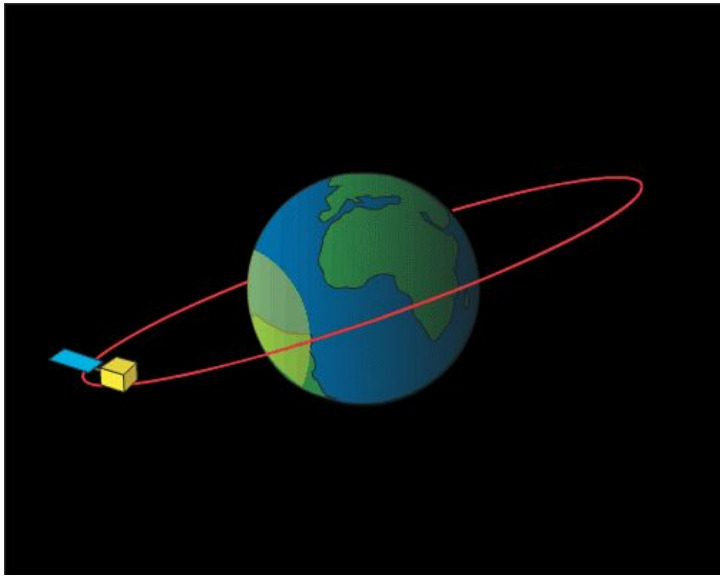
Geostationary Cloud Tracking

Jonathan Kim, Kyler Norton, AJ Geckle

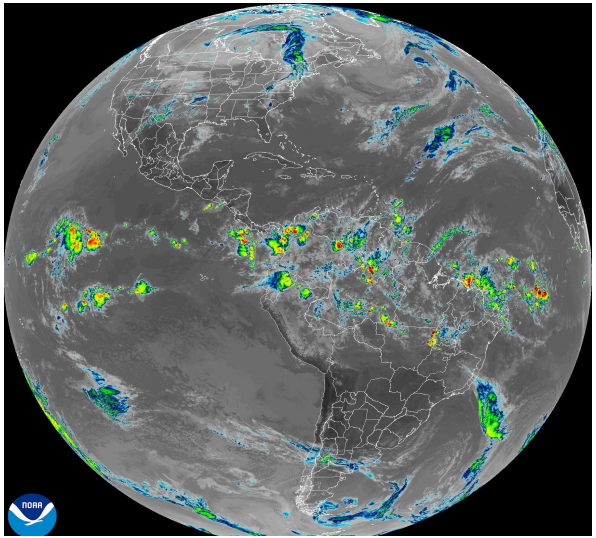
Geostationary Satellites

- Geostationary satellites are in orbit about 22,000 miles above the Earth's equator and spin at the same rate of the Earth
- This allows them to continuously gather data from the same location on Earth



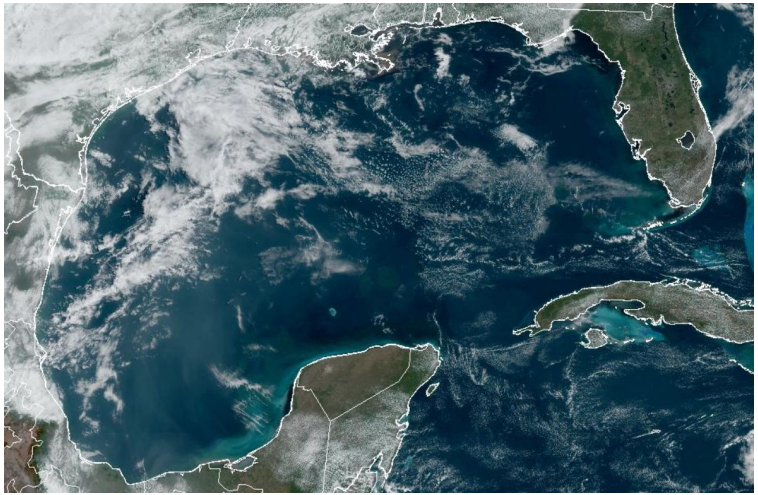
Background

- Geostationary satellites take data about every 30 minutes under normal conditions
- NOAA GOES-16 Satellite takes data every 10-15 minutes
- Captures data across 16 visual/reflective, near-infrared, and infrared bands



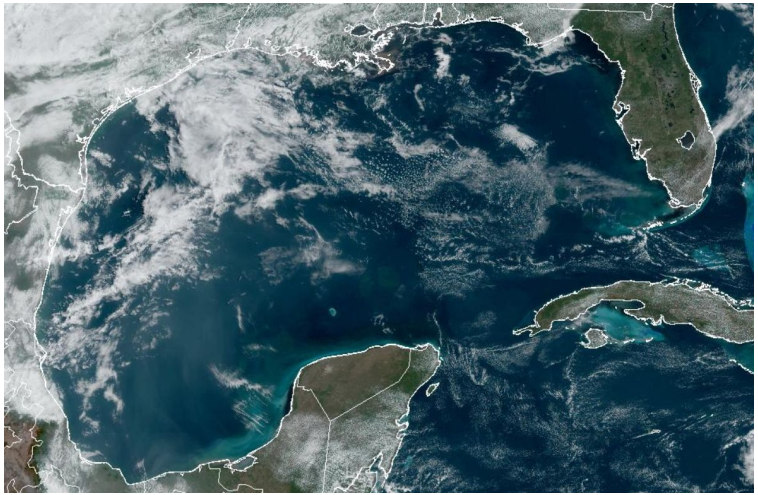
Goal

Use Machine Learning for cloud detection and tracking in the Gulf of Mexico using NOAA's Geostationary Operational Environmental Satellite GOES-16



Objectives

- Create a predictive model that can fill in the missing satellite information between snapshots
- Then use predictive model for future cloud movement prediction

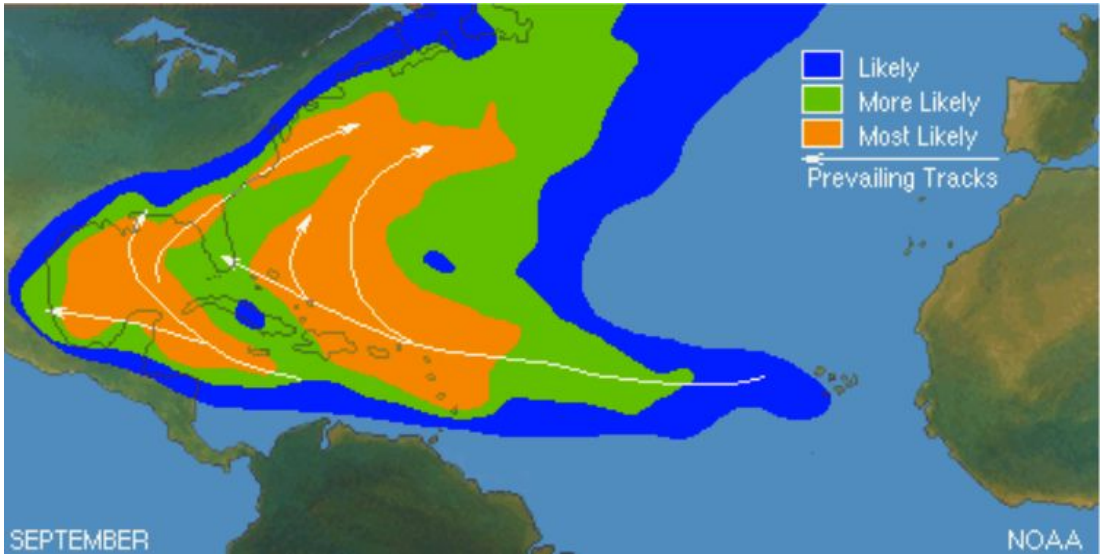


Project Motivation

- Fill in missing satellite data for more accurate weather modeling and forecasting
- Use this to track hurricanes to warn people sooner of the hurricane's path

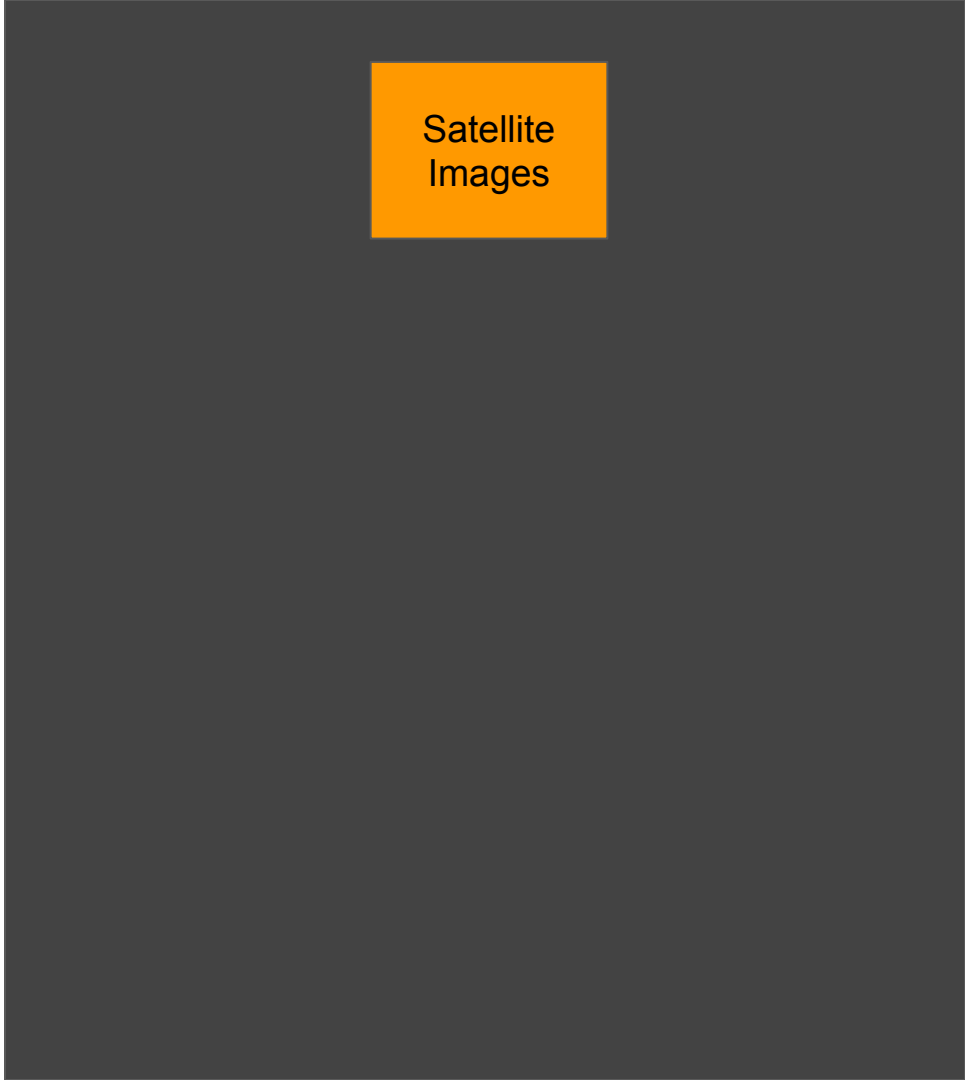
Why the Gulf of Mexico?

- Hot spot for hurricanes
- Hurricanes “funnel” through Mexico and Cuba
- GOES-16 only takes images of the Western Hemisphere



Data Pipeline

- Get satellite images



Earth Engine Code Editor

Pre-processing was initially done in the Earth Engine Code Editor, a Javascript web-based IDE for the Earth Engine API

- Integrated data catalog: Direct access to large catalog of satellite imagery and geospatial datasets allowing easy implementation of GOES or SENTINEL-2 imagery preprocessing.
- Scalability: GEE Platform handles computational load on Google servers
- Real-time data access: Beneficial for time-sensitive projects like monitoring current tropical storm movement

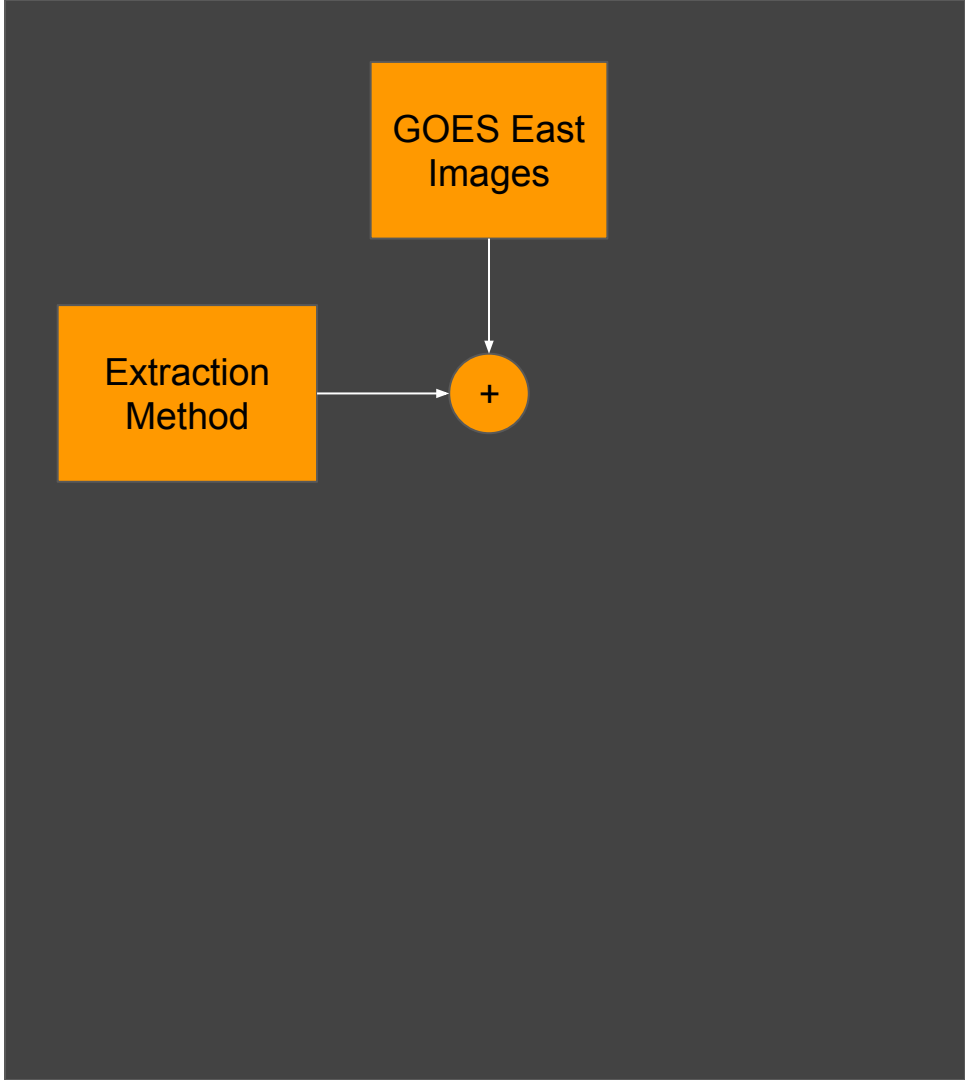
Google Colab

Pre-processing was eventually moved to Google Colab

- Machine Learning Library Support: Python supports all popular machine learning libraries such as TensorFlow, PyTorch, and Keras
- Improves Exporting Time: Easier and more efficient to import/export data directly within Google Drive

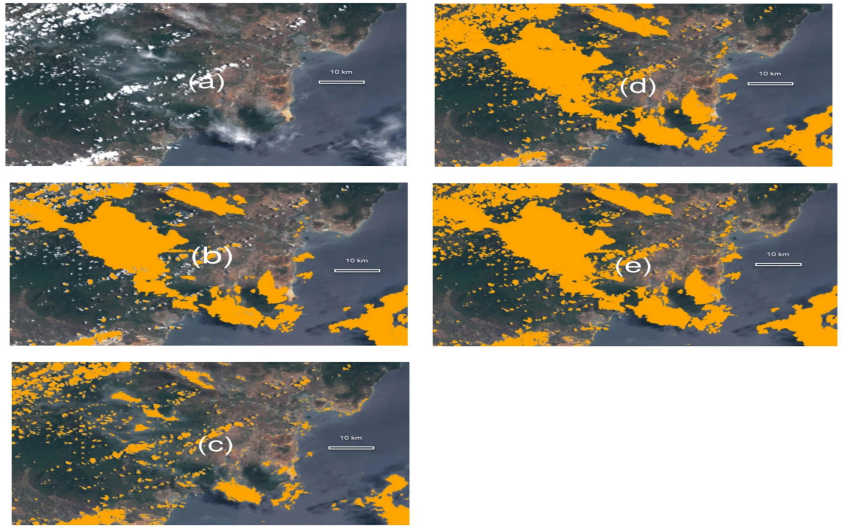
Data Pipeline

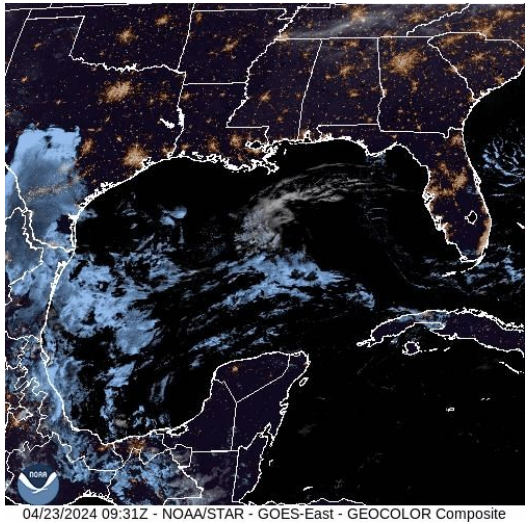
- Receive GOES East images
- Extract clouds



Cloud Extraction

- Additive Masking vs Background Subtraction
- Building a background would prove difficult and time intensive





