import numpy as np

import tensorflow as tf

from tensorflow.keras.applications import ResNet50

from tensorflow.keras.preprocessing.image import img\_to\_array, load\_img

from tensorflow.keras.applications.resnet50 import preprocess\_input

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.models import Model, load\_model

from tensorflow.keras.layers import Input, Dense, LSTM, Embedding, Dropout

from tensorflow.keras.utils import to\_categorical

# Load pre-trained ResNet model for feature extraction

resnet\_model = ResNet50(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

# Function to preprocess images

def preprocess\_image(image\_path):

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

img = img\_to\_array(img)

img = np.expand\_dims(img, axis=0)

img = preprocess\_input(img)

return img

# Function to extract features from images

def extract\_features(image\_path):

img = preprocess\_image(image\_path)

features = resnet\_model.predict(img)

features = np.reshape(features, features.shape[1:])

return features

# Load tokenizer and maximum sequence length

tokenizer = Tokenizer()

tokenizer.word\_index = np.load('word\_index.npy', allow\_pickle=True).item()

max\_sequence\_length = 30

# Define model architecture for caption generation

inputs1 = Input(shape=(2048,))

fe1 = Dropout(0.5)(inputs1)

fe2 = Dense(256, activation='relu')(fe1)

inputs2 = Input(shape=(max\_sequence\_length,))

se1 = Embedding(len(tokenizer.word\_index) + 1, 256, mask\_zero=True)(inputs2)

se2 = Dropout(0.5)(se1)

se3 = LSTM(256)(se2)

decoder1 = tf.keras.layers.add([fe2, se3])

decoder2 = Dense(256, activation='relu')(decoder1)

outputs = Dense(len(tokenizer.word\_index) + 1, activation='softmax')(decoder2)

caption\_model = Model(inputs=[inputs1, inputs2], outputs=outputs)

# Load trained caption generation model

caption\_model.load\_weights('caption\_model\_weights.h5')

# Function to generate caption for an image

def generate\_caption(image\_path):

# Extract features from the image

photo\_features = extract\_features(image\_path)

# Initialize caption with start token

in\_text = 'startseq'

for \_ in range(max\_sequence\_length):

# Encode caption text

sequence = tokenizer.texts\_to\_sequences([in\_text])[0]

sequence = pad\_sequences([sequence], maxlen=max\_sequence\_length)

# Predict next word

yhat = caption\_model.predict([photo\_features.reshape(1, -1), sequence], verbose=0)

# Convert probability to word index

yhat = np.argmax(yhat)

# Map index to word

word = [word for word, index in tokenizer.word\_index.items() if index == yhat]

# Break if end of caption is reached

if not word or word[0] == 'endseq':

break

# Update caption sequence

in\_text += ' ' + word[0]

return in\_text

# Example usage

image\_path = 'example\_image.jpg'

caption = generate\_caption(image\_path)

print("Generated caption:", caption)