MAS Journal of Applied Sciences
Uygulamalı Bilimler Dergisi
ISSN: 2757-5675
masjaps.com

OPEN ACCESS

DOI: http://dx.doi.org/10.5281/zenodo.15099262

Research Article

Enhancing EMG Signals for Amputee People with Deep Neural Network and Optimization Algorithms

Çağdaş ÖZER [1*], Zeynep ORMAN [1]
[1] Istanbul University-Cerrahpaşa, Muhendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, İstanbul
*Corresponding author: cagdas.ozer@ogr.iuc.edu.tr

Received: 12.01.2025 **Accepted**: 26.02.2025

Abstract
Individuals with amputations often rely on prosthetic limbs to maintain daily functionality; however, over time, the performance of these devices can be compromised by wear, signal degradation, or other technical issues. In this study, we investigate the enhancement of electromyography (EMG) signals to mitigate changes in signal characteristics associated with long-term use by amputees. Our approach employs deep neural networks (DNN) integrated with various optimization algorithms. Data were acquired using an MYO Armband on the right arms of seven volunteers performing repeated fist clenching until muscle fatigue set in. The acquired data were augmented using synthetic data generation techniques and subsequently processed with a DNN that incorporated methods such as Principal Component Analysis (PCA), low variance and high correlation filters, nonlinear convolution layers, ensemble learning, bagging, batch normalization, and optimization algorithms including Stochastic Gradient Descent (SGD), Adagrad, RMSprop, Adam, and Particle Swarm Optimization (PSO). The performance was evaluated using metrics such as accuracy, precision, recall, and F-measure. Without optimization, the precision was 0.76; however, after extensive testing of various algorithmic combinations and synthetic data augmentation, the best configuration achieved a precision of approximately 0.98. These findings demonstrate that, with carefully selected deep learning and optimization strategies, EMG signals can be processed in near real-time, thereby significantly reducing the impact of mobility limitations.

Keywords: EMG signals, deep neural networks, optimization, data preprocessing

1. Introduction

Since the 1970s, prosthetic limbs have been instrumental in restoring mobility for millions of individuals with limb losses (Fukuda et al., 1998). Advances in prosthetic technology have enabled improved functionality and greater independence for users. Early work, such as that by Wiener, introduced the concept of using electromyogram (EMG) signals for real-time control of prosthetic arms. Today, EMG signals—owing to their natural correlation with muscle contraction and relaxation (Artemiadis et al., 2007; Li-Zhi et al., 2018)—remain a cornerstone for controlling advanced prosthetic systems, including the Boston Arm (MIT) and the Utah Artificial Arm (Moradi et al., 2008).

In this study, we employ an MYO Armband, a device that utilizes wireless gesture and motion control technology, to capture EMG signals from the right arms of seven volunteers. Our primary objective is to maximize the detection of muscle signals during prolonged use, particularly when signal characteristics change due to muscle fatigue. To address these challenges, we propose a machine learning framework that leverages deep neural networks (DNN) combined with advanced preprocessing and optimization techniques.

The collected dataset is first subjected to preprocessing steps, including removal of redundant entries, elimination of low-variance and highly correlated features, normalization, and Principal Component Analysis (PCA). These steps are designed to enhance the quality of the data and reduce training time while improving classification accuracy. Subsequently, the preprocessed data are input into a DNN that integrates various methods such as low variance filtering, data augmentation, nonlinear convolution layers, ensemble learning, bagging, batch normalization, and multiple optimization algorithms including Stochastic Gradient Descent (SGD), Adagrad, RMSprop, Adam, and Particle Swarm Optimization (PSO). The results of this integrated approach are discussed in the Results and Discussion section. Overall, our study demonstrates that with appropriate machine learning and optimization strategies, EMG signals can be processed accurately and in near real-time, thereby significantly mitigating the impact of signal degradation on prosthetic control.

The interpretation and processing of EMG signals have evolved significantly over the past decades. Early studies, such as those by Hardyck and colleagues and J.G. Kreifeldt (Kreifeldt, 1971), pioneered the use of surface EMG for signal amplification and noise reduction. Initial methods focused on enhancing the signal-to-noise ratio and involved techniques like low-pass filtering and autoregressive-moving average models (Graupe et al., 1975; Nelder et al., 1994). Subsequent research further advanced prosthetic control systems. For example, D. Graupe et al. (1975, 1989) and Peter C. Doerschuk et al. (1983) demonstrated the feasibility of using EMG signals for the control of prosthetic limbs by applying digital signal processing techniques. These studies laid the groundwork for understanding the role of EMG in translating muscle activity into control commands for devices such as the Boston Arm and the Utah Artificial Arm (Moradi et al., 2008).

Throughout the 1980s and 1990s, researchers investigated various methods to enhance EMG signal processing. Hershler and Milner (1978) analyzed EMG characteristics during locomotion, while studies by Saridis et al. (1982) and Doerschuk et al. (1983) focused on minimizing the cognitive load required for prosthetic control. Subsequent work by Hultman and Sjöholm (1983), Studer et al. (1984), and Gerber et al. (1984) introduced more sophisticated frameworks for quantitative EMG analysis and optimal filterbank adaptation. Advances continued with contributions such as Zhou et al. (1986), who proposed methods for deriving intramuscular EMG signals from surface measurements, and Paiss and Inbar (1987), who utilized autoregressive models for spectral analysis of EMG data. In the late 1980s and early 1990s, research by Winter and Yack

(1987) and Reucher et al. (1987) further refined EMG modeling by applying spatial filtering and statistical analysis techniques.The 1990s saw an increased focus on applying neural network techniques to EMG signal classification. Early work by Hiraiwa et al. (1989) and subsequent studies by Park and Lee (1998) employed neural networks to classify EMG patterns for prosthetic control. These approaches were later augmented with fuzzy systems and wavelet transforms (Zhang et al., 2002; Vuckovic et al., 2002), enhancing the robustness of EMG-based interfaces.

More recent studies have concentrated on real-time applications and advanced optimization methods. Chang et al. (1996) and Fukuda et al. (1997, 1998) explored EMG discrimination systems for human-machine interfaces, while later work by Ouyang (2013) and Goen (2014) addressed dynamic signal properties and neuromuscular disorder evaluation. Despite the considerable progress, relatively few studies have integrated advanced machine learning techniques with a comprehensive set of optimization algorithms for EMG signal enhancement.Our work extends this body of literature by combining deep neural networks with a suite of preprocessing and optimization techniques. By incorporating methods such as PCA, low variance and high correlation filtering, and various optimization algorithms (SGD, Adagrad, RMSprop, Adam, and Particle Swarm Optimization), our study aims to improve EMG signal prediction accuracy, even in scenarios involving significant signal degradation. This approach not only builds on prior research but also introduces novel algorithmic combinations that enhance the overall performance of prosthetic control systems.

