Final Project

Case Study in Natural Science: Detecting Tropical Systems from Satellite Images

Gory, Brendon

CSCI E-89 Deep Learning, Fall 2023

Harvard University Extension School

Prof. Zoran B. Djordjević

Introduction

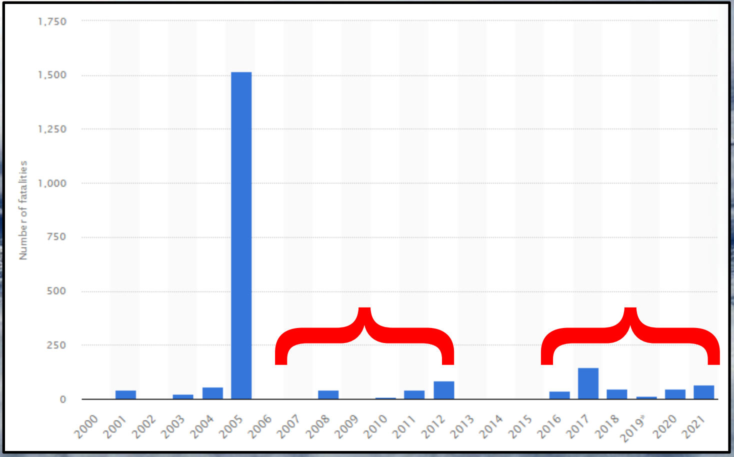
- Climate change has made tropical storms more numerous and stronger. [1]

(b)	Disaster type	Year	Country	Economic losses (in US$ billion)
1	Storm (*Katrina*)	2005	United States	163.61
2	Storm (*Harvey*)	2017	United States	96.94
3	Storm (*Maria*)	2017	United States	69.39
4	Storm (*Irma*)	2017	United States	58.16
5	Storm (*Sandy*)	2012	United States	54.47
6	Storm (*Andrew*)	1992	United States	48.27
7	Flood	1998	China	47.02
8	Flood	2011	Thailand	45.46
9	Storm (*Ike*)	2008	United States	35.63
10	Flood	1995	Democratic People's Republic of Korea	25.17

WMO | Most expensive disasters from 1970-2019.

- Tropical storms rank as the costliest weather disasters. [2]

- Despite better weather forecasting, US experienced more deaths 2016-2021 than 2007-2012. [3]



- ***Can AI detect storms prior to visual confirmation on satellite images?***

[1] https://climate.nasa.gov/explore/ask-nasa-climate/2956/how-climate-change-may-be-impacting-storms-over-earths-tropical-oceans/
[2] https://news.un.org/en/story/2021/09/1098662
[3] https://www.statista.com/statistics/203729/fatalities-caused-by-tropical-cyclones-in-the-us/

Software and Technology

- Python (Version 3.10.13)
- Jupyter Notebook
- Tensorflow (Version 2.10.1)
- Python Packages:
  - PANDAS
  - Matplotlib
  - Numpy
  - Scikit-learn
  - Pillow
  - BeautifulSoup
- Operating Systems:
  - Windows 11
- Computer:
  - 64GB RAM (3600 DIMM)
  - Intel Core i9-9900K (8 core)
  - NVIDIA GeForce RTX 3060 (12 GB)

Image Sources:

https://www.clipartkey.com

http://clipart-library.com

https://www.jagranimages.com/

https://blog.ryanwcummings.com

https://media.licdn.com/

https://becominghuman.ai

Download Data Set

Get all available websites from nhc.noaa.gov from 7/9/2014 to 11/30/2023

```
%%time

url_bases = {'https://archive.org/wayback/available?url=nhc.noaa.gov/gtwo.php?basin=atlc&fdays=2&timestamp=':'atlc',
             'https://archive.org/wayback/available?url=nhc.noaa.gov/gtwo.php?basin=epac&fdays=2&timestamp=': 'epac'}

df = pd.DataFrame()
locations, urls, json_responses, timestamps, outlook_texts = [], [], [], [], []

for url_base, loc in url_bases.items():
    start_date = datetime.strptime('20140709', '%Y%m%d')
    end_date = datetime.strptime('20231130', '%Y%m%d')

    curr_date = start_date

    while curr_date <= end_date:
        date_str = curr_date.strftime('%Y%m%d')
        curr_date += timedelta(days=1)
        url = f'{url_base}{date_str}'

        # Send a GET request to the URL
        response_for_image = requests.get(url)

        # Check if the request was successful (status code 200)
        if response_for_image.status_code == 200:
            json_string = response_for_image.text.replace("'", "\"")
            json_data = json.loads(json_string)
            snapshot_data = json_data['archived_snapshots']

            if len(snapshot_data) > 0:
                url_value = snapshot_data['closest']['url']

                if not url_value in urls:
                    # Ensure snapshot is unique

                    # Parse the HTML content with BeautifulSoup
                    try:
                        response_for_text = requests.get(url_value)
                        soup = BeautifulSoup(response_for_text.text, 'html.parser')
                        pre_tag = soup.find('pre')
                        if pre_tag:
                            # Find the specific text based on the HTML structure
                            outlook_texts.append(soup.find('pre').get_text().replace('\n','|'))
                            locations.append(loc)
                            json_responses.append(json_data)
                            urls.append(url_value)
                            timestamps.append(snapshot_data['closest']['timestamp'])

                            if len(timestamps) % 100 == 0:
                                print(f'Processing {loc}: {len(timestamps)} total records')

                    except requests.exceptions.RequestException as e:
                        pass
        else:
            # If the request was not successful, print an error message
            print(f"Error: Unable to fetch data. Status code: {response.status_code}\n{data_str}")

df['location'] = locations
df['JSON'] = json_responses
df['URL'] = urls
df['timestamp'] = timestamps
df['outlook_text'] = outlook_texts
df.to_csv('../data/available_urls.csv', index=False)
print(f'Days checked {len(timestamps)}')
```

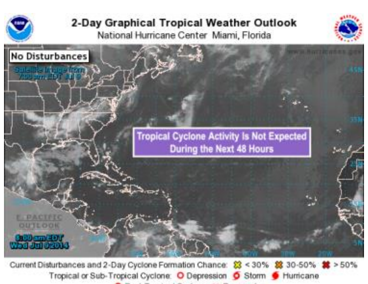
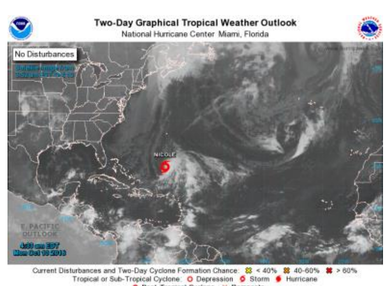
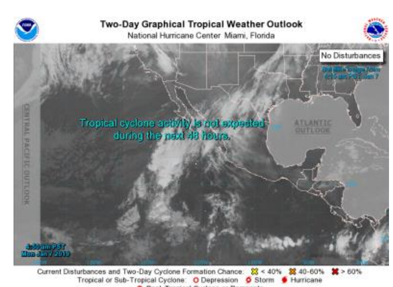
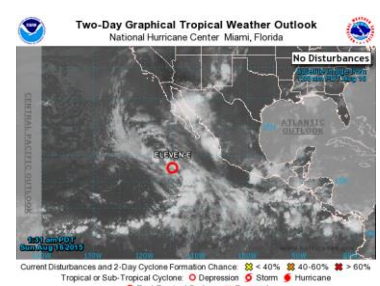
```
Processing atlc: 100 total records
Processing atlc: 200 total records
Processing atlc: 300 total records
Processing epac: 400 total records
Processing epac: 500 total records
```

- Available years by web.org 2014 – 2023
- Target website National Hurricane Center: nhc.noaa.gov
- Save JSON file with URLs to NOAA’s archived webpage that contains satellite images and forecast discussion text
- Save information in data frame to CSV file since web scraping takes time

