Summer 2021: CSEE5590 –Special Topics (Extra Credit Project)

Image classification Project (Report)

Student Name: Yousef Almutairi

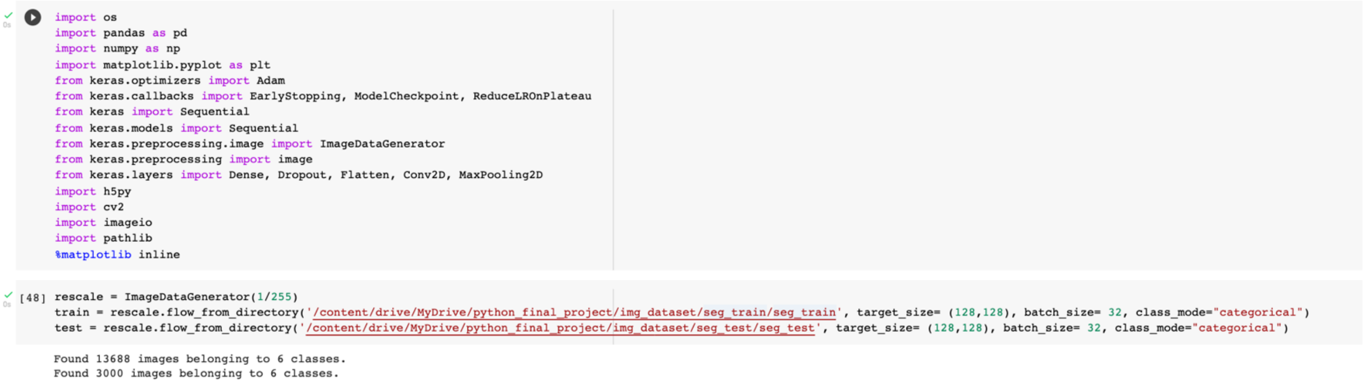
Student ID: 16305693

Project Overview:

Since we have worked on the Convolutional Neural Networks assignments (image classification in particular) during this semester, I have decided to move forward of doing my project on the same CNN model but with a different dataset. The dataset has been downloaded from Kaggle website. In addition and after classified all the images, I worked on the Streamlit framework for building a web-app for my saved model to predict the uploaded images.

Build the CNN model:

First of all, I have downloaded my dataset from Kaggle which contains around 25k images of size 150x150 distributed under 6 categories [buildings, forest, glacier, mountain, sea, street]. After that I rescaled the images based on their folder (train and test) before start processing the images. Also, I added the target_size= (128,128) parameter into it to prepare the size of the images for the model. As shown in the screenshot below I got 13688 images on the training and 3000 images on the testing folder, and all of them are belonging 6 classes.



```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
from keras import Sequential
from keras.models import Sequential
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
import h5py
import cv2
import imageio
import pathlib
%matplotlib inline
```

