Convolutional Neural Networks VS Autoencoders for Automatic Feature Extraction in HAR

Natascia Caria[†], Claudia Cozzolino[‡]

***Abstract*—Human body motion analysis based on sensors is a trending topic in recent years and it is receiving increasing attention due to its role in several and different applications ranging from sport or domotic to health and security. Nevertheless, processing time-series data from wearable inertial measurement units (IMUs) still poses challenges in the representation learning, a fundamental step for Machine Learning algorithms generalization performances. In this paper several Deep Learning architectures are proposed to automatically extract discriminative features from raw signal, avoiding exhausting and tedious manual engineering. For the activity recognition task proposed in [1], a deep Convolutional Neural Network (CNN) and a recurrent version with a Gated Recurrent Unit (GRU) layer are tested versus a Denoising Autoencoder (AE) and a Variational AE stacked with a Shallow Classifier (SC). The CNNs are able to achieve F1-scores higher than 0.98 in the majority of the classes, while the AEs have proven to be capable of comparable results with the advantage of a shorter prediction time, a fundamental aspect for real-time applications.**

***Index Terms*—Activity Recognition, Deep Learning, Convolutional Neural Networks, Recurrent Neural Networks, Denoising Autoencoder, Variational Autoencoder**

I. Introduction

Automatic Human Activity Recognition (HAR) from time-series sensor data is a growing area of research. The commercial spread of competitive and inexpensive Inertial Measurement Units (IMU) and sensor equipped smartphones is leading to a slow and progressive decay of vision based HAR. As a matter of fact, the sensor approach with wearable devices presents a multitude of benefits like non-intrusiveness, portability and high processing power. IMUs are widely used for complex motion analysis applications in a wide range of fields as sports, health care, video-games, smart homes and security.

The current challenge in HAR is then to provide methods and model architectures to robustly extract and analyse features from time-series data independently from subjects, references systems and possibly from the acquisition device. This is a fundamental step in order to build accurate classifier and implement commercial application, but often requires long preliminary studies in the task of interest context. In [1] for example, extensive human evaluation of the signals has been performed to manually define features for activity classification.

[†] Mat. n°: 1225874, email: natascia.caria@studenti.unipd.it
[‡] Mat. n°: 1227998, email: claudia.cozzolino@studenti.unipd.it
Department of Mathematics, University of Padova

The purpose of this work is to present and test strategies for the same problem, but, on the contrary, automatically extracting features directly from raw data. Guided from [2] we supposed that *ad-hoc* deep learning architectures instead of feature engineering could lead to better representation learning and then to better performances. In detail, the classification goal consists in recognising the activity of a subject among the following seven: "RUNNING", "STANDING", "LYING", "JUMPING", "WALKING", "SITTING", "FALLING", given the current spatial signals from accelerometer, gyroscope and magnetometer. In this paper four different models have been proposed to accomplish it, trying to take into account both predicting time and accuracy. We firstly started with a 2D Convolutional Neural Network (CNN) and then created an enriched version (CRNN) adding a Gated Recurrent Units (GRU) layer. In addition we wanted to experiment Autoencoder (AE) capabilities, already established in the vision domain, also in HAR setting. In particular, two AEs architectures have been compared: simple, but effective well known Denoising (DenAE) versus Variational (VAE), a more recent class belonging to Differentiable Generetor Networks. In both the cases, the pretrained encoder part was stacked with a Shallow Classifier network (SC) to conclude the predicting task. All the methods have been then tested on a standardized and on a Discrete Cosine Transform (DCT) versions of the dataset.

Before moving on to the technical discussion, the structure of the report is presented: in Section II we describe the state of the art in HAR in particular with respect to automatic feature extraction. An high level description of all the experimental process is firstly introduced in Section III and then detailed reported in Section IV, regarding the signal data preparation, and in Section V for the deep learning architectures. Finally, performances evaluation and results are presented in Section VI, while concluding remarks are discussed in Section VII.

II. Related Work

Activity recognition from time-series sensory data has a well established pipeline involving sliding window segmentation, feature extraction and finally classification. The traditional approach extensively explored hand-crafted features considering for example norm and combinations of signals [1], statistics values as mean, quantiles, correlation [3] or biomechanical attributes [4]. Despite these strategies have successfully acquired satisfying results for specific recognition tasks fully exploiting the domain knowledge and are compu-

tationally lighter, they present several drawbacks. As pointed out in [5], their generalization performance is too excessively limited by reliance on domain expert and experimental design such as the position and the number of sensors or the subjects characteristics (e.g. adults, hospitalized, parkinsonian patients, etc). An additional crucial aspect is that feature engineering requires very long and laborious procedures by researchers with the risk that the designed features might not be best for the objective at hand. Given all the previous motivations and the emerging of Deep Learning (DL) paradigms, HAR, like various research areas, is recently switching to automated end-to-end feature extraction. As a matter of fact, DL methods have boosted the state-of-the-art in many tasks [6] thanks to its innate abilities to provide effective data representation, fundamental for a successful learning [2]. However, while for the Visual Object and the Speech Recognition field the feature learning evolution has been traced in the literature as described in [7] and [8], for the HAR standards common rules still have to be defined. The most adopted techniques for effective high-level features are Convolutional Neural Networks (CNN). In this framework, each window segment is considered as an image and features are extracted by mean of convolutional and pooling stages [9] possibly exploiting more than one channel to split the signal along the tridimensional axes or by sensor type [10] [11]. Regarding Recurrent Neural Networks (RNN), several experiments have been conducted to test their capacity of explicitly modeling the temporal dynamics on activities recording sequences. In practice Long Short Term Memory (LSTM) layer is often preferred [12], eventually coupled with covolution [13] showing state-of-the-art recognition performance. A more recent trending topic in representation learning are Autoencoders (AE), which can be thought as supervised methods for unsupervised learning. Their ability to provide robust and compressed latent representation of the input data is very appealing for HAR, and more in general for sensor based applications, since it allows to reduce both energy and memory consumption maintaining comparable performances [14]. Many proposals have been published in the last years considering both shallow, deep and stacked variant of Denoising and Sparse AE, demonstrating their effectiveness [15] [16] [17] [18]. Finally, in this *excursus* about representation learning, it is worth mentioning generative models like Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs). Although these represent a significant evolution in the way of extracting features by learning the probability distribution of the data, still few experiences have been made in activity recognition problems. The majority of them exploits these modern tool for complex body position reconstruction with several wireless IMUs embedded in clothes suit [19] [20] or to perform reliable data augmentation overcoming the shortage of labelled data [21]. However, in the very recent [22], authors describe how VAEs can learn smooth representation improving Signal-to-Noise Ratio, demonstrating that HAR can be achieved in real-time even with a single IMU device like smartphone. To conclude, this work aims to give a definitive unifying overview

