Image Recognition with IBM Cloud Visual Recognition

Phase 4: Development Part 2

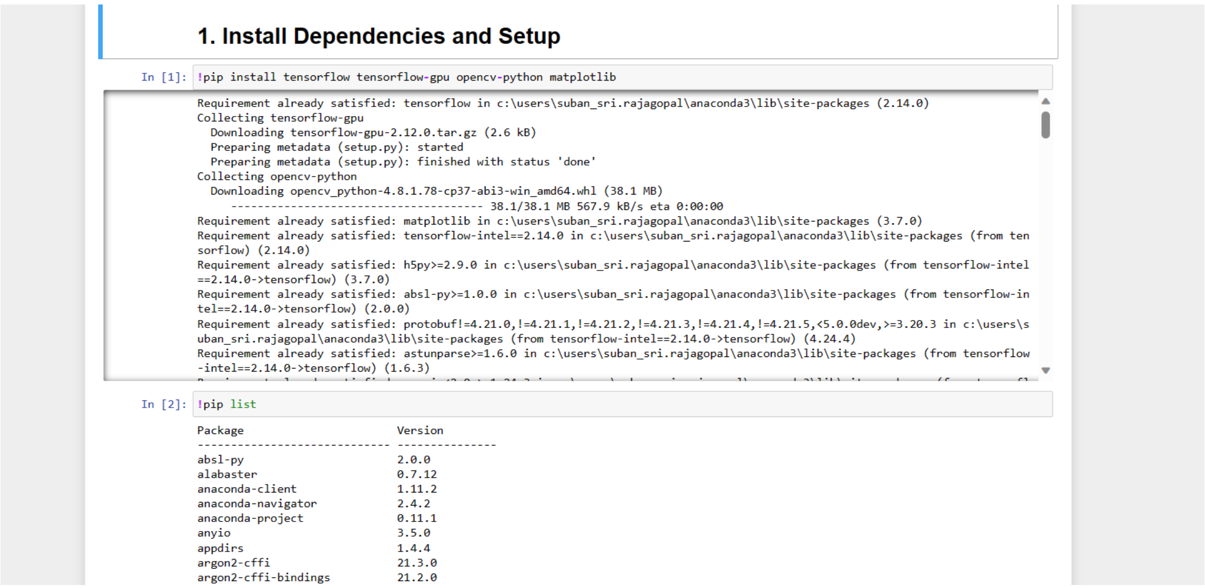
OVERVIEW:

The following actions must be taken in order to construct an image recognition system that integrates AI-generated captions with IBM Cloud Visual Recognition:

Become an IBM Cloud user: Make sure you have an IBM Cloud account if you don't have. Establish a Visual Recognition Service: Launch an IBM Watson Visual Recognition service instance on IBM Cloud. You can use this service to analyze photos and find items in them. Obtain API Credentials: Once the Visual Recognition service has been created, get your API Key and URL. To authenticate and gain access to the service, you will require these credentials. Create the Image Classification Component: To connect to the Visual Recognition service, use Python and the IBM Watson SDK. This element examines the photos and provides a list of things it has identified.

Integrate Natural Language Generation (NLG): To generate captions for the objects that have been identified, utilize a library for natural language generation. For every object, you can create a coherent statement that describes it. Using the generated captions and the results of the image categorization, show the captions on a web page or user interface.

CODE FOR IMAGE RECOGNITION:



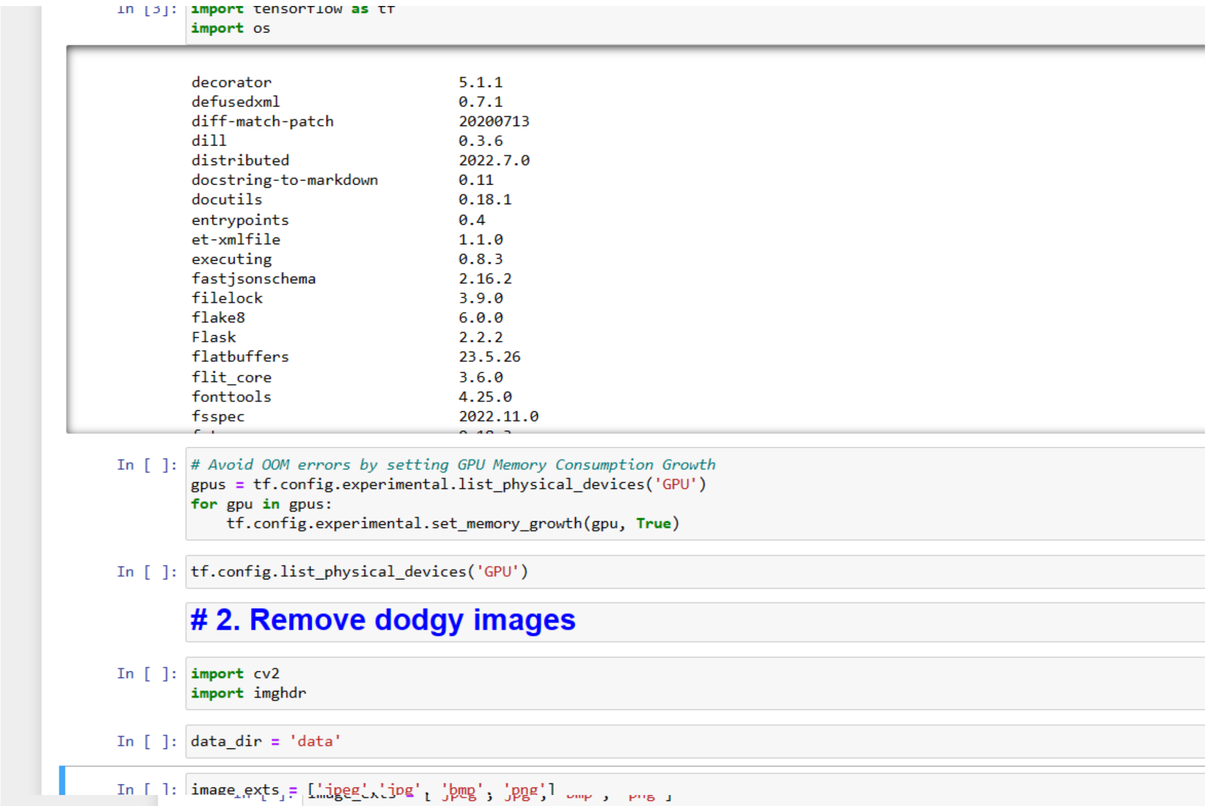
1. Install Dependencies and Setup

```
In [1]: !pip install tensorflow tensorflow-gpu opencv-python matplotlib
```

```
Requirement already satisfied: tensorflow in c:\users\suban_sri.rajagopal\anaconda3\lib\site-packages (2.14.0)
Collecting tensorflow-gpu
  Downloading tensorflow-gpu-2.12.0.tar.gz (2.6 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting opencv-python
  Downloading opencv_python-4.8.1.78-cp37-abi3-win_amd64.whl (38.1 MB)
     -------------------------------------- 38.1/38.1 MB 567.9 kB/s eta 0:00:00
Requirement already satisfied: matplotlib in c:\users\suban_sri.rajagopal\anaconda3\lib\site-packages (3.7.0)
Requirement already satisfied: tensorflow-intel==2.14.0 in c:\users\suban_sri.rajagopal\anaconda3\lib\site-packages (from ten
sorflow) (2.14.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\suban_sri.rajagopal\anaconda3\lib\site-packages (from tensorflow-intel
==2.14.0->tensorflow) (3.7.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\suban_sri.rajagopal\anaconda3\lib\site-packages (from tensorflow-in
tel==2.14.0->tensorflow) (2.0.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\users\s
uban_sri.rajagopal\anaconda3\lib\site-packages (from tensorflow-intel==2.14.0->tensorflow) (4.24.4)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\suban_sri.rajagopal\anaconda3\lib\site-packages (from tensorflow
-intel==2.14.0->tensorflow) (1.6.3)
```

```
In [2]: !pip list
```

```
Package                       Version
----------------------------- ---------------
absl-py                       2.0.0
alabaster                     0.7.12
anaconda-client               1.11.2
anaconda-navigator            2.4.2
anaconda-project              0.11.1
anyio                         3.5.0
appdirs                       1.4.4
argon2-cffi                   21.3.0
argon2-cffi-bindings          21.2.0
```



```
In [3]: import tensorflow as tf
        import os
```

```
decorator                          5.1.1
defusedxml                         0.7.1
diff-match-patch                   20200713
dill                               0.3.6
distributed                        2022.7.0
docstring-to-markdown              0.11
docutils                           0.18.1
entrypoints                        0.4
et-xmlfile                         1.1.0
executing                          0.8.3
fastjsonschema                     2.16.2
filelock                           3.9.0
flake8                             6.0.0
Flask                              2.2.2
flatbuffers                        23.5.26
flit_core                          3.6.0
fonttools                          4.25.0
fsspec                             2022.11.0
```

```
In [ ]: # Avoid OOM errors by setting GPU Memory Consumption Growth
        gpus = tf.config.experimental.list_physical_devices('GPU')
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
```

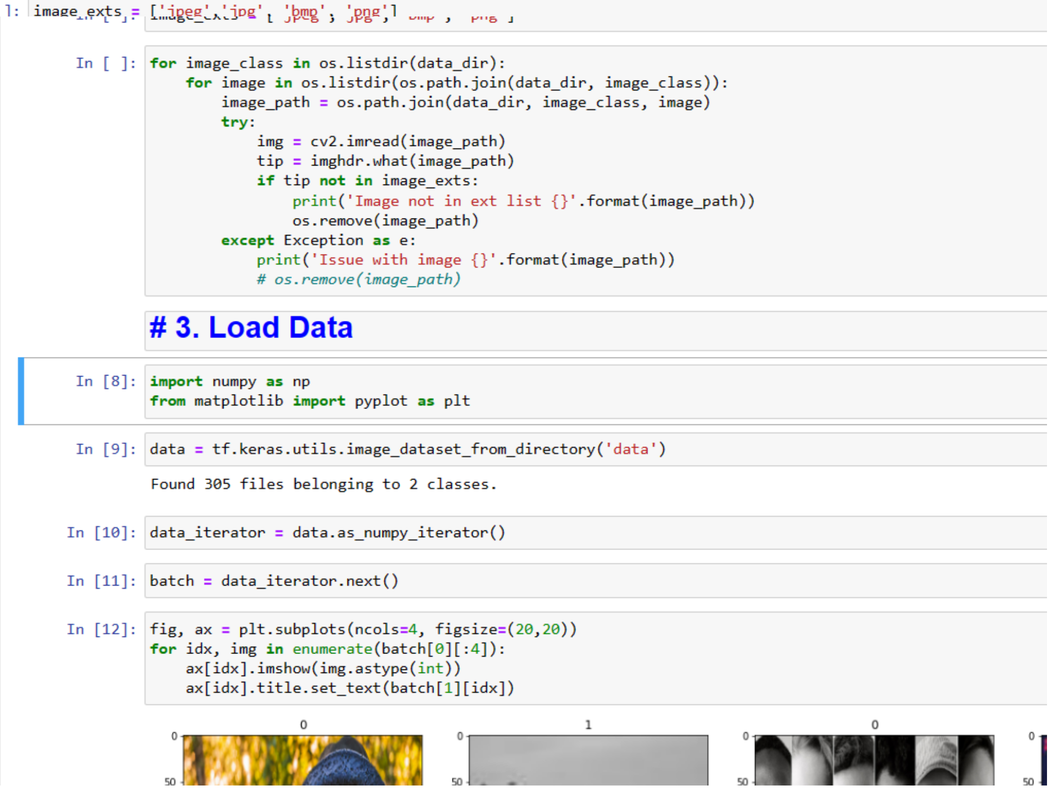
```
In [ ]: tf.config.list_physical_devices('GPU')
```

