DenseNets for Time Series Classification: towards automation of time series pre-processing with CNNs

Guillaume Richard
guillaume.richard@cmla.ens-cachan.fr
ENS Paris-Saclay / EDF R&D
Cachan, France / Palaiseau, France

Georges Hébrail
georges.hebrail@edf.fr
EDF R&D / IRT SystemX
Palaiseau, France

Mathilde Mougeot
mathilde.mougeot@cmla.ens-cachan.fr
ENS Paris-Saclay

Nicolas Vayatis
nicolas.vayatis@cmla.ens-cachan.fr
ENS Paris-Saclay

ABSTRACT

It is well known that data normalization is a fundamental pre-processing step for learning using Convolutional Neural Networks (CNN). Multiple normalization techniques have been proposed and finding an appropriate one is not an easy task. Motivated by applications in the energy consumption field, we study Time Series Classification (TSC) with deep learning techniques. We adapt DenseNets to a new convolutional architecture for TSC. We conduct an experimental study the impact of different data normalization techniques on this architecture. We propose a solution to mitigate different pre-processing methods and show its applicability across various fields.

KEYWORDS

Time Series Classification, Convolutional Neural Networks, Data Normalization

ACM Reference Format:
Guillaume Richard, Georges Hébrail, Mathilde Mougeot, and Nicolas Vayatis. 2018. DenseNets for Time Series Classification: towards automation of time series pre-processing with CNNs. In *MileTS '19: 5th KDD Workshop on Mining and Learning from Time Series, August 5th, 2019, Anchorage, Alaska, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Sequential and time series data mining remains one of the most important problem in Data Mining. Learning to represent and classify time series has led to applications in numerous fields. TSC is defined as the task of training a classifier on a dataset $\{\mathbf{X}, \mathbf{Y}\}$ in order to map a time series to a class. The UCR/UEA archive [5] opened the possibility of comparing TSC algorithms on a wide range of domains. Pre-processing is a crucial step in any application dealing with data. It includes cleaning, missing values, transformation, ... For image data, pixel intensities are frequently rescaled into a given range. For time series, instance standardization (also called z-normalization) is commonly used [5]. It was noted in a recent paper [8] that *"this traditional pre-processing step should be further studied [...] since normalization is known to have a huge effect on DNNs' learning capabilities"*.

A traditional TSC algorithm is a Nearest Neighbor classifier with a distance function specific to time series data such as Dynamic Time Warping, which was shown to be the best among several distance measures [3]. Learning new representations is also very common as time series data is high dimensional and subject to noise. For instance, Bag of SFA Symbols (BOSS) [16] builds a classifier upon the symbolic Fourier approximation. Shapelets, introduced in [18] and refined later [15] [4], are discriminative subsequences and allow a new representation for time series that can be fed into a classifier. Ensemble methods have been implemented to leverage on different representations and classifiers and are the current state-of-the-art for TSC: Elastic Ensemble [10], COTE [2], HIVE-COTE [11]. Recently deep learning methods have been applied to TSC problems. Using similar architectures to the computer vision community, Convolutional Networks, ResNets and Multi-Channel Convolutional Networks [19] were proposed. An experimental review [8] of these methods showed that ResNets lead to the most accurate results.

Finding the appropriate pre-processing is not an easy task and generally depends on inner data characteristics. Each technique may discard some information and should be used with caution. To the best of the authors' knowledge the impact of time series pre-processing on TSC has not been thoroughly studied even though it remains an important issue for many industrial practitioners. In this work, we introduce 4 different pre-processing methods and study their impact. We also propose DenseNets [6] to Time Series, expanding the family of deep learning architectures for TSC. Finally, we introduce a new way to mix information from different pre-processing and show its efficiency on an energy dataset and UCR data. The paper is organized as follows. Section II reviews the common pre-processing techniques for time series. Section III describes our novel convolutional architecture and its extensions. Section IV presents experimental results for UCR time series and energy consumption data, followed by concluding remarks.

2 PRE-PROCESSING FOR TSC

Data pre-processing can cover many subfields from sampling, infering missing values, denoising, detrending, ... In this paper we will consider only univariate time series uniformly sampled with no missing values.

MileTS '19, August 5th, 2019, Anchorage, Alaska, USA

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

2.1 Definitions

The literature shows a wide range of terms used interchangeably, hence we first define two of the most commonly used normalization methods: min-max normalization and standardization. Each of them is broken down into per instance and global normalization. For the rest of the paper, $\mathbf{X} \in \mathbb{R}^{n \times l}$ is the full dataset where n is the number of samples and l the time series length, $\mathbf{Y} \in \mathbb{R}^{n \times C}$ is the label vector. $X_i = \{X_i^1, ..., X_i^l\}$ denotes the i^{th} element of $\mathbb{X}$

Global min-max normalization (**GN**) normalizes the values of $\mathbf{X}$ according to its minimum and maximum converting it into the range $[0, 1]$. Typical issues would be out-of-sample minimum and maximum and outliers, which happens often in many time series applications. Global standardization (**GS**) standardizes the values of $\mathbf{X}$ according to its mean and standard deviation. This very common normalization is usually done per variable in order to give the same scale for each variable but for time series it would mean to normalize the data per time stamp, destroying temporal structure. Hence global transformations are a rescaling of data. Instance normalization differs since each time series of the dataset is normalized using its own statistics. We define Instance min-max Normalization (**IN**) and Instance Standardization (**IS**) in the following table.

