Problem Statement:
Developing an AI system for predicting earthquakes, floods, and hurricanes using historical data and real-time inputs. This system also optimizes resource allocation by suggesting effective response strategies. The objectives are to enhance disaster preparedness through accurate predictions and to provide actionable insights to emergency responders and affected communities. (Banafa, 2023)

Resilience in Action: Pre- and Post-Disaster Management Systems

Anvi Tandon Jain Vidit

Abstract

This report presents the development of an AI system designed to predict earthquakes, floods, and hurricanes by leveraging historical data and real-time inputs. The core objectives are twofold: to enhance disaster preparedness through accurate predictions and to optimize resource allocation by suggesting effective response strategies. The system employs a combination of dataset training and a large language model (LLM) to generate reliable forecasts and actionable insights. By integrating advanced AI techniques with comprehensive data analysis, this project aims to provide emergency responders and affected communities with the tools needed to mitigate the impacts of natural disasters and improve overall resilience.

The system's architecture incorporates machine learning algorithms to analyse patterns in historical data, while real-time inputs are processed to update predictions dynamically. Additionally, the LLM component enhances the interpretability of predictions by offering context-aware recommendations for emergency response. This dual approach not only ensures precision in forecasting but also facilitates informed decision-making during critical situations. The findings and methodologies detailed in this report underscore the potential of AI-driven solutions in transforming disaster management and response strategies. (Banafa, 2023)

```
import pandas as pd
import geopandas as gpd
from shapely import wkt
import matplotlib.pyplot as plt
import numpy as np
```

```
conda install geopandas
```

```
[2]                                                                    Python

Channels:
 - defaults
Platform: osx-64
Collecting package metadata (repodata.json): done
Solving environment: done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

```
print(df.dtypes)
```

```
Unnamed: 0        int64
id               object
country          object
iso3             object
gwno            float64
geo_id          float64
geolocation      object
level             int64
adm1             object
adm2             object
adm3             object
location         object
historical      float64
hist_country     object
disastertype     object
disasterno       object
Shape_Length    float64
Shape_Area      float64
geometry         object
dtype: object
```

```
print(df.head)
```

```
<bound method NDFrame.head of       Unnamed: 0     id   country iso3   gwno   geo_id    geolocation  level  \
0              0    109   Albania  ALB 339.00   346.00    Ana E Malit      3
1              1    109   Albania  ALB 339.00   351.00         Bushat      3
2              2    175    Angola  AGO 540.00   760.00         Onjiva      3
3              3    187    Angola  AGO 540.00   710.00          Evale      3
4              4    187    Angola  AGO 540.00   749.00           Mupa      3
...          ...    ...       ...  ...    ...      ...            ...    ...
39948      39948  06-94  Dominica  NaN    NaN 45253.00  Saint Patrick      1
39949      39949  06-94  Dominica  NaN    NaN 45254.00   Saint Andrew      1
39950      39950  06-94  Dominica  NaN    NaN 45255.00   Saint George      1
39951      39951  06-94  Dominica  NaN    NaN 45256.00    Saint David      1
39952      39952  06-94  Dominica  NaN    NaN 45257.00     Saint Paul      1

                adm1      adm2         adm3             location  historical  \
0            Shkoder  Shkodres  Ana E Malit          Ana E Malit        0.00
1            Shkoder  Shkodres       Bushat               Bushat        0.00
2             Cunene  Cuanhama       Onjiva               Onjiva        0.00
3             Cunene  Cuanhama        Evale                Evale        0.00
4             Cunene   Cuvelai         Mupa                 Mupa        0.00
...              ...       ...          ...                  ...         ...
39948  Saint Patrick       NaN          NaN  St. Patrick province        1.00
39949   Saint Andrew       NaN          NaN   St. Andrew province        1.00
39950   Saint George       NaN          NaN   St. George province        1.00
39951    Saint David       NaN          NaN    St. David province        1.00
39952     Saint Paul       NaN          NaN     St. Paul province        1.00
...
39951  MULTIPOLYGON (((-61.25374984699994 15.43152713...
39952  MULTIPOLYGON (((-61.39041519199998 15.31719684...

[39953 rows x 19 columns]>
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Python

	country	iso3	geo_id	geolocation	level	historical	disastertype	Shape_Length	Shape_Area	geometry
0	Albania	ALB	346.00	Ana E Malit	3	0.00	flood	0.30	0.00	MULTIPOLYGON (((19.384269714000027 41.99633407...
1	Albania	ALB	351.00	Bushat	3	0.00	flood	0.50	0.01	MULTIPOLYGON (((19.559324265000043 41.93292236...
2	Angola	AGO	760.00	Onjiva	3	0.00	flood	1.73	0.18	MULTIPOLYGON (((15.361865997000052 -17.2997074...
3	Angola	AGO	710.00	Evale	3	0.00	flood	1.72	0.17	MULTIPOLYGON (((16.11335372900004 -16.40042877...
4	Angola	AGO	749.00	Mupa	3	0.00	flood	2.03	0.18	MULTIPOLYGON (((16.11335372900004 -16.40042877...
...	...	...	...	...	...	...	...	...	...	...
39948	Dominica	NaN	45253.00	Saint Patrick	1	1.00	storm	0.50	0.01	MULTIPOLYGON (((-61.31208419799998 15.23930454...
39949	Dominica	NaN	45254.00	Saint Andrew	1	1.00	storm	0.80	0.02	MULTIPOLYGON (((-61.40016937299998 15.50179290...
39950	Dominica	NaN	45255.00	Saint George	1	1.00	storm	0.33	0.00	MULTIPOLYGON (((-61.301326751999966 15.3246822...
39951	Dominica	NaN	45256.00	Saint David	1	1.00	storm	0.69	0.01	MULTIPOLYGON (((-61.25374984699994 15.43152713...
39952	Dominica	NaN	45257.00	Saint Paul	1	1.00	storm	0.37	0.01	MULTIPOLYGON (((-61.39041519199998 15.31719684...

39953 rows × 10 columns

```
        country iso3   geo_id    geolocation  level  historical disastertype  \
0       Albania  ALB   346.00    Ana E Malit      3        0.00        flood
1       Albania  ALB   351.00         Bushat      3        0.00        flood
2        Angola  AGO   760.00         Onjiva      3        0.00        flood
3        Angola  AGO   710.00          Evale      3        0.00        flood
4        Angola  AGO   749.00           Mupa      3        0.00        flood
...         ...  ...      ...            ...    ...         ...          ...
39948  Dominica  NaN 45253.00  Saint Patrick      1        1.00        storm
39949  Dominica  NaN 45254.00   Saint Andrew      1        1.00        storm
39950  Dominica  NaN 45255.00   Saint George      1        1.00        storm
39951  Dominica  NaN 45256.00    Saint David      1        1.00        storm
39952  Dominica  NaN 45257.00     Saint Paul      1        1.00        storm

       Shape_Length  Shape_Area  \
0              0.30        0.00
1              0.50        0.01
2              1.73        0.18
3              1.72        0.17
4              2.03        0.18
...             ...         ...
39948          0.50        0.01
39949          0.80        0.02
39950          0.33        0.00
39951          0.69        0.01
39952          0.37        0.01
...
39951     15.41     -61.28
39952     15.37     -61.36

[39953 rows x 15 columns]
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

```
temp = df_cent[df_cent['disastertype'] == 'flood']
sal = pd.DataFrame(temp['year'])
pd.set_option('display.float_format', lambda x: '%.2f' % x)
sal.describe()
```

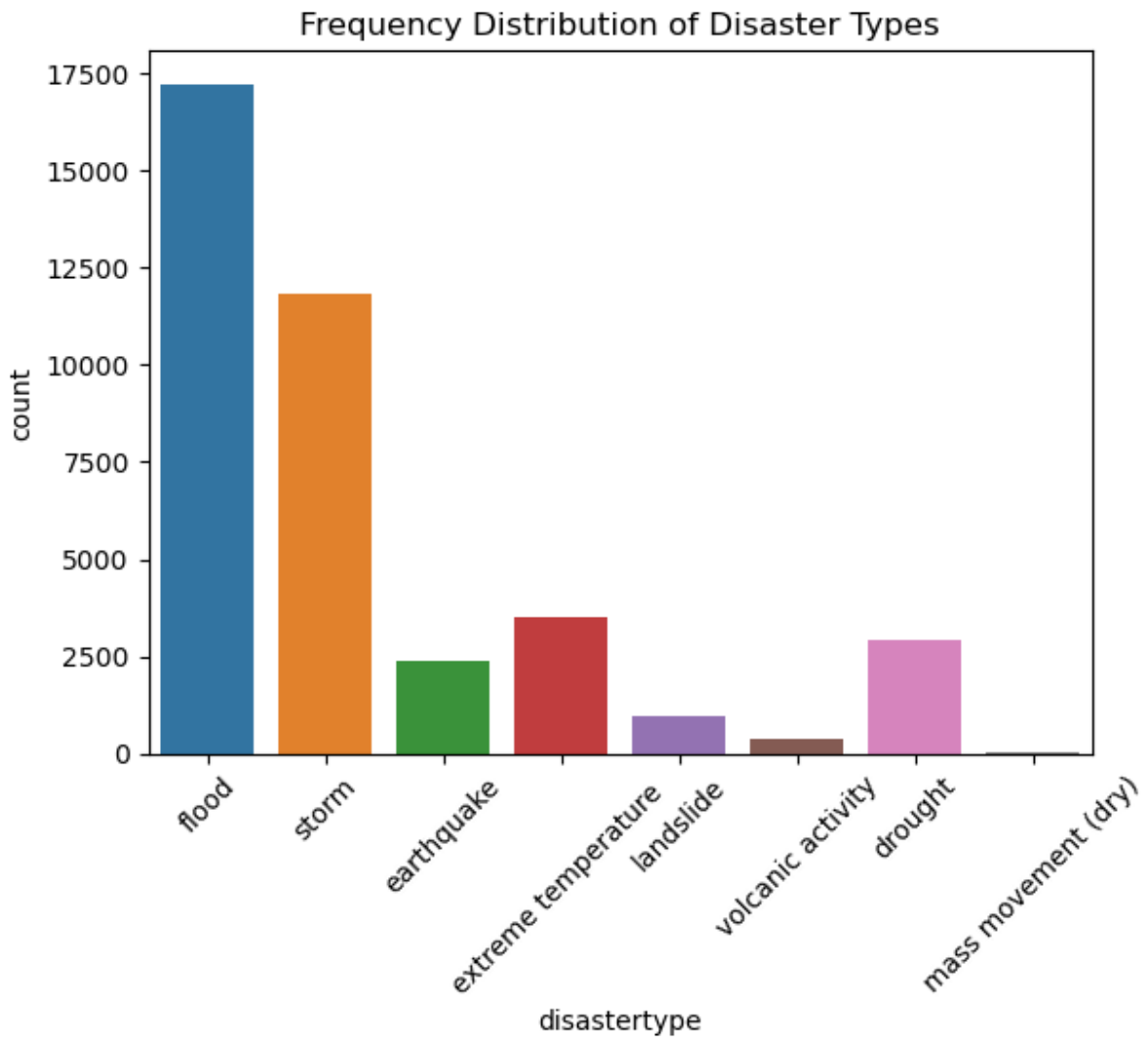
	year
count	17347.00
mean	2005.06
std	9.98
min	1960.00
25%	2000.00
50%	2007.00
75%	2012.00
max	2018.00