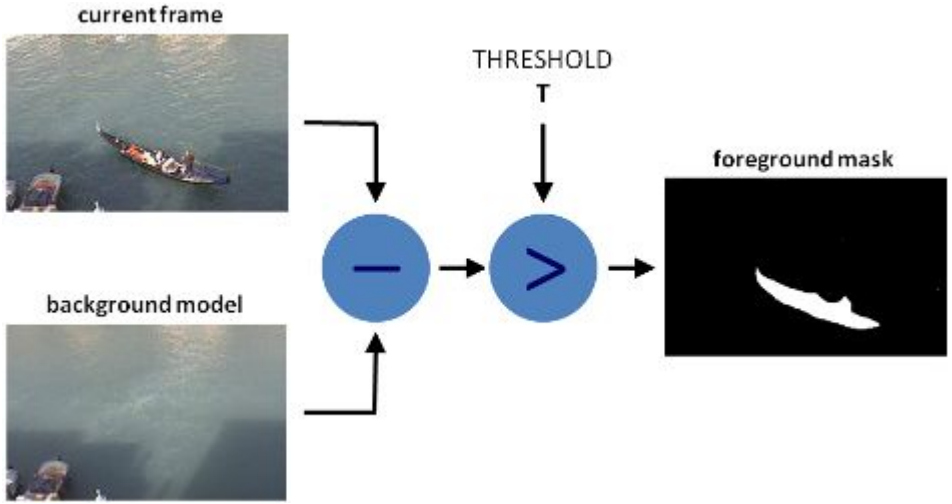


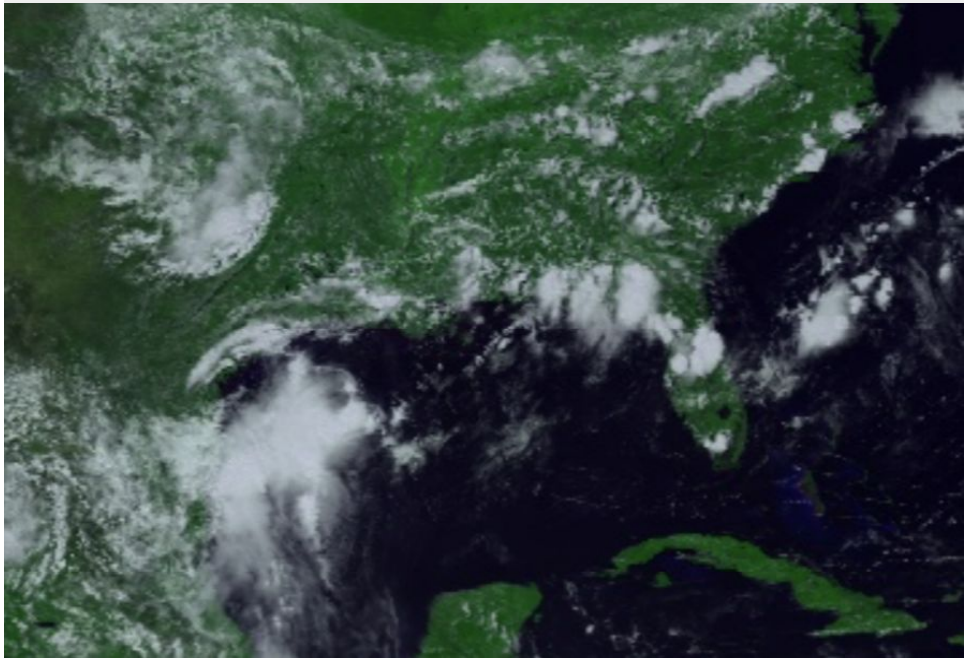
Image Bands

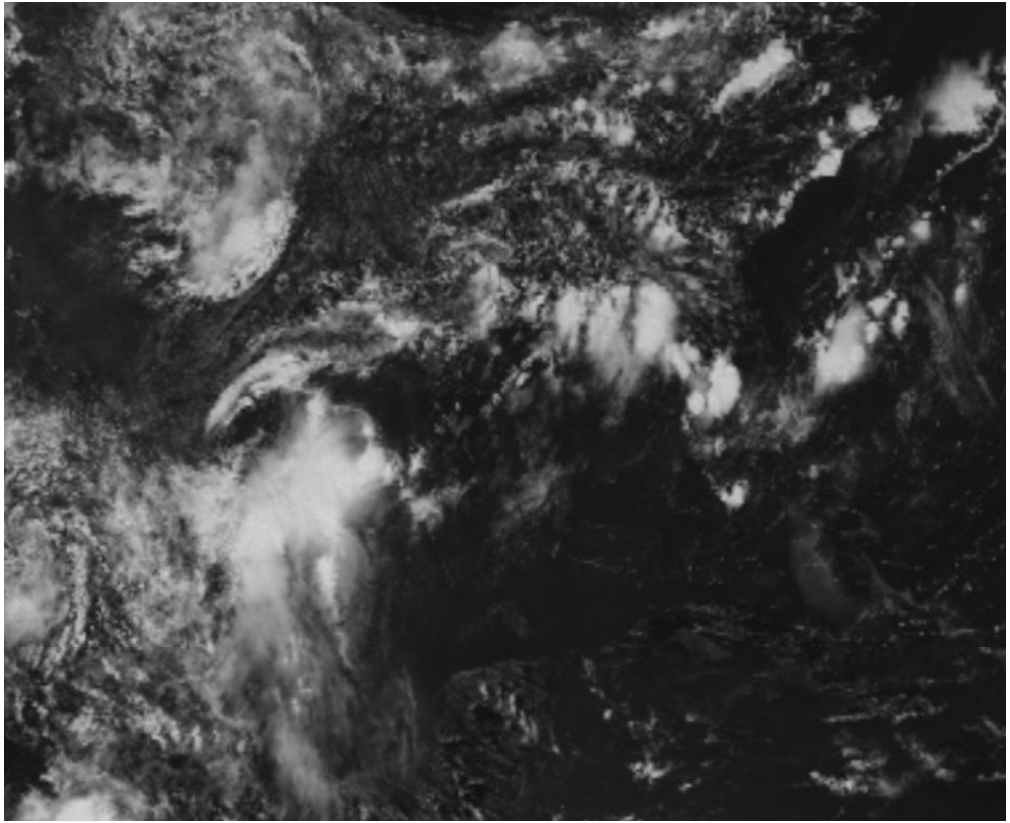
Three Types of Bands:

Visible (450 - 690 nm)

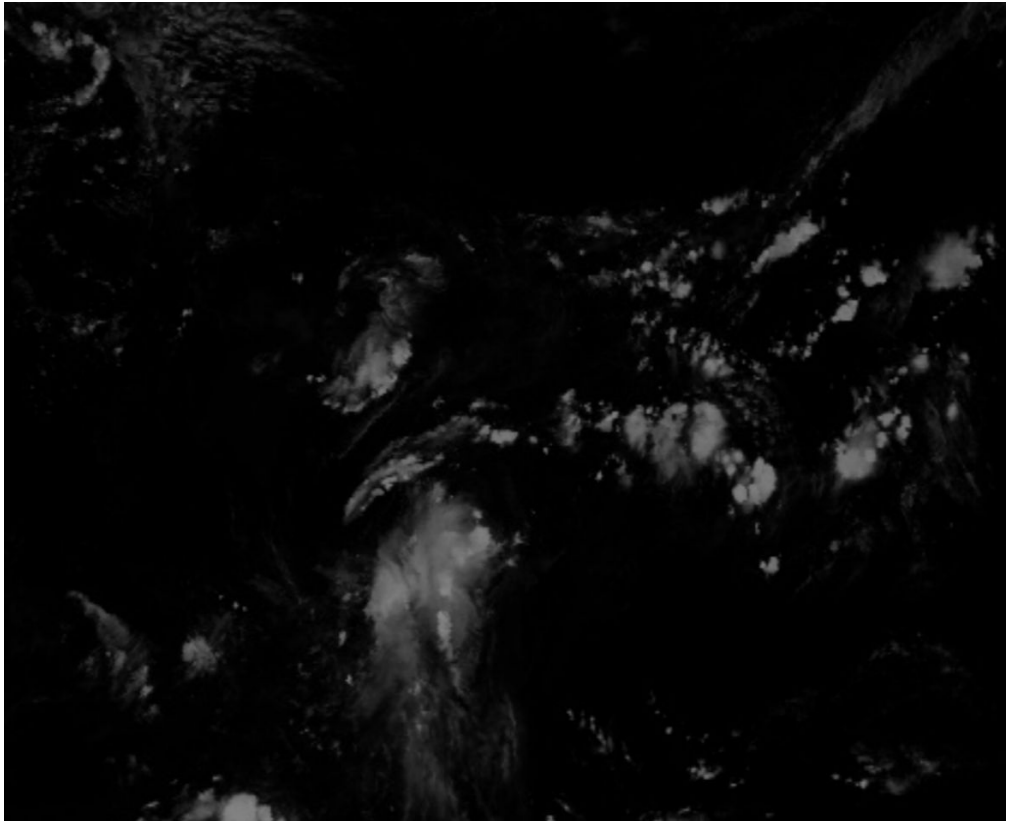
Near-Infrared (846 nm - 2.275 μm)

Infrared (3.80 - 13.6 μm)

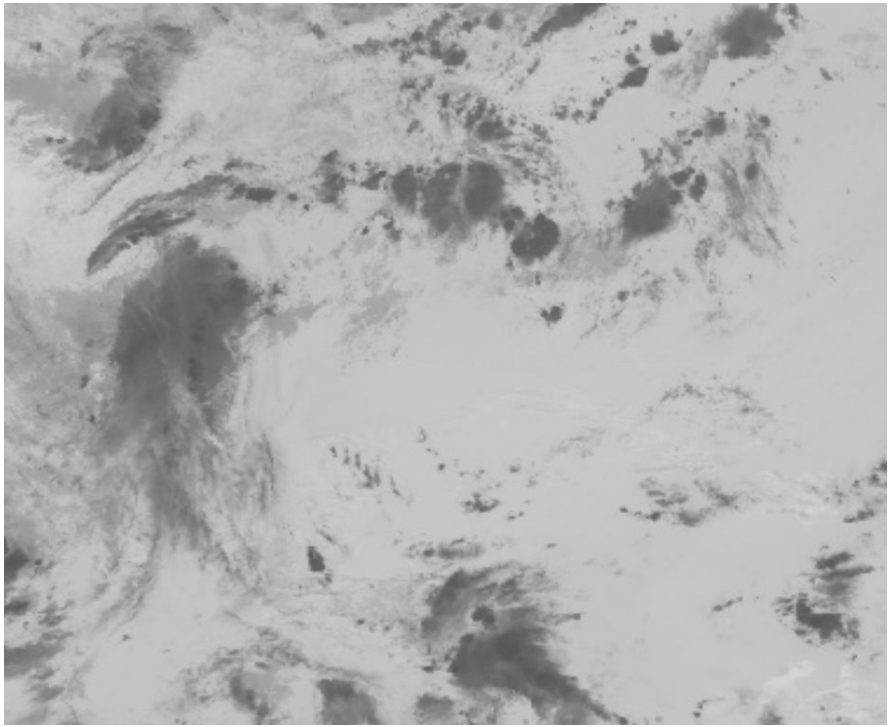


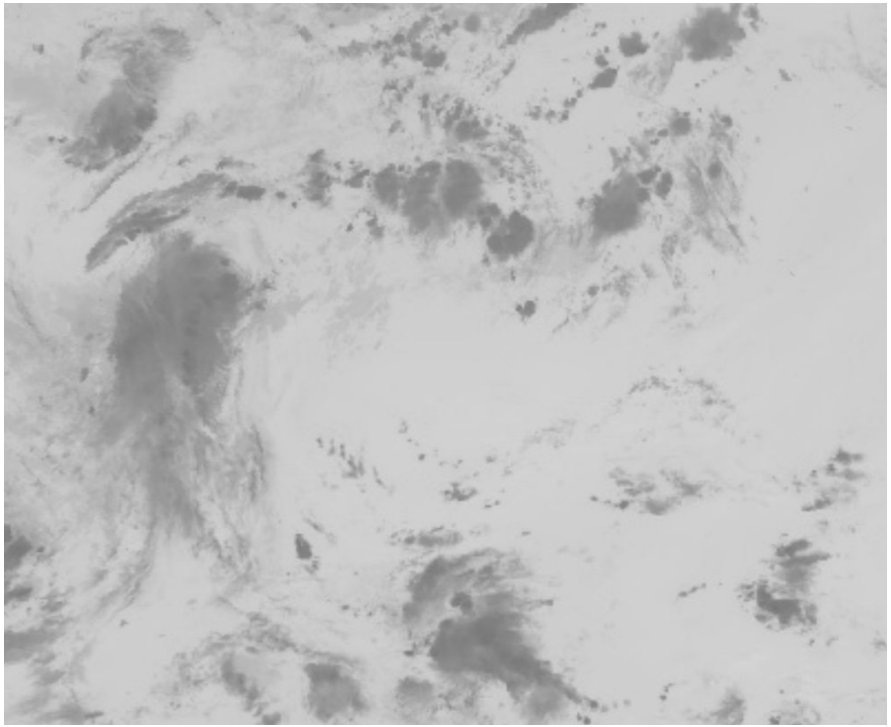


Visible Bands: Visual Blue

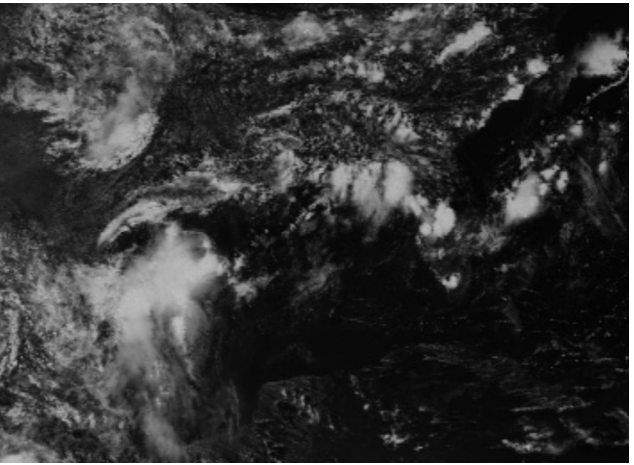


Near-Infrared Bands: Cirrus

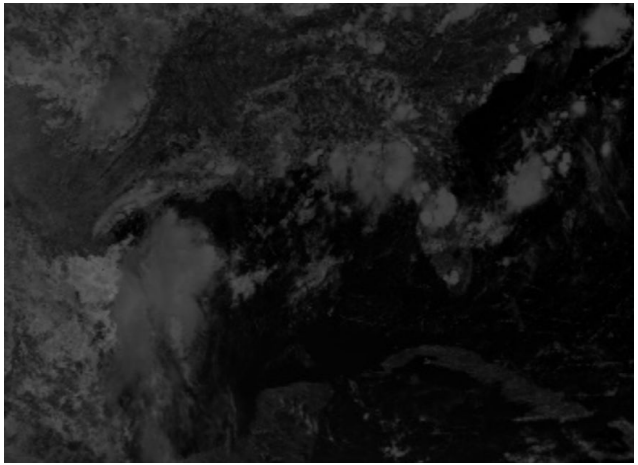




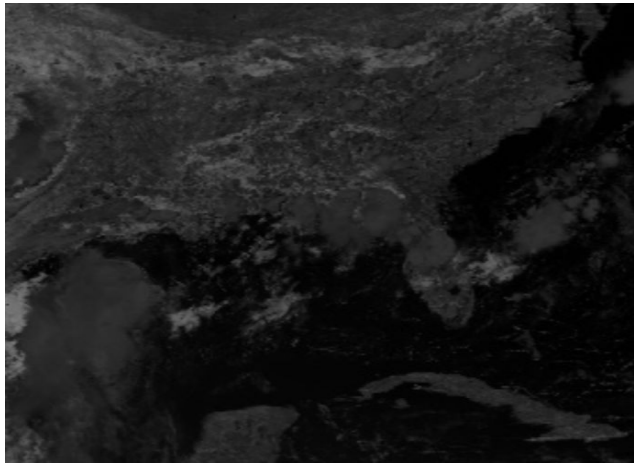
Infrared Bands: Cloud Top Phase, CO2 Longwave



