Convolutional Long-Short-Term Memory Networks (ConvLSTM) for Weather Prediction using Radar and Satellite Images [⋆]

Nícolas de Araújo Moreira [*] **Rubem Vasconcelos Pacelli** [*] **Yuri Carvalho Barbosa Silva** [*] **Tarcísio Ferreira Maciel** [*] **Ingrid Simões** [*] **João César Moura Mota** [*] **Cerine Hamida** [**] **Rodrigo Zambrana Prado** [***] **Emilie Caillault** [****] **Modeste Kacou** [†] **Marielle Gosset** [†]

[*] *Federal University of Ceara, Fortaleza, Brazil (e-mail: nicolas@ufc.br, rubem070@gmail.com,yuri@gtel.ufc.br,maciel@gtel.ufc.br,ingrid@gtel.ufc.br, mota@gtel.ufc.br).*
[**] *Massachusetts Institute of Technology, Cambridge, United States (e-mail: chamida@mit.edu).*
[***] *The Weather Force, Toulouse, France (e-mail: rodrigo.zambrana@weatherforce.org).*
[****] *Université du Littoral Côte d'Opale, Dunkerque, France (e-mail: ecaillault@gmail.com).*
[†] *Institut de Recherche pour le Développement, Toulouse, France (e-mail: modeste.kacou@ird.fr,marielle.gosset@ird.fr).*

Abstract: Artificial Intelligence techniques, mainly machine and deep learning ones, are becoming the most common approach for data prediction. In this context, using these techniques instead of approaches based on classical statistics has shown interesting and important contributions to weather prediction. The present paper discusses the prediction of rainfall and clouds direction based on a sequence of 10 and 14 frames of radar images with a loss inferior to 0.06. Two different Convolutional Long-Short Term Memory Networks configurations were tested and this work presents the estimated frames resulting from these algorithms and presents comparisons between them, real data, and with the performance of other works. The results show that these algorithms can be suitable for short-term weather forecasting.

Keywords: Artificial intelligence; machine learning; ConvLSTM; image processing; weather forecasting.

1. INTRODUCTION

Climate change is one of the most important concerns of the 21st century and it is a well-known fact that weather modeling and prediction are a mathematical and computational challenge. Meteorological phenomena are complex dynamic systems and involve different fields of physics and mathematics.

According to Shi et al. (2015) and Kumar et al. (2020), nowcasting, i.e. short-term prediction of rainfall intensity in a local region, involves non-uniform modeling of meteorological phenomena over time. More precisely, rainfall is a continuous but intermittent random process usually recorded as a cumulative quantity over constant time intervals as is declared by Ramesh (1998); Poornima and Pusgpalatha (2019) while precipitation fields are complex and irregular according to Rasmussen and Abbasnezhadi (2014). All these features increase the difficulty of a precise nowcasting.

Rainfall has an influence on ecosystems, agriculture, energy production, and even tourism. It is an important resource for human life: long drought periods, monsoons, and floods have a significant impact on the economy and may lead to real disasters. Hence, meteorological prediction can help public managers and monitoring organizations to make decisions to prevent such major impacts as is shown in Poornima and Pusgpalatha (2019). The improvement in predicting the behavior of hurricanes, for example, can impact directly early warning systems in regions affected by this kind of event and the anticipation of the behavior of hurricanes can help local governments and emergency teams in planning mitigation schemes for reducing impacts, avoiding bigger tragedies, and loss of lives.

Until the end of the 20th century, the modeling and prediction of meteorological events were based on classical probability and statistics methods, such as Coxer process and Neyman-Scott process as is shown in Smith and Karr (1985); Ramesh (1998) (usually used to model frontogenesis processes), Gaussian Random Field (GRF, used for representing precipitation in time and space) as is shown in Liu et al. (2019); Rasmussen and Abbasnezhadi (2014),

[⋆] The authors would like to thank *Institut de Recherche pour le Développement* for the financial support.

fractals, as is discussed by Sivakumar (2000); Breslin and Belward (1999); Lovejoy and Mandelbrot (1985); Pathirana (2001), etc. For instance, the authors in Lovejoy and Mandelbrot (1985) state that the projection areas on the Earth of clouds and rain have shapes whose boundaries are fractal curves while the spatiotemporal structure of rain has frequently hyperbolically distributed features (random variables), which also characterizes a fractal. The authors in Gyasu Agyei and Pegram (2014) used GRF for interpolating daily rainfall using simulated radar fields for realistic hydrological modeling of spatial rain fields.

