Spatio-Temporal Forecasting With Remote Sensing Observations Using Recurrent Feed-Backward Decoding

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

Jacinto Estima
Universidade Europeia and INESC-ID, Lisbon Portugal

Bruno Martins
IST and INESC-ID - University of Lisbon, Portugal

Abstract

This article presents a novel deep learning approach for spatio-temporal forecasting with remote sensing observations. We 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 convolutional layers, extending it in several directions. In particular, besides considering optimizations such as the Mish activation function or training based on AdaMod, we propose to integrate squeeze-and-excitation blocks with the convolutional blocks of STConvS2S. We also 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 convolution layers used in 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 architecture significantly outperforms STConvS2S and other baseline models.

2012 ACM Subject Classification Computing methodologies → Neural networks; Computing methodologies → Supervised learning by regression; Applied computing → Forecasting

Keywords and phrases Deep Learning, Spatio-Temporal Forecasting, Remote Sensing

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

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, we focus our attention on forecasting tasks, consisting on the prediction of future values conditioned by past observations. Common examples of applications for these methods include forecasting meteorological variables, such as precipitation, air temperature, or wind speed, among many others. The collected data often contains a mixture of spatial and temporal dependencies, indicating how different spatial locations exhibit related patterns (in particular, spatial neighbors tend to be related) and how the observed variables evolve over time. Due to the stochastic nature of the underlying variables, forecasting models must be capable of capturing complex and non-linear patterns, while simultaneously leveraging the aforementioned dependencies in order to obtain accurate results.

Forecasting models for remote sensing data based on deep neural networks are currently gaining increased popularity. These include approaches such as ConvLSTM [18], 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 has become a basic building block for a variety of deep learning architectures proposed in recent studies dealing with sequence forecasting from spatio-temporal data [21, 10, 24].

More recently, Nascimento et al. [15] studied alternatives to capture both spatial and temporal context 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 kernels. Each convolutional 3D filter is factorized into a 2D and 1D filter, 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, we propose the following extensions:

- Replace the ReLU activation function with the Mish [14] function, previously shown to improve performance in deep learning models for computer vision applications;
- Replace the RMSProp optimizer, used by Nascimento et al. when training STConv2S2 models, with the AdaMod [4] optimizer;
- Incorporate squeeze-and-excitation [8] blocks with the convolutional blocks of the original STConv2S2 architecture, adaptively recalibrating the channel-wise feature responses and strengthening the representational power of the convolutional layers;
- Replace the decoding component from STConv2S2, based on 1D dilated convolutions, with a recurrent block that leverages feed-backward decoding [20] (i.e., an idea recently proposed in the context of semantic segmentation models, in which weights from the convolutions used in encoder layers are used in the reverse direction to form the decoder, reducing the total number of 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. Tests relied on the CHIRPS [5] rainfall dataset and on the CFSR [17] air temperature dataset, also used in the experiments from Nascimento et al. [15]. The results show that the proposed extensions all contribute to improved predictions, leading to new state-of-the-art results on the aforementioned datasets.

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 considered datasets and the obtained results. Finally, Section 5 provides concluding remarks and discusses possible directions to be explored in future work.

2 Related Work

Conventional Recurrent Neural Network (RNN) architectures, e.g., based on Long Sort-Term Memory (LSTM) units [6], are comonly 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. [18] 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 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 neighbors. 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. 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 as 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 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, Wang et al. [21] proposed a 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:

$$
\begin{aligned}
I_t &= \sigma\left(W_{hi} * H_{t-1}^{l} + W_{xi} * X_t + b_i\right) \\
F_t &= \sigma\left(W_{hf} * H_{t-1}^{l} + W_{xf} * X_t + b_f\right) \\
G_t &= \tanh\left(W_{hg} * H_{t-1}^{l} + W_{xg} * X_t + b_g\right) \\
C_t^l &= F_t \odot C_{t-1}^l + I_t \odot G_t \\
I_t^{'} &= \sigma\left(W_{mi} * M_t^{l-1} + W_{xi}^{'} * X_t + b_i^{'}\right) \\
F_t^{'} &= \sigma\left(W_{mf} * M_t^{l-1} + W_{xf}^{'} * X_t + b_f^{'}\right) \\
G_t^{'} &= \tanh\left(W_{mg} * M_t^{l-1} + W_{xg}^{'} * X_t + b_g^{'}\right) \\
M_t^l &= F_t^{'} \odot M_t^{l-1} + I_t^{'} \odot G_t^{'} \\
O_t &= \sigma\left(W_{ho} * H_{t-1}^{l} + W_{xo} * X_t + W_{co} * C_t^l + W_{mo} * M_t^l + b_o\right) \\
H_t^l &= O_t \odot \tanh(W_{1\times1} * [C_t^l, M_t^l])
\end{aligned}
\tag{2}
$$

