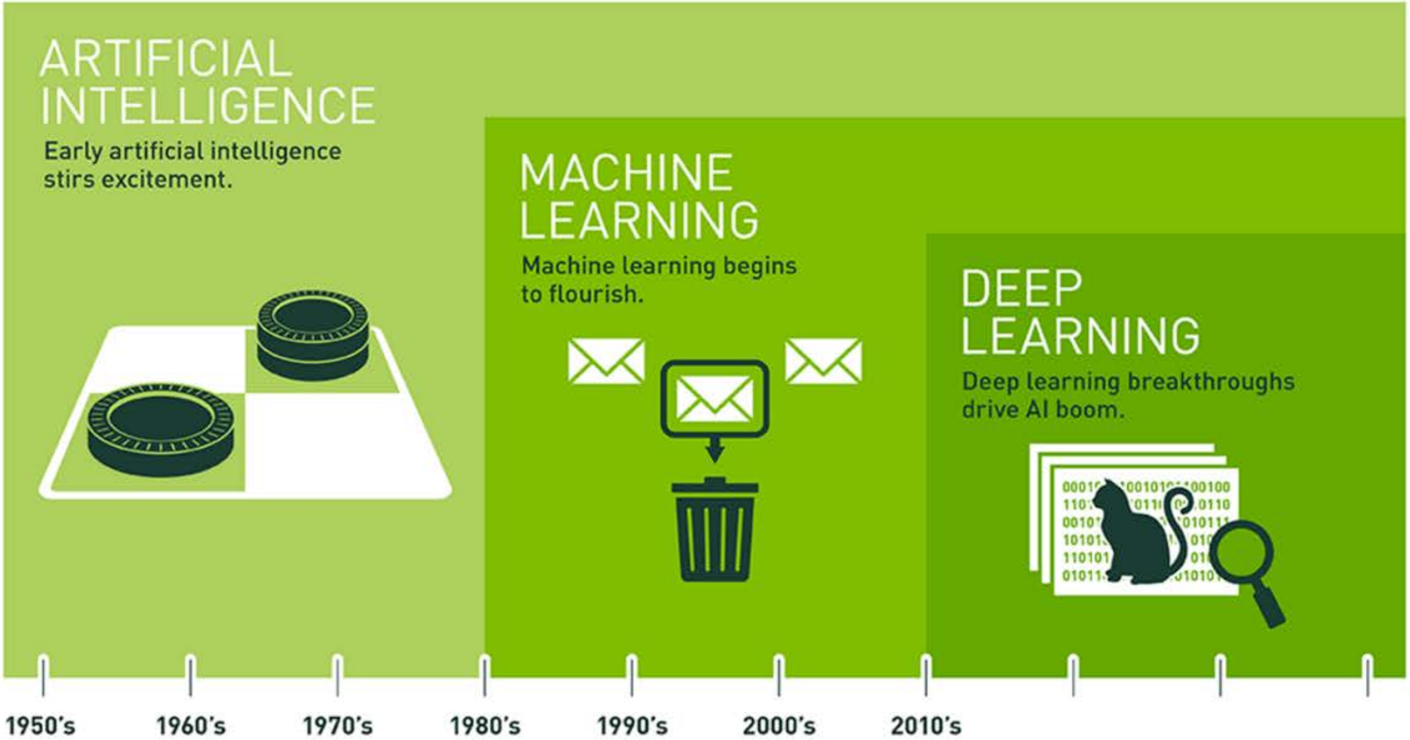
Improving Satellite Data Utilization Through Deep Learning

Jebb Stewart[*], Christina Bonfanti[**], David M. Hall[***], Isidora Jankov[*], Lidia Trailovic[**], Stevan Maksimovic[*], Mark Govett

September 26th, 2018

NOAA Earth System Research Laboratory (ESRL), Boulder, CO
[*]Cooperative Institute for Research in the Atmosphere (CIRA)
[**]Cooperative Institute for Research in Environmental Sciences (CIRES)
[***]NVIDIA Coorporation

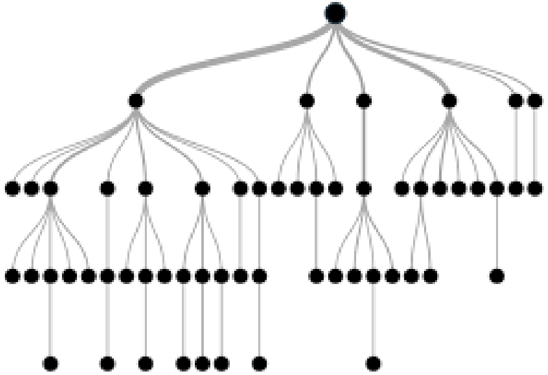
AI, MACHINE LEARNING, AND DEEP LEARNING



Source : Michael Copeland, Journalist for WIRED, Fortune, and Business 2.0

Machine Learning

- *".. at its most basic is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world." -- Nvidia*
- It's been around a while ~ 1959
- Closely related to statistics



Recent explosion lead by advances in processing power, availability of data, techniques, and tools

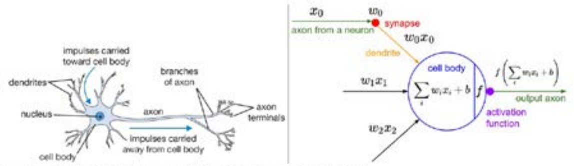
Deep Learning

- Deep learning is a subset of machine learning and uses a layered structure called an artificial neural network
- The design is inspired by the biological neural network similar to the human brain. Learns and makes decisions on it’s own
- Similar to humans, we never get to the bottom of their thoughts from a cellular point of view, hard to explain why it works
- Very good at:

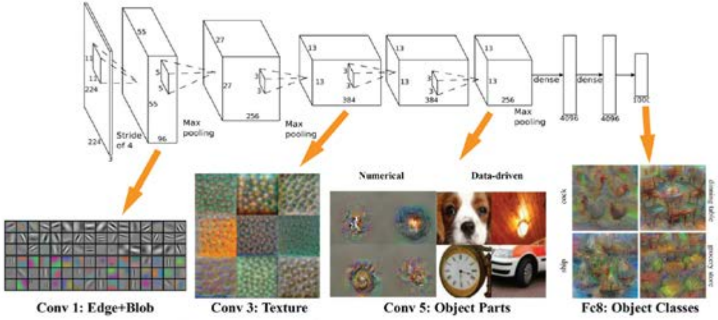
  Object Detection

  Time Series Data - Natural Language Processing

Another way to write software



A cartoon drawing of a biological neuron (left) and its mathematical model (right).



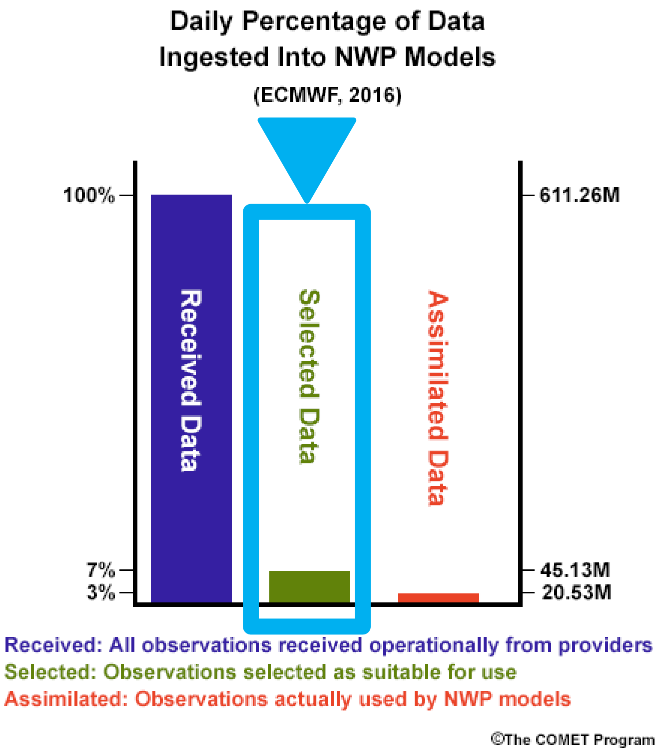
A visual representation of a convolutional neural net from the mNeuron plugin created for MIT's computer vision courses/teams.

