

EMG Signal Classification Based on Light-Weight Neural Network with FPGAs for Wearable Application

Hyun-Sik Choi ^{1,*}

¹ Department of Electronic Engineering, College of IT Convergence Engineering, Chosun University, Gwangju, South Korea; hs22.choi@chosun.ac.kr

* Correspondence: hs22.choi@chosun.ac.kr

Abstract: Recently, the application of bio-signals to health management, human-computer interaction (HCI), and user authentication is increasing. This is due to the development of artificial intelligence technology that can analyze bio-signals in many fields. In the case of bio-signal, there is a tendency for the results to vary depending on the analysts due to the large amount of noise. However, when neural network is used, extraction of the features is possible, enabling more accurate analysis. However, if the time series of bio-signal is analyzed as it is, the total neural network becomes large. In this study, to accomplish light-weight neural network, maximal overlap discrete wavelet transform (MODWT) and the smoothing technique are used for better feature extraction. In addition, the learning efficiency is increased by using the augmentation technique. In the case of neural network design, a 1-dimensional convolution layer is used so that the neural network is simple while using light-weight. Through this, light-weight can be implemented, and neural network can be implemented in edge devices such as field programmable gate array (FPGA) with low power consumption, high security, fast response time, and high user convenience for wearable application. Electromyogram (EMG) signal is used as a representative bio-signal in this study.

Keywords: bio-signals; artificial intelligence technology; light-weight neural network; maximal overlap discrete wavelet transform (MODWT); smoothing technique; augmentation technique; edge devices; field programmable gate array (FPGA); electromyogram (EMG).

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Electronics* **2023**, *11*, x. <https://doi.org/10.3390/xxxxx>

Academic Editor: Firstname Last-name

Received: date

Accepted: date

Published: date

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2023 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

There are various bio-signals such as electrocardiogram (ECG), electromyogram (EMG), photo-plethysmography (PPG), and electroencephalogram (EEG). Many artificial intelligence networks using these bio-signals are being developed, and the field of application is also diverse [1,2]. Typical application fields are health care, human-computer interaction (HCI), and user authentication [3-6]. In this paper, EMG signal is dealt as a representative bio-signals. In the case of EMG signal, measurement is simple, so it is expected to be used in fields such as user authentication and HCI [7,8]. EMG is measured by amplifying a small signal coming from the movement of the muscle, and the measurement equipment have three input terminals [9]. The two terminals are used to measure the signal in the middle and end of the muscle. The last terminal is used as a reference signal which is far from the muscle. This is amplified by the differential amplification unit to determine muscle movement. EMG is easy to attach, and it can be used for HCI by attaching it to the most active hand easily. For HCI application of EMG signal, classification using artificial intelligence was studied in many researches [10,11]. This bio-signal utilization application is due to the rapid progress of artificial intelligence and big data technology.

Basically, in the case of EMG signals, like other bio-signals, various noise components are included in the measurement [12]. For this reason, there was a problem that the analysis of the results differs depending on the analysts. However, when the neural network is used, feature extraction is easily possible and more accurate and consistent results can be derived [13]. In [14], based on the EMG signals measured in 16 channels, support vector machine (SVM) and generalized regression neural network (GRNN) artificial intelligence algorithms were used using root mean square (RMS), autoregressive model (AR), and slope sign change (SSC) corresponding to 6 representative time series analysis. Through this, the six hand gestures were classified with 98% accuracy. In [15], a user recognition study was conducted using the EMG signal measured when drawing the unlock pattern on the smartphone screen. EMG signal was measured in Flexor Digitorum Superficialis (FDS) muscle of the forearm using OpenBCI. Using one-class SVM (OCSVM), extracting features such as mean absolute value (MAV), variance (VAR), and  in the time domain, the user was recognized as 98.2%. In [16], fast Fourier transform (FFT) was performed using the EMG signal, and based on this, it was confirmed that diseases such as neuropathy muscle disease could be predicted. However, if the time series is used as it is or FFT is used, the neural network for feature extraction becomes very large, and the weight is also used more than 100,000, making it unsuitable for wearable applications [17,18]. For this heavy EMG signal analysis, most of them are implemented in a structure that transmits data to the server and analyzes it in the server. In this case, a lot of hardware resources are used in server, and power consumption is large. In addition, in this case, security is emerging as a big problem, and in the case of authentication, problems such as slow response time may occur [19,20].

To solve this problem, the field of bio-signal processing in edge devices is receiving great attention [21–23]. This is because, when using an edge device, low power, fast response speed, and security can be expected [24]. However, there is a problem in that it is difficult to achieve low power in the case of CPU and GPU based artificial intelligence system. This is because it is a clock-based operation and consumes a lot of power in the process of accessing the memory. To prevent this, field programmable gate array (FPGA) having a distributed structure can be an alternative, and low power and fast response time can be secured through various structure optimizations [25,26]. In the case of implementing an edge device with CPU/GPU or an embedded system, only one execution is performed, whereas in the case of an FPGA, it has the advantage of executing multiple instructions in parallel, so it is receiving great attention in the edge device [27,28].

In the case of an inference accelerator using FPGA, many research studies are being conducted [29]. However, in this case, an external memory such as dynamic random access memory (DRAM) is used to store the weight values, and in this case, a lot of energy is consumed for reading memory processes. To prevent this, more compressed edge devices are being studied, and this paper also focuses on this part. This paper focuses on hardware configuration that can be applied in real life. The hardware compression methods are largely based on 1) compression of the model network itself using algorithms, 2) compression of computation methods (e. g. MobileNet [30], systolic array of tensor processing unit (TPU) [31]), and 3) pruning method using weight sparsity and bit quantization [32]. In this paper, hardware compression is carried out by focusing on method 1) and 3). Therefore, the main purpose is to design a light-weight neural network suitable for edge device such as FPGA capable of parallel operation and low memory access. Therefore, the memory components for storing weight values can be exist inside FPGA without external DRAM memory access (e. g. ResNet minimum storage requirement for 26 million weight values: 104 MB).

For this purpose, a frequency-filtered signal is used by a signal processing. The maximal overlap discrete wavelet transform (MODWT) and the smoothing algorithm for this signal are used. In addition, augmentation technique is used to increase the learning efficiency. For neural network design, 1-dimensional convolution layer is used for light-weight system suitable for wearable application. The neural network implemented in this

way is deployed to FPGA, which is an edge device, and performs appropriate inference operation. Through this, it is confirmed that it is possible to implement a low power, high speed, and high security system with an edge device even in low-cost FPGAs for artificial intelligence. In other words, it is possible to reduce the size of the model network by using an efficient feature extraction method which is suitable to EMG signals, and through this, it is possible to operate with high accuracy and low power consumption without access to external memory. In addition, bit quantization is performed to further secure the compression of the edge device. The main application area is wearable system for HCI or user authentication. Section 2 of this paper deals with the signal processing of EMG signals, and Section 3 deals with the structure of the neural network. Section 4 shows the overall hardware structure and the verification process through high level synthesis (HLS) for hardware deployment. Section 5 deals with conclusions, discussions, and future research directions.

