Deep Learning for Weather and Climate Science

A Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Technology

by

Rakesh Kumar
(111701024)

COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

CERTIFICATE

*This is to certify that the work contained in the project entitled "**Deep Learning for Weather and Climate Science**" is a bonafide work of **Rakesh Kumar (Roll No. 111701024)**, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Palakkad under my guidance and that it has not been submitted elsewhere for a degree.*

Dr. Chandra Shekar Lakshminarayanan
Assistant Professor
Department of Computer Science & Engineering
Indian Institute of Technology Palakkad

Contents

List of Figures

Chapter 1

Introduction

Weather forecasting on a very short term period upto 6 hours is known as Nowcasting. Nowcasting precipitation has been an important problem in the field of weather forecasting. The goal is to provide good accuracy and timely prediction of precipitation, sometimes evolution of storm structure, hail potential, etc. over a region for a short time period (0-2 hours). These predictions are useful for producing rainfall, storms, hails, etc alerts, providing weather guidance for air traffic, marine, etc.

In 21st century, most weather nowcasting methods are NWP (Numerical Weather Prediction) based methods and some are radar reflectivity extrapolation based methods.

Weather Radar's reflectivity is used by scientists to detect precipitation, determine hail potential, evaluate storm structure, etc. Sequence of radar reflectivity over a region for some time duration has spatio temporal nature. Weather nowcasting can also be seen as a spatiotemporal sequence forecasting problem with the sequence of past few hours reflectivity maps as an input and the sequence of future reflectivity maps as an output. Building an effective model that could learn this problem is challenging due to chaotic nature of the atmosphere as well as due to high dimensionality of the sequences. Doing a multi-step predictions further increases the difficulty of the problem.

Recent advances in deep learning, especially RNNs (more precisely LSTMs) can help to tackle this sequence-to-sequence prediction problem. The LSTM encoder-decoder architecture is a general framework used in various sequence-to-sequence learning problems such as machine translation, etc. The NeurIPS 2015 [2] paper provides the ConvLSTM (Convolutional LSTM) Encoder-Decoder Network based model for this problem. In my project, I have implemented the same for the problem. This deep learning model would come under radar reflectivity extrapolation based methods.

1.1 Organization of The Report

This chapter provides a brief introduction about the weather nowcasting problem. Chapter 3 gives the background of the components (LSTM and ConvLSTM) which are used in models for the sequence problem in later chapters. The next chapter describes about the models used for the image frame sequence problem. In chapter 5, the models were trained and tested on an artificially generated moving MNIST dataset. Chapter 6 shows the result of models prediction on radar images.

Chapter 2

Review of Prior Works

2.1 Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting [2]

The paper explains the Encoding-Forecasting Structure of Convolutional LSTM Networks for precipitation nowcasting problem. It compares the performance of Convolutional LSTM with FC-LSTM, ROVERs, etc. over the radar echo dataset. It also compares the model for the synthetic Moving MNIST Dataset with some other models.

2.2 Unsupervised Learning of Video Representations using LSTMs [7]

The paper proposes variants of LSTM Encoder-Decoder model : LSTM Autoencoder Model, LSTM Future Predictor Model, Composite Model etc. The paper compares the performance of various models on moving MNIST dataset, UCF-101 dataset, etc.

In this project, I have implemented the ConvLSTM Encoder-Decoder models (described in [2]) and compared the performances on moving MNIST dataset and Radar reflectivity.

Chapter 3

Background

3.1 LSTM Networks [1]

Long Short Term Memory (LSTM) Networks are well suited to classifying, processing and making predictions based on time series data. LSTM Networks are a special kind of RNN which are capable of learning long-term dependencies much better than RNNs. In theory, RNNs are capable of learning long-term dependencies in sequences but in practice it doesn't seem to be able to learn. If the gap between relevant information and the point where it is needed is small, RNNs can learn and perform well. But if the gap is large, RNNs fail to learn to connect the information. In other words, RNNs can have vanishing gradient problem during training. LSTMs are explicitly designed to avoid the vanishing gradient problem. All RNNs have the form of a chain of repeating modules of neural networks. In standard RNNs, the repeating module has a simple structure such as a single tanh layer. LSTMs also have this chain structure, but the repeating module has a different and more complex structure.

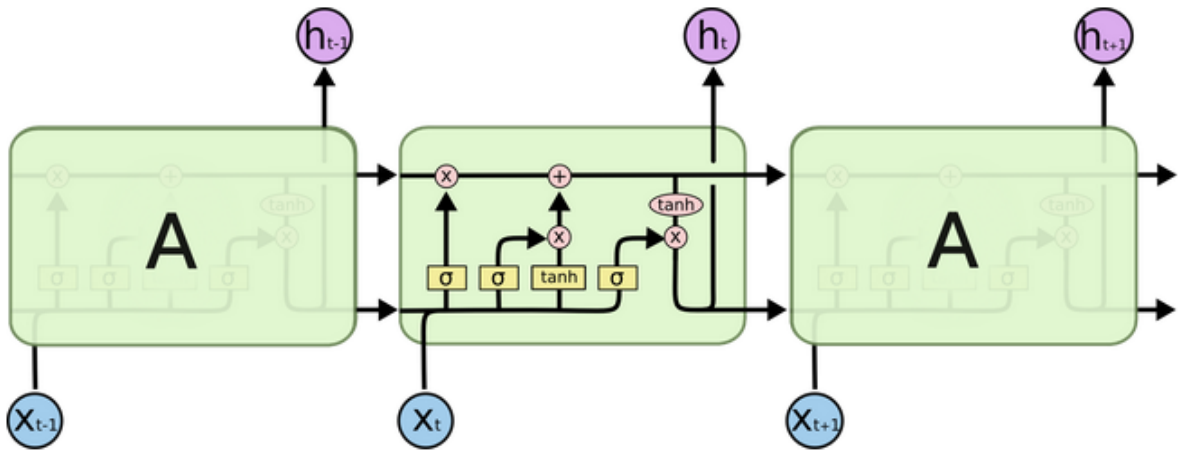


Fig. 3.1 The internal structure inside an LSTM cell [1]