Visual: Red

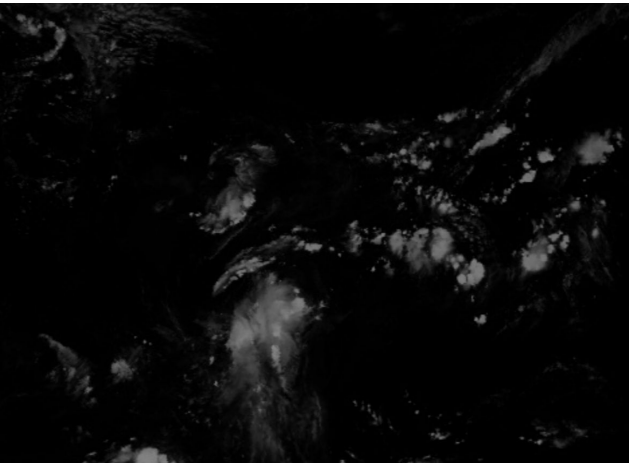


Near-IR: Cloud Particle Size

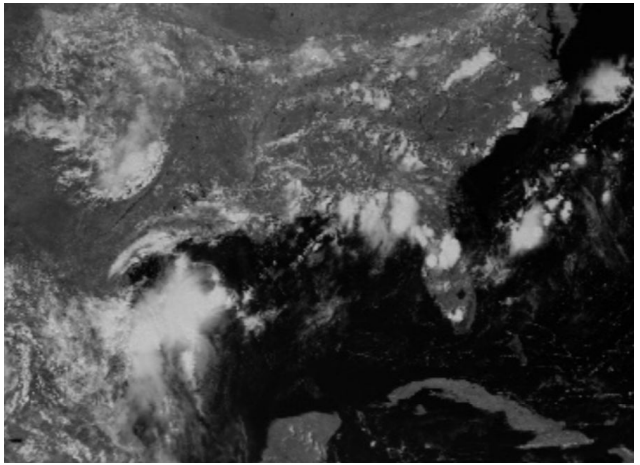


Near-IR: Snow/Ice

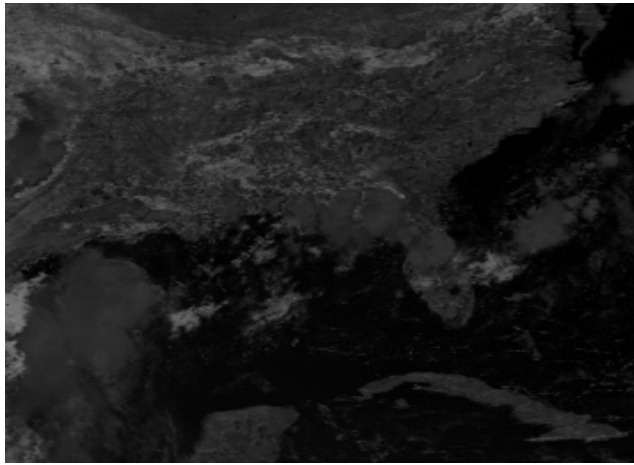
Undemonstrated Bands



Near-IR: Cirrus

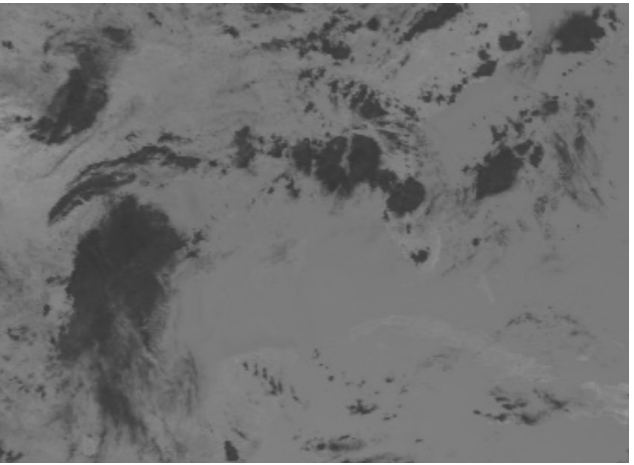


Near-IR: Veggie

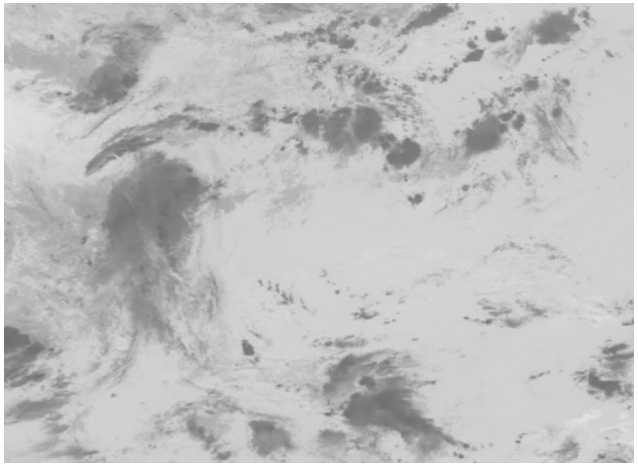


Near-IR: Snow/Ice

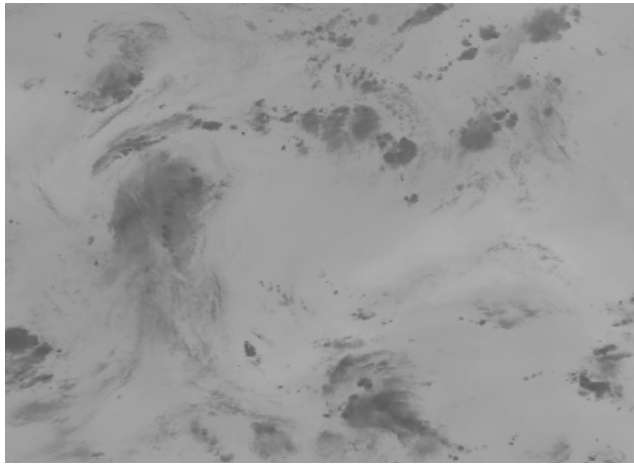
Undemonstrated Bands



IR: Shortwave

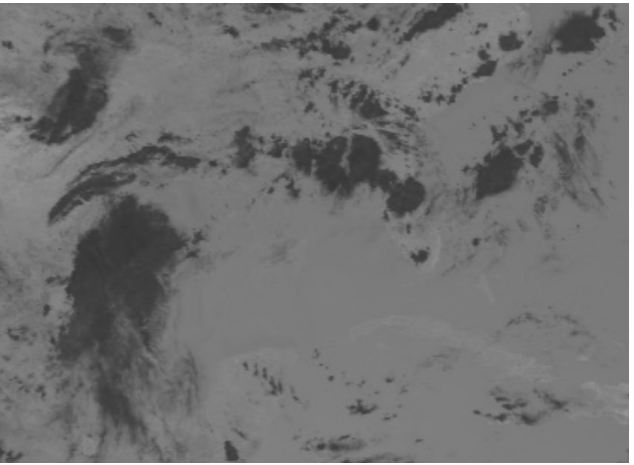


IR: Longwave

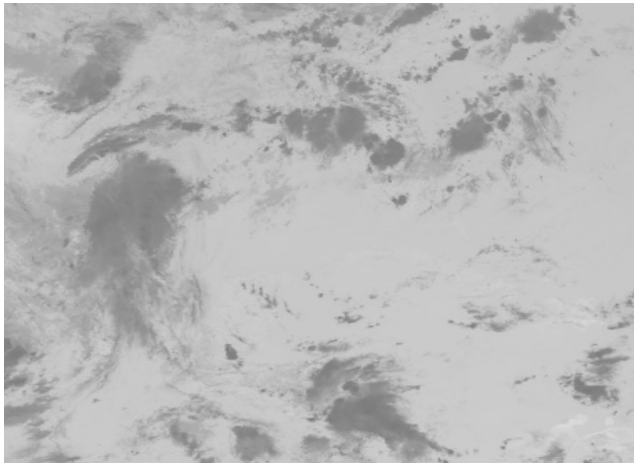


IR: Water Vapor

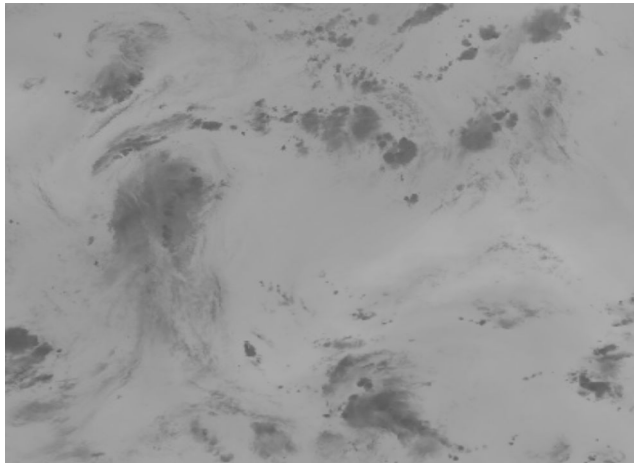
Undemonstrated Bands



IR: Shortwave

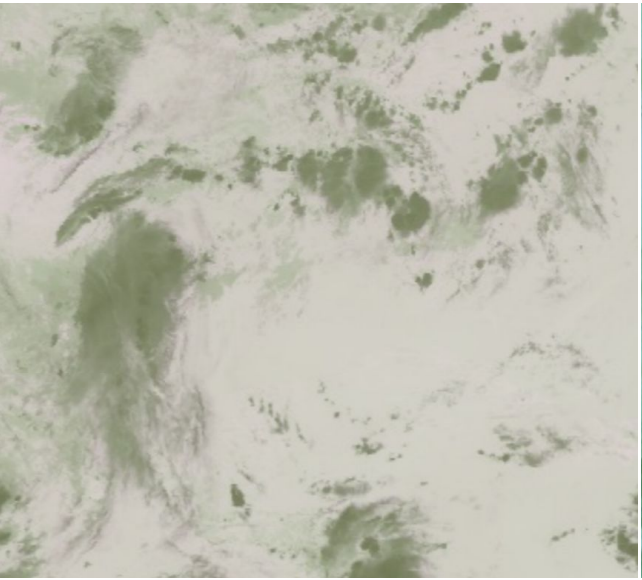


IR: Ozone

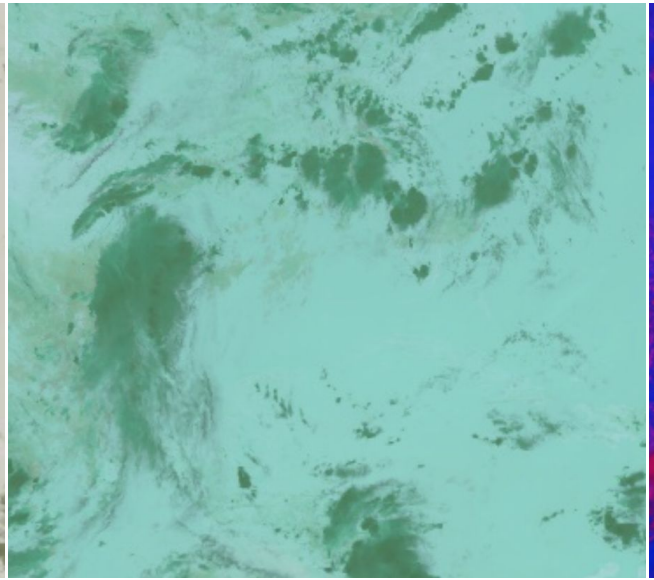


IR: Water Vapor

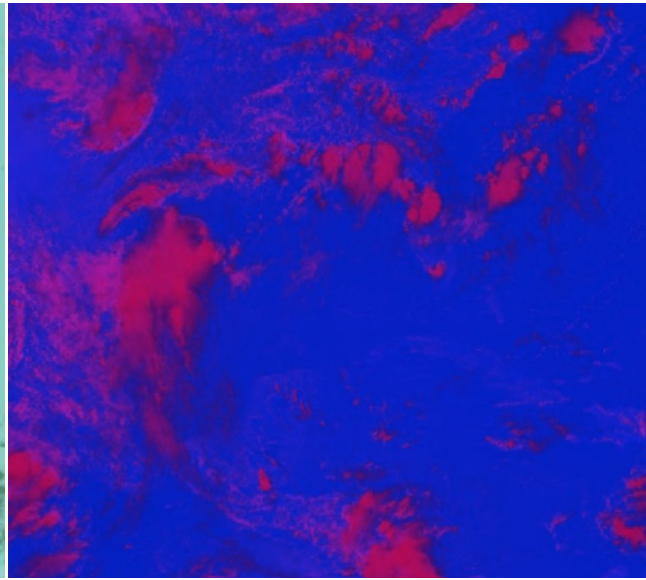
Undemonstrated Bands



Longwave, CO2, Cloud-top phase



Shortwave, CO2, Cloud-top phase



Visible Red, Cloud-top phase

Tested Combinations

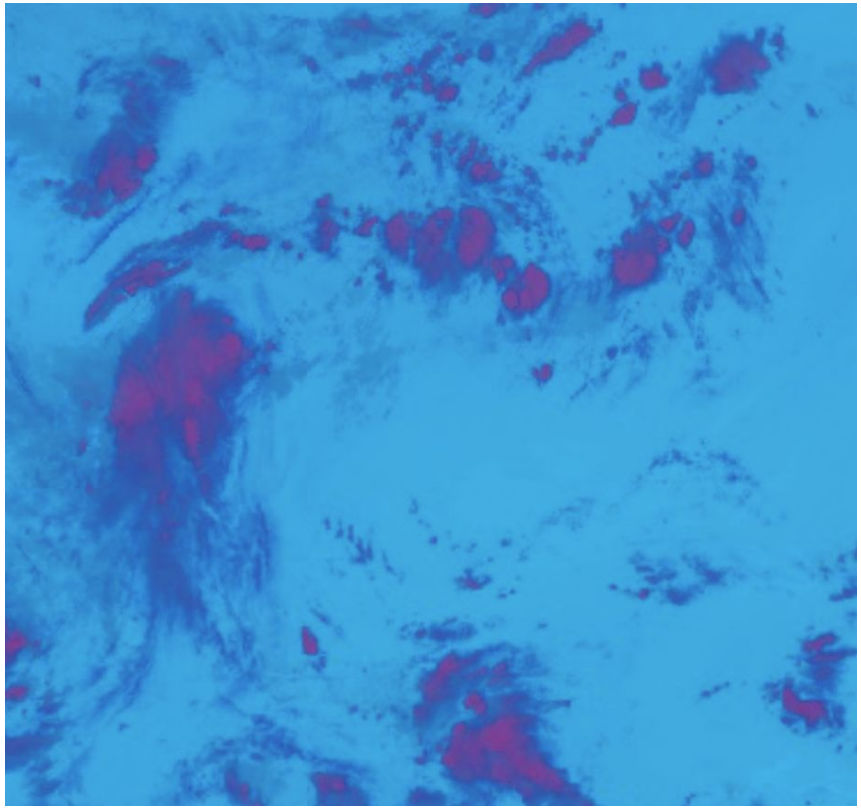
Our Banding Approach

Cirrus, Cloud Phase, CO2 Longwave

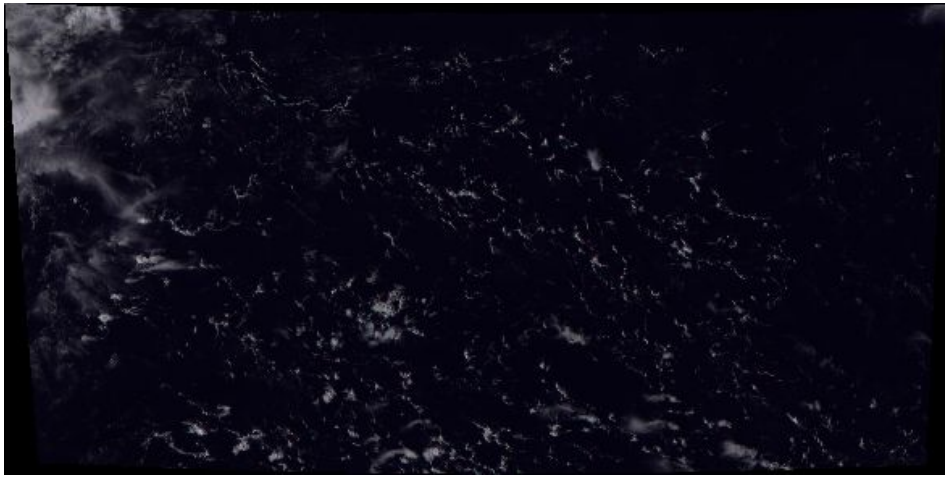
- Near-IR Cirrus: Great for excluding land and able to see cirrus clouds, but can only see daytime clouds

- IR Cloud-Top Phase: Can see clouds at any time and get the general shape of each cloud

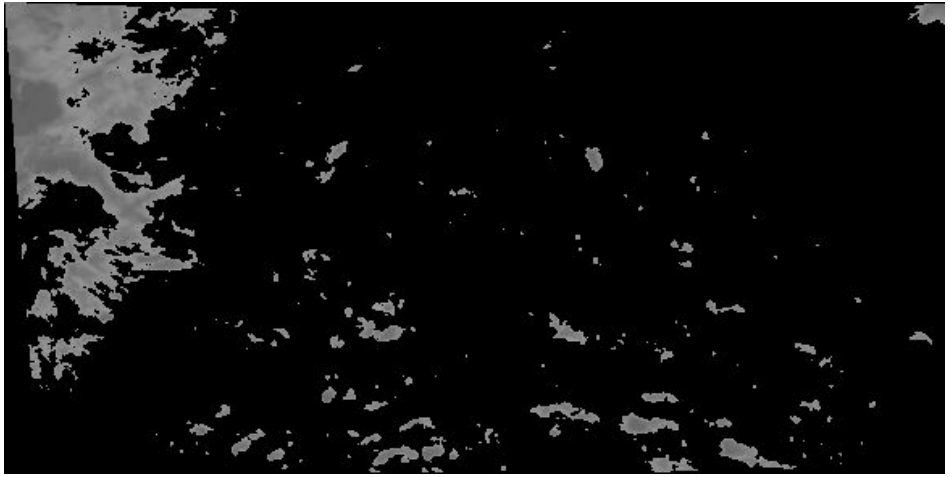
- IR CO2 Longwave: Able to differentiate clouds by height and aid in night time cloud detection



RGB Satellite Image



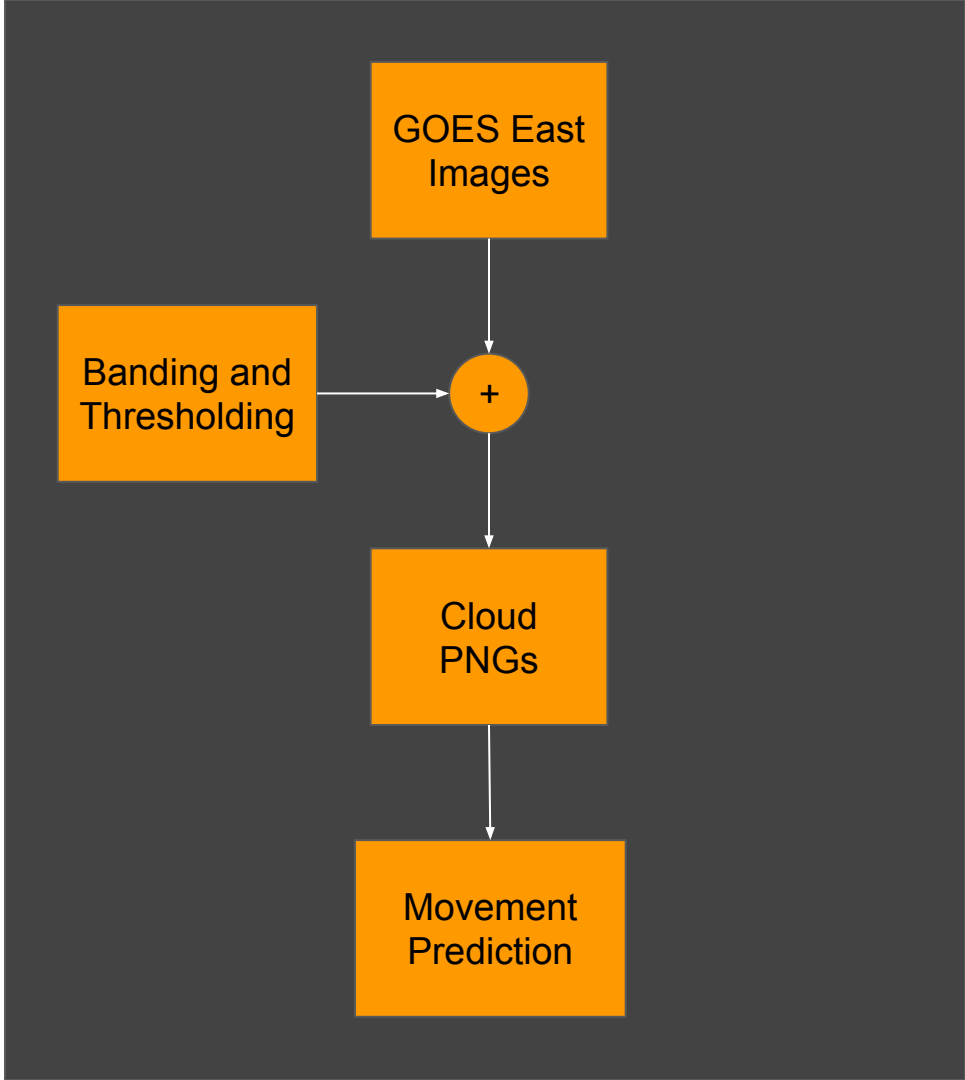
Grayscale Cloud Mask



Cloud Extraction Output

Data Pipeline

- Receive GOES East images
- Apply Banding and Thresholding
- Export image and convert image to usable format for Keras
- Input image(s) into movement prediction model