2. Signal processing

2.1. Data Augmentaion

For the EMG signal input, “sEMG for Basic Hand movements Data Set” of the existing UCI machine learning repository was used [33]. This is the measurement data for 5 people, and the hand gestures are composed of 6, so the final goal is to classify the 6 movement by the measured EMG signals. The measurement sampling frequency is 500 Hz, and measurement was performed using two channels. The reason why the sampling frequency is 500 Hz is that most important data in the case of EMG signals are included in the range of 10 to 250 Hz. However, since a wearable device is assumed in this paper, classification was performed using only data from channel 1. This causes a decrease in accuracy, but since 1 channel is mostly used in the actual HCI environment, 1 channel is used. In a real environment, it would be difficult to distinguish 6 hand gestures with 1 channel data. In actual wearable device application, classification for about 2 to 3 classes is considered. Both the EMG sensor and artificial intelligence will be produced in the form of a smart watch. Actual application fields through this will be HCI implementation for two or three hand motions, analysis of carpal tunnel syndrome, and user authentication. However, for comparison with other neural networks, the accuracy was obtained by performing the classification of all 6 hand gestures. In addition, in the case of measurement data, the number of data is insufficient because it is data for 5 people, but it was used only for the feasibility test of the proposed neural network. The dataset “EMG data for gestures Data Set” of 36 subjects and 8 channels for 6 hand gestures is also verified with the similar sequences. It shows 98 % classification accuracy for validation data (not shown here). The increased accuracy is related to the increased channel numbers [34]. In the future, the EMG signal will be measured by own compact EMG modules and it will be used as wearable device. A high-efficiency wearable device for EMG signal acquisition is currently under development.

In the case of the EMG input dataset, since the number of datasets is small, it is not suitable for training, so data augmentation was implemented using additive noise and magnitude warping techniques. In the case of additive noise, Gaussian noise was added. The entire signal was processed after normalization. The equations below describe how to create a new signal by adding additive Gaussian noise [35].

$$x_i^* = x_i + n \quad (1)$$

$$n \sim \text{Gaussian}(\mu = 0, \sigma = \sqrt{\frac{x_i^2}{\text{SNR}}}) \quad (2)$$

where, x_i is the original signal, and x_i^* is the augmented signal. n is the additive noise component, and Gaussian noise is randomly added according to the signal-to-noise ratio (SNR). Figure 1 shows a representative EMG signal and an augmentation signal using additive noise method.

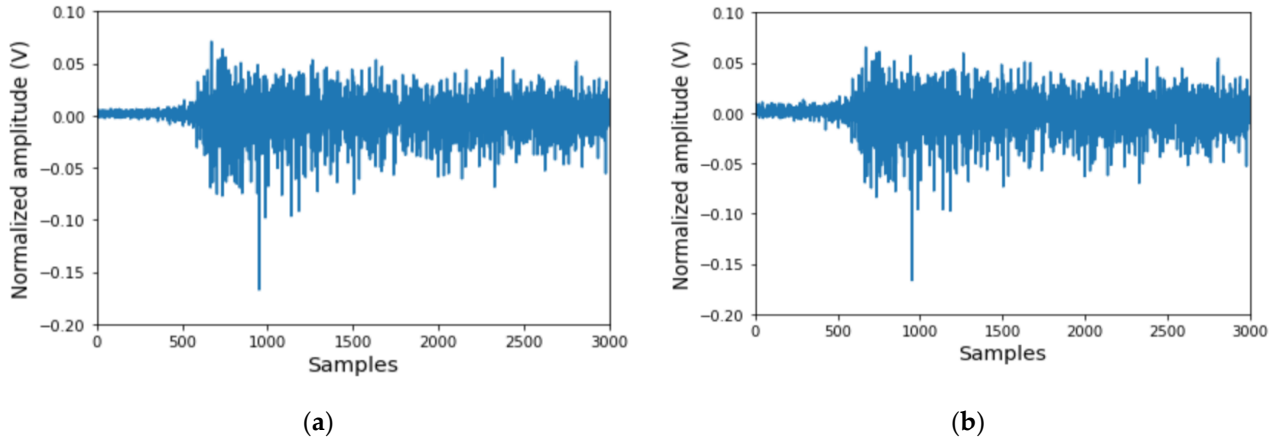


Figure 1. Representative (a) EMG signal, (b) augmentation signal using additive noise of “sEMG for Basic Hand movements Data Set”.

Additionally, moving average, permutation, and magnitude warping techniques are possible for data augmentation. In this paper, only magnitude warping technique, which is known to be the most efficient, was used [35]. As a result of actual verification, the improvement in accuracy due to the moving average technique was insignificant, and in the case of the permutation technique, the accuracy is rather reduced. The magnitude warping method is shown in the equations below.

$$x_i^* = x_i \cdot \text{CubicSpline}(r) \quad (3)$$

$$r = \{r(t_1), r(t_2), \dots, r(t_T)\} \quad (4)$$

where t means sampling time. A random curve is generated using CubicSpline method, which is an interpolated curve, and warping is performed on the magnitude. Through this augmentation method, training was performed by increasing the size of the dataset by about 3 times. This resulted in an increase in classification accuracy of about 3 % compared to when the data augmentation method was not used.

2.2. Filtering and Smoothing

Additionally, EMG signals were completely removed 50 Hz and 0 Hz signals using an additional filter. They are basically signals that have passed through the band pass filter, but when the signal analysis is performed through FFT, DC and 50 Hz components still exist, so 50 Hz and 0 Hz signals were additionally removed using a high-performance additional filter for signal processing. The first filter is an active second-order high pass filter with cutoff frequency (f_c) of 10 Hz. The second filter is an active notch filter with f_c of 50 Hz. These are designed for easy hardware implementation for EMG sensor system. The active notch filter will be designed using active twin-T notch filter method [36]. Figure 2 shows a representative signal before and after the additional filters. It is displayed as a frequency spectrum for easy comparison.

