Journal of
*Low Power Electronics and Applications*

MDPI

*Article*

# Low-Power Embedded System for Gait Classification Using Neural Networks

**Francisco Luna-Perejón *, Manuel Domínguez-Morales, Daniel Gutiérrez-Galán and Antón Civit-Balcells**

Architecture and Computer Technology Department, Universidad de Sevilla, E.T.S Ingeniería Informática, Reina Mercedes Avenue, 41012 Seville, Spain; mjdominguez@us.es (M.D.-M.); dgutierrez5@us.es (D.G.-G.); civit@us.es (A.C.-B.)

* Correspondence: fralunper@atc.us.es

Received: 19 March 2020; Accepted: 17 April 2020; Published: 1 May 2020


**Abstract:** Abnormal foot postures can be measured during the march by plantar pressures in both dynamic and static conditions. These detections may prevent possible injuries to the lower limbs like fractures, ankle sprain or plantar fasciitis. This information can be obtained by an embedded instrumented insole with pressure sensors and a low-power microcontroller. However, these sensors are placed in sparse locations inside the insole, so it is not easy to correlate manually its values with the gait type; that is why a machine learning system is needed. In this work, we analyse the feasibility of integrating a machine learning classifier inside a low-power embedded system in order to obtain information from the user's gait in real-time and prevent future injuries. Moreover, we analyse the execution times, the power consumption and the model effectiveness. The machine learning classifier is trained using an acquired dataset of 3000+ steps from 6 different users. Results prove that this system provides an accuracy over 99% and the power consumption tests obtains a battery autonomy over 25 days.

**Keywords:** machine learning; neural networks; gait analysis; embedded system

## 1. Introduction

Foot and ankle pain are very common in the population. Studies indicates that around 24% of people aged over 45 years report frequent foot pain [1]. Moreover, other studies indicate that more than 70% of the population over 65 years old present chronic foot pain [2].

It has also been demonstrated that abnormal foot postures and gait are associated with foot pain [3] as well as with lower limb injuries and pathologies [4]. Additionally, problems and disabilities associated with abnormal gait and foot posture include fractures, ankle sprain, pimple pain or plantar fasciitis, among others [5].

These previously named abnormalities due to bad foot postures have recently been related in several experiments to the pressure received at the base of the foot [4,6]. Therefore, a professional specialized in foot problems can perform a walking study of the patient's footprint in order to detect these problems, prevent the injuries occasioned by prescribing insoles and/or indicate physical exercises to correct them.

For that reason, it is very important to characterize the static foot posture and the foot function with a gait analysis. In that concern, there are available various methods in the literature [7].

The classic gait study consists of walking in a straight line through a sensorized surface that emulates a several meter long path—the surface measures and records the pressure obtained for each step during the

gait for posterior analysis. The main problem using this mechanism is the psychological component—the patient knows that he/she is being observed and walks, without any intention, in a different way (better or worse, it depends on the patient's mood). So, because of that, in many cases the recorded information does not correspond to the usual patient's way of walking [8].

To avoid the problems found in the classical methods (the psychological component mainly), many recent studies try to embed the sensorized surface in the patient's shoes [9–16].

These developments are mainly focused on designing an instrumented insole that includes pressure sensors, and demonstrate that these devices may have multiple applications in several fields such as in orthopaedic, orthoprosthetic, footwear designing, prostheses, pathology, or even in sports medicine, for the study of the most appropriate footwear in each athletic modality.

As detailed before, the use of instrumented insoles improves the data-recollection process during the gait while the patient is doing his daily-living activities (with freedom of movement and without space limitation). Nevertheless, to achieve good results collecting useful data, these insoles should have a good battery life; otherwise, data will be lost, and the gait analysis study will not be complete.

Additionally, the works developed until now use the footwear insole only to collect data and send it to a processing system like a smartphone or a computer. Due to that, the data is transmitted using a wireless connection in a continuous way and, therefore, the battery life is reduced significantly. Theoretically, if the information is processed locally inside the embedded system, the battery life increases because of the absence of data transmissions—works like that in References [17–20] demonstrate the battery-life improvement.

Recently, we developed an instrumented insole able to receive the pressure information obtained during the gait and send it to a computer via Bluetooth. Running in the computer, a local neural-network system classified the gait type as pronator, supinator or neutral and store that information [21]. Although there is no consensus on the terminology, we will use the common terms "pronation" to indicate when the foot undergoes greater lowering of the medial longitudinal arch and more medial distribution of plantar loading during gait and "supination" when the foot undergoes greater elevation of the medial longitudinal arch and more lateral distribution of plantar loading during gait [3] (see Figure 1).

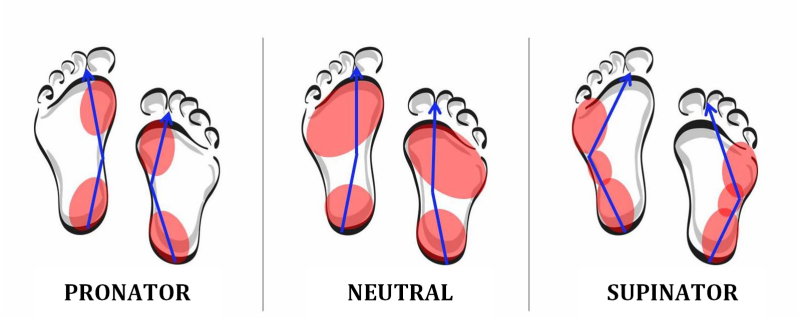


**Figure 1.** Gait type.

The main goal of that work was to study the feasibility of the proposed gait-type classification, without taking into account any battery-life restrictions. Although we demonstrated that our classification accuracy was better than that obtained in other projects, our work shared the problems related to short battery life.

So, the aim of this work is to the reduce the power-consumption requirements of the instrumented insole by implementing the neural-network classifier into the microcontroller attached to the instrumented

insole. To do that, several neural-networks architectures have been trained and tested with Tensorflow and Keras, using a database of 3000+ steps. We evaluate the effectiveness of the classifier in terms of the accuracy, among other metrics. After that, this architecture is compiled and integrated in the embedded system using STM32Cube.AI (artificial intelligence plugin used in STM32CubeIDE software for STMicroelectronics microcontrollers) in order to check the correct behaviour when running on the microcontroller, as well as to assess the power-consumption reduction when classifying with the low-power microcontroller.

The rest of the paper is divided in the following way—first, the acquisition and evaluation processes are described in the Materials and Methods section, presenting the used embedded system , the collected database, and evaluated the neural-networks architectures. Next, the results obtained after the training process with the different neural-networks architectures in Keras, the classification from the neural network deployed into the embedded system and the power-consumption study are detailed and explained in the Results and Discussion section. Finally, conclusions are presented.

## 2. Materials and Methods

As detailed in the previous section, the system used in this work requires an instrumented insole composed of a set of force sensitive resistors (FSRs) and a microcontroller for the acquisition step and for the final implementation.

Moreover, the main feature that makes the system to reduce drastically the power-consumption requirements consists in the implementation of the machine learning classifier in the microcontroller, avoiding the continuous data transmissions.

