Tiny Machine Learning for Human Activity Recognition

Gianluca Di Tuccio
University of Bologna
Cesena, Italy
gianluca.dituccio@studio.unibo.it

Lorenzo Orsini
University of Bologna
Bologna, Italy
lorenzo.orsini4@studio.unibo.it

***Abstract*—Tiny Machine Learning (TinyML) has emerged as a promising approach for deploying machine learning models on resource-constrained devices, enabling real-time inference and edge computing capabilities. Human Activity Recognition (HAR) is a significant application domain where TinyML models can provide valuable insights for healthcare monitoring, sports analysis, and human-computer interaction. In this report, we present a comparative analysis of Dense Neural Networks (DNN) and 2D Convolutional Neural Networks (2D CNN) for TinyML in HAR, considering feature selection based on variance as a preprocessing step. Our study utilizes a dataset comprising samples of a 561-feature vector with time and frequency domain variables, collected from wearable sensors. To reduce dimensionality and optimize model performance, we apply feature selection based on variance to select the most informative features. We investigate the impact of different feature subset sizes on the performance of DNN and 2D CNN models. For each model type, we trained and evaluated the models on the reduced feature set using as performance metrics the micro F1 score. To further optimize the models for deployment on microcontrollers, we applied a post-training quantization technique aimed at reducing the size of the models. This quantization process converts the model's floating-point parameters to fixed-point representations, effectively reducing memory requirements without significant loss of accuracy. We evaluated various parameters, including Multiply-Accumulate operations (MACC), latency, and flash usage, to assess the effectiveness of the quantization process in optimizing the model for resource-constrained environments.**

The experimental results demonstrate that both DNN and 2D CNN models, with the applied feature selection and post-training quantization, achieve competitive accuracy in HAR tasks. We observed that the models strike a balance between complexity and accuracy, considering the limitations of resource-constrained environments.

***Index Terms*—Human Activity Recognition (HAR), Internet of Things (IoT), Edge Intelligence, Feature Selection, Embedded systems, TinyML**

I. Introduction

Human Activity Recognition (HAR) has emerged as a vital research area in the field of wearable technology and the Internet of Things (IoT). The ability to accurately recognize and classify human activities has numerous applications, including healthcare monitoring, fitness tracking, gesture control, and context-aware systems [1]. Traditional HAR systems typically rely on complex and resource-intensive algorithms that are executed on powerful computing platforms. However, the advancements in microcontrollers and the growing popularity of Tiny Machine Learning (TinyML) techniques [2] have opened up new possibilities for performing HAR directly on resource-constrained devices. TinyML refers to the deployment of machine learning models on low-power microcontrollers, enabling on-device inference with reduced latency and improved privacy. The application of TinyML to HAR brings several advantages, such as real-time activity recognition, reduced energy consumption, and enhanced user privacy by minimizing data transmission to external servers. This has paved the way for developing wearable devices that can autonomously recognize and interpret human activities without relying on constant connectivity to the cloud.

In this work, we focus on the application of TinyML for HAR, aiming to investigate the effectiveness of deploying lightweight machine learning models directly on wearable IoT devices. Specifically, we compare the performance of Dense Neural Network (DNN) models with 2D Convolutional Neural Network (2D CNN) models in terms of accuracy, model size, and inference time. To achieve this, we employ a dataset consisting of a feature vector with 561 dimensions, collected from various wearable sensors. Given the high dimensionality of the data, we apply feature selection techniques to reduce the input size while preserving the discriminative information. We analyze the quantized models in two aspects. First, we evaluate their performance in terms of the micro F1 score, which measures the accuracy of the models in inference tasks. Second, we examine the impact of the quantized models on the parameters of the microcontrollers, including latency, Multiply-Accumulate operations (MACC), and flash usage. These analyses provide insights into both the model's performance and its compatibility with resource-constrained microcontrollers. The remainder of this paper is organized as follows.

The remainder of this paper is organized as follows. Section 2 provides an overview of our project architecture, while Section 3 describes the project implementation. Section 4 presents the experimental results and performance evaluation, including the analysis of the quantized models on the microcontroller. Finally, Section 5 discusses the findings and implications of the study, concluding the paper with a summary of the contributions.

