Final Project - Adverse External Influence Recognition for Residential Properties

Author - Ryan Lucero

```
# Importing important packages
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from keras.layers import Dropout, Dense, Flatten, BatchNormalization, GlobalAverageP
from keras.models import Sequential
from keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import matplotlib.pyplot as plt
%matplotlib inline
```

```
2024-08-20 22:35:59.469801: E external/local_xla/xla/stream_executor/cuda/cuda_d
2024-08-20 22:35:59.469908: E external/local_xla/xla/stream_executor/cuda/cuda_f
2024-08-20 22:35:59.623541: E external/local_xla/xla/stream_executor/cuda/cuda_b
```

Data Input - No Augmentation

```
import os
print(os.getcwd())

# change the directory to match your dateset location
data_dir = '/kaggle/input/subjects/Subjects' ###kaggle

Name = os.listdir(data_dir)
print(Name)
print(len(Name))

N=list(range(len(Name)))
normal_mapping=dict(zip(Name,N))
reverse_mapping=dict(zip(N,Name))
```

```
/kaggle/working
['Neutral', 'Adverse']
2
```

```
preprocess_input = tf.keras.applications.resnet.preprocess_input

# Rescaling the input image. No augmentation
train_datagen = ImageDataGenerator(rescale = 1./255,
        validation_split=0.2,
        preprocessing_function=preprocess_input,
        fill_mode='nearest')

validation_datagen = ImageDataGenerator(rescale = 1./255,validation_split=0.2)

train_batches = train_datagen.flow_from_directory(
    data_dir,
    target_size=(255,255),
    class_mode='binary',
  batch_size=5,subset = "training")

validation_batches = validation_datagen.flow_from_directory(
    data_dir,
    target_size=(255,255),
    class_mode='binary',
  batch_size=5,subset = "validation")
```

```
Found 400 images belonging to 2 classes.
Found 100 images belonging to 2 classes.
```

```
validation_batches.class_indices.keys()
```

```
dict_keys(['Adverse', 'Neutral'])
```

```
unique_values, counts = np.unique(validation_batches.labels, return_counts=True)

print(unique_values)
print(counts)
```

```
[0 1]
[50 50]
```

Basic Model

```
model = tf.keras.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape = (255, 255, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(3))
model.add(layers.Dense(1, activation="sigmoid"))
```

```
/opt/conda/lib/python3.10/site-packages/keras/src/layers/convolutional/base_conv
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 253, 253, 32)	896
max_pooling2d (MaxPooling2D)	(None, 126, 126, 32)	0
conv2d_1 (Conv2D)	(None, 124, 124, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 64)	36,928
flatten (Flatten)	(None, 230400)	0
dense (Dense)	(None, 64)	14,745,664
dense_1 (Dense)	(None, 3)	195
dense_2 (Dense)	(None, 1)	4

Total params: 14,802,183 (56.47 MB)
Trainable params: 14,802,183 (56.47 MB)
Non-trainable params: 0 (0.00 B)

```
model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

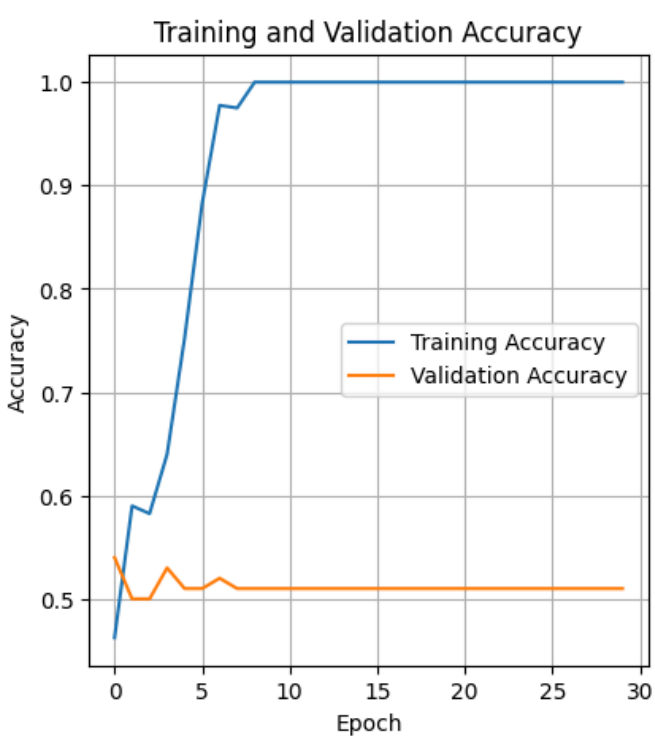
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath="model.h5.keras",
        save_best_only=True,
        monitor="val_loss")
]

history = model.fit(
    train_batches,
    epochs=30,
    validation_data=validation_batches,
    callbacks=callbacks
)
```

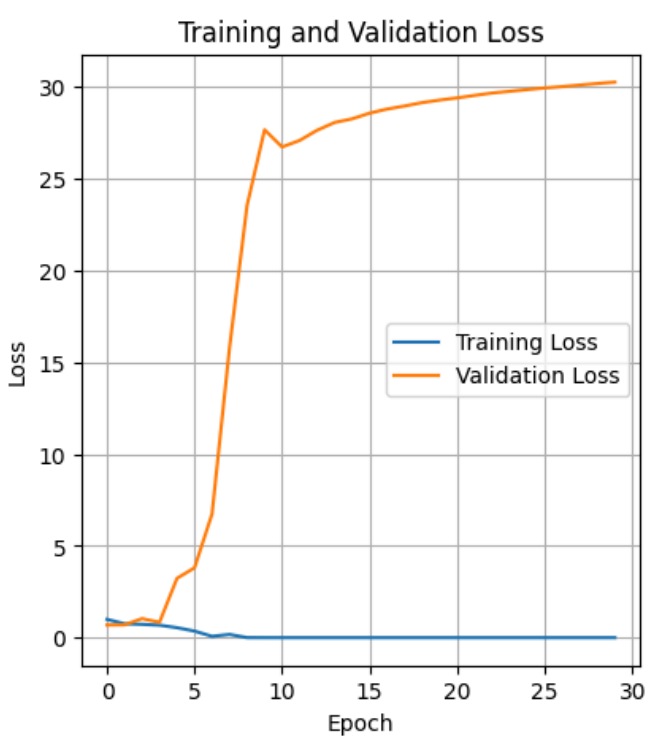
```
Epoch 1/30
/opt/conda/lib/python3.10/site-packages/keras/src/trainers/data_adapters/py_da
  self._warn_if_super_not_called()
 3/80 ━━━━━━━━━━━━━━━━━━━━ 4s 55ms/step - accuracy: 0.3000 - loss: 3.4710WARNI
I0000 00:00:1724193427.166665     125 device_compiler.h:186] Compiled cluster
W0000 00:00:1724193427.187768     125 graph_launch.cc:671] Fallback to op-by-o
79/80 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - accuracy: 0.4532 - loss: 1.5119W0000
80/80 ━━━━━━━━━━━━━━━━━━━━ 15s 97ms/step - accuracy: 0.4534 - loss: 1.4992 - v
Epoch 2/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 47ms/step - accuracy: 0.5502 - loss: 0.7683 - va
Epoch 3/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 47ms/step - accuracy: 0.5549 - loss: 0.7480 - va
Epoch 4/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 46ms/step - accuracy: 0.6083 - loss: 0.6588 - va
Epoch 5/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 0.7553 - loss: 0.5400 - va
Epoch 6/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 0.8872 - loss: 0.2875 - va
Epoch 7/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 48ms/step - accuracy: 0.9804 - loss: 0.0665 - va
Epoch 8/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 46ms/step - accuracy: 0.9954 - loss: 0.0332 - va
Epoch 9/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 46ms/step - accuracy: 1.0000 - loss: 0.0044 - va
Epoch 10/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 5s 53ms/step - accuracy: 1.0000 - loss: 1.2410e-04 -
Epoch 11/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 46ms/step - accuracy: 1.0000 - loss: 5.3255e-06 -
Epoch 12/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 1.0000 - loss: 2.9238e-06 -
Epoch 13/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 1.0000 - loss: 1.8296e-06 -
Epoch 14/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 47ms/step - accuracy: 1.0000 - loss: 9.2527e-07 -
Epoch 15/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 47ms/step - accuracy: 1.0000 - loss: 1.0012e-06 -
Epoch 16/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 46ms/step - accuracy: 1.0000 - loss: 8.3266e-07 -
Epoch 17/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 46ms/step - accuracy: 1.0000 - loss: 6.0407e-07 -
Epoch 18/30
```

```
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 46ms/step - accuracy: 1.0000 - loss: 5.0386e-07
Epoch 19/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 46ms/step - accuracy: 1.0000 - loss: 5.3372e-07
Epoch 20/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 1.0000 - loss: 4.9404e-07
Epoch 21/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 1.0000 - loss: 3.5674e-07
Epoch 22/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 47ms/step - accuracy: 1.0000 - loss: 4.5917e-07
Epoch 23/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 46ms/step - accuracy: 1.0000 - loss: 2.2350e-07
Epoch 24/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 1.0000 - loss: 3.7713e-07
Epoch 25/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 1.0000 - loss: 4.3802e-07
Epoch 26/30
```

Basic Model - Plots