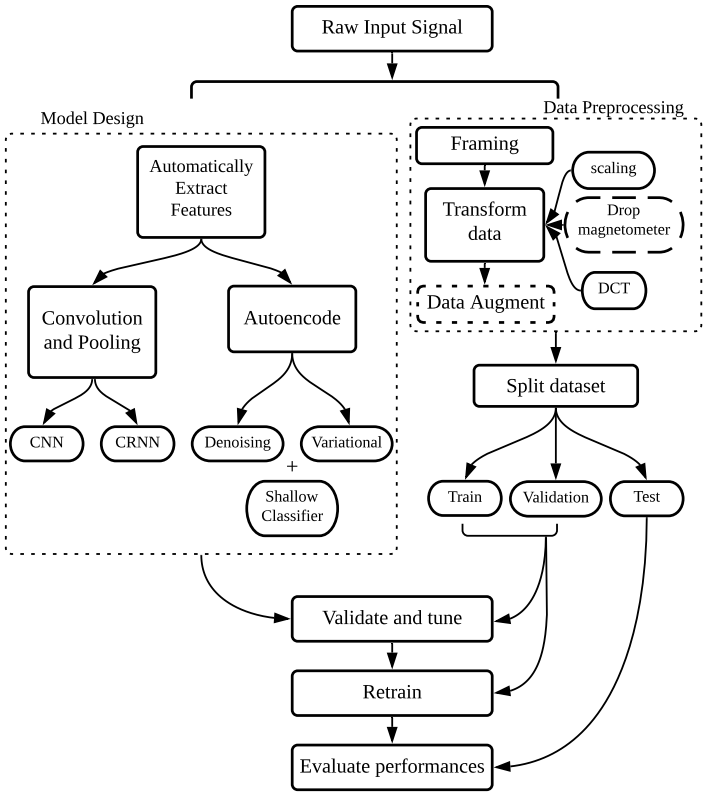

Fig. 1: Complete pipeline chart showing the theoretical and practical process and elaboration conducted during the study.

fair comparing in the same experimental setting CNN, GRU based CRNN, Denoing and Variational AEs, highlighting their strengths and weaknesses for the single IMU activity recognition task, considering efficiency in terms of time and precision.

III. Processing Pipeline

The whole study is conducted using the dataset (available here) collected by the German Aerospace Center (DLR), which is composed of more then four hours of labeled activities. In detail, it records 3D measurements for accelerometer, gyroscope and magnetometer sensors in a time-series format. For this reason, the first data manipulation that has been carried out is framing: fixed the window size w and the stride step o, the time signal is divided in frames of length w consecutively overlapped of o units. Moreover, before or after this stage many transformation could be applied in order to improve numerical stability, accelerate computations or amplify the information signal. Below, the functions and techniques that has been taken into account here during data preprocessing are listed:

- scaling: a common practice in Machine Learning to normalize the range of independent variables. In this study several possibilities have been considered and compared like min-max normalization, unit Euclidean length scaling and standardization, which turned out to

be the best.

- magnetometer data dropping: since in some of the previous works in the literature cited above, authors have shown that it is possible to achieve good results in HAR only using accelerometer and gyroscope, we also wanted to investigate how much the magnetometer data increase the expressiveness of the data by trying both to maintain and remove them.
- Discrete Cosine Transform (DCT): a widely used transformation technique in signal processing and data compression, which expresses the signal image in terms of a sum of cosine functions oscillating at different frequencies.

Subsequently, analyzing the data in detail, we observed that the cardinality of the various classes were not balanced. In this regard, some data augmentation strategies have been experimented on the less represented categories like "JUMPING" and "FALLING".

At this point, features extraction and engineering crafting is commonly performed in order to obtain the optimal and most significative attributes to tackle the task of interest. Nevertheless, since the aim of this work is to experiment with strategies for automatic representation learning, this phase has been moved in the model design part. The dataset has been then straightly shuffled and randomly split in train, validation and test sets.

Regarding the models, deep learning architectures have been exploited in order to automatically extract high-level representation. The designed methods can be mainly divided in two classes according the feature learning approach:

- **Convolutional Neural Networks:** the mathematical operation of convolution applies a set of locally shared filters (or kernels) to obtain the most representative features. The combination with a max-pooling stage then produce a subsampling downsizing the frame and introducing translational invariance to the system, which improves its robustness. Technically speaking, we experimented a network with three stacked convolutional layer with increasing number of filters alternating with pooling and Rectified Linear Unit (ReLU) activation function. After that, a fully connected layer has been applied combining all features' maps previously obtained and a seven neurons Softmax layer concludes the task computing the probability of each activity class.
  A recurrent variant has been also proposed inserting just before the last dense connections a GRU layer. The Gated Recurrent Units proposed by Cho *et al.* in [23] aims to simplify the complex Long Short Term Memory (LSTM) maintaining its quality for long-term temporal dependencies avoiding vanishing or exploding gradient problems. Here it has been put to test to try to provide a better representation of data by explicitly exploiting the time domain.
- **Autoencoders:** a compressed, sparse or smoothed version of the data, called code, is produced learning the identity function on the input space. However, since the real intent is not to learn the trivial identity map, but to produce a robust hidden representation or to capture interesting properties of the input manifold, the AE's capabilities are properly limited with architectural constraints or with regularization techniques. The first model that has been studied from this class here is a Denoising AE. This type of overcomplete AE is fit to clean a corrupted version of the input optimizing a reconstruction criterion. As said above, it is important to note that the goal is not the task of denoising *per se*, but rather denoising is investigated as a method for learning to extract useful features. In practice, the main building blocks can be separated into two parts, an encoder and a decoder, which here have been defined with three flat dense layers to each. While the former has decreasing dimensions, the latter increases the size starting from the compressed code until it reaches the input size.
  An additional AE architecture that has been implemented in our comparison study for automatic feature learning with DL is Variational Autoencoder. VAEs are generative models that produce output data sampling from probability distributions given a latent variables z. Differently from GANs, VAEs exploit differentiable generator networks to perform explicit density estimation of the input data. In our case the encoder learn the mean and the covariance matrix of the hidden variable z with a network composed by two convolutional and one dense layers. On the other hand, the decoder, once received a sampling of z performs the inverse operation to return an example that could reasonably belong to the distribution of modeled data.
  As expected, the AEs alone are not enough to complete the activity recognition task. Therefore a Shallow Classifier network has been defined to conclude the prediction process. Strictly speaking, two fully connected dense layers receive in input the compressed code learnt by the encoder part, which has been previously pre-trained, and output a seven neurons long Softmax layer similarly to the CNN. The training of this new ensemble network is then firstly performed only on the SC, freezing the encoder weights, and suddenly on all its part to fine tune the model and improve the overall performances.