	Min-Max	Standardization
Global	$GN(X_i^j) = \frac{X_i^j - min(\mathbf{X})}{max(\mathbf{X}) - min(\mathbf{X})}$	$GS(X_i^j) = \frac{X_i^j - mean(\mathbf{X})}{std(\mathbf{X})}$
Instance	$IN(X_i^j) = \frac{X_i^j - min(X_i)}{max(X_i) - min(X_i)}$	$IS(X_i^j) = \frac{X_i^j - mean(X_i)}{std(X_i)}$

Table 1: Normalization methods

2.2 Why does normalization matter?

Most existing approaches use the instance standardization (also called z-normalization) to pre-process time series. For instance UCR archive included only z-normalized datasets until a recent update. The incentive behind this choice is that similarity between two time series can be meaningless without proper pre-processing in presence of an offset or a scale variation [14]. But we argue that this choice might not be optimal for every domain, especially when the scale or the offset are discriminative.

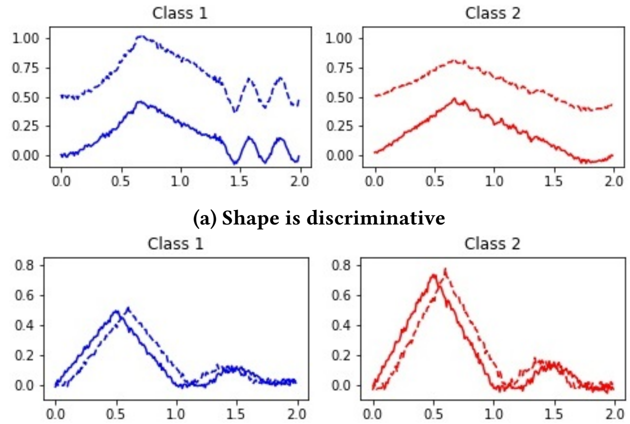
It can be seen on a toy example: in Figure 1a, without normalization, a euclidean or DTW-based classifier would not discriminate the classes properly: dotted time series is classified in the same class. With classes from Figure 1b, instance standardization will have the opposite effect. One can see that there is no obvious choice of normalization and for more complex data, balancing shape and scale information is challenging.

2.3 Non-linear scaling

Another common data transformation is non linear scaling. Non-linear scaling is a common pre-processing technique for non normal data. The main motivation is to make data more normal, making it easier to manipulate. Box-Cox transformation is a classical technique and its formula is:

$$boxcox(x) = \begin{cases} \frac{(X_i^j)^\lambda - 1}{\lambda} & \text{if } \lambda > 0 \\ \log(X_i^j), & \text{if } \lambda = 0 \end{cases}$$

The Box-Cox test finds the optimal λ to make data the most normal as possible. The optimal λ is computed globally or per instance depending on the subsequent normalization. It has been succesfully used for time series forecasting [13], but not applied to TSC to the best of the authors' knowledge.



(a) Shape is discriminative

(b) Scale is discriminative

Figure 1: Toy examples

3 CONVOLUTIONAL NEURAL NETWORKS FOR TSC

TSC is defined as the task of training a classifier on a dataset $\{\mathbf{X}, \mathbf{Y}\}$ mapping a time series to a class. A recent review has experimented different architectures [8] and pointed out ResNets [17] and Fully Convolutional Neural Networks to be the most efficient across datasets. We will briefly introduce DenseNets, a new architecture inspired from image models that we apply to time series and discuss how normalization impacts neural networks.

3.1 DenseNets for TSC

A deep neural network is a composition of parametric functions (layers) aiming to predict a target from an input for a given task. Convolutional Neural Networks (CNN) are a specific neural network made of convolutional layers. Convolutional architectures have shown good results as they extract meaningful local features from their input. A simple CNN is a composition of convolutional and fully connected layers. Namely it can be expressed as

$$\tilde{y} = f_d(f_{d-1}(...(f_1(x))) \quad (1)$$

where each $f_.$ is a non-linear transformation. Convolutional layers and intermediate pooling are successively applied in order to extract features at different scales before a global pooling operation and a fully connected layer that predicts a label. We present an architecture inspired from the computer vision community [6] that has not been proposed for TSC to the best of the authors' knowledge.

