Satellite to Radar: Sequence to Sequence Learning for precipitation nowcasting

MARK BRUDERER, University of Twente, The Netherlands

Fig. 1. A supercell thunderstorm at twilight in SW Oklahoma.

ABSTRACT

The forecasting of rain is a complex problem with centuries of scientific work. The implications of weather for individuals and companies continue to be important. Machine Learning approaches have been shown to outperform state of the art physics based models of weather for short term predictions. We introduce a new type of model **Sat2Rad**. Our model takes as input multi-spectral satellite data and outputs radar reflectivity at a set of time-steps ranging from 15 to 180 minutes in the future. This model is novel for precipitation nowcasting because it uses several satellite spectral bands instead of using radar data as input.

Additional Key Words and Phrases: Machine Learning, Sequence to Sequence, Radar, Satellite, Storms, Forecasting

1 INTRODUCTION

Precipitation forecasting is essential to reduce the risk of life threatening situations. Different types of rainfall ranging from mist to heavy rain have a major impact for different societal sectors including agriculture, aviation, outdoor events, and the energy industry. By having timely and accurate predictions of rainfall which in turn indicate the potential for destructive storms we can prevent injuries, assist companies in predicting energy production and use resources efficiently.

A particularly strong threat is posed by rain storms and thunderstorms. Storms are one of the most destructive weather events in nature, capable of destroying human structures and even lead to loss of life [12]. Predicting storms is crucial and presents it's own set of challenges.

TScIT 39, July 8, 2023, Enschede, The Netherlands

At present meteorologists are able to successfully predict many instances of precipitation. Techniques that are used in practice range from manual analysis of current weather data (e.g radar or satellite images) to complex physics based simulations of our atmosphere with Numerical Weather Prediction (NWP) models. Various traditional precipitation nowcasting methods are based on *optical flow*. Optical flow functions in two steps, first cumulonimbus clouds are identified, and then their movement is tracked to predict the location of precipitation. Thus in this the case, the *cell-lifecycle* [15] is not taken into account [13].

Machine Learning (ML) approaches have also been developed to predict precipitation. An improvement of machine learning models over NWP models is that they are much faster to produce predictions, thus ML models are more suitable for real-time or near-real-time predictions, such as required in disaster response and energy management. According to the universal approximation theorem [6], deep neural networks have the property of being able to approximate any function provided they have the correct weights, thus it is suggested that machine learning models can incorporate sources of predictability beyond optical flow such as the cell-lifecycle among others [13].

Thus far most machine learning approaches for precipitation nowcasting have focused on predicting a next frame in a time series of radar reflectivity data [2, 16, 17]. However taking this approach may eliminate the possibility of learning the cell-lifecycle, due to the fact that the model only sees precipitation itself but not the cloud that is causing the precipitation.

We propose to use multi-spectral satellite data to learn spatio-temporal mappings between sequences of satellite data and precipitation data in the near future. If this is successful *cumulonimbus* clouds could be predicted from when they are mere *cumulus* clouds and prior. An additional advantage is that contrary to radar data, satellite data is readily available over oceans and remote communities which allows for the prediction of precipitation over these regions.

1.1 Problem Formulation

We consider precipitation nowcasting as a self-supervised problem. In self-supervised tasks, explicit labels are not provided, but rather we can derive labels from the raw data itself, which is often the case with time-series data. Consequently, we can utilize established techniques in supervised learning to address our research problem. The prediction of labels can be accomplished through two approaches: predicting discrete classes that correspond to different rain intensity intervals, or conducting pixel-level regression to learn the precise values of precipitation.
Predicting the labels can be done in two ways, either predicting discrete classes mapping to rain intensity intervals, or by performing pixel level regression to learn the exact values of precipitation.

