Precipitation Nowcasting: Leveraging Deep Recurrent Convolutional Neural Networks

Alexander Heye*, Karthik Venkatesan†, Jericho Cain‡
Cray, Inc.
Seattle, Washington
**aheye@cray.com*
†kvenkatesa@cray.com
‡jcain@cray.com

***Abstract*—Automating very short-term precipitation forecasts can prove a significant challenge in that traditional physics-based weather models are computationally expensive; by the time the forecast is made, it may already be irrelevant. Deep Learning offers a solution to this problem, as that a computationally dense machine can train a neural network ahead of time using historical data and deploy that trained network in real-time to produce a new output within seconds or minutes. Our team intends to prove the capabilities of Deep-Learning in short-term forecasting by leveraging a model built on Convolutional Long Short-Term Memory (convLSTM) networks. By designing a 3D sequence-to-sequence convLSTM model, we hope to offer accurate precipitation forecasts at minute scale time resolution and neighborhood scale spatial resolution. Our work will be accelerated by the GPU-dense Cray® CS-Storm™ system for training and the Cray® Urika-GX™ for real-time processing of radar data.**

***Keywords*-Analytic; Deep Learning; Meteorology**

I. Introduction

High accuracy, short-term precipitation forecasts have the potential to not only keep us dry on our way to work, but also to save lives in extreme precipitation events. Heavy localized precipitation over a narrow river valley can produce massive flash floods rapidly and with insufficient warning if high resolutions predictive models are not in place. When a difference of 1-2km can be the difference between a normal rainy day and flood conditions, it is vital that a specialized system exists to inform meteorologists and decision makers.

A. Precipitation Nowcasting

Weather Nowcasting is term used to refer to very short term weather forecasts at high time and spatial resolutions.[12] Contemporary short-term weather forecasting tends to focus on making predictions with forecast lengths of 1-5 days into the future every 3-6 hours and with spatial accuracy on the order of 1-5 kilometers. Nowcasting in comparison aims to make reliable predictions up to 6 hours into the future as often as every 10 minutes and spatial accuracy on the order of 1 kilometer.

B. Deep Learning

Deep learning has recently become a popular technique for high level feature extraction from large, complex data-sets. With recent advancements in training techniques and new found architectures for new data formats, deep learning has found applications in any data-heavy field of study.

Deep learning refers to the training of deep neural networks, or a neural network with more than a single layer. These neural networks are generally trained on supervised data-sets where each input data has a corresponded known solution that we want the network to learn to predict. This learning occurs through an iterative process of parameter updates based on an error value representing how far off a prediction was to the expected value. This tends to be a very computationally demanding process, taking anywhere from hours to days to properly train a sufficiently complex network.

The primary driver of the widespread success of deep learning is the expanded capabilities to capture and process large amounts of data. A larger dataset provides a neural network to train on a larger subset of the complete set of possible input-output pairs, and thus can more readily generalize to an even larger subset of possible inputs. Managing and processing a sufficiently large dataset is a large undertaking, and thus is likely to be the primary pain point when applying deep learning to a new problem.

Because of this, we intend to explore the full workflow of a deep learning task, from data sourcing to data processing to model training and inference. Deep Learning is inherently a big data problem and therefore the deep learning approach we demonstrate will follow the pipeline from initial data extraction though a final trained neural network model.

II. Previous Work

Methods for precipitation nowcasting have traditionally been focused on techniques based on the extrapolation of radar observations. [15], [14], [13]. More recently techniques which blend information from both extrapolation of radar observations and output from NWP models have emerged. [16]

