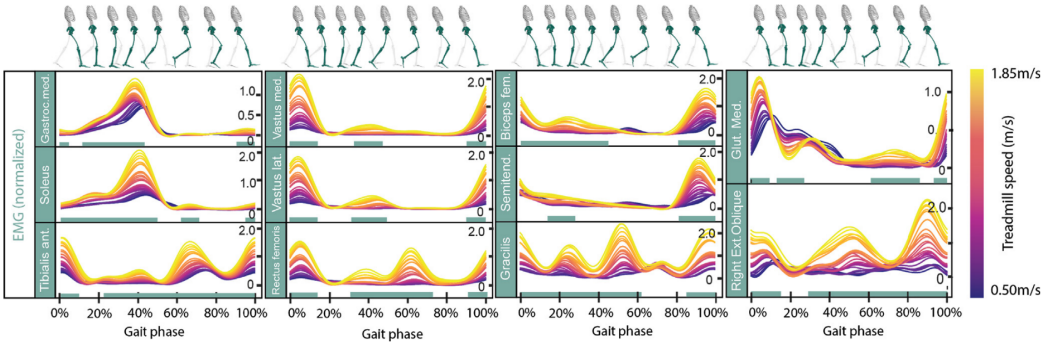

Figure 4: Plots of the average rectified EMG signals for each of the 11 muscles in the dataset, reproduced from [1]. The goal of this system is to predict the three plots on the left using measurements of the remaining muscles.

Model Section	Parameter	Value
TCN	number of filters	128
	d dilations	(1, 2, 4, 8)
	k kernel size	8
	kernel initializer	glorot uniform
	use batch normaliztion	True
	loss function	MSE
	dropout rate	0
	number of stacks	1
FCL 1	output size	64
	bias initialization	0
	activation	ReLU
FCL 2	output size	3
	bias initialization	0
	activation	ReLU

Table 6: Model parameters used for the TCN and Fully Connected Layers (FCL).

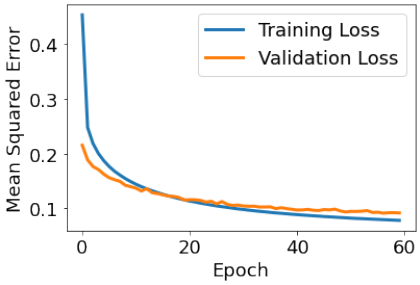

Figure 5: Training and validation loss for the TCN model fitting. Note that validation loss does not start increasing after even 60 epochs, suggesting that the model does not overfit to the training data.