electronics

Article

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

Hyun-Sik Choi

Department of Electronic Engineering, College of IT Convergence Engineering, Chosun University, Gwangju 61452, Republic of Korea; hs22.choi@chosun.ac.kr

Abstract: Recently, the application of bio-signals in the fields of health management, human–computer interaction (HCI), and user authentication has increased. This is because of the development of artificial intelligence technology, which can analyze bio-signals in numerous fields. In the case of the analysis of bio-signals, the results tend to vary depending on the analyst, owing to a large amount of noise. However, when a neural network is used, feature extraction is possible, enabling a more accurate analysis. However, if the bio-signal time series is analyzed as is, the total neural network increases in size. In this study, to accomplish a light-weight neural network, a maximal overlap discrete wavelet transform (MODWT) and a smoothing technique are used for better feature extraction. Moreover, the learning efficiency is increased using an augmentation technique. In designing the neural network, a one-dimensional convolution layer is used to ensure that the neural network is simple and light-weight. Consequently, the light-weight attribute can be achieved, and neural networks can be implemented in edge devices such as the field programmable gate array (FPGA), yielding low power consumption, high security, fast response times, and high user convenience for wearable applications. The electromyogram (EMG) signal represents a typical 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: Choi, H.-S. Electromyogram (EMG) Signal Classification Based on Light-Weight Neural Network with FPGAs for Wearable Application. *Electronics* **2023**, *12*, 1398. https://doi.org/10.3390/electronics12061398

Academic Editors: Andres Upegui, Andrea Guerrieri and Laurent Gantel

Received: 3 February 2023
Revised: 28 February 2023
Accepted: 1 March 2023
Published: 15 March 2023

1. Introduction

Various bio-signals have been developed, 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 fields of application are diverse [1,2]. Typical application fields include health care, human–computer interaction (HCI), and user authentication [3–6]. In this study, the EMG signal was treated as a representative bio-signal. In the case of the EMG signal, the measurement is simple; therefore, 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 has three input terminals [9]. Two terminals were used to measure the signal in the middle and end of the muscle. The last terminal was used as a reference signal, which is far from the muscle. This is amplified by the differential amplification unit to determine muscle movement [10]. The EMG sensor is easy to attach and can be used for HCI by attaching it to the most active hand. For HCI applications that use EMG, the classification of the signals using artificial intelligence has been the subject of many studies [11,12]. The increasing use of bio-signals is due to the rapid progress of artificial intelligence and big data technology.

In the case of EMG signals, as with other bio-signals, various noise components are included in the measurement [13]. This creates a problem in that the analysis of results can differ depending on the analyst. However, when a neural network is used, feature extraction is possible and more accurate, and consistent results can be obtained [14]. In [15], based on EMG signals measured in 16 channels, support vector machine (SVM) and generalized regression neural network (GRNN) artificial intelligence algorithms were employed using the root mean square (RMS), an autoregressive model (AR), and the slope sign change (SSC). In this manner, the six hand gestures were classified with 98% accuracy. In [16], a user recognition study was conducted using the EMG signal measured when drawing an unlock pattern on a smartphone screen. The EMG signal was measured in the flexor digitorum superficialis (FDS) muscle of the forearm using OpenBCI. Using one-class SVM (OCSVM) and extracting features, such as the mean absolute value (MAV), variance (VAR) in the time domain, the user was recognized 98.2% of the time. In [17], a fast Fourier transform (FFT) was performed using EMG signals, 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 is or if FFT is used, the neural network for feature extraction becomes significant, with more than 100,000 weight values, making it unsuitable for wearable applications [18,19]. Most of the heavy EMG signal analyses are implemented in a structure that transmits data to a server and analyzes them on the server. This means that many hardware resources are used in the server, and the power consumption is high. Moreover, security is emerging as a big problem in EMG analysis, while, regarding authentication, problems such as slow response times may occur [20,21].

To solve this problem, the field of bio-signal processing in edge devices has received considerable attention [22–24]. This is because when using an edge device, low power, a fast response speed, and security can be expected [25]. However, difficulty in achieving low power in the case of CPU- and GPU-based artificial intelligence systems is a problem. This is because their operation is clock-based and consumes considerable power in the process of accessing memory. To prevent this, a field programmable gate array (FPGA) with a distributed structure is a possible alternative, and low power and fast response times can be secured through various structure optimizations [26,27]. In the case of an edge device with a CPU/GPU or an embedded system, only one execution is performed, whereas an FPGA has the advantage of executing multiple instructions simultaneously. Therefore, FPGAs have received significant attention in the field of edge devices [28,29].

In the case of an inference accelerator using an FPGA, many studies have been conducted [30]. However, when an FPGA is used, external memory such as dynamic random access memory (DRAM) is required to store the weight values, and considerable energy is therefore consumed for reading memory processes. To prevent this, edge devices that are more compressed are being studied, which is also the focus of this study. This study focuses on hardware configurations that can be applied in real life. The hardware compression methods used are largely based on (1) compression of the model network itself using algorithms, (2) compression of the computation methods (e.g., MobileNet [31], a systolic array structure for the tensor processing unit (TPU) [32]), and (3) a pruning method that uses weight sparsity and bit quantization [33]. In this study, hardware compression was carried out by focusing on methods (1) and (3). Therefore, the main purpose is to design a light-weight neural network suitable for an edge device, such as an FPGA, capable of parallel operation and with low memory access. With such a neural network, memory components for storing weight values can exist inside the FPGA without any external DRAM memory access (e.g., ResNet minimum storage requirement for 26 million weight values of 104 MB).

