Application of Convolutional Recurrent Neural Network in Precipitation Forecasting

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

Submitted by:

Namit Abhishek
Indian Institute of Science Education and Research Pune

Under the supervision of:
Dr. Bipin Kumar (IITM, Pune)
Co-Supervisor: Dr. Rajib Chattopadhyay (IITM, Pune)
TAC Member: Dr. Joy Merwin Monteiro (IISER, Pune)

Certificate

This is to certify that this dissertation entitled Application of Convolutional Recurrent Neural Network in Precipitation Forecasting towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Namit Abhishek at Indian Institute of Tropical Meteorology, Pune, under the supervision of Dr. Bipin Kumar during the academic year 2020-2021.

Dr. Bipin Kumar
Scientist E, HPCS
IITM, Pune

Committee:

Supervisor: Dr. Bipin Kumar
Co-Supervisor: Dr. Rajib Chattopadhyay
TAC Member: Dr. Joy Merwin Monteiro

Declaration

I hereby declare that the matter embodied in the report entitled Application of Convolutional Recurrent Neural Network in Precipitation Forecasting are the results of the work carried out by me at the Indian Institute of Science Education and Research, Pune, under the supervision of Dr. Bipin Kumar and the same has not been submitted elsewhere for any other degree.

Namit Abhishek

Acknowledgments

I would like to express my gratitude to Dr. Bipin Kumar and Dr. Rajib Chattopadhyay for their constant guidance and insightful discussions throughout the project. I would also like to thank Dr. Joy Merwin Monteiro for his valuable suggestions and comments in the project. I thank Mr. Manmeet Singh and Mr. Bhupendra Bahadur Singh for their timely inputs and participation. I would like to acknowledge DST, Govt. of India for supporting me with INSPIRE scholarship throughout my BS-MS. Finally, I would like to thank my friends and family for the encouragement and support throughout my studies.

Abstract

In this work, we use convolutional recurrent neural network-based architectures involving ConvLSTM and ConvGRU for precipitation forecasting over the Indian region. We first compare direct and iterative approach for forecasting and find the iterative approach better for our model. We use multiple variables such as specific humidity, orography, soil moisture and surface pressure as input features for better capturing the underlying dynamics. We also analyse the forecasts over different homogeneous rainfall regions. Finally, we compare ConvLSTM and ConvGRU based models. We find similar performance in both the models, ConvGRU being faster.

Contents

List of Figures

Abbreviations

CNN Convolutional Neural Network.

ConvGRU Convolutional Gated Recurrent Unit.

ConvLSTM Convolutional Long Short-Term Memory.

GRU Gated Recurrent Unit.

LSTM Long Short-Term Memory.

NWP Numerical Weather Prediction.

RNN Recurrent Neural Network.

Chapter 1

Introduction

The components of the Earth system interact in complex ways and on many different temporal and spatial scales. The high variability across these scales poses a major challenge in forecasting weather. One of the most difficult to predict meteorological variables is rainfall. Accurate prediction of rainfall is essential for the agricultural sector in increasing crop productivity, for flood warning systems to minimize potential loss of life and property. Rainfall is also an important indicator of drought in a region. Forecasting approaches can be broadly classified into three categories:

1. Numerical Weather Prediction (NWP)
   Typical NWP workflow (figure 1.1) is as follows: Initially, observations from multiple sources such as weather stations and remote sensing products such as satellite and radar are converted to gridded form using suitable interpolation techniques. This process is called data assimilation. In data assimilation, a forecast is combined with newly available observations every few hours to produce a new best estimate of the atmosphere, called analysis. This is further used to give a new forecast. The current state of the weather obtained is given as the initial condition to the NWP model. The physical laws of the atmosphere in the form of differential equations are then solved to some approximation on the discrete grids, conserving certain properties of the system. In addition to the dynamical cores which solve these equations, some parameterizations are also done, which involve processes(usually subgrid) not included in the dynamical core. Further, downscaling can be done using statistical/dynamical

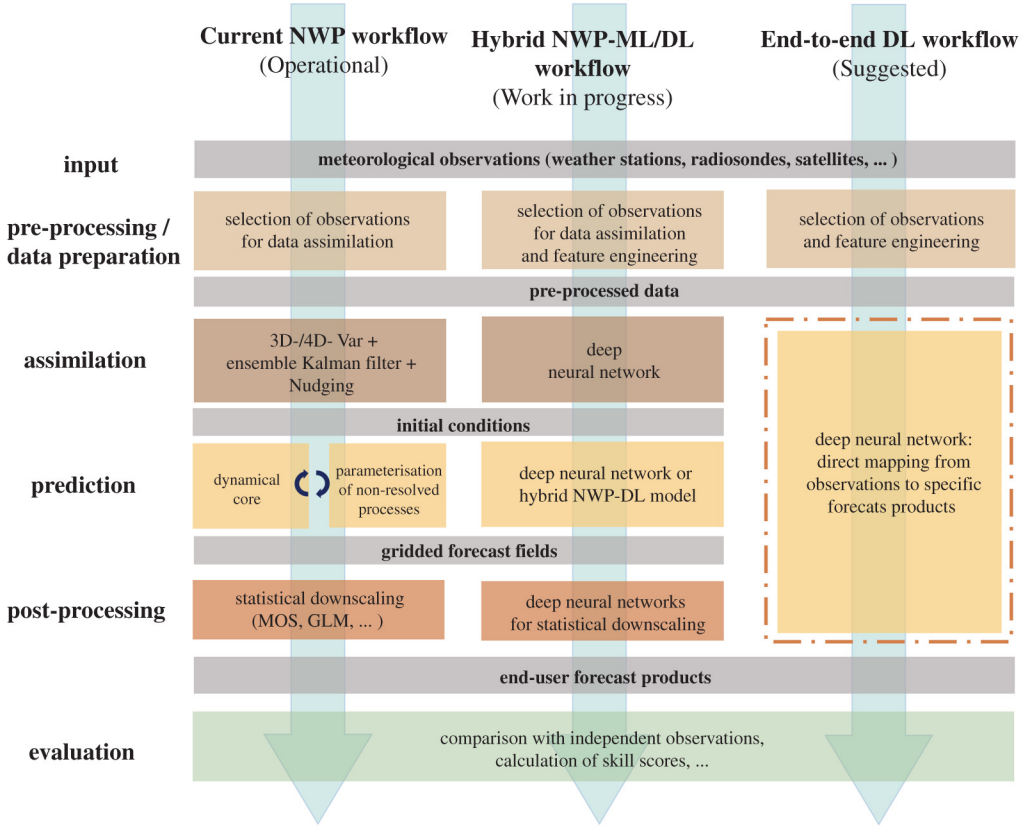

Figure 1.1: Workflow of NWP in comparison to a hybrid NWP-ML/DL and End-to-end DL model (Schultz et al., 2021)

techniques to obtain information at a local level since the prediction is usually made on coarse grids due to limitations in computational power. Currently, weather forecasting is mostly done using NWP.

2. Statistical
   In these models, the current state of the weather is extrapolated using statistical approaches. Some of the statistical methods that have been used are the grey model (GM)(Ju-long,1982 Kayacan et al.,2010) and autoregressive integrated moving average model (ARIMA) (Box et al.,1976, Tektas,2010, Li et al.,2013). However, the precipitation forecasting skill of these models is high only upto few hours of lead time (Zahraei et al., 2012)

3. Machine Learning (ML)
   In recent years, data-driven models have been explored for improving the simulation

and prediction of nonlinear dynamical systems. Classical machine learning algorithms such as kernel methods (Support Vector Machines etc.) and random forests leverage handcrafted domain-specific features to account for spatial and temporal dependencies but often are not able to exploit these spatiotemporal dependencies exhaustively (Reichstein et al., 2019). Deep learning (DL) models have shown substantial advantages in extracting abstract features from large datasets and have shown promise for modelling chaotic dynamical systems (Chattopadhyay et al., 2020). In certain conditions, deep learning models have been shown to perform comparably if not better than NWP((Qiu et al., 2017),(Sønderby et al., 2020),(Frnda et al., 2019)).

(Scher, 2018) used a deep CNN to emulate the dynamics of a simple general circulation model (GCM), the Portable University Model of the Atmosphere(PUMA) and was able to predict it several time steps ahead. The network was trained for each lead time separately.

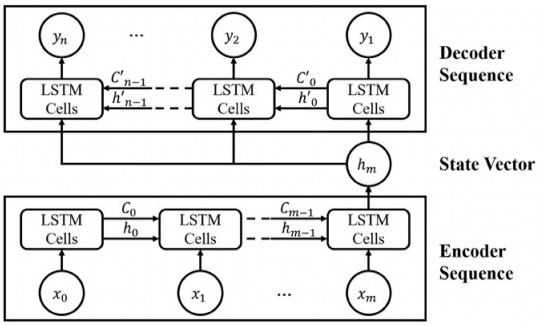

Figure 1.2: One layer of LSTM based Seq2Seq model with m timesteps as input and n timesteps as output (Xiang et al., 2020)

(Sutskever et al., 2014) proposed an LSTM autoencoder model, which consisted of 2 LSTM cells: one for encoding the input sequences into a single vector and one for decoding and obtaining a sequence out of the vector for a language translation problem. In weather forecasting, this approach can be used to produce a sequence of forecasts while also utilizing the properties of LSTM. This model has shown some success, for example, in hourly weather forecast(Zaytar and Amrani, 2016), rainfall-runoff modelling(Xiang et al., 2020) and flood forecasting (Kao et al., 2020).

(Viswanath et al., 2019) used LSTM and Seq2Seq models to identify break and active monsoon spells in India for a lead day of 1 to 5 days. The models were found

to outperform classifiers like KNN(K-nearest Neighbors) and SVM(Support Vector Machine).

