Spatio-Temporal Forecasting With Gridded Remote Sensing Data Using Feed-Backward Decoding

Mário Cardoso
IST and INESC-ID
University of Lisbon
Lisbon, Portugal

ABSTRACT

This article presents a novel deep learning approach for spatio-temporal forecasting with remote sensing data. I specifically propose a neural network architecture derived from a previous model named Spatio-Temporal Convolutional Sequence to Sequence Network (STConvS2S), which is entirely based on convolutions, extending it in several directions. In particular, besides considering optimizations such as training based on AdaMod or recently introduced normalization-activation layers, I propose to replace the decoder component of STConvS2S, based on temporal convolutions, with an alternative that leverages a recurrent backbone based on LSTMs together with the idea of feed-backward decoding, specifically by re-using the weights of the convolutions from the encoder, when generating the predictions from intermediate representations. Experiments using datasets from previous studies, consisting of observations of air temperature and rainfall, show that the proposed Spatio-Temporal Recurrent Feed-Backward Decoding (ST-RFD) architecture significantly outperforms STConvS2S and other baseline models, in tasks related to forecasting future observations. On experiments related to the reconstruction of missing time-steps, some of the proposed extensions lead to improvements over the original STConvS2S, although ST-RFD failed to outperform other models. I believe the two tasks differ in the way short-term and long-term context should be used, motivating additional research into how models can be made to better aggregate both types of information.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; **Supervised learning by regression**; • **Applied computing → Earth and atmospheric sciences**.

KEYWORDS

Deep Learning, Spatio-Temporal Forecasting, Remote Sensing

ACM Reference Format:
Mário Cardoso. 2020. Spatio-Temporal Forecasting With Gridded Remote Sensing Data Using Feed-Backward Decoding. In *28th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '20), November 3–6, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3397536.3422247

1 INTRODUCTION

A widespread collection of satellites are nowadays being used to observe the Earth, monitoring natural systems and man-made structures by measuring a variety of variables at consistent time intervals and spatial resolutions. Given this influx of remote sensing data, machine learning approaches are becoming commonplace for various practical applications. In this study, I focus specifically on forecasting tasks, consisting on the prediction of particular time-steps in a series of gridded remote sensing data (e.g., forecasting future values conditioned by past observations, or reconstructing missing time-steps). Common examples of applications for these methods include forecasting meteorological variables, such as precipitation, air temperature, wind speed, among many others, or reconstructing missing data, e.g. due to cloud coverage, for environmental indices such as NDVI. This type of data often contain a mixture of spatial and temporal dependencies, indicating how different spatial locations exhibit related patterns (in particular, spatial neighbours tend to be related) and how the observed variables evolve over time. Due to the stochastic nature of the underlying variables, prediction models must be capable of capturing complex and non-linear patterns, while simultaneously leveraging the aforementioned dependencies in order to obtain accurate results.

Neural network models for predicting gridded values of remote sensing observations are currently gaining increased popularity. These include approaches such as ConvLSTM [25], capable of capturing both spatial and temporal contexts by combining ideas from Recurrent Neural Network (RNN) architectures with the convolution operation typically seen in Convolutional Neural Networks (CNNs) for image processing. Given these properties, ConvLSTM units have become a basic building block for a variety of neural architectures proposed in recent studies dealing with forecasting from gridded spatio-temporal data [16, 29, 35].

More recently, Nascimento et al. [20] studied effective and computationally efficient alternatives to capture both spatial and temporal patterns using exclusively convolutional structures, proposing an encoder-decoder model named Spatio-Temporal Convolutional Sequence to Sequence Network (STConvS2S), comprised of sequential convolutional blocks with factorized filters. Each convolutional 3D filter is factorized into separate 2D and 1D filters, with the encoder using the former to model the spatial context, and the decoder using the latter to model the temporal context.

This paper proposes several extensions to the STConvS2S architecture, leveraging recent advances in the literature in order to further improve results. Specifically, I propose the following main extensions to the original model:

- Replace the RMSProp optimizer, originally used by Nascimento et al. when training STConv2S2 models, with the more recent AdaMod [8] optimizer;
- Replace the traditional normalization and activation operations with an adapted version of the batch-based evolved normalization-activation layer from Liu et al. [17].
- Replace the decoding component from STConvS2S, based on dilated temporal convolutions, with a recurrent block that leverages feed-backward decoding [28] (i.e., an idea recently proposed in the context of semantic segmentation models for images, in which weights from the 2D convolutions used in encoder layers of the model are used in the reverse direction to form the decoder, reducing the total number of model parameters required for processing).

The proposed additions to the STConv2S2 architecture were evaluated on tasks related to predicting patches of observations that are one or five time-steps in the future, given a sequence of patches with previous observations. Moreover, the proposed model was also evaluated in a missing data completion scenario, where the task corresponds to reconstructing a complete input sequence with a random time-step missing. For both scenarios, tests relied on the CHIRPS [10] rainfall dataset and on the CFSR [24] air temperature dataset, also used in the forecasting experiments from Nascimento et al. [20]. The results show that the proposed extensions all contribute to improved predictions, leading to new state-of-the-art results on the aforementioned datasets. In particular, the model using Spatio-Temporal Recurrent Feed-Backward Decoding (ST-RFD) significantly outperformed STConvS2S and other baselines in tasks related to forecasting future observations. On the experiments related to the reconstruction of a missing time-step, some of the proposed extensions lead to improvements over the original STConvS2S, although ST-RFD failed to outperform other models. I believe the two tasks differ in the way short-term and long-term context should be used, motivating additional research into how models can be made to better aggregate both types of information.

The rest of this article is organized as follows: Section 2 provides an overview on previous research in the area. Section 3 details the extensions to STConvS2S considered in our approach. Section 4 presents the evaluation methodology, detailing the datasets and the obtained results. Finally, Section 5 provides concluding remarks and discusses possible directions for future work.

2 RELATED WORK

Conventional Recurrent Neural Network (RNN) architectures, e.g., based on Long Short-Term Memory (LSTM) units [12], are commonly employed on forecasting tasks involving time-series data. However, these models consider the input data as sequences of vectors, thus not exploiting the spatial context present in spatio-temporal structures (e.g., raster representations for remote sensing data, consisting of patches with neighbouring cells) provided as input. To address this limitation, Shi et al. [25] combined convolution operations with LSTMs, simultaneously exploiting the abilities of Convolutional Neural Networks (CNNs) and RNNs to effectively model spatial and temporal information, respectively. In the proposed Convolutional LSTM (ConvLSTM) approach, all input data structures are 3D tensors, with the first dimension corresponding to either the number of measurements or the number of feature maps, and the last two dimensions representing the spatial dimensions (i.e., width and height). By replacing the product operation in the original LSTM with the convolution operation, denoted as $*$ in the equations shown next, the future states of a certain cell are now defined as a function of the inputs and past states of its local neighbours. Consider that I, F, O, and G denote the four standard LSTM gates/operations, t represents a time-step, $\odot$ denotes an element-wise product, H and C represent the hidden state and cell state, respectively, and that X represents the input. The ConvLSTM is formally defined as follows.

$$\begin{aligned}
I_t &= \sigma\left(W_{hi} * H_{t-1} + W_{xi} * X_t + W_{ci} \odot C_{t-1} + b_i\right)\\
F_t &= \sigma\left(W_{hf} * H_{t-1} + W_{xf} * X_t + W_{cf} \odot C_{t-1} + b_f\right)\\
O_t &= \sigma\left(W_{ho} * H_{t-1} + W_{xo} * X_t + W_{co} \odot C_t + b_o\right)\\
G_t &= \tanh\left(W_{hg} * H_{t-1} + W_{xg} * X_t + b_g\right)\\
C_t &= F_t \odot C_{t-1} + I_t \odot G_t\\
H_t &= O_t \odot \tanh(C_t)
\end{aligned} \tag{1}$$

A variety of complex architectures can be built using ConvLSTM building blocks [1, 13, 16, 29, 35]. In the original study, the authors developed a typical encoder-decoder structure comprised of multiple stacked ConvLSTM layers, with each decoder layer initializing its hidden states from the output of the corresponding encoder layer. Final predictions are given by the concatenation of the hidden states from the decoder network, followed by a 1×1 convolution. This architecture was applied on a radar echo dataset for precipitation nowcasting, considering the task of predicting the next 5 time-steps given the previous 5. Although the predictions were blurrier than those from other baseline approaches, the ConvLSTM model clearly outperformed every model under comparison, reacting better to sudden changes (i.e., more extreme values) in the inputs, and overall achieving more accurate results.