All these components and tools will be detailed in depth in this section, as well as the process followed from data acquisition to the final implementation.

### *2.1. Data Acquisition*

The data registered for the gait analysis consists of a set of pressure measures obtained through a footwear insole connected to an embedded device. After an in depth study of the walking process and the more adequate distribution of the sensors tested on previous works [21,22], seven FSRs are disposed in different parts of the foot. For each footstep, sensors samples at 50 hertz frequency since the first contact of the foot with the ground until the moment the foot is lifted. Thus, the information stored refers to the medium pressure received by each sensor during each step. These values are normalized after each step ends using the sensor's value with the highest pressure received as 100% pressure and modifying the other sensors' values to a percentage value relative to that sensor.

Next, both the footwear insole and the dataset obtained after the acquisition phase are detailed.

#### 2.1.1. Footwear Insole

The hardware system used for the acquisition is based on a low-power consumption microcontroller, seven force sensitive resistors (FSRs) and a low-energy Bluetooth module. The selected microcontroller was a STMicroelectronics MCU used for the acquisition and testing phases (model STM32L476RG, operating at a frequency up to 80 MHz, with 1 Mbyte of flash memory and 128 Kbytes of SRAM), with features that allow real-time capabilities, digital signal processing and low-power operation. The FSRs were connected to the analog inputs of the microcontroller using a voltage divider with a 10 KΩ resistor. These sensors provide information about the maximum pressure point and the load forces. Finally, a HM-10 BLE (Bluetooth Low-Energy) module was connected to the microcontroller as a wireless communication port to send the information to the computer (see Figure 2).

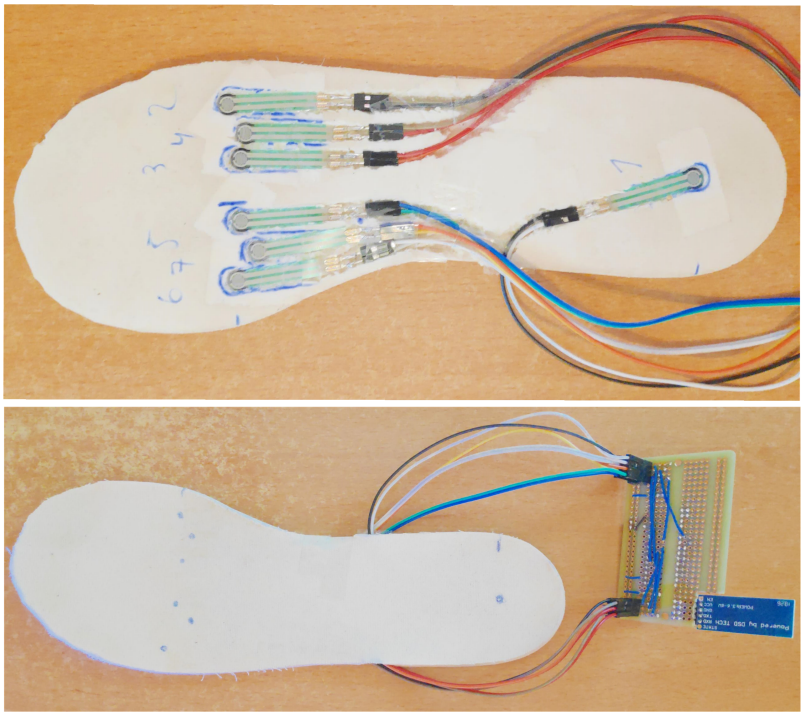

**Figure 2.** (Left) force sensitive resistors (FSRs) location and microcontroller with wireless module.

The location of the sensors in the footwear insole was established based on the anatomy of the foot, the types of footprints to classify (shown in the previous section) and the tests performed in previous works [22]. Results showed that the metatarsus area gives more information about the footprint types than the other areas, so six sensors were placed in this foot region while one last sensor was placed in the heel to determine the moments of contact with the ground and foot lifted (see Figure 2).

In order to recover the pressure measures for each footstep and obtain an appropriate dataset, the microcontroller implements a FreeRTOS https://www.freertos.org/ based firmware to manage the sensors reading and the communication without information loss. FreeRTOS allows us to manage the implemented functions using OS functionalities, such as semaphores, queues and tasks. The sent data was adequately collected by a computer application. In order to understand the data acquisition process, a graphic diagram is shown in Figure 3-up. In this diagram, the other two phases of this work are also shown. They will be explained later in this paper.

