Enhancing sEMG-based Movement Recognition Using Data Augmentation: An Empirical Study

Anonymous Authors

***Abstract*—Surface Electromyography (sEMG) is a non-invasive technique which may be leveraged in prostheses to improve quality of life for amputees. To address the significant huddle of limited sEMG data availability for the success of machine learning, this study provides an empirical evaluation of random transformation, pattern mixing and generative time series data augmentation techniques applied to sEMG signals of individual finger movements of four test subjects. We evaluate augmentation methods using two state of the art deep neural networks and a classical machine learning approach, targeting improved classification accuracy in three models: LSTM, TCN and XGB. We search for the most effective model, augmentation ratio and augmentation method. In addition to this, we analyze the impact of data availability on augmentation techniques by limiting the amount of data used training. Our results suggest a classical machine learning model, XGB, is the most effective approach to modeling individual finger movements, independent of the amount of data available during training. The results, too, suggest a naïve, random transformation augmentation technique is generally more effective than more complicated approaches such as generative and pattern mixing.**

***Index Terms*—Data augmentation, movement recognition, sEMG signals, time series.**

I. Introduction

A 2005 study found that 1.6 million persons were living with a missing limb in the United States and projected that number to more than double by 2050 [1]. There are 50,000 to 100,000 new amputations per year, and of those, approximately 7% (3,500 to 7,000) occur at the hand or wrist [2]. For those afflicted with such a disability, robotic prostheses can be of substantial value. For example, control of the prosthesis can be provided to the user using surface Electromygraphy (sEMG) sensors. sEMG is a non-invasive technique to acquire myoelectric signals [3]. These signals capture a user's intent to contract a part of the body (e.g. finger, toe, hand). Studies have found myoelectric powered prostheses to be rejected by amputees at rates ranging from 0% to 75% with complaints including excessive weight, comfort, lack of function and durability [4].

Myoelectric prostheses which are more functional are required to lower the rejection rates of these devices and improve quality of life for amputees. A critical step in facilitating such innovations is developing methods to recognize individual finger movements. However, to date, the majority of hand sEMG classification research has focused on hand gesture or multi-finger movement recognition [5], [6]. Moreover, many robotic prostheses [7]–[9] rely on conventional sEMG sensors with flat electrode pads which are held in place using a band [10] or adhesive [11], but tend to be uncomfortable or poorly affixed to the skin [12]. Such configurations, however, are susceptible to electrode shift which may affect the signals acquired by each electrode [13] and, thus, adversely affect control of the prosthesis. Therefore, in this work, we focus on individual finger movement recognition based on nano-sEMG signals collected from subjects' antebrachium (forearm) as shown in [12], with a sensor that may be directly embedded onto the user's skin and is held in place using only van der Waals forces [12], which eliminates the potential of electrode shift and any potential discomfort of bands or adhesive materials. In addition to this, the sensor we study can withstand daily activities such as bathing and exercising [12], increasing the practicality of integrating a prosthesis into the daily life of a user.

To automate the finger movement recognition in real time, machine learning techniques can be adopted. It is well known that machine learning algorithms often require sufficiently large training data to successfully generalize to the target population. One major challenge in individual finger movement recognition using nano-sEMG is limited training data, due to the fact that collecting sEMG data requires expertise, human supervision, and a considerable amount of time.

Data augmentation is a technique that has yielded great performance gains in, most notably, computer vision [14]. Time series data augmentation, on the other hand, has received significantly less attention. A recent survey [14] proposed a taxonomy of time series data augmentation techniques with four (4) distinct approaches: random transformation, pattern mixing, generative models, and decomposition. Ordinary time series data augmentation involves randomly transforming the signals. These transformations include Gaussian noise injection, time warping, frequency warping, magnitude warping, scaling, permutation, cropping, and flipping or rotating signals (e.g. [15], [16]). Pattern mixing techniques produce a synthetic sample by combining two or more samples (e.g. [17], [18]). Generative models augment datasets by extracting features from samples and using those features to construct new samples (e.g. [19]–[25]). Finally, decomposition methods decompose samples into features which are used to create new samples (e.g. [6], [26]). Furthermore, [14] provides an evaluation of random transformation and pattern mixing techniques on an array of time series datasets using several well known neural network architectures. However, this survey does not evaluate an sEMG dataset or a GAN, the models used are not tuned for each dataset, and the accuracy improvements presented are limited using the evaluated augmentation techniques. Our study focuses exclusively on an sEMG dataset, evaluating a DCGAN [20] on said data, tuning each of our models to our dataset to maximize classification accuracy, and demonstrating considerable performance improvements using augmentation. Additionally, while [14] considered only artificial neural networks (ANNs), we evaluate our dataset on two state of the art ANNs and a classical machine learning model, and find the latter to be the most performant.

Several works have evaluated the performance of augmentation techniques on sEMG data. sEMG simulation models have been introduced in [27]–[29]. Problems with these models include: 1) the signals they generate depend upon many domain-specific variables, which requires a solid understanding of their meaning [20], and 2) the validation of said models is challenging [30]. The effects of Gaussian noise applied to sEMG hand gesture movements [31] was evaluated on a simple convolution neural network (CNN) in [32]. Two generative adversarial networks (GANs) have also been proposed for sEMG data. In [20] a DCGAN with Style Transfer is used to augment a Parkinson's Disease dataset by transforming sEMG samples collected from healthy subjects into data closely resembling those of the Parkinson's data. More recently, another GAN variant was proposed in [21]. Furthermore, a thorough survey and evaluation of time series data augmentation for deep neural networks is available in [14]. In [6] wavelet decomposition, two simulation models, a random transformation method, and three (3) random combination methods are evaluated on two CNNs. To date there is no work which provides a thorough analysis of current state of the art data augmentation techniques on an sEMG dataset.

Random Transformation	Generative Model	Pattern Mixing
Gaussian Noise	DCGAN [20]	SPAWNER [17]
Window Slicing	-	-

TABLE I: Summary of data augmentation techniques, organized by taxonomy family (see [14]), evaluated in this study.

Deep Model	Classical Model
LSTM [33], [34]	XGB [35]
TCN [36]	-

TABLE II: Summary of architectures evaluated on our sEMG dataset in this study.

We propose a comprehensive comparison study of current state of the art time series augmentation methods for on sEMG data of individual finger movements for real-time applications. We are interested in studying the impact augmentation has on classification accuracy and how the amount of data available affects the gains yielded by augmentation. To this end, we evaluate a subset of augmentation techniques from each taxonomy family, with exception to decomposition. The selected augmentation techniques may be found in table I. Each method was selected because either it showed promising results in previous studies (e.g. SPAWNER, DCGAN) or is a typical method applied in sEMG/time series (e.g. Gaussian Noise, Window Slicing). The models we chose to evaluate each augmentation technique may be found in table II. Each of these models are state of the art.

II. Augmentation Methods

In this section, we formally describe each of the data augmentation methods we evaluate.

