Filling Gaps in Micro-Meteorological Data

Antoine Richard, Lior Fine, Offer Rozenstein, Joseph Tanny, Matthieu Geist, Cedric Pradalier

▶ **To cite this version:**

Antoine Richard, Lior Fine, Offer Rozenstein, Joseph Tanny, Matthieu Geist, et al.. Filling Gaps in Micro-Meteorological Data. 2020. hal-03091151

HAL Id: hal-03091151
https://hal.science/hal-03091151

Preprint submitted on 30 Dec 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Antoine Richard, Lior Fine, Offer Rozenstein, Joseph Tanny, Matthieu Geist, Cedric Pradalier

Filling Gaps in Micro-Meteorological Data

Antoine Richard[1] [✉], Lior Fine[2,3], Offer Rozenstein[2], Joseph Tanny[2,4], Matthieu Geist[5], and Cedric Pradalier[6]

[1] Georgia Institute of Technology, USA antoine.richard@gatech.edu
[2] Institute of Soil, Water and Environmental Sciences, Agricultural Research Organization, Volcani Center, Israel
[3] The Robert H Smith Faculty of Agriculture, Food and Environment, The Hebrew University of Jerusalem
[4] HIT – Holon Institute of Technology, Holon, Israel
[5] Google Research, Brain Team
[6] GeorgiaTech Lorraine – UMI2958 GT-CNRS, France

Abstract. Filling large data-gaps in Micro-Meteorological data has mostly been done using interpolation techniques based on a marginal distribution sampling. Those methods work well but need a large horizon of the previous events to achieve good results since they do not model the system but only rely on previously encountered iterations. In this paper, we propose to use multi-head deep attention networks to fill gaps in Micro-Meteorological Data. This methodology couples large-scale information extraction with modeling capabilities that cannot be achieved by interpolation-like techniques. Unlike Bidirectional RNNs, our architecture is not recurrent, it is simple to tune and our data efficiency is higher. We apply our architecture to real-life data and clearly show its applicability in agriculture, furthermore, we show that it could be used to solve related problems such as filling gaps in cyclic-multivariate-time-series.

Keywords: Evapo-Transpiration · Gap-Filling · Attention-Models.

1 Introduction

Gap filling is a crucial task in environmental science. In many fields such as precision agriculture or environmental monitoring, sophisticated sensor systems record data in remote areas over long periods of time. Often, those systems suffer from power breaks, sensor malfunction, or reduced data quality that requires binning some of the measurements. This implies that the data are frequently incomplete. As an example, approximately 30% of eddy-covariance flux measurements are missing or binned [6]. Setting up experiments to acquire environmental data is a time consuming and very tedious process, requiring highly skilled personnel, expensive equipment, and a long time to set up, process and analyze. Hence precisely filling gaps is of paramount importance to maximize the available data. Additionally, when data aggregation needs to be done, for example for calculating the seasonal water budget or the amount of daily evapotranspiration (ET) for irrigation purposes, it is necessary to complete the gaps in the time-series. In this paper, we aim to fill gaps in the ET measured by an Eddy-Covariance (EC) device on short-life-span crops (3 to 4 months): tomatoes, cotton, and wheat. The

training data consists of a set of real meteorological variables: the net-radiation, the relative-humidity, the air-temperature, and the wind-speed, along with the measured ET. Our dataset (training and testing) is limited to those 3 crops with 2 independent recording seasons and sites for each crop, more details are provided in section 4.1. Here, due to the warm climate resulting from the geographical location of the considered fields (middle-east), and the rapid development of the crops over the course of a short growing season, our system shows rapidly changing dynamics. Additionally, since we study seasonal crops, we cannot rely on site specific data recorded over multiple years as it is typically done when filling in EC measurements over forests.

Formally, the problem that we are studying consists of multiple variables that have dependencies with one or more periodic variables. Those periods are not necessarily known and may require a large time horizon to observe. We consider that the dependencies between the variables and the time cannot be modeled. To further focus our problem, we will only try to fill gaps in one variable, as shown in figure 1. This variable depends on the other remaining variables. Hence the other variables are always considered to be intact. Even-though this assumption can seem bold, neighboring weather stations can be used to fill missing meteorological variables if a sensor malfunction were to occur on the EC tower. To do so one may have to compensate for the bias in-between the EC measurements and the ones of the weather station.

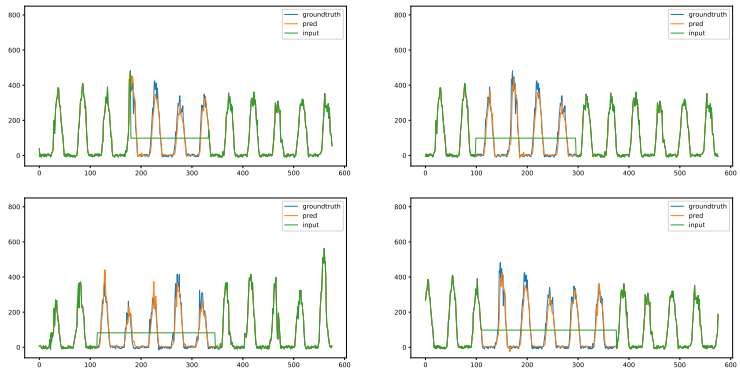

Fig. 1: Gap filling in a latent heat flux recording with gaps of 3 to 6 days (144 to 288 points). The results were obtained using our models for wheat in Saad, Israel. Blue: real sequence, orange: gap filled sequence using our method, green: sequence with gap. Other variables not shown.