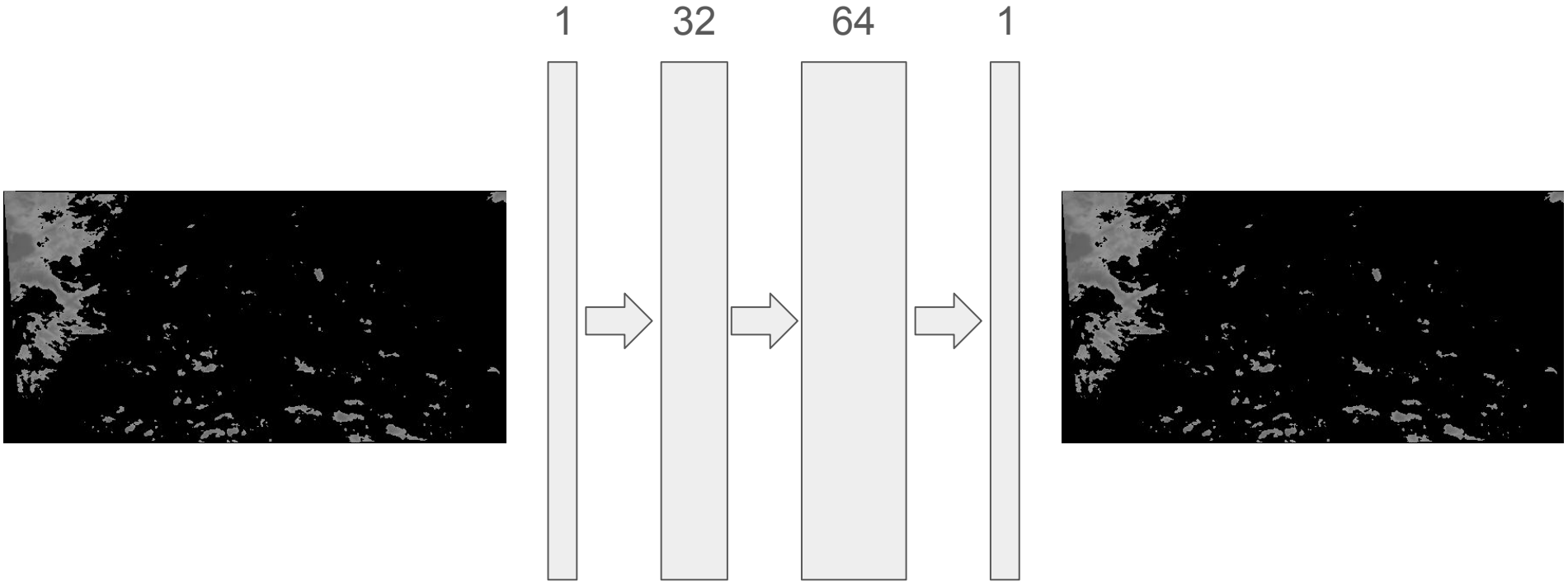
Movement Prediction Models

Initial Approaches

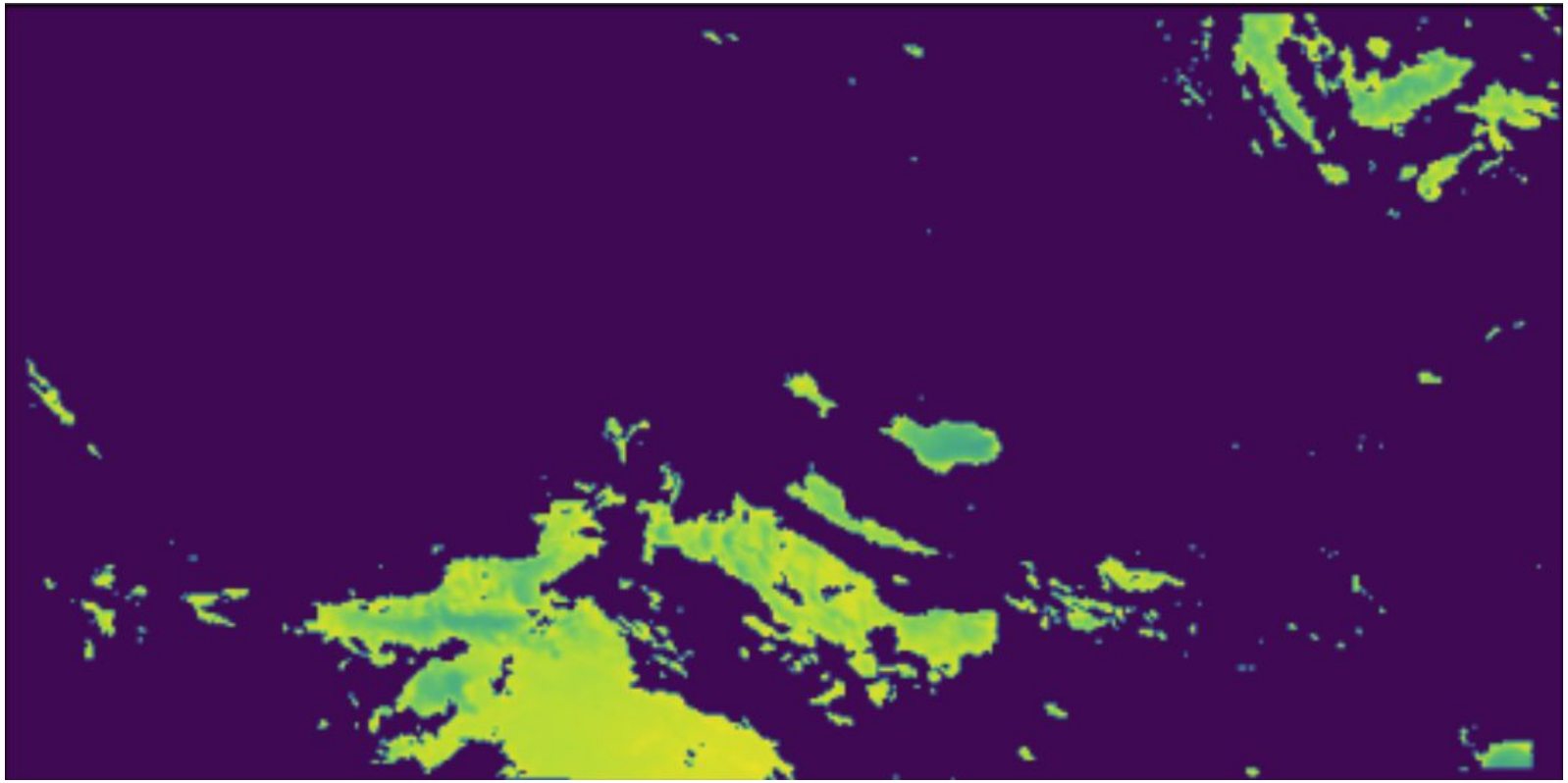
- Traditional autoregressive model seemed to be ineffective for filling in the image data between GOES satellite screenshots
    - AR models typically used for time-series numerical data
- Similarly, LSTMs on their own were not usable for the project due to our needs for spatial processing
- Possible alternatives could involve the output of such models into CNNs or RNNs

Convolution Model

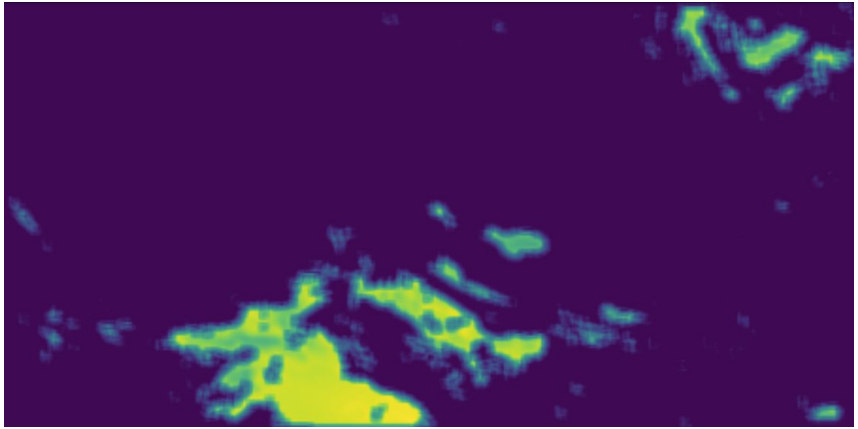
- One input frame to one output frame



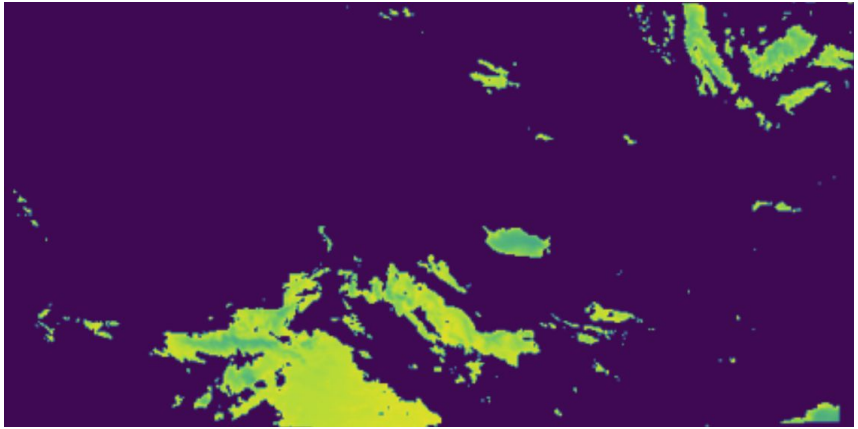
Convolutional Model Results: Single Predictions



