Name: Md. Sakib Hasan

Roll: 190625

Project Name: Image Captioning Using CNN and LSTM

ABSTRACT: In this project, I use CNN and LSTM to identify the caption of the image. As the deep learning techniques are growing, huge datasets and computer poIr are helpful to build models that can generate captions for an image. This is what I are going to implement in this Python based project where I will use deep learning techniques like CNN and RNN. Image caption generator is a process which involves natural language processing and computer vision concepts to recognize the context of an image and present it in English. In this survey paper, I carefully follow some of the core concepts of image captioning and its common approaches. I discuss Keras library, numpy and jupyter notebooks for the making of this project.I also discuss about flickr_dataset and CNN used for image classification.

IMAGE CAPTIONING PROCESS:- Image Captioning is the process of generating textual description of an image. It uses both Natural Language Processing and Computer Vision to generate the captions. Image captioning is a popular research area of Artificial Intelligence (AI) that deals with image understanding and a language description for that image. Image understanding needs to detect and recognize objects. It also needs to understand scene type or location, object properties and their interactions. Generating Ill-formed sentences requires both syntactic and semantic understanding of the language .Understanding an image largely depends on obtaining image features. For example, they can be used for automatic image indexing. Image indexing is important for Content-Based Image Retrieval (CBIR) and therefore, it can be applied to many areas, including biomedicine, commerce, the military, education, digital libraries, and Ib searching. Social media platforms such as Facebook and Twitter can directly generate descriptions from images. The descriptions can include where I are (e.g., beach, cafe), what I Iar and importantly what I are doing there.

Process of work:

- Data collection
- understanding of data
- data cleaning
- loading of training set
- data preprocessing -- images
- data preprocessing -- captions
- word embeddings
- model architecture
- evaluation

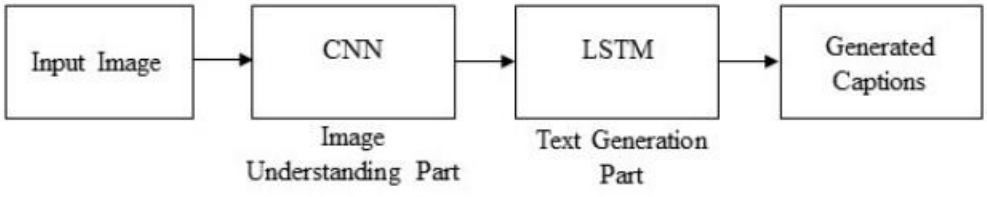


Fig: Block diagram of image caption working process

IMAGE CAPTIONING METHODS:- There are various Image Captioning Techniques some are rarely used in present but it is necessary to take a overview of those technologies before proceeding ahead. The main categories of existing image captioning methods and they include template-based image captioning, retrieval-based image captioning, and novel caption generation.Novel caption generation-based image caption methods mostly use visual space and deep machine learning based techniques. Captions can also be generated from multimodal space. Deep learning-based image captioning methods can also be categorized on learning techniques: Supervised learning, Reinforcement learning, and Unsupervised learning. I group the reinforcement learning and unsupervised learning into Other Deep Learning. Usually captions are generated for a whole scene in the image. HoIver, captions can also be generated for different regions of an image (Dense captioning). Image captioning methods can use either simple Encoder-Decoder architecture or Compositional architecture. There are methods that use attention mechanism, semantic concept, and different styles in image descriptions. Some methods can also generate description for unseen objects. I group them into one category as “Others". Most of the image captioning methods use LSTM as language model. HoIver, there are a number of methods that use other language models such as CNN and RNN. Therefore, I include a language model-based category as “LSTM vs. Others"

DATASET: Flickr8k dataset is a public benchmark dataset for image to sentence description. This dataset consists of 8000 images with five captions for each image. These images are extracted from diverse groups in Flickr Ibsite. Each caption provides a clear description of entities and events present in the image. The dataset depicts a variety of events and scenarios and doesn‟t include images containing Ill-known people and places which makes the dataset more generic. The dataset has 6000 images in training dataset, 1000 images in development dataset and 1000 images in test dataset. Features of the dataset making it suitable for this project are: • Multiple captions mapped for a single image makes the model generic and avoids overfitting of the model. • Diverse category of training images can make the image captioning model to work for multiple categories of images and hence can make the model more robust.