(Shi et al., 2015) proposed a novel ConvLSTM network for precipitation nowcasting. The model outperformed Fully connected LSTM and the state of the art ROVER algorithm. (Kim et al., 2017) found ConvLSTM based model to be more accurate than Linear Regression and FC-LSTM for 3 dimensional, 4 channel radar data. (Kim et al., 2019) used ConvLSTM based Network for Trajectory Prediction of Tropical Cyclone.

The work in this thesis is a continuation of the Master's thesis work done by (George, 2020). In (George, 2020), precipitation was predicted from input consisting of precipitation from previous days using an end-to-end ConvLSTM based model. In this thesis, we have used multiple input variables(section 4.2) for forecasting. In addition, a ConvGRU based model was tried(section4.6).
The thesis is organized as follows:
In the next section, we define the problem statement. Chapter 2 consists of the theory related to our work. Chapter 3 discusses the data as well as the methods used. In chapter 4, we present the results consisting of comparison between models(sections 4.1, 4.2, 4.6) as well as the skill(sections 4.3, 4.4, 4.5). Finally, we end with the conclusion and future work in chapter 5.

1.1 Problem Statement

Weather forecasting can be thought of as a spatiotemporal sequence forecasting problem. Suppose we have a spatial region represented by an M $\times$ N grid, and inside each grid cell, P variables can be measured. Therefore the observation at any given time is from the space R^{PXMXN}, where R is the domain of observed variables. Let $X_1, X_2, ..., X_t$ represent a sequence of observations. The spatiotemporal sequence forecasting problem is to predict the most likely length-K sequence in the future given the previous J observations(including the current one): $X_{t+1}, ..., X_{t+K} = argmax\ p(X_{t+1}, .., X_{t+k} \mid X_{tJ+1}, ..., X_t)$

The formulation is the same as that given in (Shi et al., 2015).

Chapter 2

Theory

2.1 Deep Neural Network

Deep learning is a subfield in machine learning that has gained a lot of popularity in recent years due to the advances in hardware and software that have made processing large amounts of data faster. These refer to neural network architectures which have multiple hidden layers. A layer takes one or more tensors as input and produces one or more tensors. Mathematically, a layer operation results in a transformation

$$f : \mathbb{R}^{n_1 \times n_2 \times ..... \times n_k} \rightarrow \mathbb{R}^{m_1 \times m_2 \times ..... \times m_p}$$

Multiple layers are used to learn different features with multiple levels of abstraction. Low-level features are useful in defining higher level features. A deep learning model usually has a lot of parameters and therefore can take a long time to train. As deep learning can easily be parallelized, the use of GPUs has tremendously benefitted in reducing the training time of these algorithms. Following are some of the common terms associated with deep neural networks:

- Weight and bias: These are parameters whose values are learnt with the help of data. These are responsible for linear transformation of the input, after which an activation function can be applied so that the model can learn nonlinearity in the data. At the beginning of training, the weights are given values randomly.

- Hyperparameter: These are parameters in the learning algorithm that need to be specified by the user rather than learnt from data.
- Activation function: These are functions used in layers mostly for non-linear transformation of the inputs. These are also sometimes used to restrict the output in a certain range. Activation function needs to be differentiable for backpropagation to work. Following are the activation functions that are commonly used:
  - sigmoid: $\frac{1}{1+e^{-x}}$
  - tanh: $\frac{2}{1+e^{-2x}} - 1$
  - Relu(Rectified linear unit): $max(0, x)$
- Loss function: This is a function that is used to calculate the difference between the output of the network and the desired output. The model is trained to find values of the weights that minimize the loss function. The cost(error) function, J, which is generally the average of loss function over all the training examples is then calculated.

  There are two steps associated with the computations involved in a neural network:

  1. Forward Propagation: In this step, the output of the neural network is calculated. The computations start from left to right(input layer to the output layer).
  2. The error at the output layer is backpropagated to the hidden layers and consequently, the weights are updated by the optimizer. Here, the computations start from right to left(chain rule).
- Optimizer: After the loss is calculated, weights need to be updated in order to lower the loss and move towards the minima with each iteration. This updation of weights is done by a specific algorithm called the optimizer. These are some of the optimizers:

  1. Gradient descent:
     In Gradient descent algorithm, With each iteration, the weights are updated as follows: $w_j \rightarrow w_j - \alpha \frac{\partial J}{\partial w_j}$
     where w_j is a weight, J is the cost function and α is called the learning rate.
  2. Momentum:
     $m_j \rightarrow \beta m_j + (1 - \beta) \frac{\partial J}{\partial w_j}$
     $m_j \rightarrow m_j - \alpha m_j$

3. RMSProp(Root Mean Square Propagation):
   $v_j \rightarrow \beta v_j + (1-\beta) * (\frac{\partial J}{\partial w_j})^2$
   $w_j \rightarrow w_j - \frac{\alpha}{(v+\epsilon)^{1/2}} * \frac{\partial J}{\partial w_j}$

4. Adam(Adaptive moment estimation):
   $m_j \rightarrow \beta_1 m_j + (1-\beta_1)(\frac{\partial J}{\partial w_j})$
   $v_j \rightarrow \beta_2 v_j + (1-\beta_2)(\frac{\partial J}{\partial w_j})^2$
   $\hat{m_j} \rightarrow \frac{m_j}{1-\beta_1^t}$
   $\hat{v_j} \rightarrow \frac{v_j}{1-\beta_2^t}$
   $w_j \rightarrow w_j - \frac{\alpha}{(\hat{v_j}+\epsilon)^{1/2}} \hat{m_j}$

Gradient descent works by updating the weights in the steepest direction. It is desirable that the algorithm converges faster to the minima. The basic idea of a momentum gradient descent is to compute an exponentially weighted average(controlled by the parameter β) of the gradients and then use that to update the weights instead. The velocity component, m_j dampens the weight update when the gradient changes sign while it accelerates the weight update when the gradient is in the same direction of velocity. Due to this, there is a faster convergence than simple Gradient descent. The most common value of β used is 0.9. RMSprop can also speed up gradient descent. It keeps exponentially weighted average(again controlled by β) of square of the gradient. Here, v_j is the velocity parameter. Different from momentum, the use of square of gradient helps in damping out the directions in which there are large oscillations. This also allows using a larger learning rate, thereby making the algorithm faster. Combining Momentum and RMSprop, we get Adam optimizer. Adam is a commonly used learning algorithm that is proven to be very effective for many different neural networks of a very wide variety of architectures. $\hat{m_j}$ and $\hat{v_j}$ are calculated to account for bias seen in the moving average mostly at the start of the sequence. ϵ is a small constant for numerical stability.

- Metric: Apart from the loss function values, we can also look at other functions to evaluate the model's performance. Some of the commonly used metrics are RMSE, Accuracy, MAE and Correlation Coefficient.

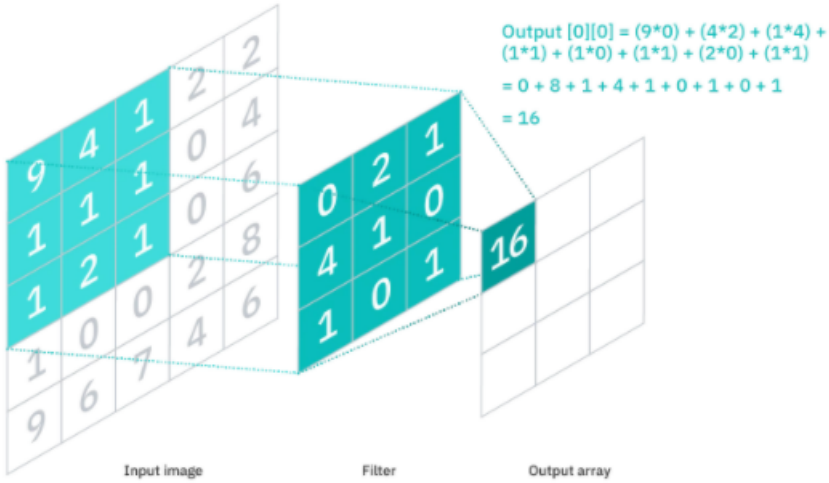

Figure 2.1: Convolution operation between a 5x5 image and a 3x3 kernel, https://www.ibm.com/cloud/learn/convolutional-neural-networks

2.2 Convolutional Neural Network (CNN)

These are neural networks that use convolution operation to detect patterns that span multiple data points in the input. An image can be considered a matrix of values. A convolutional neural network makes use of multiple kernels (also referred to as filters, which are also matrices but with learnable parameters/weights) to extract information from the image. Convolution between an image and a kernel happens by moving the kernel over the image and at each step doing an elementwise multiplication between a part of the image (part which is covered by the kernel) and the kernel. Figure 2.1 shows the convolution between a 5x5 image and a 3x3 kernel to produce a 3x3 output. For a multichannel image, we use a multichannel kernel with the same number of channels as input and corresponding to each kernel we obtain an image as output. This means that the number of kernels we use for an image is same as the number of channels we obtain for the output image. After the convolution operation, bias is usually added to the output and an activation function is applied.

Hyperparameters:

– Kernel size: height and width of the kernel.

– Number of kernels

– Stride: The number of values for the kernel to move in each direction.

– Padding: After convolution, the output image is usually smaller than the input image. Padding refers to the addition of extra values at the boundary of the image to change the size of the output. Two common choices of padding are "Same" and "Valid". "Same" ensures that the size of the output is same as that of the input, while "Valid" does not use any padding. Usually zero padding (values added at the boundary are all zeros) is done assuming no prior knowledge about the outside of the image. Convolution between an nxn image and k x k filter, with padding, p (zeros padded along a dimension at either end) and stride, s results in image of size $\lfloor \frac{n+2p-k}{s} \rfloor \times \lfloor \frac{n+2p-k}{s} \rfloor$, where $\lfloor \rfloor$ denotes the floor function.