However, the use of artificial intelligence techniques is becoming more and more popular in environmental monitoring and meteorological survey, mainly in weather forecasting and predicting the behavior of meteorological phenomena such as hurricanes and typhoons as discussed in Ruttgers et al. (2019). These methods were shown to be simple, feasible, and present a good level of accuracy. In this context, radar and satellite images play an important role in addition to historical data.

The objective of the present work is to verify the performance of Convolutional Long-Short-Term Memory (ConvLSTM) networks for digital image processing for predicting the short-term path of clouds and rainfall obtained from a sequence of frames of meteorological radar and/or satellite images.

The remainder of this paper is organized as follows. Section II presents a literature review and the state of the art related to weather prediction and other applications of artificial intelligence, machine and deep learning, statistics, probability, and stochastic processes on meteorology, as well as an overview of key concepts related to this topic. Section III presents in detail how Convolutional Networks, LSTM, and ConvLSTM work and the methodology used. Section IV presents and discusses the results. Finally, in Section V, the paper presents the conclusions and some perspectives for future works.

2. LITERATURE REVIEW AND STATE OF THE ART

The authors in Shi et al. (2015) extended Fully Connected LSTM networks (FC-LSTM) to have convolutional structures in input-to-state and state-to-state transitions (Convolutional LSTM - ConvLSTM). The results have shown that ConvLSTM presents better performance than using FC-LSTM, capturing spatiotemporal correlations.

Prediction of rainfall using intensified LSTM is discussed in Poornima and Pusgpalatha (2019), comparing it with Holt–Winters, Extreme Learning Machine (ELM), and Autoregressive Integrated Moving Average (ARIMA). The evaluation was based on the following parameters: Root Mean Square Error (RMSE), accuracy, number of epochs, loss, and learning rate of the networks.

Convolutional LSTM has been used with Wavelet decomposition for solar irradiance forecasting in Wang et al. (2018), presenting a high potential for future practical applications, according to the authors.

In Kumar et al. (2020), the authors propose a precipitation nowcasting architecture called ”Convcast” for short-term predictions of precipitation using satellite images composed by three Convolutional LSTM layers for spatial and temporal feature learning followed by a 3D convolutional layer. The training was done over NASA’s IMERG precipitation data sets, which contain data from passive microwave sensors of satellite precipitation measurements comprising global precipitation. The algorithm can predict the next frame of a 10-frames sequence, nowcasting the precipitation of up to 150 minutes. The accuracy obtained was 0,95 with and RMSE of 0.805 mm/h for 30 min, and overall accuracy of 0.87 with RMSE of 1.389 mm/h for 150 min.

The use of convolutional Recurrent Neural Networks (RNN) with limited training data for frame prediction is explored by authors in Zhang (1999).

The authors in Gamboa-Villafruela et al. (2021) describe a ConvLSTM architecture composed of three layers followed by a 3DConvLSTM with ReLU activation function to predict the 16th precipitation data from a sequence of 15 images and up to a time interval of 180 min. The authors concluded that the increase in the number of layers and the amount of training data enhanced the performance.

Image series prediction using convolutional LSTM applied to precipitation nowcasting is also discussed in Wu (2019).

Some other recent results on deep learning-based approaches for precipitation nowcasting are summarized in Gao et al. (2021), where the authors initially describe mathematically the forecasting problem as a spatiotemporal problem, present a literature review on the state of the art and introduce a systematic benchmark for evaluating the performance of the discussed works. Other approaches related to video frame prediction are discussed in Hong et al. (2017).

3. THEORETICAL INTRODUCTION

In this section, the key concepts of Convolutional, Recurrent, Long-Short Term Memory, and ConvLSTM networks will be introduced.

3.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs), or ConvNets, are feed-forward deep learning algorithms designed to work with bi-dimensional data, particularly images, that present many advantages such as less demanding pre-processing and image reduction without loss of significant features. It is used mainly for image processing due to its ability to learn a large number of filters in parallel. It is usually applied to predictions and image classification.