A. General Augmentation Algorithm Description

The input to each augmentation algorithm, the sEMG data of an individual finger movement, may be formalized as a sequence of T time steps, each of which containing F features. Note, because we record our data with a single channel, the number of features per timestep F is one (1). Specifically, we denote an individual sample as $x = x_0, x_1, \cdots, x_{t-1}$ where x_t is a vector containing the features recorded at timestep t. Each augmentation method accepts an individual finger movement x as an input and outputs a synthetic sample with the same shape as x. More formally, the output of each augmentation algorithm is $\hat{x} = \hat{x}_0, \hat{x}_1, \cdots, \hat{x}_{t-1}$ where $\hat{x}_t$ is a vector containing the features of the synthetic sample at timestep t containing the same number of features as the input x_t.

B. Gaussian Noise

Gaussian noise is a common way to augment signal processing data sets [6], [14]–[16], [32]. Another common name for this augmentation technique is 'jitter' or 'jittering' [14]. The idea is to inject random samples from a Gaussian (Normal) distribution into an authentic sample to produce a unique synthetic sample. The implicit assumption here is that signal noise follows the Gaussian distribution: $N(\mu, \sigma)$. We choose $\mu = 0$ to minimize deviation from the original sample. σ is slightly more challenging to define because nominal values will yield synthetic samples which closely relate to the originals (and thus a model which overfits the training set). In contrast, larger values will increase the signal to noise ratio to obfuscate the interesting parts of the signal (and thus an underfitting model). Our model of Gaussian noise is as follows:

$$\mathcal{N}\left(\mu = 0, \sigma = \sqrt{\frac{x_i^2}{SNR}}\right)$$

where x_i is the ith timestep of the sample for which we produce a synthetic sample and SNR is the signal to noise ratio. Signal to noise ratio will be an explored free variable in our experiments.

We generate a synthetic for this method following

$$\hat{x} = x_0 + n_0, x_1 + n_1, \cdots, x_{t-1} + n_{t-1}$$

where x is the original finger movement sample and n is a vector containing T random samples from the aforementioned Gaussian distribution.

C. Window Slicing

Window Slicing or slicing is an augmentation technique which generates a synthetic by cropping a window W of consecutive time steps of a signal [37]. Specifically, a synthetic sample $\hat{x}$ is produced from x by

$$\hat{x} = x_\omega, x_{\omega+1}, ..., x_{\omega+W-1}$$

where $\omega \in \mathbb{N}$ such that $0 \leq \omega \leq W - T - 1$, T is the number of time steps in x, and $W \leq T$. Note when $W = T$ the synthetic sample is identical to the original. For our experiments, we use a ninety percent (90%) window and interpolate the resulting synthetic such that it has the same number of time steps as the original sample.

D. SPAWNER

SPAWNER aligns two signals $X_0 = [x_0^1, x_0^2, \cdots, x_0^n], X_1 = [x_1^1, x_1^2, \cdots, x_1^m]$ via Dynamic Time Warping (DTW) [38]. DTW computes the minimal alignment path between X_0 and X_1 or the *warping path*. The *warping path* is found by computing the element-wise cost matrix C using some arbitrary distance metric (e.g., Euclidean Distance). We use L1 loss as

$$C[i,j] = |x_0^i - x_1^j| + min(C[i-1,j], C[i,j-1], C[i-1,j-1])$$

DTW assumes the first points and last points of both sequences are aligned and yields warping paths with monotonically increasing indices. To reduce computational overhead, a warping window ξ is used to limit the temporal advance and temporal delay of the alignment. We use 10 percent of the longest sequence (i.e., $\xi = \lceil max(n,m)/10 \rceil$). Note that in our work, all sequences have the same length by front padding zeros to samples where necessary. In addition to the assumptions made by DTW, SPAWNER assumes a randomly selected point (excluding end points) from X_0 and X_1 are aligned. More formally, SPAWNER forces the warping path to contain $C_p = (R_0, R_1)$ where $R_0 = \lceil rn \rceil$, $R_1 = \lceil rm \rceil$ and r is uniformly distributed random number between 0 and 1. This is accomplished by computing two alignments. First, $X_0^0 = [x_0^1, x_0^2, \cdots, x_0^{n-R_0}]$ is aligned with $X_1^0 = [x_1^1, x_1^2, \cdots, x_1^{m-R_1}]$. Then, $X_0^1 = [x_0^{n-R_0+1}, x_0^{n-R_0+2}, \cdots, x_0^n]$ is aligned with $X_1^1 = [x_1^{m-R_1+1}, x_1^{m-R_1+2}, \cdots, x_1^m]$. The alignments yielded by X_0^0 and X_0^1 are concatenated together yielding X_0^*. Similarly, X_1^0 and X_1^1 are concatenated together yielding X_1^*. X_0^* and X_1^* are then merged using their means. Finally, we inject Gaussian Noise on top of the aligned sample using a small σ as $\mathcal{N}(\mu = 0, \sigma = 1 \times 10^{-7})$.

E. EMG-GAN

Generative Adversarial Networks (GANs) [22] are comprised of two networks, a discriminator D and a generator G. The generator produces synthetic samples by encoding the samples into a latent space z. The discriminator, on the other hand, distinguishes between real and fake samples for a dataset x. The network plays a minimax game during training

$$\min_G \max_D V(G,D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \quad (1)$$

EMG-GAN [20] adapts the Deep Convolutional GAN (DCGAN) [19], which was developed for a computer vision task, to the time series domain and we make use of their work here. The generator accepts two-hundred (200) timesteps from an original sample as input and consists of six (6) convolutional blocks. Each convolutional block begins with a one dimensional convolutional layer and is followed by a batch normalization layer before being activated by ReLU. Additionally, after the first three convolutional blocks, upsampling is performed. After the last convolutional block, a flattening layer and a fully connected hidden layer are activated with the hyperbolic tangent (tanh) function. Finally, an output layer selects the value for each time step. The architecture may be found in figure 1a.

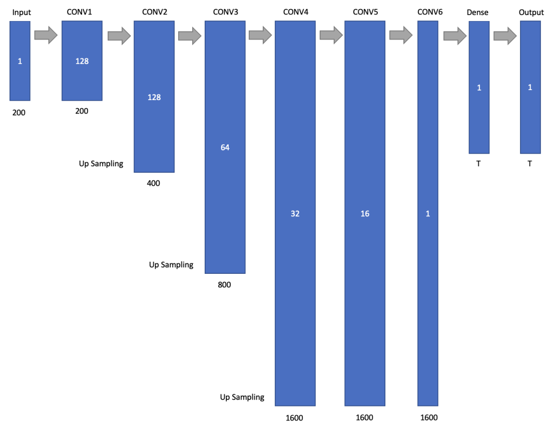
The discriminator network uses a pattern similar to the generator, however it leverages four (4) convolutional stacks. Each stack is comprised of four (4) convolutional layers, each of which having one dimensional convolutional, batch normalization and drop out layers activated by Leaky ReLU. The input to each stack is what differentiates them. The raw input signal, Fast Fourier transform (FFT), signal envelope, and a two-level discrete wavelet transformation (DWT) using the Daubechies wavelet db7 as wavelet mother are used as inputs to the stacks. Furthermore, a mini-batch discriminator [39] is used in the network to address mode collapse. The architecture may be found in figure 1b.

