Automatic Features Extraction for Human Activity Recognition through Deep Learning

Giovanni Zanella[†], Marco Ceccato[‡]

Abstract—**Human Activity Recognition (*HAR*) with wearable sensors is an open challenge leading to several possible solutions and wide applications range, from everyday usage, to medical support. The strategies to deal with *HAR* are evolving with new technologies, and in this work we propose three novel deep learning schemes to achieve high-performing activity classifiers, comparing their performance with existent ones. This work has been done on a *sensor-based* dataset taken from the German Aerospace Center (DLR) collected using a single 9-axis IMU. The first proposed scheme uses a CNN taking as input an old-fashioned set of *Hand-Crafted Features*, the second one exploits an *Automatic Features Extraction (AFE)* via CNN after an input adaptation into a 2D virtual image. The last scheme perform an AFE through Convolutional Stacked Autoencoders and then an hybrid classifier composed of LSTM and DNN layers. An experimental demo is furthermore implemented, using the proposed schemes, to obtain a smartphone based online activity recognition classifier.**

Index Terms—**Human Activity Recognition, Automatic Feature Extraction, Convolutional Neural Network , Stacked Autoencoders, Deep Learning**

I. Introduction

THE knowledge of current activity of a subject can be a really useful information in many situations [1]. For example activity and position awareness can proactively help subjects on tasks of daily living [2]. If combined with portable technologies, Human Activity Recognition, can lead to interesting implications on smartphone Application that are based on automatic feed-back, bringing these technologies in an uncontrolled environment valorizing their flexibility [3]. On the other side, static and controlled environment such as factories and work places could exploit these technologies as well. Having information about employees activities would allow to optimize industrial processes for the wellness of companies and workers [4]. Activity and gesture recognition are also spreading in the video games technologies. Full-body movements are widely integrating in this field , through applications via video or inertial sensors [5] [6]. Biological data comprehension can be useful also in sports technologies where the understanding of an athlete movement can lead to a higher awareness of sports-related techniques. This can be true for a better personalization of his tools or garements [7], but can also be a helpful to the subject himself to improve his movements [8]. Great strides on HAR have been made in *Pattern Recognition (PR)* using *Hand-Crafted Features (HCF)* extracted from wearable sensors data, however this approach has inherent weakness, like expensive preprocessing [3], generalization problems [9], and performances highly dependent from the problems knowledge. In this paper will be presented three different algorithms for Human Activity Classification all of them based on a CNN approach. Other attempts have been done with same or different techniques. For instance in [3] K. Frank et Al used static/dinamic Bayesian Networks (BN) as classifier. They hand crafted 19 features functions that, if applied to the raw data, will return a set of coefficients that will feed the BN. This strategy has two main weakness spots. The first one is that the definition of the features functions is a very problem-wise challenge, thus the performances of the network will depend on knowledge of the designer about the classification problem. In this paper we will propose two strategies that better generalize the *HAR*, and thus, more flexible to other classification problem setup. The second notable issue in [3] is that the feature extraction represent an uncontrolled loss of information from raw data to the feature set. By using an automatic features extraction to reduce the dimension of the problem we claim to maintain a higher information with a better compression. In [3] even having very good results as an overall recall and precision over 90 % they lacked a clear and exhaustive validation procedure. Our validation technique follows the *K-Fold Cross Validation* (KFC) validation philosophy, leading to results that are clearer and more robust [10].

This paper propose a solution for the following problem. Classifying the activity of a subject between these seven activities: ”RUNNING”, ”STANDING”, ”LYING”, ”JUMPING”, ”WALKING”, ”SITTING”, ”FALLING”. Thanks to the signal easy collection this classification can be part for instance of several applications reacting to the user’s current activity [3]. The data are collected with an Inertial Measurement Unit (IMU) ”Xsens MTx-28A53G25” providing the measurements in the sensor frame. The whole dataset (available here https://www.dlr.de/kn/desktopdefault.aspx/tabid-12705/22182_read-50785/) is composed of more then four hours of labeled activities of people encouraged to act normally. The first strategy we analyzed was defining 29 features functions, applying them on the dataset, and using their output to feed a 1-D CNN. The second strategy includes a *signal to image preprocessing* where each temporal segment is represented by a $49 \times 37 \times 3$ matrix, where 49 are the time samples and 37 are a combination of the raw data axis. From every matrix FFT and PCA coefficients are put beside the raw measurements and will feed a 2-D CNN. The third strategy uses a convolutional autoencoder to find, in a unsupervised manner, a latent representation of raw

[†]Giovanni Zanella, email: zanellagio@dei.unipd.it
[‡]Marco Ceccato, email: ceccatom@dei.unipd.it

signals and then an RNN-based classifier exploiting a fine-tuning scheme to classify each activity. CNNs technologies are fairly recent: born in the sixties started to outperform other algorithms since the beginning of the new millennium [11]. Being such newborns, Neural Networks *(NN)* potentials still have not been properly understood, that's why is important to explore different and known problems from a NN perspective with a strict and rigorous validation process [11]. The three proposed techniques are tested through a KFC validation process [10].

These techniques can be easily implemented for example on portable devices as they need a really compact and cheap set of sensors. Furthermore we propose with this paper a smartphone demo implementation based on MATLAB® computation engine. Our application proved to work on a non ideal uncontrolled scenario with simple sensors from relatively cheap smartphones. Futures works might be an optimization of the computational burden and a characterization of the performance.

The paper is structured as follows. The second section will be about related works, having a sight of the state-of-art techniques. The third section will be about a brief overview of the proposed algorithms structure. The fourth section will clarify the dataset structure and main features. The fifth section will technically describe the learning layers of the CNNs. The sixth section will provide our validation techniques and results. The seventh section will report considerations and implication of our work having a look of possible future related works.

