Attention mechanism on disentangled contextual factor representations for time series generation

Perrine Cribier-Delande[1,2], Raphael Puget[2] and Vincent Guigue[1], Ludovic Denoyer[1] *[1]

[1] MLIA, Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
[2]Renault, DEA-IR, Technocentre, 1 avenue du Golf 78084 Guyancourt , France

12 novembre 2020

Résumé

Learning & predicting sequence dynamics remain an outstanding research issue. Many application domains ranging from sales forecast to smart cities face challenging problems. Most of the approaches in the literature have weaknesses regarding the modelling of the time series context. We choose to tackle long-term forecasting with a focus on learning disentangled representations of contextual factors. We turned the forecasting problem into a generative problem : given spatial, temporal and other factors, we are able to generate the corresponding time series. Not only are we able to predict the sequence dynamics for a specific day or location, but we also show that our architecture is very robust to the introduction of outliers in the training set. We propose different variants of our approach, including an encoder-decoder architecture that enables us to extrapolate all contextual combinations corresponding to a new factor observed only once. This ambitious formulation raises theoretical questions about how to aggregate information associated with the different factors. This article examines several options and highlights the value of introducing an attention mechanism to improve the effectiveness of this critical step.

Mots-clef :Representation Learning Factor Disentangling Time Series Forecasting

1 Introduction

The domain of disentangled representations learning, where each learned feature matches a real life meaningful concept, has been getting more and more attention in the last years. Applied to the field of raw time series, representation learning architectures offer new opportunities to improve both dynamics modelling and understanding. The range of real life applications that can benefit from a better modelling of time series is wide, going from finance and market prediction to user modelling and recommender systems, from smart cities and transportation systems to supply chain and sales prediction.

In many of these domains, the forecasting issue must be considered in parallel with the cost of data acquisition : collecting the data needed to learn models and extract sufficient information can be very expensive. For instance, in the field of intelligent transportation systems, placing sensors in stations and/or vehicles is very costly ; this is why the datasets often do not cover all situations that happen in the real world. We often have precise measurements of a quantity only during a short period or at a specific location.

Time series are associated with various problems : the most classic is the question of forecasting, which has long been the subject of much attention. The first statistical models tackling this issue are old [1]. Later on, many famous machine learning architectures have been declined for it [21, 22]. Deep learning clearly represents a major breakthrough in the field with recurrent architectures [13, 4] that offer new opportunities [25]. Data imputation is also a standard issue in time series. It can be done inside a series but it is a more challenging task at the signal level. For example, given a context or content, trying to generate the complete series [6]. On this point, we draw inspiration from disentangled representation learning architectures to generate realistic new signals [3]. We also want to work

*This work is partially supported by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 780754, "Track & Know".

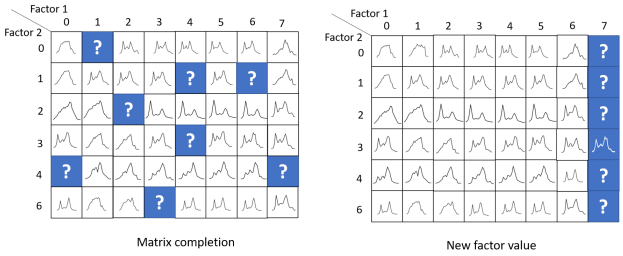


FIGURE 1 – A representation of our dataset, organised according to two factors of variation. On the left, our first task : matrix completion, on the right our second task : new factor value. A series in white background is in the training set while one in dark blue background is only available at inference time (in the testing or validation set). The question marks '?' are the series we are evaluating on.

with contextual factors that have not been observed during training. This additional constraint leads us to study few-shot learning approaches such as [24]. In this article, we aim at tackling both time series forecasting and data completion but do not consider time series classification.

Our proposal relies on a general encoder-decoder architecture. The encoder acts as factor extractor : it enables us to represent the explicit contextual factors associated with the time series –for instance, the location and day of the record. The decoder is the generation module : from the different factors it produces a time series. Depending on the shape of the decoder, we are limited to fixed-size signals (with MLP and CNN) or able to deal with any time series (RNN). The latter setting also allows us to tackle classical forecasting by feeding the recurrent network with ground truth past values to predict the future ones. We exploit this general architecture in two different tasks : (1) the generation of missing series from the dataset corresponding to contextual factors that are present in the training set. (2) The generation of time series corresponding to factor value never seen in the training set and only seen once at inference time. In the specific case where we consider only two factors –spatial and temporal for instance–, the first task can be seen as a matrix completion where each cell is a time series. The second task would then correspond to adding a line or a column in the dataset from a single new instance. A schematic representation of our datasets and the associated tasks is shown in Figure 1.

At inference step, some specific problematics arise on how to aggregate the representations of the latent factors. Indeed, depending on the encoder architecture, we may obtain different representations associated with a single factor. To generate a new time series including this factor, we investigate several strategies to aggregate those representations. To summarise, we can average representations in the latent space or average generated time series in the temporal domain. We show that the first option can be highly effective when combined with an attention mechanism [5]. In particular, when we introduce some noise in the dataset by switching a fraction of the factor labels, we demonstrate that our attention enables us to minimise the impact of the outliers and to preserve a nearly optimal performance.

Our architecture has several advantages. The disentanglement part in our model allows us to construct meaningful latent spaces ; it defines a topology for each factor in their latent space. It allows us to efficiently generate the missing series of our dataset. It also enables us to generate new time series corresponding to a new value of factor collected after training. Our model can then extrapolate all the series corresponding to this new factor associated with the observed value of all the other existing factors. For example, from a single instance collected on a new location, our model can generate series for this location associated with all observed temporal contexts.

