Team Ciaran

Using ML Solutions to
predict Storm behaviour

Contents

Part 1: Introduction

Part 2: The Data

Part 3: The Models

Part 4: Surprise Storm

Part 5: How to use the Software

Conclusion

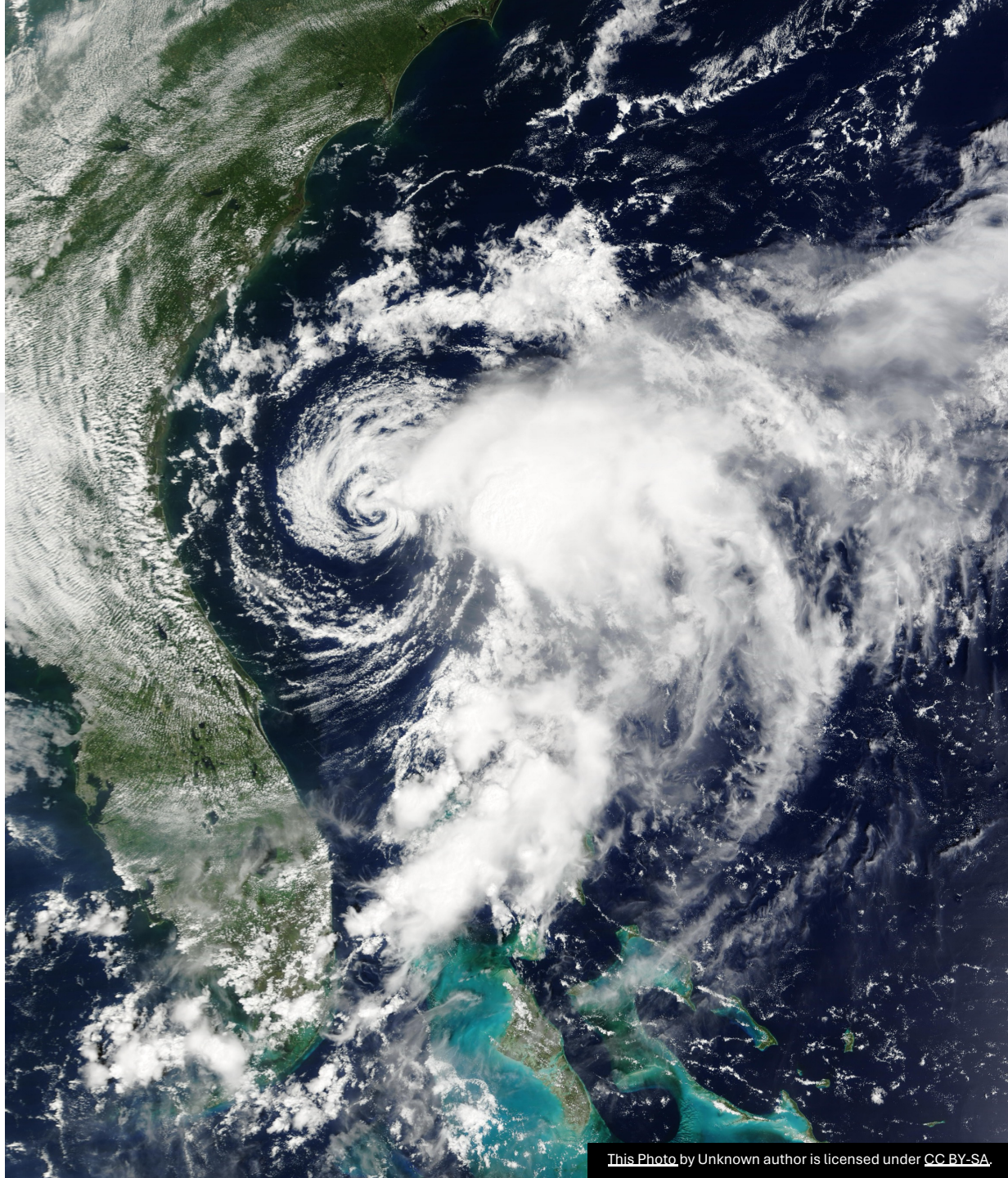
Part 1: Introduction

Motivation and Background

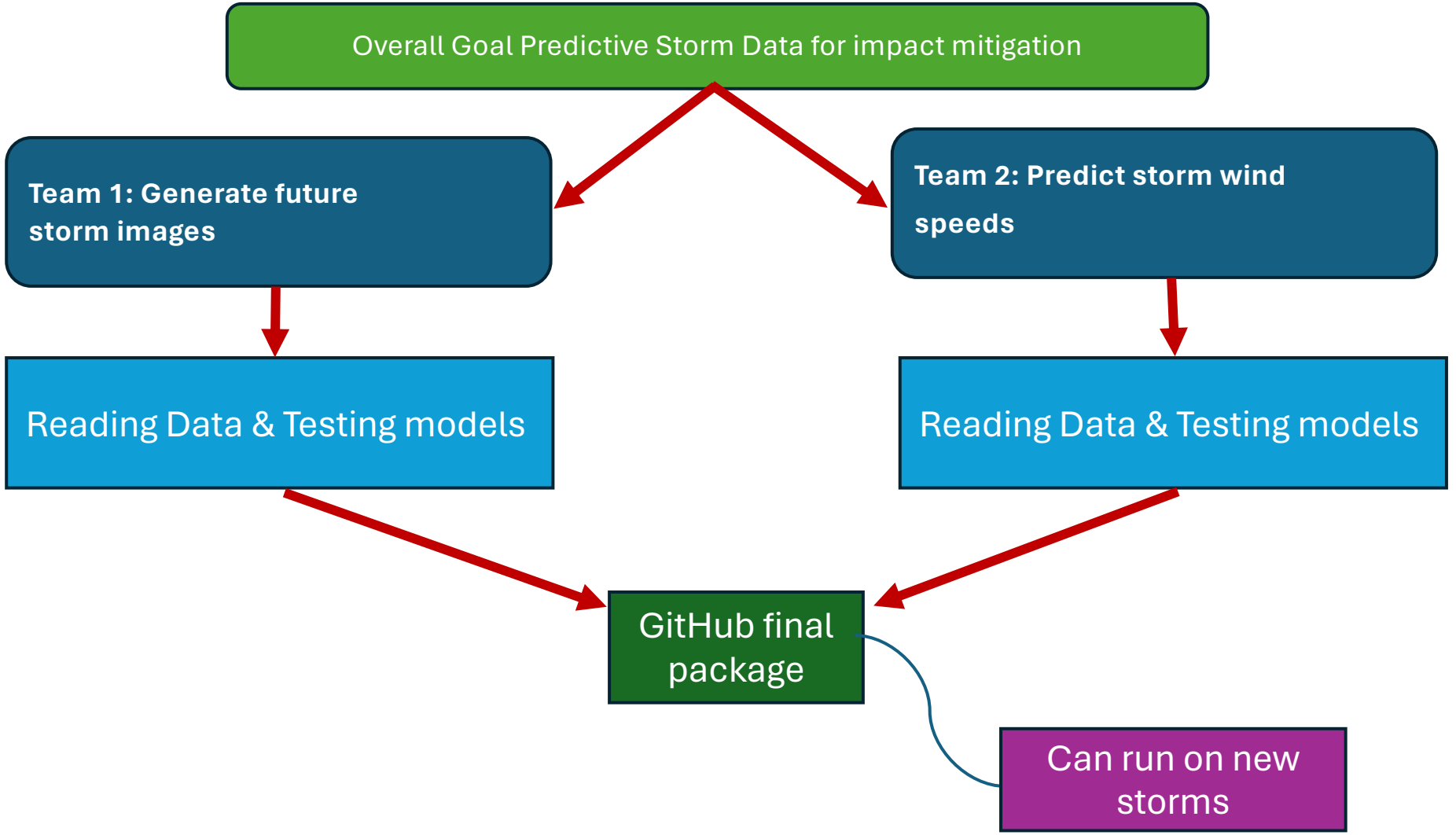
FEMA

- **Goal**: Develop ML methods to predict behavior of storm events

- Why prediction of storms important?
- Early Warning and Evacuation
- Risk Mitigation
- Infrastructure Protection
- Emergency Response Planning
- Agricultural Planning
- Scientific Research