Challenges – Operational Constraints

- The satellite data assimilation process is computationally expensive and data are often reduced in resolution to allow timely incorporation into the forecast
- Not all observations have equal value.
- With limited time, how do we best to extract the observations with greatest impact?
- Increasing forecast model resolution (ie sub 3 KM) is dependent on improved model assimilation

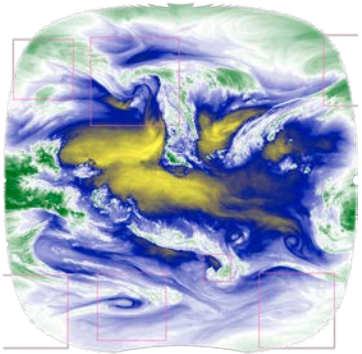
Satellite Data Assimilation Today

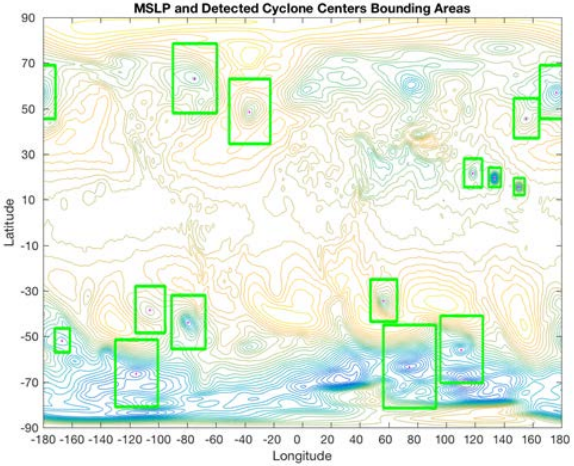
- There are far more satellite data than can be assimilated into the models
- At present, we use only ~3% of the available satellite data
- Next generation Geostationary Satellites (GOES-16, GOES-17)
  - Order of magnitude increase in data volume!!



Regions of Interest (ROI's)

- Use deep learning object detection to identify areas of atmospheric instability from satellite observation data.
- Focus the extraction of observations on these regions of interest.
- Identify other important phenomena for forecasting from satellite data:
    - convective instability
    - baroclinic instability
    - convective initiation

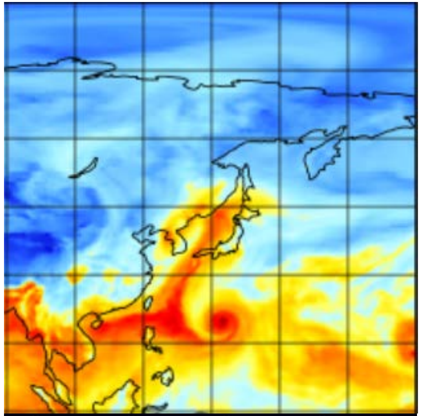


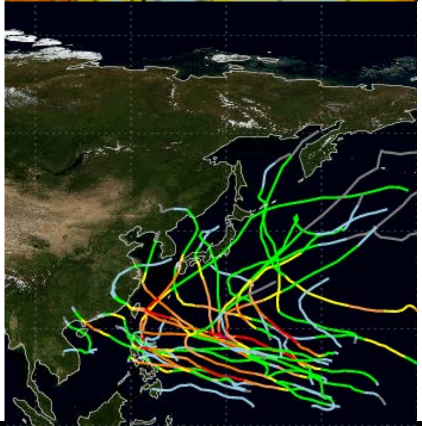


Development of Training Data

Labeling Data

- Global Forecast System (GFS) analysis data provides various data (etc… temperature, pressure, humidity)
- International Best Track Archive for Climate Stewardship (IBTrACS) provides tropical cyclone track data, including timestamps, positions, strength, radius, etc…
- Precipitable Water is a good proxy for Water Vapor observations from satellites





Development of Training Data for all Cyclones

Creating a labeled dataset of tropical cyclones and worldly low/cyclonic systems

- GFS analysis data
- Matrix of 1’s and 0’s for True and False label
- Next steps: cyclogenesis and convection initiation

Literature background on formal definitions of the following characteristics for cyclones:

- Previous cyclone tracking and forecasting methods
  - Create our own for labeling purposes
- Formal area, size, intensity, and physical features
- Meteorologists know what these systems look like, so how would a meteorologist tell a computer what to look for?
  - Signatures in water vapor, pressure fields, vorticity, etc.

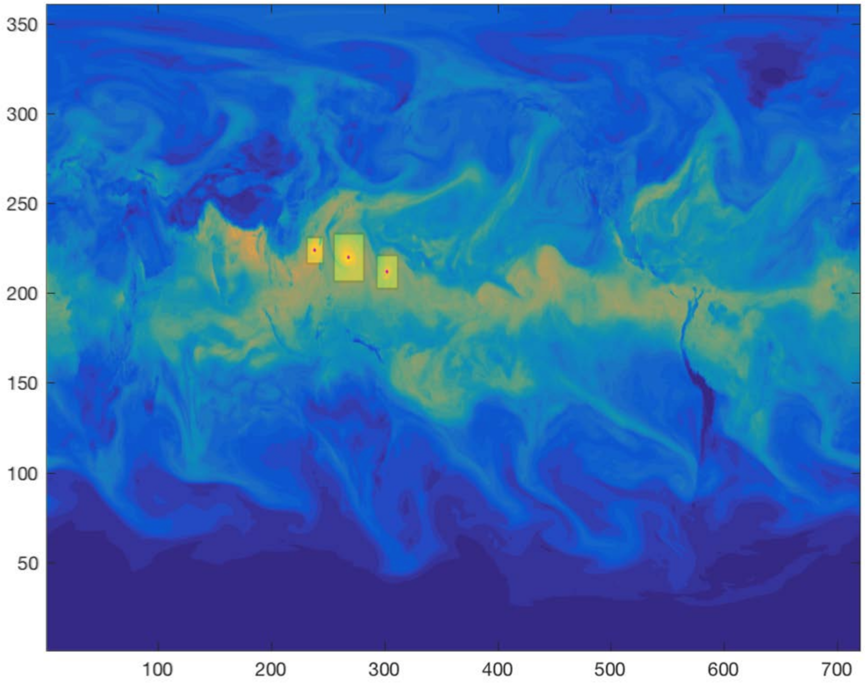
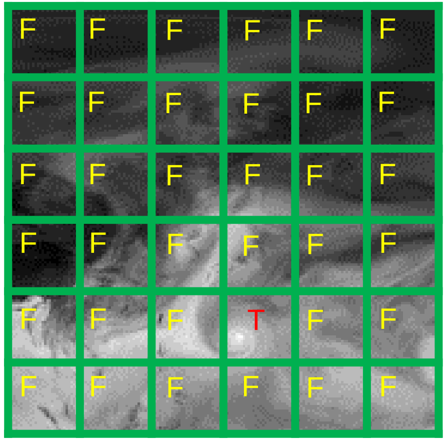
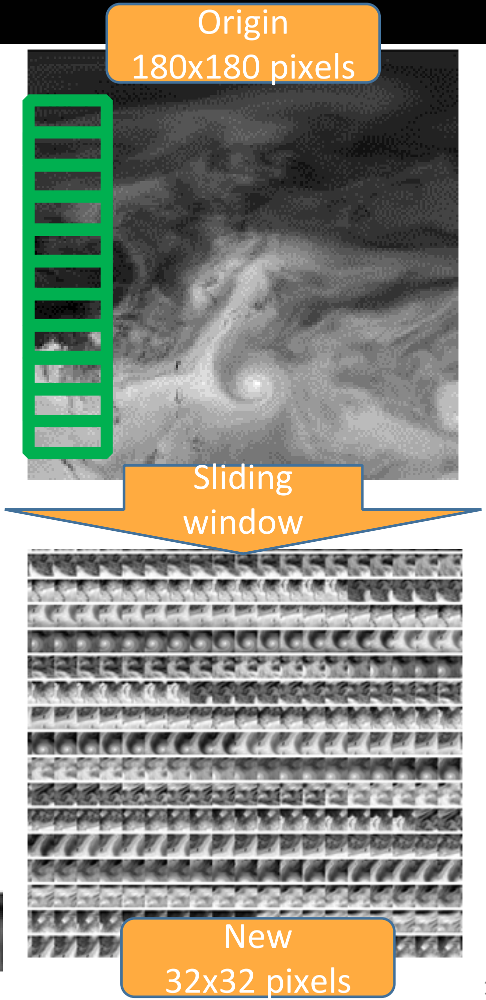