LSTMs have structures called gates which regulates the information flowing through the cells. These gates in the cell decide what to forget, what to remember from the state and what to output (if anything). The cell state c_t acts as an accumulator of the state information. The forget gate f_t tells which information should be kept from previous cell state c_{t-1}. The input gate i_t decides what new information has to be stored. The cell state c_t is updated using the above two gates. The output gate o_t controls the output (also called hidden state) of the cell h_t. The following are equations for the above:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}\ o\ c_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}\ o\ c_{t-1} + b_f)$$
$$c_t = f_t\ o\ c_{t-1} + i_t\ o\ tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}\ o\ c_t + b_o)$$
$$h_t = o_t\ o\ tanh(c_t)$$

Here, $'o'$ denotes the Hadamard product (element-wise product)

$x_t \in \mathbb{R}^d$ and $i_t, f_t, o_t, c_t, h_t \in \mathbb{R}^h$ where d and h refers to the number of input features and number of hidden units respectively.

3.2 Convolutional LSTM [2]

In Convolutional LSTM, the input will be a 3D vector (row, col and channel of an image frame). Original LSTM gates equations are modified a bit to work the image inputs. The matrix multiplications in LSTM equations are replaced by convolutional operations in ConvLSTM. The inputs $X_1, X_2, .., X_t$, the cell outputs $C_1, C_2, .., C_t$, the hidden states $H_1, H_2, .., H_t$, and the gates i_t, f_t, o_t of the ConvLSTM are 3D vectors with last two dimensions being rows and columns ($\mathbb{R}^{rows \times cols \times h}$). The ConvLSTM determines the future state of a certain cell in the grid by the inputs and past inputs of its local neighbors. The equations are,

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

where $'o'$ denotes the Hadamard product (element-wise product) and $'*'$ denotes the convolution operator

The states of a ConvLSTM can be viewed as a hidden representation of moving objects. A larger kernel size is more suitable for capturing faster motions while smaller kernel works well for slower motions.

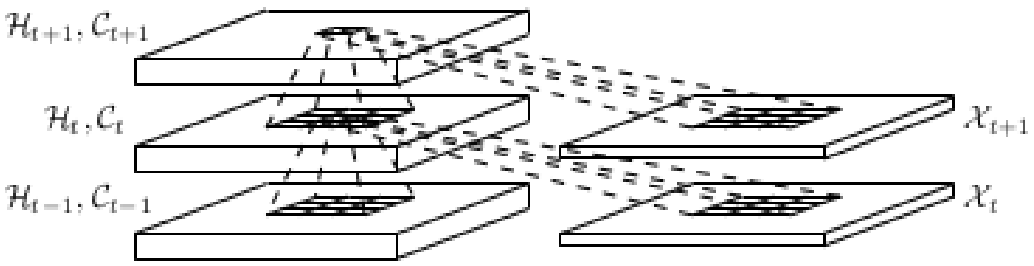


Fig. 3.2 Inner structure of Convolutional LSTM [2]

Chapter 4

Models

4.1 Stacked ConvLSTMs

In this model, Convolutional LSTM layers are stacked over one another with a 1 x 1 Convolutional layer at the top.The input frame sequences are feed to the bottommost ConvLSTM layer. The output sequence (or hidden state sequence) from one ConvLSTM layer is served as the input to the next ConvLSTM layer. The outputs from the topmost layer are passed to the Convolutional layer which produces the next frame in the sequence. The 1 x 1 convolutional layer is used to generates the output frame with same channel dimensionality as the in input frames. The model is many-to-one which means it predicts the next one frame after the given input frame sequence. For next two frame predictions, the predicted frame is added to the given input frame sequence which is then fed to the network as the new input frame sequence. So, many-to-many sequence predictions can be done in a similar manner.

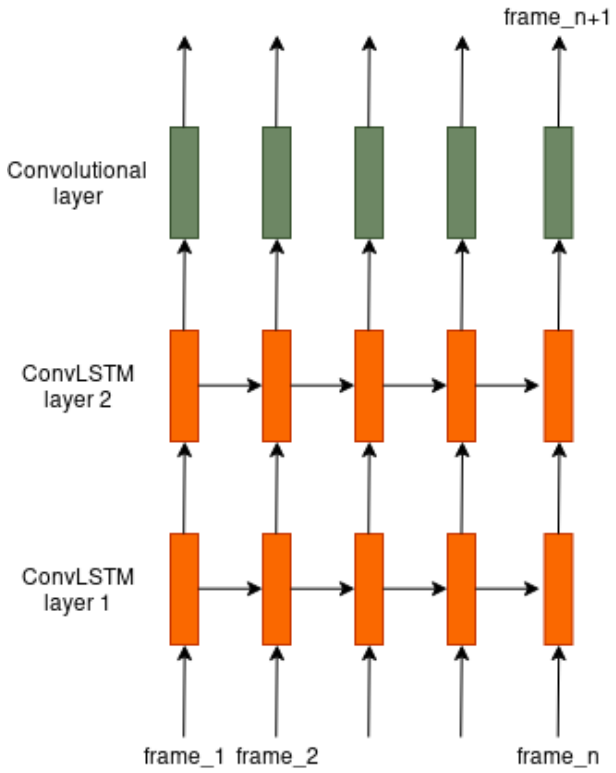


Fig. 4.1 2 layer Stacked LSTM

4.2 Encoder-Decoder Network [2]

The above model can't be used for many-to-many predictions when the output sequence is not from the same input sequence domain as the outputs can't be feed as inputs to the network. Machine translation (eg., English to French translation) is one such different domain sequence-to-sequence prediction problem. Encoder-Decoder network is a general model used for predicting output sequence of required length.

