A deep learning baseline for spatiotemporal precipitation predictions

Gerson C. Kroiz[1, a)] and Valentine Anantharaj[2]
[1)] *University of Maryland, Baltimore County, Baltimore, MD 21250*
[2)] *Oak Ridge National Laboratory, Oak Ridge, TN 37830*

(Dated: 2 August 2021)

Accurate forecasting of weather and climate depends on physics-based numerical simulations. These computational models in most cases create reliable forecasts. But they are computationally expensive due to the billions of degrees of freedom in the physical models, and hence require some of the fastest supercomputers in the world. Recently, deep learning techniques have shown significant promise in many scientific domains. In an effort to reduce computational cost and potentially improve forecasting accuracy, researchers are now exploring various deep learning architectures to work in combination with the current numerical models. However, comparisons between deep learning models are difficult because they are trained and optimized on different datasets. In this project, we develop a benchmark to facilitate comparisons among different deep learning architectures. The benchmark includes a dataset of precipitation rates and an implementation of a deep learning architecture applied to the dataset. We use data from the Multi-RADAR/Multi-sensor System (MRMS) which contains precipitation rates in 5-minute intervals, for a 1 Mi sq. km region within Southwestern U.S. from 2001 to 2011. The dataset is split into smaller tiles to create more samples, which allows for scalability studies. With this dataset, we design a convolutional long short-term memory (convLSTM) deep-learning architecture and defined this as our baseline model. We also provide training and prediction results using the convLSTM on the benchmark dataset. Ultimately, this benchmark dataset and architecture will be released to the community. Researchers can use them to explore new deep learning architectures and can use the benchmark dataset to compare the performances of other deep learning models with our baseline convLSTM implementation.

I. INTRODUCTION

Weather forecasts have become a necessity to everyday life, whether it helps in choosing appropriate clothing for the day, or provides warning of incoming natural disasters. Current forecasts are created using complex physics-based numerical simulations. The most accurate numerical simulations use billions of degrees of freedom, and hence require the computational power of some of the fastest supercomputers in the world. While current forecast methods are extremely accurate in most cases, there are many examples of where there still needs improvement. Before forecasts are even created, there are observational and measurement-based errors from satellites and weather stations. These small inaccuracies accumulate throughout the simulations and can result in large amounts of unwanted error. Because of this, forecasting accuracy decreases as you forecast farther into the future, as errors within forecasts accumulate over time.

Due to the limitations of current forecasting techniques, researchers are exploring deep learning methods to work in combination with numerical models. Several deep learning architectures have shown promise in regards to weather forecasting. In particular, there has been moderate success with encoder-decoder convolution neural networks (CNNs) such as UNets[1,2]. These structures are designed to capture spatial correlation through down and upsampling of the image resolution. As shown in the cited examples, the UNets are also capable of capturing spatial correlation across time. Another promising deep learning architecture is the convolutional long short-term memory (convLSTM) model[3,4] that is designed to combine the advantages of LSTMs and CNNs. More specifically, LSTMs are strong for predicting patterns in a time series, and CNNs are good with predictions using images as they can track spatial patterns. One potential issue with the convLSTM structure is that they are location invariant due to the convolutional layer. This can be problematic as natural motions in weather forecasting such as rotations are generally location variant. Shi et al. developed the Trajectory Gated Recurrent Unit (TrajGRU) architecture[5], where instead of having fixed recurrent connections, the connections are dynamically determined. Their initial studies showed improvement of spatiotemporal predictions in comparison to more traditional convolutional-based networks.

While there are already several papers that discuss deep learning techniques for weather forecasting, there are little to no empirical comparisons between architectures. This is because each paper presents a deep learning architecture trained and optimized on a unique dataset. One solution to this problem is to create benchmarks, which can facilitate comparisons among different deep learning architectures. One example of this is WeatherBench, a benchmark dataset for weather forecasting[6]. The WeatherBench dataset consists of hourly data of different atmospheric variables simulated over 40 years. Similar to WeatherBench, this report proposes a benchmark dataset focusing on precipitation rates, rather than the entire atmosphere.