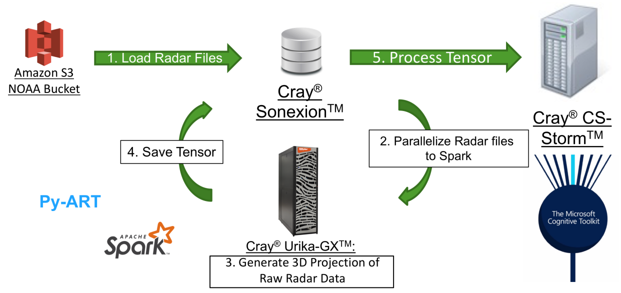


Figure 1: Deep Learning Data Pipeline

Extrapolation techniques tend to rely on linear relationship and a limited amount of local observational data. Because of this, the time to a prediction is low, but the skill of the prediction degrades rapidly from the initial state. [8]

Short term Numerical Weather Prediction (NWP) uses physics-based models relying on the same methods as for global forecasting models. Based on an initial atmospheric state extracted from weather observations, NWP makes predictions of the future states of the atmosphere. Observational weather data comes from many sources and data points are scattered inconsistently both spatially and vertically in the atmosphere. A data assimilation step is therefore necessary to combine data sources and project known data points into a gridded structure for ingestion into the physical model. Whereas global NWP models have large grid-spacing on the order of 10km and a limited time resolution of around 6-12 hours between predictions, short-term NWP models will have grid-spacing on the order of 1-5km and time resolution closer to 1 hour. In order to make a prediction in this time frame, these models are regionally focused to limit the computational time demand required by a higher resolution grid and allow a prediction within the hour.

The delay caused by the data assimilation and simulation steps limits the capacity of NWP Nowcasting systems to make very short term predictions, however the skill of predictions 1-6 hours into the future will degrade more slowly then the extrapolation techniques.

Machine Learning approaches to weather nowcasting, specifically with neural networks, have been an active research area recently. It has the potential to model non-linear relationships and shifts most of the computational burden to well before the predictions need to be made. The deep learning approach utilized in our system was originally introduced by X. Shi et al. in *Convolutional LSTM Network: A Machine Learning Approach For Precipitation Nowcasting*[1] which introduced the novel architecture of the Convolutional LSTM and provided evidence of improvement over traditional Nowcasting techniques.

III. Data Pipeline

A. Data Source

For our initial testing of this system, we have been working with radar data provided by the National Oceanic and Atmospheric Administration (NOAA). To improve public accessibility, NOAA has partnered with Amazon Web Services (AWS) to host historical NEXRAD (Next-generation Radar) level-II data in a S3 bucket, a data storage location in the AWS cloud identified by a user-defined key. Our data pipeline originates here at our data source. We download all relevant radar archive files through this service and store them on a Cray® Sonexion™ system.

B. Dataset Processing

Deep learning is inherently data intensive. Much of the time spent developing a trained neural network model is spent acquiring, labeling and processing an extensive dataset. Additionally, iterative changes to and augmentation of the dataset can be a vital part of improving training results as more information becomes available. With this in mind, the Cray® Urika-GX™ system was selected as the platform for building and managing our radar data sets before and during training.

The analytics software tools on the Cray® Urika-GX™ system allowed us to collect and process the data with ease. The raw, compressed radar files in radial format are parallelized as a Resilient Distributed Dataset (RDD) in Apache Spark. These files are extracted and transformed using python libraries such as numpy and py-art[5]. The final format of the radar data will be a saved numpy file accessible to the Cray® CS-Storm™ system for quick ingestion into the neural network training process.

C. Model Training

Training of Neural Networks on multi-dimensional data is extremely computationally expensive. Tools such as NVIDIA® CUDA™ and CuDNN™ libraries have allowed GPU computing to accelerate the training of deep neural networks, and we make full use of GPU acceleration in our training process.

The density of high end GPUs provided by the Cray® CS-Storm™ Cluster Nodes allows us to easily distribute model training and parameter tuning on each node and among multiple nodes. Each node has 8 NVIDIA® Tesla™ M40 GPUs, each with 24GB on device memory, and 2 Intel® Xeon™ host processors.