Inspired by the aforementioned ConvLSTM architecture, Wang et al. [29] proposed another recurrent model named PredRNN, which captures spatial and temporal features in a unified memory pool. In PredRNN, the states of an adapted LSTM cell can travel along both vertically between layers and horizontally across states. The authors introduced a new cell state in each LSTM unit, i.e. the spatio-temporal memory cell state M_t, that flows in a zigzag direction, first upwards across layers and then forwards over time. This extension to standard LSTM units, called Spatio-Temporal Long Short-Term Memory (ST-LSTM), allows simultaneous flow of both spatial and temporal memory, enabled by the state M_t, and the standard temporal memory, enabled by the state C_t, present in traditional ConvLSTMs. In order to maintain both memory cell states, a new set of gates was constructed to manage the information flow for M_t, in addition to the original gates that handle C_t. Let C_t^l denote the standard temporal cell state at layer l, delivered from the previous unit at $t-1$, and let M_t^l denote the spatio-temporal memory cell state, delivered vertically from the $l-1$ layer at time-step t. The ST-LSTM cells are formally defined as follows.

$$\begin{aligned}
I_t &= \sigma\left(W_{hi} * H^l_{t-1} + W_{xi} * X_t + b_i\right)\\
F_t &= \sigma\left(W_{hf} * H^l_{t-1} + W_{xf} * X_t + b_f\right)\\
G_t &= \tanh\left(W_{hg} * H^l_{t-1} + W_{xg} * X_t + b_g\right)\\
C^l_t &= F_t \odot C^l_{t-1} + I_t \odot G_t\\
I'_t &= \sigma\left(W_{mi} * M^{l-1}_t + W'_{xi} * X_t + b'_i\right)\\
F'_t &= \sigma\left(W_{mf} * M^{l-1}_t + W'_{xf} * X_t + b'_f\right)\\
G'_t &= \tanh\left(W_{mg} * M^{l-1}_t + W'_{xg} * X_t + b'_g\right)\\
M^l_t &= F'_t \odot M^{l-1}_t + I'_t \odot G'_t\\
O_t &= \sigma\left(W_{ho} * H^l_{t-1} + W_{xo} * X_t + W_{co} * C^l_t + W_{mo} * M^l_t + b_o\right)\\
H^l_t &= O_t \odot \tanh(W_{1\times1} * [C^l_t, M^l_t])
\end{aligned} \tag{2}$$

The final hidden states are defined as the concatenation of each memory state derived from different directions, represented as $[C^l_t, M^l_t]$, followed by a 1×1 convolution to ensure consistent dimensionality between states. The authors defined the PredRNN as a multi-layer architecture employing ST-LSTMs, having tasked the model with predicting 10 future observations from a radar echo dataset, given the previous 10 observations. Results showed that while other baselines provided more accurate results for the near future, they quickly deteriorated afterwards. The PredRNN was able to maintain a consistent level of performance, achieving better results overall when compared to other approaches.

Zhao et al. [35] described two baseline architectures for spatio-temporal forecasting of air pollutants conditioned on metereological variables, reporting on experiments with a dataset containing a variety of air pollutant and metereological data from China. The first architecture, called ReducedLSTM, is a relatively simple application of LSTM units, disregarding spatial information. Let f, i, c and h denote the forget gate, the input gate, the cell state and the hidden state, respectively. Consider also that x denotes the input metereological variables (e.g., temperature, humidity, etc...) for a given cell, and let mean_t denote the moving average value of the air pollutant concentration for that same cell. The ReducedLSTM architecture can be formally defined as follows.

$$\begin{aligned}
f_t &= \sigma\left(W_{hf} \cdot h_{t-1} + W_{xf} \cdot x_t + b_f\right)\\
i_t &= W_{hi} \cdot h_{t-1} + W_{xi} \cdot x_t + b_i\\
c_t &= f_t \odot \text{ReLU}\left(c_{t-1} + i_t\right)\\
h_t &= c_t - \text{mean}_t
\end{aligned} \tag{3}$$

The second architecture proposed by Zhao et al., called WipeNet, exploits spatial information by incorporating the convolution operation with the ReducedLSTM, similarly to the aforementioned ConvLSTM. The authors simulate pollutant transportation in the atmosphere through the use of location-specific redistribution filters, denoted as RK in the equation shown next, calculated using the meteorological variables associated to wind, such as wind speed and wind direction. These redistribution filters are afterwards combined with the predicted pollutant concentration, denoted as $\hat{C}_t$, to yield location sensitive pollutant values. The WipeNet architecture is formally defined as follows.

$$\begin{aligned}
F_t &= \sigma\left(W_{hf} \cdot H_{t-1} + W_{xf} \cdot X_t + b_f\right)\\
I_t &= W_{hi} \cdot H_{t-1} + W_{xi} \cdot X_t + b_i\\
\hat{C}_t &= F_t \odot \text{ReLU}\left(C_{t-1} + I_t\right)\\
RK_t &= \text{reshape}\left(\text{softmax}\left(W_{rk} * X^{Wind}_t\right)\right)\\
C_t &= RK_t * \hat{C}_t\\
H_t &= C_t - \text{mean}_t
\end{aligned} \tag{4}$$

Experiments showed that both proposals achieved superior results compared to simpler baselines. In particular, WipeNet outperformed all the other models, which the authors attribute to the correct formulation of the redistribution filters, enabling the model to better leverage spatial dependencies in the data.

Das et al. [3] proposed an architecture inspired by the Deep Stacking Network (DSN) from Deng and Yu [6], adapting the original idea for spatio-temporal forecasting. The proposed model, called Deep-STEP, is comprised of T stacked modules, where T corresponds to the number of time-steps in the data. The input data for each module is first prepared, so that each instance represents a cell in terms of its spatio-temporal features (i.e., in terms of the values for spatio-temporally neighbouring cells, from previous time-steps and nearby locations). The first module receives as input the raw cell representations, whereas the subsequent modules process a concatenation of the cell representations plus the results from the previous module. Each module is a Multi-Layer Perceptron (MLP) with a single hidden layer and two weight matrices, W and U, representing a lower-layer weight matrix connecting the input and the hidden layer, and an upper-layer weight matrix connecting the hidden layer and the output, respectively. The output Y of each module is thus defined as follows.

$$Y = \sigma\left(\sigma\left(X \cdot W^T\right) \cdot U^T\right) \tag{5}$$

The sigmoid functions ensure the output values are contained within the range $[0, 1]$. At each module, to ensure consistency between the outputs and the raw cell representations, the resulting merged input tensor X is normalized prior to being processed by the module. The normalization is defined as follows.

$$\text{norm\_}x_{ij} = \frac{\left(x_{ij} - \min(X)\right)}{(\max(X) - \min(X))} \tag{6}$$

The final module outputs each cell's prediction for the desired time-step, as a value within the range $[0, 1]$ which is then mapped to the original scale, resulting in the final predictions.

The Deep-STEP architecture was evaluated in comparison against a traditional MLP, the original DSN [6], and the NARNET model (i.e., the nonlinear autoregressive neural network model that is provided by MATLAB), on a task related to predicting the Normalized Difference Vegetation Index (NVDI) in the year 2011, given past annual observed NVDI measurements for 2004 – 2010. During training, the images from 2004 – 2009 were used to prepare the feature set supporting a prediction for the year of 2010. In turn, during testing, the images from 2004 – 2010 were used to prepare the feature set supporting predictions for the year of 2011. The obtained results showed that Deep-STEP produced lower errors at competitive values for the execution time, clearly outperforming the original DSN architecture [6].