2. Remove dodgy images

```
In [ ]: import cv2
        import imghdr
```

```
In [ ]: data_dir = 'data'
```

```
In [ ]: image_exts = ['jpeg','jpg', 'bmp', 'png']
```



```
In [ ]: for image_class in os.listdir(data_dir):
            for image in os.listdir(os.path.join(data_dir, image_class)):
                image_path = os.path.join(data_dir, image_class, image)
                try:
                    img = cv2.imread(image_path)
                    tip = imghdr.what(image_path)
                    if tip not in image_exts:
                        print('Image not in ext list {}'.format(image_path))
                        os.remove(image_path)
                except Exception as e:
                    print('Issue with image {}'.format(image_path))
                    # os.remove(image_path)
```

3. Load Data

```
In [8]: import numpy as np
        from matplotlib import pyplot as plt
```

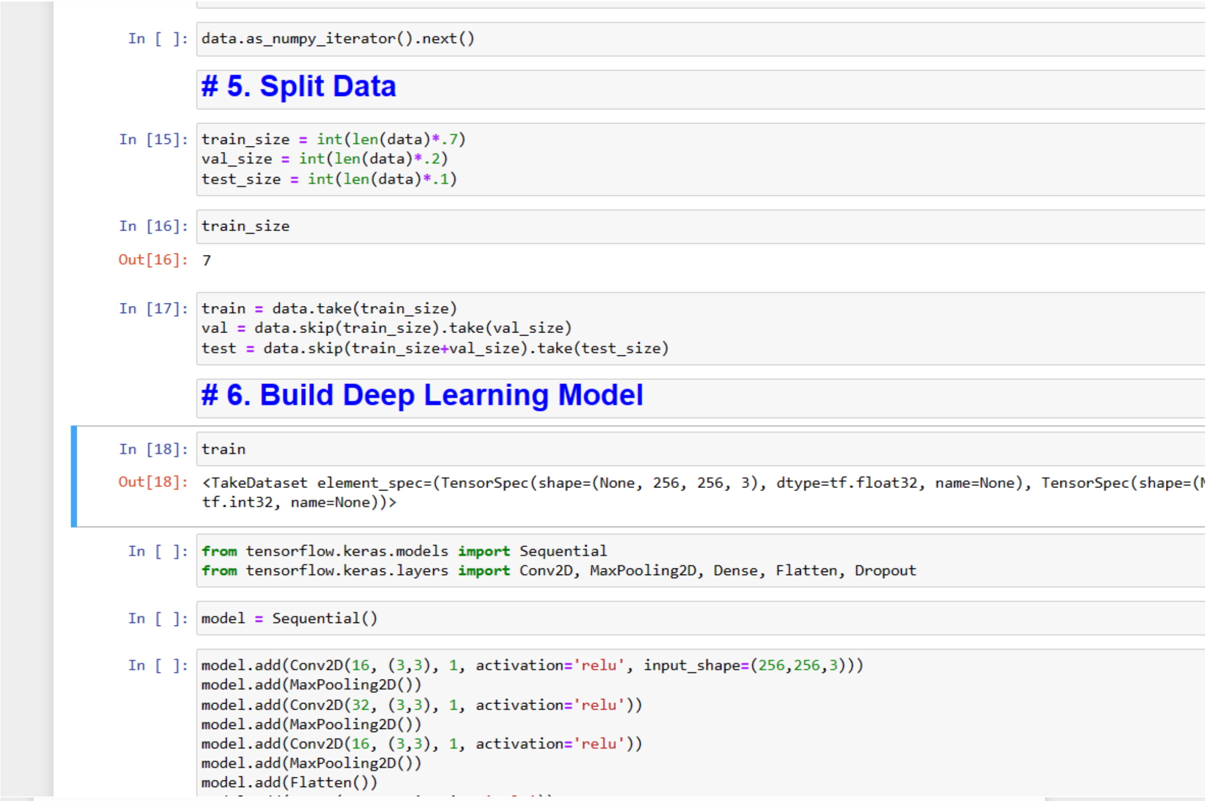
```
In [9]: data = tf.keras.utils.image_dataset_from_directory('data')
```

```
Found 305 files belonging to 2 classes.
```

```
In [10]: data_iterator = data.as_numpy_iterator()
```

```
In [11]: batch = data_iterator.next()
```

```
In [12]: fig, ax = plt.subplots(ncols=4, figsize=(20,20))
         for idx, img in enumerate(batch[0][:4]):
             ax[idx].imshow(img.astype(int))
             ax[idx].title.set_text(batch[1][idx])
```



```
In [ ]: data.as_numpy_iterator().next()
```

5. Split Data

```
In [15]: train_size = int(len(data)*.7)
         val_size = int(len(data)*.2)
         test_size = int(len(data)*.1)
```

```
In [16]: train_size

Out[16]: 7
```

```
In [17]: train = data.take(train_size)
         val = data.skip(train_size).take(val_size)
         test = data.skip(train_size+val_size).take(test_size)
```