II. Project Architecture

In this Section, we introduce our HAR system for wearable IoT devices. The pipeline is shown in Figure 1. Our system architecture comprises two types of computational nodes: an IoT wearable device and an external server. The Human Activity Recognition (HAR) process is divided into three main phases: data gathering, model training, and HAR inference. The wearable IoT device handles the data gathering and HAR inference phases, while the model training phase is carried out on an external server. This division is necessary because the model training phase requires substantial computational power and storage resources that are beyond the capabilities of the microcontroller. However, to simplify our work, we only implemented the last two steps, skipping the data acquisition. Indeed, we used an already existing dataset [3] to train our Deep Learning model. This dataset consists of a training set with 7352 samples and a test set with 2947 samples, where each sample is a 561 feature vector. In particular, each sample can belong to one of the six following classes: walking, walking upstairs, walking downstairs, sitting, standing and laying. Since each input has 561 features, which is impractical for a model to run on a resource-constrained microcontroller, we performed a data pre-processing to reduce their dimensionality before the training process.
For this work, we used the microcontroller STM32F4-DISCOVERY, with CPU STM32F412ZGT6 Arm Cortex M4-Core 100MHz, 1 MB flash and 256KB ram.

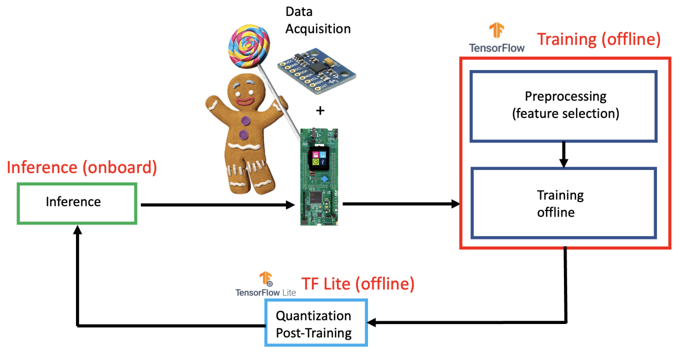

Fig. 1. Pipeline of the data: from data acquisition to an inference onboard.

III. Project Implementation

In this chapter, we will delve into the steps of our pipeline.

A. Data Pre-processing

To address the issue of high dimensionality in our data, which comprises a feature vector of dimension 561, we utilize feature selection techniques based on variance as a criterion. Our objective is to identify the optimal number of features by tuning the selection process on the validation set. We specifically explore the utilization of a neural network architecture consisting of only Dense Layers for our model. Intuitively, features with low variance contribute less to the overall information content and are likely to be less informative for classification tasks. Hence, we aim to identify a subset of features that capture the most salient information while discarding irrelevant or redundant features. To determine the optimal number of features for our model, we perform extensive experiments on the validation set. We systematically vary the number of features in the selected subset, exploring values ranging from 50 to 170 using as metrics the micro F1 score. As a model for the evaluation, we adopted a neural network with two Dense layers, where the number of nodes in each layer is respectively half and one-fourth of the input dimension. To mitigate overfitting, we inserted a dropout layer with a dropout rate of 0.4 between these two Dense layers. The output from this architecture is then passed to the classification head of the network, which comprises 6 nodes responsible for producing the probabilities of membership in each class (Figure 3). The results of the fine-tuning are shown in Table I, where it is also reported the micro F1-score of a baseline Neural Network that takes as input the raw vectors of 561 features, whose architecture is shown in Figure 2.