For all the described networks, the Adam optimizer has been adopted to perform the training, while regularization techniques such as early stopping and L_2 norm penalty are used at different stages in order to avoid overfitting to the training data.

Just after the end of the model design phase, all the architectures and choices about framing and data preprocessing have been validated. At the same time, all the hyper-parameters like number of epochs, layer units and batch size have been tuned manually or by means of a greedy search.

Finally, once a stable and satisfactory combination has been reached, the models have been definitively retrained on the train jointly with the validation set and subsequently they have

been evaluated considering the unseen samples in the test set.

The flow-chart in Figure 1 depicts the whole process described in this section.

IV. Signals and Features

The original dataset consists of time-series measurements from the accelerometers, gyroscopes and magnetometers of an IMU sensor (an Xsens MTx-28A53G25) with respect to the sensor frame (SF); moreover the direction cosine matrix extracted from the sensor is provided in order to rotate the features to the global frame (GF). In total it contains about 4.5 hours of manually labelled activities collected from 16 male and female subjects aged between 23 and 50. Each sample of the dataset is associated with an activity label among "walking up", "walking down" and "walking" (here grouped in "WALKING"), "jumping forward", "jumping backward", "jumping vertical" (grouped in "JUMPING"), "RUNNING", "STANDING", "LYING", "SITTING", "FALLING" or "TRANSITION" (here discarded).

As a first approach in this work, acceleration, angular velocity and magnetic field measures along the x, y and z axis related to the SF have been selected to be processed. The first preprocessing step on raw data has been framing by window: fixed the window-size w and an overlapping value o a time-serie signal is divided in consecutive frames of length w with overlap of o samples. The window-size value w has to be chosen on the basis of the sample frequency of the sensor (100Hz in this case), and considering that it should be large enough to capture activities peculiarities and small enough to be within the minimum activity duration. Based on these considerations, the value $w = 128$ has been chosen, which is slightly more than one second per frame. For what concerns the overlapping value o, large values have been discarded in order to avoid redundant data; after testing different values, $o = 8$ has been selected. After the framing phase, the new dataset consists of frames of size 128×9.

At this point, two different strategies have been adopted to generate the following two datasets, one based on time domain features and the other based on frequency domain ones:

1) *RawSignal* dataset: keeping frames with raw signals and applying scaling on each axis.
2) *DCT* dataset: applying the Discrete Cosine Transform of type II (DCT-II) and scaling on each axis, where DCT-II of a datasequence $x = (x_1, \ldots, x_N)$ is defined as follows:

$$y_k = 2 \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi k(2n+1)}{2N}\right)$$

In the Machine Learning background, DCT has been successfully used in wide range of applications such as image compression, high-speed face recognition [24] and in signal processing, involving the HAR framework [25]. The DCT, and in particular the DCT-II, is so popular in signal processing for its compression properties: most of the signal information tends to be concentrated in a few low-frequency components of the DCT, thus it is possible to discard the higher frequency components to reduce data size without loss of useful information. This is what is done in [25], where the most significant low-frequency components are further selected with PCA. In this work, the extraction of the main components is completely left to deep learning architectures. Thus, it is not necessary to adapt the model structures to the size of the inputs and this allows to have a fair comparison of their performance on the *RawSignal* and *DCT* dataset.

Different scaling approaches have been tested and compared, including min-max normalization, unit Euclidean length scaling and standardization with zero mean and unitary variance; the latter has been found to be the most effective in terms of models performance. In addition, on both feature sets, the option of not considering the magnetic field measures has been evaluated: the loss of accuracy of the models was such that it was decided to maintain the data related to all the sensors.

Both the *RawSignal* and the *DCT* dataset have been divided into 64% training, 16% validation and 20% test set, maintaining the class proportion of the original dataset. The number of frames per activity in the partitions is shown in Table 1.

In order to address the issue of class imbalance, different data-augmentation strategies have been considered. Indeed, numerous studies have revealed the importance of data augmentation to improve the performances in machine-learning tasks and recently attention has been focused on the specific case of time-series data [26]. In particular, for the HAR problem, *ad-hoc* data augmentation techniques for wearable sensors must be employed to ensure that data label semantics are maintained [12], [27]. In this work, the following techniques have been applied on the training set of the *RawSignal* dataset:

- rotation: applying arbitrary rotations with respect to each axis, which can be interpreted as several sensor placement scenarios;
- permutation: a single window is first sliced into same length segments that are randomly permuted to form a new window, as a way to randomly perturb the temporal location of within-window events;
- noise injection: adding to the sensor data random noise sampled from the normal standard distribution in order to increase models robustness.

Starting from the *RawSignal* frames, different combinations of techniques have been used to generate new artificial training data for the less represented classes, until reaching for each of them at least 50% of the most frequent class. However, none of the techniques applied has been found to be effective on the models discussed in this project.

V. Learning Framework

The Deep Learning models designed in this work have been implemented in `Keras`, an open-source Python library to build neural networks that runs on top of `Tensorflow`. The working environment has been Colaboratory, or "Colab" for short, a product from Google Research well suited to

	Standing	Sitting	Falling	Lying	Jumping	Walking	Running
Training	3742	1866	43	885	203	2253	468
Validation	936	466	11	221	51	563	117
Test	1170	583	14	276	64	704	146

TABLE 1: Number of frames per activity on training, validation and test set

machine learning, which provides free access to accelerated computing resources (see here for more). While the training has been executed on its GPU, perfectly integrated to speed up `Tensorflow` operations (> 30 times faster), the evaluation phase instead, has been conducted on basic CPU to better mimic the timings results of standard devices.