This technique is impractical for our real time individual finger movement identification application for two reasons. First, each EMG-GAN is trained to generate synthetic samples for a single class. Our classification task involves five (5) classes. To generate new samples for each class would thus require the training of five (5) distinct GANs. Second, training a single EMG-GAN takes a considerable amount of time. However, we evaluate this technique to compare with pattern mixing, random transformation and combination methods.

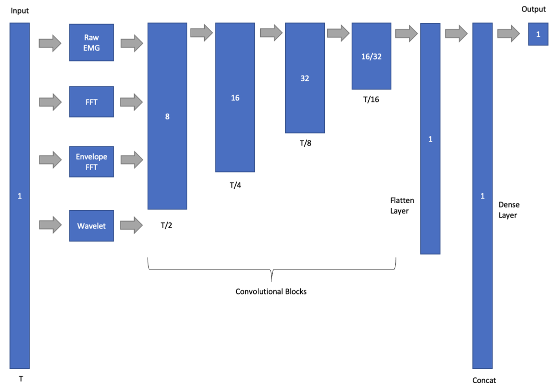
F. Random Combinations

Inspired by the work of [6] we investigate the effects of a random combination method that mixes pattern mixing and random transformation methods. Pattern mixing techniques are computationally expensive because they depend on DTW. The method constructs a sample by selecting the pattern mixing method with probability p and the random transformation technique with complement $1-p$. Note only one augmentation method is used to construct a single sample. We consider two combinations: 1) SPAWNER and Gaussian Noise 2) SPAWNER and Window Slicing.

Specifically, according to the desired ratio of augmentation γ, we aim to get a synthetic data set of size γN, where N is the total number of examples in the sEMG training data $\{X_i, y_i\}_{i=1}^N$. X_i is the i-th signal and y_i is the signal's finger class (e.g., 'pinky'). By distributing different augmentation numbers for each finger class, augmentation can also help solve the class-imbalance problem during the training. Motivated by this, we can calculate the total number of training examples after augmentation as $(\gamma+1)N$. Then, with k finger



(a) DCGAN generator architecture which accepts timesteps from a finger movement as input and applies a sequences of one dimensional convolutions, up-sampling, batch normalization and a final dense layer.



(b) DCGAN discriminator architecture extracts four features before applying four blocks consisting of four convolutional layers, batch normalization and activation layers. The last convolutional layer in each block applies 16 or 32 filters depending on the input feature. All extracted feature maps are flattened and merged prior to passing through a dense layer with sigmoid activation.

Fig. 1: Architectures of DCGAN generator 1a and discriminator 1b.

classes, the number of augmented examples for the j-th class can be easily calculated as $(\gamma+1)N/k - N_j$, where N_j is the number of original training examples of the j-th class. Note that γ should be large enough such that $(\gamma+1)N/k - N_j$ does not yield a negative for each class j.

Next, we randomly select $(\gamma+1)N/k - N_j$ samples from the corresponding class to perform data augmentation. Note when $N_j < (\gamma+1)N/k - N_j$, one or more samples will be used at least twice to achieve our target number of synthetic samples. Additionally, pattern mixing requires a reference sample for mixing as described in the above subsection (II-D). During the procedure, the reference sample is randomly obtained from the same finger class. However, we ensure that the reference sample is not the augmentation sample, and once X_i and X_j have been mixed once, they will not mix again. Most importantly, for each augmentation sample, we set a probability p to generate a synthetic sample using a random

Algorithm 1 Random Combination

```
1:  **Input:** sEMG Data set $\{X_i, y_i\}_{i=1}^{N}$, augmentation ratio $\gamma$,
    probability $p$ to use random transformation
2:  **Output:** Synthetic Data set $Syns$
3:  Initialize $Syns \leftarrow \emptyset$
4:  Initialize $sampsToAug \leftarrow \emptyset$
5:  **for** each finger class $j = 1$ to $k$ **do**
6:      $sampsToAug \leftarrow$ Randomly sample $(\gamma+1)N/k - N_j$
        from class $j$
7:  **end for**
8:  **for** each $X_i \in sampsToAug$ **do**
9:      $rand \leftarrow$ random value s.t. $0 \leq rand \leq 1$
10:     **if** $rand \leq p$ **then**
11:         $Syns \leftarrow$ RandomTransformation($X_i$)
12:     **else**
13:         Sample $X_r$ from the finger class of $X_i$
14:         $Syns \leftarrow$ PatternMixing($X_i, X_r$)
15:     **end if**
16: **end for**
17: **return** $Syns$
```

transformation method, otherwise the pattern mixing algorithm is applied. Algorithm 1 summarizes the procedure.

III. Evaluation

In this section, we introduce the models we evaluate, describe our dataset, and present the results of our study.

A. Models

1) Gradient Boosting: Gradient Boosting is an ensemble method that makes use of Decision Trees. This classifier is trained using the Boosting methodology. In Boosting, K classifiers are trained (of arbitrary type) sequentially with sample weights, where K is a hyperparameter. The sample weights are uniformly initialized. After each iteration, samples that were misclassified are given additional weights. Gradient Boosting uses gradient descent to calculate the hyperparameters of the next Decision Tree by moving along the gradient of the loss function.

We leverage XGBoost library's implementation of Gradient Boosting (proposed in [35]) with the default parameters to evaluate the dataset. Each Gradient Boosting classifier performs one-hundred rounds of boosting with a learning rate of 0.3, growing a tree to a maximum depth of six during training.

2) Long Short-Term Memory: Long Short-Term Memory (LSTM) [33] is a specialized Recurrent Neural Network (RNN) which learns long term patterns in temporal data [40] by choosing what to remember and forget at each time step. The LSTM expands on the basic RNN's concept of memory by passing a memory cell along each time step [40]. The memory cell is updated at each time step via three distinct gating units: forget gate, input gate, and output gate [40]. Bidirectional LSTM (BLSTM) [34] introduces backward recurrent connections to learn temporal patterns by walking backward through each time step.

Our BLSTM is constructed with the Keras backend. Each model has two BLSTM layers, each outputting thirty-two (32) units after being activated by the hyperbolic tangent function (Tanh) and L2 regularization (0.05). The first of these BLSTM layers pass the previous state as an input to the second BLSTM layer. The output of the second BLSTM layer is passed to a batch normalization layer before a sixteen (16) neuron hidden layer activated by the rectified linear activation function (ReLU). The final layer is a fully connected layer with five (5) neurons (one for each finger) activated by the softmax function. The model uses the Adam optimizer to optimize categorical cross-entropy with gradient clipping to address the exploding gradient problem (clipnorm = 0.5) and an initial learning rate of 0.001. The learning rate is scaled by $\frac{5}{6}$ every 100 epochs. Each model is trained for eight hundred (800) epochs with a batch size of 128. Henceforth we will refer to BLSTM as LSTM unless otherwise explicitly stated.