2. Materials and Methods

In this study, muscle signal measurements were performed with MYO Armband produced by Thalmic Labs. MYO Armband is a wristband with 8 EMG electrodes, a three-axis accelerometer, a three-axis gyroscope, and a three-axis magnetic force measurement. The ARM Cortex M4 processor manages the armband operation, while the data transmission is performed through the BLE NRF51822 chip, which exchanges data with the HM-11 BLE module mounted on the bottom side of the prosthesis driving/control unit. The MYO armband has 8 separate electromyography (EMG) sensors, which are used to read the muscles and figure out what the limb is doing at that moment. This wrist strap can also show the orientation of the arm in 3-dimensional space. The orientation data from the wristband is transmitted to the computer via wireless communication (Bluetooth). After the data is processed with the software prepared in Python programming language, it is sent to the computer in real-time via TCP / IP communication (Erin and Boru, 2018). The EMG data is 200Hz, while the IMU data is 50Hz. Those speeds are constant and cannot be changed. And this collection only works while working with a single armband. Dataset collected by MYO Armband is processed and used by Deep Neural Network with PCA, Low variance Filter, Data Augmentation, Nonlinear Convolution Layers, Data Augmentation, Ensembling, Bagging, Batch Normalization, SGD, Adagrad, RMSprop, Adam and Particle Swarm optimization. The main idea is to classify these data properly and prepare the prediction models based on this accurate classification. Since there are 8 sensors, there are 8 fields in the data showing as emg1, emg2, emg3, emg4, emg5, emg6, emg7, and emg8. These fields can be seen from the sample dataset given in Figure 2. You can see the device and a little bit of data distribution from the device in the picture Figure 1 and the data distribution from the graphical Figure 7 below. From the sample row of the dataset, as we can see in Figure 7, 8 different sensors have different values according to the movement of the arm between the timestamp of start and the end. These sensor values have a minimum value of -128 and a maximum value of 127.

Figure 1. MYO Device (Bernhardt, 2015)

2.1. Used techniques

2.1.1. Principal component analysis

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to transform the observation set of interrelated variables into values of linearly unrelated variables called principal components. The number of major components is less than or equal to the number of original variables or observations. This transformation is defined perpendicular to the previous components such that the first principal component has the largest possible variance, and the resulting components all have the highest possible variance under constraint. The resulting vectors are a set of unrelated orthogonal principal components. PCA is sensitive to the relative scaling of the original variables. Analysts often use PCA as a tool for data analysis and building predictive models. It is often used to visualize genetic distance and the relationship between populations. PCA can be done by eigenvalue decomposition of a data matrix, a data covariance matrix, or singular value decomposition after the mean average of the data matrix for each feature. Results of a PCA are often discussed in terms of factor scores and component scores called loading. PCA is the simplest of true eigenvector-based multivariate analysis. It is often thought that their work reveals the internal structure of the data in a way that best explains the variance in the data. If a multivariate data set is visualized as a set of coordinates in a high dimensional data field, the PCA can present a sub-dimensional picture of this object, a projection from the most informative point of view. This is done by using only the first few basic components, thus reducing the dimensionality of the converted data.

2.1.2. Low variance filter

The Low Variance Filter node calculates each column variance and removes those columns with variance values below a specified threshold.

2.1.3. High correlation filter

This technique establishes the specification of a description step that will potentially remove variables that have large absolute correlations with other variables.

2.1.4. Data augmentation

Data augmentation is a useful technique that is used in almost every cutting-edge machine learning model in applications such as image and text classification. Heuristic data augmentation schemes are often set manually by human experts with extensive domain knowledge, resulting in inadequate augmentation policies. This resulted in new algorithms to automate the search process of transformation functions, new theoretical knowledge that enhances the understanding of various enhancement techniques widely used in practice, and a new framework for harnessing data enhancement to patch a flawed model and improve performance on the key data subpopulation.

2.1.5. Ensemble learning

Ensemble methods work best when algorithms are independent of each other. One way to get different classifiers is to train those using different algorithms. This increases the likelihood of making different types of mistakes, which increases the accuracy of the group. Ensemble itself is a supervised learning algorithm because it can be trained and then used to make predictions. Therefore, the trained ensemble represents a single hypothesis. However, this hypothesis is not necessarily included in the hypothesis space of the models from which it is constructed. Thus, it can be shown that ensembles have more flexibility in the functions they can represent. Theoretically, this flexibility enables them to fabricate education data more than a single model. Still, some ensemble techniques (especially bagging) tend to mitigate problems associated with overfitting training data in practice. Empirically, ensembles tend to show better results when there is significant variation between patterns. Therefore, many community methods try to encourage diversity among the models they combine. However, it has been shown that using a variety of powerful learning algorithms is more effective than using techniques that try to simplify models to encourage diversity.

2.1.6. Bagging (variance improving)

Bagging is a technique that does boot clustering. One way to reduce the variance of an estimate is to average multiple guesses. For example, we can train M different trees on different subsets of data (randomly selected by replacement) and compute the ensemble. Bagging uses boot sampling to obtain subsets of data to train basic students. It uses the voting for bagging, classification, and the average for regression, to gather the outputs of the core learners.

Given a training set D= {(x_1, y_1), . . . (x_n, y_n)}

Sample T sets of elements from D (with replacement) D_1, D_2, . . . D_T→T quasi replica training sets.

train a machine on each Di,i= 1, ..., T and obtain a sequence of T outputs $f_1(x)$, . . . $f_T(x)$.

The final aggregate classifier can be

For Regression:

$$\underline{f}(x) = \sum_{i=1}^{T} \quad fi(x) \qquad (1)$$

the average of fi for i= 1, ..., T;

For Classification

$$\underline{f}(x) = sign(\sum_{i=1}^{T} \quad fi(x)) \qquad (2)$$

2.2. Optimization algorithms

In deep learning applications, the absolute minimum value of the error function must be found for the learning process to result healthily. This process is carried out using optimization methods. Optimization is the method used to make the difference between the output value produced by the network and the actual value, the error to the smallest. One of the most used methods for optimizing artificial neural networks is gradient descent. There are three gradient descent methods (Mini-Batch Gradient Descent, Stochastic Gradient Descent, and Batch Gradient Descent) depending on the size of the data set used in a single iteration. Various algorithms (Rmsprop, Adagrad, Adam, etc.) are based on the gradient descent method (Kurt, 2018). While training data, calibrating the learning coefficient is critical in terms of optimization. However, it is not possible to fully adjust the learning coefficient in the model with every algorithm. Various gradient methods have been proposed to solve this problem (Li, 2017).