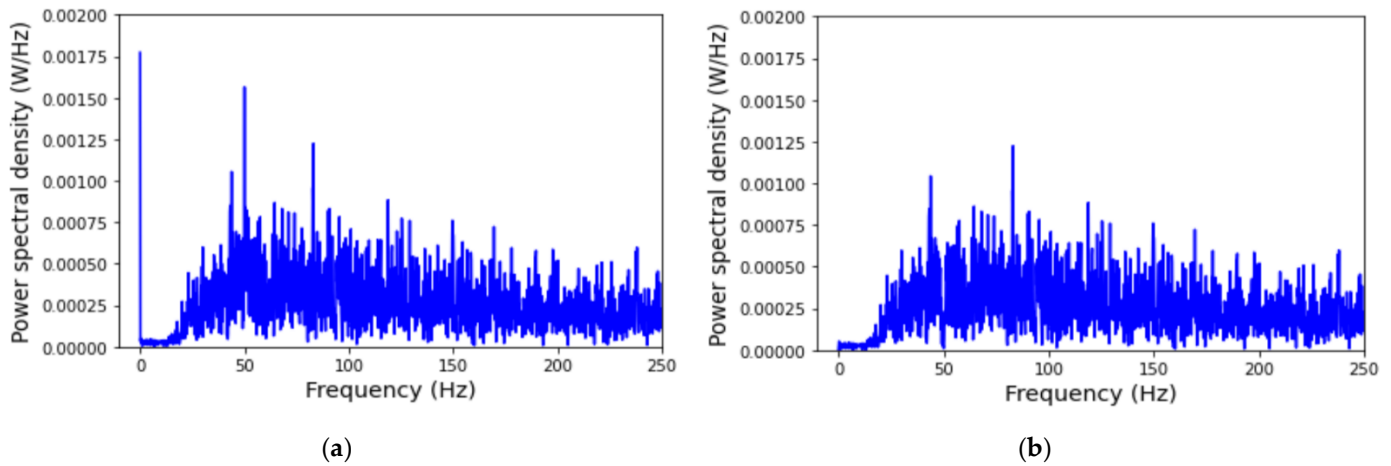


Figure 2. Frequency spectrum of EMG signal (a) before use of additional filters, (b) after use of additional filters.

In addition to filtering method for time series data, it is possible to reduce the effective noise components through smoothing technique. In this study, Savitzky-Golay filter was used for the smoothing technique. Although several smoothing techniques exist, the Savitzky-Golay method was used because it is easy for a simple hardware implementation. The Savitzky-Golay filter can replace smoothing using a polynomial regression model mathematically accurately by providing a specific impulse response without calculating the regression model within the window of every time step in performing smoothing using a regression model. The time window was 42 ms, and a cubic equation was used for polynomial regression. Through filtering method and smoothing technique, it was possible to bring about an accuracy improvement of about 2 %.

2.3. Feature extraction

For feature extraction of EMG signal, the MODWT method was used for easy implementation in the FPGA. Wavelet transform was born to simultaneously perform time and frequency domain analysis. The wavelet transform has the advantage of being able to deal with information in the time domain, instead of sacrificing some accuracy in the frequency domain. Among them, discrete wavelet transform (DWT) based on orthonormal wavelet is used a lot, but MODWT is more sensitive to circular shift than general DWT. It can be interpreted as a linear filter result. In hardware implementation, the finite impulse response (FIR) filter will be used using digital logic. Through the above results, it can be said that MODWT, which is easy to interpret in combination with events that actually occurred in nature, is useful in time series analysis. Also, unlike the DWT, MODWT is defined naturally for all sample sizes. The MODWT method is expanded while preserving the size of the data. Due to MODWT method, feature extraction with light-weight can be realized. If the time-series are directly analyzed, the size of the entire neural network for feature extraction increases, whereas in MODWT method, similar performance can be secured even with a small neural network. The basic implementation algorithm of MODWT method is as follows. For a time series X with arbitrary sample size N , the j th level MODWT wavelet (W_j^*) and scaling (V_j^*) coefficients are defined as follows [37].

$$W_{j,t}^* = \sum_{l=0}^{L_j-1} h_{j,l}^* X_{t-l \bmod N} \quad (5)$$

$$V_{j,t}^* = \sum_{l=0}^{L_j-1} g_{j,l}^* X_{t-l \bmod N} \quad (6)$$

where $h_{j,l}^* = h_{j,l}/2^{j/2}$ are the MODWT wavelet filters, and $g_{j,l}^* = g_{j,l}/2^{j/2}$ are the MODWT scaling filters.

In this study, the used filter is the second order Daubechies filter (db2), and it was confirmed that the original signal was restored by using inverse MODWT. In addition to db2, filters that can be used in MODWT method include the first order Daubechies filter (db1), the forth order Daubechies filter (db4), Haar filter, Symlets filter, and Coiflets filter [38]. Except for Haar filter and db1 filter, all showed similar excellent performance. In addition, the de-composition level was selected as 4 because the performance increased up to level 4, but the performance decreased by about 2 % when the level was 5 or higher. This is related to the frequency domain being decomposed. For example, the frequency spectrum of level 1 MODWT signal is between 125 Hz and 250 Hz.

The results of MODWT with the decomposition level of 4 is shown in Figure 3. Like the artificial intelligence network, the MODWT module will also be implemented inside the FPGA using digital logic. The results of each MODWT are given as inputs to the artificial intelligence network.

