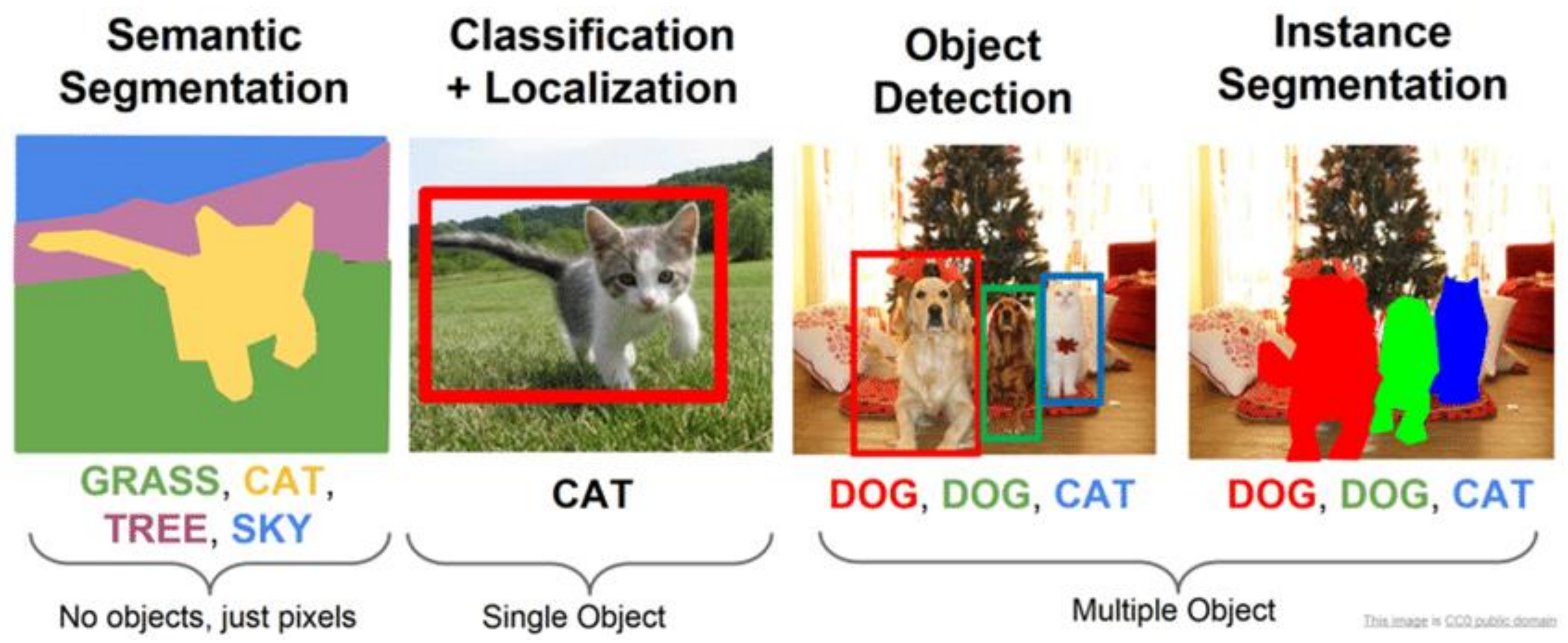
ML for Segmentation of Atmospheric Phenomenon

AI4ESS: AI for Earth System Science Summer School

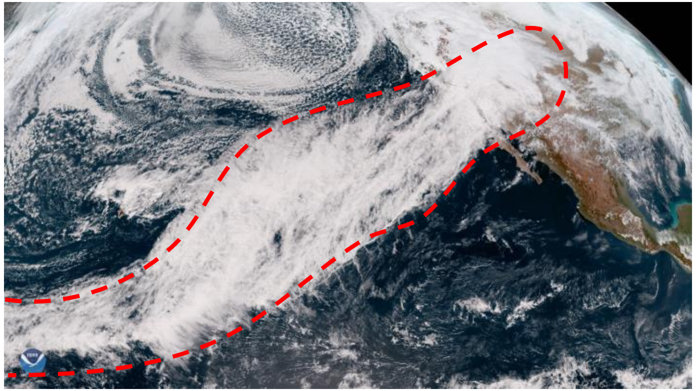
Jebb Stewart,
Christina Kumler[*], Isidora Jankov[**], David M. Hall[***]
Many, many, others

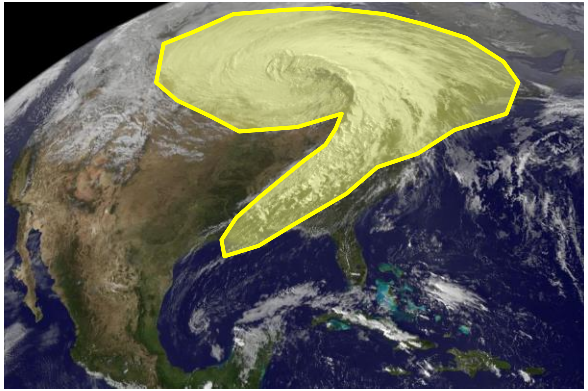
NOAA Earth System Research Laboratories (ESRL)
Global Systems Laboratory (GSL)
Boulder, CO
[*]Cooperative Institute for Research in Environmental Sciences (CIRES)
[**]Cooperative Institute for Research in the Atmosphere (CIRA)
[***]NVidia Corp

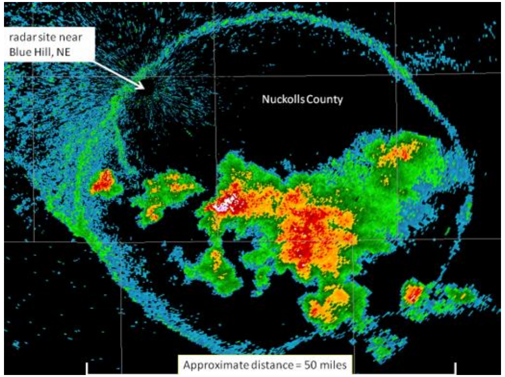
What is Segmentation?

