Precipitation Nowcasting: Leveraging Deep Convolutional Recurrent Neural Networks

Alexander Heye, Karthik Venkatesan, Jericho Cain

Agenda

- Precipitation Nowcasting
- Motivation
- Data Pipeline
- Deep Learning Architecture
- Experiments
- Results
- Summary
- Q&A

Precipitation Nowcasting

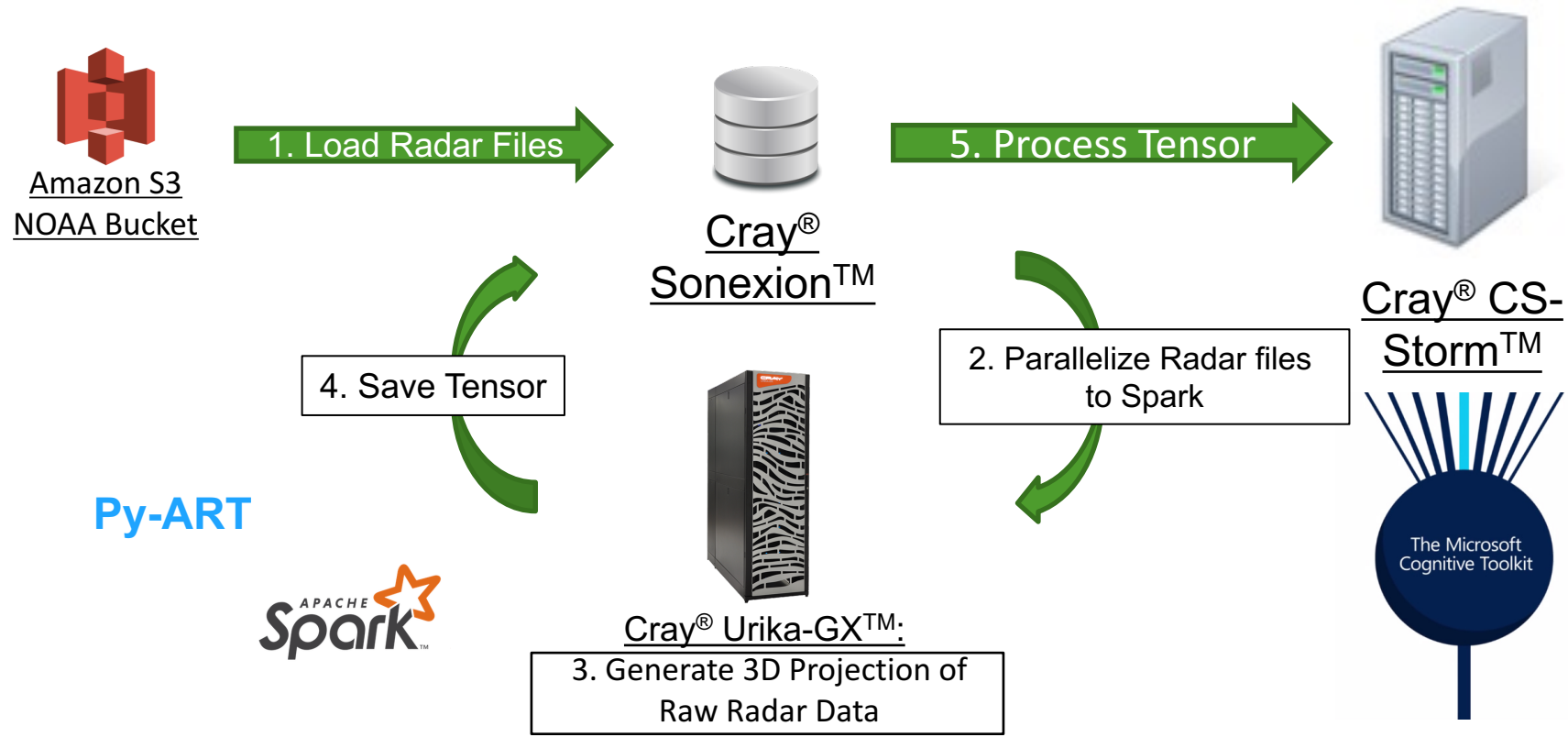
- **Problem: Predict precipitation locations and rates at a regional level over a short timeframe**
  - Neighborhood level predictions
  - T+0 – T+6 hours
- **Standard Approach: Numerical Weather Prediction**
  - Physics based simulations
  - High computational cost limits performance and accessibility
- **Cutting edge approach: Deep Learning**
  - Predict rainfall by learning from historical data
  - Heavy computation occurs ahead of time
  - Pre-Trained models can be deployed as soon as data is available