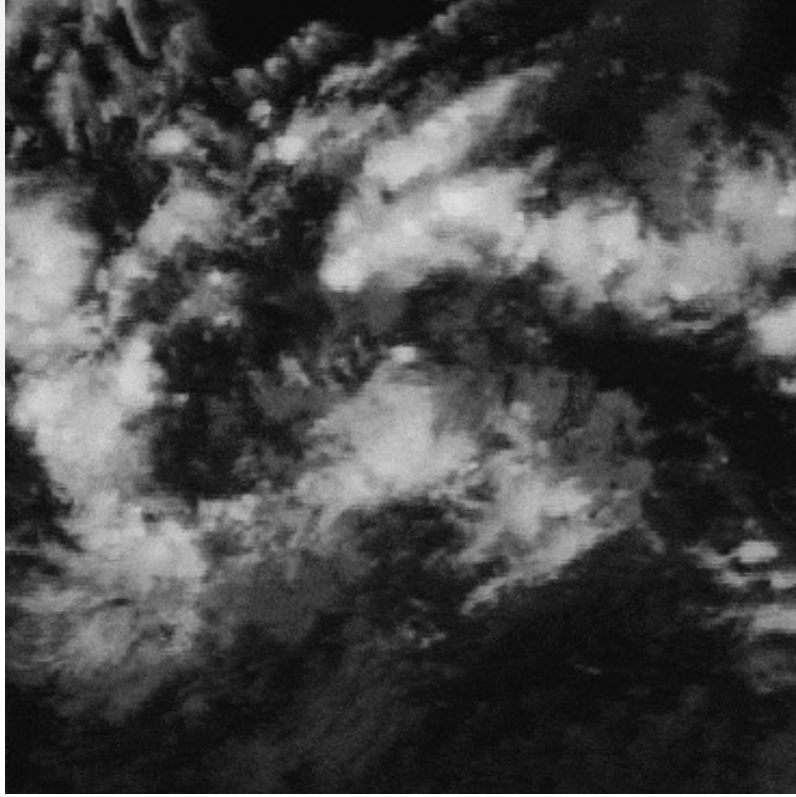


Team organisation & Workflow



Part 2: The Data

Data Cleansing



Problems

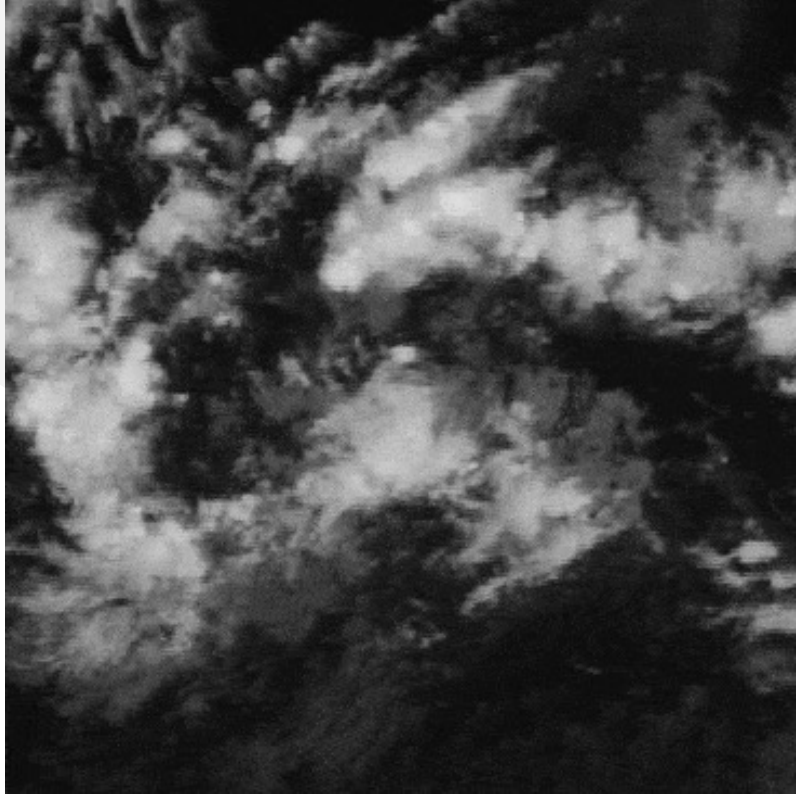
Storm images all saved in different sub-folders

Fragmented storage of storm metadata

Reduces ease of access to the data

Increases chances of conflicts in accessing the data within the team (naming conventions, method of opening the data etc.)

Data Cleansing

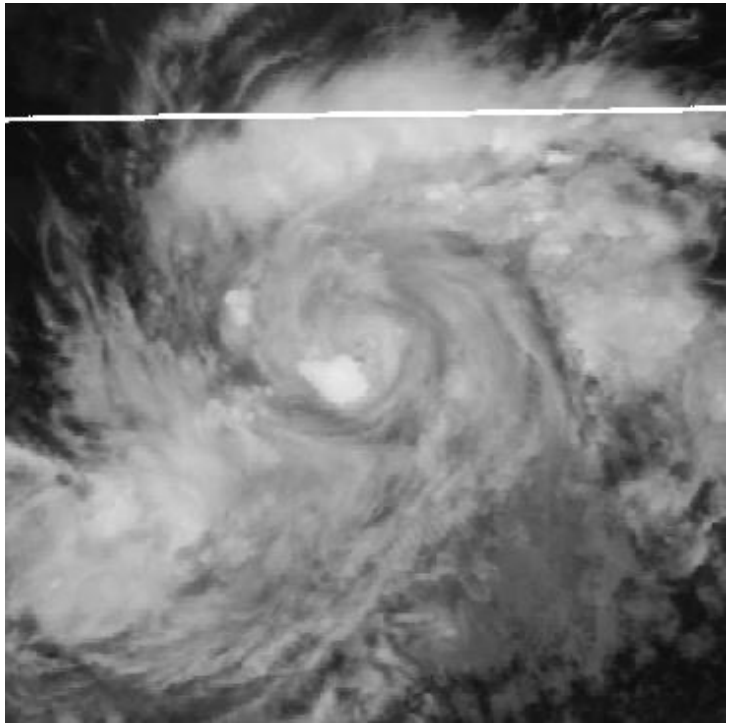


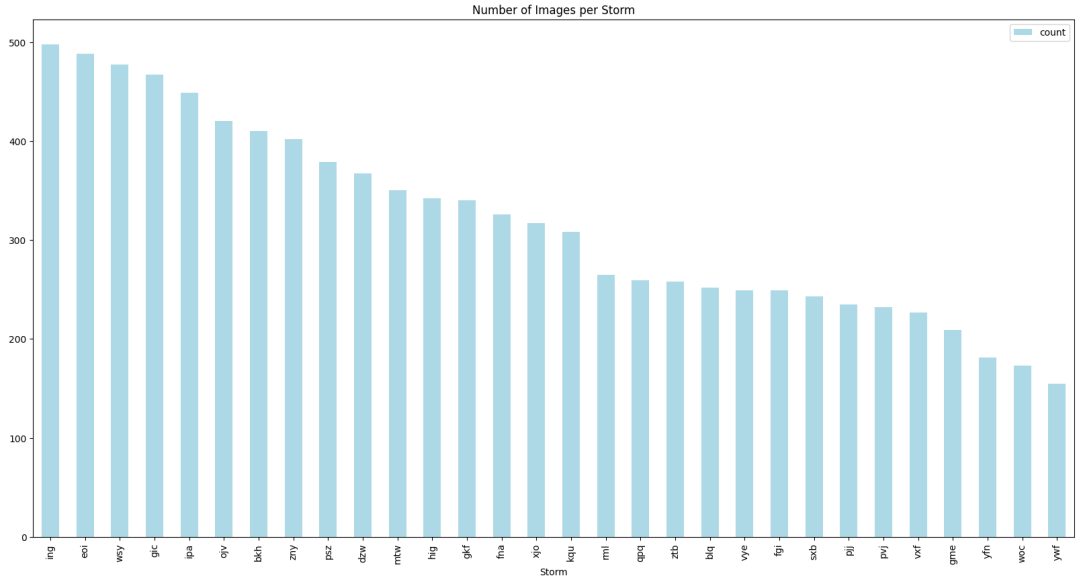
Solution

Collated all metadata of storms into a *.csv* file for easy access and manipulation

Sorted all storms images based on their relative time in a singular folder uploaded to a shared storage