II. Related Work

HAR recognition has become an huge research field, especially on the last years after the exponential growth of the machine learning tools. The first generic works on *HAR* date back to the early 90' [12] [13], where a computer vision framework were used to distinguish some human activities. The use of inertial sensors came some years later in [14] with multiple wearable accelerometers placed on the body. From there the research on *HAR*, using *wearable* and/or *external* sensors, moved on developing more complex frameworks and facing the most challenging design issues. Conventional PR approach have made huge progress on *HAR* through the extraction of complex HCF from raw signals and the use of learning algorithms such as *Support Vector Machine*, *Decision Tree*, *K-Nearest Neighbors*, *Naive Bayes*, and *Hidden Markov Models* [3] [15] [16]. The performance archived with these PR methods are pretty good, however, the recent advancements of deep learning make possible to perform *Automatic feature extraction (AFE)* from raw signals, thus achieving promising improvements overcoming some weakness of the PR approach:

- The feature selection may be a tricky task [17], existent methods work but they are not robust enough [18] and require a lot of resources for exhaustive searches. With AFE instead features selection is not needed anymore, it is done automatically by deep learning structures.
- HCF are limited by human domain knowledge and for high-level activities, such as having a coffee, is not easy to define dedicated features [19], meanwhile high-level features could be extracted by AFE without any expert knowledge and, all of it, easily through a more lightweight pre-processing.
- To train PR models a high dimension of precisely-labeled data is needed, where for AFE models also unlabeled data can be used in the training of unsupervised learning algorithms.
- In PR is hard to correlate the performance with the designer's choices due to different chosen features and learning algorithm used in addition to distinct experimental backgrounds. Is hard to understand the results due to the greater degree of freedom with respect to AFE pre-processing. In the AFE case the results are affected mainly by the choice of learning models and are less prone to the differences of the experimental backgrounds.

The samples generation process for AFE for raw data usually consists of splitting a raw signal into small windows, referred to as temporal windows. In [20] some comparison on sliding window methods, especially on overlapping effects, are investigated. In literature several deep learning models exist, the most recents and commons are: *Convolutional Neural Network (CNN)*, *Recurrent Neural Network (RNN)*, *Autoencoders*, *Restricted Boltzmann machine (RBM)* and *Hybrid Models*.

A. Convolutional Neural Network

In [21] a CNN was used treating each dimension of the accelerometer as one channel (like an RGB image) including also a modified weight sharing technique, called *partial weight sharing*, that adds a sparsity constraint which is not only good for extracting local dependencies, but also reduces the computational complexity. In [22] a similar approach was used, with the addition of unifying the weight sharing among multiple sensors to improve performance. [23] presents interesting results on how the kernel and pooling sizes of 1D convolutional layers and other hyper-parameters, impact on the performance. Treating each dimension as a channel is also known in literature as *Data-driven* [24] whereas resizing the input to form a 2D virtual image is called *Model-driven* approach.One of the first Model-driven work is [25] where they grouped same type signals to capture spatial dependency over signals via 2D convolutional kernel, for different sensor type instead they separated them by zero-padding. In [26] the *input adaptation* is more complex, they have separated acceleration signal into body motion and gravity acceleration then some dimensions are duplicated in order to look at spatial relation between over all the axis of the temporal window and finally they computed a DFT on each axis to obtain a virtual 2D image. An interesting work that helps to understand how the features are automatically extracted via deep CNN is [27] using 2D virtual images as input they investigate the effects of sequential conv layers on the relationship between activity and extracted features.

B. *Autoencoder*

Autoencoders are powerfull learning structures able to obtain latent representation through hidden layers via an unsupervised encoding-decoding procedure. *Stacked Autoencoder (SAE)* is the stack of several autoencoders [28]. Although SAE exists since a few years, their application on HAR came in the last two years, in [29] they built two stacked autoencoders composed of *Dense* layers and a simple one-layer NN was attached after the encoder part to act as classifier. In [30] a *Denoising* SAE was built adding sparsity constraints on a deep structure, using as input several sensors.

C. *Recurrent neural network and Hybrid Models*

RNN and its evolution LSTM are widely used in other fields, and in the last couple of years they have been used in *HAR* both, alone or integrated with CNN and DNN [31] [32] [33]. Other hybrid models combine CNN with RBM [34]

A cross-cutting work [20] explains that *HAR* isn't yet a standardized field, compared to other classification domains, and considers important issues that prevent a direct comparison of previous works focusing on some weakness like: the employed metrics, the validation protocol used, the samples generation process, the quality of the dataset and the experimental backgrounds.

Summarizing, being relatively newborn, CNNs for *HAR*, are still facing many uncertainties regarding their performance and validation, and standards common rules still have to be defined [20].

III. Processing Pipeline

The dataset is composed of 18 axis, respectively 3 1D measurements for each sensor (accelerometer, gyroscope and magnetometer) and 9 axis of attitude data. The accelerometer measurement, *total acceleration (TA)* can be separated into two components, the *body acceleration (BA)* and the *gravity acceleration (GA)*, using an high pass filter in the first case and a low pass filter in the second one, both the proposed filters have 0.3 Hz as cut-off frequency. In this paper will be proposed three different strategies to the given classification problem.