DenseNets. We introduce skip connections from different levels in the network through concatenation. Namely, the k^{th} layer receives the outputs of some preceding layers as an input ; denoting z_k the output of the k^{th} layer and $[z_{k-m}, ..., z_{k-1}]$ the concatenation of the m previous layers, the output of the k^{th} layer is

$$z_k = f_k([z_{k-m}, ..., z_{k-1}])$$

DenseNets allow to produce more complex information from layer outputs than ResNets. On the other hand, they tend to have

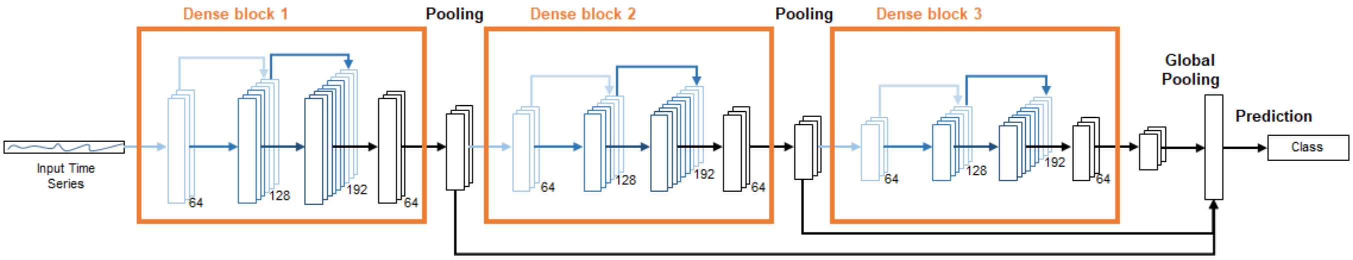


Figure 2: DenseNet: each block is made of successive convolutions and skip connections ended by a bottleneck convolution

larger layer inputs, due to the successive concatenations, depending on the number m of preceding layers being concatenated. In our architecture, we use bottleneck layers: after a dense block, a bottleneck layer brings back the number of inputs to the initial number of feature maps, as described on Figure 2. Hence for a fixed number of feature maps K and maximum m_{max} size of dense block, the number of inputs never exceed $K \times m_{max}$.

Another novelty of our architecture is to explicitly feed features from different scales to the final predictor. Namely we concatenate the outputs of each dense block to create the input of the last fully connected layer. As the number of dense blocks is relatively small, the input size stays tractable. In the next section, we discuss normalization for neural networks and propose solutions to balance shape and scale information.

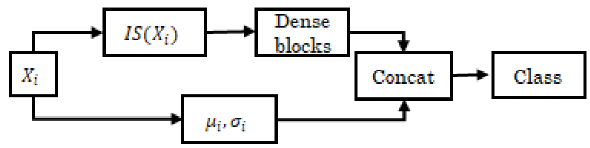
3.2 Data normalization for Neural Networks

In theory, it is rarely strictly necessary to standardize the inputs of a neural network and most pre-processing tricks are hard to analyze properly. In practice, standardization allows non-fittable networks to be fittable [9] . Recently batch normalization [7] has had a great impact on neural network training and works as a speedup technique for neural networks.

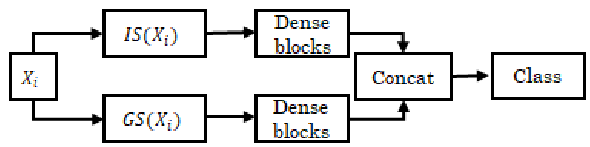
One can note that global standardization is simply a rescaling of the data and could be replaced by an adequate weight initialization. As seen in Section 2, for some data we would like to use information from the scale of each time series without compromising information derived from local shapes. Hence we propose two solutions to balance those pieces of information.

FeatNet. The first solution is to create a new architecture with two entries as summarized in Figure 3a. One entry corresponds to the instance-normalized time series, which is passed into a convolutional architecture. At the fully-connected level, the output of the convolutional blocks is concatenated with the other entry, containing the scale information from the time series (mean and standard deviation for standardization ; minimum and maximum for min-max normalization)

Ens-Norm Network. The second solution is to create an architecture with different entries corresponding to the input time series, normalized and scaled differently for each entry. Each entry is passed into convolutional blocks with no weight sharing. The outputs are then concatenated into a fully connected layer that gives the final prediction. Creating separate channels with different information is similar to the Multi-Channel Neural Network introduced in [19].



(a) FeatNet



(b) EnsNormNet

Figure 3: Proposed architectures

4 EXPERIMENTS

We ran experiments on two examples: a benchmark TSC archive and an appliance recognition problem. The second dataset motivated this study as it is a domain where both shape and scale of the data are discriminative.

4.1 UCR archive

Data. Firstly we use the UCR archive [5] to attest the impact of normalization techniques across datasets with varying characteristics. The first version of the archive contains only datasets that have already been standardized per instance but 37 of the more recent datasets are not instance standardized and are included in this study.

Architecture. Time series lengths and training size vary across datasets but we keep the same architectures and training parameters for every dataset. It is made of 3 dense blocks, each of them composed of 3 convolutional layers with 64 filters of size 7, 5 and 3. It may not be optimal but we emphasize that our goal is not to get the best performance for each dataset but to show that normalization has a tremendous impact on TSC and illustrate that our new architecture can benefit from ensembling two normalizations. Each complete architecture, implemented with Keras-Tensorflow, can be found in Appendix A.

Experiments. For each dataset, we have trained 8 neural networks corresponding to the following scenarios. When it is not specified, the architecture used is the DenseNet.

