A Deep Learning Approach for Precipitation Nowcasting with RNNs using Analytics Zoo on BigDL

Alexander Heye, Cray and Ding Ding, Intel

Agenda

- **Introduction**
  - Urika-XC
  - Analytics Zoo on BigDL
  - Precipitation Nowcasting
- **Models**
  - Convolutional Long Short-Term Memory Network
  - Sequence to sequence model
- **Results**
- **Q&A**

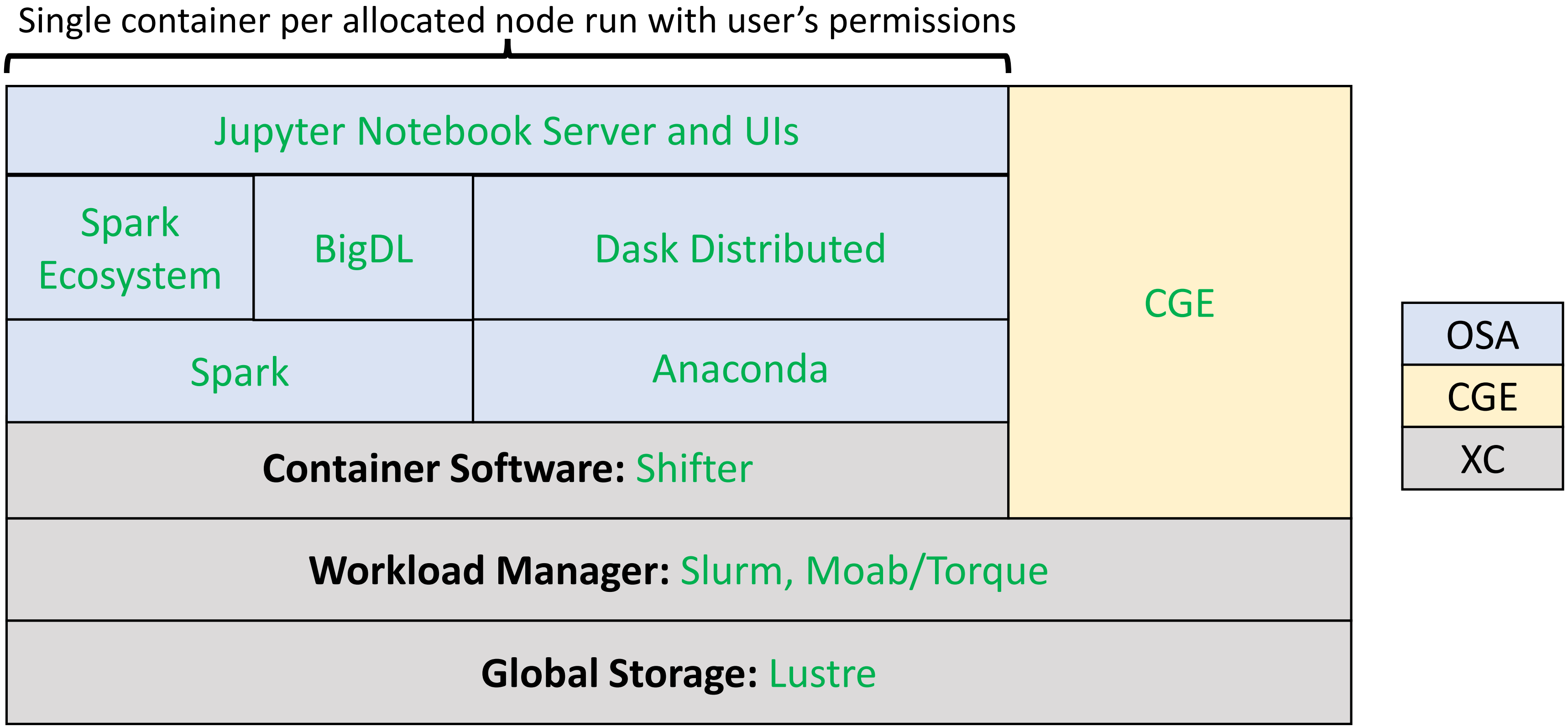
Introduction - Urika-XC

- Enabled through containerization – Shifter
- Brings Analytics software to the Cray XC
  - Apache Spark
  - Anaconda Python
  - Intel BigDL
  - Cray Graph Engine (CGE)
  - Dask Distributed
- Productivity Tools
  - Jupyter Notebooks, Tensorboard