For this purpose, a frequency-filtered signal is used by signal processing. The maximal overlap discrete wavelet transform (MODWT) and smoothing algorithm for this signal are used. Moreover, an augmentation technique is used to increase learning efficiency. For the neural network design, a one-dimensional convolution layer is used for light-weight systems that are suitable for wearable applications. The neural network implemented in this manner is deployed to the FPGA, which is an edge device, and performs an appropriate inference operation. Through this, a low-power, high-speed, and high-security system with an edge device can be implemented even in low-cost FPGAs for artificial intelligence. In particular, the size of the model network can be reduced using an efficient feature extraction method that is suitable for EMG signals; through this, operation with high accuracy and low power consumption without access to external memory is possible. Additionally, bit quantization was performed to secure the compression of the edge device. The main application area is wearable systems for HCI or user authentication. Section 2 investigates the signal processing of EMG signals, and Section 3 reports on the structure of the neural network. Section 4 shows the overall hardware structure and verification process through high level synthesis (HLS) for hardware deployment. Section 5 presents the 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 [34]. This is the measurement data for five people, and the hand gestures are composed of six; thus, the final goal is to classify the six movements using the measured EMG signals. The measurement sampling frequency was 500 Hz, and the measurement was performed using two channels. The sampling frequency was 500 Hz because the most important data in the case of EMG signals are included in the range from 10 to 250 Hz. However, because a wearable device is assumed in this study, classification was performed using only data from channel 1. This causes a decrease in accuracy, but because one channel is mostly used in the actual HCI environment, one channel is used. In a real environment, the six hand gestures would be difficult to distinguish with one channel of data. In actual wearable device applications, classification for approximately two to three classes is considered. Both the EMG sensor and artificial intelligence will be produced in the form of a smartwatch. The 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 six hand gestures. Additionally, in the case of measurement data, the amount of data is insufficient because it is data from five people; however, the data were used only for the feasibility test of the proposed neural network. The dataset "EMG data for gestures Data Set" of 36 subjects and eight channels for six hand gestures were also verified with similar sequences. A 98% classification accuracy was observed for the validation data (not shown here). The increased accuracy is related to the increased channel numbers [35]. In the future, the EMG signal will be measured via its own compact EMG modules and will be used as a wearable device. A high-efficiency wearable device for EMG signal acquisition is currently under development. Figure 1a shows a prototype of the EMG sensor that is currently under development. A differential amplifier was used, and signals from 10 to 250 Hz were obtained using frequency filtering. Figure 1b shows the measured EMG signals of the wrist. In the future, the signals will be produced in the form of a wearable watch through miniaturization and low power consumption and will be combined with an edge device for artificial intelligence to detect wrist movements.

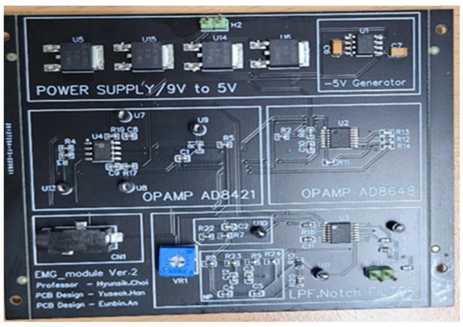
In the case of the EMG input dataset, because the number of datasets is small, it is unsuitable for training. Therefore, data augmentation was implemented using additive noise and magnitude warping. In the case of additive noise, Gaussian noise was added, and the entire signal was processed after normalization. The equations below describe the creation of a new signal by adding additive Gaussian noise [36].

$$x_i^* = x_i + n \tag{1}$$

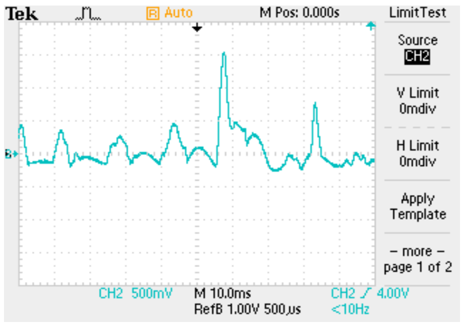
$$n \sim Gaussian(\mu = 0,\ \sigma = \sqrt{\frac{x_i^2}{SNR}}) \tag{2}$$

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 2 shows a representative EMG signal and an augmentation signal using the additive noise method.

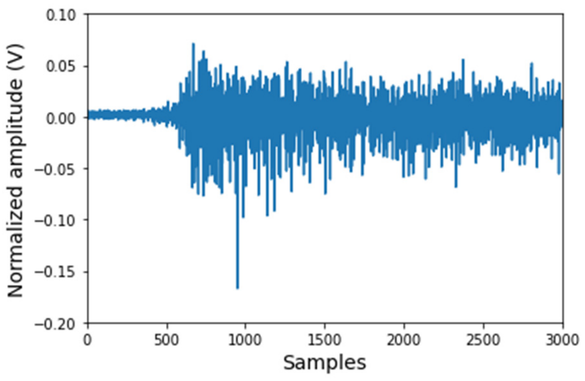


(**a**)

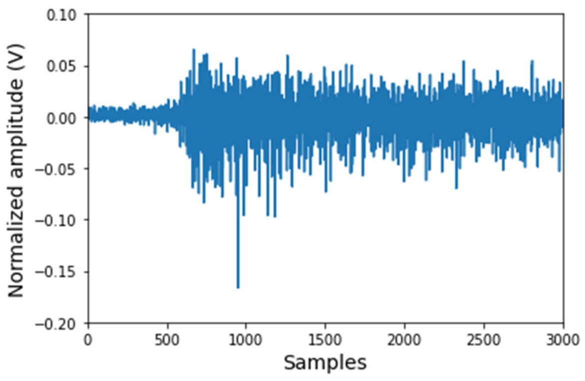


(**b**)

Figure 1. (**a**) Prototype for electromyogram (EMG) sensor and (**b**) measurement results in the wrist.



(**a**)



(**b**)

Figure 2. Representative (**a**) EMG signal, (**b**) augmentation signal using additive noise of "sEMG for Basic Hand movements Data Set".

Moreover, the moving average, permutation, and magnitude warping techniques can be used for data augmentation. In this study, only the magnitude warping technique, which is known to be the most efficient, was used [36]. 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 following equations:

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

$$\mathrm{r} = \{\mathrm{r}(t_1),\ \mathrm{r}(t_2),\ \ldots,\ \mathrm{r}(t_T)\} \tag{4}$$

where t is the sampling time. A random curve is generated using the 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 approximately three times. This resulted in an increase in the classification accuracy of approximately 3% compared to when the data augmentation method was not used.

2.2. Filtering and Smoothing

The EMG signals were completely removed at 50 Hz and 0 Hz using an additional filter. They are signals that have passed through the band pass filter; however, when the signal analysis is performed through FFT, DC and 50 Hz components still exist; therefore, 50 Hz and 0 Hz signals were additionally removed using an additional high-performance filter for signal processing. The first filter was an active second-order high-pass filter with a cutoff frequency (f_c) of 10 Hz. The second filter was an active notch filter with f_c of 50 Hz. These are designed for easy hardware implementation in an EMG sensor system. The active notch filter is designed using the active twin-T notch filter method [37]. Figure 3 shows a representative signal before and after applying additional filters. This is displayed as a frequency spectrum for ease of comparison.