Motivation

- **Increase the quality and availability of very short term (0-1 hour) precipitation forecasts**
  - Will it rain on my walk home from work if it I leave right now?
  - Which bike-route should I take to avoid the rain?
- **Improve tracking quality of severe precipitation events**
  - Where do we issue severe weather warning?
  - Is a flash flood imminent? Do we need to evacuate?

- **Gain insights into the full deep learning workflow**
- **Accelerate the integration of deep learning in operational meteorology**

Data Pipeline

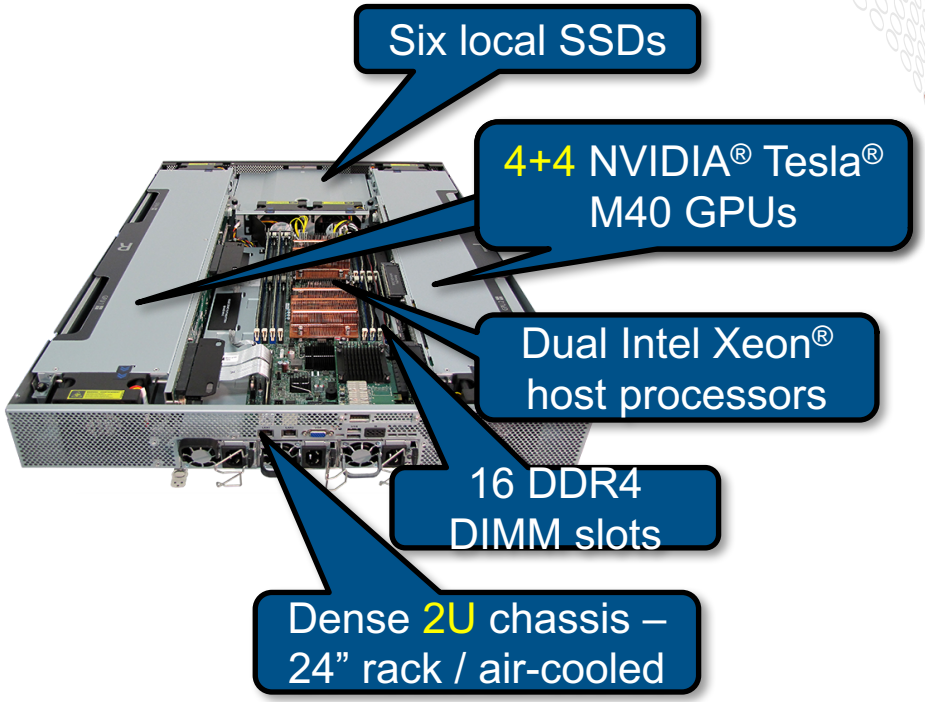


Data Preprocessing – Urika GX

- **Allowed us to easily leverage multiple analytics software frameworks and libraries**
  - Apache Spark
  - pyData
  - Jupyter Notebooks
- **Data processing was easily distributed among dozens of nodes and hundreds of cores.**
- **Full year of radar data processed and stored for training in just over 3 hours**