- Support for most HPC workload managers
  - Slurm, Moab Torque, PBS Pro
- Example (Slurm): salloc –N 34 ./start_analytics
  - Starts an interactive shell on a XC compute node and will bring up Spark and Dask Distributed clusters
  - Experience for users will be similar to running jobs from a login node on the Urika-GX analytics platform

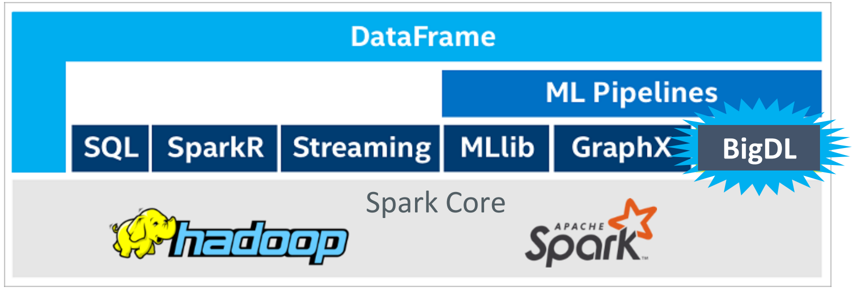
https://www.cray.com/products/analytics/urika-xc

Architecture



Introduction - Intel BigDL

HIGH PERFORMANCE DEEP LEARNING FOR APACHE SPARK* ON CPU INFRASTRUCTURE



BigDL is an **open-source** distributed deep learning library for Apache Spark* that can run directly on top of existing Spark or Apache Hadoop* clusters

Ideal for DL Models TRAINING and INFERENCE

Designed and Optimized for Intel® Xeon®

No need to deploy costly accelerators, duplicate data, or suffer through scaling headaches!

Feature Parity & Model Exchange with TensorFlow*, Caffe*, Keras, Torch*

Lower TCO and improved ease of use with existing infrastructure

Deep Learning on Big Data Platform, Enabling **Efficient Scale-Out**

Powered by Intel® MKL and multi-threaded programming

Introduction - Intel Analytics Zoo

Build and Productionize Deep Learning Apps for Big Data at Scale

Reference Use Cases	• Anomaly detection • Sentiment analysis • Fraud detection • Chatbot, sequence prediction, etc.
Built-In Deep Learning Models	• Image classification • Object detection • Text classification • Recommendations • Sequence-to-sequence, GAN, etc.
Feature Engineering	Feature transformations for • Image, text, 3D imaging, time series, speech, etc.
High-Level Pipeline APIs	• Native deep learning support in Spark DataFrames and ML Pipelines • Autograd, Keras and transfer learning APIs for model definition • Support for model serving/inference pipelines
Backbends	Spark, BigDL etc.

https://github.com/intel-analytics/analytics-zoo/ https://analytics-zoo.github.io/

Introduction – Intel Analytics Zoo

Build end-to-end deep learning applications for big data

- E2E analytics + AI **pipelines** (natively in Spark DataFrames and ML Pipelines) using ***nnframes***
- Flexible **model definition** using ***autograd, Keras-style & transfer learning APIS***
- **Data preprocessing** using built-in ***feature engineering operations***
- Out-of-the box **solutions** for a variety of problem types using ***built-in deep learning models and reference use cases***
- Large-scale distributed **TensorFlow model** inference using ***TFNet***

