applied sciences

MDPI

Article

Real-Time Machine Learning for Human Activities Recognition Based on Wrist-Worn Wearable Devices

Alexandru Iulian Alexan [1,*], Anca Roxana Alexan [1] and Stefan Oniga [1,2,*]

[1] North University Center of Baia Mare, Technical University of Cluj-Napoca, 400114 Cluj-Napoca, Romania; anca.osan@yahoo.com
[2] Faculty of Informatics, University of Debrecen, 4032 Debrecen, Hungary
* Correspondence: alexanalexandru@gmail.com (A.I.A.); stefan.oniga@ieec.utcluj.ro (S.O.)

Abstract: Wearable technologies have slowly invaded our lives and can easily help with our day-to-day tasks. One area where wearable devices can shine is in human activity recognition, as they can gather sensor data in a non-intrusive way. We describe a real-time activity recognition system based on a common wearable device: a smartwatch. This is one of the most inconspicuous devices suitable for activity recognition as it is very common and worn for extensive periods of time. We propose a human activity recognition system that is extensible, due to the wide range of sensing devices that can be integrated, and that provides a flexible deployment system. The machine learning component recognizes activity based on plot images generated from raw sensor data. This service is exposed as a Web API that can be deployed locally or directly in the cloud. The proposed system aims to simplify the human activity recognition process by exposing such capabilities via a web API. This web API can be consumed by small-network-enabled wearable devices, even with basic processing capabilities, by leveraging a simple data contract interface and using raw data. The system replaces extensive pre-processing by leveraging high performance image recognition based on plot images generated from raw sensor data. We have managed to obtain an activity recognition rate of 94.89% and to implement a fully functional real-time human activity recognition system.

Keywords: human activity recognition; plot image analysis; ML.NET; real-time; cloud

Citation: Alexan, A.I.; Alexan, A.R.; Oniga, S. Real-Time Machine Learning for Human Activities Recognition Based on Wrist-Worn Wearable Devices. *Appl. Sci.* **2024**, *14*, 329. https://doi.org/10.3390/app14010329

Academic Editors: Affan Yasin, Javed Ali Khan and Lijie Wen

Received: 8 December 2023
Revised: 24 December 2023
Accepted: 26 December 2023
Published: 29 December 2023

1. Introduction

The concept of a smart house or smart living environment is more current than ever, as more and more everyday devices are equipped with smart capabilities. The importance of activity recognition research lies in the advantages of being able to monitor and assist a person who uses smart sensors [1]. Internet connectivity, which is, in most cases omnipresent, has even increased the range of these smart devices as they can now communicate with any internet node and can even be controlled or supervised remotely.

The security aspect of this emerging connectivity is still a very important one, as personal data are sent around via the internet, so any device or platform that is exposed over the internet needs to be properly secured to make sure that the personal data are only available to the correct system and are stored only according to the user's agreement. Human Activity Recognition (HAR) is one area that greatly benefits from this emerging trend of having smart capable devices around the living environment. The amount of available data for human activity recognition greatly increases as we are surrounded by smart capable devices, thus we can detect human activity more precisely. This increase in HAR precision can be especially helpful in environments that are inhabited by multiple persons. In most cases, smart devices can pinpoint the current user, providing additional context when trying to resolve the multiple inhabitant HAR issue, more precisely, to identify a certain person and link them to a specific complex activity in a multi-person space.

Internet of Things (IoT) technology is applied in multiple domains such as medicine, manufacturing, emergency response, logistics, and daily life activity recognition. Further-

more, emerging IoT technologies allow us to not only supervise HAR but also to actively react, based on these and other data, by controlling smart-enabled devices or contacting other persons. This kind of complex supervision and environment control, together with all the smart-enabled devices and systems, make up the building blocks of the smart home, a home that can help its occupants in day-to-day tasks and especially during emergencies or life-threatening situations.

Activity recognition can be performed using three main sensor categories: ambient, vision, and body-worn, as also shown in [2–4]. Ambient sensors are a great way to supervise relatively small areas and require changes across the living environment to function. The main disadvantage of using ambient sensors is the fact that they require a specially prepared living area for HAR analysis. This kind of ambient sensor implementation can help when dealing with multiple inhabitant spaces where we need to identify and pinpoint each user, as the entire living space is supervised at any moment and acts as a single unit. As the supervised living environment increases, the required number of nodes increases as well. A particular type of ambient sensor is vision-based, which can offer great user recognition in multi-inhabitant spaces. This type of ambient sensor requires multiple vision devices installed across the living space (at least one vision-based sensor for every room) and raises some potential privacy issues, especially if the data are analyzed outside the local living environment network.

Body-worn sensors, also known as wearable devices, can directly gather and even process the sensor data closest to the user's body. Since the main data-gathering device is worn on the user's body, this approach will work fully or partially even in non-controlled environments. For this body-worn sensor type, we have only network or minimum infrastructure requirements for the system to work, even if there are multiple types of devices that send data via many types of communication technologies [5]. A person usually performs many types of activities, based on simple and complex movements [6], and this physical information is easily collected through wearable devices like commercial mobile phones and portable devices [7].

The smartphone is one of the most used devices for HAR as it can record and process information itself. Also, since the processing power is enough for most tasks, this widely available device has quickly gained popularity for HAR-related tasks. The major downside of using a smartphone for detecting the user's activity is data downtime for not wearing the device. If the smartphone is not worn or directly used by the user, the system does not receive any relevant data regarding the current activity and, thus, the HAR precision decreases. Smartphones are not necessarily worn consistently, as the wear position will greatly vary depending on the person and situation. A watch has a more stable wear pattern, as it is primarily worn on the user's wrist and is usually worn extensively for long periods of time. A smartwatch is a small device that can be easily and non-intrusively worn for long periods of time, making it ideal for data acquisition [8,9].

For HAR based on accelerometer and gyroscope data, provided by sensors also found in a smartwatch, the classic approach is to use the raw sensor data and preprocess it. Features are then extracted and used to train a neural network for activity recognition. In this scenario, the neural network input is represented by a series of numeric values that try to capture the essence of that particular activity. Based on the raw sensor data or even extracted features, plot images can be generated and fed to the neural network as input data instead of numerical values. In this scenario, the human activity recognition task becomes an image classification task, trying, in essence, to identify the activity based on the plot image using specific image classification neural networks. The numerical data that are turned into a plot image can be graphically represented in multiple ways depending on the type of plot image and the input raw data structure; these variations can have a significant impact on the activity recognition rate [10,11].

Data collected from different devices are stored in various databases that are freely available. These databases are helpful to improve and test new algorithms and methods. One of the most common databases is WISDM [12], as this database contains data

collected from the accelerometer and gyroscope of Android-based smartphones and smartwatches [13]. The data can be affected by the differences in the residence's layout or human behavior. Some of the causes of these differences are health conditions changes, sensor displacement, and activities performed differently over time, as mentioned in [13].

For data segmentation, two major types are used: fixed window size and dynamic segmentation. For both of them, the main challenge is to determine the proper window length. Data segmentation is very important, a smaller segment can allow faster activity recognition and real-time recognition, minimizing the time of the entire required process [14]. On the other hand, choosing a window that is too small may lead to a low activity recognition rate as the system cannot properly identify the activity due to the data window's similarities with other activities. So the length of the data window chosen is very important as we need to allow real-time process recognition and maintain a proper activity recognition rate. The data segmentation process can be time-based or feature-based. In activity recognition for daily tasks, the most common method is time-based segmentation. The main problem for dynamic-size windows is identifying the correct temporal separation for an activity. For the fixed-size windows, the main problem is to identify the proper size to include all the data recorded for the same activity [15–17].

The HAR is very important in real-time systems that include smart gadgets and deep learning techniques [18,19]. To recognize activities from a real environment with continuous data streaming remains a major challenge, as shown in [20]. Recognition can be implemented on wearable devices like mobile phones or personal computers. An important aspect that can affect the performance of the system is the computational cost. Although some minimal computational steps are still required, reducing the multiple features extraction and windowing process is detailed in [21]. The data from the body-worn sensor module can be relayed to a local data hub or to the cloud directly. A socket or HTTP request can be used for data transfer on the local or cloud data processing device.

Our objective is to create a cloud-based real-time human activity recognition system powered by a neural network. The system should be able to work with raw movement data retrieved from a smartwatch in order to identify human activities. The HAR process will be accomplished using an image recognition ML.NET neural network that handles plot images generated based on both raw accelerometer and gyroscope data. The system should be available for local network usage or cloud-based deployment and usage. Data privacy and security are very important aspects and supporting on-premises deployment and usage provides added layers of security for special requirements and use cases.