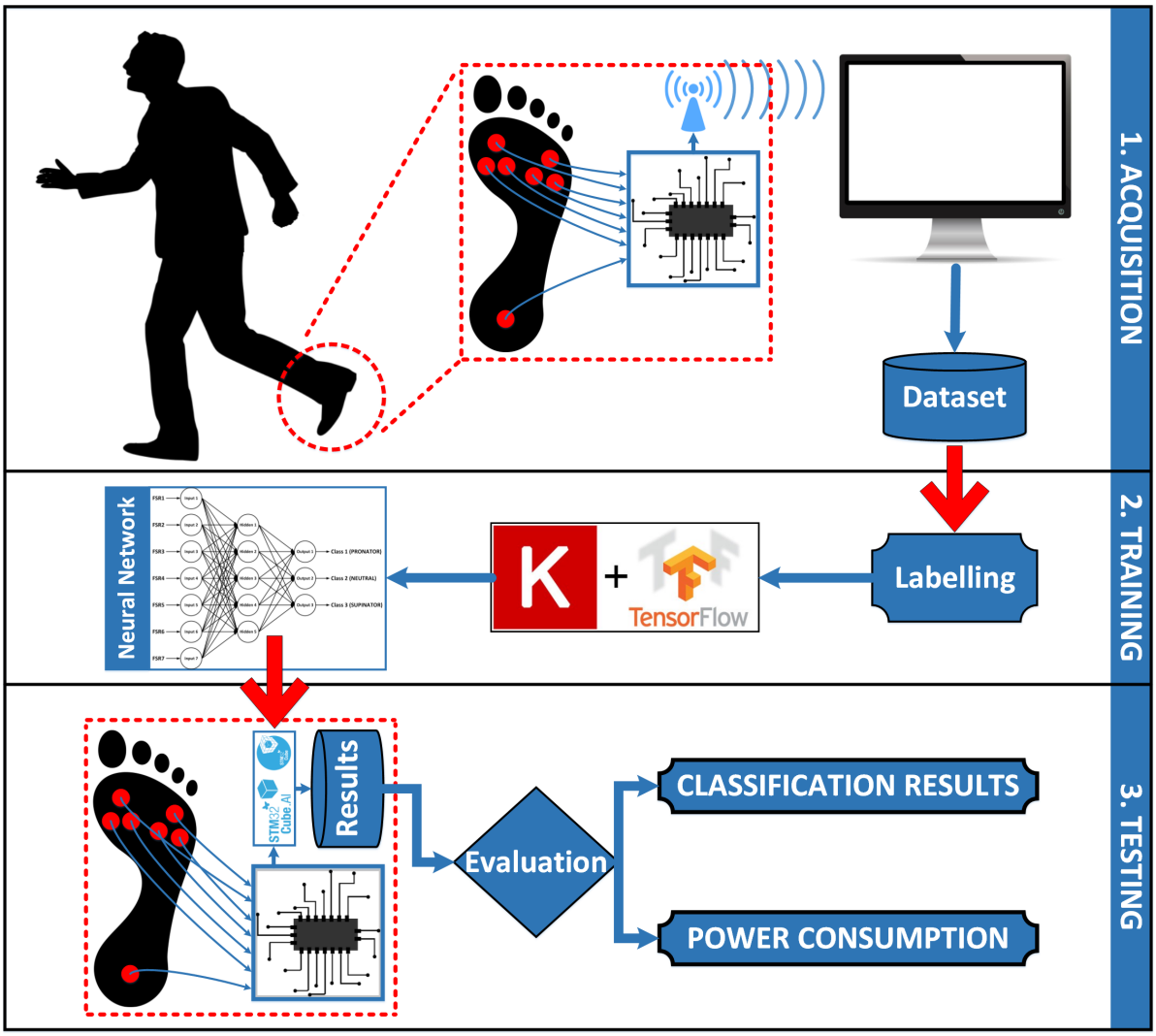


**Figure 3.** Full system implemented in this work: (**up**) data acquisition phase, (**middle**) training phase and (**bottom**) testing phase.

Once the information is stored, it is important to give further details on the collected database .

### 2.1.2. Dataset

To obtain a useful database for this work, we need users with different footprint characteristics. To be sure their footprint type has been correctly identified, only previously diagnosed patients have been used. Thus, finally, we recruited six volunteers to acquire the dataset, two users for each type of footprint; that is, two pronators, two supinators and two users with neutral gait. Although these volunteers had been diagnosed previously, we also used a classical pressure platform to verify their footprint type.

Even though the acquisition system (see Figure 3-up) allows the user to configure the sensors acquisition frequency, but the dataset was elaborated with samples taken at 50 Hz (in the next sections we will show that results justify that there is no need for using a higher frequency). With this configuration, the total number of stored footprint samples was approximately 3100, 1020 $\pm$ 90 samples per each type. The results of the data acquisition phase are further detailed in the Results and Discussion section.

### 2.2. Artificial Neural Network Classifier

Artificial Neural Networks (ANNs) have been used in several works recently to find the relationship between some input data and the desired response at the system output (called the system's inference or classification). This machine learning mechanism is very useful in applications where there are large amounts of input data and the relationship between that data and the expected output cannot be easily appreciated [23]. Thus, after an initial 'training' phase, the ANN is configured with a set of weights at both outputs and inputs of the different neuron layers with which the desired outputs can be obtained. ANNs have been demonstrated to obtain very good results in previous works, especially when used as a supervised machine learning method [24–26].

Their structure, based solely on arithmetic operations (except perhaps in inference phases in classification problems), allows them to be combined with other architectures, making them a fundamental component in several Deep Learning algorithms. It is also possible to create very efficient implementations, which can be optimized for low performance devices, such as low-power microcontrollers [27]. This fact allows acceptable execution times with very low power consumption. In this section we describe the ANN architectures analysed in terms of effectiveness, as well as their performance when embedded in a low-power device.

#### 2.2.1. Architecture Design

An ANN architecture with three layers is used in this gait classification study (see Figure 4). The first layer, that is, the input, contains seven nodes that receive information from the different FSRs, for each footstep. The last layer, the output, consisting of three nodes that return the degree of confidence of a sample to belong to one of the three footprint classes (supinator, pronator or neutral). A final output function called softmax is used. It implements a multi-class sigmoid and is typically used to normalize the results of the network output layer, limiting each output to the 0 to 1 range. The sum of the values of all the output nodes for this function is always 1. This allows us to interpret the output directly as a probability or confidence. Thus, the predicted class would be that with a greater confidence. The intermediate layer, called the hidden layer, is connected to the input and the output layers. Each node of this layer receives the information of all the seven input nodes, processes it and transmits its result to the three nodes in the output layer. Thus it is called a Fully Connected or Dense layer. In this study, architectures with different numbers of nodes in the hidden layer were considered, in order to reduce the complexity of the architecture while maintaining a good classification effectiveness. Therefore we have to look for a high-accuracy network with low complexity to implement it inside a microcontroller as this implies less memory usage and a smaller power consumption.

In this study, TensorFlow https://www.tensorflow.org together with Keras https://keras.io/ have been used to implement, train and test the architectures with the previously detailed dataset (see Figure 3-middle for a global description of the training phase). Tensorflow is a library created by Google for distributed numerical computation, that allows to design, train, evaluate and run models based on neural networks. Keras is a high-level API that simplifies model implementation with Tensorflow, by efficiently managing the connection between model layers and simplifying the Tensorflow code. The resulting models are compatible with STM32Cube utilities, allowing the creation of a C-compiled version that can be embedded in STM32 microcontrollers.

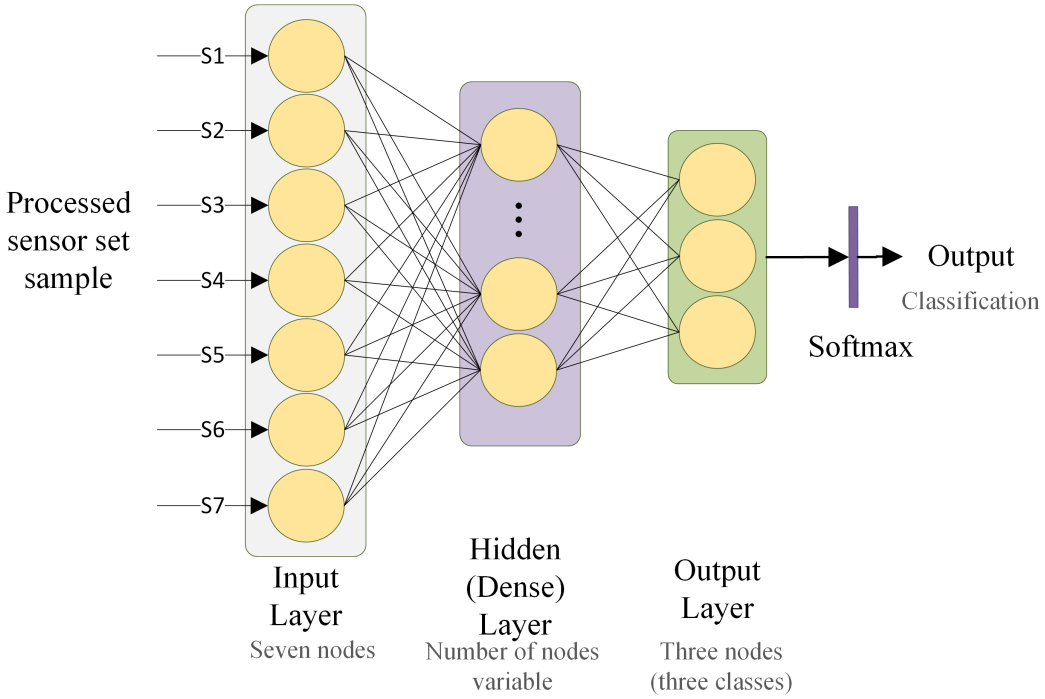


**Figure 4.** Artificial Neural Network (ANN) architecture used in the study for classification.

#### 2.2.2. Embedded Model Analysis

After the acquisition and training phases (see Figure 3-up and Figure 3-middle, respectively), the ANN is integrated into the embedded system in order to evaluate whether it provides the same effectiveness results as the original implementation on a general-purpose computer, as well as the improvements in energy consumption. For that purpose, STM32CubeIDE development environment https://www.st.com/en/development-tools/stm32cubeide.html was used. It provides tools for power consumption analysis when using different components and modules. Additionally, their STM32Cube.AI expansion plug-in https://www.st.com/en/embedded-software/x-cube-ai.html allows to generate C-compiled versions of pre-trained Neural Networks models, optimized for STM32 microcontrollers.