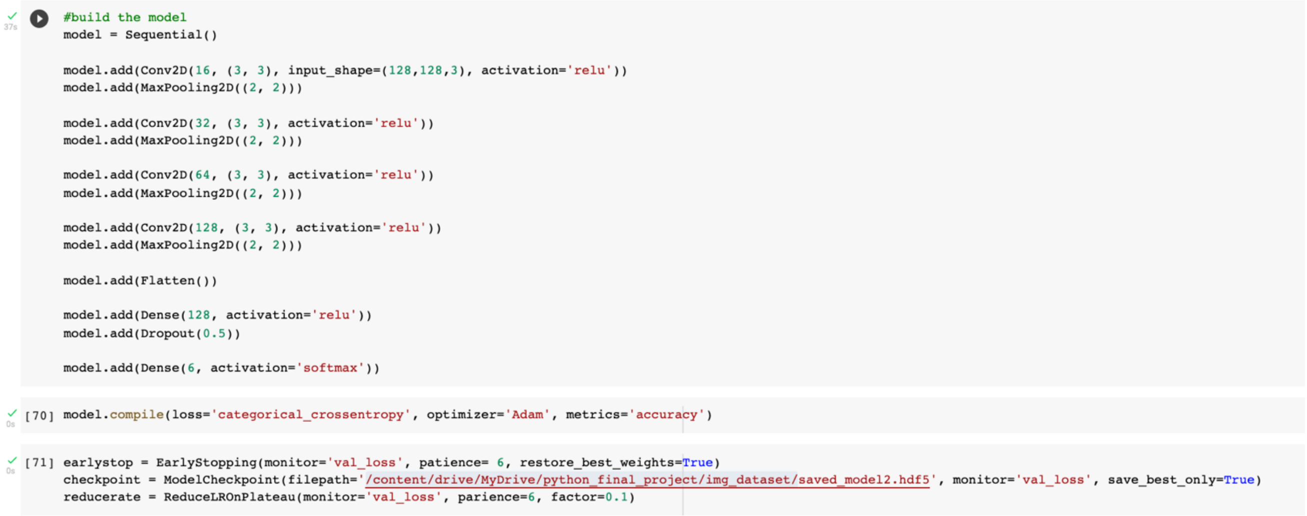
```
[48] rescale = ImageDataGenerator(1/255)
     train = rescale.flow_from_directory('/content/drive/MyDrive/python_final_project/img_dataset/seg_train/seg_train', target_size= (128,128), batch_size= 32, class_mode="categorical")
     test = rescale.flow_from_directory('/content/drive/MyDrive/python_final_project/img_dataset/seg_test/seg_test', target_size= (128,128), batch_size= 32, class_mode="categorical")
```

```
Found 13688 images belonging to 6 classes.
Found 3000 images belonging to 6 classes.
```

Then, I start building my CNN Sequential model with 4 convolution layers with a maxPooling, and different number of filters, started with 16 and end it with 256. In addition, I added the input_shape parameter to 128*128 as I've added on the target_size earlier. Regarding the MaxPooling2D, I gave it (2,2) for the pool_size in order to let the size of the future map to be reduced.

In this screenshot, I've created the EarlyStopping ,ModelCheckpoint, and ReduceLROnPlateau Callbacks to customize my model performance.

- EarlyStopping: since I do have 6 classes I gave the patience argument “6” which mean if the epochs are no longer improvement will stop as well as “restore_best_weights” to restore the weight of the model in the best value.
- ModelCheckpoint: The first parameter “filepath” is used to save the model/weights.
- ReduceLROnPlateau: This callback is used to reduce the learning rate.

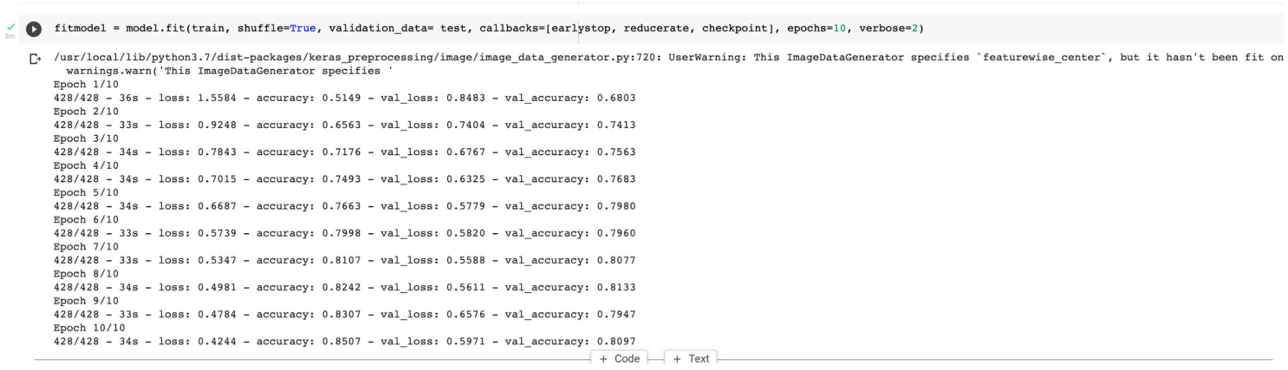


```
#build the model
model = Sequential()

model.add(Conv2D(16, (3, 3), input_shape=(128,128,3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(6, activation='softmax'))
```

```
[70] model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics='accuracy')
```

```
[71] earlystop = EarlyStopping(monitor='val_loss', patience= 6, restore_best_weights=True)
     checkpoint = ModelCheckpoint(filepath='/content/drive/MyDrive/python_final_project/img_dataset/saved_model2.hdf5', monitor='val_loss', save_best_only=True)
     reducerate = ReduceLROnPlateau(monitor='val_loss', parience=6, factor=0.1)
```