To tackle this problem, an intuitive approach would be to use Fourier's analysis. However, the problems to which we aim to apply our algorithms may consist of very long periods, at the scale of growing seasons (*i.e.* 2-6 months) or years. Furthermore, the measurements we are using often consist of only one growing season per location. Our measurements are performed in different locations and different times of the year. Since the beginning and the length of the growing season depend on the type of crop, the weather and other agronomic considerations, the measurements do not necessarily start at the same time, nor are they performed for the same duration. All those factors

are the reason that techniques based on Fourier's analysis were not considered in this study.

The first consideration in the development of the algorithm to be described in section 3 was the need for the irrigation research community that relies on evapotranspiration (loss of water from the soil by evaporation and plant transpiration) measurements to estimate the irrigation demand. Since the evapotranspiration depends on the air temperature, relative humidity, wind speed, and radiation, previous research in this field relied on the diurnal cycles of these variables to fill gaps in water vapor flux measurements from EC. To do so, they used Mean Diurnal Variation (MDV) [6] and Marginal Distribution Sampling (MDS) [15,18]. Those methods use neighboring full non-contaminated data to try and patch the missing data. Despite the very good performance of techniques such as Look-Up Tables (LUT) [13], they are often tailor-made solutions that only solve particular types of problems. Additionally, since they rely on neighboring data, they require a fairly large amount of non-corrupted neighboring data to work.

As far as we know, there is no general machine learning paper dealing with the gap filling problem: some of the work focused on making sense of the data despite the gaps [2], while the rest of the work was mostly using old architectures such as Multi-Layer Perceptron (MLP)[4]. However, the applicability of Recurrent Neural Networks (RNNs) to perform time-series-forecasting is not left to demonstrate, hence, we will present how those approaches could be used in the context of filling gaps. Yet, both MLPs and RNNs suffer from multiple drawbacks. The MLPs are not "context-aware"; thus, they cannot extract local trends, which leads to making them inapplicable in systems with rapidly changing dynamics like ours. As for the RNNs, they are context-aware, but they can suffer from vanishing gradient. There is also a limit to the reach and the amount of information that can be propagated. Additionally, they are hard to tune and expensive to train and infer, especially on long sequences. In this study, we will be considering sequences ranging from 200 to 600 elements.

To solve the aforementioned problems, we propose to rely on the most recent advances in Natural Language Processing (NLP) [17,5]. These models provide both the modeling capacities of the neural-networks but also the context understanding that offers the MDSs and RNNs without suffering from their respective limitations. Our model uses the multi-head attention layers defined in [17] with a modified positional encoding to account for the cyclicity of the data. We show that our approach outperforms current state of the art to fill gaps in evapo-transpiration data. And we also show that it can be applied to solve more general problems on a toy case.

In the end, this paper contributions can be summarized as follows:

- We derive a model for gap filling based on attention networks that account for the periodicity within the data.
- We apply our model to a real-world problem defining a new state-of-the-art performance despite a very limited training set.

2 Related Work

Reliably filling missing data-points in time-series is of the utmost importance in the field of environmental sciences. However, despite the importance of the task, this field still heavily relies on old methods.

Initially, the method used to replace missing data was LUT. It consists in using the previous occurrence that resembles most the point we are trying to recover. However, it has since been replaced by a smoother approach named MDS. In our problem, we consider that only the target data $Y \in \mathbb{R}$ is corrupted but that the remaining data $X \in \mathbb{R}^n$ is correct. We fit a probability distribution over $Y|X$. The missing values are recovered by sampling the marginal distribution over X and computing $P(Y|X)$. In practice, this is computed using K-Nearest-Neighbour (KNN). It amounts to performing a local weighted average over available values of $Y|X$. The weights are proportional to their distance from X, and only samples in the vicinity of the gap are used. The main drawback of those methods is that they solely rely on the neighboring points. Hence if the weighting window is too small, there may not be occurrences similar to the points to be filled. To avoid this problem, a large time horizon is required, as shown in [18]. However, when the horizon increases, the impact of non-observable variables (latent variables) may significantly deteriorate the filling quality. This is particularly true in systems with high paced dynamics, where the latent variables are responsible for amplitudes changes.

However, with the rise of deep neural-networks, those methods are being challenged by parametric approaches. Standard MLPs have been successfully used to fill gaps in slowly changing micro-meteorological time-series [14,4]. Those time-series have no seasonal trend or fixed seasonal trends and have large data-banks that range over multiple years. In those cases, MLPs are using the observation of the system X at a given point to predict the value of Y for that point. Hence it does not rely on neighboring values. Despite their positive results, the MLPs suffer from two significant drawbacks: they cannot natively embed temporal relations, and they cannot make long term relations. This means that if those networks were used for rapidly changing systems, they would not be able to extract local trends to re-scale their forecasts. Additionally, the use of MLP on those applications requires a large data-banks of previous years because those network leverages information like the day of the year. In our case this is not feasible as we do not have such information.