To verify that the model effectiveness is maintained after conversion to a C-compiled version and integration into the microcontroller, we compared the output confidence results for each class with those obtained with the original Keras model. We used the same MCU that the one used for the acquisition phase for the integration and performance analysis of the trained models.

For the power efficiency analysis, two different scenarios were considered (see Figure 5). In the first one, the embedded system collects data from the sensors at 50Hz and sends it via Bluetooth (every 20 ms); the information is received by the host, which processes and classifies it using an external ANN classifier (this scenario is similar than the one used for the acquisition phase, but now the ANN classifier is implemented in the host). In this case, the system spends almost 4ms for each data transmission (sending 56 bytes at 115,200 bauds), which is 20% of the total time; and, moreover, the transmission process takes more than 43 mA of power consumption (much more than the average power consumption of the system).

In the second scenario, the classification process is done inside the embedded system with the internal ANN implementation. The information read from the sensors is processed and stored internally and, only once per step, the ANN classifies the footstep type and, after that, the classification result is transmitted to the host using Bluetooth. Two main improvements are obtained in this second scenario—first, the transmitted data size is significantly reduced (from 56 bytes to 3 bytes) and, thus, the transmission

time is much lower that in the first scenario (0.2 ms); and, second, the transmission is only done after each classification (only one per step), so the number of transmissions is much less than that in the first scenario. However, the number of transmissions depends on the gait cadence of the user, so it must be studied in depth.

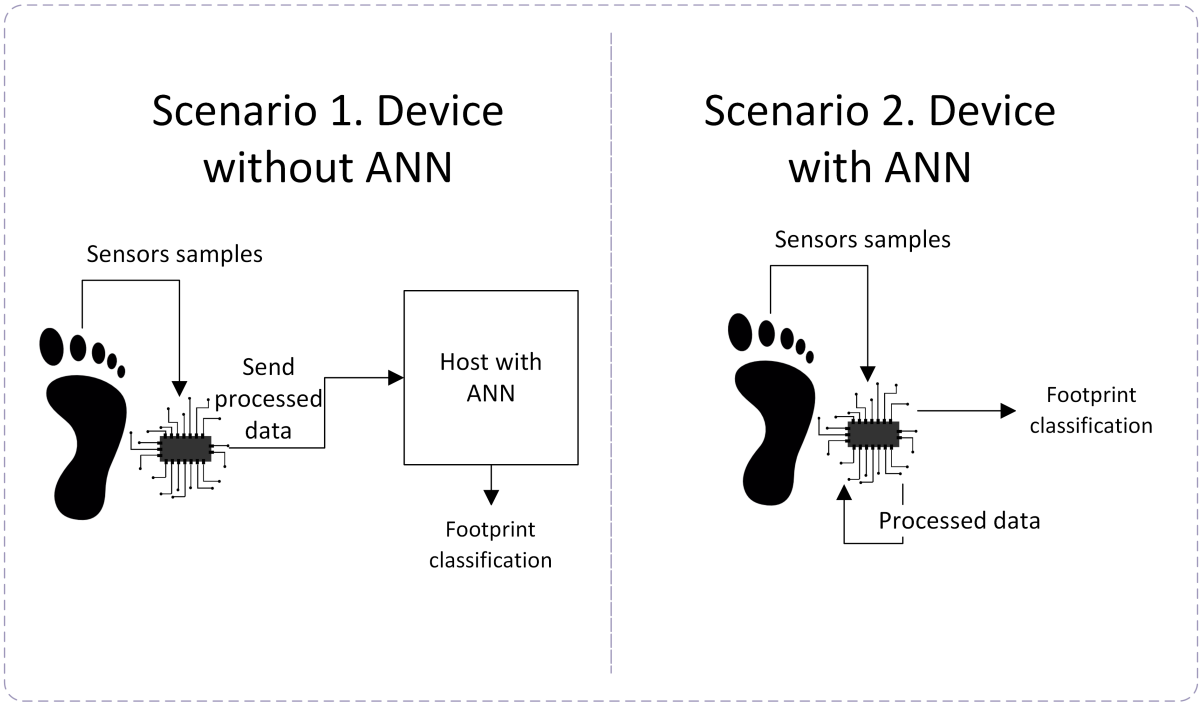


**Figure 5.** Two scenarios proposed for power consumption analysis.

To easily understand both scenarios, the implemented firmwares are described. The first one (see Figure 6-left) was used to acquire the data for the database and it was also used as "scenario 1" for the final power-consumption comparison. The second one (see Figure 6-right) is used for the embedded ANN implementation and hence it corresponds to "scenario 2" in the testing phase.

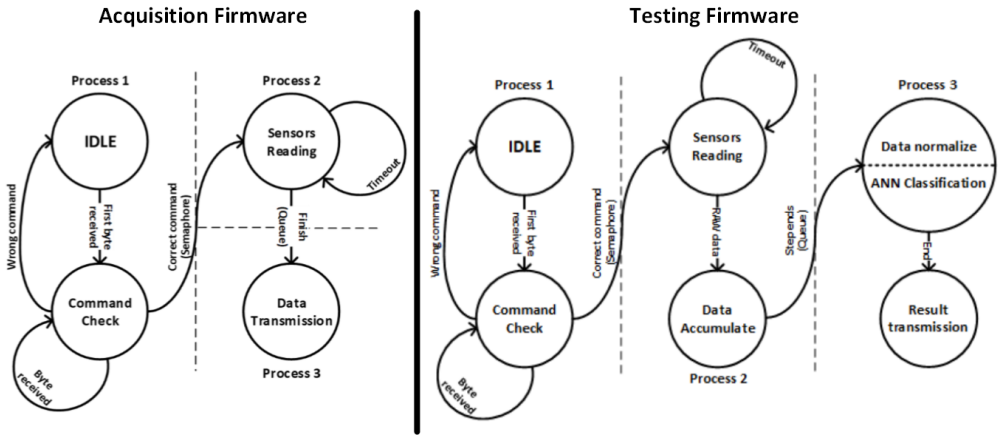


**Figure 6.** Implemented Firmwares for the acquisition (**left**) and testing phase (**right**).

The firmware implemented in the MCU for the acquisition phase is a FreeRTOS based implementation that allows a bi-directional serial communication with the bluetooth module (process 1 for reception

and process 3 for transmission) and a periodic sensor readings with data transformation (process 2), using a binary semaphore and a queue to communicate between these process. So, in the acquisition phase, there is no recording inside the MCU memory; but, after each reading, the data is packed and sent to an external computer where the information is stored in a database.