Same naming convention for retrieval of data used

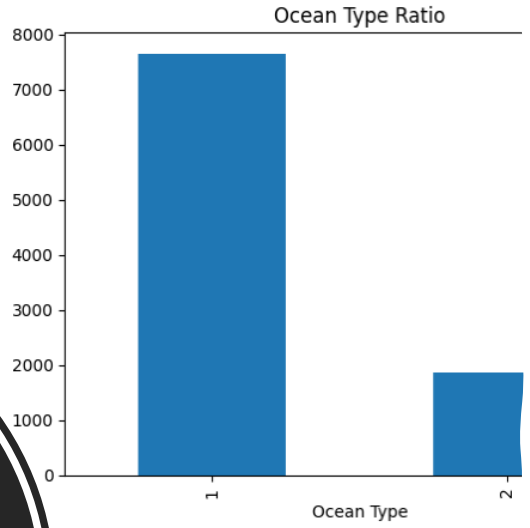


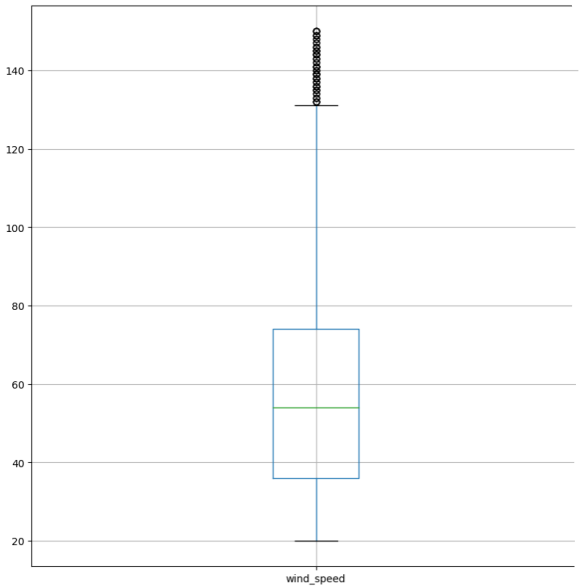


Exploring the Data - Images

- Missing pixels
- Different image sizes

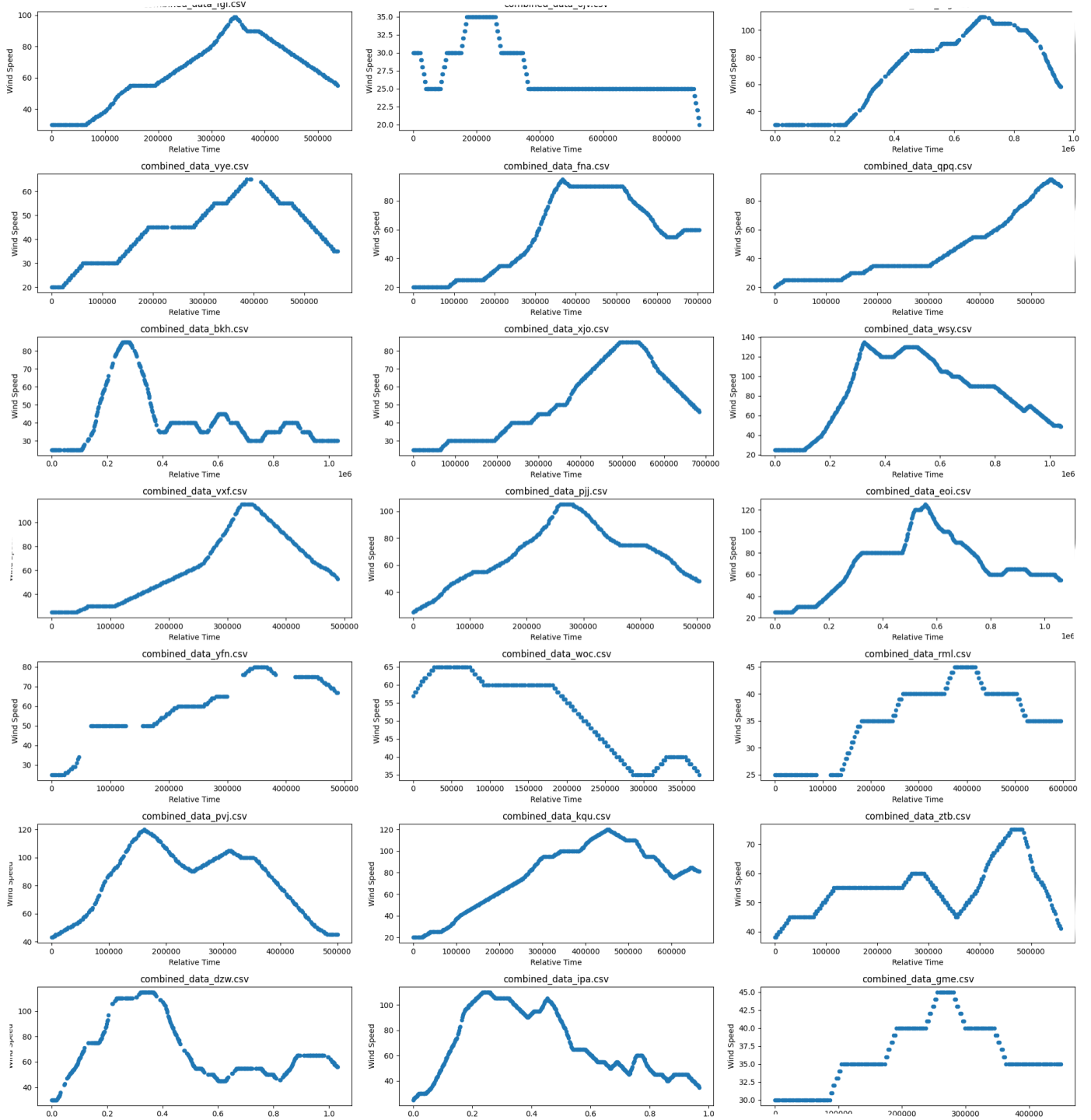
Exploring the data - metadata





Challenges:

- Very different wind speed profiles
- Large spread
- Missing Data
- Different timesteps



Storm types

- Unimodal
  - Only one peak in the speed image
  - Usually reach a higher speed at the peak
- Bimodal
  - Two peaks in the speed image
  - Usually reach a lower speed at the peak
- Other
  - Some storms lack some values, so they might not be the suitable training data

Data analysis - feature extractor

```python
class FeatureExtractor(nn.Module):
    def __init__(self):
        super(FeatureExtractor, self).__init__()
        resnet = models.resnet18(pretrained=True)
        self.features = nn.Sequential(*list(resnet.children())[:-1])

    def forward(self, x):
        return self.features(x)
```

- Pretrained ResNet18 model to extract 512 features
- Dimension reduction
  - Original image size: 366×366
  - Number of feature extracted using ResNet18: 512
  - Almost 1000 times reduction

	storm_id	relative_time	ocean	wind_speed	image_id	feature_1	feature_2	feature_3	feature_4	feature_5	...	feature_503	fea
0	mtw	0	1	30	mtw_000	0.944125	0.698270	0.183240	0.679063	0.177494	...	0.091661	
1	mtw	1800	1	30	mtw_001	0.749517	0.621735	0.379228	0.968356	0.125981	...	0.028216	
2	mtw	3600	1	30	mtw_002	1.092820	0.764792	0.187647	0.542171	0.093731	...	0.023827	
3	mtw	5400	1	30	mtw_003	0.719312	1.078920	0.470365	0.394211	0.036117	...	0.109494	
4	mtw	7200	1	30	mtw_004	0.477592	0.986226	0.205190	0.275070	0.062318	...	0.054120	
...	...	...	...	...	...	...	...	...	...	...	...	...	
9522	ing	1071001	1	65	ing_493	0.322762	0.273399	0.399614	0.615337	0.106063	...	0.593126	
9523	ing	1072799	1	65	ing_494	0.281159	0.573465	0.190713	1.865466	0.640884	...	0.383187	
9524	ing	1074600	1	65	ing_495	0.072380	0.411040	0.042893	1.488231	0.124031	...	0.265255	
9525	ing	1076399	1	65	ing_496	0.040104	0.287674	0.045845	1.416777	0.356741	...	0.114409	
9526	ing	1079999	1	65	ing_497	0.193574	0.505008	0.024042	1.857281	0.872901	...	0.003637	

9527 rows × 517 columns

Part 3: Models

Task 1: Generate predictive Storm Images

Objective:

Given a sequence of storm images, predict and generate three new images

Challenges:

- Memory Management
- Image accuracy
- Image sharpness
- *Generalization ability

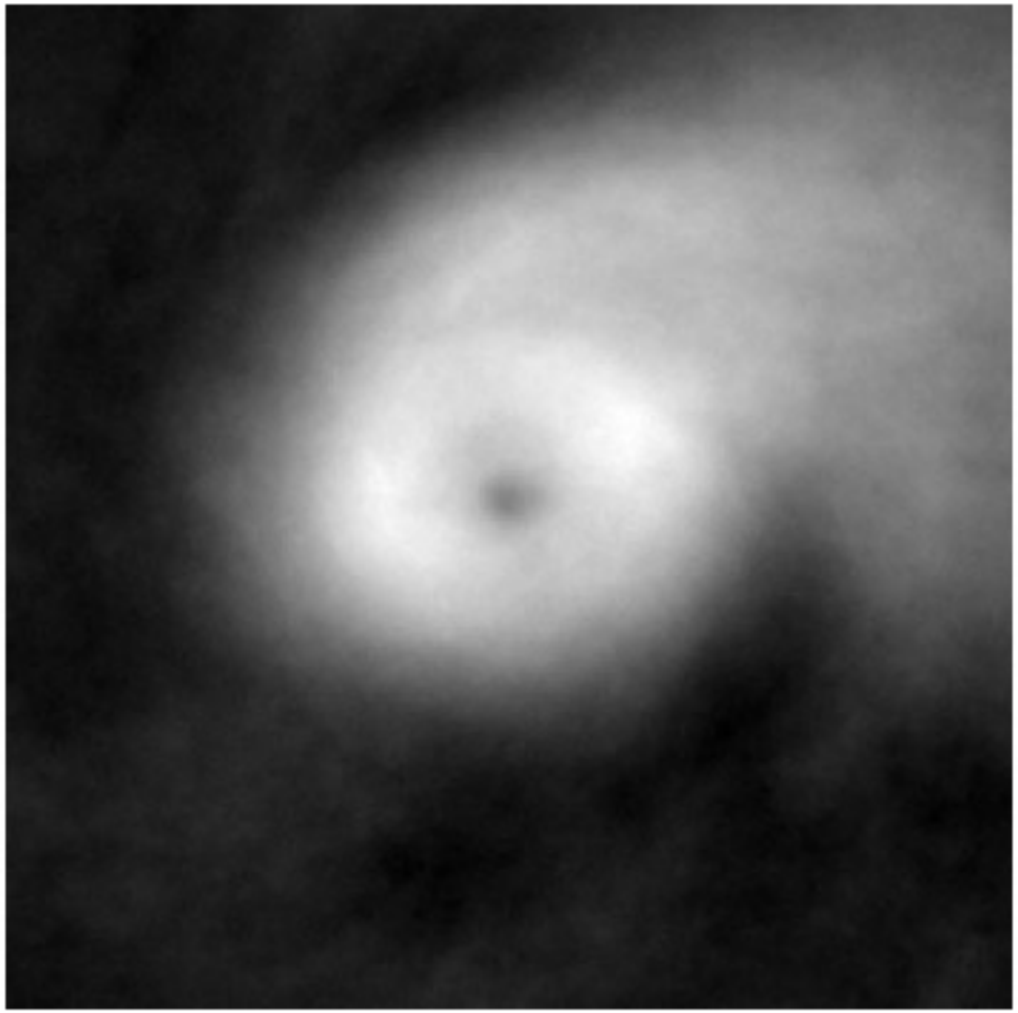
Task 1: Main Models comparison

Model Type/ Description	Average RMSE	Average SSIM	Training Time	# epochs
Encoder-decoder ConvLSTM	0.0838	**0.8199**	**~8 mins**	20
ResNet-18	**0.010371941**	0.7088	2hrs	200
3D CNN	0.0756	0.2408	~40 mins	20
Ensemble encoder-decoder ConvLSTM	0.1183	0.66171	1.5 hrs	

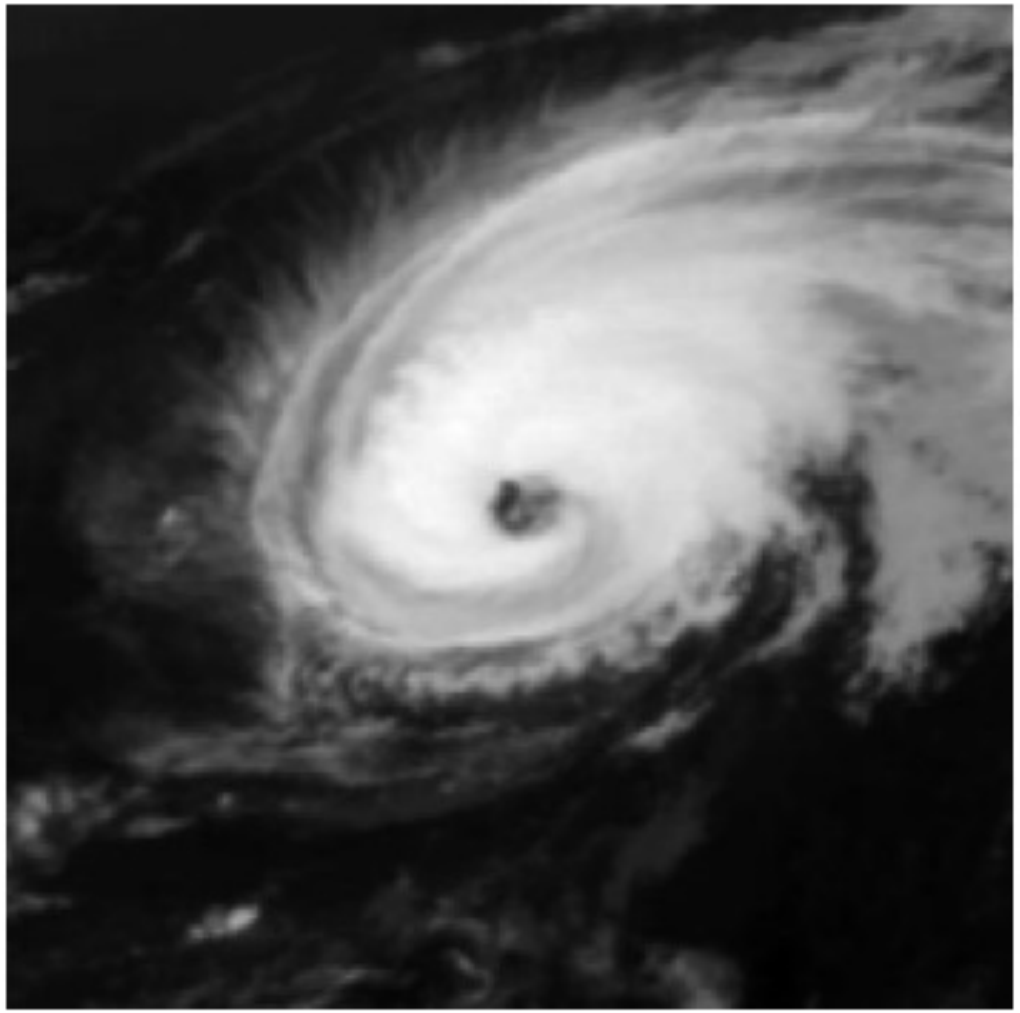
SSIM (Structural Similarity Index)