In subsequent work, Das et al. [5] proposed a self-adaptive architecture capable of dynamically adjusting the network structure based on the input data. The architecture features three modules, capturing temporal and spatial features in parallel. The first module computes a representation for each cell based on its neighbouring values, and the results are then arranged as inputs to the subsequent modules. The second module, corresponding to a self-adaptive RNN variant named SARDINE that relies on teacher forcing and on the ReLU activation function in the hidden layers, captures temporal dependencies in the results from the first module, processing the cell representations for each timestamp. In turn, the third module captures spatial dependencies in neighbouring cells.

Contrarily to a standard RNN, SARDINE can dynamically change the number of hidden layers and units, by assessing the module's ability to generalize with a Network Significance (NS) method [4], defined as the sum between the variance and squared bias of the predictions at each time-step. A high NS value indicates over-fitting, and the module reacts by pruning the least significant hidden unit of the top-most hidden layer. In turn, a low NS value indicates under-fitting, to which the module reacts by growing new hidden units at the top-most hidden layer. Besides pruning/growing hidden units, SARDINE is also capable of growing new layers, allowing for more complex functions to be learned. Whenever a drift in spatial context is detected, through an adapted version of Hoeffding's error bound technique [9], a new layer is added on top of the current top-most hidden layer, with the same number of hidden units, enhancing the network's ability to react to sudden spatial changes. The weight matrix between the new layer and the output layer is initialized with the same values as the weights between the previous layer and the output layer. In turn, the weights between the new layer and previous layer are initialized with an identity matrix.

The third module captures spatial dependencies (i.e., how neighbouring locations affect each cell). It receives the spatial information from a given neighbourhood for each target cell at a specific time-step, and outputs the predicted value for the cell. The module corresponds to a standard MLP, with the hidden layer containing half the units of the input layer, and the output layer containing a single unit that outputs the final predicted value for each cell.

The model was evaluated on three randomly selected areas from two datasets of annual NVDI time series imagery, one pertaining to the state of West Bengal, in India, for 2004 – 2011, and the other pertaining to the Northern part of Brazil, for 2012 – 2019. The model was compared to baselines such as a traditional MLP, a mixed CNN-RNN architecture, the aforementioned standard DSN, and Deep-STEP, among others. Results showed that the proposed model outperformed the baselines in almost all scenarios, being considerably faster than CNN/LSTM baselines, while slower than models based on MLPs (e.g., DSN or Deep-STEP).

Zhang et al. [32] proposed a deep learning model based on residual connections to predict the traffic flow of crowds. The model takes as input sequences of 2-channel matrices, with each cell being associated to a geo-spatial region, and the two channels corresponding, respectively, to crowd inflow and outflow (i.e., traffic entering or leaving the region, respectively). The input is sliced into three chronologically ordered subsets of data, denoting distant history, near history, and recent time. The proposed neural architecture, named ST-ResNet and leveraging 2D convolutions, is comprised of four components, respectively modeling temporally *close* information, information from a longer *period*, *trends*, and influence from *external* factors, with the former three components sharing a network structure. The three subsets of data are fed into a respective component, with the trends component receiving the distant history, the period component receiving the near history, and the closeness component receiving the remaining subset. These three components are comprised of two convolutional blocks, with a sequence of residual units between them. The first convolutional block processes a tensor corresponding to the concatenation of the input patches for the different time-steps. Each residual unit combines multiple convolution operations together with ReLU activation functions and batch normalization operations, featuring also a skip connection that adds the input of the unit to the result of the last convolution. The residual units allow for the construction of a deeper network, better modeling spatially distant dependencies (i.e., each cell in the final feature map depends on all cells of the input grid). The final convolutional block produces an output patch for the component, with a number of channels corresponding to the prediction objective.

The outputs from the closeness, period, and trend components are combined through learnable weight matrices that assign different degrees of influence to each aspect. The external influence component models potentially useful information from external datasets (e.g., meteorological conditions, patterns associated to holidays or specific days of the week, etc.), with basis on a two-layer MLP. The output of this component is added to the aforementioned combined output, resulting in the predictions for the respective time-step (i.e., the model can be applied sequentially, in order to produce predictions for multiple time-steps from past observations).

ST-ResNet was evaluated on two datasets, each containing information regarding trajectories and external conditions (i.e., meteorological data and holidays). The first dataset contained taxi trajectories in Beijing, considering four different time intervals from 2013 – 2016. In turn, the second dataset contained bike trajectories in New York City from April to September 2014. Comparisons were made against traditional approaches such as ARIMA, SARIMA, and VAR models, and against neural models corresponding to a MLP or to a previous state-of-the-art approach for crowd flows prediction, named DeepST [33]. The authors also evaluated different configurations for ST-ResNet, varying the external information and the number and/or internal structure of the residual units. Results showed that the ST-ResNet model outperformed all the baselines, with the best versions obtaining significant improvements over the previous best model for each dataset.

Recently, Nascimento et al. [20] proposed an encoder-decoder model for spatio-temporal forecasting that is comprised exclusively of convolutional layers, which are suited to capture spatial features by design. In STConvS2S, illustrated in Figure 1, convolutions are performed with factorized 3D kernels $K = t \times d \times d$, where t is the size of the temporal kernel and d is the size of the spatial kernel. The

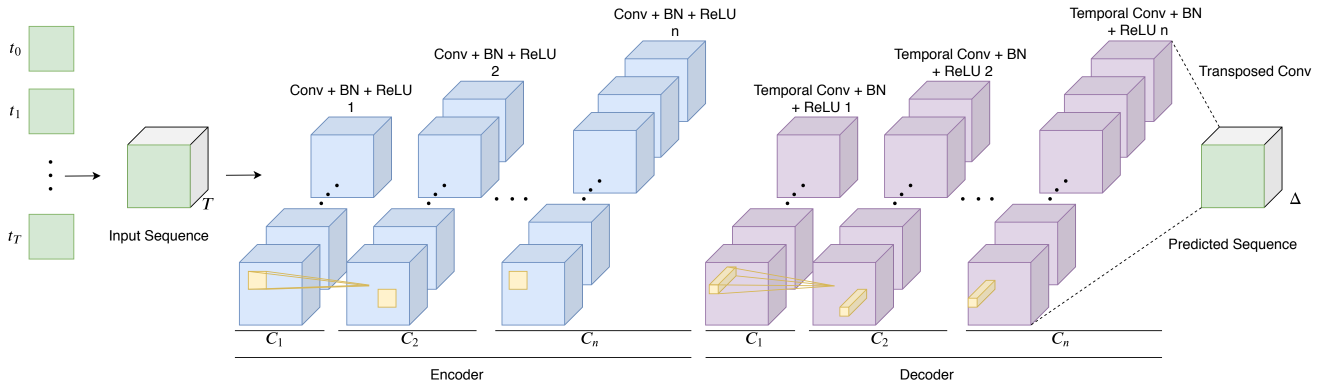

Figure 1: Illustration depicting the STConvS2S architecture, adapted from Nascimento et al. [20]. The yellow elements represent the factorized 3D kernel, T represents the number of time-steps, and C_l is the number of channels.

encoder processes the input sequence by performing convolutions using just the spatial kernel (i.e., $1 \times d \times d$, with $d = 5$ in the best configuration reported by Nascimento et al.), creating a sequence of meaningful spatial representations. These spatial representations are received as input by the decoder, which applies temporal convolutions with the temporal kernel (i.e., $t \times 1 \times 1$) in order to learn the temporal features, resulting in the predicted future sequence. Both the encoder and the decoder are comprised of successive convolutional blocks with batch normalization [15] and followed by a ReLU activation function, with the decoder utilizing causal temporal convolutions [21] to maintain temporal coherency during prediction (i.e., predictions for a time-step t make no use of future information from time-steps $t + 1$ onward). Given that standard temporal convolution operations output either a shorter sequence or a sequence of the same size as the input, the authors introduce a transposed convolution operation before the final convolutional layer, allowing for predictions that exceed the sequence length of the input. Experiments were performed on two subsets from the CFSR [24] air temperature dataset and the CHIRPS [10] precipitation dataset. The authors considered the tasks of predicting the next 5 and 15 time-steps, given the previous 5, for both datasets. Results showed that STConvS2S consistently outperformed the aforementioned ConvLSTM, with the best reported version on the air temperature dataset obtaining a 20% improvement in terms of the Root Mean Square Error (RMSE) metric, and training approximately 2.5× faster.