The final hidden states are defined as the concatenation of each memory state derived from different directions, represented as $[C_t^l, M_t^l]$, 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. [24] proposed a new dataset containing a variety of air pollutant and metereological data from China, together with two baseline architectures for spatio-temporal forecasting of air pollutants conditioned on metereological variables. 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 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

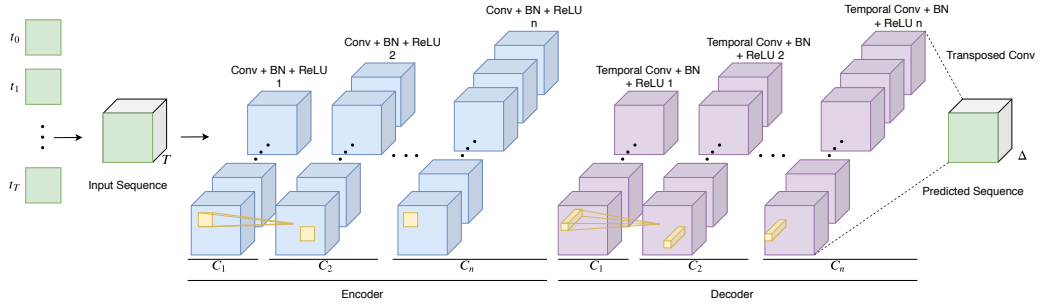

Figure 1 Illustration of the STConvS2S architecture, adapted from [15]. The yellow elements represent the factorized 3D kernel, and C_{l-1} is the number of filters used in the previous layer.

values. The WipeNet architecture is formally defined as follows:

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

Experiments showed that both proposals achieved superior results compared to simpler baselines. In particular, the WipeNet architecture 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.

More recently, Nascimento et al. [15] proposed an encoder-decoder model for spatio-temporal forecasting 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 encoder processes the input sequence by performing convolutions using just the spatial kernel (i.e., $1 \times d \times d$), creating a sequence of meaningful spatial representation. 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 [9] and followed by a ReLU activation function, with the decoder utilizing temporal convolutions to maintain temporal coherency during prediction. 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 [17] air temperature dataset and the CHIRPS [5] 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 STConvS2S architecture, first describing the squeeze-and-excitation blocks that were used to extend 3D convolutional blocks, and then presenting the novel decoding strategy based on combining a LSTM with the idea of feed-backward decoding.

3.1 Squeeze-and-Excitation Blocks

First introduced by Hu et al. [8], squeeze-and-excitation (SE) blocks can enhance the spatial representations created by CNNs by readjusting learned features based on the global information derived from channel relationships. SE blocks involve two crucial steps, namely a squeeze operation and an excitation operation. The squeeze operation produces a channel descriptor $\mathbf{z} \in \mathbb{R}^C$, where C corresponds to the number of channels, by applying an aggregation function (e.g., global average pooling) to each channel's spatial dimensions. Let H and W represent the spatial dimensions (i.e., height and width respectively), and let $u_c \in \mathbb{R}^{H \times W}$ represent the 2D feature map resulting from a convolution with the c^{th} filter. Using global average pooling as the aggregation function, each channel statistic z_c can be calculated as follows:

$$z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j) \tag{5}$$

The descriptor contains informative per-channel statistics, which are afterwards exploited by the excitation operation in order to capture channel-wise dependencies. The excitation operation consists of two fully-connected layers together with the corresponding activation functions, with the first layer projecting the input into a lower dimensionality based on a tunable reduction rate, and the second layer projecting the outcome back to the original channel dimensionality. With σ denoting the logistic sigmoid activation function, these interactions are formally defined as follows:

$$s = \sigma\left(W_2 \cdot \text{ReLU}\left(W_1 \cdot z\right)\right) \tag{6}$$

The resulting collection of per-channel weight scalars $s \in \mathbb{R}^C$ are afterwards used to readjust the original feature maps through a channel-wise multiplication.