[a)] Electronic mail: gkroiz1@umbc.edu

The remainder of this report is designed as follows. Section II describes how the benchmark dataset is structured. Using the benchmark dataset, we define a baseline model in Section III, which allows for comparisons between different deep learning architectures. Section IV discusses the predictions produced by the baseline model. The results and overall findings of the paper are then summarized in Section V.

II. BENCHMARK DATASET

For the proposed benchmark dataset, we use the Multi-RADAR/Multi-sensor System (MRMS) dataset[7]. The MRMS dataset provides precipitation rates (mm/hr) in 5-minute intervals over the entire U.S. from 2001-2011. The benchmark dataset takes a 1024×1024 km^2 portion of the MRMS dataset located in Southwestern U.S. Figure 1 shows the location of the dataset within the U.S.

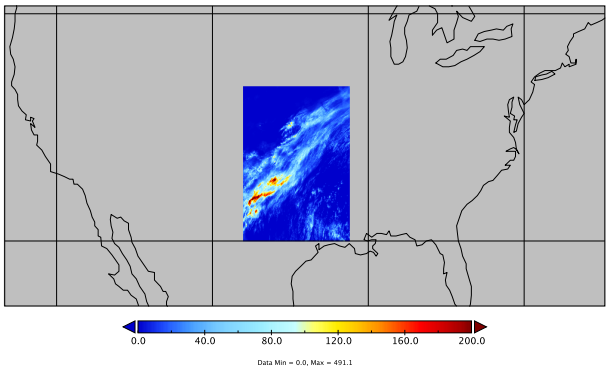

FIG. 1. Precipitation rates (mm/day) for entire 1024×1024 km^2 region on 08/14/2005

Within the dataset, the 1024×1024 km^2 portion is further divided into 256 64×64 km^2 tiles. Creating 256 subtiles provides two advantages: (1) this results in many smaller files, which can be advantageous to those with memory-bound environments, and (2) based on how training and testing datasets are created for machine learning, the number of potential samples increases by a multiple of 256.

One complication with creating 256 tiles within the 1024×1024 km^2 region is that the amount of precipitation each tile receives is not equal. For this dataset, this is observable when comparing tiles closer to the coastline in Lousiana versus. To account for this, we plotted the probability density function as a line histogram over precipitation rate (mm/hr) as shown in Figure 2.

When creating training and testing data using the benchmark dataset, Figure 2 can be used to identify tiles that most accurately represent the precipitation amounts and rates of the entire 1024×1024 km^2 region.

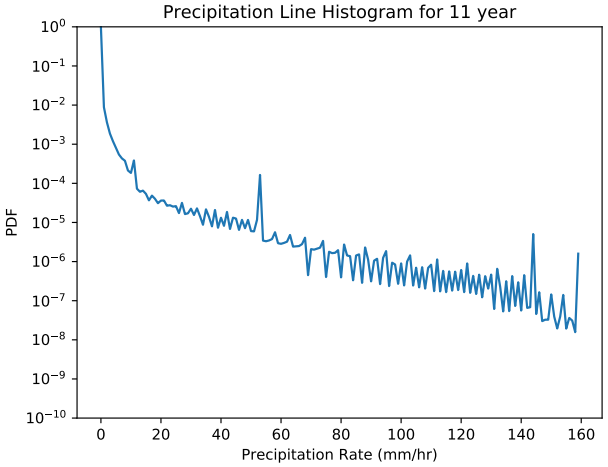

FIG. 2. Line histogram of precipitation rate (mm/hr) for entire benchmark dataset (2001-2011).

III. BASELINE MODEL

As explained in Section I, there is a need for benchmark datasets for researchers to compare different deep learning model architecture's forecasting and nowcasting performance. This section provides a baseline model that can be comparable with future models optimized on the benchmark dataset.