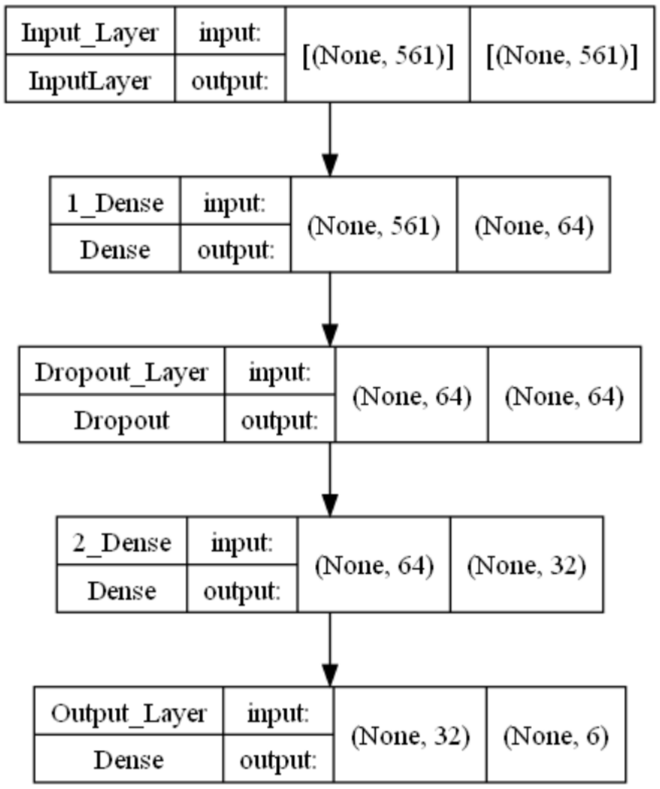

Fig. 2. Architecture of the baseline model.

B. Models adopted

Once the data pre-processing phase was completed, we proceeded to select three models for further analysis. These models were chosen based on different input sizes: the model with the smallest input size (50), the model with the largest input size (170), and the model with the best micro F1 score

TABLE I
VARIANCE FINE-TUNING ON THE VALIDATION SET USING THE DENSE MODEL.

Features	N. Params	Micro F1
50	**1665**	**0.899**
60	2391	0.918
70	3205	0.926
80	4186	0.937
90	5245	0.936
100	6481	0.944
110	7785	0.937
120	9276	0.954
130	10825	0.969
140	12571	0.969
150	14365	0.974
160	**16366**	**0.978**
170	**18405**	**0.975**
561	**38246**	**0.981**

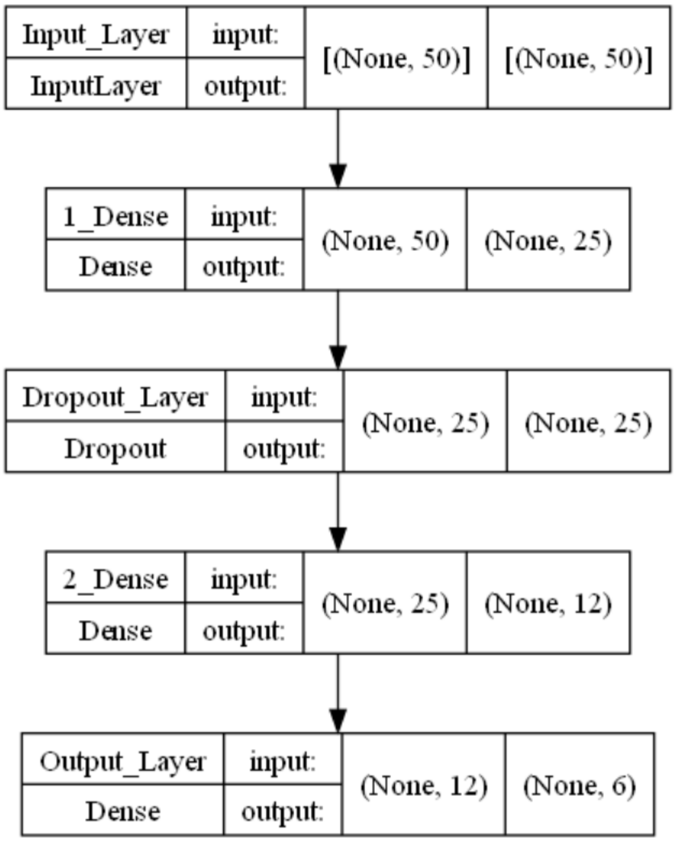

Fig. 3. Architectural overview of the Baseline Model, which is a Dense Neural Network. The model illustrated in the figure exhibits an input size of 50.

on the validation set (160). Our objective was to conduct a comparative study to understand the impact of varying input sizes on model performance. In addition, we extended our analysis to include three 2D Convolutional Neural Network (CNN) models, each with the same input sizes as the selected Dense models. The architecture of the CNN models consisted of a single convolutional layer with 16 filters of size 5x5. This was followed by a pooling layer and a dense layer with 16 nodes. To prevent overfitting, a dropout layer with a rate of 0.3 was introduced between the dense layer and the classification head (Figure 4). Notably, the samples in our dataset were

