Precipitation Nowcasting Using Deep Learning

Sarwan Shah[1]

Abstract— **Precipitation nowcasting for short-term storm forecasting (0–6 hours) is essential for timely severe weather warnings. Traditional methods such as numerical weather prediction (NWP) and radar extrapolation, often lack accuracy at short scales and are computationally intensive. Recent deep learning models, such as ConvLSTM and TrajGRU have offered promising advances by capturing complex spatiotemporal dynamics. This paper aims to evaluate these models on satellite data, addressing the limitation posed radar's limited global coverage, while focusing on the region of Sindh, Pakistan — a region with minimal meteorological infrastructure. Thus, by contributing towards the improvement of global nowcasting capabilities this work addresses critical forecasting needs heightened by climate change.**

INTRODUCTION

Precipitation nowcasting aims to predict the movement, evolution, and intensity of storm systems over short timescales (0–6 hours), enabling timely and accurate severe weather warnings. Such warnings can be life-saving and help minimize public disruption by supporting faster decision-making. This capability is becoming increasingly critical as climate change leads to more frequent and unstable weather events.

Typically, weather forecasting is conducted using numerical weather prediction (NWP) systems. However, these systems are computationally demanding at short spatial and temporal scales and can yield inaccurate results. This is because short-scale meteorological phenomena, like convection (which often leads to precipitation), are turbulent and challenging to model accurately. Consequently, precipitation nowcasting is conventionally performed using extrapolation techniques on Doppler radar and satellite images to predict future storm motion and evolution. More advanced methods have incorporated optical flow estimation algorithms on radar images for improved accuracy. For example, ROVER (Real-time Optical flow by Variational methods for Echoes of Radar) was developed for the SWIRL (Short-range Warning of Intense Rainstorms in Localized System) project in Hong Kong. However, even with these advances, the accuracy of such estimation algorithms remains below 60% and decreases significantly for longer forecast windows (over an hour).

In recent years, deep learning has produced impressive results across complex tasks such as segmentation, recognition, and generation involving semantic and visual data. Following this trend, deep learning has been applied to precipitation nowcasting, using neural network architectures capable of capturing spatial and temporal relationships, such as Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) cells, and Vision Transformers. These approaches have shown significant improvement in precipitation nowcasting literature compared to the conventional methods that were discussed. The following section provides a more detailed discussion of recent deep learning research in precipitation nowcasting.

This paper's contributions include the following: (1) We evaluate the state-of-the-art ConvLSTM and TrajGRU models for precipitation nowcasting using satellite data as current experiments with these models have been limited to radar data, which has limited global coverage. (2) We perform experiments in the region of Sindh, Pakistan - a region with minimal meteorological infrastructure and strong monsoonal influence (i.e. heavy precipitation during monsoon). This because prior studies [4, 5] tested these models only at single location and that was identified as a limitation by the authors. (3) (Secondary Objective) We aim to explore a potential transfer learning approach which would involve training models on regions with both radar and satellite data, as seen in [6], and then retraining and testing them for satellite-based nowcasting in regions that only have satellite data for improved results. These contributions will help address a critical infrastructure gap in global precipitation nowcasting, meeting the amplified needs driven by accelerating climate change.

[1]Sarwan Shah, Graduate Student, Electrical Engineering, Arizona State University

Literature Review

Recurrent Neural Networks for Visual Sequence Prediction. The task of precipitation nowcasting is inherently sequential, as it requires using a set of past states (of a storm system) to predict a set of future states. In the context of deep learning, this requirement can be met using recurrent neural networks (RNNs). To handle spatiotemporal visual data, [1] introduces a recurrent convolutional neural network (RCNN) as a baseline model for unsupervised and supervised learning on video data. The paper claims that by training the model to recreate or predict missing/future frames in a fixed-length visual sequence, it learns the underlying spatial and temporal dynamics such as motion, without labeled data. However, this model is limited to very short temporal scales and is tested only on the UCF-101 human action recognition dataset.