1.1.1 Regression Formulation. Consider a dataset {X, Y} consisting of pairs of input-output sequences indexed by $i \in \mathbb{N}$, Let

$$X = \{x^{(i)} \in \mathbb{R}^{t\times h\times w\times c}\}\forall i$$

where $x^{(i)}$ is a tensor of dimension $t \times h \times w \times c$ representing the sequence of satellite images at position i, having t time-steps, h height, w width and c channels. We decided to conduct our experiments using $t = 5$, allowing for 1 hour and 15 minutes of temporal data, $h = 256$, $w = 256$, and $c = 12$. The set of output sequences is denoted as

$$Y = \{y^{(i)} \in \mathbb{R}^{w\times h}\}\forall i$$

where $y^{(i)}$ represents the i^{th} $h \times w$ dimensional tensor with each pixel being a real number collected by the radar reflectivity reading. Here the width and height are the same as in the input sequence: $w = 256$ and $h = 256$. The problem is formulated as finding a probability mass function f(x). This function must minimize a chosen distance function $\mathcal{D}$ as follows: Let $\hat{Y} = \{p(x^{(i)})\}\forall i$, representing the predicted outputs for each sequence of satellite images and identical in dimensions to $y^{(i)}$. find f(x) such that $\mathcal{D}(\hat{Y}, Y)$ is minimized.

1.1.2 Classification Formulation. Consider a dataset {X, Y} consisting of pairs of input-output sequences indexed by $i \in \mathbb{N}$, Let

$$X = \{x^{(i)} \in \mathbb{R}^{t\times h\times w\times c}\}\forall i$$

where $x^{(i)}$ is a tensor of dimension $t \times h \times w \times c$ representing the sequence of satellite images at position i, having t time-steps, h height, w width and c channels. The set of output sequences is denoted as

$$Y = \{y^{(i)} \in \mathbb{N}^{w\times h}\}\forall i$$

where $y^{(i)}$ represents the i^{th} h $\times w$ dimensional tensor containing discrete integers mapping to rainfall intensity classes. The problem is formulated as finding a probability mass function p(x). This function must minimize the Cross Entropy Loss expressed as follows:

$$E = \frac{1}{h+w}\sum_{i=1}^{h}\sum_{j=1}^{w} t_{ij} log(p_{ij})$$

Let $\hat{Y} = \{p(x^{(i)})\}\forall i$, representing the predicted outputs for each sequence of satellite images and identical in dimensions to $y^{(i)}$. find p(x) such that $E(\hat{Y}, Y)$ is minimized.

1.2 Research Question

RQ: *How can a deep learning model be trained to predict radar data with multi-spectral satellite data ?*

This research question will be answered by looking at the following sub research questions:

(1) *what is necessary to build a well performing model ?*
(2) *what methods can be used to create this model ?*
(3) *what is the effectiveness of the trained model ?*
(4) *what are the conditions under which the model can and cannot be used ?*

2 CONTRIBUTION

In this research we focus on testing the hypothesis if a model trained on satellite sequences is capable of forecasting precipitation. This is unclear as of now because most other studies rely on only radar or a combination of satellite and radar. In the precipitation nowcasting field of research, there is a symbiosis between the task of video-to-video prediction and precipitation forecasting, models created for one are used in the other and vice-versa. It is unclear whether these techniques can be successfully applied when using input and output data from different domains. We compare three different model architectures, UNet, ConvLSTM + Attention and ConvLSTM. All of these techniques are analyzed in both pixel-classification and pixel-regression implementations.

3 RELATED WORKS

In this section we will discuss the existing work in precipitation nowcasting via machine learning. This section is structured by the type of input data used in the approaches, first we list radar based learning and then we discuss satellite based approaches.

3.1 Radar Based Nowcasting

Recurrent neural networks (RNNs) have been created to learn temporal relationships in data, therefore they are a natural candidate to the task of learning spatio-temporal patterns of weather. The LSTM architecture was developed by Hochreiter and Schmidhuber [11], to solve the problem of vanishing and exploding gradients in RNNs and is widely used. Taking LSTM as a base and adapting the weights to kernels, ConvLSTM [16] was introduced for the task of precipitation nowcasting. Multiple layers of ConvLSTM are used in this paper to obtain a sequence to sequence architecture. A further improvement of ConvLSTM is trajGRU which was proposed by Shi et al. [17] to be able to learn the *location-variant* structure for recurrent connections.