2.2.1. Adagrad

Adagrad makes different updates for each parameter by using t different learning coefficients for each step. Thus, it eliminates the need to adjust the learning coefficient manually. In Adagrad, each parameter has its

learning speed. The learning coefficient becomes excessively smaller because of the growth of the expression in which the learning coefficient value is divided in the update process during training (Çarkacı, 2018). Adagrad uses a different learning rate for every parameter θi at every time step t; first, it is being shown that Adagrad's per-parameter update, which then is vectorized. For brevity, g_t has been used to denote the gradient at time step t. g_t, i is then the partial derivative of the objective function w.r.t. to the parameter θi at time step t:

$$g_{t,i}=\nabla\theta J(\theta_{t,i}). \quad (3)$$

The SGD update for every parameter θi at each time step t then becomes:

$$\theta_{t+1,i}=\theta_{t,i}-\eta\cdot g_{t,i}. \quad (4)$$

In its update rule, Adagrad modifies the general learning rate η at each time step t for every parameter θi based on the past gradients that have been computed for θi:

$$\theta_{t+1,i}=\theta_{t,i}-\eta / \sqrt{G_{t,ii}+\epsilon}\cdot g_{t,i.} \quad (5)$$

Gt∈Rd×d here is a diagonal matrix where each diagonal element i, i is the sum of the squares of the gradients w.r.t. θi up to time step t, while ϵ is a smoothing term that avoids division by zero (usually on the order of 1e−8). As G_t contains the sum of the squares of the past gradients w.r.t. to all parameters θ along its diagonal, it can now be vectorized for the implementation by performing a matrix-vector product ⊙ between G_t and g_t:

$$\theta_{t+1}=\theta_t-\eta / \sqrt{G_t+\epsilon}\odot g_t. \quad (6)$$

2.2.2. Rmsprop

Rmsprop has been developed as a solution to the problem of over-minimizing the learning coefficient in the Adagrad algorithm. Instead of using all the values obtained from the squares of all past slopes in Adagrad, it restricts the value amount to a certain frame size (Kurt,2018; Ruder,2016).

$$E[g^2]_t=0.9E[g^2]_t-1+0.1g^2_t\theta_{t+1}=\theta_t-\eta / \sqrt{E[g2]_t+\epsilon}g_t \quad (7)$$

Rmsprop is similar to AdaGrad, and the difference is that the denominator is also decayed.

2.2.3. Adam optimization

ADAM, Adaptive Moment Estimation, is a more widely used method that adds momentum to the RMSprop method. Momentum updating is done with the exponential moving average, and when dealing with β, it is not necessary to change the learning rate. Like in RMSprop, here, the exponential moving average of the gradient square is taken. At the beginning of the training process of neural networks, SGD often goes in the wrong direction, while RMSprop is heading in the right direction. However, RMSprop is also affected by a noise like classical SGD, i.e. jumps around optimum when it finds a local minimum, leading to critical consequences. Just as we add momentum to the SGD, the same improvement is achieved with ADAM. The ADAM Algorithm was also generally regarded as quite robust in choosing hyperparameters (Studer et al.,1984).

$$m_t=\beta_1 mt-1+(1-\beta_1)gt \quad (8)$$
$$v_t=\beta_2 vt-1+(1-\beta_2)g_t^2 \quad (9)$$

This update rule is like the RMSProp. The difference is that the cumulative history of gradients is being looked at as well(m_t). ADAM is required to train some networks while using language models. SGD or ADAM with momentum is generally preferred to optimize neural networks. However, ADAM's theory in publications is not well understood, and it has some drawbacks. In elementary test problems, the method does not converge. It is known to give generalization errors. If the neural network is trained to provide zero loss in data used for training, it will not lose zero in other data points it has never seen before. It is

quite common to get worse generalization errors, especially when using SGD for image problems. For example, ADAM or factors in its structure may include finding the closest local minimum, i.e. being less noisy. While we need 3 buffers in ADAM, SGD requires 2 buffers. It is not very important unless we train a model that is a few gigabytes in size, but in such a case, it may not fit into memory. Instead of 1, 2 momentum parameters should be set. In this method, in addition to using the exponentially weighted averages (At) of the squares of past slopes, as is done in Rmsprop, it also stores the momentum changes (mt) in the cache. So, it combines Rmsprop and momentum. The default values are 0.9 for β_1; It is stated as 0.999 for β_2 and 10^8 for epsilon (Li, 2017; Çarkacı, 2018).

2.2.4. Stochastic gradient descent

SGD is an iterative optimization technique that uses mini data stacks to generate the expectation of the gradient rather than the full gradient using all available data. A large dataset containing randomly sampled samples probably contains redundant data. The larger the batch size, the more likely the surplus becomes. Some redundancies can be useful for smoothing noisy gradients, but massive batches tend to be of much less predictive value than large batches (Song et al., 2013). What if we could get the correct average gradient for much less computation. We can predict a larger average from a much smaller average by picking random samples from our dataset. Stochastic gradient descent (SGD) takes this idea to the extreme - using only one instance (batch size 1) for each iteration. Given enough iterations, SGD works but is too noisy. Stochastic means that a sample containing each batch is randomly selected. SGD performs a parameter update for each training example labeled x(i) and y(i):

$$\theta=\theta-\eta\cdot\nabla\theta J(\theta;x(i);y(i)). \quad (10)$$

In the SGD algorithm, the calculation is made over a training sample instead of all training data. In this way, possible memory deficiency problems are prevented (Kurt, 2018).

2.2.5. Particle swarm optimization

In the search field, particles are presented as potential solutions, and the fitness function is proposed as the basic mechanism for driving particle movements. The PSO, which includes the following people who are part of a community, has an idea that is part of a "field of belief" (search space) shared by neighboring individuals. Individuals can change this "state of vision" based on three factors: environmental knowledge; the individual's previous state history, and previous situations of the individual's neighborhood. Considering the rules of interaction, PSO, individuals in society adapt their belief schemes to more successful social networks. The rules of one-dimensional particle motion are shown in equations as follows:

$$v_i^{t+1}=w * v_i^t + d_1 * \varepsilon_1 * (p_{best} - x_i^t) + d_2 * \varepsilon_2 * (g_{best} - x_i^t) \quad (11)$$

$$x_i^{t+1}=x_i^t+v_i^{t+1} \quad (12)$$

Where the local best configuration (p_{best}), global best (g_{best}), new velocities ($v_{it}+1$), and positions ($x_{it}+1$) where x_{it} and v_{it} are the current position and velocity. ε1 and ε2 are chosen randomly in between (0,1). The tendency of a particle to remain in its current position is called inertia coefficient denoted by w, d_1, and d_2 (which can be modified as per requirement), which are referred to as the individual coefficient of acceleration and global coefficient of acceleration, respectively.