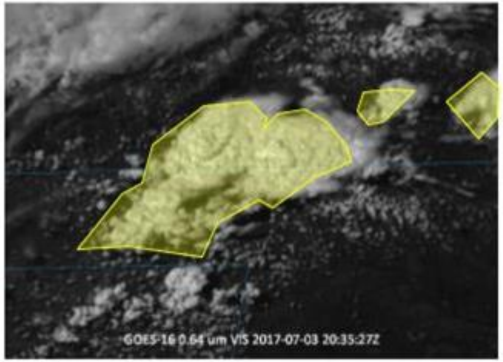


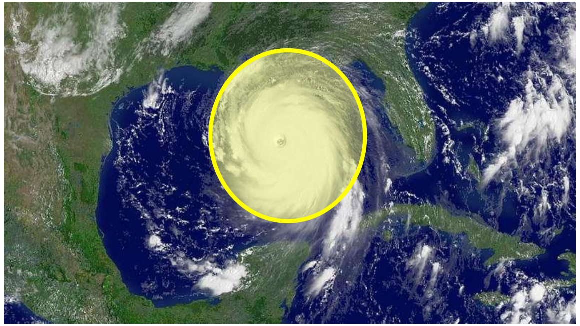
Segmentation of Atmospheric Phenomenon











Why

Identification, Classification, and Tracking

- Early warning
- Verification for specific atmospheric schemes

Automation of processes

Analytics

- Counting and comparing numbers or size of features

Targeted Data Extraction

- Identify features for further analysis

Creation of Labels

Hand Drawn

- Expert derived - Subjective
- Manually Intensive

Heuristics

- Rule based - Objective
- Can be fast
- May miss some features
- May include erroneous features

Crowdsourcing

Cat or Croissant?

Discussion on Labeling

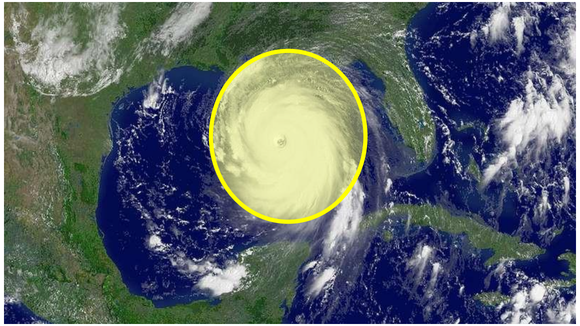
How precise do you need to be?

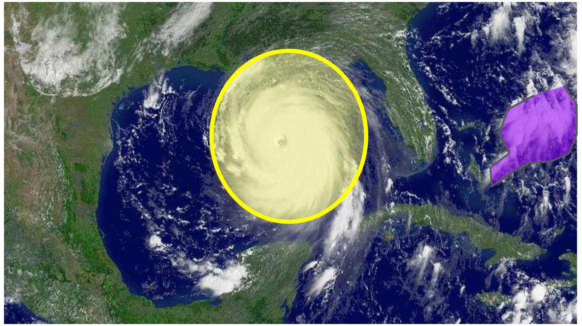
- Exact - pixel for pixel for feature
- Bounding box - general area of feature
- Disagreements between experts

If I have heuristics, why do I need Machine Learning?

- Heuristics often derived from other data sources not from target dataset
- Inference is fast, depending on algorithm, can be significantly faster

What does a label look like?





```
00000000000000000000000000
00000000000111000000000000
00000000001111100000000000
00000000011111110000000000
00000000001111100000000000
00000000000111000000000000
00000000000000000000000000
```

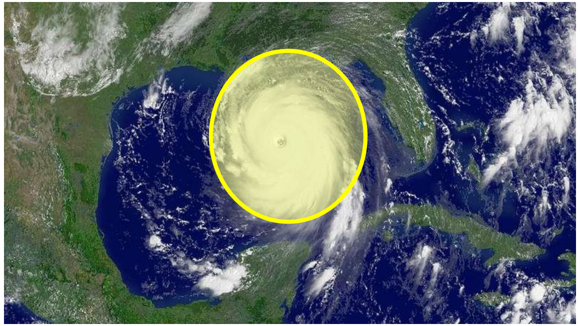
```
00000000000000000000000000
00000000000111000000000222
00000000001111100000000222
00000000011111110000000222
00000000001111100000002000
00000000000111000000000000
00000000000000000000000000
```

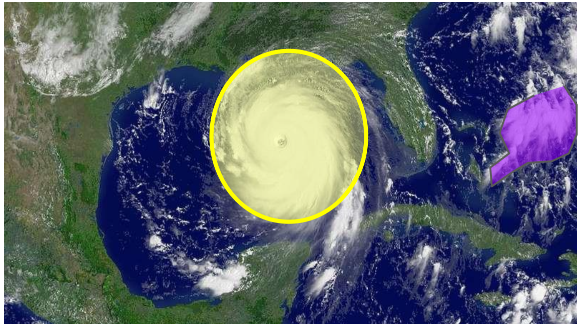
Integer Encoding Method:

0 = background
1 = Tropical Cyclone
2 = Cloud

N = Number of Classes
LABELS = (X, Y, 1)

What does a label look like?





```
00000000000000000000000000
00000000000111000000000000
00000000001111100000000000
00000000011111110000000000
00000000001111100000000000
00000000000111000000000000
00000000000000000000000000
```

```
00000000000000000000000000
00000000000111000000000222
00000000001111100000000222
00000000011111110000000222
00000000001111100000002000
00000000000111000000000000
00000000000000000000000000
```

Integer Encoding Method:

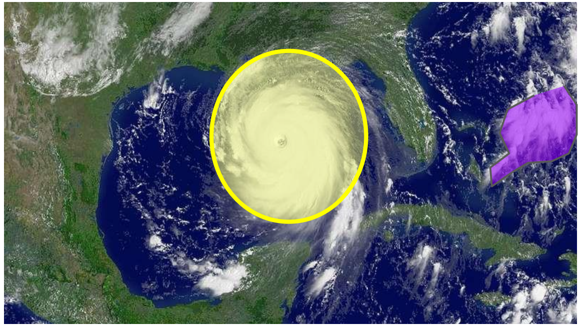
0 = background
1 = Tropical Cyclone
2 = Cloud

N = Number of Classes
LABELS = (X, Y, 1)

Problem: Model can interpret values order is meaningful and higher values could be interpreted as higher importance

One Hot Encoding

0 = background
1 = Tropical Cyclone
2 = Cloud



```
00000000000000000000000000
00000000000111000000000222
00000000001111100000000222
00000000011111110000000222
00000000001111100000002000
00000000000111000000000000
00000000000000000000000000
```

Background

```
11111111111111111111111111
11111111111000111111111000
11111111110000011111111000
11111111100000001111111000
11111111110000011111110111
11111111111000111111111111
11111111111111111111111111
```

Cyclone

```
00000000000000000000000000
00000000000111000000000000
00000000001111100000000000
00000000011111110000000000
00000000001111100000000000
00000000000111000000000000
00000000000000000000000000
```

