MULTIVARIATE TIME SERIES CLASSIFICATION: UNDERSTANDING HOUSEHOLD BEHAVIOUR THROUGH THEIR ELECTRICITY CONSUMPTION

MANUSCRIPT IN PREPARATION

Noah Sarfati
Ecole Polytechnique
noah.sarfati@polytechnique.edu

Jesse Read
Ecole Polytechnique
jesse.read@polytechnique.edu

Mohamed Alami Cheboune
Ecole Polytechnique
Mohamed.alami-chehboune@polytechnique.edu

ABSTRACT

Multivariate Time Series (MTS) classification is a branch of machine learning aiming at giving labels to an agent when one collected multiple pieces of information through time. In the past years, with the increase in computing powers, deep learning received enormous attention and allowed an exponential increase in areas such as Computer Vision and Natural Language Processing. In our study, we were given a newly created dataset generated by EDF, the French leading electricity utility. The goal then was to understand household behaviour via their electricity consumption. Influenced by the progress in Computer Vision and more particularly in image classification and object detection, we propose a new implementation of the InceptionTime model, using the Inception module as an encoder and a machine learning classifier to decode the transformed data. We leverage the focal loss for a multiclass, multilabel and regression problem. While the focal loss was first described for an object detection task, we achieved a non-negligible increase in precision compared to the traditional cross-entropy. We also propose a data-augmentation method by cutting the time series and show the robustness of our predictions with an increase in precision. Finally, in this study, we were able to reach high accuracy with an unbalanced and a little dataset and show that it is possible to learn jointly households characteristics and present a robust model for transfer learning.

Keywords Multivariate time series classification · Deep learning · Inception · Focal loss · Multioutput-multiclass classification

1 Introduction

Recently, there was an explosion in data availability and computing powers. Time series are present in many real-world applications ranging from health care, energy consumption, epidemiology, finance, automated disease detection, anomaly detection etc. For industries like EDF, relying mainly on time series data, a better understanding is crucial. Predicting for example the number of inhabitants could prevent some households that lie to pay less taxes. It could also help understanding their behaviour to improve the green production.

A variety of methods have been developed for MTS classification. These approaches range from statistical to machine learning. They leverage the ground truth data to learn the trends and patterns. Popular families of statistical methods for univariate time series study include auto-regression (AR), autoregressive moving average (ARMA), and autoregressive integrated moving average (ARIMA). These models extend to vector autoregressive model (VAR) for multivariate problems. The issues with that type of methods is that they perform poorly on dataset where points are not dependant from the past.

Traditional machine learning methods such as the Distance-based KNN with dynamic time warping or Interval-based random forests classifiers were developed. But they need manual feature extractions and are computationally expensive for long time series with several dimensions. Time series is one of the most complicated data to understand and visualise. For example it is not hard to classify an image, but we wouldn't be able to classify a temporal pattern only based on summary features such as mean, standard deviation, and slope. This is why, deep learning is now widely used since it is capable to learn features extraction. One of the most famous method for temporal data such as texts, is Recurrent Neural Network (RNN). This sequence-aligned models are natural choices for modeling time series data. However, when trained on long time series, RNN typically suffer from the vanishing gradient problem, that means that the parameters in the hidden layers either don't change that much or they lead to numeric instability and chaotic behavior.

In this work, we leverage Convolutions Neural Networks mainly used in Computer Vision. Indeed, it consists in sliding one-dimensional filters over the time series, thus enabling the network to extract non-linear discriminant features that are time-invariant. Therefore it has the potential to model complex dynamics of time series data that are challenging for sequence models like LSTM or GRU. We adapted InceptionTime [1], a SOTA ensemble model for time series classification. It is based on the Inception layer [2] originally used for end-to-end image classification. This model is capable to adapt itself on input of different length via transfer learning. Indeed one issue with time series is that often some data points are missing one doesn't always have input of the same length. We also show that that the use of convolutions outperforms traditional machine learning algorithms and RNN. Finally we show that the focal loss [3] and our proposed architecture increases the accuracy on unbalanced time series dataset and show that it is possible to learn jointly on multiclass tasks. Mainly our contribution are the following:

- We developed a general architecture for a MTS, multioutput-multiclass classification and regression task based on the Inception with a strong capability for transfer learning on MTS of different length than the ones it was trained with.
- We demonstrate the effectiveness of the focal loss on the problem of MTS classification on unbalanced data compared to the standard categorial cross-entropy.
- We show that deep CNN can be used to generalise a little dataset with an appropriate learning rate and the use of an ensemble method.
- We propose a data augmentation method to increase the accuracy on unrepresented classes.
- Using EDF dataset as a case study, we demonstrate that it is possible to learn jointly information one can find on a MTS. We also showed that electricity consumption can tell a lot on household composition and behaviour and that one does not need an entire year of data to have excellent results.

2 Background

In this first part we will explain the basic notions one need to understand the problem and to evaluate properly the different models. We will define a multivariate time series, a multi-output-multi-class problem, residual and convolutional neural networks.

2.1 Multivariate Time Series

Before introducing the different types of neural networks architectures, we go through some formal definitions for TSC.

Definition 1: Let $n \in \mathbb{N}$ and $t_i \in \mathbb{R}\ \forall i \in \{1, .., n\}$. We define a univariate time serie as an ordered sequence of the form $T = [t_1, ..., t_n]$.

Definition 2: Let $(n, M) \in \mathbb{N}^2$, and $t_i \in \mathbb{R}^M\ \forall i \in \{1, .., n\}$. We define a M-dimension multivariate time serie as an ordered sequence of the form $T = [t_1, ..., t_n]$.

Definition 3: Let $m \in \mathbb{N}$, we call $\mathcal{T}_m$ the set of real valued m-dimension multivariate time series.

2.2 Multivariate Time Series Classification

Definition 4: Let $(n, m, p) \in \mathbb{N}^3$ and $(T_1, ..., T_n) \in (\mathcal{T}_m)^n$. We define a p-multiclass label, an integer $y_i \in [0, 1, .., p]\ \forall i \in [1, 2, .., n]$. A multiclass classification problem is then an association of the form $\mathcal{D} = \{(T_1, y_1), ..., (T_n, y_n)\}$.

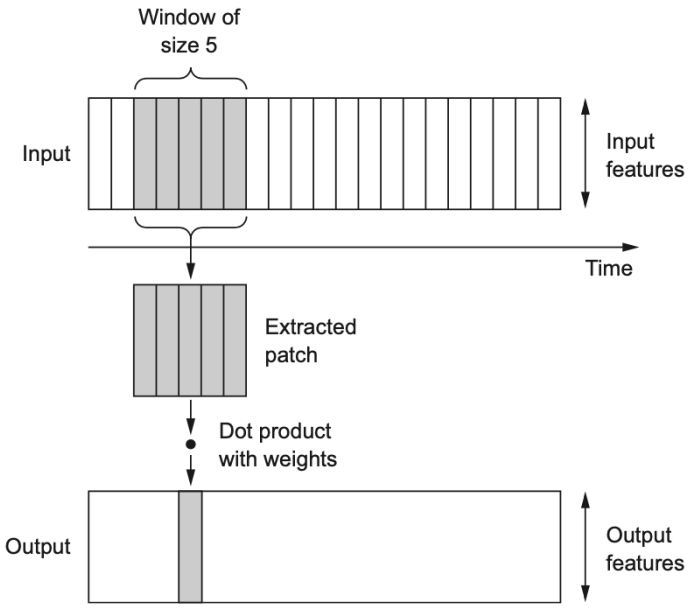


Figure 1: From Deep Learning with Python by Francois Chollet.

Definition 5: Let $(n, k) \in \mathbb{N}^2$, A multiouput-multiclass classication problem is an association of the form $\mathcal{D} = \{(T_1, y_{1,1}, .., y_{1,k}), ..., (T_n, y_{n,1}, .., y_{n,k})\}$, where for a fixed $m \in \{1, .., k\}$ and $\forall i \in \{1, 2, ..., n\}$ $y_{i,m}$ is a p-multiclass label for some $p \in \mathbb{N}$. Here n represents the number of availaible MTS in the dataset and k represents the number of labels one wants to attribute to each one of them.