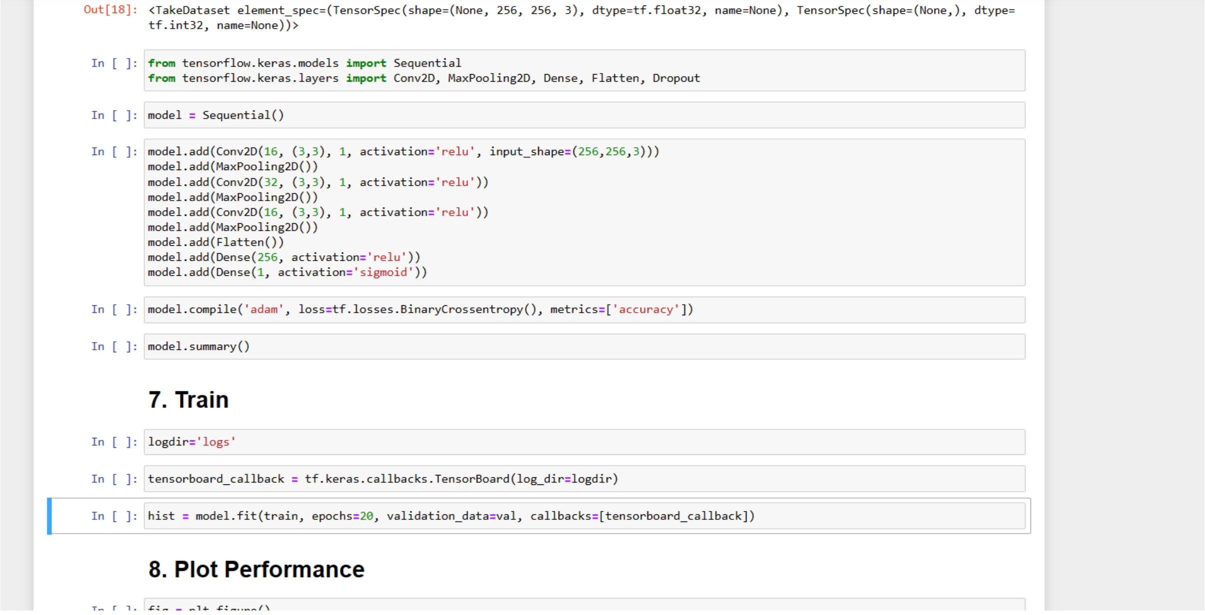
6. Build Deep Learning Model

```
In [18]: train

Out[18]: <TakeDataset element_spec=(TensorSpec(shape=(None, 256, 256, 3), dtype=tf.float32, name=None), TensorSpec(shape=(N
         tf.int32, name=None))>
```

```
In [ ]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
```

```
In [ ]: model = Sequential()
```

```
In [ ]: model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(256,256,3)))
        model.add(MaxPooling2D())
        model.add(Conv2D(32, (3,3), 1, activation='relu'))
        model.add(MaxPooling2D())
        model.add(Conv2D(16, (3,3), 1, activation='relu'))
        model.add(MaxPooling2D())
        model.add(Flatten())
```



```
Out[18]: <TakeDataset element_spec=(TensorSpec(shape=(None, 256, 256, 3), dtype=tf.float32, name=None), TensorSpec(shape=(None,), dtype=
         tf.int32, name=None))>
```

```
In [ ]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
```

```
In [ ]: model = Sequential()
```

```
In [ ]: model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(256,256,3)))
        model.add(MaxPooling2D())
        model.add(Conv2D(32, (3,3), 1, activation='relu'))
        model.add(MaxPooling2D())
        model.add(Conv2D(16, (3,3), 1, activation='relu'))
        model.add(MaxPooling2D())
        model.add(Flatten())
        model.add(Dense(256, activation='relu'))
        model.add(Dense(1, activation='sigmoid'))
```

```
In [ ]: model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])
```

```
In [ ]: model.summary()
```

7. Train

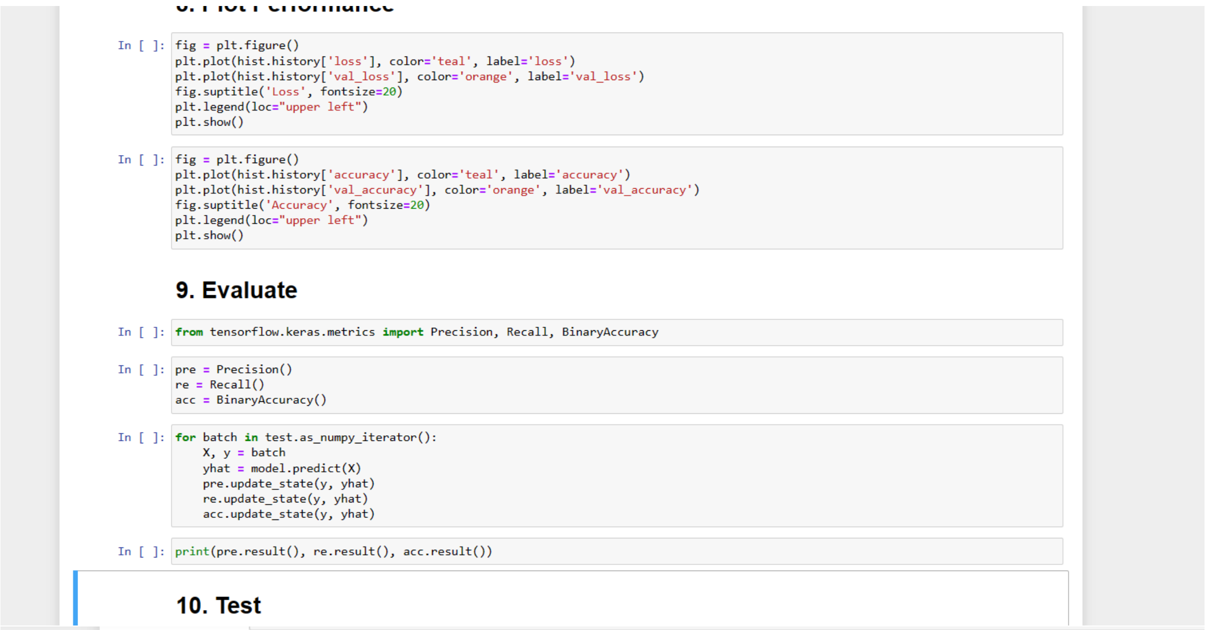
```
In [ ]: logdir='logs'
```

```
In [ ]: tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
```

```
In [ ]: hist = model.fit(train, epochs=20, validation_data=val, callbacks=[tensorboard_callback])
```

8. Plot Performance

```
In [ ]: fig = plt.figure()
```



8. Plot Performance

In [ ]:
```
fig = plt.figure()
plt.plot(hist.history['loss'], color='teal', label='loss')
plt.plot(hist.history['val_loss'], color='orange', label='val_loss')
fig.suptitle('Loss', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```

In [ ]:
```
fig = plt.figure()
plt.plot(hist.history['accuracy'], color='teal', label='accuracy')
plt.plot(hist.history['val_accuracy'], color='orange', label='val_accuracy')
fig.suptitle('Accuracy', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```