In the field of machine learning, some works have shown the applicability of RNNs to time-series forecasting. In general, Long Short-Term Memorys (LSTMs) [9] and Gated Recurrent Units (GRUs) [3] are common tools used in time-series forecasting. Unfortunately, to the best of our knowledge, in the case of gap filling, the machine learning literature has little to offer. This is most possibly due to the lack of interest of the community in this problem. Usually, when performing time-series forecasting, a sequence of chronologically ordered elements is embedded in a higher-dimensional space using a learned projection such as a linear transformation. It is then processed by a few stacked recurrent-layers, and, finally the results of the recurrent-layers are projected back into the desired shape. The recurrent nature of RNNs allows them to remember past information.

While these methods address the problems encountered with MLPs, they have limitations of their own. Indeed, LSTMs and GRUs suffer from limited memory capacity and range. Because they are recurrent architectures, the hidden-state of the network, also known as its memory, will go through n transformations, with n the number of elements in the sequence. This means that correlating elements at the beginning of the sequence with those at the end will be hard, especially since we consider processing sequences of 600 points. Additionally, properly initializing the RNNs' hidden state is tricky and leads to errors, especially in the case of a limited dataset. Finally, in the case of gap filling, it would make sense to use bidirectional RNNs [8]. Indeed standard RNN only leverages past information when bidirectional RNN can leverage past and future observations, making it a more appropriate choice. However, those architectures are not well suited for our task because the variable we are trying to fill gap in exhibits a cyclicity. We are confident that the RNN would be able to learn this cyclicity but other architectures can account for it natively if modified correctly. Directly accounting for the cyclicity should reduce the need for learning samples and make the overall learning process easier.

Recent advances in the field of NLP and Natural Language Generation (NLG) address those issues with the introduction of attention-based deep neural networks. Models such as Transformer [17], and now Bert [5], are the reference for many sequence-to-sequence (seq2seq) tasks in NLP. The mechanism at the root of these neural-networks is called self-attention. It offers the benefits of recurrent models without their downsides. Also, because the model is not iterative, it does not have to iterate through each of the sequence elements but can perform all the operation concurrently leveraging tensor-dot operations. This makes attention-based models much more efficient than their recurrent counterparts. Finally, these models can be modified to account for the cyclicity of the data. As they rely on a positional encoding to know where the different elements are located in the sequence, this encoding can modified to create a cyclicity. In the case of filling gaps in ET time-series we can use the time of the day. This results in all the values for a given time of the day to be located at the same position in the sequence from the perspective of the attention network. In comparison, a recurrent network iterates through the whole sequence and stores in its hidden state all that happened before to make its prediction. It is not aware that there might be a cyclic behavior unless it learns it. This consideration, associated with the one stated earlier on, explains why we did not consider RNNs for this study.

3 Filling Gaps

In order to leverage information from both past, present, and future, we chose to develop seq2seq attention-based models dedicated to filling gaps. Our architecture can be seen in figure 2.

From a formal point of view, our system can be formulated as in (1), where $x_i(t)$ is the set of observed variables such that $\forall i \in [1,n], \forall t \in [1,T],\ x_i(t) \in \mathbb{R}$ and $z_i(t)$ is the set of non-observed variables such that $\forall i \in [1,m], \forall t \in [1,T],\ z_i(t) \in \mathbb{R}$.

$$\begin{aligned} y(t) &= f(x_1(t), .., x_n(t), z_1(t), .., z_m(t)) \in \mathbb{R} \\ &\forall i \in [1,n], \exists \tau_i \in \mathbb{R} \text{ s.t. } \mathrm{x_i(t) = x_i(t + \tau_i) + \epsilon} \end{aligned} \tag{1}$$

We will denote as x the "sequence of observations" of our system, and y the "sequence of targets" of our system. The goal of our model will be to recover the missing values from the target sequence using the sequence of observations and the sequence of targets with missing values. Our architecture leverages both past and future information using the attention mechanism.

3.1 Architecture

When building our model, we took inspiration from the canonical attention architecture: Transformer. An architecture like transformer does not iteratively process all the elements of a sequence but instead processes a whole sequence at once. Additionally, it features an encoder-decoder structure, where, the encoder is used to process the input sequence, and the decoder is used to generate the output sequence based on the processed input sequence. The encoder share a fairly similar structure where each element of the sequence is embedded using the same transformation and then a process called self attention is applied on the sequence. This process results in a sequence of similar length as the input sequence, and allows the network to make sequence wide correlations. Then a stack of shared feed-forward layer are applied on each element of the previously processed sequence. Complete details of the architecture is given in [17, sec3.1].

We only relied on a single part of the transformer model: the encoder. This simplification of the encoder/decoder architecture effectively reduces the number of parameters within the model, reducing its complexity and making it easier to train. Similarly to transformer our architecture features multiple attention-heads which process embedded input. This embedding is of size d_{model}.

An attention-head (or Scaled Dot-Product Attention) computes the correlation for every combination of elements pairs in the sequence under the form of an attention matrix. This propagates information between two elements, even if they are far apart in the sequence. The exact implementation of this method is described in [17, sec3.2]. Multi-head attention consists in using multiple attention-heads in parallel (with the same input) and then concatenate their output.

Initial results with a single head showed poor performances: the network was having a hard time separating the different variables. Indeed, in the case of the observation sequence, the model should look everywhere, but in the case of the target sequence, it should not pay attention to the gap. Hence, based on that observation, and in an effort to minimize the learning complexity, we chose to manually assign different variables to our network heads. We used three attention heads: one head processes the observation sequence, another head processes the target sequence, and finally, the last head processes a concatenation of observation and target sequences. Ideally, we would let the

model learn how to separate the variables on its own, but, due to the limited training samples this was not feasible in the case of our dataset.