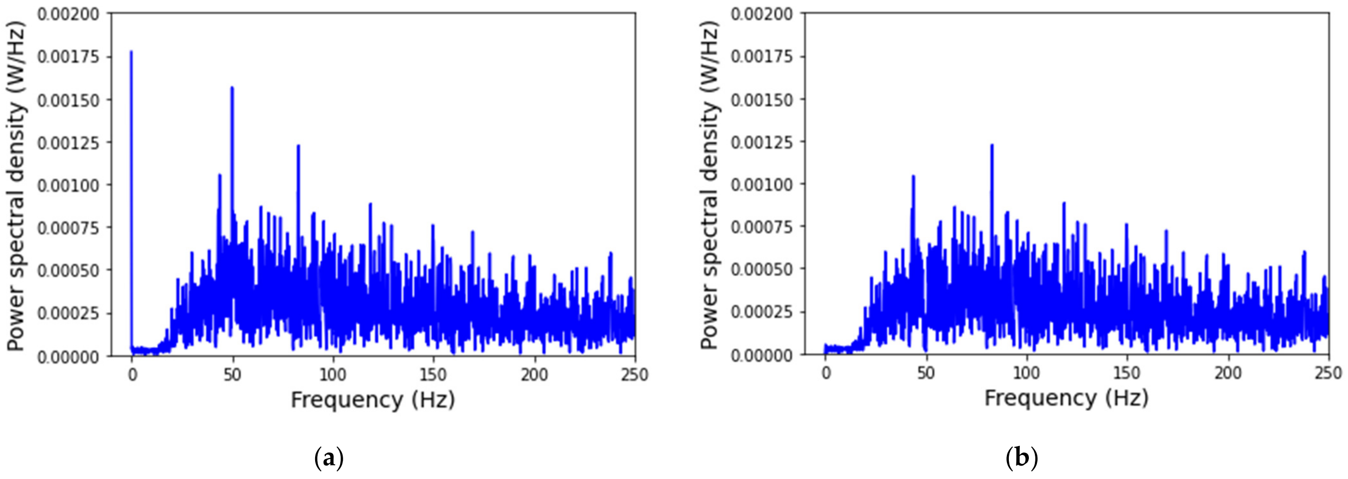


Figure 3. Frequency spectrum of EMG signal (**a**) before and (**b**) after the use of additional filters.

In addition to using the filtering method for the time series data, the effective noise components can be reduced through a smoothing technique. In this study, the Savitzky–Golay filter was used as the smoothing technique. Although several smoothing techniques exist, the Savitzky–Golay method was used because it is easy to perform for a simple hardware implementation. The Savitzky–Golay filter can mathematically replace smoothing using a polynomial regression model 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 the filtering method and smoothing technique, the accuracy could be improved by approximately 2%.

2.3. Feature Extraction

For feature extraction of the EMG signal, the MODWT method was used for easy implementation in the FPGA. The wavelet transform was developed to perform time and frequency domain analyses simultaneously. 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, the discrete wavelet transform (DWT) based on orthonormal wavelet is frequently used; however, MODWT is more sensitive to circular shifts than the general DWT. This can be interpreted as a linear filter result. In hardware implementation, a finite impulse response (FIR) filter is used using digital logic. The above results indicate that MODWT, which is easy to interpret in combination with events that actually occur in nature, is useful in time series analysis. Moreover, unlike DWT, MODWT is defined naturally for all sample sizes. The MODWT method was expanded while preserving the size of the data. There are various studies related to feature extraction [38]. Based on the dataset measured at Chosun University, a scalogram based on the continuous wavelet transform (CWT) is confirmed to be useful as an ECG feature [39]. By extending this, MODWT was selected as a feature suitable for the EMG analysis at edge devices. Owing to the 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 the MODWT method, a similar performance can be secured even with a small neural network. The basic implementation algorithm of the MODWT method is as follows. For a time series X with an arbitrary sample size N, the j-th level MODWT wavelet (W_j*) and scaling (V_j*) coefficients are defined as follows [40].

$$W_{j,t}^{*} = \sum_{l=0}^{L_j-1} h_{j,l}^{*} X_{t-l \ mod \ N} \tag{5}$$

$$V_{j,t}^{*} = \sum_{l=0}^{L_j-1} g_{j,l}^{*} X_{t-l \ mod \ N} \tag{6}$$

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 second-order Daubechies filter (db2) was used, and the original signal was restored using inverse MODWT. In addition to db2, filters that can be used in the MODWT method include the first-order Daubechies (db1), fourth-order Daubechies (db4), Haar, Symlets, and Coiflets filters [41]. Except for the Haar and db1 filters, all exhibited excellent performance. Moreover, the decomposition level was selected as 4 because the performance increased up to level 4; however, the performance decreased by approximately 2% when the level was 5 or higher. This is related to the decomposition of the frequency domain. For example, the frequency spectrum of the level 1 MODWT signal is between 125 Hz and 250 Hz.