And, for the testing phase, the implemented firmware is also FreeRTOS based and uses periodic readings, value normalization and performs an ANN classification after each steps ends. The result of this classification is transmitted (once per step) to the external computer.

## 3. Results and Discussion

In this section, the results obtained at the end of each phase are detailed. For the acquisition phase (see Figure 3-up), the collected dataset is presented. For the training step (see Figure 3-middle), the classification results are detailed and, finally, the embedded model accuracy and the power consumption study are presented as results of the testing phase (see Figure 3-bottom).

### *3.1. Dataset*

The dataset used in this study was split for training and assessment purposes. We used the Hold-out technique, by randomly selecting a sample subset for the training of the models, and using the remaining subset to validate the model performance. A subset with the 85% of dataset samples was used for training, while the remaining 15% subset was used for evaluation. The distribution was made to ensure that there was a balanced percentage of each type of footprint in both subsets. Table 1 shows the distribution.

**Table 1.** Dataset distribution for each subset.

| Subset | Neutral | Pronator | Supinator | Total |
|---|---|---|---|---|
| **Total** | 1067 | 1129 | 928 | 3124 |
| **Training** | 917 | 955 | 784 | 2656 |
| **Test** | 150 | 174 | 144 | 468 |

### *3.2. ANN Model Assessment*

This section presents the trained ANN architectures, their implementation and training specifications and the effectiveness evaluation results.

#### 3.2.1. ANN Architectures and Parameters

We analyse the architecture introduced in Section 2.2.1 with different numbers of nodes in its hidden layer. In previous studies, 5 hidden nodes were used in the hidden layer, based on [28]. For the current analysis, we assess the effectiveness by reducing the number of nodes looking for an improvement of classification times and power consumption reduction. Sigmoid function was used as activation function for the nodes in the hidden layer, while Rectified Linear Unit (ReLU) function was used for the output layer nodes. The model training was performed with a learning rate of 0.001, a batch size of 8 and 75 epochs. The used optimizer was a Root Mean Square Prop (RMSProp).

#### 3.2.2. Effectiveness Results

We compared the effectiveness using different metrics—accuracy, sensitivity (also named macro recall), specificity, macro precision and macro F1-score [29]. This last metric measures the relation is the harmonic mean of macro precision and macro recall.

$$Specificity = \sum_c \frac{TN_c}{TN_c + FP_c}, c \in classes \quad (1)$$

$$Precision_m = \sum_c \frac{TP_c}{TP_c + FP_c}, c \in classes \quad (2)$$

$$Recall_m(sensitivity) = \sum_c \frac{TP_c}{TP_c + FN_c}, c \in classes \quad (3)$$

$$F1 - score_m = 2 * \frac{precision_m * recall_m}{precision_m + recall_m}, \quad (4)$$

where $m$ index refers to macro metric and $classes = \{Pronator, Neutral, Supinator\}$. The term $TP_c$ refers to the number of samples with class $c$ that were classified correctly as $c$ by the trained model. $TN_c$ denotes the number of samples with a different class of $c$ that were not classified as $c$. The term $FP_c$ determines the set of samples with different class of $c$ that were classified as $c$ by the model and, finally, $FN_c$ refers to the set of samples with class $c$ that were wrongly classified as other different class.

The results obtained with each architecture are shown in Table 2. As can be seen, the reduction in the number of nodes not only maintains effectiveness, but also improves when compared to larger models. The greatest effectiveness is achieved with three nodes in the hidden layer. This may be due to the fact that this hidden layer reduction diminishes the so-called over-fitting phenomenon [30], preventing the model from adjusting too closely to the particular characteristics of the used training subset. The architecture, however, can assimilate enough footprint characteristics even with only two hidden nodes with slightly worse effectiveness.

**Table 2.** Metrics results with different numbers of nodes in the hidden layer. The model trained with only one node in its hidden layer is not able of distinguish one footprint class, so specificity and sensitivity cannot be obtained for this case.

| Nodes in Hidden Layer | Accuracy | Precision | F1-Score | Specificity | Sensitivity |
|---|---|---|---|---|---|
| Five nodes | 0.987 | 0.987 | 0.994 | 0.987 | 0.987 |
| Four nodes | 0.994 | 0.994 | 0.997 | 0.994 | 0.994 |
| Three nodes | 0.996 | 0.996 | 0.998 | 0.996 | 0.996 |
| Two nodes | 0.991 | 0.992 | 0.996 | 0.992 | 0.992 |
| One node | 0.678 | 0.653 | 0.842 | - | - |

### *3.3. Embedded System Results*

In this section, the results obtained from the embedded models running in a low power STM32L476RG board are presented. Three aspects were analysed in relation to the embedded device performance. First, we assessed the accuracy of the C-compiled model obtained with CUBE-AI package extension. Second, we estimated the inference time, that is, the execution times obtained when the embedded model classifies a sample. Finally, we calculated the consumption of the device for the two scenarios established in the Methods section, that is, when the device has the integrated ANN model and when it only sends the data to an external computer with higher computing performance and less power usage limitations.

#### 3.3.1. Embedded Model Accuracy

We analysed the similarity of the outputs from the Keras model and those from its C-compiled version. For this purpose, we assessed the differences on the inference outputs of the models. This was obtained by calculating the relative L2 error:

$$e = \frac{\|F_{generated} - F_{original}\|}{\|F_{generated}\|}, \quad (5)$$

where $F_{generated}$ is the flattened array of the generated model last output layer and $F_{original}$ the flattened array of the original model. In other words, we compare the reliability results returned by the last layer of the two model implementations, prior to classification.

Results for each model are presented in Table 3. We showed the results when each C-compiled model was compressed to occupy less flash memory in the microcontroller. Compression is carried out using a weight sharing-based algorithm. A clustering technique (K-means) is used to calculate values centroids for the layer weights and bias. The compression with factor x4 uses 256 centroids codified on 8 bits, while the compression with factor x8 uses 16 centroids codified on 4bits. In most of the models the L2 error was very low, under $6.8 \times 10^{-7}$, which implies the C-compiled models maintain a very close classification behaviour. It should be noted that, for the models with the highest number of nodes in the hidden layer, their more compressed version provides reliability values relatively further from the corresponding model in Keras. Increasing the complexity of the hidden, fully connected layer may have caused this effect. The results obtained may also have been influenced as a consequence of the ov-erfitting effect mentioned above, which could imply an improvement in effectiveness, due to the fact that specific features to classify particular cases of the training set could be forgotten. This may have occurred with the four hidden node compressed model, which has improved its accuracy over the original Keras model. However, the best results continue to be found with the model with three nodes in the hidden layer, obtaining an L2 lower than $1.0 \times 10^{-8}$.