Primarily there are two main advantages of convolutional layers:

1. Lesser number of parameters

2. Translational invariance: Response is same the irrespective of the location of the object in the image.

2.3 Recurrent Neural Network (RNN)

These are neural networks that deal with sequential data. The basic operation in a recurrent neural network layer is performed by a cell which takes the output from the previous cell (called hidden state, H_t) as well as the input, which is part of the input sequence (X_t). Consequently, a hidden state has to be initialized as zeros or random values from a distribution for input to the first cell. The same cell operation is performed on every element of the sequence. Figure 2.2 shows a simple representation of an RNN.

A major drawback of the basic RNN is the vanishing/exploding gradient problem, due to which it can fail to capture correlations in the sequence which are far apart. This is because the weights remain constant when going from one time step to the next time step(horizontal arrows in 2.2). Due to this, while backpropagating the error, the gradient of cost function

gets multiplied by the same weight multiple number of times(equal to the number of elements in the sequence). If the weight is less than 1, then the product of these weights tends to go to zero (as we move towards the beginning of the sequence) quickly. Consequently the weight update at the beginning of the sequence slows down exponentially and, therefore, hardly get involved in learning. If the weight is more than 1, then the product can blow up with time. To solve this problem, LSTM cell (Hochreiter and Schmidhuber 1996, 1997) was proposed. The problem does not occur here as there is no repeated multiplication with same weight.

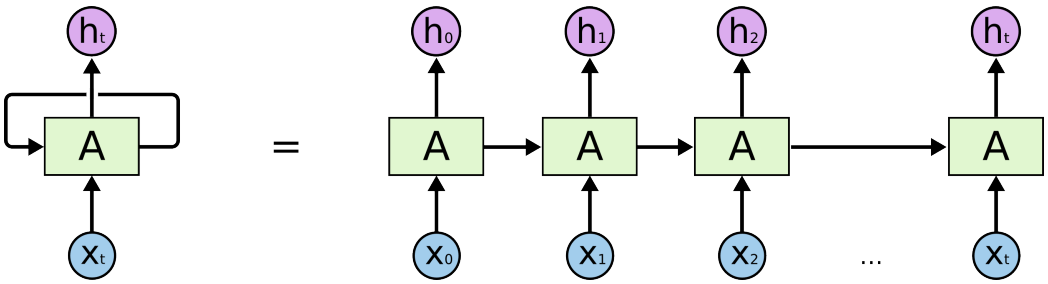

Figure 2.2: Unrolled RNN (Olah, 2015)

2.3.1 Long Short-Term Memory (LSTM) cell

In addition to the hidden state, H_t, an LSTM cell makes use of memory cell, C_t and three gates (input gate, forget gate and output gate)(figure 2.3). The gates use sigmoid activation function to output values between 0 and 1.

$$
\begin{aligned}
i_t &= \sigma(W_{xi}X_t + W_{hi}H_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}X_t + W_{hf}H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\
\tilde{C}_t &= tanh(W_{xc}X_t + W_{hc}H_{t-1} + b_c) \\
C_t &= f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \\
o_t &= \sigma(W_{xo}X_t + W_{ho}H_{t-1} + W_{co} \circ C_t + b_o) \\
H_t &= o_t \circ tanh(C_t)
\end{aligned}
\tag{2.1}
$$

where, X_t is the 1D input(features) to the LSTM cell. Consequently, the input to the LSTM layer is 2D(timesteps, features). $\circ$ is the hadamard product(element wise multiplication), other products involved are matrix multiplications, b_i, b_f, b_c and b_o are the bias terms. Terms involving W are the weight matrices. The forget gate, f_t uses X_t and H_{t-1} to output a number

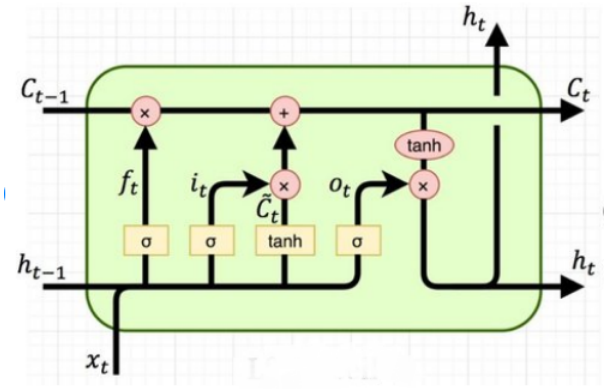

Figure 2.3: LSTM Cell (Varsamopoulos et al., 2018)

between 0 and 1 for each element in C_{t-1}.This is responsible in how much information from previous cell state C_{t-1} is to be kept or discarded. Similarly the input gate is responsible for how much information from current cell is to be kept. The input gate also uses X_t and H_{t-1}. After this, a set of candidate values, $\tilde{C}_t$ are created using X_t and H_{t-1}. tanh activation function is typically used here.

The new cell state, C_t is then obtained using the previous cell state C_{t-1} and the candidate, $\tilde{C}_t$. Finally, the output gate, o_t decides what information from the cell state is to be output.

If the input vector is of length d(i.e. $X_t \in \mathbb{R}^d$) and hidden state, H_t is of length h($H_t \in \mathbb{R}^h$), then:

- $f_t, i_t, o_t, \tilde{C}_t, C_t, b_i, b_f, b_c, b_o \in \mathbb{R}^h$
- $W_{xi}, W_{xf}, W_{xc}, W_{xo} \in \mathbb{R}^{h \times d}$
- $W_{hi}, W_{hf}, W_{hc}, W_{ho} \in \mathbb{R}^{h \times h}$

2.3.2 Convolutional Long Short-Term Memory (ConvLSTM) cell

The ConvLSTM cell is very similar to the LSTM cell but makes use of convolution operation rather than matrix multiplication that we see in LSTM. The use of convolution operation is to be able to extract spatial patterns from some spatiotemporal data. Following are the

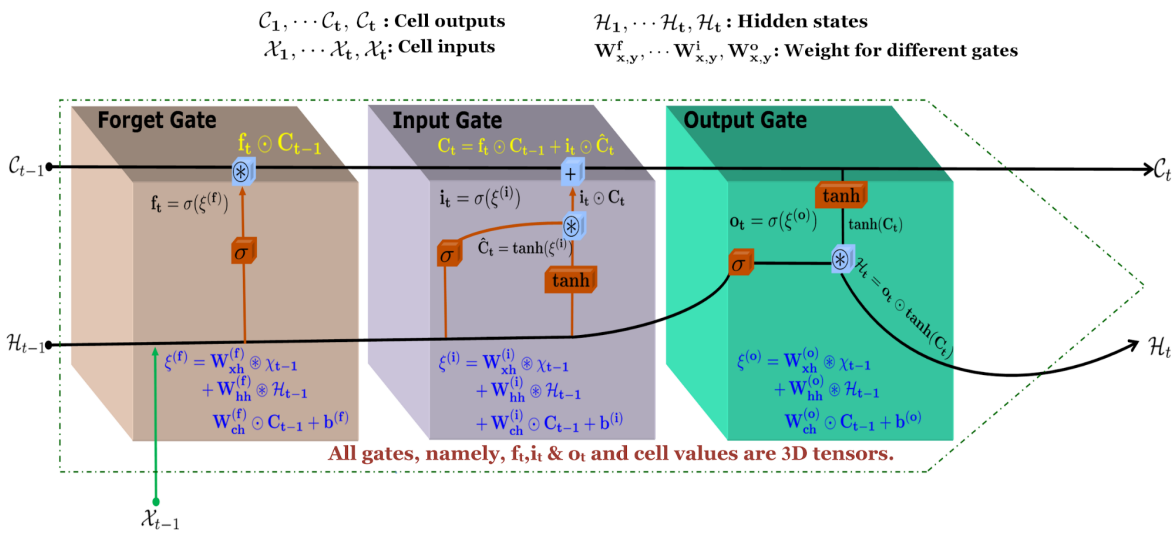

Figure 2.4: ConvLSTM cell

equations involved in a ConvLSTM cell as given in (Shi et al., 2015):

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\
\tilde{C}_t &= tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \\
C_t &= f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \\
o_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o) \\
H_t &= o_t \circ tanh(C_t)
\end{aligned}
\tag{2.2}
$$

* denotes convolution operation while o is hadamard product. Input to the ConvLSTM cell (X_t) is 3D(channels, height, width) instead of 1D in LSTM. Consequently, the Input to the ConvLSTM layer is 4D(timesteps, channels, height, width). In implementation, we can provide multiple inputs at once, so input can be 5D(samples, timesteps, channels, height, width). Figure 2.4 is a representation of the ConvLSTM cell equations given in Eq. 2.2.

In Eq. 2.2, memory cell C_t is used in obtaining the input, forget and output gates. This is known as peephole connection, first introduced by (Gers and Schmidhuber, 2000)). Although, the current implementation in tensorflow(Abadi et al., 2015) does not use peephole

connection and is by the following equations:

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + b_f) \\
\tilde{C}_t &= \hat{\sigma}(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \\
C_t &= f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \\
o_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + b_o) \\
H_t &= o_t \circ \hat{\sigma}
\end{aligned}
\tag{2.3}
$$

2.3.3 Convolutional Gated Recurrent Unit (ConvGRU) cell

Gated Recurrent Unit (GRU) was introduced by (Cho et al., 2014) as a simplified version of LSTM. Consequently ConvGRU(Ballas et al., 2016) was introduced as an extension to GRU, similar to ConvLSTM. ConvGRU has only two gates(reset gate, R_t and update gate, Z_t) instead of three (One can think of this as forget and input gates of LSTM are combined to get an update gate), hidden state is same as memory cell and compared to ConvLSTM, fewer parameters are involved. For input at a given time in the sequence, the reset gate controls how much information from the previous state(H_{t-1}) to clear. Using the current input, the candidate($\tilde{H}$) is calculated. The update gate Z_t controls how much of the new information is to be taken and old information to be discarded. The equations are as follows:

$$
\begin{aligned}
Z_t &= \sigma(W_{xz} * X_t + W_{hz} * H_{t-1} + b_z) \\
R_t &= \sigma(W_{xr} * X_t + W_{hr} * H_{t-1} + b_r) \\
\tilde{H}_t &= \hat{\sigma}(W_{xh} * X_t + R_t \circ (W_{hh} * H_{t-1}) + b_h) \\
H_t &= Z_t \circ H_{t-1} + (1 - Z_t) \circ \tilde{H}_t
\end{aligned}
\tag{2.4}
$$

* denotes convolution operation while o is hadamard product.