Save Images and Assign Labels

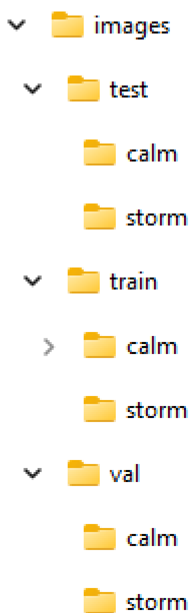
```python
### Literals for text below images with no supposed storms
no_storms = []
# Pattern 1
no_storms.append('Tropical cyclone formation is not expected during the next 5 days.')
no_storms.append('Tropical cyclone formation is not expected during the next 7 days.')

# Pattern 2
no_storms.append('During the off-season, Special Tropical Weather|Outlooks will be issued as conditions warrant.')
no_storms.append('During the|off-season, Special Tropical Weather Outlooks will be issued as|conditions warrant.')
no_storms.append('During the off-season, Special Tropical Weather Outlooks will be|issued as conditions warrant.')
no_storms.append('During the off-season, Special Tropical Weather Outlooks|will be issued as conditions warrant.')

# Pattern 3
no_storms.append('while Special Tropical Weather Outlooks will |be issued as necessary during the off-season.')
no_storms.append('while Special Tropical Weather Outlooks |will be issued as necessary during the off-season.')

for idx, row in df.iterrows():
    loc = 'pac'
    if row['location'] == 'atlc': loc = 'atl'
    timestamp = row['timestamp']
    outlook_text = row['outlook_text']

    if no_storms in outlook_text:
        # No active storms label
        label = 0
    else:
        # Active storms label
        label = 1

    image_filename = f'data/{loc}/{timestamp}_{label}.png'
    if not os.path.isfile(image_filename):
        image_url = f'https://web.archive.org/web/{timestamp}im_/http://www.nhc.noaa.gov/xgtwo/two_{loc}_2d0.png'
        response = requests.get(image_url)
        time.sleep(5)
        img = Image.open(BytesIO(response.content))
        time.sleep(5)
        img.save(image_filename)
```

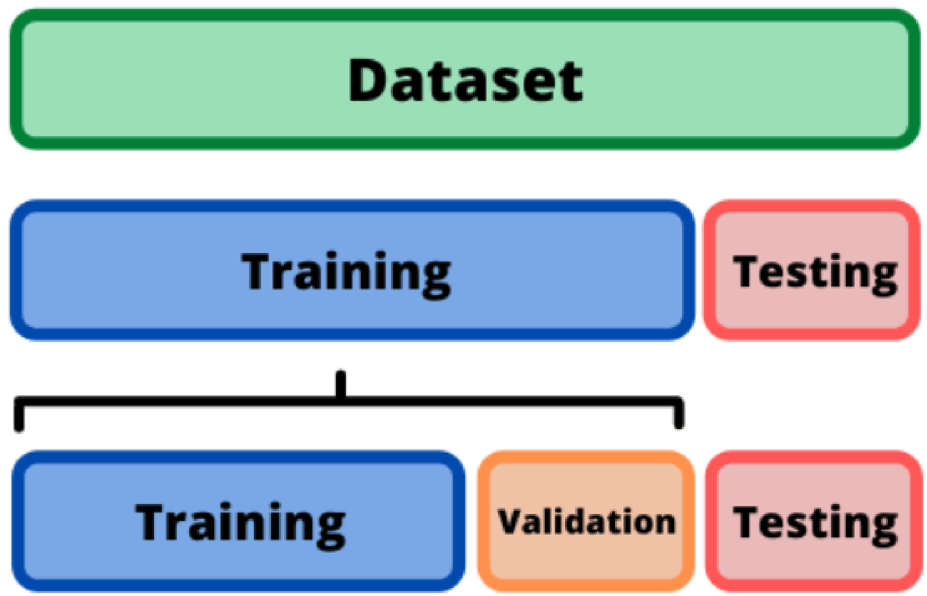
- Get satellite images for two geographic locations, Eastern Pacific and Northern Atlantic Basin
- Check forecast discussion text for literals to assign label
- Save png files locally
- 252 total Atlantic images
- 274 total Pacific images