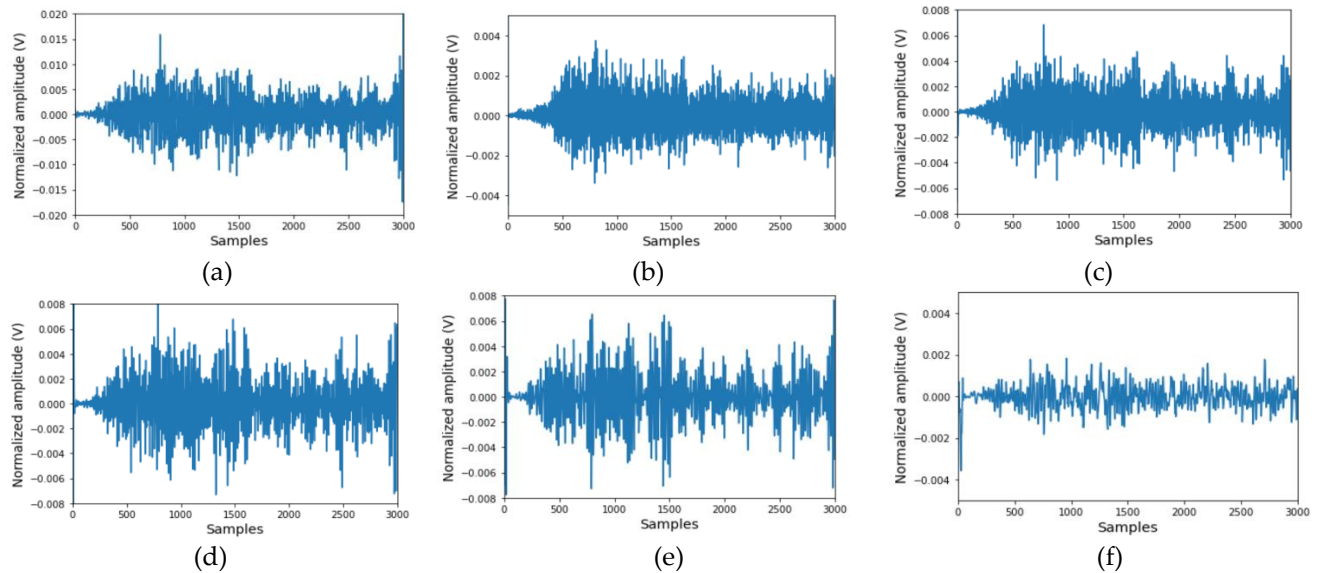


Figure 3. MODWT execution result. (a) Original signal, (b) Level 1, (c) Level 2, (d) Level 3, (e) Level 4, (f) Residual.

3. Artificial Intelligence Network

The MODWT signal for feature extraction has 5 channels and is given as input to 1-dimensional convolution layer as shown in Figure 4. The three 1-dimensional convolution layer is used. The 1-dimensional convolution layer is suitable for realizing compressed neural network because the amount of computation is smaller than that of 2-dimensional convolution layer. To implement light-weight, while keeping neural network as simple as possible, the accuracy was aimed at above 93 %. For this purpose, verification of various neural networks was performed, and among them, neural network with 93 % or more accuracy was selected while implementing light-weight.

Activation function of the 1-dimensional convolution layer uses the rectified linear unit (ReLU), and the padding is not used. Kernel for convolution layer uses L1 regulation and the used parameter is 0.001. Three 1-dimensional convolution layers are used and the number of channels is optimized. The max-pooling layer after the convolution layer is used to reduce the number of features. In the case of the fully connected layer (dense layer), the minimum features were used as input, and sigmoid function and softmax function were used for the activation functions. These were implemented in the form of look-up table when implementing these to hardware resources in Section 4. Because the dense

layer uses many parameters, the structure is implemented as concise as possible. The focus in this case is to ensure high accuracy with few parameters and a simple structure.

Model: "model_14"

Layer (type)	Output Shape	Param #
input_15 (InputLayer)	[(None, 3000, 5)]	0
conv1d_50 (Conv1D)	(None, 2998, 32)	480
batch_normalization_50 (Batch Normalization)	(None, 2998, 32)	128
activation_50 (Activation)	(None, 2998, 32)	0
max_pooling1d_49 (MaxPooling1D)	(None, 999, 32)	0
conv1d_51 (Conv1D)	(None, 498, 16)	2560
batch_normalization_51 (Batch Normalization)	(None, 498, 16)	64
activation_51 (Activation)	(None, 498, 16)	0
max_pooling1d_50 (MaxPooling1D)	(None, 166, 16)	0
conv1d_52 (Conv1D)	(None, 82, 16)	1024
batch_normalization_52 (Batch Normalization)	(None, 82, 16)	64
activation_52 (Activation)	(None, 82, 16)	0
max_pooling1d_51 (MaxPooling1D)	(None, 20, 16)	0
flatten_14 (Flatten)	(None, 320)	0
dense_28 (Dense)	(None, 12)	3852
dense_29 (Dense)	(None, 6)	78

=====
 Total params: 8,250
 Trainable params: 8,122
 Non-trainable params: 128

Figure 4. The structure and parameters for artificial intelligence network

In this case, the adaptive moment estimation (adam) optimizer was used for training, and the categorical cross-entropy function was used as the loss function. In addition, learning process was performed using the validation data of 30 % of the total training data. The learning rate (lr), a representative hyper parameter, used step decay function, and the lr change factor was set to 0.5. The initial lr was set to 0.0001. The minimum value of lr was set to 10^{-7} . In this way, the appropriate learning rate was adjusted. The total trainable parameters of the designed neural network are 8,122. This corresponds to the light-weight neural network, and 96 % accuracy for the validation data was possible with the help of algorithms such as MODWT, data augmentation, and smoothing technique. Through this, the implementation of light-weight neural network that can be operated with a small amount of resources even when implemented as an edge device was achieved. The accuracy and loss of training data and validation data according to the increase of training epochs are shown in Figure 5, respectively. The accuracy of training data and validation data is about 99 % and 96 %, respectively.

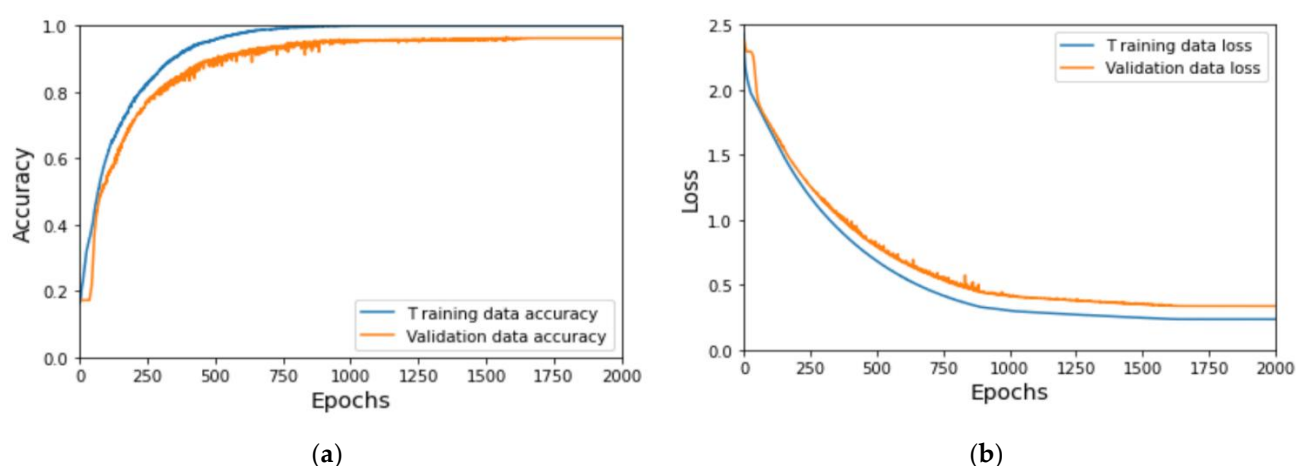


Figure 5. (a) Accuracy of training data and validation data, (b) Loss of training data and validation data.

