Image Recognition for 10 Boston Skyscrapers

A Tensorflow model that can recognize images of 10 skyscrapers with more than 60% accuracy





Problem statement

The problem statement for my project is: When you look at a tall building in a city, how can you figure out the name of that building?

Solution

With Image recognition for buildings you can take a picture and it will tell you what building it is. I used Tensorflow to build a machine learning model for 10 buildings and then ran the model against a new picture to identify the building. For each building, I used an average of 150 pictures to build the model.

The model was able to identify pictures with > 60% accuracy.

Tensorflow

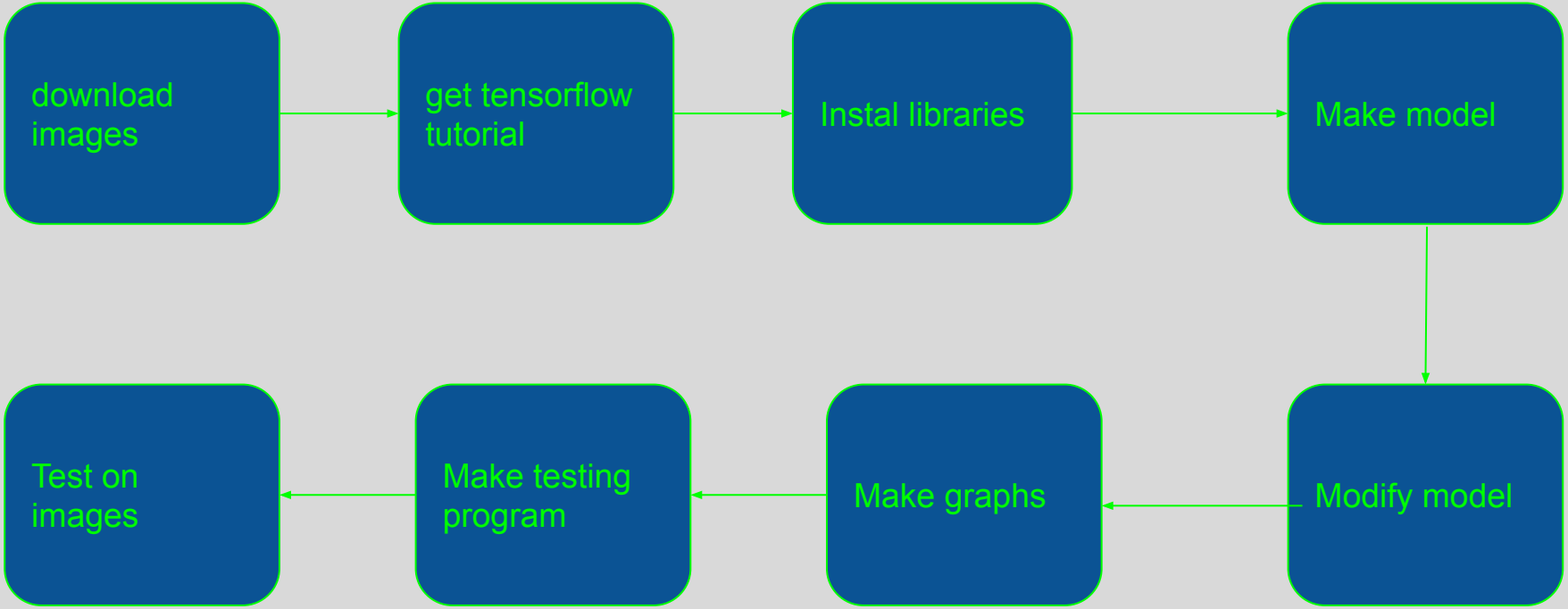
Tensorflow is a platform for machine learning made by Google. It comes in many computer languages.

Link: https://www.tensorflow.org/

Why I picked this project

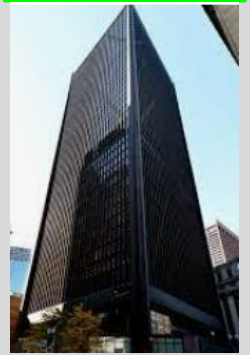
When I moved to Boston I would look at the buildings and wonder what the name was, by the time I tried to see the name of the building and a map, the car would have passed the building. So, I wanted to solve this problem.