- Global Standardization (**GS**)
- Global min-maxNormalization (**GN**)
- Instance Standardization (**IS**)

- Instance min-max Normalization (**IN**)
- Box-Cox Transformation + Global Standardization (**BC-GS**)
- Box-Cox Transformation + Instance Standardization (**BC-IS**)
- Instance Standardization + FeatNet (**IS-Feat**)
- Instance Standardization - Global Standardization + NormEnsNet (**IS-GS-NEN**)

Each of them was run 10 times with different seeds for weight initialization and we report the mean classification accuracy in Appendix B. Training times differ for each dataset but a complete run (8 networks on 37 dataset) takes approximately 5 hours on our GPU cluster.

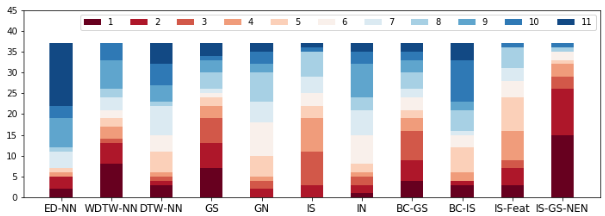


Figure 4: Rank distribution of each method over the 37 UCR un-normalized datasets. Each bar corresponds to a method. Red shows a high rank and blue a low rank.

Results. In order to compare with existing methods, we report the results available online [1] using Nearest Neighbour classifiers associated with Euclidean Distance (ED-NN), Dynamic Time Warping (DTW-NN) and Dynamic Time Warping with a learned window (WDTW-NN). The full results are available in Appendix B and we plot the rank distribution of each algorithm on Figure 4.

The first observation is that different data preparations affect the performance of a convolutional neural network. Moreover there is no universal choice of normalization to get the best classification accuracy. For most datasets, the Ens-Norm Network achieves better accuracy than the simple DenseNets, which indicates that this architecture is able to derive the best from both standardizations. Standardization seems to be more efficient than min-max normalization in general. Finally, non-linear scaling with box-cox transformation does not bring any significant improvement for most domains.

Overall we achieve slightly better results than the existing nearest neighbour classifiers. We do believe that better accuracies can be achieved, notably with ensemble-based methods, whose voting scheme could even include neural networks. We conducted experiments with ResNets and FCN architectures that generally did not lead to better accuracies but showed similar observations for the impact of pre-processing. We plan to produce a full comparison in a future work.

4.2 Appliance recognition

A field where time series scale is of particular importance is energy consumption. We study the example of appliance recognition using a load monitoring dataset: REFIT [12]. Appliance recognition corresponds to classify devices given their consumption profiles. REFIT project monitors household electricity consumption in 20 homes in the UK.

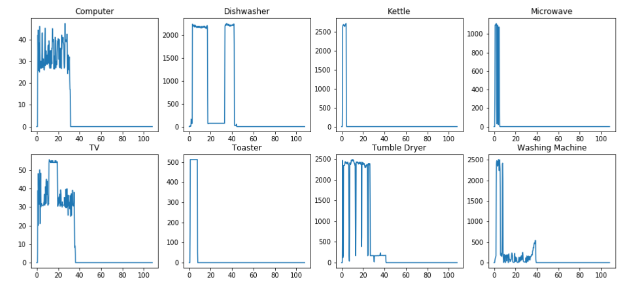


Figure 5: REFIT: extracted signatures and classes (y-scales are different)

From each device in each house, we extract appliance signatures (consumption pattern when the device is on). Our task is to efficiently classify appliance signatures. We only work on devices with sufficient training data. As the sampling is not uniform, we uniformly resample data every 10*s*. After extracting signatures, they are cropped or padded with zeros so that every signature has the same length 900, corresponding to 2 hours.

The same notations as in previous experiments are kept. Architectures and training procedures are the same. The results are produced using a leave-one-out procedure, using all houses except one for training and testing on the remaining one so that devices in the test set have not been seen during training.

In Table 2, we represent the macro F1-score for different classifiers. One can see that the Norm-Ens-Net performs the best. In Appendix C, we report confusion matrices for **GS**, **IS** and **IS-GS-NEN**. One can see that instance standardization has a bad impact on classification performance, especially for discriminating devices similar in shape such as Toaster/Kettle/Microwave or Computer/TV. At the same time, **GS** is not optimal for separating Dishwashers from Washing Machines for instance. For this application, we highlight that **IS-GS-NEN** gets the best of both worlds.

GS	GN	IS	IN
78,37 (0,63)	77,37 (0,58)	75,69 (0,89)	75,48 (0,76)
BC-GS	BC-IS	IS-Feat	**IS-GS-NEN**
77,99 (0,43)	75,83 (0,73)	76,11 (0,97)	**83,39 (0,54)**

Table 2: F1 score (%) for different architectures with standard deviation over 10 runs

5 CONCLUSION AND FUTURE WORK

Instance standardization (or z-normalization) should be carefully used as a pre-processing step for TSC. Depending on the application better normalization techniques can be used. A first option is to try different methods and chose the best one. Ensembling different normalizations seems to be a robust alternative. We showed for electricity datasets that an ensemble can extract meaningful information. This approach can also be extended to other fields.