CNNs are multilayer architecture composed by connected nodes. Each layer is composed of a two-dimensional plan containing many different neurons. The input layer receives the initial data and each neuron extracts the local features using a convolution kernel. The training process does a continuous modification of this convolution kernel weights. The convolution operation extracts high-level features such as edges. The network is composed by one or more pairs of convolutional layers and a max pooling layer. The max pooling enhances the effect of translation invariance and tolerance to deviations, reduces the spatial size (dimensionality reduction) of the convolved feature, reducing the demanded computational power, and suppresses the noise. The max pooling returns the maximum value from the portion of the image covered by the kernel. The addition of a fully-connected layer is a simple way to learn non-linear combinations of the high-level features. The input image is flattened into a column vector and then fed to a feed-forward neural network and backpropagated in every iteration of training. Over a series of epochs, the model can distinguish dominating and low-level features and classify them using softmax classification.

3.2 Long-Short-Term Memory Networks (LSTM)

Recurrent Neural Networks are networks with a feedback loop that allows information to persist. So, from an architectural point of view, it is a kind of densely connected neural network with the output layer on the recurrent neural network connected to itself and with the output of the hidden layer passing through a delay block. We can create a chain of these blocks with multiple copies of these structures. Denoting by $\mathbf{U}$ and $\mathbf{V}$ the matrices connecting inputs and recurrent outputs, and with x_t, h_{t-1} denoting an input and the output of the previous cell, we have:

$$\mathbf{h}_t = \sigma(\mathbf{U}\mathbf{x}_t + \mathbf{V}\mathbf{h}_{t-1}). \tag{1}$$

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network used in deep learning and widely applied tn time-series analysis. The main feature which distinguishes it from other types of neural networks is the introduction of a forget gate that controls which states are remembered or forgotten.

A gate is a structure that adds, removes, or transmits information. They are composed of a layer of a sigmoid neural network (here denoted by σ) and a Hadamard product (denoted by $\circ$). The sigmoid layer outputs a number within the interval $[0, 1]$, where 0 denotes not allow the transmission of any information, and 1 denotes that all information is re-transmitted. Each cell is composed of five parts:

- Cell state (c_t): represents the internal memory of the cell that stores short and long-term memories.
- Hidden state (c_h): decides to retain short and/or long-term information on cell state to make the prediction;
- Input gate (i_t): conditionally decides which input values will be updated on memory state, or, in other words, decides which information to forget and which will be set on the input of cell state;
- Forget gate (f_t): decides how much information of current input and from previous cell state is forwarded to current cell state;
- Output gate (o_t): conditionally decides which will be the output according to input and memory block.

So, the first step of LSTM is to decide which information will be ignored and then which new information will be stored.

The previous cell c_{t-1} is updated into a new cell c_t and then the old state is multiplied by a forgetting factor f_t. A new candidate value is generated, so it is necessary to decide what will be transmitted to output based on the cell state by applying a sigmoid layer. The tanh layer puts the value on the interval $[-1, 1]$. Denoting $\mathbf{U}_g, \mathbf{V}_g$ the weights for input and previous cell output, respectively, and $\mathbf{b}_g$ an input bias, we have:

$$\mathbf{g} = \tanh(\mathbf{b}_g + x_t\mathbf{U}_g + \mathbf{h}_{t-1}\mathbf{V}_g). \tag{2}$$

The input gate is basically a hidden layer of sigmoid activation nodes with weighted inputs $\mathbf{x}_t, \mathbf{h}_{t-1}$. So, the expression for the input gate is given by:

$$\mathbf{i} = \sigma(\mathbf{b}_i + \mathbf{x}_t\mathbf{U}_i + \mathbf{h}_{t-1}\mathbf{V}_i). \tag{3}$$

Denoting by s_t the internal state of a LSTM cell, the forget gate is a set of sigmoid activation nodes multiplied by s_{t-1} to determine what must be remembered (output of forget gate close to 1) and what must be forgotten (output of forget gate close to 0). So, LSTM can learn according to the context and the forget gate expression is given by:

$$\mathbf{f} = \sigma(\mathbf{b}_f + \mathbf{x}_t\mathbf{U}_f + \mathbf{h}_{t-1}\mathbf{V}_f). \tag{4}$$

Then, the output of the forget gate acts as a weight for the internal states. So, the output of this state s_t is expressed by:

$$\mathbf{s}_t = \mathbf{s}_{t-1} \circ \mathbf{f} + \mathbf{g} \circ \mathbf{i}. \tag{5}$$

The output gate is the final stage of the LSTM cell. The output gate has two components: a sigmoid layer and a hyperbolic tangent (tanh) layer (which creates a vector with new candidates to be added to a $\tilde{c}_t$ state). The output is expressed by:

$$\mathbf{o} = \sigma(\mathbf{b}_o + \mathbf{x}_t\mathbf{U}_o + \mathbf{h}_{t-1}\mathbf{V}_o). \tag{6}$$

So, the final expression for cell output is:

$$\mathbf{h}_t = \tanh(\mathbf{s}_t) \circ o \tag{7}$$

A piece of information is forgotten only when a new entry takes its place.