Input Frame

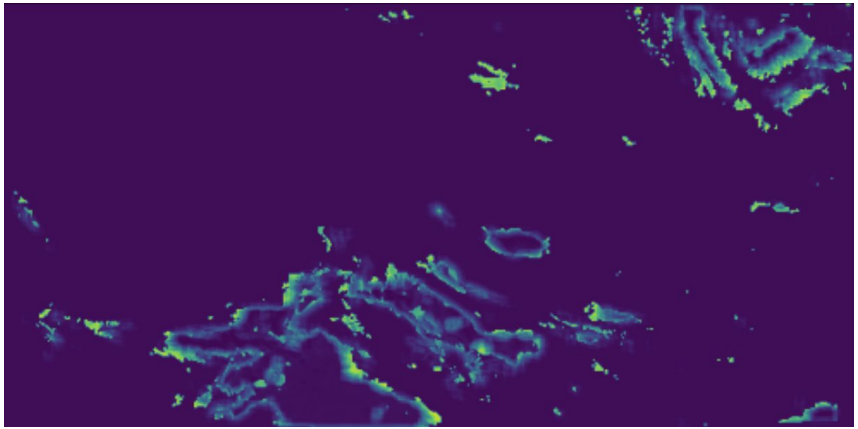


Predicted Frame



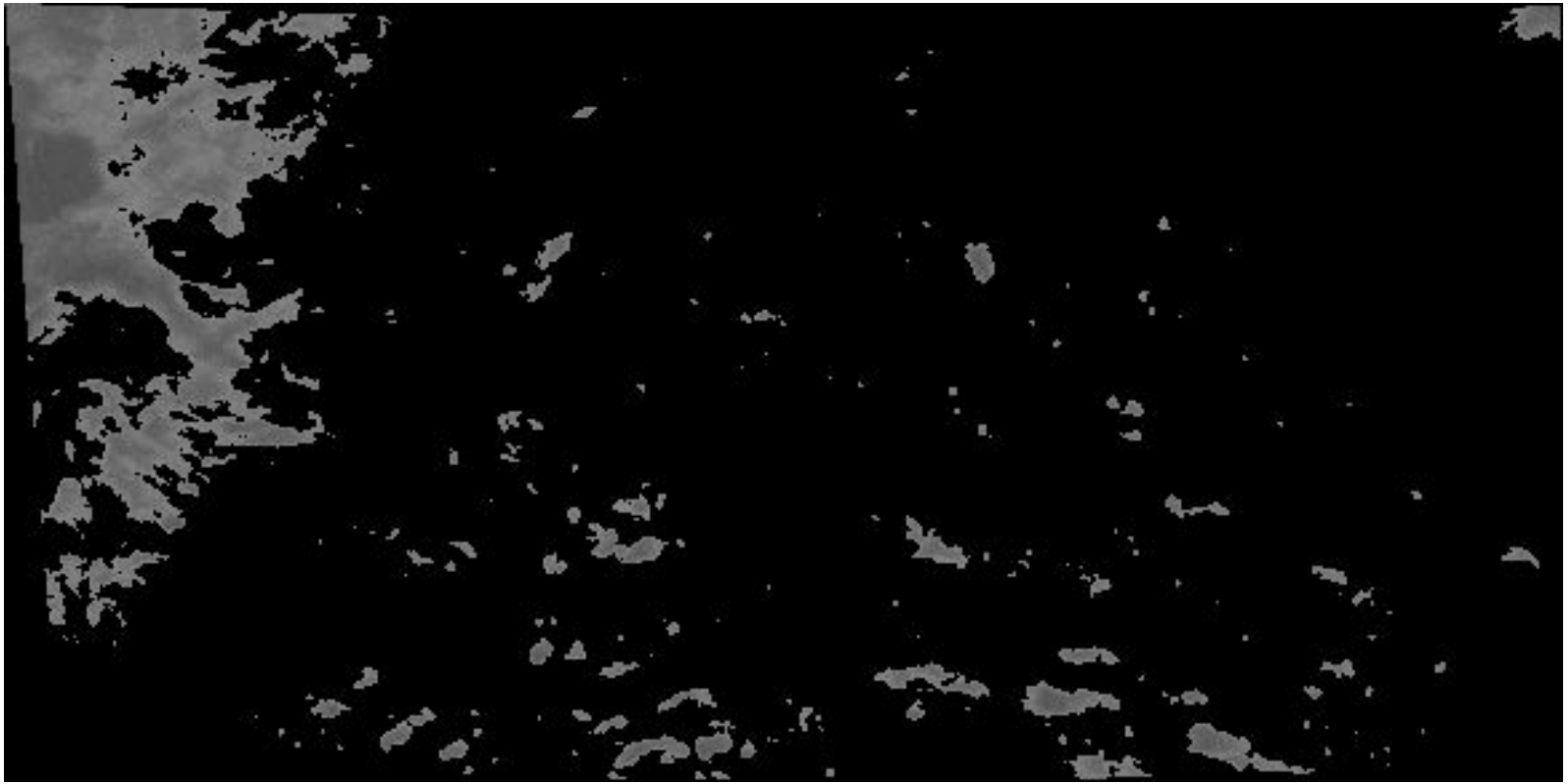
Expected Output

$$(prediction - expectecd)^2$$



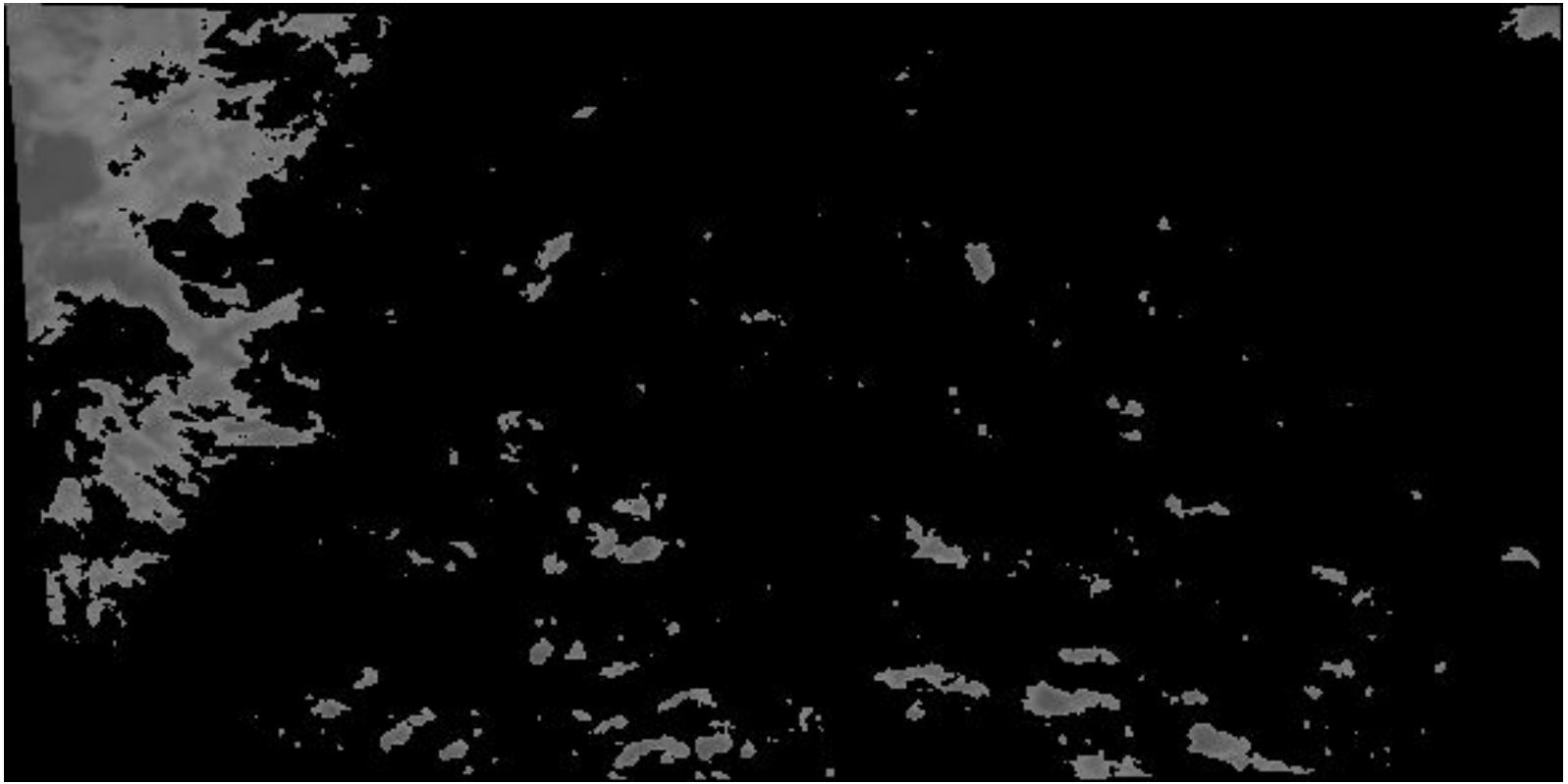
Squared Error

Convolutional Model Results: Consecutive Predictions



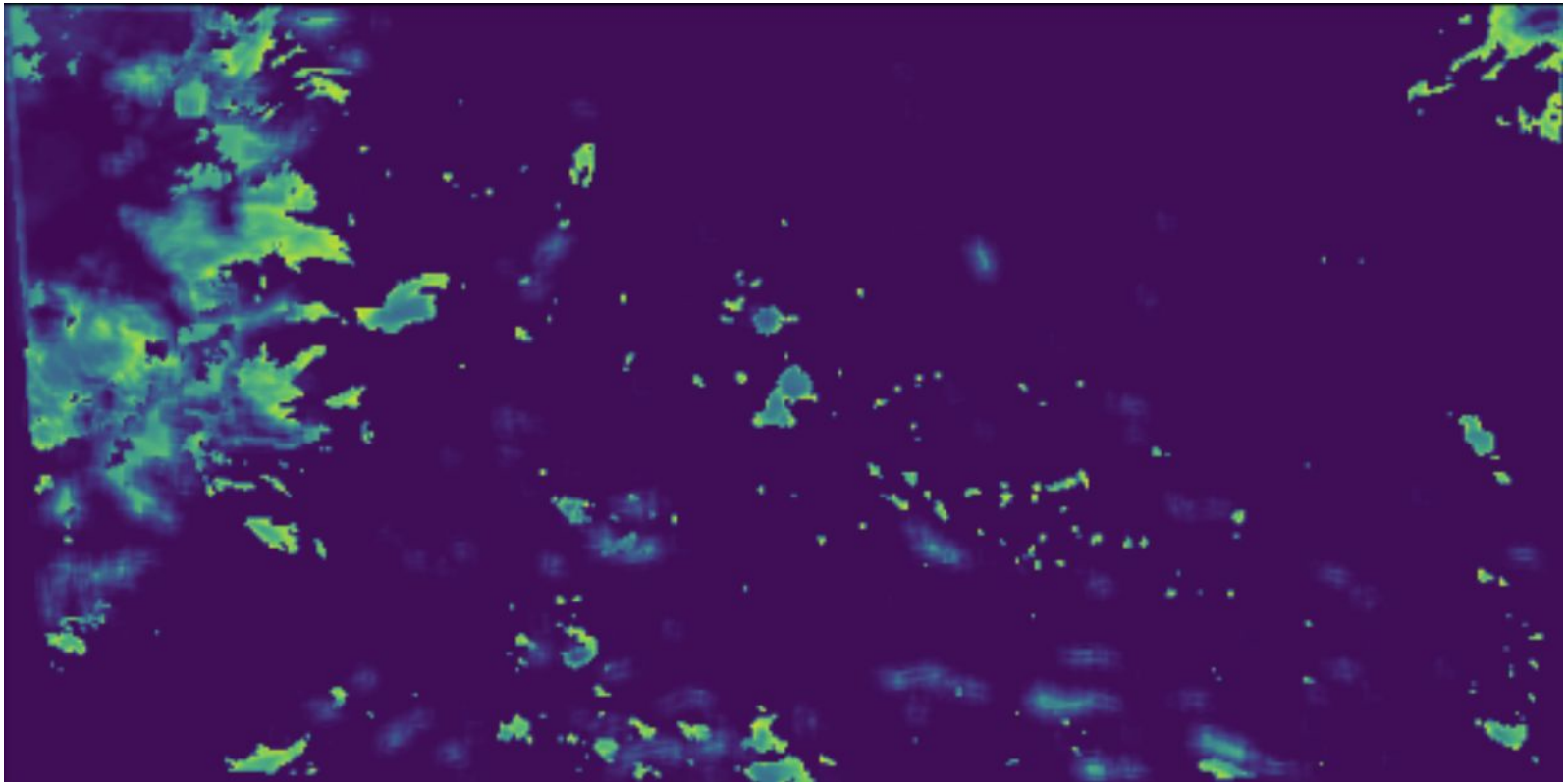
Expected Output

Convolutional Model Results: Consecutive Predictions



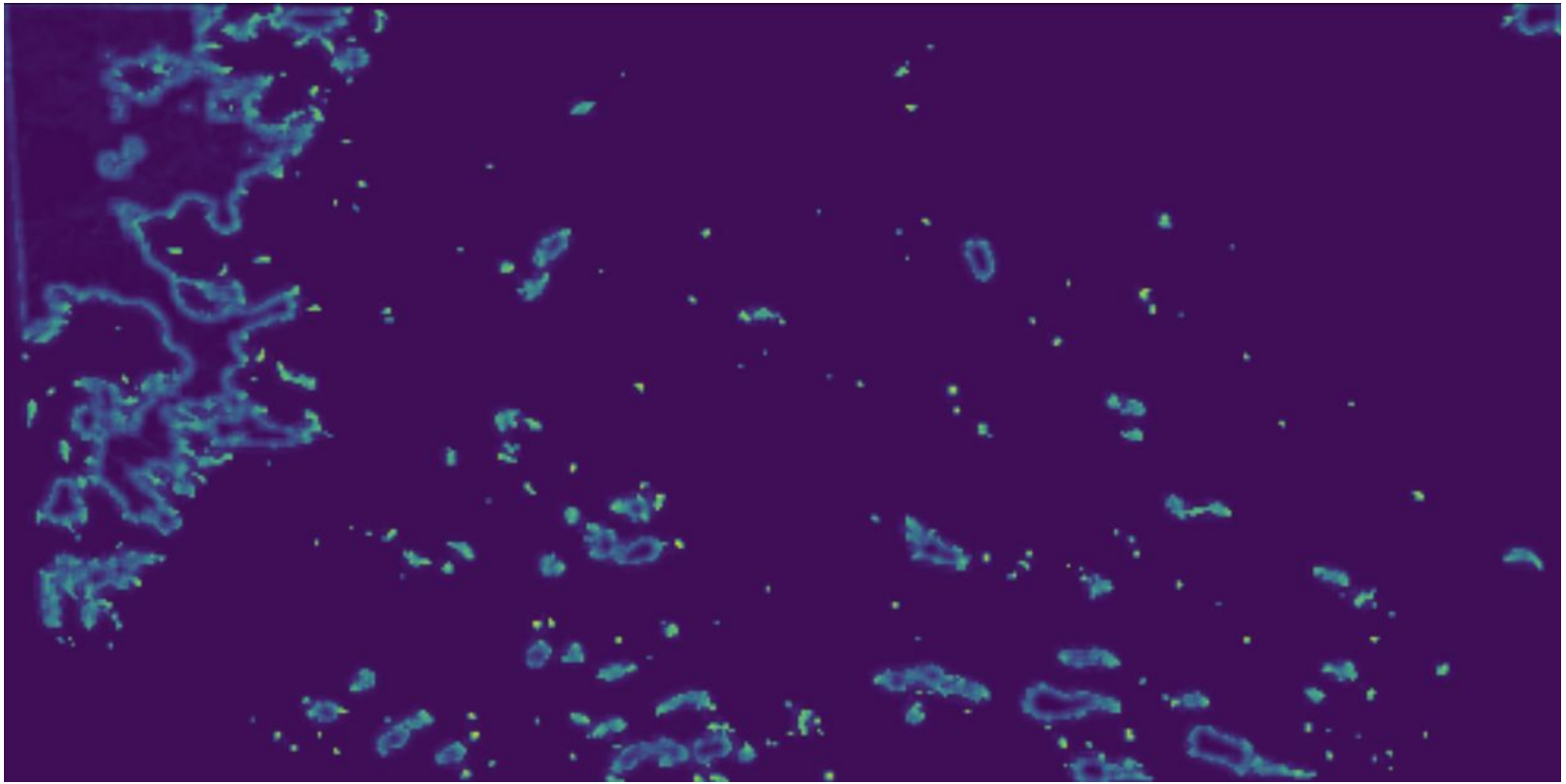
Predictions

Convolutional Model Results: Consecutive Predictions



Mean Squared Error (predictions, expected)

Convolutional Model Results: Consecutive Predictions

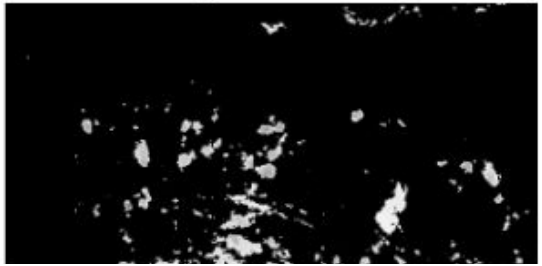


Squared Error (predictions[-1], input)

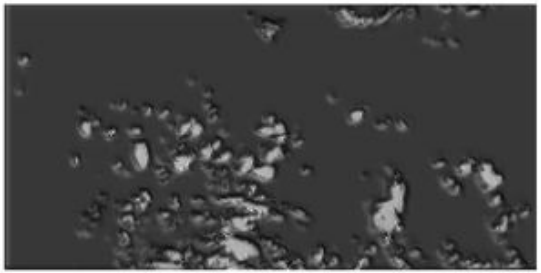
Convolutional Long Short-Term Memory Model

- Shows promise, effective at capturing general movements and changes over
- Struggles with rapid atmospheric changes due to their sequential data processing nature
  - Clouds dissipating, forming
- Temporal smoothing inherent in LSTM blurs distinctions between different stages of cloud dynamics, producing averaged or smoothed predictions

Original Frame 6



Predicted Frame 6



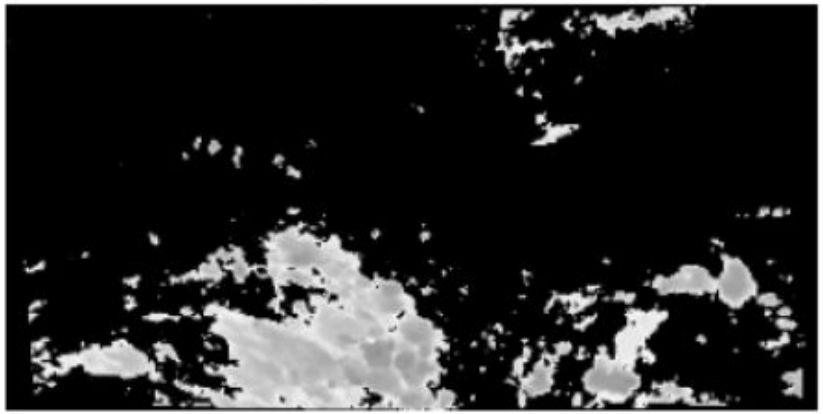
Convolutional Long Short-Term Memory Model

```
Epoch 1/20
359/359 [====] - 260s 687ms/step - loss: 0.1169 - val_loss: 0.2056 -
Epoch 2/20
359/359 [====] - 242s 674ms/step - loss: 0.1020 - val_loss: 0.1164 -
Epoch 3/20
359/359 [====] - 242s 674ms/step - loss: 0.1013 - val_loss: 0.1121 -
Epoch 4/20
359/359 [====] - 242s 675ms/step - loss: 0.1011 - val_loss: 0.1119 -
Epoch 5/20
359/359 [====] - 242s 675ms/step - loss: 0.1009 - val_loss: 0.1172 -
Epoch 6/20
359/359 [====] - 242s 675ms/step - loss: 0.1008 - val_loss: 0.1118 -
Epoch 7/20
359/359 [====] - 242s 675ms/step - loss: 0.1007 - val_loss: 0.1115 -
Epoch 8/20
```

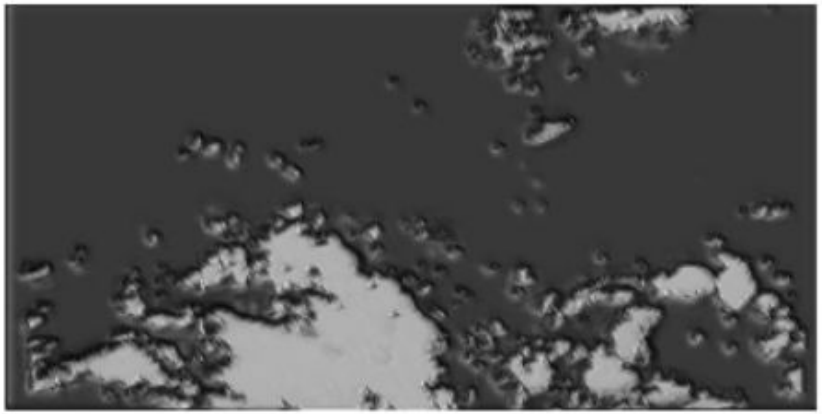
```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, None, 160, 320,   0
                              1)]

conv_lstm2d (ConvLSTM2D)     (None, None, 160, 320,    416256
                              64)

batch_normalization (Batch   (None, None, 160, 320,    256
Normalization)                64)

conv_lstm2d_1 (ConvLSTM2D)   (None, None, 160, 320,    295168
                              64)

batch_normalization_1 (Bat   (None, None, 160, 320,    256
chNormalization)              64)

conv_lstm2d_2 (ConvLSTM2D)   (None, None, 160, 320,    33024
                              64)

batch_normalization_2 (Bat   (None, None, 160, 320,    256
chNormalization)              64)

conv3d (Conv3D)              (None, None, 160, 320,    1729
                              1)
```