Procedures


download images
get tensorflow tutorial
Instal libraries
Make model
Modify model
Make graphs
Make testing program
Test on images

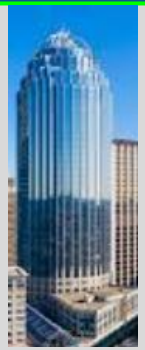
Download images step

1 Boston Place



Federal Reserve Bank

111 Huntington Ave

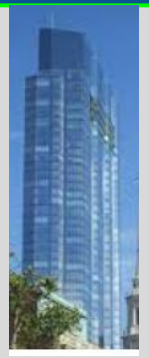


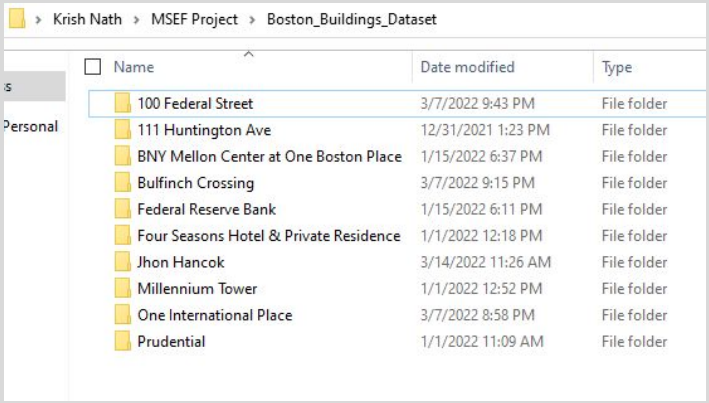
Bulfinch crossing

1 Dalton

1 International Place

Millenium Tower

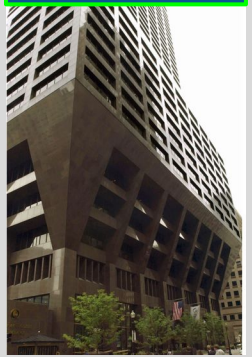




Average number of images per directory are 150

Used Chrome extension called Imagi that helps download multiple images in 1 go

100 Federal Street



John Hancock

Prudential

Get tensorflow tutorial step

I started to look at tutorials and the one I liked the most was https://www.tensorflow.org/tutorials/images/classification#compile_and_train_the_model because it was from the tensorflow and it had explanations of what the functions do and why they are using it.

Installing libraries step

IDE: Spyder

```
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
from PIL import Image
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
import pathlib
```

Make model step

My program split the images into training and validation to ensure model accuracy

```
batch_size = 32
img_height = 500
img_width = 500
train_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
val_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

Modify model step

```
AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
num_classes = len(class_names)
data_augmentation = keras.Sequential(
  [
    layers.experimental.preprocessing.RandomFlip("horizontal",
                      input_shape=(img_height,
                                   img_width,
                                   3)),
    layers.experimental.preprocessing.RandomRotation(0),
    layers.experimental.preprocessing.RandomZoom(0.1)])
model = Sequential([
  data_augmentation,
  layers.experimental.preprocessing.Rescaling(1./255,
  input_shape=(img_height, img_width, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.1505),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

This modifies the model so it can have better accuracy then a regular model.