Image above gives an example of the tropical cyclone labeled dataset. Shown are total precipitable water in entire atmosphere, pink labeled tropical cyclone centers, and then highlighted regions that define the area of the storm

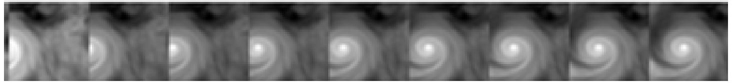
Data processing



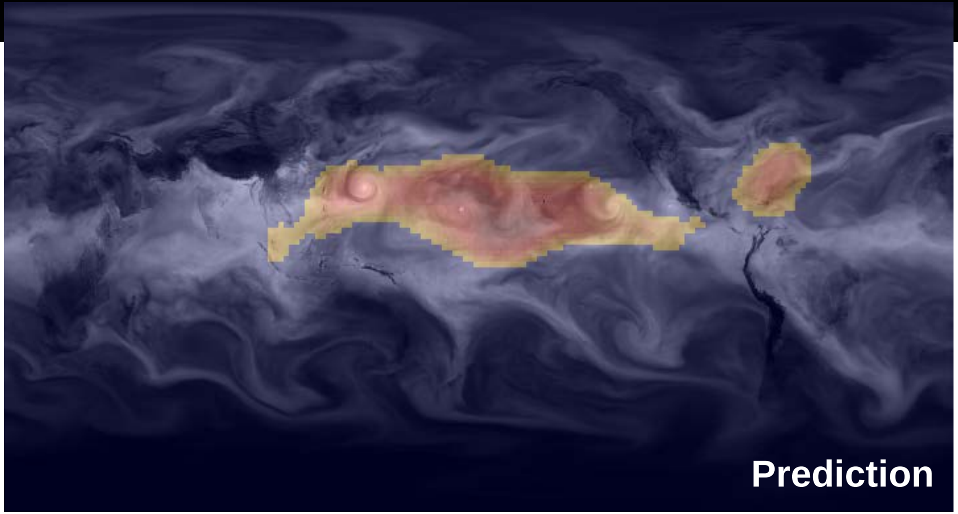


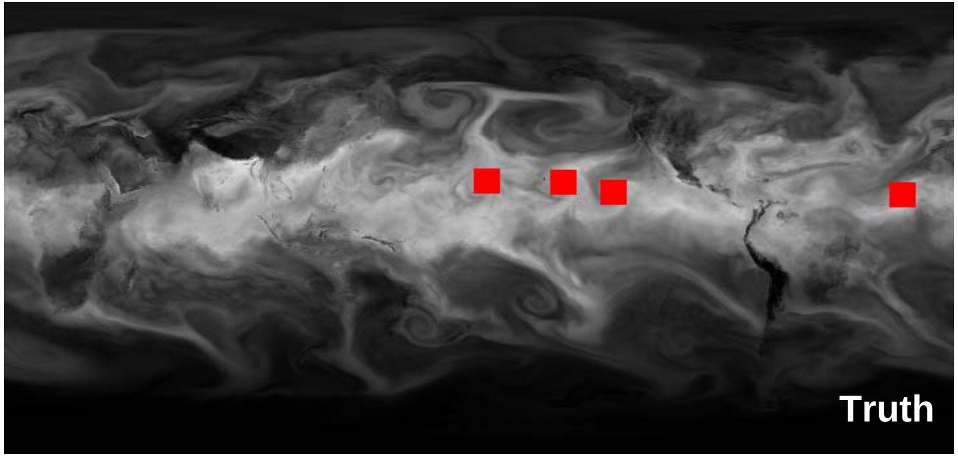
- Images processing concept:
  Sliding Windows

- Sliding windows help:
  - Small / Unbalanced data set problem
    - Only a few true but many false samples
  - boundary problem
    - A cyclone is not located in the center of cell, is cut by the boundary



Classification with Sliding Window





Total Precipitable Water from GFS

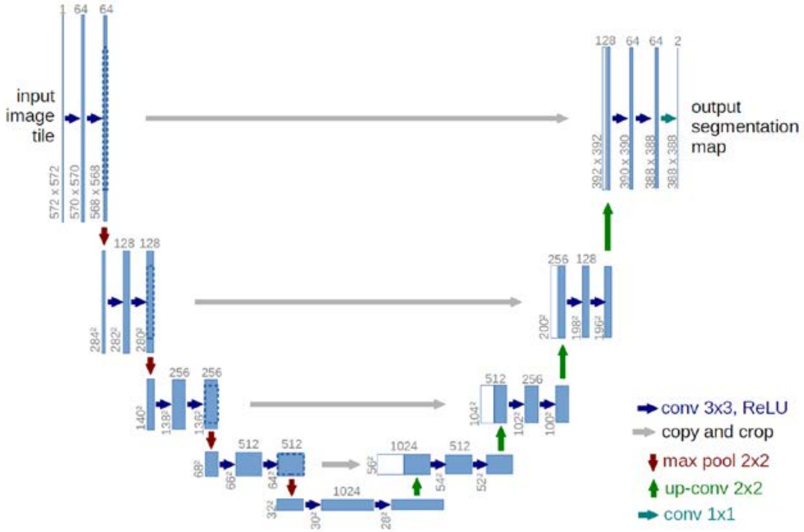
Deep Learning Development for Image Segmentation

Keras

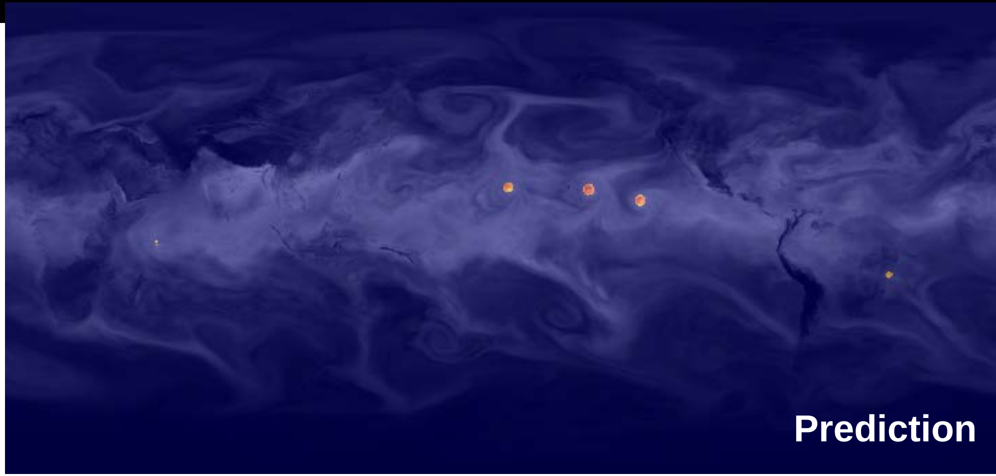
- https://keras.io
- High Level Deep Learning Library using Google Tensorflow under the hood

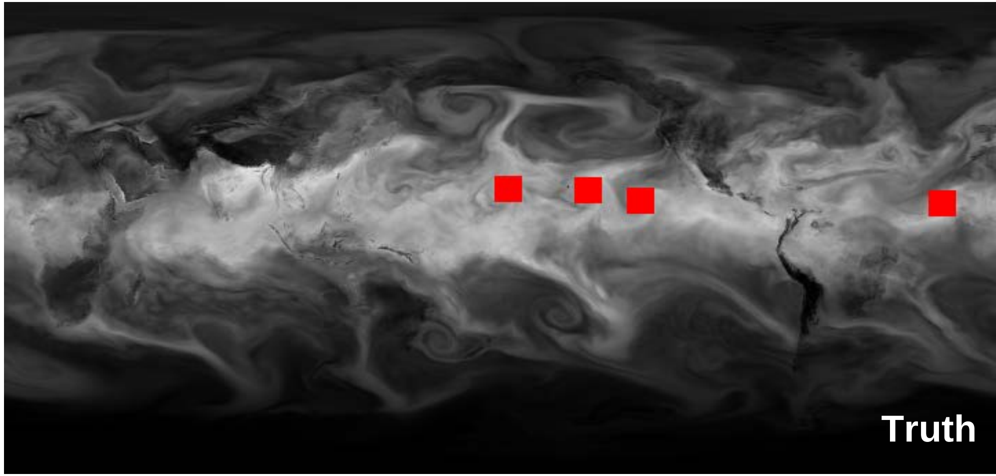
Unet - Deep Neural Network

- Links larger features before compression with smaller features after compression
- Commonly seen in image segmentation challenges on Kaggle.com