```
missing_values = result.isnull().sum()

print(missing_values)
```

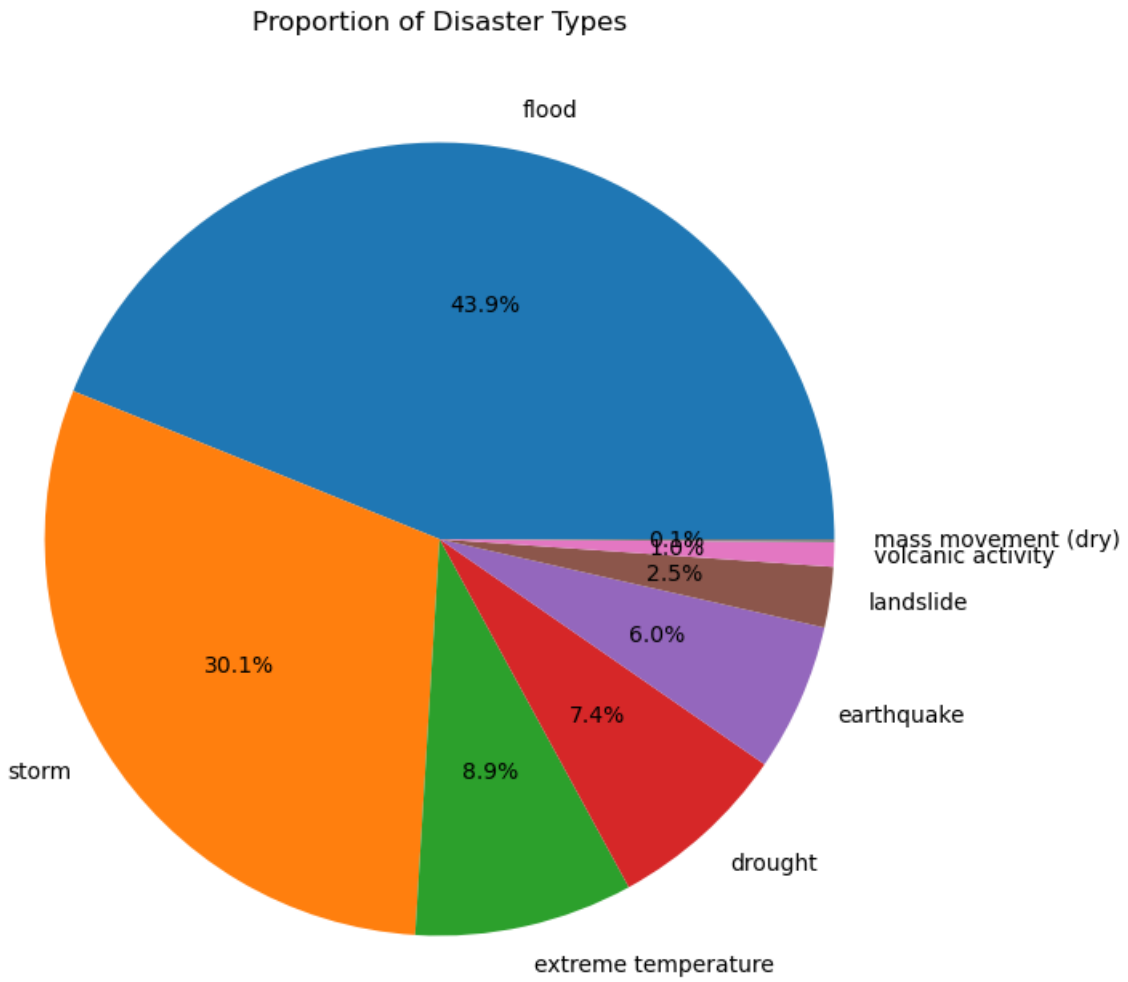
```
country         0
iso3            0
geo_id          0
geolocation     0
level           0
historical      0
disastertype    0
Shape_Length    0
Shape_Area      0
geometry        0
gwno            0
year            0
geo_id          0
latitude        0
longitude       0
dtype: int64
```

```
result = result.dropna()
```

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.countplot(data=result, x='disastertype')
plt.title('Frequency Distribution of Disaster Types')
plt.xticks(rotation=45)
```



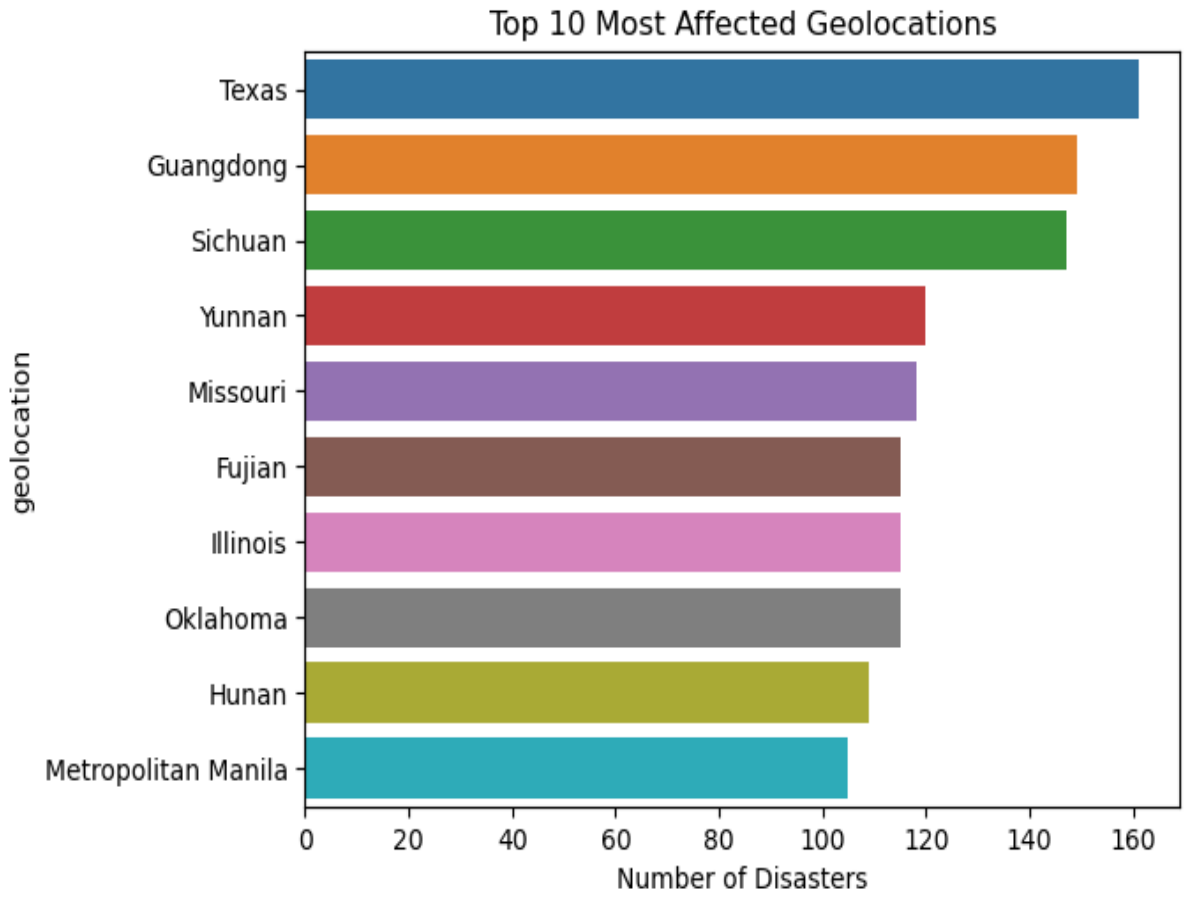

```
result['disastertype'].value_counts().plot.pie(autopct='%1.1f%%',
figsize=(8, 8))
plt.title('Proportion of Disaster Types')
plt.ylabel('')
plt.show()
```



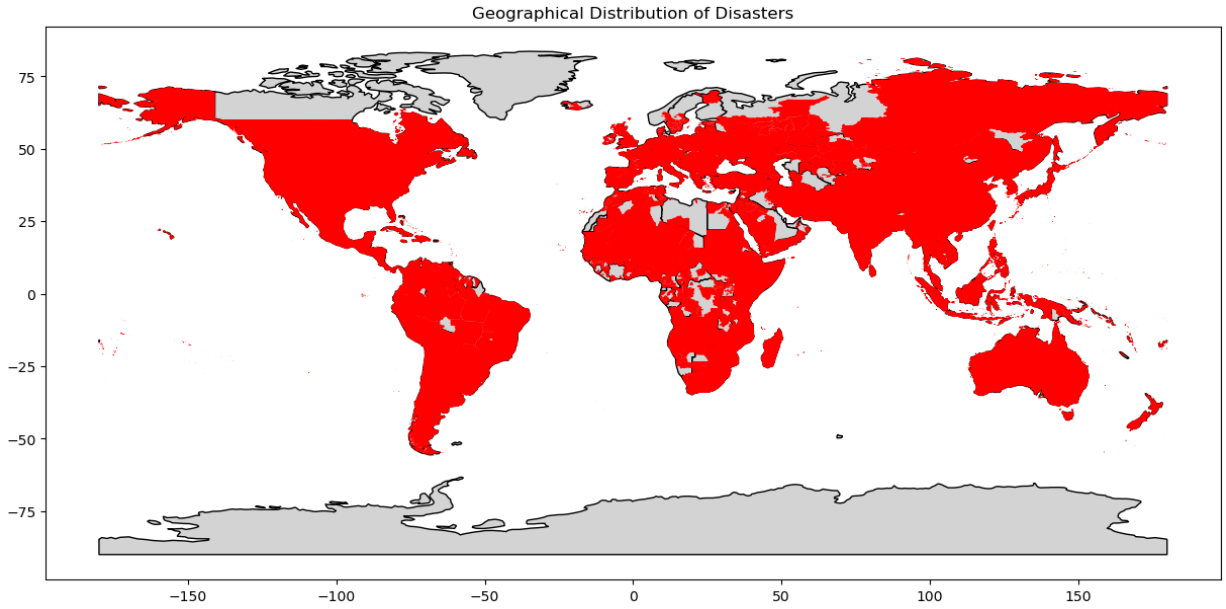

```
unique_locations = result['geolocation'].nunique()
print(f'Number of unique geolocations: {unique_locations}')
```

```
Number of unique geolocations: 9660
```

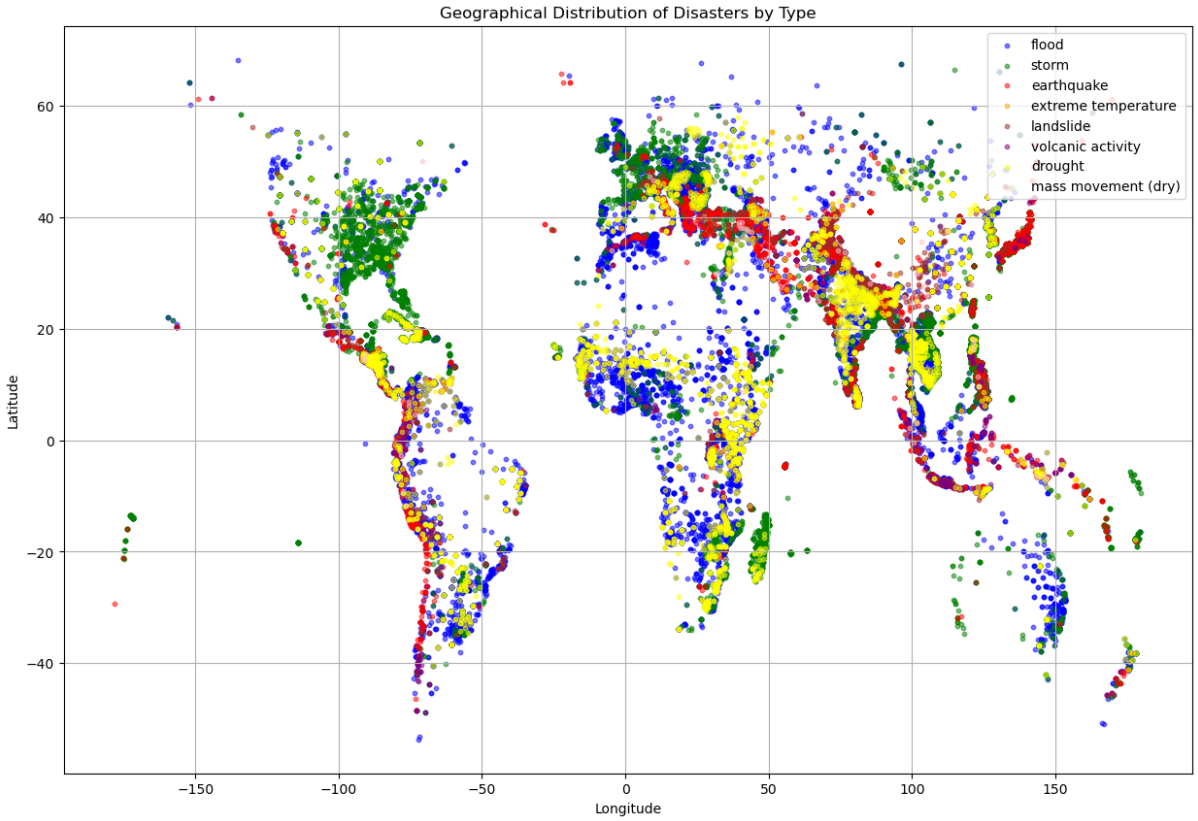
```
top_locations = result['geolocation'].value_counts().head(10)
sns.barplot(x=top_locations.values, y=top_locations.index)
plt.title('Top 10 Most Affected Geolocations')
plt.xlabel('Number of Disasters')
plt.show()
```