```
def generate_plots(model_history):
    fig, axes = plt.subplots(1, 2, figsize=(10, 5))

    ax1 = axes[0]
    ax2 = axes[1]

    ax1.plot(model_history.history['accuracy'], label='Training Accuracy')
    ax1.plot(model_history.history['val_accuracy'], label='Validation Accuracy')
    ax1.set_title('Training and Validation Accuracy')
    ax1.set_xlabel('Epoch')
    ax1.set_xticks(range(0,31,5))
    ax1.set_ylabel('Accuracy')
    ax1.legend()
    ax1.grid(True)

    ax2.plot(model_history.history['loss'], label='Training Loss')
    ax2.plot(model_history.history['val_loss'], label='Validation Loss')
    ax2.set_title('Training and Validation Loss')
    ax2.set_xlabel('Epoch')
    ax2.set_xticks(range(0,31,5))
    ax2.set_ylabel('Loss')
    ax2.legend()
    ax2.grid(True)

    plt.show()


generate_plots(history)
```





Improved Model

```
model_2 = tf.keras.Sequential()
model_2.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape = (255, 255, 3)
model_2.add(layers.MaxPooling2D((2, 2)))
model_2.add(layers.Conv2D(64, (3, 3), activation='relu'))
model_2.add(layers.MaxPooling2D((2, 2)))
model_2.add(layers.Conv2D(128, (3, 3), activation='relu'))
model_2.add(layers.MaxPooling2D((2, 2)))
model_2.add(layers.Conv2D(256, (3, 3), activation='relu'))
model_2.add(layers.MaxPooling2D((2, 2)))
model_2.add(layers.Conv2D(256, (3, 3), activation='relu'))
model_2.add(layers.Flatten())
model_2.add(layers.Dropout(.5))
model_2.add(layers.Dense(64, activation='relu'))
model_2.add(layers.Dense(3))
model_2.add(layers.Dense(1, activation="sigmoid"))
```

```
model_2.summary()
```

Show hidden output

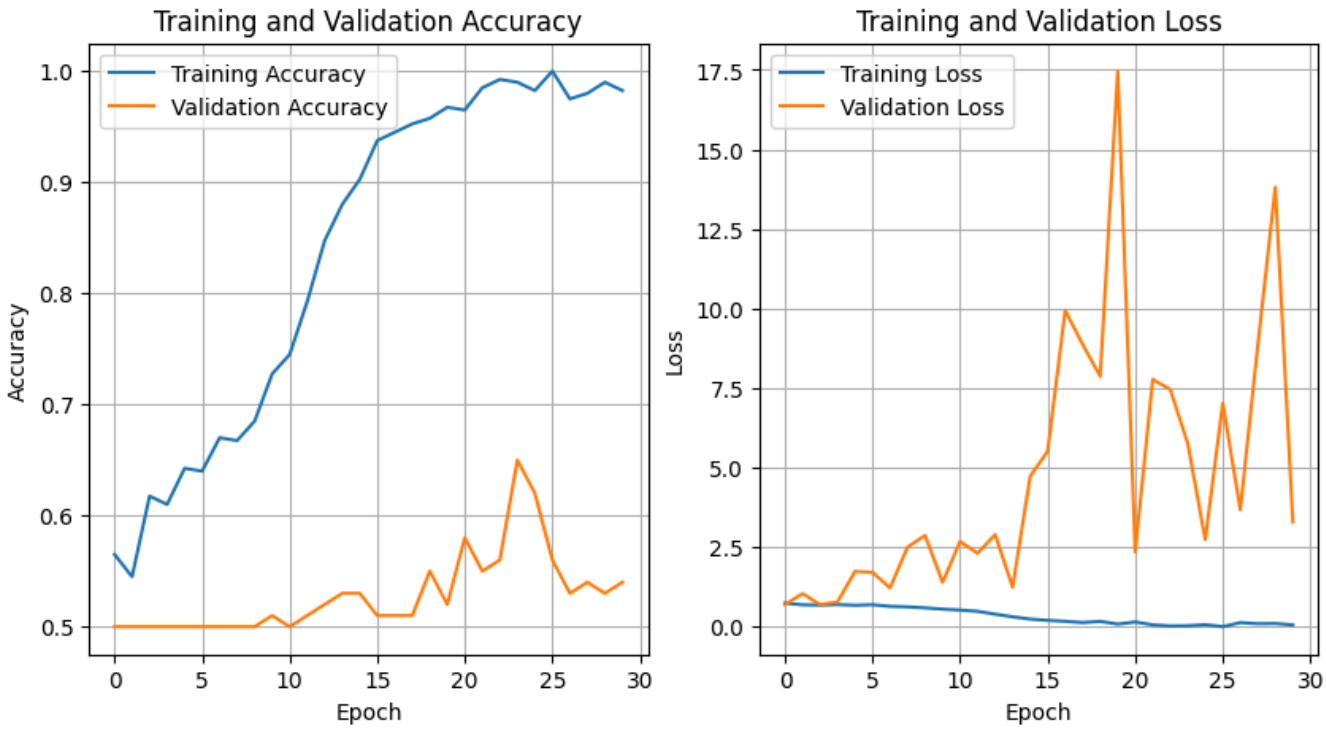
```
model_2.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])

early_stopping = EarlyStopping(monitor='val_accuracy', patience=10, mode='max')

model_checkpoint = ModelCheckpoint(filepath='model_2.h5.keras',
                                   save_best_only=True,
                                   monitor='val_accuracy',
                                   mode='max')

#callbacks = [early_stopping, model_checkpoint]
callbacks = [model_checkpoint]

history_2 = model_2.fit(
    train_batches,
    epochs=30,
    validation_data=validation_batches,
    callbacks=callbacks
)
```

```
Epoch 1/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 55ms/step - accuracy: 0.5415 - loss: 0.8024 - va
Epoch 2/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 0.4911 - loss: 0.6825 - va
Epoch 3/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 5s 45ms/step - accuracy: 0.6209 - loss: 0.6876 - va
Epoch 4/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 46ms/step - accuracy: 0.6019 - loss: 0.6900 - va
Epoch 5/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 46ms/step - accuracy: 0.6539 - loss: 0.6646 - va
Epoch 6/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 46ms/step - accuracy: 0.6463 - loss: 0.6875 - va
Epoch 7/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 50ms/step - accuracy: 0.6935 - loss: 0.5983 - va
Epoch 8/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 0.6191 - loss: 0.6398 - va
Epoch 9/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 0.6648 - loss: 0.5907 - va
Epoch 10/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 47ms/step - accuracy: 0.7283 - loss: 0.5359 - va
Epoch 11/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 0.7311 - loss: 0.5122 - va
Epoch 12/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 0.8021 - loss: 0.5044 - va
Epoch 13/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 46ms/step - accuracy: 0.8484 - loss: 0.3429 - va
Epoch 14/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 47ms/step - accuracy: 0.8981 - loss: 0.2737 - va
Epoch 15/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 46ms/step - accuracy: 0.9267 - loss: 0.1943 - va
Epoch 16/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 44ms/step - accuracy: 0.9549 - loss: 0.1829 - va
```