2.3. Deep neural network

The deep neural network algorithms used in the study are multi-layered versions of artificial neural networks that are inspired by the information processing method of the human brain. Since the single-layer perceptron, which was first developed and

evaluated as the most primitive artificial neural network, is not sufficient in solving nonlinear problems, a multi-layer perceptron has been developed (Fernandez et al., 2006). Most machine learning and deep learning algorithms use the idea of gradient descent. And that is based on the Newton algorithm that is about finding the roots of a 2d function. To achieve that, a point is randomly picked and slide to the left or right along the x-axis, based on the positive or negative value of the derivative of the slope of this function at the chosen point until the value of the y axis function comes to zero. The same idea is used in gradient descent, where we move or descend along a given path in multidimensional weight space. The cost function continues to decrease and stops when the error rate stops decreasing. Newton's method tends to get stuck at the local minimum if the derivative of the function at the current point is zero. Similarly, this risk exists when gradient descent is used on a non-convex function. The effect is multidimensional (each dimension represents a weight variable) and is amplified in the multilayer environment of DNN, resulting in a non-optimal weight set. In standard SGD, the learning rate is used as a constant multiplier of the gradient to calculate step size or update weight. This can cause the update to exceed a potential minimum if the slope is too steep or delay convergence if the slope is noisy. Using the concept of momentum in physics, the momentum algorithm presents a velocity variable v that is structured as an exponentially decreasing mean of the gradient. This helps prevent costly landings in the wrong direction. In the equation below, a Ɛ [0; 1) is the momentum parameter, and ε is the learning rate. The multi-layer perceptron contains three layers which is the input layer, hidden layer(s), and an output layer. Unlike the single-layer sensor, nonlinear classification can be made. Depending on the nature of the problem solved, the number of hidden layers and the number of neurons in these layers may change. To create a good generalization in the modeling process of architecture, the input data set is divided into two or three separate parts (Fukuda et al., 1998; Cichosz, 2015, Matt et al., 2015). In deep neural network algorithms, the entire data set is generally divided into two datasets called training data set and test data set (testing). The role of the training data set is to calculate the estimates of the weights in the neural network, and the role of the test data set is to test the accuracy of the obtained weight values with data hidden from the model (Priddy and Keller, 2005; Okut, 2018) works with. In the first stage of the backpropagation algorithm, variables are presented to the training network. By assigning a weight value for each neuron of each variable, after the values are multiplied by the weights and summed, it is sent to all neurons in the next layer by an activation function in the neurons in the hidden layer by adding the bias value (Moradi et al., 2008; Song et al., 2013). These values constitute the input values for the neurons in the output layer. After similar operations are performed in the output layer, an output value is obtained, the output and the actual value are compared, and the error value is obtained. In the second stage, errors obtained from the output layer are propagated to other layers in a backward direction. These errors are used to calculate the slope of the loss function relative to the weights in the network. Them, to update the weights for the idea of minimizing the loss function, optimization methods are used (Okut et al., 2011; Li, 2017). In this process, the partial derivative of the error according to the weights is made by using the chain rule in the derivative. The incoming information $x = [x_1, x_2, \ldots, x_M]$.

The weights $w = [w_1, w_2, \ldots, w_M]$ at the incoming connections and the bias b for the neuron, both of which constitute the parameters of the neuron. An intermediate processing step within the body of the neuron is implemented simply as a linear combination of the incoming data x using the weights w and bias b. The activation function $\sigma(\cdot)$ of the neuron, which applies a nonlinear transformation to the intermediate result z, thus creating the neuron's output information, y,. Therefore, the output of the ANN model can be viewed as

$$y = \sigma (w^t x + b).$$

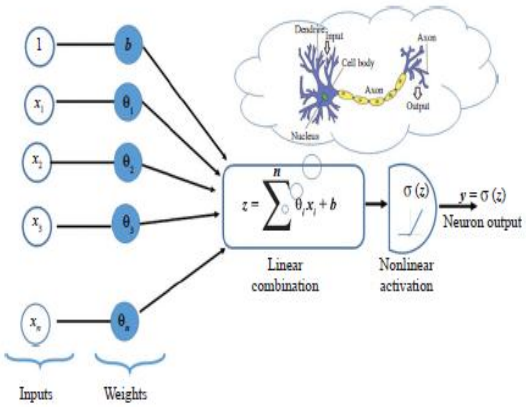

Figure 2. Single ANN Model (Gürbüz, 2020)

Figure 2 shows a representation of an artificial neuron, pointing out its following main components. Some highly used activation functions, namely sigmoid, tangent hyperbolic, and rectified linear unit, are shown in Figure 5, which also provides their definitions. The weights and bias of each neuron are parameters to learn to approach optimal matching between input and output data samples. Nonlinear activation functions are particularly important to allow more complex nonlinear mappings to be represented by a network of simple neurons.

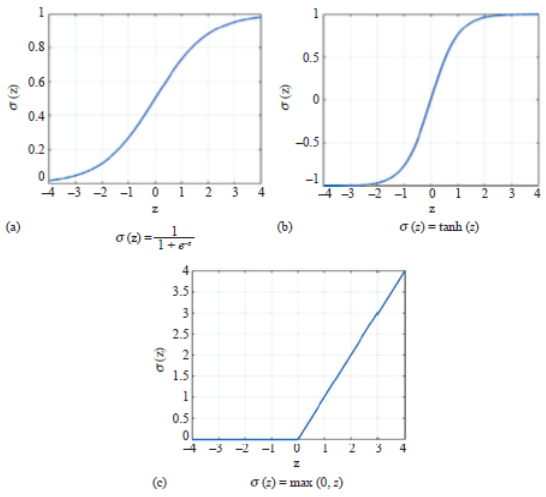

Figure 3. Three of the most commonly used activation functions: (a) sigmoid σ(z) = 1 1+e−z, (b) tangent hyperbolic σ(z) = tanh (z), and (c) rectified linear unit (ReLU) σ(z) = max (0, z) (Gürbüz, 2020)

An ANN consists of interconnections of neurons in a layered structure. The first layer of neurons takes data samples as input, while the last layer produces output samples, whether classification or regression results. There are one or more hidden layers that progressively transform the first layer neuron activations into final output samples. Figure 4 shows an ANN example with two hidden layers.

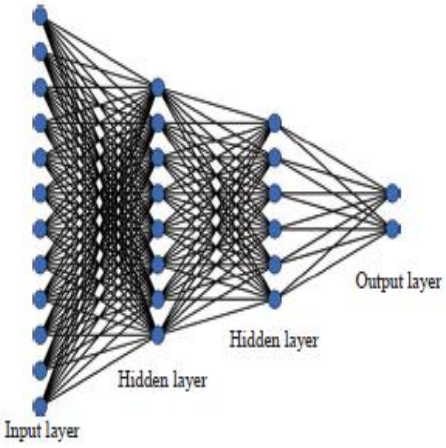

Figure 4. An ANN structure with two hidden layers

As the number of hidden layers increases, the ANN structure gets deeper and turns into a DNN. Although ANNs with many hidden layers are called DNN in the literature, deep learning and DNN structures are mostly related to the ability to learn features from raw sensor data and the models and structures that facilitate this ability.

2.4. Proposed work

It is aimed to use optimization algorithms to the fullest to accurately predict EMG signals with the proposed deep learning model. In this section, the processes in the proposed model will be explained in detail. The pseudocode of the deep learning framework used in the study is as shown below in Figure 5:

```
{
Identify the deep learning framework.
Ensure relevant instrumentation data is generated.
Decide upon the batch, epoch and iteration sizes.

Start the training.
Plot the training accuracy versus iteration values.
Is prescribed accuracy attained?
    If yes,
    {   Go for testing
    }
    Else
    {   Start improvement
        Switch method
        Improve upon Data:
            Look for co-relation and distinction.
        Improve upon Algorithm Tuning:
            Decide batch size, learning rate, weights etc.
        Use Ensemble:
            Use other models:
        Default:
        Goto Training
    }
    Save The Neural Network Model
}
```

Figure 5. Pseudo-code of the deep learning framework

The parameters that it has been used are 60% training 20% test, 20% validation, 100 epochs, 0.2 learning rate, Sigmoid activation function, low signal data as the input value, and to predict the correct signal data as the output value. Multiple approaches with many epochs and learning rates have been tried to find the optimized parameters. The result obtained from the proposed model is the calculated mean of those many tries; therefore,

it can be seen that this result was not for one time and was calculated mathematically to show its robustness.