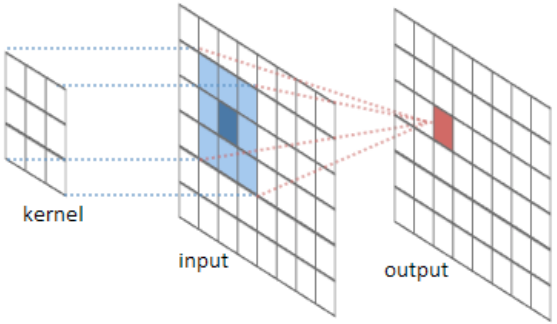
Model Training – CS-Storm

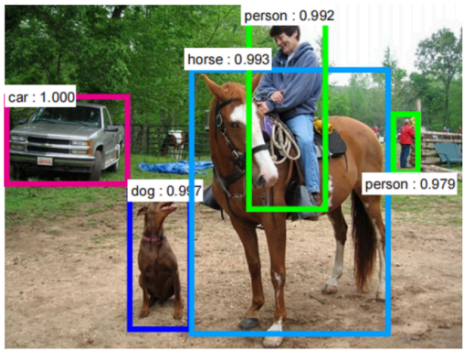
- Training a Neural Network is computationally demanding
- NVIDIA® CUDA® and CuDNN® libraries help accelerate the training time
- Cray CS-Storm cluster provides a GPU dense environment with 8 NVIDIA® Tesla® GPUs per node
- Training is performed in a data parallel setting
  - Copy of model weights stored on each GPU
  - Mini-batch split and distributed amongst them
- Deep Learning with Docker
  - Microsoft Cognitive Toolkit on a Docker image
  - Highly Portable and scalable

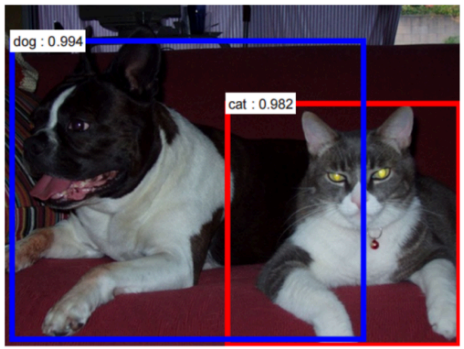


Convolutional Neural Networks

- Rely on a convolutional operation
- Strong ability to extract spatial relationships
  - Computer Vision
  - Board Games
- Examples: LeNet, AlexNet, ResNet

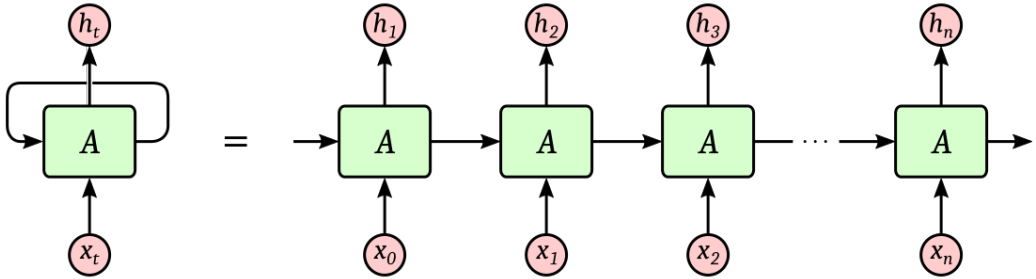






Recurrent Neural Networks

- Includes a recurrent connection and a mechanism of retaining state
- Designed to extract temporal relationships
  - Language Modeling
  - Speech Recognition
  - Machine Translation
- Examples: Simple-RNN, Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM)



Convolutional Long Short-Term Memory Network

- **Long Short-Term Memory (LSTM)**
  - RNN with a defined cell-state representing an encoded version of the sequential history.
  - Cell-State is updated through “gating functions” that control information retention, loss and acquisition.
  - LSTMs have a remarkable ability to retain and apply long-term dependencies of a sequence.
- **Convolutional LSTM**
  - Variant of the standard LSTM
  - Embedded convolutional operations
  - State vectors replaced with N-D tensors

LSTM Gate and Output Functions

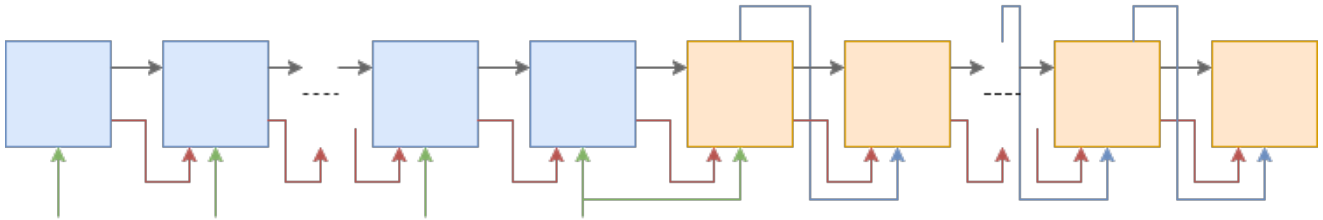
$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t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