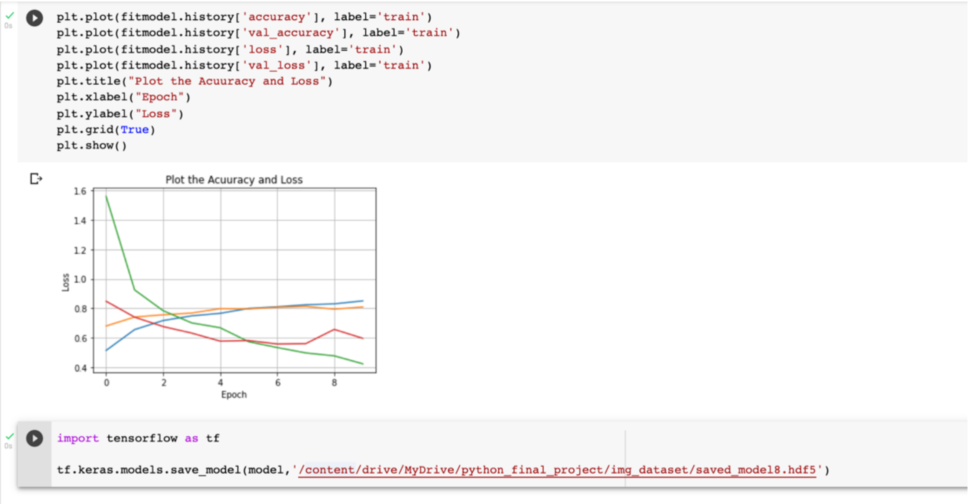
In this step, I compile and train the model



```
fitmodel = model.fit(train, shuffle=True, validation_data= test, callbacks=[earlystop, reducerate, checkpoint], epochs=10, verbose=2)
```

```
/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/image_data_generator.py:720: UserWarning: This ImageDataGenerator specifies `featurewise_center`, but it hasn't been fit on
  warnings.warn('This ImageDataGenerator specifies '
Epoch 1/10
428/428 - 36s - loss: 1.5584 - accuracy: 0.5149 - val_loss: 0.8483 - val_accuracy: 0.6803
Epoch 2/10
428/428 - 33s - loss: 0.9248 - accuracy: 0.6563 - val_loss: 0.7404 - val_accuracy: 0.7413
Epoch 3/10
428/428 - 34s - loss: 0.7843 - accuracy: 0.7176 - val_loss: 0.6767 - val_accuracy: 0.7563
Epoch 4/10
428/428 - 34s - loss: 0.7015 - accuracy: 0.7493 - val_loss: 0.6325 - val_accuracy: 0.7683
Epoch 5/10
428/428 - 34s - loss: 0.6687 - accuracy: 0.7663 - val_loss: 0.5779 - val_accuracy: 0.7980
Epoch 6/10
428/428 - 33s - loss: 0.5739 - accuracy: 0.7998 - val_loss: 0.5820 - val_accuracy: 0.7960
Epoch 7/10
428/428 - 33s - loss: 0.5347 - accuracy: 0.8107 - val_loss: 0.5588 - val_accuracy: 0.8077
Epoch 8/10
428/428 - 34s - loss: 0.4981 - accuracy: 0.8242 - val_loss: 0.5611 - val_accuracy: 0.8133
Epoch 9/10
428/428 - 33s - loss: 0.4784 - accuracy: 0.8307 - val_loss: 0.6576 - val_accuracy: 0.7947
Epoch 10/10
428/428 - 34s - loss: 0.4244 - accuracy: 0.8507 - val_loss: 0.5971 - val_accuracy: 0.8097
```