**Table 3.** L2 error for each model (trained model vs. generated c-model) with different compression factors.

| Nodes in Hidden Layer | Not-Compressed | x4 Compression | x8 Compression |
|---|---|---|---|
| Five nodes | $6.816 \times 10^{-8}$ | $6.816 \times 10^{-8}$ | $9.629 \times 10^{-2}$ |
| Four nodes | $2.586 \times 10^{-7}$ | $2.586 \times 10^{-7}$ | $2.012 \times 10^{-1}$ |
| Three nodes | $9.099 \times 10^{-8}$ | $9.099 \times 10^{-8}$ | $9.099 \times 10^{-8}$ |
| Two nodes | $4.346 \times 10^{-7}$ | $4.346 \times 10^{-7}$ | $4.346 \times 10^{-7}$ |
| One node | $7.191 \times 10^{-7}$ | $7.191 \times 10^{-7}$ | $7.191 \times 10^{-7}$ |

#### 3.3.2. Execution Times

We estimated the time spent on classifying a sample, that is, the execution time for one ANN classification. This value is important to determine the power consumption for the process 3 in the scenario 2 (see Figure 6-right). We also analysed the results for each compressed model version. The results, which can be seen in Table 4, show that there is a slight variation in power consumption as the number of nodes decreases. The compressed version of each model does not seem to disturb the execution times, except again in the case of the models with the highest number of nodes in the hidden layer, which take longer to perform a classification. Considering the previous results, the architecture with the greatest effectiveness for this problem and with a good classification time is the one with three nodes in its hidden layer, which in turn can be compressed by a factor $\times 8$ without altering performance.

Regarding the power consumption analysis the ANN with three nodes in the hidden layer and a $\times 8$ compression is used.

#### 3.3.3. Power Consumption Analysis

As detailed in previous sections, the main differences of both analysed scenarios are the communications frequency and the amount of data transmitted (see Figure 5).

**Table 4.** Estimated execution times per classification (milliseconds), of each model integrated and running in STM32L476RG board.

| Nodes in Hidden Layer | Not-Compressed | ×4 Compression | ×8 Compression |
|---|---|---|---|
| Five nodes | 0.071 | 0.071 | 0.075 |
| Four nodes | 0.067 | 0.067 | 0.070 |
| Three nodes | 0.061 | 0.061 | 0.061 |
| Two nodes | 0.056 | 0.056 | 0.056 |
| One node | 0.052 | 0.052 | 0.052 |

In the first scenario (see Figure 5-left), every 20 ms (50 Hz reading frequency) the embedded system takes: 0.07 ms reading the sensors' values (that consumes 6.1 mA), 3.9 ms transmitting via Bluetooth (that consumes 43.16 mA) and the rest of the time (16.03 ms) in sleep mode (that consumes only 18 μA). So, using an average button battery of 125 mAh capacity, the system has a battery life of 14 h.

In the second scenario, the calculation is not that easy because the system only transmits information once per step. So, two possibilities are evaluated: first, sensors are read but the step is not finished; and, second, sensors are read and the step is finished. In the first possibility (step is not end yet): sensors are read and values are accumulated (in this case, the embedded system does not classify and does not transmit). In the second possibility: sensors are read, values are accumulated, the final amount of data for the full step is normalized and classified using the ANN; and, finally, the classification result is transmitted.

Both possibilities are very different in the power-consumption analysis—the second one spends much more power that the first one because of the ANN classification and the transmission.

Moreover, if we compare the power-consumption between the first scenario (always transmitting) and the second possibility of the second scenario (transmission only once per step), there is a big difference too—in the second scenario, only one data transmission per step is done and the time spent in the transmission is less than in the first scenario because the amount of data transmitted is much lower (3 bytes versus 56 bytes), taking only 0.2 ms in the transmission process.

So, evaluating the second scenario, 0.07 ms are spent for sensors' reading (6.1 mA), 0.061 ms are spent for the ANN classification (255.1 μA), 0.2 ms are spent for the transmission (43.16 mA) and the rest of the time (19.66 ms) the system is in sleep mode (18 μA). However, if the step is not ended, the system does not transmit and there is no classification process (only periodic sensors' reading); so, during the time spent in these two phases, the system is in sleep mode too.

The first scenario is relatively easy to evaluate in the power-consumption study, but the second scenario is much more difficult because it depends on the user's gait cadence. So, in order to obtain a more accurate power consumption study for it, the gait cadence of the user must be evaluated. Using the information obtained after the study done in [31], we can observe than a gait cadence less than 100 steps/min corresponds to a low intensity (walking), a cadence between 100 and 130 steps/min corresponds to medium intensity (jogging) and a cadence higher than 130 steps/min corresponds to high intensity (running).

Thus, in our case, we have analysed the power consumption with a sensors' reading frequency fixed at 50 Hz and cadence values between 30 steps/min and 160 steps/min, obtaining the results presented in Table 5. The first column indicates the gait cadence in number of steps per minute and varies from 30 to 160; in the second one, the time spent for each step (in seconds) for each gait cadence is detailed; the third one calculates the total number of samples collected for each step using the data from the previous columns and using only one foot (as one instrumented insole collects information from one foot).

Instead of evaluating the power consumption of both possibilities in the second scenario (step ends or not), for this power-consumption estimation we assume that all the sensors' readings imply a classification

step and a data transmission and, depending on the number of samples for each step, the consumption of those processes are multiplied by a factor (between 0 and 1) that indicates the proportion of transmissions depending on the gait cadence. For example, if we always transmit, this factor will be 1; or, if we have 10 samples per step, we transmit only 10% times, so this factor is 0.1. So, the fourth column of Table 5 represents the result of the power consumption of the classification and transmission processes (43.16 mA approx.) multiplied by this factor. Finally, the fifth and sixth columns indicate the average consumption of the full system (in µA) and the final battery life (in hours), respectively.

**Table 5.** Number of steps, transmissions and power consumption varying the gait cadence of the user.

| Cadence (Steps/min) | Sec. Spent for Each Step | # Samples for Each Step (1 foot) | Tx Power Consumption for Each Reading (µA) | System Average Consumption (µA) | Battery Life (h) |
|---|---|---|---|---|---|
| 30 | 2.00 | 50.00 | 0.86 | 75.11 | 1639 |
| 40 | 1.50 | 37.50 | 1.15 | 85.26 | 1488 |
| 50 | 1.20 | 30.00 | 1.44 | 95.41 | 1297 |
| 60 | 1.00 | 25.00 | 1.73 | 105.57 | 1171 |
| 70 | 0.86 | 21.43 | 2.01 | 115.37 | 1071 |
| 80 | 0.75 | 18.75 | 2.30 | 125.53 | 984 |
| 90 | 0.67 | 16.67 | 2.59 | 135.68 | 909 |
| 100 | 0.60 | 15.00 | 2.88 | 145.84 | 845 |
| 110 | 0.55 | 13.64 | 3.17 | 155.64 | 792 |
| 120 | 0.50 | 12.50 | 3.45 | 165.80 | 743 |
| 130 | 0.46 | 11.54 | 3.74 | 175.95 | 709 |
| 140 | 0.43 | 10.71 | 4.03 | 186.11 | 670 |
| 150 | 0.40 | 10.00 | 4.32 | 196.26 | 636 |
| 160 | 0.38 | 9.38 | 4.60 | 206.06 | 605 |

Hence, as can be seen in Table 5, in the worst case (highest gait cadence evaluated) the battery life exceeds 25 days. So, it improves the battery life of the first scenario by more than 43 times (an improvement of 4321%). This comparison can be observed in Figure 7.