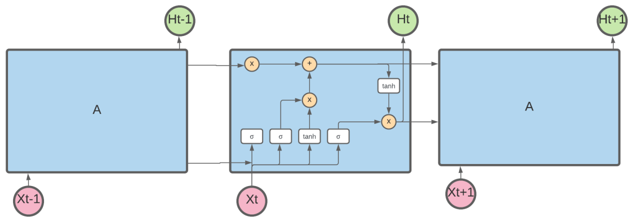

Figure 1. LSTM architecture.

A gentle introduction tn LSTM networks using Python can be found in Brownlee (2017). Further discussions on RNN can be found in Medsker and Jain (2001); Koutník et al. (2014); Josefowic et al. (2015), stochastic RNN in Bayer and Osendorfer (2015), Depth-Gated RNN in Yao et al. (2015), RNN for image generation in Gregor et al. (2015), about LSTM in Hochreiter and Schmidhuber (1997); Greff et al. (2017) and Grid LSTM in Kalchbrenner et al. (2015).

Other works deal with the problem of prediction using other approaches. Predictive Recurrent Neural Networks (PredRNN) are introduced in Wang et al. (2017). PredRNN is based on the idea that prediction strategies should store spatial and temporal data on a unique memory pool. This idea demands a new network architecture called Spatiotemporal LSTM (ST-LSTM) that extracts and memorizes spatiotemporal representations simultaneously. The authors in Kwon and Park (2019) use Generative Adversarial Networks for predicting video frames.

3.3 Convolutional LSTM (ConvLSTM)

Convolutional LSTM is an adequate approach to spatiotemporal prediction: while LSTM is used for a sequence of collected data along time that is modeled as a temporal series, Convolutional Neural Networks are widely used for image processing. Thus, for a sequence of images, a Convolutional LSTM (ConvLSTM) combines the features and advantages of LSTM and CNN, which makes ConvLSTM the best approach for image prediction.

Cell outputs, hidden states, and gates of the ConvLSTM are represented by 3D tensors whose last two dimensions are spatial dimensions (rows and columns). ConvLSTM calculates the future state of a cell by the inputs and past states of its local neighbors using convolution in the state-to-state and input-to-state transitions. The convolution is preceded by a padding operation to ensure the dimension compatibility between states and inputs as is stated in Shi et al. (2015).

The first step in frame prediction is the visualization and analysis of files, extracting rainfall image data. An input example for estimators is shown in Figure 2, containing meteorological radar images obtained from Cearense Foundation of Meteorology and Hydric Resources (*Fundação Cearense de Meteorologia e Recursos Hídricos* - FUNCEME), website accessed on April 23rd, 2021.

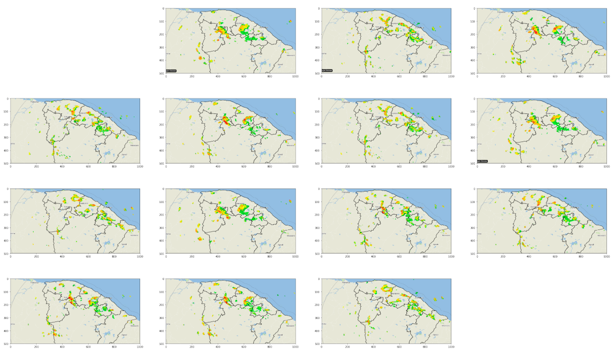
Figure 2. Sequence of frames obtained from meteorological radar (Quixeramobim) images from FUNCEME (accessed on April 23rd, 2021).

This work uses ConvLSTM to predict the next frame of a sequence of satellite / meteorological radar images through two different approaches to data processing and ConvLSTM parameter selection.

4. METHODOLOGY

Two approaches were used and compared, with different settings applied to a dataset composed of 14 colored and 10 grayscale images of rainy days.