Step 1. To prepare for better results, preprocessing was done before the model. At the start, Min-Max feature scaling was applied because this step is also a requirement for standardization. So, the data of EMG signals were normalized with the formula below.

$$X_{scaled} = (x - \min(x)) / (\max(x) - \min(x))$$

X represents a single feature/variable vector.

Since there are no missing values from the dataset that we have taken from the MYO Armband device, no techniques are needed to fill them.

Step 2. When the feature scaling is over, the following techniques were applied to enhance the results further. These techniques are PCA, low variance filter, high correlation filter, data augmentation, ensemble learning, and bagging.

Step 3. In this step, data is divided into training, testing, and validation. This process is not random, as the data is handpicked to avoid the randomness of the results.

Step 4. To see the prediction values from the Deep Neural Network, a standard model was applied without any optimization model or preprocessing techniques. It can be seen the accuracy result of the current data, with the correct picking of weights (random at first) and the number of hidden layers (in our case, it seems 8) and using the different activation functions (in our case, sigmoid worked the best) and the modifying the learning rates and the weights with the help of backpropagation algorithm to reach the upper numbers of the prediction.

Step 5. After the model gives a prediction result, it is time to use optimization algorithms and preprocessing techniques to increase the prediction value of the model further. This is the step that took most of the time in the research. First, it started with one optimization algorithm only, and that is Adam Optimization because it has been known before that Adam Optimization works well with Neural Networks. After that, many Optimization algorithms were applied single, and then combinations of these optimization algorithms were tested to reach the best result; this step took 916 tries. And the best-combined result in the model with the dataset comes from the combination of Adagrad, RMSProp, Adam Optimization, Stochastic Gradient Descent, and PSO with Deep Neural Networks.

3. Findings and Discussion

The Deep Neural Network and all other techniques were implemented via Python, and the results are shown in Table 1 and Table 2. Table 1 shows the results of the Deep Neural Network, without any of the preprocessing and optimization methods applied. It can be seen from Table 1 that the raw data results are not considered high enough to be useful. Table 2 shows the results of the Deep Neural Network with PCA, Low variance Filter, Data Augmentation, Nonlinear Convolution Layers, Data Augmentation, Ensembling, Bagging, Batch Normalization, SGD, Adagrad, RMSprop, Adam and Particle Swarm optimization. It can be seen from Table 2 that with more Synthetic data, and more specified hidden layers, with the help of preprocessing optimization methods, results are reached a point where they could be seen as reasonable. Future work can test and study more classification techniques, optimization methods, and different movements. Also, more feature extraction methods and other machine learning algorithms can be chosen to improve the results, or these methods can be used in a situation other than EMG signals.

Table 1. Model results before proposed method (DNN only)

Model	Accuracy	F Measure	Precision	Recall
Deep Neural Network	0.74	0.73	0.76	0.71
	Standard Deviation	**Standard Deviation**	**Standard Deviation**	**Standard Deviation**
	0.002	0.005	0.009	0.003

Table 2. Model combination results

Model	Accuracy	F Measure	Precision	Recall
Deep Neural Network	0.74	0.73	0.76	0.71
	Standard Deviation	Standard Deviation	Standard Deviation	Standard Deviation
	0.002	0.005	0.009	0.003
Deep Neural Network + Preprocessing	0.77	0.76	0.79	0.74
	Standard Deviation	Standard Deviation	Standard Deviation	Standard Deviation
	0.004	0.007	0.007	0.005
Deep Neural Network + Preprocessing + Adam	0.82	0.81	0.84	0.76
	Standard Deviation	Standard Deviation	Standard Deviation	Standard Deviation
	0.004	0.008	0.007	0.006
Deep Neural Network + Preprocessing + SGD	0.8	0.79	0.77	0.73
	Standard Deviation	Standard Deviation	Standard Deviation	Standard Deviation
	0.003	0.007	0.009	0.006
Deep Neural Network + Preprocessing + SGD + PSO	0.84	0.83	0.82	0.78
	Standard Deviation	Standard Deviation	Standard Deviation	Standard Deviation
	0.004	0.003	0.008	0.005
Deep Neural Network (With Synthetic Data) + Preprocessing + SGD + Adam	0.87	0.84	0.86	0.81
	Standard Deviation	Standard Deviation	Standard Deviation	Standard Deviation
	0.004	0.008	0.004	0.005
Deep Neural Network (With Synthetic Data) + Preprocessing + SGD + RMSProp + PSO	0.91	0.9	0.91	0.88
	Standard Deviation	Standard Deviation	Standard Deviation	Standard Deviation
	0.004	0.007	0.009	0.006

From Figure 7, you can see the differences between the normal model and the proposed model graphically.

Metric based results are accuracy,

Accuracy defines the distance between the actual value and the measured value (Sammut and Webb, 2015).

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

While

True Positive (TP): is where the signal is strong, and our model correctly predicts a strong signal

True Negative (TN) is where the signal is weak, and our model correctly predicts a weak signal

False Positive (FP): is where the signal is strong, and our model predicts a weak signal

False Negative (FN): is where the signal is weak, and our model predicts a strong signal

Precision,

Precision defines the distance between the measured values (Sammut and Webb, 2015).

$$Precision = \frac{(TP)}{(TP + FP)}$$

Recall,

$$Recall = \frac{(TP)}{(TP + FN)}$$

and F Measure.

$$F\ Measure = \frac{(2 * TP)}{(2 * TP + FP + FN)}$$

It can be seen from Table 1 that with the dataset of 152,842 rows, our Deep Neural Network Model has shown 0,74 Accuracy, 0,73 F Measure, 0,76 Precision and 0,71 Recall results before techniques (with 152,842 rows)

In Table 2, results of some of the combination that has been made can be seen. Including parameter tuning, preprocessing steps, optimization algorithm applications, and everything that is included in this search took a total of 916 tries. With the creation of synthetic data for further training, the dataset reached 24,728,319 rows. The best result, with the combination of the selected Optimization Algorithms, our model shows 0,93 Accuracy, 0,96 F Measure, 0,98 Precision, and 0,92 Recall. The significant improvement of the results within the model can be seen in Table 3.

Table 3. Model results after the proposed method (AdaGrad + RMSProp, adam, stochastic gradient descent + PSO + DNN)

Model	Accuracy	F Measure	Precision	Recall
Deep Neural Network (With Synthetic Data) + Adam + Adagrad + RMSProp + SGD + PSO	0.93	0.96	0.98	0.92
	Standard Deviation	Standard Deviation	Standard Deviation	Standard Deviation
	0.005	0.017	0.009	0.01

From Figure 6, it can be seen that, between the proposed model and the raw results, there is a big difference. It shows that with more data, preprocessing and optimization algorithm combinations make a big difference in the prediction result of the model.