The Microsoft® Cognitive Toolkit™ (CNTK) was utilized to declare, build and train a neural network as a computational graph. By utilizing this toolkit, the work necessary to design and build the neural network was greatly simplified in that many common functions were built-in and predefined and all gradient calculations and modifications are automated.

To support such an effort along a broad range of neural network and machine learning techniques, the list of prerequisites and necessary hardware and software configurations is extensive. To aid with deployment, the toolkit and its prerequisites were containerized through docker. A container is able to utilize portions of the underlying operating system as well as all software installed and configured within its separate environment. When the toolkit is containerized, the prerequisites and environment comes with it and therefore should offer portability from system to system.

Training was performed on multiple GPUs in parallel. There are two primary methods for distributed neural network training: Data Parallel and Model Parallel. Data parallel creates a copy of the network on each device and distributes the dataset (or minibatch) among the devices for training. The gradients are then combined and applied to the master copy and the copies on each device are updated. Model parallel splits the network, generally by layers, among the devices and samples from the dataset and gradients move through each device during the forward and backward passes.

The nowcasting model will run in a data parallel setting. This choice was heavily influenced by the recurrent structure as well as the number of GPU devices available. Each convolution recurrent layer (described in the next section) is very large and stacking a large number of layers required to keep each device active would decrease flexibility as a larger network may not train to the same accuracy due to the vanishing gradient problem common in large networks.

IV. Neural Network Architecture

A. *Long Short-Term Memory Network*

To understand the complex dynamics of convective precipitation in the atmosphere, temporal relationship must be considered. The ability to consider not only the initial atmospheric state prior to the prediction, but also the sequence of events that led to that state should provide more predictive power.

Recurrent Neural Networks (RNN) are designed to retain temporal data in a hidden state and use that knowledge for each subsequent prediction. RNNs come in various forms, however the Long Short-Term Memory (LSTM) network has in recent years proven itself again and again as an exceptional tool for learning long term dependencies.[9] The LSTM is able to do this by retaining and managing sequential information in a cell state, or a numeric vector held over between time steps. The cell state is managed through gating functions that can control the influence of input vectors as well as prior cell and hidden state vectors on it while also controlling the rate at which information is lost.

The cell state is managed by the input, forget and output gates and are represented in (3) by i_t, f_t and o_t. These gates are functions of the input vector, previous hidden state. A sigmoid function is applied to confine the gating function between 0 and 1. In the extreme cases, a gated vector of all 0s would ignore all influence from the vector it is applied to and a value of 1 would do the opposite: allow full influence from the vector it is applied to.

A minor variant of the standard LSTM is to include what are called "peephole" connections as introduced in [5]. These are represented by the third piece of each sigmoid function in (3), $W_c \circ c_{t-1}$. The idea is that each gate function has access to the current cell state at some capacity.

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

B. *Convolutional LSTM*

To understand precipitation trends in a certain location, you must have a spatial understanding of where it is raining and where it is not. Learning the location of the edges of a band of precipitation and how those edges move over time is vital to extrapolating into the future.

Convolutional Neural Networks (CNN) have been widely adapted in the area of computer vision due to the distinct ability of a convolutional operation to extract spatial context of each pixel rather than just the independent value. With radar reflectivity data, the shape of a convective cell can have significant impacts on its direction, speed and intensity over time. Combining this strength with the sequential processing capabilities of the LSTM will allow a neural network to effectively make predictions from a sequence of multi-dimensional reflectivity values.

We followed the approach introduced by [1] in building a Convolutional LSTM network. This network follows much of the same structure as the peephole LSTM, however the core internal equations of the LSTM utilize convolutional operations rather than matrix multiplications. In order to keep the operations consistent, the hidden state and cell state are also represented as 3 dimensional tensors matching the size and shape of the input tensor. By expanding the dimensionality, we are likely also improving the potential representative power of the hidden and cell states.

The Convolutional LSTM is represented in (2). Here the weight matrix multiplications have been replaced by convolutional operations represented by $*$ and the input, hidden state and cell state have taken the form of 3 dimensional tensors $\mathcal{X}_t$, $\mathcal{H}_t$ and $\mathcal{C}_t$.