We propose to improve the feature extraction of the STConvS2S convolution blocks by inserting the SE blocks after the activation function in both the encoder and decoder. Instead of the ReLU activation function used in the original studies that proposed the STConvS2S architecture and SE blocks, we instead used the Mish activation function [14], 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.

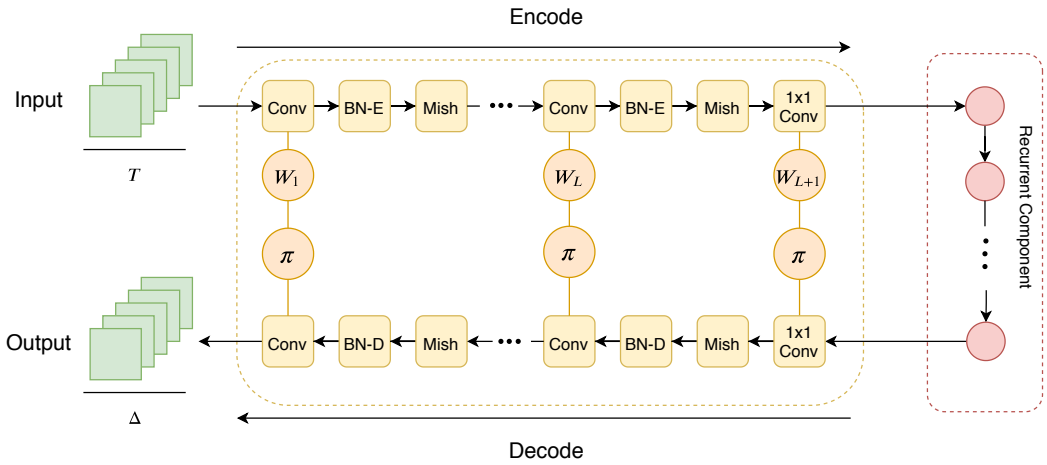

Figure 2 Illustration of the proposed extension leveraging feed-backward decoding, with L layers. BN-E and BN-D denote the batch normalization operations for the encoder and decoder, respectively. The orange circles indicate weight-sharing, with π corresponding to a permutation of dimensions.

3.2 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. [20] 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. If the channel dimension of the input remains with the same size in a convolutional layer L, the same filters of L can be used during decoding.

Leveraring the idea of feed-backward decoding, we propose a variation of the STConvS2S model incorporating the weight-sharing technique between the encoder and the decoder, as illustrated in Figure 2. This alternative allows us to exploit recurrent architectures for capturing temporal features, without incurring in an explosion of learnable parameters. Given that the encoder is now also used for decoding, the proposed adaptation on the STConvS2S network 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 prediction for the next time-step. During decoding, the encoder receives the predicted

	CHIRPS				CFSR			
	$\Delta = 1$		$\Delta = 5$		$\Delta = 1$		$\Delta = 5$	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
ConvLSTM	6.4999	2.4065	6.3874	2.3694	2.6459	1.6836	2.2369	1.5171
STConvS2S	6.3769	2.3443	6.3311	2.3421	1.4172	0.9904	1.5896	1.0491
STConvS2S$_{\text{MA}}$	6.3226	2.3419	6.3172	2.3466	1.3548	0.9327	**1.4674**	1.0397
STConvS2S$_{\text{SE}}$	6.2660	2.3106	**6.2942**	**2.3352**	1.2864	0.8872	1.4727	**0.9945**
STConvS2S$_{\text{RFD}}$	**6.1193**	**2.2336**	6.3761	2.3396	**1.2662**	**0.8312**	1.5584	1.0413

Table 1 Results of the studied models on the test sets. The subscripts MA, SE and RFD correspond to the Mish and Adamod, squeeze-and-excitation and recurrent feedbackward decoding extensions, respectively. The bold values indicate the best performing model for each column.

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 of length differing from the input sequence, so the original transposed convolutional layer is removed. Besides the aforementioned changes, the original encoder structure can be maintained (or instead augmented with squeeze-and-excitation blocks), with the addition of a 1×1 convolutional layer at the end to reduce the channel dimension back to one while still preserving the spatial dimensions.

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. [15], namely the CFSR air temperature dataset and the CHIRPS precipitation dataset. The different models were tasked with 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 time-step and next 5 time-steps, respectively.