2. Related Work

Visually representing data from a dataset is quite common. Symbolic representation is used to increase the recognition rate and uses high-performance neural networks specialized in image recognition. A significant method of representing data is Human Activity Recognition on Signal Images (HARSI). As described in [22], this method is very efficient and the research describes the method as converting the raw data into an understandable image. The images were generated by plotting the raw data from an accelerometer. The WISDM data set was used for this work and six types of activities were selected: jogging, moving downstairs, walking, moving upstairs, standing, and sitting. In this study, different types of convolutional neural networks were tested. The experimental algorithms tested the following versions: HARSI-ResNet50, HARSI-ResNet101, HARSI-AlexNet, HARSI-DenseNet121, HARSI-SqueezeNetv1.0, HARSI-SqueezeNetv1.1, HARSI-VGG16, and HARSI-VGG19. The most efficient was the HARSI-VGG19 method with a 98% accuracy.

Another study using the WISDM data set was described in [23]. Two types of algorithms were tested: Convolution Neural Networks (CNNs) and Long Short-Term Memory Networks (LSTMs). Through these algorithms, the performance of recognizing hand movements was tested using smartwatch data. The following activities were studied: kicking, stairs, standing, jogging, sitting, walking, sandwiching, chips, drinking, soup, pasta, fold-

ing, typing, teeth, catching, clapping, dribbling, and writing. The data from 44 subjects were selected from the raw data set. In the preprocessing phase, a time-based sliding window was used with a length of 10 s with 5 s of data overlapping. Three types of data sets were tested: data from the accelerometer, data from the gyroscope, and combined data from both the accelerometer and gyroscope. The best results were obtained from combined data from the accelerometer and the gyroscope with a hybrid CNN-LSTM algorithm with an overall recognition rate of 96.20%. Analyzing the confusion table, we observed that the activities with the lower recognition rate were sitting, standing, sandwiching, and writing.

The research described in [24] presents a human activity recognition method based on symbolic representation (HAR-SR). This method has two essential parts: the first is signal segmentation-based data fusion and the second is symbolic representation based on patterns from time series. The steps in symbolic representation presented in this research start with data segmentation using a sliding window based on time with a specific length, after the dimension of the segments database is reduced by symbolic approximation and noise removal. Next, a search table was used for the discretization process to obtain the symbolic representation of the segment and, in the final step, each discretized segment was used to obtain a frequency histogram of the symbols. For classification, a K-Nearest Neighbor (kNN) algorithm was used and the recognition rate obtained was 77.82%.

An interesting view of human activity recognition was described in [25]. A new architecture of CNNs network was proposed in this research. A new type of classification was added to the convolutional layer. This layer contains a matching filter (MF) based on adding a signal noise comparison function. The activation function from this method is based on calculating the maximum between the known signal or the template signal and the unknown signal or the signal to be known. This function was included in a layer named Conv1D. The Conv1D layer is followed by a GlobalMaxPooling layer and, in the final step, a fully connected layer with softmax activation was added. For the most common activities (walking, jogging, moving downstairs, moving upstairs, sitting), this research obtained a very good total recognition rate of 97.32%. Analyzing the confusion matrix for this method applied to the WISDM data set (only accelerometer data), we observed a lower recognition rate for moving downstairs and moving upstairs activities. Moving downstairs activity obtained an 89.00% recognition rate and moving upstairs a 91.00% recognition rate. The algorithm was unable to obtain a very good recognition rate for both of these activities, probably due to the similarities between the signals.

The labeling of the activity data set is the most important part of improving activity recognition performance according to [26]. This research proposes a system based on user feedback regarding the recognized activity. The user was informed through a notification. The recognition algorithm used is a long short-term memory (LSTM) model which was trained using an open dataset. The accelerometer data that are read from the smartphone are recognized by the trained algorithm and the result is sent to the user via a notification. The user approves the newly labeled data and, in this way, the raw data set increases in size. This implementation improves the recognition rate, according to this research, with 10% more than the classical method with the same model. For the kNN model, the recognition rate was improved by 13.40%, and for LSTM, by 16.20%.

The concept of the Internet of Things can help improve people's life nowadays, by handling and maximizing the data collected from multiple sensors. The authors of [27] propose a human activity recognition real-time system based on data collected from smartphones and smartwatches. The real-time data are collected from the accelerometer and gyroscope sensors. The watch is placed on the wrist and the smartphone is in a pocket. For the accelerometer, the signals are separated into body and gravity components. The data are segmented into overlapped windows. After the new features are extracted, the data scaling factors and normalization are applied. Four types of algorithms were tested for human activity recognition: Random Forest (RF), Multi-layer Perception (MLP), Support Vector Machines (SVM), and Naive Bayes (NB). The training was implemented with a dataset collected by researchers. The best recognition rate was obtained by the RF model at 98.72%.

Also, this method was tested with the WISDM dataset, and a 98.56% recognition rate was obtained.

Using wearable sensors built into everyday objects is common practice, even reaching the level of the incorporation of stretch sensors into clothes. Combining stretch sensors and inertial sensors can yield good results with an accuracy rate of 97.51% [28]. This approach implies that the user wears custom add-on modules that can also be built into the clothes, and it requires specialized hardware modules for this, making the system a bit more difficult to wear for extensive periods of time.

Options to unobtrusively monitor a user's activity can also be used, without the subject carrying a dedicated device; for example, the authors of reference [29] highlight an innovative system that uses indoor WiFi signals to detect users' activity by the different patterns generated. This system can achieve a good recognition rate; the CNN-ABiLSTM model reached an accuracy of 98.54% but it has some drawbacks as it can be used only in a specific setup environment.

3. Proposed System

We propose a HAR system able to perform real-time activity recognition based on internally generated plot images. The proposed system has multiple components:

- A data preprocessing app;
- A machine learning core processor;
- A machine learning processor Web API;
- A real-time Cloud human activity recognition system

The main components of the proposed system are shown in Figure 1.

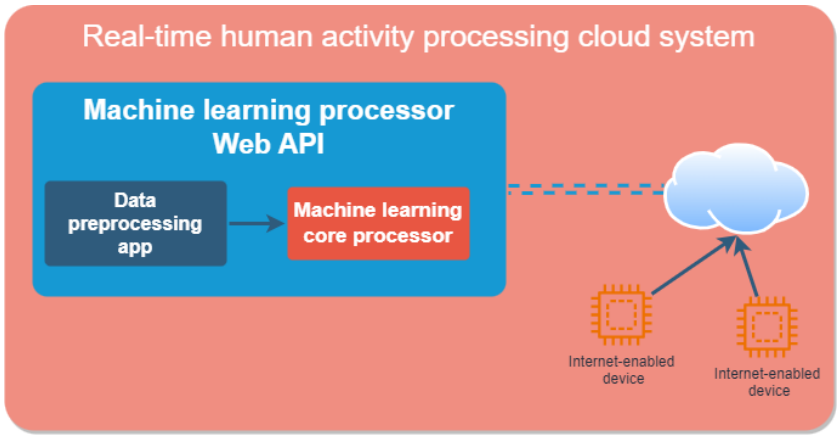

Figure 1. Main components of the proposed system.

The "Data preprocessing app" handles the conversion of raw movement data from an accelerometer and gyroscope to a plot image. This conversion transforms a movement data window to a single image that will be used as input for the machine learning algorithm. The "Machine learning core processor" represents the main computing logic that is a machine learning implementation trained using the previously generated movement images. After training, this component is capable of receiving an image and predicting its source activity. This behavior is supported only locally, as this component does not support network interactions or advanced conversions from raw data to images. The "Machine learning processor Web Api" is the component that incorporates the "Machine learning core processor" and is able to support network connections and recognize human activity. The recognition process can be based on a generated image but this component is also able to generate the image itself based on raw accelerometer and gyroscope data. So, in order to recognize what activity a series of movement data is part of, a simple API call is sufficient. The "Real-time Cloud human activity recognition system" represents the "Machine learning processor Web Api" cloud correspondent, which is able to handle requests from

multiple computer networks via the Internet. This component is not limited to a single local network and can be easily scaled and enhanced for the human activity recognition process. In this way, we can easily recognize real-time human activity from any source application that can make a web API request that contains movement data.

3.1. Our Contributions

Our contributions to the HAR field are as follows:

- The implementation of a real-time system for human activity recognition that can operate locally and in the cloud via rest API calls based on image plot recognition;
  - The implementation and usage of a .NET C# console application to generate label images based on raw accelerometer and gyroscope sensor data;
  - The creation of a .NET C# application that contains a deep neural network that was created based on a pre-trained TensorFlow DNN model and trained for HAR using plot images;
  - The integration of the created and trained neural network in a .NET Web API application capable of real-time activity recognition based on rest API calls;
  - The further extension of the HAR Web API application capabilities to allow cloud-based activity recognition.
- The analysis of multiple scenarios for plot image generation configuration and plot types and the evaluation of the obtained activity recognition precision results;
- We concluded that a real-time HAR system, based on plot image recognition and REST requests, can be a good system architecture for real-time activity recognition.