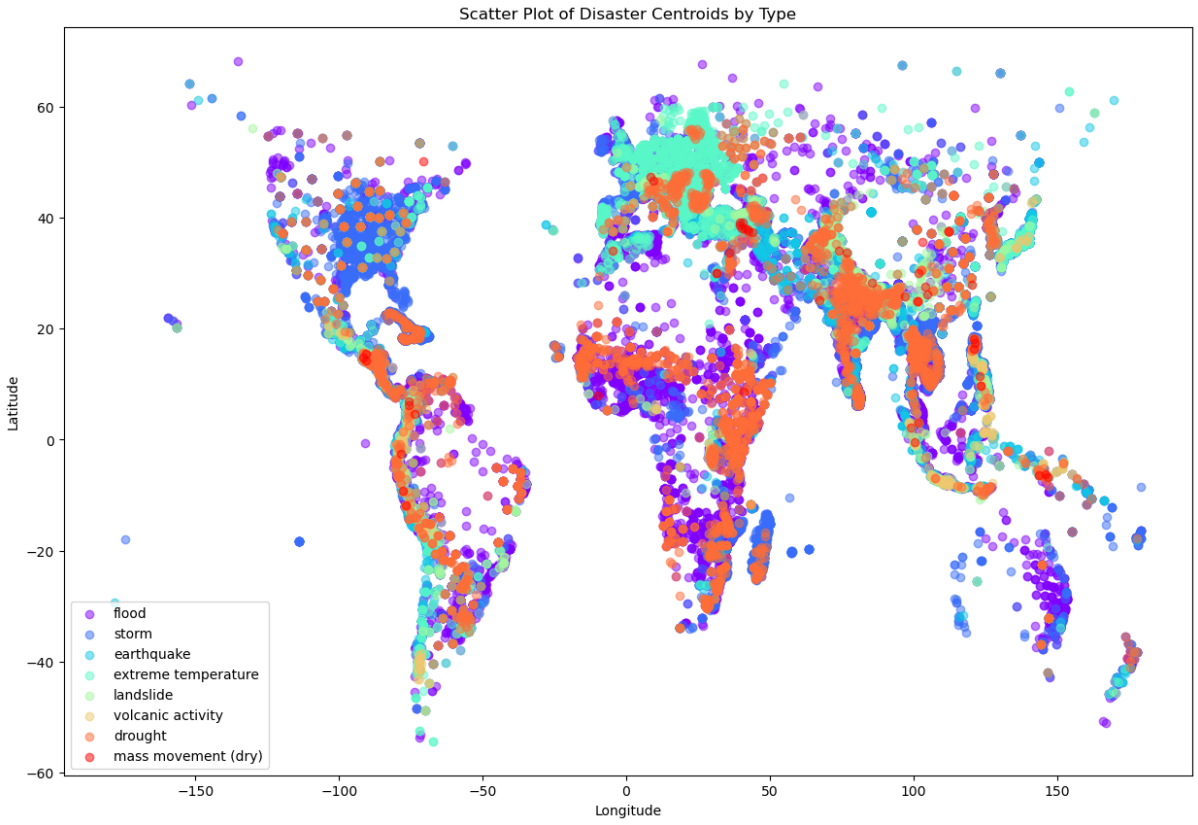

```
result['geometry'] = result['geometry'].apply(wkt.loads)

gdf = gpd.GeoDataFrame(result, geometry='geometry')

world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

ax = world.plot(figsize=(15, 10), color='lightgrey', edgecolor='black')

gdf.plot(ax=ax, marker='o', color='red', markersize=5)

plt.title('Geographical Distribution of Disasters')
plt.show()
```




```
import pandas as pd
import matplotlib.pyplot as plt
from shapely import wkt

result['geometry'] = result['geometry'].apply(wkt.loads)
result['longitude'] = result['geometry'].apply(lambda geom: geom.centroid.x)
result['latitude'] = result['geometry'].apply(lambda geom: geom.centroid.y)

disaster_colors = {
    'flood': 'blue',
    'storm': 'green',
    'earthquake': 'red',
    'extreme temperature': 'orange',
    'landslide': 'brown',
    'volcanic activity': 'purple',
    'drought': 'yellow',
    'mass movement (dry)': 'pink'
}

fig, ax = plt.subplots(figsize=(15, 10))

for disaster_type, color in disaster_colors.items():
    subset = new_df[result['disastertype'] == disaster_type]
    ax.scatter(subset['longitude'], subset['latitude'], s=10, c=color,
label=disaster_type, alpha=0.5)

plt.title('Geographical Distribution of Disasters by Type')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.legend(loc='upper right')
plt.grid(True)
plt.show()
```




```
# plt.scatter(result['longitude'], result['latitude'], alpha=0.5)

# plt.title('Scatter Plot of Disaster Centroids')
# plt.xlabel('Longitude')
# plt.ylabel('Latitude')
# plt.show()

fig, ax = plt.subplots(figsize=(15, 10))

disaster_types = result['disastertype'].unique()
colors = plt.cm.rainbow(np.linspace(0, 1, len(disaster_types)))

for disaster_type, color in zip(disaster_types, colors):
    subset = result[result['disastertype'] == disaster_type]
    ax.scatter(subset['longitude'], subset['latitude'], alpha=0.5,
label=disaster_type, color=color)

plt.title('Scatter Plot of Disaster Centroids by Type')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.legend()
plt.show()
```



Floods and Storms: Floods (purple) and storms (blue) are prevalent in coastal and low-lying regions. High concentration in Southeast Asia, particularly in countries like India, Bangladesh, and the Philippines. Significant occurrences in the eastern United States and Central America.

Earthquakes: Earthquakes (red) are concentrated along tectonic plate boundaries. High density in the Pacific Ring of Fire, affecting countries like Japan, Indonesia, and the west coast of the Americas. Significant occurrences in southern Europe and parts of the Middle East.

Extreme Temperatures: Extreme temperature events (green) are scattered but more frequent in inland and northern regions. Noticeable in parts of North America, Europe, and central Asia.

Landslides: Landslides (brown) are more common in mountainous regions. High frequency in the Himalayan region, Andes, and certain parts of Southeast Asia.

Volcanic Activity: Volcanic activities (cyan) are concentrated in the Pacific Ring of Fire and volcanic island chains. Prominent in Indonesia, Japan, and parts of Central America and the Caribbean.

Droughts: Droughts (orange) are more common in arid and semi-arid regions. Significant occurrences in Sub-Saharan Africa, Australia, and parts of the United States and Brazil.

Mass Movements (Dry): Dry mass movements (pink) are less frequent but present in various regions. Scattered occurrences with some concentration in parts of Asia and South America.

Spatial Patterns: Cluster Analysis: Clustering of disaster types suggests regions with multiple overlapping disaster risks.

For example, Southeast Asia experiences frequent floods, storms, and earthquakes, indicating a need for integrated disaster management.

Distribution Gaps: Certain regions, such as central Africa and central Australia, show fewer disaster events, which could be due to lower data availability or genuinely lower occurrence rates.

Implications for Prediction: High-Risk Areas: Regions with high concentrations of disaster events should be prioritized for disaster prediction and preparedness. Coastal regions and areas along tectonic plate boundaries require robust early warning systems and infrastructure resilience.

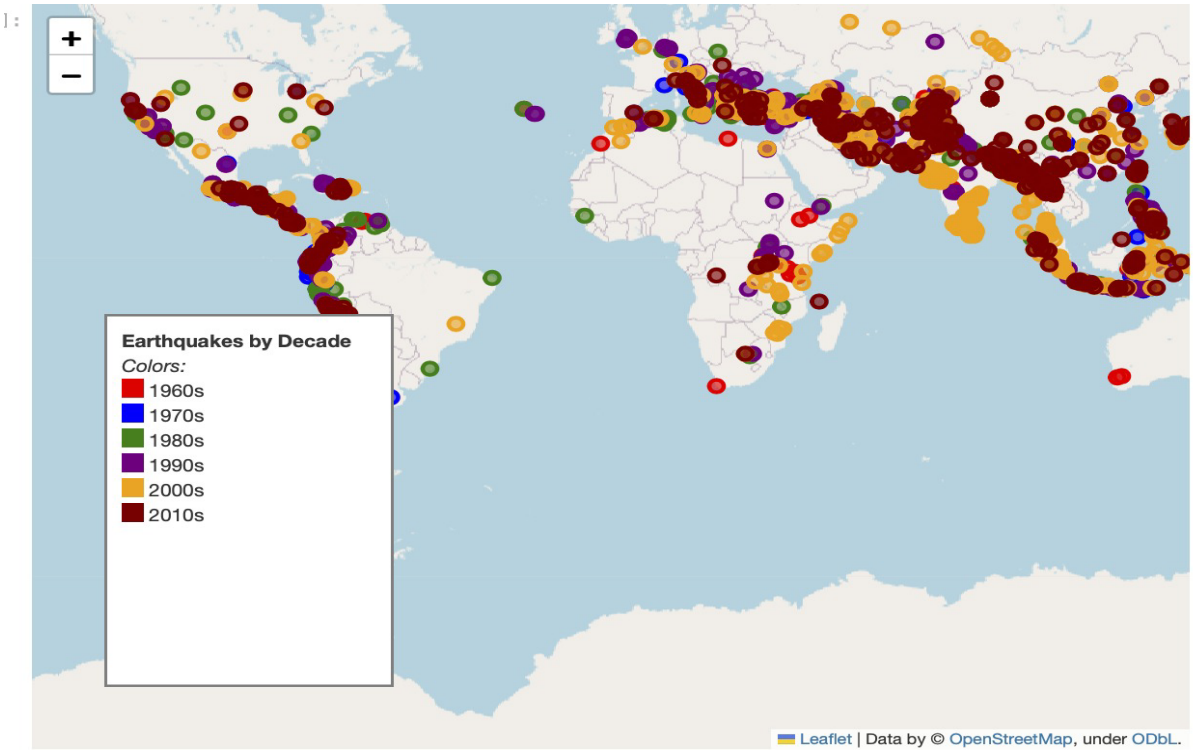
Seasonal and Environmental Factors: Understanding the environmental and seasonal factors contributing to these patterns can enhance prediction models. For example, the monsoon season in Asia correlates with higher flood and storm events.

```
import folium
from folium.plugins import HeatMap
from IPython.display import IFrame

map_center = [result['latitude'].mean(), result['longitude'].mean()]

map_folium = folium.Map(location=map_center, zoom_start=2)

heat_data = [[row['latitude'], row['longitude']] for index, row in
result.iterrows()]

HeatMap(heat_data).add_to(map_folium)

map_folium.save('heatmap.html')

IFrame('heatmap.html', width=800, height=600)
```

Temporal Analysis: Conduct a temporal analysis to identify trends and changes in disaster frequency over time. Determine if certain regions are becoming more prone to specific types of disasters.

Integration with Socio-Economic Data: Integrate geographical data with socio-economic indicators to understand the impact on vulnerable populations. This can help in designing targeted interventions and resilience-building measures. By analyzing these insights, you can enhance your understanding of disaster patterns and improve prediction and preparedness strategies for natural disasters.

```
earthquakes = result[result['disastertype'] == 'earthquake']
earthquakes['decade'] = (earthquakes['year'] // 10) * 10
map_center = [earthquakes['latitude'].mean(),
earthquakes['longitude'].mean()]

map_folium = folium.Map(location=map_center, zoom_start=2)

colors = ['red', 'blue', 'green', 'purple', 'orange', 'darkred',
'lightred', 'darkblue', 'darkgreen', 'cadetblue']

unique_decades = sorted(earthquakes['decade'].unique())

for i, decade in enumerate(unique_decades):
    decade_data = earthquakes[earthquakes['decade'] == decade]
    for _, row in decade_data.iterrows():
        folium.CircleMarker(
            location=[row['latitude'], row['longitude']],
            radius=5,
            color=colors[i % len(colors)],
            fill=True,
            fill_color=colors[i % len(colors)],
            fill_opacity=0.6,
            popup=f"Year: {row['year']}\nType: {row['disastertype']}"
        ).add_to(map_folium)

legend_html = '''
<div style="position: fixed;
     bottom: 50px; left: 50px; width: 200px; height: 300px;
     border:2px solid grey; z-index:9999; font-size:14px;
     background-color:white; padding: 10px;">
     <b>Earthquakes by Decade</b><br>
     <i>Colors:</i><br>
'''
for i, decade in enumerate(unique_decades):
    legend_html += f'<i style="background:{colors[i %
len(colors)]};width:15px;height:15px;display:inline-block;"></i>
{decade}s<br>'
legend_html += '</div>'

map_folium.get_root().html.add_child(folium.Element(legend_html))

map_folium.save('earthquake_map.html')

IFrame('earthquake_map.html', width=800, height=600)
```



For a comprehensive temporal analysis of earthquake occurrences, we visualized their geographic distribution across different decades. By categorizing earthquakes by decade and color-coding each period, we created an interactive map that highlights the temporal progression and concentration of seismic activity. This map, cantered on the average earthquake coordinates, allows users to observe changes in earthquake patterns over time and facilitates a better understanding of the temporal dynamics of seismic events. The included legend aids in interpreting the map, providing clear insights into how earthquake activity has evolved across different periods.

```
Observations...
We realise that the dataset we are using is a little skewed
for past few decades. But still, it is easily identifiable
from the above map that the major epicenters clusters are
present in the Circum-Pacific belt, the Alpide belt and the
mid-Atlantic Ridge.
The epicenter belts are located near mountain ranges and on
tectonic plate boundaries.
Another hypothesis I would like to put forth is expanding
clusters sizes during the last couple of decades.
```