3D CNN

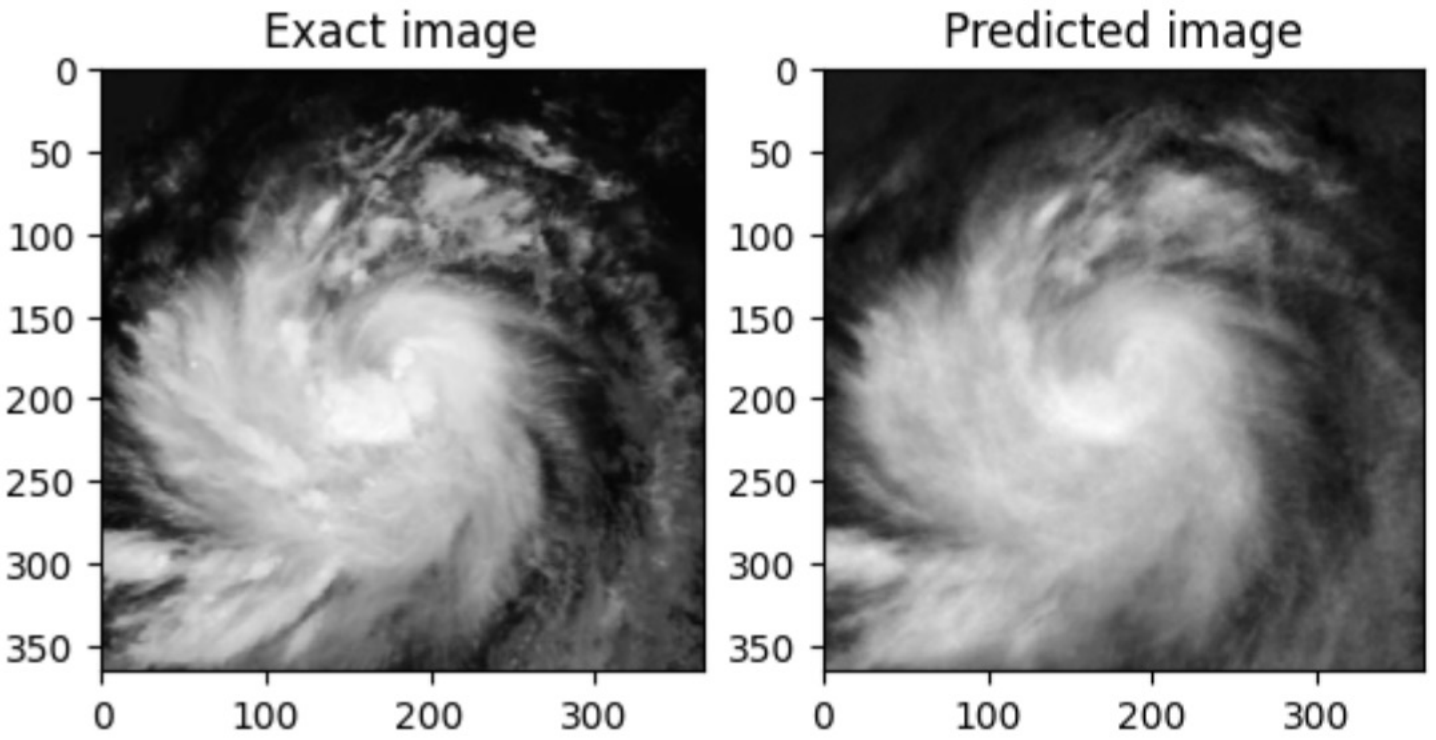
Output

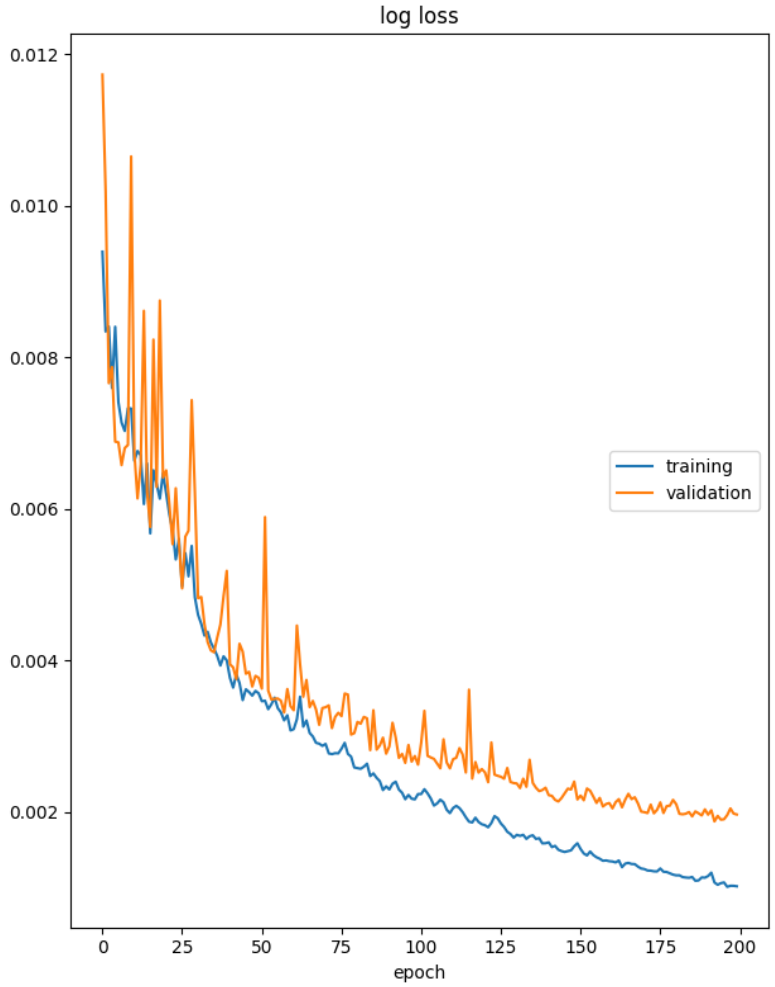


Target

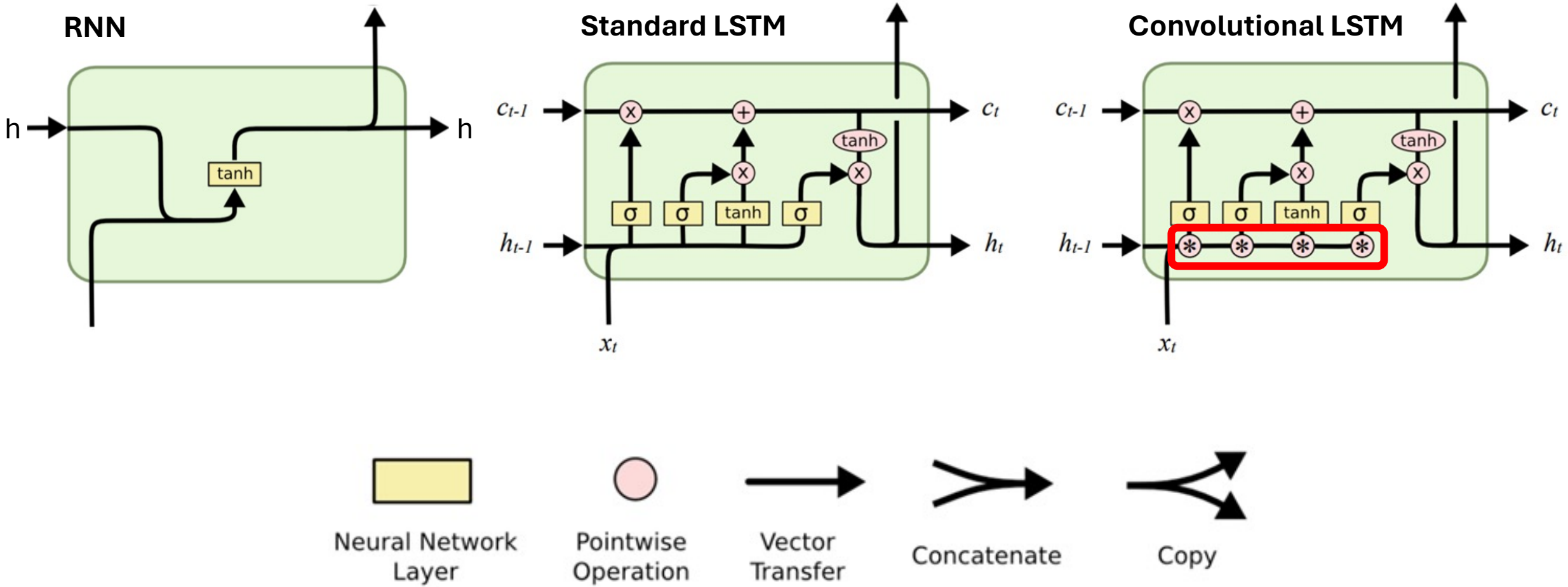


Resnet-18

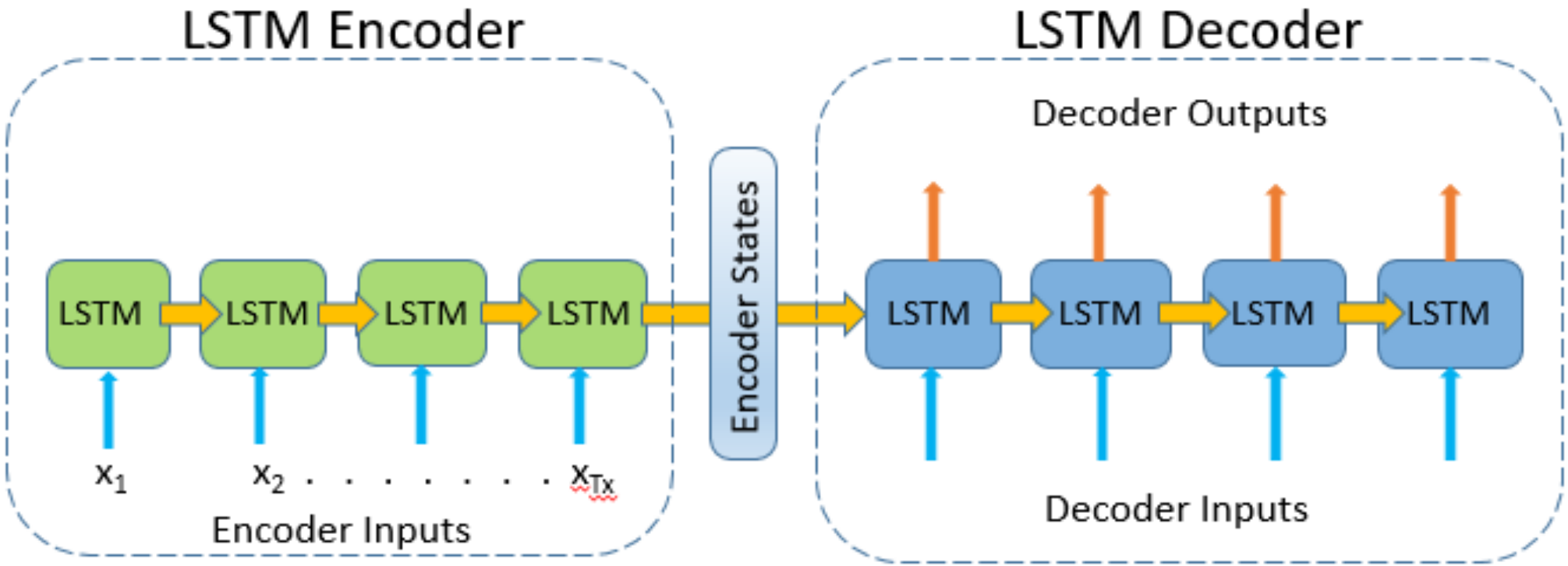

Exact image
Predicted image
0
50
100
150
200
250
300
350
0
100
200
300


log loss
0.012
0.010
0.008