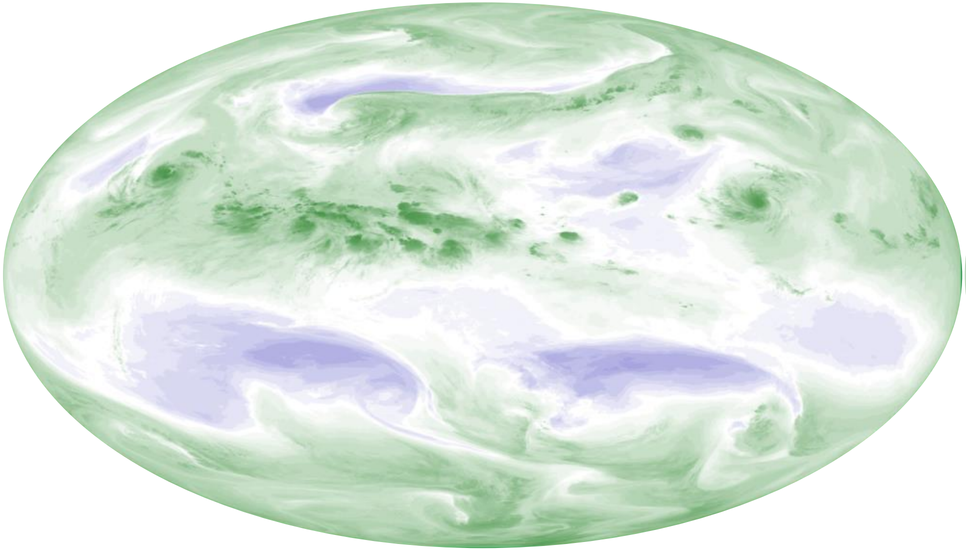
Cloud

```
00000000000000000000000000
00000000000000000000000111
00000000000000000000000111
00000000000000000000000111
00000000000000000000001000
00000000000000000000000000
00000000000000000000000000
```

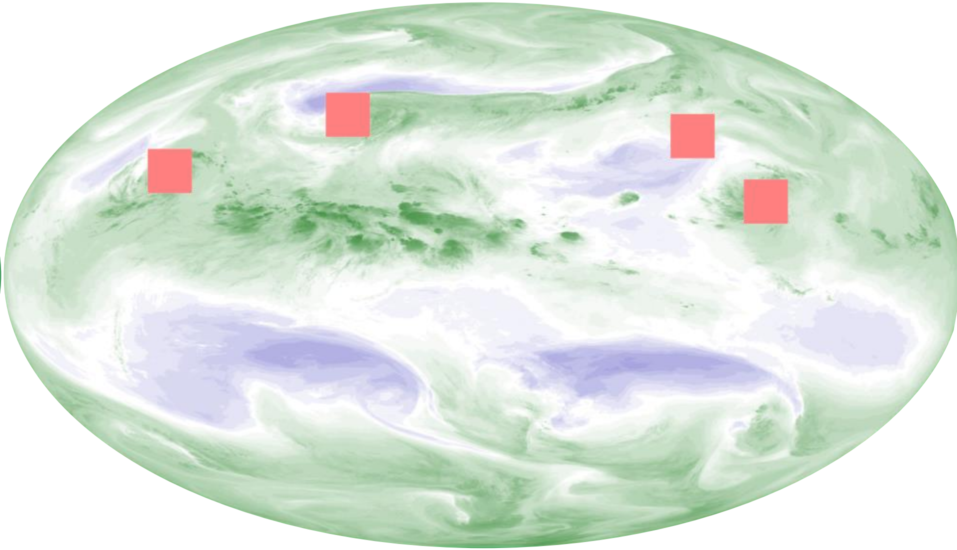
Splits integer array into an array for each class.
Many tools exists to perform this conversion.

N = Number of Classes
Label = (X, Y, N)

Dataset Challenges

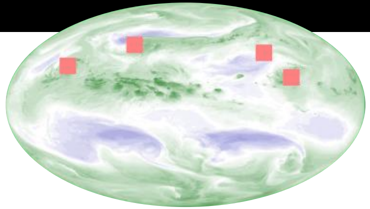


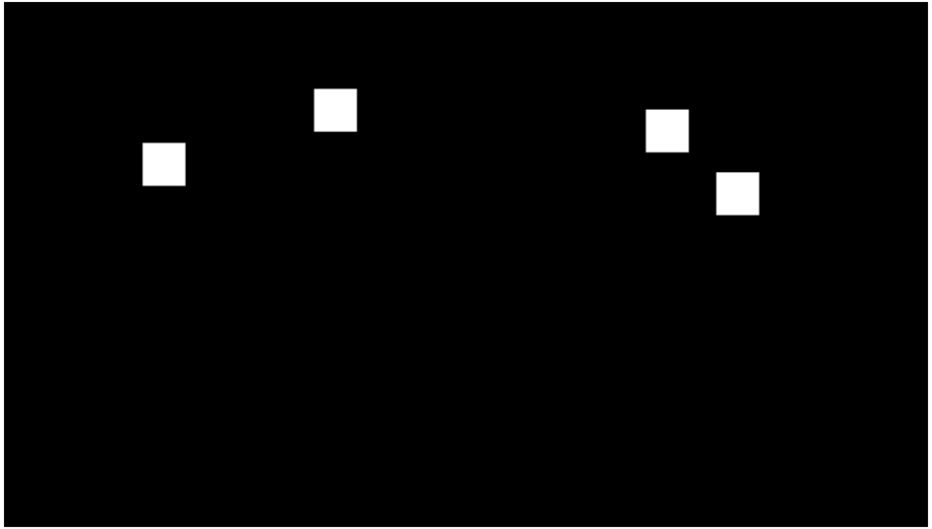
Water Vapor Image



Water Vapor Image with
Tropical Cyclone Labels

Dataset Imbalance





Truth (White are Labeled Cyclones)