```
Epoch 17/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 0.9415 - loss: 0.1799 - va
Epoch 18/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 0.9449 - loss: 0.1260 - va
Epoch 19/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 47ms/step - accuracy: 0.9639 - loss: 0.1442 - va
Epoch 20/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 47ms/step - accuracy: 0.9781 - loss: 0.0606 - va
Epoch 21/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 49ms/step - accuracy: 0.9648 - loss: 0.1459 - va
Epoch 22/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 47ms/step - accuracy: 0.9895 - loss: 0.0372 - va
Epoch 23/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 47ms/step - accuracy: 0.9913 - loss: 0.0265 - va
Epoch 24/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 48ms/step - accuracy: 0.9972 - loss: 0.0177 - va
Epoch 25/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 0.9873 - loss: 0.0558 - va
Epoch 26/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 1.0000 - loss: 0.0018 - va
Epoch 27/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 0.9868 - loss: 0.0709 - va
Epoch 28/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 4s 45ms/step - accuracy: 0.9695 - loss: 0.0988 - va
Epoch 29/30
```

Improved Model - Plots

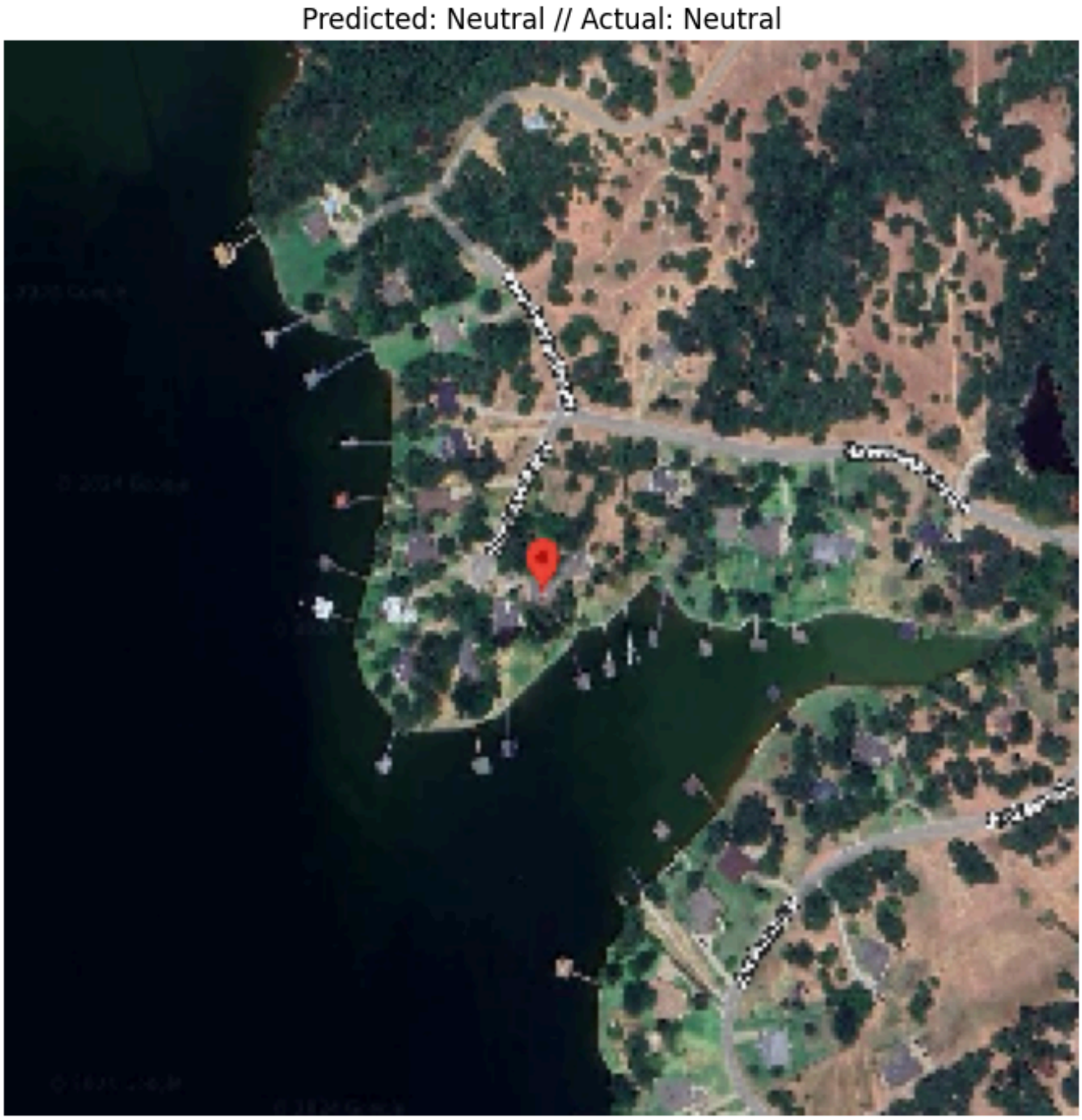
```
generate_plots(history_2)
```

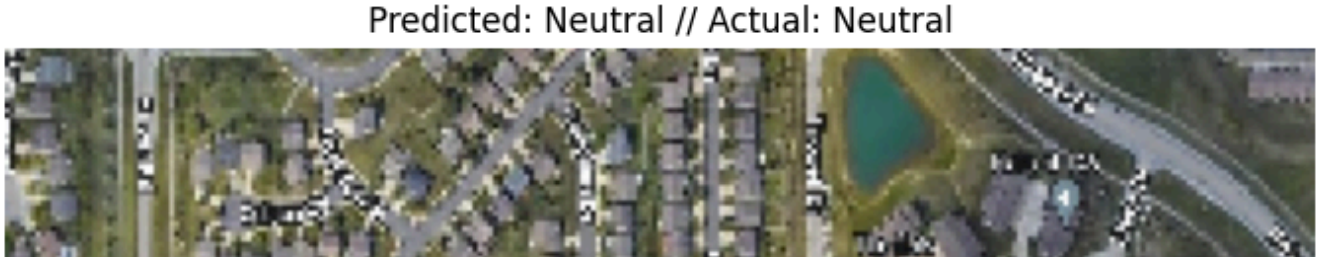


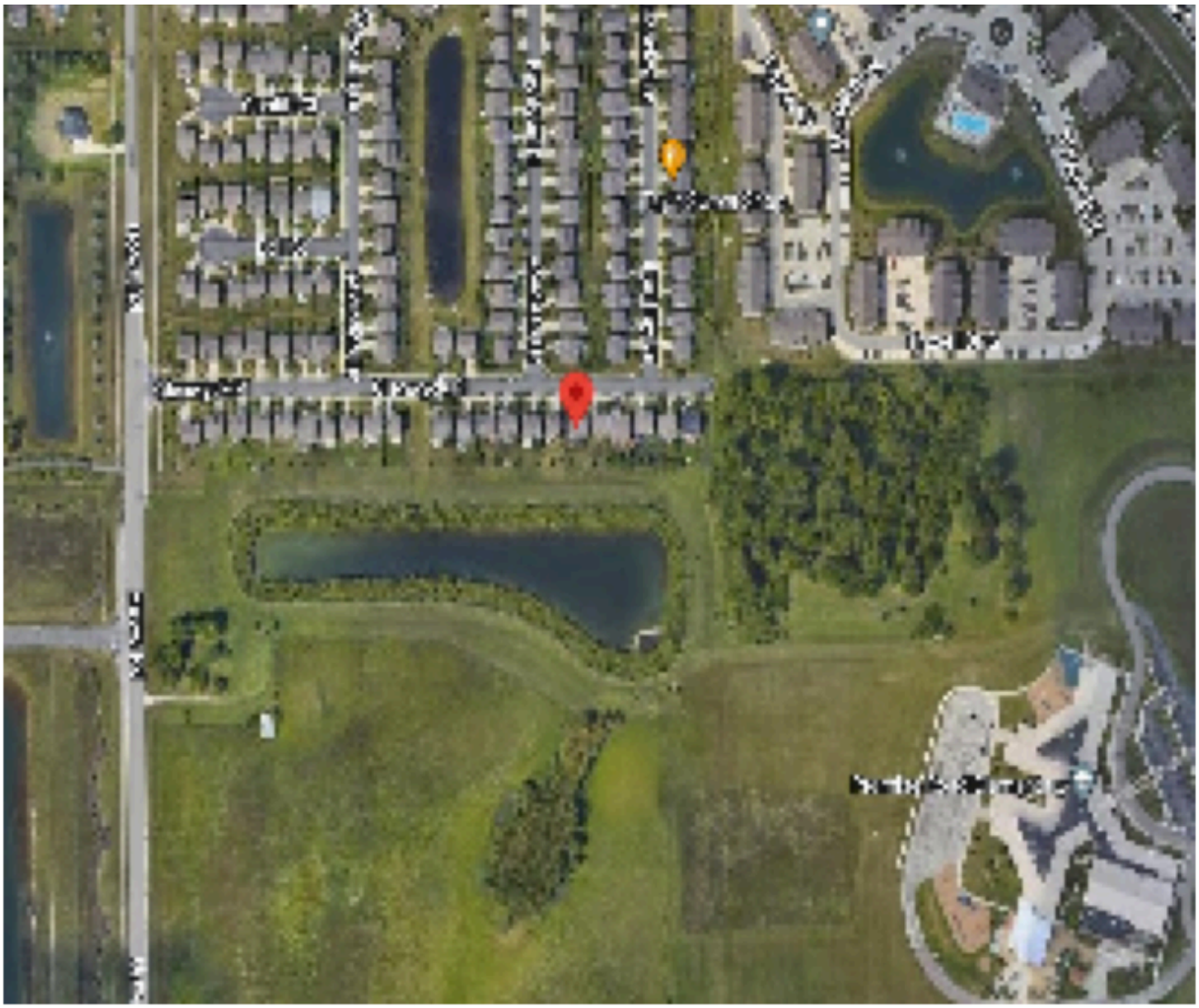
Improved Model - Sample Batch Predictions