3.2. WISDM Biometrics Dataset

The "WISDM Smartphone and Smartwatch Activity and Biometrics Dataset" dataset was chosen for this implementation. This dataset was introduced and described in [30] and is extensively used for human activity recognition.

This dataset was chosen as it is one of the most important and used datasets for human activity recognition based on wearable devices, as also mentioned in [22]. Since it was used in multiple studies, it is a perfect candidate to be used to compare different approaches and to analyze the obtained accuracy rates across multiple different implementations. Since the data collection process was closely supervised by the dataset research creators, to ensure proper data quality and movement consistency, this dataset can be clearly used for the in-depth analysis of any of the available activities. This dataset is even suitable for detecting symmetric activities. Based on the fact that the dataset used is an existing one that is freely available for usage, the research creators handled the appropriate approvals from required entities as it involved human subjects research.

The dataset contains many different activities carried out by 51 subjects. We have 18 different activities, each with a length of about 3 min. The monitoring devices used to track the subject's movements are a smartwatch and a smartphone. The smartwatch is placed on the subject's dominant hand and is non-obstructive, due to its relatively small form factor. The smartphone is placed in the pocket, simulating the everyday carrying of this omnipresent device. Both data-gathering devices run a custom-made app designed specifically for this purpose. Both tracking devices gather accelerometer and gyroscope data using a number of four separate sensors, two for each device. Each sensor gathers data with a frequency of 20 Hz, reading a measurement every 50 ms. This is the data polling rate that was set for the operation but the actual polling rate of the sensor data can be delayed if the processor is busy.

Accelerometer and gyroscope data folders are present for both smartwatch and smartphone data. In each data sub-directory, there will be a file for each subject, so a total of 51 files will be present in each sub-directory. The naming of the files is standard and uses underlining to delimit different parts of the name components.

Each data file contains one sensor reading per line using a comma-separated value data format, as the data format is maintained across the different sensor types and sources. Each line has the following features:

- Subject-id: a unique identifier for the subject, with a range between 1600 and 1650;
- Activity Code: a unique identifier for the activity, using a single letter from A to S skipping the "N" value;
- Timestamp: the sensor reading timestamp in Linux format;
- Sensor value for the x, y, and z axis represented by numeric real values.

Each file contains approximately 54 min of readings, based on the expected 3 min for each activity multiplied by the total number of 18 activities; in an ideal case, we would have a total number of 64,800 lines for the file. The data collection process was not perfect, so these borderline values will still fluctuate across the dataset somewhat.

The raw total number of samples in this dataset is 15,630,426 and the distribution of data across the sensor types and sources is as follows:

- Raw phone accelerometer number of data readings: 4,804,403;
- Raw phone gyroscope number of data readings: 3,608,635;
- Raw watch accelerometer number of data readings: 3,777,046;
- Raw watch gyroscope number of data readings: 3,440,342.

The smartwatch data-gathering device runs Android Wear 1.5 and the smartphone runs Android 6.0.

As shown in [31], this dataset groups all the previously mentioned activities into three main activity categories based on the type of activity and each recorded activity in this dataset has a code assigned. These three major activity categories, alongside the corresponding activities, are shown in Table 1.

Table 1. Dataset activity categories.

Activity Category No.	Activity Category	Activities
1	Non-hand-oriented activities	walking, jogging, stairs, standing, kicking
2	General hand-oriented activities	dribbling, playing catch, typing, writing, clapping, brushing teeth, folding clothes
3	Eating hand-oriented activities	eating pasta, eating soup, eating a sandwich, eating chips, drinking

For this research, we plan to use only the smartwatch dataset containing accelerometer and gyroscope raw data and we do not plan to use any of the already available extracted features.

3.3. Data Preprocessing App: Movement Data Plot Images Generation

In order to simplify the classic time-series data classification task, based on the sensor data, we can handle the human activity recognition task as an image classification one. The numeric sensor data from the smartwatch are used to transform a series of values, consisting of a data window, into a single data image. This way we can provide a visual representation of a data chunk that is easier for a human to analyze and interpret the data manually. Each image can show certain characteristics for that particular activity type and different plot styles can be used. The main preprocessing application is written in C# 6 and uses the "ScottPlot" plotting library for .NET. The preprocessing application is written as a .NET console application.

In order to reduce the amount of data generated in the preprocessing phase, only certain activity types and users are selected for processing. The following activities are selected for processing from the source dataset: walking, jogging, stairs, sitting, standing,

typing, and brushing teeth. To further decrease the number of generated plot images, the users that performed the activities are also filtered and only the first five users are used.

The "ScottPlot" 4.1.59 plotting library was chosen as it is open-source, free, and provided under the permissive MIT license. This library makes displaying and saving images easy and fast. As mentioned in [32], there are multiple platforms of .NET that are supported by this library (including .NET core) which means that it can even be used in Azure cloud directly, under the form of Azure function.

Based on the chosen plot image type from the ScottPlot library, we have two main scenarios:

- Scatter plot images;
- Population images.

The generated plot images have a different width and height based on the chosen scenario, as shown in Table 2. The window size chosen for overlapping scenarios is 40 records.

Table 2. Generated image output dimensions.

Scenario	Image Width	Image Height
Scatter plot images with overlapping scatter plots	600	400
Scatter plot images with non-overlapping scatter plots	500	300
Population images	500	300

An example of generated scatter plot images with overlapping scatter plots with a data sequence for walking activity is shown in Figure 2.

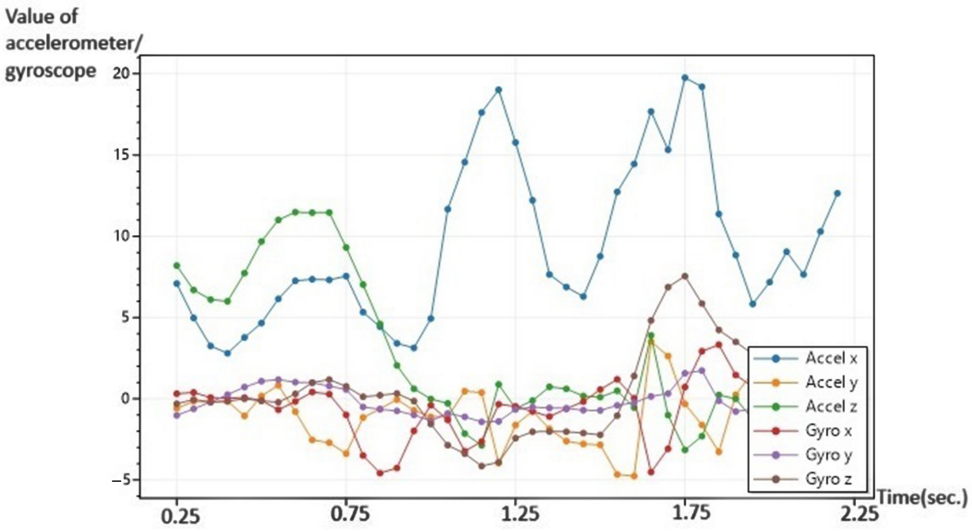

Figure 2. The scatter plot image for a data sequence with overlapping windows for an activity using the .NET plotting library.

An example of generated scatter plot images with non-overlapping scatter plots with a data sequence for the walking activity is shown in Figure 3.

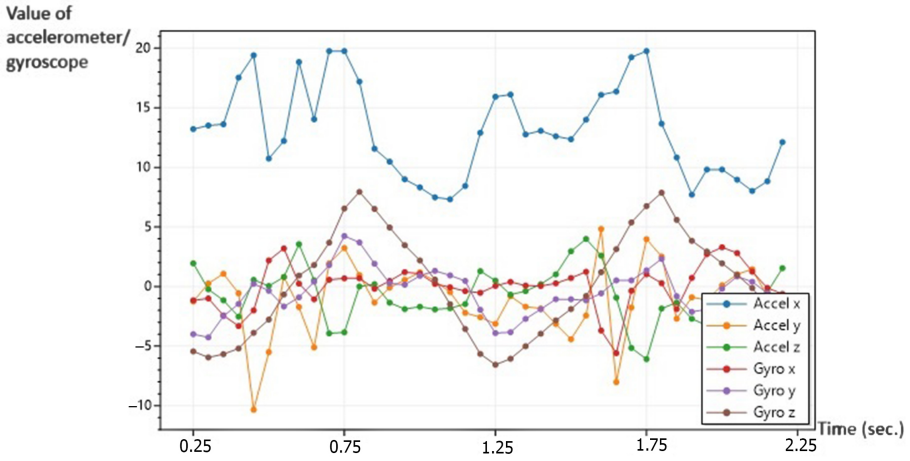

Figure 3. The scatter plot image for a data sequence without overlapping windows for an activity using the .NET plotting library.