Prediction from Model

= 95% Accuracy

Working with Dataset Imbalances

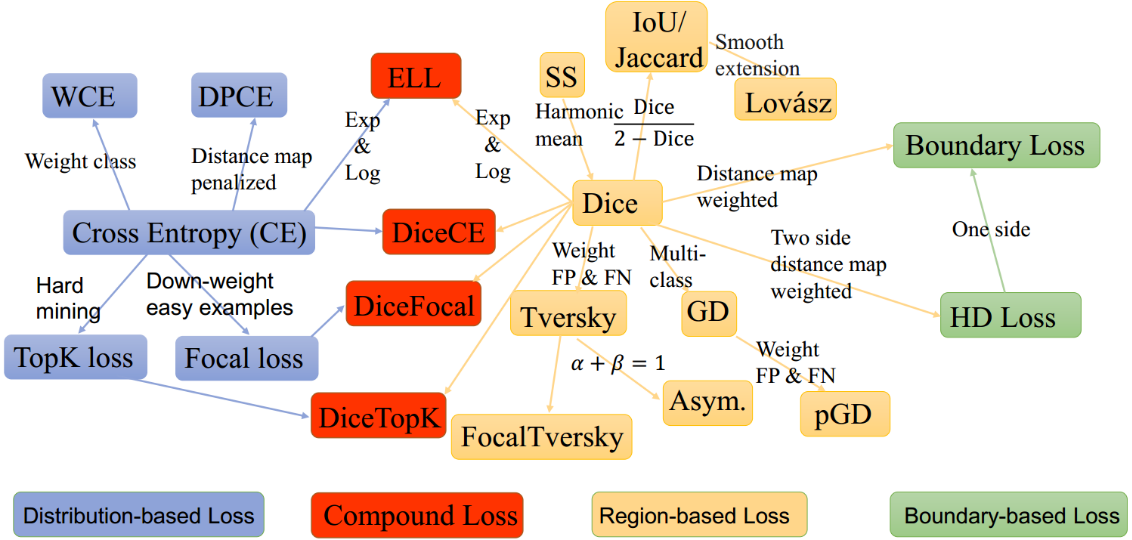
Image Processing

Sampling Techniques

- Undersample majority class
- Oversample minority class

Modify Labels

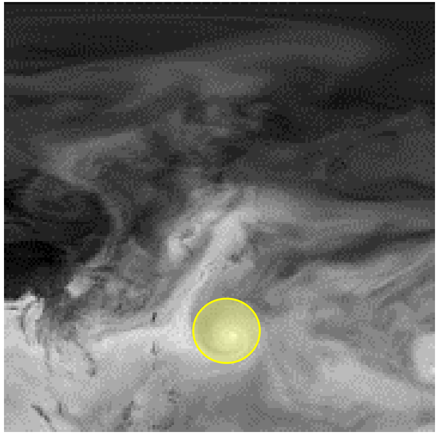
Different Loss Functions

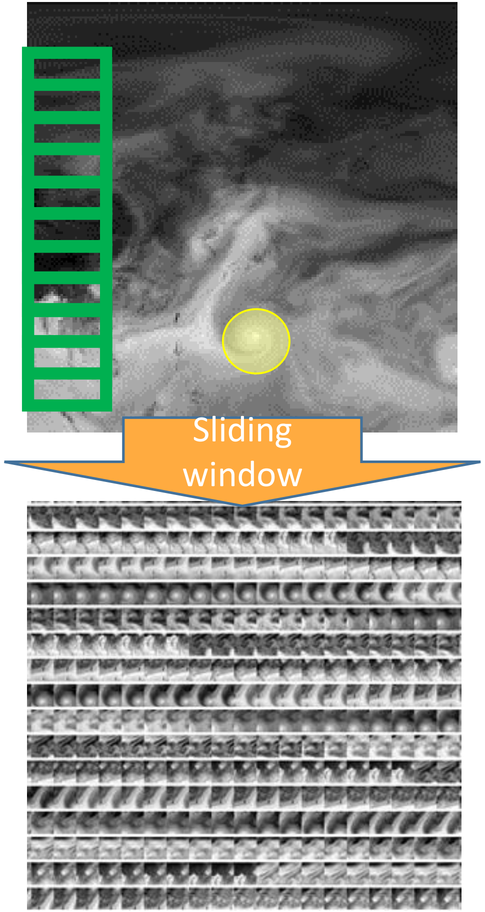


Source: https://github.com/JunMa11/SegLoss

Image Processing

Sliding Window Technique

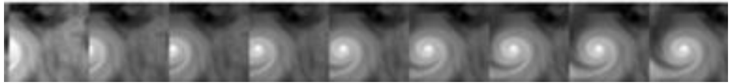




Training dataset can now contain more equitable distribution of both positive and negative segmentation

Potential Downsides:

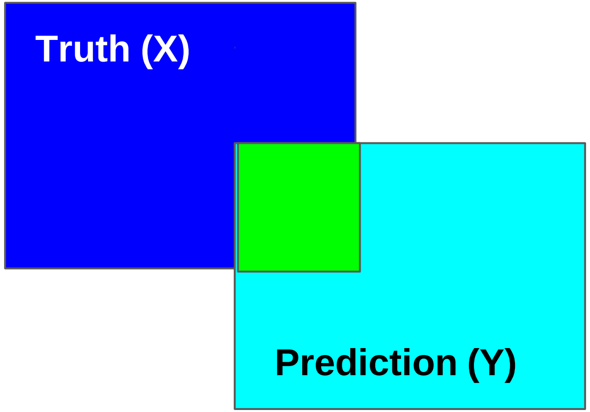
- More processing
- Not efficient
- Convolutional Layers are doing this internally

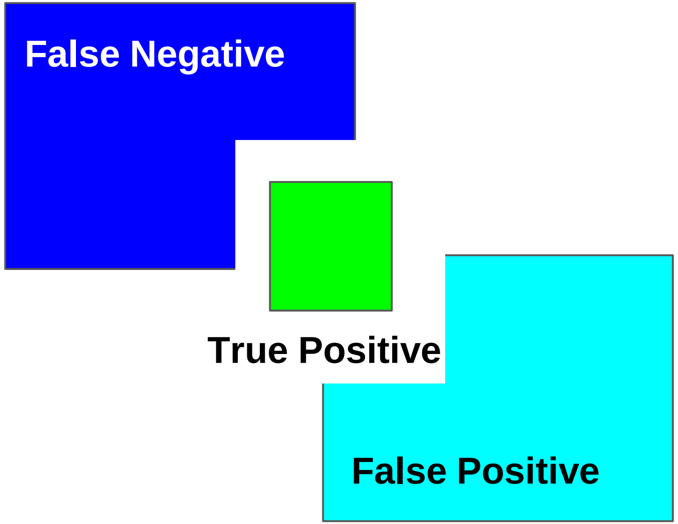


Loss Function - Dice Coefficient

$$\text{Dice Coefficient} = \frac{2 * |X \cap Y|}{|X| + |Y|}$$

DC = 1.0 Identical Image
DC = 0.0 No overlap





Dice Coefficient in Code

```
# using keras

def dice_coeff(y_true, y_pred):
   smooth = 1.
   y_true_f = K.flatten(y_true)
   y_pred_f = K.flatten(y_pred)
   intersection = K.sum(y_true_f * y_pred_f)
   score = (2. * intersection + smooth) / (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)
   return score

def dice_loss(y_true, y_pred):
   return (1 - dice_coeff(y_true, y_pred))

# smooth variable helps optimizer and avoids division by zero
```