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f) \\
o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_{t-1} + b_o) \\
\mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \\
\mathcal{H}_t &= o_t \circ tanh(\mathcal{C}_t)
\end{aligned}
\tag{2}
$$

C. *Sequence to Sequence*

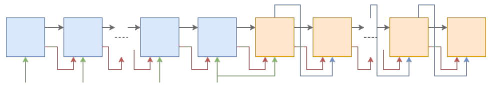


Figure 2: Encoder/Decoder Network

A simple Convolutional LSTM network is sufficient to make single predictions based on the state of the atmosphere leading up to that time, however the true goal of nowcasting is to forecast multiple time steps into the future and provide predictions of when it will be raining with a time resolution in the minutes to an hour. We want a neural network architecture that can readily provide predictions arbitrarily far into the future.

We can build on the techniques used to solve vector based sequence to sequence problem such as [2] and develop a decoder/encoder structure with more than one recurrent cell. In this approach, one recurrent cell is used exclusively to process the input sequence, the encoder. The decoder recurrent cell is applied generatively to produce a sequence of predictions. The encoder cell synthesizes the input sequence into its cell and hidden states; these tensors are concise representations of the entire input sequence. The decoder's hidden and cell states are initialized with these tensors allowing for an efficient transfer of information between recurrent cells.

The decoder input sequence is made up of either actual or predicted reflectivity tensors and a "start symbol" that primes the network to make it's initial prediction. During training, the label, or ground-truth reflectivity sequence is used such that each time step can provide valuable training information from the start and thus accelerating the training process. When evaluating the model, we switch to prior predictions which are fed back into the network at each time step after priming. The priming step involves feeding a unique image through the network that will initiate the first prediction. We ran experiments with different start symbols to evaluate the performance effect. Input data was scaled to the range of [0,1], so the easiest choices for a start symbol are tensors of all 0s and of all 1s, representing the extremes at each location. On top of this we tried using the last time step of the encoder input sequence to represent the current state of the atmosphere. This option allowed the decoder to assume a consistent input format and thus simplify the task. The potential drawback is that the network may learn to mimic the input as a faster route to a lower loss value rather than actually learning the underlying pattern. The results can be found in Table 1, and are discussed in the results section.

V. Experiments

A. *NOAA NEXRAD data*

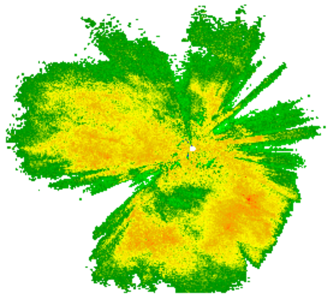


Figure 3: Example Radar Image

NOAA NEXRAD-II data is used in training and validation of our model. NEXRAD data has the benefit of being high-quality, consistent and openly accessible. Specifically, our focus is on reflectivity scores captured from the radar station that represent the amount of a radar beam that is scattered back to the radar station by precipitation and small objects in the atmosphere.

Each data point used in training and evaluation of the neural network model is a Cartesian projection of the reflectivity scores onto a 3 dimensional grid. This grid was of the shape 5x100x100 where 5 is the vertical dimension and 100x100 is the horizontal plane. The data used was limited to 100km from the radar site in the north, south, east and west directions and 12km vertically. That gave us a horizontal resolution of 2km and vertical resolution of 2.4km.

With the initial projection in place, the tensor was further processed for the sake of training efficiency. The reflectivity values were clipped to the range of [0, 80], and any location without a reading was set to 0. 80 was chosen as a max value simply because the likelihood of an event producing a reflectivity value of 80dBZ is very unlikely. The goal is to predict severe precipitation, exact predictions at this level will not make a significant affect on decisions. In order to keep weight values and gradients small, the reflectivity was projected down to the range of [0, 1] by simply divided the each data point by 80.