Segmentation with Sliding Window





Total Precipitable Water from GFS

Problems with Sliding Window

Duplicate Data

- Each window contains identical information from previous window

Slower Inference

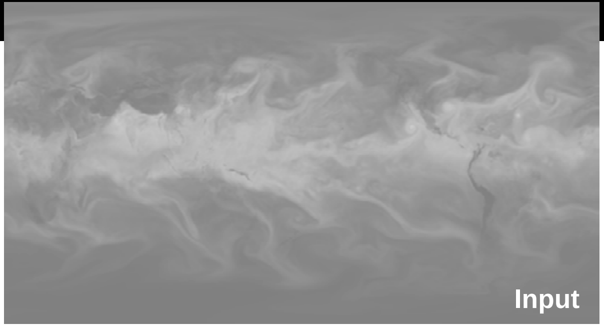
- Need to split up incoming data into windows
- Need to process prediction back into result

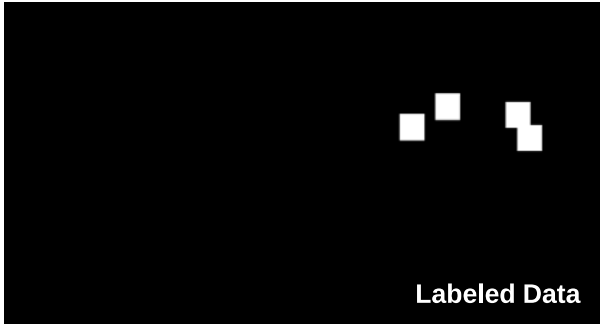
Classification results in “Smeared” Picture

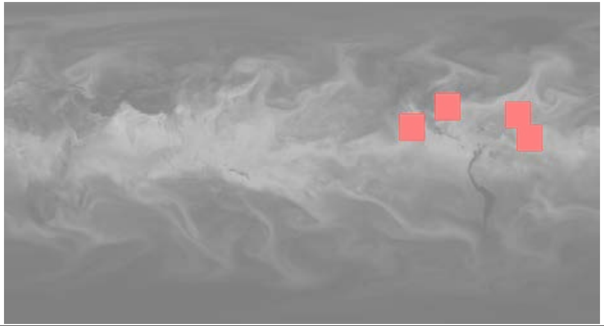
Training Data - Using GFS

- Precipitable water from Global Forecast System (GFS) model analysis
- Storm Centers from IBTracs dataset
- Input data normalized to range from -1 to +1
- Trained 2010-2013 Validation 2014, Test 2015
- Image segmentation 20x20 pixel segmentation box centered on tropical systems
- Only use storms classified as Tropical Storm or greater on Saffir Simpson Scale
    - 34 knots and above