An example of generated population images with the "BarMeanStDev" option is shown in Figure 4.

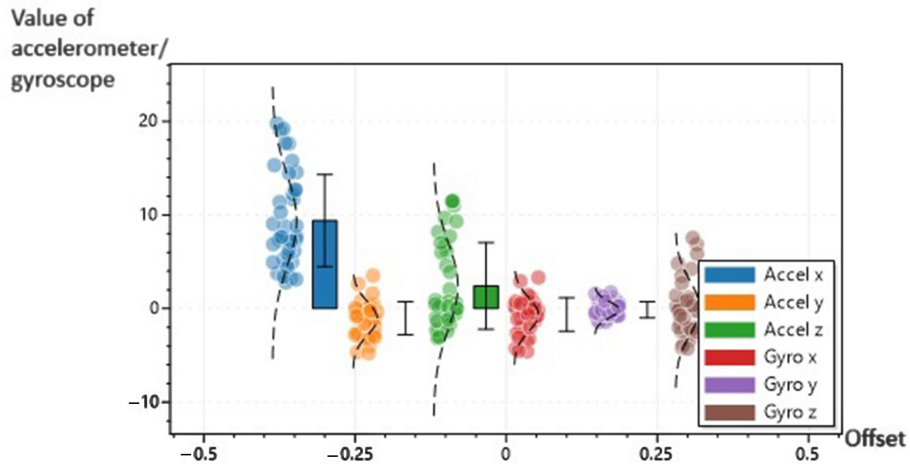

Figure 4. Generated population image with the "MeanAndStdev" option.

3.4. Machine Learning Core Processor

The machine learning core processor is the main component that is able to perform human activity recognition. This component can train a neural network based on the generated plot images and allows this trained neural network to be used directly from a .NET core application. The hosting application where the training takes place is the same one as where the trained neural network is placed afterward; it is implemented in the form of a .NET core console application project. This allows the neural network to be created, trained, saved, and tested, all in one place. The neural network can be later moved into another project to allow further development. This machine learning core processor project thus also contains the required logic for the model consumption and the logic required for the model to be retrained.

3.4.1. Machine Learning Training Process

The training phase was executed locally using the local GPU as the main training processing device. The used training environment machine was equipped with an Intel(R) Core(TM) i5-8300H CPU running at 2.30GHz. The available system memory was 16.0 GB. The GPU used for the main training process was a NVIDIA GeForce GTX 1050.

In order for the training process to support local GPU, the graphics card needs to be CUDA compatible and the system needs to have multiple software components, like

extensions and a toolkit, installed for this particular purpose. The "ML.NET Model Builder GPU Support 2022" GPU Visual Studio extension needs to be installed in the system so that we can use the local GPU for training image classification models. The ML.NET Model Builder 2022 extension was updated to the latest version at implementation time, 16.14.2255902, as this extension provides a simple and fast UI tool to build, train, and ship custom machine learning models in .NET applications. The CUDA Toolkit version 10.1 also needs to be installed as this is the recommended version at implementation time and it provides a development environment for creating and running high-performance GPU-accelerated applications. Alongside the CUDA Toolkit, the CUDA Deep Neural Network library (known as cuDNN) also needs to be installed and, in this case, version 7.6.4 was used. cuDNN is a library that contains GPU-accelerated primitives for deep neural networks. The cuDNN library contains implementations for standard routines: forward and backward convolution, pooling, normalization, and activation layers.

3.4.2. Training Time

The training time is different across different runs and ranges from 1.12 to 5.8 h depending on the size and number of the images used for training. All training processes were executed on the same machine described in the previous section. The training times obtained for the executed scenarios are presented in Table 3.

Table 3. Training time.

Scenario	Number of Images	Training Time (Seconds)
Scatter plot images with overlapping scatter plots	125,973	20,975.79
Scatter plot images with non-overlapping scatter plots	125,973	5750.11
Population images with the "BarMeanStDev" option	111,418	4437.96
Population images with the "MeanAndStdev" option	111,478	4050.35

3.4.3. Local Activity Recognition

After the training phase has been completed, the machine learning core processor can be used to run the activity recognition process locally. From the project's console application, any logic can be added to leverage the activity recognition functionality. The image data can be generated on the fly based on raw accelerometer data or the system can use a database as a buffer for the image files or movement data. The core processor functionality can be incorporated into any .NET project type, like a desktop application or even a web application.

3.4.4. The Trained Deep Neural Network (DNN)

The deep neural network model chosen for the image classification task is "ImageClassificationMulti". The available trainer for this image classification task is "ImageClassificationTrainer" and it trains a DNN network by using pre-trained models for classifying images, in this case, Resnet50. For this, trainer normalization and the cache are not required. A supervised ML task predicts the category or class of the image representing the activity type that we want to recognize. Each label starts as text and is converted into a numeric key via the "TermTransform". The image classification algorithm output is a classifier that can recognize the class and the activity type for a provided image. Figure 5 shows the generated ML model diagram, as can be seen using the Netron viewer software.

Since ML.NET also uses a pre-trained deep neural network (DNN), called transfer learning, the trained model also includes pre-trained knowledge, making it perfect for improving accuracy and decreasing training time. ML.NET internally retrains the Tensorflow

layers based on the used data images input dataset. A small portion of the entire Resnet50 model used is shown in Figure 6.

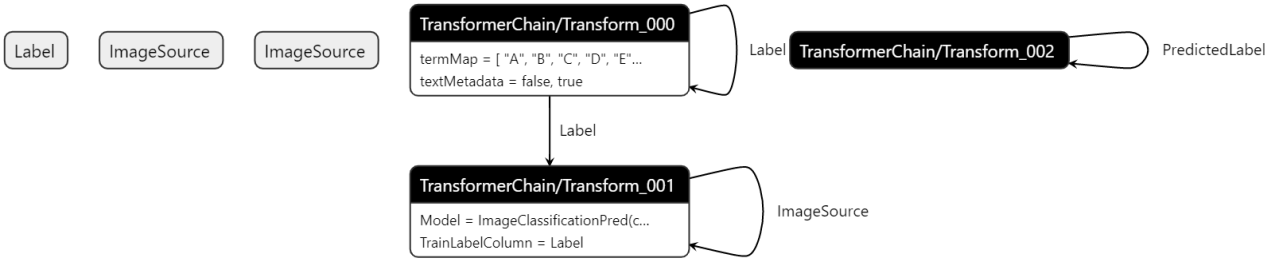

Figure 5. Neural network ML model diagram.