Pure convolutional neural networks have also been used to predict precipitation. As demonstrated by Bai et al. and Gering et al. [3, 7] convolutional neural architectures can outperform recurrent neural networks for a variety of sequence modelling tasks. This is the reason why many works on precipitation nowcasting have opted for pure convolutional networks [1, 2].

Due to machine learning models attempting to minimize loss, *blurry predictions* can be produced by models. This can be alleviated by using generative models which sample from the possible futures and do not seek to provide a best average fit. Generative Adversarial Networks have been successfully applied to the task of precipitation nowcasting [14].

3.2 Satellite Based Nowcasting

In their study Chen et al. built a [5] MLP to forecast radar data from satellite data. The researchers used a combination of low earth orbit satellite passive microwave and infrared channels from two different satellites. Their model is developed to predict up to 1.5 hours in the future by recursive predictions of the model.

A study that does not predict precipitation but uses lightning as a marker for extreme precipitation was performed by Brodehl et al. [4] this study uses a convolutional network to predict lightning events, and contributes the important observation that both the visual and infrared channels are significant in differing ways to predict lightning.

The approach taken by the researchers of MetNet [18] is to combine a convolutional block for spatial downsampling, then a ConvLSTM block for temporal encoding and finally a Axial attention block [19]. MetNet is able to perform more accurate forecasts than NWP models for up to 8 hours. In this study the input data that is used is both satellite and radar data as well as the elevation, time of the year and latitude and longitude values.

4 BACKGROUND

In this section, we will explore the machine learning techniques utilized in this study. Initially, we will delve into convolutional neural networks (CNNs) and their advantages compared to multilayer perceptrons (MLPs). Following the introduction of CNNs, we will focus U-Net a widely used fully convolutional network architecture. Additionally, we will examine LSTM networks and their extension to ConvLSTM that incorporates convolutions.

4.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) were initially introduced by Yann LeCun in 1998 and applied to the challenge of handwritten digit classification [10]. However, the breakthrough for CNNs came later with the remarkable success achieved by Krizhevsky et al. in the ImageNet paper of 2012. This pivotal work revolutionized the field of image classification by significantly advancing the state of the art on the ImageNet dataset [8].

CNNs are a class of deep learning models strongly capable of solving computer vision tasks. Unlike fully connected neural networks, which treat input data as a one dimensional vector, CNNs are designed to process higher dimensional data such as images. This distinction enables CNNs to exploit more spatial relationships and patterns in visual data as opposed to a flattened vector where these patters are not recoverable.

Additionally Convolutional neural networks reduce the amount of parameters that are needed for each layer. This reduction is caused by parameter sharing, in a traditional multi-layer perceptron

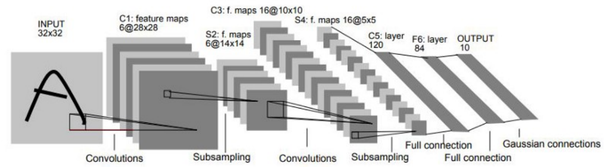

Fig. 2. Architecture of LeNet-5: used for digit recognition and introduced by Yan LeCun [10]

weights exist for each connection while in CNNs a set of kernels are applied to the input. The kernels used in CNNs are small and repeatedly applied across the input. This reduces the amount of parameters the network has.