- The first one is similar to the one proposed by the authors of the dataset and will be referred as *FeaturesConvNet*. From raw signal $M = 29$ HCF are computed on different window sizes and a sliding window is adopted with a dimension finally settled at 32 samples. The obtained $N \times M$ features matrix, where N is the number of samples, is then normalized by rows in order to arrange values between 0 and 1. The learning framework take as input the rows of the features matrix and its structure is composed of the following sequential layers: *1D Convolutional* layer with 16 filters of kernel size 3, *MaxPooling* with size 2 and *tanh* activation, *1D Convolutional* layer with 8 filters of kernel size 3 and *tanh* activation, *FullyConnected* layer with 32 neurons and *tanh* activation, *FullyConnected* layer with 7 neurons and *softmax* activation. This model is essentially done for the sake of analyzing and compare the old style approach with respect to the new advances in this field using the same dataset to make the analysis comparable.
- The second strategy, so-called *AutoConvNet*, aims to convert the raw data to an image-like signal. As anticipated before, the *AutoConvNet* uses a sliding window method to obtain temporal windows of size 69×6, where 69 are number of samples and 6 are the following measures: the first three columns are the TA measurements, the next three columns are gyroscope axis. From each temporal window we obtain three different channels each of them of size 128×37. The first channel is obtained as follows: from the original matrix a 69×9 matrix is obtained splitting the TA in BA and GA. Each of the nine column is repeated until every axis has at least one adjacency with all the others as explained in [26]. Doing so, the first channel reaches a dimension of 69×37. We called this strategy *Augment and shuffle*, and it proved to consistently increase the performance of the *AutoConvNet*. After that a DFT is performed along each axis using padding to increase resolution, obtaining the final structure of the first channel of shape 128×37. The second channel is obtained as the first channel, but without applying the DFT. It has a shape of 69×37. The third channel simply represents the PCA coefficients of the second channel with a shape of 37×37. Zero padding technique has been applied to the second and third channels to make their shape consistent with the first one. Then final shape of the pre-processed dataset is finally $128 \times 37 \times 3 \times numberOfSamples$. The learning framework is composed of: two *2D Convolutional /AveragePooling* layers followed by an *LSTM* layer and two *FullyConnected* layers with a final *softmax* activation. The details about the structure can be found in Figure 2.
- The last proposed structure, called *AutoencoderNet*, take as input the temporal windows of size 96×12 with 50% overlap using a *data driven* approach hence considering 12 different channels. The first nine channels are the same measures of the second structure whereas the last three are orientation signals obtained from the attitude data. The learning framework is composed of two parts: The first is a convolutional SAE composed of 4 autoencoders, its structure can be seen in Figure 3; the second one use the encoder part of the previous autoencoder and, with a fine-tuning technique, attach a classifier composed of one *LSTM* and a *FullyConnected* layer with a *softmax* activation.

IV. Signals and Features

As mentioned before this paper proposes three different strategies to deal with the *HAR* problem, three different networks with different features. The following paragraphs will describe the three different ad-hoc preprocessing approaches. From the original 18 axis dataset just a few of them have been selected. We found the most relevant information regarding activity recognition in the accelerometer and the gyroscope, while the magnetometer information (representing the orientation between the subject and the magnetic field), useless and sometimes harmful considering a supervised training approach. After a *PSD* analysis, we found the signals without relevant noise artifacts, making a pre-filtering not strictly necessary. The choice of the windowing for the six axes is non trivial. Every window should contain the maximum information possible relating its label, and avoid the transition segments. Having 7 heterogeneous activities is hard to say which temporal window is best. For impulsive activities as 'FALLING' a window size over a second is very likely to overlap undesired transition segments, while for activities as 'WALKING' having a window size longer than a second would provide much accurate information than a short term sight.

A. *FeaturesConvNet*

The *FeaturesConvNet* is based on the work of K. Frank et al. [3] and has been taken into account mostly as a reference. The purpose of this paper is to analyze other strategies and exploring other novel approaches, that's why we will present just a brief description of the *FeaturesConvNet*. The preprocessing in this case aims to extract some features from the raw signal and using them as input for a classifier. In [3] they used a Bayesian classifier, and we propose a 1-D CNN instead. We designed a set of 29 feature functions each of them trying to extract different and relevant coefficients from the raw dataset. The 29 features were selected sorting them by importance evaluated using random forest and Relief-F algorithm [35]. That's a crucial point in this network, and from our point of view its main weakness. From raw data to features coefficients there is an uncontrolled loss of information depending on the reliability of the features functions, the *AutoConvNet* and the *AutoencoderNet* will overcome this approach. Describing the feature functions would lie outside this paper topic. For the sake of comprehension, this brief explanation will be enough. The feature vector $\in \mathbb{R}^1$ is obtained by applying the 29 different feature functions centered on the same sample. The functions consider from 8 to 128 samples around the pivot, and apply many different transformations, exploiting both well-known and ad-hoc techniques, extracting only one coefficient each. To reduce redundancy and to make this network comparable with our proposed ones we under-sampled the raw data with a step size of 32 finding it a reasonable trade-off between performance and computational burden. After a simple normalization between 0 and 1 we obtain a $29 \times NumberOfSamples$ matrix that will be the input of our CNN.

B. *AutoConvNet*

For the *AutoConvNet* will be used a 69 samples window size. Considering the sensor sampling frequency $@100Hz$ it means that each matrix feeding the CNN will have a temporal sight around the labeled sample about $0.69s$. The step size is 33 to allow about 50% overlap and a temporal resolution of $0.33s$. CNNs have proved to be really efficient describing correlation in input data, but they lack a higher level analysis. The aim of this preprocessing has been to extract information otherwise hard to be reached by CNNs. That is why the raw temporal data are placed aside PCA and DFT coefficients. DFT coefficients provide a frequency description of each window, and PCA coefficients describe the signal in term of variation along orthogonal axis. As explained above the *Augment and Shuffle* transformation, leads to an image-like signal in some case very recognizable. In Figure 1 it's clear

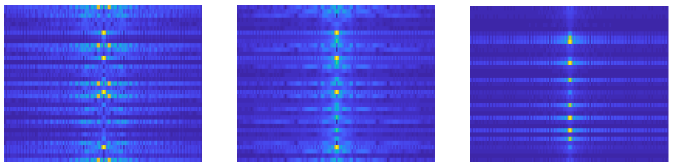
(a) Running (b) Falling (c) Lying

Fig. 1: DFT channel of three different activities

the difference between different activities even by the human eye, while on others is more difficult.