LSTM Architectures for Sequence Learning. For longer temporal scales, it is important for the model to account for both short- and long-term dependencies in visual sequences. Additionally, it is desirable for input and output sequences to have variable lengths. These requirements are addressed in [2], which uses LSTM cells in an encoder-decoder architecture for sequence-to-sequence learning. Here, the encoder produces a context vector based on an input sequence, which is then used by the decoder to generate a variable-length output sequence.

Building on [1] and [2], [3] presents an encoder-decoder LSTM architecture where the encoder network creates a fixed-length vector representation of the visual input sequence, which the decoder then uses to generate a variable-length output sequence. This study explores both unconditional (where learning relies solely on the encoded vector representation without loss computation on generated output) and conditional decoders, creating a composite model that uses one decoder to reconstruct the input sequence (unconditional) and another to predict future frames (conditional).

ConvLSTM for Spatiotemporal Prediction. [4] identifies that while [3] captures temporal relationships in the visual sequence, its fixed-length vector representation fails to capture spatial relationships, which are crucial for the precipitation nowcasting task. Thus, [4] introduces an encoder-decoder ConvLSTM architecture with convolutional layers between input-to-state and state-to-state transitions, allowing spatial relationships to be captured. It is noted that by increasing the convolution kernel length, the model can capture faster motions. The model demonstrated significant improvement over the ROVER algorithm in predicting future rain intensity using radar images, achieving an average RMSE of 1.42 mm/h with a lead time of 1.5 hours. The work also validates its model using the Moving MNIST dataset.

Trajectory Gated Recurrent Units for Location-Variant Transformations. One limitation highlighted by [5] in ConvLSTM [4] is its inability to capture location-variant transformations, such as rotation and scaling, which are significant in precipitation radar images. In ConvLSTM, these transformations can be lost due to the fixed receptive field of the convolutional kernel, which captures neighboring information uniformly. [5] introduces a TrajGRU (Trajectory Gated Recurrent Unit) architecture that addresses this limitation by dynamically adjusting the spatial dependencies applied at each input location using an adaptive kernel (vs standard convolution kernel). The adaptive kernel uses a structure-generating sub-network to create flow fields based on the current input and previous hidden state. These flow fields define a trajectory that determines neighbors of each cell instead of a fixed convolutional kernel, allowing the model to flexibly adapt to location-variant transformations and capture complex motion patterns. Additionally, this paper also introduces the Moving MNIST++ dataset, which incorporates rotational and scaling motions, and a balanced MSE metric to evaluate precipitation nowcasting performance on rarer and heavier rainfall events.

Data Fusion and Transfer Learning Approaches. Building on these advancements, which have largely focused on precipitation nowcasting using radar data and are therefore limited in spatial applicability, [6] introduces a machine learning-based data fusion system that improves precipitation estimation by integrating satellite (IR + PWM) and ground radar network observations within a multi-layer perceptron (MLP) model. This approach enhances satellite-based precipitation accuracy, especially for light rain events, by using temporally averaged radar-based rainfall data as target labels. Similarly, [7] utilizes the ConvLSTM architecture from [4] to create the ConvCast model, which is trained and tested on radar data. This model achieves an RMSE of 0.648 mm/h at a 0.5-

hour lead time and 1.929 mm/h at a 2.5-hour lead time.

Emerging Methods: Convolutional and Vision Transformers. Very recently, advanced deep learning approaches involving convolutional and vision transformers have also been applied to precipitation nowcasting [8][9][10]; however, these methods are beyond the scope of this review.

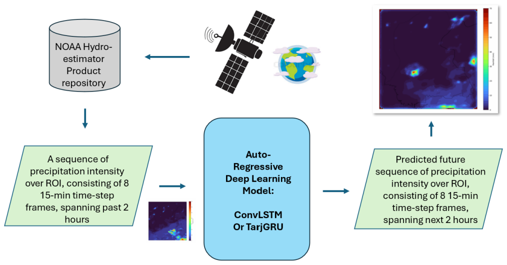
NOAA Hydro-estimator Product repository
A sequence of precipitation intensity over ROI, consisting of 8 15-min time-step frames, spanning past 2 hours
Auto-Regressive Deep Learning Model:
ConvLSTM Or TarjGRU
Predicted future sequence of precipitation intensity over ROI, consisting of 8 15-min time-step frames, spanning next 2 hours

Fig. 1. This figure highlights the intended goal and approach toward solving the precipitation nowcasting problem for the ROI.