Atlantic Calm	Atlantic Storm	Pacific Calm	Pacific Storm
			

Train, Validation & Test

- 60% of images used for training
- 30% used for validation (evaluation metrics on model)
- 10% reserved for test (predict on unseen data)
- TensorFlow ImageDataGenerator will automatically assign labels for properly organized directory structure
- Each directory has a “calm” and “storm” directory housing the appropriate images



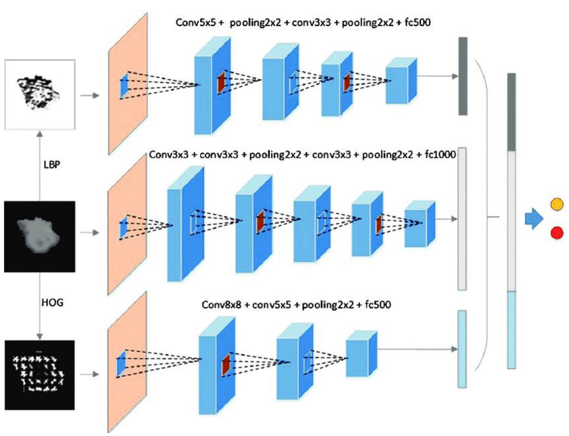


velog.io

CNN Model

- Input image 500w X 390h, 3 channels (RGB)
- 3 Convolution layers
- MaxPooling for each layer
- Increased nodes for each successive convolution
- Training parameters increase for after each convolution layer
- Dense layer for labeling output: 0, calm; 1, storm

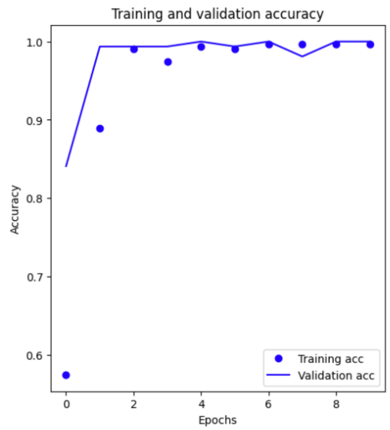
```
Model: "Base_Model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 388, 498, 32)      896

 max_pooling2d (MaxPooling2D  (None, 194, 249, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 192, 247, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 96, 123, 64)      0
 2D)

 conv2d_2 (Conv2D)           (None, 94, 121, 128)      73856

 max_pooling2d_2 (MaxPooling  (None, 47, 60, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 45, 58, 128)       147584

 max_pooling2d_3 (MaxPooling  (None, 22, 29, 128)      0
 2D)

 flatten (Flatten)           (None, 81664)             0

 dense (Dense)               (None, 512)               41812480

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 42,053,825
Trainable params: 42,053,825
Non-trainable params: 0
_________________________________________________________________
```