The article is classically organised as follows : first, we describe related works combining disentanglement, data generation and one-shot learning with attention. Then, we present our baselines and models and we detail the experimental setting. We conclude with several analyses on the results from the experimental section.

2 Related Work

Historically, statistical methods such as ARIMA and Box-Jenkins [1] have achieved great success in time series modelling via forecasting. Their weakness arises when large amounts of non-linear data have to be analysed. In comparison, this is one of the strength of deep learning which is increasingly used in such cases. Recurrent Neural Network (RNN) were designed specifically to handle sequences. Robust implementations such as Long Short Term Memory(LSTM) and Gated Recurrent Units (GRU) are classically used to model time series [13, 4]. Both these techniques have the ability to *memorise* the past trend and recent changes, which makes them particularly suited to handle time series at multiple scales. They have been used with great success in very various domain such as Natural Language Processing –to learn contextualised word re-

presentations [19]–, video –to predict the next frame for example [23]–, sound processing –to predict the continuation of a piece of music [15]– or sale prediction [9]. Spatio-temporal data processing also benefits from deep architecture flexibility to model complex dependencies or contextual factors [16, 25, 7].

Broadly speaking, the recent work on disentanglement has been using two techniques : the Variational Autoencoder –VAEs– [20, 14] and the Generative Adversarial Network –GAN– [11]. Although focused at first on computer vision tasks, the field has been largely broadened to tackle time series, video, speech, sounds and many other. For the VAE approach, the β-VAE [12] improved on the classical VAE and has become a classic of the domain. Slightly modifying the classical loss of the VAE, by adding weight β to the KL-divergence forces the latent factors to be disentangled. [17] then showed that the β-VAE controls the latent overlap of the disentangled factors and how the choice of prior can impact the decomposition in factors. Using the adversarial technique, InfoGAN [3] improves on GAN [11] by imposing a structure on the latent space through the minimisation of the mutual information between generated samples. This allows them to generate images conditioned on the latent variables. For example, they can generate the image of a specific number instead of a random one.

VAE can be used to learn universal embeddings to represent time series in an unsupervised manner [10]. But mixing time series analysis and disentanglement leads to investigate new approaches to model the temporal aspects of an input. The focus thus shifted to disentangling the dynamic (the temporal component) from the content (that does not have a temporal component). For speech, the content can be the sentence being pronounced and the dynamics the voice saying it. This allows [15] to perform "feature swapping" via a new generative model, based on VAE, that explicitly learns to disentangle content and dynamic and generate speech with a voice different from the one of the original speaker. Using adversarial training [23] and [8] focused on predicting the next frame of a video by using different auto-encoders architectures and disentangling dynamics and content. On one hand, [23] learned to represent the motion by comparing two consecutive frames. On the other hand, [8] designed a new training objective to learn disentangled representations of content and dynamics on only one frame.

Improving the modelling of dynamics and contextual factors leads to a better understanding of the object to process. This offers some opportunities to tackle one-shot or few-shot learning issues where we aim at learning or adapting a model from very little data [18]. The matching neural networks exploit an attention mechanism to measure a similarity between the representation of a new class and the already learned classes [24], similarly to what is done with Siamese networks [2] but with an asymmetric architecture so as to have two different representation functions.

This eclectic overview of the literature is a good illustration of our objective : combining recent advances from various fields to tackle new tasks associated with time series.

3 Model

In this article, we restricted ourself to datasets made of temporal series that are organised according to two factors of variation [1]. Each factor can have several values. We call each possible value of a factor a factor class, and the integer associated to it is called class value. The factor 1 class can take values in $[0, u]$ and the factor 2 class can take values in $[0, v]$. In this paper $\mathbf{x}_{i,j}$ denotes the series that correspond to the class value i for the factor 1 and j for the factor 2 ($i \in [0, u], j \in [0, v]$). Each temporal series is of fixed size T such that $\mathbf{x}_{i,j} \in \mathbb{R}^T$ but the model, in its recurrent version, can deal with series of different lengths. The series are indexed through a mask m such that $m_{i,j} = 1$ if $x_{i,j}$ is observed (and is in the training set), and $m_{i,j} = 0$ if $x_{i,j}$ is not observed (in the validation or test set). The Figure 1 shows a representation of this setting for two tasks.

In one of our experimental setting, to add some noise, we mislabelled some series in the training set, meaning that at least one of the factors class does not match the real ground truth of the series. In Figure 1 this corresponds to swapping a fraction of the series with white background. In practice, we randomly sample 15% of the training set and cycle the labelled factor so that the examples would have at least one mislabelled factor.

Our architectures are based on an encoder-decoder architecture. They all contain an encoder and a decoder. Between them, an attention mechanism can be used. In the following, we denote the encoder function associated with the factor 1 and 2 respectively e_1 and e_2. The decoder is denoted d.

— **Encoder** : takes as input a time series $\mathbf{x}_{i,j}$ and outputs the disentangle representation of each factor, noted $\mathbf{z}_i^{(1)}$ and $\mathbf{z}_j^{(2)}$. Our encoder can take

1. Our model can handle an arbitrary number of factors. However, for the sake of clarity, it is easier to demonstrate its ability on a two-factor dataset.

several forms. In this paper, we use either neural networks (MLP, CNN or RNN) or matrices of embeddings. Using embeddings leads to a simple transductive formulation that is able to tackle the matrix completion task (Fig. 1, left). Obviously, each class value corresponds to a unique representation. Exploiting a neural network encoder asks the question of factor representation uniqueness ; indeed, denoting e_1 the encoder associated to factor 1, we are going to obtain different values for different inputs : $e_1(\mathbf{x}_{i,j}) \neq e_1(\mathbf{x}_{i,k})$. Thus, we face an aggregation issue to determine $\mathbf{z}_i^{(1)}$. It will be discussed in depth in this section (see sous-section 3.2). This point leads us to use the intermediate notation $\mathbf{z}_{i,j}^{(1)}$ corresponding to the output of $e_1(\mathbf{x}_{i,j})$ even if $\mathbf{z}^{(1)}$ is supposed to be independent from j.