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

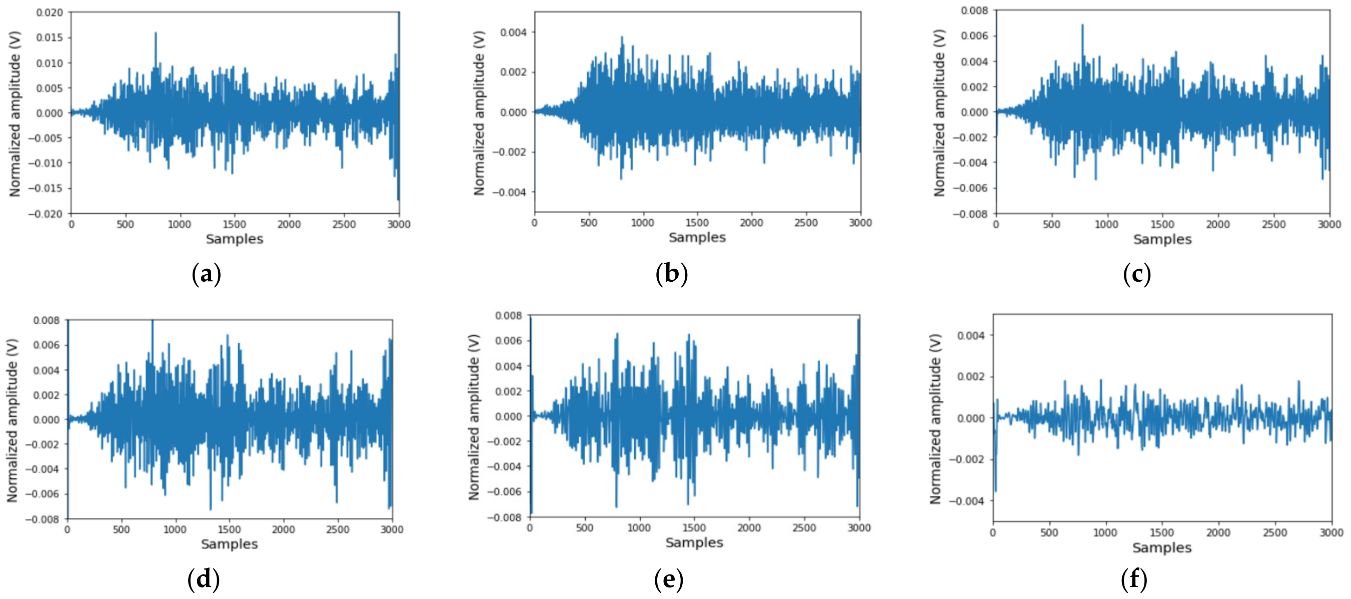
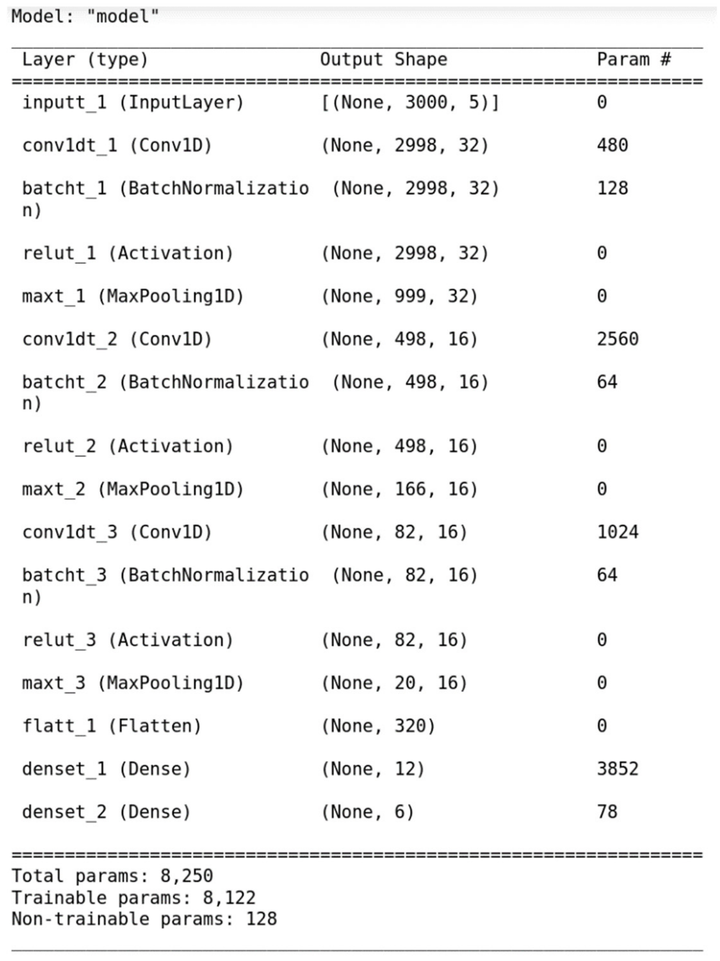


Figure 4. Maximal overlap discrete wavelet transform (MODWT) execution result. (**a**) Original signal, (**b**) level 1, (**c**) level 2, (**d**) level 3, (**e**) level 4, and (**f**) residual.

3. Artificial Intelligence Network

The MODWT signal for feature extraction has five channels and is given as an input to the one-dimensional convolution layer, as shown in Figure 5. The three one-dimensional convolution layers were used. The one-dimensional convolution layer is suitable for realizing a compressed neural network because the amount of computation is smaller than that of the two-dimensional convolution layer. To implement the light-weight characteristic while keeping the neural network as simple as possible, the accuracy was aimed at above 93%. Therefore, the verification of various neural networks was performed, and among them, a neural network with 93% or more accuracy was selected while achieving the light-weight characteristic. KerasTuner was used for the network design.



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inputt_1 (InputLayer)       [(None, 3000, 5)]         0

 conv1dt_1 (Conv1D)          (None, 2998, 32)          480

 batcht_1 (BatchNormalizatio  (None, 2998, 32)         128
 n)

 relut_1 (Activation)        (None, 2998, 32)          0

 maxt_1 (MaxPooling1D)       (None, 999, 32)           0

 conv1dt_2 (Conv1D)          (None, 498, 16)           2560

 batcht_2 (BatchNormalizatio  (None, 498, 16)          64
 n)

 relut_2 (Activation)        (None, 498, 16)           0

 maxt_2 (MaxPooling1D)       (None, 166, 16)           0

 conv1dt_3 (Conv1D)          (None, 82, 16)            1024

 batcht_3 (BatchNormalizatio  (None, 82, 16)           64
 n)

 relut_3 (Activation)        (None, 82, 16)            0

 maxt_3 (MaxPooling1D)       (None, 20, 16)            0

 flatt_1 (Flatten)           (None, 320)               0

 denset_1 (Dense)            (None, 12)                3852

 denset_2 (Dense)            (None, 6)                 78

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

Figure 5. Structure and parameters for an artificial intelligence network.

The activation function of the one-dimensional convolution layer used the rectified linear unit (ReLU), and padding was not used. The kernel for the convolution layer used L1 regulation, and the used parameter was 0.001. Three 1-dimensional convolution layers were used, and the number of channels was optimized. The max-pooling layer after the convolution layer was 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 the sigmoid and softmax functions were used for the activation functions. These were implemented in the form of a look-up table when being implemented to the hardware resources in Section 4. Because the dense layer uses many parameters, the structure is implemented as concisely as possible. The focus in this case was to ensure high accuracy with few parameters and a simple structure.

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. Moreover, the learning process was performed using the validation data of 30% of the total training data. The learning rate (*lr*), a representative hyperparameter, used a step decay function, and the *lr* change factor was set to 0.5. The initial *lr* value was set to 0.0001. The minimum value of *lr* was set to 10^{-7}. Thus, the appropriate learning rate was adjusted. The total number of trainable parameters of the designed neural network was 8122. This corresponds to the lightweight neural network, and 96% accuracy for the validation data was possible with the help of algorithms such as MODWT, data augmentation, and smoothing techniques. Thus, the implementation of a light-weight neural network that can be operated with a small number 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 in training epochs are shown in Figure 6. The accuracies of the training data and validation data were approximately 99% and 96%, respectively.