The full dataset focuses on a single station outside of Seattle, Washington with the NOAA station code of KATX. This location was chosen for the initial investigation due to the number of localized weather phenomena, such as rain shadows and convection zones, common in western Washington due to the unique terrain and proximity to the

Pacific Ocean. For training and evaluation, all sequences from all days of the 2015 calendar year with over 0.1 inches of rain at the KSEA observation site were selected, which came out to 76 days for the year, or about 138GB of raw data. This cutoff was used to ensure the training set was not over-saturated with near zero reflectivity on clear days. Including full days with precipitation ensures that the network will train on a small number of sequences with little to no precipitation and thus not saturate training in the other direction.

B. Tuning and Hyperparameters

Hyperparameter selection was performed the open version of Spearmint[11], which provides a Bayesian optimization mechanism for quickly finding a near optimal configuration of hyperparameter values. Spearmint was allowed to run for 100 iterations at which time the best model was selected and used. The hyperparameters key to this approach include the convolutional kernel size, the number of convolutional filters, learning rate and momentum constant.

C. Evaluation Metrics

During the training process, either Mean Absolute Error (MAE) or Mean Squared Error (MSE) loss functions were used for back-propagation. MSE is a common metric in Machine Learning, however MAE has strengths in Computer Vision[10]. MSE and MAE are able to describe the similarity of two tensors, however the goal is to accurately predict rainfall. In order to do that, we convert the predicted reflectivity values into actual rainfall rates in millimeters per hour (mm/hr). This was done for each grid-point in the reflectivity tensor using the Marshall-Palmer formula [7] (3).

$$\frac{\mathrm{mm}}{\mathrm{h}} = \left(\frac{10^{(dBZ/10)}}{200}\right)^{\frac{5}{8}} \tag{3}$$

A threshold was defined at 0.5mm/hr in order to convert the rainfall rates into a binary grid where 1 represents precipitation at that location and 0 represents no precipitation. This threshold is set to filter out virga conditions and barely detectable precipitation. With the binary precipitation grid for both the prediction (P) and observed reflectivity (T) tensors, we can compute further metrics by computing hits ($P_{i,j} = T_{i,j} = 1$), misses ($P_{i,j} = 0$, $T_{i,j} = 1$) and false-alarms ($P_{i,j} = 1$, $T_{i,j} = 0$). With these values, we can compute the Probability of Detection (POD)(4), False Alarm Rate (FAR)(5) and Critical Success Rate (CSI)(6).

$$POD = \frac{hits}{hits + misses} \tag{4}$$

$$FAR = \frac{false - alarms}{hits + false - alarms} \tag{5}$$

$$CSI = \frac{hits}{hits + misses + false - alarms} \tag{6}$$

	Observed Precipitation	No Observed Precipitation
Predicted Precipitation	Hit	False Alarm
No Predicted Precipitation	Miss	Ignored

Table I: Criterion Table for Nowcasting Metrics

VI. Results

In order to gauge the efficacy of various techniques attempted, comparison tests were run holding all hyper-parameters constant. Techniques explored include Decoder priming techniques (Table II), training loss function (Table III) and the sequence to sequence method utilized. The metrics reported represent the evaluation at 1 hour after the last radar scan was taken.

Technique	FAR	POD	CSI
Zeros	**0.29**	0.48	0.41
Ones	0.31	**0.51**	**0.43**
Last input	**0.29**	0.49	0.42

Table II: Effect of Different Priming Techniques

The first experiment evaluated the effect of priming techniques for the generative decoder network. In vector based problems, the use of a unique start symbol, either a special class or an extreme value, can be used to kick-start the string of predictions. As that this is a very detailed and data heavy regression model, making the encoder learn two tasks (detecting the start symbol and making a prediction from a prior radar image) may have a detrimental effect on overall training.