— **Decoder** : from the factor representations, we generate a new time series using three neural architectures : MLP, CNN and RNN. The latter is able generate arbitrary length signals whereas the first two are limited to fixed-length signals. The generation process is straightforward from embeddings, however, it is more complex when combined with neural encoders. We investigate three options to generate $\hat{\mathbf{x}}_{i,j}$: (1) aggregating signals generated from $e_1(\mathbf{x}_{i,.})$ and $e_2(\mathbf{x}_{.,j})$ in the temporal domain, (2) aggregating embeddings at the output of the encoding process and generate a single time series, (3) learning an attention process over the factor representation in an end-to-end manner so as to build a smart aggregation of the factors coming from the encoded signals.

3.1 Training procedure

As it stands, the model could devolve into a simple auto-encoder where all the information about the series is encoded into only one of the factors. Indeed, there is no guarantee that the information pertaining to the first factor (respectively the second factor) is encoded in $\mathbf{z}^{(1)}$ (respectively $\mathbf{z}^{(2)}$). To enforce disentanglement, we use a specific training procedure. Instead of using a single series as input and target, we sample three series. The first one, noted $\mathbf{x}_{i,j}$ is our target. The second one must have the same value for the first factor as our target, we denote it $\mathbf{x}_{i,k}$. The third one, must have the same value as our target for the second factor, we denote it $\mathbf{x}_{k',j}$.

We then encode only the second and third series $\mathbf{x}_{i,k}$ and $\mathbf{x}_{k',j}$ to estimate the targeted one.

$$\hat{\mathbf{x}}_{i,j} = d(e_1(\mathbf{x}_{i,k}), e_2(\mathbf{x}_{k',j})) \quad (1)$$

The error between $\hat{\mathbf{x}}_{i,j}$ and $\mathbf{x}_{i,j}$ is back-propagated through both the decoder and the encoders. MLP and CNN rely on MSE for the whole signal while RNN error is computed for each value of the time series.

The three possible architectures of our model are illustrated in Figure 2, where we show the reconstruction of a time series from embedding matrices. The training procedure optimises both the decoder and the factor value representations. Figure 3 introduces a neural encoder that enables us to tackle the new task depicted in Fig. 1, right.

Indeed, the ability to encode new series in the inference step offers new applicative opportunities. As explained above, this architecture requires an aggregation step that can be improved using an attention mechanism to focus on factor value representations that are close to the targeted case. This latter architecture is represented in Figure 4.

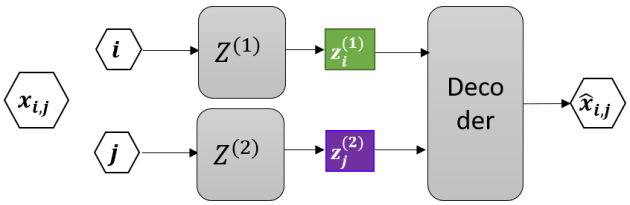


FIGURE 2 – Our simplest encoder-decoder model based on unique embeddings associated with each factor value.

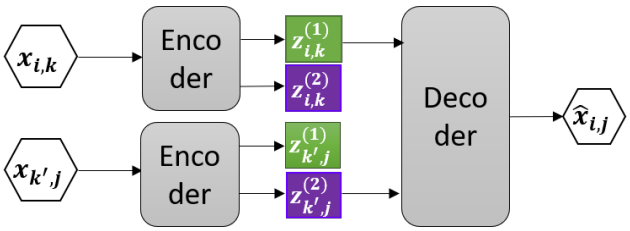


FIGURE 3 – Our encoder-decoder model with a neural encoder. Encoder and decoder are always based on the same architecture and we consider MLP, CNN and RNN.

3.1.1 Encoder

Our encoder learns to represent each factor in the latent spaces. For two factors, we have two latent spaces $\mathbb{R}^{\ell_1}$ and $\mathbb{R}^{\ell_2}$ and two encoders. Note that ℓ_1 and ℓ_2

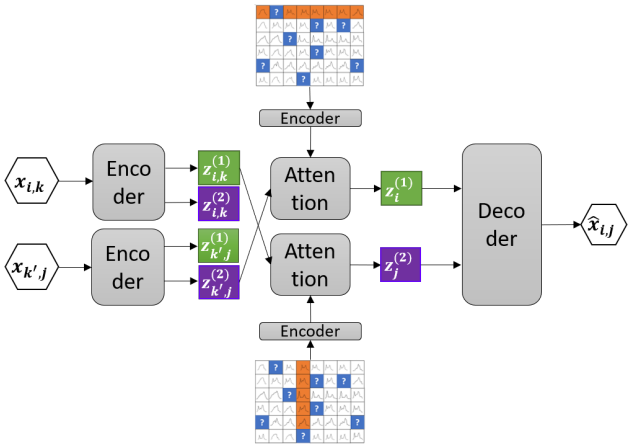


FIGURE 4 – Our encoder-decoder model with the attention mechanism

have to be optimised separately : their optimal values are closely linked to the number of factor values. In the case where our encoders are matrices of embeddings, we denote these matrices $Z^{(1)}$ and $Z^{(2)}$. If they are neural networks, they are denoted e_1 and e_2. e_1, e_2 and d are always based on the same architecture and the encoders corresponding to different factors can share some parameters, like the first few layers of a MLP.