ConvLSTM Gate and Output Functions

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

Sequence To Sequence

- **Nowcasting is a sequence to sequence problem**
  - Input: Sequence of radar images leading up to the current time
  - Output: Sequence of predicted radar images arbitrarily far in the future
- **Solution: Encoder-Decoder Networks**
  - Encoder (Blue) digests the input sequence and produces a single tensor representation
  - This tensor is used to initialize the decoder
  - Decoder (Orange) takes previous images as input and produces predictions of the next image.



Data

- **Source: NOAA NEXRAD Level-II Reflectivity Data from station KATX, outside of Seattle, Wa.**
  - 2015 calendar year
  - All sequences from full days with over 0.1 inches of precipitation
- **Radial Reflectivity data projected onto a 3-dimensional Cartesian grid with help from pyart**
  - Grid shape: 5 x 100 x 100
  - Area covered: 12km x 200km x 200km
  - Spatial resolution of roughly 2km

Metrics

- Hit: Correct prediction of precipitation at a location
- Miss: Failure to predict precipitation at that location
- False-Alarm: Prediction of Precipitation when none was detected
- True-Negative: No Precipitation was observed nor predicted - ignored

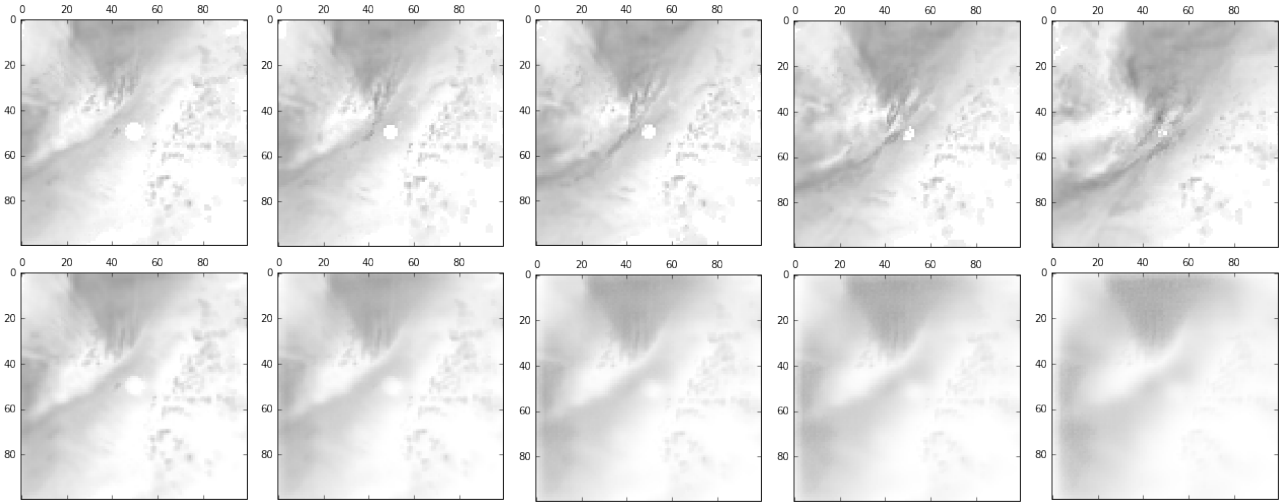
	Observed Precipitation	No Observed Precipitation
Predicted Precipitation	Hit	False Alarm
No Predicted Precipitation	Miss	True Negative

- False Alarm Rate: Fraction of false alarms to predicted precipitation
  - FAR = false-alarms / (hits + false-alarms)
- Probability of Detection: Fraction of hits to observed precipitation
  - POD = hits / (hits + misses)
- Critical Success Index: Fraction of hits to measured and observed precipitation
  - CSI = hits / (hits + misses + false-alarms)