3 PROPOSED EXTENSIONS TO STCONVS2S

This section details the main extensions proposed over the original STConvS2S architecture, first describing the novel decoding strategy based on combining a LSTM with the idea of feed-backward decoding, and then presenting new normalization-activation layers.

3.1 Feed-Backward Decoding

When using an encoder-decoder model for tasks involving spatial structures (e.g., tasks such as semantic segmentation of image inputs), it is common to have the encoder comprised of CNNs that process the input, progressively creating increasingly compact representations that capture meaningful features, which are afterwards sent as input to the decoder. In situations where retaining the spatial dimensions of the original input is crucial, such as spatio-temporal forecasting or semantic segmentation, the decoder typically applies transposed convolutions or some other interpolation technique to upscale the intermediate representations (e.g., bi-linear up-sampling followed by a traditional convolution operation).

Wang et al. [28] proposed a novel technique to maintain the original spatial dimensions, by using an encoder in the opposite direction to decode (i.e., feed-backward decoding). During decoding, the existing convolutional layers and filters of the encoder are re-used, allowing learned features to be mapped from smaller dimensions to larger dimensions, without additional parameters. To apply this technique, a couple of considerations need to be taken into account. For instance, any pooling operations used in the encoder need to be replaced by interpolation operations (e.g., bi-linear interpolation) during decoding. Also, if the channel dimension of the input has its size altered in a convolutional layer L, the weights used during decoding are the weights from L with the input and output channel dimensions permuted. Figure 2 illustrates the feed-backward decoding procedure, showing on the top part of the figure a typical convolution between a zero-padded input X with 2 channels and spatial dimensions of 4×4, and a set of 3, 3×3 filters with 2 channels each (i.e., a filter tensor with 2 input channels and 3 output channels). The result is a tensor Y with 3 channels and the same spatial dimensions as the input. During the decoding phase, the input-output flow is reversed, and the convolution is now performed over an input with the same dimensions as the output of the original convolution (plus zero-padding to guarantee consistent spatial dimensions). The same set of filters are re-utilized by swapping the input and output channel dimensions, yielding a set of 2, 3×3 filters with 3 channels each. With this technique, the network learns to both capture spatial features during encoding and also reverse its effect back to the original representation during decoding, with the regrouping of the filters dictating the behaviour.

Leveraging the idea of feed-backward decoding, I propose a variation of the STConvS2S model, here denoted as ST-RFD, incorporating the weight-sharing technique between the encoder and the decoder, as illustrated in Figure 3. This alternative allows us to exploit different approaches for capturing temporal features,

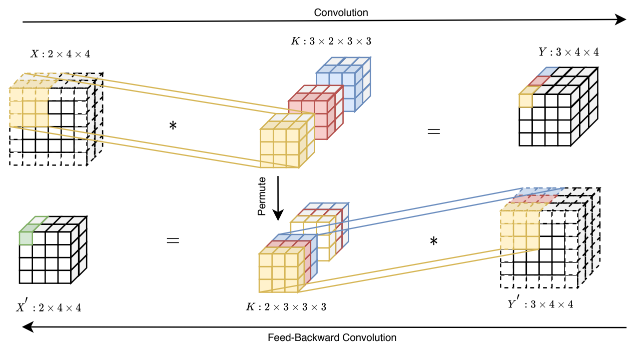

Figure 2: Illustration for how the dimensions of a filter can be permuted to transform the channel dimension. The cells with dotted borders correspond to zero-padding, and $*$ denotes a standard convolution with stride one. The middle row illustrates the normal vs. permuted convolution filters.

without incurring in an additional increase of learnable parameters through separate convolutions in the encoder and decoder parts.

In particular, given that the encoder is now also used for decoding, ST-RFD no longer captures temporal features through factorized temporal filters. Instead, we include an additional simple recurrent layer comprised of an LSTM unit that, at each time-step, receives a flattened representation of the learned spatial features, and outputs a representation for the prediction of the next time-step(s). During decoding, a component that shares its parameters with the encoder receives the predicted flattened representations and outputs the final predictions, with the information flowing from the last layer to the first layer. The recurrent component trivializes the prediction of sequences with a length differing from that of the input sequence, and the transposed convolutional layer from the STConvS2S architecture is not used. Besides the aforementioned changes, the original encoder structure from STConvS2S, relying on multiple 5×5 convolution operations, can be maintained, with the addition of a 1×1 convolutional layer at the end to reduce the dimensionality of the representations passed to the LSTM (i.e., we considered a final channel dimensionality of one while still preserving the spatial dimensions).

3.2 Novel Normalization-Activation Layers

In both the encoding and decoding parts of the ST-RFD architecture, novel normalization-activation layers are located after every convolution operation, except for the final convolutional layers along each direction (i.e., encoding or decoding). Specifically, instead of the ReLU activation function used in the original STConvS2S architecture, we can alternatively use the Mish activation function [19], which can be computed as shown next.

$$\text{Mish}(x) = x \cdot \tanh\left(\ln(1 + e^{x})\right) \tag{7}$$

Previous experiments have shown that Mish tends to work better than other activation functions such as ReLU, in many deep networks and across several challenging tasks/datasets.

We also considered replacing the activation functions plus the normalization operations, executed after the 5×5 convolutions

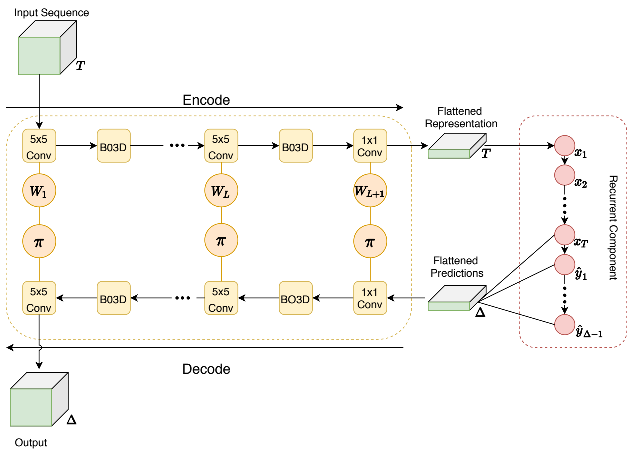

Figure 3: The ST-RFD model leveraging feed-backward decoding with $L + 1$ convolutional layers. The boxes labeled as B03D correspond to the proposed EvoNormB0 extension. The orange circles indicate weight-sharing, with π corresponding to a permutation of dimensions.