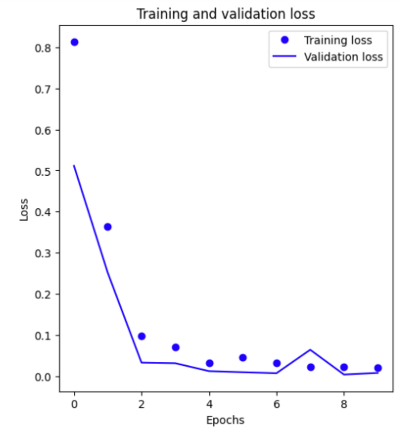


Reasearchgate.net

CNN Model Results

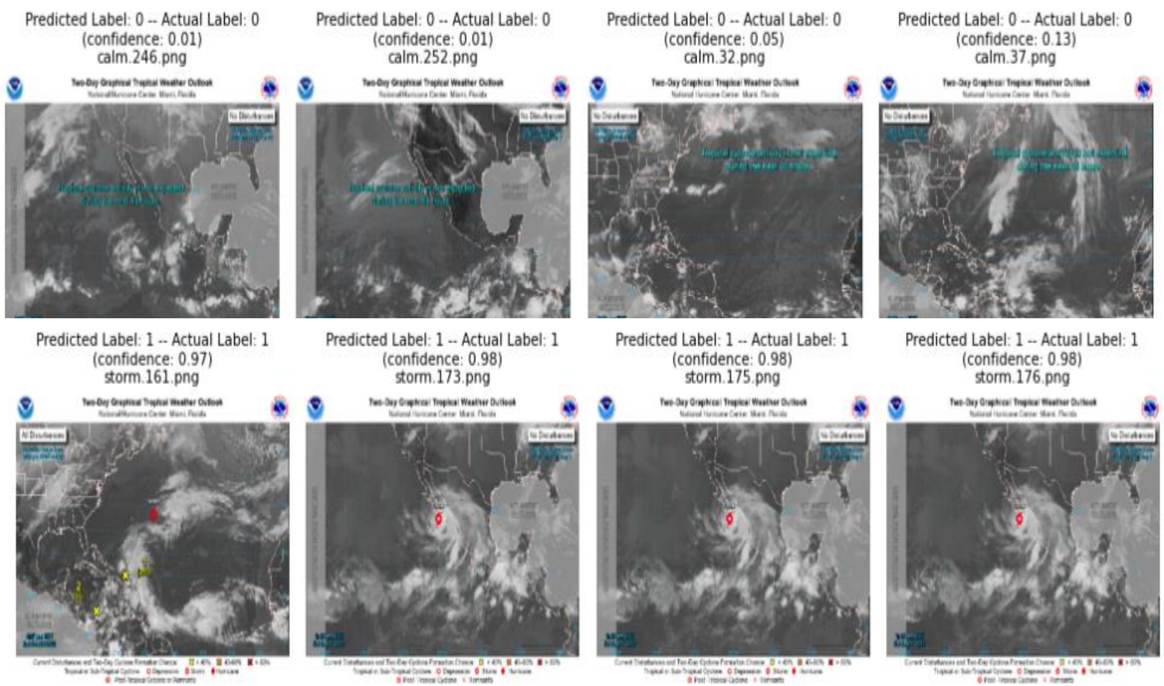
- Model quickly hit peak after a few epochs
- Possible overfitting after epoch 7
- Not enough images!