Furthermore, it is worth noting that those heads do not use a causal mask[7]. In attention heads, causal masks prevent the association of an element with other elements that happen later in the sequence. Removing it allows each attention head to look at the past, present, and future at the same time. This enables our model to extract local trends, and correlate information from elements far apart in the sequence in a way an RNN could not.

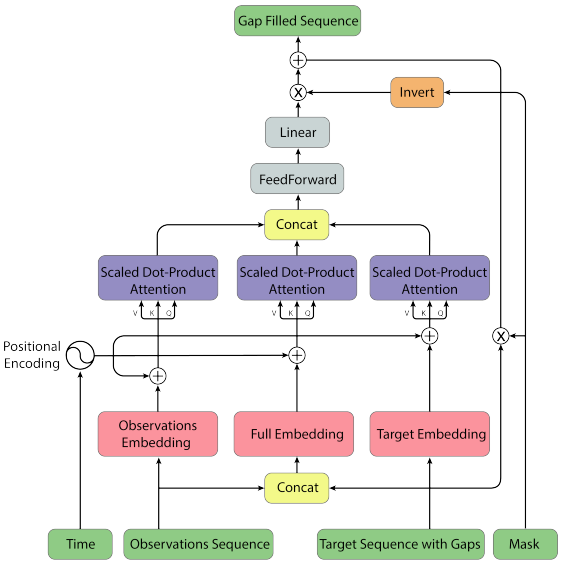

Fig. 2: Our architecture for gap filling

3.2 Feed Forward Layer and Copy Task

After applying the multi-head attention mechanism, the output heads are concatenated and passed to a feed-forward layer: two dense layers with Leaky-Rectified-Linear-Unit (Leaky-ReLU) [12] activation function, and a dropout layer. Similarly to [16], our models performed approximately 10% better when using Leaky-ReLUs over normal Rectified-Linear-Units (ReLUs). After the feed-forward layer, a dense layer is used to project back the output of our feed-forward layer to a one-dimensional sequence: the target sequence with its missing data filled.

Finally, we implemented a full skip-branch directly copying the original target value onto the output. This prevents the network from learning a complicated function, where

[7] Please note that this is only true in attention heads. We use a mask in the overall structure to copy the non-gap-points

part of the target sequence is copied, and the rest is changed to fill the gap. This is achieved by providing our network with a binary mask (a sequence of binary numbers). Where, True means that the data is to be copied, and False means the network should fill in the point. As of now, the detection of the gaps in the data is handled by the EddyPro software[8]. This software is one of the most prevalent in the field and features a wide panel of failure detection methods. Nevertheless, future work may focus on learning areas of the sequence that need correction, *i.e.* using a similar architecture to learn the sensor filtering.

3.3 Positional encoding

Unlike RNNs, which recurrently process the elements of the sequence, attention models cannot know where the different elements are positioned inside the sequence. To alleviate this issue, Transformer uses a positional encoding, which gives a unique value (an identifier) to each element of the sequence. However, to account for the cyclicity of our variables we modified the positional encoding. When Transformer uses the position of the elements in the sequence to perform the positional encoding, we chose to use the value of the primary periodic variable: in the case of the micro-climatic data, the time of the day. Hence, the positional encoding becomes cyclic. This lets us account for the periodic time dependencies natively. Similarly to the Transformer model, we use sine and cosine functions with the modifications mentioned above, as can be seen in (2). $pos \in \mathbb{N}$ is the position in the sequence, i is the dimension, and $t(pos) \in \mathbb{N}$.

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(\frac{t(pos)}{10000^{\frac{2i}{d_{model}}}}\right) \\ PE_{(pos,2i+1)} &= \cos\left(\frac{t(pos)}{10000^{\frac{2i}{d_{model}}}}\right) \end{aligned} \tag{2}$$

Despite the appeal to learn the encoding [7], which should, in theory, allow the network to learn the frequencies that make most sense for the problem at hand, we chose not to. Firstly, based on the conclusion of [17], it seems that there are no benefits from using a learned positional encoding. Secondly, this lets us reduce the complexity of the learning process in regard to the limited amount of training examples at our disposal.

From a practical perspective, the positional encoding depends on the period from the different sequences of the batch, which have different time offsets. This means that it cannot be computed ahead of time. Thus to maximize performance, the positional encoding is computed directly within the network's graph.

3.4 Training

To train our models we used gap-free-data and generated artificial gaps inside them. Since gap-free-data almost do not exist in the real-world, we used linear interpolation to fill small gaps (1 to 2 points). In the case of larger gaps, we simply removed the

[8] https://www.licor.com/env/products/eddy_covariance/software.html

days during which those occurred and did not took samples that overlapped with the gaps. The exact details of the dataset generation are given in section 4.1. Please note we do not aim to fill those small gaps, this problem is considered trivial and we are only looking to fill continuous gaps larger than 24 points (or half a day) and up to 288 points.

We then had to put some values inside the gaps to tell our models where points are missing in the sequence. Unlike NLP, we cannot choose to set an unknown character to indicate the areas where we want to recover data as our variables are defined in $\mathbb{R}$. However, since the data is normalized to a zero-centered normal distribution with unit variance we can make the assumption that most of our values will lie within the $]-2, 2[$ interval. Hence, we could pick a value outside of this interval to fill our gaps. Yet, after some experimentation, we found that it was more reliable to fill the gaps with the mean of the non-corrupted points within that same sequence. Not only did it make the models predictions more reliable, but it also increased the convergence speed of our architectures.