(i.e., the batch normalization operation), with the recently proposed Evolved Normalization-Activation (EvoNorm) layers [17]. We specifically use an adapted version of the EvoNorm B0 layer, which was the best performing batch-based version reported by Liu et al. [17]. Let v_1, γ and β denote learnable parameter vectors, and let $s_{b,h,w}$ and $s_{h,w}$ represent the variance of a mini-batch and the variance of a single instance, respectively. EvoNorm B0 uses the following computation over inputs x.

$$\text{B0} = \frac{x}{\max\left(\sqrt{s^2_{b,w,h}(x) + \epsilon}, v_1 \cdot x + \sqrt{s^2_{w,h}(x) + \epsilon}\right)} \cdot \gamma + \beta \tag{8}$$

The proposed extension to EvoNorm B0 explicitly models spatio-temporal scenarios, considering sequences d of two-dimensional inputs when calculating both the batch and instance variance (i.e., the values associated to each time-step in the input sequence are considered separately when computing the variance). This extension, denoted here as B03D, is defined as follows.

$$\text{B03D} = \frac{x}{\max\left(\sqrt{s^2_{b,d,w,h}(x) + \epsilon}, v_1 \cdot x + \sqrt{s^2_{d,w,h}(x) + \epsilon}\right)} \cdot \gamma + \beta \tag{9}$$

4 EXPERIMENTAL EVALUATION

We evaluated all the proposed extensions against baselines corresponding to the ConvLSTM and the original STConvS2S models, on the two datasets that were also used in the study by Nascimento et al. [20], namely the CFSR air temperature dataset and the CHIRPS precipitation dataset. The different models were evaluated in two different scenarios, with the first task corresponding to predicting a future sequence up to a fixed horizon Δ, given the previous sequence of 5 time-steps. In our experiments, we considered $\Delta = 1$ and $\Delta = 5$, i.e., predicting image patches corresponding to the next

	CHIRPS						**CFSR**					
	$\Delta = 1$			$\Delta = 5$			$\Delta = 1$			$\Delta = 5$		
	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2
ConvLSTM	6.4999	2.4065	0.1341	6.3874	2.3694	0.1628	2.4206	1.6836	0.9060	2.2369	1.5171	0.9217
STConvS2S	6.3769	2.3443	0.1701	6.3311	2.3421	0.1800	1.4172	0.9904	0.9684	1.5896	1.0491	0.9603
STConvS2S$_{\text{B03D}}$	6.3319	2.3445	0.1772	6.3424	2.3517	0.1734	1.3846	0.9520	0.9699	1.4603	1.0289	0.9665
ST-RFD	5.9244	2.2275	**0.2766**	6.1889	2.3351	0.2139	1.0479	0.7335	**0.9835**	**1.4450**	**1.0192**	**0.9672**
ST-RFD$_{\text{B0}}$	5.9280	2.2269	0.2721	6.1773	2.3227	0.2076	1.0837	0.7809	0.9820	1.4708	1.0259	0.9660
ST-RFD$_{\text{B03D}}$	**5.9205**	**2.2253**	0.2706	**6.1682**	**2.3182**	**0.2149**	**1.0461**	**0.7278**	0.9824	1.4582	1.0279	0.9666

Table 1: Results for the different models under consideration, on the two datasets and for the tasks related to predictign future time-steps. The subscripts B0 and B03D respectively correspond to the EvoNormB0 and EvoNormB03D extensions. The bold values indicate the best performing model for each dataset, prediction horizon, and evaluation metric.

time-step and next 5 time-steps, respectively. The second task corresponds to reconstructing a sequence with missing information. Specifically, we randomly remove an entire time-step from an input sequence of 10 time-steps, and the models are required to output the entire sequence without incomplete information.

The CHIRPS [10] dataset describes 13, 960 sequences of 10 instances for precipitation measurements on a 50×50 grid, each corresponding to a subset of the South American region (latitudes between 10°N and 39°S and longitudes between 84°W and 35°W). Data was collected from 1981 to 2019, measuring daily precipitation at a spatial resolution of 0.05°, and the original data was interpolated to a resolution of 1° per cell. In turn, the CFSR dataset [24] describes 54, 047 sequences for air temperature measurements in a 32×32 grid, pertaining to a similar region as the previous dataset (latitudes between 8°N and 54°S and longitudes between 80°W and 25°W). Data was collected from 1979 to 2015, measuring temperature every 6 hours at a spatial resolution of 0.5°. For both datasets, and similarly to Nascimento et al. [20], we adopted a splitting strategy involving 60% - 20% - 20% of the data respectively for training, validation and testing, in chronological order.

In the tests with the STConvS2S baseline architecture, we used the best performing version reported by Nascimento et al. [20], consisting of 3 convolutional layers on both the encoder and decoder (plus the final 1×1 convolution), each containing 32 filters of dimensionality 5×5. Similarly, the ConvLSTM baseline consists of 3 layers with 32 hidden states and filters of dimensionality 5×5.

In terms of hyper-parameter choices, we tried to keep most of the values from the study by Nascimento et al. [20], without extensive fine-tuning. Specifically, in all experiments over the CHIRPS dataset, we apply dropout with a probability value of 0.8 for the ConvLSTM model, and 0.2 for the remaining models. With the CFSR dataset, dropout was not applied. Both baseline models (i.e., ConvLSTM and STConvS2S) use traditional batch normalization, the ReLU activation function, and the RMSProp optimizer for training, as reported by Nascimento et al., while the proposed extensions use the AdaMod [8] optimizer. The extensions also use either the traditional batch normalization operation together with the Mish activation function (ST-RFD), or the evolved normalization-activation layer explained in Section 3.2, either in the ST-RFD$_{\text{B0}}$ or ST-RFD$_{\text{B03D}}$ configuration. In all scenarios, we set the learning rate to 0.01 and the batch size to 25, using the mean squared error as the loss function. We apply an early stopping procedure to all models, terminating the training when the validation loss function stops decreasing with a patience threshold of 5. Model training for the recurrent feed-backward decoding extension, in all the tests with $\Delta = 5$ (i.e., when predicting the next 5 time-steps), used the outputs from the last time-step $t - 1$ as input for the recurrent unit at the current time-step t (i.e., training did not rely on a teacher-forcing strategy).

Results were measured in terms of the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE) between estimated and observed values, and also in terms of the coefficient of determination R^2. The corresponding formulas are as follows.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}} \tag{10}$$

$$\text{MAE} = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n} \tag{11}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{12}$$

In Equations 10, 11 and 12, $\hat{y}_i$ corresponds to a predicted value, y_i corresponds to a true observed value, $\bar{y}$ is the mean of the observed values, and n is the number of predictions (i.e., we sum across all cells in the raster representations for the regions under analysis). While the MAE gives the same weight to all errors, the RMSE penalizes models with a higher variance, as it gives errors with larger absolute values more weight than errors with smaller absolute values. The coefficient of determination R^2 measures the proportion of total variation in the observations that is explained by the model, assigning a value of one to a model whose predictions exactly match the observed values, a value of zero to a model that always predicts the average value, and a negative value to a model worse than the baseline corresponding to the average.

4.1 Forecasting Future Time-Steps

The RMSE, MAE and R^2 results for each model over the test sets, for both prediction horizons and datasets, are shown in Table 1. The