C. *AutoencoderNet*

The set of signals taken by this scheme as input are listed in Figure 2, all of them are firstly filtered by a low pass filter (cutoff 30 Hz). In this model the features are automatically extracted by sequential stacked autoencoders that are forced to learn a latent representation able to reconstruct the original input data. The strong suit of this scheme is that this can be done in a unsupervised way, and then also unlabeled data of the dataset can be used to learn. Nine of the 18 axes that the sensor computes, using a Kalman-based scheme [36], are drift-free attitude signals. From the attitude data that is represented by the *Direction Cosine Matrix (DCM)*, namely R

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{1}$$

Euler rotations (a.k.a. yaw, pitch, and roll) can be easily obtained as:

$$\begin{aligned} \alpha &= \arctan\left(\frac{r_{21}}{r_{11}}\right) \\ \beta &= \arctan\left(\frac{r_{31}}{\sqrt{{r_{32}}^2 + {r_{33}}^2}}\right) \\ \gamma &= \arctan\left(\frac{r_{32}}{r_{33}}\right) \end{aligned} \tag{2}$$

The approach used here is to treat each axis as an independent channel in input of the SAE: the simpler 1D convolutional structure makes the computation easier to perform, however that might imply a lack of correlation analysis among axis in the first layers. To overcome this limit a 2D convolutional framework (model-driven approach) was implemented, however we noted lower reconstruction performance at the decoder

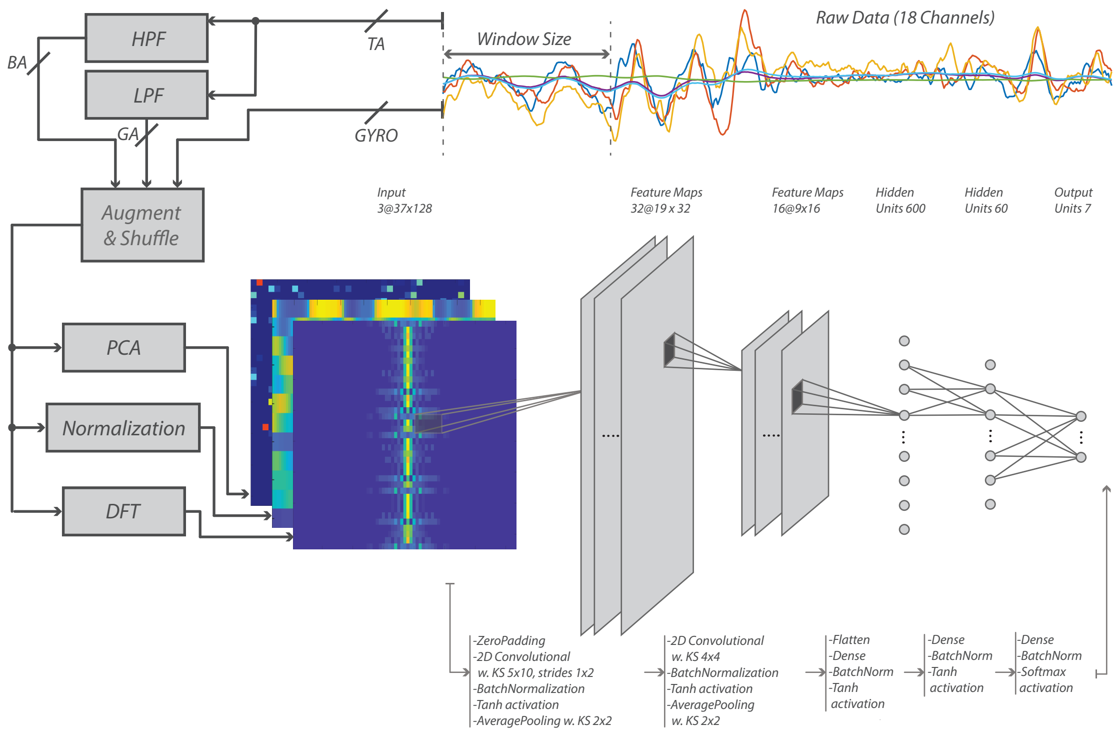

Fig. 2: *AutoConvNet* pipeline. KS stands for Kernel Size

side, and at the classification phase.

V. Learning Framework

In this paper will be proposed three different CNNs with different features and different performance. The three of them are based on an high-level API called KERAS based on TENSORFLOW™ libraries. The three strategies share the evaluation process: a 10 *Fold Cross* validation splitting the dataset in 90% of *Train Set* and 10% of *Test Set*. Dealing with a very unbalanced dataset we needed an adaptive algorithm for the optimization process of our CNNs. Our best performance were reached using optimizer based on [37], with an algorithm called *Adam*. Some of the sizes and hyper-parameters turned out from a grid search, note that some of them were assumed independent to drastically reduce the computational complexity, otherwise too high for our equipment. Some regularization techniques were adopted on several layers to avoid overfitting: *Dropout* to enhance the generalization property of neurons, *L1 Regularization* to increase the sparseness of our networks and *L2 Regularization* in order to limit the weights magnitude. Using these techniques we noted higher values on the loss function, *categorical cross-entropy*, but a remarkable improvement on the validation test accuracy. Other choices regarding sizes of the kernel filters and the number of neurons of each layer were instead inferred after several attempts.

A. *FeaturesConvNet*

This first CNN is very simple: is composed by two blocks composed of *Convolution1D*, *MaxPooling1D*, *BatchNormalization* and *Tanh* as activation. The second block reduces the dimensions of the network until it reaches a dimension of seven with an activation *Softmax* to be comparable with the *one-hot-encoding* of the seven different activities. The structure is shown in Table 1

Layer Name	Kernel Size	Output Size
Input	–	25@16
Zero Padding	–	27@16
Convolution1D	3	25@ 16
MaxPooling1D	2	12@ 16
BatchNormalization	–	12@ 16
Tanh activation	–	12@ 16
Zero Padding	–	14@16
Convolution1D	3	12@ 8
BatchNormalization	–	12@ 8
Tanh activation	–	12@8
Flatten	–	96@1
Dense	–	32@ 1
BatchNormalization	–	32@1
Tanh activation	–	32@1
Dense	–	7@1
Softmax activation	–	7@1

