import os

import zipfile

import json

import numpy as np

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras import layers, models

from sklearn.model\_selection import train\_test\_split

from PIL import Image

!pip install roboflow

from roboflow import Roboflow

rf = Roboflow(api\_key="EiumqQQO33oWqjhBJiJo")

project = rf.workspace("nidha-rahiman").project("clothing-color-detection")

dataset = project.version(1).download("coco")

!pip install tensorflow numpy matplotlib

import os

# List files in the dataset folder

print(os.listdir("clothing-color-detection-1"))

import json

# Load the COCO annotations

with open("clothing-color-detection-1/train/\_annotations.coco.json") as f:

    annotations = json.load(f)

# Print some annotation details

print(annotations.keys())  # Should contain 'images', 'annotations', 'categories'

# Create a dictionary mapping image IDs to file names

image\_id\_to\_filename = {img['id']: img['file\_name'] for img in annotations['images']}

print(image\_id\_to\_filename)

# Create a dictionary mapping category IDs to color labels

category\_id\_to\_label = {cat['id']: cat['name'] for cat in annotations['categories']}

print(category\_id\_to\_label)

import matplotlib.pyplot as plt

import matplotlib.patches as patches

from PIL import Image

# Load a sample image

sample\_image\_id = annotations['annotations'][425]['image\_id']

sample\_image\_name = image\_id\_to\_filename[sample\_image\_id]

image\_path = f"clothing-color-detection-1/train/{sample\_image\_name}"

# Open and display the image

image = Image.open(image\_path)

plt.figure(figsize=(6, 6))

plt.imshow(image)

# Add bounding boxes

ax = plt.gca()

for ann in annotations['annotations']:

    if ann['image\_id'] == sample\_image\_id:

        x, y, w, h = ann['bbox']

        rect = patches.Rectangle((x, y), w, h, linewidth=2, edgecolor='r', facecolor='none')

        ax.add\_patch(rect)

        label = category\_id\_to\_label[ann['category\_id']]

        plt.text(x, y, label, color='white', fontsize=12, backgroundcolor='red')

plt.show()

cropped\_images = []

labels = []

for ann in annotations['annotations']:

    image\_id = ann['image\_id']

    category\_id = ann['category\_id']

    bbox = ann['bbox']  # Bounding box coordinates

    # Load the image

    img\_path = f"clothing-color-detection-1/train/{image\_id\_to\_filename[image\_id]}"

    image = Image.open(img\_path).convert("RGB")

    # Crop the image using bbox

    x, y, width, height = bbox

    cropped\_image = image.crop((x, y, x + width, y + height))

    # Resize to 224x224 for training

    cropped\_image = cropped\_image.resize((224, 224))

    # Convert to numpy array

    image\_array = np.array(cropped\_image)

    # Store the processed image and label

    cropped\_images.append(image\_array)

    labels.append(category\_id\_to\_label[category\_id])

# Convert to NumPy arrays

cropped\_images = np.array(cropped\_images)

labels = np.array(labels)

print(f"Processed {len(cropped\_images)} images for training.")

from sklearn.preprocessing import LabelEncoder

from tensorflow.keras.utils import to\_categorical

# Convert string labels to numeric values

label\_encoder = LabelEncoder()

encoded\_labels = label\_encoder.fit\_transform(labels)  # Converts 'red', 'blue' → [0, 1, 2, ...]

# Convert to one-hot encoding

num\_classes = len(label\_encoder.classes\_)  # Number of unique colors

one\_hot\_labels = to\_categorical(encoded\_labels, num\_classes)

print("Unique Classes:", label\_encoder.classes\_)  # Check mapping of labels

print(f"Encoded Labels Shape: {one\_hot\_labels.shape}")

from sklearn.model\_selection import train\_test\_split

# Split the dataset

X\_train, X\_val, y\_train, y\_val = train\_test\_split(cropped\_images, one\_hot\_labels, test\_size=0.2, random\_state=42)

print(f"Training data: {X\_train.shape}, Validation data: {X\_val.shape}")

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

# Define CNN model

model = Sequential([

    Conv2D(32, (3,3), activation='relu', input\_shape=(224, 224, 3)),

    MaxPooling2D(2,2),

    Conv2D(64, (3,3), activation='relu'),

    MaxPooling2D(2,2),

    Conv2D(128, (3,3), activation='relu'),

    MaxPooling2D(2,2),

    Flatten(),

    Dense(128, activation='relu'),

    Dropout(0.5),  # Prevent overfitting

    Dense(num\_classes, activation='softmax')  # Output layer (multi-class)

])

# Compile the model

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Show model summary

model.summary()

# Train the model

history = model.fit(

    X\_train, y\_train,

    validation\_data=(X\_val, y\_val),

    epochs=20,  # Increase for better results

    batch\_size=32,

    verbose=1

)

# Evaluate the trained model

loss, accuracy = model.evaluate(X\_val, y\_val)

print(f"Validation Accuracy: {accuracy\*100:.2f}%")

model.save("color\_model.keras")

print("Model saved successfully!")

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing import image

import tensorflow as tf

# Load the trained model

model = tf.keras.models.load\_model("/content/drive/My Drive/color\_model.h5")

# Upload a new image

from google.colab import files

uploaded = files.upload()

# Load and preprocess the image

img\_path = list(uploaded.keys())[0]

img = image.load\_img(img\_path, target\_size=(224, 224))

img\_array = image.img\_to\_array(img) / 255.0  # Normalize

img\_array = np.expand\_dims(img\_array, axis=0)  # Add batch dimension

# Predict the color

prediction = model.predict(img\_array)

predicted\_class = np.argmax(prediction)  # Get index of highest probability

predicted\_color = label\_encoder.inverse\_transform([predicted\_class])[0]  # Convert index to label

# Display the image with prediction

plt.imshow(img)

plt.axis("off")  # Hide axis

plt.title(f"Predicted Color: {predicted\_color}", fontsize=14, color="blue")

plt.show()

print(f"Predicted Color: {predicted\_color}")

from google.colab import drive

drive.mount('/content/drive')

model\_save\_path = '/content/drive/My Drive/color\_model.keras'

model.save(model\_save\_path)

model.summary()

from tensorflow.keras.models import load\_model

# Load the model

model = load\_model(model\_save\_path)

# Recompile the model

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

print("Model loaded and recompiled successfully!")

from tensorflow.keras.models import load\_model

model\_save\_path = '/content/drive/My Drive/color\_model.keras'

# Load the model

model = tf.keras.models.load\_model(model\_save\_path)

# Recompile the model

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

print("Model loaded and recompiled successfully!")