**Project Description: Building an AI-Powered Image Recognition System with Flask and TensorFlow**

Our project is an innovative image recognition system that harnesses the capabilities of Python, TensorFlow, and Flask to provide detailed descriptions of uploaded images. This project represents the culmination of cutting-edge technology, offering a seamless and practical solution for image analysis.

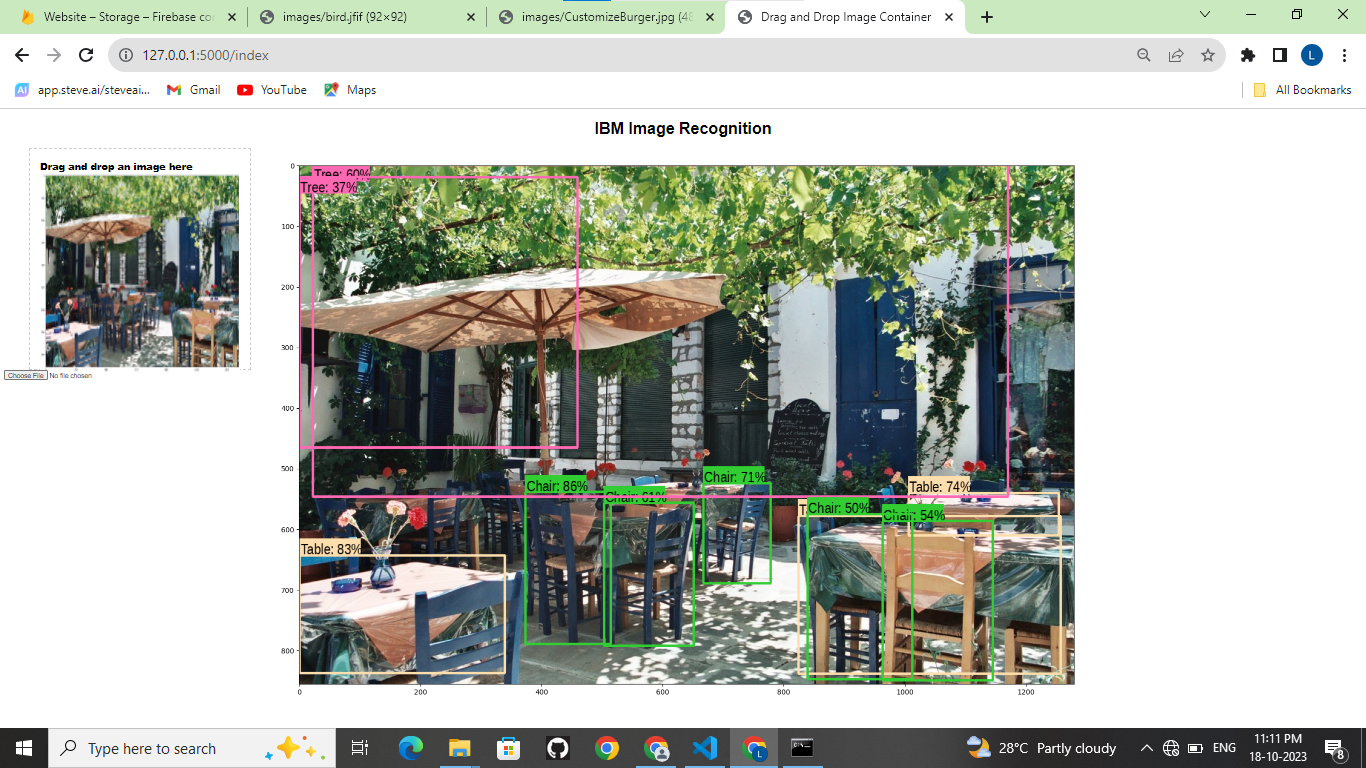
**Key Technical Components:**

1. **Flask Web Application:** Our project is centered around a Flask web application. Flask is a lightweight and efficient web framework for Python, making it ideal for serving as the backbone of our system. It handles user interactions, image uploads, and result delivery.
2. **TensorFlow-Powered AI Model:** At the core of our solution is a sophisticated AI model built using TensorFlow, a widely-used deep learning framework. This model has been trained to recognize and describe the contents of images, providing human-like descriptions based on the visual content.
3. **IBM Cloud Integration:** We've chosen IBM Cloud as our hosting and storage platform. This cloud infrastructure ensures high availability, scalability, and data security. Uploaded images are stored and accessed from the IBM Cloud, guaranteeing robust performance.