Our future work will study multivariate time series, as normalization techniques are even more crucial in this case. We plan to conduct a more general work to confirm this first intuitive and experimental study. In particular, we assume that normalization has effects on both data characteristics and neural network optimization procedure which are hard to quantify.

REFERENCES

[1] 2019. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/

[2] A. Bagnall, J. Lines, J. Hills, and A. Bostrom. 2015. Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles. *IEEE Transactions on Knowledge and Data Engineering* 27, 9 (Sep. 2015), 2522–2535. https://doi.org/10.1109/TKDE.2015.2416723

[3] Anthony J. Bagnall and Jason Lines. 2014. An Experimental Evaluation of Nearest Neighbour Time Series Classification. *CoRR* abs/1406.4757 (2014). arXiv:1406.4757 http://arxiv.org/abs/1406.4757

[4] Aaron Bostrom and Anthony Bagnall. 2015. Binary shapelet transform for multiclass time series classification. In *International Conference on Big Data Analytics and Knowledge Discovery*. Springer, 257–269.

[5] Hoang Anh Dau, Anthony J. Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn J. Keogh. 2018. The UCR Time Series Archive. *CoRR* abs/1810.07758 (2018). arXiv:1810.07758 http://arxiv.org/abs/1810.07758

[6] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.

[7] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[8] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2018. Deep learning for time series classification: a review.

[9] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. 2012. Efficient backprop. In *Neural networks: Tricks of the trade*. Springer, 9–48.

[10] Jason Lines and Anthony Bagnall. 2015. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery* 29, 3 (2015), 565–592.

[11] Jason Lines, Sarah Taylor, and Anthony Bagnall. 2018. Time Series Classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles. *ACM Trans. Knowl. Discov. Data* 12, 5, Article 52 (July 2018), 35 pages. https://doi.org/10.1145/3182382

[12] David Murray and Lina Stankovic. 2015. REFIT: Electrical Load Measurements. University of Strathclyde. (2015).

[13] Tommaso Proietti and Helmut Lütkepohl. 2013. Does the Box–Cox transformation help in forecasting macroeconomic time series? *International Journal of Forecasting* 29, 1 (2013), 88–99.

[14] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 2012. Searching and Mining Trillions of Time Series Subsequences Under Dynamic Time Warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. ACM, New York, NY, USA, 262–270. https://doi.org/10.1145/2339530.2339576

[15] Thanawin Rakthanmanon and Eamonn Keogh. 2013. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 668–676.

[16] Patrick Schäfer. 2015. The BOSS is Concerned with Time Series Classification in the Presence of Noise. *Data Min. Knowl. Discov.* 29, 6 (Nov. 2015), 1505–1530. https://doi.org/10.1007/s10618-014-0377-7

[17] Zhiguang Wang, Weizhong Yan, and Tim Oates. 2017. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*. IEEE, 1578–1585.

[18] Lexiang Ye and Eamonn Keogh. 2011. Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data mining and knowledge discovery* 22, 1-2 (2011), 149–182.

[19] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. 2014. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*. Springer, 298–310.

A ARCHITECTURES

For each convolution, elu is used as an activation function. The last activation is a $softmax$. Moreover, each bottleneck convolution is associated with BatchNormalization. Weight initialization is done using Glorot procedure. The chosen optimizer is Adam with early stopping and decay.

Layers	Input shape	Output shape	Filter shape
Input	Time series of length l		
Conv (1)	$l \times 1$	$l \times 64$	1×7
Conv (2)	$l \times 64$	$l \times 64$	64×5
Conv (3)	$l \times 128$	$l \times 64$	128×3
Conv (4)	$l \times 192$	$l \times 64$	192×3
Pooling (1)	$l \times 64$	$l/2 \times 64$	2
Conv (5)	$l/2 \times 64$	$l/2 \times 64$	64×7
Conv (6)	$l/2 \times 64$	$l/2 \times 64$	64×5
Conv (7)	$l/2 \times 128$	$l/2 \times 64$	128×3
Conv (8)	$l/2 \times 192$	$l/2 \times 64$	192×3
Pooling (2)	$l/2 \times 64$	$l/4 \times 64$	2
Conv (9)	$l/4 \times 64$	$l/4 \times 64$	64×7
Conv (10)	$l/4 \times 64$	$l/4 \times 64$	64×5
Conv (11)	$l/2 \times 128$	$l/4 \times 64$	128×3
Conv (12)	$l/4 \times 192$	$l/4 \times 64$	192×3
Pooling (3)	$l/4 \times 64$	$l/8 \times 64$	2
Merge	Pooling (1)+(2)+(3)		
Dense	$56l$	n_{class}	$56l$