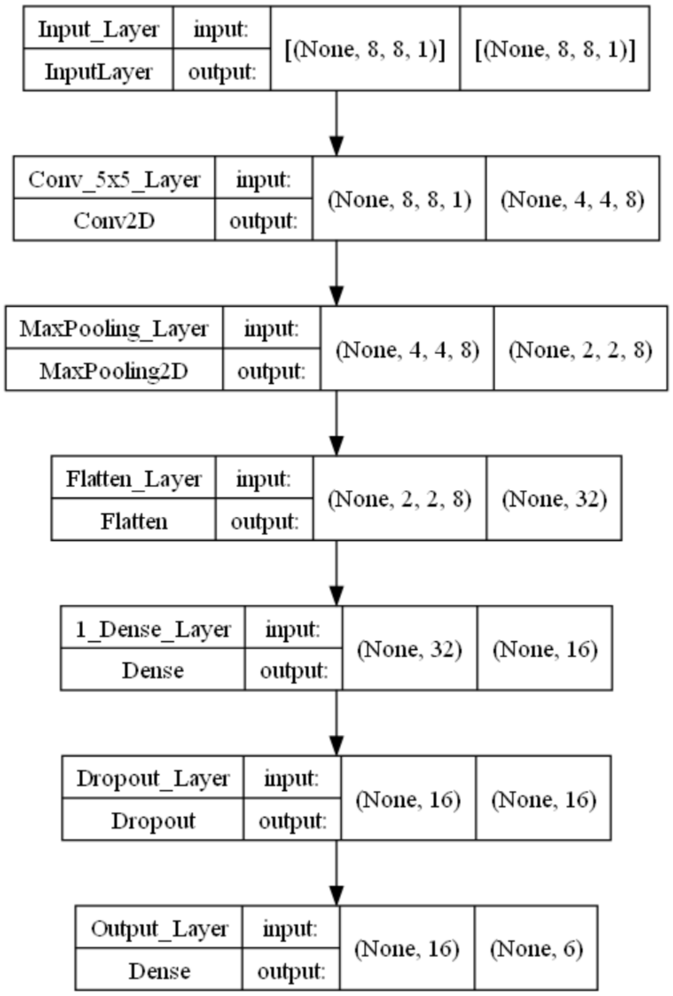

Fig. 4. Architecture of the CNN model with Input Size 50. The input is visualized as an 8x8 square matrix, obtained through zero padding on the original input.

transformed into 2D matrices, resembling black and white images, using zero padding to achieve square matrices.
Finally, we compare all the models by using the micro F1 score on the test set, as highlighted in Table II.

TABLE II
COMPARISON OF MICRO F1 SCORES ON TEST SET: DENSE AND 2D CNN MODELS WITH DIFFERENT INPUT SIZE

Model-Features	N. Params	Micro F1
Dense-50	1665	0.855
CNN-50	838	0.822
Dense-160	16366	0.933
CNN-160	2374	0.924
Dense-170	18405	0.929
CNN-170	3526	0.917
Dense-561	38246	0.952

C. Quantization

In the end, to deploy our model on the microcontroller, we need to apply Post-training quantization, which is a conversion technique that can reduce model size with little degradation in model accuracy. In particular, we adopted Dynamic range quantization, which statically quantizes only the weights from floating point to integer at conversion time, which provides 8 bits of precision. All the models were quantized with this technique by using Tensorflow Lite. Then, we evaluate the performance of the quantized models on the test set, as shown in Table III, which also reports the respective memory size.

TABLE III
COMPARISON OF MICRO F1 SCORES BEFORE AND AFTER DYNAMIC QUANTIZATION ON TEST SET. DQ=DYNAMIC QUANTIZATION

Model-Features	Size Before DQ (KB)	Size After DQ (KB)	Micro F1 Before DQ	Micro F1 After DQ
Dense-50	54	5	0.855	0.856
CNN-50	49	6	0.822	0.822
Dense-160	231	20	0.933	0.933
CNN-160	67	6	0.924	0.923
Dense-170	255	22	0.929	0.930
CNN-170	81	7	0.917	0.914
Dense-561	494	41	0.952	0.948

The "Size After DQ" column now includes the corresponding sizes in kilobytes for each model after dynamic quantization.