METHODOLOGY

In this section, we discuss in detail the methodology that was used to achieve the intended goals of this paper. This methodology is briefly highlighted in figure 1. We begin by formulating the precipitation nowcasting problem mathematically.

Formulating The Problem

Input Data: Each of our input sample is a 2-dimensional (2D) spatial grid X (frame) of size HxW retrieved from satellite data, where each point in the grid represents a precipitation intensity value at a spatial point. The input to our model will be a sequence of t such frames (grids) separated by a fixed time-step.

Output Data: Similarly, our anticipated output from the model is also a sequence of n frames separated by a fixed time-step, where each frames represents precipitation intensity values in 2D spatial grid of size HxW.

Mathematically, the problem of precipitation nowcasting can be stated as: that given a sequence of input frames (from satellite data) X_1, X_2, ... X_t, where $X_t \in R^{H*W}$, how do we predict the future sequence $\hat{X}_t + 1, \hat{X}_{t+2}$,... $\hat{X}_{t+n}$ [4]. Although the lengths of the input and output sequence does not need to be the same i.e. $n \neq t$, to keep things simple we will work with the case where they are equal i.e $n = t$. More formally, our learning problem can be summarized by equation (1).

$$\hat{X}_{t+1}...\hat{X}_{t+n} = argmax\ p(X_{t+1}...X_{t+n}|X_1...X_t) \tag{1}$$

Precipitation Product and Parameter Selection

Beyond our mathematical formulation of the problem, it was also important to define parameters such as the time-step, length of the input-output sequence, and the expanse of our spatial grid. These parameters were largely defined and constrained by the precipitation product (by product we mean big satellite-based datasets that maintain precipitation data, often at a global scale) used and region of interest.

The precipitation nowcasting literature [6][7] used satellite products that had a time-step of 30-min between consecutive frames. The spatial resolution of the NOAA CMORPH product used in [6] was 8km x 8km, and it combined IR retrievals with PWM-based retrievals for higher accuracy of intensity values. [7] Meanwhile, NASA's IMERG product had a spatial resolution was 11km x 11km, and although it had 30-min temporal resolution, it's more calibrated/mature products had a temporal resolution 4-hours.

Since precipitation nowcasting is most useful at short timescales (due to the explosive nature of convective/storm activity) and higher spatial resolutions. After much consideration, for our experiment we used the NOAA Hydro-estimator product, which has a temporal resolution of 15-min and spatial resolution of 4km x 4km, and thus demonstrates a significant advantage over the products used in existing literature. Moreover, a sequence length of $t = n = 8$ (2 hours), as it was deemed sufficient as a baseline for predicting the evolution of storm systems. Additionally, the expanse of spatial grid based on our region of interest (ROI) as approximately shown in figure 2.

Dataset Preparation

To create a dataset that was localized to the ROI and to make sure that the data was not dominated by samples consisting of non-rainy days the approach shown in figure 3 was employed. The script to execute this approach was created in the Google Colab environment and NOAA Hydro-estimator product hosted through the Azure cloud was used. A total of approx

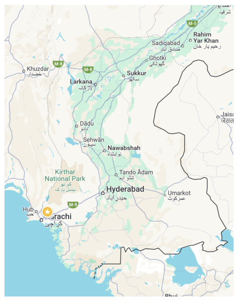

Fig. 2. This figure shows the region of interest between 23°39'03.7"N 66°24'20.5"E and 28°34'18.5"N 71°16'22.9"E for which the data was collected.