Encoder-Decoder network consists of two networks: Encoder and Decoder. Encoder network processes the input sequence and returns its states (hidden state + cell state) as vectors. The outputs of the Encoder network are discarded. Decoder network gets the state vectors from

the encoder as its initial states and it generates the output sequence with the required length. Both encoder and decoder are made up of stacked ConvLSTMs but encoder network doesn't have the convolutional layer at the top while the decoder has a 1 x 1 convolutional layer at the top.

In the decoder network, the input frame at any time step is the output frame from the previous timestep . During the training, instead of feeding the output as the input to the decoder at the next timestep, the target or actual frame is passed as the input. This training process is known as a teacher forcing. Thus, the decoder is trained to predict the next frame after the given input frame.

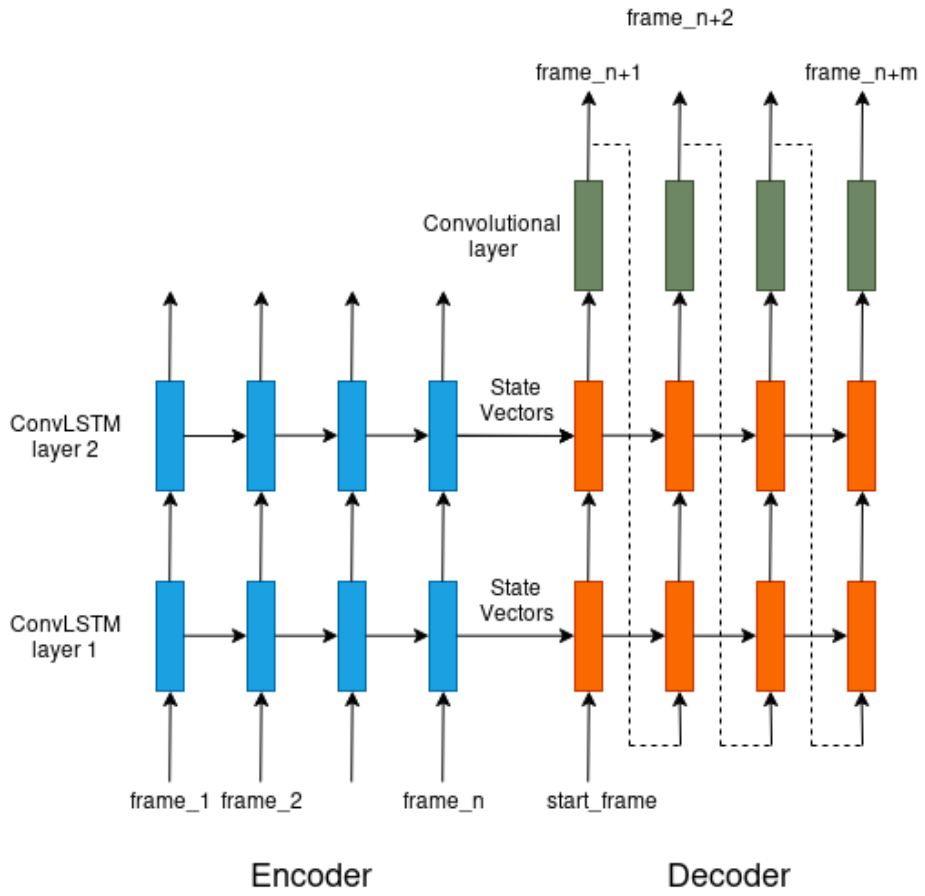


Fig. 4.2 2 layer Encoder-Decoder Structure

4.3 Experiment on a simple Artificial Data

This artificially generated dataset contains image frame sequences (or movies) with 3 to 7 moving squares in a 40 x 40 sized frame. The squares are of shapes 1 x 1 or 2 x 2 and move with constant velocity. The squares have intensity 1 while background has intensity 0. The dataset generation code was taken from the Keras next-frame prediction tutorial [3]. The dataset was used to simply test the working of the models.

4.3.1 Stacked ConvLSTMs

Four layer ConvLSTM stacks with 40 hidden units (or no. of filters) with filter size of 3 x 3 was trained. The top convolutional layer had sigmoid activation for non-linearity as well as to keep the intensities in the output frame between 0 and 1. It was trained to minimize the binary crossentropy loss and Adadelta optimizer was used during the training. The model was trained

for 50 iterations over 900 samples. Each sample had a 20 frames sequence. The final loss was 0.0055.