4.1 Application of ConvLSTM Models, First Approach

In the present work, two different scenarios were tested, during February 2019 and February 2020 for the Fortaleza region with slightly different Convolutional LSTM architecture configurations for frame prediction given an image sequence. On the first approach, the selected coordinates were 7°00'00.0"S 40°00'00.0"W, 2°00'00.0"S 37°00'00.0"W, and the samples were obtained between February 1st, and 2nd 2019.

Three 2D convolutional LSTM networks using tanh activation function and a convolution window of size 7×7 were used. The number of output filters in the convolution for the first network was 128 and for the remaining was 64. A last 3D convolutional LSTM network using only one output filter and with a convolution window with dimension $1 \times 1 \times 1$ and using ReLU activation function was used, as precipitation level assumes only values equal or superior to zero, so, the negative activations were mapped to zero.

Adam algorithm was used as optimization function. Adam optimization is a stochastic gradient descent technique used for large data and parameter problems based on adaptive estimation of first and second-order moments. It is computationally efficient, demanding low memory. It is invariant to diagonal re-scaling of gradients according to Kingma and Ba (2015).

In neural networks, we aim to minimize error, so, in this scenario, the objective function (in the optimization sense) is referred as a cost function, and the result of this cost function is called loss. Let x denote the error ($y_{predicted} - y_{real}$), the loss function is defined as the logarithm of the hyperbolic cosine of the prediction error:

$$\log(\cosh(x)) = \log\left(\frac{e^{x} + e^{-x}}{2}\right) \tag{8}$$

The algorithm runs during 1000 epochs. The code was developed using Python, Keras and Tensorflow libraries. It initially defines latitude and longitude limits and separates the dataset into training and test datasets and then starts getting the current day as a copy (datetime object), gets the oldest day, and set as current day. A slice (latitude / longitude) data is captured for the current day, getting the precipitation matrix. For each pair of matrices, two periods (k as t and $k+1$ as $t + 30min$) are registered as X and Y. The batch size was set to 2. Then the training period starts and a model is generated and stored. The training history is transformed into a dataframe and saved in a ".csv" format. Then the algorithm advances to the next training day, setting the training dataset to its original state.

4.2 Application of ConvLSTM Models, Second Approach

For the second method, reflectivity data from the X-band radar in Fortaleza, Brazil, obtained from FUNCEME was used to create the dataset.

The selected coordinates were 5°00'00.0"S 40°00'00.0"W, 3°00'00.0"S 37°00'00.0"W, and 184 sample data points separated by a time-step of 7 to 8 minutes were obtained every day for 11 days between February 1st, 2020 and February 11th, 2020.

For pre-processing, we first retrieved radar images of size 360×360 from the raw data file and transformed the intensity values into gray-level pixels. To address computational limitations and the absence of data in the outer edges of the images, we decided to crop and cut the images. For each image, we retrieved the inner square of dimensions 256×256, which we then cut in four squares of equal dimension.

Patches from the same position were used to create sequences of 10 images. Such a sequence is shown in figure 4. Each sequence was then added to the final dataset, which was saved as a N-dimensional array of shape (808,10,128,128,1).

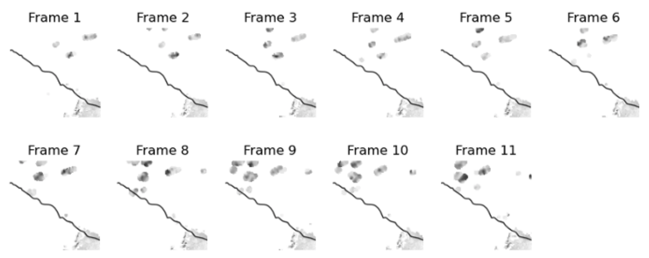

Figure 3. Sequence of frames obtained from meteorological radar (Fortaleza) images from FUNCEME (accessed on June 15, 2021).

The model was constructed using three 2D convolutional LSTM (ConvLSTM2D) layers with batch normalization, followed by a 3D Convolutional (Conv3D) layer for the spatiotemporal outputs. For the three ConvLSTM2D layers, the number of output filters was 64 and the ReLU activation function was used. The kernel sizes were the following: 5×5 for the first layer, 3×3 for the second layer, and 1×1 for the third layer. The Conv3D layer used one output filter, a kernel size of $3 \times 3 \times 3$, and the sigmoid activation function.

Similarly to the first model, the Adam optimization function was used. The loss function used was binary cross entropy. A cross-entropy loss increases the performance (in terms of learning rate and generalization) of a model with sigmoid and softmax output when compared with the mean squared error loss, which presents problems such slow learning rate and saturation.