In case of new data, test data, the confusion matrix is shown in Figure 6. The accuracy for test data is 95 % after training. Although this value has a relatively low accuracy, it is high considering that it is implemented using a small amount of hardware resources. In the case of the other light-weight neural network example, the accuracy of “jet tagging” is about 74 % [39]. By using MODWT, feature extraction is performed efficiently and high accuracy can be secured with few parameters. In Figure 6, “spher” represents the action for holding spherical tools, “tip” for holding small tools, “palm” for grasping with palm facing the object, “lat” for holding thin, flat objects, “cyl” for holding cylindrical tools, and “hook” for supporting a heavy load.

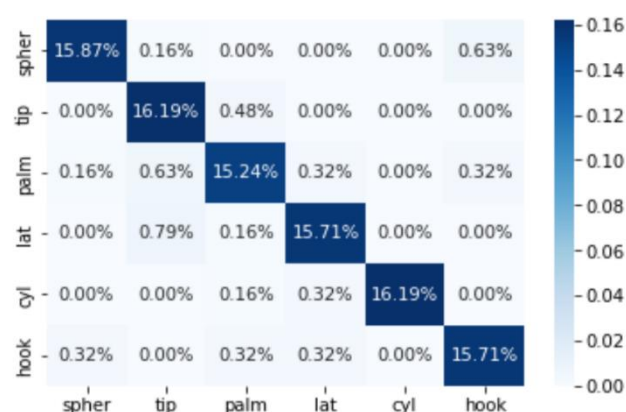


Figure 6. Confusion matrix for test data.

Finally, Figure 7 shows receiver operating characteristic (ROC) curve and area under curve (AUC). ROC curve is often used to evaluate the performance of a model that distinguishes classes, and there is false positive rate (FPR) on the x-axis and true positive rate (TPR) on the y-axis. Through this, it is possible to represent various performance indices. Currently, the AUC is over 99 % for test data.

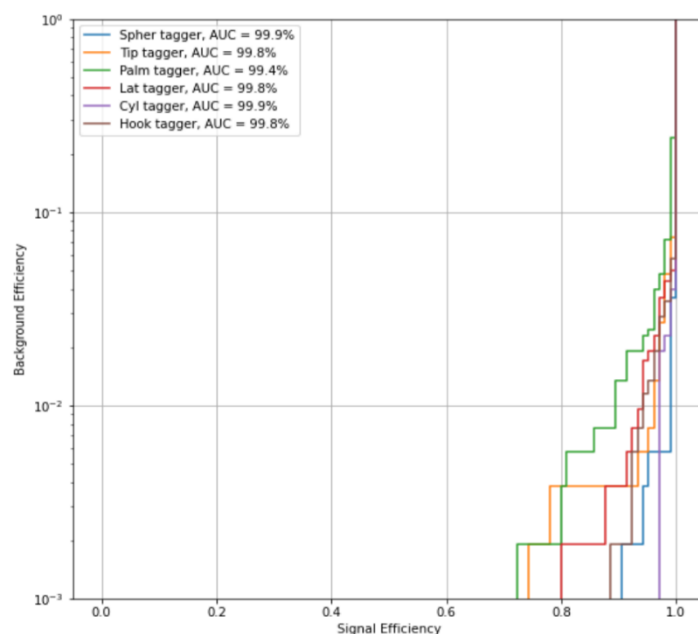


Figure 7. ROC curve for test data (6 categories).

4. Hardware deployment

Using the light-weight neural network model created by software using Keras tool, it is converted to Verilog hardware description language (HDL) using HLS [40]. Verilog HDL is changed to NAND gates, NOR gates, etc. through synthesis to create digital logic. HLS is a method that automatically converts software code to Verilog HDL when it is written in C/C++, which the user is familiar with. Because it takes a long time for large logic to be written directly in a language such as Verilog HDL, HLS can greatly reduce hardware development time and increase user convenience. In addition, if the user uses a command such as “#pragma HLS PIPELINE”, which is a separate grammar in HLS, the parallel computation of the FPGA can be accomplished. Therefore, the latency reduction in FPGA can be obtained. In such way, the structure optimization is performed together in HLS. Figure 8 shows the HLS code for 1-dimensional convolution layer written in C/C++, representatively. The 1-dimensional convolution layer proposed by Keras is configured using C/C++ and converted to Verilog HDL using HLS. The main algorithm is the process of adding the convolution operation of the input and the filter again after storing the values of the bias in the buffer. Each layer of batch normalization layer, max-pooling layer, activation layer, and dense layer was implemented using HLS to be similar to the proposed neural network by Keras. In the case of sigmoid and softmax functions, the amount of exponential calculation is large, so it is implemented in the form of a look-up table. Similar to the above method, there are several hardware implementation methods such as Vivado SDAccel and NVIDIA Deep Learning Accelerator (NVDLA). The core part has the advantage of directly converting a neural network written in Caffe or the like into register transfer level (RTL). In this study, it was confirmed that the artificial intelligence part for EMG signal classification in real life could be implemented in the edge device. In this study, it was designed to have light weight parameters through neural network design, which resulted in a reduction of hardware resources, and additional resource reduction was performed through additional bit optimization. The designed digital logic can be easily changed to application specific integrated circuits (ASICs).

```

void conv1d(DTYPE in[NUM_INCHAN][IN_ROWS],
            const DTYPE filt[NUM_OUTCHAN][NUM_INCHAN][KSIZE],
            const DTYPE bias[NUM_OUTCHAN],
            DTYPE out[NUM_OUTCHAN][OUT_ROWS]){
    OFM: for(int ofm=0; ofm<NUM_OUTCHAN; ofm+=2){
        ROW_CLR: for(int r=0; r<OUT_ROWS; r++){
#pragma HLS PIPELINE
            acc_buf_0[r]=bias[ofm]; acc_buf_1[r]=bias[ofm+1];
            IFM: for(int ifm=0; ifm<NUM_INCHAN; ifm++){
                ROW: for(int r=0; r<OUT_ROWS; r++){
#pragma HLS PIPELINE
                    acc_0=0; acc_1=0;
                    K_ROW: for(int k_r=0; k_r<KSIZE; k_r+=STRIDE){
                        acc_0 += filt[ofm][ifm][k_r]*in[ifm][r + k_r];
                        acc_1 += filt[ofm+1][ifm][k_r]*in[ifm][r + k_r];
                    }
                    acc_buf_0[r] += acc_0; acc_buf_1[r] += acc_1;
                }
            }
            ROW_COPY: for(int r=0; r<OUT_ROWS; r++){
#pragma HLS PIPELINE
                out[ofm][r] = acc_buf_0[r];
                out[ofm+1][r] = acc_buf_1[r];
            }
        }
    }
}

```

Figure 8. HLS code for 1-dimensional convolution layer.