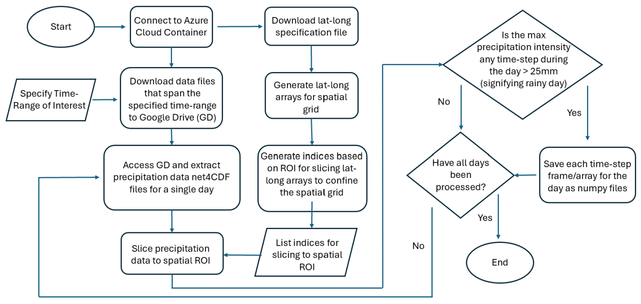
Start
Connect to Azure Cloud Container
Download lat-long specification file
Specify Time-Range of Interest
Download data files that span the specified time-range to Google Drive (GD)
Generate lat-long arrays for spatial grid
Access GD and extract precipitation data net4CDF files for a single day
Generate indices based on ROI for slicing lat-long arrays to confine the spatial grid
Slice precipitation data to spatial ROI
List indices for slicing to spatial ROI
Is the max precipitation intensity any time-step during the day > 25mm (signifying rainy day)
No
Yes
Have all days been processed?
Save each time-step frame/array for the day as numpy files
Yes
No
End

Fig. 3. This figure shows a flow diagram of the proccess that was utilized to create a dataset for the experiments performed in this paper.

$\tilde{1}$30 GB of compressed data ($\tilde{8}$ TB uncompressed) was downloaded from Azure cloud spanning 6 years (2019, 2020, 2021, 2022, 2023, 2024) and for the months of July to September, which mark the Monsoon season in the ROI. Using the aforementioned, a two datasets based on $\tilde{3}$00+ rainy days were created of 1473 and 779 sequential samples. In the first dataset each sequence was 16 frames long (8 as input, 8 as prediction target) and in the second each frame was 32 frames lon (16 as input, 16 as prediction target) represented by a 2D array of 153 by 135 (HxW) covering the ROI ($\tilde{3}$75 km x 375 km). The final size of the dataset was $\tilde{2}$.1 GB. The details of the final dataset are summarized in table I.

To visualize samples from the dataset, a cartographic projection was applied on the 2D array to overlay geographic boundaries on it using the cartopy and matplotlib libraries. Moreover, a color map was used to translate precipitation intensity values into visual features. A partial sequence from the dataset is visualized in figure 4.

Finally, a dataset class was created using Pytorch that organized the data and loaded the precipitation intensity

Parameter	Dataset A	Dataset B
Total Seq.	1560	779
Training Seq.	1092	623
Validation Seq.	312	116
Testing Seq.	156	40
Seq. Length	16	32
Frame Size (HxW)	153x135	153x135
Min Rain Intensity	0	0
Max Rain Intensity	100	100

TABLE I

THIS TABLE SUMMARIZES THE DETAILS ON THE DATASETS THAT WERE CREATED TO CARRY OUT EXPERIMENTS ON THE DISCUSSED PRECIPITATION NOWCASTING PROBLEM.

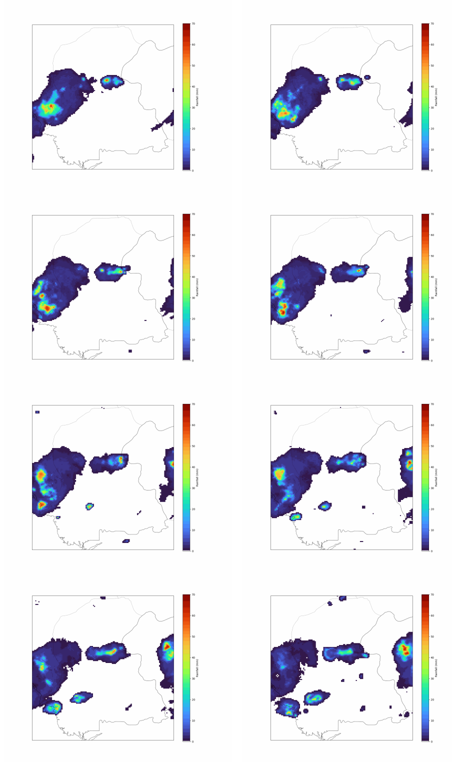

Fig. 4. This figure shows a sequence of 8 frames where each frame shows a precipitation map over the region of Sindh, Pakistan and represents a time-step of 15-minutes (Starting from the top-left, moving left to right, and then top to bottom). With each frame we can see the progressive spatio-temporal evolution of the storm systems.