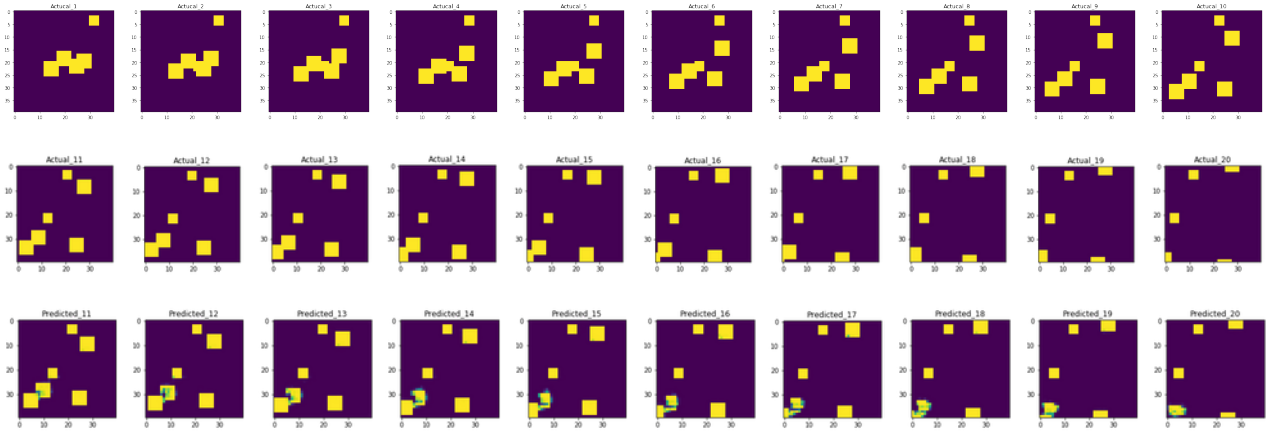


Fig. 4.3 An example from the test set. From left to right: input frames; ground truth; prediction by the model

4.3.2 Encoder-Decoder Model

One layer model with 128 hidden units with filter size of 3 x 3 was trained. The top convolutional layer in the decoder structure had sigmoid activation. The model was trained to minimize the average binary crossentropy loss using back-propagation through time and RMSProp optimizer (learning rate = 0.001 and decay rate = 0.9) was used. It was trained for 20 iterations over 1000 samples. The final loss was 0.0001.

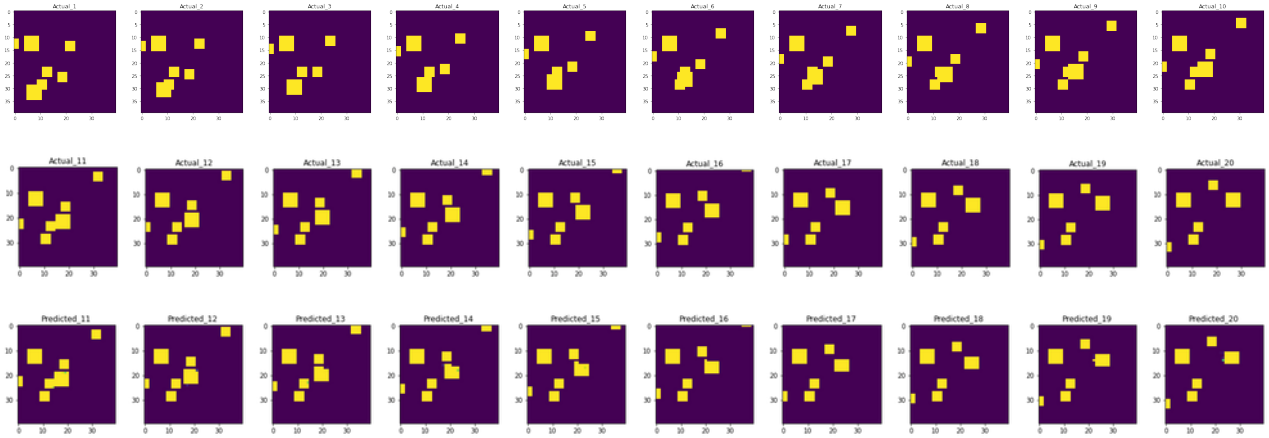


Fig. 4.4 An example from the test set. From left to right: input frames; ground truth; prediction by the model

Chapter 5

Moving MNIST Dataset

Moving MNIST Dataset consists of sequences of 20 image frames showing 2 handwritten digits from the MNIST database moving in the 64 x 64 frame. Each sequence is generated by randomly choosing 2 digits from the MNIST database and each of them is assigned a random velocity in the beginning. The digits move inside the frame and bounce back when they reaches the edge of the frame. Moving MNIST dataset of 10,000 sequences can also be found at `http://www.cs.toronto.edu/~nitish/unsupervised_video/`

The Moving MNIST dataset frame has 256 levels (0 - 255). The frames were thresholded to have binary intensities (0 and 1). Each 20 frames sequence was splited into two sequence of 10 frames (first 10 frames for input and other 10 for prediction). The models were trained to minimize the binary crossentropy loss and RMSProp optimizer (learning rate = 0.001 and rho = 0.9) was used during the training. Also, early stopping was performed on the validation set.

- 2 Layers Encoder-Decoder network with 32, 32 hidden units and (5 x 5) filter size in each layer: This model was trained over 1,000 sequences and tested on 200 sequences (200 validation sequences). The average binary crossentropy loss was 0.2998.

Stacked LSTM was also tested on the dataset . Both models performed satisfactorily on the first 2-3 frames prediction but in the next 7-8 frames, digits were distorted and weren't recognizable.