Plot the Accuracy and Loss and saved the model

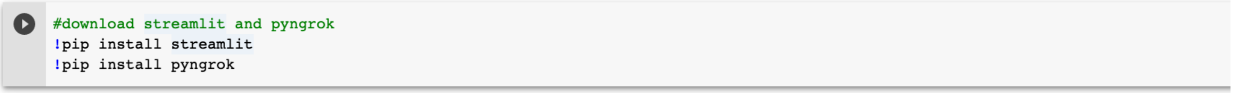


```
plt.plot(fitmodel.history['accuracy'], label='train')
plt.plot(fitmodel.history['val_accuracy'], label='train')
plt.plot(fitmodel.history['loss'], label='train')
plt.plot(fitmodel.history['val_loss'], label='train')
plt.title("Plot the Acuuracy and Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.grid(True)
plt.show()
```

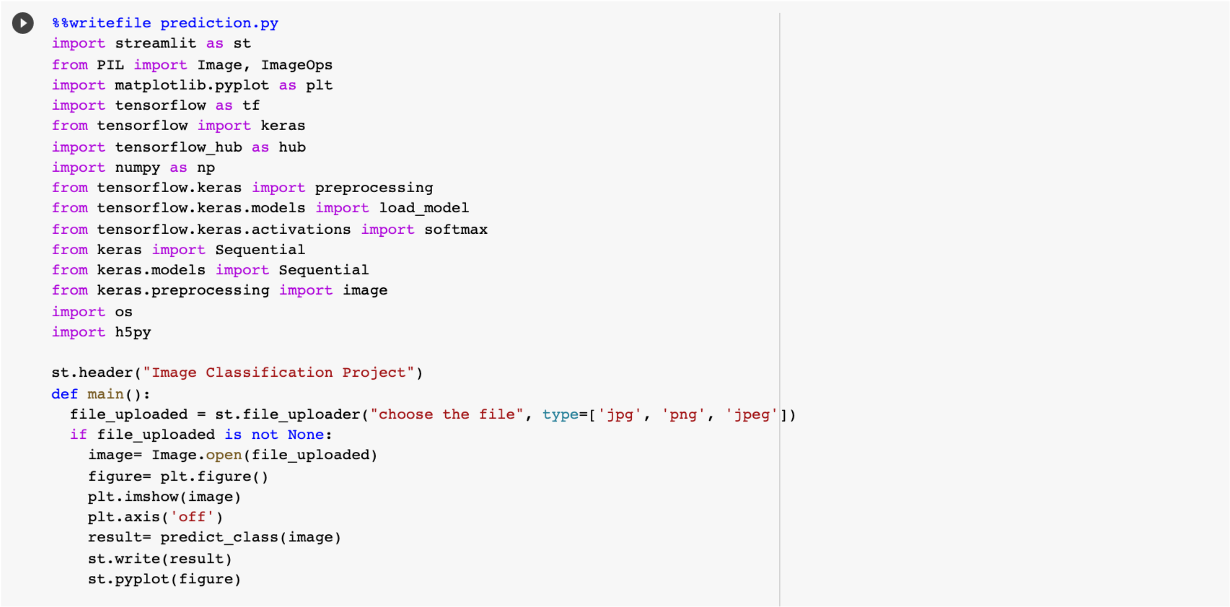
```
import tensorflow as tf

tf.keras.models.save_model(model,'/content/drive/MyDrive/python_final_project/img_dataset/saved_model8.hdf5')
```

Build the web-app framework:

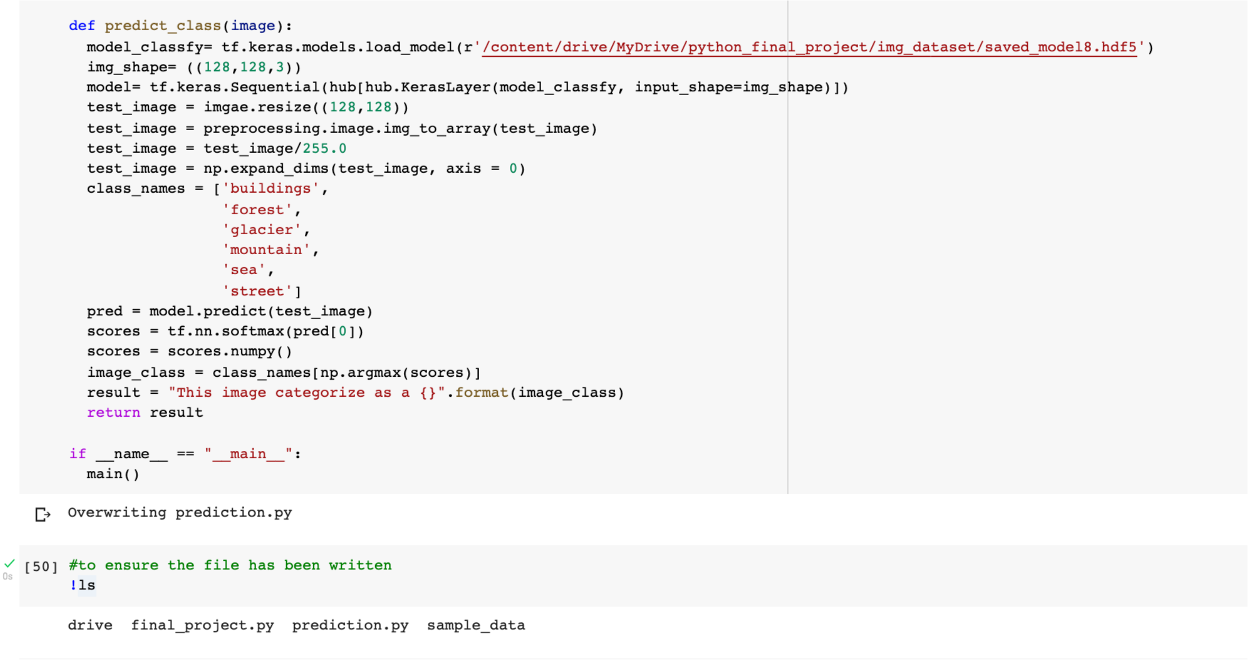
In this step, I used the Streamlit framework to create a web page for the model that I saved using the ngrok web server. In the following screenshot is showing that how I built the wep-page



```
#download streamlit and pyngrok
!pip install streamlit
!pip install pyngrok
```



```
%%writefile prediction.py
import streamlit as st
from PIL import Image, ImageOps
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
import tensorflow_hub as hub
import numpy as np
from tensorflow.keras import preprocessing
from tensorflow.keras.models import load_model
from tensorflow.keras.activations import softmax
from keras import Sequential
from keras.models import Sequential
from keras.preprocessing import image
import os
import h5py

st.header("Image Classification Project")
def main():
  file_uploaded = st.file_uploader("choose the file", type=['jpg', 'png', 'jpeg'])
  if file_uploaded is not None:
    image= Image.open(file_uploaded)
    figure= plt.figure()
    plt.imshow(image)
    plt.axis('off')
    result= predict_class(image)
    st.write(result)
    st.pyplot(figure)
```