4.2 U-Net

Semantic segmentation, the task of assigning a class label to each pixel in an image, is key to understand an image from a computer vision perspective. This is the starting point for U-Net which was developed by Ronneberger et al. [9] to segment images from microscopes in biomedical applications. U-Net is a fully convolutional architecture that provides accurate and detailed pixel-level predictions. The U-Net architecture is specifically designed to capture both local and global context information. The U-Net architecture gets its name from its U-shaped design (Figure 3), which consists of an encoder path and a decoder path. The encoder path resembles a traditional CNN and serves to capture spatial information and learn feature representations at various scales. It typically comprises multiple convolutional and pooling layers, where each convolutional layer extracts increasingly abstract features by convolving with learnable filters and applying non-linear activation functions. The decoder path, on the other hand, aims to recover the spatial information lost during the pooling and down-sampling operations of the encoder. It employs a series of upsampling and transposed convolutional layers to gradually increase the spatial resolution and reconstruct the detailed predictions. The skip connections between the corresponding encoder and decoder layers help preserve fine-grained details and enable the fusion of local and global context information, facilitating more accurate segmentation. These skip connections bridge the gap between low-level and high-level features, allowing the network to leverage both local and global context for precise pixel-wise predictions. One of the key advantages of U-Net is its ability to capture contextual information effectively. By using skip connections, the network can combine low-level and high-level features, enabling the model to refine predictions by incorporating local details and global context simultaneously. This property is especially beneficial for tasks like semantic segmentation, where precise boundary delineation and accurate classification of object categories are essential.

4.3 Convolutional LSTM

4.4 Attention

Designed for machine translation Bahadanau et al. can be applied to sequence to sequence encoder decoder models. how to pay the right amount of attention to the input sequence.

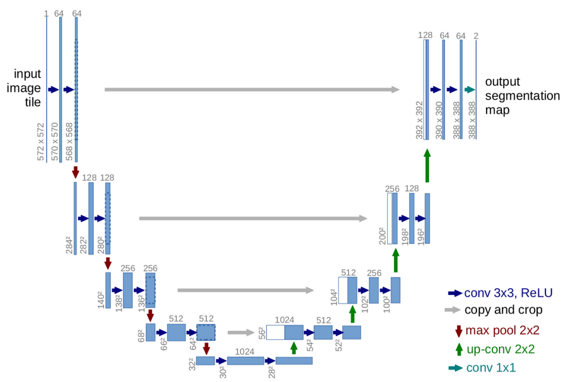

Fig. 3. U-Net

5 METHODOLOGY

In this section, we will provide detailed explanations of the methods, techniques, and procedures employed in our experiments. We will begin by discussing the data preprocessing steps, followed by an explanation of the training process. Finally, we will describe how we evaluated the performance of our models.

5.1 Engineering For Machine Learning

In order to streamline our experimental process and minimize effort, we focused on developing the training and preprocessing code in a way that allows for easy experimentation. We also utilized tools that enhance efficiency and reproducibility in the research process, reducing time-consuming tasks.

We adopted the `zenml` framework to structure our training and preprocessing pipelines. This framework brings several advantages to our workflow. One notable benefit is the promotion of modularity in the design of our pipelines. Each pipeline consists of individual steps, which enhances the code's modularity. By defining a series of functions with clear inputs and outputs, we ensure that each step can be easily understood and modified as needed. Moreover, the `zenml` web application offers a convenient way to monitor the status of our pipelines. This feature provides transparency and improves comprehension by providing insights into the outputs generated at each step.

For experiment tracking we used the `MLFlow` framwork. This makes it easy to track all metrics over all experiments. We can even track and visualize metrics during training of the models. With `MLFlow` it is also possible to compare different parameters that were used during training to experimentally find the best combinations. In `MLFlow` we save each finished model as an artifact which can be directly served as a server endpoint to start providing end users access to our models.

We used the pytorch lightning library as well. Using patterns such as the `DataModule` and LightningModule to accelerate the research process. The **`class LightningModule`** can be used by a configurable **`class Trainer`** to create a training loop, this ensures that we do not have to manually handle backpropagation or updating parameters. Furthermore pytorch ligthning handles switching from training devices automatically from for example `cpu` and `cuda` which solves a lot of overhead in a programmer trying to remember which device each Tensor is on.