Our networks were trained using the ADAM optimizer [10] (without using the advanced modification of transformer). As in most regression problems, we use an L_2-loss that we average over the whole sequence excluding the points outside of the gap.

4 Experiments

4.1 Datasets

We tested our model on two cases: a real use case from the field of agriculture and a toy case to demonstrate that our approach also scales to other related problems. For the application on real data, we aim to demonstrate that our methodology is capable of strong generalization by learning on a growing season in different crop fields and using this knowledge to fill gaps in situations that were not encountered before: a different crop at a different season at a different location.

Eddy Covariance (EC) Data To evaluate our model on a real-world scenario, we chose to apply it to ET measurements. These measures of ET are acquired using an EC device. An EC tower measures latent heat flux (*i.e.* water vapor flux) from a crop. This can be used to infer the crop water consumption and hence its evapo-transpiration. Additionally, the tower also records the relative humidity in the air, along with the sun radiation, the wind speed and the air temperature. These 4 variables, which we will call meteorological data, are the only variables our neural network has access to, to reconstruct the missing point in the ET. Please note that sometimes the tower had sensors failures as well. In order to avoid having gaps in our input data, we used values from a nearby meteorological station to fill in the missing meteorological values. The files used to train and evaluate our model come from six different measurement campaigns[9] and feature two growing seasons in three different types of crops: processing tomatoes, cotton, and wheat. These recordings were acquired at different seasons: winter and summer, and in different regions in Israel: north and south. The direct consequence is that

[9] data available upon request

the amplitudes of the variables change significantly between the different recordings. Each recording has 5 variables, the latent heat flux (i.e. our target), the net radiation, the relative humidity, the air temperature, and the wind speed. Those files are recorded with a half-hourly rate over a period of 3 to 4 months. In total, this makes for about 3000 to 4000 continuous gap-free-points per recording.

To evaluate our networks on those crops, we could not train and test on each recording individually. This would result in too little data to train or evaluate our model properly. Also, it would mean that our model would be tuned for this specific recording, and would not be able to generalize to other crops, making our approach impractical. We verified this hypothesis using tomato crops and then chose to do 6 different train/test sets. To do so, we put all our recordings but one in the training and the remaining one in the test. We ran the 6 possible combinations and obtained 6 different datasets.

This dataset exhibits interesting behaviors when compared to similar problems, for instance in forestry. Here, the hot middle-eastern climate coupled to the spring season creates rapid changes in the plant canopy, increasing its leaf area index which in-turns increases the water it consumes. Ideally, we would include a vegetation variable, like the leaf-area index to our model, but this variable is very tedious to acquire, and in most cases is not measured. Using vegetation indices (e.g. NDVI - Normalized Difference vegetation index) derived from remote sensing data[11] could also be considered, but they often exhibits gaps and are not applicable to small fields. This is why in the end, we did not consider any vegetation variables in our experiments. On this dataset we only compare ourselves to REddyProc. As the MLP performed worsed than the REddyProc its results are not presented here.

Toy Problem To test our architecture, we developed a small toy problem that features similar construction to our general problem. We create 3 variables $x_i(t) \in \mathbb{R}, \mathrm{t} \in \mathbb{N}$ s.t. $x_i(t) = \alpha_i + \sin(t * \gamma + \tau_i) * \beta_i + \epsilon_i$ and γ is computed such that the periodicity of the variables is 48 points. Additionally, we create a latent variable (which will not be observed by our methods) $z_1(t) \in \mathbb{R}$ defined as $z_1 = \sin(t * \omega)$ where ω is set such that the periodicity of z_1 is 720 points. We chose 720 as this is larger than the maximum scope of our neural-networks. We then combine those variable to form (3).

$$y(t) = \left| x_1(t)^2 \times e^{x_2(t)} \times \log\left(x_3(t)\right) \right| \times \left(z_1(t) + 2\right) + \epsilon \tag{3}$$

This problem is interesting because the cyclicity of the positional encoding of our networks is set to 48 points and not 720 points. Hence, our model will have to adapt to the amplitude change created by z_1 and extract local trends to estimate the correct values. A total of 70,000 points were generated. On this problem, we compare ourselves to a method with a maximum window of 300 points on each side of the gap. It is set to use up to 15 points as long as their L_∞-norm is below 5% error. If no points matching this condition are found, it then works as a LUT. We are also comparing ourselves to an MLP model that we acquired doing a grid-search for the optimal set of dense layers.

4.2 Evaluation

We chose to evaluate our approach on different sequences and gap sizes. To do so, we generated sequences with 3 different lengths: 192, 384, and 576 points, equivalent

respectively to 4, 8, and 12 days on the real data. For each of these lengths, one model is learned. We will refer to sequences of 192 points as small (S), 384 points as medium (M), and 576 as large (L).

When considering our real data, the limited quantity of data-points was a problem. Even for small sequences, we only had 82 unique non-overlapping sequences. To increase that number, we generated sequences using a moving window with a stride of one. This allowed us to generate about 2500 sequences out of one recording. This high redundancy in our data explains why we aimed to minimize the network's parameters: to prevent overfitting. Additionally, we chose to generate gaps of random size, and at random positions when sampling batches during both training and testing. This further increases the quantity of available data and further mitigates the risk of overfitting. In the small sequences, the gaps ranged from 24 to 72 points; in the medium ones, the gaps ranged from 72 to 144 points, and in the large ones, the gaps ranged from 144 to 288. This is slightly higher than the usual 30% missing data in average in EC measurements.