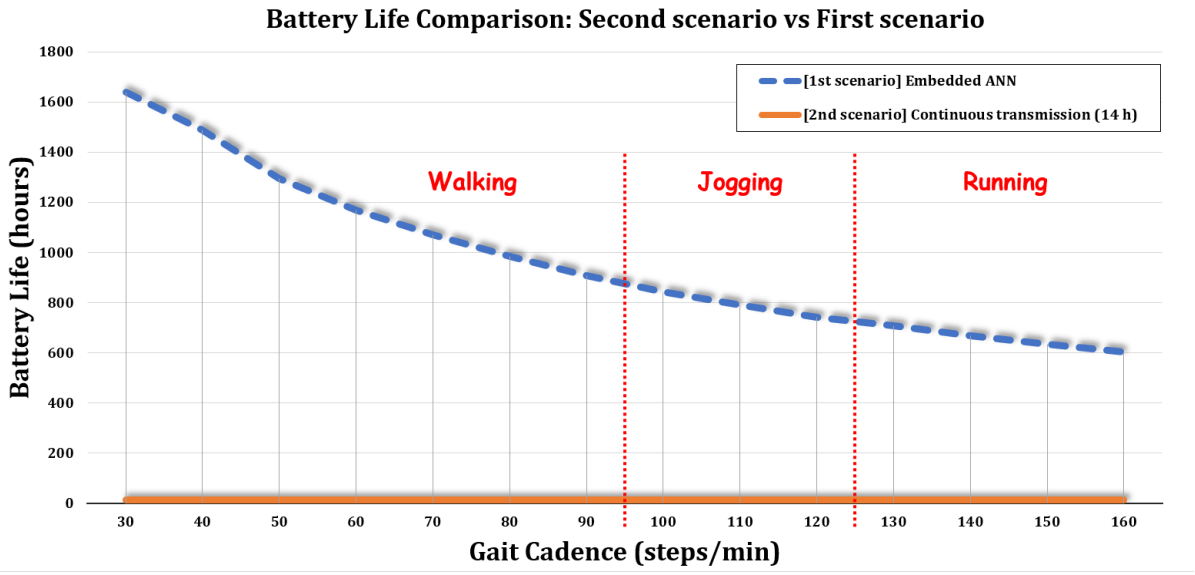


**Figure 7.** Battery life (in hours) with the power consumption estimation for each scenario.

The results presented in Figure 7 show that the higher the gait cadence, the lower the battery life is. Despite this, the life of the used battery (125 mAh) even for cases in which the user is running high enough to allow a biomechanical gait study without recharges.

## 4. Conclusions

In this work, a performance analysis of a low-power footwear insole for the detection of abnormal foot postures is presented. The device implements an embedded Machine Learning model based on ANN for real-time footprint type inference. The inputs of the model consist of average FSR measures obtained during a footstep, and the outputs correspond to the three gait types described in the Introduction section—pronator, supinator and neutral.

First, a model study was performed. The effectiveness of the ANN architecture, consisting of three neural layers, was assessed using a different number of nodes in the hidden layer. The architecture with three nodes obtained the best results, with effectiveness metrics above 99.6%. The architectures with a greater number of nodes showed slightly less classification ability, possibly due to overfitting the training dataset.

Finally, as the main point of the study, a complete analysis of the classifier performance has been performed when it is integrated into a low-power embedded device. The L2 error obtained when comparing the Keras and the C-compiled model outputs showed that the conversion does not have a significant impact on the effectiveness of the model, even when the model is compressed, thus saving memory space on the microcontroller. This can be an important aspect if we intend to include other models with different functionalities in the device in future works. Regarding the inference execution times, the best model is able to classify a footstep sample in 0.61 ms, even when it is compressed. This is much less than the time needed to read a sample of the insole sensors, thereby achieving real-time execution. Based on this, and considering that the classifier execution and result transmission only take place when a full step is performed, the battery life estimation is over 25 days (considering the higher gait cadence).

**Author Contributions:** Conceptualization, M.D.-M., D.G.-G. and A.C.-B.; methodology, F.L.-P., M.D.-M. and A.C.-B.; software, F.L.-P. and D.G.-G.; validation, F.L.-P., M.D.-M. and A.C.-B.; investigation, F.L.-P., D.G.-G. and M.D.-M.; resources, F.L.-P. and M.D.-M.; writing–original draft preparation, F.L.-P., D.G.-G. and M.D.-M.; writing–review and editing, D.G.-G. and A.C.-B.; supervision, M.D.-M. and A.C.-B.; funding acquisition, A.C.-B. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Robotics and Computer Technology Lab. (RTC) of the Universidad de Sevilla, Spain.

**Acknowledgments:** This work has been supported by the Robotics and Computer Technology Lab. (RTC) of the Universidad de Sevilla, Spain.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| OS | Operating System |
| ANN | Artificial Neural Network |
| FSR | Force Sensitive Resistors |
| BLE | Bluetooth Low Energy |
| MCU | Microcontroller |
| ReLU | Rectified Linear Unit |