2.3 Residual and convolutional neural networks

Residual and convolutional neural networks (CNN) are the basic blocks of the Inception model [2, 1]. In this part we explain briefly how they work on a MTS.

2.3.1 1D Convolution

2D CNN are widely used in Computer Vision for their ability to extract local patterns : in the case of images, patterns found in small 2D windows of the inputs. With this idea in mind, we thought about adapting this to our problem. In the same way, one can use 1D convolutions, extracting local 1D patches (subsequences) from sequences (see figure 1).

Definition 6: More formally, let $(n, k) \in \mathbb{N}^2$, k odd, and $T \in \mathcal{T}_1$ with length n . A kernel is a real vector of the form $K = [w_{-\lfloor \frac{k}{2} \rfloor}, ..., w_{\lfloor \frac{k}{2} \rfloor}]$. We define $\forall p \in \{1, ..., n\}$:

$$(T * K)[p] = \sum_{i=-\lfloor \frac{k}{2} \rfloor}^{\lfloor \frac{k}{2} \rfloor} T[p - i]w_i$$

When the index is out of bound, we define it to be 0, that is what we call the method of *padding*. If one wants to have an output of several dimensions i.e transforming T to have a tensor $T' \in \mathcal{T}_m$ for some $m \in \mathbb{N}$, one can reproduce the process defined above with m different kernels.

Finally, we would like to extend this process having as input $T \in \mathcal{T}_n$ and output $T' \in \mathcal{T}_m$ for some $(m, n) \in \mathbb{N}^2$, both of fixed size $j \in \mathbb{N}$.

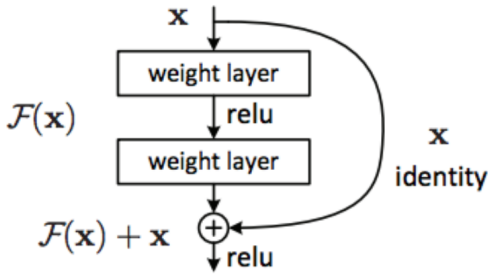


Figure 2: Residual neural network

Definition 7: $\forall k \in \{1, ..., n\}$ and some $p \in \mathbb{N}$ odd, we associate a real kernel K_k of size p and define $K := [K_1, ...K_n]$. We call T_i with $i \in \{1, .., n\}$, the i-th univariate time serie of T. Thus, we can define a new univarite time serie:

$$(T * K)[p] = \sum_{i=1}^{n} (T_i * K_i)[p]$$

Now to extend this result and obtain T', we repeat this process m times with m different vector of Kernels. At the end we get $n * m$ different kernels.

2.3.2 Residual neural network

When adding convolutions on top of each others, researchers faced the issue of a decrease in accuracy because of a loss of information instead of a gain with a vanishing gradient descent problem. Deep convolutional neural networks were first introduce in 2012 by Krizhevsky et al [4] for the image classification challenge Imagenet [6]. Their DCNN, named AlexNet, contained 8 neural network layers, 5 convolutional and 3 fully-connected. The intuition behind their function is that these layers progressively learn more complex features. In their paper [5], He et al. show this effect and propose residual neural networks (see figure 2).

Definition 7: Let $T \in \mathcal{T}_n$, and $\mathcal{F}$ a function applied to it such as a convolution and activation functions. $\mathcal{F}'$ a convolution with kernel of size 1 and output dimension to be the same as the one of $\mathcal{F}$. We define the residual to be:

$$T' = F(T) + F'(T)$$

We need $\mathcal{F}'$, because as explained above, most of the time a convolution will change the dimension of our MTS. Then in order to use this identity mapping and keep our input data, we need it to make a pointwise sum.