To evaluate our approach on the real data, we compare ourselves to the most prevalent tool in the field: REddyProc [18]. This tool, developed by the Max-Planck Institute, is strictly dedicated to fill gaps within flux measurements by EC systems. Embedded as an R package, it relies on the MDS, and in some extreme cases on the MDV to recover the missing points. The main restriction of this tool comes from the techniques it uses. As it is based on MDS, a window of at least 7 days on each side of the gap is being used. Since the size of this window cannot be changed, the two approaches were compared using different sequence lengths to fill the gaps: Our network sees a much smaller horizon of points due to memory limitation of our GPUs. Also, using 14-day windows (7 on each side) and 6-day gaps would be equivalent to use sequences of 960 points, which we could not do with our data-recordings as it would result in too few sequences.

Finally, to evaluate the results of the different methods, we sample 100 gaps from the test set. On these gaps, we compute the Root Mean Squared Error (RMSE) and the Mean Bias Error (MBE) between the methods results and the ground-truth EC measurements. The MBE is the mean of all the errors on a given gap. For each gap the percentage of improvement is computed and the mean and standard deviation of the improvement is also reported. The metrics are computed per gap and then averaged. Additionally, we compute the standard deviation for each of the metrics. The objective of the MBE metric is to make sure that the model has a zero centered error. In irrigation, it is used to indicate the bias induced on the daily or seasonal sums of the water loss. Finally, to ensure the repeatability of our results, all our models are trained 3 times with a different optimizer seed, different test samples, and different training batch orders. The presented results are an average of the 3 runs.

4.3 Neural networks and Training

When training on the small sequences, we used a batch size of 256, a learning rate of 0.0001, and dropout of 0.85. When training on medium sequences, the batch size was reduced to 128, and when training on large sequences, the batch size was set to 64. Going over those values resulted in tensors too large to be processed. For our model on the real data, the input embedding is of size 128, d_{model} is of size 512, the first dense layer after it has a size of 128, and the last dense layer as a size of 32.

Seq Size	Methods	RMSE		MBE	
		mean	std	mean	std
S	Ours	**0.09**	**0.021**	**0.008**	**0.021**
	MLP	0.16	0.041	0.0016	0.044
	KNN	0.48	0.21	-0.016	0.21
M	Ours	**0.12**	**0.04**	**-0.004**	**0.026**
	MLP	0.25	0.08	-0.007	0.096
	KNN	0.54	0.21	0.011	0.23
L	Ours	**0.25**	**0.09**	-0.007	**0.033**
	MLP	0.41	0.09	-0.016	0.15
	KNN	0.69	0.19	**0.04**	0.34

Table 1: RMSE and MBE of our model and KNN on the toy problem. Our problem easily bests the KNN approach.

We implemented our model in TensorFlow [1] 1.14 with a tensorboard front-end allowing us to visualize the evolution of the attention-heads over time along with the loss and accuracy. All experiments were carried out on an IBM Power Systems AC922 with 256 GB of RAM and 4 NVIDIA V100 16 GB GPU (using only a single GPU).

5 Results

5.1 Toy Case

On the toy problem, our approach outperforms both MLP and the KNN algorithm despite the larger view horizon of the KNN. As can be seen in table 1, which summarizes the results for the different gap-size, our approach is consistently better than KNN and MLP across all metrics except for the MBE on large sequence sizes. This can be explain by the nature of the MBE metric: it is the mean of the error, thus one value can slightly change the overall result even more here since the error values are very small. Hence it is more important to focus on the variance of the MBE as it depicts how is fluctuates over various gaps. Fig 3 compares the KNN method (in blue) to our architecture (in orange). We can see that our approach better fits the data. Our attention-based model is able to extract the local trend, whereas KNN is being tricked by the long term amplitude changes created by the latent variable z_1.

5.2 Evapotranspiration Data

Table 2 summarizes our models' performance on the EC datasets. Based on the RMSE results our model performs statistically better or as well as the reference method: REddyProc. Only on large gaps on Cotton2 does it performs slightly worse in average but the difference is not statistically significant. On an other hand, our model performs much better than the baseline on the Tomato crops. This is particularly visible on the Tomato 2 experiment, which yields an average improvement of 30% across all gaps. This is interesting as tomatoes are summer crops with a quick canopy growth. This,

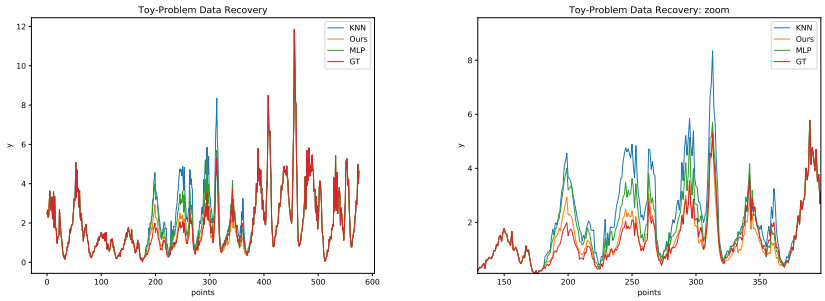

Fig. 3: Gap filling quality comparison of our model (NN), with KNN on the toy problem on large gaps.