## References


1. Thomas, M.J.; Roddy, E.; Zhang, W.; Menz, H.B.; Hannan, M.T.; Peat, G.M. The population prevalence of foot and ankle pain in middle and old age: A systematic review. *Pain* **2011**, *152*, 2870–2880. [CrossRef]
2. Menz, H.B. Chronic foot pain in older people. *Maturitas* **2016**, *91*, 110–114. [CrossRef] [PubMed]
3. Menz, H.B.; Dufour, A.B.; Riskowski, J.L.; Hillstrom, H.J.; Hannan, M.T. Association of planus foot posture and pronated foot function with foot pain: The Framingham foot study. *Arthritis Care Res.* **2013**, *65*, 1991–1999. [CrossRef]
4. Buldt, A.K.; Allan, J.J.; Landorf, K.B.; Menz, H.B. The relationship between foot posture and plantar pressure during walking in adults: A systematic review. *Gait Posture* **2018**, *62*, 56–67. [CrossRef] [PubMed]
5. Perry, J.; Davids, J.R. Gait analysis: Normal and pathological function. *J. Pediatr. Orthop.* **1992**, *12*, 815. [CrossRef]
6. Buldt, A.K.; Forghany, S.; Landorf, K.B.; Levinger, P.; Murley, G.S.; Menz, H.B. Foot posture is associated with plantar pressure during gait: A comparison of normal, planus and cavus feet. *Gait Posture* **2018**, *62*, 235–240. [CrossRef]
7. Razeghi, M.; Batt, M.E. Foot type classification: A critical review of current methods. *Gait Posture* **2002**, *15*, 282–291. [CrossRef]
8. Frelih, N.G.; Podlesek, A.; Babič, J.; Geršak, G. Evaluation of psychological effects on human postural stability. *Measurement* **2017**, *98*, 186–191. [CrossRef]
9. Morris, S.J.; Paradiso, J.A. Shoe-integrated sensor system for wireless gait analysis and real-time feedback. In Proceedings of the Second Joint 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society, Houston, TX, USA, 23–26 October 2002; Volume 3, pp. 2468–2469.
10. Bamberg, S.J.M.; Benbasat, A.Y.; Scarborough, D.M.; Krebs, D.E.; Paradiso, J.A. Gait analysis using a shoe-integrated wireless sensor system. *IEEE Trans. Inf. Technol. Biomed.* **2008**, *12*, 413–423. [CrossRef]
11. Shu, L.; Hua, T.; Wang, Y.; Li, Q.; Feng, D.D.; Tao, X. In-shoe plantar pressure measurement and analysis system based on fabric pressure sensing array. *IEEE Trans. Inf. Technol. Biomed.* **2010**, *14*, 767–775.
12. Wahab, Y.; Mazalan, M.; Bakar, N.; Anuar, A.; Zainol, M.; Hamzah, F. Low power shoe integrated intelligent wireless gait measurement system. *J. Phys. Conf. Ser. IOP Publ.* **2014**, *495*, 13–14. [CrossRef]
13. Crea, S.; Donati, M.; De Rossi, S.M.M.; Oddo, C.M.; Vitiello, N. A wireless flexible sensorized insole for gait analysis. *Sensors* **2014**, *14*, 1073–1093. [CrossRef] [PubMed]
14. Talib, N.; Rahman, M.; Najib, A.; Noor, M. Implementation of Piezoelectric Sensor in Gait Measurement System. In Proceedings of the 2018 8th IEEE International Conference on Control System, Computing and Engineering (ICCSCE), Penang, Malaysia, 23–25 November 2018; pp. 20–25.
15. Oshimoto, T.; Abe, I.; Kikuchi, T.; Chijiwa, N.; Yabuta, T.; Tanaka, K.; Asaumi, Y. Gait Measurement for Walking Support Shoes with Elastomer-Embedded Flexible Joint. In Proceedings of the 2019 IEEE/SICE International Symposium on System Integration (SII), Paris, France, 14–16 January 2019; pp. 537–542.
16. Ngamsuriyaroj, S.; Chira-Adisai, W.; Somnuk, S.; Leksunthorn, C.; Saiphim, K. Walking gait measurement and analysis via knee angle movement and foot plantar pressures. In Proceedings of the 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE), Nakhonpathom, Thailand, 11–13 July 2018; pp. 1–6.
17. Dominguez-Morales, J.P.; Rios-Navarro, A.; Dominguez-Morales, M.; Tapiador-Morales, R.; Gutierrez-Galan, D.; Cascado-Caballero, D.; Jimenez-Fernandez, A.; Linares-Barranco, A. Wireless sensor network for wildlife tracking and behavior classification of animals in Doñana. *IEEE Commun. Lett.* **2016**, *20*, 2534–2537. [CrossRef]
18. Henkel, J.; Pagani, S.; Amrouch, H.; Bauer, L.; Samie, F. Ultra-low power and dependability for IoT devices (Invited paper for IoT technologies). In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Lausanne, Switzerland, 27–31 March 2017; pp. 954–959.
19. Andres-Maldonado, P.; Ameigeiras, P.; Prados-Garzon, J.; Ramos-Munoz, J.J.; Lopez-Soler, J.M. Optimized LTE data transmission procedures for IoT: Device side energy consumption analysis. In Proceedings of the 2017 IEEE International Conference on Communications Workshops (ICC Workshops), Paris, France, 21–25 May 2017; pp. 540–545.

20. Deepu, C.J.; Heng, C.H.; Lian, Y. A hybrid data compression scheme for power reduction in wireless sensors for IoT. *IEEE Trans. Biomed. Circuits Syst.* **2016**, *11*, 245–254. [CrossRef]
21. Domínguez-Morales, M.J.; Luna-Perejón, F.; Miró-Amarante, L.; Hernández-Velázquez, M.; Sevillano-Ramos, J.L. Smart Footwear Insole for Recognition of Foot Pronation and Supination Using Neural Networks. *Appl. Sci.* **2019**, *9*, 3970. [CrossRef]
22. Pineda-Gutiérrez, J.; Miró-Amarante, L.; Hernández-Velázquez, M.; Sivianes-Castillo, F.; Domínguez-Morales, M. Designing a Wearable Device for Step Analyzing. In Proceedings of the 2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS), Cordoba, Spain, 5–7 June 2019; pp. 259–262.
23. Anderson, D.; McNeill, G. Artificial neural networks technology. *Kaman Sci. Corp.* **1992**, *258*, 1–83.
24. Luna-Perejón, F.; Domínguez-Morales, M.J.; Civit-Balcells, A. Wearable Fall Detector Using Recurrent Neural Networks. *Sensors* **2019**, *19*, 4885. [CrossRef]
25. Nasser, I.M.; Abu-Naser, S.S. Lung Cancer Detection Using Artificial Neural Network. *Int. J. Eng. Inf. Syst.* **2019**, *3*, 17–23.
26. Sadek, R.M.; Mohammed, S.A.; Abunbehan, A.R.K.; Ghattas, A.K.H.A.; Badawi, M.R.; Mortaja, M.N.; Abu-Nasser, B.S.; Abu-Naser, S.S. Parkinson's Disease Prediction Using Artificial Neural Network. *Int. J. Acad. Health Med. Res.* **2019**, *3*, 1–8.
27. Gutierrez-Galan, D.; Dominguez-Morales, J.P.; Cerezuela-Escudero, E.; Rios-Navarro, A.; Tapiador-Morales, R.; Rivas-Perez, M.; Dominguez-Morales, M.; Jimenez-Fernandez, A.; Linares-Barranco, A. Embedded neural network for real-time animal behavior classification. *Neurocomputing* **2018**, *272*, 17–26. [CrossRef]
28. Sheela, K.G.; Deepa, S.N. Review on methods to fix number of hidden neurons in neural networks. *Math. Probl. Eng.* **2013**, *2013*, 425740. [CrossRef]
29. Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manag.* **2009**, *45*, 427–437. [CrossRef]
30. Finnoff, W.; Hergert, F.; Zimmermann, H.G. Improving model selection by nonconvergent methods. *Neural Netw.* **1993**, *6*, 771–783. [CrossRef]
31. Tudor-Locke, C.; Aguiar, E.J.; Han, H.; Ducharme, S.W.; Schuna, J.M.; Barreira, T.V.; Moore, C.C.; Busa, M.A.; Lim, J.; Sirard, J.R.; et al. Walking cadence (steps/min) and intensity in 21–40 year olds: CADENCE-adults. *Int. J. Behav. Nutr. Phys. Act.* **2019**, *16*, 1–11. [CrossRef]