Example Predictions

Observed Reflectivity

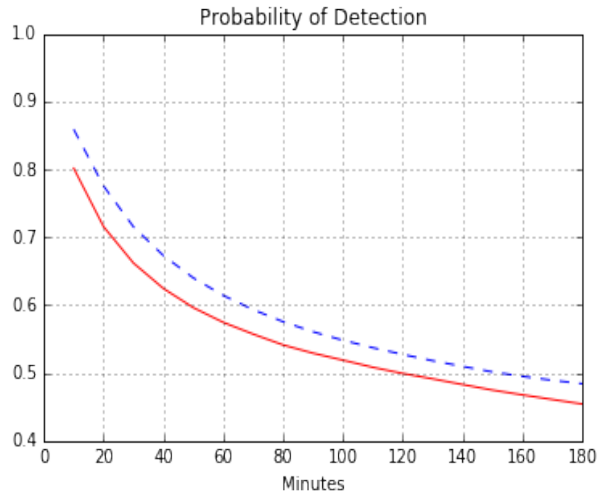
Predicted Reflectivity

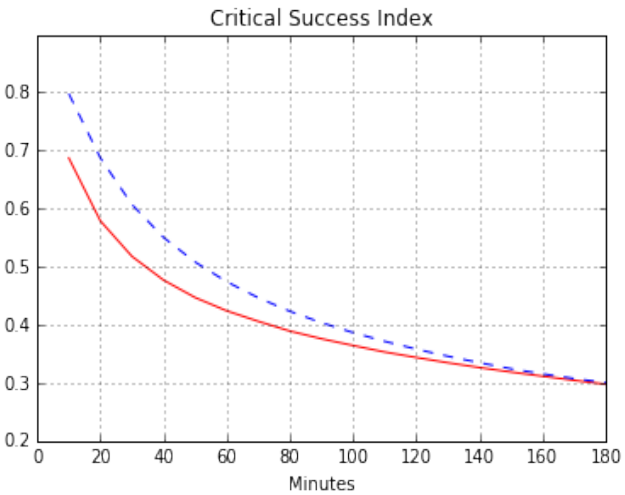


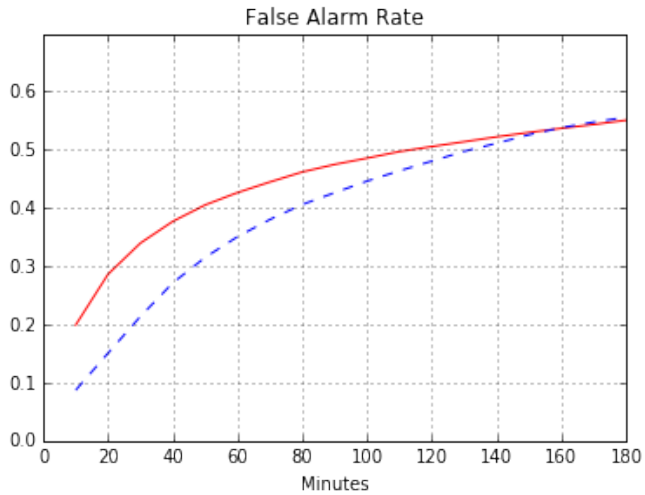
Time Since Last Obs. (Min)

10 30 50 70 90

Results over time







Summary

- **Build a Precipitation Nowcasting deep learning model**
  - Gain insights into the full deep learning workflow
  - Improve the quality and speed of precipitation nowcasting
- **Big-Data task**
  - Exploring possible data-sources
  - Building a trainable dataset from raw radar data
- **Neural Network Model**
  - Convolutional Long Short-Term Memory Neural Network
  - Encoder Decoder
- **Results**
  - Beating persistence
  - More to come…

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, and URIKA. The following are trademarks of Cray Inc.: APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, REVEAL, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.


Q&A
Alexander Heye
aheye@cray.com
CUG.2017.CAFFEINATED COMPUTING
Redmond, Washington May 7-11, 2017