The following sections describe in detail the topology and the main parameters involved in the four machine learning models proposed here to solve the HAR task. For all cases, before being fed to the networks, the input data was reshaped as tensor adding a third dimension for compatibility reasons with the GPU backend of Keras. Moreover, note that all models have been trained using mini-batch gradient descent in order to have a as fast and reliable as possible learning. Regarding the weights update, the adaptive momentum based Adam optimizer [28] has been adopted calibrating the rate between 0.001 and 0.008 observing the convergence results on the training set. The weights initialization has been instead done randomly, using the Xavier-Glorot uniform distribution for the parameters and zero for the biases, default kernel initializers in `Keras`. Moreover, the Early Stopping technique has been exploitend during the validation phase to prevent overfitting.

Finally, all hyperparameters have been tuned either via an exhaustive grid search or by hand, validating the value choices with a 5-folds cross-validation on all the samples except the test set.

A. Convolutional Neural Network

This is the first architectures explored in this paper and involves both temporal and spatial signals convolutions, the topology of the model is graphically described in Figure 2. The 2D frame of size 128×9 is processed following the steps listed below:

- 3 2D Convolutional layers characterized by 5×5 locally shared kernels, stride step set to 1 and zero padding to keep the dimension equal. The number of filters is double at each layer starting from 64. Moreover, weight decay is applied to add to the training loss a penalty proportional to $L_2(\theta) = ||\theta||_2$, i.e. the euclidean norm of the weights, as regularizer.
- each convolutional layer terminates applying the Rectified Linear Unit activation function defined as

$$\text{ReLU}(x) = \max\{0, x\}$$

  and is interspersed with a pooling stage. In particular the Max-pooling is iteratively applied on 2×2 blocks on the passed frames to halve their size while extracting significant and robust information. This step it also has the advantage of making the network approximately invariant to small variations like noise or translation.
- a fully connected dense layer with 512 units receives then the flattened results of previous operation, again the ReLU function is chosen as activation.
- to conclude the predicting process, a fully connected dense layer of 7 units, jointly with a Softmax activation is used. Note that since it is defined as

$$\sigma(\boldsymbol{z})_j = \frac{\exp(z_j)}{\sum_k^K \exp(z_k)},$$

  where K is the number of classes, 7 in our case, the output is a vector of probability where each entry represents how likely the input belongs to the correspondent class.

Finally, note that considered the output type and the multi-class classification task, the network is trained optimizing the Sparse Categorical Crossentropy defined as

$$L(\theta) = -\frac{1}{N} \sum_{x \in Tr} \sum_{k \in K} \mathbf{1}_{x \in k} \log(P_\theta(x \in k)),$$

where Tr is the training, N its cardinality and P_θ is the probability distribution, parametrized in the model weights θ, that a sample x belongs to class k. This is the Categorical Crossentropy version for mutually exclusive classes prediction, which allows to conserve time and memory avoiding the useless summation over all the contributions.

B. Convolutional Recurrent Neural Network

The second proposed model is almost entirely identical to the previous one. The only and decisive change consists in having inserted a recurrent layer just after the flattening, while the rest of the structure choices and training procedures remains unaltered.

Technically speaking, a Gated Recurrent Units with 32 units (the green dashed rectangle in Fig. 2) has been added to enrich the CNN with more dynamics in order to enforce the temporal sequential nature of the human activity time-series signal. Although the GRU is simpler than the long short-term memory (LSTM), its performance on certain tasks as speech signal modeling and natural language processing was found to be similar to that of LSTM [29], albeit maintaining a faster trainability thanks to its fewer parameters. In detail, while the LSTM exploits a memory cell and three gates, the GRU controls the information flows via a single gating unit. This simultaneously monitors the forgetting factor and the update decision, which in practice are implemented as continuous $0-1$ gates. The recurrent hidden state representation for every

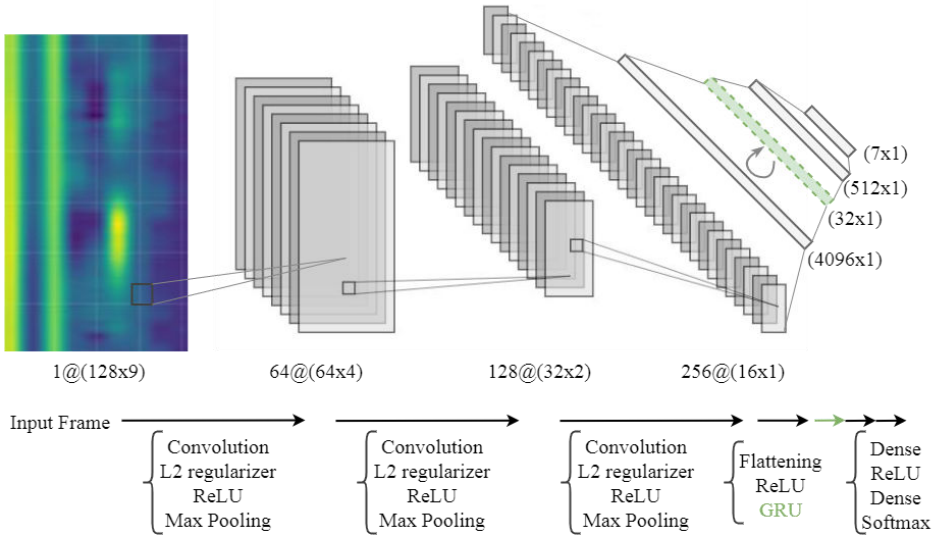

Fig. 2: CNN and CRNN models architecture pipeline with kernel size = 5, pooling size = 2 and increasing number of filters. The green layer corresponding to GRU is only exploited in the recurrent version.

time step t can be then formalized as

$$\boldsymbol{h}^{(t)} = z^{(t)} \odot \boldsymbol{h}^{(t-1)} + (1 - z^{(t)}) \odot f(\boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{W}(r^{(t)} \odot \boldsymbol{h}^{(t-1)})),$$

where $\boldsymbol{x}^{(t)}$ is the input of the recurrent layer at time t, $\boldsymbol{U}$ is the input to hidden connection weights matrix, $\boldsymbol{W}$ is the hidden to hidden connection weights matrix and $z, r \in [0, 1]$ are respectively the update and reset gate factors. Regarding f, it is the activation function, usually both $\tanh$, the hyperbolic tangent, and Sigmoid, which is defined as

$$s(x) = \frac{1}{1 + \exp(-x)},$$

are possible. In `Keras` implementation the GRU layer has by default the former as output activation and the latter as recurrent activation. Moreover, from the expression of the recurrent state, it can be noticed that z selects whether the hidden state is to be be update with the new one, while r decides if the previous hidden state is to be ignored; as expected, the combination $z = 0$ and $r = 1$ gives indeed the classic update of the simple recurrent layer.