Loss Function - Tversky Coefficient

$$\text{Tversky Coefficient} = \frac{2*|X\cap Y|}{(|X\cap Y| + \alpha|X-Y|+\beta|Y-X|)}$$

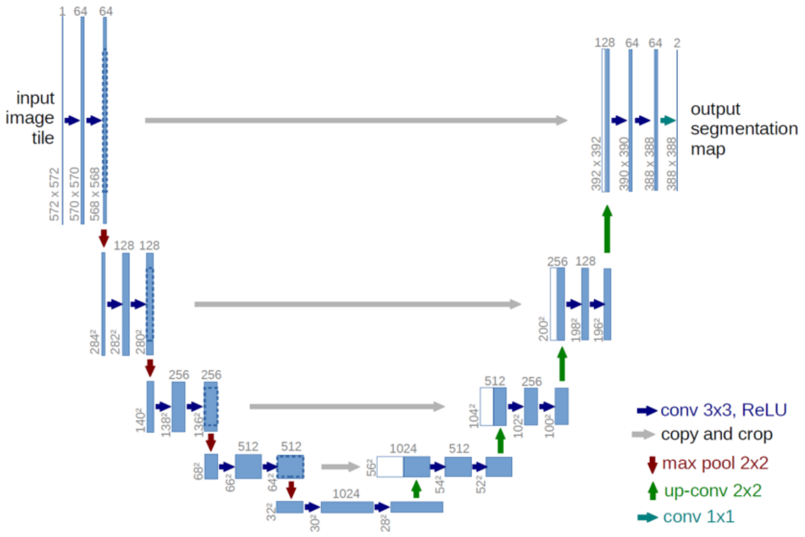
$\alpha < \beta$ penalizes false negatives more

```
def tversky_coeff(alpha=0.3, beta=0.7, smooth=1e-10):

    def tversky(y_true, y_pred):
       y_true = K.flatten(y_true)
       y_pred = K.flatten(y_pred)
       truepos = K.sum(y_true * y_pred)
       fp_and_fn = alpha * K.sum(y_pred * (1 - y_true))
                  + beta * K.sum((1 - y_pred) * y_true)
       return (truepos + smooth) / ((truepos + smooth)
                    + fp_and_fn)
    return tversky
```

Neural Network Structures - U-Net

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

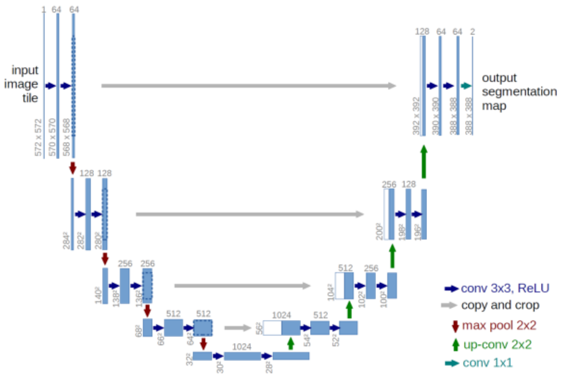


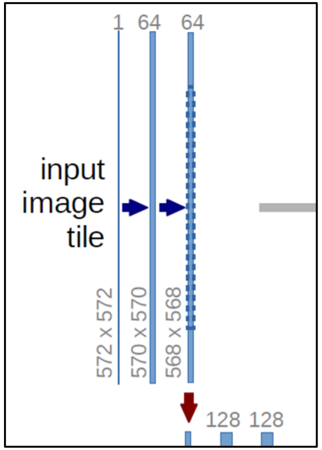
U-Net in Code

```
def conv2d_block(input_tensor, n_filters=64, kernel_size=3, layers=2, batchnorm=True):
    X = input_tensor

    for l in range(0,layers):
       x = Conv2D(filters=n_filters, kernel_size=(kernel_size, kernel_size), kernel_initializer="he_normal",
padding="same")(x)
       if batchnorm:
          x = BatchNormalization()(x)
       x = Activation("relu")(x)

    return x
```






```
Single Channel Input
INPUT = (BATCH_SIZE, 572, 572, 1)

If you had RBG (ie 3 Channels)
INPUT = (BATCH_SIZE, 572, 572, 3)
```

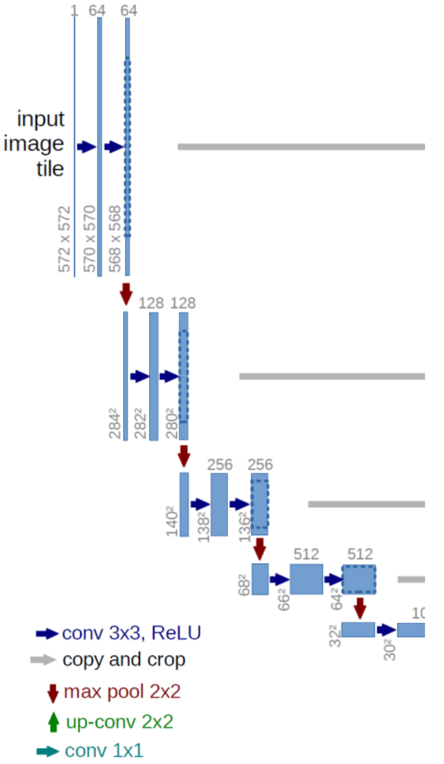
U-Net in Code

```
# contracting path
# Block 1
c1 = conv2d_block(input_img, n_filters=n_filters*1, kernel_size=3, layers=2,
                  batchnorm=batchnorm)
p1 = MaxPooling2D((2, 2)) (c1)
p1 = Dropout(dropout*0.5)(p1)

# Block 2
c2 = conv2d_block(p1, n_filters=n_filters*2, kernel_size=3, batchnorm=batchnorm)
p2 = MaxPooling2D((2, 2)) (c2)
p2 = Dropout(dropout)(p2)

# For the depth of U-Net, repeat for each block c3, c4
# increasing multiplier on n_filters by factor of 2 each block

c5 = conv2d_block(p4, n_filters=n_filters*16, kernel_size=3, batchnorm=batchnorm)
```