A. Deep Learning Architecture

Based on its success with precipitation predictions, we design the baseline model using a convLSTM architecture. Additionally, there is strong code-based documentation for the proposed convLSTM architecture[4], which allows for a direct comparison when developing the model. As explained in the introduction, convLSTMs are designed for spatiotemporal data as they are capable of learning long-term dependencies with respect to location[4]. ConvLSTM's are structured as recursive blocks.

Figure 3 displays one of the convLSTM recursive blocks. Each block takes in three inputs, the cell state (C), the hidden state (H), and the input state (X). C carries important information between each of the blocks, and is the core of how the model learns. Within each block, there are three gates, the forget gate (F), the input gate (I), and the output gate (O). These control what information from H and X should added to C. More specifically, F controls what information should be removed from the cell state, I controls what information in the cell state should be updated, and O controls the values within H for the next convLSTM block. More formal definitions for each of the different components of each convLSTM block are shown below, where '$*$' denotes the

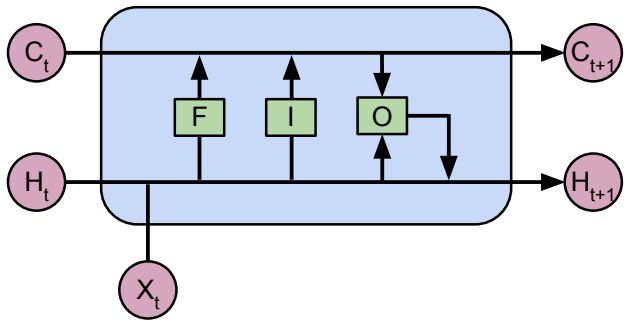

FIG. 3. Visual representation of convLSTM recursive block

convolution operator, '$\circ$' denotes the Hadamard product, and $t = 1, \ldots, n$, where n is the total number of convLSTM blocks:

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + Wcf \circ \mathcal{C}_{t-1} + b_f) \\
\mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o) \\
\mathcal{H}_t &= o_t \circ \tanh(\mathcal{C}_t)
\end{aligned}
$$

While a convLSTM can use an arbitrary number of convLSTM blocks, our baseline model includes 3. Additionally, the baseline model includes a 10% dropout rate between each convLSTM block to prevent overfitting as well as padding='same' such that the input dimensions for each convLSTM block match the output dimensions. Having a nonzero padding layer is beneficial as padding can be interpreted as the *state of the outside world*, providing additional information of how clouds and other causation of rainfall travel through a region[4]. Besides the first and last layers, each layer uses 32 filters and a kernel size of 3. Within the architecture, the last recursive block, rather than outputting a sequence, returns a single time instance. We then apply a final convolution to this time instance to reduce the number of filters to 1, so the output is comparable to the ground truth.

As a whole, the baseline model typically takes in 5D tensors (batch size, time, rows, cols, channels) or the data of a region over time and uses the information to predict the region's data for the next time frames. The output is another 5D tensor (batch size, time, rows, cols, channels) where time = 1 as the model only predicts the next time frame. In both the input and output tensors, the number of channels equals 1. However, within the model, this is increased to 32.

B. Data Generation using Benchmark Dataset

For the baseline model, we need to define a training, validation, and testing dataset. Using the benchmark dataset, we defined a sample as five consecutive time frames, where the first four time frames are considered as the data fed into the model (x_data), and the last time frame is used to compare the output of the model (y_data). Any sample that was missing any of the five consecutive time frames or had any NaN values within the precipitation rates data was removed. Using this method of sampling results in around 20,000 samples per year for each tile. For the baseline model, the training dataset consisted of samples from 2001-2003, the validation dataset consisted of samples from 2004, and the testing dataset consisted of samples from 2005. Sampling as such results in approximately a 60%, 20%, 20% split for training, validation, and testing, respectively.

Based on the general distribution of precipitation rates shown in Figure 2, any dataset that does not use preprocessing techniques to balance samples with and without precipitation will be extremely unbalanced, where the vast majority of samples contain little to no precipitation. From a scientific standpoint, the unbalanced ratio between samples without and samples with rain makes sense for the Southwest region of the U.S. as there are many more instances where the is no rain compared to instances when there is rain.