We made use of the `typed-settings` library to allow cleanly structuring and validating settings for models. This ensures that to train a new version of a model in most cases only adjustments to the configuration file needs to be done. `typed-settings` supports passing settings through toml confiuration files (See listing 1), environment variables and command line options.

5.2 Data Preprocessing

For the preprocessing of data we created a pipeline which distinguishes between satellite images and radar images to preprocess each following their own needs (See figure 4).

The pipeline begins by obtaining all necessary files, from a remote storage bucket. This is done by the **`class BucketService()`** which uses the `boto3` library to interface with the bucket and downloads files in the required date ranges. In total we downloaded `3332` satellite files and `103348` radar files, this covers all data from March 1st 2023 until April 5th 2023. This requires `387.6` Gigabytes of storage for only the preprocessed data.

In the case of satellite data, the obtained files are in compressed in zip files. The pipeline handles the extraction of these files deleting any files which are not needed along the way to ease the storage requirements. Then we *reproject* the satellite images using the `satpy` package. This downsampling is done by a combination of cropping and interpolation via a nearest-neihbor alorithm.

By reprojecting we reduce the dimensions of each satellite image to `256 x 256` pixels from it's original dimensions of (3712, 3712). Via the reprojection we also obtain only the geographical area of interest, specified by the coordinates for the lower corner `(50°0'0"N 0°0'0"E)` and the upper corner `(55°0'0"N 10°0'0"E)` of the region, this gives us the area centered on the netherlands with other bordering countries see figure 8. Additionally the projection is matched between the satellite image and radar image. After reprojecting we perform a *statistics* step where we aggregate the dataset by finding the minimum and maximum values for each channel. The statistics are necessary for the next step which is normalization. During the normalization step we perform the *Min-Max* Normalization (equation 1).

$$x_{normalized} = \frac{x - min_{\bar{x}}}{max_{\bar{x}} - min_{\bar{x}}} \tag{1}$$

After finalizing the normalization step we sample an image from the dataset which is visualized to check for errors in the pipeline.

The radar pipeline begins with the downloading of radar files in the form of `h5` files. The collected files are each converted to decibels relative to Z (dBZ) from their previous unit.

$$dBZ(x) = x \cdot 0.5 - 32 \tag{2}$$

The preprocessing pipeline then splits into two, one step will normalize values between 0 and 1 using *Min-Max* normalization and the other step will use levels of *dBZ* to create discrete ranges

```toml
[model]
name = "SAT2RAD_UNET"
classes=8

[model.input_size]
height = 256
width = 256
channels = 12
sequence_length = 8

[model.output_size]
height = 256
width = 256
channels = 1
sequence_length = 1

[model.unet]
kernel_size = [3, 3]
layers = 3
filters = 64

[model.training]
max_epochs = 100
class_weights = [
            0.01081153,
            0.13732371,
            0.13895907,
            0.1416087,
            0.14272867,
            0.14285409,
            0.14285709,
            0.14285714,
        ]
metrics = [
    'acc',
    'precision',
    'recall',
    'exact',
    'f1',
    'jaccard'
]

[mlflow]
experiment_name = "sat2rad_unet"
experiment_tracker = "Infoplaza MLFlow"


[visualize]
output_dir = '../../../../../logs/'
```

Listing 1. toml configuration file for U-Net Model.