~ 7200 Total Labeled data



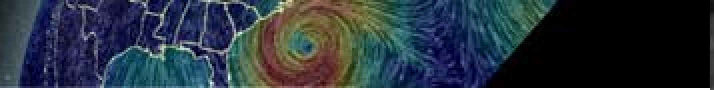




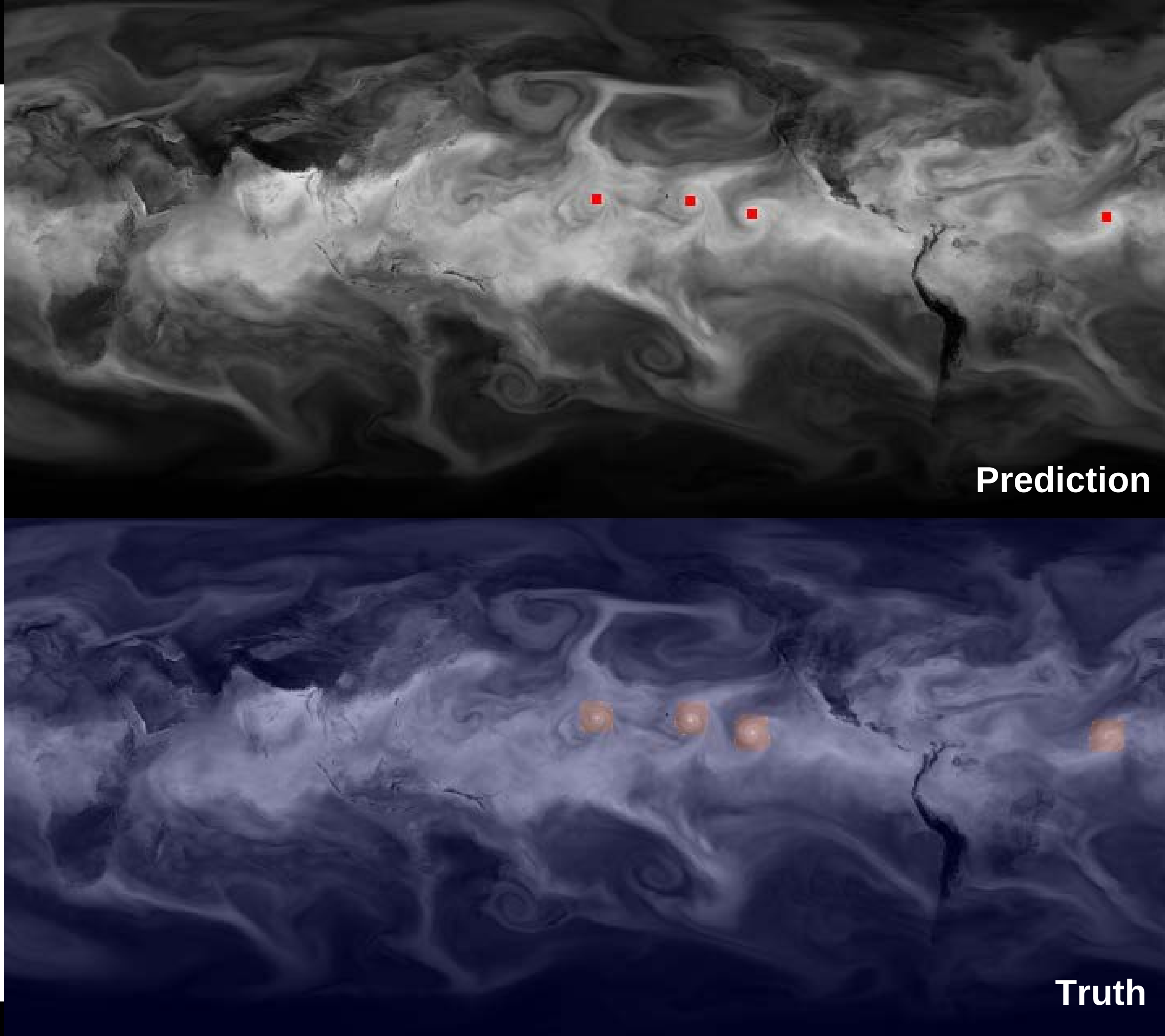
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 320, 704, 1)	0	
conv2d_1 (Conv2D)	(None, 320, 704, 32)	320	input_1[0][0]
batch_normalization_1 (BatchNor	(None, 320, 704, 32)	128	conv2d_1[0][0]
activation_1 (Activation)	(None, 320, 704, 32)	0	batch_normalization_1[0][0]
conv2d_2 (Conv2D)	(None, 320, 704, 32)	9248	activation_1[0][0]
batch_normalization_2 (BatchNor	(None, 320, 704, 32)	128	conv2d_2[0][0]
activation_2 (Activation)	(None, 320, 704, 32)	0	batch_normalization_2[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 160, 352, 32)	0	activation_2[0][0]
conv2d_3 (Conv2D)	(None, 160, 352, 64)	18496	max_pooling2d_1[0][0]
batch_normalization_3 (BatchNor	(None, 160, 352, 64)	256	conv2d_3[0][0]
activation_3 (Activation)	(None, 160, 352, 64)	0	batch_normalization_3[0][0]
conv2d_4 (Conv2D)	(None, 160, 352, 64)	36928	activation_3[0][0]
batch_normalization_4 (BatchNor	(None, 160, 352, 64)	256	conv2d_4[0][0]
activation_4 (Activation)	(None, 160, 352, 64)	0	batch_normalization_4[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 80, 176, 64)	0	activation_4[0][0]
conv2d_5 (Conv2D)	(None, 80, 176, 128)	73856	max_pooling2d_2[0][0]
batch_normalization_5 (BatchNor	(None, 80, 176, 128)	512	conv2d_5[0][0]
activation_5 (Activation)	(None, 80, 176, 128)	0	batch_normalization_5[0][0]
conv2d_6 (Conv2D)	(None, 80, 176, 128)	147584	activation_5[0][0]
batch_normalization_6 (BatchNor	(None, 80, 176, 128)	512	conv2d_6[0][0]
activation_6 (Activation)	(None, 80, 176, 128)	0	batch_normalization_6[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 40, 88, 128)	0	activation_6[0][0]
conv2d_7 (Conv2D)	(None, 40, 88, 256)	295168	max_pooling2d_3[0][0]
batch_normalization_7 (BatchNor	(None, 40, 88, 256)	1024	conv2d_7[0][0]
activation_7 (Activation)	(None, 40, 88, 256)	0	batch_normalization_7[0][0]
conv2d_8 (Conv2D)	(None, 40, 88, 256)	590080	activation_7[0][0]
batch_normalization_8 (BatchNor	(None, 40, 88, 256)	1024	conv2d_8[0][0]
activation_8 (Activation)	(None, 40, 88, 256)	0	batch_normalization_8[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 20, 44, 256)	0	activation_8[0][0]
conv2d_9 (Conv2D)	(None, 20, 44, 512)	1180160	max_pooling2d_4[0][0]
batch_normalization_9 (BatchNor	(None, 20, 44, 512)	2048	conv2d_9[0][0]
activation_9 (Activation)	(None, 20, 44, 512)	0	batch_normalization_9[0][0]
batch_normalization_19 (BatchNo	(None, 80, 176, 128)	512	conv2d_19[0][0]
activation_19 (Activation)	(None, 80, 176, 128)	0	batch_normalization_19[0][0]
conv2d_20 (Conv2D)	(None, 80, 176, 128)	147584	activation_19[0][0]
batch_normalization_20 (BatchNo	(None, 80, 176, 128)	512	conv2d_20[0][0]
activation_20 (Activation)	(None, 80, 176, 128)	0	batch_normalization_20[0][0]
conv2d_21 (Conv2D)	(None, 80, 176, 128)	147584	activation_20[0][0]
batch_normalization_21 (BatchNo	(None, 80, 176, 128)	512	conv2d_21[0][0]
activation_21 (Activation)	(None, 80, 176, 128)	0	batch_normalization_21[0][0]
up_sampling2d_4 (UpSampling2D)	(None, 160, 352, 128	0	activation_21[0][0]
concatenate_4 (Concatenate)	(None, 160, 352, 192	0	activation_4[0][0] up_sampling2d_4[0][0]
conv2d_22 (Conv2D)	(None, 160, 352, 64)	110656	concatenate_4[0][0]
batch_normalization_22 (BatchNo	(None, 160, 352, 64)	256	conv2d_22[0][0]
activation_22 (Activation)	(None, 160, 352, 64)	0	batch_normalization_22[0][0]
conv2d_23 (Conv2D)	(None, 160, 352, 64)	36928	activation_22[0][0]
batch_normalization_23 (BatchNo	(None, 160, 352, 64)	256	conv2d_23[0][0]
activation_23 (Activation)	(None, 160, 352, 64)	0	batch_normalization_23[0][0]
conv2d_24 (Conv2D)	(None, 160, 352, 64)	36928	activation_23[0][0]
batch_normalization_24 (BatchNo	(None, 160, 352, 64)	256	conv2d_24[0][0]
activation_24 (Activation)	(None, 160, 352, 64)	0	batch_normalization_24[0][0]
up_sampling2d_5 (UpSampling2D)	(None, 320, 704, 64)	0	activation_24[0][0]
concatenate_5 (Concatenate)	(None, 320, 704, 96)	0	activation_2[0][0] up_sampling2d_5[0][0]
conv2d_25 (Conv2D)	(None, 320, 704, 32)	27680	concatenate_5[0][0]
batch_normalization_25 (BatchNo	(None, 320, 704, 32)	128	conv2d_25[0][0]
activation_25 (Activation)	(None, 320, 704, 32)	0	batch_normalization_25[0][0]
conv2d_26 (Conv2D)	(None, 320, 704, 32)	9248	activation_25[0][0]
batch_normalization_26 (BatchNo	(None, 320, 704, 32)	128	conv2d_26[0][0]
activation_26 (Activation)	(None, 320, 704, 32)	0	batch_normalization_26[0][0]
conv2d_27 (Conv2D)	(None, 320, 704, 32)	9248	activation_26[0][0]
batch_normalization_27 (BatchNo	(None, 320, 704, 32)	128	conv2d_27[0][0]
activation_27 (Activation)	(None, 320, 704, 32)	0	batch_normalization_27[0][0]
conv2d_28 (Conv2D)	(None, 320, 704, 1)	33	activation_27[0][0]

Total params: 34,613,793
Trainable params: 34,599,777
Non-trainable params: 14,016