C. Denoising Autoencoder

The idea behind the third model is radically different from those of the above. The convolutional operations are abandoned to switch to a simpler but very effective architecture: Denoising Autoencoder (DenAE). As a matter of fact, the prediction task is temporarily suspended to firstly learn a good and compressed representation. To this end, the DenAE is trained to denoise a corrupted version of the input data. In practice the model pipeline is composed by the following steps:

- first inject random noise summing a 128×9 matrix whose elements are sampled from the standard Gaussian and then multiplied by a factor of $\nu = 0.6$.
- after a Flattening operation the vector data is passed to a series of dense fully connected layers respectively with sizes 512, 256 and 128. All of them exploits the ReLU activation and the L_2 norm penalty similarly to the CNN. The produced output is the code, i.e. the compressed representation and concludes the encoder part.
- to obtain the signal frame back, the decoder is defined with a symmetric structure with the only exception that the last dense layer as a linear activation. The motivation to this choice is that reconstructing the initial image is like a regression task since the signal values vary in a continuous range.

The structure above described is depicted in detail in Figure 3, where a graphical example of the noisy versus restored frame is also shown.

In conclusion, as explained above, given the real output, the AE network is trained element-wisely minimizing the Mean Squared Error (MSE) loss defined as

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{x}_i)^2,$$

where the x_i's are the *pixels* of the original *images* and $\hat{x}_i$'s those of the denoised output of the respectively corrupted version.

D. Variational Autoencoder

The latest created model exploits all the ideas of the previous ones, but in a different fashion. The fourth way to automatically extract reliable features tries to explicitly estimate the underlying distribution of the data. A 2D Variational Autoencoder (VAE), as stochastic generative model, has been then experimented. In general a VAE is a model that produces

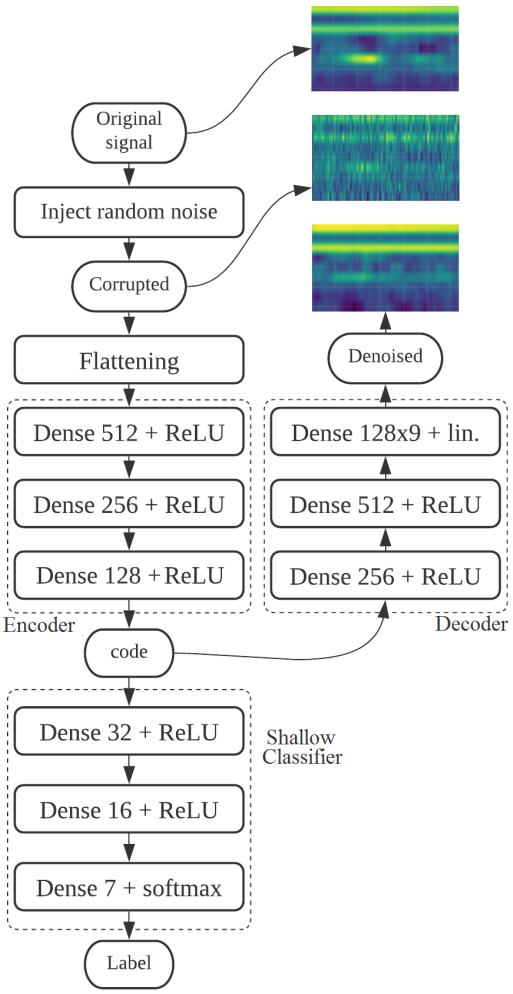

Fig. 3: Predicting process via the Denoising Autoencoder architectures and the Shallow Classifier. In the decoder chart, "lin." is the abbreviation of linear.

output data which can be generated by a latent variables $\boldsymbol{z}$. In practice, a simple prior distribution $p_\theta(\boldsymbol{z})$ is picked, e.g. the Gaussian, and a generator network, which in fact constitutes the decoder, is used to compute the posterior $p_\theta(\boldsymbol{x}|\boldsymbol{z})$. To estimate the best parameters θ, the likelihood

$$p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{z}) p_\theta(\boldsymbol{x}|\boldsymbol{z}) dz$$

should be maximized. However, that computation is intractable determining also the intractability of the conditioned $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. An encoder part is therefore necessary to approximate the latter. In practice this is done exploiting the Evidence Lower Bound (ELBO), a variational approach, which gives the name to the architecture.

In our work the VAE has been implemented with the operation described below, the complete pipeline is also illustrated in Figure 4.

- two 2D Convolutional layers with 5×5 kernels, stride = 2, zero padding and respectively 32 and 64 filters. As usual the ReLU activation is adopted.
- a flattening and fully connected dense layer of 16 units is consequently applied.
- to conclude the encoder part and produce the latent variable, firstly two dense layer of size equal to that of $\boldsymbol{z}$ and ReLU are used to produce an estimate of its mean $E(\boldsymbol{z})$ and of its *variance*. In practice the logarithmic transformation $\log(Var(\boldsymbol{z}))$ is preferred instead for numerical stability reasons. Finally, $\boldsymbol{z}$ is computed as follows

$$\boldsymbol{z} = E(\boldsymbol{z}) + \epsilon \odot \exp\left(\frac{1}{2}\log(Var(\boldsymbol{z}))\right)$$

where ϵ is a probability vector with the same dimension of $\boldsymbol{z}$ sampled from the multivariate standard normal. This technique is called reparametrization trick and allows to implement sampling from a Gaussian distribution with custom mean and standard deviation, while keeping trainability via gradient descent method.
- as for the Denoising AE, the decoder part is defined in the opposite way of the encoder in order to return back to