3 The EDF dataset

For this study, which was carried out in collaboration with EDF, we received a set of generated data containing information on 1000 households. From this large data set, we limited the study to important labels such as:

- Number of inhabitants: $[1, 2, .., 6]$
- Electric heater: $[0, 1]$
- Ecological investment or effort: $[confort, ecoresponsable, ecoresponsable++]$
- Type of accommodation: $[Maison\ individuelle, Logement\ collectif]$
- Housing area: $x \in \mathbb{R}^+$

Based now on their electricity consumption in 2018, we will try to build a model capable of correctly predicting the labels defined above and at the same time. We have used 15 univariate time series:

{total (W), Chauffage (W), Chauffe-eau (W), Lumiere (W), Veilles (W), Ordinateur (W), Seche-linge (W), Lave vaisselle (W), Lave linge (W), Aspirateur (W), Appareils de cuisine (W), Seche-serviette (W), Refrigerateur (W), Vehicule electrique (W), Groupe Television (W)}

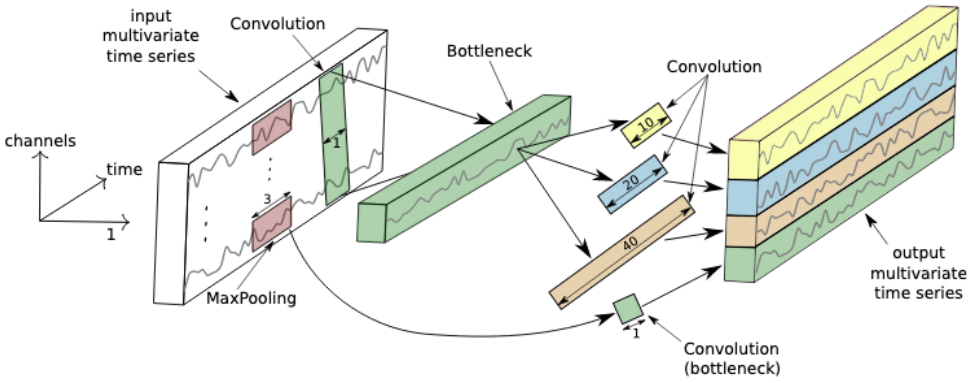


Figure 3: Inception module, image from [1]
. For simplicity they illustrate a bottleneck layer of size m = 1.

Each univariate time series represents electricity consumption every 30 minutes during the year with a length of 17520.

The format of the dataset was in the form of a csv file with 1000 lines describing each household. We then have a 1 csv file for each household with its electricity consumption. We then designed an interface that could easily read all this data and take in the information we needed. For example, the number of households, the use of interest and the label we want to provide. It also transforms these data into a machine readable one i.e. transforming words into numbers and columns into vectors that respect the format of definition 4.

As you have seen, the housing area is given by a real number and not by an integer to be classified. We consider it a challenge for us to build a model capable of regressing and classifying multiclass labels at the same time.

4 Model

4.1 Problem Description

We formulate the problem as a multiouput problem, the output is either a binary or multiclass classification but can also be a regression. Given a time series $T \in \mathcal{T}_{17}$ of length 17520 representing 1 year, we classify it with 5 different labels (see section 3). We added two dimensions, indeed we thought it would be helpful to give as information the day of the week and the moment in the day. We also divided by a scalar of 10000 each electricity consumption point.

4.2 Inception Module

In this section we refer to [2, 1]. The input is a time series $T \in \mathcal{T}_n$ which first passes by a bottleneck (a convolution with kernels of size 1) layer to reduce the dimensionality. This has been proven very efficient in MobileNets [7], a feature extraction network for object detection and image classification. Indeed it reduces computational costs. The output is a new time series $T' \in \mathcal{T}_m$ which is fed to three 1D convolutional layers of kernel size 10, 20 and 40 and gives $T'_1, T'_2, T'_3 \in \mathcal{T}_m$. In parallel T is also passed trough a Max Pooling layer of size 3 before being passed to another bottleneck layer to yield $T'' \in \mathcal{T}_m$. Finally, T'_1, T'_2, T'_3 and T'' are depth concatenated and form a new encoded time series $G \in \mathcal{T}_{4m}$. See figure 3.