Convolutional Long Short-Term Memory Model

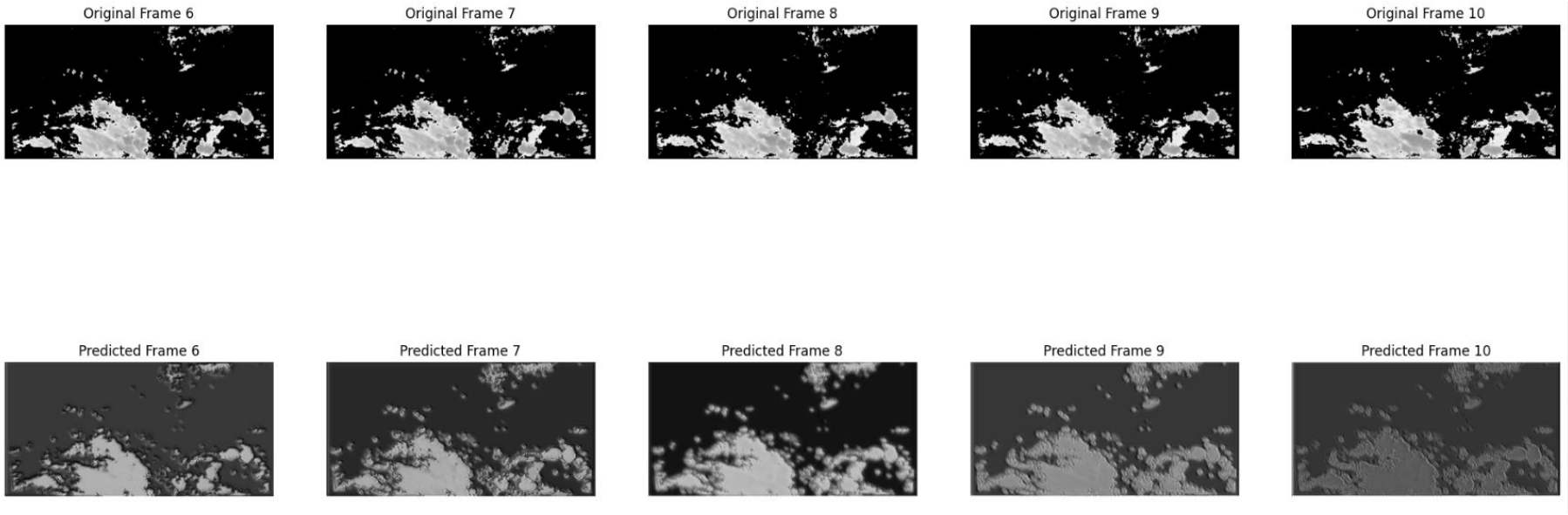
Original Frame 6



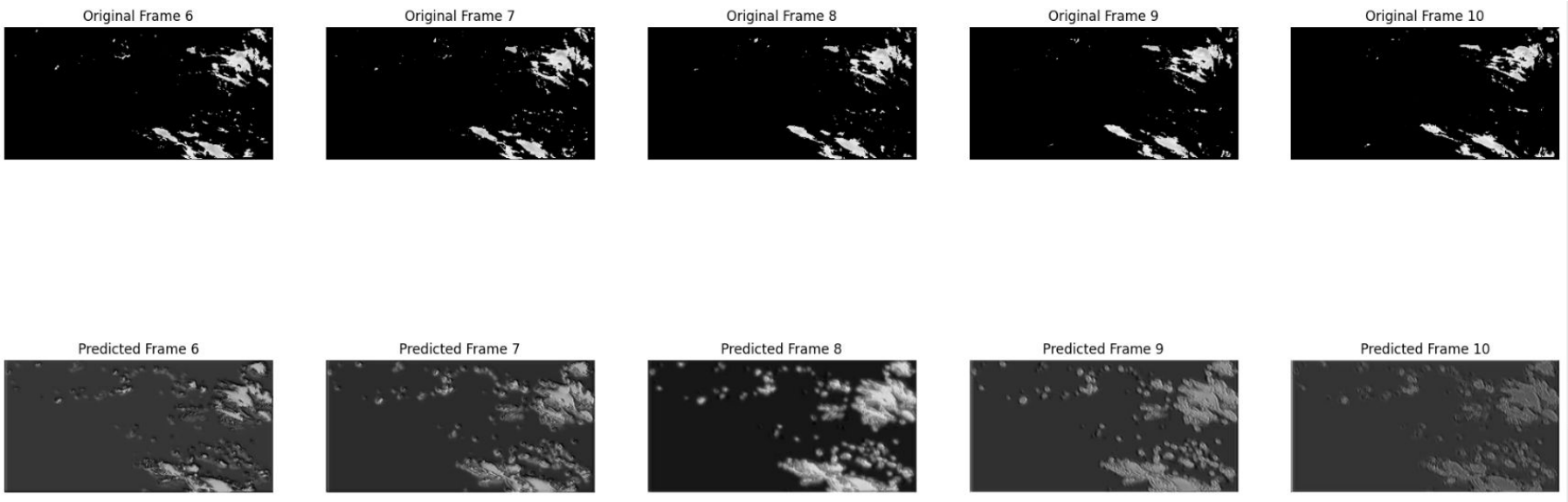
Predicted Frame 6



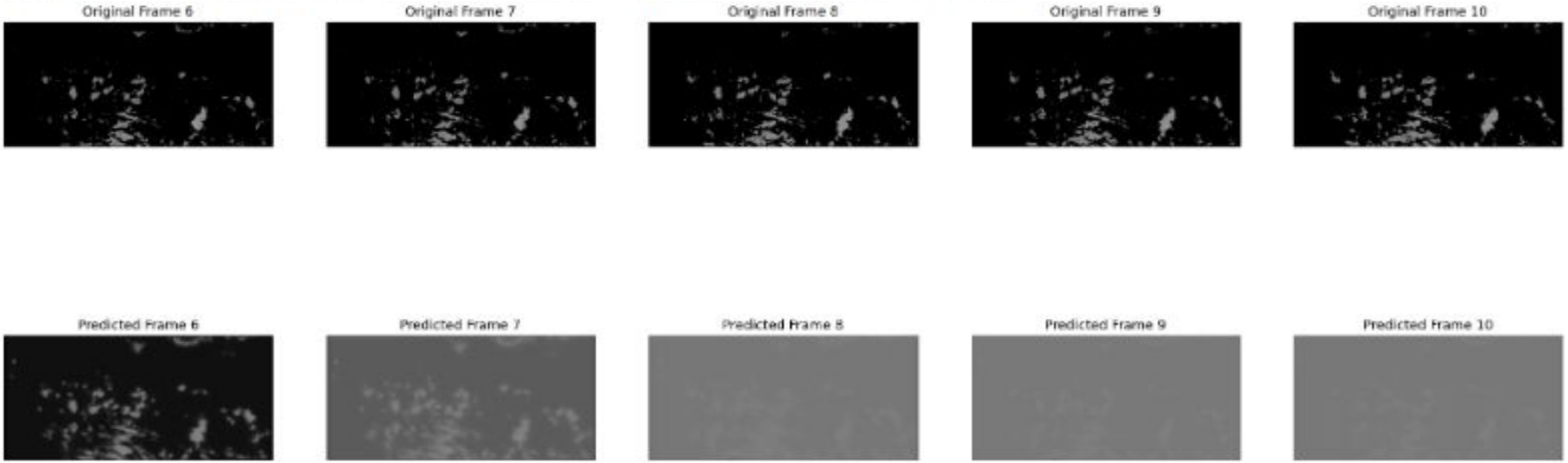
Convolutional Long Short-Term Memory Model



Convolutional Long Short-Term Memory Model

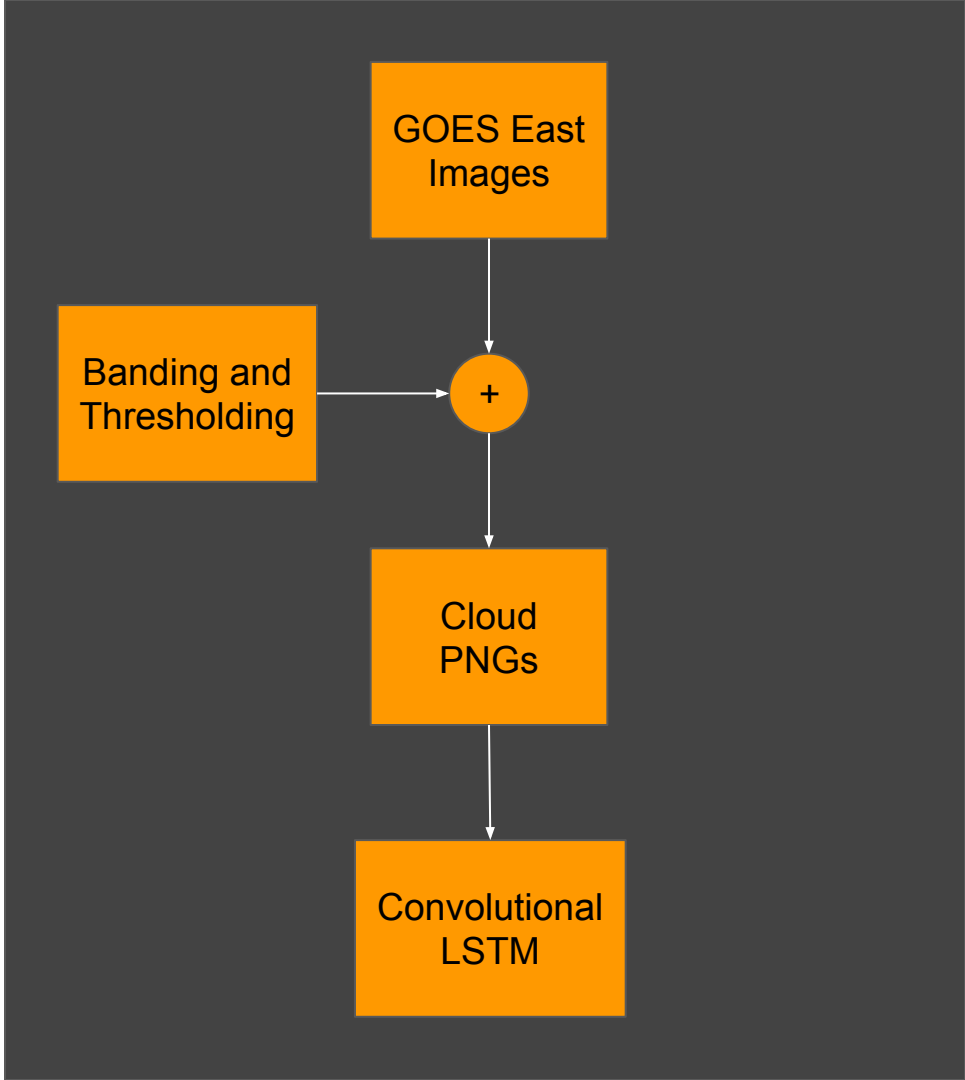


Convolutional Long Short-Term Memory Model



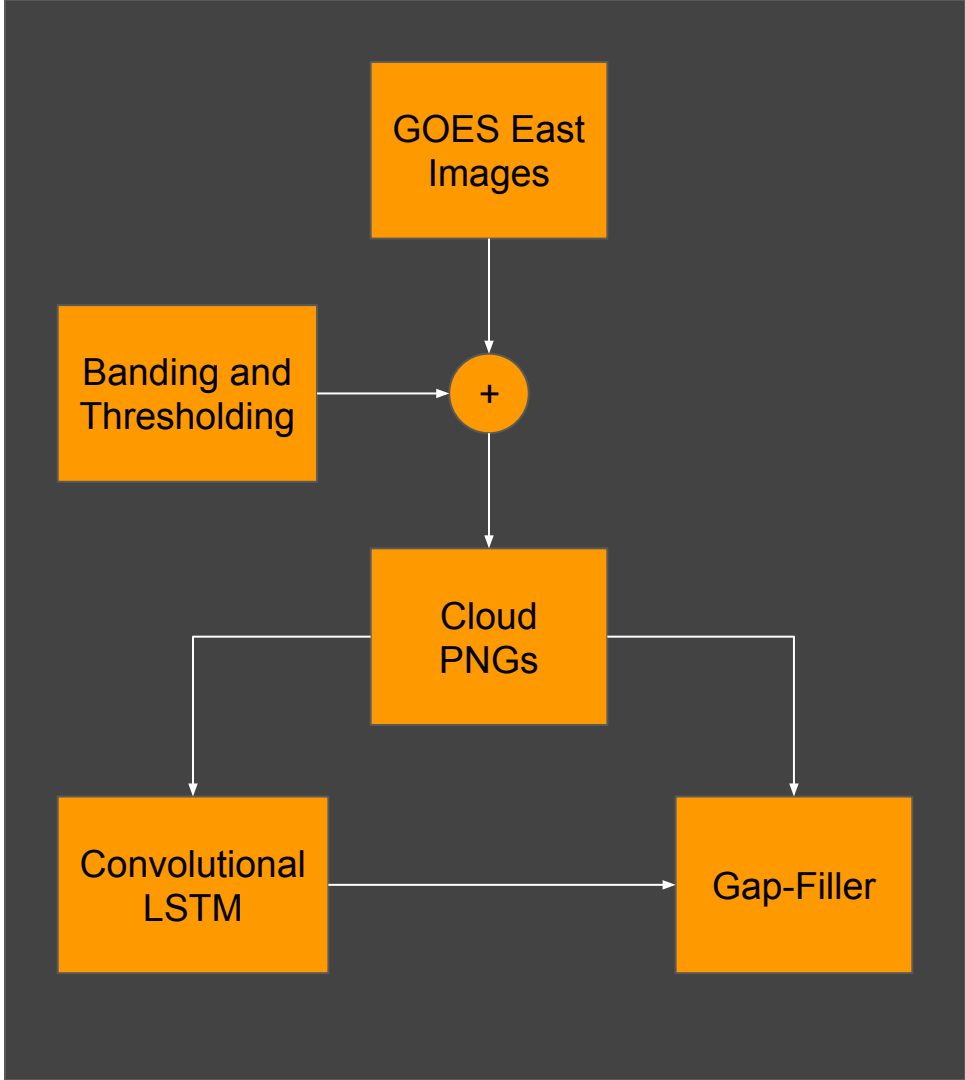
Data Pipeline

- Receive GOES East images
- Apply Banding and Thresholding
- Export image and convert image to usable format for Keras
- Input image(s) into Convolutional LSTM
  - ❏ Reduce required number of predictions



Data Pipeline

- Receive GOES East images
- Apply Banding and Thresholding
- Export image and convert image to usable format for Keras
- Input image(s) into Convolutional LSTM
  - ❏ Reduce required number of predictions
- Input combination of images and predictions into gap-filling model

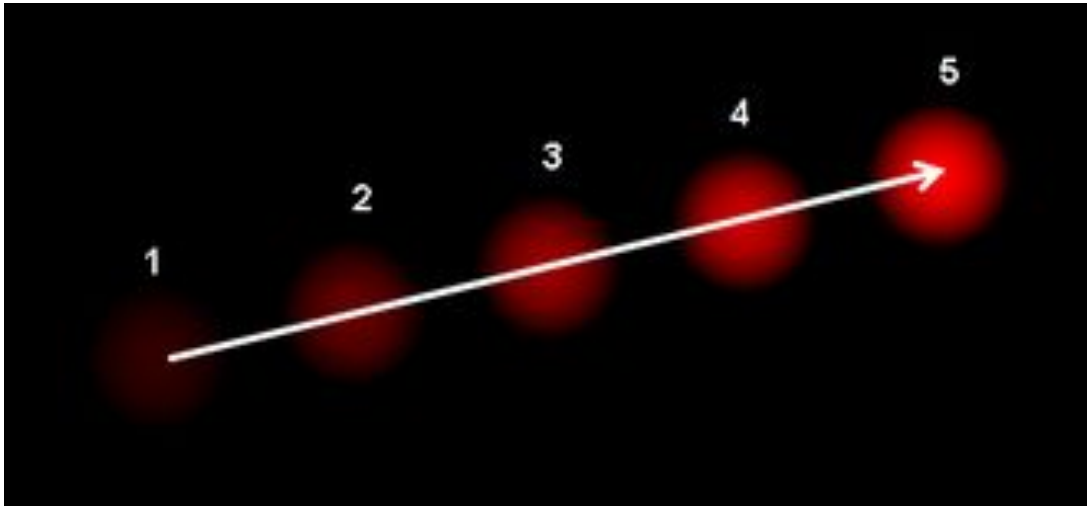


Models to Fill Data Gaps

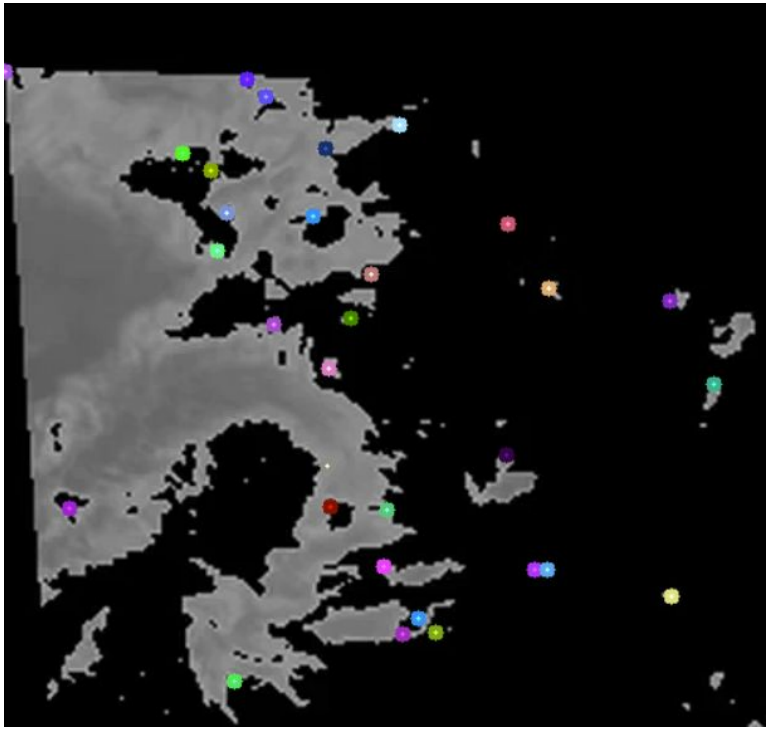
Optical Flow

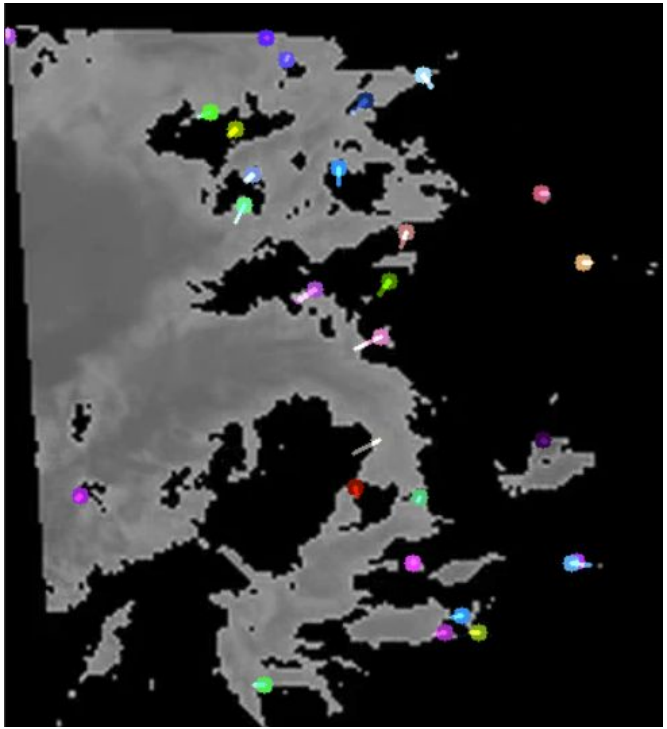
Motion in image processing defined in three ways:

- Changes in light
- Changes in an object
- Changes in the camera



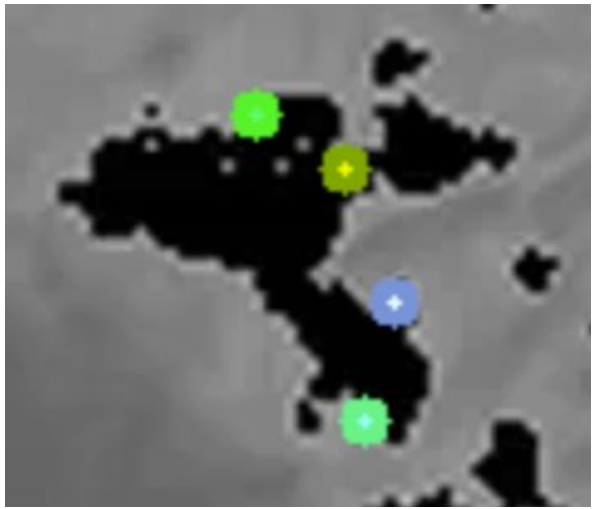
Optical Flow via OpenCV

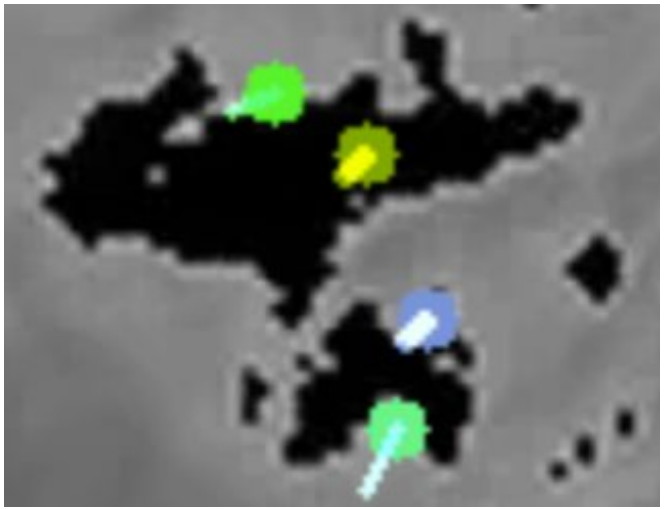




Optical Flow Issues

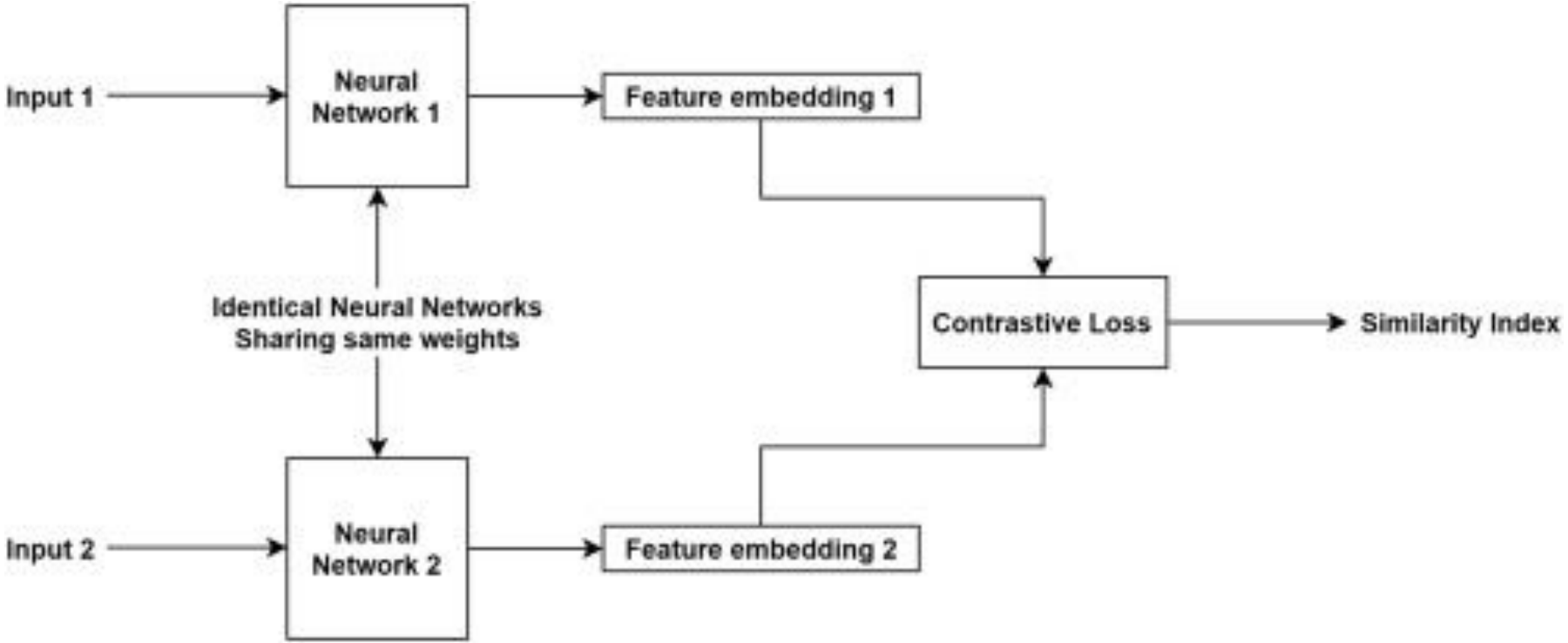
- 3 variables (clouds shape, lighting, earth rotating)
- Images too far apart in time (15 minutes vs. 33ms)
- Optical Flow not meant for fluids