0.006
0.004
0.002
0
25
50
75
100
125
150
175
200
epoch
training
validation

Convolutional Long Short-Term Memory (ConvLSTM)

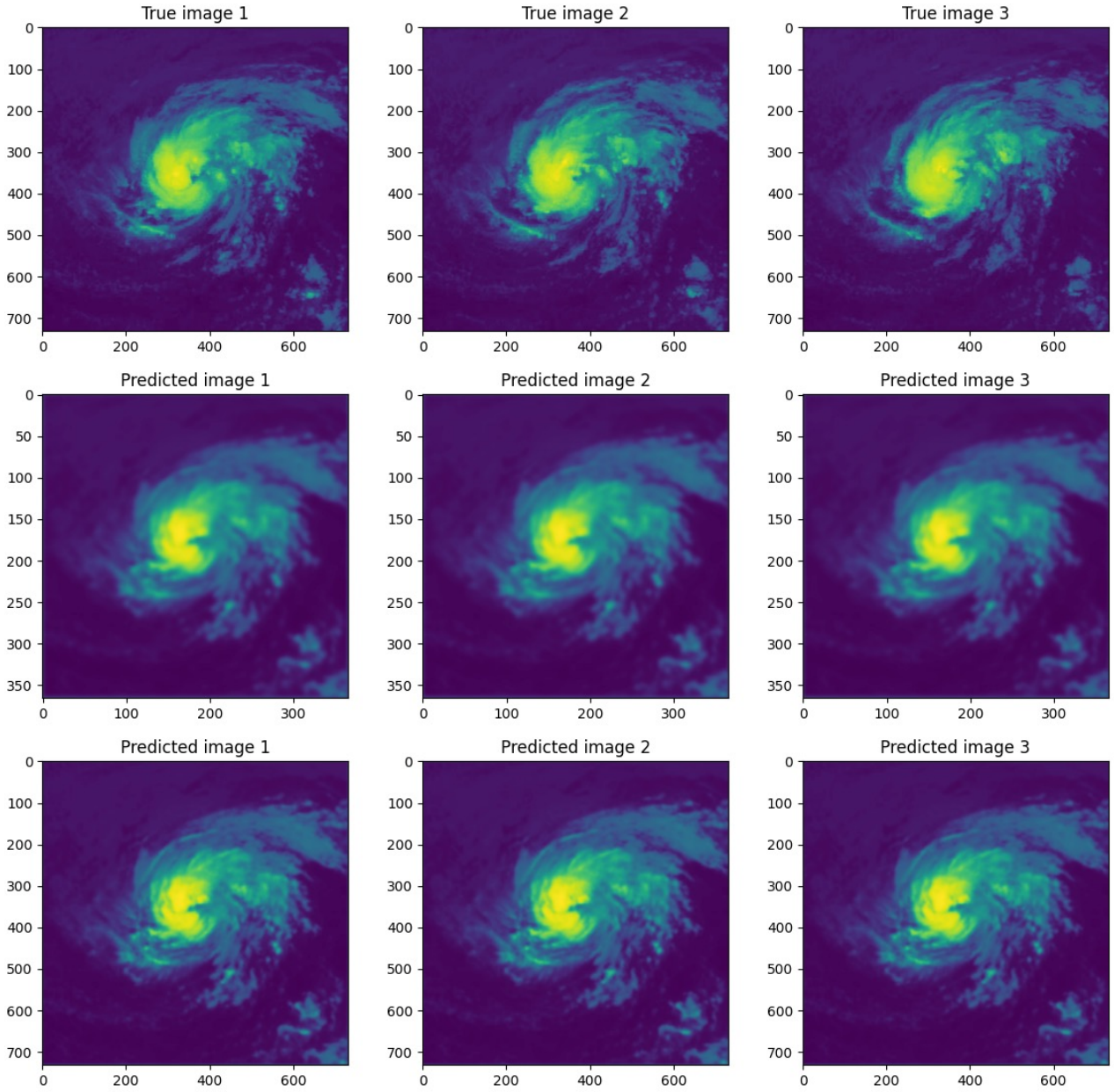


Encoder-decoder ConvLSTM



Ref: https://towardsdatascience.com/video-prediction-using-convlstm-with-pytorch-lightning-27b195fd21a2