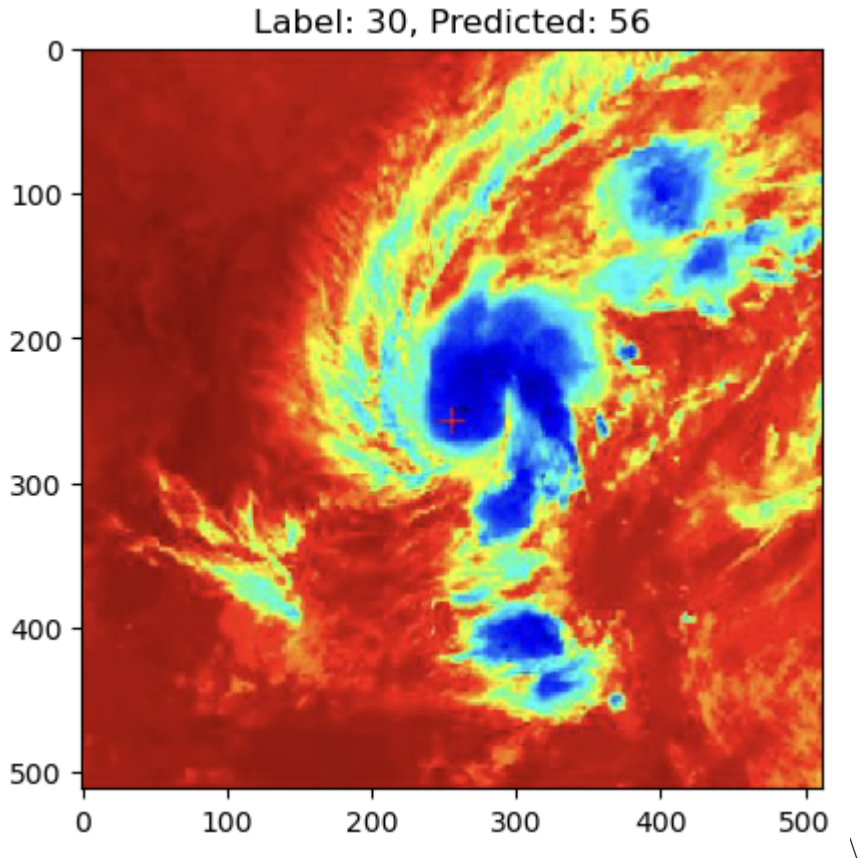
```
def spatial_pyramid_pool(previous_conv, num_sample, previous_conv_size,
out_pool_size):
    spp = []
    for i in range(len(out_pool_size)):
        h_wid = int(math.ceil(previous_conv_size[0] / out_pool_size[i]))
        w_wid = int(math.ceil(previous_conv_size[1] / out_pool_size[i]))
        h_pad = int(math.ceil((h_wid * out_pool_size[i] -
previous_conv_size[0] + 1) / 2))
        w_pad = int(math.ceil((w_wid * out_pool_size[i] -
previous_conv_size[1] + 1) / 2))
        maxpool = nn.MaxPool2d((h_wid, w_wid), stride=(h_wid, w_wid),
padding=(h_pad, w_pad))
        x = maxpool(previous_conv)
        x = x.reshape(num_sample, -1)  # Use reshape instead of view
        spp.append(x)
    return torch.cat(spp, 1)

class SPP_NET(nn.Module):
    def __init__(self, input_nc, ndf=64):
        super(SPP_NET, self).__init__()
        self.output_num = [4, 2, 1]

        self.conv1 = nn.Conv2d(input_nc, ndf, 4, 2, 1)
        self.LReLU1 = nn.LeakyReLU(0.2)

        self.conv2 = nn.Conv2d(ndf, ndf * 2, 4, 1, 1)
        self.BN1 = nn.BatchNorm2d(ndf * 2)
        self.dropout1 = nn.Dropout(0.5)

        self.conv3 = nn.Conv2d(ndf * 2, ndf * 4, 4, 1, 1)
        self.BN2 = nn.BatchNorm2d(ndf * 4)
        self.dropout2 = nn.Dropout(0.5)
```

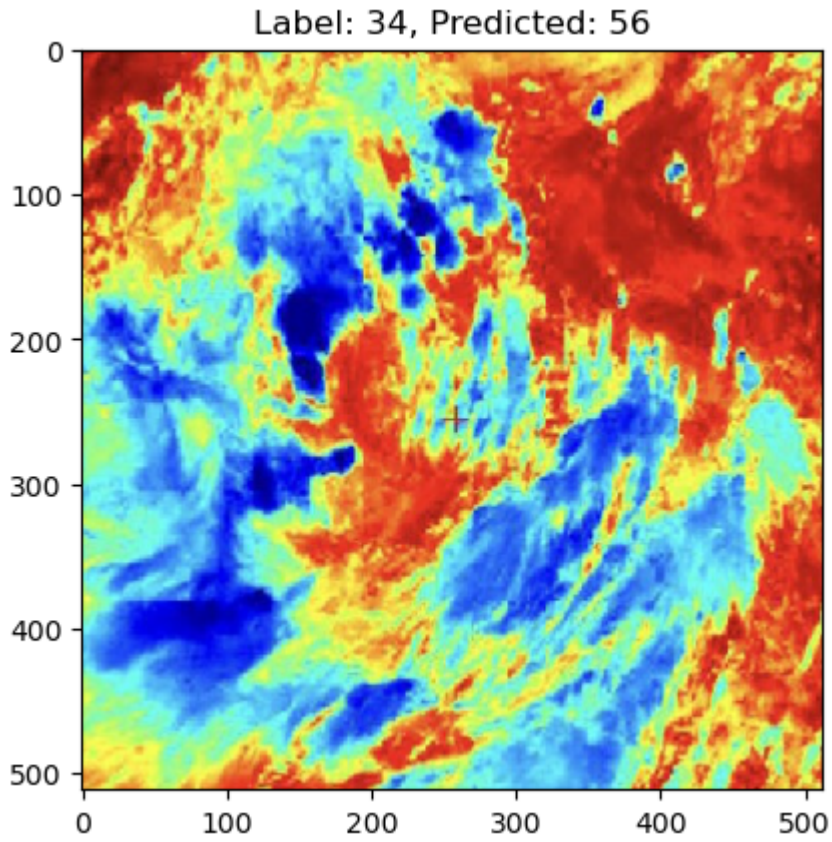
```python
from torchvision import transforms

transform = transforms.Compose([
    transforms.RandomHorizontalFlip(),
    transforms.RandomVerticalFlip(),
    transforms.RandomRotation(20),
    transforms.ToTensor(),
])

class CustomDataset(Dataset):
    def __init__(self, dataframe, datagen, directory, x_col, y_col,
target_size, class_mode, transform=None):
        self.datagen = datagen.flow_from_dataframe(
            dataframe,
            directory=directory,
            x_col=x_col,
            y_col=y_col,
            target_size=target_size,
            class_mode=class_mode,
            batch_size=1,  # Single sample for each call
            shuffle=True
        )
        self.transform = transform

    def __len__(self):
        return len(self.datagen)  # Number of batches available

    def __getitem__(self, idx):
        batch = next(self.datagen)
        images, labels = batch
        images = torch.tensor(images, dtype=torch.float32).permute(0, 3, 1,
2)  # Convert to PyTorch tensor and permute dimensions

        if self.transform:
            images = self.transform(images)

        labels = torch.tensor(labels, dtype=torch.long)
        return images[0], labels[0]  # Return single sample
```

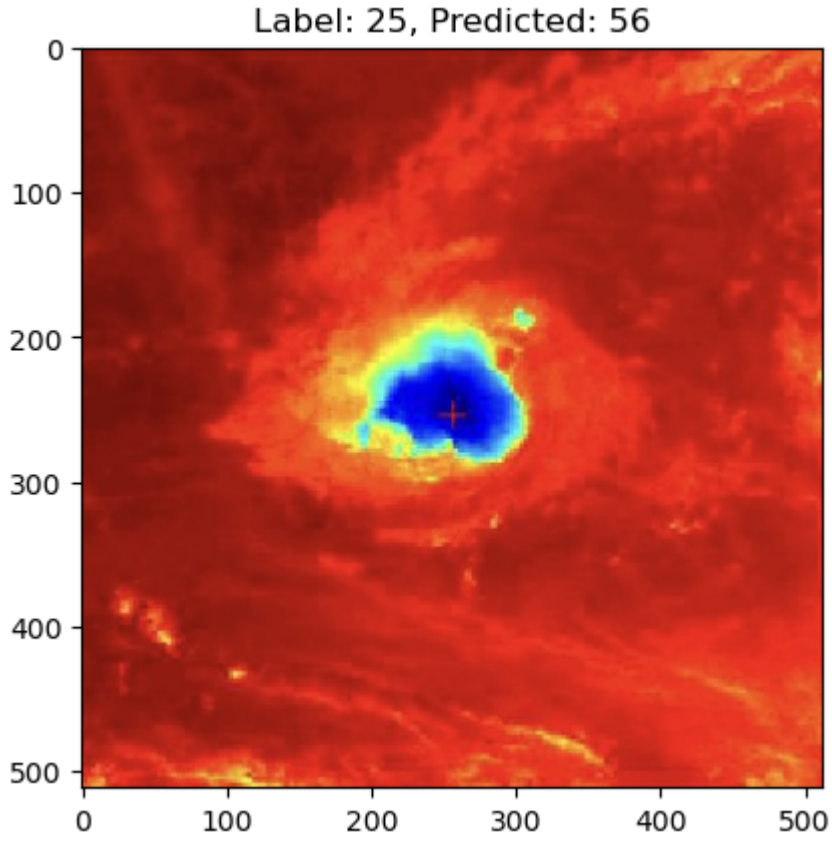
```
train_datagen = ImageDataGenerator(rescale=1.0/255.0)
test_datagen = ImageDataGenerator(rescale=1.0/255.0)

train = pd.read_csv("/Users/adityajain/Downloads/archive/insat_3d_ds - Sheet.csv")
train_df, test_df = train_test_split(train, test_size=0.2, random_state=2)

train_dataset = CustomDataset(
    train_df,
    train_datagen,
    "/Users/adityajain/Downloads/archive/insat3d_ir_cyclone_ds/CYCLONE_DATASET_INFRARED",
    "img_name",
    "label",
    (512, 512),
    'raw'
)

test_dataset = CustomDataset(
    test_df,
    test_datagen,
    "/Users/adityajain/Downloads/archive/insat3d_ir_cyclone_ds/CYCLONE_DATASET_INFRARED",
    "img_name",
    "label",
    (512, 512),
    'raw'
```

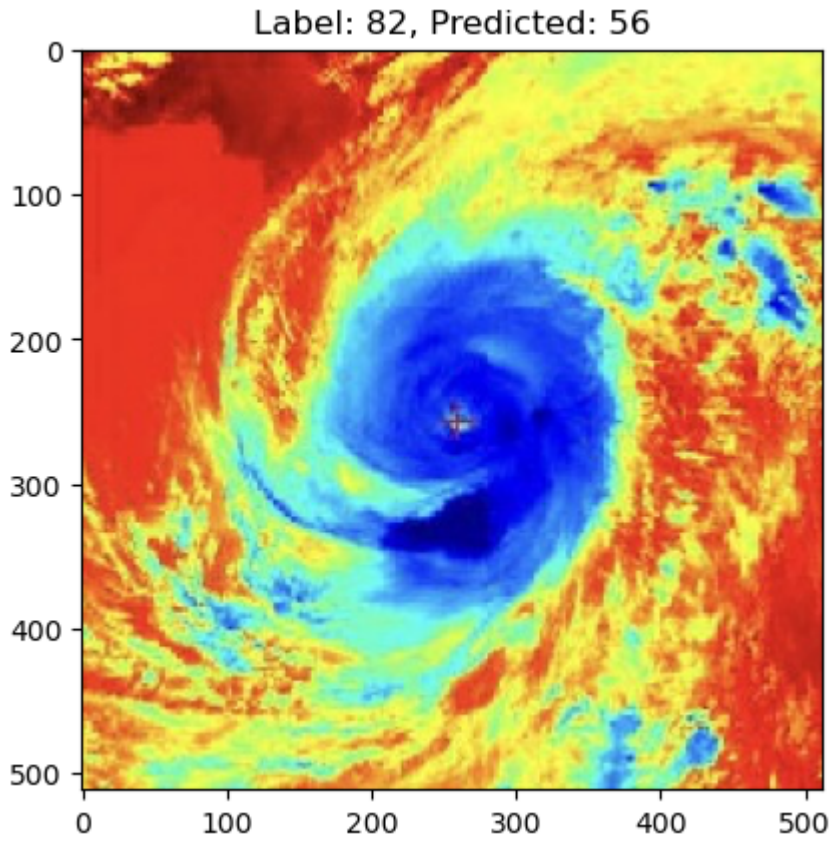
```
import torch

import torch.nn as nn
import torch.optim as optim

def train_model():
    learning_rate = 1e-4
    weight_decay = 1e-5  # Add weight decay for regularization
    num_epochs = 10
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

    model = SPP_NET(input_nc=3).to(device)
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=learning_rate, weight_decay=weight_decay)
```