4.3 Proposed Architecture

Originally, the InceptionTime [1] architecture is made of 6 different Inception module [2] on top of the other with a residual layer every 3 module. And the number of filter was set to 32, the m in the description above. This proposed architecture didn't fit our problem, since it contains a lot of parameters, which is not ideal for the little dataset we have, and would have require to increase the dimensionality of our time series, being originally 17. It leads to a fast overfitting and the model doesn't learn anything. When reducing m to 4, the problem was still remaining, even though an improvement was noticeable. We then decided to use only 1 Inception module. The model was able to learn and reach good accuracy, but not that high for every output. We then realised that the Inception module should be used as an encoder to transform our MTS to a new tensor that should be treated as a simple data and then we could use traditional basic machine learning algorithm. We propose to use an average pooling layer on the encoded MTS and

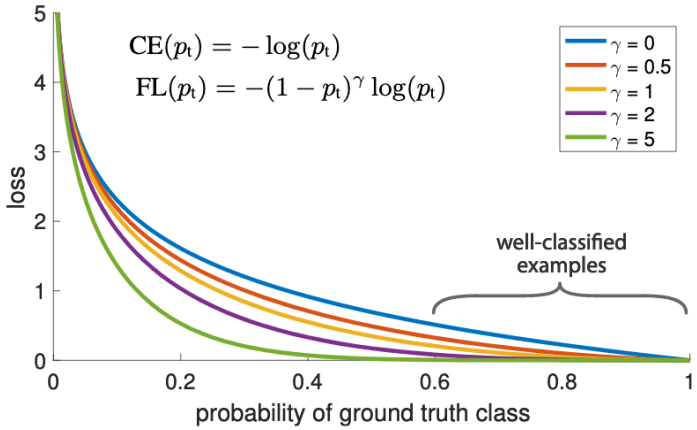


Figure 4: Focal Loss visualisation for different γ. Image from [3]

from it a simple fully connected layer with 128 neurons for each of the output, 5 in our case. The increase in accuracy was not negligible. We show the results in table 1.

4.4 Focal Loss

The focal loss [3] was presented in order to address class imbalance in the task of object detection. It reshapes the standard cross entropy loss such that it down-weights the loss assigned to well-classified examples.

In the following, we define p_t to be the predicted probability on the true label. The cross entropy is defined as follow:

$$\text{CE}(p_t) = -log(p_t)$$

The focal loss then add a modulating factor$(1 - p_t)^\gamma$ to the cross entropy loss, with tunable focusing parameter $\gamma \geq 0$. The focal loss is defined as follow:

$$\text{FL}(p_t) = -(1 - p_t)^\gamma \log(p_t)$$

When an example is misclassified and p_t is small, the modulating factor is near 1 and the loss is unaffected. As p_t tends to 1, the factor goes to 0 and the loss for well-classified examples is down-weighted. The focusing parameter γ smoothly adjusts the rate at which easy examples are down-weighted. When $\gamma = 0$, FL is equivalent to CE, and as γ is increased the effect of the modulating factor is likewise increased. Intuitively, the modulating factor reduces the loss contribution from easy examples and extends the range in which an example receives low loss. We chose $\gamma = 2$ for our study. See figure 4 for a concrete visualtion of the focusing parameter.

5 Experiments

5.1 Metrics

In this section we present the different experiments we have done. But but first we define on what is the metric we used to compare the models, apart the accuracy.

We use the precision, it measures how correct the model is when it predicts one specific class. The formula is:

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

We also use the recall which measures the proportion of correct labels on a given class.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Model	Inhabitants			Effort			Electric heater			Accommodation			Area	
	Pr	**Re**	**F_1**	**Pr**	**Re**	**F_1**	**Pr**	**Re**	**F_1**	**Pr**	**Re**	**F_1**	**MAE**	**RMSE**
LSTM_dense, FL	0.23	0.24	0.22	0.36	0.35	0.28	0.98	0.96	0.97	0.64	0.63	0.59	30.0	1271
	0.46	0.55	0.49	0.37	0.33	0.27	0.97	0.97	0.97	0.68	0.59	0.58		
Inception Time, FL	0.66	0.56	0.56	0.38	0.42	0.36	0.94	0.93	0.93	0.68	0.69	0.67	87.0	9496
	0.82	0.81	0.80	0.38	0.39	0.34	0.84	0.94	0.94	0.71	0.67	0.68		
Ours,without FC, FL	0.48	0.49	0.47	0.42	0.44	0.40	0.98	0.95	0.96	0.65	0.66	0.65	74.4	6792
	0.76	0.76	0.75	0.42	0.42	0.39	0.97	0.97	0.97	0.68	0.66	0.66		
Ours, CE	0.67	0.67	0.66	0.45	0.45	0.40	0.99	0.98	0.99	0.80	0.80	0.80	21.1	667.1
	0.91	0.90	0.90	0.45	0.43	0.39	0.99	0.99	0.99	0.81	0.81	0.81		
Ours, FL	0.66	0.64	0.65	0.38	0.40	0.36	0.95	90.0	0.92	0.73	0.74	0.71	22.2	736.8
	0.88	0.88	0.88	0.38	0.39	0.35	0.93	0.93	0.92	0.76	0.72	0.72		

Table 1: Comparison of different models described in 5.4. For each model we have the macro average and the weighted average as described in 5.1, when it is a classification task

Model	Inhabitants			Effort			Electric heater			Accommodation			Area	
	Pr	**Re**	**F_1**	**Pr**	**Re**	**F_1**	**Pr**	**Re**	**F_1**	**Pr**	**Re**	**F_1**	**MAE**	**RMSE**
Transfer Learning, CE	0.59	0.58	0.58	0.45	0.44	0.40	0.95	0.90	0.92	0.73	0.73	0.73	22.85	756.3
	0.86	0.86	0.86	0.45	0.42	0.39	0.94	0.93	0.93	0.75	0.74	0.74		
Transfer Learning, FL	0.66	0.64	0.65	0.38	0.40	0.36	0.95	0.90	0.92	0.73	0.74	0.71	23.28	799.4
	0.88	0.88	0.88	0.38	0.39	0.35	0.93	0.93	0.92	0.76	0.72	0.72		
Data augmentation, CE	0.81	0.76	0.76	0.53	0.51	0.50	0.96	0.94	0.95	0.78	0.77	0.77	22.16	710.16
	0.94	0.94	0.93	0.54	0.51	0.51	0.95	0.95	0.95	0.79	0.79	0.79		
Data augmentation, FL	0.82	0.78	0.79	0.57	0.57	0.55	0.93	0.93	0.93	0.76	0.78	0.77	20.99	666.5
	0.96	0.95	0.95	0.57	0.56	0.55	0.93	0.93	0.94	0.79	0.77	0.77		

Table 2: Metrics on test data of 3 months length. The first half is our proposed architecture trained on 1 year and used as transfer learning. And the second one is our architecture directly trained with the data augmentation described in 5.2. For each model we have the macro average and the weighted average as described in 5.1, when it is a classification task.

Finally to gather recall and precision as one metric, we can use the F_1 score, defined as follow:

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Since we are faced with a multiclass problem, we have two choices. Either we take the average of the metrics measured on each class, or we take a weighted average based on their representation in the dataset.

5.2 Data Augmentation

We remarked by cutting each time series in 4 i.e every 3 months or by seasons lead to a more stable training and more accurate results on unrepresented labels. We conclude that our model is able to understand households composition and behavior with just few months. We can't compare it with the model trained on 1 year as the number of data is not the same, but it shows that if one doesn't have a large dataset it can cut its MTS to create new one and get a better precision. See table 2.

5.3 Ensemble method

Like in InceptionTime [1] we propose to leverage the efficiency of Ensemble Methods. Indeed, the model exhibits high standard deviation in accuracy. In their paper they explain that they believe that this variability comes from both the randomly initialized weights and the stochastic optimization process itself. As rather than training only one potentially very good or very poor, instance of the proposed architecture, we decided to leverage this instability through ensembling. The following equation explains the ensembling of classification made by a network with different initialization:

$$\hat{y}_{i,c} = \frac{1}{n}\sum_{j=1}^{n} p_c(T_i, \theta_j) \ \ | \ \forall c \in [1, C]$$

With $\hat{y}_{i,c}$ denoting the ensemble's output probability of having the input time series T_i belonging to class c, which is equal to the probability output p_c averaged over the n randomly initialized models, and θ_j representing the $j-$th model. Like in InceptionTime [1] we opt for ensembling 5 different models. As mention in their paper, the ensembling process help in the control of a little dataset like the one we have. Indeed it helps leveraging the variance of the error, which is likely to reduce when increasing the training set's size. For the regression task, we can simply take an average of the predicted values.