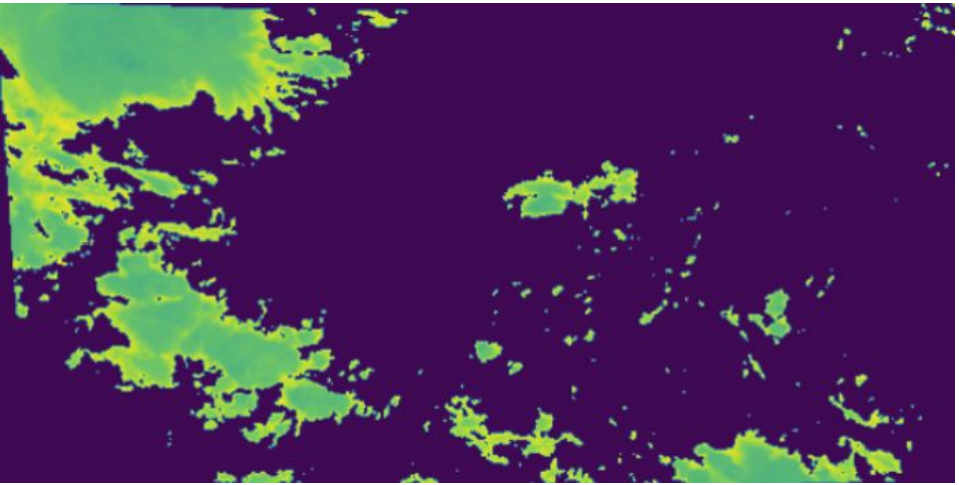


Siamese Network

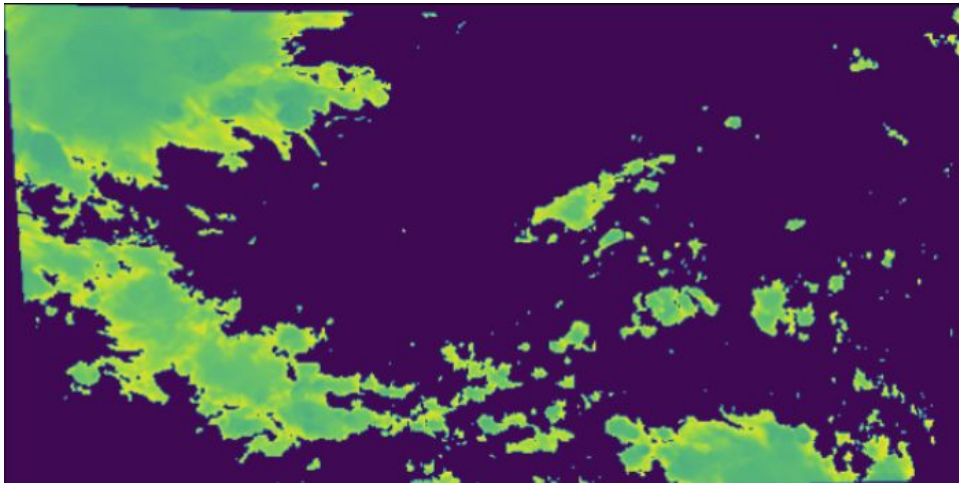
- Two input frames to one output frame



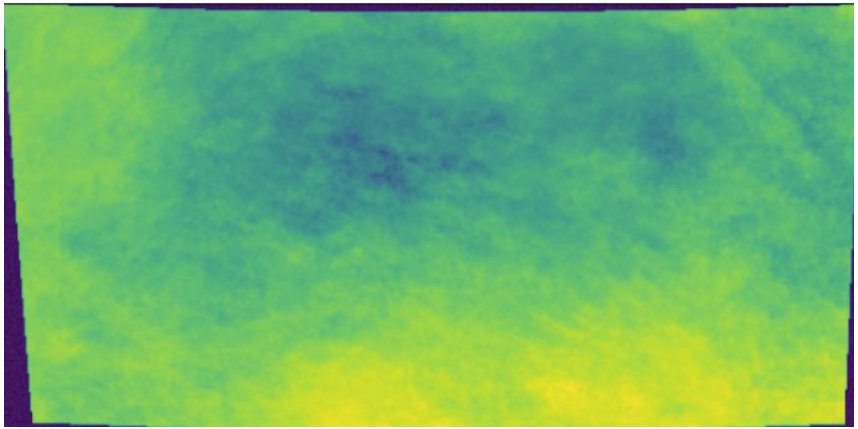
Siamese Model Results: Single Predictions



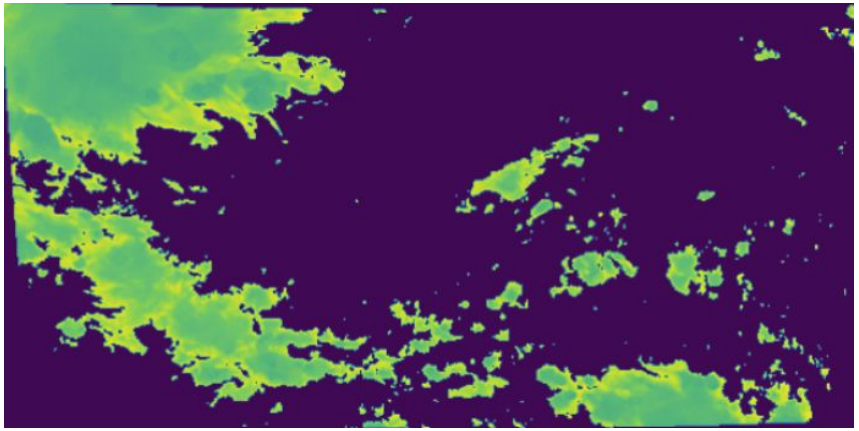
Starting Frame



Ending Frame

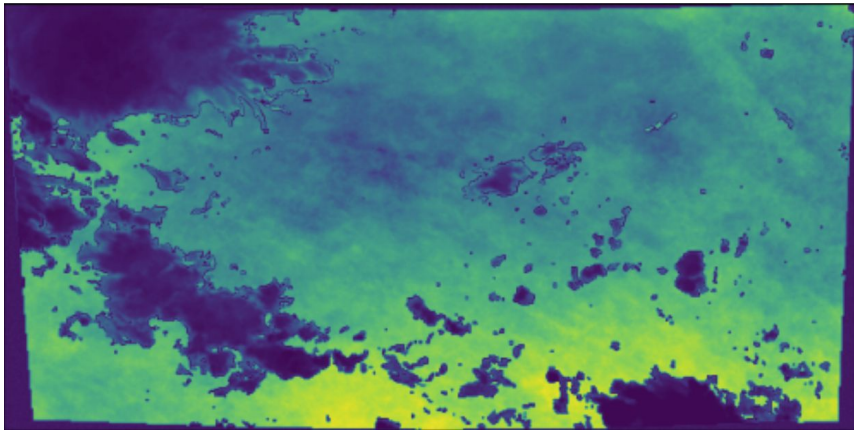


Predicted Frame



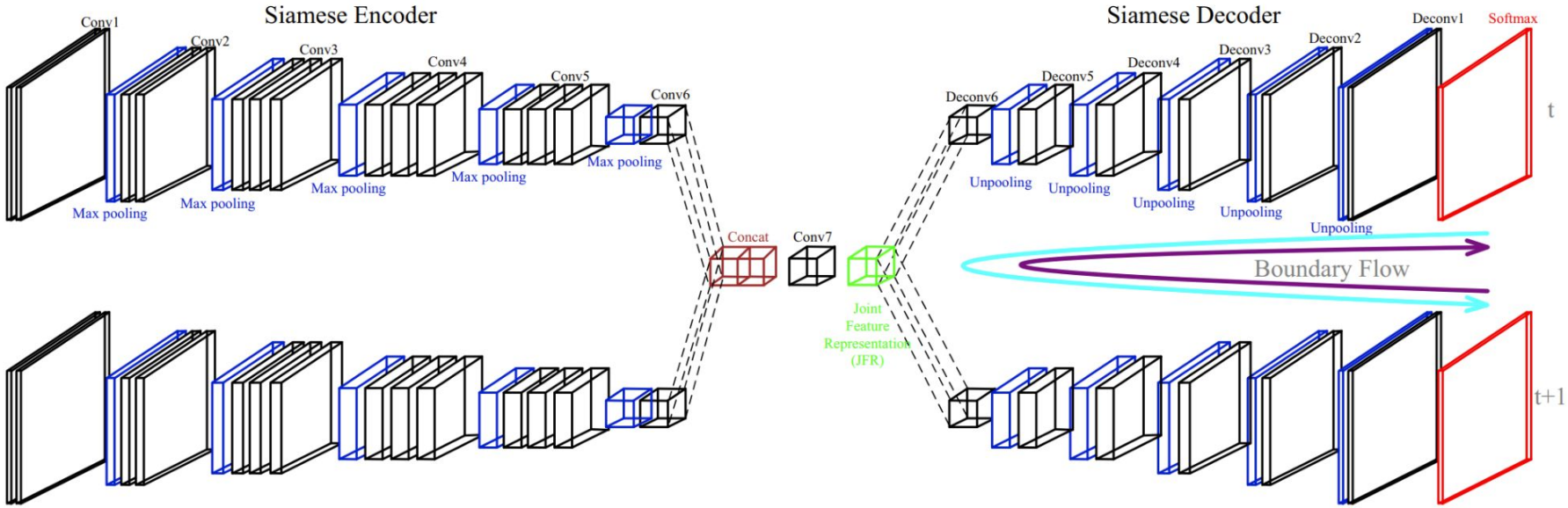
Expected Output

$$(prediction - expectecd)^2$$



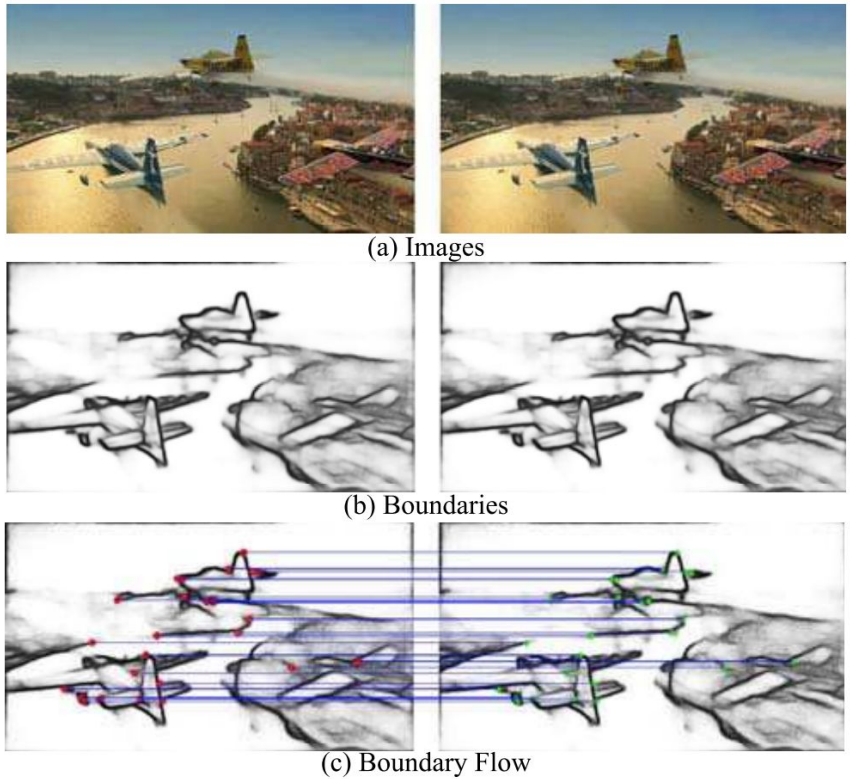
Squared Error

Siamese Model: What is Possible



Peng Lei, Fuxin Li, and Sinisa Todorovic (2018)

Siamese Model: What is Possible



(a) Images

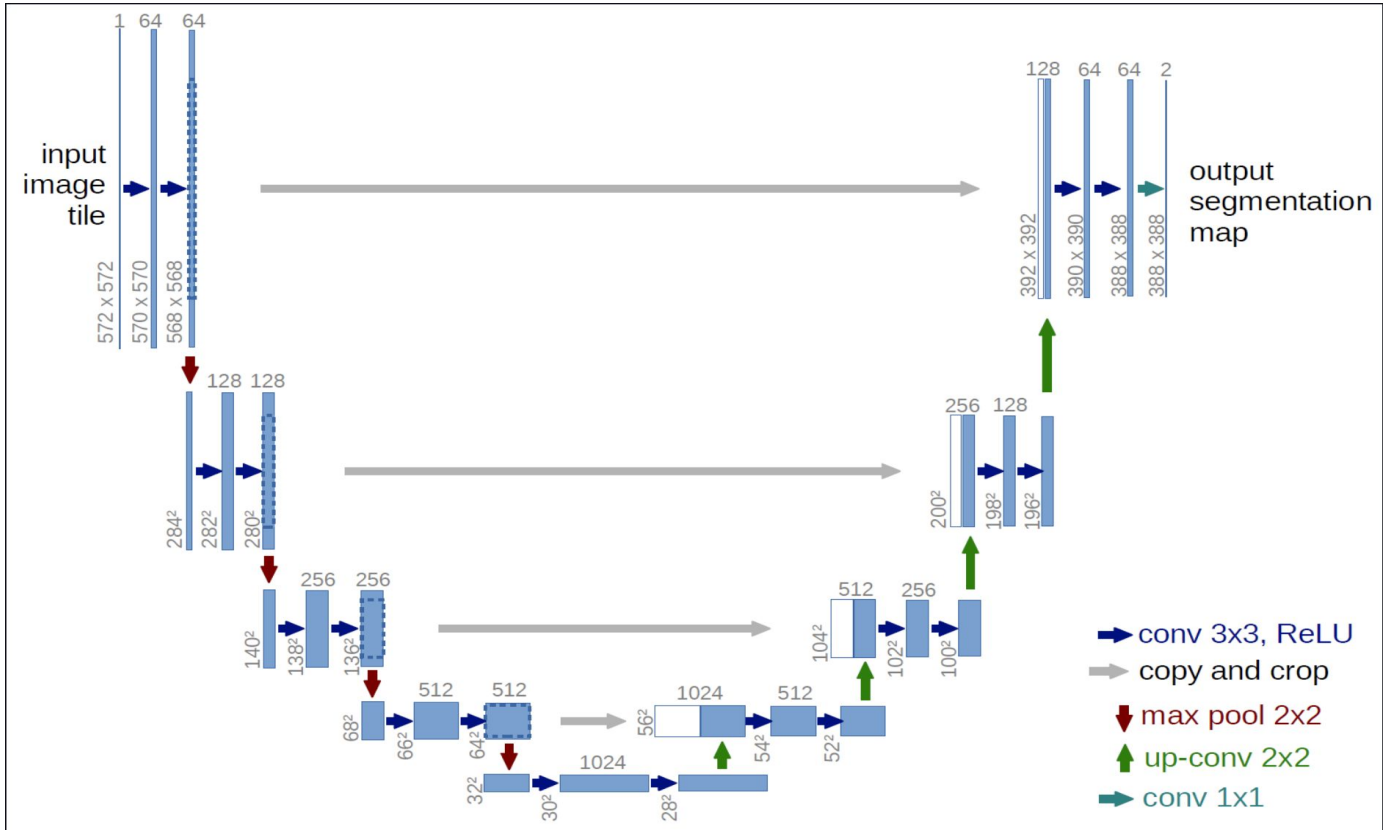
(b) Boundaries

(c) Boundary Flow

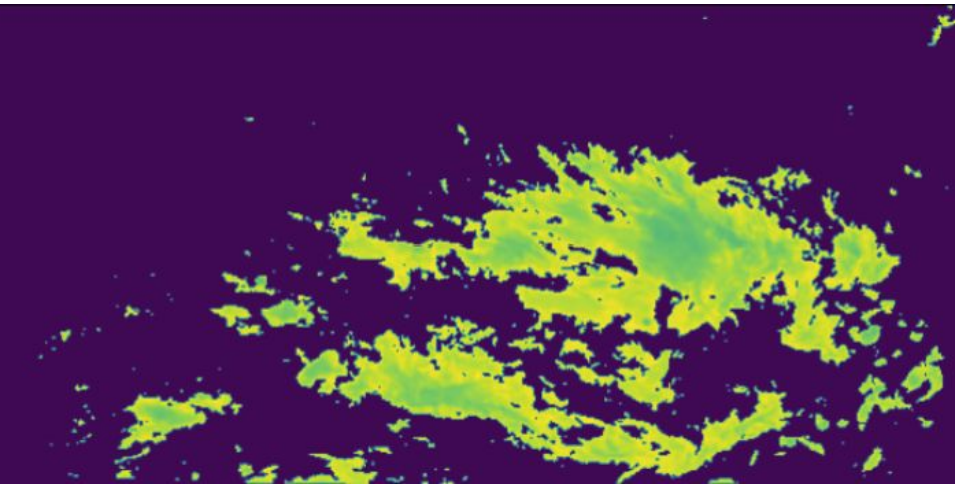
Peng Lei, Fuxin Li, and Sinisa Todorovic (2018)

