1. Importing Libraries

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

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

from keras.preprocessing.text import Tokenizer

from keras.preprocessing.sequence import pad\_sequences

from keras.utils import to\_categorical

from keras.models import Model

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

from keras.applications.vgg16 import VGG16

from keras.applications.vgg16 import preprocess\_input

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

import pickle

import os

from tqdm import tqdm

**2. Data Preprocessing**

**Load and preprocess the Flickr8k dataset:**

def load\_doc(filename):

with open(filename, 'r') as file:

text = file.read()

return text

def load\_descriptions(doc):

mapping = dict()

for line in doc.split('\n'):

tokens = line.split()

if len(tokens) < 2:

continue

image\_id, image\_desc = tokens[0], tokens[1:]

image\_id = image\_id.split('.')[0]

image\_desc = ' '.join(image\_desc)

if image\_id not in mapping:

mapping[image\_id] = list()

mapping[image\_id].append(image\_desc)

return mapping

def clean\_descriptions(descriptions):

for key, desc\_list in descriptions.items():

for i in range(len(desc\_list)):

desc = desc\_list[i]

desc = desc.lower()

desc = desc.replace('[^A-Za-z]', '')

desc = ' '.join(word for word in desc.split() if len(word) > 1)

desc\_list[i] = 'startseq ' + desc + ' endseq'

def to\_vocabulary(descriptions):

all\_desc = set()

for key in descriptions.keys():

[all\_desc.update(d.split()) for d in descriptions[key]]

return all\_desc

filename = 'Flickr8k\_text/Flickr8k.token.txt'

doc = load\_doc(filename)

descriptions = load\_descriptions(doc)

print('Loaded: %d ' % len(descriptions))

clean\_descriptions(descriptions)

vocabulary = to\_vocabulary(descriptions)

print('Vocabulary Size: %d' % len(vocabulary))

# Save descriptions to file

pickle.dump(descriptions, open('descriptions.pkl', 'wb'))

3. Extracting Features using CNN (VGG16)

def extract\_features(directory):

model = VGG16()

model = Model(inputs=model.inputs, outputs=model.layers[-2].output)

features = dict()

for name in tqdm(os.listdir(directory)):

filename = directory + '/' + name

image = load\_img(filename, target\_size=(224, 224))

image = img\_to\_array(image)

image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))

image = preprocess\_input(image)

feature = model.predict(image, verbose=0)

image\_id = name.split('.')[0]

features[image\_id] = feature

return features

directory = 'Flickr8k\_Dataset'

features = extract\_features(directory)

print('Extracted Features: %d' % len(features))

pickle.dump(features, open('features.pkl', 'wb'))

4. Prepare Data for Training

def to\_lines(descriptions):

all\_desc = list()

for key in descriptions.keys():

[all\_desc.append(d) for d in descriptions[key]]

return all\_desc

def create\_tokenizer(descriptions):

lines = to\_lines(descriptions)

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts(lines)

return tokenizer

def max\_length(descriptions):

lines = to\_lines(descriptions)

return max(len(d.split()) for d in lines)

# Load descriptions from file

descriptions = pickle.load(open('descriptions.pkl', 'rb'))

tokenizer = create\_tokenizer(descriptions)

vocab\_size = len(tokenizer.word\_index) + 1

max\_length = max\_length(descriptions)

5. Define the Model

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

inputs1 = Input(shape=(4096,))

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

model = define\_model(vocab\_size, max\_length)

model.summary()

6. Data Generator

def data\_generator(descriptions, photos, tokenizer, max\_length):

while 1:

for key, desc\_list in descriptions.items():

photo = photos[key][0]

for desc in desc\_list:

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

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

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

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

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

yield [photo, in\_seq], out\_seq

# Load features from file

features = pickle.load(open('features.pkl', 'rb'))

### 7. Train the Model

epochs = 20

steps = len(descriptions)

for i in range(epochs):

generator = data\_generator(descriptions, features, tokenizer, max\_length)

model.fit(generator, epochs=1, steps\_per\_epoch=steps, verbose=1)

model.save('model\_' + str(i) + '.h5')

8. Generate Captions for New Images

def word\_for\_id(integer, tokenizer):

for word, index in tokenizer.word\_index.items():

if index == integer:

return word

return None

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

in\_text = 'startseq'

for \_ in range(max\_length):

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

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

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

yhat = np.argmax(yhat)

word = word\_for\_id(yhat, tokenizer)

if word is None:

break

in\_text += ' ' + word

if word == 'endseq':

break

return in\_text

photo = extract\_features('your\_image.jpg')

description = generate\_desc(model, tokenizer, photo, max\_length)

print(description)