Chapter 3

Methods

3.1 Data

3.1.1 IMD

The IMD (Indian Meteorological Department) dataset is a gridded daily dataset over the Indian region obtained by interpolation of data recorded at rain gauge stations across India using Shepard's method(Pai et al., 2014). The rainfall dataset is available in 2 spatial resolutions: 0.25°x 0.25° and 1°x1° for 1901-present. For our study, 0.25° x 0.25° data was used. The region outside India is filled with NAN due to nonavailability of data.

3.1.2 ERA5 Reanalysis Data

ERA5 is the latest version of gridded global Reanalysis[1] dataset produced by ECMWF (European Centre for Medium-Range Weather Forecasts) generated using 4D-Var data assimilation in CY41R2 version of the ECMWF IFS(Hersbach et al., 2020). The data for many atmospheric, land-surface and sea-state parameters is available at 0.25°x 0.25° resolution. The dataset is available for download from the Climate Data Store(CDS) of the ECMWF[2]. In

[1]Reanalysis is the data assimilation of historical observations using a consistent and modern numerical model generally extending for decades

[2]`https://cds.climate.copernicus.eu/`

addition to obtaining observations from multiple sources, it uses an NWP model to generate data where its missing. This results in the data being spatially and physically consistent.

We are interested in the single level and pressure level variables of ERA5. These are available hourly from 1979-present. For our study, the hourly data is resampled by taking mean to obtain daily data. Both the datasets are available in netCDF format. The datasets are taken to cover the Indian region between 6.5° N and 38.5° N, 66.5° E and 100° E. Both the datasets have the format (timesteps, latitude, longitude). Due to the methodology(although different) used in generating both the datasets, there is no missing value with the only exception of IMD rainfall values in ocean areas.

The data was split into training, validation and test.

Training	**Validation**	**Test**
30 years(1979-2008)	**2 years(2009-10)**	**5years(2011-15)**

3.2 Data Preprocessing

Rainfall data is significantly skewed. This makes it difficult for the model to output high values. A suitable transformation of data can help in predicting higher rainfall values. Log transformation helps in reducing skewness but the data remains significantly skewed even after the transformation. The use of log transformation is seen in (Rasp and Thuerey, 2020) for medium range precipitation forecasting. We find that the more the skewness is reduced, the lower is the value of higher rainfall values in the predicted dataset.

When no transformation is applied, assigning a value close to minimum rainfall helps in reducing boundary effects characterized by higher values at the boundary areas due to a sharp change in value at the boundary. Although, it should not be too close in which case, the model will simply predict zero. When exponential transformation was used, boundary effect was always seen whenever the performance of the model was found to be good.

Exponential transformation (e^{kx}), described below, was used in order to output high values. NAN values need to be assigned some value for the ConvLSTM based model to be able to learn. It is not right to replace it with a value corresponding to a possible rainfall

value in the transformed dataset. Following method was implemented:
The rainfall data was initially normalized, after which e^{kx} transformation was applied to non-NAN grid points. NAN was then assigned the value 0. k was chosen to be 7 so that e^{kR} was close to R (George, 2020), where R is the maximum rainfall in the dataset. Precipitation data from IMD is combined with variables from ERA5 reanalysis data to obtain data with dimensions (timesteps, variables, latitude, longitude). Other variables used are:

1. Orography: Orography(Geopotential height) is a measure of elevation of surface of the earth. It does not vary with time. At height h, geopotential is defined as $\Phi(h) = \int_0^h g(\phi, z)dz$, where g is acceleration due to gravity and z is the elevation. Geopotential height is then defined as $\frac{\Phi(h)}{g_o}$, where g_o= 9.806 65 ms^{-2} is the globally averaged value of gravity at sea level.

2. Specific humidity: mass of water vapor per kg of moist air.

3. Surface Pressure:Pressure of atmosphere at the surface of land or sea.

4. Soil Moisture: Water content in soil.
   These variables have been shown to affect precipitation and in some cases used in it's prediction.(Georgakakos and Bras1984,(Larraondo et al., 2020), (Hunt and Turner, 2017), (Pangaluru et al., 2019), (Zantedeschi et al., 2020)).

Neural networks are not scale independent so normalization is done for all variables including transformed rainfall. Normalization also speeds up the optimization of the loss function.

$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$

3.3 Model Architecture

For our model, the spatiotemporal problem is being considered under supervised learning with RMSE as the loss function. To capture the long term spatiotemporal correlations, we are using ConvLSTM layers in our model. The Convolutional operations allow for learning spatial features in the data. In addition, the memory cells can help in retaining long term patterns.

Layer no.	Layer	Activation	Kernel size	Number of kernels	Output Shape
1	ConvLSTM2D_1	tanh	(3,3)	4	(None, None, 4, 129, 135)
2	ConvLSTM2D_2	tanh	(3,3)	8	(None, None, 8, 129, 135)
3	ConvLSTM2D_3	tanh	(3,3)	8	(None, None, 8, 129, 135)
4	ConvLSTM2D_4	tanh	(3,3)	16	(None, None, 16, 129, 135)
5	ConvLSTM2D_5	tanh	(3,3)	16	(None, 16, 129, 135)
6	Conv2D_1	relu	(3,3)	16	(None, 16, 129, 135)
7	Conv2D_2	relu	(3,3)	1	(None, 1, 129, 135)

Table 3.1: Model architecture

By activation function of ConvLSTM, we mean $\hat{\sigma}$ in 2.3. We found that increasing the ConvLSTM layers upto 5 layers helped in increasing the performance, while further addition of a ConvLSTM layer didn't improve the performance significantly. It was also found that 3 x 3 sized kernels worked best and performance declined with increase in kernel size. 'None' implies the dimension can have any size. Only the input of last time step is taken from the last ConvLSTM2D layer as input for the Conv2D layer thereafter as it contains information from the previous timesteps as well as layers. Model Architecture consists of five ConvLSTM2D layers to capture spatiotemporal patterns. This is followed by two Conv2D layers to learn spatial features in the prediction (figure 3.1). Table 3.2 consists of the hyperparameter values used but not given in table 3.1.

Hyperparameter	**Value**
Input Timesteps	5 days
Input feature dimension	5 x no. of input variables x 129 x 135
Batch size	32
Number of epochs	1000-2000 (higher for more input variables)
Learning rate	10^{-4}
Adam optimizer parameters	$\beta_1 = 0.9$, β_2=0.999

Table 3.2: Network Hyperparameters

3.4 Implementation

All of the Implementation was carried out using Keras(Chollet et al., 2015) with Tensorflow(Abadi et al., 2015) as backend. After preprocessing the data, training examples

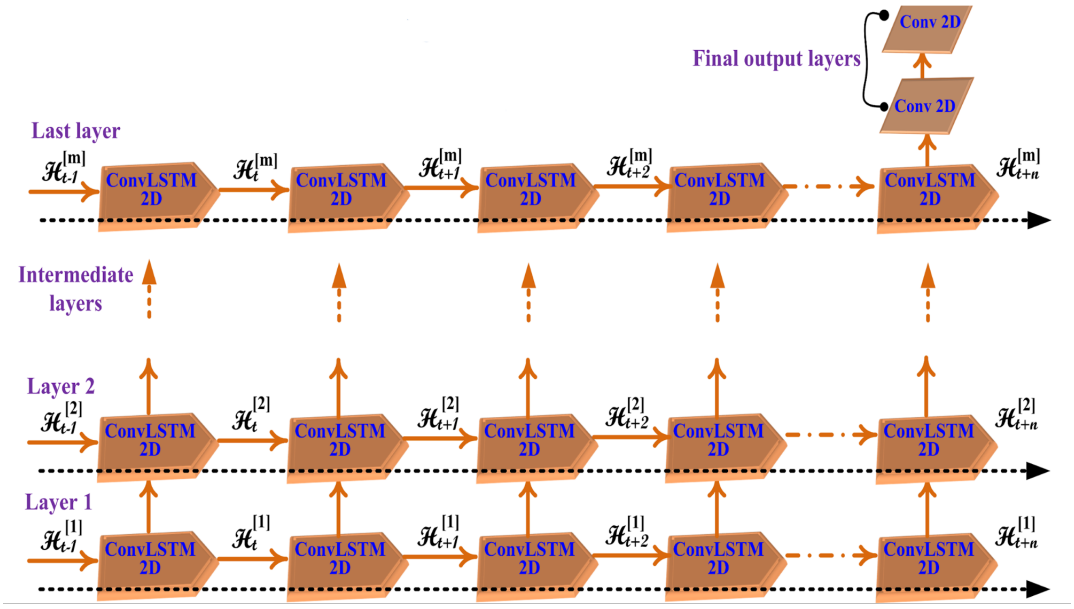

Figure 3.1: Model Architecture used for training

(X,Y) are created from the dataset. ConvLSTM layer requires the input, X to be 5 dimensional(Samples, timesteps, Variables, latitude, longitude). Samples refers to the training examples. We have considered timesteps=5, i.e. we are taking 5 days of data as input and the corresponding output is the prediction of a future day. Therefore for our model, the output Y in the training example is being taken as the groundtruth for the future day. In addition to this, we are currently only interested in rainfall prediction. So Y is 4 dimensional (Samples, 1, Variable=rainfall, latitude, longitude). From hereon, we will refer to predicted days as lead day 1, ...,5 depending on the day which is being predicted, lead day 1 implying 1 day after the last observation. The input variables consist of rainfall from IMD dataset and other variables(sections 3.2,4.2) from ERA5. Due to the data exceeding the RAM on adding extra variables, we use a DataGenerator class[3] which feeds only a small part of the data which is required by the network at a given time and allows us to add multiple input variables without any memory issues.