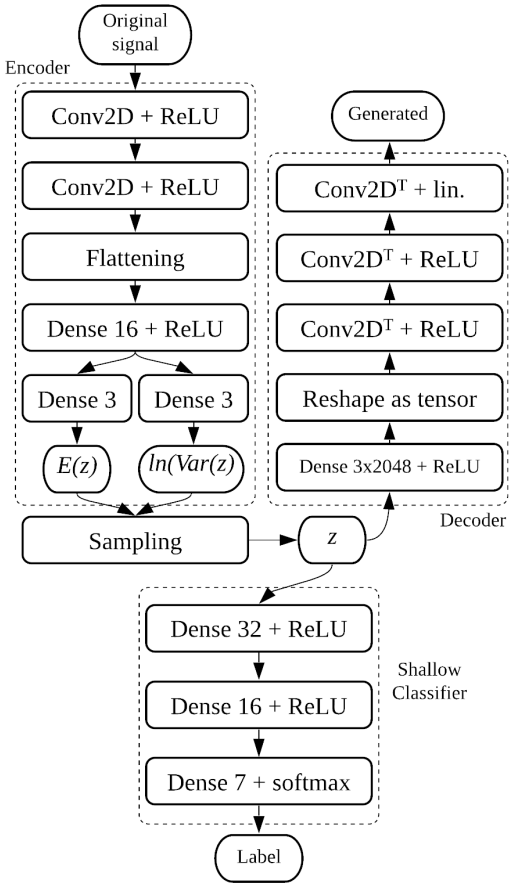

Fig. 4: Predicting process via the Variational Autoencoder architectures and the Shallow Classifier. In the encoder part, $E(\boldsymbol{z})$ stands for mean of $\boldsymbol{z}$ and $Var(\boldsymbol{z})$ for its covariance matrix. In the decoder chart, "lin." is the abbreviation of linear.

the original frame format. In this case, after a dense layer and a reshaping, three 2D transpose convolutions, also called deconvolutions, are applied. While the first two have the same kernel size, stride and reversed number of filters of the encoder convolutions to revert their effect, the last one has a single 3×3 filter. Padding is also applied to ensure the right input dimensions are restored. Concerning the activation functions, as usual the ReLU is used for the hidden layers and the liner for the output.

The VAE is trained optimizing a custom loss which takes in consideration both a *image* reconstruction error expressed by means of the MSE, as in the DenAE, and the ELBO constraints. The minimized objective hence results in a weighted sum of the MSE and a reguralization loss. Note that the reconstruction loss forces the decoded samples to match the original inputs, while the regularization loss, enforced via D_{KL}, the Kullback-Liebler Divergence, helps to learn well-formed latent spaces and reduces overfitting to the training data. Strictly speaking, the D_{KL} as a measure of dissimilarity between probability distributions, forces the encoder to have a specific distribution; usually the Gaussian prior is preferred since it has closed-form solutions. A final consideration should be done on the latent dimension, i.e. the size of $\boldsymbol{z}$. This is an hyperparameter to validate, attempting for a good trade-off between representation and good compression. In our case a latent dimension of 3 turned out to be optimal, an example of the 3D latent space learnt by the VAE is shown in Figure 5, where its effectiveness can be deducted by the fine clustering capabilities of the different activities signals.

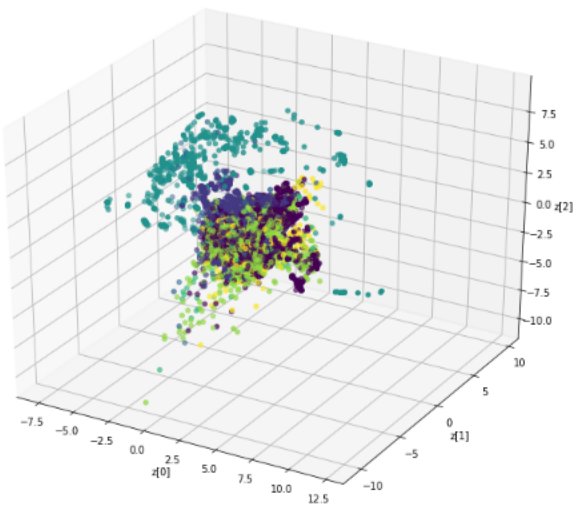

Fig. 5: Example of 3D latent space learnt by the Variational Autoencoder. Each color represent a different activity class. As expected, exploiting a Gaussian prior, the majority of points tend to be close to the origin. Nevertheless, it can be seen that the classes are almost always well clustered.

E. Shallow Classifier

Since the AEs are not sufficient to bring to fruition the activity prediction task, a Shallow Classifier (SC) has been implemented. Its task is to take in input the code obtained from the encoder (built as according to one of the two versions proposed before) and to return the corresponding class label. The composition of this block is really minimal: two fully connected dense layers of sizes 32 and 16 units follow each other before the usual final seven long predicting layer. As for the model described above ReLU and weight decay regularization have been adopted for the hidden layers, while Softmax activation for the output one. The classification power of this very simple model relies in its training strategy and in the encoder ability to provide a code that is as compressed and representative as possible. To this end, the two blocks, the encoder and the SC, are firstly trained separately. As a matter of fact, the encoder weights are learned during the AE's training, after that they are temporarily freezed in order to train the SC as a small network from the hidden code to the respectively class. As for the CNN and CRNN train to predict means in practice to minimize the Sparse Categorical Crossentropy. Finally the pretrained parts are joint together and the performances are boosted fine tuning the obtained larger model training on all its weights.

VI. Results

In this section the numerical results related to all the architectures on both *RawSignal* and *DCT* dataset are provided. Bearing in mind the unbalance of the classes and aiming at finding a good trade-off between Precision and Recall, the performance of the models are evaluated in terms of the F1-score, which is defined as

$$\text{F1-score} = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad .$$

A. Convolutional Neural Networks results

As far as Convolutional Neural Networks are concerned, the hyper-parameters defining the models have been rigorously tuned with a grid search procedure with 5-fold cross-validation, performed on the union of the training and validation set, which together constitute 80% of the data. The process led to the optimal parameters as described in the previous section; the optimal models have been finally tested in the remaining 20% of the dataset and related F1-scores are given in Tables 2 and 3, which also provide the models average prediction time per sample.