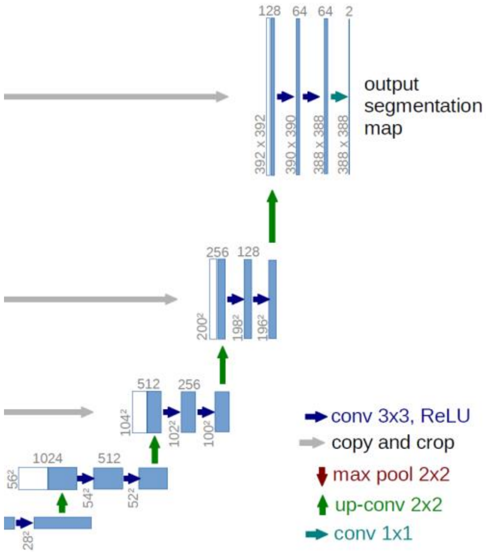
U-Net in Code

```
# expansive path
u6 = Conv2DTranspose(n_filters*8, (3, 3), strides=(2, 2), padding='same') (c5)
u6 = concatenate([u6, c4])
u6 = Dropout(dropout)(u6)
c6 = conv2d_block(u6, n_filters=n_filters*8, kernel_size=3, layers=2, batchnorm=batchnorm)

u7 = Conv2DTranspose(n_filters*4, (3, 3), strides=(2, 2), padding='same') (c6)
u7 = concatenate([u7, c3])
u7 = Dropout(dropout)(u7)
c7 = conv2d_block(u7, n_filters=n_filters*4, kernel_size=3, batchnorm=batchnorm)

# For the depth of U-Net, repeat for each block c8, c9
# decreasing multiplier on n_filters by factor of 2 each block

##
outputs = Conv2D(output_channels, (1, 1), activation='sigmoid') (c9)
model = Model(inputs=inputs, outputs=outputs)
```



Discussion on Neural Nets for Segmentation

Things to try:

- Use Gaussian Noise versus Dropout
- Vary Number of Filters (n_filters) value
- Keep consistent Number of Filters (n_filters) between blocks
- Vary Depth of U-Net
- Vary final Activation

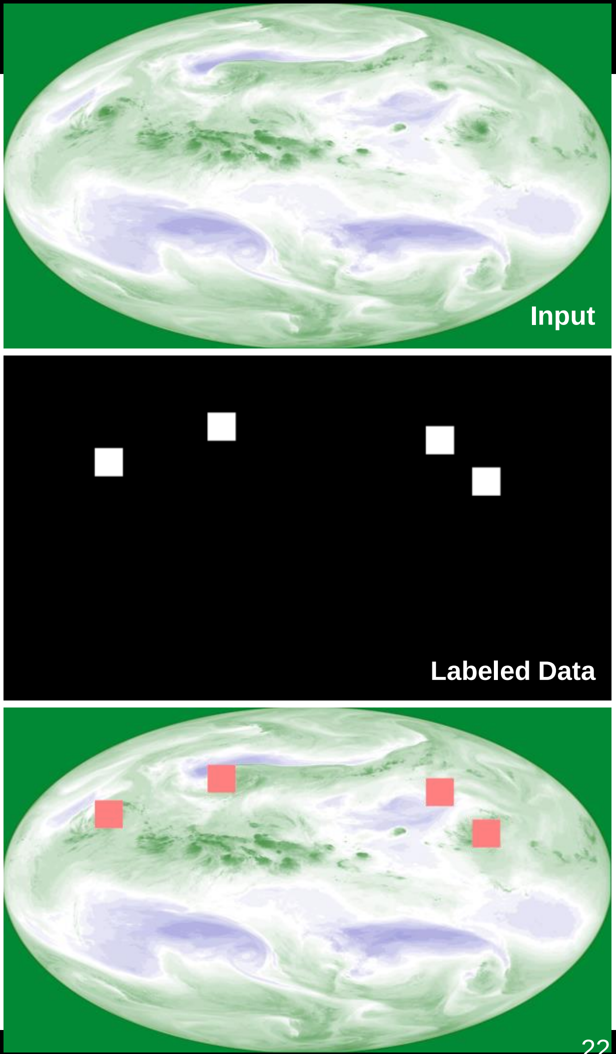
Things to consider:

- Depth and Number of Filters impact memory usage
- U-Net one of many deep neural networks for image segmentation

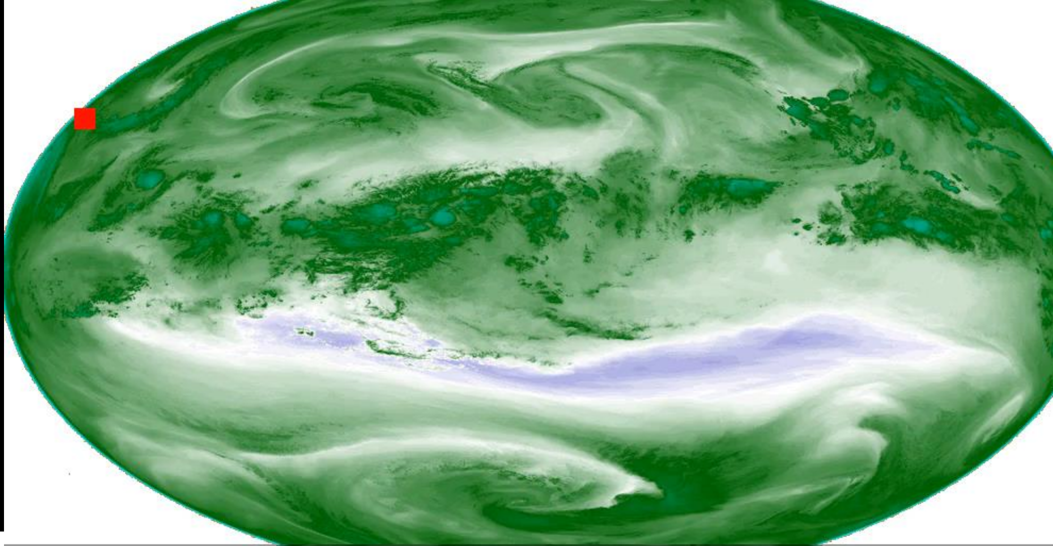
U-Net in Action - Tropical Cyclones

- Water Vapor Channel from GOES Imager (Previous Generation)
- Storm centers from IBTracks Dataset
- Data for 2008 through 2016
- Image segmentation 25x25 pixel box segmentation centered on storm
- Only used storms classified as Tropical Storm or greater on Saffir Simpson Scale
    - 34 knots and above