```
x, y = validation_batches[0]

class_names = ('Adverse', 'Neutral')

preds = model_2.predict(validation_batches)
pred_labels = np.where(preds < .5, 0, 1)

plt.figure(figsize=(10, 50))

for i in range(0,5):
    image = x[i]
    label = int(y[i])
    plt.subplot(5, 1, i+1)
    plt.imshow(image)
    plt.title(f'Predicted: {class_names[int(pred_labels[label])]} // Actual: {class_
    plt.axis('off')

plt.show()
```

```
20/20 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step
/tmp/ipykernel_34/1490707011.py:15: DeprecationWarning: Conversion of an array w
  plt.title(f'Predicted: {class_names[int(pred_labels[label])]} // Actual: {clas
```

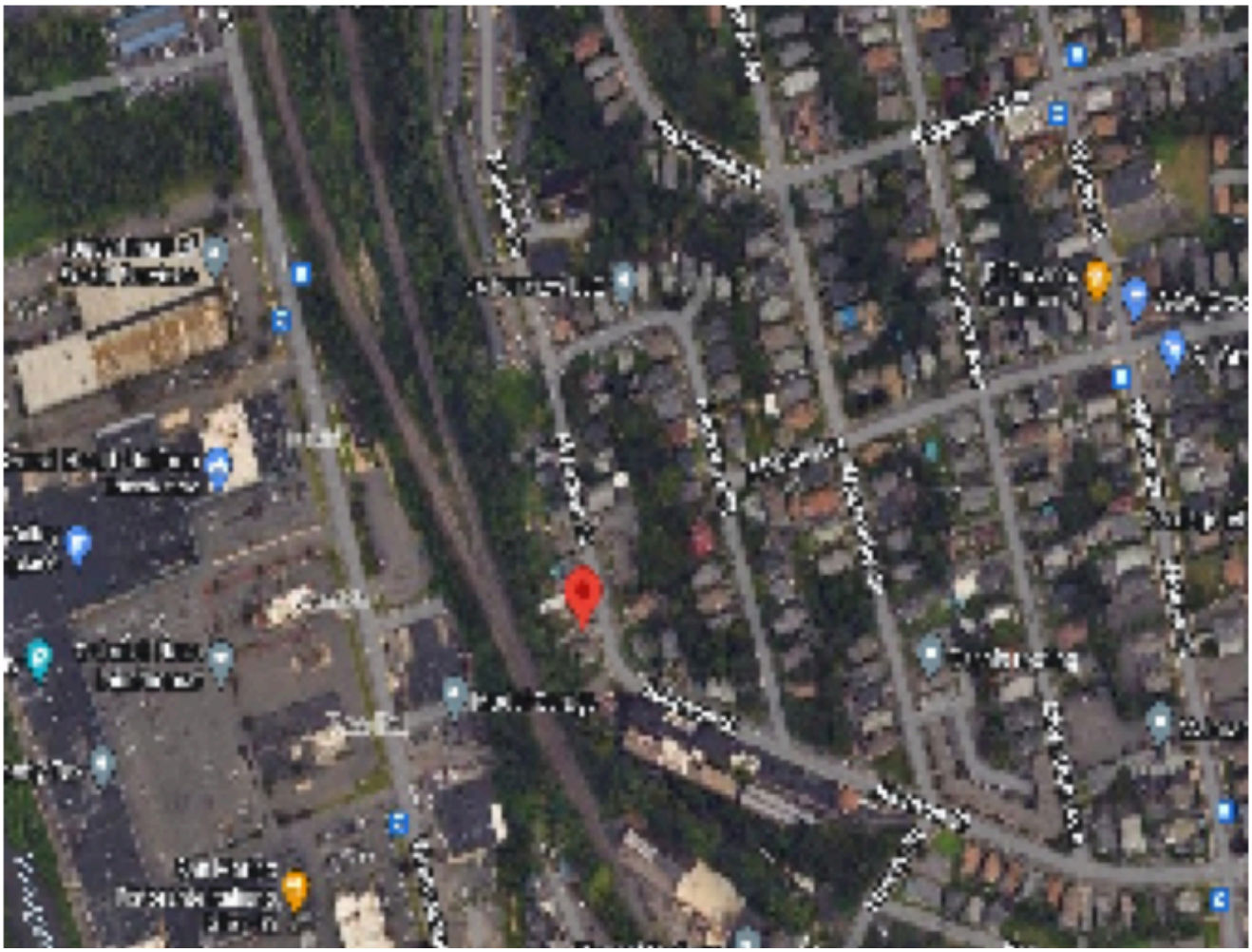


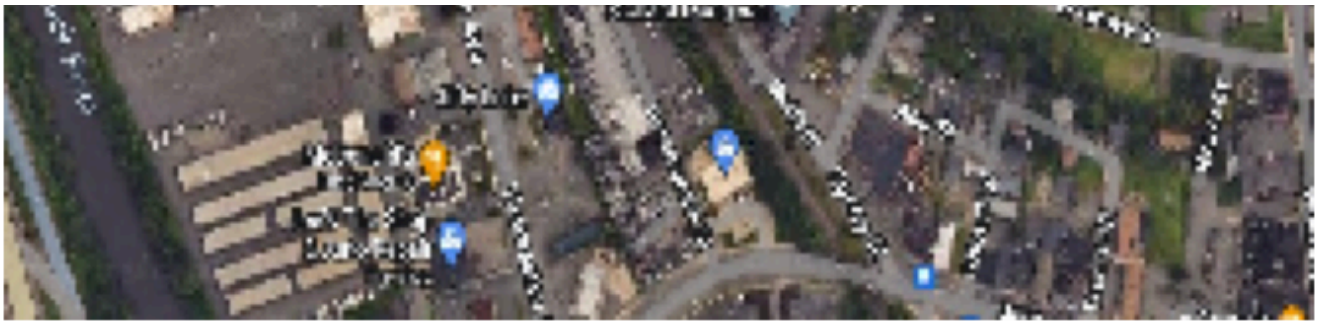


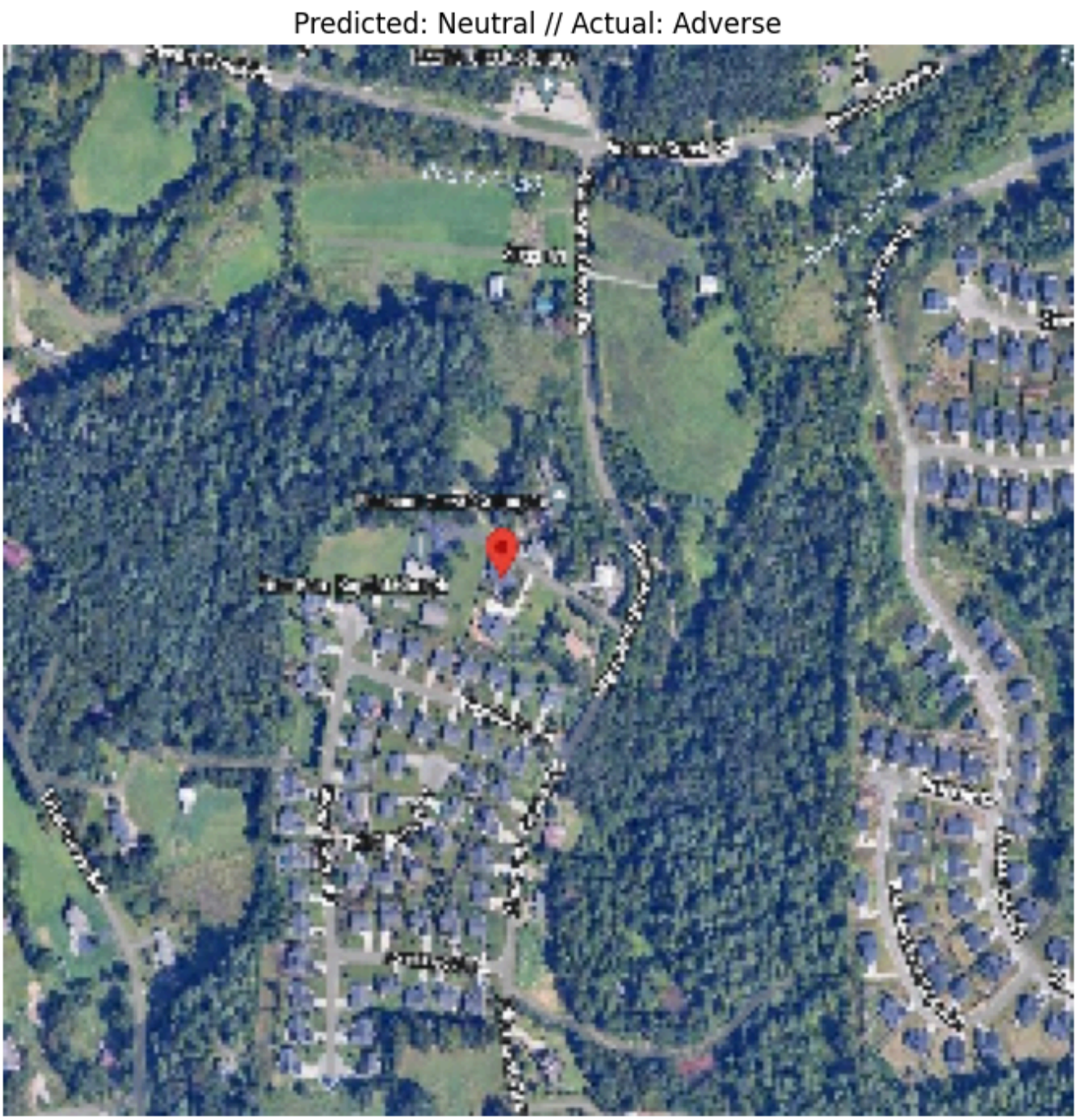


Predicted: Neutral // Actual: Neutral

Predicted: Neutral // Actual: Adverse






Predicted: Neutral // Actual: Adverse

Data Augmentation

```
####Augmenting####

# Rescaling the input image as well as Data Augmentation
train_datagen = ImageDataGenerator(rescale = 1./255,
        validation_split=0.2,
        rotation_range=30,
        width_shift_range=0.2,
        preprocessing_function=preprocess_input,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')

validation_datagen = ImageDataGenerator(rescale = 1./255,validation_split=0.2)

train_batches = train_datagen.flow_from_directory(
    data_dir,
    target_size=(255,255),
    class_mode='binary',
  batch_size=5,subset = "training")

validation_batches = validation_datagen.flow_from_directory(
    data_dir,
    target_size=(255,255),
    class_mode='binary',
  batch_size=5,subset = "validation")
```