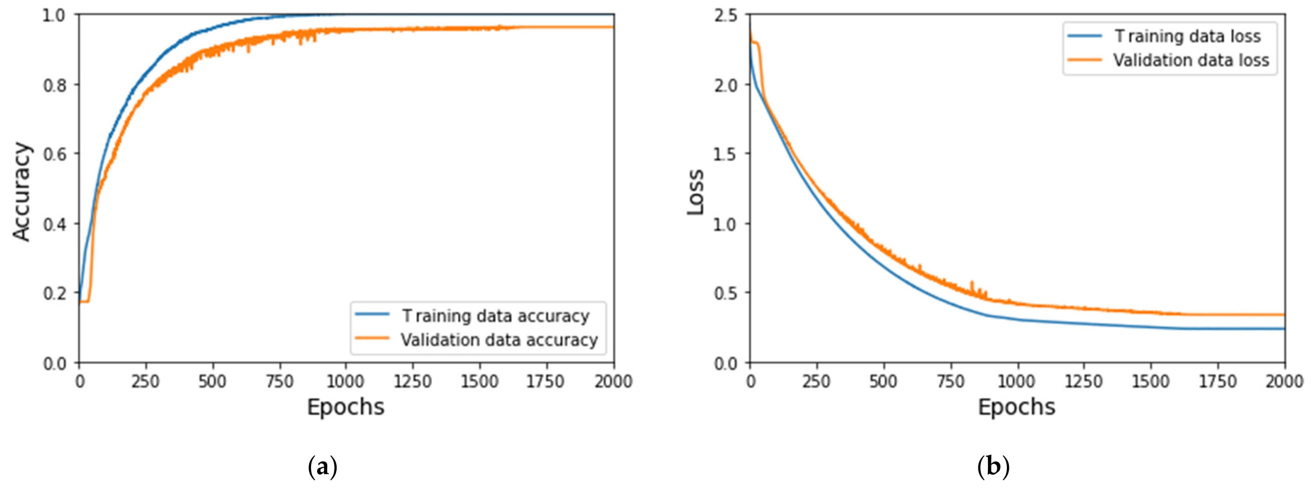


Figure 6. (**a**) Accuracy of training and validation data. (**b**) Loss of training and validation data.

In the case of new data (test data), the confusion matrix is shown in Figure 7. The accuracy of the test data was 95% after training. Although this value has relatively low accuracy, the value is high considering that it is implemented using a small number of hardware resources. In the case of the other light-weight neural network example, the accuracy of "jet tagging" was approximately 74% [42]. Using MODWT, feature extraction is performed efficiently, and high accuracy can be secured with few parameters. In Figure 7, "spher" represents the action for holding spherical tools, "tip" for holding small tools, "palm" for grasping with the palm facing the object, "lat" for holding thin, flat objects, "cyl" for holding cylindrical tools, and "hook" for supporting a heavy load.

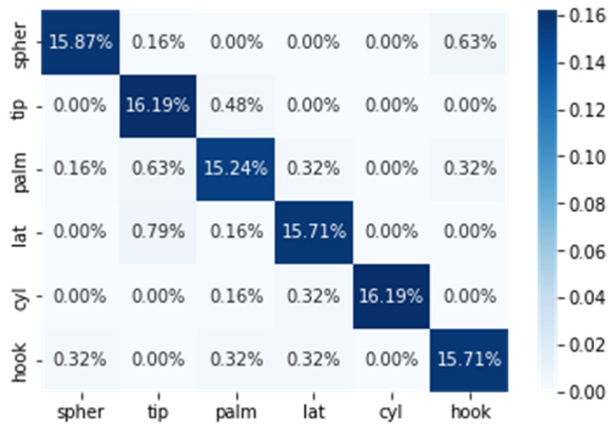


Figure 7. Confusion matrix for test data.

Finally, Figure 8 shows the receiver operating characteristic (ROC) curve and area under curve (AUC). The ROC curve is often used to evaluate the performance of a model that distinguishes classes, with a false positive rate (FPR) on the x-axis and a true positive rate (TPR) on the y-axis. Thus, various performance indices can be represented. Currently, the AUC is over 99% for the test data. In the previous study carried out by the authors of [43] using the State-of-the-Art (SOTA), the achievable accuracy with FPGAs was approximately 95.4% when using more parameters in the proposed network. In this study, higher accuracy was achieved with fewer parameters using various feature extraction techniques.

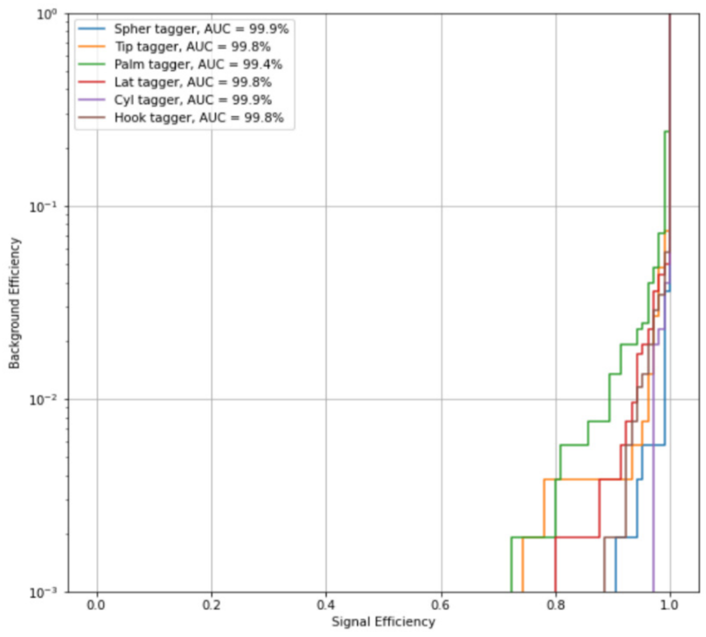
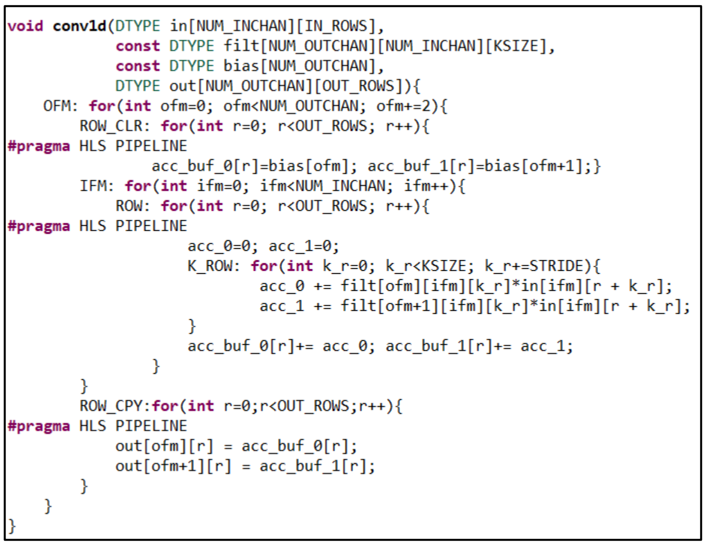