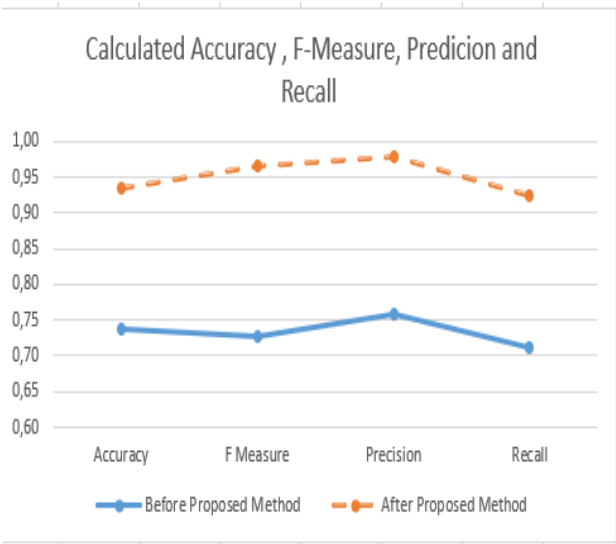

Figure 6. Calculated accuracy, f-measure, precision and recall

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9
timestamp	emg1	emg2	emg3	emg4	emg5	emg6	emg7	emg8
1570174405084153	-1	-4	-5	-5	0	-3	0	-1
1570174405084153	-1	2	11	5	1	2	0	-2
1570174405090136	-6	-3	-17	-10	-4	0	-2	-2
1570174405090136	5	4	4	2	1	1	0	0
1570174405090136	-2	-3	-1	-1	1	-2	-2	-2
1570174405090136	-2	-2	-1	-4	-8	0	-2	0
1570174405092131	1	-1	-4	-2	-8	0	2	0
1570174405092131	-3	-4	-2	-2	-4	-3	-1	0
1570174405103177	1	-8	-16	-8	-3	1	-2	0
1570174405103177	2	6	18	14	7	-1	0	-1
1570174405114074	-1	-1	-2	-2	-2	-2	-2	-2
1570174405114074	-8	-9	-9	-6	-1	-1	-2	-3
1570174405118321	-1	2	2	-1	0	1	0	1
1570174405118321	-2	1	-20	-11	-2	-4	-3	-3
1570174405134082	-1	-2	14	4	3	1	1	1
1570174405134082	-2	0	-1	2	4	0	0	-3
1570174405142098	-1	-6	-2	1	-5	-2	0	-1
1570174405142098	-3	-7	-5	-1	0	-2	-1	0
1570174405157106	-1	0	3	0	1	-3	-2	-3

Figure 7. Sample from the dataset numeric

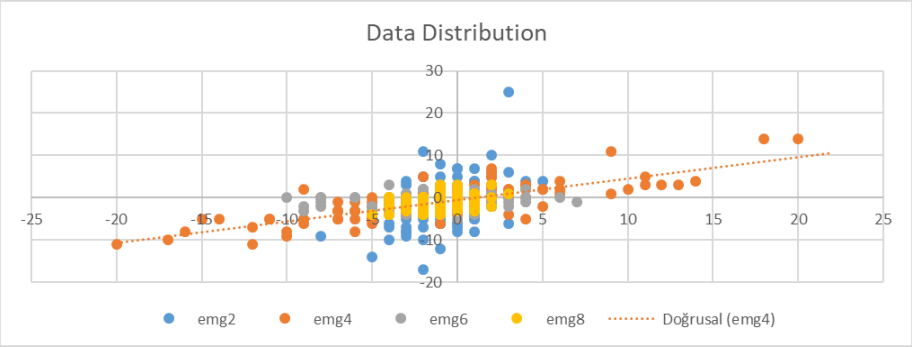

Figure 8. Data Distribution of 100 rows from dataset graphical

4. Conclusion

This study demonstrates that integrating deep neural networks with advanced preprocessing and optimization techniques significantly enhances the prediction of electromyography (EMG) signals. By employing methods such as Principal Component Analysis, low variance and high correlation filters, data augmentation, and ensemble approaches, combined with optimization algorithms—including Adagrad, RMSprop, Adam, Stochastic Gradient Descent, and Particle Swarm Optimization—we achieved substantial improvements in performance metrics. In particular, the precision of the model improved from 0.76 without optimization to approximately 0.98 after applying the proposed methodology.

The results indicate that robust EMG signal processing can be achieved in near real-time, potentially reducing the impact of signal degradation on the control of prosthetic devices. This advancement not only contributes to the field of prosthetic control but also opens avenues for further research on real-life implementations using compact hardware platforms such as Arduino or Raspberry Pi. Future work will explore alternative classification techniques, additional optimization methods, and further refinements in feature extraction to enhance model performance. The promising results of this study underscore the potential for machine learning applications to improve the quality of life for individuals with amputations by ensuring more reliable and accurate prosthetic control.

References

Abel, E.W., Zacharia, P.C., Forster, A., Farrow, T.L., 1996. Neural network analysis of the EMG interference pattern. *Medical Engineering and Physics*, 18(1): 12-17.

Andreas, G., Studer, R.M., de Figueiredo Rui J.P., Moschytz George, S., 1984. A new framework and computer program for quantitative emg signal analysis. *IEEE Transactions on Biomedical Engineering*, 31(12).

Artemiadis, P.K., Kyriakopoulos, K.J., 2007. EMG-based position and force control of a robot arm: Application to teleoperation and orthosis. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp:6.

Bernhardt, P., 2015. MYO Developer Blog. (https://developerblog.myo.com/), (Access Date: 03.01.2025).

Bodruzzaman, M., Devgan, S., Kari, S., 1992. Chaotic classification of electromyographic (EMG) signals via correlation dimension measurement. *IEEE Proceedings of South-East Conference*, Birmingham, pp. 95-98.

Cichosz, P., 2015. Data Mining Algorithms: Explained Using R. United States: John Wiley & Sons.

Çarkacı, N., 2018. Derin öğrenme uygulamalarında en sık kullanılan hiper parameteler. (https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4), (Accessed: 05.01.2025).

Daniel, G., William, K.C., 1975. Functional separation of EMG signals via ARMA identification methods for prosthesis control purposes. *Conference Proceedings of Systems, Man and Cybernetics International Conference on IEEE*, pp. 252-259.

Doerschuk Peter, C., Donald E., Gustafon, A., Willsky, S., 1983. Upper extremity limb function discrimination using EMG signal analysis. *Biomedical Engineering*, 18-29.

Dwyer, G., Noguchi, Y., Szeto, H.H., 1989. EMG burst waveform recognition procedure. *Proceedings of the 1989 Fifteenth Annual Northeast on Bioengineering Conference*, 27-28th March, Boston, MA, pp. 235-236.

Edward, A., Clancy Neville H., 1995. Multiple site electromyograph amplitude estimation. *IEEE Transactions on Biomedical Engineering*, 42(2).

Erin, K., Boru, B., 2018. Real-time control of industrial robot arm with EMG and gyroscope data. *Sakarya University Journal of Science*, 22(2): 509-515.