```
Found 400 images belonging to 2 classes.
Found 100 images belonging to 2 classes.
```

Improved Model with Augmentation

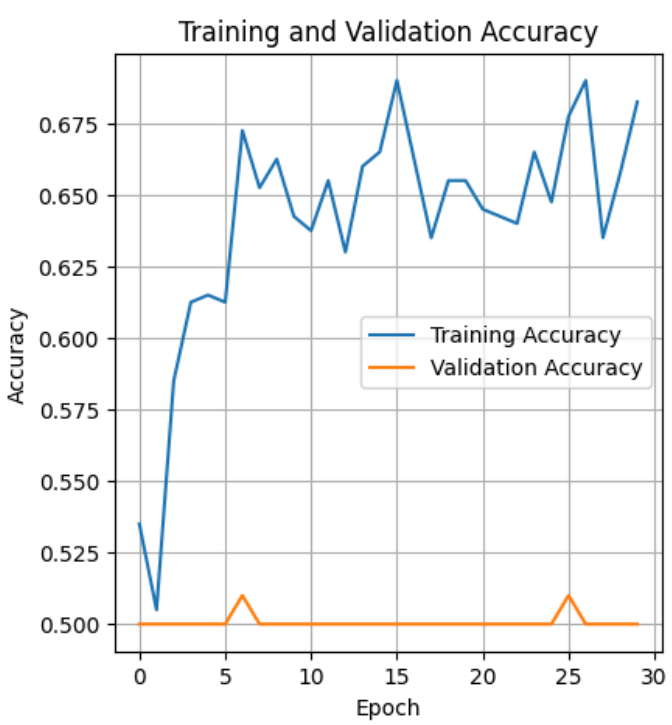
```
model_2.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

early_stopping = EarlyStopping(monitor='val_accuracy', patience=10, mode='max')

model_checkpoint = ModelCheckpoint(filepath='model_2_augmentation.h5.keras',
                                   save_best_only=True,
                                   monitor='val_accuracy',
                                   mode='max')

#callbacks = [early_stopping, model_checkpoint]
callbacks = [model_checkpoint]

history_2_with_augmentation = model_2.fit(
    train_batches,
    epochs=30,
    validation_data=validation_batches,
    callbacks=callbacks
)
```

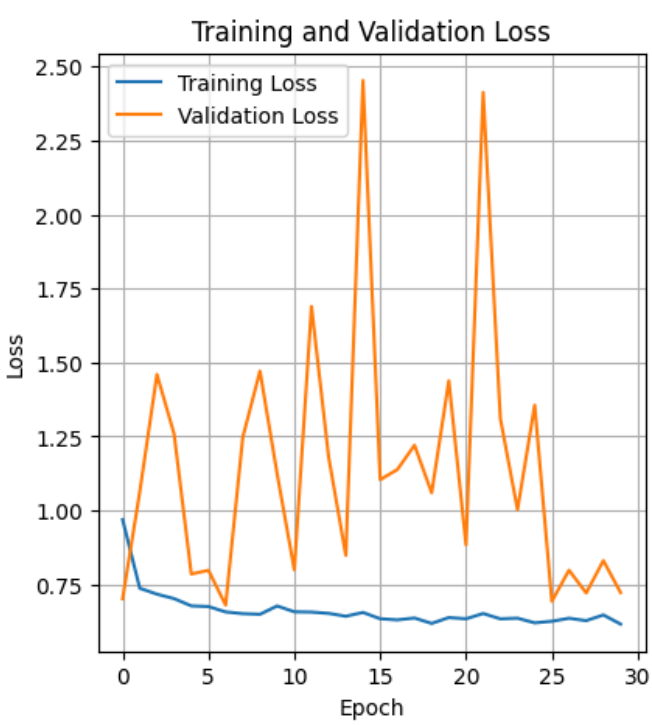
```
Epoch 1/30
/opt/conda/lib/python3.10/site-packages/keras/src/trainers/data_adapters/py_da
  self._warn_if_super_not_called()
80/80 ━━━━━━━━━━━━━━━━━━━━ 14s 121ms/step - accuracy: 0.5370 - loss: 1.7180 -
Epoch 2/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 116ms/step - accuracy: 0.4415 - loss: 0.7137 -
Epoch 3/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 115ms/step - accuracy: 0.5876 - loss: 0.7002 -
Epoch 4/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 114ms/step - accuracy: 0.6208 - loss: 0.7798 -
Epoch 5/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 116ms/step - accuracy: 0.5957 - loss: 0.6771 -
Epoch 6/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 112ms/step - accuracy: 0.6108 - loss: 0.6706 -
Epoch 7/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 115ms/step - accuracy: 0.7001 - loss: 0.6276 -
Epoch 8/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 117ms/step - accuracy: 0.6293 - loss: 0.6408 -
Epoch 9/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 113ms/step - accuracy: 0.6941 - loss: 0.6420 -
Epoch 10/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 113ms/step - accuracy: 0.6644 - loss: 0.6370 -
Epoch 11/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 114ms/step - accuracy: 0.6654 - loss: 0.6377 -
Epoch 12/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 112ms/step - accuracy: 0.6138 - loss: 0.6728 -
Epoch 13/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 113ms/step - accuracy: 0.6429 - loss: 0.6385 -
Epoch 14/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 115ms/step - accuracy: 0.6308 - loss: 0.6782 -
Epoch 15/30
```

```
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 113ms/step - accuracy: 0.6546 - loss: 0.6767 -
Epoch 16/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 111ms/step - accuracy: 0.6764 - loss: 0.6223 -
Epoch 17/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 113ms/step - accuracy: 0.6895 - loss: 0.5974 -
Epoch 18/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 112ms/step - accuracy: 0.6215 - loss: 0.6401 -
Epoch 19/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 111ms/step - accuracy: 0.6399 - loss: 0.6037 -
Epoch 20/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 112ms/step - accuracy: 0.6744 - loss: 0.6460 -
Epoch 21/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 115ms/step - accuracy: 0.6484 - loss: 0.6214 -
Epoch 22/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 113ms/step - accuracy: 0.6656 - loss: 0.6305 -
Epoch 23/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 112ms/step - accuracy: 0.6422 - loss: 0.6401 -
Epoch 24/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 114ms/step - accuracy: 0.6380 - loss: 0.6432 -
Epoch 25/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 119ms/step - accuracy: 0.5838 - loss: 0.6500 -
Epoch 26/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 112ms/step - accuracy: 0.7155 - loss: 0.5963 -
Epoch 27/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 116ms/step - accuracy: 0.6821 - loss: 0.6264 -
Epoch 28/30
```

Improved Model with Data Augmentation - Plots

```
generate_plots(history_2_with_augmentation)
```





Pre-Trained Model - Xception

```
####Pre-Trained Keras Model####
pre_trained = keras.applications.Xception(
    include_top=False,
    weights="imagenet",
    input_shape=(255, 255, 3)
)


for layer in pre_trained.layers:
    layer.trainable = False #freezing wieghts and biases


pre_trained.summary()
```

Show hidden output

```
pre_trained.layers[-1]
```

```
<Activation name=block14_sepconv2_act, built=True>
```

```
pre_trained.layers[-1].output
```

```
<KerasTensor shape=(None, 8, 8, 2048), dtype=float32, sparse=False,
name=keras_tensor_283>
```

```
last_layer = pre_trained.get_layer('block14_sepconv2_act')
last_output = last_layer.output


x = layers.Dense(2048, activation='relu')(last_output)
x = layers.Dropout(.2)(x)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Flatten()(x)
x = layers.Dense(1, activation='sigmoid')(x)

model_pt = keras.Model(pre_trained.input, x) #appending dense network to base model