Figure 8. Receiver operating characteristic (ROC) curve for test data (six categories).

4. Hardware Deployment

Using the light-weight neural network model created by software using the Keras tool, it is converted to the Verilog hardware description language (HDL) using HLS [44]. Verilog HDL is changed to NAND or NOR gates through synthesis to create digital logic. HLS is a method that automatically converts the software code to Verilog HDL when written in C/C++, with which the user is familiar. Because large logic requires longer duration when written directly in a language such as Verilog HDL, HLS can significantly reduce the hardware development time and increase user convenience. Moreover, 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, latency reduction in the FPGA can be obtained. Thus, the structure optimization is performed together in the HLS. Figure 9 shows the HLS code for one-dimensional convolution layer written in C/C++, representatively. This 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 filter after storing the values of the bias in the buffer. Each batch normalization, max-pooling, activation, and dense layer was implemented using HLS to be similar to the neural network proposed by Keras. In the case of the sigmoid and softmax functions, the number of exponential calculations is large; therefore, it is implemented in the form of a look-up table. Similar to the abovementioned method, several hardware implementation methods are available, 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 similar frameworks into a register transfer level (RTL). In this study, the artificial intelligence part for EMG signal classification in real life can be implemented in an edge device, which was designed to have light-weight parameters through neural network design. This resulted in a reduction in 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_CPY: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 9. High level synthesis (HLS) code for one-dimensional convolution layer.

The advantage of hardware deployment is that the number of systems can be freely used. Because the hardware is designed by us, the number of systems used inside can be defined and used; thus, the hardware resources can be reduced [45]. In the case of using the floating point number used in software in Section 3, 95% accuracy can be obtained for the test data using the hardware resource. This is the same accuracy result as that of the software. However, the number of bits used for the resource reduction is reduced to make it suitable for wearable devices. In this case, bit optimization was performed. In the case of a fixed point number, an error occurs, unlike in Keras, and a large difference is shown depending on the number of bits used. In the 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 the FPGA. In particular, the entire bit becomes 16 bits, the integer is 6 bits, including the sign bit, and 10 bits represent the value below the decimal point. In this case, if ap_fixed<24, 6> is used, it can be implemented with a small number of hardware resources without degradation in accuracy. However, when ap_fixed<22, 6> was used, 80% of the hardware resources were used compared to ap_fixed<24, 6>, which was regarded as the optimal structure. In this case, the difference in accuracy was approximately 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 the weights used to optimize the bit was examined, as shown in Figure 10. The total number of bits used for the optimal structure was 22. Bit optimization can also be performed using QKeras. QKeras is a quantization extension of Keras.

The network was implemented using HLS with ap_fixed<22, 6>. The accuracy of the test data was approximately 94%, and the AUC could be secured by more than 99%. If the basic ap_fixed<16, 6> is used, the accuracy is reduced to approximately 18%, and the AUC becomes more than 45%. No change is observed in accuracy when using floating-point numbers or ap_fixed<24, 6>. Moreover, a reuse factor from 10 to 20 was used to implement a structure that reuses resources completely. Figure 11 shows the ROC curve in the digital logic implemented in the FPGA with apiece<22, 6> compared with the Keras result. The AUC decreases with a decrease in the number of bits; an AUC can be secured by more than 99%.

When configured in an actual FPGA, communication was performed using the AXI structure, and the generated IP of the artificial neural network was used. The FPGA chipset "xcku5p-ffvb676-1-i" was used, and the device utilization is shown in Figure 12. 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 approximately 2.47 s when using the CPU and approximately 480 ms when using the proposed FPGA. Therefore, the operating speed for inference is approximately five times that of the proposed FPGA. In the case of FPGA, because it is composed of a wearable system, the data transmission time to the server is not required; therefore, a fast response time is guaranteed. Because security can be secured in healthcare services, the application of edge devices using FPGAs is expected to be of great help.

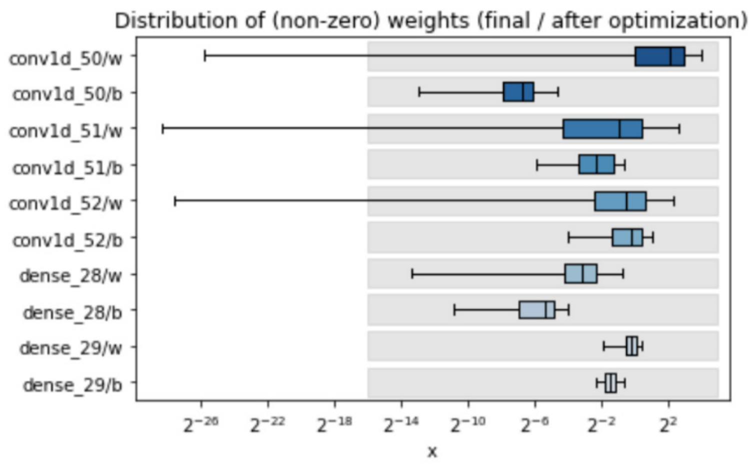


Figure 10. Bit selection using numerical profiling.