Introduction - 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

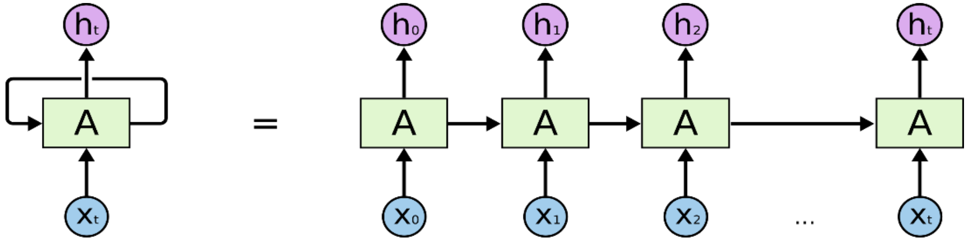
Precipitation Nowcasting - Motivation

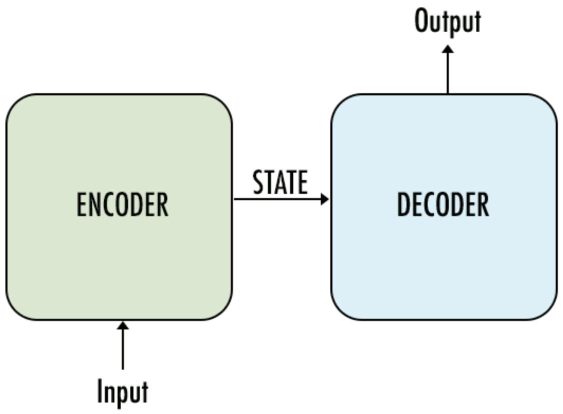
- Increase the quality and availability of very short term (0-1 hour) precipitation forecasts
  - Will it rain on my walk home from work if it I leave right now?
  - Which bike-route should I take to avoid the rain?
- Improve tracking quality of severe precipitation events
  - Where do we issue severe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

Precipitation Nowcasting Model

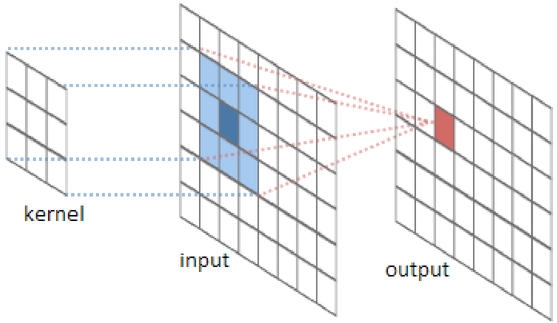
- Convolutional Recurrent Neural Network
  - Convolutional Neural Network – Spatial Patterns
  - Recurrent Neural Network – Temporal Patterns
  - ConvLSTM – Convolutional Long Short-Term Memory Network
- Sequence to Sequence
  - Encoder Decoder
  - Use recent history to predict future changes