in sequences of the specified length and passed it to the dataloader function for creating the train, validation, and testing splits.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \tag{2}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f) \tag{3}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{4}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o) \quad (5)$$

$$h_t = o_t \circ \tanh(c_t) \quad (6)$$

Training ConvLSTM

The model creation was simplified by utilizing the code provided by [7], which had already organized the ConvLSTM model as a Pytorch class. The LSTM cell in the ConvLSTM is governed by the equations (2)(3)(4)(5)(6). Compared to the traditional LSTM cell, the ConvLSTM replaces the matrix multiplication between the input-to-gate and gate-to-hidden-state transactions with convolutions. This essentially enables the incorporation of spatial dependencies in the LSTM cell. Moreover, this network was arranged in an encoder-decoder architecture. The architecture for the network is shown in figure 5, the train loss curves can be seen in 6, and the hyperparameters for the best trained model are detailed in table II.

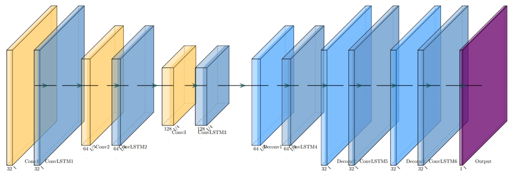

Fig. 5. This figure shows the encoder-decoder convolutional LSTM architecture that gave us the best results during prelimary training.

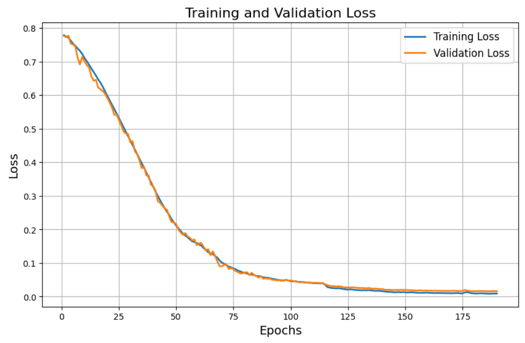

Fig. 6. This figure shows the training and validation loss during training on the best ConvLSTM model with layer normalization.

The model was trained in the Google Colab Pro environment on an A100 GPU using on both datasets: 16 frame sequence vs 32 frame sequence.

During the training of this model it was identified that lowering the learning rate to encourage slower convergence was critical for the model to learn the spatial and temporal features underlying the data. With higher learning rates even though the model was converging

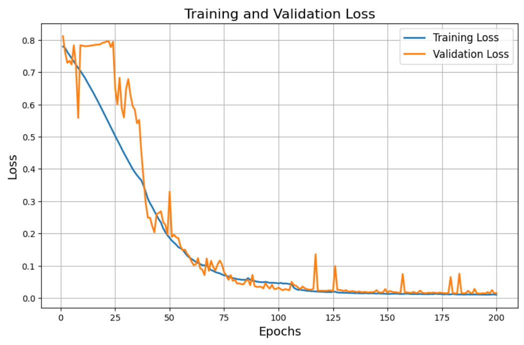

Fig. 7. This figure shows the training and validation loss during training on the best ConvLSTM model with batch normalization.

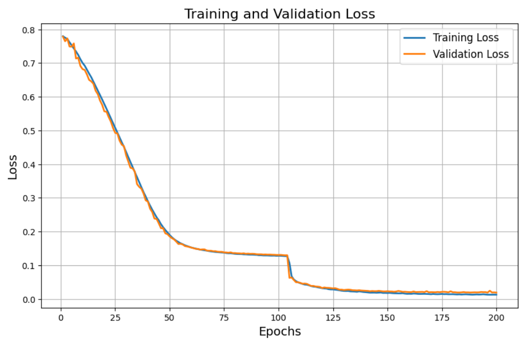

Fig. 8. This figure shows the training and validation loss during training on the best TrajGRU model with layer normalization.

quickly in-terms of loss, but it was failed to capture the visual representation with accuracy.