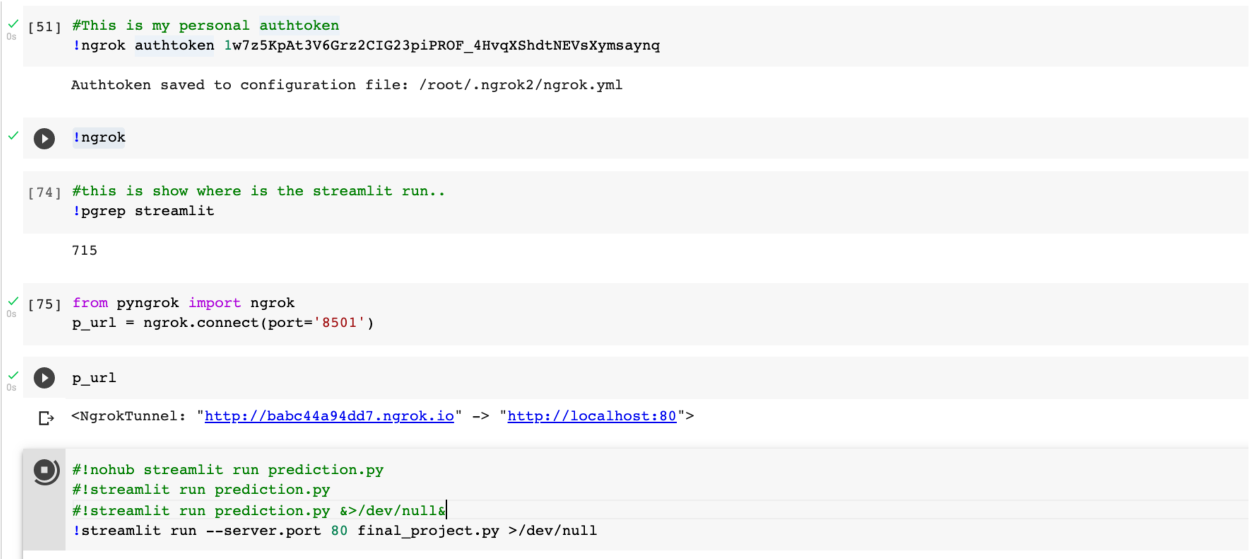
```
def predict_class(image):
  model_classfy= tf.keras.models.load_model(r'/content/drive/MyDrive/python_final_project/img_dataset/saved_model8.hdf5')
  img_shape= ((128,128,3))
  model= tf.keras.Sequential(hub[hub.KerasLayer(model_classfy, input_shape=img_shape)])
  test_image = imgae.resize((128,128))
  test_image = preprocessing.image.img_to_array(test_image)
  test_image = test_image/255.0
  test_image = np.expand_dims(test_image, axis = 0)
  class_names = ['buildings',
                 'forest',
                 'glacier',
                 'mountain',
                 'sea',
                 'street']
  pred = model.predict(test_image)
  scores = tf.nn.softmax(pred[0])
  scores = scores.numpy()
  image_class = class_names[np.argmax(scores)]
  result = "This image categorize as a {}".format(image_class)
  return result

if __name__ == "__main__":
  main()
```

```
Overwriting prediction.py
```

```
[50] #to ensure the file has been written
     !ls
```

```
drive   final_project.py   prediction.py   sample_data
```

After creating the streamlit framework and connected with the model, I start working on the ngrok command to ensure the web page is running successfully.



```
[51] #This is my personal authtoken
     !ngrok authtoken 1w7z5KpAt3V6Grz2CIG23piPROF_4HvqXShdtNEVsXymsaynq
```

```
Authtoken saved to configuration file: /root/.ngrok2/ngrok.yml
```

```
!ngrok
```

```
[74] #this is show where is the streamlit run..
     !pgrep streamlit
```

```
715
```

```
[75] from pyngrok import ngrok
     p_url = ngrok.connect(port='8501')
```

