**2. Task (IMAGE CAPTIONING)**

# IMAGE CAPTIONING FULL PROJECT CODE (RNN BASED)

# Section 1: Imports

import os

import numpy as np

import tensorflow as tf

from tensorflow.keras.applications import ResNet50

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

from tensorflow.keras.preprocessing import image

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.utils import pad\_sequences

from tensorflow.keras.models import Model

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

from tensorflow.keras.utils import to\_categorical

# Section 2: Feature Extraction

def extract\_features(img\_path):

base\_model = ResNet50(weights='imagenet')

model = Model(inputs=base\_model.input, outputs=base\_model.layers[-2].output)

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

x = image.img\_to\_array(img)

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

x = preprocess\_input(x)

features = model.predict(x, verbose=0)

return features

# Section 3: Load Captions

def load\_captions(captions\_file):

captions = {}

with open(captions\_file, 'r') as file:

for line in file:

tokens = line.strip().split('\t')

if len(tokens) == 2:

img, caption = tokens

if img not in captions:

captions[img] = []

captions[img].append("startseq " + caption + " endseq")

return captions

# Section 4: Tokenizer and Preprocessing

def create\_tokenizer(captions\_dict):

all\_captions = []

for cap\_list in captions\_dict.values():

all\_captions.extend(cap\_list)

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts(all\_captions)

return tokenizer

def max\_length(captions\_dict):

all\_captions = []

for cap\_list in captions\_dict.values():

all\_captions.extend(cap\_list)

return max(len(caption.split()) for caption in all\_captions)

# Section 5: Data Generator

def data\_generator(captions\_dict, features, tokenizer, max\_len, vocab\_size, batch\_size=32):

while True:

X1, X2, y = [], [], []

for key, caption\_list in captions\_dict.items():

for caption in caption\_list:

seq = tokenizer.texts\_to\_sequences([caption])[0]

for i in range(1, len(seq)):

in\_seq, out\_seq = seq[:i], seq[i]

in\_seq = pad\_sequences([in\_seq], maxlen=max\_len)[0]

out\_seq = to\_categorical([out\_seq], num\_classes=vocab\_size)[0]

X1.append(features[key][0])

X2.append(in\_seq)

y.append(out\_seq)

if len(X1) == batch\_size:

yield ([np.array(X1), np.array(X2)], np.array(y))

X1, X2, y = [], [], []

# Section 6: Build Model

def build\_model(vocab\_size, max\_length):

inputs1 = Input(shape=(2048,))

fe1 = Dropout(0.5)(inputs1)

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

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

se1 = Embedding(vocab\_size, 256, mask\_zero=True)(inputs2)

se2 = Dropout(0.5)(se1)

se3 = LSTM(256)(se2)

decoder1 = add([fe2, se3])

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

outputs = Dense(vocab\_size, activation='softmax')(decoder2)

model = Model(inputs=[inputs1, inputs2], outputs=outputs)

model.compile(loss='categorical\_crossentropy', optimizer='adam')

return model

# Section 7: Generate Caption

def generate\_caption(model, tokenizer, photo, max\_length):

in\_text = 'startseq'

for \_ in range(max\_length):

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

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

yhat = model.predict([photo, seq], verbose=0)

yhat = np.argmax(yhat)

word = tokenizer.index\_word.get(yhat)

if word is None:

break

in\_text += ' ' + word

if word == 'endseq':

break

return in\_text.replace("startseq", "").replace("endseq", "").strip()