D. Performances on microcontroller

Finally, we present a comprehensive evaluation of the performance of quantized models in terms of latency, Multiply-Accumulate (MACC) operations, and flash memory usage (Table IV). Latency refers to the time it takes for a model to process a single inference. Reduced latency ensures timely and responsive predictions, enhancing the overall user experience. We measure the latency of quantized models by recording the time taken for inference on representative input data. MACC operations are fundamental arithmetic operations that form the backbone of neural network computations. Each MACC operation involves multiplying two values and adding the result to an accumulator. MACC operations provide a metric for evaluating the computational complexity and efficiency of quantized models. Flash memory refers to the non-volatile memory available on microcontrollers and plays a critical role in storing the model parameters and code. Evaluating the flash usage of quantized models allows us to understand the trade-off between model size and accuracy.

TABLE IV
EVALUATION OF THE QUANTIZED MODELS ON MICRO (STM32F4-DISCO). ALL THE MODELS HAVE BEEN TESTED WITH A CLOCK OF 16MHZ.

Model-Features	Used Flash (KB)	MACC	Latency (ms)	Micro F1
Dense-50	6.50	1792	1.02	0.856
CNN-50	3.27	4200	4.3	0.822
Dense-160	63.93	16576	7.46	0.933
CNN-160	9.27	19640	17.76	0.923
Dense-170	71.89	18622	8.42	0.930
CNN-170	13.77	25032	24.80	0.914
Dense-561	149.40	38432	16.74	0.949

IV. RESULTS

Now, let's discuss the results obtained in the previous sections.
Firstly, we will examine the process of feature selection. As depicted in Table I, the Baseline model, which operated on the raw input comprising 561 features, achieved an impressive micro F1 score of 0.98 on the validation set. Subsequently, we proceeded to explore the impact of feature selection by progressively reducing the number of features. Remarkably, as the feature count decreased, we observed a gradual decrease in performance. However, this decrease was minimal, with only a slight difference observed between each step of reducing the number of features. For instance, with 160 features, the model attained a micro F1 score of 0.978, while with 170 features, it achieved a score of 0.975. These findings underscore the effectiveness of feature selection in reducing dimensionality without compromising competitive performance. The feature selection approach enabled us to achieve nearly identical performance compared to the Baseline model while employing less than half the parameters. By discarding approximately 400 features, we accomplished comparable results with a considerably more streamlined model architecture. Furthermore, we noted that by reducing the input size by a factor of 10, from 561 to 50 features, the model achieved a micro F1 score of 0.89. Although this score was 10% lower than that of the baseline model, it remained acceptable. Additionally, we observed a significant reduction in the number of parameters, with only 1665 compared to the original 38246, highlighting the efficiency gained through feature selection.
Next, we proceed with the comparison between 2D CNN and Dense Networks. The analysis is based on Table II, which presents the performance of both models on the test set, considering different input sizes. Interestingly, the performances of the two models are remarkably similar for each input size, with the highest difference being only 3% for an input size of 50. However, what distinguishes these models is the number of parameters they employ. Since the convolutional layer utilizes parameter sharing, the Dense Network consistently has a higher parameter count. Moreover, while the number of convolutional filters remains fixed at 16, the Dense Networks' layer sizes depend on the input size. Consequently, the gap in parameter count between the two models widens as the input size increases, while the difference in performance remains insignificant. This observation is particularly evident in models with 170 features, where the micro F1 score differs by approximately 1%, while the Dense Neural Network employs six times more parameters than the 2D CNN. Comparing these results with the Baseline model, we observe that the models with 160 and 170 features exhibit minimal differences. However, for models with 50 features, a slightly larger disparity becomes evident, although the results remain acceptable. The Dense model shows a performance

difference of approximately 10%, while the 2D CNN model demonstrates a difference of 13%. Notably, the 2D CNN model achieves these results while utilizing an impressive 46 times fewer parameters compared to the Baseline model.

Then, we move on to the results of the quantization process (Table III). In terms of performance before and after dynamic quantization with TF Lite, it was observed that the results were nearly identical, with minimal differences of approximately ±0.01%, except for the 561-feature model. In the case of the 561-feature model, there was a slight decrease of 0.4% in the F1 score. What we need also to examine is the size of the models before and after the quantization process: every model is reduced by almost a factor 10. This substantial reduction in model size demonstrates the effectiveness of dynamic quantization in optimizing models for deployment on resource-constrained devices without compromising their performance.