3.1.2 Decoder

Our decoder d learns to reconstruct a series $\hat{x}_{i,j}$ from the representation of the corresponding factor. It can rely on the three previously mentioned architecture : MLP, CNN or RNN. Only the latter is able to generate arbitrary length signals, but we do not exploit this property in this article.

When we do not introduce attention, the learning procedure is very simple and we do not face the aggregation issue : we rely on a triplet of time series and back-propagate the error made on $\hat{x}_{i,j}$ through both the encoder and the decoder. Then, the difficulty is carried over to the inference step.

With the attention mechanism, the procedure is a little more complex, we give the details below.

3.1.3 Attention mechanism

As explained above, this part is optional : it aims at learning which factor value representations should be used or not to infer a new time series. The representation $\mathbf{z}_i^{(1)}$ is based on the aggregation of all the possible representations of $\mathbf{z}_{i,\cdot}^{(1)}$ that can be computed from the training set with respect to a query : $\mathbf{x}_{i,j}$. First, each example in the training set for which the factor 1 class value corresponds to i, is encoded through e_1, giving us a set of representation noted $\mathbf{z}_{i,\cdot}^{(1)}$. Our attention mechanism aims at creating the optimal linear combination of these representations for the specific instance of $\mathbf{x}_{i,j}$. Thus, we introduce an attention function that will outputs a weight that represents how important this representation is for this query. In practice :

$$\mathbf{z}_i^{(1)} = \sum_k w_k \mathbf{z}_{i,k}^{(1)}, \qquad w_k = \mathbf{z}_{i,k}^{(1)} \cdot A_{1\to 2} \cdot e_2(\mathbf{x}_{i,j}) \quad (2)$$

According to our experiments, the attention function should measures the interest of $\mathbf{z}_{i,k}^{(1)}$ with respect to the second factor associated with $\mathbf{x}_{i,j}$. The $A_{1\to 2}$ and $A_{2\to 1}$ parameter matrices are learnt during the training procedure. The matrices dimensions depends on ℓ_1 and ℓ_2, the dimensions of the latent spaces chosen for the two factors.

The Figure 5b illustrates the output of our attention mechanism on an example. All the representations of a factor class have been assigned a weight (shown in levels of grey).

3.1.4 Optimisation of parameters & hyper-parameters

To optimise the computation time during training with the attention mechanism, we adopt a method inspired by what is done in most implementations of GAN. Each batch contains a set of examples that all have one factor value in common. This allows us to only have to encode one set of examples to perform the attention query on. For instance, for a batch of examples with the same factor 1 value, we only have to encode all the examples matching this value for factor 1 and perform one query by example in the batch. We alternate the common factor at each batch.

The parametrized functions d, e_1 and e_2, the attention matrices $A_{1\to 2}$, $A_{2\to 1}$ and the embeddings matrices $Z^{(1)}$, $Z^{(2)}$ are learned jointly (end-to-end) via gradient descent and back-propagation. We trained using the classical MSE loss between the target and the reconstructed series : $MSE(\mathbf{x}, \hat{\mathbf{x}}) = ||\mathbf{x} - \hat{\mathbf{x}}||^2 = \sum_t (x[t] - \hat{x}[t])^2$.

The hyper-parameters of this model were optimised on the validation set, using a random search. The hyper-parameters are :

— ℓ_1 and ℓ_2, the dimension of the latent space of each factor. This is very dataset-dependant because it is affected by the possible number of values. We performed a search on the following possible values : 8, 16, 32, and 64.

— The number and size of the layers of the encoder and the decoder. For each architecture we experimented with MLP, CNN and RNN. We notice that the introduction of ℓ_1 * ℓ_2 parameters for the attention mechanism always coincide with a reduction of the size of the overall optimal architecture, whatever the chosen neural model.
— The learning rate : we perform our search with value 0.0001 and 0.001
— The optimiser : we performed our search with on classical SGD optimiser and Adam optimiser. We always use a mini-batch size of 32.

As an example, for the STIF dataset (presented below, see sous-sous-section 4.1.1), where there are 88 values for factor 1 and 297 values for factor 2, we found that the optimal set of hyper-parameters were : $\ell_1 = 16$ and $\ell_2 = 32$, a learning rate of 0.001 and the Adam Optimiser. The number and sizes of the layer was dependant on the task. For the first task, without noise but with attention we obtain : for the MLP architecture, an encoder with three layers of size 128 and a decoder with 2 layers of size 256. For the CNN architecture both encoder and decoder have 2 convolutional layers with 32 filters of size 5 and for the RNN architecture both encoder and decoder have two layers of LSTM cells with a hidden size of 64.

3.2 Inference

We distinguish the inference step for the two tasks illustrated in Figure 1.

3.2.1 Matrix completion task

At inference time, in the matrix completion task, we want to estimate a new series $\hat{\mathbf{x}}_{i,j}$, considering that there exist at least two series $\mathbf{x}_{i,k}$ and $x_{k',j}$ that are observed. Depending on our settings, there are two possibilities : If we use embeddings matrices, the prediction is straightforward : $\hat{\mathbf{x}}_{i,j} = d(\mathbf{z}_i^{(1)}, \mathbf{z}_j^{(2)})$. If we do not use embeddings matrices, then : $\hat{\mathbf{x}}_{i,j} = d(e_1(\mathbf{x}_{i,k}), e_2(\mathbf{x}_{k',j}))$. However, we saw that the latter option is more complicated. We can choose any k and k' or aggregate the different possibilities. We consider three options :