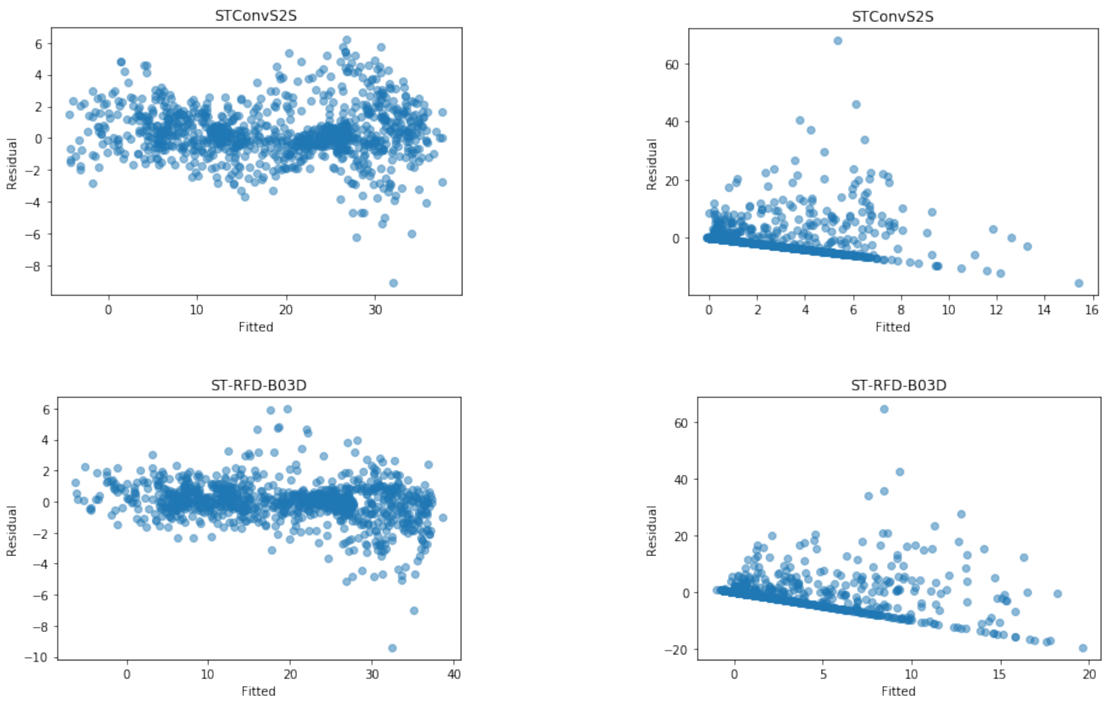

Figure 4: Analysis of fitted versus residual values for the STConvS2S (top) and ST-RFD$_{\text{B03D}}$ (bottom) models, for one time-step ahead forecasts on the CFSR (left) and CHIRPS (right) datasets.

results show that the proposed extensions consistently outperform the baselines. Notably, the recurrent feed-backward decoding extension (ST-RFD) outperforms the ConvLSTM and STConvS2S models in every setting, with the most significant performance gains occurring when considering a prediction horizon of 1. This further attests to the recurrent component's ability to capture temporal features in the short-term, while simultaneously showcasing increased difficulty in maintaining temporal features in longer sequences.

Note that although our experiments only used a simple LSTM layer, the recurrent component can be any sequence-to-sequence recurrent architecture. In particular, for future work, we plan to experiment with other recurrent architectures optimized for maintaining information across long sequences [18, 27, 31]. Although the ST-RFD model has many more parameters in comparison to the highly efficient STConvS2S architecture (i.e., mostly due to the dimensionality of the representations passed to the LSTM layer), training can still be made easily with standard GPU hardware and with reasonably-sized mini-batches of instances. Tests with the STConvS2S and ST-RFD models took similar amounts of time for training and inference, and significantly less than the tests with our implementation for the ConvLSTM architecture.

Besides feed-backward decoding, the novel normalization functions also contribute to even larger performance improvements over the baselines. Changing the normalization operations to either EvoNorm B0 or EvoNorm B03D results in considerable performance gains in the CHIRPS dataset, while still being a competitive option in the CFSR dataset. In almost all scenarios, EvoNorm B03D outperforms the original EvoNorm B0.

Figure 4 presents scatter plots with residuals on the y axis and fitted values (i.e., predictions) on the x axis, for two distinct models (i.e., STConvS2S, ST-RFD) and on the scenario of one time-step ahead forecasts over both datasets. On the CFSR dataset, the residuals roughly form a horizontal band around the value of zero in the y axis, although some outliers are visible and the spread of the residuals is slighly increasing as the fitted values change (i.e., we see some heteroskedasticity problems, slighly worse on the case of the STConvS2S model). On the CHIRPS dataset, the residuals are close to zero when the fitted value is small, and increasingly more negative when the fitted value is large, with some outliers also clearly visible. The spread of the residuals is approximately constant, but the conditional mean is not, showing that the models tend to predict higher values when compared to the true observations. The patterns on the fitted versus residual plots show that there is still significant room for model improvement.

Figure 5 provides an illustration for the results from the best reported model on average (ST-RFD$_{\text{B03D}}$), over both the CHIRPS (top) and CFSR (bottom) datasets. A consistent colour scale was used in all the images from each dataset. The six images shown in each row correspond to the last 3 time-steps used to inform the prediction of the next time-step, together with (a) the ground-truth patches for the next time-step, (b) the predicted patches, and (c) the absolute error between the ground-truth and the predictions (shown in shades of red). The images further attest to the proposed model's

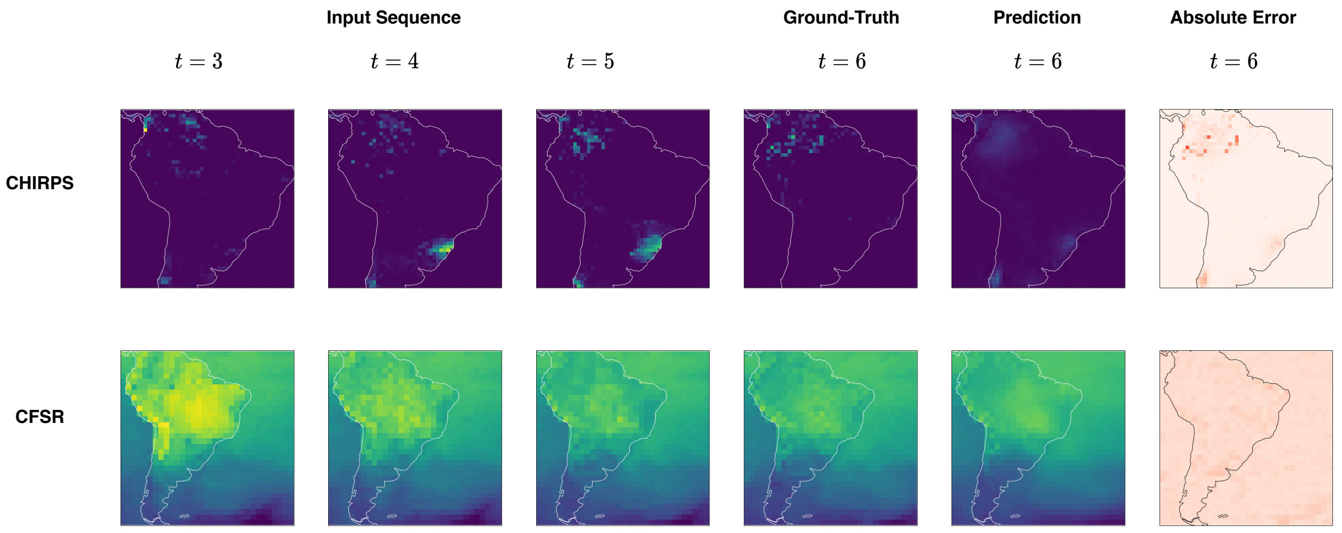

Figure 5: Example results obtained with the ST-RFD$_{\text{B03D}}$ model over the two datasets, considering a prediction horizon of one.

ability of making accurate predictions, although also exhibiting some over-smoothing issues.

It is interesting to notice that result quality when predicting the next 5 time-steps is generally worse than when predicting the immediate next time-step, although not by much – see again Table 1. Particularly on the CHIRPS dataset, and more evidently with the STConvS2S model, the errors concentrate on the cells that are originally associated to more extreme values (and thus the impact of the errors on the averaged metrics is less severe when we consider a larger number of time-steps).

4.2 Missing Data Completion

For this scenario, we created input sequences of 10 time-steps by concatenating the 5 input time-steps and the 5 ground-truth time-steps from the forecasting task detailed in the previous section, followed by the removal of a random time-step from the sequence. The architectures integrating a recurrent component are extended to consider a bi-directional LSTM, allowing for the representations at each time-step to use both future and past information. All the ST-RFD variations, as well as the ConvLSTM baseline, involve bi-directional LSTM units, with the resulting representation at each step being defined as the mean between the forward and backward states. The 1D convolutions in the decoder from the STConvS2S architecture are also allowed to use both future and past information (i.e., causal convolutions were not used in this scenario).

We tested two different loss function configurations, varying how the model is penalized. In a first configuration, denoted as Loss_1, the model is penalized exclusively based on the missing time-step. In the second configuration, denoted as Loss_2, the model is penalized based on the entire output sequence of 10 time-steps, with the loss value associated with the missing time-step being weighed by 0.9 (i.e., the missing time-step loss corresponds to 90% of the total sequence loss), and the remaining 9 time-step predictions being weighed by $\frac{0.1}{9}$ each. This second configuration explores the intuition that reconstructing the entire sequence of co-related time-steps, based on the input representations, can be beneficial for more accurate missing data completion.