5.1 Image Reshaping

The above models were trained on NVidia K80 GPU provided by Kaggle with a weekly quota of around 40 hours. The weights in the ConvLSTM cells depends on `batch_size x image_rows x image_cols x filters`. Training deeper models with many hidden units required good memory space. So, deeper Encoder-Decoder model couldn't be trained due to the limited resource. Since, the memory allocated for ConvLSTM depends on `image_size (rows x cols)`, the 64 x 64 frames were reshaped into 16 x 16 x 16 frames (increasing the channel dimension). This allowed me to train deeper Encoder-Decoder networks. Two Encoder-Decoder networks were trained on the sequences (8,000 training sequences, 1,000 validation sequences and 1,000 test sequences).

- 2 Layers with 128, 128 hidden units and (5 x 5) filter size in each layer. The average binary crossentropy loss was 0.2791.
- 3 Layers with 128, 64, 64 hidden units and (5 x 5) filter size in each layer. The average binary crossentropy loss was 0.2662.

In the below figures, 32-32 model means Encoder-Decoder network having 2 layers with 32 hidden units in each layer. 128-128 and 128-64-64 models can also be interpreted in similar ways. Deeper models performed better but there was not much difference between 128-128 and 128-64-64 model.

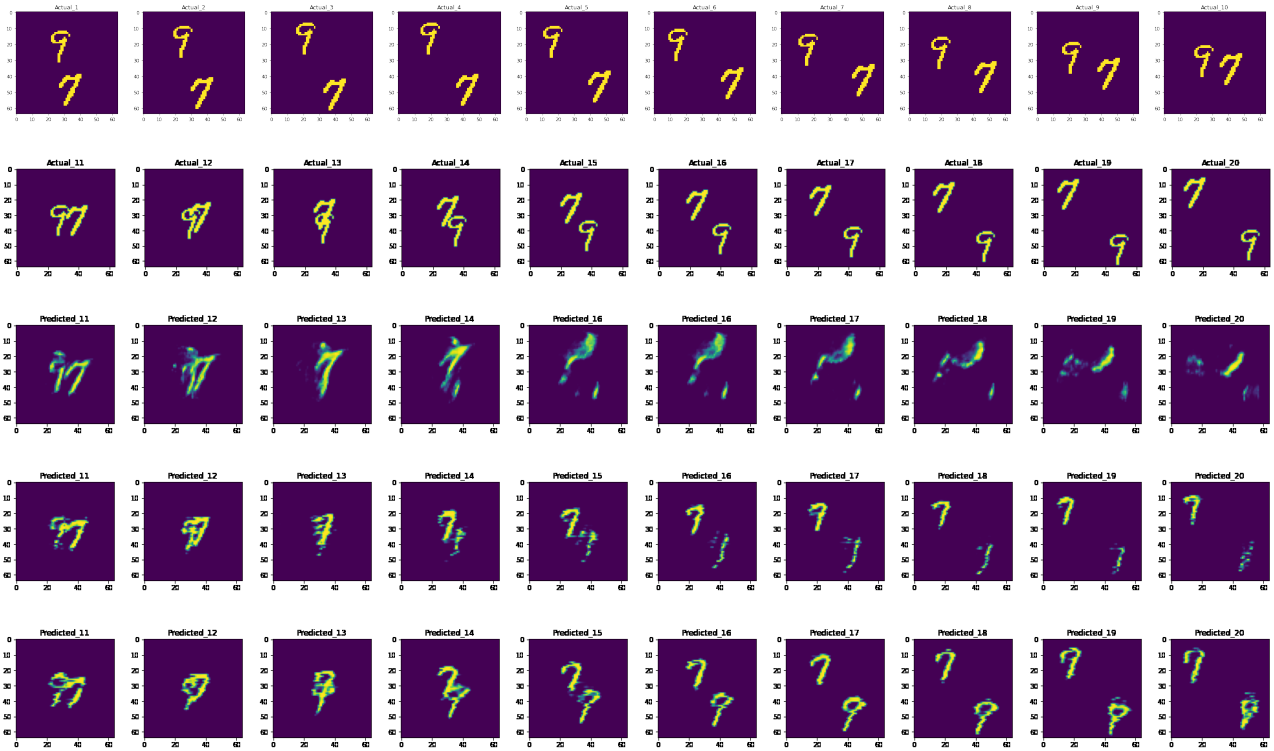


Fig. 5.1 An example from the test set. From left to right: input frames; ground truth; 32-32 model predictions (without image reshaping); 128-128 model predictions; 128-64-64 model predicitons