Comparing the two architectures, which only differ for the presence of a recursive layer, it is possible to observe that, both on *RawSignal* and *DCT* dataset, the CNN model outperforms the CRNN in the prediction of "FALLING" and "JUMPING" activities. For the other classes the results are instead comparable, with slight benefits of the simpler network. In general, the low performance of both models in the prediction of "FALLING" and "JUMPING" can be related to the problem of class imbalance, since the available data are considerably less than those of the most represented activities, in this case "STANDING", "WALKING" and "SITTING". Furthermore, one would have expected CRNN to have better

	CNN	CRNN	DnAE	VAE
Standing	0.99	0.98	0.98	0.98
Sitting	0.99	0.98	0.98	0.98
Falling	0.88	0.60	0.75	0.67
Lying	1.00	0.99	1.00	0.99
Jumping	0.89	0.82	0.76	0.82
Walking	0.98	0.98	0.95	0.98
Running	1.00	0.98	0.96	0.95
Pred Time	3e-03	3e-03	8e-05	3e-04

TABLE 2: F1-scores on *RawSignal* test set and prediction time per sample.

	CNN	CRNN	DnAE	VAE
Standing	0.98	0.98	0.96	0.97
Sitting	0.97	0.98	0.94	0.95
Falling	0.75	0.44	0.64	0.61
Lying	0.99	1.00	0.99	0.98
Jumping	0.73	0.69	0.33	0.62
Walking	0.97	0.97	0.93	0.95
Running	0.96	0.96	0.87	0.94
Pred Time	3e-03	3e-03	8e-05	3e-04

TABLE 3: F1-scores on *DCT* test set and prediction time per sample.

results than CNN, at least on the time-domain dataset. This probably does not happen because the sequential structure of the data is somehow lost in the automatic feature extraction made by the convolutional layers. A possible alternative not tested here could be to anticipate the recurrent layer. The best performances on the *RawSignal* dataset may be attributed to the fact that it would have been necessary a selection of DCT components before feeding the network. As already mentioned in Section IV, information tends to be concentrated in its few low-frequency components. Thus, discarding the higher frequency components could have better driven the automatic feature extraction, here completely entrusted by convolutional layers.

B. Autoencoders results

For what concerns the Autoencoders architectures, it has been preferred to manually choose the hyperparameters, albeit carefully, due to the difficulties in implementing an automatic folded grid search able to take into account the several compilation steps. The models tuning has been carried again by cross-validating on the union of the training and validation set. Once best models were defined, their performances have been evaluated in the usual test set (20%).

Looking at Tables 2 and 3, it is possible to see that the two models achieve similar performances both on time and frequency domain based datasets, even if exploiting very different structures. Also in this case, the problem of poor prediction capabilities in small classes remains; here it is even more pronounced in the *DCT* dataset, for which the same considerations made above hold. Nevertheless, for the majority of the activities, the F1-scores are competitive with respect to the CNN and CRNN ones, with the advantage of being more than 10 time faster. As a matter of fact, given the choice of the hyperparameters and of the layers type, the predictive process of the AEs requires less operations than the CNNs. This aspect could be very appealing in the context of real-time human activity recognition on devices with tight battery and CPU constraints (e.g. smartphones). Before concluding, it could be interesting to report some observations about the AE's behaviour in relation with the encoding dimension hyperparameters, i.e the size of the compressed code for the DenAE and of the latent variable for the VAE. During their tuning on the validation set, we noticed that although augmenting them, a correspondent improvements of the performance does not append, while, as expected, a decrease causes a worsening. We hence felt that those values were the optimal since they proved to be the sufficient but necessary sizes for a good compressed encoding.

C. Final considerations

Considering the overall results, the best model turn out to be the CNN on the *RawSignal* version of the dataset. The F1-score is greater or equal than 0.98 for all the classes except the less represented "FALLING" and "JUMPING", which respectively achieve 0.88 and 0.89 score. By observing the confusion matrix in Figure 6 we can conclude that this poorer results are to be addressed to a problem of Recall rather than Precision. Indeed, these activities are often confused with the most frequent classes; this behaviour is in general shared among all the methods. After having tried to solve the problem with *ad-hoc* data augmentation strategies for sensor frame signals and having failed, we supposed that it could be a problem not only related to a lack of samples, but caused by the specific kind of movements. Computing

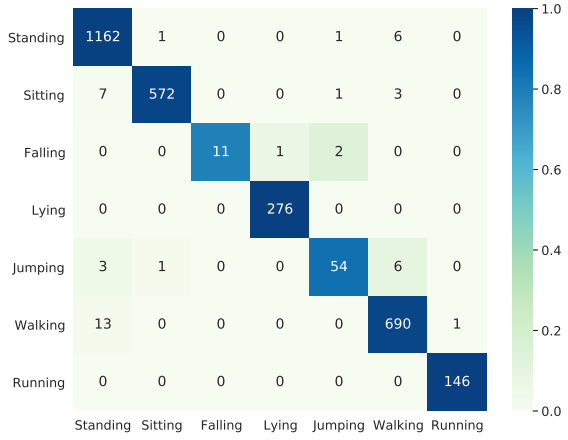

Fig. 6: Confusion Matrix of CNN2 predictions on *RawSignal* test set

the average duration among the original records, we found out that for "FALLING" and "JUMPING" the time-series are significantly shorter, as expected, since they are instantaneous actions. As a consequence, due to possible inaccuracies in the manual labelling, the head and the tail time instances of the original records are often respectively related to previous and subsequent actions. Once divided in frames, therefore, only few of them carry the useful information, probably determining poor representation learning and performances on this two classes.

VII. Concluding Remarks

Given the good results just shown in the previous section, at the end of this experimental project, we can conclude that the aim to prove Deep Learning paradigm ability to automatically extract significant features for Human Activity Recognition task has been achieved with satisfaction. The best model has turned out to be the Convolutional Neural Network, but in general the F1-scores of all the architectures have been really close to each other in the majority of the classes. In particular, the Denoising and the Variational Autoencoders performances should be taken in consideration for their remarkable capabilities in terms of time and compression of the original signals, which could be fundamental aspects when implementing real-time applications. As concerning the least notable outcomes such as the DCT transformation, the data augmentation techniques and the lower accuracy on the "FALLING" and "JUMPING" activities, we have reason to believe that a blended approach between the manual engineering proposed by [1] and the fully automatized tested here should be a valid possibility to improve them. A balance between automation and fully exploitation of the specific knowledge domain could be therefore the right direction for related future works.