The RMSE, MAE and R^2 results for each model over the test sets, for both loss configurations and datasets, are shown in Table 2. Regardless of the loss configuration being used, the reported values are calculated entirely with respect to the missing time-step. As opposed to the forecasting scenario, the ST-RFD model and extensions do not provide significant improvements over the baselines. Instead, the ST-RFD models achieve worse results on average than both baselines in both datasets.

Changing the normalization and activation operations in ST-RFD, to EvoNormB0, further deteriorates the results in every setting, while the proposed EvoNormB03D improves over the base model in the CHIRPS dataset. Interestingly, replacing the ReLU activations and standard batch normalization operations in the STConvS2S architecture, with EvoNormB03D, leads to significant improvements (i.e., this is the best performing model on the CFSR dataset).

The bi-directional ConvLSTM baseline performs exceptionally well in experiments over the CHIRPS dataset, outperforming almost every other model, with results deteriorating when considering experiments on the CFSR dataset. In every setting, the models containing a recurrent component (i.e., ConvLSTM and all ST-RFD variations) improve results when considering the second loss configuration, while the results with the STConvS2S baseline worsen.

We hypothesize that the worse results with ST-RFD are due to the fact that the output at each step, in a bi-directional recurrent component, is calculated based on the entire sequence, whereas in a convolutional component each step is only affected by its closest neighbours, depending on the filter size (i.e., with a filter size of 5, each step is affected by the two closest past and future steps). This property is especially well-suited to this scenario, since the missing values are typically much more co-related with the closest future/past observations, and the co-relations become progressively less relevant the farther an observation is. We believe that the two

	CHIRPS						CFSR					
	Loss$_1$			**Loss$_2$**			**Loss$_1$**			**Loss$_2$**		
	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2
ConvLSTM	5.6809	2.4968	0.3144	**5.6558**	2.4678	**0.3206**	1.0465	0.7716	0.9822	0.9240	0.6814	0.9861
STConvS2S	5.7150	2.4975	0.3044	5.8041	2.5865	0.2779	0.8578	0.6054	0.9879	0.8757	0.6202	0.9874
STConvS2S$_{B03D}$	5.6767	**2.4466**	0.3172	5.6955	2.4573	0.3119	**0.7865**	**0.5354**	**0.9899**	0.7903	0.5410	0.9898
ST-RFD	5.8451	2.6150	0.2679	5.8285	2.6018	0.2658	0.9589	0.6893	0.9852	0.9540	0.6822	0.9853
ST-RFD$_{B0}$	5.9236	2.6611	0.2500	5.8916	2.6195	0.2572	1.0071	0.7321	0.9837	0.9877	0.7164	0.9842
ST-RFD$_{B03D}$	5.8235	2.5798	0.2662	5.7758	2.5728	0.2844	0.9826	0.7079	0.9844	0.9648	0.6966	0.9850

Table 2: Results for the prediction of a missing time-step. The subscripts B0 and B03D respectively correspond to the EvoNormB0 and EvoNormB03D extensions. The bold values indicate the best performing model for each dataset and model.

tasks (i.e., predicting future time-steps, and predicting missing time-steps) differ in the way short-term and long-term context should be used, motivating additional research into how models can be made to better aggregate both types of information.

Finally, Figure 6 provides illustrations for a time-step reconstruction for the ConvLSTM (left), STConvS2S (middle) and ST-RFD$_{B03D}$ (right) models, on both the CHIRPS (top) and CFSR (bottom) datasets. A consistent colour scale was used in all the images from each dataset and model. For each dataset, the eight images shown in each row correspond to the ground-truth missing time-step, followed by the respective model prediction and absolute error between the two (shown in shades of red), for each model. Analyzing the images, we can verify that all models are capable of generating accurate reconstructions, although once again exhibiting some over-smoothing. For both datasets, and particularly on the CHIRPS dataset, the errors on all predictions are concentrated on regions associated with extreme values.

5 CONCLUSIONS AND FUTURE WORK

This article explored the use of encoder-decoder deep learning architectures for spatio-temporal prediction from gridded remote sensing data. We detailed and evaluated several extensions to the previously proposed STConvS2S architecture [20], based on recent techniques in the literature and considering both convolutional and recurrent structures, in an attempt to improve the simultaneous capture of spatial and temporal dependencies in the data. Experiments using datasets from previous studies, consisting of observations of air temperature and rainfall, showed that the proposed extensions significantly outperform a ConvLSTM model and the original STConvS2S architecture, in forecasting scenarios involving the prediction of future time-steps, while still being a competitive option in a missing data completion scenario.

Despite the interesting results, there are also many ideas for future work. For instance, we are interested in exploring the use of deeper convolution blocks, e.g. using recently proposed enhanced pooling methods [7, 14, 30] or using squeeze-and-excitation optimizations [23] to extract more discriminative spatial features. We also plan to experiment with different missing data completion tasks, such as the prediction of missing values associated with partial regions due to cloud coverage [11, 26, 34].

It is interesting to notice that convolution operations are equivariant to translations by design (i.e., translating an input x and convolving with a filter k yields the same result as translating the feature map resulting from a convolution between x and k), but are not equivariant to other transformations. Previous studies have proposed group convolutions as extensions to traditional convolution operations [2, 22], designed to achieve equivariance to other types of affine transformations (e.g., rotation, reflection, or scaling) and compositions of affine transformations (e.g., roto-translation). Similar ideias can also be used as extentions to the proposed model, under the intuition that equivariance to rotations or changes in the scale of the patterns are particularly interesting when processing gridded remote sensing data.

ACKNOWLEDGMENTS

This research was supported through the European Union's Horizon 2020 research and innovation programme under grant agreement No 874850 (MOOD), as well as through the Fundação para a Ciência e Tecnologia (FCT) under the project grants with references PTDC/CTA-OHR/29360/2017 (RiverCure), DSAIPA/DS/0026/2019 (MATISSE), and PTDC/CCI-CIF/32607/2017 (MIMU), and through the INESC-ID multi-annual funding from the PIDDAC programme (UIDB/50021/2020). We also gratefully acknowledge the support of NVIDIA Corporation, with the donation of two Titan Xp GPUs used in the experiments reported in this article.

REFERENCES

[1] Antoine Alléon, Grégoire Jauvion, Boris Quennehen, and David Lissmyr. 2020. PlumeNet: Large-Scale Air Quality Forecasting Using A Convolutional LSTM Network. arXiv:2006.09204

[2] Taco S. Cohen and Max Welling. 2016. Group Equivariant Convolutional Networks. arXiv:1602.07576

[3] Monidipa Das and Soumya K. Ghosh. 2016. Deep-STEP: A Deep Learning Approach for Spatiotemporal Prediction of Remote Sensing Data. *IEEE Geoscience and Remote Sensing Letters* 13, 12 (2016).

[4] Monidipa Das, Mahardhika Pratama, Andri Ashfahani, and Subhrajit Samanta. 2019. FERNN: A fast and evolving recurrent neural network model for streaming data classification. In *Proceedings of the International Joint Conference on Neural Networks*.

[5] Monidipa Das, Mahardhika Pratama, and Soumya K. Ghosh. 2020. SARDINE: A Self-Adaptive Recurrent Deep Incremental Network Model for Spatio-Temporal Prediction of Remote Sensing Data. *ACM Transactions on Spatial Algorithms and Systems* 6, 3 (2020).

[6] Li Deng and Dong Yu. 2014. Deep Learning: Methods and applications. *Foundations and Trends in Signal Processing* 7, 3–4 (2014).