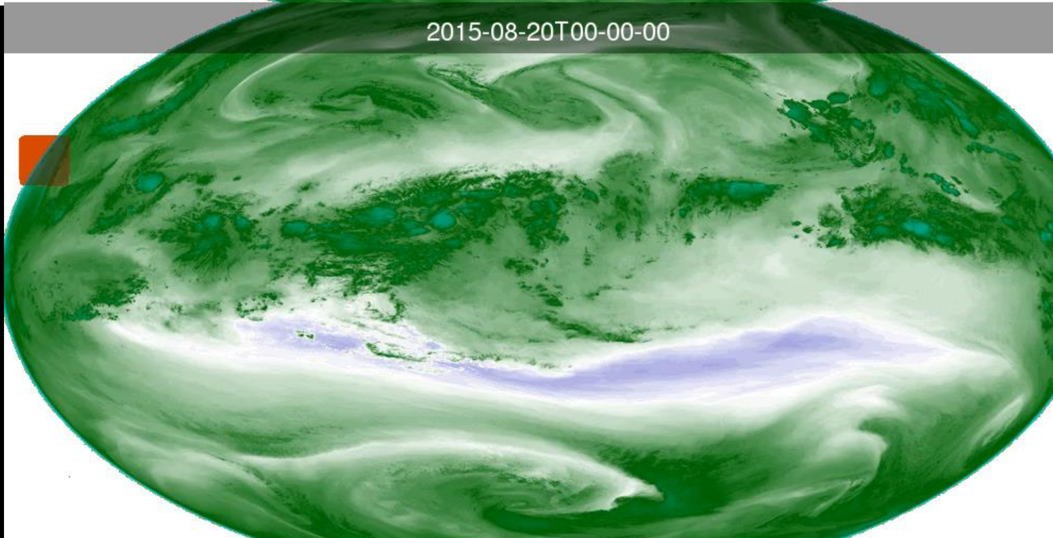
~ 10,000 Labeled Data

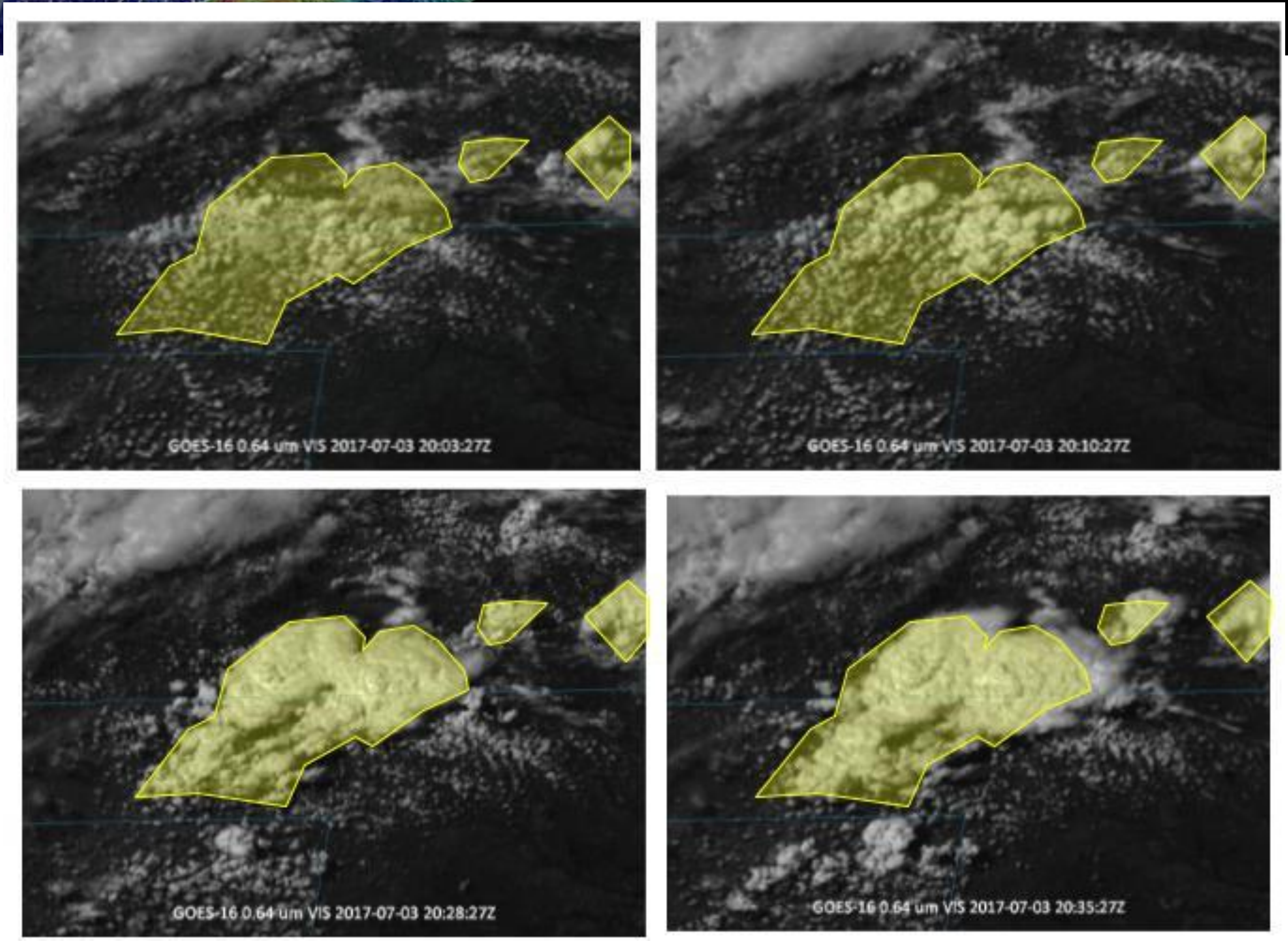


Manual Labels



Automatic Labels from
Trained Neural Network

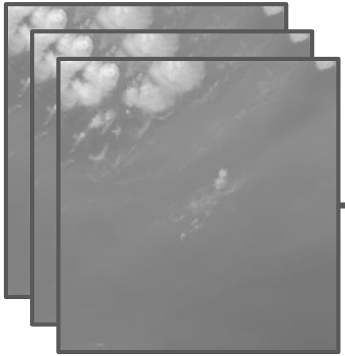


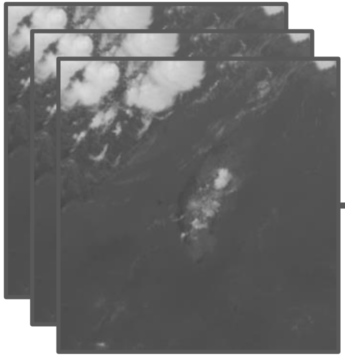


Goal - Neural Network for Automatic Detection and Labeling of Convection Initiation Areas