Table 3: DenseNet architecture used in both experiments

Layers	Input shape	Output shape	Filter shape
Input (1)	Instance-standardized time series		
Conv (1)	$l \times 1$	$l \times 64$	1×7
Conv (2)	$l \times 64$	$l \times 64$	64×5
Conv (3)	$l \times 128$	$l \times 64$	128×3
Conv (4)	$l \times 192$	$l \times 64$	192×3
Pooling (1)	$l \times 64$	$l/2 \times 64$	2
Conv (5)	$l/2 \times 64$	$l/2 \times 64$	64×7
Conv (6)	$l/2 \times 64$	$l/2 \times 64$	64×5
Conv (7)	$l/2 \times 128$	$l/2 \times 64$	128×3
Conv (8)	$l/2 \times 192$	$l/2 \times 64$	192×3
Pooling (2)	$l/2 \times 64$	$l/4 \times 64$	2
Conv (9)	$l/4 \times 64$	$l/4 \times 64$	64×7
Conv (10)	$l/4 \times 64$	$l/4 \times 64$	64×5
Conv (11)	$l/2 \times 128$	$l/4 \times 64$	128×3
Conv (12)	$l/4 \times 192$	$l/4 \times 64$	192×3
Pooling (3)	$l/4 \times 64$	$l/8 \times 64$	2
Input (2)	μ_i, σ_i		
Merge	Pooling (1)+(2)+(3) + Input (2)		
Dense	$56l + 2$	n_{class}	$56l + 2$

Table 4: FeatNet architecture used in both experiments

Layers	Input shape	Output shape	Filter shape	Layers	Input shape	Output shape	Filter shape
Input (1)	Global-Standardized TS			Input (2)	Instance-Standardized TS		
Conv (1)	$l \times 1$	$l \times 64$	1×7	Conv (13)	$l \times 1$	$l \times 64$	1×7
Conv (2)	$l \times 64$	$l \times 64$	64×5	Conv (14)	$l \times 64$	$l \times 64$	64×5
Conv (3)	$l \times 128$	$l \times 64$	128×3	Conv (15)	$l \times 128$	$l \times 64$	128×3
Conv (4)	$l \times 192$	$l \times 64$	192×3	Conv (16)	$l \times 192$	$l \times 64$	192×3
Pooling (1)	$l \times 64$	$l/2 \times 64$	2	Pooling (4)	$l \times 64$	$l/2 \times 64$	2
Conv (5)	$l/2 \times 64$	$l/2 \times 64$	64×7	Conv (17)	$l/2 \times 64$	$l/2 \times 64$	64×7
Conv (6)	$l/2 \times 64$	$l/2 \times 64$	64×5	Conv (18)	$l/2 \times 64$	$l/2 \times 64$	64×5
Conv (7)	$l/2 \times 128$	$l/2 \times 64$	128×3	Conv (19)	$l/2 \times 128$	$l/2 \times 64$	128×3
Conv (8)	$l/2 \times 192$	$l/2 \times 64$	192×3	Conv (20)	$l/2 \times 192$	$l/2 \times 64$	192×3
Pooling (2)	$l/2 \times 64$	$l/4 \times 64$	2	Pooling (5)	$l/2 \times 64$	$l/4 \times 64$	2
Conv (9)	$l/4 \times 64$	$l/4 \times 64$	64×7	Conv (21)	$l/4 \times 64$	$l/4 \times 64$	64×7
Conv (10)	$l/4 \times 64$	$l/4 \times 64$	64×5	Conv (22)	$l/4 \times 64$	$l/4 \times 64$	64×5
Conv (11)	$l/2 \times 128$	$l/4 \times 64$	128×3	Conv (23)	$l/2 \times 128$	$l/4 \times 64$	128×3
Conv (12)	$l/4 \times 192$	$l/4 \times 64$	192×3	Conv (24)	$l/4 \times 192$	$l/4 \times 64$	192×3
Pooling (3)	$l/4 \times 64$	$l/8 \times 64$	2	Pooling (6)	$l/4 \times 64$	$l/8 \times 64$	2
Merge	Pooling (1)+(2)+(3)+(4)+(5)+(6)						
Dense	Class prediction						

Table 5: EnsNormNet architecture: dense blocks are applied in parallel before being merged