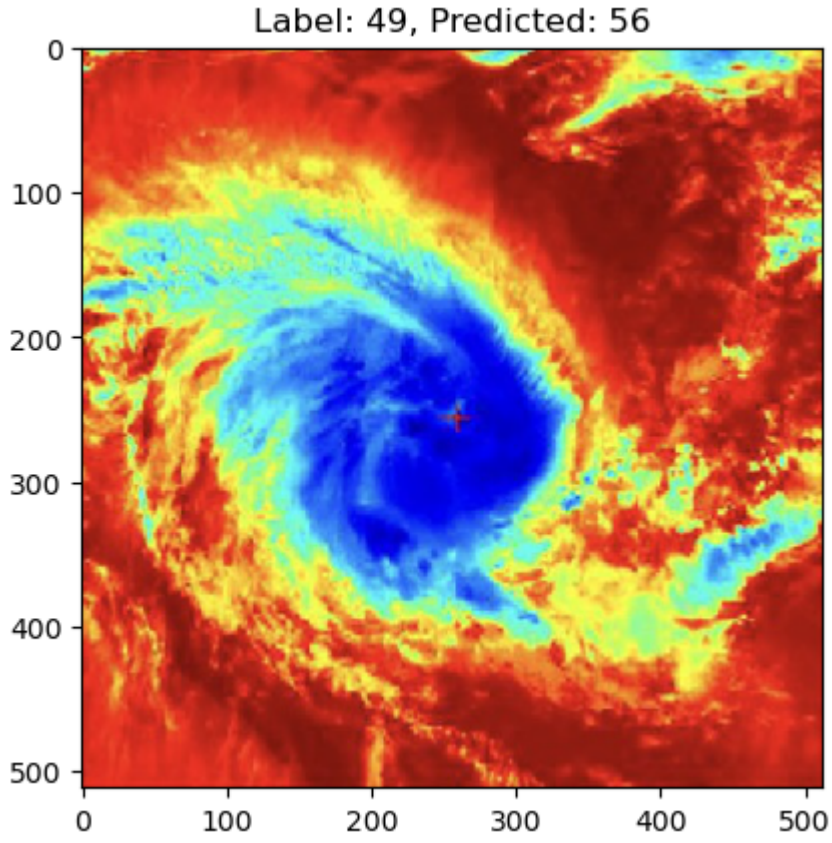
BATCH 1:

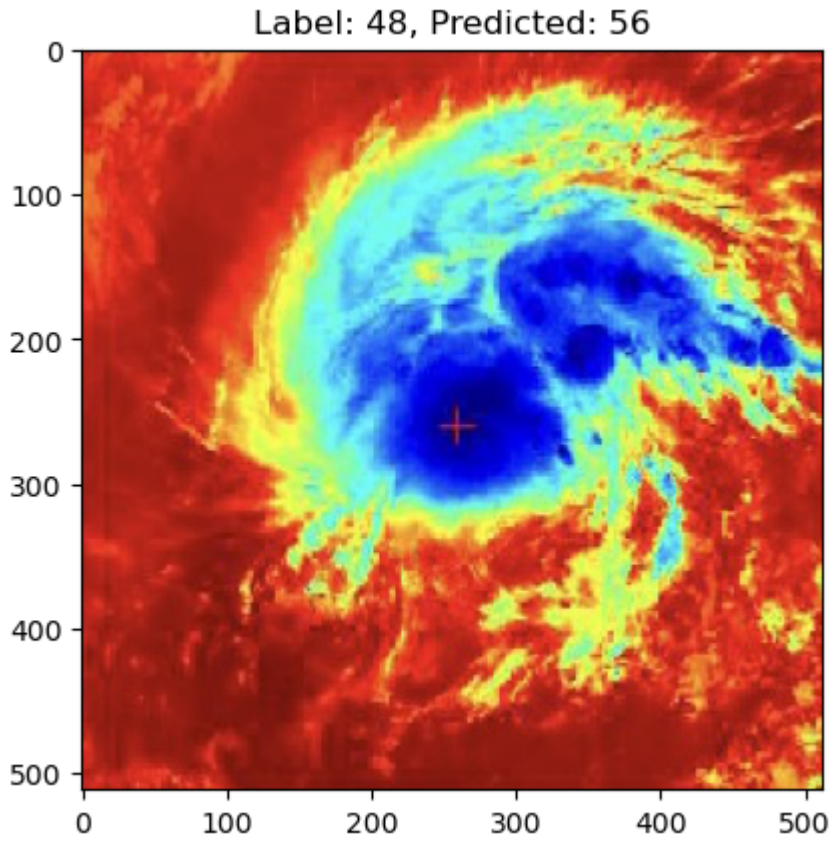

Label: 30, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

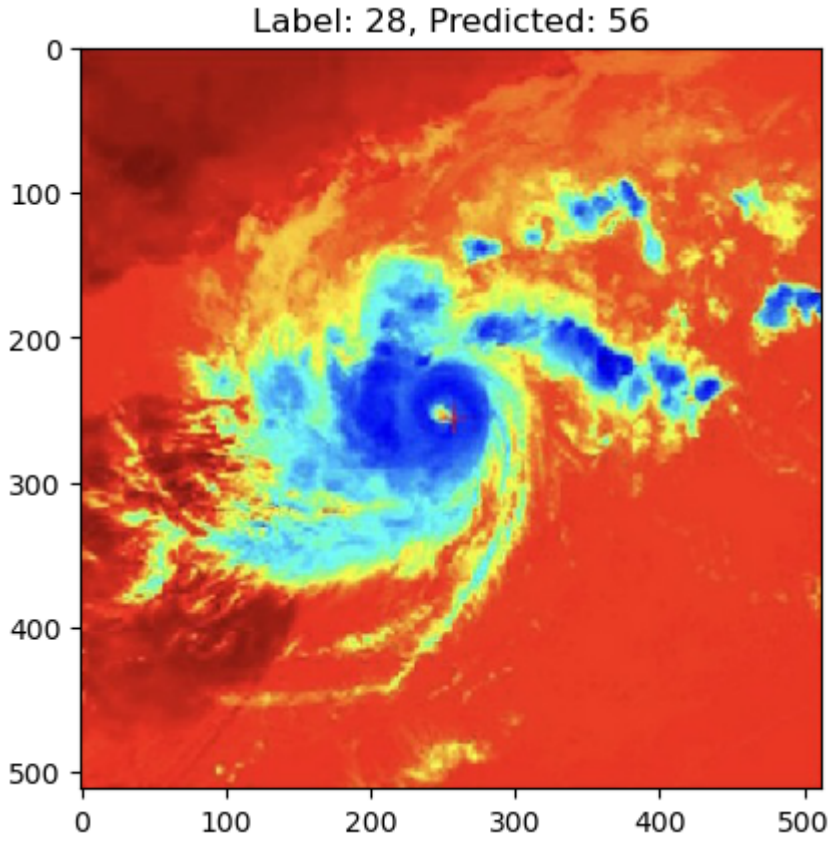

Label: 34, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

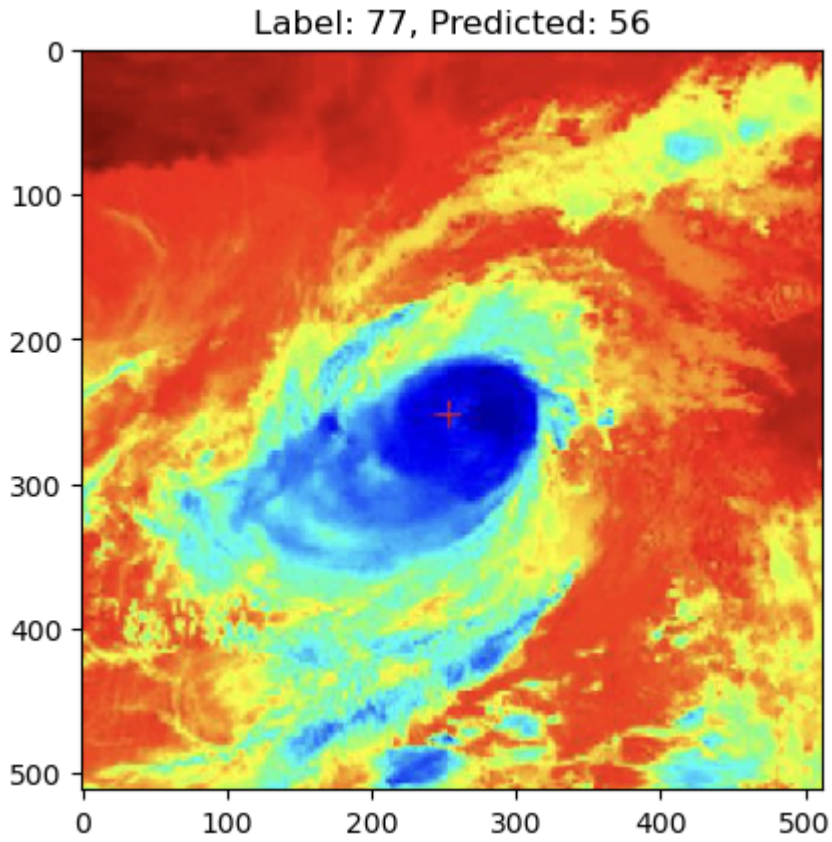

Label: 25, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

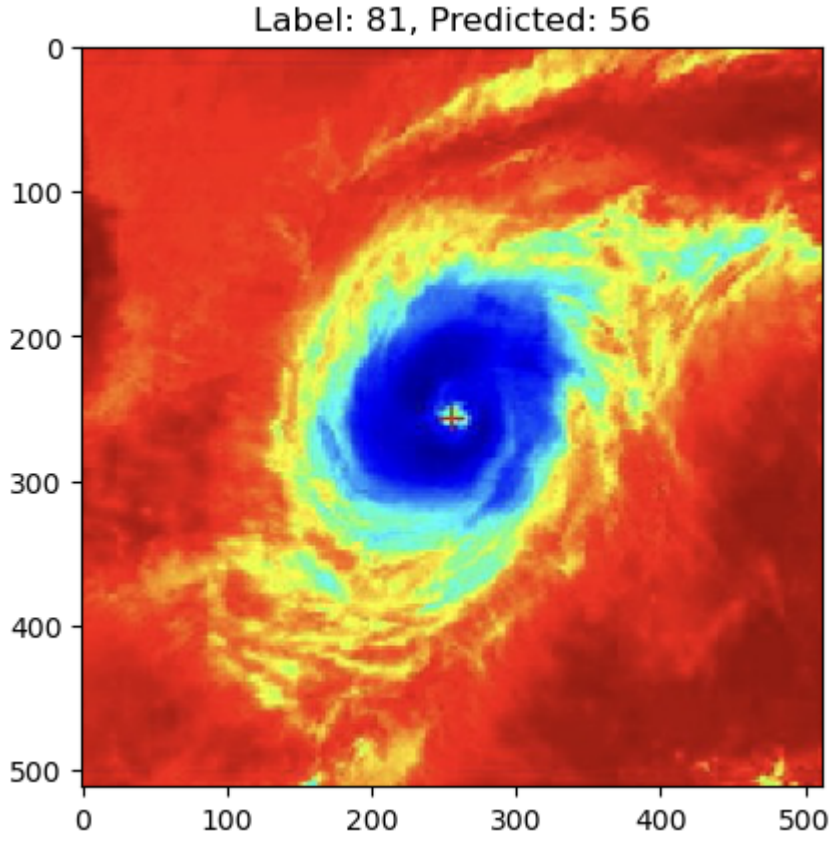

Label: 82, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

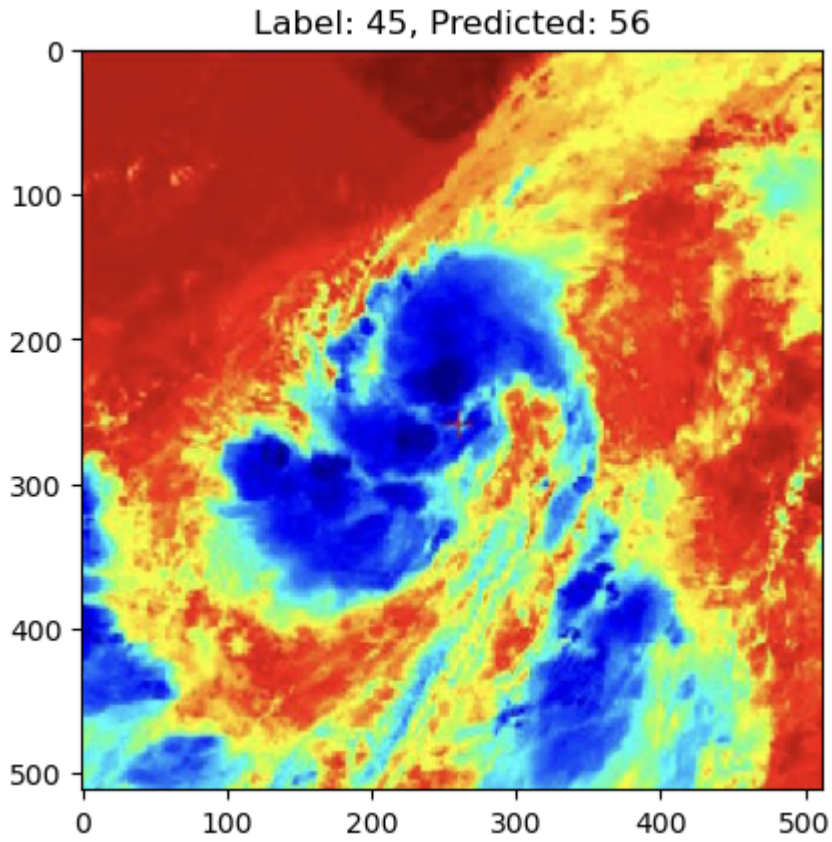

Label: 49, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

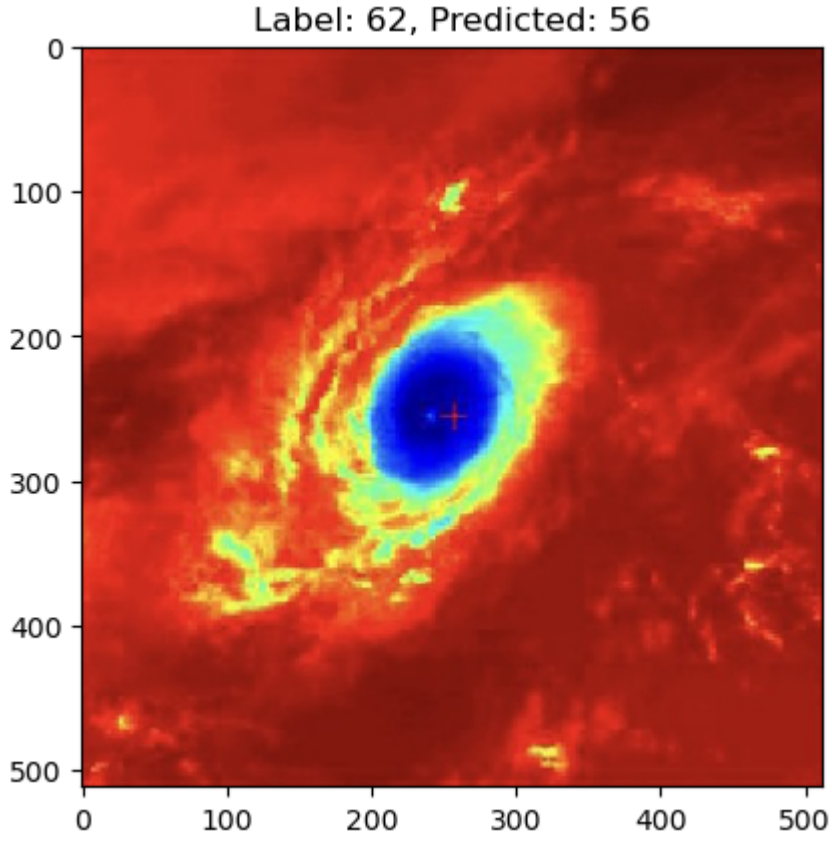

Label: 48, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

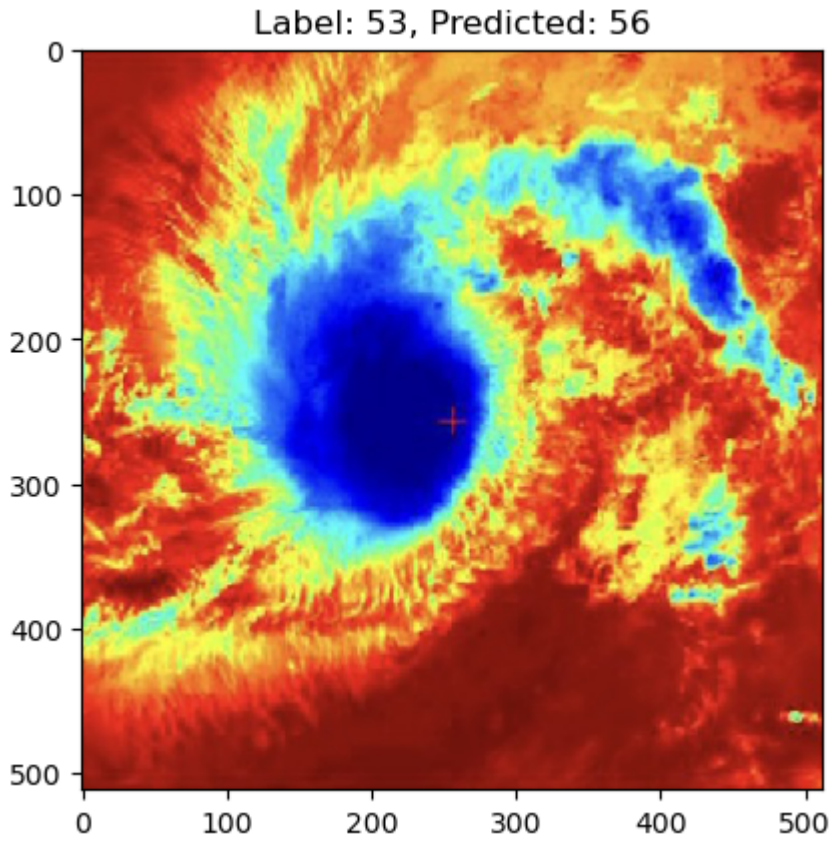

Label: 28, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

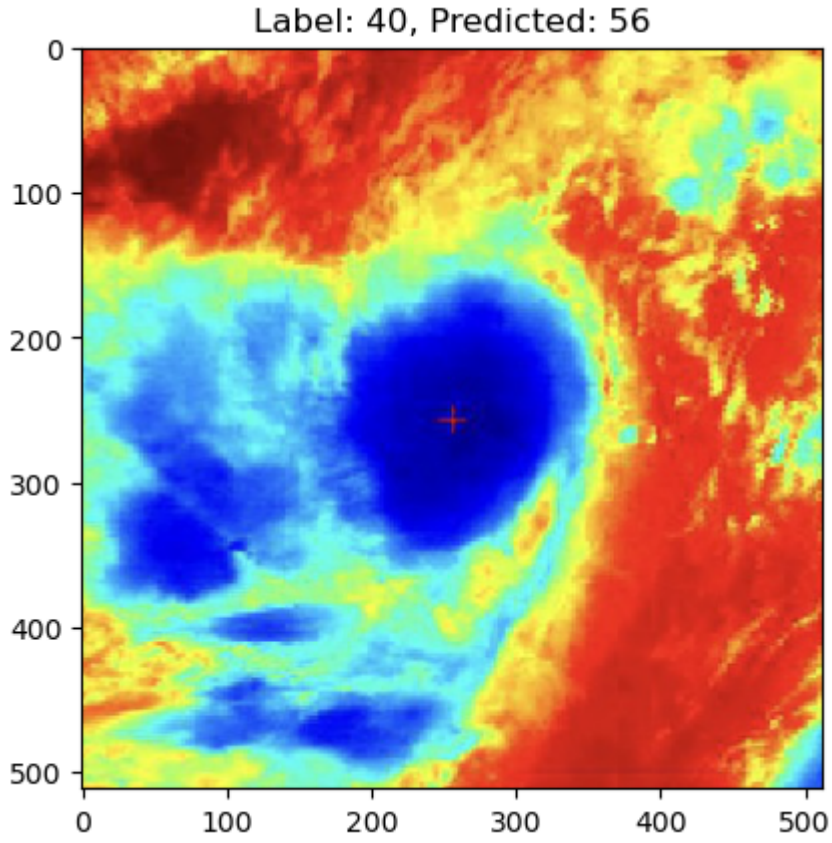

Label: 77, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

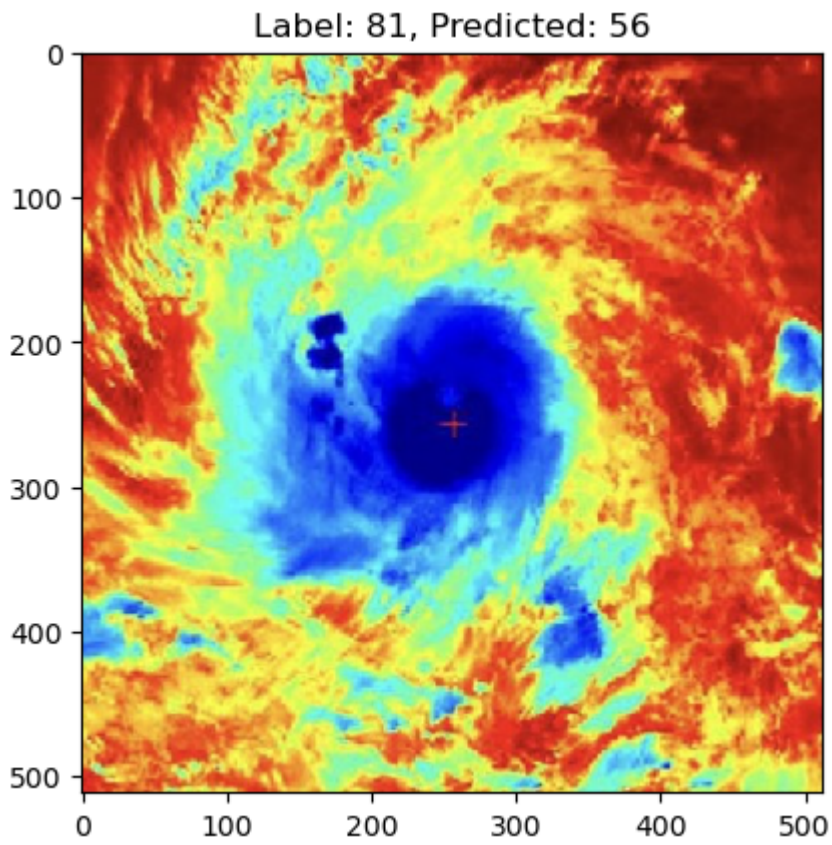

Label: 81, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

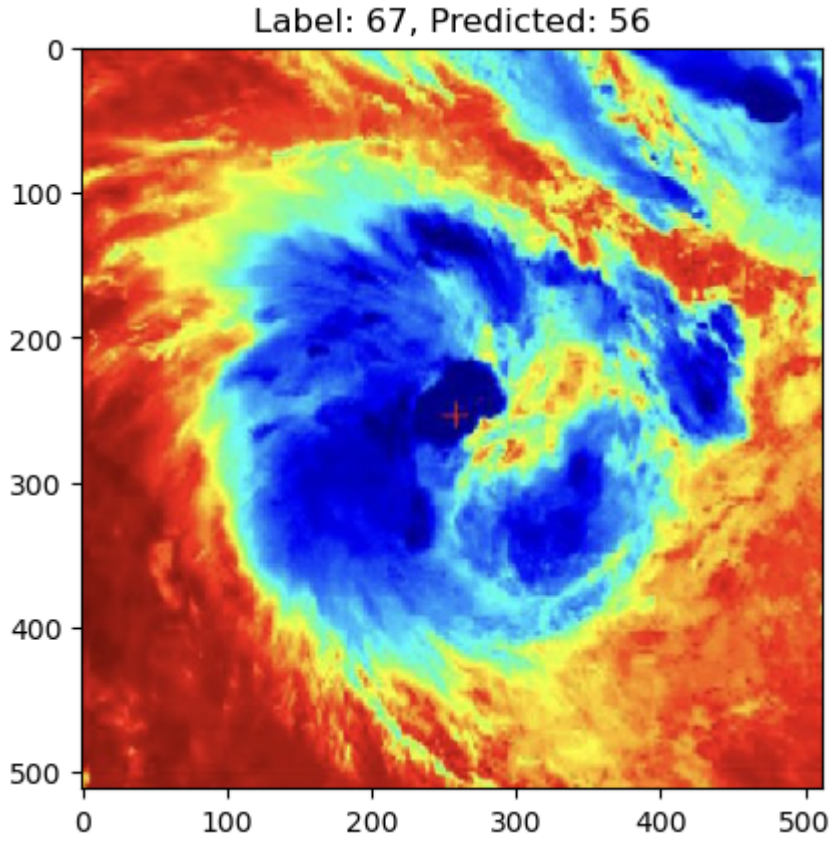

Label: 45, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