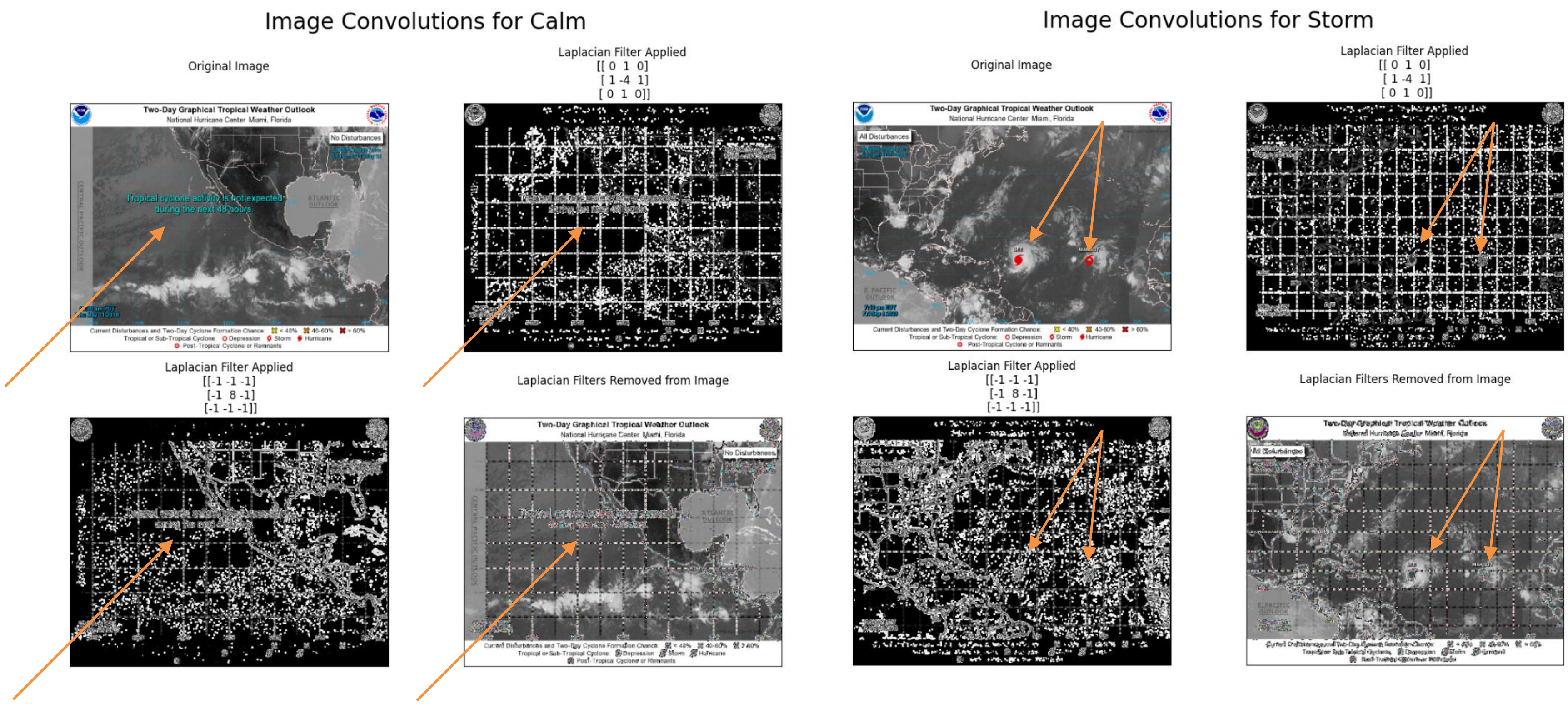
Prediction

- 100% accurate
- High confidence (threshold 0.5)
- Where is it looking?



Laplacian Edge Detection

- Apply filter to determine most prominent parts of image
- Filters of different weighting
- Calm weather might be focusing on text, not clouds
- Storms focusing on symbol, not clouds

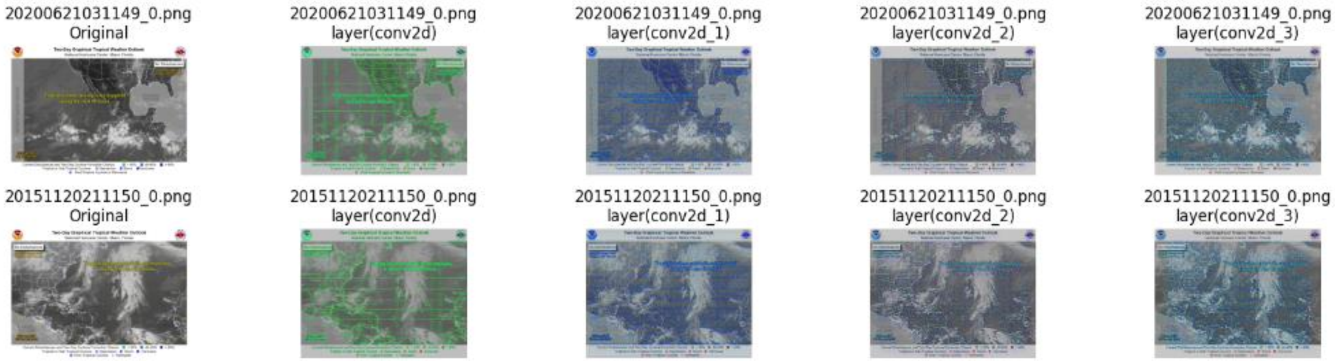


Convolutional Layer Breakdown

- Further confirmation the models seem to be training and validation on text and symbols
- These parts of the image stand out the most in each convolution