model_pt.summary()
```

Show hidden output

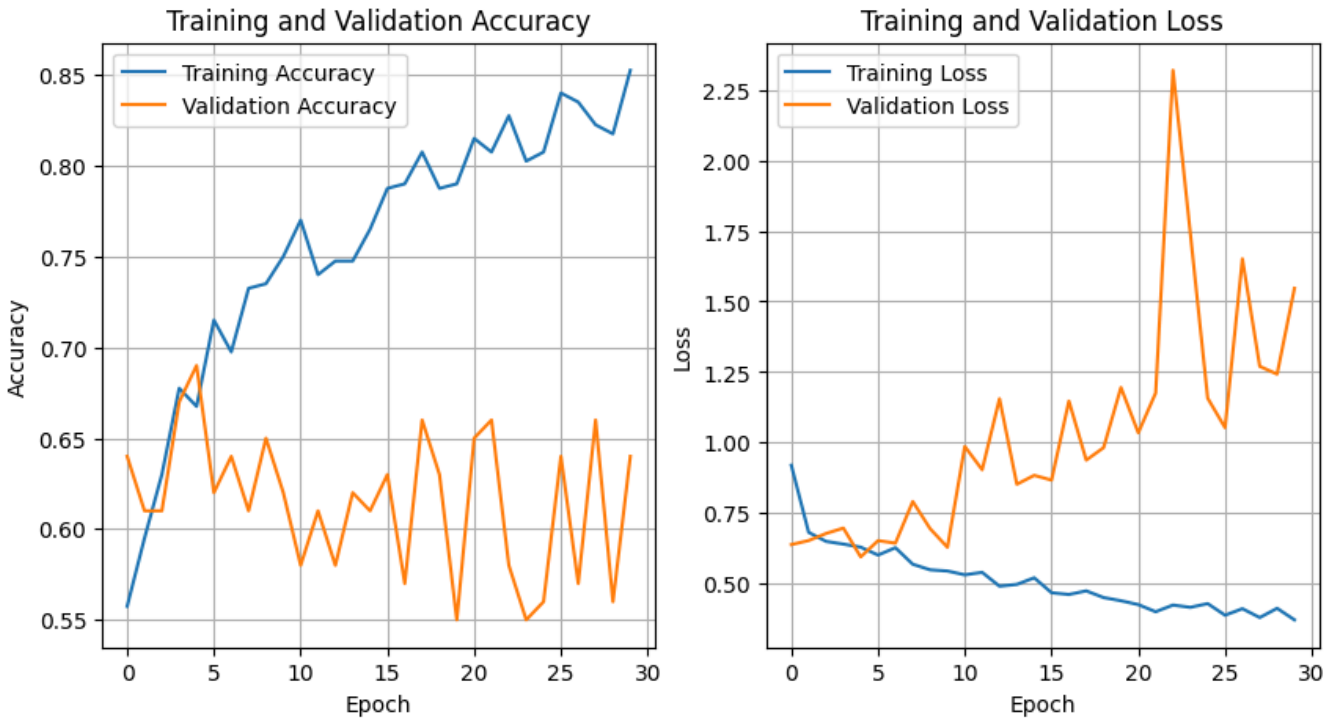
```
model_pt.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])


model_checkpoint = ModelCheckpoint(
      filepath="pre_trained_with_data_augmentation.h5.keras",
      save_best_only=True,
      monitor="val_accuracy")

callbacks = [model_checkpoint]


history_pt = model_pt.fit(
    train_batches,
    epochs=30,
    validation_data=validation_batches,
    callbacks=callbacks)
```

```
Epoch 1/30
/opt/conda/lib/python3.10/site-packages/keras/src/trainers/data_adapters/py_da
  self._warn_if_super_not_called()
 3/80 ━━━━━━━━━━━━━━━━━━━━ 6s 79ms/step - accuracy: 0.2000 - loss: 5.1901
79/80 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step - accuracy: 0.5029 - loss: 1.5755W000
80/80 ━━━━━━━━━━━━━━━━━━━━ 21s 163ms/step - accuracy: 0.5043 - loss: 1.5593 -
Epoch 2/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 116ms/step - accuracy: 0.5998 - loss: 0.6683 -
Epoch 3/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 115ms/step - accuracy: 0.6711 - loss: 0.6234 -
Epoch 4/30
```

```
80/80 ━━━━━━━━━━━━━━━━━━━━ 11s 128ms/step - accuracy: 0.6902 - loss: 0.6328 -
Epoch 5/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 11s 126ms/step - accuracy: 0.6553 - loss: 0.6518 -
Epoch 6/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 116ms/step - accuracy: 0.6831 - loss: 0.6096 -
Epoch 7/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 118ms/step - accuracy: 0.7161 - loss: 0.6154 -
Epoch 8/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 116ms/step - accuracy: 0.7474 - loss: 0.5411 -
Epoch 9/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 116ms/step - accuracy: 0.7376 - loss: 0.5282 -
Epoch 10/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 117ms/step - accuracy: 0.7471 - loss: 0.5351 -
Epoch 11/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 116ms/step - accuracy: 0.7688 - loss: 0.5195 -
Epoch 12/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 117ms/step - accuracy: 0.7586 - loss: 0.4883 -
Epoch 13/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 120ms/step - accuracy: 0.7688 - loss: 0.4813 -
Epoch 14/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 118ms/step - accuracy: 0.7679 - loss: 0.4451 -
Epoch 15/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 116ms/step - accuracy: 0.7983 - loss: 0.4632 -
Epoch 16/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 117ms/step - accuracy: 0.7937 - loss: 0.4849 -
Epoch 17/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 117ms/step - accuracy: 0.7895 - loss: 0.4392 -
Epoch 18/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 116ms/step - accuracy: 0.8061 - loss: 0.4784 -
Epoch 19/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 117ms/step - accuracy: 0.7971 - loss: 0.4515 -
Epoch 20/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 118ms/step - accuracy: 0.8048 - loss: 0.4083 -
Epoch 21/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 115ms/step - accuracy: 0.8371 - loss: 0.3982 -
Epoch 22/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 117ms/step - accuracy: 0.8419 - loss: 0.3617 -
Epoch 23/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 115ms/step - accuracy: 0.8220 - loss: 0.4722 -
Epoch 24/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 114ms/step - accuracy: 0.7925 - loss: 0.4848 -
Epoch 25/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 118ms/step - accuracy: 0.8156 - loss: 0.4097 -
Epoch 26/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 10s 118ms/step - accuracy: 0.8295 - loss: 0.4070 -
Epoch 27/30
```

Xception - Plots

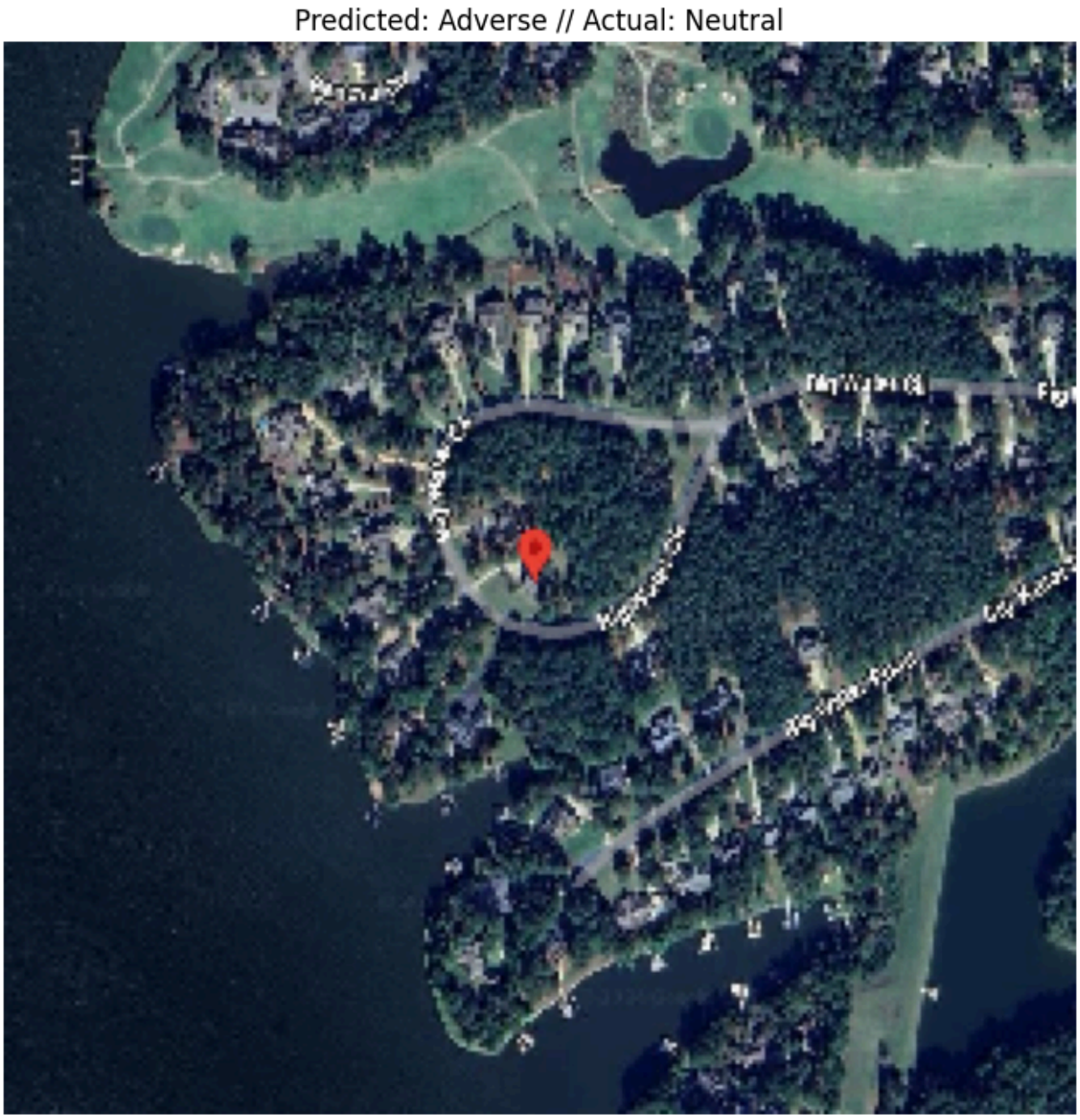
```
generate_plots(history_pt)
```



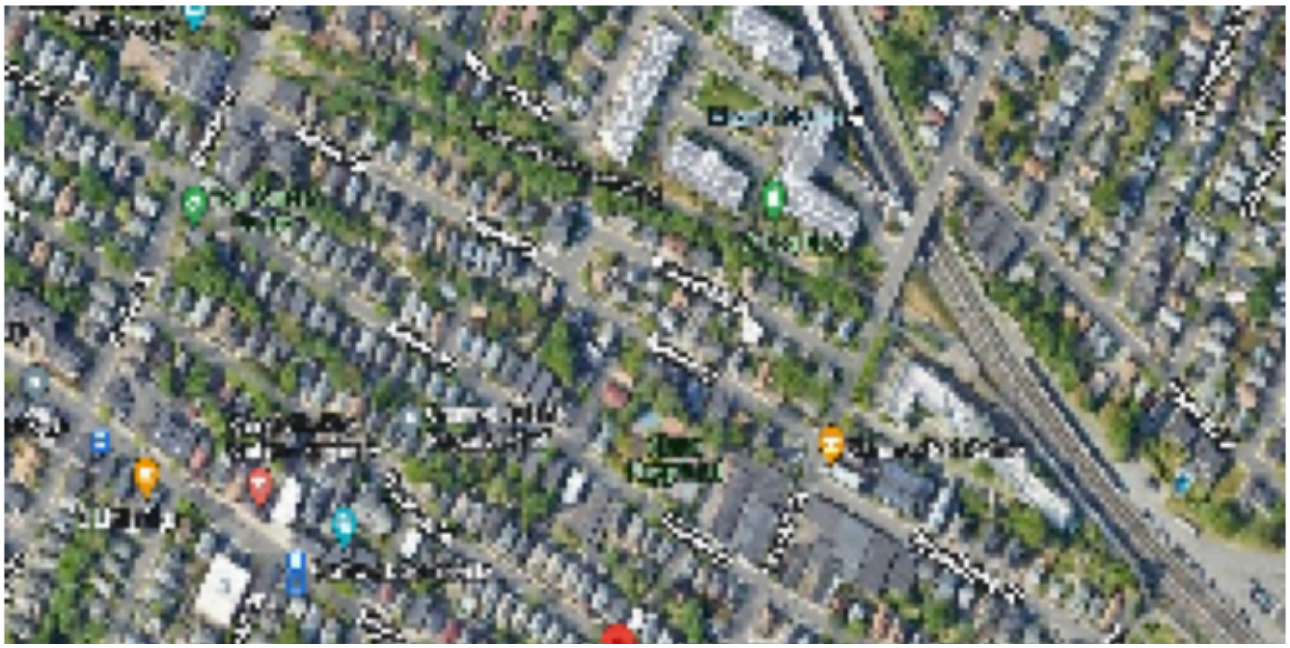
Xception - Sample Batch Predictions

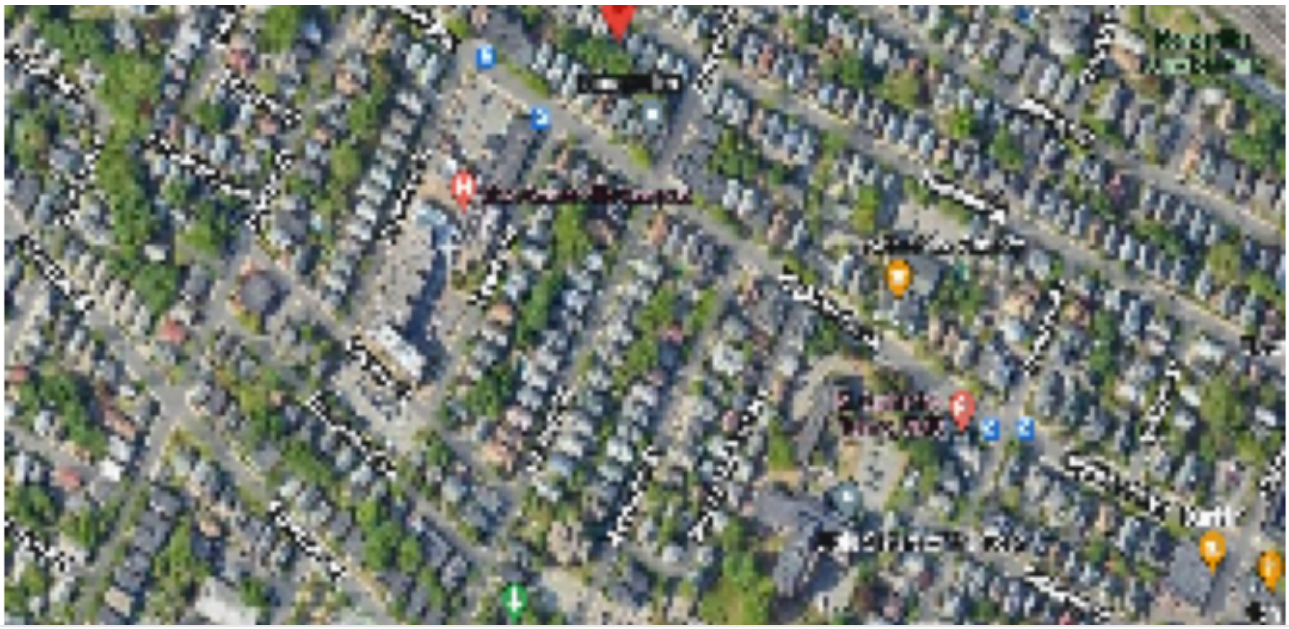
```
x, y = validation_batches[0]