TABLE 1: FeatureConvNet structure

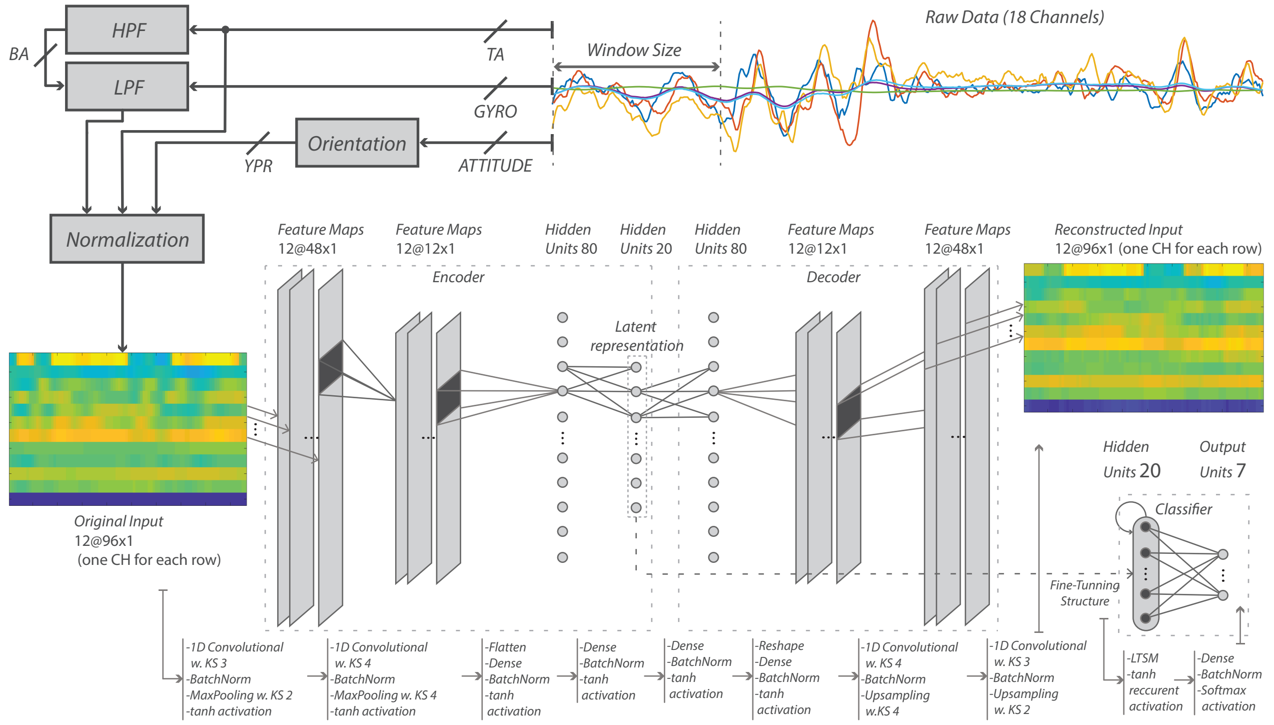
Raw Data (18 Channels)
HPF
LPF
BA
TA
GYRO
Window Size
Orientation
YPR
ATTITUDE
Normalization
Original Input
12@96x1
(one CH for each row)
Feature Maps
12@48x1
Feature Maps
12@12x1
Hidden
Units 80
Hidden
Units 20
Hidden
Units 80
Feature Maps
12@12x1
Feature Maps
12@48x1
Reconstructed Input
12@96x1 (one CH for each row)
Encoder
Decoder
Latent
representation
Hidden
Units 20
Output
Units 7
Classifier
Fine-Tunning
Structure
-1D Convolutional w. KS 3 -BatchNorm -MaxPooling w. KS 2 -tanh activation
-1D Convolutional w. KS 4 -BatchNorm -MaxPooling w. KS 4 -tanh activation
-Flatten -Dense -BatchNorm -tanh activation
-Dense -BatchNorm -tanh activation
-Dense -BatchNorm -tanh activation
-Reshape -Dense -BatchNorm -tanh activation
-1D Convolutional w. KS 4 -BatchNorm -Upsampling w.KS 4
-1D Convolutional w. KS 3 -BatchNorm -Upsampling w. KS 2
-LTSM -tanh reccurent activation
-Dense -BatchNorm -Softmax activation

Fig. 3: *AutoencoderNet* pipeline. KS stands for Kernel Size

B. *AutoConvNet*

As shown in Figure 2 the structure of the learning framework is pretty simple. The first two blocks are composed of *Conv2D*, *BatchNormalization*, *Tanh* as activation and *AveragePooling2D*. After that a *Flatten* layer was used and three *Dense* to reach the final size equal to seven to make the dimensions consistent to the *one-hot-encoding* representation of our activities. Other network more complicated proved to easily overfit our dataset.

C. *AutoencoderNet*

The *AutoencoderNet* has the most complex training process among the previous ones. An unsupervised pre-training is first of all done: each stacked autoencoder is trained in a greedy-layer-wise mode; this helps to mitigate the difficult optimization problem of deep networks by better initializing the weights of all layers reducing the possibility to get stuck in poor solution [38] and attenuating the vanishing gradient problem [39]. Since the dataset is extremely unbalanced, in order to reduce the wide gap between the number of samples of 'FALLING' and the others activities, a synthetic generation of the minority class is obtained in the training phase through the *Adaptive Synthetic Sampling (ADASYN)* algorithm [40], evolution of the most known *SMOTE algorithm* [41]. That over-sampling technique helps to improve the 'FALLING' performance, especially the recall measure as we will see in the results section.
Finally via a fine-tuning scheme the pre-trained encoder is connected to the classifier part of the *AutoencoderNet* and will be subject to a supervised training.