— **Embedding average.** In the first method, we average embedding in the latent space.Given $\mathcal{N}$ and $\mathcal{M}$ the sets of sequences corresponding respectively to the targeted factor i and j :

$$\mathbf{z}_i^{(1)\star} = \frac{1}{|\mathcal{N}|} \sum_{k \in \mathcal{N}} e_1(\mathbf{x}_{i,k})$$

$$\mathbf{z}_j^{(2)\star} = \frac{1}{|\mathcal{M}|} \sum_{k \in \mathcal{M}} e_2(\mathbf{x}_{k,j})$$

$$\hat{\mathbf{x}}_{i,j} = d(\mathbf{z}_i^{(1)\star}, \mathbf{z}_j^{(2)\star})$$

— **Series average.** In the second solution, we encode each series that match the factor 1 class values i or the factor 2 class value j and decode all possible combinations of these representations and then average on the reconstructed set of series.

$$\hat{\mathbf{x}}_{i,j} = \frac{1}{|\mathcal{N}||\mathcal{M}|} \sum_{k \in \mathcal{N}, k' \in \mathcal{M}} d(e_1(\mathbf{x}_{i,k}), e_2(\mathbf{x}_{k',j})) \tag{3}$$

— **Attention mechanism.** The last solution consists in learning the aggregation through the attention process. In that case :

$$\mathbf{z}_i^{(1)\star} = \frac{1}{|\mathcal{N}|} \sum_{k \in \mathcal{N}} a_1(j, \mathbf{x}_{i,k}) e_1(\mathbf{x}_{i,k})$$

$$\mathbf{z}_j^{(2)\star} = \frac{1}{|\mathcal{M}|} \sum_{k \in \mathcal{M}} a_2(i, \mathbf{x}_{i,k}) e_2(\mathbf{x}_{k,j})$$

$$\hat{\mathbf{x}}_{i,j} = d(\mathbf{z}_i^{(1)\star}, \mathbf{z}_j^{(2)\star})$$

where a_1 denotes the attention function computed on any time series including factor j.

3.2.2 New factor value

In the case where i has never been observed during training and we are provided $\mathbf{x}_{i,j}$ (new factor value task), we aim at predicting $x_{i,k}$ for k in $[0, v]$. Obviously in this task, the embeddings matrices cannot be used, because i was never seen during training. We then only have two settings to investigate : with and without attention mechanism. In both situations, the combinatorial is strongly reduced with respect to the matrix completion setting as $|\mathcal{M}| = 1$. Thus, the aggregation is performed on a single factor and the attention is computed with respect to the only available sample : $\mathbf{x}_{i,j}$.

Despite the fact that this task is much more complicated, calculations are simplified by the lack of data.

4 Experiments

Our architectures are tested on four different datasets. They are comprised of series of 24 timesteps series (so $T = 24$). For each one of them, the factor 1 is

the temporal context (the day on which the data was collected) and the factor 2 is the spatial context (the location where it was collected).

4.1 Datasets

4.1.1 Smart card dataset (STIF)

This first dataset, denoted STIF, was collected by Ile-de-France Mobilites (formely called STIF). It measures the number of smart card validation in the Parisian subway network (299 stations). We aggregated this data hourly so that each time series in the dataset is the number of smart card validations in one station for one day. It was collected between the 1^{st} of October and the 31^{st} of December 2015. This period includes two events that disturbed the collection of data : Paris hosted the COP21 (29 and 30 of November), during which the transportation system was free. The Bataclan terrorist attacks occurred on November the 13^{th}. These three days were removed from the dataset as well as two stations that were undergoing planned work at the time. This dataset has 88 factor 1 values and 297 factor 2 values.

4.1.2 Energy Consumption

This dataset comes from Kaggle [2] and consists of the measurements of hourly consumption of energy in the East of the United States. It was collected by PJM Interconnection LLC [3], a regional transmission organisation in the US that supplies energy to a large portion of the North East of the United States. The energy is distributed across several regions and each one reports its hourly consumption. The regions are used as our second factor. The data was collected between the 2^{nd} of June 2013 and the 2^{nd} of August 2018. A few days had recording errors and were dismissed. This dataset has 1877 factor 1 values and 10 factor 2 values.

4.1.3 Air quality

These two datasets are the split of a Kaggle dataset [4]. The original dataset was collected and opened by the city of Madrid [5]. It consists of hourly measurements of many pollutants in the air of Madrid. We selected two of them, NO and NO_2 and turned each one into its own dataset. The measurement stations recorded the NO_2 and NO level in $\mu g/m^3$ in the air. The measurements were made between the 2^{nd} of January 2011 and the 30^{th} of April 2018. Some days had missing values and were dismissed. These two datasets have 2669 factor 1 values and 24 factor 2 values.

Pre-processing All datasets were normalised between 0 and 1 via min-max scaling. Because they were inherently multi-scaled (for example some subway stations are much bigger than others and the most frequented part of town have more pollutant than others), this normalisation was done by factor 2 (spatial context). They were split into train, validation and test set with a classical 70%/15%/15% ratio.

Noise/outlier addition For the experiments where noise was added, 15% of the training set examples' factor value were swapped.

4.2 Baselines

We compare our models to two baselines and one oracle in order to provide a clear interpretation of our performances.

Baseline average (BL avg) This baseline predicts $\hat{\mathbf{x}}_{i,j}$ by averaging the observed time series sharing a common factor class value i or j. If we denote $\mathcal{N}$ the set of factor 2 class values of the observed sequences with a factor 1 class value of i and $\mathcal{M}$ the set of factor 1 class values observed sequences with a factor 2 class value of $\hat{j}$, then $\hat{\mathbf{x}}_{i,j} = \frac{1}{|\mathcal{N}|+|\mathcal{M}|}(\sum_{k\in\mathcal{N}} \mathbf{x}_{i,k} + \sum_{k\in\mathcal{M}} \mathbf{x}_{k,j})$

This baseline can be used for both our tasks. The only difference is that for matrix completion both $|\mathcal{N}|$ and $|\mathcal{M}|$ are strictly greater than 0, while for the new factor value task, either $|\mathcal{N}|$ or $|\mathcal{M}|$ is equal to 0.