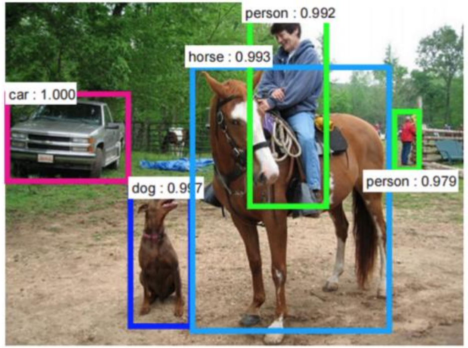


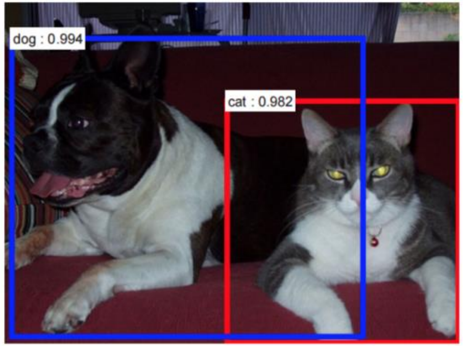


Convolutional Neural Networks

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

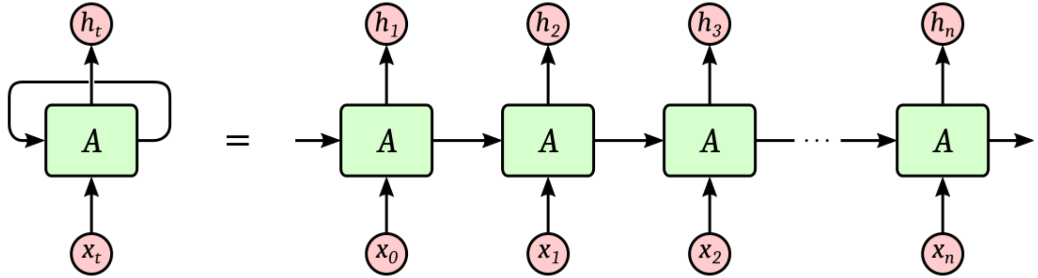






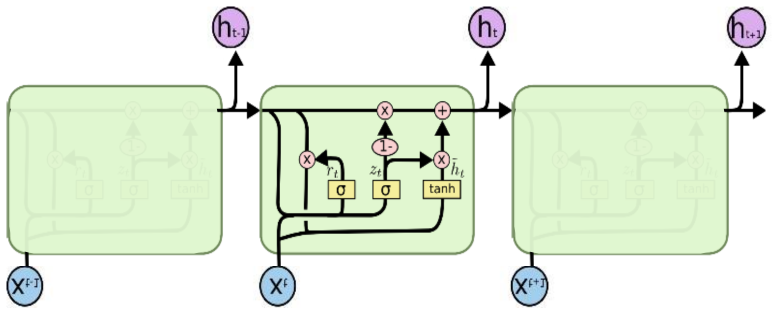
Recurrent Neural Network

- Has a "memory" which captures information about what has been calculated so far
- Designed to extract temporal relationships
  - Language Modeling
  - Speech Recognition
  - Machine Translation
- Examples: Simple-RNN, Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM)



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.



LSTM Gate and Output Functions

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

Convolutional Long Short-Term Memory Network

- Convolutional LSTM
  - Variant of the standard LSTM
  - Embedded convolutional operations
  - State vectors replaced with N-D tensors

LSTM Gate and Output Functions

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

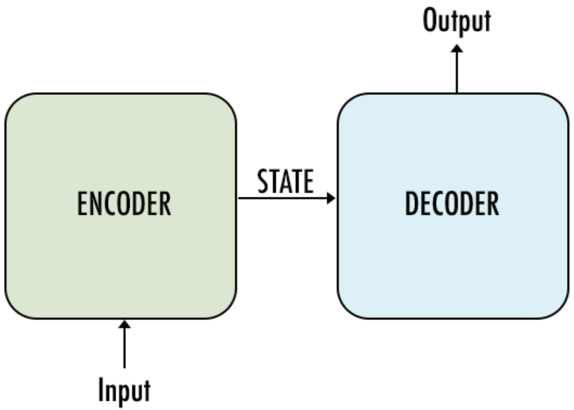
ConvLSTM Gate and Output Functions

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

Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting
https://arxiv.org/abs/1506.04214

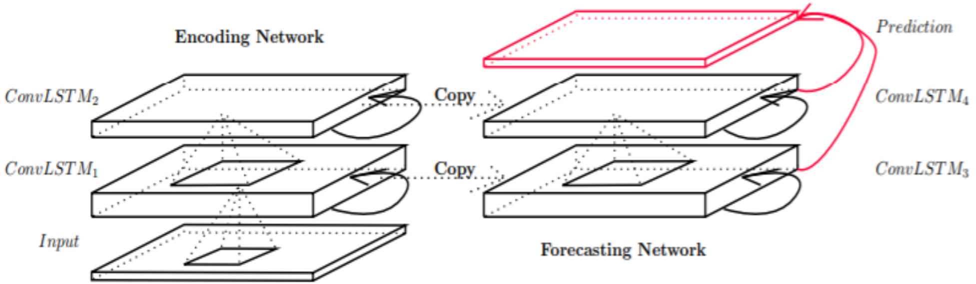
Sequence To Sequence

- **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 (Green) digests the input sequence and compress into a hidden state
  - Decoder (Blue) takes previous images as input and produces predictions of the next image.



Precipitation Nowcasting model

- encoding network and forecasting network
- formed by stacking several ConvLSTM layers



Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting
https://arxiv.org/abs/1506.04214