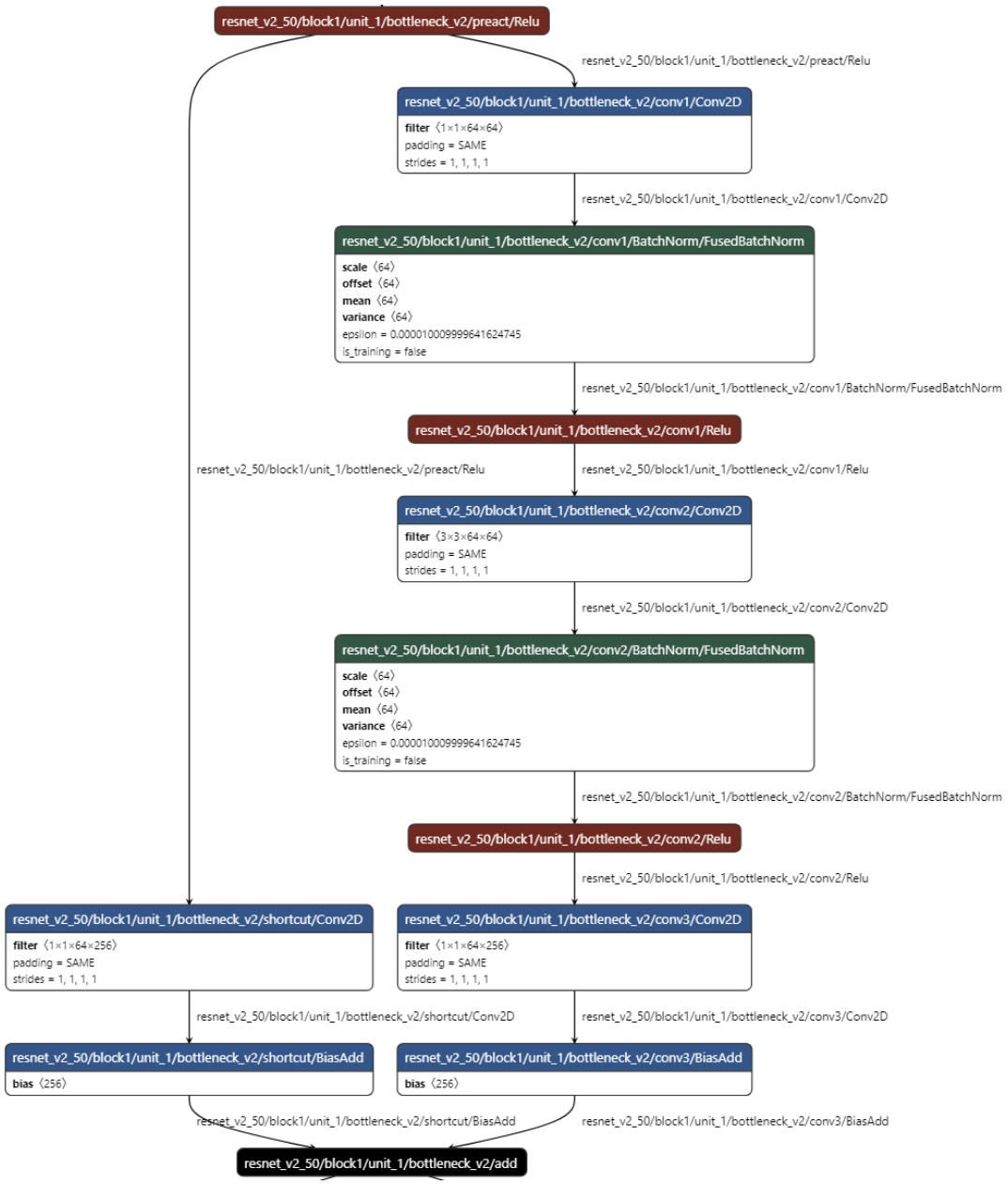

Figure 6. Resnet50 neural network detail.

3.5. Web API Activity Recognition System

Based on the machine learning core processor module, that is able to recognize human activity relying on the movement-generated plot image, a Web API application was built to expose this functionality to other components inside the local network. In this way, any device can receive the activity type as a response by making an API request containing either an already generated plot image or the raw data required to generate the plot image. Since we are handling all the main processing in a Web API application, we can save the received data, resulting in a database, and even send real-time system notifications to other linked subsystems or components based on certain events. For example, email notifications can be sent if the system encounters an activity that is out of the ordinary based on certain logic. A Web API application type component is useful to simplify the system architecture as it is scalable and allows the other subsystems to easily communicate using a fast and reliable method using proven protocols and technologies. Since we are using a stateless design, the lower components that make the data acquisition do not need to be very powerful from the computing perspective. The only requirement is to be able to generate HTTP requests, compared to a real-time system designed around sockets, where communication is achieved via a bidirectional opened channel. The lower layer of the acquisition device can gather data, based on window size, and when the data has reached the window size, an API request can be created with the entire window payload. The frequency of the API requests is clearly dependent on the chosen window size and whether we want to overlap data windows or not.

3.5.1. The Web API Project

The web API project was build in-house and is based on the .NET framework and structured in the form of a minimal API project in .NET core 7. Minimal API was chosen as it is perfect for this kind of implementation due to its low file count and clean architecture. Due to the .NET Core cross-platform nature, this project can be deployed on multiple platforms, like Windows and Linux, and supports cloud integration as well. The minimal API file structure features a small number of configuration files and one single code entry point. The already trained network is loaded from the generated machine learning model zip archive using the "FromFile" extension when registering the prediction engine pool.

3.5.2. The Web API Endpoints and OpenAPI Specification

The Web API project features two main endpoints used for activity recognition, one that is able to detect human activity based on a movement data plot image and the second that is able to detect human activity based on a window of movement data gathered from an accelerometer and gyroscope. The OpenAPI specification represents the standard for defining RESTful interfaces, providing a technology-agnostic API interface that provides API development and consumption support. Swagger is the tool that allows for OpenAPI specification generation and usage in our web API project. Swagger contains powerful tools to fully use the OpenAPI Specification, as shown in Figure 7, where the Swagger-generated documentation for the created Web API is highlighted.

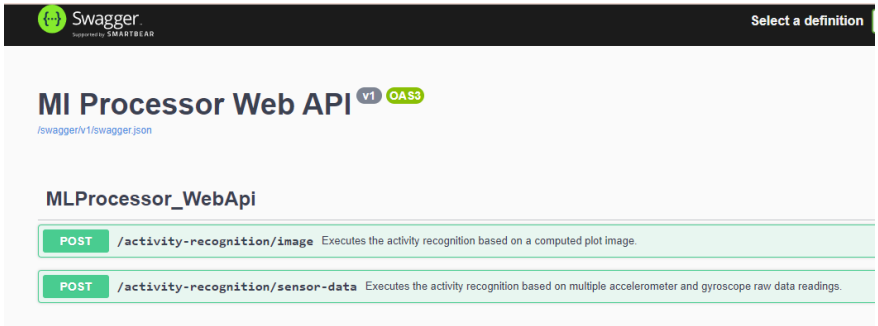

Figure 7. Swagger OpenAPI specification.

3.5.3. Real-Time Activity Recognition Process

Since we are exposing the activity recognition process via a rest API, any subsystem in the local network can call this web API to benefit from the activity recognition process. The rest request type is post and the configured endpoint route is "/activity-recognition/image". The activity recognition sensor data endpoint can detect the activity based on a window of raw movement sensor data. Internally, the movement plot image containing sensor data from both the accelerometer and gyroscope is recreated and fed to the machine learning neural network. The rest request type is post and the configured endpoint route is "/activity-recognition/sensor-data".

3.6. Real-Time Cloud Human Activity Recognition System

In order to create a real-time system, Azure cloud functionalities were leveraged to create a scalable and omnipresent service. Since the activity recognition system is deployed in the cloud, even a simple internet-enabled device with low processing power can benefit from the activity recognition functionalities, not just the ones from the local network. In order to deploy the Web API application in the cloud, a valid Azure subscription is required. The created components are structured under a hierarchical form, having as a parent the HarResourceGroup resource group that has an API management service named MLprocessorWebApi that can contain multiple APIs, an,d in this case, contain only one, named MlProcessorWebApi.

The main web API application was deployed in the cloud using the following address [33] and the postman example request is shown in Figure 8, a request that is able to determine the activity type with the activity unique identifier "E" that corresponds to the standing activity.

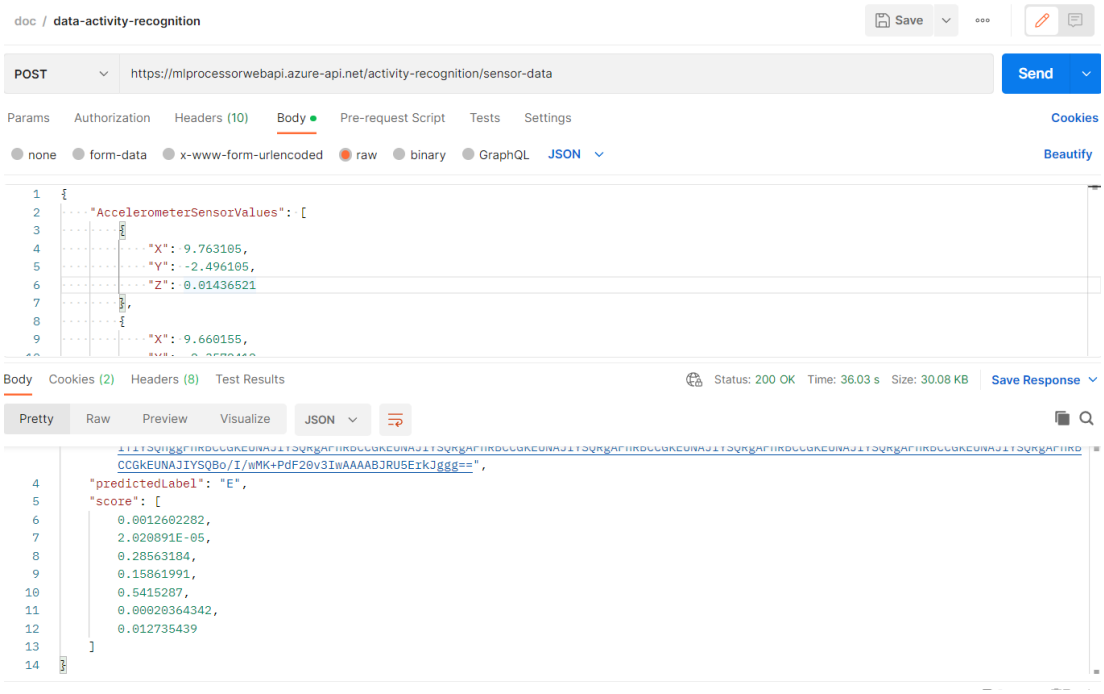

Figure 8. Azure postman post call to the activity recognition data endpoint.