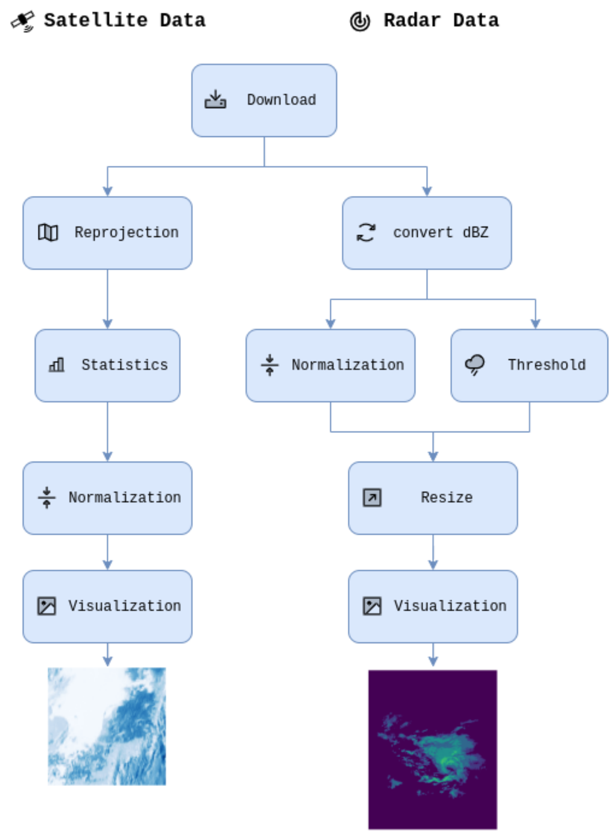

Fig. 4. Data Preprocessing Pipeline: Satellite Data and Radar data preprocessed separately

of precipitation to use as classes during training. Finally the current images in the pipeline are resized using the nearest neighbor algorithm, to avoid changing the values with bilinear interpolation. Next identically to the satellite pipeline we sample and visualize a radar image for verification purposes.

5.3 Model Training

In order to build and train our models we worked with pytorch. We used pytorch lightning to provide a higher level interface to speed up the research process. To track our experiments we connected pytorch ligthning to mlflow, using MlFlow we can log metrics, artifacts produced during training while keeping track of the used parameters during that training run. To create pipelines for preprocessing and training, we used the `zenml` library.

We created 4 different datasets for our research. These datasets result from the combination of sliding and sequential datasets with class and regression datasets. We implemented these datasets as subclasses of pytorch vision datasets. We also created time based utility functions to align the start and ending times of satellite data and radar data, such that when data is split based on training, validation and testing percentages, data is still in the same temporal range. Keeping in mind that the radar data has a different resolution and is received at a different strides.

5.4 Performance Evaluation

Different metrics were used for the regression and the classification.

Classification evaluation metrics:

(1) Precision
(2) Accuracy
(3) Recall
(4) Exact Match
(5) F1 Score
(6) Jaccard Index

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F1Score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{6}$$

$$JaccardIndex = \frac{TP}{TP + FP + FN} \tag{7}$$

Regression Evaluation Metrics

$$MSE = \frac{\sum(\hat{y}_i - y_i)^2}{n} \tag{8}$$

$$MAE = \frac{\sum|\hat{y}_i - y_i|}{n} \tag{9}$$

$$RMSE = \sqrt{\frac{\sum(\hat{y}_i - y_i)^2}{n}} \tag{10}$$

Additional metrics come from the study [17], where the authors created *B-MSE* and *B-MAE* due to the fact that the frequencies of different rainfall levels are imbalanced. Models trained with conventional *MSE* or *MAE* are biased towards predicting low amounts of precipitation. This can be solved with a loss function that weights the higher precipitation more strongly. The weighting function w can be found in the study as well.

$$BMSE = \frac{\sum w(\hat{y}_i) \cdot (\hat{y}_i - y_i)^2}{n} \tag{11}$$

$$BMAE = \frac{\sum w(\hat{y}_i) \cdot (\hat{y}_i - y_i)^2}{n} \tag{12}$$

6 RESULTS

7 CONCLUSIONS

At this point in the research we have created an introduction for the research. We have also performed a literature review of available work in the field of precipitation nowcasting. We have written the methods that we will use to perform the research and listed expected results. And we have visualized some of the data from our dataset.