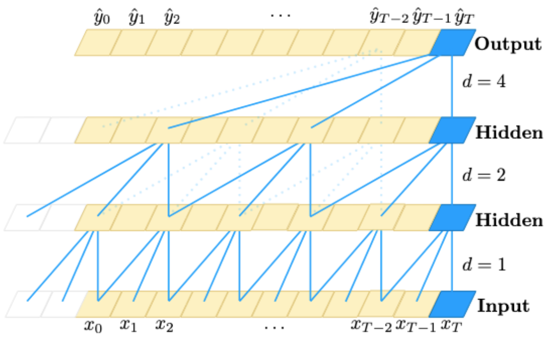
3) Temporal Convolutional Network: Temporal Convolutional Networks (TCN) [36] is a causal, one dimensional, fully-convolutional network (FCN) [41]. Here causal implies the use of causal convolutions, which only convolve previous time steps. This prevents any leakage of future information to the past. The second critical component of TCNs is dilation which increases the receptive field of the network. The dilation factor increases as a function of depth of the network 2^k where k is the depth of the network. Furthermore, this network makes use of residual connections [42] which enables layers to learn modifications to identity mappings [36]. See 2 for a visual representation of dilated casual convolutions and how residual connections are leveraged in TCN.

Our TCN consists of four FCNs with sixteen (16) features per input timestep. For example, if the input sequence were to contain forty (40) timesteps, each FCN would contain $40 * 16 = 640$ neurons. The features extracted from the FCNs are flattened into a one dimensional vector and are batch normalized. Next, we dropout thirty percent (30%) of the neurons before the output layer (with five (5) neurons). We train our model for three-hundred (300) epochs and optimize with Adam regularized with 0.001 L2 weight decay. We apply Cosine Annealing [43] over the entire cycle (300 epochs).

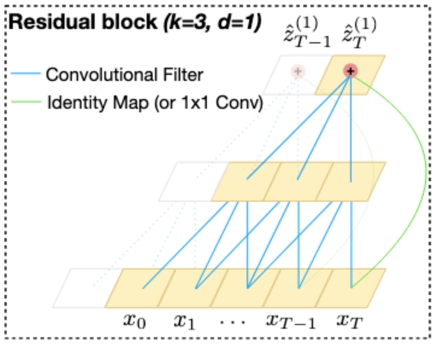
We found replacing rectified linear unit (ReLU) with scaled exponential linear units (SELU) [44] and adding batch normalization after each temporal block yielded considerable model accuracy gains. Additionally, we replaced the random uniform weight initialization with the method presented in [45].

B. Data Description

We have collected the individual finger movements of four subjects including male and female. It should be noted that three of the subjects used their left arms, while the right arm was used by the remaining one. Data is collected by an epidermal electronic system (EES) proposed in [12]. The device records single-channel data at 250 Hz via an Android application over Bluetooth transmission. Data acquisition consisted of completing N individual finger flexions at regular time intervals, where N varied for each subject. We then apply


$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ $\cdots$ $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
Output
$d = 4$
Hidden
$d = 2$
Hidden
$d = 1$
Input
x_0 x_1 x_2 $\cdots$ x_{T-2} x_{T-1} x_T

(a) Architecture of dilated casual convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$.


Residual block (k=3, d=1)
$\hat{z}^{(1)}_{T-1}$ $\hat{z}^{(1)}_T$
Convolutional Filter
Identity Map (or 1x1 Conv)
x_0 x_1 $\cdots$ x_{T-1} x_T

(b) TCN residual connection example. The blue lines are convolutional filters in the residual function with the green lines representing the identity mappings.

Fig. 2: Components of the TCN with images borrowed from [36].

three pre-processing steps (i.e., data cleaning, data filtering and smoothing, individual finger movement detection) to prepare the data for the supervised individual finger movement classification task ahead.

Data cleaning is applied by removing noise and null values. Noise in our case may be a considerable number of timesteps before the first finger flexion. Preceding noise accumulates when the test subject pauses after initiating acquisition before beginning the experimental procedure. An example of noise can be seen in Fig. 3. Then, we apply data filtering in three steps to prune unwanted noise and smooth the signal. Note that the noise we are filtering here differs from the noise removed in the data cleaning step. Here we are referring to both electrical noise and signal drift. Electrical noise is seemingly random and distorts signals of interest while drift can be seen in figure 3 as the signal value decreases approximately linearly with time. The first filtering step involves applying a first-order Butterworth Bandpass filter and extracting the high output as shown in figure 4. We then use a Hilbert transform to extract the upper envelope of the signal. Lastly, a second polynomial order Savitzky-Golay filter [46] is applied to smooth our data.

Finally, we need to split the individual finger movements, which is also a necessary step for real-time movement detection. To do this, we start by finding each of the positive peak

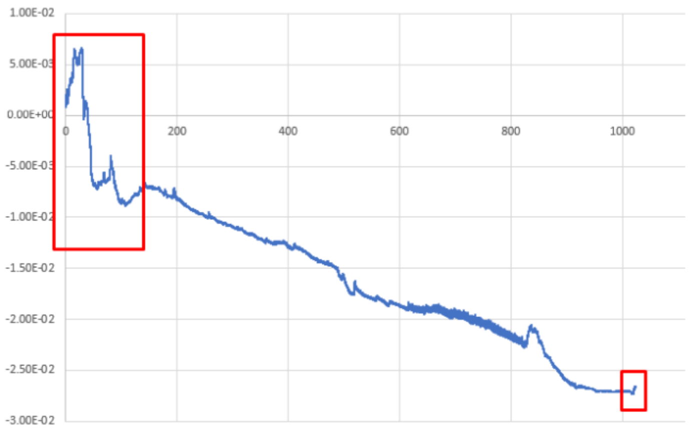


Fig. 3: A plot of raw acquisition data highlighting unwanted noise.

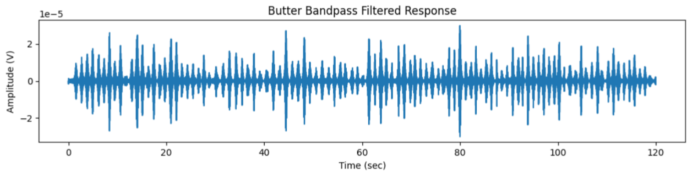


Fig. 4: The high output of a Butterworth Bandpass filter.