Loss Function	FAR	POD	CSI
MSE	**0.29**	0.49	0.42
MAE	0.30	**0.51**	**0.43**

Table III: Effect of Loss Function on Rainfall Metrics

We also chose to investigate the effects of two standard loss functions for tasks similar to this, MAE and MSE. The effects of the loss function can be significant in that while training with stochastic gradient descent, an error function that over-emphasizes very large error values, such as MSE, can provide an environment for small, potentially equally relevant errors to remain.

Method	FAR	POD	CSI
Attention	**0.28**	0.46	0.40
Encoder-Decoder	0.29	**0.48**	**0.41**

Table IV: Effect of Sequence-to-Sequence Methods

Both a standard encoder-decoder model and a more advanced attention model[3] of sequence to sequence techniques were used and evaluated in table IV. Surprisingly, the

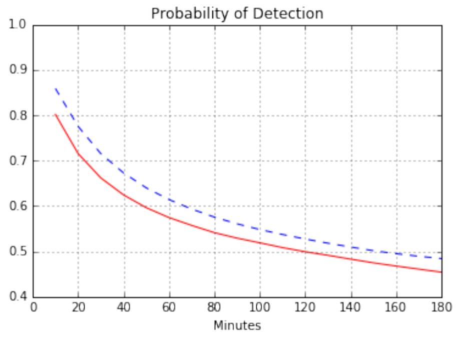


Figure 4: Probability of Detection: Predictions vs Persistence

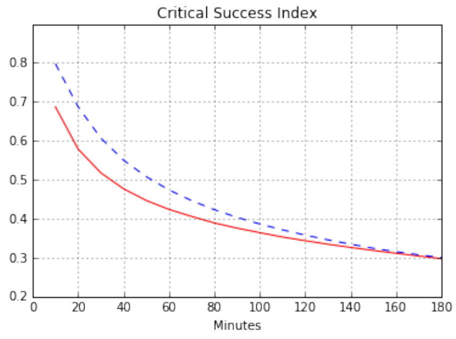


Figure 5: Critical Success Rate: Predictions vs Persistence

standard encoder-decoder method proved more successful in POD and CSI and saw only a slight increase in false alarms over the attention model.

In order to evaluate the degradation of prediction quality over time, the metrics were calculated and averaged for each timestep through 180 minutes. Figures 4, 5 and 6 display the values of these metrics for the neural network predictions (blue dashed line) over this 3 hour time period compared to a prediction of persistence (solid red line) on the best network determined by 100 iterations of spearmint.

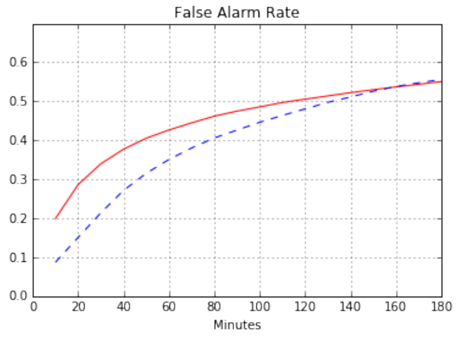


Figure 6: False Alarm Rate: Predictions vs Persistence

VII. Discussion

Differences in priming techniques on overall training performance seem to be minimal. Priming with a tensor of ones saw an increase in POD and CSI. A potential cause may be that since a input value of 1 would represent high reflectivity (80dBZ or higher), and therefore extreme precipitation, priming with this start symbol may increase the number of activations within the first few timesteps, thus propagating higher value through the output sequence. An issue we ran into regularly was a trend toward lower values for predictions further from the current time. This is likely caused by less than confident precipitation predictions being fed back into the RNN repeated causing the uncertainty to increase at each step. Priming with the last input tied for the lowest FAR and had improved CSI and POD scores relative to priming with zeros. This makes intuitive sense as that priming with the last input allows the decoder to focus on a single task: making a prediction based on the cell state and the last predicted or recorded reflectivity tensor.