U-Net Frame Interpolation

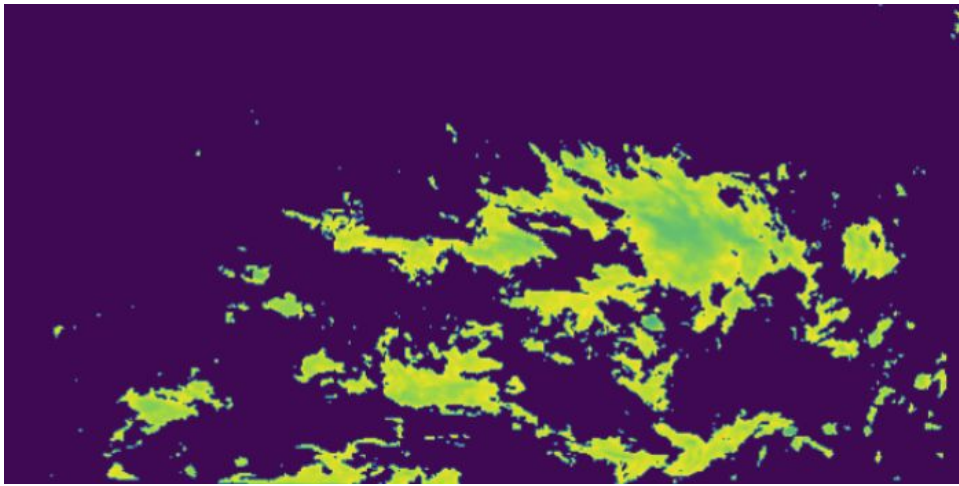
- Two input frames to one output frame



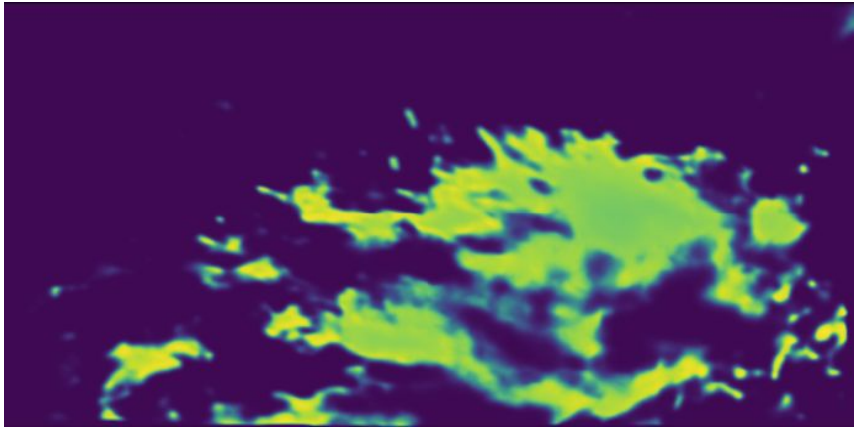
U-Net Model Results: Single Predictions



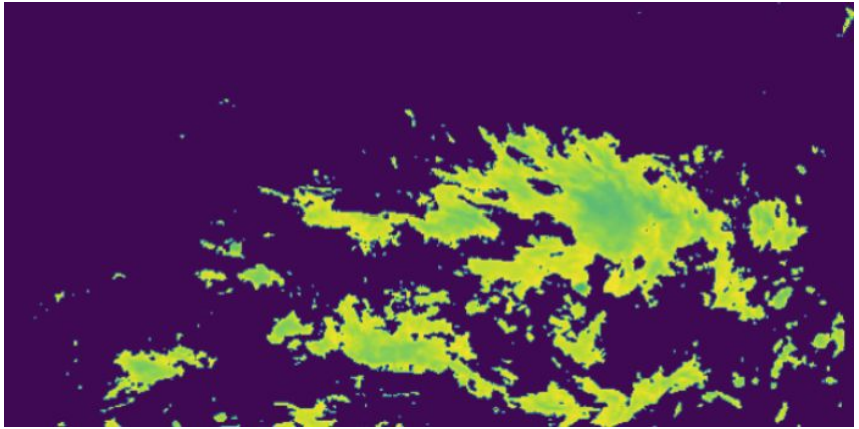
Starting Frame



Ending Frame

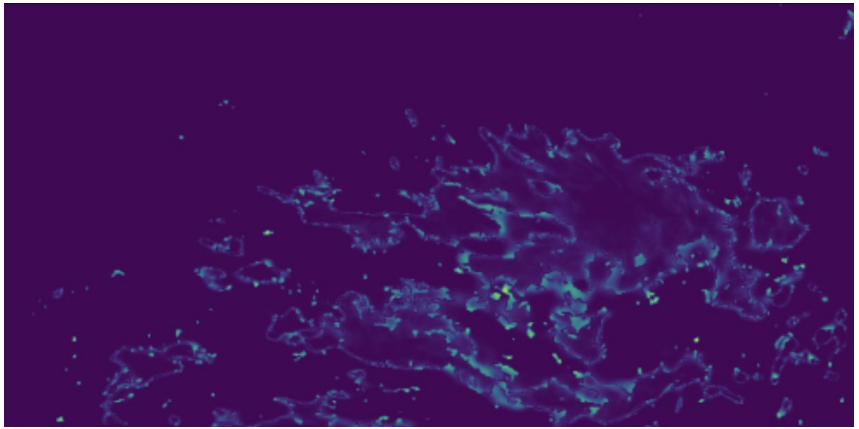


Predicted Frame



Expected Output

$$(prediction - expectecd)^2$$



Squared Error

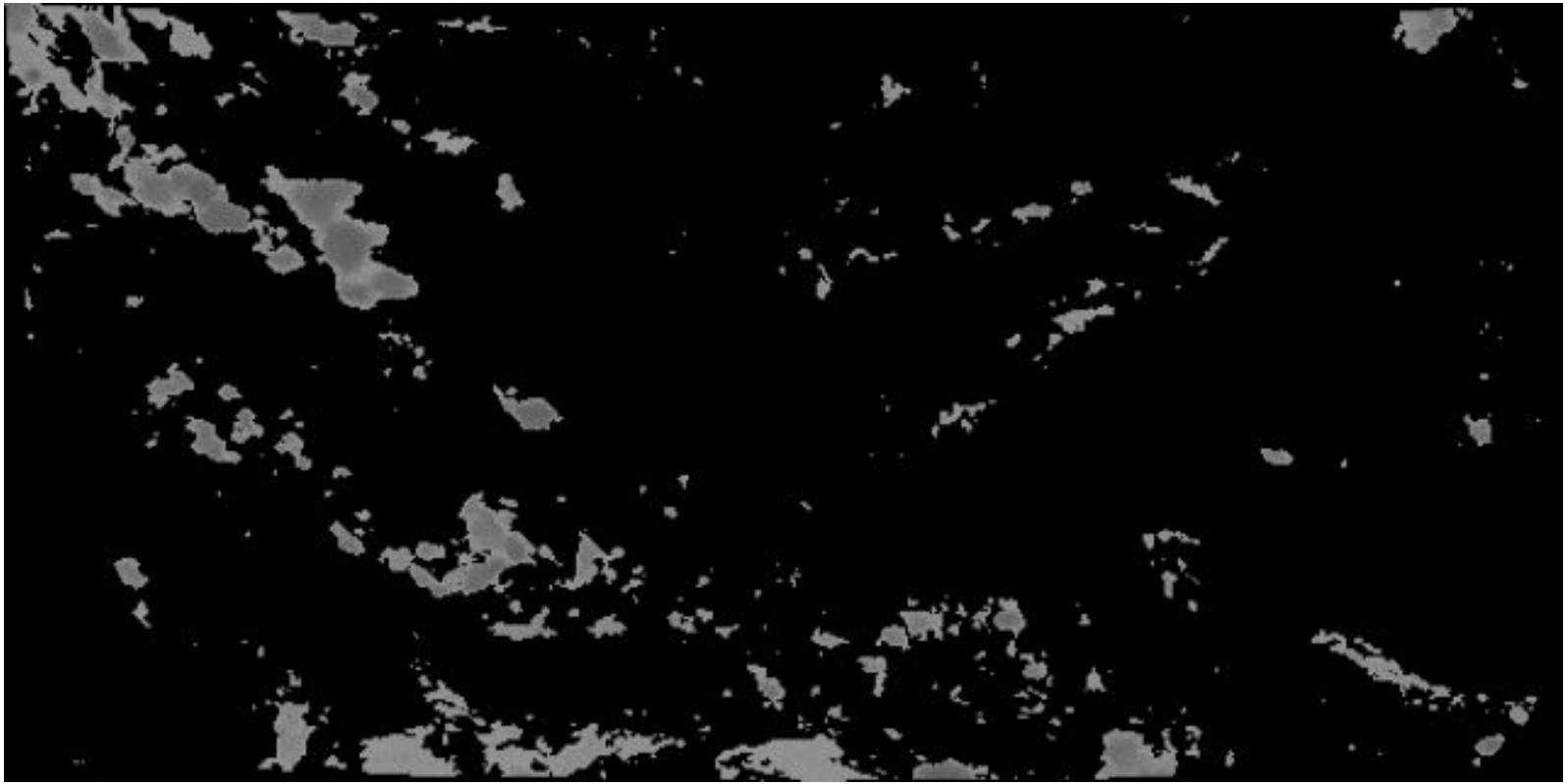
U-Net Model Results: Consecutive Predictions

Expected Output

U-Net Model Results: Consecutive Predictions

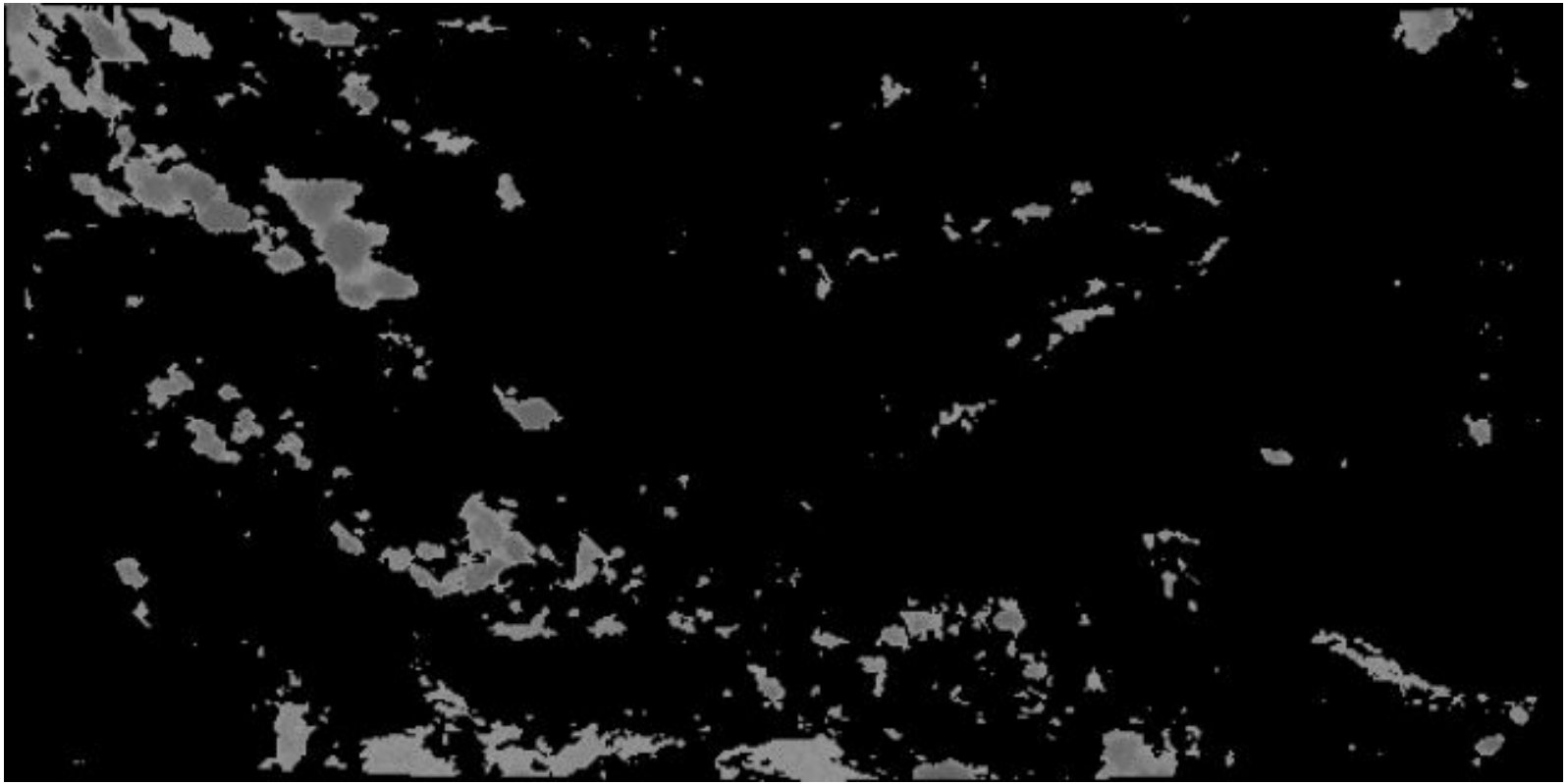
Predictions

U-Net Model Results: Consecutive Predictions



Expected Output

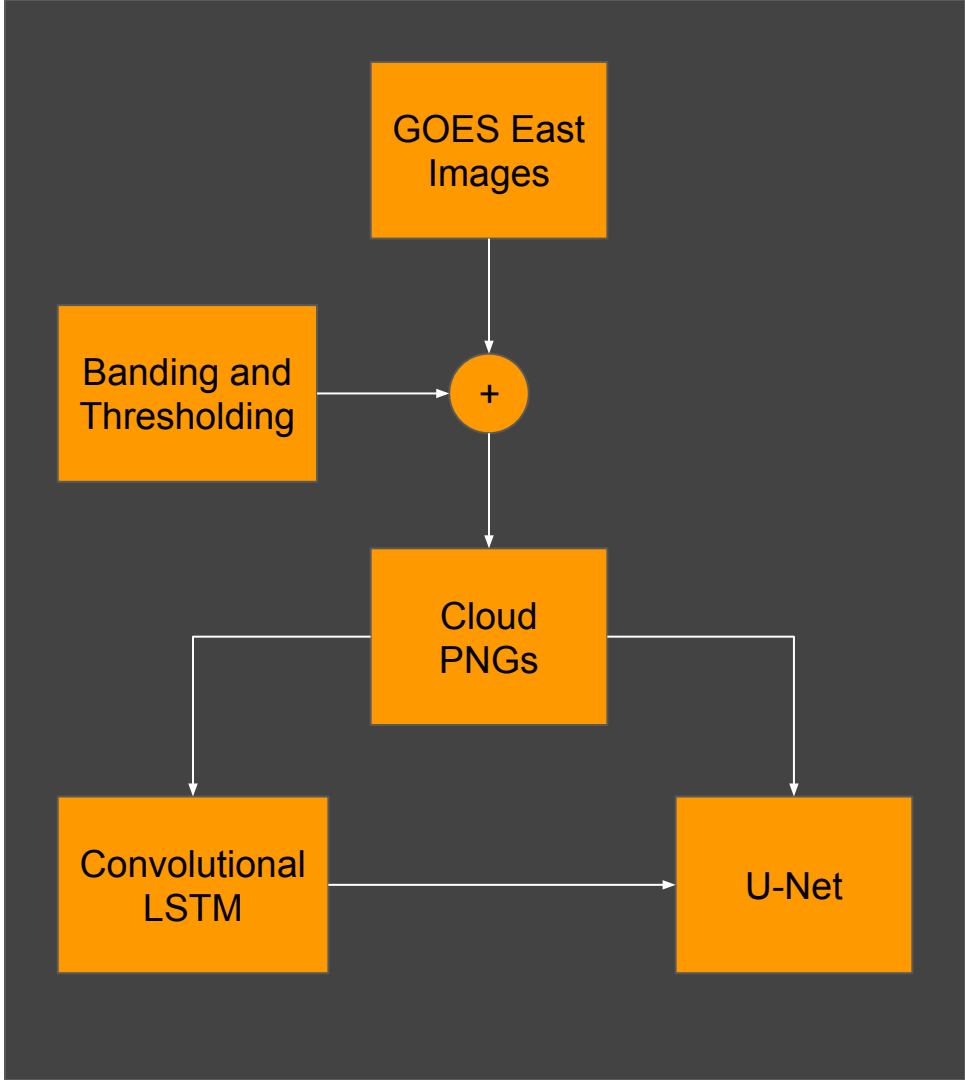
U-Net Model Results: Consecutive Predictions



Predictions

Final Data Pipeline

- Receive GOES East images
- Apply Banding and Thresholding
- Export image and convert image to usable format for Keras
- Input image(s) into Convolutional LSTM
  - ❏ Reduce required number of predictions
- Input combination of images and predictions into U-Net to fill gaps



With More Time

- Train ConvLSTM with a custom loss function that discourages “whiting out” the background to compensate for cloud dissipations
- Additional ConvLSTM layers incorporating new features related to cloud dissipation and formation
- Try new normalization techniques for models (integers 0 or 1 instead floats from 0 to 1) or customized loss function to limit blurring of predictions

With More Time

- Remodel and continue work on the Siamese architecture
- Increase region of interest
- Work with more geostationary satellites (Landsat, MODIS, Sentinel)
- Alter models to fit RGB images

Questions?

Thank you!