B FULL RESULTS ON UCR DATA

Dataset	ED-NN	WDTW-NN	DTW-NN	GS	GN	IS	IN	BC-GS	BC-IS	IS-Feat	IS-GS-NEN
AllGestureWiimoteX	51,57%	71,71%	71,57%	**77,89%**	68,34%	73,34%	66,89%	76,83%	67,14%	68,34%	77,83%
AllGestureWiimoteY	56,86%	**73,00%**	72,86%	72,60%	72,91%	73,34%	72,26%	75,80%	68,86%	72,91%	72,91%
AllGestureWiimoteZ	45,43%	65,14%	64,29%	70,06%	67,49%	69,40%	66,20%	75,80%	50,86%	67,49%	**76,31%**
BME	83,33%	98,00%	90,00%	92,13%	92,13%	95,33%	96,27%	94,27%	**99,07%**	94,53%	96,27%
Chinatown	95,36%	95,36%	95,65%	97,97%	97,97%	97,97%	97,97%	97,68%	97,97%	97,97%	**98,26%**
Crop	71,17%	71,17%	66,52%	78,30%	74,52%	76,01%	74,90%	78,72%	75,19%	74,52%	**79,26%**
DodgerLoopDay	55,00%	58,75%	50,00%	55,00%	40,75%	61,25%	42,75%	49,25%	40,75%	55,75%	**62,75%**
DodgerLoopGame	88,41%	**92,75%**	87,68%	86,96%	88,41%	84,35%	85,65%	88,41%	83,33%	88,41%	88,26%
DodgerLoopWeekend	**98,55%**	97,83%	94,93%	88,12%	95,36%	92,75%	92,75%	89,71%	93,77%	95,36%	92,75%
EOGHorizontalSignal	41,71%	47,51%	50,28%	61,27%	58,62%	58,62%	59,12%	59,12%	52,98%	61,49%	**61,82%**
EOGVerticalSignal	44,20%	47,51%	44,75%	48,67%	46,35%	46,24%	44,20%	46,35%	42,38%	**49,12%**	48,84%
Fungi	82,26%	82,26%	**83,87%**	53,87%	68,06%	64,84%	72,37%	28,82%	49,78%	68,06%	64,52%
GestureMidAirD1	57,69%	**63,85%**	56,92%	59,54%	55,85%	62,15%	58,15%	58,15%	59,23%	59,23%	63,08%
GestureMidAirD2	49,23%	60,00%	**60,77%**	46,46%	57,23%	59,23%	54,46%	42,31%	42,92%	59,23%	59,85%
GestureMidAirD3	34,62%	**37,69%**	32,31%	19,85%	30,31%	33,54%	30,77%	21,38%	26,46%	34,46%	34,31%
GesturePebbleZ1	73,26%	82,56%	79,07%	59,88%	63,37%	84,19%	62,21%	62,33%	76,74%	**84,77%**	84,42%
GesturePebbleZ2	67,09%	**77,85%**	67,09%	59,62%	55,06%	73,80%	61,01%	67,97%	73,42%	71,27%	74,43%
GunPointAgeSpan	89,87%	96,52%	91,77%	98,67%	99,37%	99,56%	**99,62%**	98,61%	98,10%	99,37%	**99,62%**
GunPointMaleVersusFemale	97,47%	97,47%	99,68%	**99,81%**	99,37%	99,05%	99,37%	99,68%	96,96%	99,37%	**99,81%**
GunPointOldVersusYoung	95,24%	96,51%	83,81%	**100,00%**	96,70%	97,08%	96,38%	**100,00%**	95,11%	96,70%	99,87%
HouseTwenty	66,39%	94,12%	92,44%	94,12%	92,44%	42,02%	42,02%	89,92%	42,02%	42,02%	**94,79%**
InsectEPGRegularTrain	67,87%	82,73%	87,15%	**100,00%**	99,68%	97,75%	98,96%	**100,00%**	96,79%	99,68%	100,00%
InsectEPGSmallTrain	66,27%	69,48%	73,49%	35,74%	35,74%	95,98%	47,39%	35,74%	90,68%	**96,71%**	96,47%
MelbournePedestrian	84,82%	84,82%	79,06%	96,44%	90,25%	90,53%	90,43%	97,02%	90,36%	95,31%	**97,11%**
PLAID	53,63%	83,61%	83,80%	83,99%	83,09%	84,21%	84,25%	83,09%	69,42%	83,09%	**84,25%**
PickupGestureWiimoteZ	56,00%	66,00%	66,00%	**76,00%**	66,80%	70,40%	60,80%	72,40%	47,60%	70,00%	75,60%
PigAirwayPressure	5,77%	9,62%	10,58%	**18,17%**	6,15%	10,48%	6,63%	15,10%	10,87%	13,56%	17,40%
PigArtPressure	12,50%	19,71%	24,52%	51,06%	16,73%	47,50%	19,52%	49,90%	44,33%	16,73%	**52,02%**
PigCVP	8,17%	15,87%	15,38%	47,21%	18,65%	50,58%	19,33%	46,44%	**52,69%**	51,15%	52,31%
PowerCons	93,33%	92,22%	87,78%	95,44%	89,89%	90,56%	89,22%	94,78%	89,00%	89,89%	**96,89%**
Rock	**84,00%**	**84,00%**	60,00%	71,60%	62,00%	59,60%	62,00%	73,20%	62,40%	62,00%	62,00%
SemgHandGenderCh2	76,17%	**84,50%**	80,17%	79,67%	65,83%	83,40%	35,00%	82,93%	81,77%	82,77%	84,00%
SemgHandMovementCh2	36,89%	**63,78%**	58,44%	54,53%	40,62%	44,71%	39,24%	49,60%	46,31%	40,62%	39,24%
SemgHandSubjectCh2	40,44%	**80,00%**	72,67%	71,69%	70,98%	74,71%	73,24%	70,27%	69,29%	72,22%	73,24%
ShakeGestureWiimoteZ	60,00%	84,00%	86,00%	88,40%	88,40%	84,40%	86,40%	87,20%	72,40%	88,40%	**89,20%**
SmoothSubspace	90,67%	94,67%	82,67%	98,00%	97,33%	96,53%	97,33%	**98,67%**	96,00%	97,33%	97,33%
UMD	76,39%	97,22%	**99,31%**	**99,31%**	98,61%	98,61%	98,61%	**99,31%**	**99,31%**	98,61%	**99,31%**