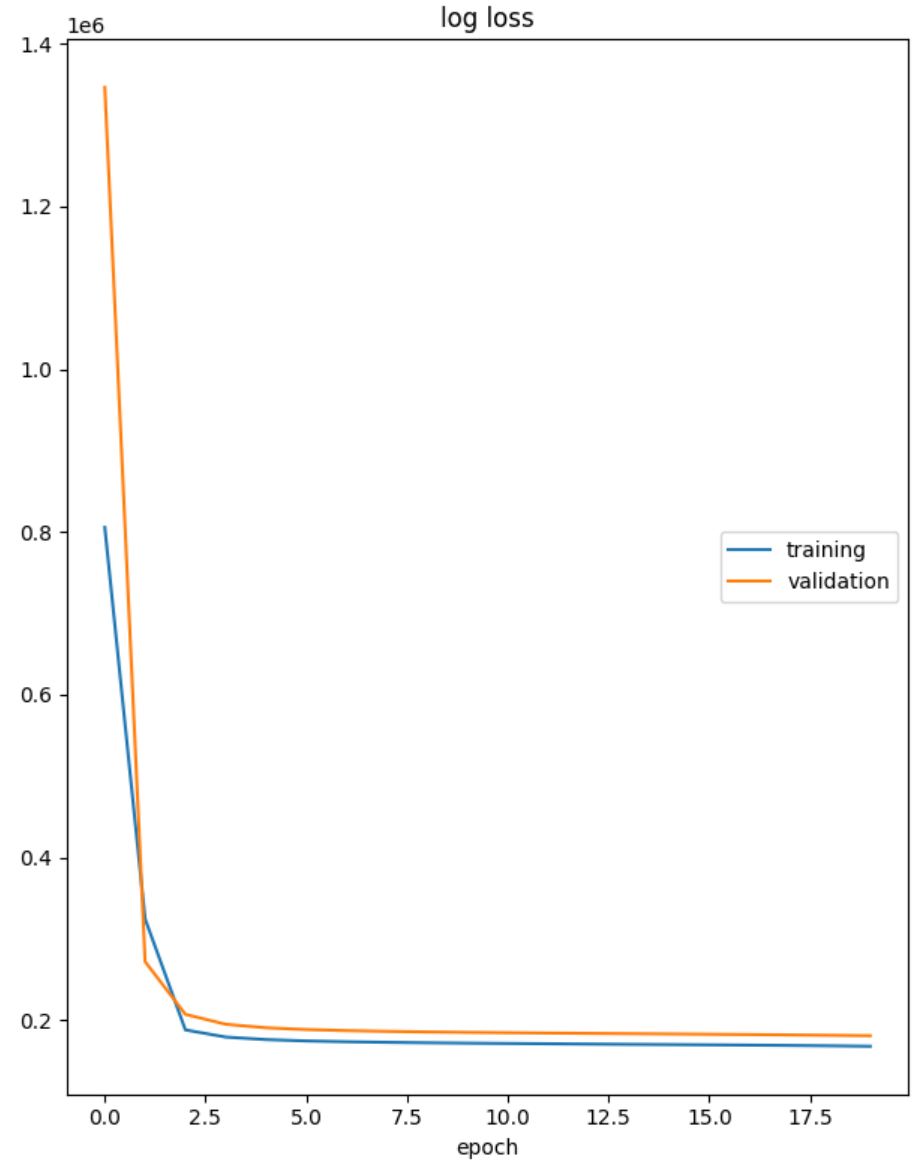
Encoder-decoder ConvLSTM



True
366x366

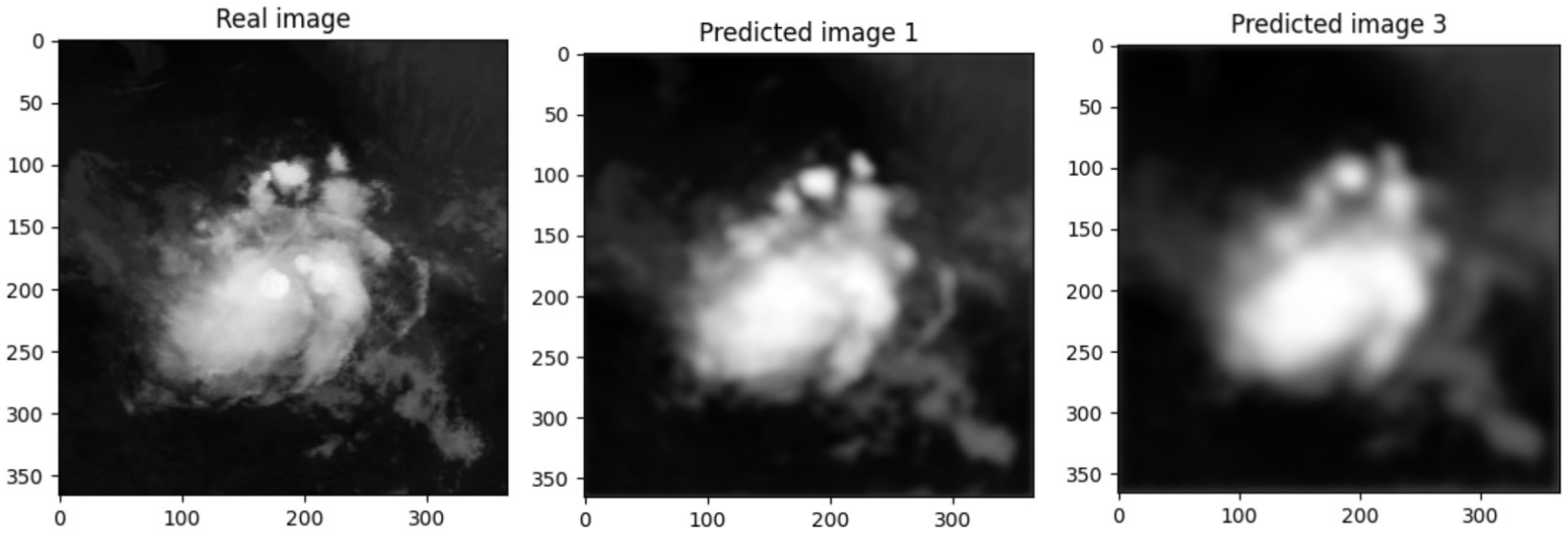
Predicted
366x366

Predicted
732x732



Ensemble Model

- Final model using Conv-LSTM architecture to predict time series images



Task 2: Predict wind speed

- Goals: using images and other features to predict the wind speed
- Metric selection justification: MSE, RMSE
  - We use the predictions and ground truth's mean square error and root mean square error as the criterion in the training and testing
- Features
  - We tried two different methods:
    - Feature extraction: using pretrained ResNet18 to generate 512 features
    - Original images: using CNN to take the original 366✖366 images as inputs

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad \mathrm{RMSE} = \sqrt{\mathrm{MSE}}$$

Task 2: Main Models comparison

Model Type/ Description	Best RMSE on whole dataset	"Training Time" on whole dataset
RandomForest: method without image analysis --> creation of cloud features using image pixels	≈ 25	9.5s
Custom Regression model: 2 layer deep neural network and image processing function.	≈ 11	2min32s
RESNET152: pre-trained deep residual learning for image recognition with task specific modifications	≈ 8	1 hour 10 mins

Task 2: Main Models comparison

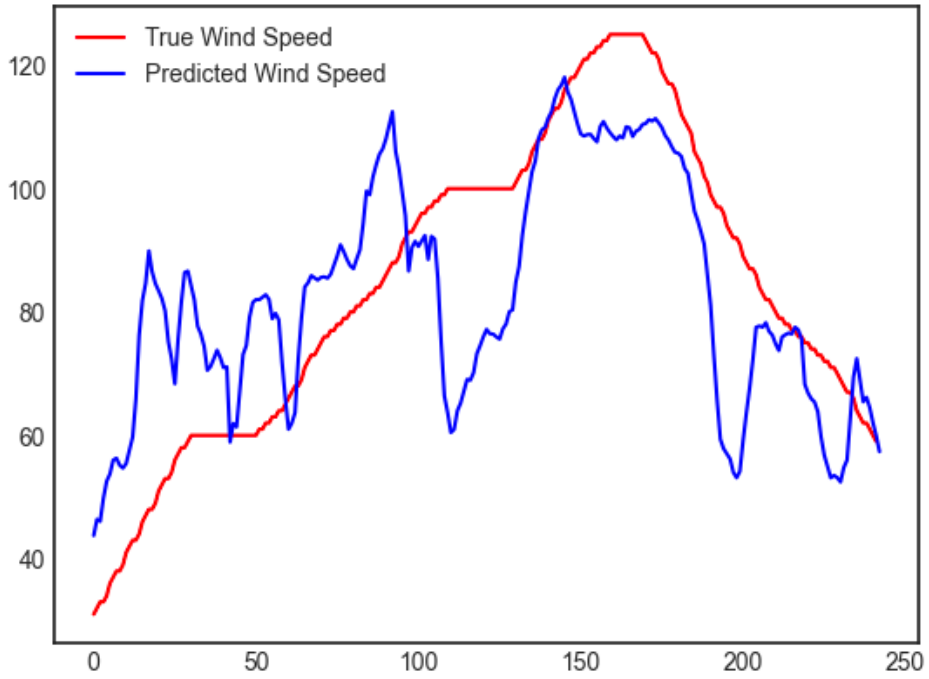
Model Type/ Description	Best RMSE on whole dataset	"Training Time"
LSTM: LSTM model with feature extractor (pretrained ResNet18)	≈ 7.5	1min
TransformerEncoder: 3 layer TransformerEncoder model with feature extractor (pretrained ResNet18)	≈ 5	19min58.3s

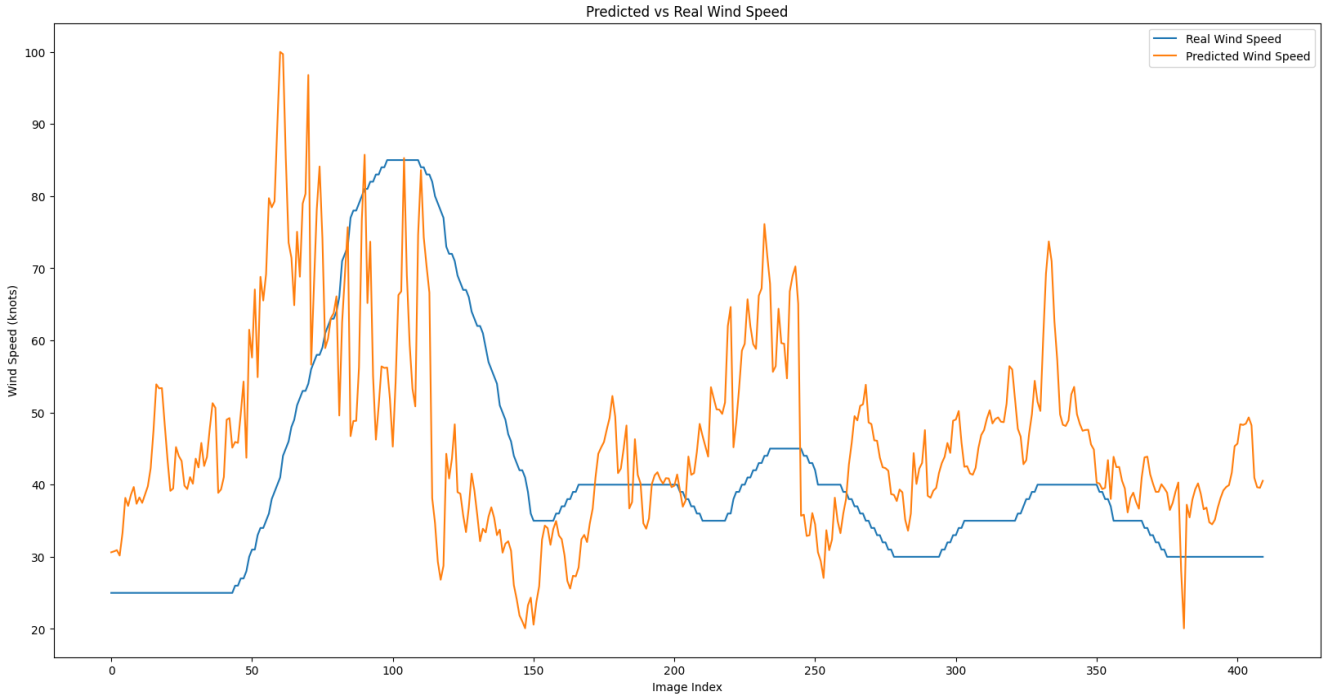
```python
class LSTMModel(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, output_size):
        super(LSTMModel, self).__init__()
        self.lstm = nn.LSTM(input_size=input_size,
                            hidden_size=hidden_size, num_layers=num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        out, _ = self.lstm(x)
        out = self.fc(out[:, -1, :])
        return out
```

```python
class TransformerModel(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, output_size, num_heads):
        super(TransformerModel, self).__init__()
        self.embedding = nn.Linear(input_size, hidden_size)
        self.encoder = nn.TransformerEncoder(nn.TransformerEncoderLayer(\
            d_model=hidden_size, nhead=num_heads, norm_first=True), num_layers=num_layers)
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x, tgt=None):
        x = self.embedding(x)
        x = self.encoder(x, tgt)
        x = self.fc(x[-1, :, :])  # Use the output of the last time step as prediction
        return x
```

Further Wind Speed models comparison

Methodology for assessing predictions and avoiding overfitting:
One whole storm removed from training and is predicted on





LSTM model predictions:

- Less oscillating, predicts using context
- Overall better predictions

Custom Regression model:

- follows trends of real data but overshooting a lot.
- Generates exclusively from image and is not time dependent (which can explain the oscillations)