What the network looks like



Segmentation with Fully Connected Network

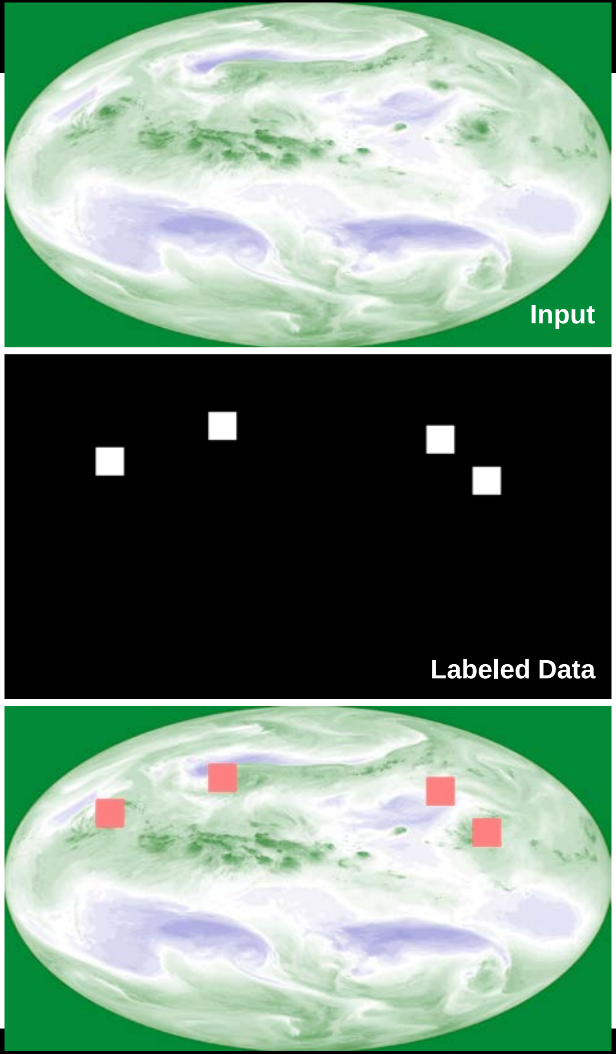


Total Precipitable Water from GFS

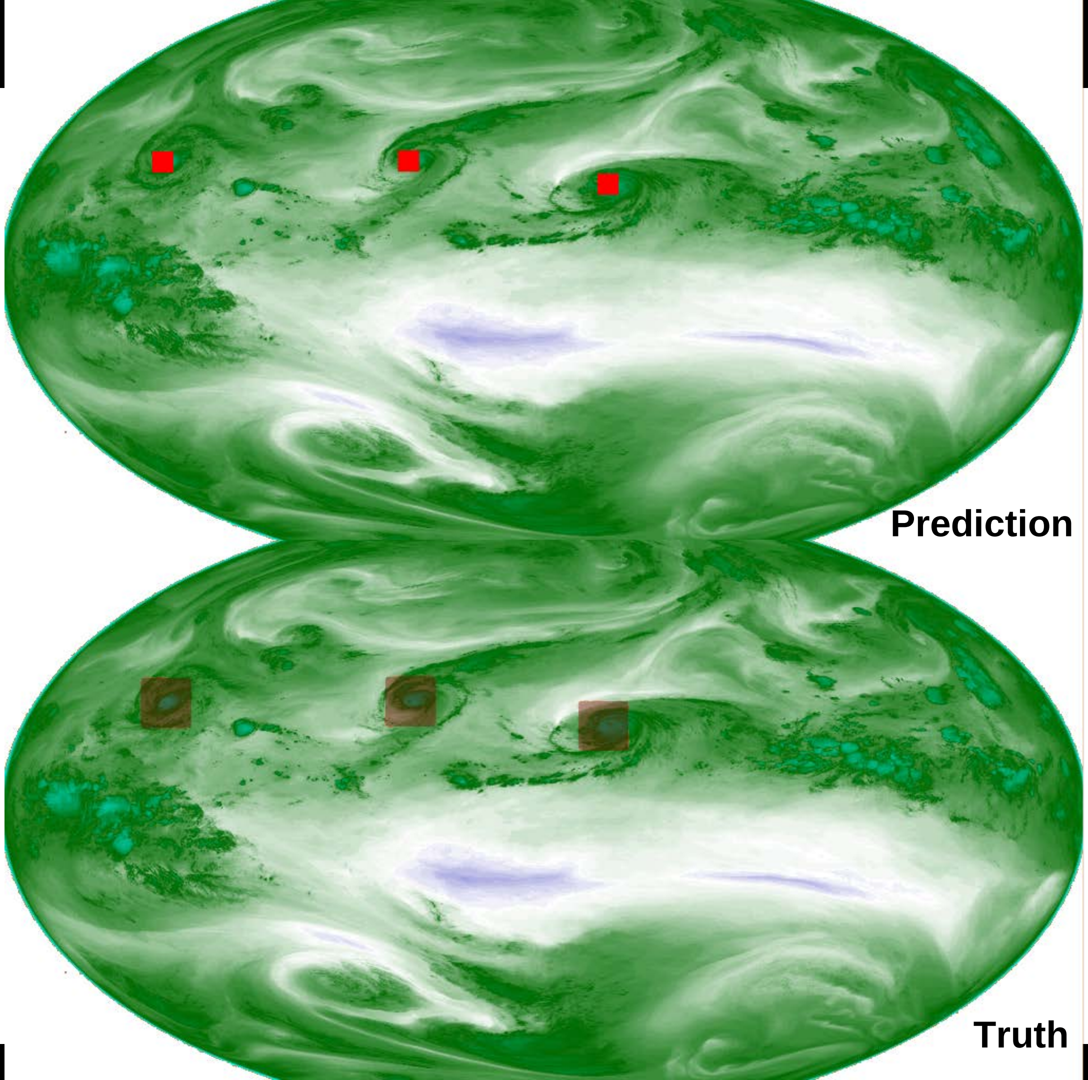
Using Satellite Data for Training

- Water Vapor Channel from GOES 10, 11,12,13,14, and 15
- Storm centers from IBTracks Dataset
- Data normalized to range from -1 to +1
- Trained 2010-2013 Validated 2014, Test 2015
- Images resized and cropped to 1024x512
- Image segmentation 25x25 pixel box segmentation centered on storm
- Only use storms classified as Tropical Storm or greater on Saffir Simpson Scale
    - 34 knots and above

~ 4500 Labeled Data



Segmentation with Fully Connected Network



Water Vapor GOES-15

Training Details

NOAA High Performance Computing System

Theia

- 100 nodes
- Each node has two 10 core Haswell processors
- Each node has 256 GB of memory
- Each node has 8 Tesla P100 (Pascal) GPUs.
    - 16 GB Memory Each

Training Time - GFS Data

- Keras Multi GPU Setting to use multi gpu single node configuration
  - GFS Model Data
    - 704x320x1
    - 1.2 GB per image
    - 72 Images per batch (~ 10.8 GB Per GPU)
    - ~ 80 seconds per epoch
    - Early stopping ~ 70 epochs (~ 1.5 hours for complete training)
    - Inference ~ 40 ms

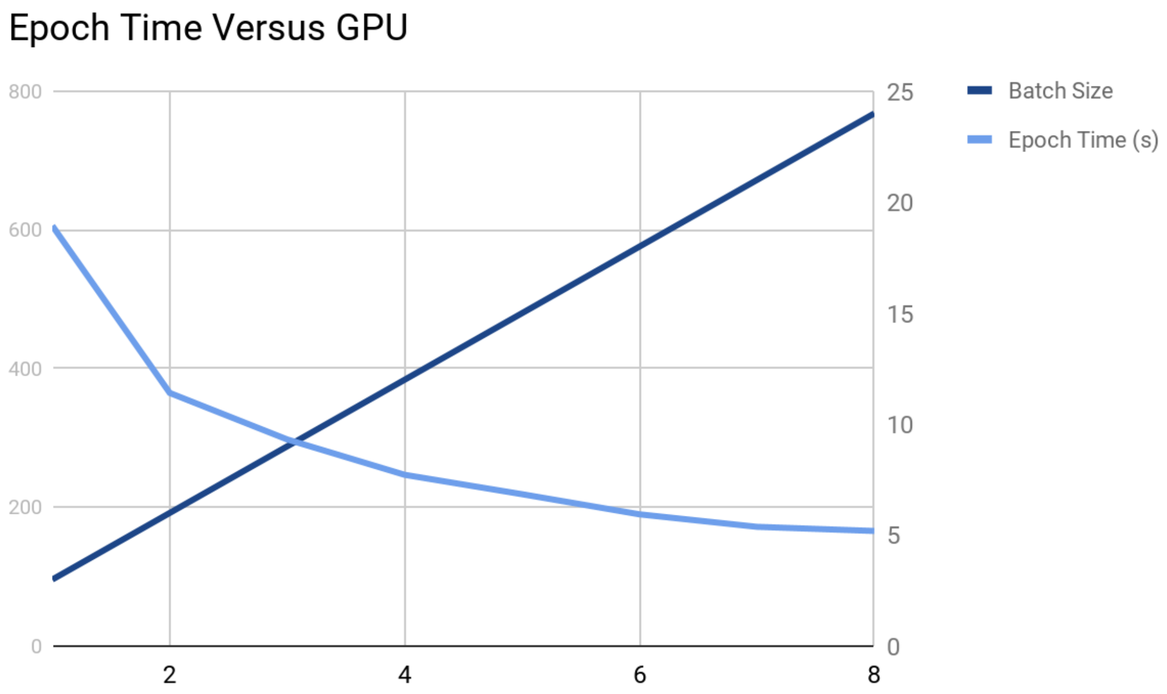
- Comparison to CPU
  - 7 Hours Per Epoch
  - ~ 500 Hours to Complete Training!
  - Inference ~ 1 Second

Training Time - Satellite Data

- Keras Multi GPU Setting to use multi gpu single node configuration
  - Satellite Data
    - 1024x512x3 (RGB)
    - 2.6 GB per image
    - 24 Images per batch (~ 7.8 GB Per GPU)
    - ~ 3 minutes per epoch
    - Early stopping ~ 70 epochs (~ 3 hours for complete training)
    - Inference ~ 40 ms

- Comparison to CPU
  - 11.5 hours per epoch
  - ~ 400 Hours to Complete Training!
  - Inference ~ 1 Second