Baseline p-Nearest Neighbours (BL NN(p)) This intuitive approach is dedicated to the 2^{nd} task (new factor value). In this task, we are given $\mathbf{x}_{\hat{i},j}$, where $\hat{i}$ corresponds to a new factor 1 class value while j corresponds to an observed factor 2 class value. We aim at predicting all other $\mathbf{x}_{i,k}$ for $k \in \mathcal{M}$, where $\mathcal{M}$ is the set of factor 2 class values observed in the training set. If we denote $\mathcal{N}$ the set of factor 1 class values of observed series that have j for factor 2 class value, the principle of this baseline is to find the p series in $\mathcal{N}$ that are the closest to $\mathbf{x}_{\hat{i},j}$. Let us define $\mathcal{U}$ the subset corresponding to these series ($\mathcal{U} \subset \mathcal{N}$, $|\mathcal{U}|$) Then, for this baseline : $\hat{x}_{i,j} = \frac{1}{p}\sum_{k\in\mathcal{U}} \mathbf{x}_{k,j}$

2. https://www.kaggle.com/robikscube/hourly-energy-consumption/
3. www.pjm.com
4. https://www.kaggle.com/decide-soluciones/air-quality-madrid/
5. https://datos.madrid.es/portal/site/egob

Oracle p-Nearest Neighbours (Oracle NN(p)) The third baseline is an oracle since it requires access the ground truth. $\mathbf{x}_{i,j}$ is simply estimated by averaging its p nearest neighbours in the training set.

For both baselines based on nearest neighbours, p is a parameter of the baselines we optimise.

4.3 Quantitative results

We first give our results for both tasks on data without noise. We then give our results on both task on noisy data (15% of the training set is labelled incorrectly).

4.3.1 Original data without noise, task 1 : matrix completion

The Table 1 shows our results for our first task (matrix completion) with the MSE between the predicted series and the ground truth. As expected, the oracle baseline achieves near perfect results. The settings with embeddings matrices outperforms the neural network settings because it encodes the exact information of the factor class values, whereas the neural network settings does not have access to it. The model using the attention mechanism also outperforms the others settings, proving that it is a big improvement of our model.

4.3.2 Original data without noise, task 2 : new context value

For task 2, we want to make prediction for all factor 2 from a single sequence associated with a new factor 1 (or respectively extrapolating predictions for all factor 1 from one sequence corresponding to a new factor 2).

The Table 2 shows our results on this task (same MSE as the previous one). Our results on the new factor 2 (location) are just as the same as our baselines for the datasets on Energy and Air quality which can be explained by the low number of factor 2 in the datasets (10 and 24). Otherwise, we outperforms our baseline. Once again, the model with attention mechanism are a clear improvement on the others.

4.3.3 Data with noise, task 1 : matrix completion

The Table 3 shows our MSE results for our first task (matrix completion) on the noisy data. The oracle baseline, though a bit disturbed by the noise, still achieves very good results.

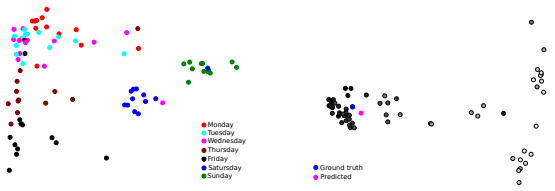
The settings without attention is comparatively more impacted in their results than the settings with attention, showing again the strength and resilience the attention mechanism bring to the model.

4.3.4 Data with noise, task 2 : new context value

The Table 4 shows our MSE results on this task. Just like when there is no noise, our models outperform the baselines, except for the Energy and NO dataset on new location (factor 2) where they achieve comparable performances. The models with attention mechanism achieve better performance that the one without and manage to limit the impact of the noise in the data.

4.4 Qualitative results

We explore the exploitative power of our model, to show that the learned latent spaces are indeed meaningful. As an example, the Figure 5a shows the Principal omponent Analysis (PCA) of the set of representation of all the days (factor 1) present in the STIF dataset. It shows that our model latent space of representation has grouped together the representations corresponding to the same day of the week (information that is not present in the dataset). Moreover, the Saturdays and Sundays are clearly differentiated from the weekdays which can be explain by the habits of the regular users of the Parisian network that are different on the week-end than there are during the week.



(a) PCA of the representations of the different day of the STIF dataset in their latent space, no attention, no noise.

(b) PCA of the representation of the different days of the STIF training set, with attention, without noise.

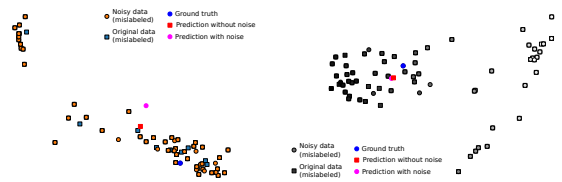
For our model with an attention mechanism, the Figure 5b show the PCA of the representation of the days (factor 1) in the training set of the STIF dataset. The level of grey indicates the weight of the attention. The blue dot is the representation of the ground truth encoded factor. The magenta dot represent the weighted sum that is the output of the attention mechanism. We can see that the output of the attention mechanism is close to the ground truth.