Table 6: Accuracies for each non-normalized UCR dataset

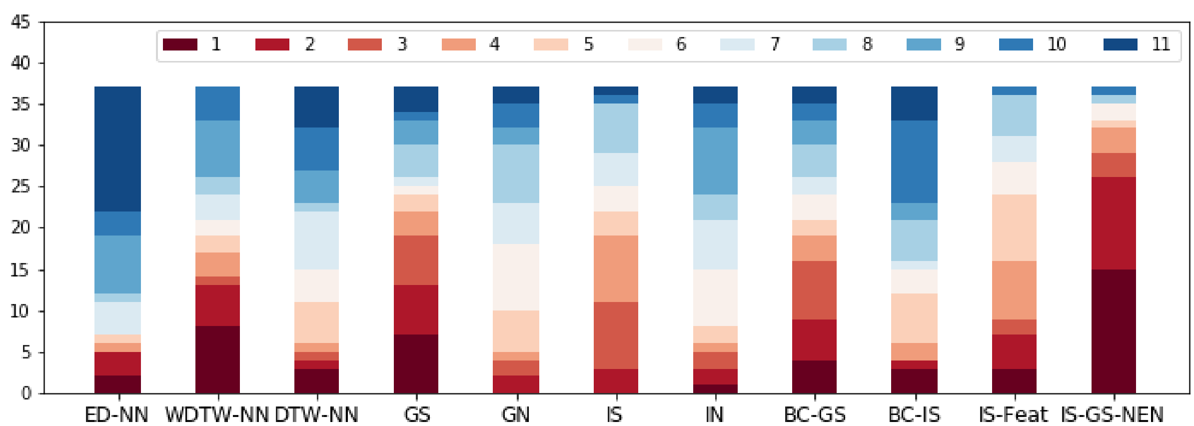


Figure 6: Rank distribution of each method over UCR non-normalized datasets. Each bar corresponds to a method. Red shows a high rank and blue a low rank.

C CONFUSION MATRICES FOR REFIT DATA

GS		Prediction								Recall
		Computer	Dishwas her	Kettle	Microwa ve	TV	Toaster	Tumble Dryer	Washing Machine	
Truth	Computer	3039	183	283	30	767	18	9	17	69,93%
	Dishwas her	17	3635	13	9	18	23	7	643	83,28%
	Kettle	174	42	8171	183	100	593	4	42	87,78%
	Microwa ve	8	67	321	7016	267	512	212	58	82,92%
	TV	1075	44	93	18	17187	25	43	3	92,96%
	Toaster	8	0	216	482	153	2460	1	52	72,95%
	Tumble Dryer	32	121	18	43	99	28	1380	99	75,82%
	Washing Machine	15	753	65	29	27	9	76	1637	62,70%
Precision		69,57%	75,03%	89,01%	89,83%	92,31%	67,07%	79,68%	64,17%	

Figure 7: Confusion matrix with Global Standardization

IS		Prediction								Recall
		Computer	Dish-washer	Kettle	Micro-wave	TV	Toaster	Tumble Dryer	Washing Machine	
Truth	Computer	2234	78	312	76	1573	43	7	23	51,40%
	Dish-washer	17	4218	19	23	15	3	2	68	96,63%
	Kettle	372	29	6440	959	256	1218	6	29	69,18%
	Micro-wave	292	42	643	6257	321	682	145	79	73,95%
	TV	2753	54	41	33	15550	17	12	28	84,11%
	Toaster	27	16	431	411	211	2227	8	41	66,04%
	Tumble Dryer	32	53	24	46	88	13	1537	27	84,45%
	Washing Machine	13	236	48	45	41	10	4	2214	84,80%
Precision		38,92%	89,25%	80,92%	79,71%	86,13%	52,86%	89,31%	88,24%	

Figure 8: Confusion matrix with Instance Standardization

IS-GS-NEN		Prediction								
		Computer	Dishwasher	Kettle	Microwave	TV	Toaster	Tumble Dryer	Washing Machine	Recall
Truth	Computer	3114	160	250	30	773	2	1	16	71,65%
	Dishwasher	20	4121	11	9	12	5	4	183	94,41%
	Kettle	168	52	8185	159	123	581	3	38	87,93%
	Microwave	8	73	281	6953	290	570	225	61	82,18%
	TV	1099	39	62	29	17214	11	22	12	93,11%
	Toaster	6	0	220	446	164	2488	0	48	73,78%
	Tumble Dryer	47	54	24	41	96	13	1518	27	83,41%
	Washing Machine	13	267	68	45	41	10	4	2163	82,84%
Precision		69,59%	86,47%	89,94%	90,16%	91,99%	67,61%	85,42%	84,89%	

Figure 9: Confusion matrix with Ens-Norm-Net