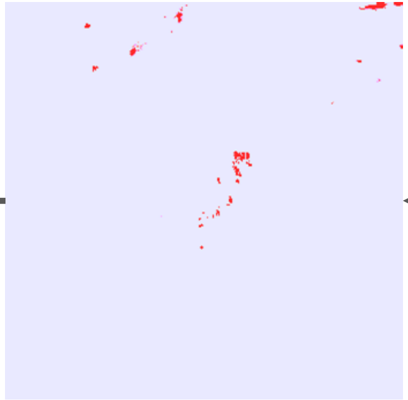
How Data is Used - 90 Minute Lead Time

T = Time for Prediction



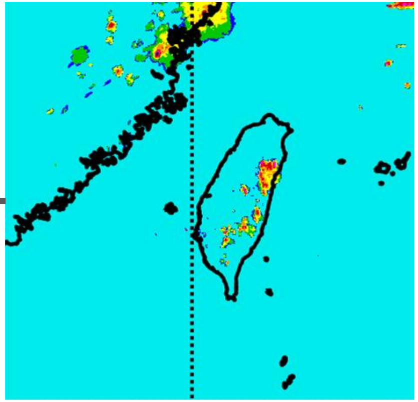


T - 110, T - 100, T - 90 (minutes)
Band 9, Band 15



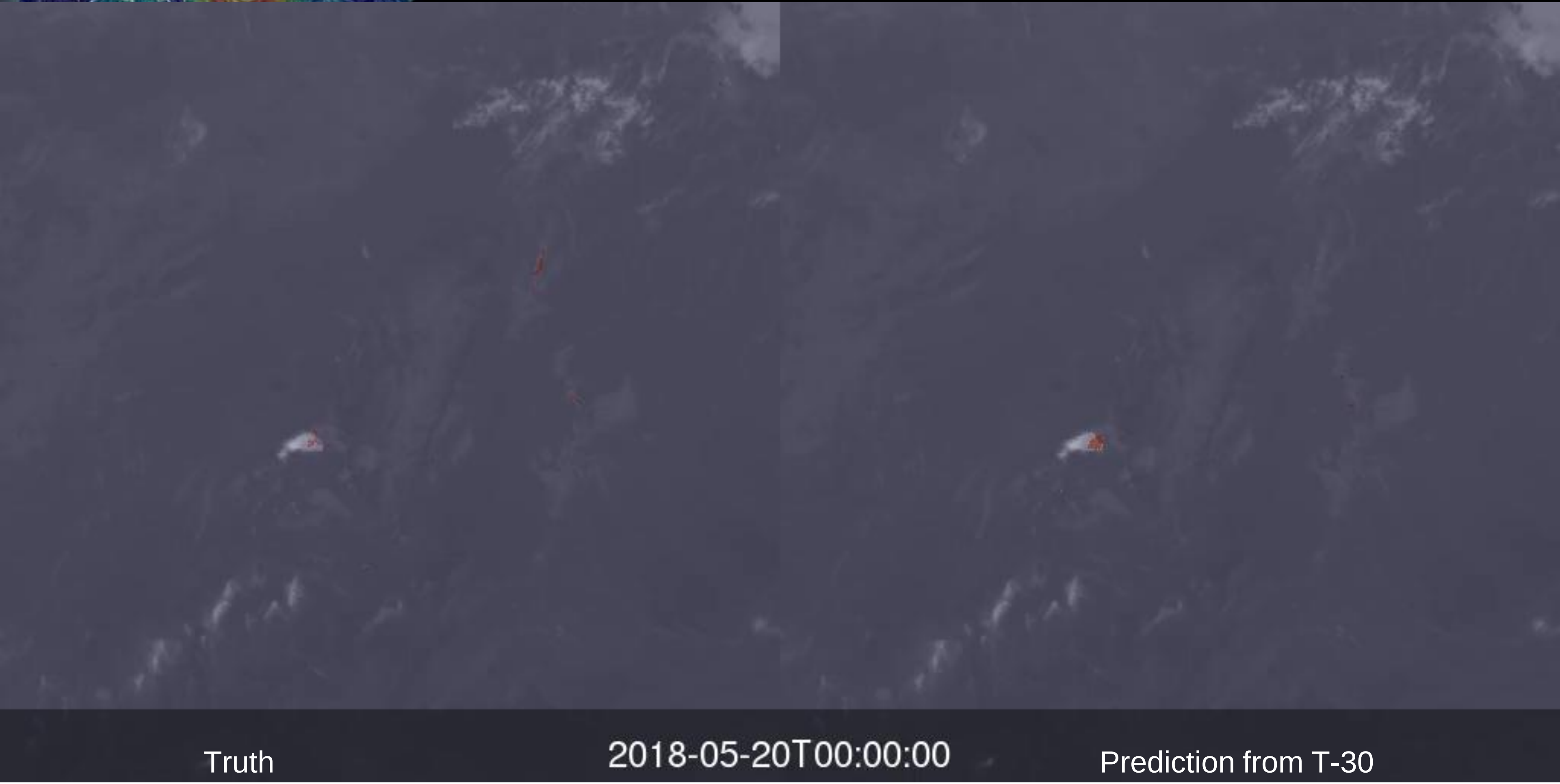
T
Binary (yes/no)
Composite Reflectivity > 35 DBz

Neural Network:
6 Inputs (Satellite Only)
1 output (Binary Mask where DBz > 35)



T
Composite Reflectivity

Prediction from T-30



Prediction from T-90

Summary

Understand the problem you are trying to solve first

- The tools you need vary on the solution you are looking for
- Many different tools in the toolbox

Labeled data can be challenging

- Not always an agreement - our objects can have fuzzy

Dataset Imbalance can skew results

- Be aware and evaluate random samples

Field is still rapidly evolving

- Exciting and it can be difficult to keep up

Thanks!

Questions?

Jebb.Q.Stewart@noaa.gov

Resources and References

Learning:
https://www.tensorflow.org/tutorials/images/segmentation
https://towardsdatascience.com/fastai-image-segmentation-eacad8543f6f
https://medium.com/analytics-vidhya/pytorch-implementation-of-semantic-segmentation-for-single-class-from-scratch-81f96643c98c
https://www.jeremyjordan.me/semantic-segmentation/

Data:
https://www.ncdc.noaa.gov/ibtracs/
https://www.bou.class.noaa.gov/saa/products/search?datatype_family=GVAR_IMG

Papers:
U-Net: Convolutional Networks for Biomedical Image Segmentation, Ronneberger, O. (2015)
https://arxiv.org/pdf/1505.04597.pdf
Evolution of Image Segmentation using Deep Convolutional Neural Network: A Survey,Sultana et al (2020),
https://arxiv.org/abs/2001.04074