The CHIRPS [5] dataset describes $13,960$ sequences for precipitation measurements on a 50×50 grid 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°. In turn, the CFSR dataset [17] 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. [15], we adopted a splitting strategy involving 60% - 20% - 20% of the data for training, validation and testing, in chronological order.

In the tests with the STConvS2S architecture, we used the best performing version reported by Nascimento et al. [15], consisting of 3 layers on both the encoder and decoder,

each containing 32 filters of dimension 5. Similarly, the ConvLSTM baseline consists of 3 layers with 32 hidden states and filters of dimension 5. For every experiment on the CHIRPS dataset, we apply dropout with a probability value of 0.8 for the ConvLSTM model and 0.2 for the remaining models. Both baseline models use the ReLU function and the RMSProp optimizer for training, while the proposed extensions use the Mish function and the AdaMod optimizer for training. In all scenarios, we set the learning rate to 0.01 and the batch size to 25. We apply an early stopping procedure to all models, with a patience threshold of 5. Model training for the recurrent feed-backward decoding extension ($\text{STConvS2S}_{\text{RFD}}$), 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. An initial set of tests showed that this strategy outperformed the also common teacher-forcing approach (i.e., leveraging the ground truth from the prior time-step as input during training for the next prediction). For future work, in order to better deal with exposure biases, we plan to experiment with alternative strategies such as scheduled-sampling [1] or professor-forcing [11].

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

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{8}$$

$$\text{MAE} = \frac{\sum_{i=1}^{n}|\hat{y}_i - y_i|}{n} \tag{9}$$

In Equations 8 and 9, $\hat{y}_i$ corresponds to a predicted value, y_i corresponds to a true value, and n is the number of predictions (i.e., we sum and normalize across all cells). 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 RMSE and the MAE of each model over the test set, for every combination of dataset and prediction horizon, is shown in Table 1.

Our experiments show that the proposed extensions consistently perform better than their baseline counterparts. Notably, the recurrent feed-backward decoding extension ($\text{STConvS2S}_{\text{RFD}}$) performs best when considering a prediction horizon of 1, with the results deteriorating on larger prediction horizons. This could be attributed to the recurrent component's difficulty in maintaining temporal information across longer sequences when compared to the convolutional counterpart. 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 [19, 13, 23].

Besides feed-backward decoding, the other proposed extensions also contribute to improved performance. Changing the ReLU activation fuction to Mish and using the AdaMod optimization strategy ($\text{STConvS2S}_{\text{MA}}$) improves the results over the ConvLSTM and STConvS2S baselines, and introducing squeeze-and-excitation blocks ($\text{STConvS2S}_{\text{SE}}$) further improve the performance. For future work we plan to further enhance this forecasting model with recent ideas from the deep learning literature, such as strip pooling [7], polynomial pooling [22], or enhanced normalization-activation layers [12].

Figure 3 provides an illustration of the results produced by $\text{STConvS2S}_{\text{RFD}}$, over both the CHIRPS (top) and CFSR (bottom) datasets. A consistent colour scale was used in

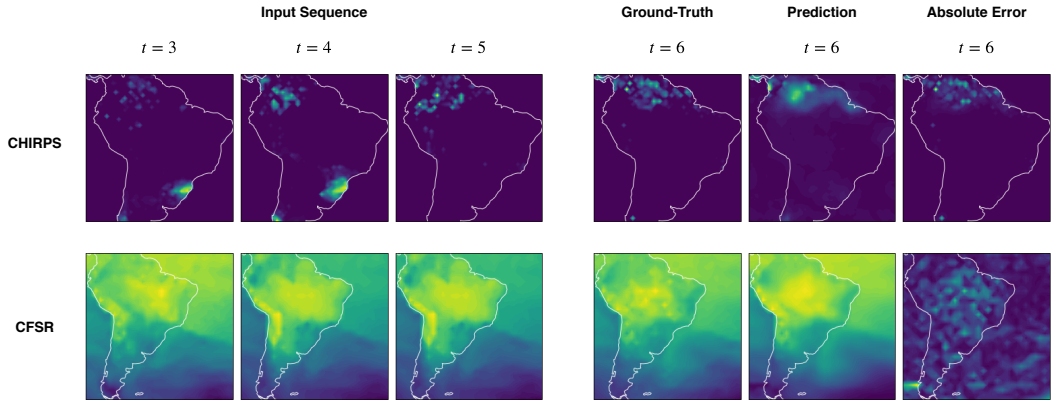

Figure 3 Illustration of the results obtained with the STConvS2S$_{\text{RFD}}$ model over the two datasets, considering a prediction horizon of one.