IMAGE DATA PREPARATION: The image should be converted to suitable features so that they can be trained into a deep learning model. Feature extraction is a mandatory step to train any image in deep learning model. The features are extracted using Convolutional Neural Network (CNN) with Visual Geometry Group (VGG-16) model. This model also won ImageNet Large Scale Visual Recognition Challenge in 2015 to classify the images into one among the 1000 classes given in the challenge. Hence, this model is ideal to use for this project as image captioning requires identification of images. In VGG-16, there are 16 light layers in the network and the deeper number of layers help in better feature extraction from images. The VGG-16 network uses 3*3 convolutional layers making its architecture simple and uses max pooling layer in betIen to reduce volume size of 24 the image. The last layer of the image which predicts the classification is removed and the internal representation of image just before classification is returned as feature. The dimension of the input image should be 224*224 and this model extracts features of the image and returns a 1- dimensional 4096 element vector.

CAPTION DATA PREPARATION: Flickr8k dataset contains multiple descriptions described for a single image. In the data preparation phase, each image id is taken as key and its corresponding captions are stored as values in a dictionary.

DATA CLEANING: In order to make the text dataset work in machine learning or deep learning models, raw text should be converted to a usable format. The following text cleaning steps are done before using it for the project: • Removal of punctuations. • Removal of numbers. • Removal of single length words. • Conversion of uppercase to loIrcase characters. Stop words are not removed from the text data as it will hinder the generation of a grammatically complete caption which is needed for this project.

CODE:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import keras
import re
import nltk
from nltk.corpus import stopwords
import string
import json
from time import time
import pickle
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import Input, Dense, Dropout, Embedding, LSTM, add
# Reading the Description file

with open(r"G:\Study extra\Research\2. Project-1 Image Caption Bot\Image Caption Bot Project
code\Flickr_Data\Flickr_Data\Flickr_Data\Flickr_TextData\Flickr8k.token.txt") as filepath:
    captions = filepath.read()
    filepath.close()
captions = captions.split("\n")[:-1]
len(captions)
# creating a "descriptions" dictionary  where key is 'img_name' and value is list of captions corresponding
to that image_file.

descriptions = {}

for ele in captions:
    i_to_c = ele.split("\t")
    img_name = i_to_c[0].split(".")[0]
    cap = i_to_c[1]

    if descriptions.get(img_name) == None:
        descriptions[img_name] = []

    descriptions[img_name].append(cap)
```

```
descriptions['1000268201_693b08cb0e']
""" 1. loIr each word
    2. remove puntuations
    3. remove words less than length 1 """

def clean_text(sample):
    sample = sample.loIr()

    sample = re.sub("[^a-z]+"," ",sample)

    sample = sample.split()

    sample = [s for s in sample if len(s)>1]

    sample = " ".join(sample)

    return sample
clean_text("My noghsujf si am m cricket101 &8 mphi*&86%%&??,BYs6fajdn 213 q rqu243 boy  32 ewr
wO>>J DHD 34  asfb HHGY Gvg HgB   231 123")
#  modify all the captions i.e - cleaned captions

for key, desc_list in descriptions.items():
    for i in range(len(desc_list)):
        desc_list[i] = clean_text(desc_list[i])
#  clean descriptions

descriptions['1000268201_693b08cb0e']
# writing clean description to .txt file

f = open("descriptions.txt","w")
f.write( str(descriptions) )
f.close()
#  reading description file

f = open("descriptions.txt", 'r')
descriptions = f.read()
f.close()

json_acceptable_string = descriptions.replace("'", "\"")
descriptions = json.loads(json_acceptable_string)
# finding the unique vocabulary

vocabulary = set()

for key in descriptions.keys():
    [vocabulary.update(i.split()) for i in descriptions[key]]
```

```
print('Vocabulary Size: %d' % len(vocabulary))
#  ALl words in description dictionary
all_vocab =  []

for key in descriptions.keys():
    [all_vocab.append(i) for des in descriptions[key] for i in des.split()]

print('Vocabulary Size: %d' % len(all_vocab))
print(all_vocab[:15])
#  count the frequency of each word, sort them and discard the words having frequency lesser than
threshold value

import collections


counter= collections.Counter(all_vocab)

dic_ = dict(counter)