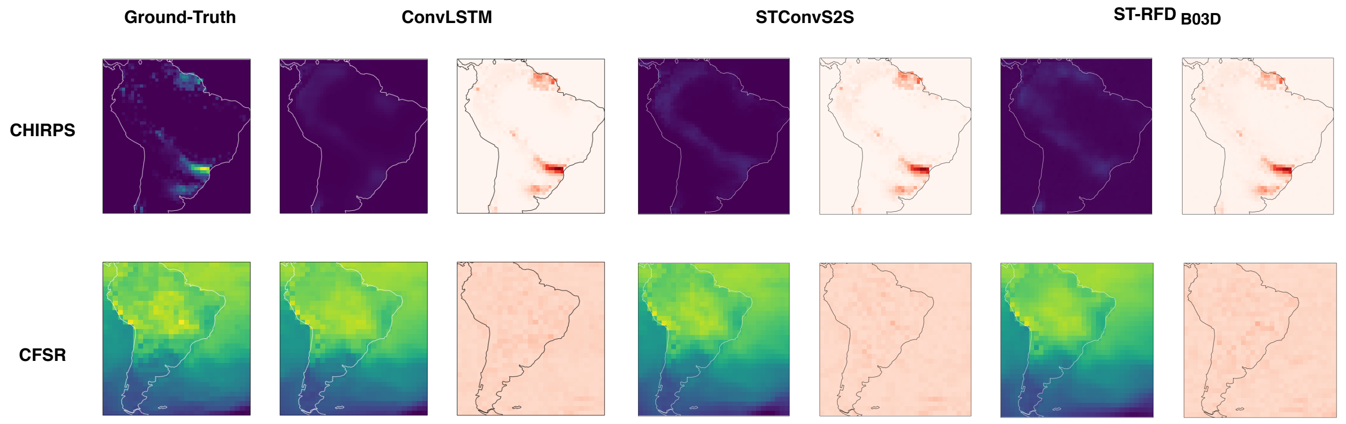

Figure 6: Illustration of the results obtained with the ConvLSTM, STConvS2S, and ST-RFD$_{B03D}$ models over the two datasets, considering the second loss configuration.

[7] Xueqing Deng, Yi Zhu, Yuxin Tian, and Shawn Newsam. 2019. Generalizing Deep Models for Overhead Image Segmentation Through Getis-Ord Gi* Pooling. arXiv:1912.10667

[8] Jianbang Ding, Xuancheng Ren, Ruixuan Luo, and Xu Sun. 2019. An Adaptive and Momental Bound Method for Stochastic Learning. arXiv:1910.12249

[9] Isvani Frías-Blanco, José del Campo-Ávila, Gonzalo Ramos-Jimenez, Rafael Morales-Bueno, Agustín Ortiz-Díaz, and Yailé Caballero-Mota. 2014. Online and non-parametric drift detection methods based on Hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering* 27, 3 (2014).

[10] Chris Funk, Pete Peterson, Martin Landsfeld, Diego Pedreros, James Verdin, J. Rowland, Bo Romero, Gregory Husak, Joel Michaelsen, and Andrew Verdin. 2014. A Quasi-Global Precipitation Time Series for Drought Monitoring. *U.S. Geological Survey* Data Series 832 (2014).

[11] Florian Gerber, Rogier de Jong, Michael E. Schaepman, Gabriela Schaepman-Strub, and Reinhard Furrer. 2018. Predicting missing values in spatio-temporal remote sensing data. *IEEE Transactions on Geoscience and Remote Sensing* 56, 5 (2018).

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997).

[13] Seungkyun Hong, Seongchan Kim, Minsu Joh, and Sa kwang Song. 2017. PSIque: Next Sequence Prediction of Satellite Images using a Convolutional Sequence-to-Sequence Network. arXiv:1711.10644

[14] Qibin Hou, Li Zhang, Ming-Ming Cheng, and Jiashi Feng. 2020. Strip Pooling: Rethinking Spatial Pooling for Scene Parsing. arXiv:2003.13328

[15] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167

[16] Yunseok Jang, Gunhee Kim, and Yale Song. 2018. Video Prediction with Appearance and Motion Conditions. arXiv:1807.02635

[17] Hanxiao Liu, Andrew Brock, Karen Simonyan, and Quoc V Le. 2020. Evolving Normalization-Activation Layers. arXiv:2004.02967

[18] Fandong Meng, Jinchao Zhang, Yang Liu, and Jie Zhou. 2019. Multi-Zone Unit for Recurrent Neural Networks. arXiv:1911.07184

[19] Diganta Misra. 2019. Mish: A Self Regularized Non-Monotonic Neural Activation Function. arXiv:1908.08681

[20] Rafaela C. Nascimento, Yania M. Souto, Eduardo Ogasawara, Fabio Porto, and Eduardo Bezerra. 2019. STConvS2S: Spatiotemporal Convolutional Sequence to Sequence Network for Weather Forecasting. arXiv:1912.00134

[21] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. WaveNet: A generative model for raw audio. arXiv:1609.03499

[22] David W. Romero, Erik J. Bekkers, Jakub M. Tomczak, and Mark Hoogendoorn. 2020. Attentive Group Equivariant Convolutional Networks. arXiv:2002.03830

[23] Abhijit Guha Roy, Nassir Navab, and Christian Wachinger. 2018. Recalibrating fully convolutional networks with spatial and channel "squeeze & excitation" blocks. *IEEE Transactions on Medical Imaging* 38, 2 (2018).

[24] Suranjana Saha, Shrinivas Moorthi, Xingren Wu, Jiande Wang, Sudhir Nadiga, Patrick Tripp, David Behringer, Yu-Tai Hou, Hui-ya Chuang, Mark Iredell, Michael Ek, Jesse Meng, Rongqian Yang, Malaquías Peña Mendez, Huug van den Dool, Qin Zhang, Wanqiu Wang, Mingyue Chen, and Emily Becker. 2014. The NCEP Climate Forecast System Version 2. *Journal of Climate* 27, 6 (2014).

[25] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Proceedings of the International Conference on Neural Information Processing Systems.*

[26] Negar Siabi, Seyed Hossein Sanaeinejad, and Bijan Ghahraman. 2020. Comprehensive evaluation of a spatio-temporal gap filling algorithm: Using remotely sensed precipitation, LST and ET data. *Journal of Environmental Management* 261 (2020).

[27] Aaron R. Voelker, Ivana Kajić, and Chris Eliasmith. 2019. Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks. In *Proceedings of the International Conference on Neural Information Processing Systems.*

[28] Beinan Wang, John Glossner, Daniel Iancu, and Georgi N. Gaydadjiev. 2019. Feedbackward Decoding for Semantic Segmentation. arXiv:1908.08584

[29] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu. 2017. PredRNN: Recurrent Neural Networks for Predictive Learning using Spatiotemporal LSTMs. In *Proceedings of the International Conference on Neural Information Processing Systems.*

[30] Zhen Wei, Jingyi Zhang, Li Liu, Fan Zhu, Fumin Shen, Yi Zhou, Si Liu, Yao Sun, and Ling Shao. 2019. Building Detail-Sensitive Semantic Segmentation Networks With Polynomial Pooling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.*

[31] Lanqing Xue, Xiaopeng Li, and Nevin L. Zhang. 2019. Not All Attention Is Needed: Gated Attention Network for Sequence Data. arXiv:1912.00349

[32] Junbo Zhang, Yu Zheng, and Dekang Qi. 2016. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. arXiv:1610.00081

[33] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, and Xiuwen Yi. 2016. DNN-Based Prediction Model for Spatio-Temporal Data. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.*

[34] Qiang Zhang, Qiangqiang Yuan, Chao Zeng, Xinghua Li, and Yancong Wei. 2018. Missing data reconstruction in remote sensing image with a unified spatial–temporal–spectral deep convolutional neural network. *IEEE Transactions on Geoscience and Remote Sensing* 56, 8 (2018).

[35] Songgang Zhao, Xingyuan Yuan, Da Xiao, Jianyuan Zhang, and Zhouyuan Li. 2018. AirNet: A machine learning dataset for air quality forecasting.