9. Evaluate

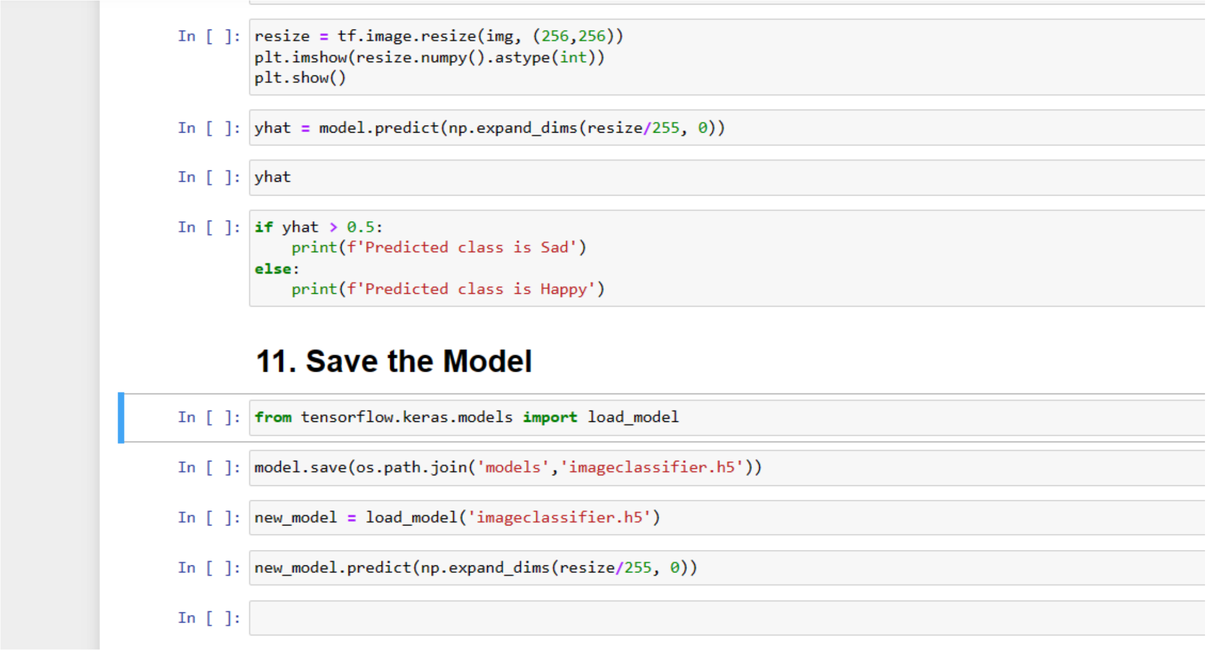
In [ ]:
```
from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy
```

In [ ]:
```
pre = Precision()
re = Recall()
acc = BinaryAccuracy()
```

In [ ]:
```
for batch in test.as_numpy_iterator():
    X, y = batch
    yhat = model.predict(X)
    pre.update_state(y, yhat)
    re.update_state(y, yhat)
    acc.update_state(y, yhat)
```

In [ ]:
```
print(pre.result(), re.result(), acc.result())
```

10. Test



In [ ]:
```
resize = tf.image.resize(img, (256,256))
plt.imshow(resize.numpy().astype(int))
plt.show()
```

In [ ]:
```
yhat = model.predict(np.expand_dims(resize/255, 0))
```

In [ ]:
```
yhat
```

In [ ]:
```
if yhat > 0.5:
    print(f'Predicted class is Sad')
else:
    print(f'Predicted class is Happy')
```

11. Save the Model

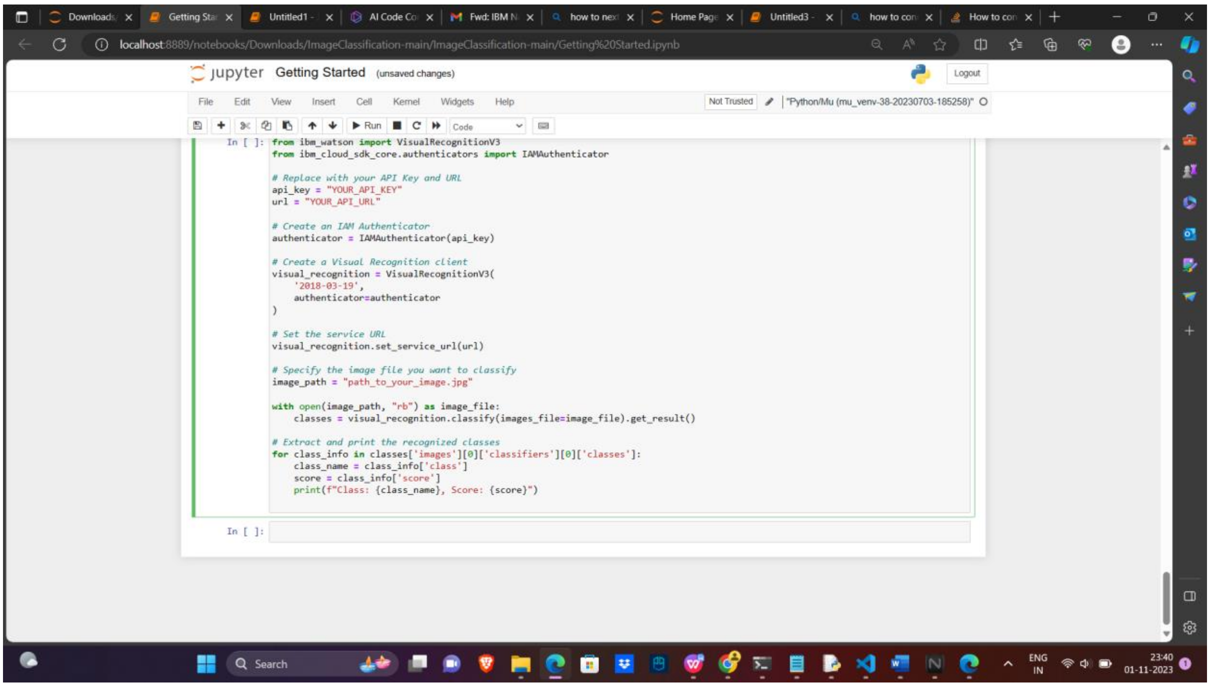
In [ ]:
```
from tensorflow.keras.models import load_model
```