To conclude, we tried our best to present a as much as possible professional and motivated solution for the project task of interest, experiencing how much a problem like activity recognition, which could be an everyday task for a current smartphone application, could be challenging and stimulating. We have understood that a good mathematical, and not only, preparation is fundamental to address a Human Data Analytic problem rigorously, in particular to provide meaningful data representation and architectures. Finally, we have learned that also dedication, passion and curiosity are necessary to produce an original peace of research in this field, which is so interconnected with many different scientific disciplines.

References

[1] K. Frank, M. Nadales, P. Robertson, and M. Angermann, "Reliable Real-Time Recognition of Motion Related Human Activities Using MEMS Inertial Sensors," in *Proceedings of the International Conference ION GNSS*, (Portland, Oregon, USA), 2010.

[2] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798 – 1828, 2013.

[3] J. Zhu, R. San-Segundo, and M. J. Pardo, "Feature extraction for robust physical activity recognition," *Human-centric Computing and Information Sciences*, vol. 7, no. 16, 2017.

[4] A. Wickramasinghe, D. C. Ranasinghe, C. Fumeaux, K. D. Hill, and R. Visvanathan, "Sequence learning with passive rfid sensors for real-time bed-egress recognition in older people," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 4, pp. 917–929, 2017.

[5] A. A. Varamin, E. Abbasnejad, Q. Shi, D. Ranasinghe, and H. Rezatofighi, "Deep auto-set: A deep auto-encoder-set network for activity recognition using wearables," *arXiv 1811.08127*, 2018.

[6] G. H. Y. LeCun, Y. Bengio, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] S. Latif, R. Rana, S. Khalifa, R. Jurdak, J. Qadir, and B. W. Schuller, "Deep Representation Learning in Speech Processing: Challenges, Recent Advances, and Future Trends," *CoRR*, vol. 2001.00378, 2020.

[8] D. K. Nithin and P. B. Sivakumar, "Generic feature learning in computer vision," *Procedia Computer Science*, vol. 58, pp. 202–209, 2015.

[9] 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 *24th International Conference on Artificial Intelligence*, 2015.

[10] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, P. W. J. Zhu, and J. Zhang, "Convolutional neural networks for human activity recognition using mobile sensors," in *6th International Conference on Mobile Computing, Applications and Services*, Nov 2014.

[11] F. M. Rueda, R. Grzeszick, G. A. Fink, S. Feldhorst, and M. ten Hompel, "Convolutional neural networks for human activity recognition using body-worn sensors," *Informatics*, vol. 5, no. 2, 2018.

[12] O. S. Eyobu and D. S. Han, "Feature representation and data augmentation for human activity classification based on wearable imu sensor data using a deep lstm neural network," *Sensors*, vol. 18, no. 9, 2018.

[13] F. J. Ordez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, 2016.

[14] D. D. Testa and M. Rossi, "Lightweight lossy compression of biometric patterns via denoising autoencoders," *IEEE signal processing letters*, vol. 22, no. 12, 2015.

[15] Y. Li, D. Shi, B. Ding, and D. Liu, "Unsupervised Feature Learning for Human Activity Recognition Using Smartphone Sensors," in *Mining Intelligence and Knowledge Exploration. Lecture Notes in Computer Science*, vol. 8891, R. Prasath, P. O'Reilly, Kathirvalavakumar T. (eds) Springer, 2014.

[16] B. Almaslukh, J. AlMuhtadi, and A. Artoli, "An effective deep autoencoder approach for online smartphone-based human activity recognition," *IJCSNS International Journal of Computer Science and Network Security*, vol. 17, no. 4, 2017.

[17] X. Gao, H. Luo, Q. Wang, F. Zhao, L. Ye, and Y. Zhang, "A human activity recognition algorithm based on stacking denoising autoencoder and lightgbm," *Sensor*, vol. 19, no. 4, 2019.

[18] M. H. M. Noor, M. A. Ahmadon, and M. K. Osman, "Activity Recognition using Deep Denoising Autoencoder," in *9th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, (Penang, Malaysia), 2019.

[19] M. Chaveroche, A. Malais, F. Colas, F. Charpillet, and S. Ivaldi, "A variational time series feature extractor for action prediction," *arXiv:1807.02350v2*, 2018.

[20] S. Mohammed and I. Tashev, "Unsupervised deep representation learning to remove motion artifacts in free-mode body sensor networks," in *IEEE 14th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, (Eindhoven), 2017.

[21] K. Chen, D. Zhang, L. Yao, B. Guo, Z. Yu, and Y. Liu, "Deep learning for sensor-based human activity recognition: Overview, challenges and opportunities," *arXiv:2001.07416*, 2020.

[22] E. V. Anazco, L. Rivera, H. Park, N. Park, and T.-S. Kim, "Human activities recognition with a single writs imu via a variational autoencoder and android deep recurrent neural nets," *Computer Science and Information Systems*, vol. 17, no. 2, pp. 581–597, 2020.

[23] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networkson sequence modeling," *arXiv:1412.3555v1*, 2014.

[24] M. J. Er, W. Chen, and W. Shiqian, "High-speed face recognition based on discrete cosine transform and rbf neural networks," in *IEEE Trans. on Neural Networks, vol.16, No.3, pp:679-691*, 2005.

[25] Z. He and L. Jin, "Activity recognition from acceleration data based on discrete consine transform and svm," in *Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics*, 2009.

[26] Q. Wen, L. Sun, X. Song, J. Gao, X. Wang, and H. Xu, "Time series data augmentation for deep learning: A survey," *arXiv:2002.12478v1*, 2020.

[27] H. Ohashi, M. Al-Naser, S. Ahmed, A. Takayuki, T. Sato, N. Nguyen, K. Nakamura, and A. Dangel, "Augmenting wearable sensor data with physical constraint for dnn-based human-action recognition," in *Time Series workshop at International Conference of Machine Learning (ICML)*, 2017.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[29] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light gated recurrent units for speech recognition," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 91–102, 2018.