The evaluation metric is Pearson correlation coefficient (PCC). The correlation for each grid cell is calculated by taking the correlation coefficient of the time series rainfall values of predicted rainfall vs the groundtruth rainfall for the grid cell in the test dataset. In this way we obtain a matrix containing the correlation at each grid cell. The density of correlation

[3] https://stanford.edu/~shervine/blog/keras-how-to-generate-data-on-the-fly

will also be useful while comparing results for different models.

The reason for taking Pearson correlation coefficient as our primary metric is because it captures the strength of linear relationship between groundtruth and the prediction.

Chapter 4

Results and Discussion

Except for section 4.2, all results in this chapter are for model with only rainfall as input. Also, we make use of density plots for comparison of correlation. This is done by taking the time series from the test period of ground-truth and prediction at each grid point and obtaining the correlation for each grid point.

4.1 Iterative vs direct prediction

Two possible ways in which our model can be implemented:

- Direct approach(marked by 'without ld1 weights' in figure 4.2): Separate models were trained for each lead day with input and output consisting of rainfall data from the IMD dataset.
- Iterative approach: Model for forecasting lead day 1 was used to predict other lead days ('with ld1 weights' in figure 4.2). This was done by first predicting the lead day 1 using the previous five days as input. The first day of the input was removed and the predicted day was concatenated to the previous days to predict lead day 2. Similarly, Other lead days were predicted.

  Figure 4.1 shows density vs correlation of the direct approach for lead days 1-5.

On using the iterative approach, there is a slight improvement in the correlation distribution as there is a right shift. Also, the height of the peak has increased, signifying lesser variance in correlation. Although true in our case, it is not always the case that iterative models perform better than models trained for a specific lead day.

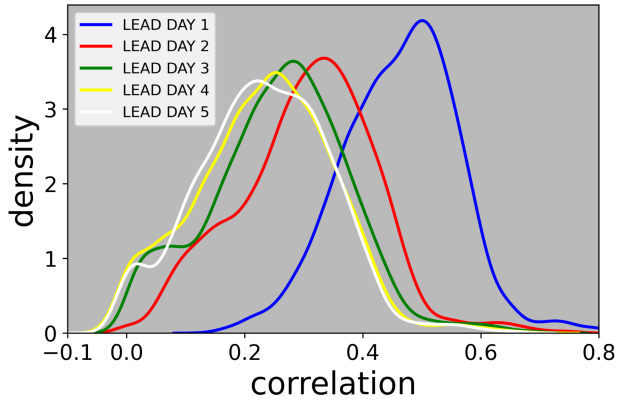

Figure 4.1: density of correlation for lead days 1-5

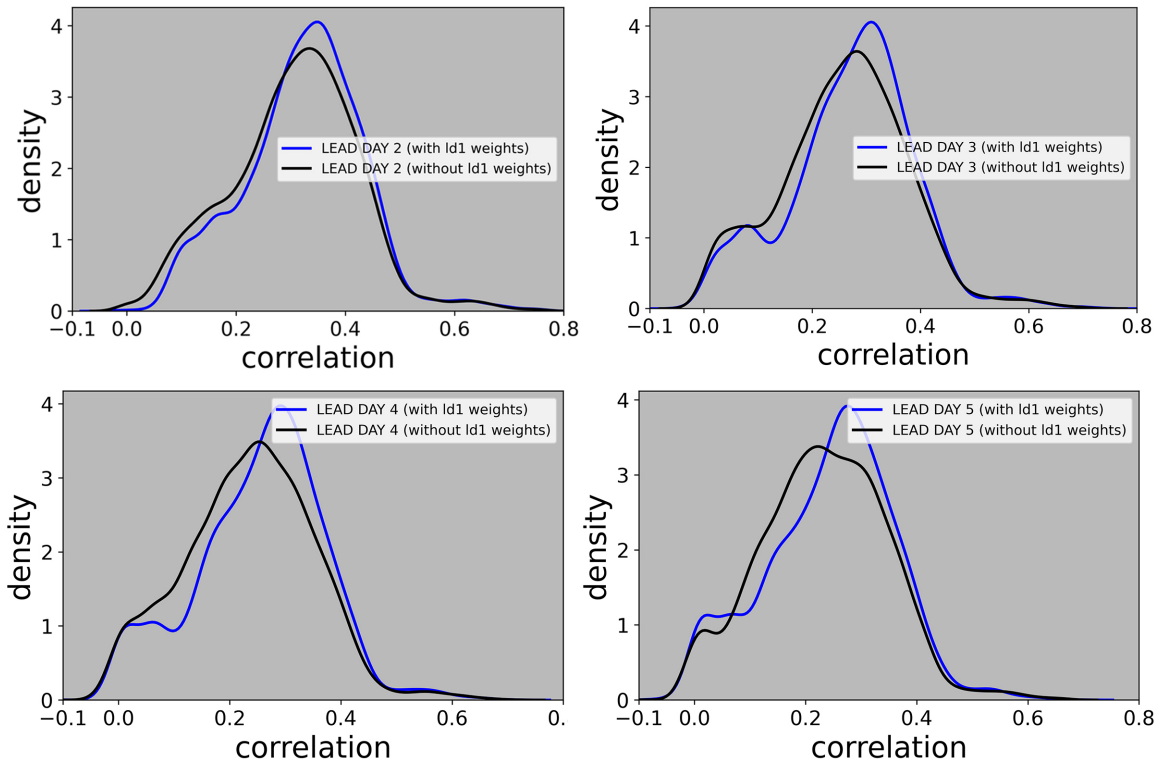

Figure 4.2: Iterative vs direct forecasting

4.2 Addition of Input Variables

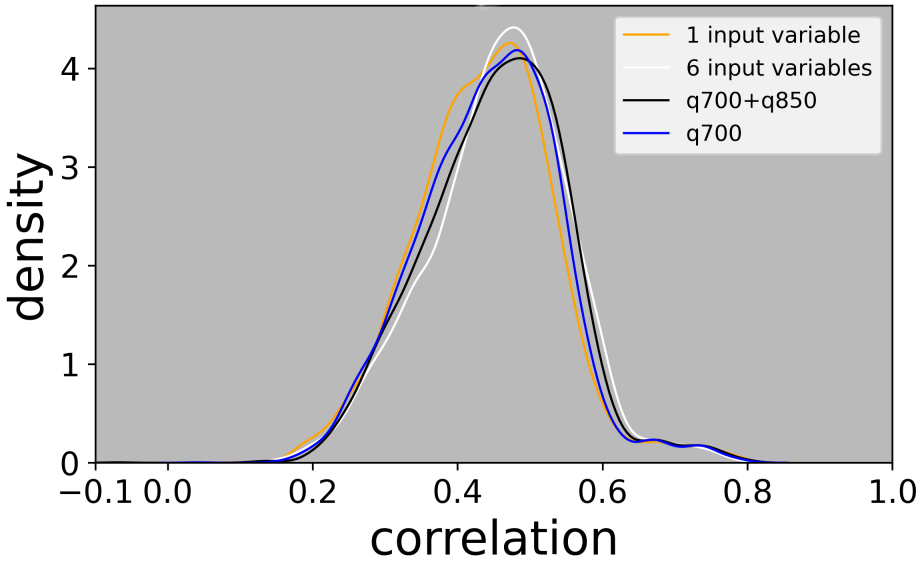

Figure 4.3: density vs correlation plot for various input variables(Lead day 1)

By using multiple input variables, we see an improvement in the prediction. In figure 4.3, by 1 input variable we mean only rainfall was used as input, 6 input variables are rainfall, orography, specific humidity at 700 hPa, specific humidity at 850 hPa, surface pressure and soil moisture. q700 and q850 denote specific humidity at 700 hPa and 850 hPa, respectively. We see that majority of the improvement in the 6 variable input model can be captured by only 2 variables(q700 and q850). However the 6 variables provide much more stability (reduced variance), signified by a higher peak.

4.3 Calibration Refinement Factorization

To assess the quality of the forecasts, the forecasts and observations can be considered to be random variables. The joint distribution of observation and forecast $p(o, f)$ contains information about their marginal distribution as well as their conditional distribution, which captures relationship between the two variables(Murphy and Epstein, 1989). The calibration-