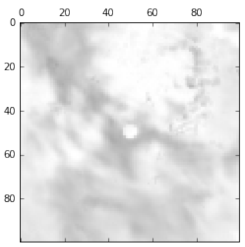
Surprisingly, the basic encoder-decoder network outperformed the attention model in both POD and CSI, but also led to more false alarms. The attention model provides a means for the decoder to access parts of the input sequence potentially relevant to that specific prediction. In this case, then entire input sequence should be relevant to each prediction. By forcing the neural network to learn a distinct single representation of the input sequence rather than depending on just a portion of that sequence, each prediction from the encoder is guaranteed to have a more complete representation of the environment leading up to that time.

Over the course of the first 3 hours of predictions, all metrics outpaced persistence. Though persistence is a low bar, we are likely to see the least change, and thus highest performance from persistence during the first 0-3 hours and thus clearing this bar is a vital first step in building a superior nowcasting model for very short term predictions. This proves that the neural network was able to understand common atmospheric motions and apply them to new situation effectively. This model performed worst on POD likely due to the rapid propagation of uncertainty that pushed predicted reflectivity values lower at each timestep.

VIII. Conclusion and Future Work

We provide a viable and scalable method for very short term precipitation nowcasting that can effectively bridge the gap between Numerical Weather Prediction and basic extrapolation techniques. This model was able to successfully beat a forecast of persistence in the 3 metrics described for the timeframe of concern.

Further evaluation should be done to compare the success of this model to traditional nowcasting techniques, both NWP at the hour level and extrapolation techniques for shorter timeframes. Improvements may be found through


(a) Title A

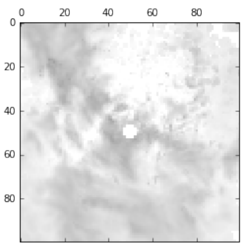
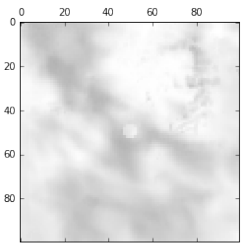

(b) Title B

Figure 7: Predicted vs Expected Reflectivity Graph at 10 minutes from t_0


(a) Title A

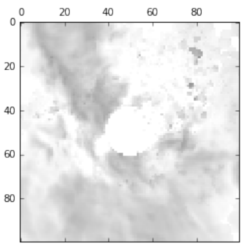
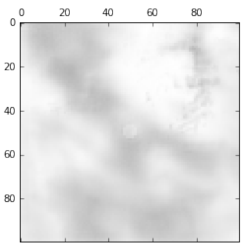

(b) Title B

Figure 8: Predicted vs Expected Reflectivity Graph at 30 minutes from t_0


(a) Title A

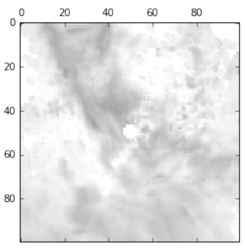
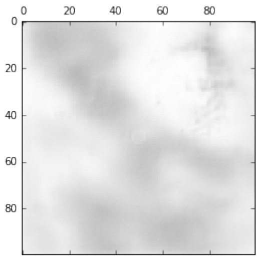

(b) Title B

Figure 9: Predicted vs Expected Reflectivity Graph at 60 minutes from t_0


(a) Title A

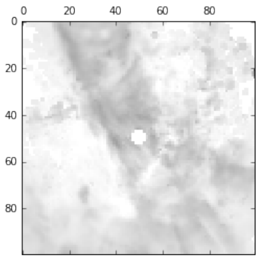

(b) Title B

Figure 10: Predicted vs Expected Reflectivity Graph at 90 minutes from t_0

further preprocessing of the radar images, including more sophisticated techniques for de-noising the raw reflectivity data and more exploration of initial resolution. A longer term goal is to implement real-time training and inference system where all new radar data is incorporated into the neural network model as it becomes available such that predictions improve continuously over time.

References

[1] X. Shi, Z. W. Chen, D. Y. Wong and W. C. Woo *Convolutional LSTM Network: A Machine Learning Approach For Precipitation Nowcasting* Neural Information Processing Systems (NIPS) 2015.