preds = model_pt.predict(validation_batches)
pred_labels = np.where(preds < .5, 0, 1)

plt.figure(figsize=(10, 50))

for i in range(0,5):
    image = x[i]
    label = int(y[i])
    plt.subplot(5, 1, i+1)
    plt.imshow(image)
    plt.title(f'Predicted: {class_names[int(pred_labels[label])]} // Actual: {class_
    plt.axis('off')

plt.show()
```

```
 7/20 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/stepW0000 00:00:1723922187.140855    3865 g
20/20 ━━━━━━━━━━━━━━━━━━━━ 3s 36ms/step
/tmp/ipykernel_3778/1719383123.py:13: DeprecationWarning: Conversion of an arra
  plt.title(f'Predicted: {class_names[int(pred_labels[label])]} // Actual: {cla
```



Predicted: Adverse // Actual: Neutral





Pre-Trained Model - NASNetLarge

```
pre_trained_NAS = keras.applications.NASNetLarge(
    include_top=False,
    weights="imagenet",
    input_shape=(255, 255, 3)
)


for layer in pre_trained_NAS.layers:
    layer.trainable = False


pre_trained_NAS.summary()
```

Show hidden output

```
pre_trained_NAS.layers[-1]
```

```
<Activation name=activation_259, built=True>
```

```
pre_trained_NAS.layers[-1].output
```

```
<KerasTensor shape=(None, 8, 8, 4032), dtype=float32, sparse=False,
    name=keras_tensor_1327>
```

```
last_layer = pre_trained_NAS.get_layer('activation_259')
last_output = last_layer.output


x = layers.Dense(4032, activation='relu')(last_output)
x = layers.Dropout(.2)(x)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Flatten()(x)
x = layers.Dense(1, activation='sigmoid')(x)

model_NAS = keras.Model(pre_trained_NAS.input, x)


model_NAS.summary()
```

Show hidden output

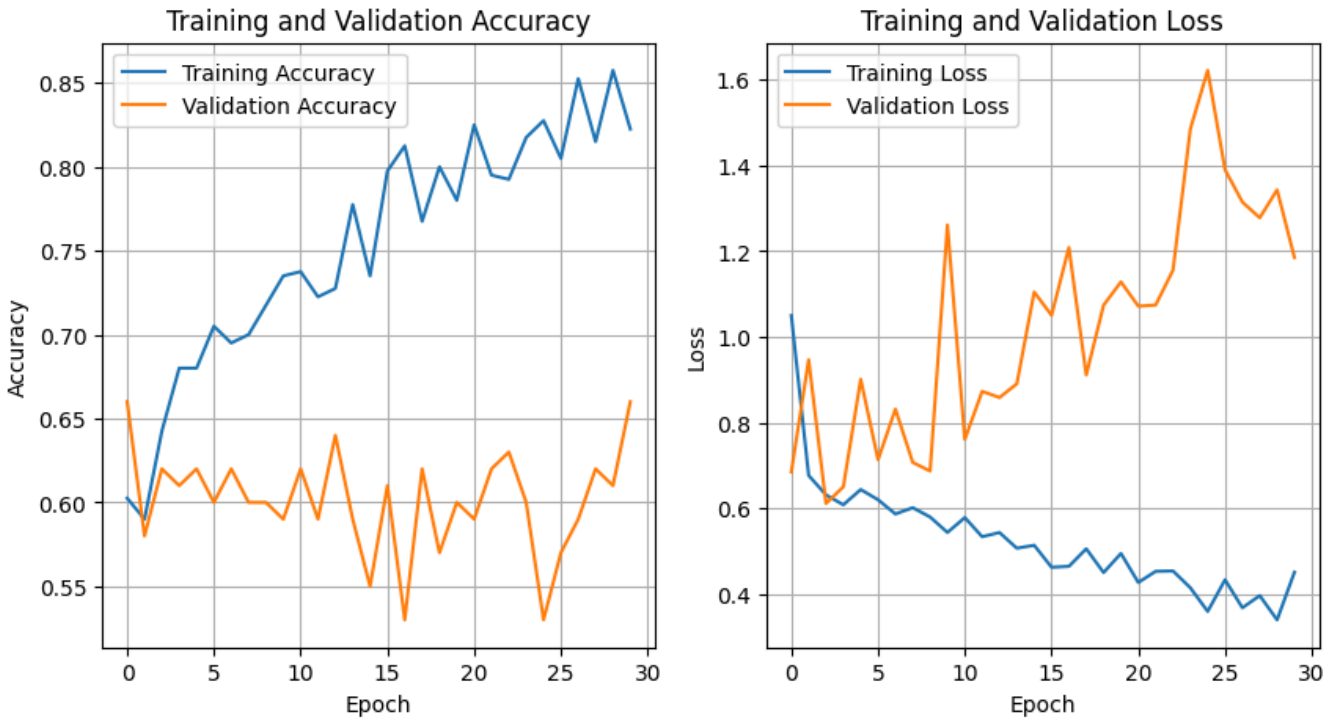
```
model_NAS.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])


model_checkpoint = ModelCheckpoint(
      filepath="pre_trained_NAS_with_data_augmentation.h5.keras",
      save_best_only=True,
      monitor="val_accuracy")

callbacks = [model_checkpoint]


history_NAS = model_NAS.fit(
    train_batches,
    epochs=30,
    validation_data=validation_batches,
    callbacks=callbacks)
```