Make graph

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
model.summary()
epochs=24
#history = model.fit(train_ds, validation_data=train_ds, epochs=epochs)
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
path=(r"C:\Users\Krish Nath\MSEF Project" +"\ mnh").replace(" mnh","")
model.save(path)
```

This is the part of the program that plots the training and validation accuracy for 24 epochs.

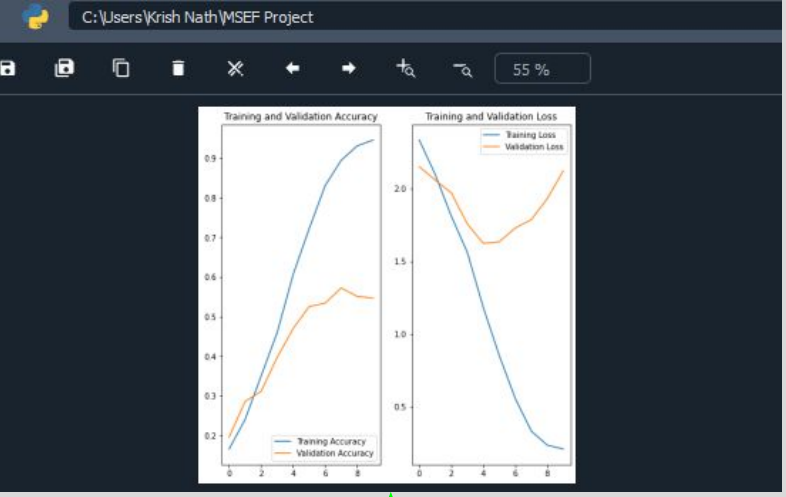
Make testing program

This is the program that tests images against the model

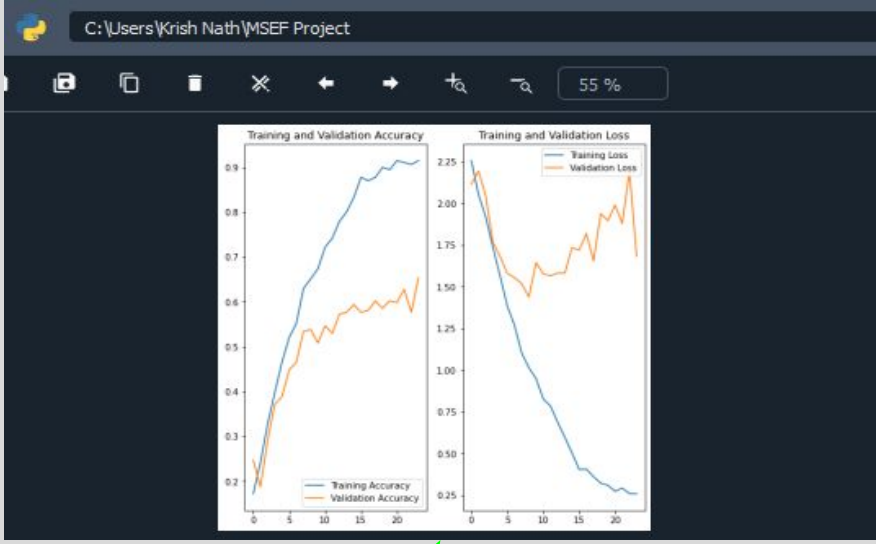
```
import PIL
from PIL import Image
import numpy as np
import tensorflow as tf
test_img=(Image.open(r"C:\Users\Krish Nath\MSEF Project\test_image_6.jpg")).resize((500,500))
model = tf.keras.models.load_model((r"C:\Users\Krish Nath\MSEF Project" +"\ mnh").replace(" mnh","")
class_names=['100 Federal Street', '111 Huntington Ave', 'BNY Mellon Center at One Boston Place',
'Bulfinch Crossing', 'Federal Reserve Bank', 'Four Seasons Hotel & Private Residence',
'Jhon Hancok', 'Millennium Tower', 'One International Place', 'Prudential']
img_array = tf.keras.utils.img_to_array(test_img)
img_array = tf.expand_dims(img_array, 0)
predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

Plots



This plot shows that the model is overfitted because validation has 50% accuracy, and the training has almost 100% accuracy.



This plot shows that the modified model is not too much overfitted because validation has 65% accuracy, and the training has a little more than 90% accuracy.

Running the model against an image

John Hancock image

```
In [4]: runfile('C:/Users/Krish Nath/MSEF Project/test_model.py', wdir='C:/Users/Krish Nath/
MSEF Project')
This image most likely belongs to Jhon Hancok with a 79.06 percent confidence.
```

Prudential image

```
In [3]: runfile('C:/Users/Krish Nath/MSEF Project/test_model.py', wdir='C:/Users/Krish Nath/
MSEF Project')
This image most likely belongs to Prudential with a 96.79 percent confidence.
```

Results

As you look at the plots, you can see that the model is not so much overfitted and so it is able to recognize images.

Next steps

I should work on making the accuracy even better, then future studies are required to expand this concept to develop a model that can recognize any skyscraper in the world. Once developed, this mechanism can be used in various applications liking pointing a phone camera towards a building, and getting the name of the building and other details on the screen.

About me

My name is Krish Nath, I live in Boston, I am a 6th grader in Boston Trinity Academy, and I am very interested in programming.

I hope you enjoyed reading my slides!