		STIF			Energy			NO			NO2		
Encoder	Decoder	Series avg	Emb. avg	Attention	Series avg	Emb. avg	Attention	Series avg	Emb. avg	Attention	Series avg	Emb. avg	Attention
Oracle NN (p)		0.001 (5)			0.00008 (4)			0.0007 (5)			0.0031 (5)		
BL avg (temporal)		0.0179			0.006			0.0107			0.0215		
BL avg (spatial)		0.0121			0.0089			0.0045			0.0098		
Emb	MLP	0.0028			0.002			0.0043			0.0098		
Emb	CNN	0.0053			0.0048			0.0046			0.0101		
Emb	RNN	0.0016			0.0015			0.0041			0.0102		
MLP	MLP	0.0084	0.0084	0.0062	0.0046	0.0047	0.0032	0.0052	0.0052	0.0039	0.0106	0.0105	0.0093
CNN	CNN	0.0073	0.0073	0.0051	0.0059	0.0059	0.0038	0.0058	0.0057	0.0035	0.0104	0.0105	0.0092
RNN	RNN	0.0026	0.0025	0.0018	0.0036	0.0034	0.0018	0.0050	0.0051	0.0036	0.0107	0.0105	0.0083

TABLE 1 – MSE errors for task 1 : sequence generation with the matrix completion setting. From top to bottom : baselines, transductive modelling & inductive modelling (context encoded on the fly).

			STIF			Energy			NO			NO2		
	Encoder	Decoder	Series avg	Emb. avg	Attention	Series avg	Emb. avg	Attention	Series avg	Emb. avg	Attention	Series avg	Emb. avg	Attention
New loc.	BL NN (p)		0.0123 (150)			0.0081 (3)			0.0049 (5)			0.0107 (5)		
	BL avg.		0.0125			0.0087			0.0052			0.0112		
	MLP	MLP	0.0059	0.0059	0.0056	0.0118	0.0118	0.0080	0.0058	0.0057	0.0052	0.0107	0.0107	0.0106
	CNN	CNN	0.0075	0.0076	0.0072	0.0099	0.0102	0.0096	0.0062	0.0064	0.0066	0.0101	0.0102	0.00100
	RNN	RNN	0.0049	0.0049	0.0044	0.0107	0.0108	0.0070	0.0062	0.0065	0.0053	0.0106	0.0108	0.0099
New day	BL NN		0.0143 (7)			0.0093 (900)			0.0099 (70)			0.0207 (60)		
	BL avg.		0.0145			0.0089			0.0103			0.0210		
	MLP	MLP	0.0063	0.0061	0.0046	0.0076	0.0076	0.0069	0.0062	0.0062	0.0063	0.0133	0.0133	0.0131
	CNN	CNN	0.0061	0.0059	0.0052	0.0065	0.0063	0.0040	0.0071	0.0071	0.0066	0.0129	0.0128	0.0122
	RNN	RNN	0.0041	0.0040	0.0029	0.0047	0.0046	0.0019	0.0063	0.0063	0.0055	0.0139	0.0139	0.0112

TABLE 2 – MSE errors for task 2 : generation of series associated with a new factor. New locations above, new days below.



(a) Model without attention (b) Model with attention

FIGURE 6 – The representation of the different day of the STIF training set with noise. The squares represent the original data, without noise while the dot represent the noisy data. The blue dot is the representation of the ground truth encoded factor. The magenta dot represent the representation given as input in the decoder. The red square is the representation that would have been given if there was no noise in the data. On the left, the level of grey indicates the weight of the attention.

Finally, when we add noise to the dataset by mixing up some of the label in the training set, we can see that our model with attention is less disturbed by the noise and outputs a representation close to the ground truth.

5 Conclusion & Perspective

We proposed a new framework to learn disentangled factor representations from time series. This new architecture is focused on context modelling which make it flexible and explainable. It allows us to tackle new tasks that correspond to a concrete requirement for economy in data collection. Furthermore, we demonstrate that the introduction of an attention mechanism brings both robustness and explanation : the attention weighting of the samples allows us to partially catch up with the oracle and provides an interesting insight into the topology of the factors.

In future works, we plan to work with heterogeneous data to bridge between time series and textual information. The idea would be to build a kind of *Time Series question answering* framework inspired both by the Visual Question Answering and the data-to-text recent approaches. Playing with encoders and decoders attached to different nature of data will be the key to face this ambitious challenge.

Références

[1] BOX, G. E., AND , G. M. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 17*, 2 (1968), 91–109.

[2] BROMLEY, J., BENTZ, J. W., BOTTOU, L., GUYON, I., LECUN, Y., MOORE, C., SÄCKINGER, E., AND SHAH, R. Signature verification using a "siamese" time delay neural network. In *IJPRAI* (1993).

		STIF			Energy			NO			NO2		
Encoder	Decoder	Series avg	Emb. avg	Attention	Series avg	Emb. avg	Atten	Series avg	Emb. avg	Attention	Series avg	Emb. avg	Attention
Oracle NN (p)		0.0011 (5)			0.0001 (5)			0.0013 (5)			0.0044 (6)		
BL avg (temporal)		0.0270			0.0089			0.0119			0.0227		
BL avg (spatial)		0.0171			0.0099			0.0098			0.0214		
MLP	MLP	0.0162	0.0163	0.0069	0.0096	0.0097	0.0080	0.0086	0.0087	0.0057	0.0185	0.0188	0.0098
CNN	CNN	0.0163	0.0163	0.0065	0.0069	0.0072	0.0045	0.0087	0.0087	0.0034	0.0193	0.0193	0.0094
RNN	RNN	0.0164	0.0164	0.0159	0.0055	0.0057	0.0032	0.0089	0.0089	0.0062	0.0178	0.0189	0.0091

TABLE 3 – MSE errors for task 1 : sequence generation with the matrix completion setting with **noisy data**. From top to bottom : baselines, transductive modelling & inductive modelling (context encoded on the fly).