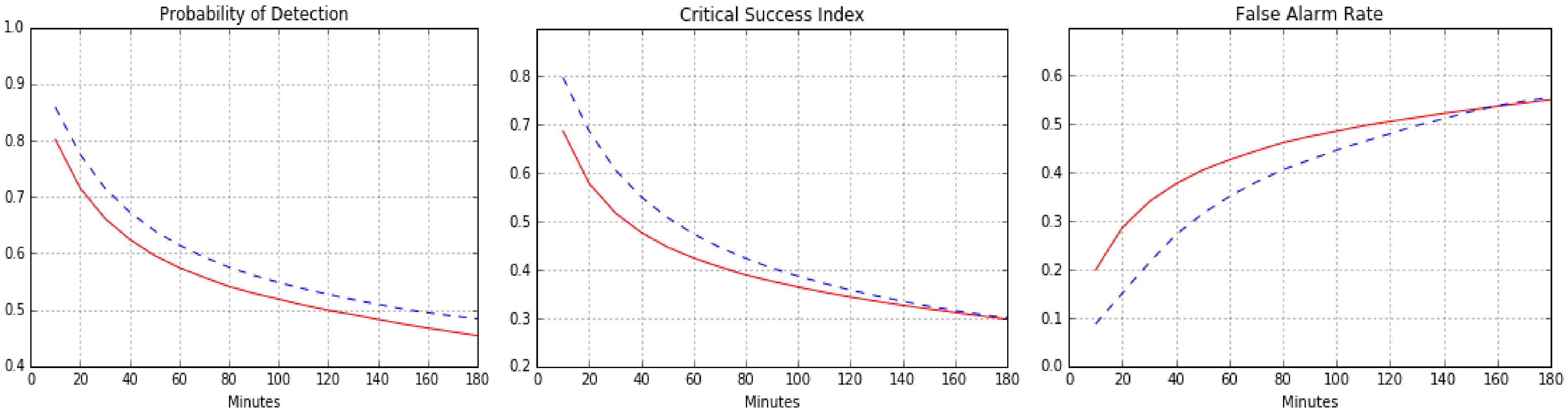
Metrics

- **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)

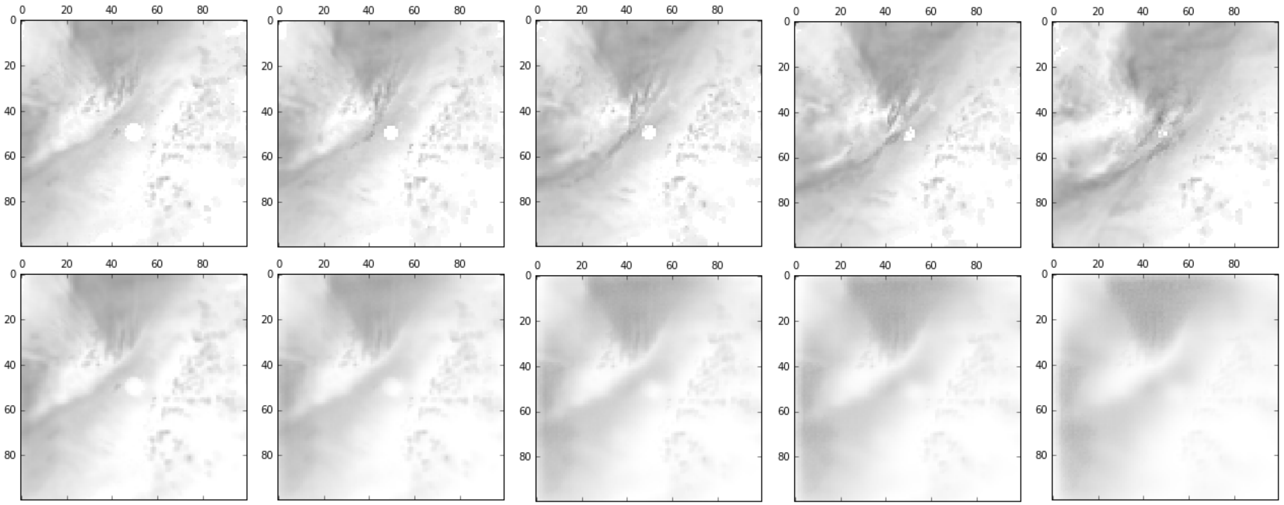
Results over time



Example Predictions

Observed Reflectivity

Predicted Reflectivity



Time Since Last Obs. (Min)

10 30 50 70 90

Further Reading ☺

https://github.com/intel-analytics/analytics-zoo

https://analytics-zoo.github.io/master/index.html

https://github.com/intel-analytics/BigDL/

https://bigdl-project.github.io/master/index.html

Legal Disclaimer

- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.
- No computer system can be absolutely secure.
- Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit **http://www.intel.com/performance**.

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.