The advantage of hardware deployment is that the number system can be freely used. Because the hardware is designed by ourselves, the number system used inside can be defined and used, and through this, the hardware resource can be reduced [41]. In case of using floating point number used in software in Section 3, it is possible to obtain 95 % accuracy for the test data using the hardware resource. This is the same accuracy result as the software. However, the number of bits used for resource reduction was reduced to be suitable for wearable devices. In this case, bit optimization was performed. In the case of a fixed point number, an error occurs unlike Keras, and a large difference is shown depending on the number of bits used. In FPGA, the size of the fixed point number is determined through the “ap_fixed” keyword, and ap_fixed<16, 6> is the basic configuration for FPGA. That is, the entire bit becomes 16 bits, the integer is 6 bits including the sign bit, and 10 bits representing the value below the decimal point. In this case, if ap_fixed<24, 6> is used, it can be implemented with a small amount of hardware resources without degradation of accuracy. However, when ap_fixed<22, 6> is used, 80 % of the hardware resources are used compared to ap_fixed<24, 6>, so it was judged as the optimal structure. In this case, the difference in accuracy is just about 1 %. However, if the number of bits is further reduced and the hardware is configured as ap_fixed<20, 6>, the accuracy is reduced to 86 %. The distribution of weights actually used to optimize the bit was examined through Figure 9. The used total bits for optimal structure are 22 bits. Bit optimization can also be performed using QKeras. QKeras is a quantization extension to Keras.

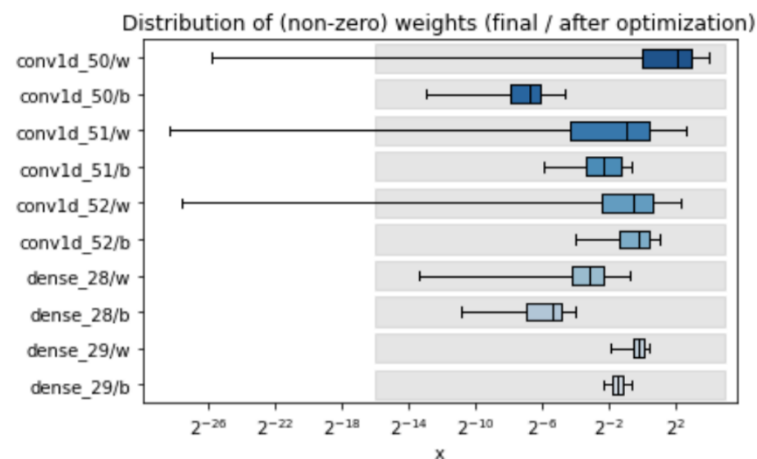


Figure 9. Bit selection using numerical profiling.

The network is implemented using HLS using $ap_fixed<22, 6>$. The accuracy of the test data is about 94 %, and the AUC can be secured more than 99 %. If the basic $ap_fixed<16, 6>$ is used, the accuracy is reduced to about 18 %. The AUC becomes more than 45 %. There is no change in accuracy when using floating point numbers or $ap_fixed<24, 6>$. In addition, the reuse factor from 10 to 20 was used to implement a structure that reuses resources as much as possible. Figure 10 shows the ROC curve in the digital logic implemented in the FPGA with $ap_fixed<22, 6>$ compared to the Keras result. It can be seen that the AUC decreases by the decrease in the number of bits. The AUC can be secured more than 99 %.

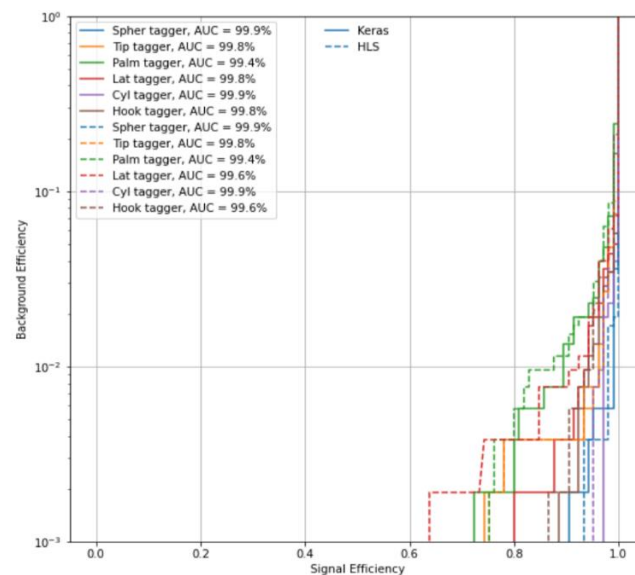


Figure 10. ROC curve for Keras and HLS (6 categories).

When configured in an actual FPGA, communication is performed using the AXI structure, and the generated IP of artificial neural network is used. The FPGA chipset “xcku5p-ffvb676-1-i” was used, and device utilization is shown in Figure 11. In the case of AUC, it is shown as 99 % or more for all classifications. In this case, the total operating speed for 4000 data at the time of inference is about 2.47 sec when using the CPU, and about 480 msec when using the proposed FPGA. Therefore, the operating speed for inference is about 5 times using the proposed FPGA. In the case of FPGA, since it is composed of a wearable system, the data transmission time to the server is not required, so the fast response time is guaranteed. Since security can be secured in healthcare services, the application of edge devices using FPGAs is expected to be of great help.

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	32	-
FIFO	480	-	11830	22236	-
Instance	163	498	67422	68379	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	36	-
Register	-	-	6	-	-
Total	643	498	79258	90683	0
Available	960	1824	433920	216960	64
Utilization (%)	66	27	18	41	0

Figure 11. Device utilization for FPGA chipset “xcku5p-ffvb676-1-i”.

Furthermore, it is possible to use less resources by using pruning technique, but in the current structure, it is not used. When pruning technique is used with 50 % sparsity, the accuracy is reduced by about 3 %, so this option was not used.

Table 1 shows a comparison of accuracy and response speed when performing inference using FPGA and CPU. CPU is Intel Core i7-7700HQ. In the case of accuracy, test data was used. Through this, it was confirmed that EMG signal classification is possible using a small amount of resources when using FPGA. Security is enhanced and a fast response time can be acquired using FPGA. However, when using HLS for hardware deployment, it seems that the performance of optimization between latency and resources is somewhat insufficient, so it is necessary to consider how to implement HLS more efficiently in the future.

EMG classification in real time is attracting attention. In this regard, the response time was about 200 ms in the case of MYO armband and 0.2 ms in the case of implementation with MCU [42,43]. In this study, a response time of 0.12 ms per sample was obtained by using FPGA's parallel computation and light weight (ap_fixed<22,6>).