The deployed Web API is secured using subscription keys. In order to consume the published APIs, it is mandatory that this subscription key be included in the HTTP requests when calling the protected APIs. If a valid subscription key is not provided in the header, the calls are not forwarded to the back-end services or rejected immediately by the API Management gateway.

4. Materials and Methods

The "WISDM Smartphone and Smartwatch Activity and Biometrics Dataset" dataset is freely available and can be downloaded from [12]. The "ScottPlot" library can be found here [34], alongside examples and demos. The corresponding GitHub page for this project is located here [32] and the Nuget package is listed here [35]. The CUDA Toolkit used for GPU neural network training is available here [36]. NVIDIA CUDA® Deep Neural Network library (cuDNN) is available here [37]. ML.NET Model Builder GPU Support 2022 is available here [38]. ML.NET Model Builder 2022 is available here [39]. The Netron viewer used is freely available here [40].

The source code for the proposed real-time cloud-based human activity recognition system that uses image classification is available in a public git repository [41]. This also contains the Web API project that was deployed and tested in the cloud. It is located in the "realTimeHarSystem" folder and includes the main Web API "MLProcessor WebApi" project and its project dependencies, like the "MlProcessorService" class library project.

The dataset processor console application that was used to generate the plot images used for training is also available in this repository. Each scenario folder contains the "DataSetProcessor" project with the required updates for that particular scenario. The input data used by the "DataSetProcessor" project is located in the "watch" folder located in the root directory and contains the raw data from the dataset smartwatch with subfolders for accelerometer and gyroscope data named "accel" and "gyro". The output-generated images are not part of this repository as their size is very large and can be easily generated with the tools and files provided in this public repository.

5. Results

We used the WISDM dataset to train a real-time human activity recognition system that is based on a Resnet50 neural network that achieved the best precision of 94.89% using scatter plot images with overlapping scatter plots. Raw accelerometer and gyroscope data from the previously mentioned WISDM dataset are both used to generate the plot images that consist of the input data for the neural network training process.

For the obtained results, the following activities were used: Walking, Jogging, Stairs, Sitting, Standing, Typing, and Brushing Teeth, for five selected users. Four major factors affect the size and number of generated plot image files:

- The number of analyzed activities;
- The number of analyzed users;
- The window type or window size;
- The plot image dimensions.

The usage of a reduced dataset is due to the large size and number of the generated image data sets, as we have reduced the available total number of 18 activities to 7 activities and the total number of 30 users to a smaller one of 5. Reducing the number of analyzed users and activities provided a decent working dataset and the decrease of the generated plot image dimensions further decreased the side of the generated plot image dataset.

In Table 4, we can observe the performance of the obtained results. We can clearly notice that Scatter plot images with the overlapping scatter plots method obtained the best result. Another method that obtained decent results is using population images with the "BarMeanStDev" option. As shown in Table 4, we managed to obtain a decent precision with a maximum value of 94.89% when using scatter plot images with overlapping scatter plots.

The algorithm managed to obtain a surprising result time-wise, as Figure 9 clearly shows. Although the method of using mean and standard-deviation-based generated images is not the best from an activity recognition standpoint, it managed to obtain a 4.7 times better training time, meaning it is the fasted trained method. Since the activity recognition difference is 0.2%, this is a negligible value as opposed to the obtained training time gain.

Table 4. Obtained results.

Scenario	Recognition Algorithm	Accuracy
Scatter plot images with overlapping scatter plots	ImageClassificationMulti	94.89%
Scatter plot images with non-overlapping scatter plots	ImageClassificationMulti	93.83%
Population images with the "BarMeanStDev" option	ImageClassificationMulti	94.69%
Population images with the "MeanAndStdev" option	ImageClassificationMulti	92.49%

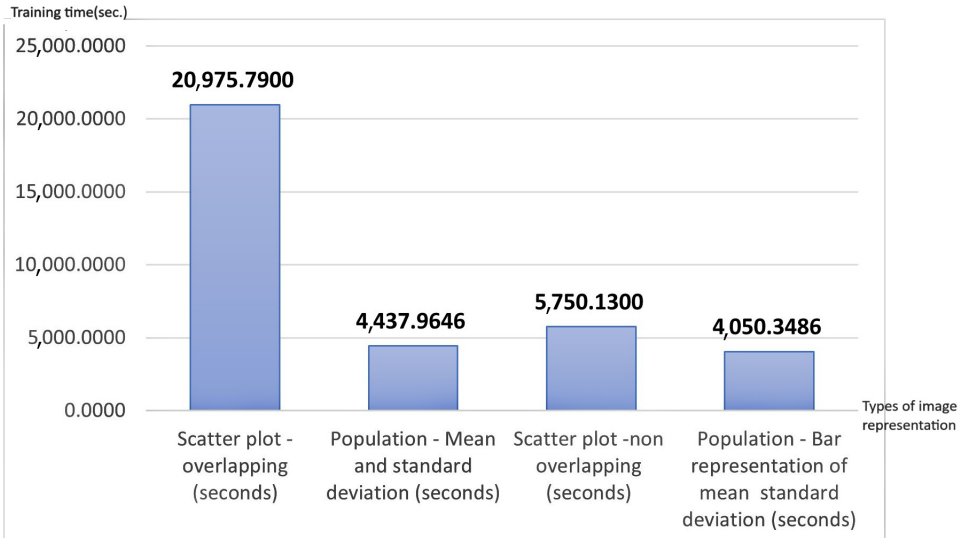

Figure 9. Training time performance analysis.

The following sections will detail the obtained results for each of the four analyzed scenarios.

5.1. Scatter-Plot-Images-Based Recognition

For the "Scatter Plot" type scenario, two main variants were tested, overlapping and non-overlapping scatter plots. For overlapping scatter plots, we obtained a micro accuracy of 0.9489 using a training session with a duration of 20,975.7900 s.

For non-overlapping scatter plots, we obtained a micro accuracy of 0.9383 using a training session with a duration of 5750.1130 s. The obtained non-overlapping scatter plot's micro precision is smaller in this case with 0.0106 compared with using overlapping scatter plots. The training time is significantly lower, with a difference of 15,225.677 s compared to scatter plots.

We obtained better human activity recognition precision using overlapping scatter plots compared with non-overlapping scatter plots but the training time was significantly higher.

5.2. Population-Based Images Recognition

For the "Population" type scenario, only overlapping window type was used, based on the previously tested scenarios. The two main variants that were tested differ in how the population plot was generated using either the BarMeanStDev or the MeanAndStdevResult generation option.

For the "BarMeanStDev" population plot generation option, we obtained a micro accuracy of 0.9469 using a training session with a duration of 4437.9640 s.

For the "MeanAndStdevResult" population plot generation option, we managed to obtain a micro accuracy of 0.9249 using a training session with a duration of 4050.3480 s.

The obtained "MeanAndStdevResult" population plot generation option micro precision is smaller in this case with 0.022 compared with using the "BarMeanStDev" population plot generation option. The training time is somewhat lower, with a difference of 387.616 s compared with the "BarMeanStDev" population plot generation option.

We managed to obtain a slightly better precision for scatter plot images with overlapping scatter plots with a micro accuracy of 0.9489 compared with the population images with the "BarMeanStDev" option where we obtained a micro accuracy of 0.9469, with a very small difference for micro accuracy of 0.002.

5.3. Comparison of Results with Other Studies

As shown in Table 5, the results obtained leveraging the researched methods obtained a satisfactory result. It can be seen that studies that used data representation as images obtained results as good as those represented in raw data windows.

Table 5. Comparison results with other research based on WISDM dataset.

Criteria Number from Table 6—Related Work Section	Algorithm	Accuracy %	Type of Data Representation
Our results	RestNet50-ImageClassificationMulti model	94.89	Image representation; Population-based MeanAndStdevResult image plotting with overlapping
Our results	RestNet50-ImageClassificationMulti model	92.49	Image representation; Population-based MeanAndStdevResult image plotting
Our results	RestNet50-ImageClassificationMulti model	94.69	Image representation; Population-based BarMeanStDev image plotting
1	HARSI-ResNet34	97.33	Image representation
2	HARSI-AlexNet	89.17	Image representation
4	CNN	89.60	Raw Data Windowing
6	Hybrid CNN-LSTM	95.20	Raw Data Windowing
7	CNN	86.40	Raw Data Windowing
9	Hybrid CNN-LSTM	95.40	Raw Data Windowing
10	CNN	93.10	Raw Data Windowing
12	Hybrid CNN-LSTM	96.20	Raw Data Windowing
14	Matching Filter CNN	97.32	Matching filter over Raw Data

Table 6. Summary of activity recognition systems based on WISDM dataset.