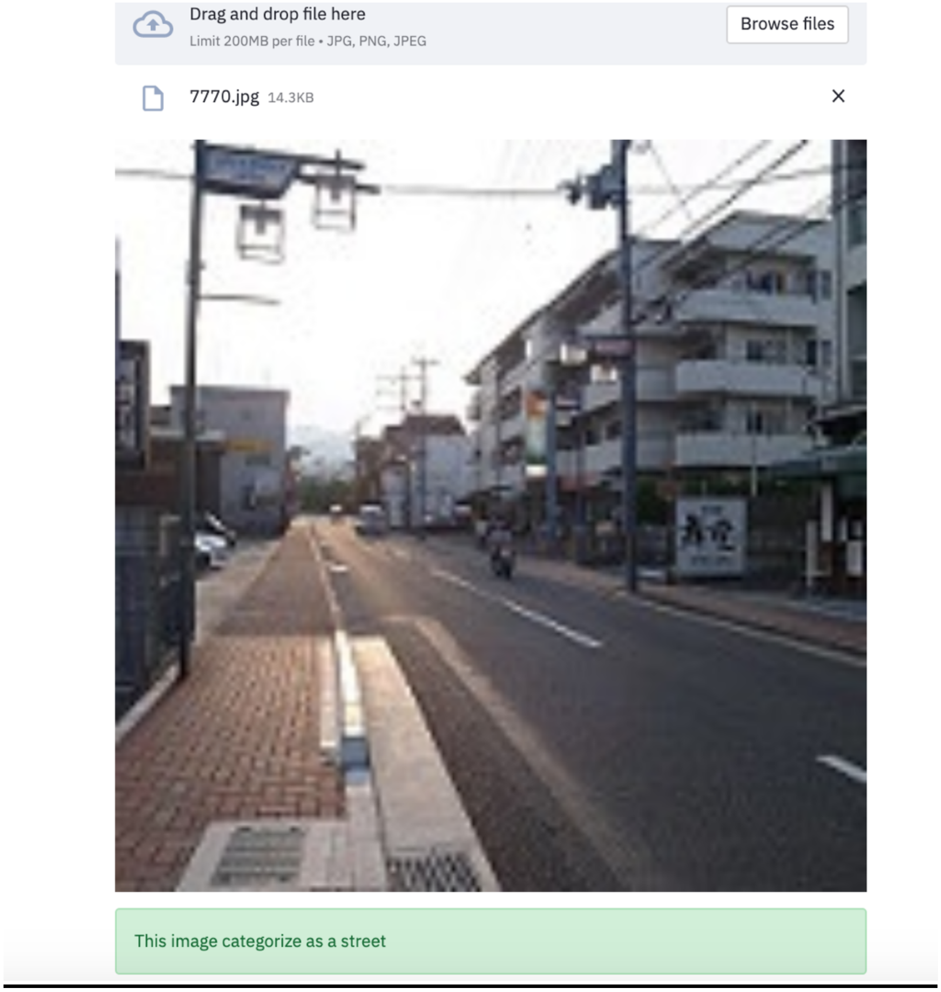
```
p_url
```

```
<NgrokTunnel: "http://babc44a94dd7.ngrok.io" -> "http://localhost:80">
```

```
#!nohub streamlit run prediction.py
#!streamlit run prediction.py
#!streamlit run prediction.py &>/dev/null&
!streamlit run --server.port 80 final_project.py >/dev/null
```

Result:

I have successfully implemented the project with no issues. Also, I've recorded a video while I'm uploading the image and get the result. (Video Link.)



Drag and drop file here

Limit 200MB per file • JPG, PNG, JPEG

7770.jpg 14.3KB

This image categorize as a street

References:

https://www.kaggle.com/puneet6060/intel-image-classification

https://medium.com/analytics-vidhya/image-classification-with-django-deployment-d18dfc224270

https://www.analyticsvidhya.com/blog/2020/10/create-image-classification-model-python-keras/

https://www.youtube.com/watch?v=VVXkbXPFzmM&ab_channel=WhenMathsMeetCoding

https://www.analyticsvidhya.com/blog/2021/01/image-classification-using-convolutional-neural-networks-a-step-by-step-guide/

https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

https://keras.io/api/preprocessing/image/

https://heartbeat.fritz.ai/a-beginners-guide-to-convolutional-neural-networks-cnn-cf26c5ee17ed

https://docs.streamlit.io/en/stable/main_concepts.html

https://medium.com/@jcharistech/how-to-run-streamlit-apps-from-colab-29b969a1bdfc

https://dashboard.ngrok.com/get-started/setup

https://www.tensorflow.org/api_docs/python/tf/keras/models/load_model

https://numpy.org/doc/stable/reference/generated/numpy.expand_dims.html