Additionally, given the regressive nature of the task, various regressive losses were experimented with: MSE, L1Loss, SmoothL1Loss. However, it was observed that with those losses the model failed to learn the visual representation and quickly converged. A perceptual loss (SSIM) was then experimented with, and while the perceptual loss lead to a significant improvement in the visual representation, it struggled to predict regions of high intensity rainfall and had an overall smoothing effect on the precipitation patterns. Finally, a combination of SSIM (70% weightage) and MSE (30% weightage) was used. The MSE was utilized with the idea of penalizing large errors associated with the higher precipitation intensity regions as the model was struggling to learn those with only the SSIM loss. This combination provided us with the best results that balanced visual representation with accuracy in precipitation intensity.

To stabilize training, batch normalization between layers was replaced with layer normalization, which provided more stable training regime as seen in the

Parameter	Value
Model	ConvLSTM
Epochs	200
Batch Size	16
Learning Rate	0.001
Optimizer	Adam
Primary Loss Function	SSIM + MSE
Training Loss	0.091
Validation Loss	0.0160
Test Loss	0.0192
Test MSE Loss	0.0014
Test RMSE mm/hr	3.69 mm/hr
GPU Used	A100
Environment Used	Google Colab Pro

TABLE II

THIS TABLE SUMMARIZES THE DETAILS ON THE HYPERPARAMETERS AND RESULTS ON THE OVERALL BEST TRAINED CONVLSTM MODEL.

Parameter	Value
Model	TrajGRU
Epochs	200
Batch Size	8
Learning Rate	0.001
Optimizer	Adam
Primary Loss Function	SSIM + MSE
Training Loss	0.0126
Validation Loss	0.0188
Test Loss	0.0220
Test MSE Loss	0.0024
Test RMSE mm/hr	4.89 mm/hr
GPU Used	A100
Environment Used	Google Colab Pro

TABLE III

THIS TABLE SUMMARIZES THE DETAILS ON THE HYPERPARAMETERS AND RESULTS ON THE OVERALL BEST TRAINED TRAJGRU MODEL.

difference between figure 6 and 7. Moreover, the training batch size was also critical in ensuring good learning. For dataset A batch sizes greater than 16 caused training stagnated (i.e. 32). Similarly, for dataset B using a batch size greater than 8 (i.e. 16) caused training to stagnate.

Training TrajGRU

The model creation was simplified by utilizing the code provided by [5], which had already organized the TrajGRU model as a Pytorch class. Thus, the ConvLSTM cell in our encoder-decoder was replaced with a TrajGRU cell while keeping the remaining hyperparameters and architecture the same as shown in figure 5 to allow for comparison. The key difference between the TrajGRU and ConvLSTM is that the TrajGRU is more memory and computation efficient while having the potential to deliver improved or similar performance owing to the adaptive convolutional kernel as discussed in the literature review. The train loss curves can be seen in 8, and the hyperparameters for the best trained model are detailed in table III.

RESULTS

ConvLSTM

The best model trained on dataset A (16 length sequences) was tested on 156 test sequences. The model had a loss of 0.0192 on the combined loss, a 0.0014 MSE loss, and an RMSE loss of 3.69 mm/hr with a lead-time of 2 hours at 15-min time steps. Similarly, the best trained model on dataset B (32 length sequences) was tested on 40 test sequences. The model had a loss of 0.007353 on the combined loss, a 0.0065 MSE loss, and an RMSE loss of 8.06 mm/hr with a lead-time of 4 hours at 15-min time steps.

The prediction sequences were visually inspected against their targets as well and promisingly, they showed a great degree of visual accuracy until the very last frame in the sequence. A visual sequence of the prediction, target, and input is shown in figure 11, 9, and 10 respectively. Additional sequences with target on the left and prediction on the right can accessed at the following Google Drive link.