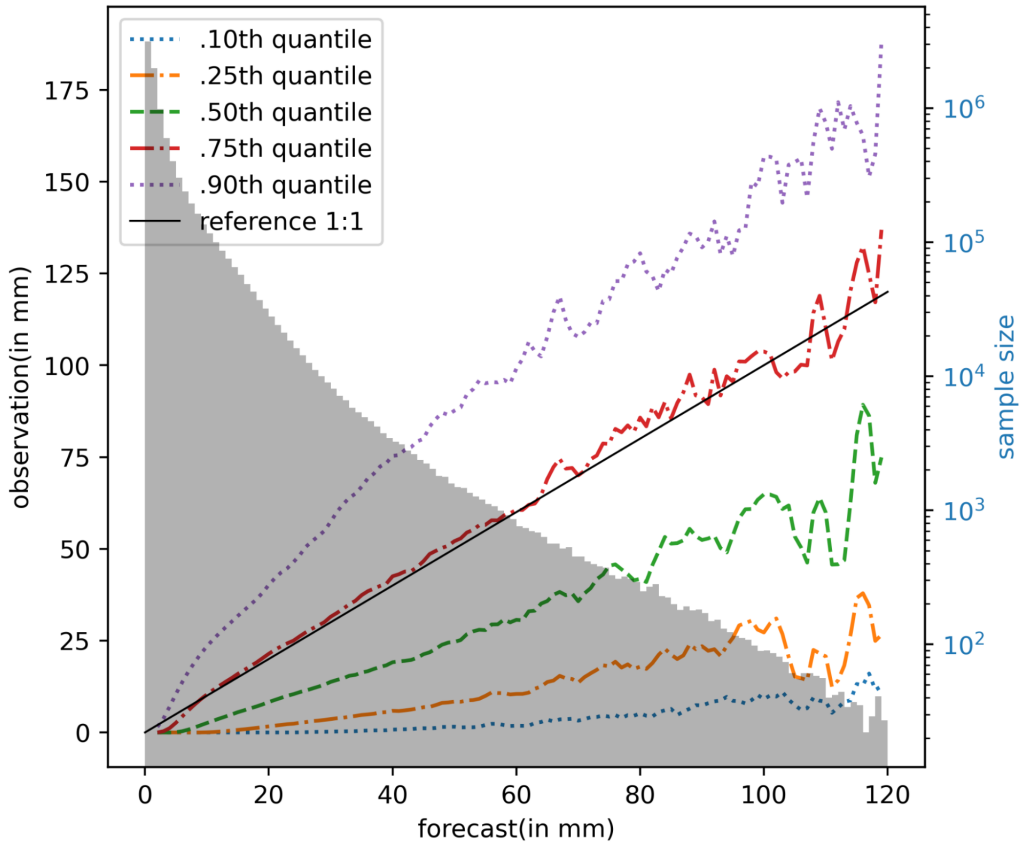

Figure 4.4: Calibration refinement factorization plot (Lead day 1 forecast)

refinement factorization plot (figure 4.4)consists of 2 parts(Murphy and Epstein,1989,Leufen et al.,2020):

1. The marginal distribution of forecast, $p(f)$ represented as a histogram with a bin size of 1mm. This gives a measure of how often different forecasts are given by the model. A model is refined if it does not always give similar values for a forecast.

2. For a given forecast, calibration is a measure of how close the observation is to the forecast. Calibration can be inferred from the conditional distribution $p(o|f)$. Forecasts are perfectly calibrated when $E(o|f) = f$

The conditional distribution $p(o|f)$ represented by quantiles of observation in each 1mm bin of forecast.

The term factorization comes from Bayes theorem: $p(o, f) = p(o|f) * p(f)$. Perfect forecast is represented by the 1:1 line. We see from figure 4.4 that the .50th quantile line remains below the 1:1 line. This means that the model underpredicts the rainfall. Also, the underprediction is higher for larger rainfall values.

4.4 Skill Comparison with Baseline

The skill of the model is compared with climatology(mean of historical observations is taken to be the forecast) and persistence(last observation is taken as the forecast) which are important baselines in weather forecasting. RMSE and correlation have been used to evaluate the skill. Note that climatology is a constant, so correlation cannot be computed for this.

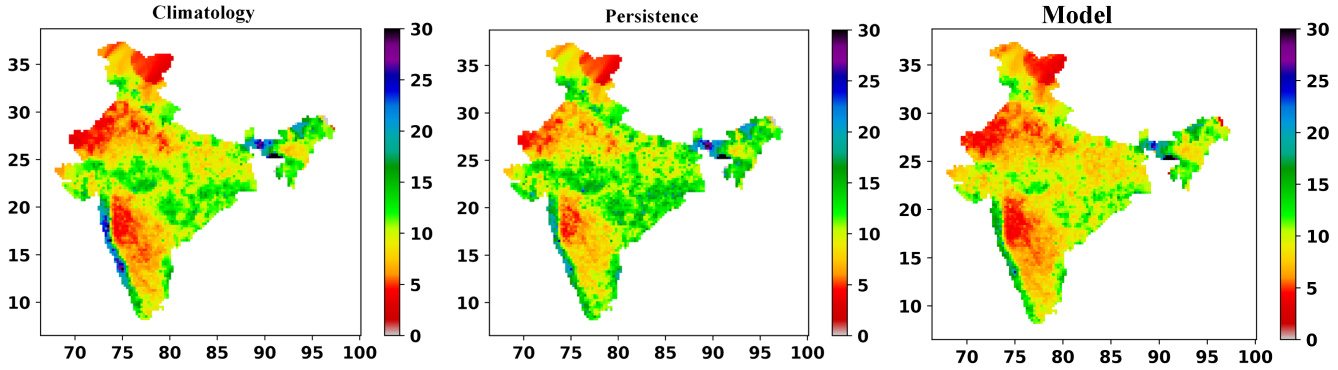

Figure 4.5: RMSE of baselines vs model forecast (Lead day 1)

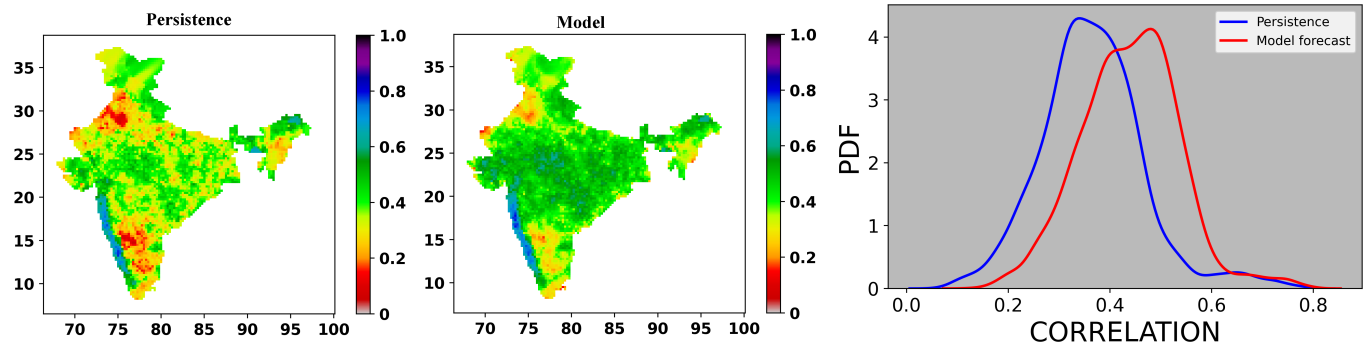

Figure 4.6: Correlation of persistence vs model forecast(Lead day 1)

We define the relative skill score of our model similar to (Leufen et al., 2020). Taking MSE as the metric, we have Skill score = 1 - (MSE of forecast)/(MSE of baseline). A positive skill score can be seen as the percentage of improvement compared to the baseline(Murphy and Daan, 1984). Skill score is calculated for each grid cell for the test period. The skill scores are represented as box and whisker plot(figure 4.7) We see the predictions of the model are better than both climatology and persistence.

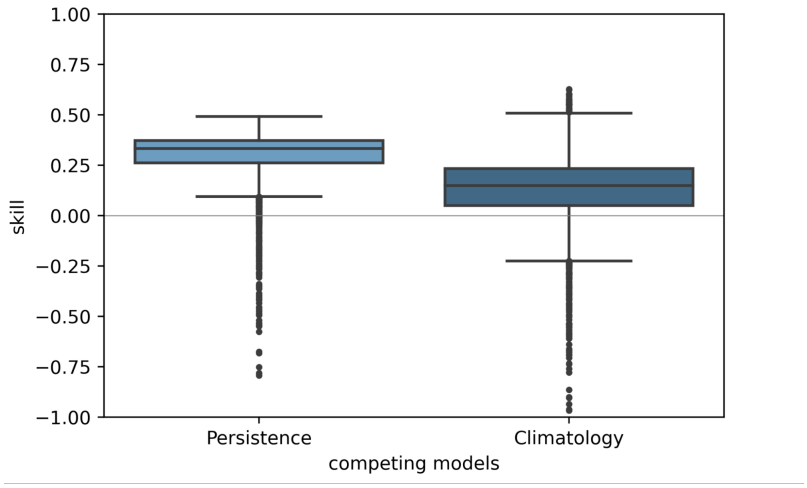

Figure 4.7: Relative skill of the model as compared to Persistence and Climatology

Figure 4.5 shows the RMSE for Climatology, persistence and model forecast at each non NAN grid point for the test period. Figure 4.6 shows the correlation for persistence and model forecast as well as the density of correlation for the test period. We see that in most places, the RMSE of the model forecast is better than both climatology and persistence (figure 4.5). Also, we find the correlation of the model to be better than persistence (figure 4.6).

4.5 Homogeneous Rainfall Regions

There are a total of 6 homogeneous rainfall regions (figure 4.8) categorized based on the amount of rainfall in monsoon seasons defined by the Indian Meteorology Department(IMD) (Kothawale and Rajeevan, 2017). Area-averaged time series of forecast and ground-truth

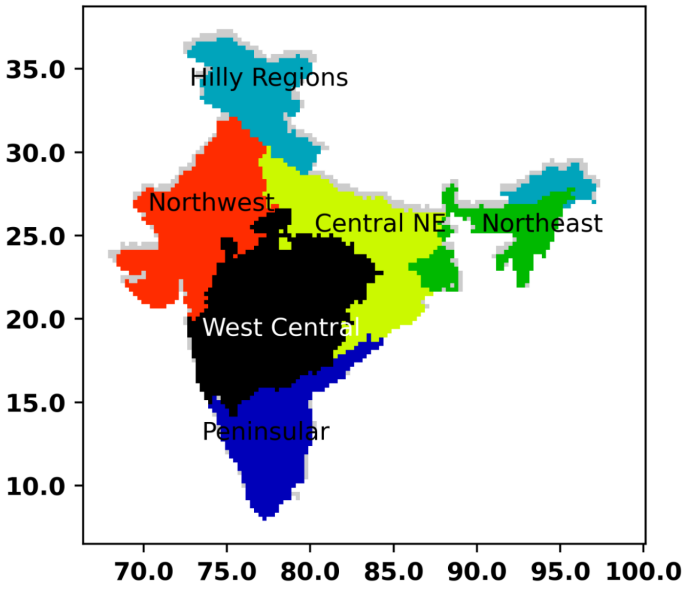

Figure 4.8: Homogeneous Rainfall regions

data was obtained for 5 years(JJAS[1] of 2011-2015) for all the 6 regions using which correlation coefficient and RMSE were calculated for lead days 1 and 2(figure 4.9).