C. Weighted Loss Function

To account for the unbalanced data, we designed a weighted mean squared error (WMSE) loss function. This WMSE function is similar to the balanced MSE and MAE functions optimized for the radar echo precipitation data used by Shi et al, where each pixel was weighted based on precipitation intensity[5]. The WMSE loss function is defined as the following:

$$
\text{WMSE} = \frac{\sum_{i=1}^{n} w_i (Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n} (w_i)}, \tag{1}
$$

where w_i is the weight, Y_i is the ground truth for the precipitation rates, and $\hat{Y}_i$ is the predicted precipitation rates for each sample $i = 1, \ldots, n$. To define the weight of each sample, we designed a piece-wise function based on Figure 4.

Figure 4 provides insight on the distribution of samples in terms of number of pixels with nonzero precipitation rates, n. The piece-wise function $f(n_i)$ is defined based on n_i in the y_label for the ith sample:

$$
f(n_i) = \begin{cases} \text{if } n_i \leq 0.83t \text{ then } w_i = 285 \\ \text{if } 0.83t < n_i \leq 0.93t \text{ then } w_i = 3 \\ \text{if } 0.93t < n_i \leq 0.975t \text{ then } w_i = 2 \\ \text{if } n_i > 0.975t \text{ then } w_i = 1 \end{cases} \tag{2}
$$

where t is the total number of pixels in each sample, and w_i is the weight. The weights are based on the major

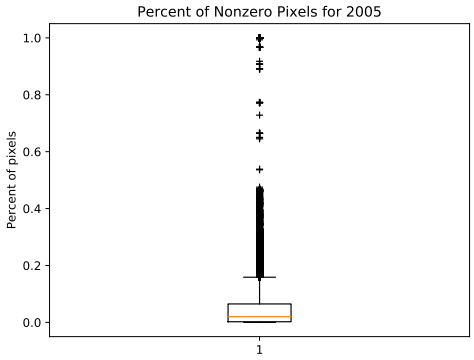

FIG. 4. Box plot of percent of pixels within the 1024×1024 km^2 region that have a nonzero precipitation rate for 2005.

points within the boxplot. We define samples that have n smaller than the median as samples with virtually no rain. These have a default weight of 1. The remainder of samples either fall the third quartile group, the fourth quartile group, or are classified as outliers. The weight of the samples that fall in these ranges are either 2, 3, or 285 respectively. These values are based on the proportion of data that falls within the respective range relative to the data with virtually no rain. Using these weights balances the influence of samples so the loss is not so heavily influenced by the large number of samples with virtually no rain.

IV. RESULTS

The model defined in Section III was implemented using Tensorflow's library and was trained in a multi-GPU environment using Tensorflow's `MirroredStrategy`, a standard distributive strategy[8]. Training and evaluation used 4 NVIDIA Tesla V100-SXM2 GPUs with 16GB high bandwidth memory. The model was trained over 50 epochs with a learning rate of 10^{-3} using the Adam optimizer.

With the trained model, we developed a direct 24-hour precipitation prediction. Figure 5 shows a visual comparison of the observed and predicted precipitation rates (mm/day) for three different amounts of precipitation: light, medium, and heavy. Tables I and II show the distribution of precipitation rates for the observed and predicted data for the three days respectively. These tables use the same categorization for the different precipitation levels as the tables used in a previous related paper[5].