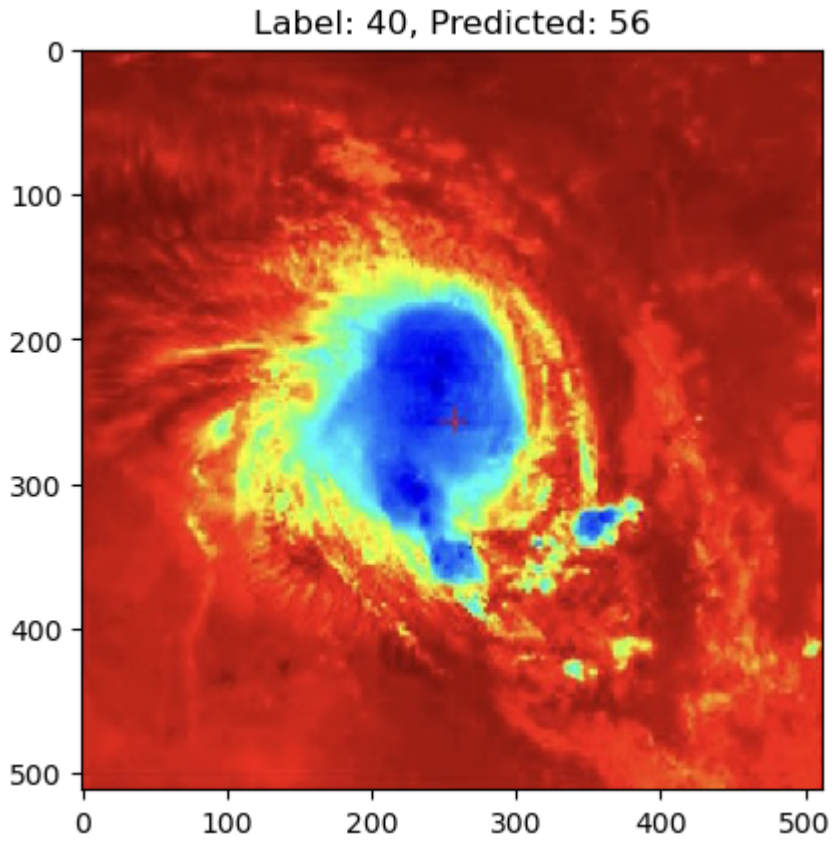

Label: 62, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

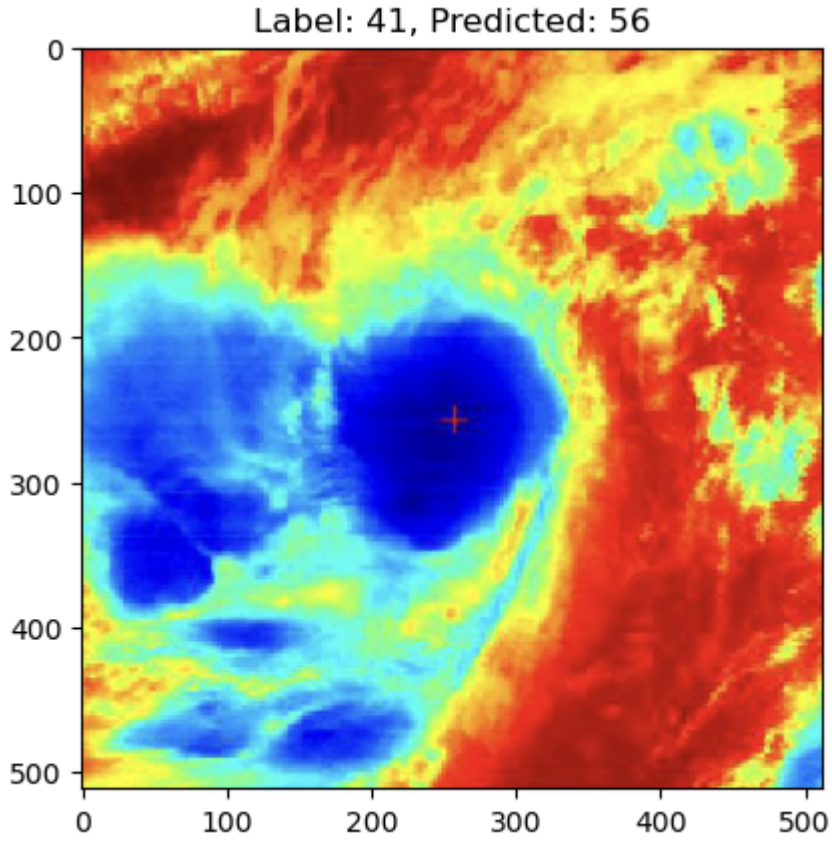

Label: 53, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

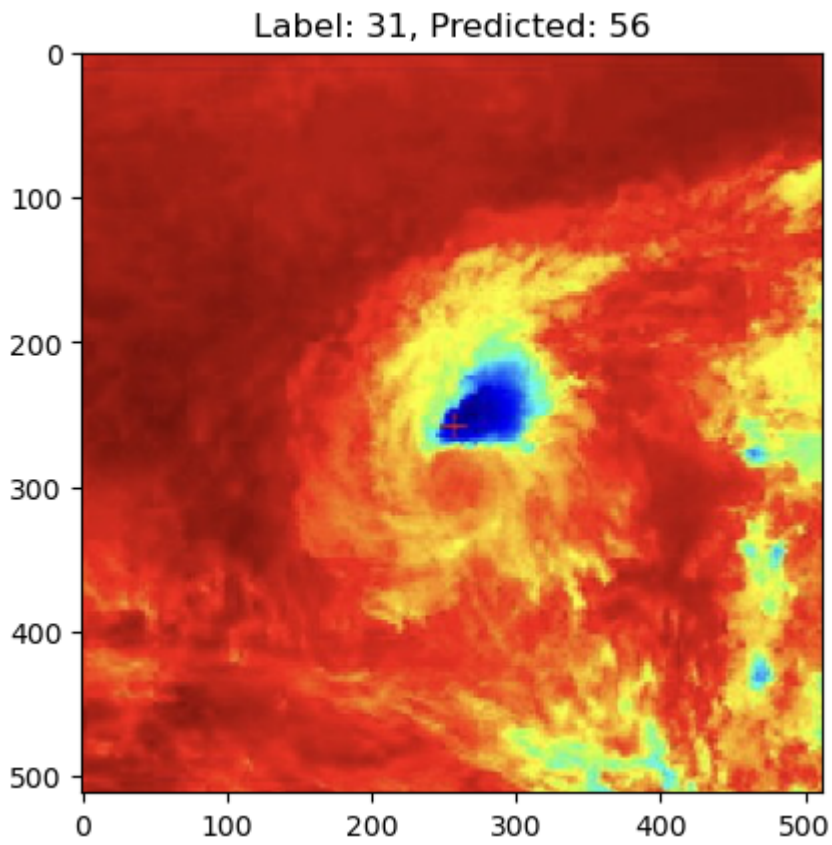

Label: 40, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

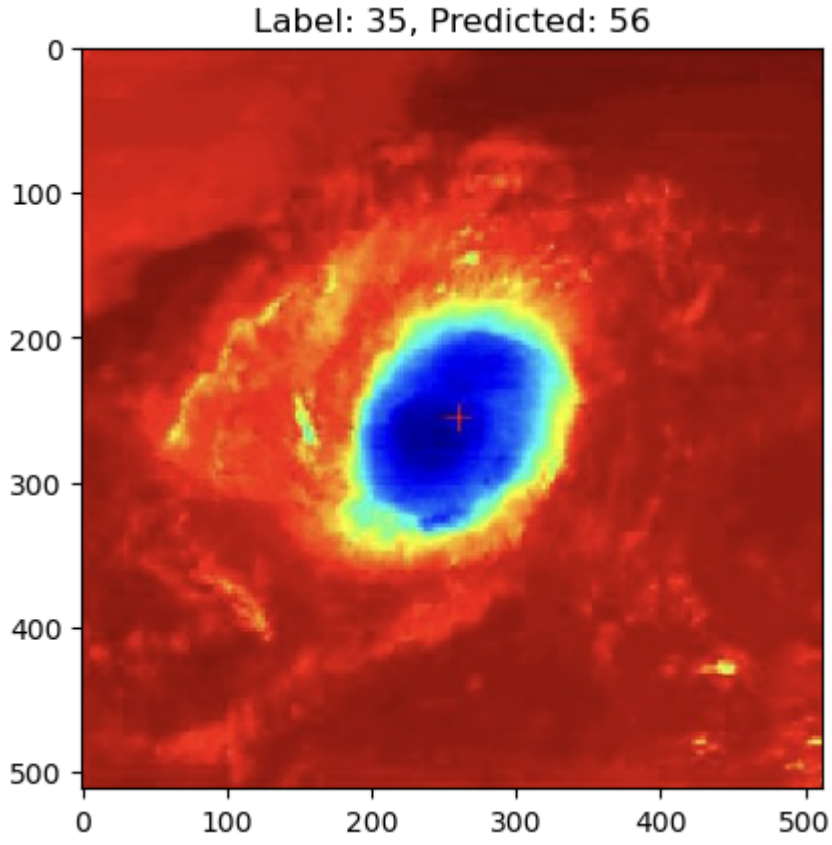

Label: 81, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

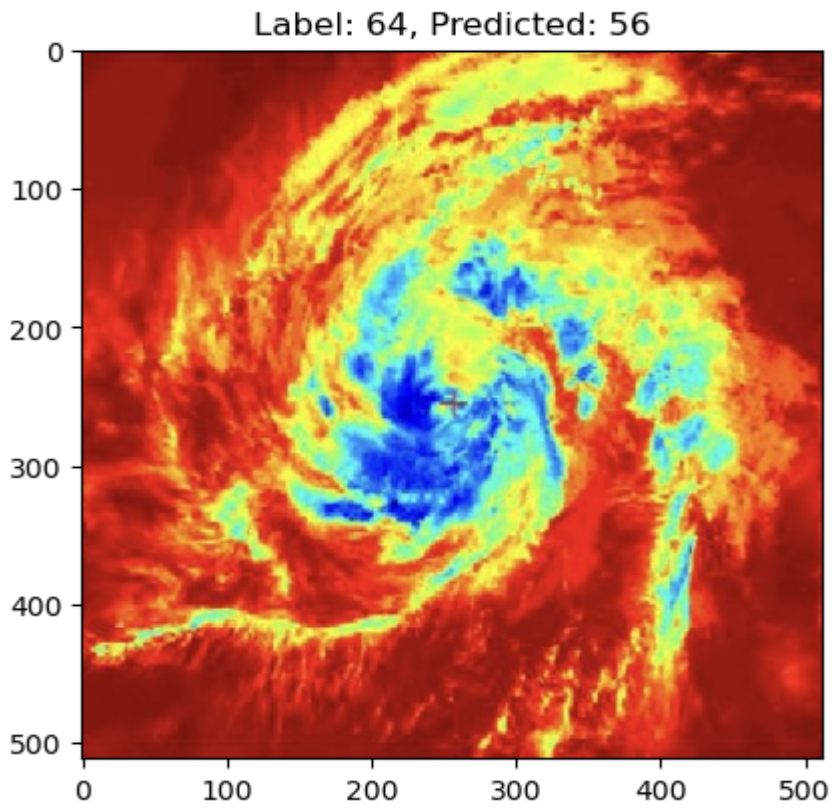

Label: 67, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

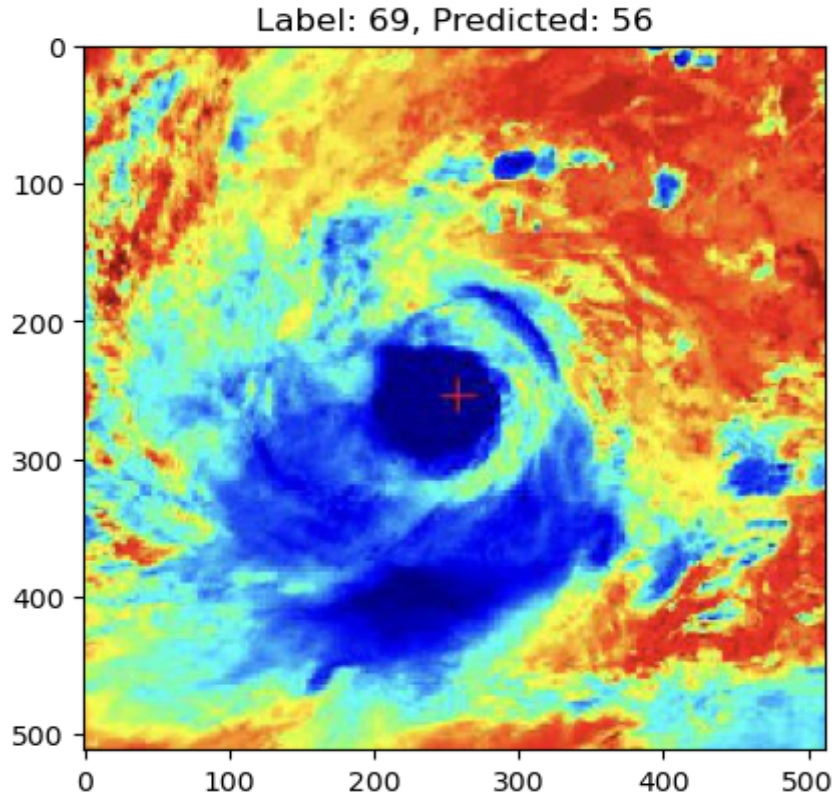

Label: 40, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

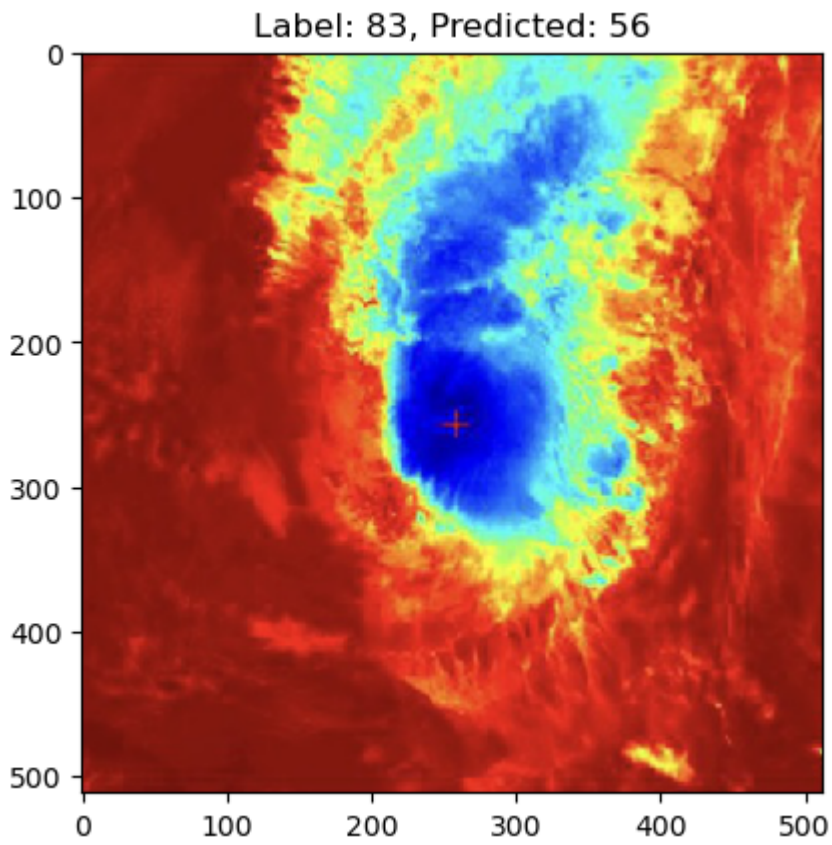

Label: 41, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

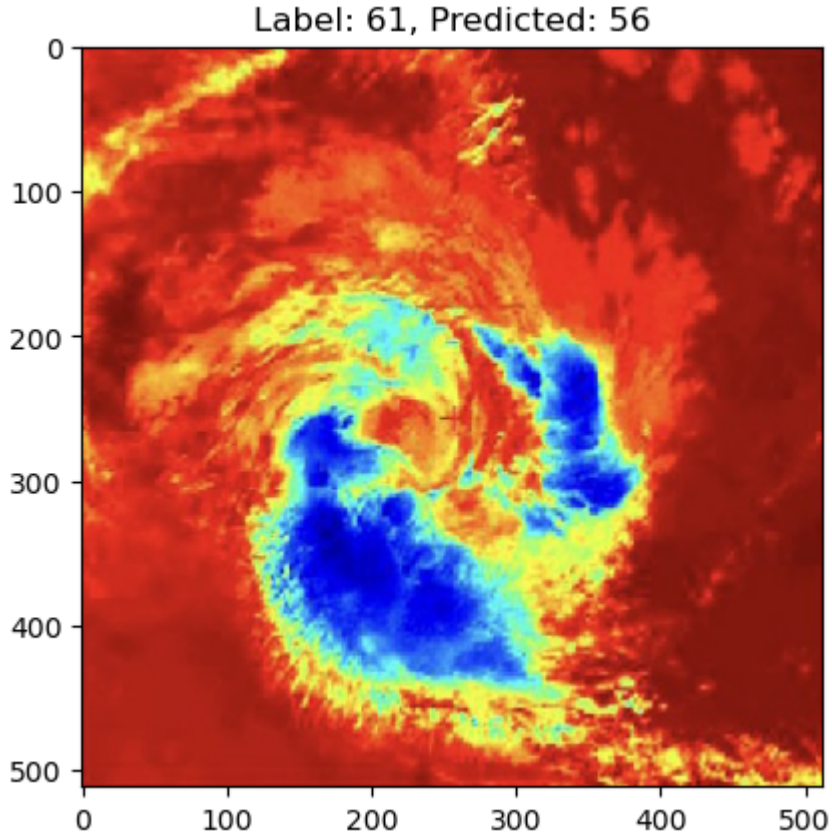

Label: 31, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

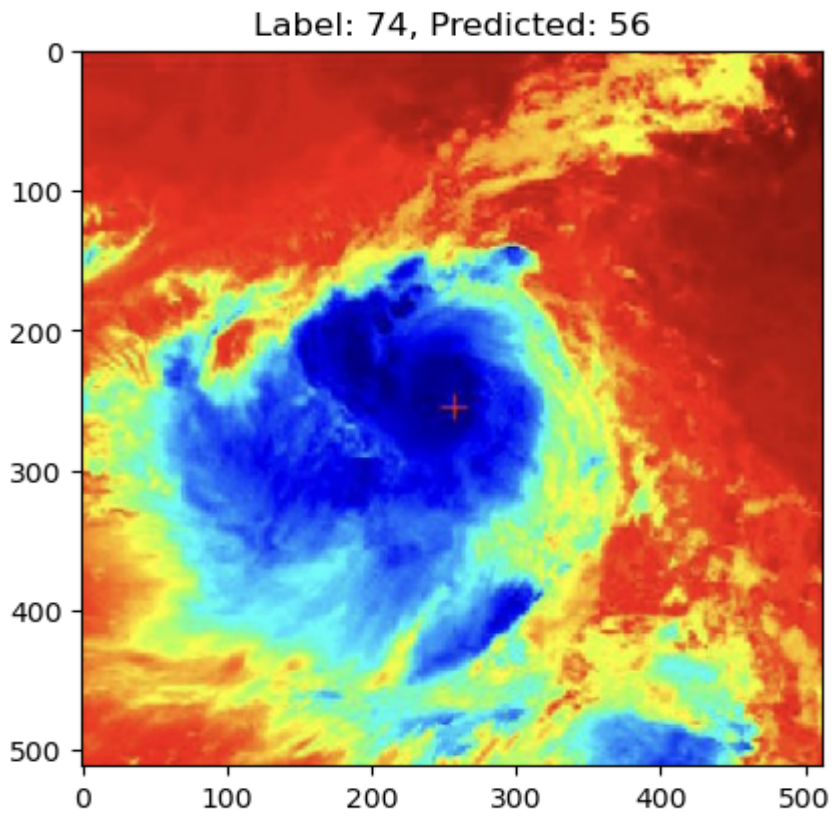

Label: 35, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

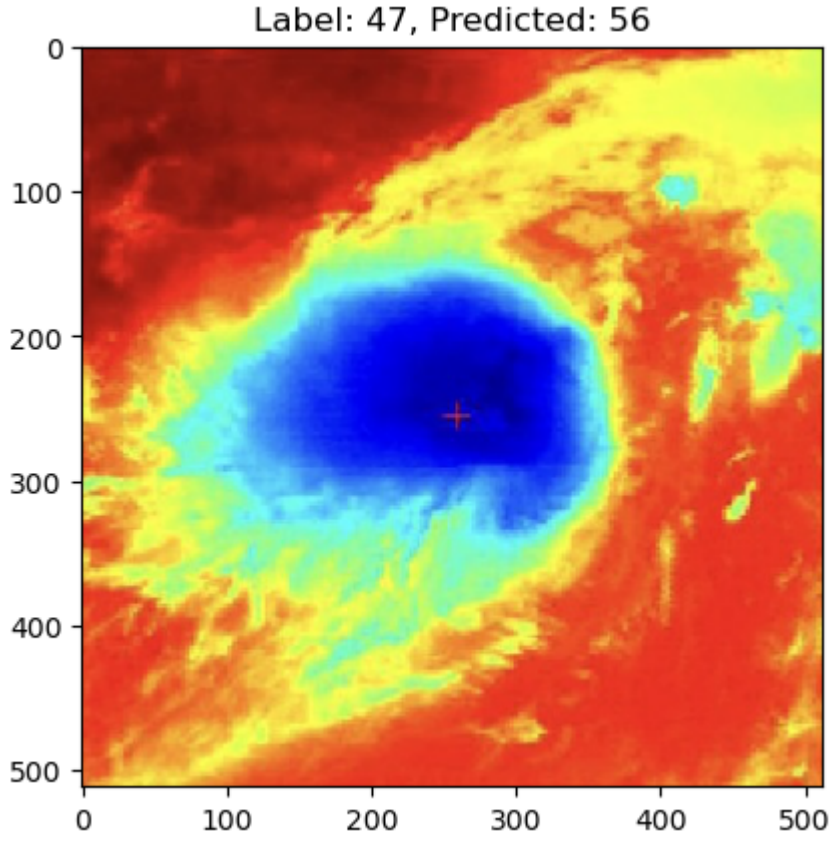

Label: 64, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

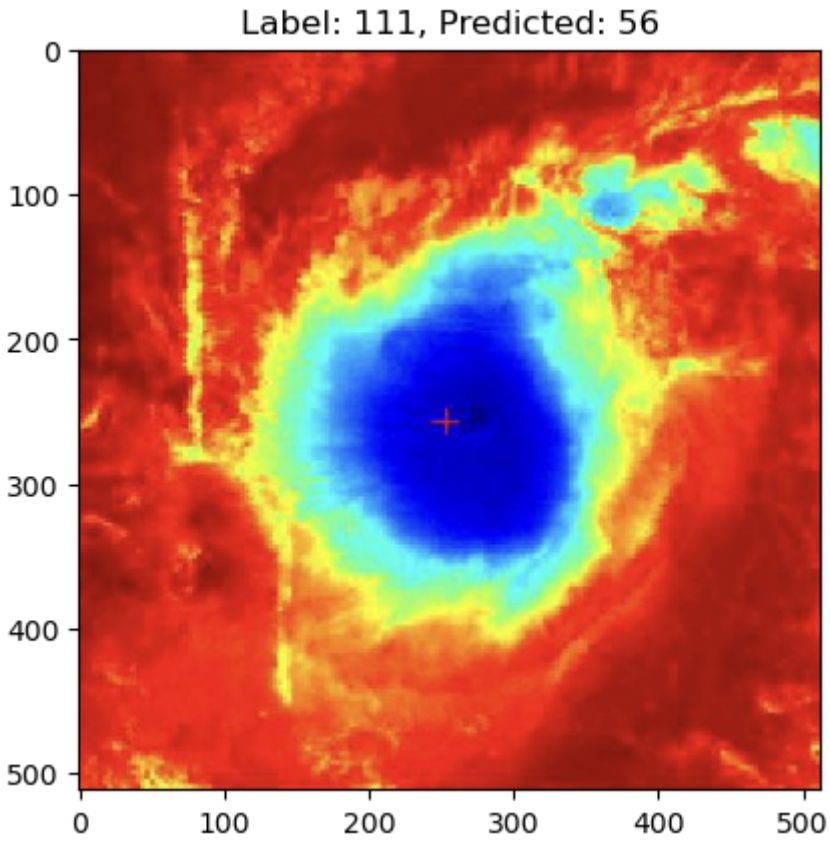