ACKNOWLEDGMENTS

I would like to thank my supervisor from Elena Mocanu, for her help. Furthermore I would like to thank I would like to thank Dina Lazorkeno for proofreading my thesis.

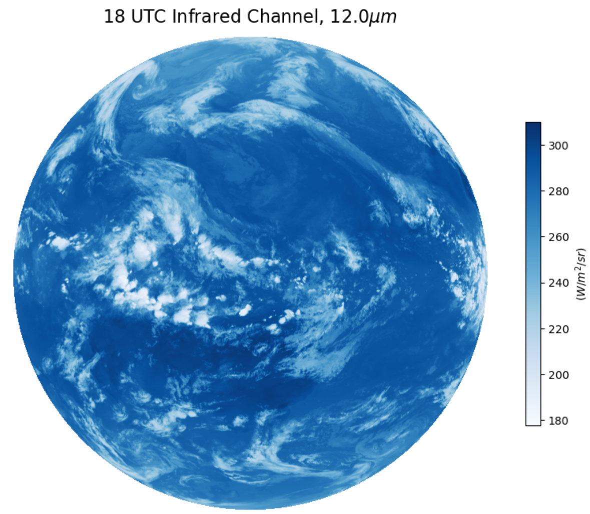

Fig. 5. Satellite Image: Infrared Channel 18UTC 12.0μm

REFERENCES

[1] Shreya Agrawal, Luke Barrington, Carla Bromberg, John Burge, Cenk Gazen, and Jason Hickey. 2019. Machine Learning for Precipitation Nowcasting from Radar Images. arXiv:1912.12132 [cs.CV]

[2] G. Ayzel, T. Scheffer, and M. Heistermann. 2020. RainNet v1.0: a convolutional neural network for radar-based precipitation nowcasting. *Geoscientific Model Development* 13, 6 (2020), 2631–2644. https://doi.org/10.5194/gmd-13-2631-2020

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. arXiv:1803.01271 [cs.LG]

[4] Sebastian Brodehl, Richard Müller, Elmar Schömer, Peter Spichtinger, and Michael Wand. 2022. End-to-End Prediction of Lightning Events from Geostationary Satellite Images. *Remote Sensing* 14, 15 (2022). https://doi.org/10.3390/rs14153760

[5] Haonan Chen, V. Chandrasekar, Robert Cifelli, and Pingping Xie. 2019. A Machine Learning System for Precipitation Estimation Using Satellite and Ground Radar Network Observations. *IEEE Transactions on Geoscience and Remote Sensing* PP (10 2019), 1–13. https://doi.org/10.1109/TGRS.2019.2942280

[6] George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems* 2, 4 (12 1989), 303–314. https://doi.org/10.1007/bf02551274

[7] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. arXiv:1705.03122 [cs.CL]

[8] family=Krizhevsky given i=A, given=Alex, family=Sutskever given i=I, given=Ilya, and family=Hinton given i=GE, given=Geoffrey E. [n. d.]. ImageNet classification with deep convolutional neural networks. 60, 6 ([n. d.]), 84–90. https://doi.org/10.1145/3065386

[9] family=Ronneberger given i=O, given=Olaf, family=Fischer given i=P, given=Philipp, and family=Brox given i=T, given=Thomas. [n. d.]. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* Springer Science+Business Media. 234–241 pages. https://doi.org/10.1007/978-3-319-24574-4_28

[10] family=LeCun given i=Y, given=Yann, family=Bottou given i=L, given=Léon, family=Bengio given i=Y, given=Yoshua, and family=Haffner given i=P, given=Patrick. [n. d.]. Gradient-based learning applied to document recognition. 86, 11 ([n. d.]), 2278–2324. https://doi.org/10.1109/5.726791

[11] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (12 1997), 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735

[12] NOAA National Severe Storms Laboratory. [n. d.]. Thunderstorm Basics. https://www.nssl.noaa.gov/education/svrwx101/thunderstorms/

[13] Rachel Prudden, Samantha Adams, Dmitry Kangin, Niall Robinson, Suman Ravuri, Shakir Mohamed, and Alberto Arribas. 2020. A review of radar-based nowcasting of precipitation and applicable machine learning techniques. arXiv:2005.04988 [physics.ao-ph]

[14] Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, Rachel Prudden, Amol Mandhane, Aidan Clark, Andrew Brock, Karen Simonyan, Raia Hadsell, Niall Robinson, Ellen Clancy, Alberto Arribas, and Shakir Mohamed.