Clipartbest.com

Calm



Storm

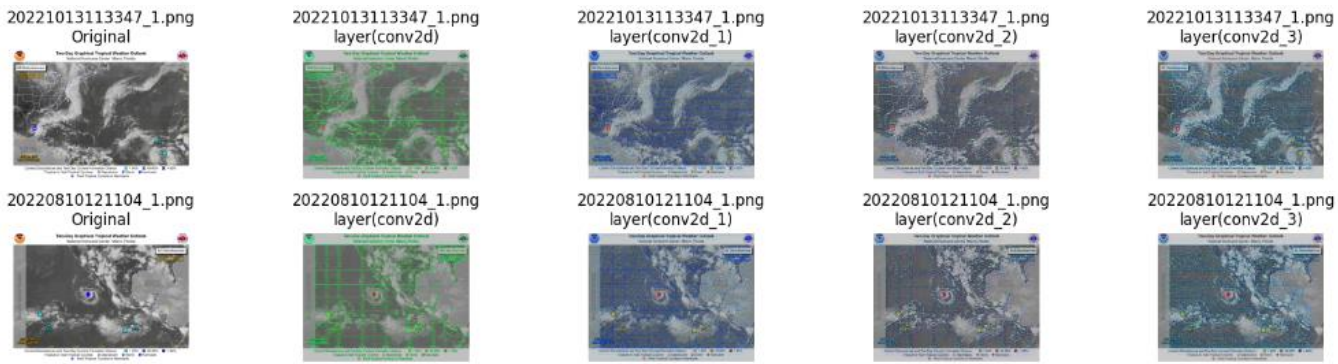
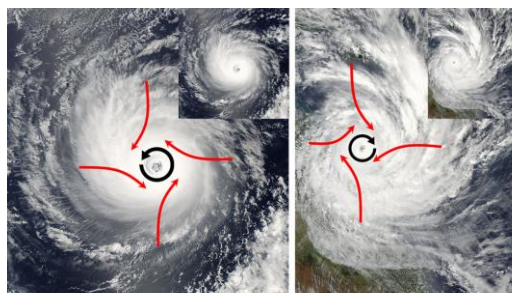


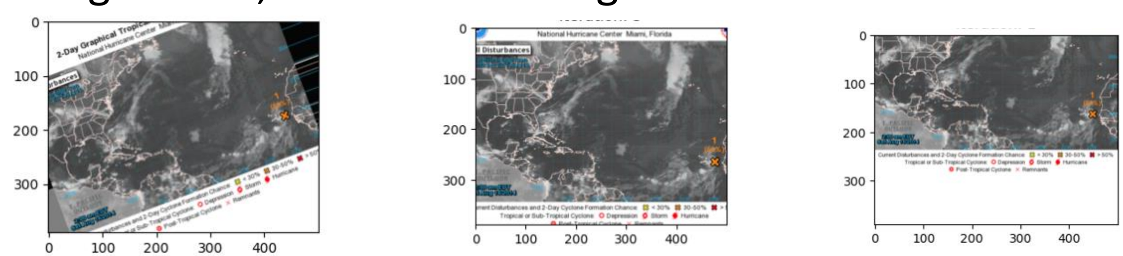
Image Augmentation

- Limited dataset
  - Images only were available since 2014 (imitation of wayback machine, not NOAA)
  - Hurricane season in Pacific and Atlantic six months of the year
- Because of a limited image dataset, alter images for the model to train
- Not all augmentations make sense
  - Tropical systems in Northern Hemisphere rotate counterclockwise
  - Tropical systems in Southern Hemisphere rotate clockwise