For the light amount of precipitation on 08/16/2005, the maximum observed precipitation rate is 89.1 mm/day. In comparison, the maximum predicted precipitation rate is 33.2 mm/day. However, the overall average precipitation rate between the two is much closer, with an average of 15.6 mm/day and 24.2 mm/day for the observation and predicted data, respectively. From a

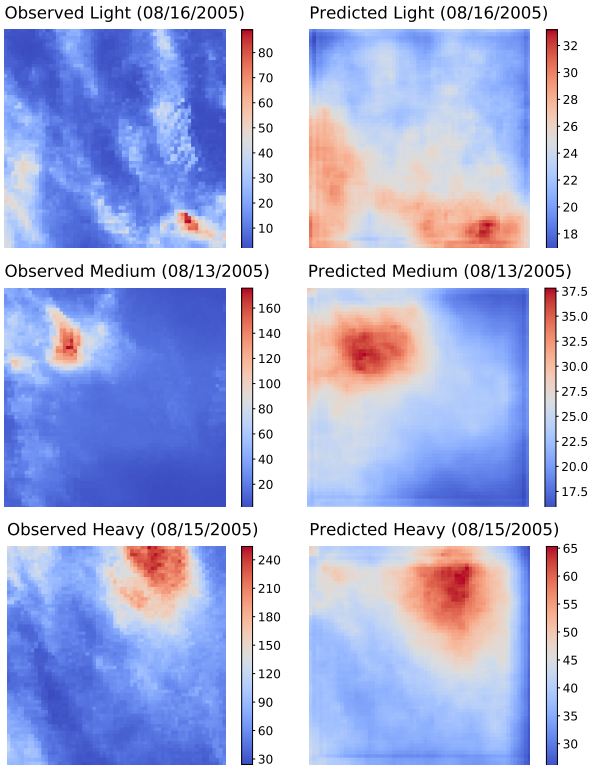

FIG. 5. Comparison of observed and predicted precipitation rates (mm/day) for light, medium, and heavy amounts of precipitation. Note the different color scale for each plot.

visual perspective, we see in Figure 5 that the observed precipitation rates are fairly low throughout, where there is some increase along the left side, and the bottom right corner is where the most precipitation is. Similarly, the predicted precipitation rates capture the general patterns both on the left side and the bottom right corner. However, these areas are visually more spread out.

For the medium amount of precipitation on 08/13/2005, we see that the maximum observed precipitation rate of 175.8 mm/day is significantly larger than the maximum predicted precipitation rate of 37.8 mm/day. On the other hand, the average precipitation rate between the observed and predicted data, 22.9 mm/day and 23.9 mm/day, respectively, is extremely close, with a 1 mm/day difference. Visually, in the observed data, the most precipitation occurs in the top left corner and is relatively sparse throughout the remainder of the tile. Similarly, the predicted data shows the most precipitation in the top left corner, capturing the general spatial location of the event but not the magnitude.

For the heavy amount of precipitation on 08/15/2005, the maximum observed precipitation rate is 254.1 mm/day, whereas the maximum predicted precipitation rate is 65.3 mm/day. When comparing the observed and predicted data for the large amount of precipitation, we see that the difference between the two maximums is the

Precip. Rate (mm/hr)	Low Proportion (%)	Medium Proportion (%)	High Proportion (%)	Precipitation Level
$0 \leq x < 5$	84.42	81.19	53.28	No / Hardly noticeable
$0.5 \leq x < 2$	9.75	6.91	20.73	Light
$2 \leq x < 5$	3.12	8.11	11.10	Light to moderate
$5 \leq x < 10$	1.34	2.34	6.56	Moderate
$10 \leq x < 30$	1.01	0.81	6.24	Moderate to heavy
$30 \leq x$	0.36	0.63	2.10	Rainstorm warning

TABLE I. Precipitation rate statistics for light, medium, and heavy amounts of precipitation in the observed data averaged based on each 5-minute interval

Precip. Rate (mm/hr)	Low Proportion (%)	Medium Proportion (%)	High Proportion (%)	Precipitation Level
$0 \leq x < 5$	0	0	0	No / Hardly noticeable
$0.5 \leq x < 2$	92.51	89.20	70.61	Light
$2 \leq x < 5$	7.36	10.49	25.86	Light to moderate
$5 \leq x < 10$	0.12	0.31	3.32	Moderate
$10 \leq x < 30$	0.01	0.01	0.21	Moderate to heavy
$30 \leq x$	0	0	0	Rainstorm warning