Label: 69, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

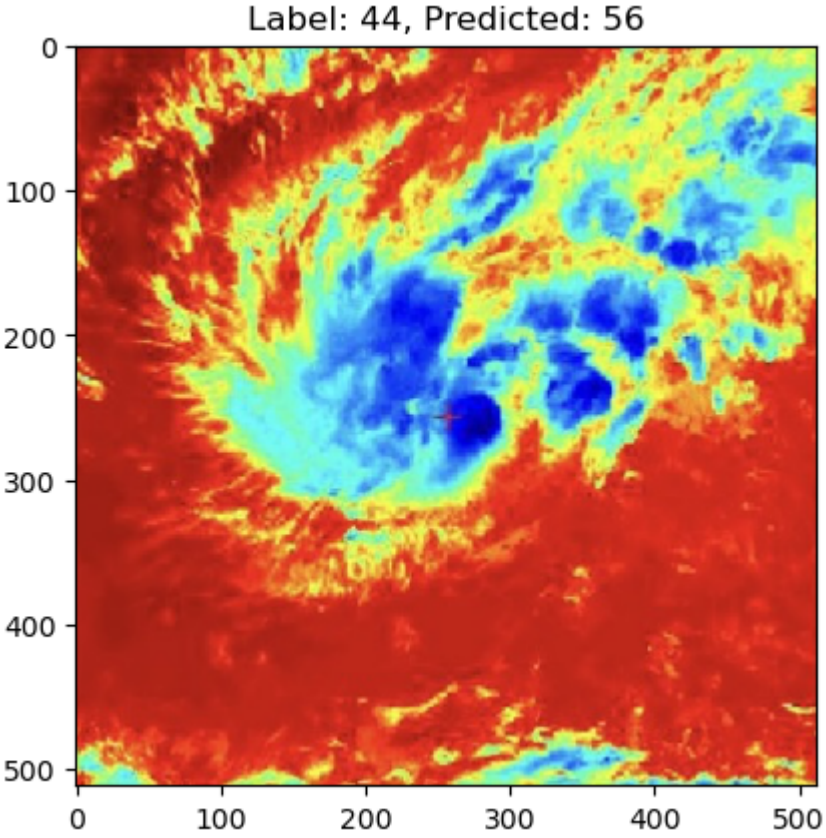

Label: 83, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500


Label: 61, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500


Label: 74, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500


Label: 47, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500


Label: 111, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500


Label: 44, Predicted: 56
0
100
200
300
400
500
0
100
200
300
400
500

Conclusion:

In conclusion, our project demonstrates the potential of machine learning in predicting natural disasters. By leveraging historical data and advanced modelling techniques, we can improve disaster preparedness and response, ultimately saving lives and reducing damage. Thank you for your attention, and I am happy to answer any questions you may have.

Bibliography:

Data Set from-

- Nasa: https://www.nasa.gov/

- https://sedac.ciesin.columbia.edu/downloads/data/pend/pend-gdis-1960-2018/pend-gdis-1960-2018-disasterlocations-gdb.zip