Table 1. A comparison of accuracy for test data and response speed between CPU and FPGA.

Category	Accuracy	Inference time (4000 samples)
CPU	95 %	2.47 sec
FPGA, ap_fixed<24, 6>	95 %	520 msec
FPGA, ap_fixed<22, 6>	94 %	480 msec
FPGA, ap_fixed<20, 6>	86 %	435 msec
FPGA, ap_fixed<16, 6>	18 %	378 msec

5. Conclusions and Discussions

The implementation of edge devices using EMG signals was studied. Among various edge devices, FPGAs is considered because it can easily perform parallel computation and can be implemented to ASICs in the future. In this case, since there are resource limitations in the FPGA, optimization was performed on the structure that can maintain accuracy while implementing the artificial neural network as simply as possible. For this, data augmentation was performed on the EMG signal, and MODWT capable of time and frequency domain analysis was used for the feature vector. In the case of convolution neural network and deep neural network, the structure is optimized so that the number of parameters does not exceed 10,000, and the number of bits is optimized to maintain the accuracy. Through this, it was confirmed that HCI, disease diagnosis, and user authentication can be performed quickly and with low power by performing artificial intelligence on the edge device with light-weight. By implementing a wearable device, it can contribute to security enhancement. For future studies, the wearable system with bio-signal sensor system and the edge device will be manufactured together. This system will help the self-diagnosis at the user's desired time. Furthermore, the analysis of power consumption of this wearable system will be accomplished. Also, the studies on Siamese networks that can learn with little data are also planned to be added.

Author Contributions: Conceptualization, H.-S. Choi; methodology, H.-S. Choi; software, H.-S. Choi; validation, H.-S. Choi; formal analysis, H.-S. Choi; investigation, H.-S. Choi; resources, H.-S. Choi; data curation, H.-S. Choi; writing—original draft preparation, H.-S. Choi; writing—review and editing, H.-S. Choi; visualization, H.-S. Choi; supervision, H.-S. Choi; project administration, H.-S. Choi; funding acquisition, H.-S. Choi. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2017R1A6A1A03015496).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data supporting the findings of the article are available in reference number [33].

Conflicts of Interest: The author declares no conflict of interest.

References

- Swapna, M.; Viswanadhula, U.M.; Aluvalu, R.; Vardharajan, V.; Kotecha, K. Bio-Signals in Medical Applications and Challenges Using Artificial Intelligence. *Journal of Sensor and Actuator Networks*. **2022**, *11*(1), 17.
- Hammad, M.; Pławiak, P.; Wang, K.; Acharya, U.R. ResNet-Attention model for human authentication using ECG signals. *Expert Systems*, **2021**, *38*(6), e12547.
- Choi, E.J.; Kim, D.K. Arousal and Valence Classification Model Based on Long Short-Term Memory and DEAP Data for Mental Healthcare Management. *Healthc. Inform. Res.* **2018**, *24*(4), 309–316.
- Shahid, H.; Butt, A.; Aziz, S.; Khan, M.U.; Naqvi, S.Z.H. Emotion Recognition System featuring a fusion of Electrocardiogram and Photoplethysmogram Features. In 14th International Conference on Open Source Systems and Technologies. 1–6, December 2020.
- Yu, J.; Park, S.; Kwon, S.H.; Cho, K.H.; Lee, H. AI-Based Stroke Disease Prediction System Using ECG and PPG Bio-Signals. *IEEE Access*. **2022**, *10*, 43623–43638.
- Muhammad, G.; Alshehri, F.; Karray, F.; El Saddik, A.; Alsulaiman M.; Falk, T.H. A comprehensive survey on multimodal medical signals fusion for smart healthcare systems. *Information Fusion*. **2021**, *76*, 355–375.
- Raurale, S.A.; McAllister, J.; Del Rincon, J.M. EMG biometric systems based on different wrist-hand movements. *IEEE Access*. **2021**, *9*, 12256–12266.
- Rahim, M.A.; Shin, J. Hand movement activity-based character input system on a virtual keyboard. *Electronics*. **2020**, *9*(5), 774.
- Antonelli, M.G.; Beomonte Zobel, P.; Durante, F.; Zeer, M. Modeling-Based EMG Signal (MBES) Classifier for Robotic Remote-Control Purposes. *Actuators*. **2022**, *11*(3), 65.
- Albadawi, Y.; Takruri, M.; Awad, M. A review of recent developments in driver drowsiness detection systems. *Sensors*. **2022**, *22*(5), 2069.
- Toro-Ossaba, A.; Jaramillo-Tiguerros, J.; Tejada, J.C.; Pena, A.; Lopez-Gonzalez, A.; Castanho, R.A. LSTM Recurrent Neural Network for Hand Gesture Recognition Using EMG Signals. *Applied Sciences*. **2022**, *12*(19), 9700.
- Schluter, C.; Caraguay, W.; Ramos, D.C. Development of a low-cost EMG-data acquisition armband to control an above-elbow prosthesis. *Journal of Physics: Conference Series*. **2022**, *2232*(1), 012019.
- Alsolai, H.; Qureshi, S.; Zeeshan Iqbal, S.M.; Ameer, A.; Cheaha, D.; Henesey, L.E.; Karrila, S. Employing a Long-Short-Term Memory Neural Network to Improve Automatic Sleep Stage Classification of Pharmacoe-EEG Profiles. *Applied Sciences*. **2022**, *12*(10), 5248.
- Sun, Y.; Xu, C.; Li, G.; Xu, W.; Kong, J.; Jiang, D.; Chen, D. Intelligent human computer interaction based on non-redundant EMG signal. *Alexandria Engineering Journal*. **2020**, *59*(3), 1149–1157.
- Li, Q.D.P.; Zheng, J. Enhancing the security of pattern unlock with surface EMG-based biometrics. *Applied Sciences*. **2020**, *10*(2), 541.
- Sadikoglu, F.; Kavalcioglu, C.; Dagman, B. Electromyogram (EMG) signal detection, classification of EMG signals and diagnosis of neuropathy muscle disease. *Procedia Computer Science*. **2017**, *120*, 422–429.
- Ngai, W.K.; Xie, H.; Zou, D.; Chou, K.L. Emotion recognition based on convolutional neural networks and heterogeneous bio-signal data sources. *Information Fusion*. **2022**, *77*, 107–117.
- Saikia, A.; Mazumdar, S.; Sahai, N.; Paul, S.; Bhatia, D. Performance analysis of artificial neural network for hand movement detection from EMG signals. *IETE Journal of Research*. **2022**, *68*(2), 1074–1083.