**Technical Workflow:**

1. **User Image Upload:** Users interact with our platform through a user-friendly interface, uploading images of their choice. These images are transmitted to our Flask application for processing.
2. **AI Image Analysis:** The uploaded images are processed by our TensorFlow-based AI model. This involves pre-processing the images and sending them through the model for analysis.
3. **Image Description Generation:** Once the AI model has completed its analysis, it generates textual descriptions of the images. These descriptions are then sent back to the user via the Flask application.
4. **IBM Cloud Storage:** All images are securely stored in the IBM Cloud. This integration allows for easy and efficient access to images and ensures data integrity and reliability.
5. **Deployment:** Our Flask application is deployed on the IBM Cloud, ensuring it's accessible to users across the internet. The deployment is configured to handle incoming requests and return results promptly.
6. **Continuous Improvement:** Our project is designed with a commitment to continuous improvement. The TensorFlow model will be updated and fine-tuned over time to maintain and enhance recognition accuracy.

Our project is a testament to the synergy of modern technologies. By combining the power of TensorFlow's deep learning capabilities with Flask's flexibility, and deploying the system on the IBM Cloud, we've created a robust and user-friendly image recognition solution that has the potential to transform how we interact with visual content. This project reflects the future of image analysis, with a focus on accuracy, accessibility, and scalability.



**Image recognition:**

import tensorflow as tf

import tensorflow\_hub as hub

# For downloading the image.

import matplotlib.pyplot as plt

import tempfile

from six.moves.urllib.request import urlopen

from six import BytesIO

# For drawing onto the image.

import numpy as np

from PIL import Image

from PIL import ImageColor

from PIL import ImageDraw

from PIL import ImageFont

from PIL import ImageOps

# For measuring the inference time.

import time

# Print Tensorflow version

print(tf.\_version\_)

# Check available GPU devices.

print("The following GPU devices are available: %s" % tf.test.gpu\_device\_name())

def display\_image(image):

fig = plt.figure(figsize=(20, 15))

plt.grid(False)

plt.imshow(image)

def download\_and\_resize\_image(url, new\_width=256, new\_height=256,

display=False):

\_, filename = tempfile.mkstemp(suffix=".jpg")

response = urlopen(url)

image\_data = response.read()

image\_data = BytesIO(image\_data)

pil\_image = Image.open(image\_data)

pil\_image = ImageOps.fit(pil\_image, (new\_width, new\_height), Image.LANCZOS)

pil\_image\_rgb = pil\_image.convert("RGB")

pil\_image\_rgb.save(filename, format="JPEG", quality=90)

print("Image downloaded to %s." % filename)

if display:

display\_image(pil\_image)

return filename

image\_url = "https://upload.wikimedia.org/wikipedia/commons/6/60/Naxos\_Taverna.jpg" #@param

downloaded\_image\_path = download\_and\_resize\_image(image\_url, 1280, 856, True)

module\_handle = "https://tfhub.dev/google/faster\_rcnn/openimages\_v4/inception\_resnet\_v2/1" #@param ["https://tfhub.dev/google/openimages\_v4/ssd/mobilenet\_v2/1", "https://tfhub.dev/google/faster\_rcnn/openimages\_v4/inception\_resnet\_v2/1"]

detector = hub.load(module\_handle).signatures['default']

def load\_img(path):

img = tf.io.read\_file(path)

img = tf.image.decode\_jpeg(img, channels=3)

return img

def run\_detector(detector, path):

img = load\_img(path)

converted\_img = tf.image.convert\_image\_dtype(img, tf.float32)[tf.newaxis, ...]

start\_time = time.time()

result = detector(converted\_img)

end\_time = time.time()

result = {key:value.numpy() for key,value in result.items()}

print("Found %d objects." % len(result["detection\_scores"]))

print("Inference time: ", end\_time-start\_time)

image\_with\_boxes = draw\_boxes(

img.numpy(), result["detection\_boxes"],

result["detection\_class\_entities"], result["detection\_scores"])

display\_image(image\_with\_boxes)

run\_detector(detector, downloaded\_image\_path)

image\_urls = ["https://upload.wikimedia.org/wikipedia/commons/thumb/0/0d/Biblioteca\_Maim%C3%B3nides%2C\_Campus\_Universitario\_de\_Rabanales\_007.jpg/1024px-Biblioteca\_Maim%C3%B3nides%2C\_Campus\_Universitario\_de\_Rabanales\_007.jpg",

]

def detect\_img(image\_url):

start\_time = time.time()

image\_path = download\_and\_resize\_image(image\_url, 640, 480)

run\_detector(detector, image\_path)

end\_time = time.time()

print("Inference time:",end\_time-start\_time)

detect\_img(image\_urls[0])