It is seen that on average, the model is able to capture higher peaks. The peaks captured is much lower for 2nd lead day(figure 4.9). It is seen that the correlation is significantly better for West Central, Central NE and Northwest as compared to other regions for both the lead days(figure 4.9). The performance decreases with lead day. The RMSE for lead day 1 is better than lead day 2 for all the regions.

Receiver Operating Characteristics(ROC) curve is one of the methods to evaluate the quality of continuous forecasts. It is a plot of True Positive Rate(TPR) vs False Positive Rate(FPR). TPR=TP/P FPR=FP/N For a given threshold value of rainfall, TP is the number of days when both area-averaged values of ground-truth and prediction are above the threshold, P is the number of days when area-averaged ground-truth is above the threshold. Similarly for FPR, FP is the number of days when area-averaged value of prediction is above the threshold and the corresponding value of ground-truth is below the threshold, N is the

[1]Monsoon period , June to September

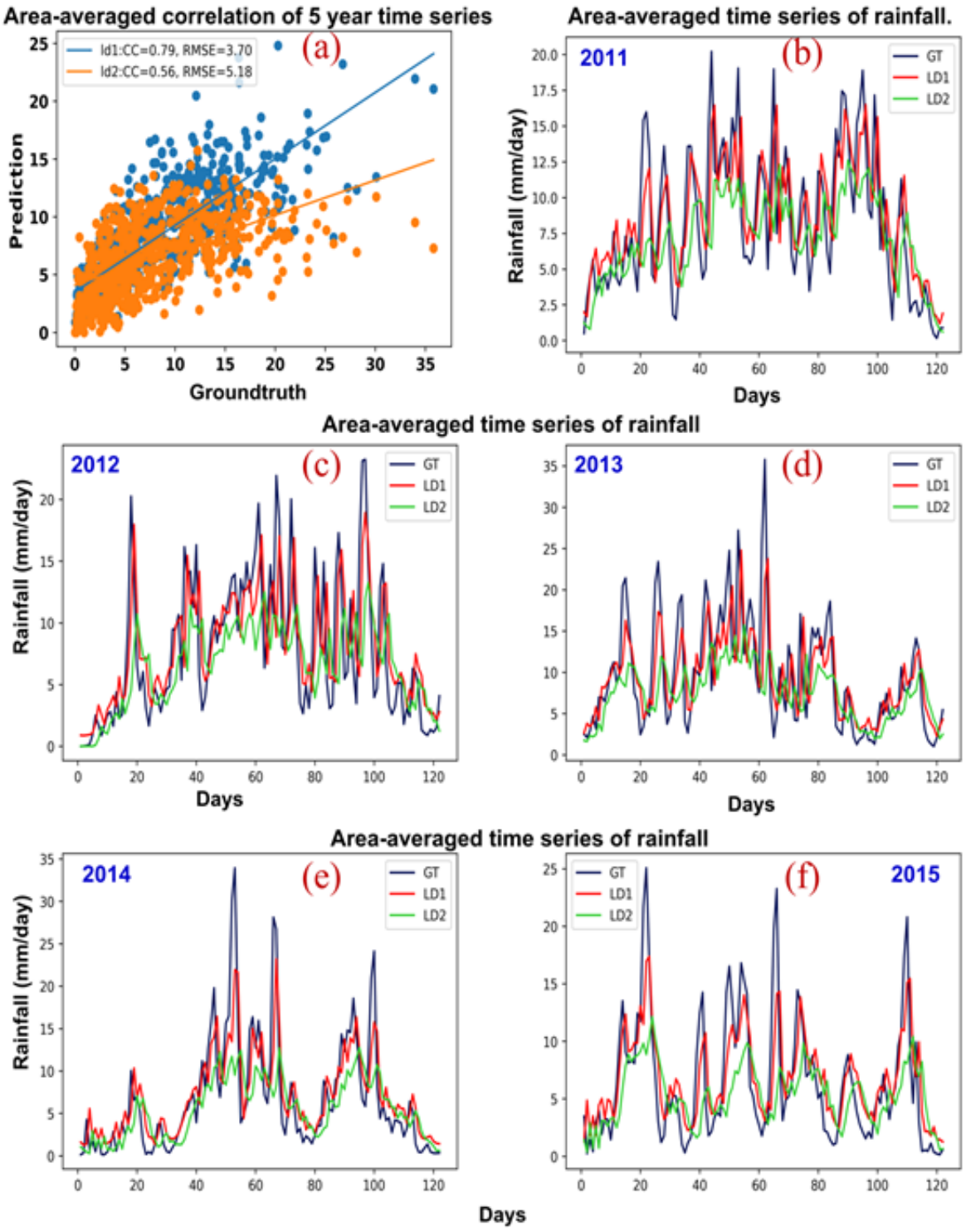

Figure 4.9: Timeseries of area-averaged rainfall of Central North East for 5 years JJAS (2011-15)

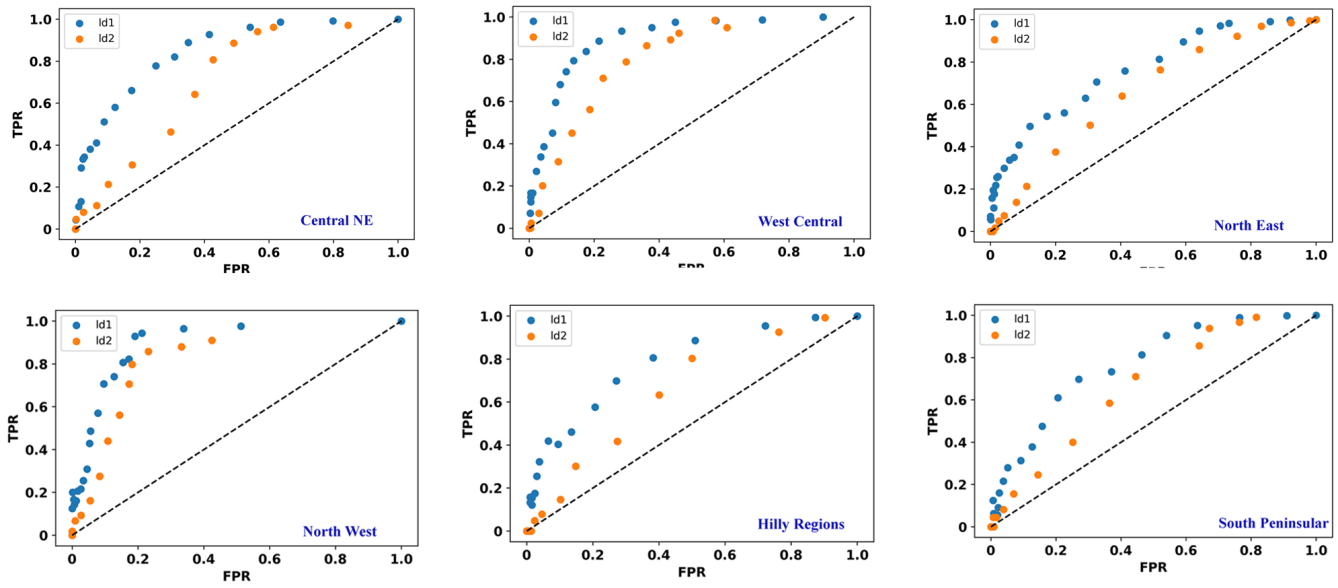

Figure 4.10: ROC curves for Homogeneous regions

number of days when area-averaged groundtruth rainfall is below the threshold. We vary the threshold to obtain the curve. We want the model to predict high rainfall when groundtruth rainfall is high, similarly we want the model to not predict high rainfall when groundtruth rainfall is low. TPR and FPR capture this and so, the ROC curve can help in comparing two models as well as comparing the same model for multiple regions. The curve for which AUC(area under the curve) is higher is better.

For obtaining the ROC curve(figure 4.10), 610 days(JJAS months of 2011-15) of area-averaged rainfall were used.

The Area under the curve(AUC) for lead day 1 is higher than lead day 2 for all the regions. AUC for West Central, Central NE and Northwest are higher compared to other regions agreeing with the table 4.1.

4.6 Comparison with ConvGRU

ConvGRU based model was created after replacing ConvLSTM cells in fig M with ConvGRU cells. Hyperparameters were taken to be the same as that of that of the ConvLSTM model.

Region	LD1 CC(RMSE)	LD2 CC(RMSE)
West Central	0.79(3.70)	0.56(5.18)
Central NE	0.7(3.92)	0.42(5.11)
Northwest	0.76(3.77)	0.58(4.64)
Hilly Regions	0.53(4.19)	0.24(4.93)
Northeast	0.55(5.85)	0.3(6.84)
South Peninsular	0.52(4.15)	0.31(4.46)

Table 4.1: Skill for homogeneous rainfall regions

Figure 4.11 shows the density of correlation for the ConvLSTM and ConvGRU based models. We see a slight if not any improvement over the ConvLSTM based model. Another advantage is a significant reduction in training time over the ConvLSTM model (takes about 70% of the time taken by ConvLSTM) due to a lesser number of parameters.