Farina, D., Merletti, R., 2000. Comparison of algorithms for estimation of EMG variables during voluntary isometric contractions. *Journal of Electro-myography and Kinesiology*, 10(5): 337-349.

Fernandez, C., Soria, E., Martin, J.D., Serrano, A.J., 2006. Neural networks for animal science applications: Two case studies. *Expert Systems With Applications*. 31: 444-450.

Friedman, J.H., 2001. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5): 1189-1232.

Fukuda, O., Tsujı, T., Ohtsuka A., Kaneko, M., 1998. EMG-based human-robot ınterface for rehabilitation aid. *IEEE International Conference on Robotics and Automation*, Proceedings Book, pp. 3492-3497.

Fukuda, O., Tsuji, T., Kaneko, M., 1997. An EMG controlled robotic manipulator using neural networks. *6th IEEE International Workshop on Robot and Human Communication*, Proceedings Book, 29 September-1 October, pp.442-447.

Fukuda, O., Tsuji, T., Ohtsuka, A., Kaneko, M., 1998. EMG-based human-robot interface for rehabilitation aid. *IEEE International Conference on Robotics and Automation*, Proceedings Book, 16-20 May, pp. 3492-3497.

Geoffrey, L.S., 2012. Application of time-varying analysis to diagnostic needle electromyography, *Medical Engineering & Physics*, 34(2): 249-255.

Goen, A., 2014. Classification of EMG signals for assessment of neuromuscular disorders. *International Journal of Electronics and Electrical Engineering*, 2(3).

Graupe, D., 1989. EMG Pattern analysis for patient-responsive control of fes in paraplegics for walker-supported walking. *IEEE Transactions on Bio-medical Engineering*, 36(7): 711-721.

Graupe, D., Kohn, K.H., Basseas, S.P., 1989. Control of electrically-stimulated walk-ing of paraplegics via above- and below-lesion EMG signature identification. *IEEE Transactions on Automatic Control*, 34(2): 130-138.

Gurbuz, S., 2020. Deep Neural Network Design for Radar Applications. The Institution of Engineering and Technology: Stevenage, UK.

Gut, R., Moschytz George S., 2000. High-precision EMG signal decomposition using communication techniques. *IEEE Transactions on Signal Processing*, 48(9): 2487-2494.

Gwo-Ching, C., Wen-Juh, K., Jer-Junn, L., Cheng-Kung, C., Jin-Hae-Jeong, P., Sung-Hoon, K., Hee-Chan K., Kwang-Suk, P., 1999. Adaptive EMG-driven communication for the disabled. *Proceedings of Engineering in Medicine and Biology, 21st Annual Conference and of the Bio-medical Engineering Society Conference*, 13-16 October, Atlanta, GA.

Hefftner, G., Zucchini, W., Jaros, G.G., 1988. The electromyogram (EMG) as a control signal for functional neuromuscular stimulation. I. Autoregressive modeling as a means of EMG signature discrimination. *IEEE Transactions on Biomedical Engineering*, 35(4): 230-237.

Henneberg, K.A., Plonsey, R., 1993. Boundary element analysis of the directional sensitivity of the concentric EMG electrode. *IEEE Transactions on Bio-medical Engineering*, 40(7), 621-631.

Hershler, C., Morris M., 1978. An optimality criterion for processing electromyographic (EMG) signals relating to human locomotion. *Biomedical Engineering, IEEE Transactions*, pp. 413-420.

Hiraiwa, A., Yukio Tokunaga, K.S., 1989. EMG Pattern Analy-sis and Classification by Neural Network. *Conference Proceedings of Systems, Man and Cybernetics International Conference on IEEE*, pp. 1113-1115.

Hultman, E., Sjöholm, H., 1983. Electromyogram, force and relaxation time during and after continuous electrical stimulation of human skeletal muscle in situ. *The Journal of Physiology*, 33-40.

Ito, K., Tsuji, T., Kato, A., Ito, M., 1992. An EMG controlled prosthetic forearm in three degrees of freedom using ultrasonic motors. *14th Annual International Conference of the IEEE on Engineering in Medicine and Biology Society*, Oct 29-Nov 1, Paris, pp.1487-1488.

Jingdong, Z., Zongwu, X., Li, J., Hegao, C., Hong, L., Hirzinger, G., 2006. EMG control for a five-fingered prosthetic hand based on wavelet trans-form and autoregressive model. *Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation*, 25-28 June, Luoyang, Henan, pp.1097-1102.

Jun-Uk C., Inhyuk, M., Yun-Jung, L., Shin-Ki, K., Mu-seong, M., 2007. A supervised feature-projection-based real-time EMG pattern recognition for multifunction myoelectric hand control. IEEE/ASME Transactions on-Mechatronics.

Kamen, G., Caldwell Graham E., 1996. Physiology and ınterpretation of the electromyogram. *Journal of Clinical Neurophysiology*, 13(5): 366-384.

Kang, W.J., Cheng, C.K., Lai, J.S., Shiu, J.R., Kuo, T.S., 1996. A comparative analysis of various EMG pattern recognition methods. *Medical Engineering & Physics*, 18(5): 390–395.

Kermani, M.Z., Tehran, Iran., Badie, K., 1989. An intelligent strategy for motion interpretation in an EMG-controlled prosthesis. *International Conference of the IEEE Engineering in Engineering in Medicine and Biology Society*, 9-12 November, 5: 1682-1683.

Kreifeldt John, G., 1971. Signal versus noise characteristics of filtered EMG used as a control source. *Biomedical Engineering, IEEE Transactions,* 16-22.

Kurt, F., 2018. Evrişimli sinir ağlarında hiper parametrelerin etkisinin incelenmesi Yüksek Lisans Tezi, Hacettepe Üniversitesi, Fen Bilimleri Enstitüsü. Ankara, Türkiye.

Kutlu, H., 2018. Biyoistatistik temelli bilimsel araştırmalarda derin öğrenme uygulamaları. Yüksek Lisans Tezi, Yakındoğu Üniversitesi, Sağlık Bilimleri Enstitüsü, Lefkoşa, Kıbrıs.

Lee, S., Sankai, Y., 2002. Power assist control for walking aid with HAL-3 based on EMG and impedance adjustment around knee joint. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2: 1499-1504.

Li, P., 2017. Optimization algorithms for deep learning. department of systems engineering and engineering management.

Li-Zhi, L., Yi-Li, T., Hsin-Han, C., 2018. EMG-based control scheme with svm classifier for assistive robot arm. *International Automatic Control Conference* (CACS), pp.1-5.

Maksutov, R., 2018. Deep study of a not very deep neural network. part 3b: choosing an optimizer. (https://medium.com/@maksutov.rn/deep-study-of-a-not-very-deep-neural-network-part-3b-choosing-an-optimizer-de8965aaf1ff), (Accessed: 05.01.2025).

Manabe, H., Zhang, Z, 2004. Multi-stream HMM for EMG-based speech recognition. *26th Annual International Conference of the IEEE engineering in Medi-cine and Biology Society*, San Francisco, pp.4389-4392.