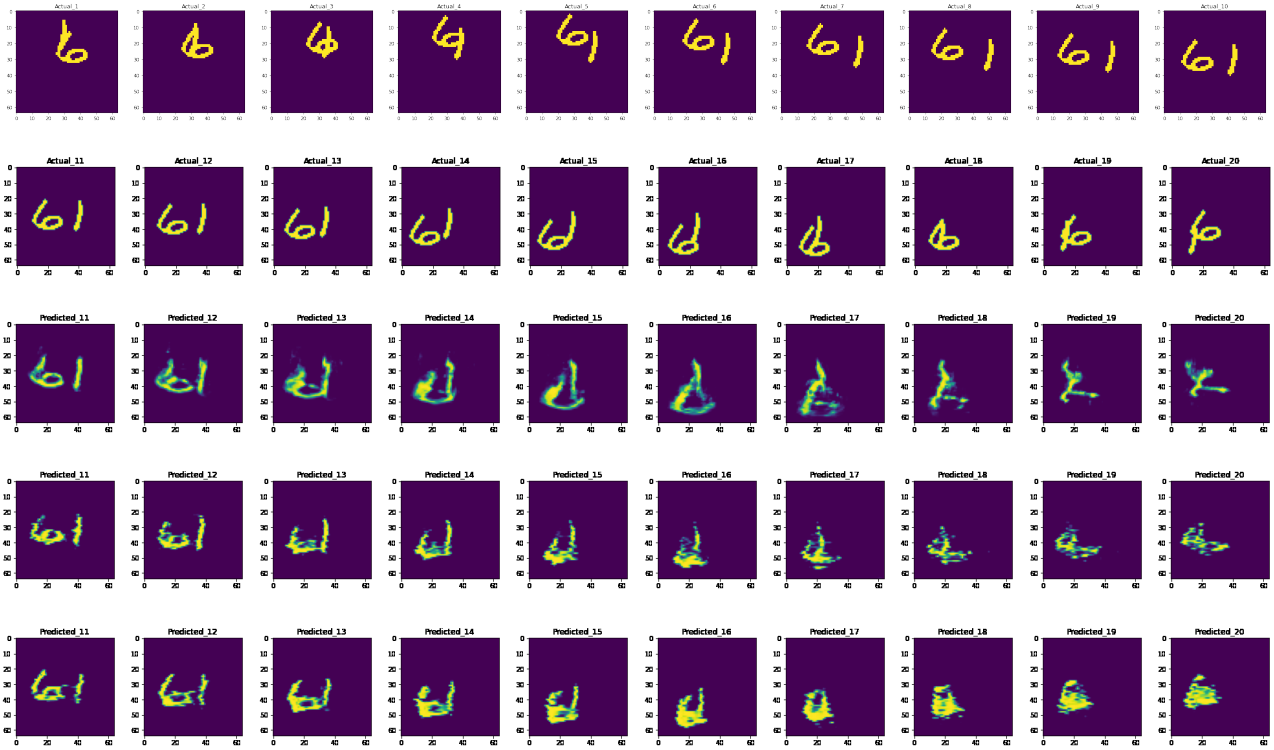


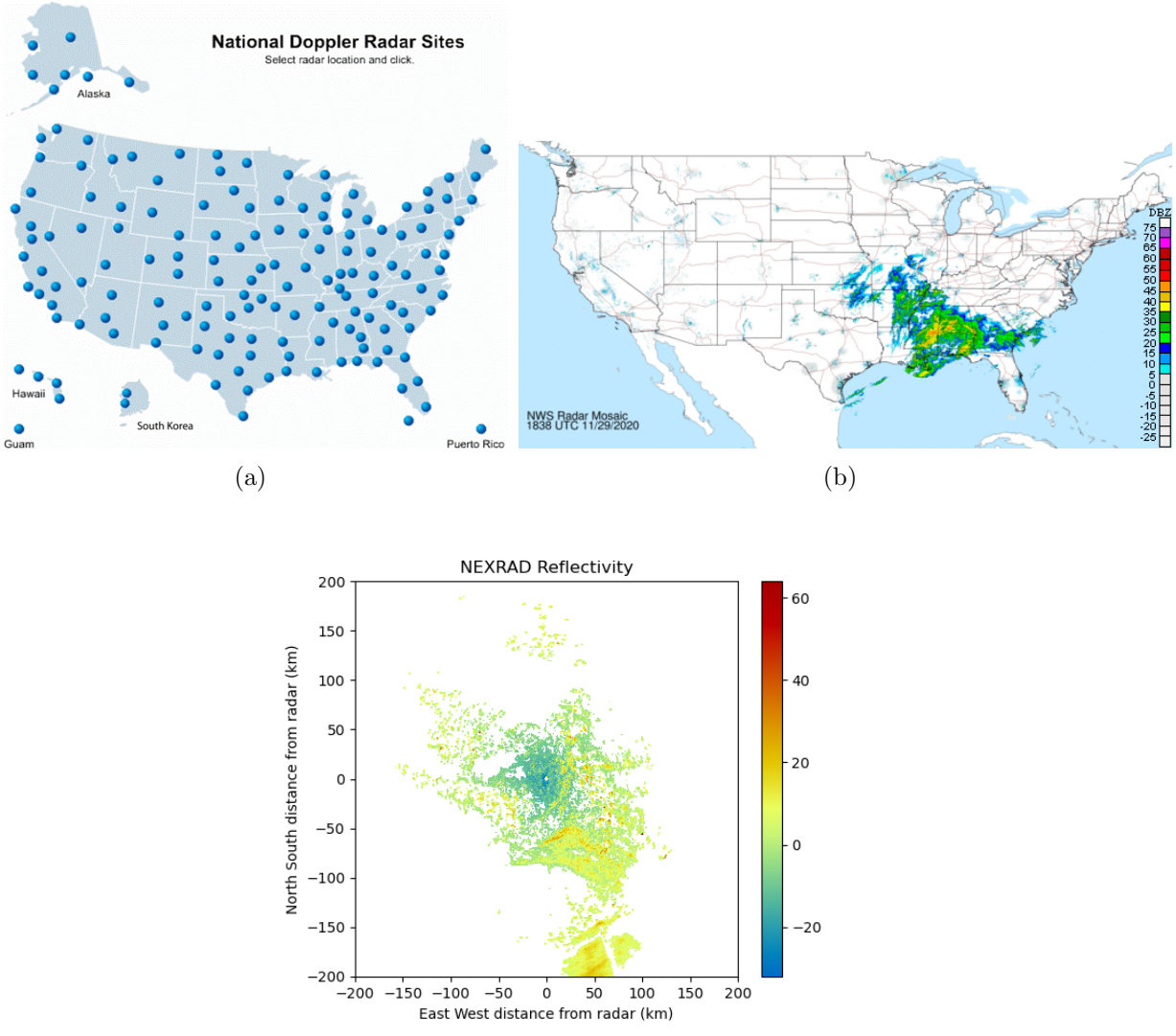
Fig. 5.2 An example from the test set. From left to right: input frames; ground truth; 32-32 model predictions (without image reshaping); 128-128 model predictions; 128-64-64 model predicitons