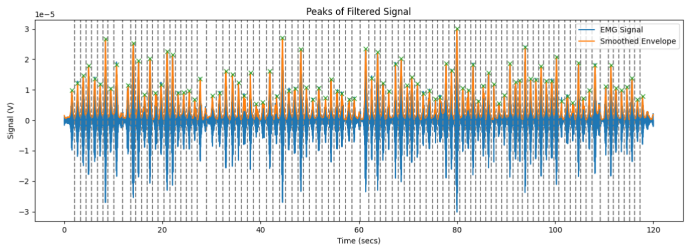


Fig. 5: A visualization of the midpoints between each peak.

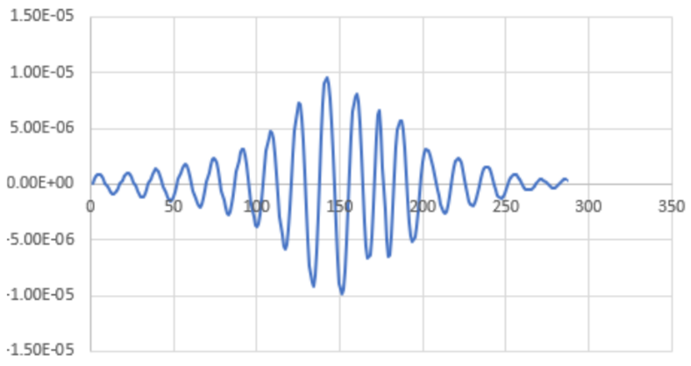
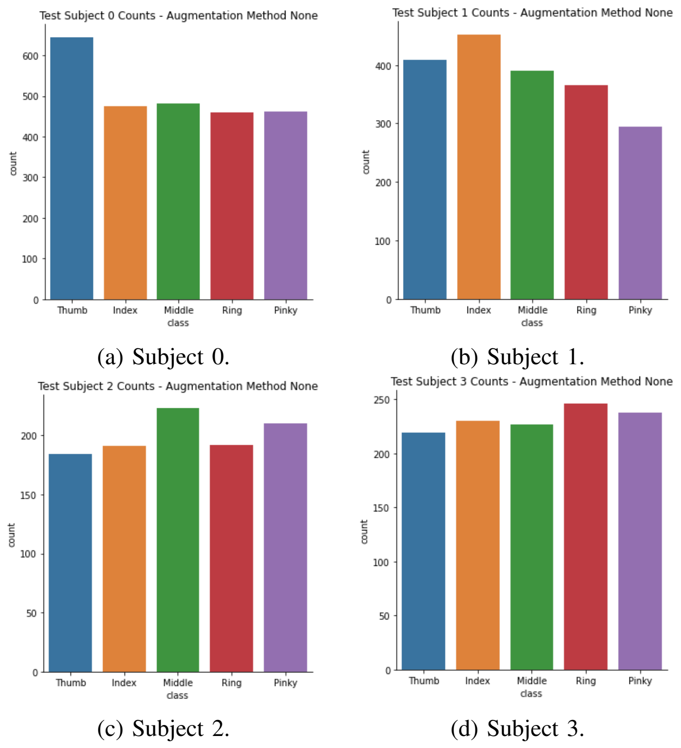


Fig. 6: An example of one of the extracted individual finger movements.

values in our signal as shown in Fig. **??**. Peaks are discovered via thresholding, where the threshold is the average value of our data (i.e., the output of the Savitzky-Golay filter). Next, we find the mid-point between each two peaks (Fig. 5) to cut each movement out. An individual signal is depicted in Fig. 6.



(a) Subject 0. (b) Subject 1. (c) Subject 2. (d) Subject 3.

Fig. 7: Extracted finger movement counts of for each subject.

Fig. 7 summarizes the number of finger movements detected for each subject using the above strategy.

It should be noted that each signal obtained from this procedure can be of different lengths. Table III provides a statistical summary of the sample lengths by test subject. However, considering the signal strength, it is reasonable to pad zeros on either end of the samples to have the same length for all movements. Furthermore, we leave our signal in time domain for classification considering our real time application and the order of time to necessary to convert to the frequency domain using FFT is $O(N \log_2(N))$.

Subject	Max	Mean	Std	Min
TS 0	441	383.54887	22.96217	256
TS 1	488	366.86311	49.30736	214
TS 2	424	365.71029	29.70054	255
TS 3	446	376.34453	26.56463	209

TABLE III: Statistical summary of individual finger movement sample lengths by subject.

C. Setup

To fairly evaluate the selected models we use fixed, stratified splits across all experiments. We train our models with 5-fold cross validation and evaluate performance on a holdout set. To evaluate the impact of data availability on each data augmentation method, we vary the size of the holdout set. We use the following five (5) holdout percentages: 10%, 30%, 50%, 70%, and 90%. Because we have four (4) test subjects, we then have a total of twenty (20) sets of data we evaluate, each of which is divided into five (5) folds for cross validation.

Through ablation study we selected the range of augmentation ratios to evaluate. Augmentation ratio is the ratio of synthetic sample count to original sample count. We found the benefits of augmentation to plateau at augmentation ratio 2.0 for a random forest classifier and two random transformation methods (Gaussian Noise and Magnitude Warping).

D. Results and Discussion

1) Model Performance: In table IV we present observed classification accuracy for each model by augmentation method at augmentation ratio 2.0 on the 10% holdout dataset. Each accuracy presented is the mean result of each of the four (4) test subjects and each of the five (5) cross validation folds. Model labels 'XGB', 'LSTM', and 'TCN' refer to Gradient Boosting, Bidirectional Long Short-Term Memory and Temporal Convolutional Network models, respectively. The best result is presented in bold by classification model. Baseline accuracies for each model may be found in the column labeled 'No Aug'. The labels 'RGS', 'SPA', 'GN', and 'WS' are used to denote the random combination of SPAWNER and Gaussian Noise, SPAWNER, Gaussian Noise, and Window Slicing augmentation methods, respectively. From this table three interesting observations may be made. First, the Gradient Boosting model outperforms the neural network models with and without augmentation. Even with augmentation, the two neural networks often do not surpass the baseline XGB model. The one exception to this is the TCN model with RGS augmentation which only marginally beats XGB without augmentation. Second, the application of each of the evaluated augmentation methods facilitated the development of a model with greater ability to generalize. Third, the most performant augmentation method for each model involved a pattern mixing technique. The random combination of Gaussian Noise and SPAWNER (RGS) resulted in the greatest gains for both TCN and XGB, while SPAWNER assisted best for the LSTM.