VI. Results

The validation of a classifier is not always an easy task. In the literature have been proposed many different strategies but still today data scientists lack a standardized method to validate CNN performance [20]. In [3] the authors chose to validate their classifier by testing it on the data acquired from two different people not present in the training set. We found this method inadequate. How were these two people chosen? Are this two people a valid representation

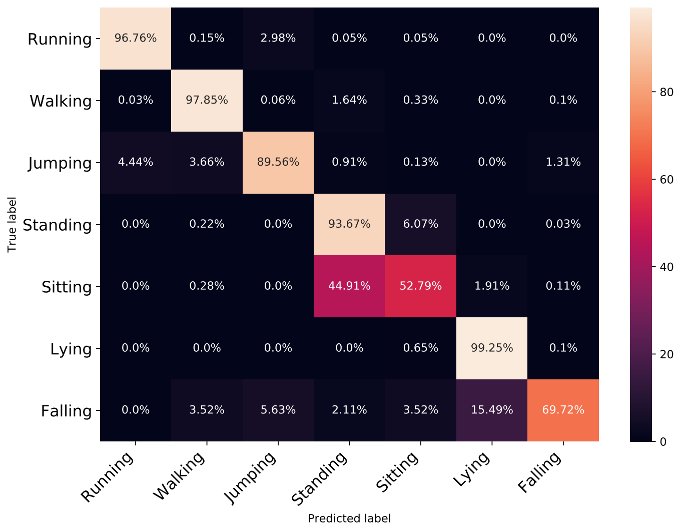

Fig. 4: Confusion matrix of the *AutoencoderNet* scheme, normalized by rows.

Activity	F-Score (%)			Precision (%)			Recall (%)		
	FCN	ACN	AEN	FCN	ACN	AEN	FCN	ACN	AEN
Running	96.93	97.27	97.45	98.75	98.52	98.15	95.18	96.04	96.76
Walking	97.52	97.35	98.37	97.65	95.93	98.90	97.40	98.82	97.85
Jumping	91.04	89.16	90.09	87.48	88.71	90.62	94.91	89.62	89.56
Standing	85.44	87.99	88.70	83.64	86.86	84.22	87.33	89.14	93.67
Sitting	59.74	66.82	61.97	63.81	70.46	75.02	56.15	63.55	52.79
Lying	97.37	99.18	97.87	96.53	99.42	96.53	98.23	98.95	99.25
Falling	79.25	68.93	73.33	75.90	82.43	77.34	82.89	59.22	69.72

TABLE 2: Cross-validation performance for three proposed algorithms, where FCN stands for *FeaturesConvNet*, ACN for *AutoConvNet* and AEN for *AutoencoderNet*

of a generic individual? Isn't it possible to validate the classifier more exhaustively? We found the results proposed in [3] non informative as a validation, but maybe a best-case scenario evaluation. In this paper we validate our networks through a *K-fold Cross* validation (KFC) so that every bit of the our dataset is used both as train set and test set [20]. Looking at the results we have to take into account the dataset shape. This is a case of very unbalanced set where the most frequent label 'STANDING' represent the 38.08% of the dataset and 'FALLING' just the 0.71%. To evaluate the performance of our classifiers we looked mainly at three parameters for each activity: recall, precision and F-score. The details of the performance for our three proposal are listed in Table 2. Due to the inequality of our dataset is better to stress the concept of precision for the sake of understanding the results. Note that the precision of the classes minorly represented is strongly affected by the irregularity of the dataset. For instance a tiny percentage of the majority class misclassification would remarkably reduce the precisions of the minority classes bringing to a misleading comprehension at first glance. Regarding the recall metric, as can be seen from the confusion matrix normalized by rows of Figure 4, we observe two important facts. The first one is that most of the errors are committed among 'STANDING' and 'SITTING' and the explanation is quite simple: the two activities are really similar from an accelerometer perspective, because the torso is oriented the same way during both activities; the only difference occurs whenever the pelvis is slightly rotated. A possible way to improve this misclassification, not covered here, is to use the transitions detection to track the context as did in [42]. The second comment regards the classification errors between 'LYING' and 'FALLING', here the problem is inherent the labeling process since is not easy to label perfectly the impulsive and short activities. From an analysis on the raw data we noted that before, and especially after the true falling moments, the signals are labeled as 'FALLING' even if the subject is still standing or after the fall is on the ground in horizontal position, hence considered as 'LYING' from ours classifiers. The 'FALLING' problem implies poor performance, however we should consider that on 'FALLING' activity, being an high-level activity, is not really important to detect all its duration and sub-phases but just the fact that is occurred might be interesting for most of the applications. As in [3], we chose to remove the transition samples as not taking part of our classification problem and as their quality is prone to the imprecisions of the personnel performing the labeling. In Figure 5 is reported

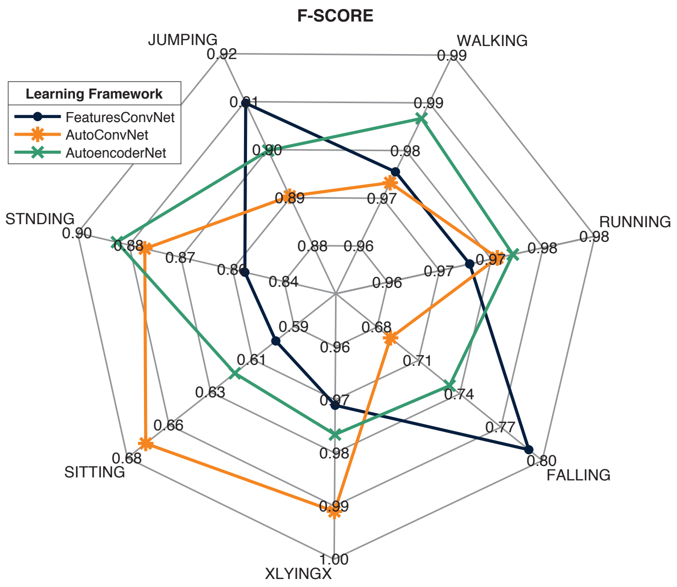

Fig. 5: Comparison of the F-Score results between the three structures proposed

the *F-score* of the three classifier. Here is clear how the three algorithms outperform one another on different activities. The *FeaturesConvNet* performed an higher *F-score* on impulsive activities as 'FALLING' and 'JUMPING'. That is because this framework has many feature functions defined on small time intervals, helping the recognition of impulsive activities. The second strategy *AutoConvNet* proved to be reliable in the static activities as 'LYING', 'SITTING' and 'STANDING'. The DFT channel is strongly influencing the discrimination between static and dynamic activities while the PCA and RawData channels carry good representation of the orientation of the subject. The *AutoencoderNet* achieves the best results on repetitive-dynamic activities such as 'RUNNING' and