Chapter 6

Weather Forecasting using NEXRAD

The Next Generation Weather Radar (NEXRAD) [4] system currently has around 160 radar sites throughout the United States and some nearby overseas locations. NEXRAD detects precipitation and atmospheric movement or wind. This radar data which after being processed can be displayed in a mosaic map which shows patterns of precipitation and its movement. NEXRAD Level-II (Base) data include the original three meteorological base data quantities: reflectivity, mean radial velocity, and spectrum width. Data is collected from the radar sites usually at the interval of 4, 5, 6 or 10 minutes depending upon the volume coverage. Radar Data can be accessed at `https://www.ncdc.noaa.gov/nexradinv/`. There are different ways to access the data. For eg., Data can accessed by Single Site and Day, Multiple Sites and Days, etc.

Radar's reflectivity is used to detect precipitation, determine hail potential, evaluate storm structure, etc. Reflectivity is expressed in dBZ. Higher value of reflectivity tells heavy precipitation or hail at that place and lower value tells light precipitation. So, reflectivity component from the Level-II data can be used for weather forecasting for short duration.



(a) (b)

(c)

Fig. 6.1 (a) Available Radar Sites [5]; (b) Reflectivity of Radars combined [6]; (c) Reflectivity plot of Seattle site radar

In this project, weather forecasting was done for two regions : Seattle, WA and South Shore, Hawaii. For each region, radar level-II data was collected for some duration and reflectivity plots were extracted. These plots or images were resized into 100 x 100 images using nearest-neighbor interpolation. Further, the images were converted to gray scale and later thresholded to have binary intensities. These image sequences were later used for training and testing the models. For each region, weather forecasting was done independently. (For simpler dataset, the images were modified to have only two intensities and the model were trained to predict only the shapes not the intensities)

6.1 PHWA-SOUTH SHORE, HAWAII

PHWA is the id of radar at South Shore, Hawaii. A dataset of 959 sequences with 20 radar maps or images in each sequence was created by collecting radar data of around 30 days from August-2020 to October-2020. Time gap between each frame of a sequence was around 5 minutes. These sequences were separated into 700 training sequences, 100 validation sequences and 159 test sequences. The following Encoder-Decoder networks were trained for forecasting.

- 2 Layers with 64, 48 hidden units and (3 x 3) filter size in each layer. The input frames were reshaped into 50 x 50 x 4 vectors. The average binary crossentropy loss was 0.1491.

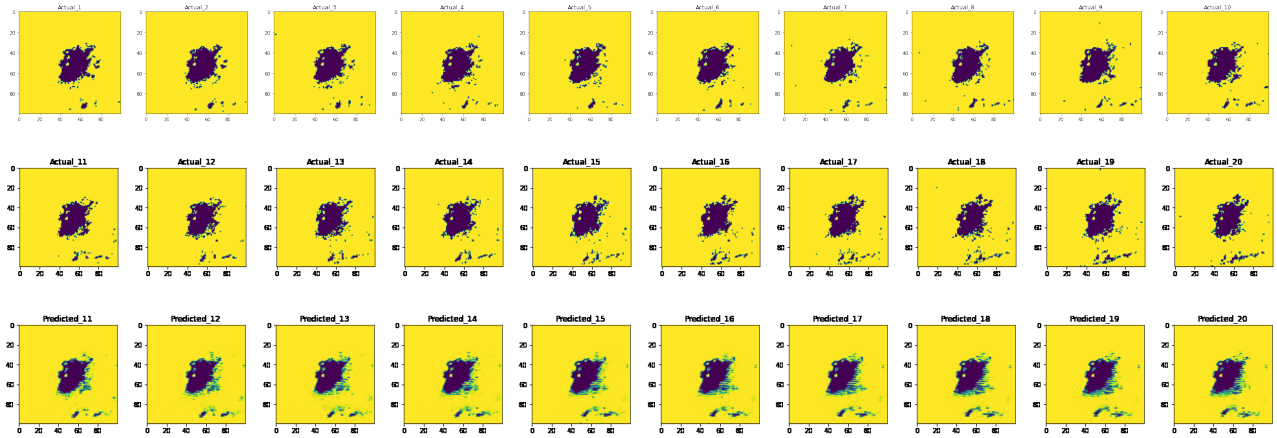


Fig. 6.2 An example from the test set. From left to right: input frames; ground truth; prediction by the model; Click here for better visualization

- 4 Layers with 96, 64, 64, 32 hidden units and (3 x 3) filter size in each layer. The input frames were reshaped into 25 x 25 x 16 vectors. The average binary crossentropy loss was 0.1790.