Table V compares classification accuracy achieved by each model for various holdout set percentages (i.e. varying the number of samples used to train the classification model). For each model and holdout percentage we present mean accuracy with the most effective augmentation method, the name of the augmentation method used and results without augmentation. The augmentation ratio applied for the results which are presented was 2.0. Accuracies reported are the mean of each of the four (4) test subjects and the five (5) cross validation folds. The best results are reported in bold by holdout percentage. A number of observations may be made from this table. First, the Gradient Boosting model dominates the other two models across all holdout percentages, with and without augmentation. Second, the TCN consistently outperforms the LSTM with and without augmentation. A third interesting observation, both XGB and TCN, using only 50% of the available data and the most effective augmentation method, are able to achieve nearly the same performance as augmentation with 90% of the data used for training. The XGB's mean result with RGS at holdout percentage 50% differs by only 0.236%. We may also observe the pattern mixing techniques were most impactful in general for these models. The XGB model prefers RGS for holdout percentages 10, 30, 50, and 70, while TCN prefers RGS for 10 and SPA for 30, 50, 70, and 90. The LSTM is less consistent in this respect as it prefers SPA for holdout percentages 10, 30, and 70, GN for 50, and RGS for holdout 90. These LSTM results seem to be inline with [14] which also observed mixed results for augmentation on their LSTM. Lastly, the XGB model preferred WS at 90% holdout and received an approximate 18.4% accuracy improvement, while TCN and LSTM only saw around 13.7% and 7.5% improvements, respectively. WS was only applied to XGB, so it is possible that the LSTM and TCN models would similarly benefit from WS at this holdout percentage.

2) Augmentation Method Performance: We evaluate augmentation ratio using the XGB model and the 10% holdout dataset in table VI. Augmentation ratio is evaluated in the inclusive range of 0.0 to 2.0 in increments of 0.25. Accuracies reported are the mean of each of the four (4) test subjects and the five (5) cross validation folds. The best results are reported in bold by augmentation ratio. The results indicate each of the evaluated augmentation techniques improve the XGB model's ability to generalize for each of the augmentation ratios on our sets of data. We observe for augmentation ratios less than or equal to 1.0, Gaussian Noise outperforms the more informed SPAWNER method. However, for augmentation ratios greater than 1.0 SPAWNER bests Gaussian Noise. We also observe the random combination of Gaussian Noise and SPAWNER outperforming each of the other methods more often than not. Interestingly, we find Window Slicing to be the best method for two of the larger augmentation ratios (i.e. 1.5 and 1.75).

In table VII we study the impact of each augmentation method on the XGB model with varied amounts of available training data. Augmentation ratio was fixed at 2.0. Each accuracy reported is the mean of our four (4) test subjects and the five (5) cross validation folds. Similar to our previous observations, we find each evaluated augmentation method improves generalization of the XGB model. Next, we observe the random combination method and window slicing to be the two most impactful methods across the holdout percentages. The random combination method is found to me most effective at holdout percentages 10, 30, and 70, while Window Slicing is dominant at 50 and 90. The most staggering result found in this table is the relative performance of Window Slicing at 90% holdout. We observe nearly a 5.3% advantage over the next closest method (RGS). Whereas the results of the augmentation methods differ at most by approximately 2.0% for all of the other holdout percentages.

In table VIII mean accuracies are presented by test subject and augmentation method for augmentation ratio 2.0 on the 10% holdout dataset. We evaluated a generative augmentation technique, DCGAN, on a single subject and holdout percentage due to the impracticality of this approach on our dataset described in section II-E and the relatively poor results we observed. While the DCGAN results indicate an improvement over the baseline, it is the least effective method of those

Model	No Aug	RGS	SPA	GN	WS
XGB	0.95235 ± 0.04372	$\mathbf{0.97984 \pm 0.02310}$	0.97752 ± 0.02486	0.97594 ± 0.02617	0.97702 ± 0.02443
LSTM	0.81712 ± 0.12133	0.86385 ± 0.07545	$\mathbf{0.88893 \pm 0.06274}$	0.85848 ± 0.07311	-
TCN	0.91236 ± 0.06445	$\mathbf{0.95302 \pm 0.04385}$	0.94760 ± 0.04789	0.94229 ± 0.04374	-

TABLE IV: Accuracies for each model and augmentation method for 10% holdout dataset. Each reported accuracy is the mean of each of our four (4) test subjects. The best performing augmentation method is presented in bold for each model. The augmentation ratio used for each augmentation method was 2.0.

	XGB			LSTM			TCN		
Holdout %	No Aug	Aug	Method	No Aug	Aug	Method	No Aug	Aug	Method
10%	0.95235 ± 0.04372	$\mathbf{0.97984 \pm 0.02310}$	RGS	0.81712 ± 0.12133	0.88893 ± 0.06274	SPA	0.91236 ± 0.06445	0.95302 ± 0.04385	RGS
30%	0.93714 ± 0.04477	$\mathbf{0.97804 \pm 0.02623}$	RGS	0.77201 ± 0.11657	0.86657 ± 0.07867	SPA	0.90639 ± 0.06157	0.95050 ± 0.04082	SPA
50%	0.90015 ± 0.05545	$\mathbf{0.97748 \pm 0.02275}$	RGS	0.79649 ± 0.11619	0.86813 ± 0.07812	GN	0.86164 ± 0.04563	0.94362 ± 0.04208	SPA
70%	0.80274 ± 0.08039	$\mathbf{0.94711 \pm 0.03712}$	RGS	0.59066 ± 0.14262	0.77810 ± 0.12785	SPA	0.68302 ± 0.05163	0.90617 ± 0.03098	SPA
90%	0.52775 ± 0.05004	$\mathbf{0.79106 \pm 0.07671}$	WS	0.45756 ± 0.07240	0.53219 ± 0.09780	RGS	0.39009 ± 0.04394	0.52736 ± 0.08426	SPA

TABLE V: Accuracies grouped by no augmentation (No Aug) and best augmentation method for each model and holdout percentage. Each reported accuracy is the mean of each of our four (4) test subjects.

Aug Ratio	RGS	SPA	GN	WS
0.00	$\mathbf{0.95235 \pm 0.04372}$	$\mathbf{0.95235 \pm 0.04372}$	$\mathbf{0.95235 \pm 0.04372}$	$\mathbf{0.95235 \pm 0.04372}$
0.25	0.96276 ± 0.03029	0.96341 ± 0.02977	$\mathbf{0.96445 \pm 0.02928}$	0.96089 ± 0.03117
0.50	$\mathbf{0.97271 \pm 0.02428}$	0.96430 ± 0.03101	0.97243 ± 0.02430	0.97025 ± 0.02943
0.75	$\mathbf{0.97613 \pm 0.02423}$	0.96852 ± 0.03159	0.97400 ± 0.02408	0.97384 ± 0.02448
1.00	$\mathbf{0.97874 \pm 0.02063}$	0.97322 ± 0.02567	0.97440 ± 0.02493	0.97696 ± 0.02439
1.25	$\mathbf{0.97664 \pm 0.02633}$	0.97529 ± 0.02535	0.97233 ± 0.02578	0.97475 ± 0.02654
1.50	0.97778 ± 0.02334	0.97526 ± 0.02605	0.97153 ± 0.02808	$\mathbf{0.97867 \pm 0.02496}$
1.75	0.97746 ± 0.02550	0.97593 ± 0.02785	0.97318 ± 0.03004	$\mathbf{0.97867 \pm 0.02485}$
2.00	$\mathbf{0.97984 \pm 0.02310}$	0.97752 ± 0.02486	0.97594 ± 0.02617	0.97702 ± 0.02443

TABLE VI: XGB accuracies for each augmentation ratio and method on 10% holdout. Each reported accuracy is the mean of each of our four (4) test subjects.