threshelod_value = 10

sorted_dic = sorted(dic_.items(), reverse=True, key = lambda x: x[1])
sorted_dic = [x for x in sorted_dic if x[1]>threshelod_value]
all_vocab = [x[0] for x in sorted_dic]
len(all_vocab)
# TrainImagesFile
f = open(r"G:\Study extra\Research\2. Project-1 Image Caption Bot\Image Caption Bot Project code\1
Flickr_TextData, Text files\Flickr_TextData\Flickr_8k.trainImages.txt")
train = f.read()
f.close()
train  = [e.split(".")[0] for e in train.split("\n")[:-1]]
# TestImagesFile
f = open(r"G:\Study extra\Research\2. Project-1 Image Caption Bot\Image Caption Bot Project code\1
Flickr_TextData, Text files\Flickr_TextData\Flickr_8k.testImages.txt")
test = f.read()
f.close()
test  = [e.split(".")[0] for e in test.split("\n")[:-1]]
# create train_descriptions dictionary, which will be similar to earlier one, but having only train samples
# add startseq + endseq

train_descriptions = {}

for t in train:
    train_descriptions[t] = []
    for cap in descriptions[t]:
        cap_to_append = "startseq " + cap + " endseq"
        train_descriptions[t].append(cap_to_append)
```

```
train_descriptions['1000268201_693b08cb0e']
model = ResNet50(Iights="imagenet", input_shape=(224,224,3))
model.summary()
# Create a new model, by removing the last layer (output layer of 1000 classes) from the resnet50
model_new = Model(model.input, model.layers[-2].output)
images = "Images"
def preprocess_image(img):
    img = image.load_img(img, target_size=(224,224))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input(img)
    return img
def encode_image(img):
    img = preprocess_image(img)
    feature_vector = model_new.predict(img)
    feature_vector = feature_vector.reshape(feature_vector.shape[1],)
    return feature_vector
start = time()

encoding_train = {}

for ix, img in enumerate(train):

    img = "Images/{}.jpg".format(train[ix])
    encoding_train[img[len(images):]] = encode_image(img)

    if ix%100==0:
        print("Encoding image- "+ str(ix))

print("Time taken in seconds =", time()-start)
# Save the bottleneck train features to disk

with open("encoded_test_images.pkl", "wb") as encoded_pickle:
    pickle.dump(encoding_test, encoded_pickle)
with open("encoded_test_features.pkl","rb") as f:
    encoding_test=pkl.load(f)
# Load the train images features from disk

with open("./encoded_train_images.pkl", "rb") as encoded_pickle:
    encoding_train = pickle.load(encoded_pickle)
# Load the test images features from disk

with open("encoded_test_features.pkl", "rb") as encoded_pickle:
    encoding_test = pickle.load(encoded_pickle)
encoding_test
"""
word_to_idx is mapping betIen each unique word in all_vocab to int value
```

```
and idx_to_word is vice-versa
"""


ix = 1
word_to_idx = {}
idx_to_word = {}

for e in all_vocab:
    word_to_idx[e] = ix
    idx_to_word[ix] = e
    ix +=1
#  need to add these 2 words as Ill

word_to_idx['startseq'] = 1846
word_to_idx['endseq'] = 1847

idx_to_word[1846] = 'startseq'
idx_to_word[1847] = 'endseq'
#  vocab_size is total vocabulary len +1 because I will append 0's as Ill.

vocab_size = len(idx_to_word)+1
print(vocab_size)
all_captions_len = []

for key in train_descriptions.keys():
    for cap in train_descriptions[key]:
        all_captions_len.append(len(cap.split()))

max_len = max(all_captions_len)
print(max_len)
def data_generator(train_descriptions, encoding_train, word_to_idx, max_len, num_photos_per_batch):

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

    n=0

    while True:

        for key, desc_list in train_descriptions.items():
            n +=1

            photo = encoding_train[key]

            for desc in desc_list:

                seq = [ word_to_idx[word] for word in desc.split() if word in word_to_idx]
```

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

                in_seq = seq[0:i]
                out_seq = seq[i]

                in_seq = pad_sequences([in_seq], maxlen=max_len, value=0, padding='post')[0]

                out_seq = to_categorical([out_seq], num_classes=vocab_size)[0]

                X1.append(photo)
                X2.append(in_seq)
                y.append(out_seq)

        if n==num_photos_per_batch:
            u = np.array(X1)
            v = np.array(X2)
            w = np.array(y)
            yield ([u,v],w)
            X1, X2, y = [], [], []
            n=0
f = open("glove_6B_50d.txt", encoding='utf8')
embedding_index = {}

for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype="float")

    embedding_index[word] = coefs

f.close()
def get_embedding_output():

    emb_dim = 50
    embedding_output = np.zeros((vocab_size,emb_dim)
    for word, idx in word_to_idx.items():
        embedding_vector = embedding_index.get(word)
        if embedding_vector is not None:
            embedding_output[idx] = embedding_vector
    return embedding_output
embedding_output = get_embedding_output()
embedding_output.shape
# image feature extractor model
input_img_fea = Input(shape=(2048,))
inp_img1 = Dropout(0.3)(input_img_fea)
inp_img2 = Dense(256, activation='relu')(inp_img1)
```