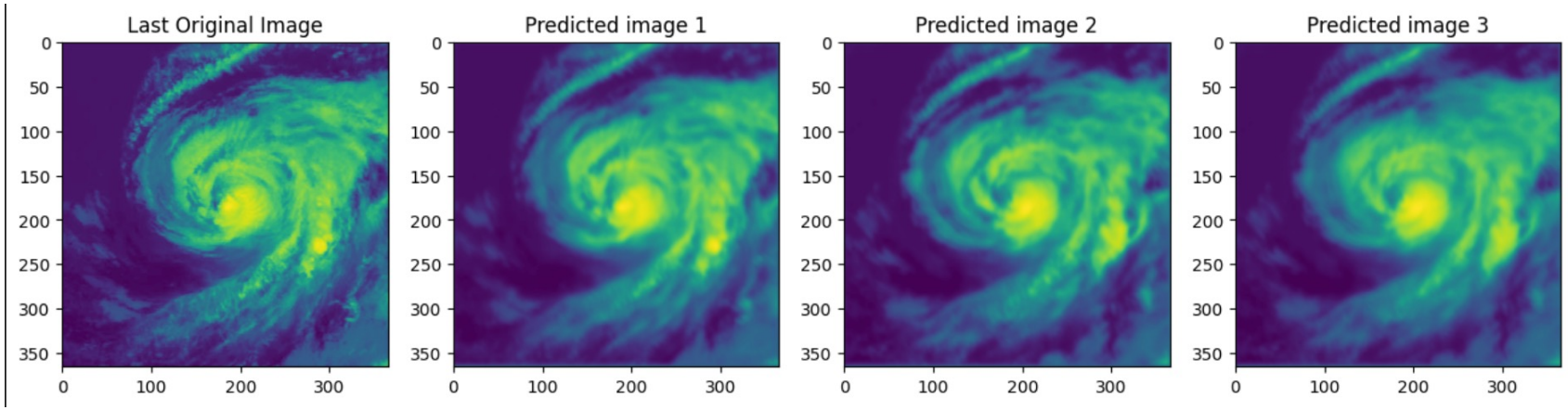
Wind speed Best model ?

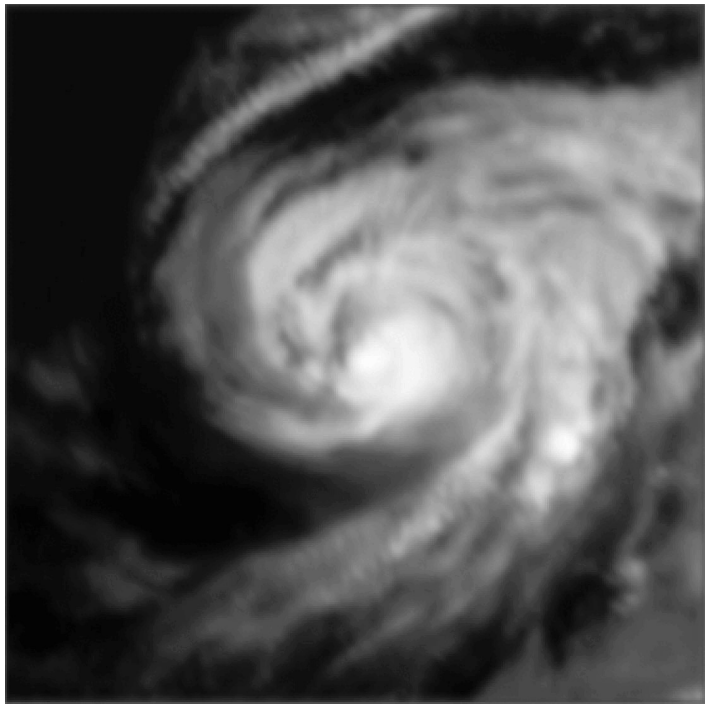
Before the Surprise storm, the best prediction model was between the ResNet152, the LSTM and the Transformer Encoder

The final model would depend on how the robust the model is when dealing with the predicted images

Part 4: Results on Surprise Storm

Surprise Storm results Generated Images- ConvLSTM





Future Scope

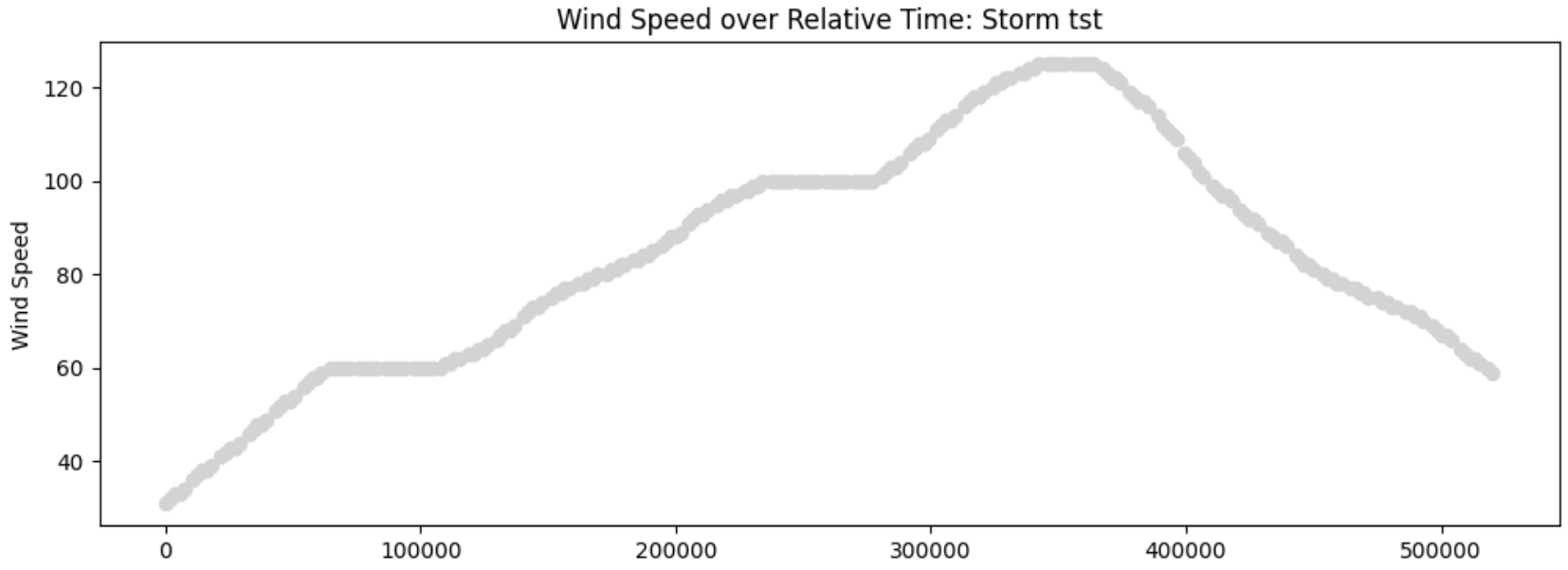
- Higher Resolution of Images
  - Can extract common features from previous images, incorporate and generate new images
  - Post-process Predictions
  - Sharpening kernel and parameters

- Try different losses for comparison
  - VGG (Visual Geometry Group) network architecture
  - Total Variation Loss
  - Edge-Enhanced Loss

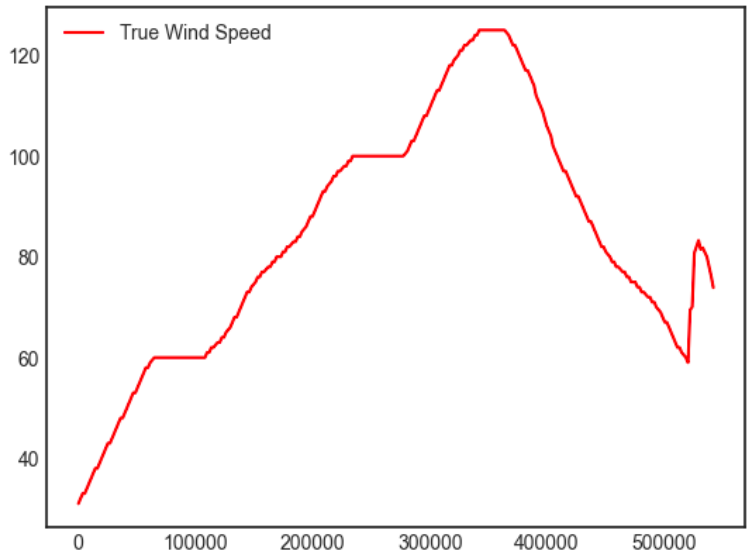
- Memory Management
  - Trying with different resolution of images

- Generalised model for all types of oceans
  - Besides Pacific and Atlantic , can train on storms in Indian Ocean, Artic Ocean and Antarctic Ocean

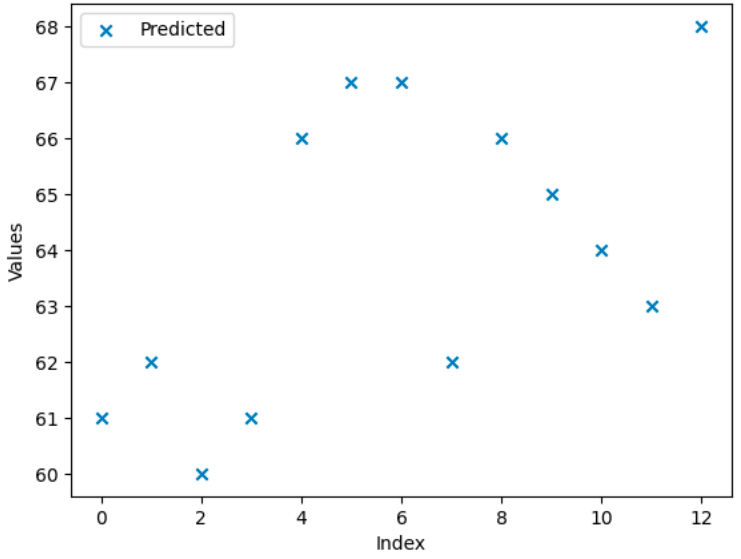
Surprise storm EDA: wind speed



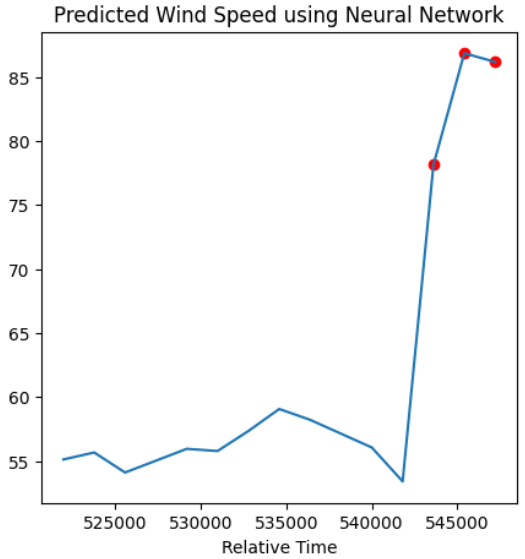
Surprise Storm results Wind Speed



LSTM Model predictions on last 10 + 3 images which were really sharp and sudden.