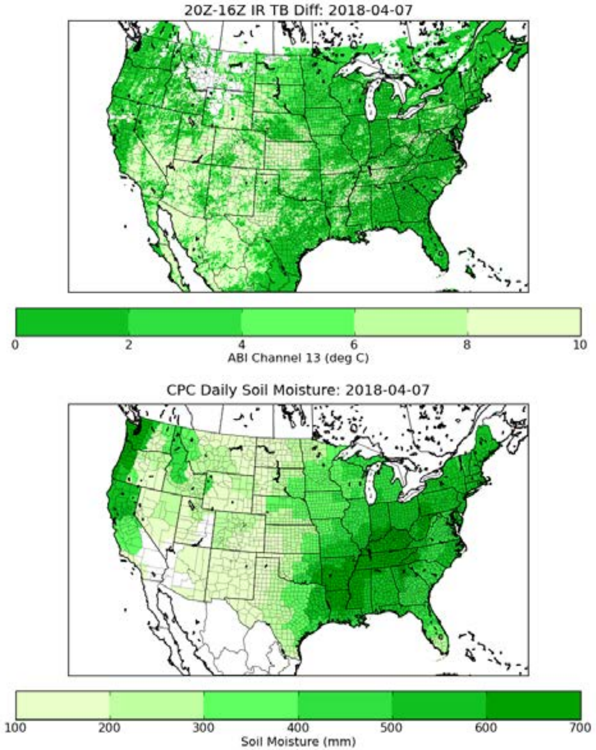
Multi GPU Scaling



Other Deep Learning Applications

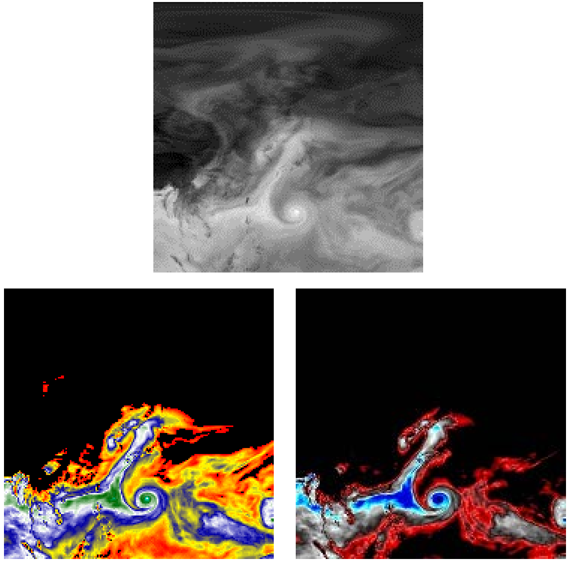
Soil Moisture from Satellite Radiances

- Use machine learning to correlate radiances from GOES-16 ABI to generate soil moisture product for model assimilation
- Working with CIRA, Kyle Hilburn and Steve Miller



Future Work for Deep Learning/AI

- Different color palette (RGB) channels
- Use actual storm radius values for segmentation
- Time series (3+ time steps into channels)
- Trim labels near oblique angles (edges) for satellite data
- Multi Node Multi GPU using Horovod
  https://github.com/uber/horovod



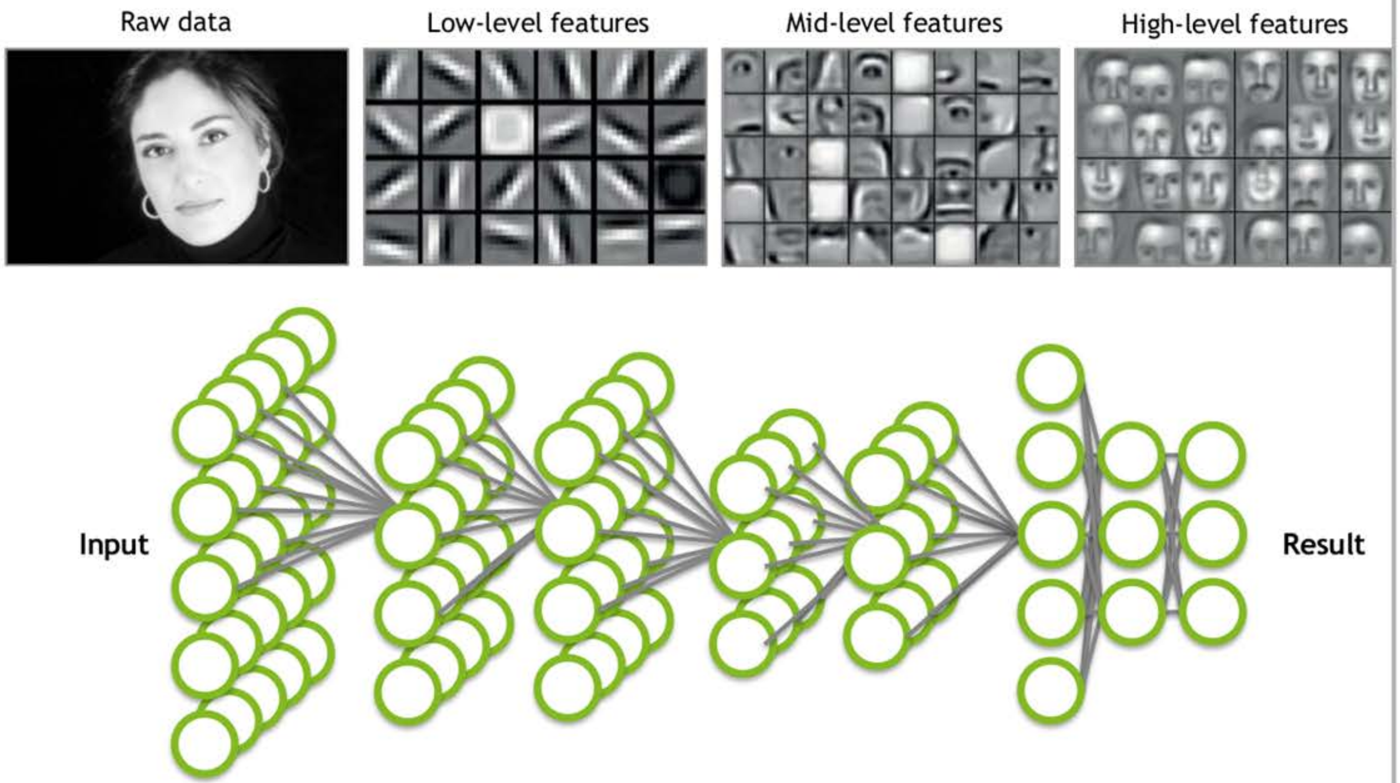
Many Other Potential Applications

- Improved Speed, performance, accuracy of Model Functions like Radiative Transfer Models, Convection Parameterization, or other parameterized functions
- Classification of Atmospheric Conditions from Satellite for Improved Model Verification
- Air Quality Probabilities for Taiwan
- Bias correction from observations
- Assist HRRR Smoke in cleaning fire detection observations (prescribed vs fire vs other)
- Chemistry modeling optimization for different species
- Investigate hurricane intensification probabilities. (NWS SSDs)

Questions?

Jebb.Q.Stewart@noaa.gov

DEEP Convolutional NEURAL NETWORK (DNN)



Application components:

Task objective
e.g. Identify face
Training data
10-100M images
Network architecture
~10s-100s of layers
1B parameters
Learning algorithm
~30 Exaflops
1-30 GPU days

WHAT IS A CONVOLUTION?