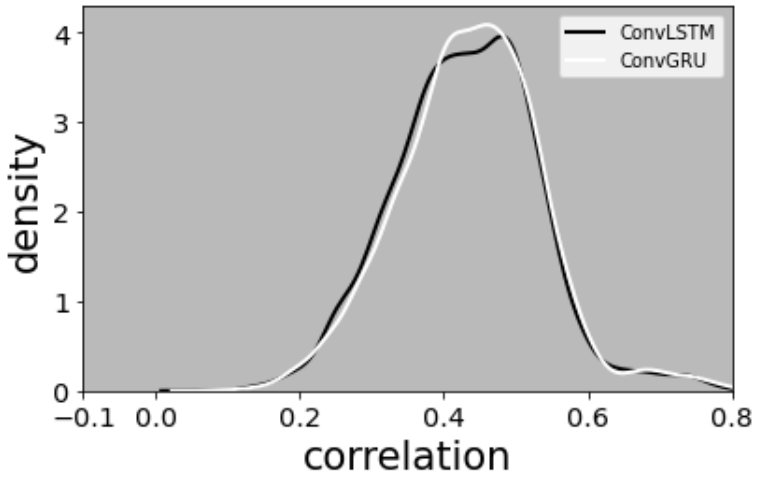

Figure 4.11: density of correlation ConvLSTM vs ConvGRU (Lead day 1)

Chapter 5

Conclusion

We see that iterative prediction is better than direct prediction. Although in (Scher, 2018), which uses only convolutional layers, direct prediction was better. Forecast improved on using multiple input variables. Particularly, specific humidity at 700 hPa and 850 hPa has resulted in significant improvement compared to other variables used. This shows that the model is able to capture the rainfall dynamics better when more variables are used. The model outperforms standard baselines such as persistence and climatology. It was found that the model performs well in West Central, Central NE and Northwest and performs comparatively poorly in the other three regions. We also implemented a ConvGRU based model and saw a similar performance to the ConvLSTM based model with the benefit of reduced computational time.

5.1 Future Work

Other variables can be explored to improve the model. Potentially, specific humidity at other pressure levels can be tried. Hyperparameters like number of layers,number of filters for different layers, learning rate, optimizer, batch size, number of epochs, regularization, dropout can be further tuned to improve the model. Also other loss functions/custom loss functions can be tried. SGCN is a generalization of CNN and views image as a graph. It has been shown to be better than CNN when there are missing values in an image (Danel et al., 2020). SGCN can thus be tried to deal with NAN values in the IMD dataset.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Ballas, N., Yao, L., Pal, C., and Courville, A. C. (2016). Delving deeper into convolutional networks for learning video representations. *CoRR*, abs/1511.06432.

Box, G., Jenkins, G., and Day, H. (1976). *Time Series Analysis: Forecasting and Control*. Holden-Day series in time series analysis and digital processing. Holden-Day.

Chattopadhyay, A., Hassanzadeh, P., and Subramanian, D. (2020). Data-driven predictions of a multiscale lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network. *Nonlinear Processes in Geophysics*, 27:373–389.

Cho, K., Merrienboer, B. V., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *ArXiv*, abs/1409.1259.

Chollet, F. (2017). *Deep Learning with Python*. Manning Publications Co., USA, 1st edition.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Danel, T., Smieja, M., Struski, L., Spurek, P., and Maziarka, L. (2020). Processing of incomplete images by (graph) convolutional neural networks. *ArXiv*, abs/2010.13914.

Frnda, J., Durica, M., Nedoma, J., Zabka, S., Martínek, R., and Kostelansky, M. (2019). A weather forecast model accuracy analysis and ecmwf enhancement proposal by neural network. *Sensors (Basel, Switzerland)*, 19.

Georgakakos, K. and Bras, R. (1984). A hydrologically useful station precipitation model: 1. formulation. *Water Resources Research*, 20:1585–1596.

George, S. (2020). Application of deep learning algorithm for monsoon forecast. *Master's Thesis, Indian Institute of Technology, Madras*.

Gers, F. and Schmidhuber, J. (2000). Recurrent nets that time and count. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, 3:189–194 vol.3.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., Simmons, A., Soci, C., Abdalla, S., Abellan, X., Balsamo, G., Bechtold, P., Biavati, G., Bidlot, J., Bonavita, M., Chiara, G., Dahlgren, P., Dee, D., Diamantakis, M., Dragani, R., Flemming, J., Forbes, R., Fuentes, M., Geer, A., Haimberger, L., Healy, S., Hogan, R., Holm, E., Janisková, M., Keeley, S., Laloyaux, P., Lopez, P., Lupu, C., Radnoti, G., Rosnay, P., Rozum, I., Vamborg, F., Villaume, S., and Thépaut, J. (2020). The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146:1999–2049.

Hochreiter, S. and Schmidhuber, J. (1996). Lstm can solve hard long time lag problems. In *NIPS*.

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

Hunt, K. M. R. and Turner, A. (2017). The effect of soil moisture perturbations on indian monsoon depressions in a numerical weather prediction model. *Journal of Climate*, 30:8811–8823.

Ju-long, D. (1982). Control problems of grey systems. *Systems & Control Letters*, 1:288–294.

Kao, I.-F., Zhou, Y., Chang, L., and Chang, F. (2020). Exploring a long short-term memory based encoder-decoder framework for multi-step-ahead flood forecasting. *Journal of Hydrology*, 583:124631.

Kayacan, E., Ulutas, B., and Kaynak, O. (2010). Grey system theory-based models in time series prediction. *Expert Syst. Appl.*, 37:1784–1789.

Kim, S., Hong, S., Joh, M., and Song, S.-K. (2017). Deeprain: Convlstm network for precipitation prediction using multichannel radar data. *ArXiv*, abs/1711.02316.

Kim, S., Kim, H., Lee, J., Yoon, S., Kahou, S., Kashinath, K., and Prabhat (2019). Deep-hurricane-tracker: Tracking and forecasting extreme climate events. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1761–1769.

Kothawale, D. and Rajeevan, M. (2017). Monthly, seasonal and annual rainfall time series for all-india, homogeneous regions and meteorological subdivisions: 1871-2016. *IITM research report no. RR-138*.

Larraondo, P. R., Renzullo, L., Dijk, A., Inza, I., and Lozano, J. A. (2020). Optimization of deep learning precipitation models using categorical binary metrics. *Journal of Advances in Modeling Earth Systems*, 12.

Leufen, L. H., Kleinert, F., and Schultz, M. (2020). Mlair (v1.0) – a tool to enable fast and flexible machine learning on air data time series.

Li, L., Ma, Z., Liu, L., and Fan, Y. (2013). Hadoop-based arima algorithm and its application in weather forecast. *International journal of database theory and application*, 6:119–132.

Murphy, A. and Daan, H. (1984). Impacts of feedback and experience on the quality of subjective probability forecasts. comparison of results from the first and second years of the zierikzee experiment. *Monthly Weather Review*, 112:413–423.

Murphy, A. and Epstein, E. (1989). Skill scores and correlation coefficients in model verification. *Monthly Weather Review*, 117:572–582.

Olah, C. (2015). Understanding lstm networks.

Pai, D., Sridhar, L., Rajeevan, M., Sreejith, O., Satbhai, N. S., and Mukhopadhyay, B. (2014). Development of a new high spatial resolution ($0.25° \times 0.25°$) long period (1901-2010) daily gridded rainfall data set over india and its comparison with existing data sets over the region.

Pangaluru, K., Velicogna, I., Geruo, A., Mohajerani, Y., Ciracì, E., Charakola, S., Basha, G., and Rao, S. V. B. (2019). Soil moisture variability in india: Relationship of land surface-atmosphere fields using maximum covariance analysis. *Remote. Sens.*, 11:335.

Qiu, M., Zhao, P., Zhang, K., Huang, J., Shi, X., Wang, X., and Chu, W. (2017). A short-term rainfall prediction model using multi-task convolutional neural networks. *2017 IEEE International Conference on Data Mining (ICDM)*, pages 395–404.

Rasp, S. and Thuerey, N. (2020). Data-driven medium-range weather prediction with a resnet pretrained on climate simulations: A new model for weatherbench. *arXiv: Atmospheric and Oceanic Physics*.

Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N., and Prabhat (2019). Deep learning and process understanding for data-driven earth system science. *Nature*, 566:195–204.

Scher, S. (2018). Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning. *Geophysical Research Letters*, 45:12616–12622.

Schultz, M., Betancourt, C., Gong, B., Kleinert, F., Langguth, M., Leufen, L. H., Mozaffari, A., and Stadtler, S. (2021). Can deep learning beat numerical weather prediction? *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 379.

Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., and Woo, W. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*.

Sønderby, C., Espeholt, L., Heek, J., Dehghani, M., Oliver, A., Salimans, T., Agrawal, S., Hickey, J., and Kalchbrenner, N. (2020). Metnet: A neural weather model for precipitation forecasting. *ArXiv*, abs/2003.12140.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *NIPS*.

Tektas, M. (2010). Weather forecasting using anfis and arima models. *Environmental Research, Engineering and Management*, 51:5–10.

Varsamopoulos, S., Bertels, K., and Almudéver, C. G. (2018). Designing neural network based decoders for surface codes.

Viswanath, S., Saha, M., Mitra, P., and Nanjundiah, R. (2019). Deep learning based lstm and seqtoseq models to detect monsoon spells of india. In *ICCS*.

Xiang, Z., Yan, J., and Demir, I. (2020). A rainfall-runoff model with lstm-based sequence-to-sequence learning. *Water Resources Research*, 56.

Zahraei, A., Hsu, K., Sorooshian, S., Gourley, J., Lakshmanan, V., Hong, Y., and Bellerby, T. (2012). Quantitative precipitation nowcasting: A lagrangian pixel-based approach. *Atmospheric Research*, 118:418–434.

Zantedeschi, V., Martini, D., Tong, C., Schroeder, C., Kalaitzis, F., Bilinski, P., Chantry, M., and Watson-Parris, D. (2020). Towards data-driven physics-informed global precipitation forecasting from satellite imagery.

Zaytar, A. and Amrani, C. E. (2016). Sequence to sequence weather forecasting with long short-term memory recurrent neural networks. *International Journal of Computer Applications*, 143:7–11.