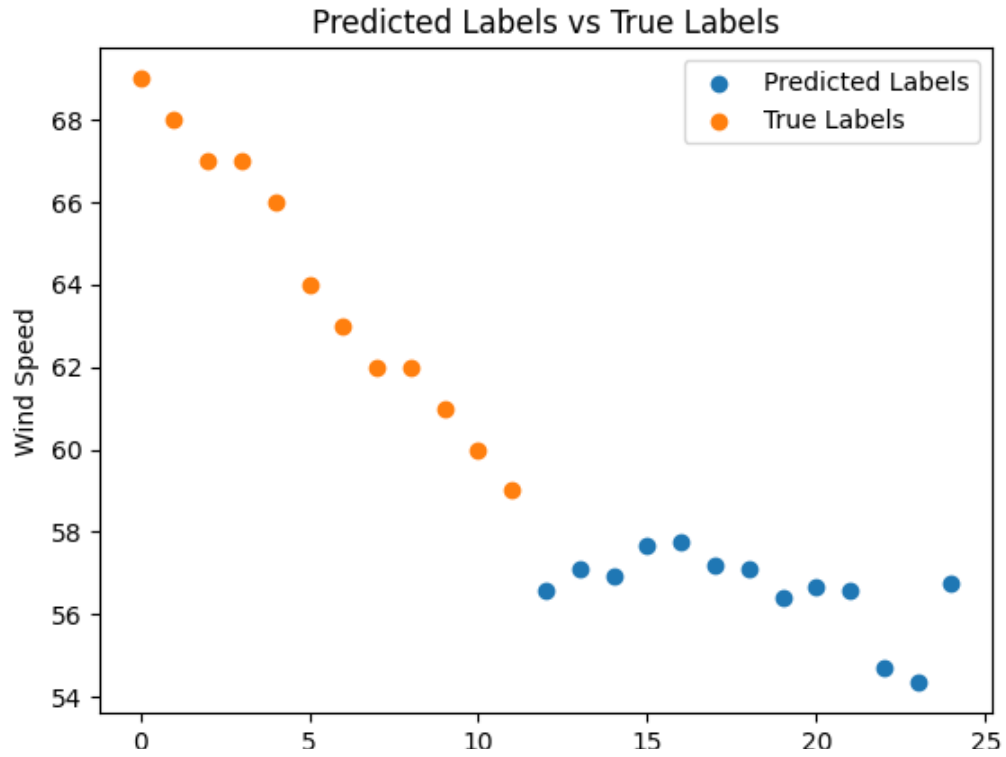
Predicted Wind Speed using ResNet + Transformer

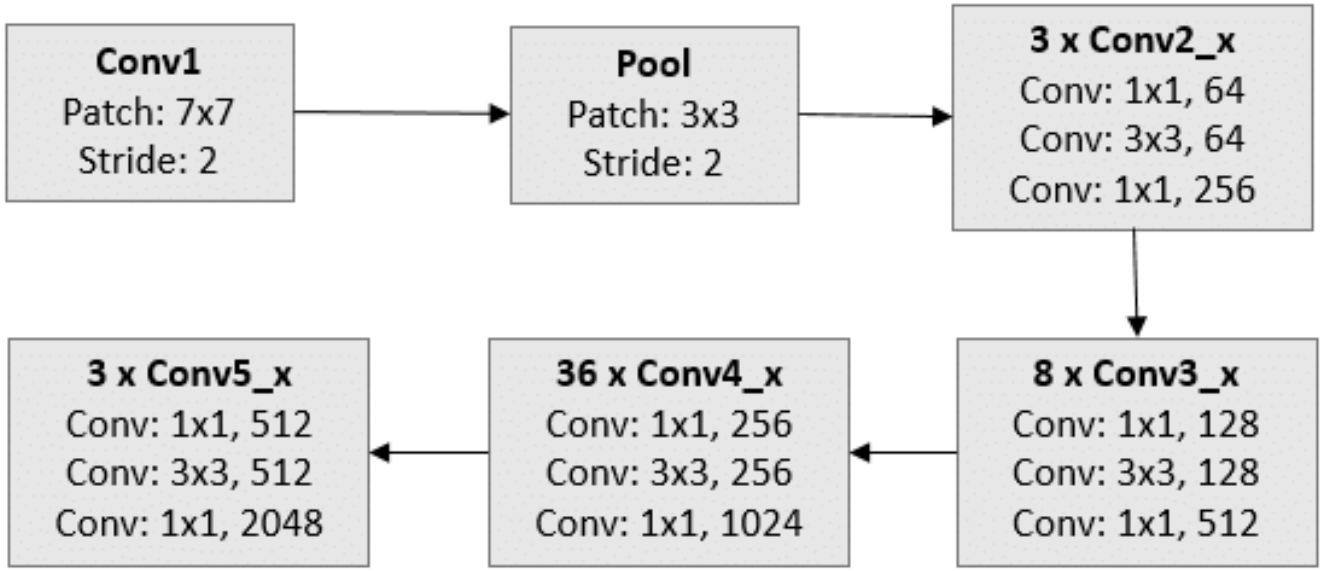


Custom Regression Model: predictions on last 10 + 3 generated images.

Best Result for surprise Storm: Resnet152

```
[[56]
 [57]
 [56]
 [57]
 [57]
 [57]
 [57]
 [56]
 [56]
 [56]
 [54]
 [54]
 [56]]
```





Future Scope

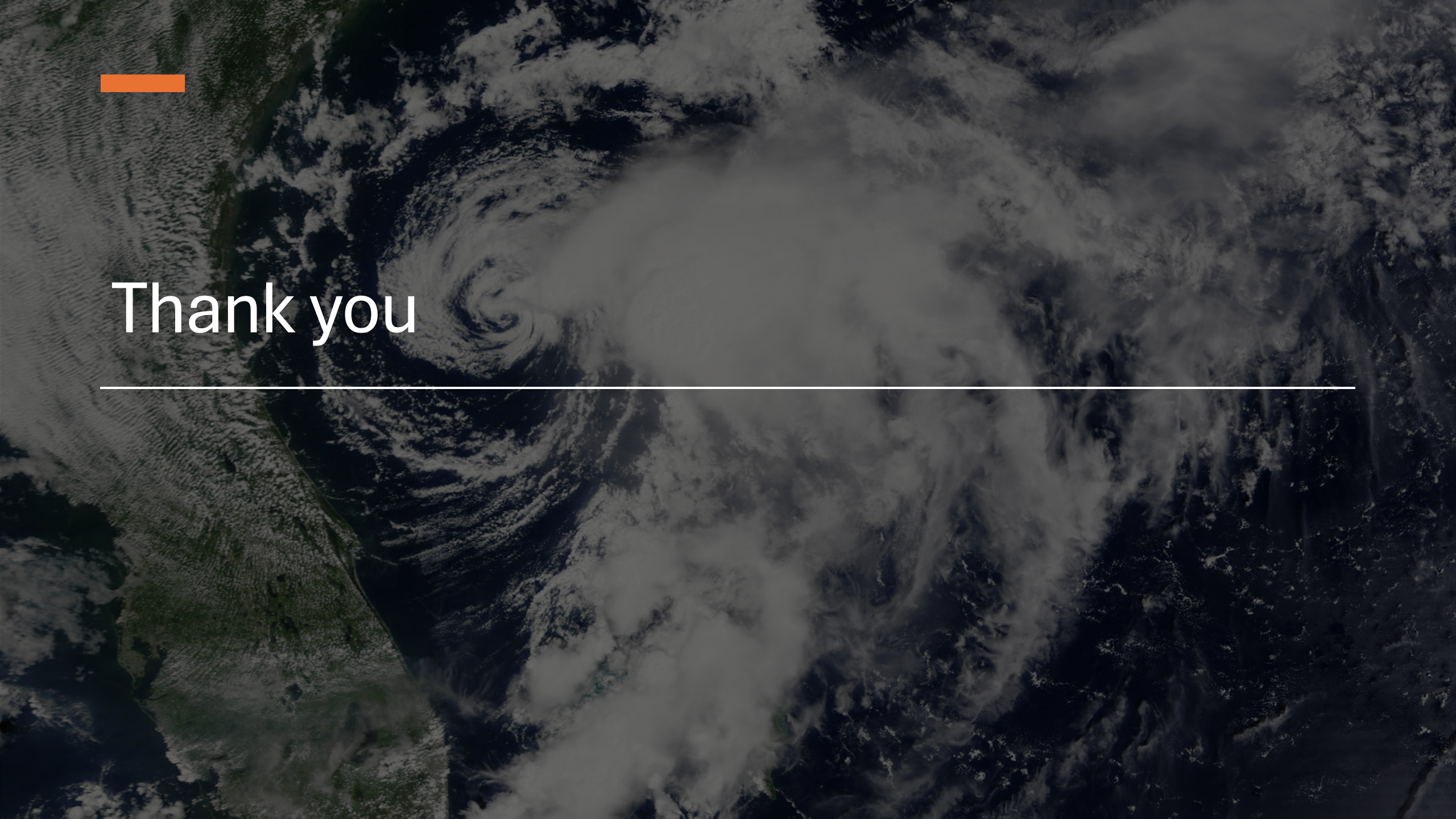
- Train model on more storms - including on storms not from Pacific/Atlantic (more versatile model)
- Could predict other features other than wind speed (ocean type, 'Risk Level': depending on urban areas affected)
- Merging multiple models for better results
- Using multiple images as one sequence for training

Part 5: How to use the software

How to use our product

Product Link:
https://github.com/ese-msc-2023/acds-the-day-after-tomorrow-ciaran.git

To generate Images	Solutions folder: Run the cells in: 'predict_storm_images_packaged.ipynb' (Uses functions in Tools)
Predict wind speeds	Solutions folder: Run the cells in: 'generated_wind_speeds.ipynb'
To find out more about the data or try different models	Miscellaneous folder (EDA, other models less efficient)
To test software	Test folder (check readme for more information on how to run them)



Thank you