Data from the first 8 days were used for training and validation. The trained model was then evaluated on testing data for the following 3 days. The algorithm was run for 15 epochs, using batch sizes of 4 instead of 2 to reduce the computational cost.

5. RESULTS AND DISCUSSION

The results for both approaches are shown below, including discussions of qualitative and quantitative aspects and comparisons with the performance of some works presented in Section 2.

5.1 Results for the First ConvLSTM Approach

Figure 4 shows the comparison between real and predicted frames at the same moment for different time instants (30, 60, 90, 120 minutes).

The losses were under 0.02. As it is possible to see, ConvLSTM can predict with good accuracy the dynamic behavior of the frame sequence.

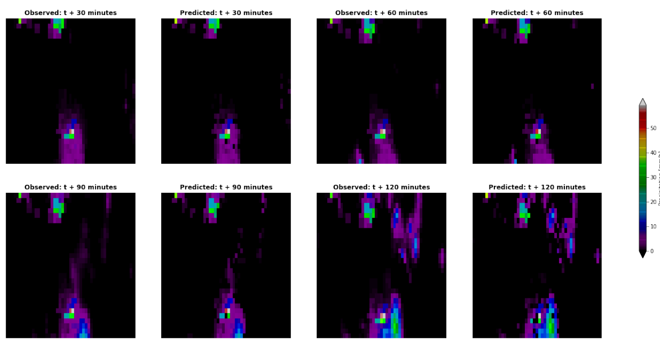

Figure 4. Frame prediction of the first ConvLSTM approach: comparison between real and predicted frames. The observed (left) and predicted (right) frames are compared side-by-side for 30 and 60 minutes ahead (first row) and 90 and 120 minutes ahead (second row).

5.2 Results for the Second ConvLSTM Approach

The trained model was able to predict images up to 30 min ahead based on 50 minutes of previous data. Figure 6 shows the comparison between real and predicted frames at the same moment for different time instants (8, 15, 23, 30 minutes).

The predicted images all had grey backgrounds; for this reason, we decided to apply filters to each image, thus matching the observed images and allowing for more accurate comparisons. Figure 7 illustrates such changes.

The losses were the following: training: 0.0538, validation: 0.0577, testing: 0.0542.

This ConvLSTM model was able to get relatively accurate results with little data and training. We found these results to be transposable, as the model did not need to be trained on the same days used to make the predictions.

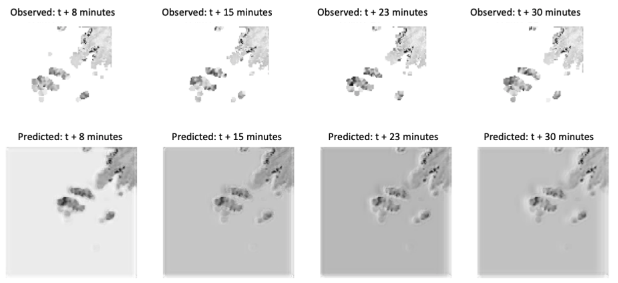

Figure 5. Frame prediction of the second ConvLSTM approach: comparison between real and predicted frames. The observed frames are shown on the first row and the respective predicted ones are shown on the second row for 8, 15, 23 and 30 minutes ahead.

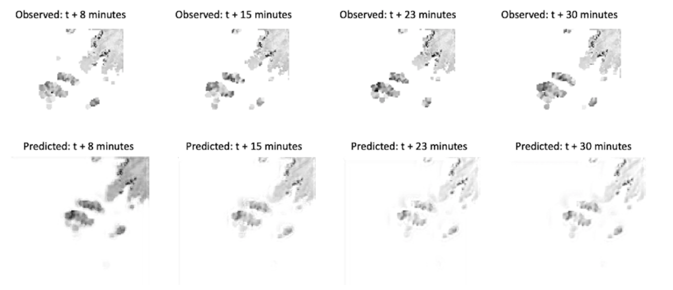

Figure 6. Frame prediction of the second ConvLSTM approach with filters applied: comparison between real and predicted frames.