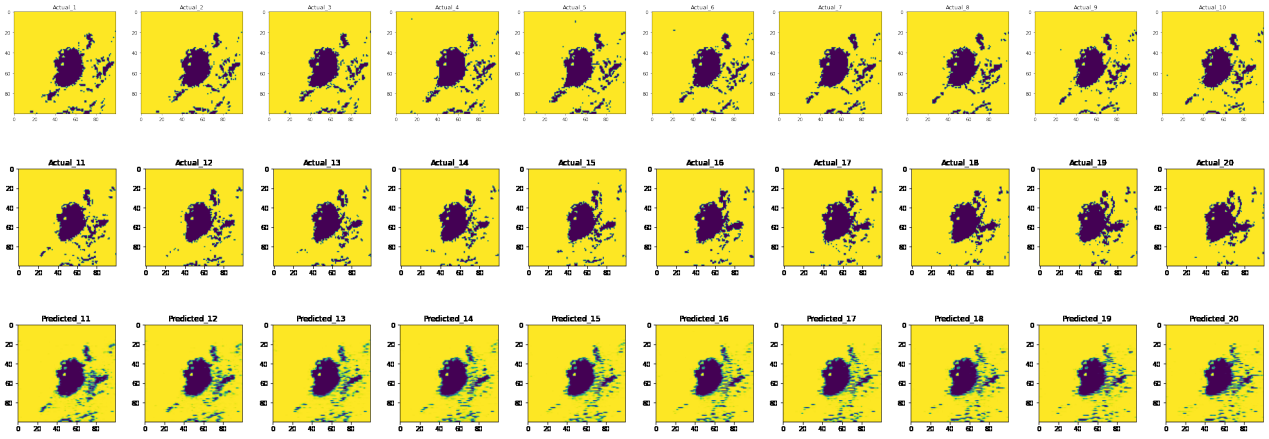


Fig. 6.3 An example from the test set. From left to right: input frames; ground truth; prediction by the model; Click here for better visualization

Frames were reshaped to increase the channel size so that deeper models could be trained on limited resources but increasing the frame channel size too much resulted in bad performance.

6.2 KATX-SEATTLE, WA

Radar id at Seattle, WA is KATX. A dataset of 499 sequences with 20 radar maps in each sequence was created by collecting radar data of around 30 days from January-2020 to April-2020. Time gap between each frame of a sequence was around 10 minutes. These sequences were separated into 350 training sequences, 75 validation sequences and 74 test sequences.

- 4 Layers with 96, 96, 32, 32 hidden units and (3 x 3) filter size in each layer. The input frames were reshaped into 25 x 25 x 16 vectors. The average binary crossentropy loss was 0.3761.

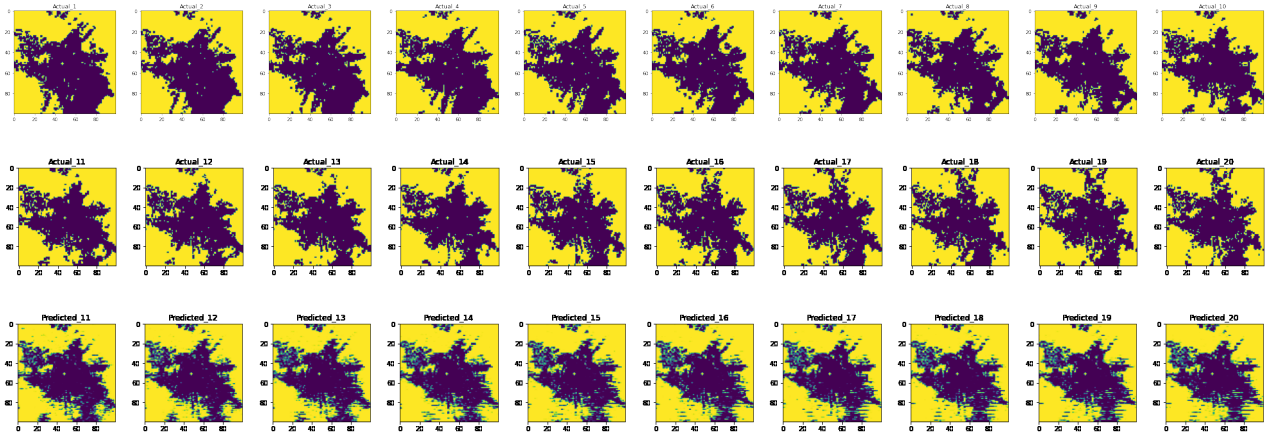


Fig. 6.4 An example from the test set. From left to right: input frames; ground truth; prediction by the model; Click here for better visualization

Radar reflectivity at a site doesn't change much in 1-2 hours. There are very changes in the radar maps which makes it difficult for the models to predict correctly. The results are not that good. To improve the result, several radar maps over a region could be combined together. This will give the model a better understanding of the nature of the weather over that region.

Chapter 7

Repository, Conclusion and Future Work

7.1 Repository

The Github repository of this project can be found at : `https://github.com/iamrakesh28/Deep-Learning-for-Weather-and-Climate-Science/`.

7.2 Conclusion and Future Work

Weather forecasting using just one radar map doesn't work well. The reason could be that several nearby radar reflectivity maps are related. For example, a storm may move from one location to another . It may move over several radar sites. So, instead of using one radar maps for prediction, several maps could be combined to produce reflectivity map over larger regions. This map may be used for training and could produce better results. In this project, ConvLSTM Encoder-Decoder was used. Other models such as LSTM Autoencoder, LSTM Future Predictor Model, etc (described in [7]) could also be used.

References

[1] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[2] Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting

[3] https://keras.io/examples/vision/conv_lstm/

[4] https://www.ncdc.noaa.gov/data-access/radar-data/nexrad

[5] https://www.ncdc.noaa.gov/nexradinv/map.jsp

[6] https://radar.weather.gov/Conus/index.php

[7] Unsupervised Learning of Video Representations using LSTMs