In [ ]:
```
model.save(os.path.join('models','imageclassifier.h5'))
```

In [ ]:
```
new_model = load_model('imageclassifier.h5')
```

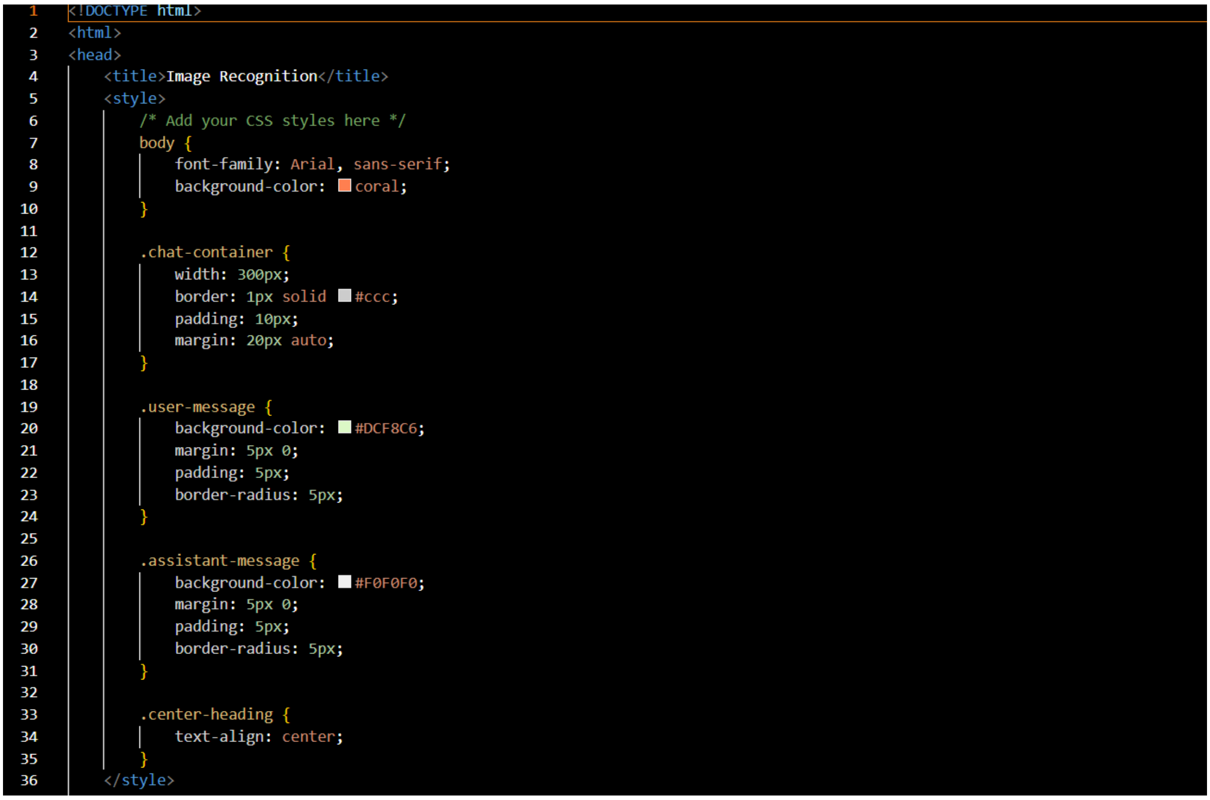
In [ ]:
```
new_model.predict(np.expand_dims(resize/255, 0))
```

In [ ]:

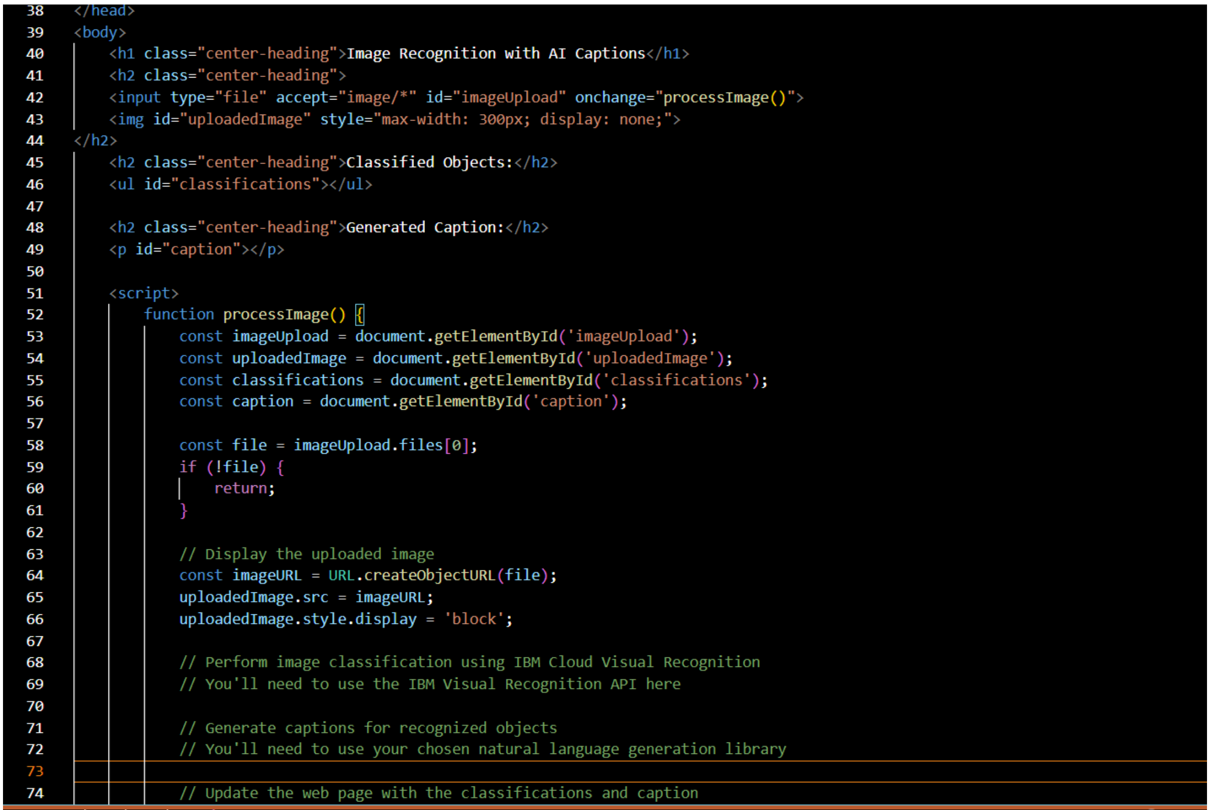

localhost:8889/notebooks/Downloads/ImageClassification-main/ImageClassification-main/Getting%20Started.ipynb
jupyter
Getting Started
(unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
"Python/Mu (mu_venv-38-20230703-185258)"
Run
Code
In [ ]:
Search
ENG IN
23:40
01-11-2023

```python
from ibm_watson import VisualRecognitionV3
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator

# Replace with your API Key and URL
api_key = "YOUR_API_KEY"
url = "YOUR_API_URL"

# Create an IAM Authenticator
authenticator = IAMAuthenticator(api_key)

# Create a Visual Recognition client
visual_recognition = VisualRecognitionV3(
    '2018-03-19',
    authenticator=authenticator
)

# Set the service URL
visual_recognition.set_service_url(url)

# Specify the image file you want to classify
image_path = "path_to_your_image.jpg"

with open(image_path, "rb") as image_file:
    classes = visual_recognition.classify(images_file=image_file).get_result()

# Extract and print the recognized classes
for class_info in classes['images'][0]['classifiers'][0]['classes']:
    class_name = class_info['class']
    score = class_info['score']
    print(f"Class: {class_name}, Score: {score}")
```

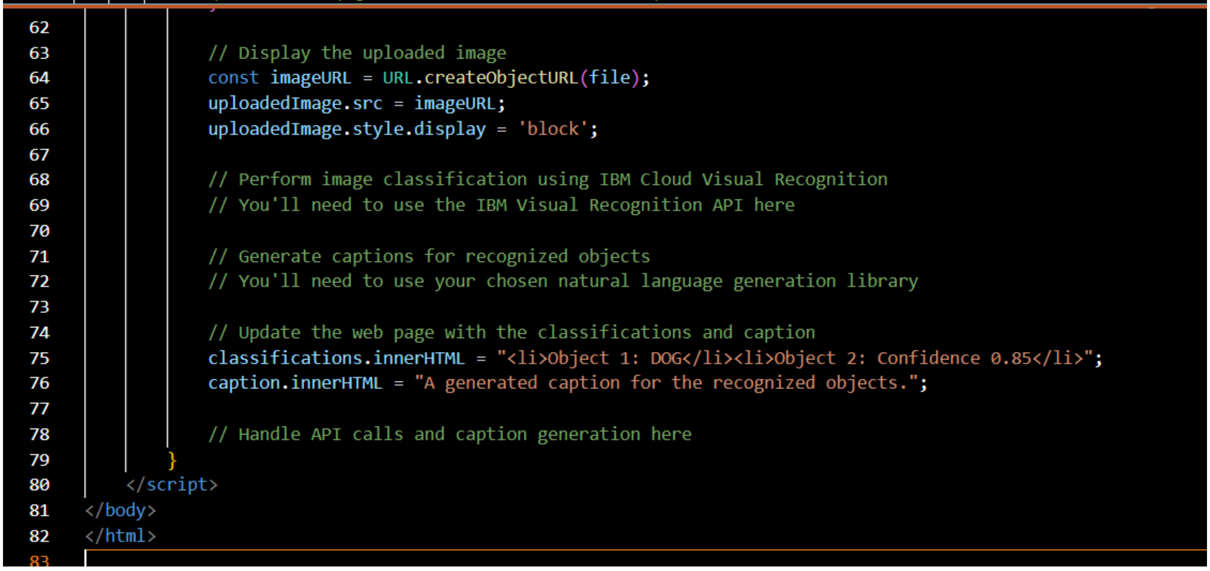
CODE FOR WEBSITE LAYOUT:



```html
<!DOCTYPE html>
<html>
<head>
    <title>Image Recognition</title>
    <style>
        /* Add your CSS styles here */
        body {
            font-family: Arial, sans-serif;
            background-color: coral;
        }

        .chat-container {
            width: 300px;
            border: 1px solid #ccc;
            padding: 10px;
            margin: 20px auto;
        }

        .user-message {
            background-color: #DCF8C6;
            margin: 5px 0;
            padding: 5px;
            border-radius: 5px;
        }

        .assistant-message {
            background-color: #F0F0F0;
            margin: 5px 0;
            padding: 5px;
            border-radius: 5px;
        }

        .center-heading {
            text-align: center;
        }
    </style>
```



```
</head>
<body>
    <h1 class="center-heading">Image Recognition with AI Captions</h1>
    <h2 class="center-heading">
    <input type="file" accept="image/*" id="imageUpload" onchange="processImage()">
    <img id="uploadedImage" style="max-width: 300px; display: none;">
</h2>
    <h2 class="center-heading">Classified Objects:</h2>
    <ul id="classifications"></ul>

    <h2 class="center-heading">Generated Caption:</h2>
    <p id="caption"></p>

    <script>
        function processImage() {
            const imageUpload = document.getElementById('imageUpload');
            const uploadedImage = document.getElementById('uploadedImage');
            const classifications = document.getElementById('classifications');
            const caption = document.getElementById('caption');

            const file = imageUpload.files[0];
            if (!file) {
                return;
            }

            // Display the uploaded image
            const imageURL = URL.createObjectURL(file);
            uploadedImage.src = imageURL;
            uploadedImage.style.display = 'block';

            // Perform image classification using IBM Cloud Visual Recognition
            // You'll need to use the IBM Visual Recognition API here

            // Generate captions for recognized objects
            // You'll need to use your chosen natural language generation library

            // Update the web page with the classifications and caption
```



```

            // Display the uploaded image
            const imageURL = URL.createObjectURL(file);
            uploadedImage.src = imageURL;
            uploadedImage.style.display = 'block';

            // Perform image classification using IBM Cloud Visual Recognition
            // You'll need to use the IBM Visual Recognition API here

            // Generate captions for recognized objects
            // You'll need to use your chosen natural language generation library

            // Update the web page with the classifications and caption
```

```
            classifications.innerHTML = "<li>Object 1: DOG</li><li>Object 2: Confidence 0.85</li>";
            caption.innerHTML = "A generated caption for the recognized objects.";

            // Handle API calls and caption generation here
        }
    </script>
</body>
</html>
```