5.3 Discussion

The tables 1, 2 and 3 summarize the architecture and parameters for the 1st and 2nd approach and ConvCast according to discussed in Kumar et al. (2020). As is possible to see, the 1st architecture has the smallest loss, followed by the 2nd approach and ConvCast. It is possible to compare this performance indicator with the present one shown in Shi et al. (2015), where it is used a Mean Square Error (MSE), resulting in 1.4. However, the 1st approach demands a significantly larger number of epochs (1,000 against 15 of 2nd architecture and ConvCast). Is important to note that the prediction of 2nd approach was based on less than one hour of observations. In addition to a reduced loss, the number of inputs necessary is expressively smaller: from 10 up to 14 against the values presented on the table 3 of ConvCast.

Table 1. Summary of architecture and parameters for the 1st approach

	1st Approach
Structure	3 ConvLSTM layers Activation function: tanh Convolutional window: 7 × 7 28, 64, 64 output filters respectively 1 3DConvLST: Activation function: ReLU Convolutional window: 1 × 1 × 1 1 output filter
Batch size	2
Optimization function	Adam
Loss function	logcosh(x)
Input	14 coloured images
Number of epochs	1,000
Loss	0.02

Table 2. Summary of architecture and parameters for the 2nd approach

	2nd Approach
Structure	3 2DConvLSTM layers Activation function: ReLU Kernel Sizes: 5 × 5, 3 × 3, 1 × 1 64 output filters 1 3DConvLSTM Activation function: sigmoid Kernel size: 3 × 3 × 3 1 output filter
Batch size	4
Optimization function	Adam
Loss function	Cross-entropy
Input	10 grayscale images
Number of epochs	15
Loss	0.05

Table 3. Summary of ConvCast architecture and parameters according to Kumar et al. (2020).

	ConvCast
Structure	Activation function: tanh (ConvLST) ReLU (3DConvLSTM) 4 hidden layers 128 input-to-state filters 64 state-to-state kernel size: 7 × 7
Batch size	2
Optimization function	Adam
Loss function	MSE
Input	1,276 in the training set 319 in the validation set 242 for the test set.
Number of epochs	15
Loss	0,6049

CONCLUSION AND FUTURE WORKS

The ConvLSTM algorithms were shown to provide feasible results with a reduced number of inputs for training and testing, with predicted and actual frames presenting similar profiles. In one of the presented configurations, the number of necessary epochs is equal to the presented in Kumar et al. (2020). The method can be used, for example, for hurricane and typhoon monitoring, allowing early warning on regions under the constant presence of these phenomena.

A possible subject to explore is the use of Adversarially Learned Inference (ALI) and Generative Adversarial Networks (GAN). A second alternative would be the use of Physics Informed Neural Networks, discussed in Raissi et al. (2019). As a next step, a second line of study is the use of a stochastic process based-approach to find a generalized model (probably an α-stable like process) - more precisely, a Probability Density Function (PDF) for climatological phenomena that converges to fractal representation on small scale, Gaussian Random Field on a medium scale and Generalized Neyman-Scott Process and Coxer process to model large scale phenomena such as frontogenesis. This PDF can be used to extract important statistical features that can be used for weather prediction as well.

Another possibility is the use of stochastic resonance to improve the quality of radar/satellite images.

6. ACKNOWLEDGEMENT

The authors would like to thank *Institut de Recherche pour le Développement* for the financial support.

REFERENCES

Bayer, J. and Osendorfer, C. (2015). Lerning stochastic recurrent networks. In *ICLR 2015*.

Breslin, M. and Belward, J. (1999). Fractal dimensions for rainfall time series. *Mathematics and Computers in Simulation*, 48, 437–446.

Brownlee, J. (2017). *Long Short-Term Memory Networks with Python.*

Gamboa-Villafruela, C.J., Fernández-Alvarez, J.C., Márquez-Mijares, M., Pérez-Alarcón, A., and Batista-Leyva, A.J. (2021). Convolutional lstm architecture for precipitation nowcasting using satellite data. *Environmental Sciences Proceedings*, 8(1). doi: 10.3390/ecas2021-10340. URL `https://www.mdpi.com/2673-4931/8/1/33`.

Gao, Z., Shi, X., Wang, H., Dit-Yang, Y., Woo, W.C., and Wong, W.K. (2021). Deep learning and the weather forecasting problem: Precipitation nowcasting. *Deep Learning for the Earth Sciences.*

Greff, K., Srivastava, R.K., Koutnik, H., Steunerbrink, B.R., and Schmidhuber, J. (2017). Ltsm: a search space odissey. *IEEE Transactions on neural networks and learning systems*, 28(10), 2222 – 2232.