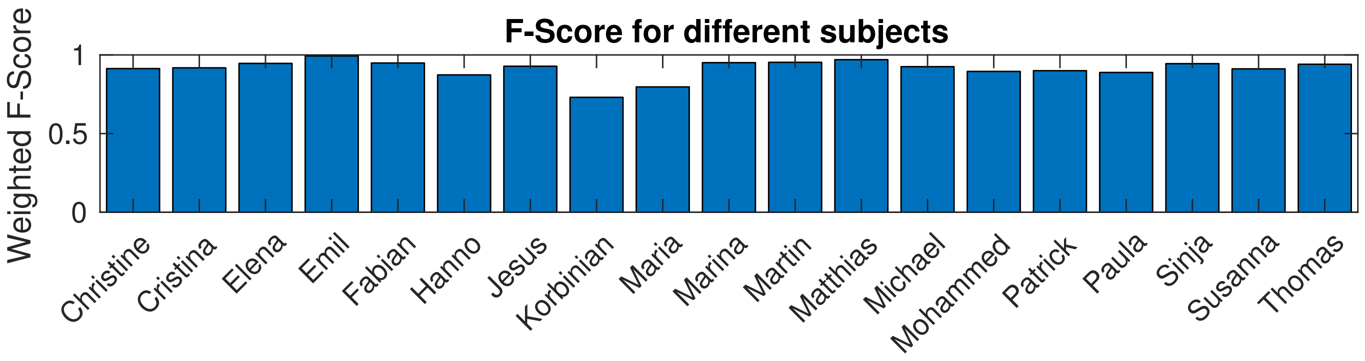

Fig. 6: Comparison of the F-Score results between different subjects, using the *AutoencoderNet* structure

'WALKING', furthermore is the most well balanced structure considering all the activities.
As a final consideration using our KFC validation scheme, we can see in Figure 6 that our proposed frameworks achieve a remarkable generalization property among different subjects: the difference among the subjects is mainly due to different amounts of activities for each user, and there are no anomalous performance.

VII. Concluding Remarks

Along this paper we proposed three different strategies to deal with *HAR*. The *FeaturesConvNet*, based on an old fashioned hand-crafted feature extraction, of very intuitive structure, but lacking of a standardized and controlled preprocessing. The *AutoConvNet* exploiting an image-like preprocessing very reliable on static activities. And the *AutoencoderNet* with high performance on long time repeated activities and with a well balanced overall performance. Futures works could be testing these nets on other dataset and on other *HAR* problems for the sake of an exhaustive validation of the algorithms performance. Furthermore the strength of these techniques derives also from the really cheap sensor set needed for the classification. It makes them implementable in many different devices as regular smartphones. Possible future works could regard the optimization and validation of the three proposed networks on small wearable devices.
In this exam project we learned all the design phases required for a deep learning work. Being our first work on ML topics we started without experience from naive methods, firstly trying to replicate the features extraction proposed in [3] and then integrating new hand-crafted features designed by us. Once all these features were extracted we focused on features selection, founding out that is a tricky task to deal with. After that we built a CNN classifier (*FeaturesConvNet*), and one of the first difficulties was to understand how to properly compute performance metrics for our highly unbalanced dataset. After this steps we realized that the level of innovation in our project was almost null, because that approach was overcome. Thus we decided moving to new approaches headed towards an automatic features extraction. Then two parallel threads have led to two different architectures: *AutoConvNet* and *AutoencoderNet*. The main difficulties encountered were related to the input adaptation of the raw signal and to the choice of the structure. The choice of the hyper-parameters has been a challenge as well, we moved heuristically to reach the best performance but sometimes struggling to understand the *cause-effect* motivation. However we learned that accurate recognition of actions is a highly challenging task due to cluttered backgrounds, occlusions, and viewpoint variations.

References

[1] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Comput. Surv.*, vol. 46, pp. 33:1–33:33, Jan. 2014.
[2] G. D. Abowd, A. K. Dey, R. Orr, and J. Brotherton, "Context-awareness in wearable and ubiquitous computing," in *Digest of Papers. First International Symposium on Wearable Computers*, pp. 179–180, Oct 1997.
[3] K. Frank, M. J. V. Nadales, P. Robertson, and M. Angermann, "Reliable real-time recognition of motion related human activities using mems inertial sensors," in *ION GNSS 2010*, September 2010.
[4] I. Maurtua, P. T. Kirisci, T. Stiefmeier, M. L. Sbodio, and H. Witt, "A wearable computing prototype for supporting training activities in automotive production," in *4th International Forum on Applied Wearable Computing 2007*, pp. 1–12, March 2007.
[5] C.-H. Wu, Y.-T. Chang, and Y. Tseng, "Multi-screen cyber-physical video game: An integration with body-area inertial sensor networks," in *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pp. 832–834, March 2010.
[6] J. Charles and M. Everingham, "Learning shape models for monocular human pose estimation from the microsoft xbox kinect," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)(ICCVW)*, vol. 00, pp. 1202–1208, Nov. 2012.
[7] K. King, S. Yoon, N. Perkins, and K. Najafi, "Wireless mems inertial sensor system for golf swing dynamics," *Sensors and Actuators A: Physical*, vol. 141, no. 2, pp. 619 – 630, 2008.
[8] M. Bächlin, K. Förster, and G. Tröster, "Swimmaster: A wearable assistant for swimmer," in *Proceedings of the 11th International Conference on Ubiquitous Computing*, UbiComp '09, (New York, NY, USA), pp. 215–224, ACM, 2009.
[9] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys Tutorials*, vol. 15, pp. 1192–1209, Third 2013.
[10] A. Vehtari, A. Gelman, and J. Gabry, "Practical bayesian model evaluation using leave-one-out cross-validation and waic," *Statistics and Computing*, vol. 27, pp. 1413–1432, Sep 2017.
[11] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015.
[12] R. Polana and R. Nelson, "Detecting activities," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2–7, June 1993.