The lower accuracy on dataset B can be attributed to limited availability of training and test data due to longer sequence length. We are confident that with a slightly more complex architecture, hyper-parameter tuning, and more data, the RMSE loss on both models could also been significantly improved. Additionally, experimentation with increasing the weight-age of the MSE loss in the combined loss could also prove to be helpful.

TrajGRU

The best model trained on dataset A (16 length sequences) was tested on 156 test sequences. The model had a loss of 0.0220 on the combined loss, a 0.0024 MSE loss, and an RMSE loss of 4.89 mm/hr with a lead-time of 2 hours at 15-min time steps. This model was not trained on dataset B. The prediction sequences were visually inspected against their targets as well and they showed a good degree of accuracy, but the results appeared less smooth and accurate compared to those achieved with the ConvLSTM.

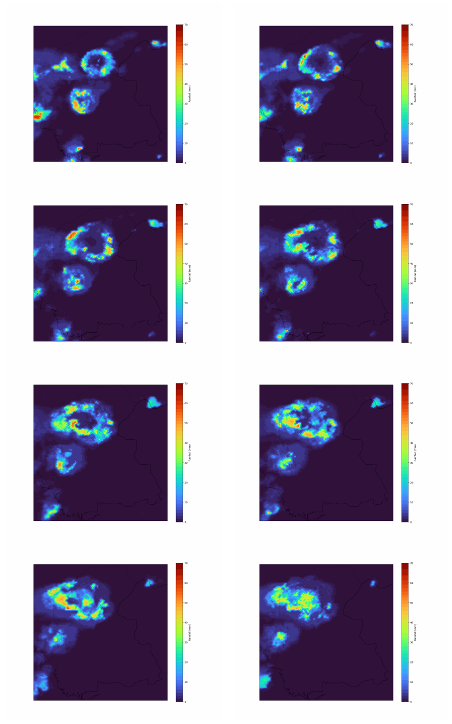

Fig. 9. **Target Sequence**. This figure shows the 8 frames of a target sequence for the test input to our model shown in figure 10 and can be compared to the prediction in figure 11

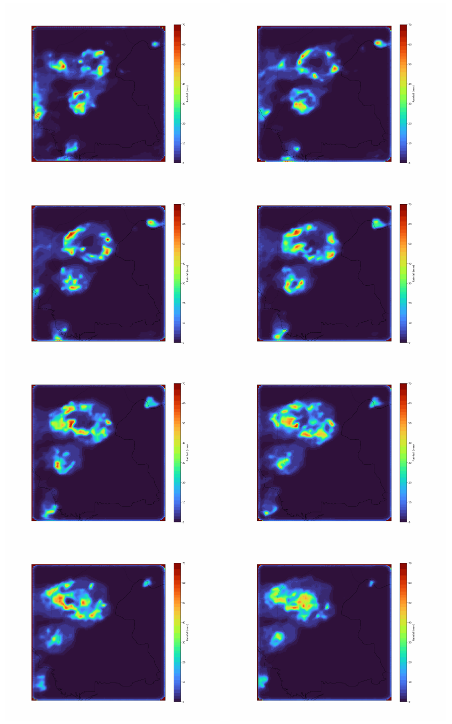

Fig. 11. **Predicted Sequence**. This figure shows the 8 frames of a predicted sequence generated by our model for the test input shown in figure 10 and can be compared to the target in figure 9

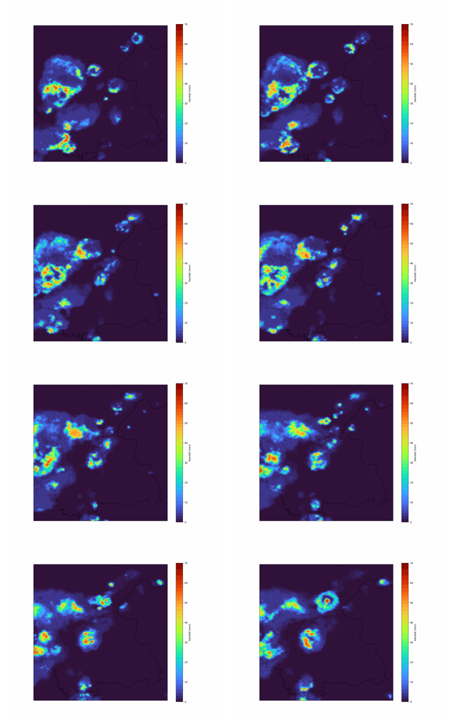

Fig. 10. **Input Sequence**. This figure shows the 8 frames of a input sequence to our model. The corresponding target and predicted sequence are shown in figure 9 and figure 11