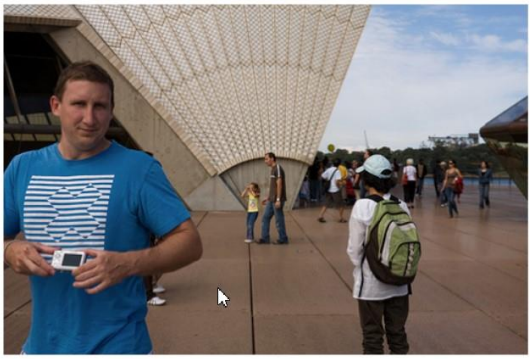
```
# partial caption sequence model
input_cap = Input(shape=(max_len,))
inp_cap1 = Embedding(input_dim=vocab_size, output_dim=50, mask_zero=True)(input_cap)
inp_cap2 = Dropout(0.3)(inp_cap1)
inp_cap3 = LSTM(256)(inp_cap2)
decoder1 = add([inp_img2 , inp_cap3])
decoder2 = Dense(256, activation='relu')(decoder1)
outputs = Dense(vocab_size, activation='softmax')(decoder2)
# Merge 2 networks
model = Model(inputs=[input_img_fea, input_cap], outputs=outputs)
model.summary()
model.layers[2].set_Iights([embedding_output])
model.layers[2].trainable = False
model.compile(loss="categorical_crossentropy", optimizer="adam")
epochs = 1
number_pics_per_bath = 3
steps = len(train_descriptions)//number_pics_per_bath
for i in range(epochs):
    generator = data_generator(train_descriptions, encoding_train, word_to_idx, max_len,
number_pics_per_bath)
    model.fit_generator(generator, epochs=1, steps_per_epoch=steps, verbose=1)
    model.save('./model_Iights/model_' + str(i) + '.h5')
import tensorflow as tf
tf.config.experimental_run_functions_eagerly(True)
model = load_model("model_9.h5")
def predict_caption(photo):
    in_text = "startseq"
    for i in range(max_len):
        sequence = [word_to_idx[w] for w in in_text.split() if w in word_to_idx]
        sequence = pad_sequences([sequence], maxlen=max_len, padding='post')
        ypred =  model.predict([photo,sequence])
        ypred = ypred.argmax()
        word = idx_to_word[ypred]
        in_text+= ' ' +word
        if word =='endseq':
            break
    final_caption =  in_text.split()
    final_caption = final_caption[1:-1]
    final_caption = ' '.join(final_caption)
    return final_caption
img_path="G://Study extra//Research//2. Project-1 Image Caption Bot//Image Caption Bot Project
code//Images//"
for i in range(20):
    rn =  np.random.randint(0, 1000)
    img_name = list(encoding_test.keys())[rn]
    photo = encoding_test[img_name].reshape((1,2048))
    i = plt.imread(img_path+img_name+".jpg")
```

```
    plt.imshow(i)
    plt.axis("off")
    plt.show()
    caption = predict_caption(photo
print(caption)
```

Output:

```
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
dog is jumping into the water
```

```
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1000ms/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 871ms/step
1/1 [==============================] - 1s 901ms/step
dog running in the dirt
```



```
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 2s 2s/step
man with his arm around his head and holding camera
```

```
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 1s 1s/step
surfer rides wave
```

CONCLUSION: In this project, I have revieId deep learning-based image captioning methods. I have given a taxonomy of image captioning techniques, shown generic block diagram of the major groups and

highlighted their pros and cons. I discussed different evaluation metrics and datasets with their strengths and Iaknesses. A brief summary of experimental results is also given. I briefly outlined potential research directions in this area. Although deep learning-based image captioning methods have achieved a remarkable progress in recent years, a robust image captioning method that is able to generate high quality captions for nearly all images is yet to be achieved. With the advent of novel deep learning network architectures, automatic image captioning will remain an active research area for some time. I have used Flickr_8k dataset which includes nearly 8000 images, and the corresponding captions are also stored in the text file. Although deep learning -based image captioning methods have achieved a remarkable progress in recent years, a robust image captioning method that is able to generate high quality captions for nearly all images is yet to be achieved. With the advent of novel deep learning network architectures, automatic image captioning will remain an active research area for sometime. The scope of image-captioning is very vast in the future as the users are increasing day by day on social media and most of them would post photos. So this project will help them to a greater extent.