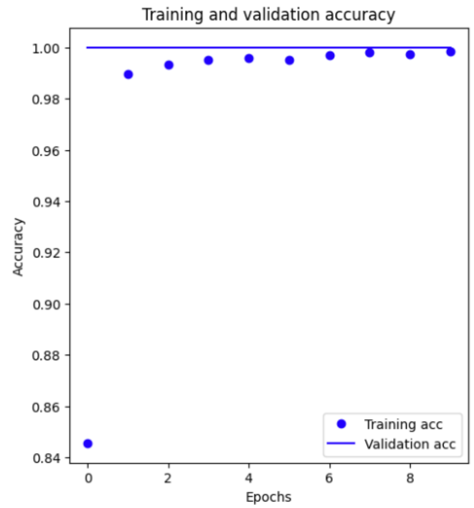
geo.libretexts.org

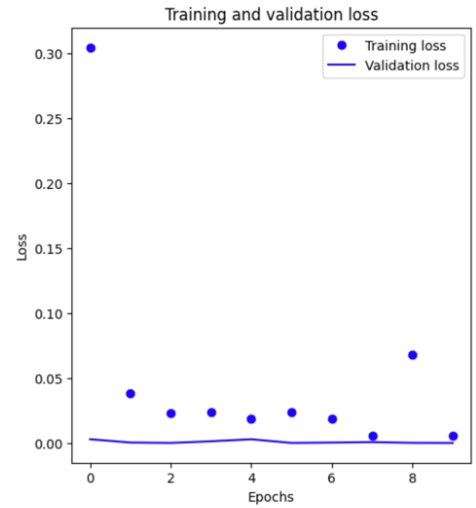
- Rotation, image width, zoom chosen augmentations



CNN Model with Augmented Images Results

- Almost immediate training accuracy after one epoch
- Signs of overfitting at epoch 8
- Model may be labeling based on orientation and not content

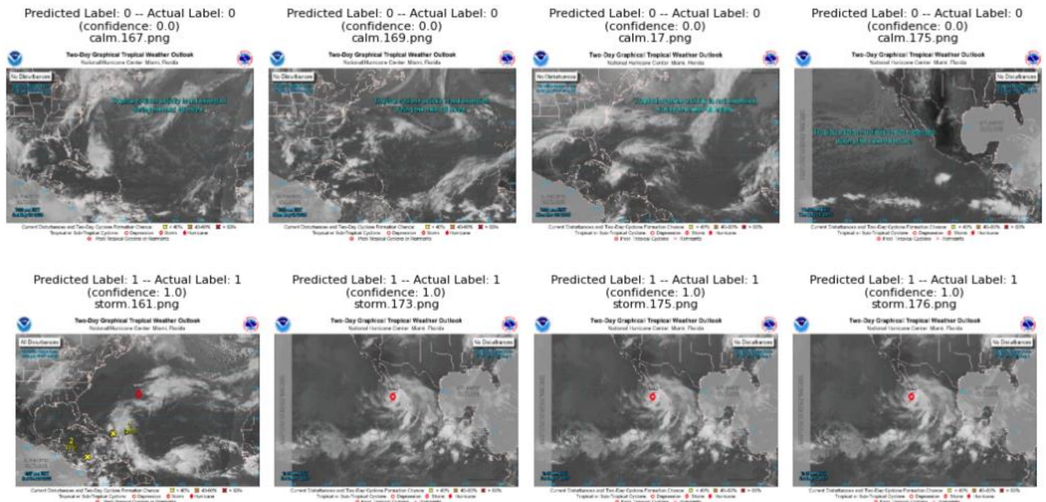




Clipartbest.com

Prediction

- 100% accurate (again)
- Complete confidence (threshold 0.5)
- Did not seem to help!



Lessons Learned

- Limited datasets can quickly overfit a model.
- Begin with a simple CNN model and build from there.
- Hyperparameter tuning along with varied, numerous images can improve accuracy and prediction.
- Layers of a convolutional network can be analyzed to visualize the innerworkings.
- The model is very accurate looking at the correct components of an image.
- Properly representative images can reduce the work of augmenting images. More is better.
- If a powerful GPU is not available, run models in a cloud environment that has a lot of computational power. TensorFlow and CNN models run very well in a parallel setting.

Future Work

- There is hope! CNN models can read satellite images before meteorologists annotate them.
- Sequence of images fed into a model with proper labeling might result in a CNN model properly forecasting a storm or at least an area of concern.
- No reason to reject the model since it trained on the wrong parts of an image. The responsibility fits on the data scientist for proper data and labeling.
- Any weather pattern in every part of the world can be interpreted by AI. Meteorologists can train models and assist in their forecasting.

Image Source: https://wikiclipart.com

YouTube URLs, Last Page

- 3 minute (short): https://youtu.be/UYajNuc7WuY
- 15 minutes (long): https://youtu.be/yBoP3bSU0dg

Image source: https://clipart-library.com