Finally, we will illustrate the performances of the quantized models on the microcontroller (Table IV). One noticeable trend is that the Dense Neural Network models consistently require higher flash usage compared to the 2D CNN models. For instance, the Dense-50 model utilizes 6.50 KB of flash, while the CNN-50 model uses only 3.27 KB. This disparity in flash usage can be attributed to the Dense Neural Network's architecture, which typically contains a higher number of parameters and thus requires more memory for storage. On the other hand, the CNN models, with their convolutional layers employing parameter sharing, exhibit a more efficient use of flash storage. Another notable observation is that the CNN models generally have higher MACC counts compared to their Dense Neural Network counterparts. For example, the Dense-50 model has 1792 MACC, while the CNN-50 model has 4200 MACC. This difference can be attributed to the inherent nature of convolutional operations in CNNs, which involve multiple multiplications and additions per element. In contrast, Dense Neural Network models rely on fully connected layers, resulting in a lower MACC count. Furthermore, the latency of the models follows a similar trend, with CNN models exhibiting higher latency compared to the Dense Neural Network models. This higher latency arises from the computational complexity of the convolutional layers, which involve convolving filters over the input data, resulting in more intensive computations compared to the multiplications in Dense Neural Network layers. It is worth mentioning that despite being the most complex model, the Dense-170 model exhibits impressive resource efficiency on the microcontroller. In comparison to the baseline model, the Dense-170 model demonstrates approximately half the flash usage, MACC count, and latency. Specifically, the Dense-170 model utilizes significantly less flash storage and performs approximately half the number of MACC operations compared to the baseline model. Furthermore, it achieves a substantially lower latency. Remarkably, these resource savings and improved latency are achieved with only a marginal decrease of 3% in the micro F1 score.

V. Conclusion

In this work, we presented a comprehensive study on Human Activity Recognition (HAR) using Tiny Machine Learning (TinyML) techniques for wearable Internet of Things (IoT) devices. Through the pipeline of data pre-processing, model selection, and quantization, we aimed to optimize the performance of HAR models while addressing the constraints of resource-constrained microcontrollers.

Our research demonstrated the effectiveness of feature selection techniques based on variance, which successfully reduced the dimensionality of the data while maintaining competitive performance. By discarding irrelevant or redundant features, we achieved streamlined models that achieved similar results to the baseline model while utilizing significantly fewer parameters. The comparison between Dense Neural Network (DNN) models and Convolutional Neural Network (CNN) models revealed that both architectures performed comparably, with CNN models exhibiting a notable advantage in terms of parameter efficiency due to parameter sharing. This finding emphasizes the potential of CNN models for HAR on wearable IoT devices, where minimizing model size and computational complexity is crucial. Furthermore, we explored the application of post-training quantization, specifically dynamic range quantization, to reduce model size without compromising accuracy. Our results demonstrated that quantization effectively optimized the models for deployment on resource-constrained microcontrollers, achieving nearly identical performance to non-quantized models. Furthermore, we conducted a comprehensive evaluation of the performance of quantized models in terms of latency, Multiply-Accumulate (MACC) operations, and flash memory usage. 2D CNN models, with their parameter sharing, demonstrated significant advantages in terms of parameter count, resulting in lower latency, reduced Multiply-Accumulate (MACC) operations, and optimized memory usage. The reduced latency ensured timely and responsive predictions, while the decreased MACC operations and memory usage made CNN models well-suited for deployment on resource-constrained microcontrollers.

In conclusion, our study highlights the efficacy of TinyML techniques in optimizing HAR models for wearable IoT devices, achieving high performance with reduced resource requirements. 2D CNN models stand out as a promising choice, offering parameter efficiency and optimized resource utilization for real-time HAR applications.

References

[1] ”Damien Bouchabou et al. “A Survey of Human Activity Recognition in Smart Homes Based on IoT Sensors Algorithms: Taxonomies, Challenges, and Opportunities with Deep Learning”. In: *Sensors, MDPI, 2021, 21* (2021). DOI: https://doi.org/10.48550/arXiv.2111.04418.

[2] ”Rakhee Kallimani et al. “TinyML: Tools, Applications, Challenges, and Future Research Directions”. In: (2023). DOI: https://doi.org/10.48550/arXiv.2303.13569.

[3] ”Jorge Reyes-Ortiz et al. “Human Activity Recognition Using Smartphones”. In: (2012). DOI: 10.24432/C54S4K.