[13] R. Polana and R. Nelson, “Recognizing activities,” in *Proceedings of 12th International Conference on Pattern Recognition*, vol. 1, pp. 815–818 vol.1, Oct 1994.

[14] F. Foerster, M. Smeja, and J. Fahrenberg, “Detection of posture and motion by accelerometry: a validation study in ambulatory monitoring,” *Computers in Human Behavior*, vol. 15, no. 5, pp. 571 – 583, 1999.

[15] O. D. Lara and M. A. Labrador, “A survey on human activity recognition using wearable sensors,” *IEEE Communications Surveys Tutorials*, vol. 15, pp. 1192–1209, Third 2013.

[16] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, “Activity recognition using cell phone accelerometers,” *SIGKDD Explor. Newsl.*, vol. 12, pp. 74–82, Mar. 2011.

[17] M. Zhang and A. A. Sawchuk, “A feature selection-based framework for human activity recognition using wearable multimodal sensors,” in *Proceedings of the 6th International Conference on Body Area Networks*, BodyNets ’11, (ICST, Brussels, Belgium, Belgium), pp. 92–98, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011.

[18] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, “Feature selection,” *ACM Computing Surveys*, vol. 50, pp. 1–45, dec 2017.

[19] N. Y. Hammerla, S. Halloran, and T. Ploetz, “Deep, convolutional, and recurrent models for human activity recognition using wearables,” *CoRR*, vol. abs/1604.08880, 2016.

[20] A. Jordao, A. C. Nazare, Jr., J. Sena, and W. Robson Schwartz, “Human Activity Recognition Based on Wearable Sensor Data: A Standardization of the State-of-the-Art,” *ArXiv e-prints*, June 2018.

[21] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, “Convolutional neural networks for human activity recognition using mobile sensors,” in *6th International Conference on Mobile Computing, Applications and Services*, pp. 197–205, Nov 2014.

[22] J. B. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, “Deep convolutional neural networks on multichannel time series for human activity recognition,” in *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI’15, pp. 3995–4001, AAAI Press, 2015.

[23] C. A. Ronao and S.-B. Cho, “Human activity recognition with smartphone sensors using deep learning neural networks,” *Expert Syst. Appl.*, vol. 59, pp. 235–244, 2016.

[24] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, “Deep Learning for Sensor-based Activity Recognition: A Survey,” *ArXiv e-prints*, July 2017.

[25] S. Ha, J. Yun, and S. Choi, “Multi-modal convolutional neural networks for activity recognition,” in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 3017–3022, Oct 2015.

[26] W. Jiang and Z. Yin, “Human activity recognition using wearable sensors by deep convolutional neural networks,” in *Proceedings of the 23rd ACM International Conference on Multimedia*, MM ’15, (New York, NY, USA), pp. 1307–1310, ACM, 2015.

[27] L. Xue, S. Xiandong, N. Lanshun, L. Jiazhen, D. Renjie, Z. Dechen, and C. Dianhui, “Understanding and Improving Deep Neural Network for Activity Recognition,” *ArXiv e-prints*, May 2018.

[28] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, “Stacked convolutional auto-encoders for hierarchical feature extraction,” in *Artificial Neural Networks and Machine Learning – ICANN 2011* (T. Honkela, W. Duch, M. Girolami, and S. Kaski, eds.), (Berlin, Heidelberg), pp. 52–59, Springer Berlin Heidelberg, 2011.

[29] B. Almaslukh, “An effective deep autoencoder approach for online smartphone-based human activity recognition,” vol. 17, 04 2017.

[30] F. Gu, K. Khoshelham, S. Valaee, J. Shang, and R. Zhang, “Locomotion activity recognition using stacked denoising autoencoders,” vol. 5, pp. 2085 – 2093, 04 2018.

[31] Y. Guan and T. Ploetz, “Ensembles of Deep LSTM Learners for Activity Recognition using Wearables,” *ArXiv e-prints*, Mar. 2017.

[32] M. Inoue, S. Inoue, and T. Nishida, “Deep Recurrent Neural Network for Mobile Human Activity Recognition with High Throughput,” *ArXiv e-prints*, Nov. 2016.

[33] N. Y. Hammerla, S. Halloran, and T. Ploetz, “Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables,” *ArXiv e-prints*, Apr. 2016.

[34] C. Liu, L. Zhang, Z. Liu, K. Liu, X. Li, and Y. Liu, “Lasagna: Towards deep hierarchical understanding and searching over mobile sensing data,” in *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking*, MobiCom ’16, (New York, NY, USA), pp. 334–347, ACM, 2016.

[35] M. Robnik-Šikonja and I. Kononenko, “Theoretical and empirical analysis of relieff and rrelieff,” *Machine Learning*, vol. 53, pp. 23–69, Oct 2003.

[36] N. R. M. Paulich, M. Schepers and G. Bellusci, *Xsens MTw Awinda: Miniature Wireless Inertial-Magnetic Motion Tracker for Highly Accurate 3D Kinematic Applications*. XSENS TECHNOLOGIES B.V, https://www.xsens.com/download/pdf/MTwAwinda˙WhitePaper.pdf.

[37] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *CoRR*, vol. abs/1412.6980, 2014.

[38] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” in *IN NIPS*, MIT Press, 2007.

[39] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,” in *A Field Guide to Dynamical Recurrent Neural Networks* (S. C. Kremer and J. F. Kolen, eds.), IEEE Press, 2001.

[40] H. He, Y. Bai, E. A. Garcia, and S. Li, “Adasyn: Adaptive synthetic sampling approach for imbalanced learning,” in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 1322–1328, June 2008.

[41] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *ArXiv e-prints*, June 2011.

[42] H. Gjoreski, S. Kozina, M. LuÅ¡trek, and M. Gams, “Using multiple contexts to distinguish standing from sitting with a single accelerometer,” in *European Conference on Artificial Intelligence (ECAI*, 2014.