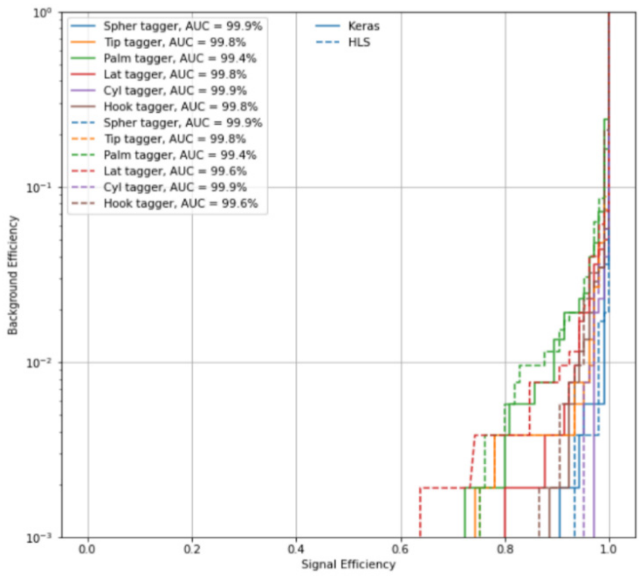
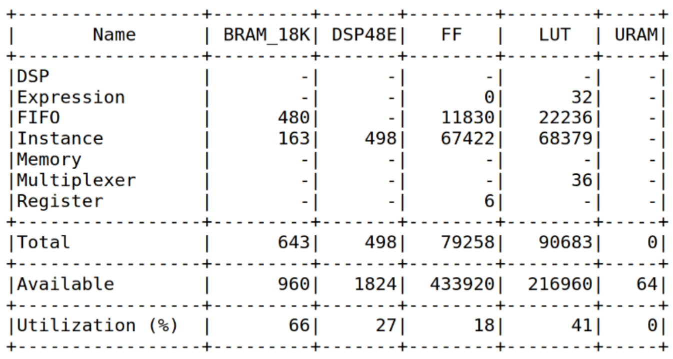


Figure 11. ROC curve for Keras and HLS (six categories).

Furthermore, fewer resources can be used by the pruning technique, although it is not implemented in the current structure. When the pruning technique is used with 50% sparsity, the accuracy is reduced by approximately 3%; hence, this option was not used.