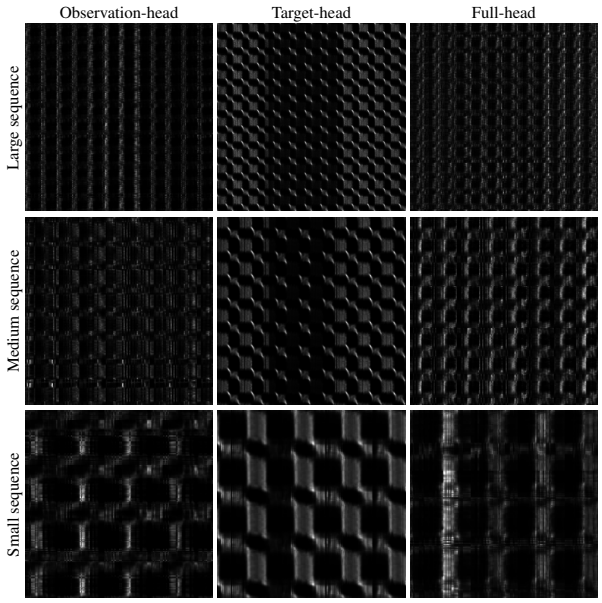

Fig. 4: Attention weights for the different sequence sizes on real data. The periodicity of the data was understood and leveraged by the attention mechanism.

particularities lead to faster ET dynamics than the one encountered in the other crops present in this dataset. The superior performance of our network on this crop shows that our approach performs well on system with high dynamics, which was our original goal. Additionally, the constant performance of our network demonstrates that, despite its smaller time horizon, our architecture is more reliable than REddyProc. Regarding the MBE our model performs as well as REddyProc. Overall, the MBE quantifies the irrigation bias but not the accuracy of the prediction, hence our prediction keeps similar performances.

Crops	Seq Size	Methods	RMSE		RMSE Improvements	MBE	
			mean	std		mean	std
Cotton 1	S	Ours	**44.2**	**13.2**	+12% ± 17%	**-0.9**	**8.4**
		REddyProc	51.8	17.4		2.1	13.8
	M	Ours	**53.9**	**13.4**	+7% ± 14%	**-3.9**	13.9
		REddyProc	57.0	13.5		9.1	**12.2**
	L	Ours	**46.7**	**7.62**	+10% ± 10%	**-0.5**	7.4
		REddyProc	52.9	12.7		0.9	**5.7**
Cotton 2	S	Ours	**43.2**	**13.9**	+10% ± 15%	-0.7	**12.0**
		REddyProc	48.1	13.9		**0.1**	14.0
	M	Ours	**48.0**	**11.9**	+4% ± 22%	**0.1**	12.1
		REddyProc	51.0	12.0		-5.5	**9.7**
	L	Ours	49.5	14.6	-10% ± 49%	2.2	12.3
		REddyProc	**47**	**10.9**		**-1.1**	**8.6**
Tomato 1	S	Ours	**30.9**	**14.0**	+29% ± 19%	-10.5	**11.7**
		REddyProc	54.6	24.0		**-4.1**	19.3
	M	Ours	**31.0**	**4.2**	+35% ± 12%	-2.6	**4.8**
		REddyProc	49.9	14.9		**0.6**	10.7
	L	Ours	**44.6**	**9.1**	+35% ± 12%	-4.4	**5.4**
		REddyProc	71.9	23.8		**-3.3**	12.8
Tomato 2	S	Ours	**32.4**	**9.1**	+19% ± 25%	**-1.2**	**9.1**
		REddyProc	42.1	13.2		-3.4	16.6
	M	Ours	**36.4**	**6.88**	+14% ± 20%	**-1.6**	**8.5**
		REddyProc	44.2	12.34		-5.9	13.0
	L	Ours	**37.8**	**4.1**	+13% ± 15%	**1.0**	**5.9**
		REddyProc	44.4	9.2		-1.6	9.8
Wheat 1	S	Ours	**37.9**	**12.2**	+7% ± 30%	-8.9	**9.4**
		REddyProc	46.44	13.58		**-7.6**	22.5
	M	Ours	38.0	**8.1**	+0% ± 25%	**-4.3**	**4.8**
		REddyProc	**37.5**	11.4		6.5	5.8
	L	Ours	**37.9**	**4.6**	+0% ± 13%	-8.9	**4.2**
		REddyProc	38.7	6.6		**-1.7**	8.5
Wheat 2	S	Ours	**26.0**	**7.2**	+0% ± 30%	4.5	**6.3**
		REddyProc	28.42	13.6		**-1.8**	10.3
	M	Ours	**27.9**	**4.8**	+2% ± 21%	7.5	**7.4**
		REddyProc	29.9	9.1		**-0.8**	9.2
	L	Ours	**29.9**	**3.3**	+7% ± 17%	7.2	**4.9**
		REddyProc	31.9	7.0		**-2.8**	6.9

Table 2: RMSE and MBE of of our model and REddyProc applied on real EC data (lower RMSE and MBE values indicate better model performance). Values range from -50 to 800.