Matt, F., Somesh, J., Thomas, R., 2015. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In ACM SIGSAC Conference on Computer and Communications Security.

Mesin, L., Farina, D., 2005. A model for surface EMG generation in volume conductors with spherical inhomogeneities. *IEEE Transactions of Biomedical Engineering*, 52(12): 1984-1993.

Moradi, M., Hashtrudi-Zaad, K., Mountjoy, K., and Morin, E., 2008. An EMG-based force control system for prosthetic arms. *Canadian Conference on Electrical and Computer Engineering*, pp.1737-1742.

Morita, S., Kondo, T., Ito, K., 2001. Estimation of forearm movement from EMG signal and application to prosthetic hand control. *Proceedings of IEEE International Conference on Robotics and Automation*, pp.3692-3697.

Nikolic, Z.M., Popovic, D.B., Stein, R.B., Kenwell, Z., 1994. Instrumentation for ENG and EMG recordings in FES systems. *IEEE Transactions on Biomedical Engineering*, 41(7): 706.

Nishikawa, D., Wenwei, Yu., Yokoi, H., Kakazu, Y., 1999. EMG prosthetic hand controller using real-time learning method. *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, 12-15 October, Tokyo, pp.153-158.

Nishikawa, D., Wenwei, Yu., Yokoi, H., Kakazu, Y., 1999. EMG prosthetic hand controller discriminating ten motions using real-time learning method. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Kyongju, pp.1592-1597.

Okut, H., 2018. Machine learning methods for big data and genomic selection: r application. *Workshop Material, International Agricultural Congress*, 8 May, Van, Turkey.

Okut, H., Gianola, D., Rosa, G.J.M., Weigel, K.A., 2011. Prediction of body mass index in mice using dense molecular markers and a regularized neural network. *Genetics Research*, 93(3): 189-201.

Ouyang, G., Zhu, X., Ju, Z., Liu, H., 2013. Dynamical characteristics of surface EMG signals of hand grasps via recurrence plot. *IEEE Journal of Biomedical and Health Informatics*, 257-265.

Paiss, Omry., Inbar, G.F., 1987. Autoregressive modeling of surface EMG and ıts spectrum with application to fatigue. *IEEE Transactions on Biomedical Engineering*, 34(10): 761-770.

Paul, G.M., Fan, Cao., Torah, R., Kai, Yang., Beeby, S., Tudor, J., 2014. A smart textile based facial EMG and EOG computer ınterface. *Sensors Journal*, 14(2): 393-400.

Priddy, K.L., Keller, P.E., 2005. Artificial neural network: An Introduction, 1st. Ed., Spie Press, Washington.

Reucher, H., Silny, J., Rau, G., 1987. Spatial filtering of noninvasivemultielec-trode EMG: Part II-filter performance in theory and modeling. *IEEE Transactions on Biomedical Engineering*, 34(2): 106-113.

Ruder, S., 2016. An overview of gradient descent optimization algorithms. (http://adsabs.harvard.edu/abs/2016arXiv160904747R), (Accessed: 05.01.2025).

Sammut, C., Webb, G.I., 2015. Encyclopedia of machine learning and data mining, Springer.

Sang-Hui, P., Seok P., 1998. EMG pattern recognition based on artificial ıntelligence techniques. *IEEE Transactions On Rehabilitation Engineering*, 6(4).

Saridis George N., Thomas, P., 1982. EMG pattern analysis and classification for a prosthetic arm. *Biomedical Engineering, IEEE Transactions*, 6: 403-412.

Sharma, A., 2017. Understanding activation functions in neural networks. (https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0), (Accessed: 05.01.2025).

Shin, L., Jia-Jin, J., Chen Te-Son, K., 1996. Real-time implementation of electromyogram pattern recognition as a control command of man-machine interface. *Medical Engineering and Physics*, 18(7): 529-535.

Song, S., Chaudhuri K., Sarwate, A.D., 2013. Stochastic gradient descent with differentially private updates. *IEEE Global Conference on Signal and Information Processing*, Austin, pp. 245-248.

Stefan Karlsson, J., Karin, R., Christer G., Andreas, H., Nils Ö., 1887. Signal processing of the surface electromyogram to gain insight into neuromuscular physiology. Medical Engineering and Physics, Elsevier.

Studer, R.M., de Figueiredo Rui J.P., Moschytz George S., 1984. An algorithm for sequential signal estimation and system ıdentification for emg signals. *IEEE Transactions on Biomedical Engineering*, 31(3): 285-295.

Su, H., Nelder, J.A., Spence, R., Ismail, M., 1994. Generalized linear models for empirical performance modeling in circuit design. *Proceedings of APCCAS'94-Asia Pacific Conference on Circuits and Systems*.

Şengöz, N., 2017. Yapay sinir ağları. (http://www.derinogrenme.com/author/nilgunsengoz/), (Accessed: 08.01.2025).

Tsuji, T., Shigeyoshi, H., Kaneko, M., 2000. Bio-mimetic impedance control of an EMG-controlled prosthetic hand. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 31 October – 5 November, Takamatsu, pp.377-382.

Vuckovic, M., Sijiang, D., 2002. Classification of prehensile EMG patterns with simplified fuzzy ARTMAP networks. *Proceedings of the 2002 International Joint Conference on Neural Networks*, 12-17 May, Honolulu, HI. pp.2539-2544.

Winter, D.A., Yack, H.J., 1987. EMG profiles during normal human walking: stride-to-stride and inter-subject variability. *Electroencephalography and Clinical Neurophysiology*, 67: 402-411.

Xiaowen, Z., Yupu, Y., Xiaoming, X., Ming Z., 2002. Wavelet-based neuro-fuzzy classification for EMG control. *IEEE 2002 International Conference on Communications, Circuits, and Systems and West Sino Expositions*, 29 June- 1 July, pp.1087-1089.

Yitong, Z., Chellappa, R., Bekey, G., 1986. Estimation of intramuscular EMG signals from surface EMG signal analysis. *Acoustics, Speech, and Signal Pro-cessing, IEEE International Conference on ICASSP*, pp.1805-1808.

Zahedi, E., Farahani, H., 1995. Graphical simulation of artificial hand motion with fuzzy EMG pattern recognition. *14th Conference of the Biomedical Engineering Society of India an International Meeting*.

Zardoshti-Kermani, M., Badie, K., Hashemi, R.M., 1995. EMG feature evaluation for movement control of upper extremity prostheses. *IEEE Transactions on Rehabilitation Engineering*, 3(4): 324-333.

Zennaro, D., Wellig, P., Koch, V.M., Moschytz George, S., Laubli, T., 2003. A software package for the decomposition of long-term multichannel EMG signals using wavelet coefficients. *IEEE Transactions on Biomedical Engineering*, 50(1): 58-69.

To Cite: Özer, Ç., Orman, Z., 2025. Enhancing EMG Signals for Amputee People with Deep Neural Network and Optimization Algorithms. *MAS Journal of Applied Sciences*, 10(1): 141-160.
DOI: http://dx.doi.org/10.5281/zenodo.15099262.