5.4 Results

5.4.1 Training

We trained the model with a batch size of 64, the Adam optimiser with a very low learning rate of 0.0001. A little increase in the learning rate leads to a fast overfitting or an unstable training. We split the dataset to have, 65% for the training, 20% for the validation and the remaining 15% for the tests. Each set contains approximately the same distribution of different labels.

We show the results of the original InceptionTime, our architecture without the fully connected layer at the end i.e without taking advantage of the encoded data on the focal loss. And our architecture with cross-entropy and the focal loss. We also compare the results with a basic RNN architecture using an LSTM and fully connected layer. See table 1. We didn't experiment on traditional machine learning classifiers such as Shapelets transform algorithms [**?**], KNN with the dynamic time wrapping algorithms or TimeSeriesForest because either they were not suitable with out multiouput-multiclass regression problem or were too computationally expensive on our very long time series.

The final model trained on 1 year had been implemented so that it can take as input a time series of any length via transfer learning. For example if EDF, can only have the consumption of a client for 5 months, it can use our model without any issue. The accuracy will be of course a bit reduced. We show the experimentation of a model trained on 1 year and tested on data of 3 months i.e a season. See table 2.

5.4.2 Observation

We observe in Table 1 and Table 2, that for this task of multiouput-multiclass, regression problem our model outperforms existing state of the art methods, even if it is still good mentioning, it is based on the Inception module presented in [2, 1]. Indeed, although InceptionTime's results are not so bad for classification tasks, they are even better than our architecture without the fully connected layers, it cannot do both tasks, i.e. classify and regress. This shows the importance of using the Inception module as an encoder before passing it to a decoder instead of using it directly before the activation function, softmax or linear in our case.

In terms of total length, there is no significant difference between the FL and the CE, with the CE results even slightly exceeding those of the FL. But using the two models with a transfer learning on the MTS cut in 4, i.e. with more data,

we can see that the macro average decreases significantly for the CE compared to the FL. This shows how robust the FL is.

Finally, we can see that the data augmentation is very useful, in fact the measurements are improved on each label and it is now the FL which is slightly higher than the CE. This can be explained by the fact that the unbalanced data were not present enough in the original dataset and even the focal loss could not make a big difference. Nevertheless, the FL helped to learn about the existence of unbalanced classes, as the transfer learning task showed. And as the data set increased, the focal loss was able to do its job and increase the accuracy compared to the CE.

6 Conclusion and Further Discussion

Through this study, we showed that it is possible to learn jointly about a regression and multiple class problem. We proposed to use the Inception module as an encoder for the MTS. It proved to be very efficient on the EDF dataset. A key point, which could be further investigated, is the use of a single fully connected layer as a decoder. We could try to use a traditional machine learning classifier for each of the labels and testify to the quality of the encoded data.

Recently, the Transformer [8] architecture based on attention layer [9] outperformed most of the tasks in NLP. MTS can be closely related to NLP tasks since it is also an embed sequence. In another study we could try to use the Transformer encoder instead of the Inception module and see how good attention would be on electricity consumption.

References

[1] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller and François Petitjean. Inceptiontime: Finding alexnet for time series classification. arXiv:1909.04939, 2019.

[2] Szegedy C, Ioffe S, Vanhoucke V, Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI* Conference on Artificial Intelligence, 2017.

[3] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He and Piotr Dollár. Focal Loss for Dense Object Detection In *ICCV*, 2017.

[4] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. 2012.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. 2015.