HOME PAGE:

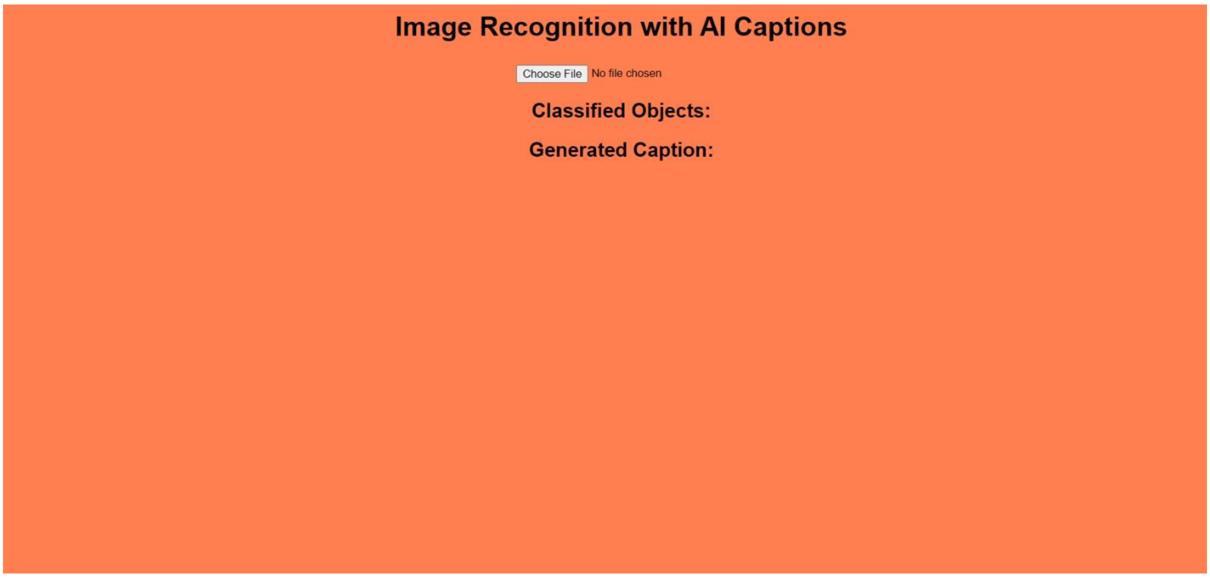


Image Recognition with AI Captions

No file chosen

Classified Objects:

Generated Caption:

OUTPUT PAGE:

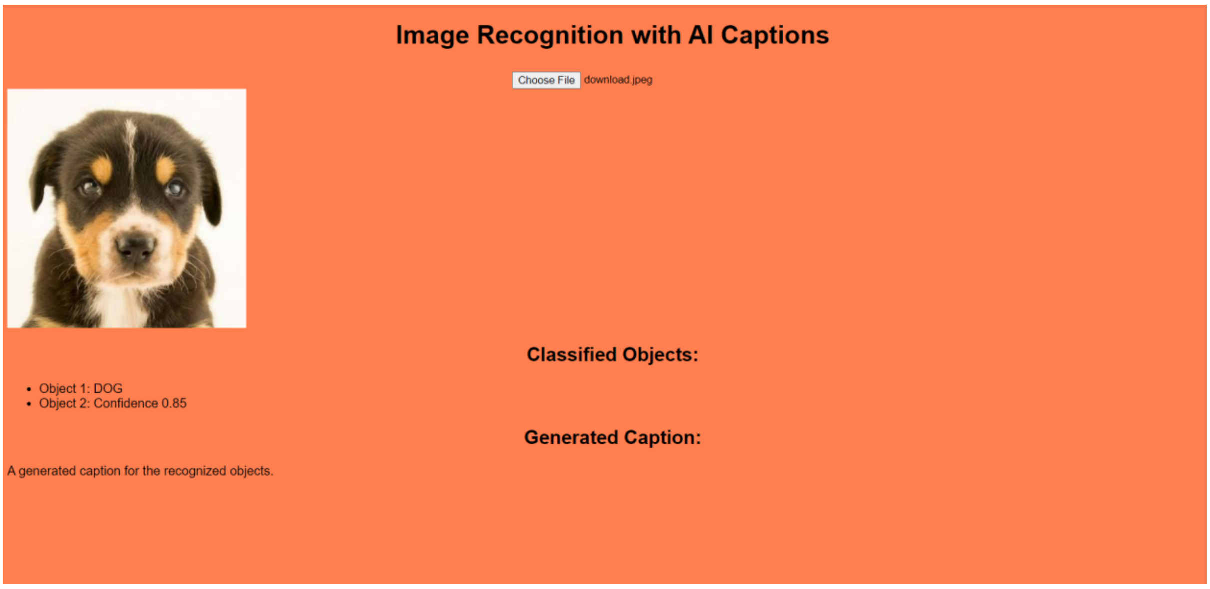


Image Recognition with AI Captions

download.jpeg

Classified Objects:

- Object 1: DOG
- Object 2: Confidence 0.85

Generated Caption:

A generated caption for the recognized objects.