Criteria Number	Reference	Year	Algorithm	Accuracy %	Recognised Activities	Data Source Sensors	Type of Data Representation
1	[22]	2022	HARSI-ResNet34	97.33	Six activities [1]	Accelerometer	Image representation
2	[22]	2022	HARSI-AlexNet	89.17	Six activities [1]	Accelerometer	Image representation
3	[22]	2022	HARSI-VGG19	98.00	Six activities [1]	Accelerometer	Image representation
4	[23]	2020	CNN	89.60	Eighteen activities [2]	Accelerometer	Raw Data Windowing
5	[23]	2020	LSTM	87.80	Eighteen activities [2]	Accelerometer	Raw Data Windowing
6	[23]	2020	Hybrid-CNN-LSTM	95.20	Eighteen activities [2]	Accelerometer	Raw Data Windowing
7	[23]	2020	CNN	86.40	Eighteen activities [2]	Gyroscope	Raw Data Windowing
8	[23]	2020	LSTM	84.10	Eighteen activities [2]	Gyroscope	Raw Data Windowing
9	[23]	2020	Hybrid-CNN-LSTM	95.40	Eighteen activities [2]	Gyroscope	Raw Data Windowing
10	[23]	2020	CNN	93.10	Eighteen activities [2]	Accelerometer, Gyroscope	Raw Data Windowing

Table 6. *Cont.*

Criteria Number	Reference	Year	Algorithm	Accuracy %	Recognised Activities	Data Source Sensors	Type of Data Representation
11	[23]	2020	LSTM	89.60	Eighteen activities [2]	Accelerometer, Gyroscope	Raw Data Windowing
12	[23]	2020	Hybrid-CNN-LSTM	96.20	Eighteen activities [2]	Accelerometer, Gyroscope	Raw Data Windowing
13	[24]	2020	kNN HAR-SR	77.82	Six activities [3]	Accelerometer	Symbolic representation
14	[25]	2022	Matching Filter CNN	97.32	Five activities [4]	Accelerometer	Matching filter over Raw Data
15	[26]	2019	Labeled CART	77.30	Six activities [3]	Accelerometer	Labels over Raw Data
16	[26]	2019	Labeled kNN	80.01	Six activities [3]	Accelerometer	Labels over Raw Data
17	[26]	2019	Labeled LDA	76.60	Six activities [3]	Accelerometer	Labels over Raw Data
18	[26]	2019	Labeled LR	77.80	Six activities [3]	Accelerometer	Labels over Raw Data
19	[26]	2019	Labeled LSTM	78.30	Six activities [3]	Accelerometer	Labels over Raw Data
20	[26]	2019	Labeled RF	81.50	Six activities [3]	Accelerometer	Labels over Raw Data
21	[26]	2019	Labeled SVM	77.50	Six activities [3]	Accelerometer	Labels over Raw Data
22	[27]	2022	RandomForest	98.56	Thirteen activities [5]	Accelerometer	New features over Raw Data

[1] Jogging, moving downstairs, walking, moving upstairs, standing, sitting. [2] Kicking, stairs, standing, jogging, sitting, walking, sandwiching, chips, drinking, soup, pasta, folding, typing, teeth, catching, clapping, dribbling, and writing. [3] Walking, running, moving downstairs, moving upstairs, sitting, standing. [4] Walking, jogging, moving downstairs, moving upstairs, sitting. [5] Biking, having coffee, walking downstairs, eating, jogging, sitting, smoking, standing, giving a talk, typing, walking upstairs, walking, writing.

It can be observed that we obtained a better result using the proposed model, with an approximately 5% recognition rate gain compared to the implementation using a hybrid HARSI-AlexNet model (Criteria number 2).

An improved result, compared with the proposed model, was obtained using a model based on RestNet, under the form of a HARSI-ResNet34 hybrid model. This showed that some hybrid models obtained better results than the classic ones, which will help us in the development of a new algorithm model. Comparing the basic model of the hybrid variants, it can be seen that those based on RestNet obtained better results than those based on AlexNet or LSTM. The fact that the HARSI-ResNet34 variant model [22] used a data plot obtained only from the accelerometer raw data confirms that the model proposed by us, which uses the data from the accelerometer and gyroscope, could be enhanced in a future implementation using a new hybrid model.

6. Conclusions

Our main objective was to implement a real-time cloud-based human activity recognition system that uses image classification at its core. The system should be able to expose HAR capabilities from the cloud or locally via rest API calls. In order to achieve this, a .NET core C#-based web API was implemented to expose the activity recognition functionality. The main HAR functionality was achieved using a deep neural network created based on a pre-trained TensorFlow Resnet50 DNN model and trained for HAR using plot images. The created system was trained using data from the WISDM dataset, which is open access. The training data were generated using a separate .NET core C# application that generated the plot images based on raw accelerometer and gyroscope data. Since the proposed system can also be deployed to the cloud, it can be easily expanded to support multiple sensor modules and users at the same time based on its rest implementation.

We managed to obtain a decent human activity recognition rate of 94.89%, using the following activities subset: Walking, Jogging, Stairs, Sitting, Standing, Typing, and Brushing Teeth, demonstrating that a real-time HAR system based on plot image recognition and REST requests can be a good system architecture for a real-time activity recognition system. We managed to improve the previously obtained human activity recognition accuracy of 93.10%, as shown in [31], to 94.89%, obtaining an increase of 1.79%.

Since only raw accelerometer and gyroscope data are used by the proposed HAR platform, we should be able to expand the usable sensor modules to any sensor module that has an accelerometer, gyroscope, and network capabilities. This creates a kind of hardware abstraction layer, as this can be simply implemented with a relatively low number of physical components. This is a very simple option to allow basic network-enabled sensor modules to gain ML.NET deep neural network capabilities. The custom hardware implementation of wrist-worn sensor modules is also a viable option to be integrated into the proposed system, as we are using a web API for the final system integration.

In order to further improve the accuracy of the implemented system, additional plot-type images can be analyzed to see if we can gain a performance boost. Other types of neural networks can also be analyzed, and other custom TensorFlow models may even provide a better implementation to further expand the system. The system could be expanded to use more data from the initial dataset, by increasing the number of analyzed activities and users and trying to optimize the training time by using a more powerful training machine, and trying to lower the training time using other pre-trained deep neural networks.

In order to further increase the system's accuracy, integration with other HAR systems may also yield better results, like integration with an environmental-based HAR system. This will increase the complexity of the system but would provide additional types of data regarding the user's activity.

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

Funding: This research received no external funding.

Data Availability Statement: Data available in a publicly accessible repository. The data presented in this study are the "WISDM Smartphone and Smartwatch Activity and Biometrics Dataset". The dataset is openly available and can be downloaded from [12].

Acknowledgments: The Intelligent Embedded Systems research laboratories supported this work at the Technical University of Cluj-Napoca, North University Center of Baia Mare.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

HAR	Human Activity Recognition
ARFF	Attribute-Relation File Format
cuDNN	CUDA® Deep Neural Network library
CUDA	Compute Unified Device Architecture
DTO	Data Transfer Object
DNN	Deep Neural Network
ML	Machine Learning
REST	Representational State Transfer
IoT	Internet Of Things
API	Application Programming Interface
Resnet50	A residual convolutional neural network with 50 layers
WISDM dataset	Wireless Sensor Data Mining dataset
LSTM	Long Short-Term Memory Networks
CNN	Convolution Neural Networks
HARSI	Human Activity Recognition on Signal Images
HTTP	Hypertext Transfer Protocol

HAR-SR	Human activity recognition method based on symbolic representation
RF	Random Forest Networks
MLP	Multi-layer Perception Networks
SVM	Support Vector Machines Networks
NB	Naive Bayes Networks
MIT	Massachusetts Institute of Technology