			STIF			Energy			NO			NO2		
	Encoder	Decoder	Series avg	Emb. avg	Attention	Series avg	Emb. avg	Attention	Series avg	Emb. avg	Attention	Series avg	Emb. avg	Attention
New loc.	BL NN (p)		0.0211 (130)			0.0135 (3)			0.0089 (6)			0.0127 (5)		
	BL avg.		0.0242			0.0146			0.0095			0.0206		
	MLP	MLP	0.0163	0.0162	0.0158	0.0159	0.0160	0.0097	0.0094	0.0095	0.0084	0.0190	0.0193	0.0180
	CNN	CNN	0.0164	0.0165	0.0158	0.0102	0.0102	0.0090	0.0090	0.0090	0.0086	0.0185	0.0187	0.00178
	RNN	RNN	0.0172	0.0173	0.0169	0.0102	0.0101	0.0089	0.0094	0.0092	0.0087	0.0184	0.0185	0.0178
New day	BL NN		0.0183 (5)			0.0094 (900)			0.0099 (70)			0.0207 (60)		
	BL avg.		0.0155			0.0102			0.0112			0.0215		
	MLP	MLP	0.0100	0.0102	0.0093	0.0091	0.0091	0.0076	0.0074	0.0074	0.0072	0.0160	0.0162	0.0159
	CNN	CNN	0.0098	0.0099	0.0098	0.0065	0.0066	0.0058	0.0074	0.0074	0.0073	0.0163	0.0163	0.0158
	RNN	RNN	0.0093	0.0093	0.0089	0.0052	0.0050	0.0042	0.0070	0.0072	0.0066	0.0162	0.0163	0.0140

TABLE 4 – MSE errors for task 2 with noisy data : generation of series associated with a new factor. New locations above, new days below.

[3] CHEN, X., DUAN, Y., HOUTHOOFT, R., SCHULMAN, J., SUTSKEVER, I., AND ABBEEL, P. Infogan : Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS* (2016).

[4] CHO, K., VAN MERRIENBOER, B., GÜLÇEHRE, Ç., BOUGARES, F., SCHWENK, H., AND BENGIO, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR abs/1406.1078* (2014).

[5] CHOROWSKI, J. K., BAHDANAU, D., SERDYUK, D., CHO, K., AND BENGIO, Y. Attention-based models for speech recognition.

[6] CHUNG, J., KASTNER, K., DINH, L., GOEL, K., COURVILLE, A. C., AND BENGIO, Y. A recurrent latent variable model for sequential data. *NIPS* (2015).

[7] CRIBIER-DELANDE, P., PUGET, R., GUIGUE, V., AND DENOYER, L. Time series prediction using disentangled latent factors. *ESANN* (2020).

[8] DENTON, E., AND BIRODKAR, V. Unsupervised learning of disentangled representations from video. *NIPS* (2017).

[9] FLUNKERT, V., SALINAS, D., AND GASTHAUS, J. DeepAR : Probabilistic Forecasting with Autoregressive Recurrent Networks.

[10] FRANCESCHI, J., DIEULEVEUT, A., AND JAGGI, M. Unsupervised scalable representation learning for multivariate time series.

[11] GOODFELLOW, I. J., POUGET-ABADIE, J., MIRZA, M., XU, B., WARDE-FARLEY, D., OZAIR, S., COURVILLE, A. C., AND BENGIO, Y. Generative adversarial nets. In *NIPS* (2014).

[12] HIGGINS, I., MATTHEY, L., PAL, A., BURGESS, C., GLOROT, X., BOTVINICK, M. M., MOHAMED, S., AND LERCHNER, A. beta-vae : Learning basic visual concepts with a constrained variational framework. In *ICLR* (2017).

[13] HOCHREITER, S., AND SCHMIDHUBER, J. Long short-term memory. *Neural Comput. 9*, 8 (Nov. 1997), 1735–1780.

[14] KINGMA, D. P., AND WELLING, M. Auto-encoding variational bayes. *CoRR* (2013).

[15] LI, Y., AND MANDT, S. A deep generative model for disentangled representations of sequential data. *arXiv abs/1803.02991* (2018).

[16] LIU, Q., WU, S., WANG, L., AND TAN, T. Predicting the next location : A recurrent model with spatial and temporal contexts.

[17] MATHIEU, E., RAINFORTH, T., SIDDHARTH, N., AND WHYE TEH, Y. Disentangling Disentanglement in Variational Autoencoders. *arXiv e-prints* (Dec 2018), arXiv :1812.02833.

[18] PALATUCCI, M., POMERLEAU, D., HINTON, G. E., AND MITCHELL, T. M. Zero-shot learning with semantic output codes.

[19] PETERS, M. E., NEUMANN, M., IYYER, M., GARDNER, M., CLARK, C., LEE, K., AND ZETTLEMOYER, L. Deep contextualized word representations.

[20] REZENDE, D. J., MOHAMED, S., AND WIERSTRA, D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *ICML abs/1401.4082* (2014).

[21] RIDGEWAY, G., MADIGAN, D., AND RICHARDSON, T. Boosting methodology for regression problems. In *AISTATS* (1999).

[22] SAPANKEVYCH, N. I., AND SANKAR, R. Time series prediction using support vector machines : a survey. *IEEE Computational Intelligence Magazine 4*, 2 (2009), 24–38.

[23] VILLEGAS, R., YANG, J., HONG, S., LIN, X., AND LEE, H. Decomposing motion and content for natural video sequence prediction. *ArXiv abs/1706.08033* (2017).

[24] VINYALS, O., BLUNDELL, C., LILLICRAP, T., KAVUKCUOGLU, K., AND WIERSTRA, D. Matching networks for one shot learning.

[25] ZIAT, A., DELASALLES, E., DENOYER, L., AND GALLINARI, P. Spatio-temporal neural networks for space-time series forecasting and relations discovery. *ArXiv abs/1804.08562*.