19. Usman, M.; Amin, R.; Aldabbas, H.; Alouffi, B. Lightweight challenge-response authentication in SDN-based UAVs using elliptic curve cryptography. *Electronics*. **2022**, *11*(7), 1026. 453
20. Shumba, A.T.; Montanaro, T.; Sergi, I.; Fachechi, L.; De Vittorio, M.; Patrono, L. Leveraging IoT-Aware Technologies and AI Techniques for Real-Time Critical Healthcare Applications. *Sensors*. **2022**, *22*(19), 7675. 454
21. Zhu, G.; Liu, D.; Du, Y.; You, C.; Zhang, J.; Huang, K. Toward an intelligent edge: Wireless communication meets machine learning. *IEEE Communications Magazine*. **2020**, *58*(1), 19–25. 455
22. Yazici, M.T.; Basurra, S.; Gaber, M.M. Edge machine learning: Enabling smart internet of things applications. *Big data and cognitive computing*. **2018**, *2*(3), 26. 456
23. Sudharsan, B.; Breslin, J.G.; Ali, M.I. Edge2train: A framework to train machine learning models (SVMs) on resource-constrained IoT edge devices. In Proceedings of the 10th International Conference on the Internet of Things. 1–8, October 2020. 457
24. Akopyan, F.; Sawada, J.; Cassidy, A.; Alvarez-Icaza, R.; Arthur, J.; Merolla, P.; Modha, D.S. TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip. *IEEE Trans. On Computer-aided design of Integrated circuits and systems*. **2015**, *34*(10), 1537–1557. 458
25. Sambas, A.; Vaidyanathan, S.; Zhang, X.; Koyuncu, I.; Bonny, T.; Tuna, M.; Alcin, M.; Zhang, S.; Sulaiman, I.M.; A. M. Awwal, A.M.; Kumam, P. A Novel 3D Chaotic System With Line Equilibrium: Multistability, Integral Sliding Mode Control, Electronic Circuit, FPGA Implementation and Its Image Encryption. *IEEE Access*. **2022**, *10*, 68057–68074. 459
26. Sambas, A.; Vaidyanathan, S.; Bonny, T.; Zhang, S.; Hidayat, Y.; Gundara, G.; Mamat, M. Mathematical model and FPGA realization of a multi-stable chaotic dynamical system with a closed butterfly-like curve of equilibrium points. *Applied Sciences*. **2021**, *11*(2), 788. 460
27. Sambas, A.; Vaidyanathan, S.; Tlelo-Cuautle, E.; Abd-El-Atty, B.; Abd El-Latif, A.A.; Guillen-Fernandez, O.; Sukono; Hidayat, Y.; Gundara, G. A 3-D multi-stable system with a peanut-shaped equilibrium curve: Circuit design, FPGA realization, and an application to image encryption. *IEEE Access*. **2020**, *8*, 137116–137132. 461
28. Liu, X.; Yang, J.; Zou, C.; Chen, Q.; Yan, X.; Chen, Y.; Cai, C. Collaborative edge computing with FPGA-based CNN accelerators for energy-efficient and time-aware face tracking system. *IEEE Trans. On Computational Social Systems*. **2021**, *9*(1), 252–266. 462
29. Guo, K.; Zeng, S.; Yu, J.; Wang, Y.; Yang, H. A survey of FPGA-based neural network accelerator. *ACM Transactions on Reconfigurable Technology and Systems*. **2019**, *12*(1), 1–26. 463
30. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition. 4510–4520, January 2018. 464
31. Norrie, T.; Patil, N.; Yoon, D.H.; Kurian, G.; Li, S.; Laudon, J.; Young, C.; Jouppi, N.; Patterson, D. The design process for Google's training chips: TPUv2 and TPUv3. *IEEE Micro*, **2021**, *41*(2), 56–63. 465
32. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. International Conference on Learning Representations. 1–14, February 2016. 466
33. Sapsanis, C.; Georgoulas, G.; Tzes, A.; Lymberopoulos, D. Improving EMG based classification of basic hand movements using EMD. In 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society 13 (EMBC 13). 5754–5757, July 2013. 467
34. Lobov, S.; Krilova, N.; Kastalskiy, I.; Kazantsev, V.; Makarov, V.A. Latent factors limiting the performance of sEMG-interfaces. *Sensors*. **2018**, *18*(4), 1122. 468
35. Tsinganos, P.; Cornelis, B.; Cornelis, J.; Jansen, B.; Skodras, A. Data augmentation of surface electromyography for hand gesture recognition. *Sensors*. **2020**, *20*(17), 4892. 469
36. Jeong, J.W.; Lee, W.; Kim, Y.J. A Real-Time Wearable Physiological Monitoring System for Home-Based Healthcare Applications. *Sensors*. **2021**, *22*(1), 104. 470
37. Adib, A.; Zaerpour, A.; Lotfird, M. On the reliability of a novel MODWT-based hybrid ARIMA-artificial intelligence approach to forecast daily snow depth (Case study: the western part of the Rocky Mountains in the USA). *Cold Regions Science and Technology*. **2021**, *189*, 103342. 471
38. Zhdanov, D.S.; Zemlyakov, I.Y.; Kosteley, Y.V.; Bureev, A.S. Choice of Wavelet Filtering Parameters for Processing Fetal Phonocardiograms with High Noise Level. *Biomedical Engineering*. **2021**, *55*(3), 194–199. 472
39. Fast Machine Learning Lab, <https://github.com/fastmachinelearning/>. 473
40. Westby, I.; Yang, X.; Liu, T.; Xu, H. FPGA acceleration on a multi-layer perceptron neural network for digit recognition. *The Journal of Supercomputing*. **2021**, *77*(12), 14356–14373. 474
41. Xia, M.; Huang, Z.; Tian, L.; Wang, H.; Chang, V.; Zhu, Y.; Feng, S. SparkNoC: an energy-efficiency FPGA-based accelerator using optimized lightweight CNN for edge computing. *Journal of Systems Architecture*. **2021**, *115*, 101991. 475
42. Zhang, Z.; Yang, K.; Qian, J.; Zhang, L. Real-time surface EMG pattern recognition for hand gestures based on an artificial neural network. *Sensors*, **2019**, *19*(14), 3170. 476
43. Liu, X.; Sacks, J.; Zhang, M.; Richardson, A.G.; Lucas, T.H.; Van der Spiegel, J. The virtual trackpad: An electromyography-based, wireless, real-time, low-power, embedded hand-gesture-recognition system using an event-driven artificial neural network. *IEEE Transactions on Circuits and Systems II: Express Briefs*, **2016**, *64*(11), 1257–1261. 477