```
Epoch 1/30
 2/80 ━━━━━━━━━━━━━━━━━━━━ 9s 120ms/step - accuracy: 0.4500 - loss: 3.8369   W
80/80 ━━━━━━━━━━━━━━━━━━━━ 0s 119ms/step - accuracy: 0.5785 - loss: 1.8258W000
80/80 ━━━━━━━━━━━━━━━━━━━━ 84s 427ms/step - accuracy: 0.5788 - loss: 1.8163 -
Epoch 2/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 140ms/step - accuracy: 0.5542 - loss: 0.7269 -
Epoch 3/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 137ms/step - accuracy: 0.6241 - loss: 0.6422 -
Epoch 4/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 137ms/step - accuracy: 0.7075 - loss: 0.5963 -
Epoch 5/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 140ms/step - accuracy: 0.6549 - loss: 0.6467 -
Epoch 6/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 137ms/step - accuracy: 0.7185 - loss: 0.6120 -
Epoch 7/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 137ms/step - accuracy: 0.6592 - loss: 0.6108 -
Epoch 8/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 140ms/step - accuracy: 0.7266 - loss: 0.5893 -
Epoch 9/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 137ms/step - accuracy: 0.7207 - loss: 0.5584 -
Epoch 10/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 140ms/step - accuracy: 0.7358 - loss: 0.5829 -
Epoch 11/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 137ms/step - accuracy: 0.7582 - loss: 0.5590 -
Epoch 12/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 141ms/step - accuracy: 0.7527 - loss: 0.5100 -
Epoch 13/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 139ms/step - accuracy: 0.7607 - loss: 0.4916 -
Epoch 14/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 137ms/step - accuracy: 0.7923 - loss: 0.4945 -
Epoch 15/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 136ms/step - accuracy: 0.7661 - loss: 0.4597 -
Epoch 16/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 138ms/step - accuracy: 0.8276 - loss: 0.4231 -
Epoch 17/30
```

```
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 138ms/step - accuracy: 0.8202 - loss: 0.4332 -
Epoch 18/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 140ms/step - accuracy: 0.7656 - loss: 0.5459 -
Epoch 19/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 137ms/step - accuracy: 0.8332 - loss: 0.4020 -
Epoch 20/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 137ms/step - accuracy: 0.7938 - loss: 0.4229 -
Epoch 21/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 138ms/step - accuracy: 0.8283 - loss: 0.4366 -
Epoch 22/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 137ms/step - accuracy: 0.8375 - loss: 0.3906 -
Epoch 23/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 21s 141ms/step - accuracy: 0.8141 - loss: 0.4232 -
Epoch 24/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 136ms/step - accuracy: 0.8140 - loss: 0.4142 -
Epoch 25/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 139ms/step - accuracy: 0.8318 - loss: 0.3634 -
Epoch 26/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 137ms/step - accuracy: 0.8081 - loss: 0.4072 -
Epoch 27/30
80/80 ━━━━━━━━━━━━━━━━━━━━ 12s 138ms/step - accuracy: 0.8485 - loss: 0.4088 -
Epoch 28/30
```

NASNetLarge - Plots

```
generate_plots(history_NAS)
```

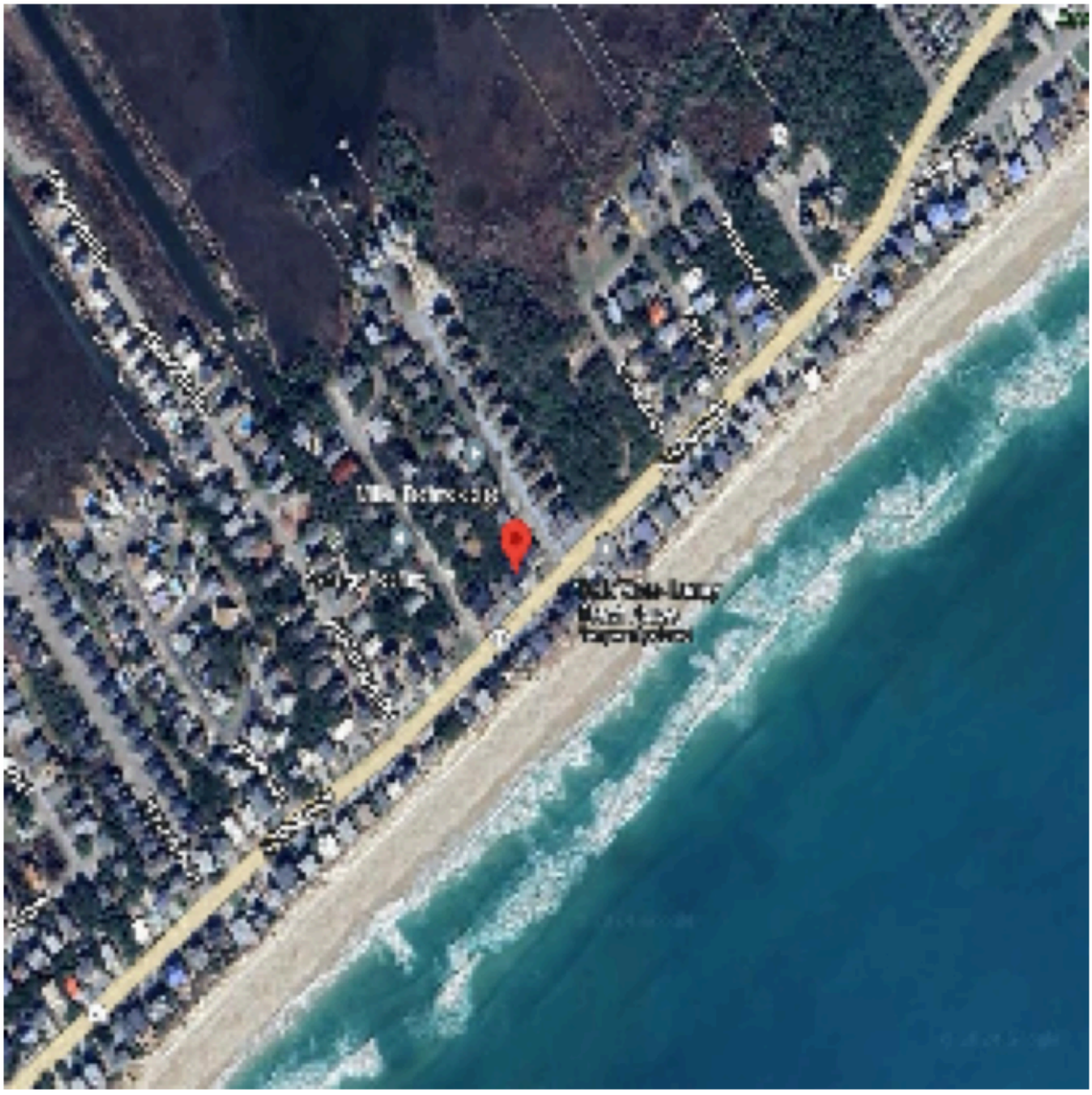


NASNetLarge - Sample Batch Predictions

```
x, y = validation_batches[0]

preds = model_NAS.predict(validation_batches)
pred_labels = np.where(preds < .5, 0, 1)

plt.figure(figsize=(10, 50))

for i in range(0,5):
    image = x[i]
    label = int(y[i])
    plt.subplot(5, 1, i+1)
    plt.imshow(image)
    plt.title(f'Predicted: {class_names[int(pred_labels[label])]} // Actual: {class_
    plt.axis('off')

plt.show()
```

```
 3/20 ━━━━━━━━━━━━━━━━━━━━ 1s 71ms/stepW0000 00:00:1723923281.808691    3866 g
20/20 ━━━━━━━━━━━━━━━━━━━━ 19s 70ms/step
/tmp/ipykernel_3778/1229623569.py:13: DeprecationWarning: Conversion of an arra
  plt.title(f'Predicted: {class_names[int(pred_labels[label])]} // Actual: {cla
```

Predicted: Adverse // Actual: Adverse



Predicted: Neutral // Actual: Neutral