References

1. Peppas, K.; Tsolakis, A.C.; Krinidis, S.; Tzovaras, D. Real-Time Physical Activity Recognition on Smart Mobile Devices Using Convolutional Neural Networks. *Appl. Sci.* **2020**, *10*, 8482. [CrossRef]
2. Kolkar, R.; Geetha, V. Human Activity Recognition in Smart Home using Deep Learning Techniques. In Proceedings of the 2021 13th International Conference on Information & Communication Technology and System (ICTS), Surabaya, Indonesia, 20–21 October 2021; pp. 230–234. [CrossRef]
3. Li, H.; Trocan, M. Deep learning of smartphone sensor data for personal health assistance. *Microelectron. J.* **2019**, *88*, 164–172. [CrossRef]
4. Dang, M.; Min, K.; Wang, H.; Piran, M.J.; Lee, C.H.; Moon, H. Sensor-based and vision-based human activity recognition: A comprehensive survey. *Pattern Recognit.* **2020**, *108*, 107561. [CrossRef]
5. Kim, B.; Kim, S.; Lee, M.; Chang, H.; Park, E.; Han, T. Application of an Internet of Medical Things (IoMT) to Communications in a Hospital Environment. *Appl. Sci.* **2022**, *12*, 12042. [CrossRef]
6. Athota, R.K.; Sumathi, D. Human activity recognition based on hybrid learning algorithm for wearable sensor data. *Sensors* **2022**, *24*, 100512. [CrossRef]
7. Gowthami, M.; Kumar, V.R. A hybrid DL with the Internet of Things to monitor human activities using wearable sensors. *Sensors* **2022**, *24*, 100496. [CrossRef]
8. Giorgi, G.; Saracino, A.; Martinelli, F. Using recurrent neural networks for continuous authentication through gait analysis. *Pattern Recognit. Lett.* **2021**, *147*, 157–163. [CrossRef]
9. Hofmann, C.; Patschkowski, C.; Haefner, B.; Lanza, G. Machine Learning Based Activity Recognition To Identify Wasteful Activities In Production. *Procedia Manuf.* **2020**, *45*, 171–176. [CrossRef]
10. Sena, J.; Barreto, J.; Caetano, C.; Cramer, G.; Schwartz, W.R. Human activity recognition based on smartphone and wearable sensors using multiscale DCNN ensemble. *Neurocomputing* **2021**, *444*, 226–243. [CrossRef]
11. Prasad, A.; Tyagi, A.K.; Althobaiti, M.M.; Almulihi, A.; Mansour, R.F.; Mahmoud, A.M. Human Activity Recognition Using Cell Phone-Based Accelerometer and Convolutional Neural Network. *Appl. Sci.* **2021**, *11*, 12099. [CrossRef]
12. Weiss, G. WISDM Smartphone and Smartwatch Activity and Biometrics Dataset. Available online: https://archive.ics.uci.edu/dataset/507/wisdm+smartphone+and+smartwatch+activity+and+biometrics+dataset (accessed on 1 February 2023).
13. Ignatov, A. Real-time human activity recognition from accelerometer data using Convolutional Neural Networks. *Appl. Soft Comput.* **2018**, *62*, 915–922. [CrossRef]
14. Bragança, H.; Colonna, J.G.; Oliveira, H.A.B.F.; Souto, E. How Validation Methodology Influences Human Activity Recognition Mobile Systems. *Sensors* **2022**, *22*, 2360. [CrossRef] [PubMed]
15. Najeh, H.; Lohr, C.; Leduc, B. Dynamic Segmentation of Sensor Events for Real-Time Human Activity Recognition in a Smart Home Context. *Sensors* **2022**, *22*, 5458. [CrossRef] [PubMed]
16. Helmi, A.M.; Al-qaness, M.A.A.; Dahou, A.; Damaševičius, R.; Krilavičius , T.; Elaziz, M.A. A Novel Hybrid Gradient-Based Optimizer and Grey Wolf Optimizer Feature Selection Method for Human Activity Recognition Using Smartphone Sensors. *Entropy* **2021**, *23*, 1065. [CrossRef] [PubMed]
17. Mekruksavanich, S.; Jitpattanakul, A. Deep Learning Approaches for Continuous Authentication Based on Activity Patterns Using Mobile Sensing. *Sensors* **2021**, *21*, 7519. [CrossRef] [PubMed]
18. Gul, M.A.; Yousaf, M.H.; Nawaz, S.; Ur Rehman, Z.; Kim, H. Patient Monitoring by Abnormal Human Activity Recognition Based on CNN Architecture. *Electronics* **2020**, *9*, 1993. [CrossRef]
19. Magdin, M.; Benc, J.; Koprda, Š.; Balogh, Z.; Tuček, D. Comparison of Multilayer Neural Network Models in Terms of Success of Classifications Based on EmguCV, ML.NET and Tensorflow.NET. *Appl. Sci.* **2022**, *12*, 3730. [CrossRef]
20. Mohamad, S.; Sayed-Mouchaweh, M.; Bouchachia, A. Online active learning for human activity recognition from sensory data streams. *Neurocomputing* **2020**, *390*, 341–358. [CrossRef]
21. Cheng, X.; Zhang, L.; Tang, Y.; Liu, Y.; Wu, H.; He, J. Real-Time Human Activity Recognition Using Conditionally Parametrized Convolutions on Mobile and Wearable Devices. *IEEE Sens. J.* **2022**, *22*, 5889–5901. [CrossRef]
22. Cengiz, A.B.; Birant, K.U.; Cengiz, M.; Birant, D.; Baysari, K.T. Improving the Performance and Explainability of Indoor Human Activity Recognition in the Internet of Things Environment. *Symmetry* **2022**, *14*, 2022. [CrossRef]
23. Mekruksavanich, S.; Jitpattanakul, A.; Youplao, P.; Yupapin, P. Enhanced Hand-Oriented Activity Recognition Based on Smartwatch Sensor Data Using LSTMs. *Symmetry* **2020**, *12*, 1570. [CrossRef]
24. Bragança, H.; Colonna, J.G.; Lima, W.S.; Souto, E. A Smartphone Lightweight Method for Human Activity Recognition Based on Information Theory. *Sensors* **2020**, *20*, 1856. [CrossRef] [PubMed]

25. Farag, M.M. Matched Filter Interpretation of CNN Classifiers with Application to HAR. *Sensors* **2022**, *22*, 8060. . [CrossRef] [PubMed]
26. Mairittha, N.; Mairittha, T.; Inoue, S. On-Device Deep Learning Inference for Efficient Activity Data Collection. *Sensors* **2019**, *19*, 3434. [CrossRef] [PubMed]
27. Issa, M.E.; Helmi, A.M.; Al-Qaness, M.A.A.; Dahou, A.; Abd Elaziz, M.; Damaševičius, R. Human Activity Recognition Based on Embedded Sensor Data Fusion for the Internet of Healthcare Things. *Healthcare* **2022**, *10*, 1084. . [CrossRef] [PubMed]
28. Wang, X.; Shang, J. Human Activity Recognition Based on Two-Channel Residual-Channel Residual–GRU–ECA Module with Two Types of Sensors. *Electronics* **2023**, *12*, 1622. [CrossRef]
29. Elkelany, A.; Ross, R.; Mckeever, S. WiFi-Based Human Activity Recognition Using Attention-Based BiLSTM. In Proceedings of the Artificial Intelligence and Cognitive Science. AICS 2022, Munster, Ireland, 8–9 December 2022; Longo, L., O'Reilly, R., Eds.; Communications in Computer and Information Science; Springer: Cham, Switzerland, 2023; Volume 1662, pp. 121–133. [CrossRef]
30. Weiss, G.M.; Yoneda, K.; Hayajneh, T. Smartphone and Smartwatch-Based Biometrics Using Activities of Daily Living. *IEEE Access* **2019**, *7*, 133190–133202. [CrossRef]
31. Alexan, A.; Alexan, A.; Oniga, Ş. Smart watch activity recognition using plot image analysis. In Proceedings of the 2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS), Debrecen, Hungary, 16–18 May 2022; pp. 1–6. [CrossRef]
32. Harden, S.W. ScottPlot Library for .NET Source Code. Available online: https://github.com/ScottPlot/ScottPlot (accessed on 1 February 2023).
33. Alexan, A. Deployed Web App. Currently Not Available Online Anymore, Was Exposed for Testing under the Form of Secured Web API Application. Available online: https://mlprocessorwebapi.azure-api.net (accessed on 1 February 2023).
34. Harden, S.W. ScottPlot Library for .NET. Available online: https://scottplot.net/ (accessed on 1 February 2023).
35. Harden, S.W. ScottPlot Library for .NET NuGet Package. Available online: https://www.nuget.org/packages/ScottPlot/ (accessed on 1 February 2023).
36. NVIDIA. CUDA Toolkit. Available online: https://developer.nvidia.com/cuda-toolkit (accessed on 1 February 2023).
37. NVIDIA. NVIDIA cuDNN. Available online: https://developer.nvidia.com/cudnn (accessed on 1 February 2023).
38. Microsoft. ML.NET Model Builder GPU Support. Available online: https://marketplace.visualstudio.com/items?itemName=MLNET.ModelBuilderGPU2022 (accessed on 1 February 2023).
39. Microsoft. ML.NET Model Builder. 2022. Available online: https://marketplace.visualstudio.com/items?itemName=MLNET.ModelBuilder2022 (accessed on 1 February 2023).
40. Roeder, L. Netron Web App. Available online: https://netron.app/ (accessed on 1 February 2023).
41. Alexan, A. Source Code. Available online: https://bitbucket.org/alexandruAlexan/publicwisdmsmartwatchhar/src/master/ (accessed on 1 February 2023).