Table 1. Metrics on Test Set for variants of classification models trained on 50 epochs.

Models	Accuracy	Precision	Recall	F1Score	Exact Match	Jaccard Index
U-Net	0.9199	0.9199	0.9199	0.9199	0.0000	0.1150
ConvLSTM	0.9999	0.9999	0.9999	0.9999	0.5385	0.1249
ConvLSTM + Attention						

2021. Skilful precipitation nowcasting using deep generative models of radar. *Nature* 597, 7878 (sep 2021), 672–677. https://doi.org/10.1038/s41586-021-03854-z
[15] NOAA's National Weather Service. [n. d.]. NWS JetStream - Life Cycle of a Thunderstorm. https://www.weather.gov/jetstream/life
[16] Xingjian SHI, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. 2015. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf
[17] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai kin Wong, and Wang chun Woo. 2017. Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model. arXiv:1706.03458 [cs.CV]
[18] Casper Kaae Sønderby, Lasse Espeholt, Jonathan Heek, Mostafa Dehghani, Avital Oliver, Tim Salimans, Shreya Agrawal, Jason Hickey, and Nal Kalchbrenner. 2020. MetNet: A Neural Weather Model for Precipitation Forecasting. arXiv:2003.12140 [cs.LG]
[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. arXiv:1706.03762 [cs.CL]

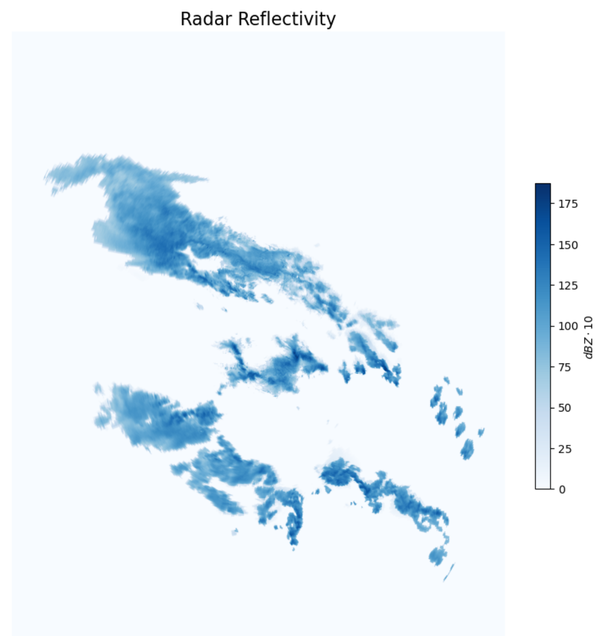

Fig. 6. Radar Reflectivity

A APPENDIX

Table 2. Reflectivity in dBZ versus Rainrate

LZ(dBZ)	R(mm/h)	R(in/h)	Intensity
5	(mm/h)	<0.01	Hardly noticeable
10	0.15	<0.01	Light mist
15	0.3	0.01	Mist
20	0.6	0.02	Very light
25	1.3	0.05	Light
30	2.7	0.10	Light to moderate
35	5.6	0.22	Moderate rain
40	11.53	0.45	Moderate rain
45	23.7	0.92	Moderate to heavy
50	48.6	1.90	Heavy
55	100	4	Very heavy/small hail
60	205	8	Extreme/moderate hail
65	421	16.6	Extreme/large hail