Holdout %	No Aug	RGS	SPA	GN	WS
10%	0.95235 ± 0.04372	$\mathbf{0.97984 \pm 0.02310}$	0.97752 ± 0.02486	0.97594 ± 0.02617	0.97702 ± 0.02443
30%	0.93714 ± 0.04477	$\mathbf{0.97804 \pm 0.02623}$	0.97646 ± 0.02815	0.97446 ± 0.02319	0.97617 ± 0.02765
50%	0.90015 ± 0.05545	0.97748 ± 0.02275	0.96996 ± 0.02748	0.97190 ± 0.02766	$\mathbf{0.97551 \pm 0.02454}$
70%	0.80274 ± 0.08039	$\mathbf{0.94711 \pm 0.03712}$	0.92897 ± 0.05542	0.93946 ± 0.04145	0.94623 ± 0.04054
90%	0.52775 ± 0.05004	0.73826 ± 0.07390	0.72256 ± 0.07249	0.72703 ± 0.08115	$\mathbf{0.79106 \pm 0.07671}$

TABLE VII: XGB accuracies for each holdout percentage and augmentation method with ratio 2.0. Each reported accuracy is the mean of each of our four (4) test subjects.

we evaluated and the performance benefit is marginal. An issue with the DCGAN for augmentation ratios greater than 1.0 is the generator provides a deterministic mapping from state to action spaces (i.e. input to output). For this reason augmentation ratios larger than 1.0 result in a training set with duplicate synthetic values, similar to the up sampling augmentation approach. Note duplicate synthetic values are highly improbable for each of the other augmentation methods due to their randomness and implementation details. We found Gaussian Noise and SPAWNER each to have been one of the most performant methods for one (1) of the four (4) test subjects. The random combination of these two methods was observed to be one of the most effective for three (3) of the four (4) test subjects.

3) *Impact of Sex:* Another interesting observation found in table VIII is the relative difficulty of classifying the individual finger movements of the female test subject (test subject zero (0)). The mean baseline accuracy is approximately 88.4% for the female subject while each of the males' mean accuracies are above 95%; nearly a 7% difference. Our experiments demonstrate augmentation helps close this gap. However, it seems accurately identifying sEMG signals is more challenging for female subjects than male subjects. Because of our limited sample size (i.e. a single (1) female test subject and three (3) male subjects), it is possible these results are not representative of the populations of all males and all females. However, other studies ([47]–[50]), too, have observed differences in sEMG signals acquired from male and female subjects.

Test Subject	Sex	No Aug	RGS	SPA	GN	WS	DCGAN
Test Subject 0	M	0.95336 ± 0.00901	0.98656 ± 0.00451	0.98419 ± 0.00559	$\mathbf{0.98735 \pm 0.00331}$	0.98656 ± 0.00354	0.95968 ± 0.01166
Test Subject 1	F	0.88438 ± 0.00856	$\mathbf{0.94271 \pm 0.00974}$	0.93750 ± 0.00974	0.93542 ± 0.01863	0.93854 ± 0.01622	-
Test Subject 2	M	0.98020 ± 0.01400	$\mathbf{0.99010 \pm 0.00000}$	$\mathbf{0.99010 \pm 0.00000}$	0.98614 ± 0.00886	0.98812 ± 0.00443	-
Test Subject 3	M	0.99145 ± 0.00855	$\mathbf{1.00000 \pm 0.00000}$	0.99829 ± 0.00382	0.99487 ± 0.00468	0.99487 ± 0.00468	-

TABLE VIII: XGB accuracies for 10% holdout for each test subject and augmentation method with ratio 2.0. Each reported accuracy is the mean of five (5) cross validation folds.

IV. Conclusion

Surface electromyography (sEMG) has great potential to improve the quality of life of amputees, however there is a need for more advanced classification for sEMG equipped prostheses to be more widely adopted. We have present an analysis of data augmentation applied to an sEMG dataset of individual finger movements to address limited data availability, a key issue faced when developing sEMG prostheses for the hand. We find the classical machine learning model XGB to be most effective at capturing the distribution of our dataset when compared to state of the art deep models. Furthermore, while we do not find a single augmentation method to consistently yield the best performance gains, we observe, in the presence of sufficient data, the random combination of pattern mixing and random transformation to be effective. Finally, a more staggering result is the effectiveness of window slicing which appears to improve as the amount of training data available is reduced.

References

[1] K. Ziegler-Graham, E. J. MacKenzie, P. L. Ephraim, T. G. Travison, and R. Brookmeyer, "Estimating the prevalence of limb loss in the united states: 2005 to 2050," *Archives of Physical Medicine and Rehabilitation*, vol. 89, no. 3, p. 422–429, 2008.

[2] M. P. Fahrenkopf, N. S. Adams, J. P. Kelpin, and V. H. Do, "Hand amputation," *Eplasty*, vol. 18, Sep 2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6173827/

[3] M. A. Cavalcanti Garcia and T. Vieira, "Surface electromyography: Why, when and how to use it," *Revista Andaluza de Medicina del Deporte*, vol. 4, pp. 17–28, 04 2011.

[4] E. A. Biddiss and T. T. Chau, "Upper limb prosthesis use and abandonment: A survey of the last 25 years," *Prosthetics and Orthotics International*, vol. 31, no. 3, pp. 236–257, 2007, pMID: 17979010. [Online]. Available: https://doi.org/10.1080/03093640600994581

[5] P. Kaczmarek, T. Mańkowski, and J. Tomczyński, "putemg—a surface electromyography hand gesture recognition dataset," *Sensors*, vol. 19, no. 16, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/16/3548

[6] P. Tsinganos, B. Cornelis, J. Cornelis, B. Jansen, and A. Skodras, "Data augmentation of surface electromyography for hand gesture recognition," *Sensors*, vol. 20, no. 17, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/17/4892

[7] Ottobock. (2022) Ottobock bebionic hand. [Online]. Available: https://shop.ottobock.us/Prosthetics/Upper-Limb-Prosthetics/bebionic/bebionic-Hands-%26-Gloves/Ottobock-bebionic-Hand/p/BB1000~50_B

[8] Ossur. (2023) i-limb®quantum. [Online]. Available: https://www.ossur.com/en-us/prosthetics/arms/i-limb-quantum

[9] Taska. (2022). [Online]. Available: https://www.taskaprosthetics.com/

[10] M. Gomez-Correa and D. Cruz-Ortiz, "Low-cost wearable band sensors of surface electromyography for detecting hand movements," *Sensors*, vol. 22, no. 16, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/16/5931

[11] A. Prakash, B. Kumari, and S. Sharma, "A low-cost, wearable semg sensor for upper limb prosthetic application," *Journal of Medical Engineering & Technology*, vol. 43, no. 4, pp. 235–247, 2019, pMID: 31414614. [Online]. Available: https://doi.org/10.1080/03091902.2019.1653391

[12] W.-H. Yeo, Y.-S. Kim, J. Lee, A. Ameen, L. Shi, M. Li, S. Wang, R. Ma, S. H. Jin, Z. Kang, Y. Huang, and J. A. Rogers, "Multifunctional epidermal electronics printed directly onto the skin," *Advanced Materials*, vol. 25, no. 20, pp. 2773–2778, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.201204426

[13] A. Rainoldi, G. Melchiorri, and I. Caruso, "A method for positioning electrodes during surface emg recordings in lower limb muscles," *Journal of neuroscience methods*, vol. 134, pp. 37–43, 04 2004.

