```python
# Install Google Cloud Vision API if not already installed
!pip install --upgrade google-cloud-vision

# Import necessary libraries
from google.cloud import vision
from google.colab import files
import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing import image
import os
import io
from PIL import Image

# Upload the JSON key for Google Cloud Vision API
uploaded = files.upload()
key_file = next(iter(uploaded))
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = key_file

# Create a client for the Vision API
client = vision.ImageAnnotatorClient()

# Upload the CNN model file
uploaded = files.upload()
model_path = next(iter(uploaded))
model = tf.keras.models.load_model(model_path)

# Set the brand names
brand_names = ['Adidas', 'Apple', 'Nike', 'Swarovski', 'Under Armour']  # Update if needed
```

```
Collecting google-cloud-vision
  Downloading google_cloud_vision-3.8.0-py2.py3-none-any.whl.metadata (5.3 kB)
Requirement already satisfied: google-api-core!=2.0.*,!=2.1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.
Requirement already satisfied: google-auth!=2.24.0,!=2.25.0,<3.0.0dev,>=2.14.1 in /usr/local/lib/python3.10/dist-packages (from goo
Requirement already satisfied: proto-plus<2.0.0dev,>=1.22.3 in /usr/local/lib/python3.10/dist-packages (from google-cloud-vision) (
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.2 in /usr/local/lib/p
Requirement already satisfied: googleapis-common-protos<2.0.dev0,>=1.56.2 in /usr/local/lib/python3.10/dist-packages (from google-a
Requirement already satisfied: requests<3.0.0.dev0,>=2.18.0 in /usr/local/lib/python3.10/dist-packages (from google-api-core!=2.0.*
Requirement already satisfied: grpcio<2.0dev,>=1.33.2 in /usr/local/lib/python3.10/dist-packages (from google-api-core[grpc]!=2.0.*
Requirement already satisfied: grpcio-status<2.0.dev0,>=1.33.2 in /usr/local/lib/python3.10/dist-packages (from google-api-core[grp
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth!=2.24.0,!=2.25.0
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth!=2.24.0,!=2.25.0,
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth!=2.24.0,!=2.25.0,<3.0.0de
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0.dev0,>=2.18
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0.dev0,>=2.18.0->google-a
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0.dev0,>=2.18.0->go
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0.dev0,>=2.18.0->go
Downloading google_cloud_vision-3.8.0-py2.py3-none-any.whl (488 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 488.5/488.5 kB 6.8 MB/s eta 0:00:00
Installing collected packages: google-cloud-vision
Successfully installed google-cloud-vision-3.8.0
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving vision_api.json to vision_api.json
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving brand_detection_model.h5 to brand_detection_model.h5
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until
```

```python
# List of brand and restaurant names to match
brand_and_restaurant_names = [
    # Brand Names
    "Adidas", "Adidas Kids", "Adidas Originals", "Apple", "Armani Exchange", "Asics",
    "Armani Beauty", "Boss", "Brooks Brothers", "Byredo", "Calvin Klein", "Calvin Klein Underwear",
    "Charles Tyrwhitt", "Clinique", "Coach", "Crocs", "Da Milano", "Diesel", "Dior",
    "Dune London", "Ed-A-Mamma", "Emporio Armani", "Ethos Summit", "Forever New",
    "Forest Essentials", "Freshpik@Jio World Drive", "GAS", "Gant", "Hamleys", "Hamleys Play",
    "Heads Up For Tails", "Hunkemöller", "Jo Malone", "Jean-Claude Biguine", "Kama Ayurveda",
    "Kate Spade", "LensCrafters", "MAC", "Maison Des Parfums", "Maje", "Marks & Spencer Lingerie",
    "Michael Kors", "Montblanc", "Mothercare", "Muji", "Mulmul", "Neeladri", "Needledust",
    "Nike", "Onitsuka Tiger", "Paul Smith", "Ritu Kumar", "Samsonite", "Sandro", "Satya Paul",
    "Scotch & Soda", "Steve Madden", "Superdry", "Superdry Sport", "Sunglass Hut", "Swarovski",
    "The White Crow", "The White Crow - Books & Coffee", "Tommy Hilfiger", "Tommy Hilfiger Kids",
    "Tira", "Tumi", "Under Armour", "Vero Moda", "Vision Express", "West Elm",

    # Restaurant Names
    "Arbab", "Bateel", "Bombay Island", "CocoCart", "Cou Cou By Oberoi", "Entisi", "Flax",
    "FOO", "Grandmama's Cafe", "Hurrem's Turkish Baklava & Confectionery", "Indigo Delicatessen",
    "Kofuku", "Legit Burgers", "MOTODO", "Neel", "Pret A Manger", "SAZ", "SEESAW",
    "Someplace Else", "Spice-O-Pedia", "Starbucks", "The Nutcracker", "The Sassy Café",
    "The White Crow Books And Coffee", "Twisting Scoops"
```

```
]


# Function to match detected text with the brand/restaurant names
def match_brand_or_restaurant(detected_text):
    for name in brand_and_restaurant_names:
        if name.lower() in detected_text.lower():
            return f"Detected with 100% confidence: {name}"
    return None

# Function to detect text using Google Vision API
def detect_text(path):
    """Detects text in the file using Google Vision."""
    with io.open(path, 'rb') as image_file:
        content = image_file.read()
    image = vision.Image(content=content)
    response = client.document_text_detection(image=image)
    texts = response.text_annotations

    if response.error.message:
        raise Exception(f'{response.error.message}\nFor more info on error messages, check: https://cloud.google.com/apis/design/errors

    # Extract the full detected text
    return texts[0].description if texts else ""



# Function to use CNN model to predict brand based on logo
def predict_brand_logo(image_path, model):
    # Load and preprocess the uploaded image
    img = image.load_img(image_path, target_size=(180, 180))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)  # Convert to a batch of size 1
    img_array = img_array / 255.0  # Rescale the image

    # Predict the class
    predictions = model.predict(img_array)
    predicted_class = np.argmax(predictions[0])  # Get the index of the class with the highest probability
    predicted_label = brand_names[predicted_class]
    confidence = np.max(predictions[0]) * 100

    return f"Predicted Brand: {predicted_label} with {confidence:.2f}% confidence"



def process_image(image_path, model):
    # Step 1: OCR - Detect text
    extracted_text = detect_text(image_path)
    match_result = match_brand_or_restaurant(extracted_text)

    if match_result:
        print(match_result)
    else:
        # Step 2: CNN - Predict brand logo
        print("No text match found in OCR. Proceeding to CNN for logo detection...")
        cnn_result = predict_brand_logo(image_path, model)
        print(cnn_result)

# Upload an image file to analyze
uploaded = files.upload()
image_path = next(iter(uploaded))

# Run the complete process
process_image(image_path, model)
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving image_4.png to image_4.png
No text match found in OCR. Proceeding to CNN for logo detection...
```