A refresher of 2D convolutions

Image

I1	I2	I3	I4	I5	I6
I7	I8	I9	I10	I11	I12
I13	I14	I15	I16	I17	I18
I19	I20	I21	I22	I23	I24
I25	I26	I27	I28	I29	I30
I31	I32	I33	I34	I35	I36

Filter/Feature

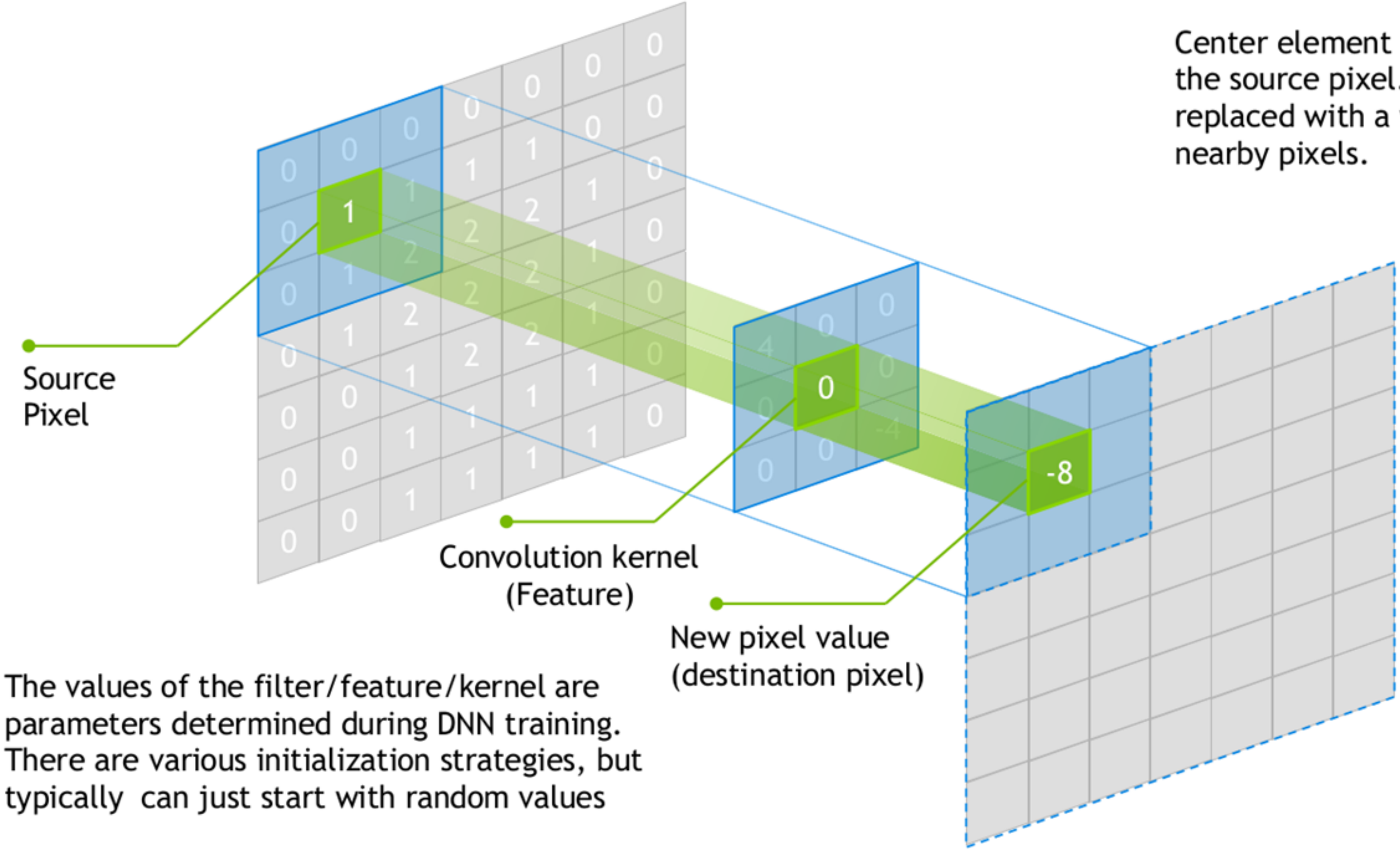
F1	F2	F3
F4	F5	F6
F7	F8	F9

```
Conv(I,F) =
  F1*I1  + F2*I2  + F3*I3 +
  F4*I7  + F5*I8  + F6*I9 +
  F7*I13 + F8*I14 + F9*I15
```

… and so on until you've covered the entire image with filter

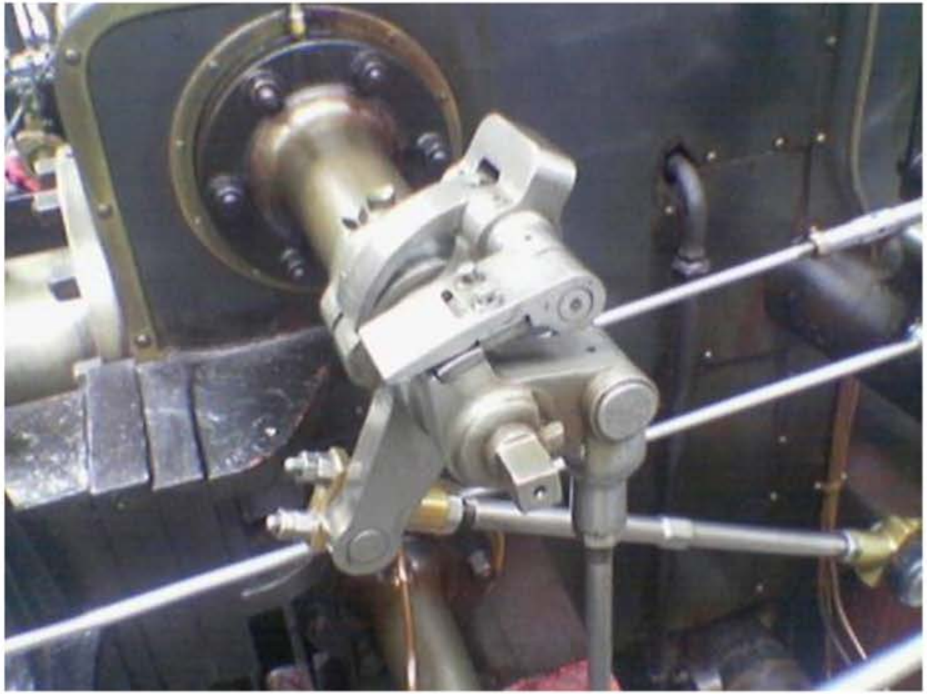
WHAT IS A CONVOLUTION?

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.



The values of the filter/feature/kernel are parameters determined during DNN training. There are various initialization strategies, but typically can just start with random values

CONVOLUTION EXAMPLE: SOBEL FILTER



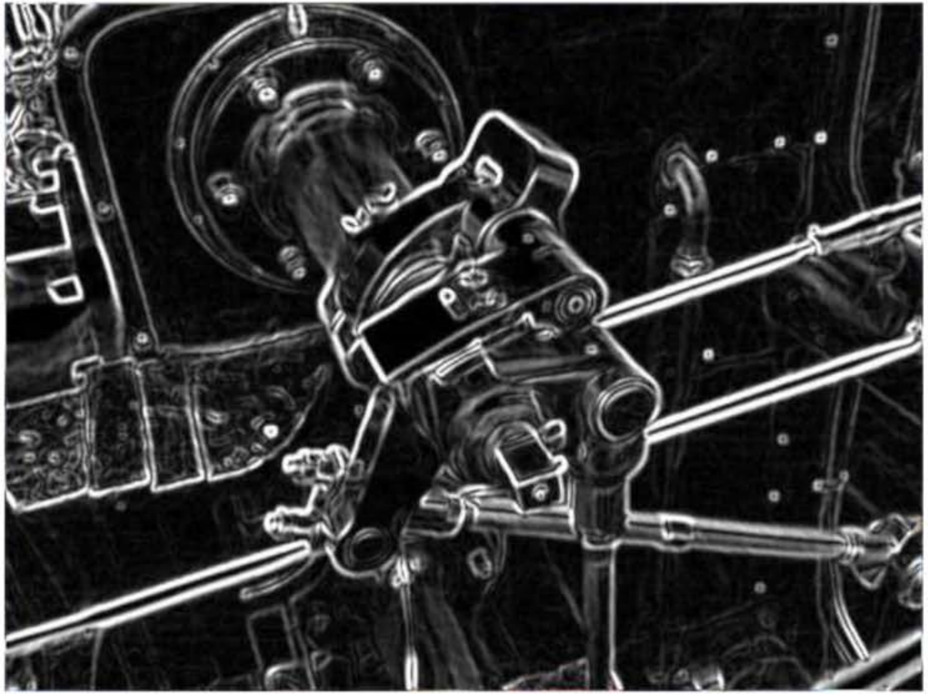
$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

Image source: https://en.wikipedia.org/wiki/Sobel_operator

CONVOLUTION EXAMPLE: SOBEL FILTER



$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$G = \sqrt{{G_x}^2 + {G_y}^2}$$

Image source: https://en.wikipedia.org/wiki/Sobel_operator