Gregor, K., Danihelka, I., Graves, A., Rezende, D.J., and Wierstra, D. (2015). Draw: a recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning.*

Gyasu Agyei, Y. and Pegram, G. (2014). Interpolation of daily rainfall networks using simulated radar fields for realistic hydrological modelling of spatial rain field ensembles. *Journal of Hydrology*, 777–791.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9, 1735–1780.

Hong, S., Kin, S., Joh, M., and Song, S.K. (2017). Psique: next sequence prediction of satellite images using a convolutional sequence-to-sequence network. In *Workshop on Deep Learning for Physical Sciences (DLPS 2017), NIPS 2017.*

Josefowic, R., Zaremba, W., and Sutskever, I. (2015). An empirical exploration of recurrent neural networks. In *Proceedings of the 32nd International Conference on Machine Learning.*

Kalchbrenner, N., Danihelka, I., and Graves, A. (2015). Grid long short-term memory.

Kingma, D.P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* URL `http://arxiv.org/abs/1412.6980`.

Koutník, J., Greff, K., Gomez, F., and Schmidhuber, J. (2014). A clockwork rnn.

Kumar, A., Islam, T., Sekimoto, Y., Mattmann, C., and Wilson, B. (2020). Convcast: An embedded convolutional lstm based architecture for precipitation nowcasting using satellite data. *PLOS ONE*, 15(3), 1–18. doi:10.1371/journal.pone.0230114. URL `https://doi.org/10.1371/journal.pone.0230114`.

Kwon, Y.H. and Park, M.G. (2019). Predicting future frames using retrospective cycle gan. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1811–1820. doi:10.1109/CVPR.2019.00191.

Liu, Y., Li, Jingfa, L., Sun, S.S., and Yu, B. (2019). Advances in gaussian random field generation: a review. *Computational Geosciences*, 777–791.

Lovejoy, S. and Mandelbrot, B. (1985). Fractal properties of rain and a fractal model. *Tellus A: Dynamic Meteorology and Oceanography*, 209–232.

Medsker, L. and Jain, L. (2001). *Recurrent neural networks: design and applications.* CRC Press.

Pathirana, A. (2001). *Fractal modeling of rainfall: downscaling in time and space for hydrological applications.* Ph.D. thesis, University of Tokyo.

Poornima, S. and Pusgpalatha, M. (2019). Prediction of rainfall using intensified lstm based recurrent neural network with weighted linear units. *Atmosphere*, 10(668).

Raissi, M., Perdikaris, P., and Karniadakis, G. (2019). Physics-informed neural networks: a deep learning framewoek for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, (378), 686–707.

Ramesh, N. (1998). Temporal modelling of short-term rainfall using cox process. *Environmentrics*, 9, 629–643.

Rasmussen, P. and Abbasnezhadi, K. (2014). Stochastic simulation of high-resolution daily precipitation using Gaussian Markov Random Fields. In *EGU General Assembly Conference Abstracts*, EGU General Assembly Conference Abstracts, 4698.

Ruttgers, M., Lee, S., Jeon, S., and You, D. (2019). Prediction of a typhoon track using a generative adversarial network and satellite images. *Nature.*

Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.k., and Woo, W.c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, 802–810. MIT Press, Cambridge, MA, USA.

Sivakumar, B. (2000). Fractal analysis of rainfall observed in two different climatic regions. *Hydrological sciences - Journal des sciences hydrologiques*, 45(2), 727–738.

Smith, J.A. and Karr, A.F. (1985). Statistical inference for point process models of rainfall. *Water resources research*, 21(1), 73–79.

Wang, F., Yu, Y., Zhang, Z., Li, J., Zhen, Z., and Li, K. (2018). Wavelet decomposition and convolutional lstm networks based improved deep learning model for solar irradiance forecasting. *Applied Sciences*, 8(1286).

Wang, Y., Long, M., Wang, J., Gao, Z., and Yu, P.S. (2017). Predrnn: recurrent neural networks for predictive learning using spatiotemporal lstms. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*.

Wu, M. (2019). *Sequential images prediction using convolutional LSTM with application in precipitation nowcasting.* Ph.D. thesis, University of Calgary.

Yao, K., Cohn, T., Vylomova, K., Duh, K., and Dyer, C. (2015). Depth-gated recurrent neural networks.

Zhang, Z. (1999). *Image series prediction via convolutional recurrent neural networks with limited training data.* Ph.D. thesis, Wegeningen University.