[14] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *PLOS ONE*, vol. 16, no. 7, pp. 1–32, 07 2021. [Online]. Available: https://doi.org/10.1371/journal.pone.0254841

[15] L. Huang, W. Pan, Y. Zhang, L. Qian, N. Gao, and Y. Wu, "Data augmentation for deep learning-based radio modulation classification," 2019.

[16] T. T. Um, F. M. J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić, "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks," *Proceedings of the 19th ACM International Conference on Multimodal Interaction*. [Online]. Available: http://dx.doi.org/10.1145/3136755.3136817

[17] K. Kamycki, T. Kapuscinski, and M. Oszust, "Data augmentation with suboptimal warping for time-series classification," *Sensors*, vol. 20, p. 98, 12 2019.

[18] B. K. Iwana and S. Uchida, "Time series data augmentation for neural networks by time warping with a discriminative teacher," *CoRR*, vol. abs/2004.08780, 2020. [Online]. Available: https://arxiv.org/abs/2004.08780

[19] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015. [Online]. Available: https://arxiv.org/abs/1511.06434

[20] R. Anicet Zanini and E. Luna Colombini, "Parkinson's disease emg data augmentation and simulation with dcgans and style transfer," *Sensors*, vol. 20, no. 9, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/9/2605

[21] F. Coelho, M. F. Pinto, A. G. Melo, G. S. Ramos, and A. L. M. Marcato, "A novel semg data augmentation based on wgan-gp," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 0, no. 0, pp. 1–10, 2022, pMID: 35862582. [Online]. Available: https://doi.org/10.1080/10255842.2022.2102422

[22] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[23] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf

[24] M. N. Fekri, A. M. Ghosh, and K. Grolinger, "Generating energy data for machine learning with recurrent generative adversarial networks," *Energies*, vol. 13, no. 1, 2020. [Online]. Available: https://www.mdpi.com/1996-1073/13/1/130

[25] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional gans," 2017.

[26] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N.-C. Yen, C. C. Tung, and H. H. Liu, "The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1971, pp. 903–995, 1998. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1998.0193

[27] A. Furui, H. Hayashi, G. Nakamura, T. Chin, and T. Tsuji, "An artificial emg generation model based on signal-dependent noise and related application to motion classification," *PLOS ONE*, vol. 12, p. e0180112, 06 2017.

[28] D. P. Botelho, K. Curran, and M. M. Lowery, "Anatomically accurate model of emg during index finger flexion and abduction derived from diffusion tensor imaging," *PLoS computational biology*, vol. 15, p. e01007267, 08 2018.

[29] A. Guerrero and J. Macías-Díaz, "A package for the computational analysis of complex biophysical signals," *International Journal of Modern Physics C*, vol. 30, p. 1950005, 01 2019.

[30] M. M. Lowery, *EMG Modeling and Simulation*. John Wiley & Sons, Ltd, 2016, ch. 8, pp. 210–246. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119082934.ch08

[31] M. Atzori, A. Gijsberts, C. Castellini, B. Caputo, A.-G. M. Hager, S. Elsig, G. Giatsidis, F. Bassetto, and H. Müller, "Electromyography data for non-invasive naturally-controlled robotic hand prostheses," *Scientific data*, vol. 1, p. 140053, 2014. [Online]. Available: https://europepmc.org/articles/PMC4421935

[32] M. Atzori, M. Cognolato, and H. Müller, "Deep learning with convolutional neural networks applied to electromyography data: A resource for the classification of movements for prosthetic hands," *Frontiers in Neurorobotics*, vol. 10, p. 9, 2016. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2016.00009

[33] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[34] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[35] T. Chen and C. Guestrin, "XGBoost," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2016. [Online]. Available: https://doi.org/10.1145%2F2939672.2939785

[36] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *CoRR*, vol. abs/1803.01271, 2018. [Online]. Available: http://arxiv.org/abs/1803.01271

[37] A. Le Guennec, S. Malinowski, and R. Tavenard, "Data Augmentation for Time Series Classification using Convolutional Neural Networks," in *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, Riva Del Garda, Italy, Sep. 2016. [Online]. Available: https://halshs.archives-ouvertes.fr/halshs-01357973

[38] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[39] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf

[40] S. J. Russell and P. Norvig, *Artificial Intelligence: a modern approach*, 4th ed. Pearson, 2020.

[41] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," 2014. [Online]. Available: https://arxiv.org/abs/1411.4038

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[43] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," 2016. [Online]. Available: https://arxiv.org/abs/1608.03983

[44] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," 2017. [Online]. Available: https://arxiv.org/abs/1706.02515

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015. [Online]. Available: https://arxiv.org/abs/1502.01852

[46] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964. [Online]. Available: https://doi.org/10.1021/ac60214a047

[47] R. Zhang, X. Zhang, D. He, R. Wang, and Y. Guo, "semg signals characterization and identification of hand movements by machine learning considering sex differences," *Applied Sciences*, vol. 12, no. 6, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/6/2962

[48] F. Meduri, M. Beretta-Piccoli, L. Calanni, V. Segreto, G. Giovanetti, M. Barbero, C. Cescon, and G. D'Antona, "Inter-gender semg evaluation of central and peripheral fatigue in biceps brachii of young healthy subjects," *PLOS ONE*, vol. 11, no. 12, pp. 1–14, 12 2016. [Online]. Available: https://doi.org/10.1371/journal.pone.0168443

[49] J. Bouffard, R. Martinez, A. Plamondon, J. N. Côté, and M. Begon, "Sex differences in glenohumeral muscle activation and coactivation during a box lifting task," *Ergonomics*, vol. 62, no. 10, pp. 1327–1338, 2019, pMID: 31282824. [Online]. Available: https://doi.org/10.1080/00140139.2019.1640396

[50] R. Martinez, J. Bouffard, B. Michaud, A. Plamondon, J. N. Côté, and M. Begon, "Sex differences in upper limb 3D joint contributions during a lifting task," *Ergonomics*, vol. 62, no. 5, pp. 682–693, May 2019.