Table 1 presents a comparison of the accuracy and response speed when performing inference using FPGA and CPU. The CPU was Intel Core i7-7700HQ. In the case of accuracy, test data were used. Thus, the EMG signal classification is possible using a small amount of resources when using an FPGA. Security is enhanced, and a fast response time can be acquired using FPGA. However, when using the HLS for hardware deployment, the performance of optimization between the latency and resources is rather insufficient; therefore, a more efficient implementation of the HLS should be considered in the future.



```
+-----------------+---------+-------+--------+--------+-----+
|       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 12. Device utilization for FPGA chipset "xcku5p-ffvb676-1-i".

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

Category	Accuracy	Inference Time (4000 Samples)
CPU	95%	2.47 s
FPGA, ap_fixed<24, 6>	95%	520 ms
FPGA, ap_fixed<22, 6>	94%	480 ms
FPGA, ap_fixed<20, 6>	86%	435 ms
FPGA, ap_fixed<16, 6>	18%	378 ms

EMG classification in real time has attracted considerable attention. In this regard, the response time was approximately 200 ms in the case of the MYO armband and 0.2 ms in the case of implementation with the MCU [46,47]. In this study, a response time of 0.12 ms per sample was obtained using the parallel computation and light weight of the FPGA (ap_fixed<22, 6>).

5. Conclusions and Discussion

The implementation of edge devices using EMG signals was studied. Among various edge devices, FPGAs were considered because they can easily perform parallel computation and can be implemented in ASICs in the future. In this case, owing to resource limitations in the FPGA, optimization was performed on the structure that could maintain accuracy while implementing the artificial neural network easily. For this, data augmentation was performed on the EMG signal, and MODWT, which is capable of time and frequency domain analysis, was used for the feature vector. In the case of convolutional and deep neural networks, the structure was optimized to prevent the number of parameters from exceeding 10,000, and the number of bits was optimized to maintain accuracy. Thus, HCI, disease diagnosis and user authentication could be performed quickly and with low power using artificial intelligence on a light-weight edge device. The implementation of wearable devices can contribute to security enhancement. The main contribution of this study is the examination of the practical applications of edge devices. However, to be grafted onto wearable devices, they must be implemented using fewer resources. In future studies, a wearable system with a bio-signal sensor system and edge device will be manufactured. This system will help the user's self-diagnosis at the desired time. Moreover, studies on Siamese or ensemble networks that can learn with less data are planned. Simultaneously, the study plans to manufacture PPG and ECG sensors to build a wearable system based on artificial intelligence.

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

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

Data Availability Statement: The data supporting the findings of the article are available in reference number [34].

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

References

1. Swapna, M.; Viswanadhula, U.M.; Aluvalu, R.; Vardharajan, V.; Kotecha, K. Bio-Signals in Medical Applications and Challenges Using Artificial Intelligence. *J. Sens. Actuator Netw.* **2022**, *11*, 17. [CrossRef]
2. Hammad, M.; Pławiak, P.; Wang, K.; Acharya, U.R. ResNet-Attention model for human authentication using ECG signals. *Expert Syst.* **2021**, *38*, e12547. [CrossRef]
3. 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*, 309–316. [CrossRef]
4. 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 Proceedings of the 14th International Conference on Open Source Systems and Technologies, Lahore, Pakistan, 16–17 December 2020; pp. 1–6.
5. 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. [CrossRef]
6. 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. *Inf. Fusion* **2021**, *76*, 355–375. [CrossRef]
7. Raurale, S.A.; McAllister, J.; Del Rincon, J.M. EMG biometric systems based on different wrist-hand movements. *IEEE Access* **2021**, *9*, 12256–12266. [CrossRef]
8. Rahim, M.A.; Shin, J. Hand movement activity-based character input system on a virtual keyboard. *Electronics* **2020**, *9*, 774. [CrossRef]
9. Antonelli, M.G.; Beomonte Zobel, P.; Durante, F.; Zeer, M. Modeling-Based EMG Signal (MBES) Classifier for Robotic Remote-Control Purposes. *Actuators* **2022**, *11*, 65. [CrossRef]
10. Mukhopadhyay, A.K.; Samui, S. An experimental study on upper limb position invariant EMG signal classification based on deep neural network. *Biomed. Signal Process. Control* **2020**, *55*, 101669. [CrossRef]
11. Albadawi, Y.; Takruri, M.; Awad, M. A review of recent developments in driver drowsiness detection systems. *Sensors* **2022**, *22*, 2069. [CrossRef] [PubMed]
12. Toro-Ossaba, A.; Jaramillo-Tigreros, J.; Tejada, J.C.; Pena, A.; Lopez-Gonzalez, A.; Castanho, R.A. LSTM Recurrent Neural Network for Hand Gesture Recognition Using EMG Signals. *Appl. Sci.* **2022**, *12*, 9700. [CrossRef]
13. Schluter, C.; Caraguay, W.; Ramos, D.C. Development of a low-cost EMG-data acquisition armband to control an above-elbow prosthesis. *J. Phys. Conf. Ser.* **2022**, *2232*, 012019. [CrossRef]
14. 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 Pharmaco-EEG Profiles. *Appl. Sci.* **2022**, *12*, 5248. [CrossRef]
15. Sun, Y.; Xu, C.; Li, G.; Xu, W.; Kong, J.; Jiang, D.; Chen, D. Intelligent human computer interaction based on non-redundant EMG signal. *Alex. Eng. J.* **2020**, *59*, 1149–1157. [CrossRef]
16. Li, Q.D.P.; Zheng, J. Enhancing the security of pattern unlock with surface EMG-based biometrics. *Appl. Sci.* **2020**, *10*, 541. [CrossRef]
17. Sadikoglu, F.; Kavalcioglu, C.; Dagman, B. Electromyogram (EMG) signal detection, classification of EMG signals and diagnosis of neuropathy muscle disease. *Procedia Comput. Sci.* **2017**, *120*, 422–429. [CrossRef]
18. Ngai, W.K.; Xie, H.; Zou, D.; Chou, K.L. Emotion recognition based on convolutional neural networks and heterogeneous bio-signal data sources. *Inf. Fusion* **2022**, *77*, 107–117. [CrossRef]
19. Saikia, A.; Mazumdar, S.; Sahai, N.; Paul, S.; Bhatia, D. Performance analysis of artificial neural network for hand movement detection from EMG signals. *IETE J. Res.* **2022**, *68*, 1074–1083. [CrossRef]
20. Usman, M.; Amin, R.; Aldabbas, H.; Alouffi, B. Lightweight challenge-response authentication in SDN-based UAVs using elliptic curve cryptography. *Electronics* **2022**, *11*, 1026. [CrossRef]
21. 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*, 7675. [CrossRef]
22. Zhu, G.; Liu, D.; Du, Y.; You, C.; Zhang, J.; Huang, K. Toward an intelligent edge: Wireless communication meets machine learning. *IEEE Commun. Mag.* **2020**, *58*, 19–25. [CrossRef]
23. Yazici, M.T.; Basurra, S.; Gaber, M.M. Edge machine learning: Enabling smart internet of things applications. *Big Data Cogn. Comput.* **2018**, *2*, 26. [CrossRef]
24. 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, Malmö, Sweden, 6–9 October 2020; pp. 1–8.

25. 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. Comput. Aided Des. Integr. Circuits Syst.* **2015**, *34*, 1537–1557. [CrossRef]
26. Sambas, A.; Vaidyanathan, S.; Zhang, X.; Koyuncu, I.; Bonny, T.; Tuna, M.; Alcin, M.; Zhang, S.; Sulaiman, I.M.; Awwal, A.M.; et al. 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. [CrossRef]
27. 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. *Appl. Sci.* **2021**, *11*, 788. [CrossRef]
28. 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. [CrossRef]
29. 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. Comput. Soc. Syst.* **2021**, *9*, 252–266. [CrossRef]
30. Guo, K.; Zeng, S.; Yu, J.; Wang, Y.; Yang, H. A survey of FPGA-based neural network accelerator. *ACM Trans. Reconfigurable Technol. Syst.* **2019**, *12*, 1–26. [CrossRef]
31. 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, Salt Lake City, UT, USA, 18–23 January 2018; pp. 4510–4520.
32. 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*, 56–63. [CrossRef]
33. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In Proceedings of the International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016; pp. 1–14.
34. Sapsanis, C.; Georgoulas, G.; Tzes, A.; Lymberopoulos, D. Improving EMG based classification of basic hand movements using EMD. In Proceedings of the 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society 13 (EMBC 13), Osaka, Japan, 3–7 July 2013; pp. 5754–5757.
35. Lobov, S.; Krilova, N.; Kastalskiy, I.; Kazantsev, V.; Makarov, V.A. Latent factors limiting the performance of sEMG-interfaces. *Sensors* **2018**, *18*, 1122. [CrossRef]
36. Tsinganos, P.; Cornelis, B.; Cornelis, J.; Jansen, B.; Skodras, A. Data augmentation of surface electromyography for hand gesture recognition. *Sensors* **2020**, *20*, 4892. [CrossRef]
37. Jeong, J.W.; Lee, W.; Kim, Y.J. A Real-Time Wearable Physiological Monitoring System for Home-Based Healthcare Applications. *Sensors* **2021**, *22*, 104. [CrossRef]
38. Chen, Y.; Xia, R.; Zou, K.; Yang, K. FFTI: Image Inpainting Algorithm via Features Fusion and Two-Steps Inpainting. *J. Vis. Commun. Image Represent.* **2023**, *1*, 103776. [CrossRef]
39. Byeon, Y.H.; Pan, S.B.; Kwak, K.C. Intelligent deep models based on scalograms of electrocardiogram signals for biometrics. *Sensors* **2019**, *19*, 935. [CrossRef]
40. Adib, A.; Zaerpour, A.; Lotfirad, 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 Reg. Sci. Technol.* **2021**, *189*, 103342. [CrossRef]
41. 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. *Biomed. Eng.* **2021**, *55*, 194–199. [CrossRef]
42. Fast Machine Learning Lab. Available online: https://github.com/fastmachinelearning/ (accessed on 1 February 2022).
43. Kang, S.; Kim, H.; Park, C.; Sim, Y.; Lee, S.; Jung, Y. sEMG-Based Hand Gesture Recognition Using Binarized Neural Network. *Sensors* **2023**, *23*, 1436. [CrossRef]
44. Westby, I.; Yang, X.; Liu, T.; Xu, H. FPGA acceleration on a multi-layer perceptron neural network for digit recognition. *J. Supercomput.* **2021**, *77*, 14356–14373. [CrossRef]
45. 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. *J. Syst. Archit.* **2021**, *115*, 101991. [CrossRef]
46. 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*, 3170. [CrossRef]
47. 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 Trans. Circuits Syst. II Express Briefs* **2016**, *64*, 1257–1261. [CrossRef]