[6] O.Russakovsky, J.Deng, H.Su, J.Krause, S.Satheesh, S.Ma, Z.Huang, A.Karpathy, A.Khosla, M.Bernstein et al. Imagenet large scale visual recognition challenge. arXiv:1409.0575, 2014.

[7] Andrew G. Howard, Menglong Zhu, Bo Chen,Dmitry Kalenichenko, Weijun Wang, TobiasWeyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural net-works for mobile vision applications. CoRR,abs/1704.04861, 2017.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia. Attention is all you need. Polosukhin In NIPS. 2017

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Attention Neural machine translation by jointlylearning to align and translate. CoRR, abs/1409.0473, 2014

[10] Hills J, Lines J, Baranauskas E, Mapp J, Bagnall. Classification of time series by shapelet transformation. Data Mining and Knowledge Discovery 28(4):851–881. 2014.

A Appendices

A.1 Data distribution

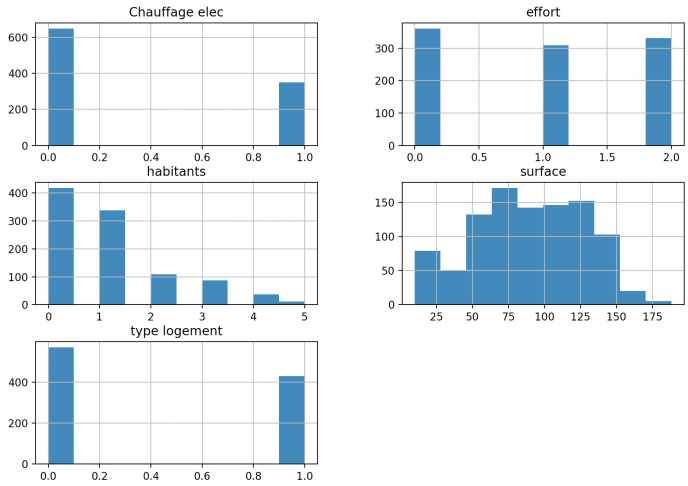
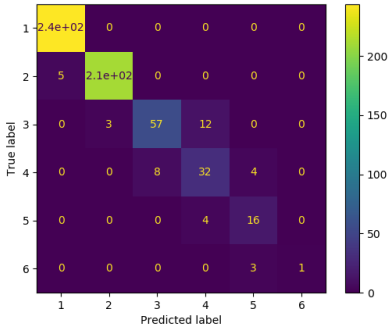
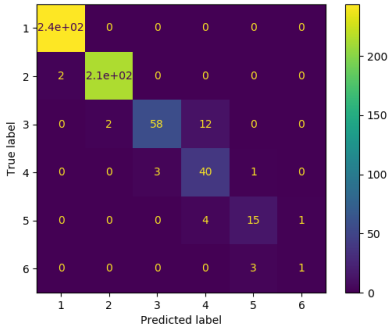


Figure 5: Distribution of the classes in the EDF dataset

A.2 Confusion matrices with the model trained with data augmentation comparing the FL and CE

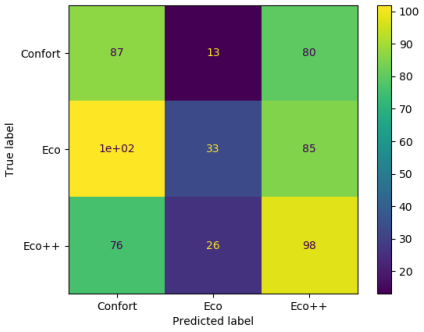


(a) CE

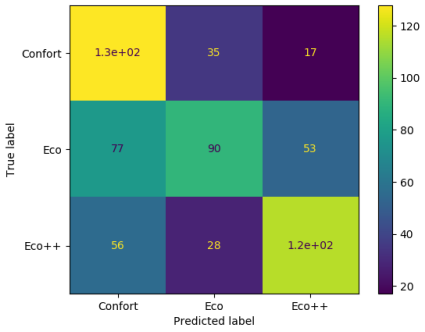


(b) FL

Figure 6: Class: number of inhabitants



(a) CE



(b) FL

Figure 7: Class: ecological effort