TABLE II. Precipitation rate statistics for light, medium, and heavy amounts of precipitation in the predicted data averaged based on each 5-minute interval

largest out of all three comparisons. This difference suggests that the model fails to capture high precipitation rates. The model's inability to predict high precipitation rates is further evident in Tables 1 and 2. For this example, the average precipitation rate between the observed and predicted data was 78.5 mm/day and 42.1mm/day, respectively. Similar to the other amounts of precipitation, the difference between the average precipitation rate is much smaller than the difference between maximum precipitation rates. In the observed visual, the most precipitation occurs in the top right corner, but there is also some precipitation in both left corners. We see the same patterns for the predicted data, where the largest amounts of precipitation are in the top right corner, followed by smaller amounts in both left corners. When comparing to the other samples, we see that the predicted data for the heavy amount of rain does a better job of capturing the size of the cloud.

As previously mentioned, Tables I and II show the distribution of precipitation rates within the tile for the observed and predicted data respectively. In the observed data, as expected, as the amount of rainfall increases, the proportion of pixels with higher amounts of precipitation rate increases. This trend is also notable within the predicted precipitation proportions. However, the distribution of precipitation rates within the observed data is not kept within the predicted data. This is because the model does not adequately capture high precipitation rates and often underestimates the rates. It is evident from the tables that the model misses both the no-rain as well as rainstorm events. This, in turn, results in a tendency to predict prevailing light rain throughout the domain during all three days.

V. CONCLUSIONS

In summary, we present a benchmark dataset for precipitation rates, described in Section II and provide a baseline deep learning architecture that utilizes the benchmark dataset, described in Section III.

Section IV shows that the model's predictions capture both spatial and temporal characteristics of the data though there are very large biases. From these examples, it appears that as the amount of rainfall increases, the difference between the maximum and average precipitation rate of the observed and predicted data increases. From a broader perspective, it appears the model struggles to capture both high as well as very light precipitation rates and generally underpredicts precipitation rate when there is a large amount of rainfall. The model completely misses no-rain and rainstorm events.

With the moderate success in the baseline model, other deep learning architectures can be optimized on this benchmark to facilitate empirical comparisons. Furthermore, due to the abundance of data in the benchmark dataset from the tiling technique, there is opportunity for scalability studies. This opens doors for comparisons not only based on prediction accuracy but also on computational efficiency with of data. Scalability studies can provide insight into how computational and cost requirements can be optimized for practical weather forecasts. Ultimately, these comparisons can help provide motivation for weather forecasting to incorporate deep learning methods.

ACKNOWLEDGMENTS

Special thanks to Junqi Yin and Aristeidia Tsaris for their technical guidance. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility, and the resources of the Compute and Data Environment for Science (CADES) at the Oak Ridge National Laboratory, both supported by the Office of Science of the U.S. Department of Energy under Contract DE-AC05-00OR22725.

REFERENCES

[1] J. Hickey, "Using machine learning to "nowcast" precipitation in high resolution," (2020).

[2] S. Samsi, C. J. Mattioli, and M. S. Veillette, "Distributed deep learning for precipitation nowcasting," (2019).

[3] N. Kalchbrenner and C. Sønderby, "A neural weather model for eight-hour precipitation forecasting," (2020).

[4] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems*, Vol. 28, edited by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Curran Associates, Inc., 2015).

[5] X. Shi, Z. Gao, L. Lausen, H. Wang, D.-Y. Yeung, W. kin Wong, and W. chun Woo, "Deep learning for precipitation nowcasting: A benchmark and a new model," in *Advances in Neural Information Processing Systems*, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017).

[6] S. Rasp, P. D. Dueben, S. Scher, J. A. Weyn, S. Mouatadid, and N. Thuerey, "Weatherbench: A benchmark data set for data-driven weather forecasting," (2020).

[7] J. Zhang and J. Gourley, "Multi-radar multi-sensor precipitation reanalysis (version 1.0)," (2018).

[8] M. A. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," (2015), software available from tensorflow.org.