[2] I. Sutskever, O. Vinyals and Q. Le *Sequence to Sequence Learning with Neural Networks* Advances in Neural Information Processing Systems (NIPS), 2014.

[3] D. Bahdanau, K. Cho, Y. Bengio *Neural Machine Translation by Jointly Learning to Align and Translate* Proc. International Conference on Learning Representations http://arxiv.org/abs/1409.0473, 2015.

[4] A. Agarwal, E. Akchurin, C. Basoglu, G. Chen, S. Cyphers, J. Droppo, A. Eversole, B. Guenter, M. Hillebrand, T. Ryan Hoens, X. Huang, Z. Huang, V. Ivanov, A. Kamenev, P. Kranen, O. Kuchaiev, W. Manousek, A. May, B. Mitra, O. Nano, G. Navarro, A. Orlov, H. Parthasarathi, B. Peng, M. Radmilac, A. Reznichenko, F. Seide, M. L. Seltzer, M. Slaney, A. Stolcke, H. Wang, Y. Wang, K. Yao, D. Yu, Y. Zhang and G. Zweig *An Introduction to Computational Networks and the Computational Network Toolkit* Microsoft Technical Report MSR-TR-2014-112, 2014.

[5] J. J. Helmus and S. M. Collis *The Python ARM Radar Toolkit (Py-ART), a Library for Working with Weather Radar Data in the Python Programming Language.* Journal of Open Research Software. 4(1), p.e25. 2016. DOI: http://doi.org/10.5334/jors.119

[6] F. A. Gers, N. N. Schraudolph and J. Schmidhuber *Learning Precise Timing with LSTM Recurrent Networks* Journal of Machine LEarning Research 3, 2002.

[7] J. S. Marshall and M. Palmer *The Distribution of Raindrops with Size* Journal of Meteorology vol. 5, Shorter Contributions, 1948.

[8] C. Mass *Nowcasting: The Next Revolution in Weather Prediction* Bulletin of the American Meteorology Society, 2011.

[9] R. Jozefowics, W. Zaremba and I. Sutskever *An Empirical Exploration of Recurrent Network Architectures* In Proceedings of the 32nd International Conference on Machine Learning (ICML-15), 2015.

[10] H. Zhao, O. Gallo, I. Frosio, and J. Kautz *Loss Functions for Neural Networks for Image Processing* arXiv:1511.08861, 2015.

[11] J. Snoek, H. Larochelle and R. P. Adams *Practical Bayesian Optimization of Machine Learning Algorithms* Advances in Neural Information Processing Systems (NIPS), 2012.

[12] World Meteorological Organization *Nowcasting* **http://www.wmo.int/pages/prog/amp/pwsp/Nowcasting.htm**, retrieved 17 May 2017.

[13] N. E. Bowler, C. E. Pierce, A Seed *Bowler NE, Pierce CE, Seed A. Development of a precipitation nowcasting algorithm based upon optical flow techniques.* hskip 1em plus 0.5em minus 0.4emJournal of Hydrology 288.1,74-91 2004.

[14] A. W. Seed *A dynamic and spatial scaling approach to advection forecasting* hskip 1em plus 0.5em minus 0.4emJournal of Applied Meteorology 42.3,381-388 2003.

[15] U. Germann, I. Zawadzki *Scale-dependence of the predictability of precipitation from continental radar images Part I: Description of the methodology* hskip 1em plus 0.5em minus 0.4emMonthly Weather Review Dec;130.12,2859-73 2002.

[16] N. E. Bowler, C. E. Pierce, A. E. Seed *STEPS: A probabilistic precipitation forecasting scheme which merges an extrapolation nowcast with downscaled NWP* hskip 1em plus 0.5em minus 0.4emQuarterly Journal of the Royal Meteorological Society. Oct 1;132(620),2127-55 2006.