CONCLUSION

In this paper we presented an evaluation of deep learning-based precipitation nowcasting methods, with a focus on ConvLSTM and TrajGRU architectures, using satellite data for the Sindh region of Pakistan. Addressing the limitations of conventional radar-based nowcasting methods, our study demonstrated the feasibility of leveraging satellite data for short-term weather prediction, an approach that is particularly beneficial for regions with limited meteorological infrastructure. Moreover, our work is the first (to the best of our knowledge) that has experimented and created model that operates at spatial scale of 4 x 4 km and a temporal resolution of 15-minutes.

Our experiments revealed that the ConvLSTM model performed better in terms of accuracy and visual representation, achieving an RMSE of 3.69 mm/hr for a 2-hour lead time and 8.06 mm/hr for a 4-hour lead time. The use of a combined SSIM and MSE loss proved effective in balancing visual fidelity with intensity accuracy, especially for capturing high-intensity precipitation regions.

Future work should explore larger datasets to train models on longer sequences effectively, experiment with more complex architectures, and investigate transfer learning approaches to adapt models trained on radar and satellite data in well-instrumented regions to less-equipped areas. Incorporating state-of-the-art methods like vision transformers and spatiotemporal fusion networks could further enhance the performance and scalability of satellite-based precipitation nowcasting.

REFERENCES

[1] Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., & Chopra, S. (2014). Video (language) modeling: a baseline for generative models of natural videos. ArXiv, abs/1412.6604.

[2] Sutskever, I., Vinyals, O., & Le, Q.V. (2014). Sequence to Sequence Learning with Neural Networks. ArXiv, abs/1409.3215..

[3] Srivastava, N., Mansimov, E., & Salakhutdinov, R. (2015). Unsupervised Learning of Video Representations using LSTMs. ArXiv, abs/1502.04681.

[4] Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W., & Woo, W. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. Neural Information Processing Systems.

[5] Shi, X., Gao, Z., Lausen, L., Wang, H., Yeung, D.Y., Wong, W., & Woo, W. (2017). Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model. Neural Information Processing Systems.

[6] Chen, H., Chandrasekar, V., Cifelli, R., & Xie, P. (2020). A Machine Learning System for Precipitation Estimation Using Satellite and Ground Radar Network Observations. IEEE Transactions on Geoscience and Remote Sensing, 58, 982-994.

[7] Kumar, A., Islam, T., Sekimoto, Y., Mattmann, C., & Wilson, B. (2020). Convcast: An embedded convolutional LSTM based architecture for precipitation nowcasting using satellite data. PLoS ONE, 15.

[8] Han, L.; Li, Z.; Ye, J.; Sun, Y.; Hou, X. Integrated nowcasting of convective precipitation with Transformer-based models using multi-source data. J. Hydrol. 2024, 623, 129867.

[9] Ravuri, S.; Lenc, K.; Willson, M.; Kangin, D.; Lam, R.; Mirowski, P.; Fitzsimons, M.; Athanassiadou, M.; Kashem, S.; Madge, S.; et al. Precipitation nowcasting using transformer-based generative models and transfer learning for improved disaster preparedness. Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. *2024, 382, 20230126.

[10] Wang, Y.; Zhang, J.; Zhu, L.; Fang, Y.; Sun, W.; Zhu, Y.; Liu, H. Spatiotemporal Feature Fusion Transformer for Precipitation Nowcasting via Feature Crossing. Remote Sens. 2024, 16, 2191