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. The images further attest to the model's ability of making accurate predictions (although also exhibiting some over-smoothing issues) with errors concentrating on the cells that are originally associated to more extreme values.

5 Conclusions and Future Work

This article explored the use of encoder-decoder deep learning architectures for spatio-temporal forecasting from remote sensing data. We detailed and evaluated several extensions to the previously proposed STConvS2S architecture [15], 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 STConvS2S and other baselines.

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, 3, 22] or optimized normalization-activation operations [12], to extract more discriminative spatial features.

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, 16], 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 STConvS2S, under the intuition that equivariance to rotations or changes in the scale of the patterns are particularly interesting when processing remote sensing data.

Acknowledgements

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 Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2015.

2 Taco S. Cohen and Max Welling. Group equivariant convolutional networks. *arXiv preprint arXiv:1602.07576*, 2016.

3 Xueqing Deng, Yi Zhu, Yuxin Tian, and Shawn Newsam. Generalizing deep models for overhead image segmentation through Getis-Ord Gi* pooling. *arXiv preprint arXiv:1912.10667*, 2019.

4 Jianbang Ding, Xuancheng Ren, Ruixuan Luo, and Xu Sun. An adaptive and momental bound method for stochastic learning. *arXiv preprint arXiv:1910.12249*, 2019.

5 Chris Funk, Pete Peterson, Martin Landsfeld, Diego Pedreros, James Verdin, J. Rowland, Bo Romero, Gregory Husak, Joel Michaelsen, and Andrew Verdin. A quasi-global precipitation time series for drought monitoring. *U.S. Geological Survey*, Data Series 832, 2014.

6 Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8), 1997.

7 Qibin Hou, Li Zhang, Ming-Ming Cheng, and Jiashi Feng. Strip pooling: Rethinking spatial pooling for scene parsing. *arXiv preprint arXiv:2003.13328*, 2020.

8 Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017.

9 Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

10 Yunseok Jang, Gunhee Kim, and Yale Song. Video prediction with appearance and motion conditions. *arXiv preprint arXiv:1807.02635*, 2018.

11 Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2016.

12 Hanxiao Liu, Andrew Brock, Karen Simonyan, and Quoc V Le. Evolving normalization-activation layers. *arXiv preprint arXiv:2004.02967*, 2020.

13 Fandong Meng, Jinchao Zhang, Yang Liu, and Jie Zhou. Multi-zone unit for recurrent neural networks. *arXiv preprint arXiv:1911.07184*, 2019.

14 Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*, 2019.

15 Rafaela C. Nascimento, Yania M. Souto, Eduardo Ogasawara, Fabio Porto, and Eduardo Bezerra. STConvS2S: Spatiotemporal convolutional sequence to sequence network for weather forecasting. *arXiv preprint arXiv:1912.00134*, 2019.

16 David W. Romero, Erik J. Bekkers, Jakub M. Tomczak, and Mark Hoogendoorn. Attentive group equivariant convolutional networks. *arXiv preprint arXiv:2002.03830*, 2020.

17 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. The NCEP climate forecast system version 2. *Journal of Climate*, 27(6), 2014.

18 Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2015.

19 Aaron R. Voelker, Ivana Kajić, and Chris Eliasmith. Legendre memory units: Continuous-time representation in recurrent neural networks. *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2019.

20 Beinan Wang, John Glossner, Daniel Iancu, and Georgi N. Gaydadjiev. Feedbackward decoding for semantic segmentation. *arXiv preprint arXiv:1908.08584*, 2019.

21 Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu. PredRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMs. *Proceedings of the Annual Conference on Neural Information Processing Systems*, 2017.

22 Zhen Wei, Jingyi Zhang, Li Liu, Fan Zhu, Fumin Shen, Yi Zhou, Si Liu, Yao Sun, and Ling Shao. Building detail-sensitive semantic segmentation networks with polynomial pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

23 Lanqing Xue, Xiaopeng Li, and Nevin L. Zhang. Not all attention is needed: Gated attention network for sequence data. *arXiv preprint arXiv:1912.00349*, 2019.

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