Figure 4 shows examples of attention matrices for the different attention heads. Each of those matrices translates the cross-correlation between the elements of the same sequence. Let us define A, an attention matrix for a sequence of size k s.t $A \in R^{k \times k}$ and $i, j \in [1, k]$, the position of two elements inside that same sequence; then the value $A(i, j)$ is a measure of how strong the correlation between the elements i and j is. The

brighter the pixels in the image, the stronger the correlation. The attention matrices of our networks present periodic patterns. The periodic patterns show that our architectures are leveraging the periodicity of the data to fill the missing values in our sequences. One can also see that on the target-heads the center of the matrix is less bright than on the other heads. This is due to the fact that the target heads avoid using the values inside the gaps.

If one looks at the last row of figure 4, one can see some patterns and at some point a variation in that pattern, this can easily be seen on the target-head's weights where a black band appears. This is where the gap is located inside the data. What this black band means, is that the network learns not to use data where there are gaps in order to fill the missing values. Similar things can be seen on the the medium and large sequences (on the target head), but the gap is not fully black. Instead during the nights the network is still trying to make use of the data. This is probably due to the network having difficulties detecting the gap or to a lack of training data. On the other heads, similar pattern can be seen but the gap is not clearly visible. However, what is visible are the nights: during the nights the values are homogeneous and the variables all have the same weights. This is represented by this darker bands that can be seen in the different heads weights.

Finally, using this training-testing split, we show that our network achieves solid performances without even training on the dataset which we aim to fill. This demonstrates the strong generalization capacities of our method and makes it almost as convenient as the MDS since its application would be training-free.

6 Conclusion

A novel, data-driven, gap filling method that relies on multi-head attention is introduced in the context of evapotranspiration measurements of field crops. Our method performed better than the current state of the art when tested on both a toy problem and a real-world scenario using evapo-transpiration data. Furthermore, the data-efficiency of our method allows to achieve these results even with very small training datasets. Finally, the generality of this innovative approach is demonstrated in a real-world scenario where we learn a model on a set of crops and use the knowledge learned on those crops to successfully fill gaps on an other crop type. Future efforts will focus on creating a simple framework for the benefit of users outside the machine learning community to train and apply these models for any time-series.

Acknowledgements

We would like to thank Lionel Clavien from our partner InnoBoost SA in Switzerland for providing us access to the server used for our experiments as well as some starting help on the platform. This work was supported by a grant from the Ministry of Science and Technology (MOST), Israel, under the France-Israel Maimonide Program, Ministry of Europe and Foreign Affairs (MEAE), and the Ministry of Higher Education, Research and Innovation (MESRI) of France.

References

1. Abadi, M., Agarwal, A., Barham, P., et al.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), http://tensorflow.org/, software available from tensorflow.org
2. Che, Z., Purushotham, S., Cho, K., Sontag, D., Liu, Y.: Recurrent neural networks for multivariate time series with missing values. Scientific reports **8**(1), 6085 (2018)
3. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
4. Coutinho, E.R., Silva, R.M.d., Madeira, J.G.F., Coutinho, P.R.d.O.d., Boloy, R.A.M., Delgado, A.R.S., et al.: Application of artificial neural networks (anns) in the gap filling of meteorological time series. Revista Brasileira de Meteorologia **33**(2), 317–328 (2018)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
6. Falge, E., Baldocchi, D., Olson, R., Anthoni, P., Aubinet, M., Bernhofer, C., Burba, G., Ceulemans, R., Clement, R., Dolman, H., et al.: Gap filling strategies for long term energy flux data sets. Agricultural and Forest Meteorology **107**(1), 71–77 (2001)
7. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1243–1252. JMLR. org (2017)
8. Graves, A., Fernández, S., Schmidhuber, J.: Bidirectional lstm networks for improved phoneme classification and recognition. In: International Conference on Artificial Neural Networks. pp. 799–804. Springer (2005)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
11. Lange, M., Dechant, B., Rebmann, C., Vohland, M., Cuntz, M., Doktor, D.: Validating modis and sentinel-2 ndvi products at a temperate deciduous forest site using two independent ground-based sensors. Sensors **17**(8), 1855 (2017)
12. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proc. icml. vol. 30, p. 3 (2013)
13. Moffat, A.M., Papale, D., Reichstein, M., Hollinger, D.Y., Richardson, A.D., Barr, A.G., Beckstein, C., Braswell, B.H., Churkina, G., Desai, A.R., et al.: Comprehensive comparison of gap-filling techniques for eddy covariance net carbon fluxes. Agricultural and Forest Meteorology **147**(3-4), 209–232 (2007)
14. Papale, D., Valentini, R.: A new assessment of european forests carbon exchanges by eddy fluxes and artificial neural network spatialization. Global Change Biology **9**(4), 525–535 (2003)
15. Reichstein, M., Falge, E., Baldocchi, D., Papale, D., Aubinet, M., Berbigier, P., Bernhofer, C., Buchmann, N., Gilmanov, T., Granier, A., et al.: On the separation of net ecosystem exchange into assimilation and ecosystem respiration: review and improved algorithm. Global Change Biology **11**(9), 1424–1439 (2005)
16. Richard, A., Mahé, A., Pradalier, C., Rozenstein, O., Geist, M.: A comprehensive benchmark of neural networks for system identification (2019)
17. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
18. Wutzler, T., Lucas-Moffat, A., Migliavacca, M., Knauer, J., Sickel, K., Šigut, L., Menzer, O., Reichstein, M.: Basic and extensible post-processing of eddy covariance flux data with reddyproc. Biogeosciences **15**(16), 5015–5030 (2018)