Image Caption Generator

Contents:

1.Introduction

In recent years, with the rapid development of artificial intelligence, image caption has gradually attracted the attention of many researchers in the field of artificial intelligence and has become an interesting and arduous task. Image caption, automatically generating natural language descriptions according to the content observed in an image, is an important part of scene understanding, which combines the knowledge of computer vision and natural language processing.

a. Overview

Being able to automatically describe the content of an image using properly formed English sentences is a challenging task, but it could have a great impact by helping visually impaired people better understand their surroundings.

We are creating a web application where the user selects the image and the image is fed into the model that is trained and generated caption will be displayed on the webpage.

b. Purpose

Image caption generator is a task that involves computer vision and natural language processing concepts to recognize the context of an image and describe them in a natural language like English.

2. Literature Survey

a. Existing Problem

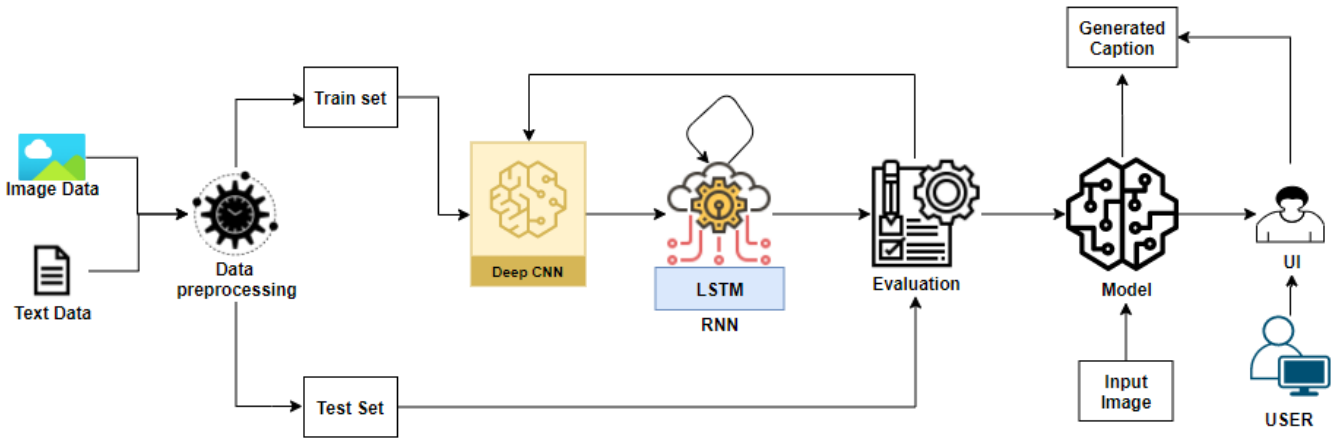
The problem introduces a captioning task, which requires a computer vision system to both localize and describe salient regions in images in natural language.

b. Proposed Solution

We will tackle this problem using an Encoder-Decoder model. Here our encoder model will combine both the encoded form of the image and the encoded form of the text caption and feed to the decoder.

3. Theoretical Analysis

a. Block Diagram



b. Hardware & Software Requirement

1. Tensorflow
2. Keras
3. Anaconda Navigator
4. Flask
5. GPU

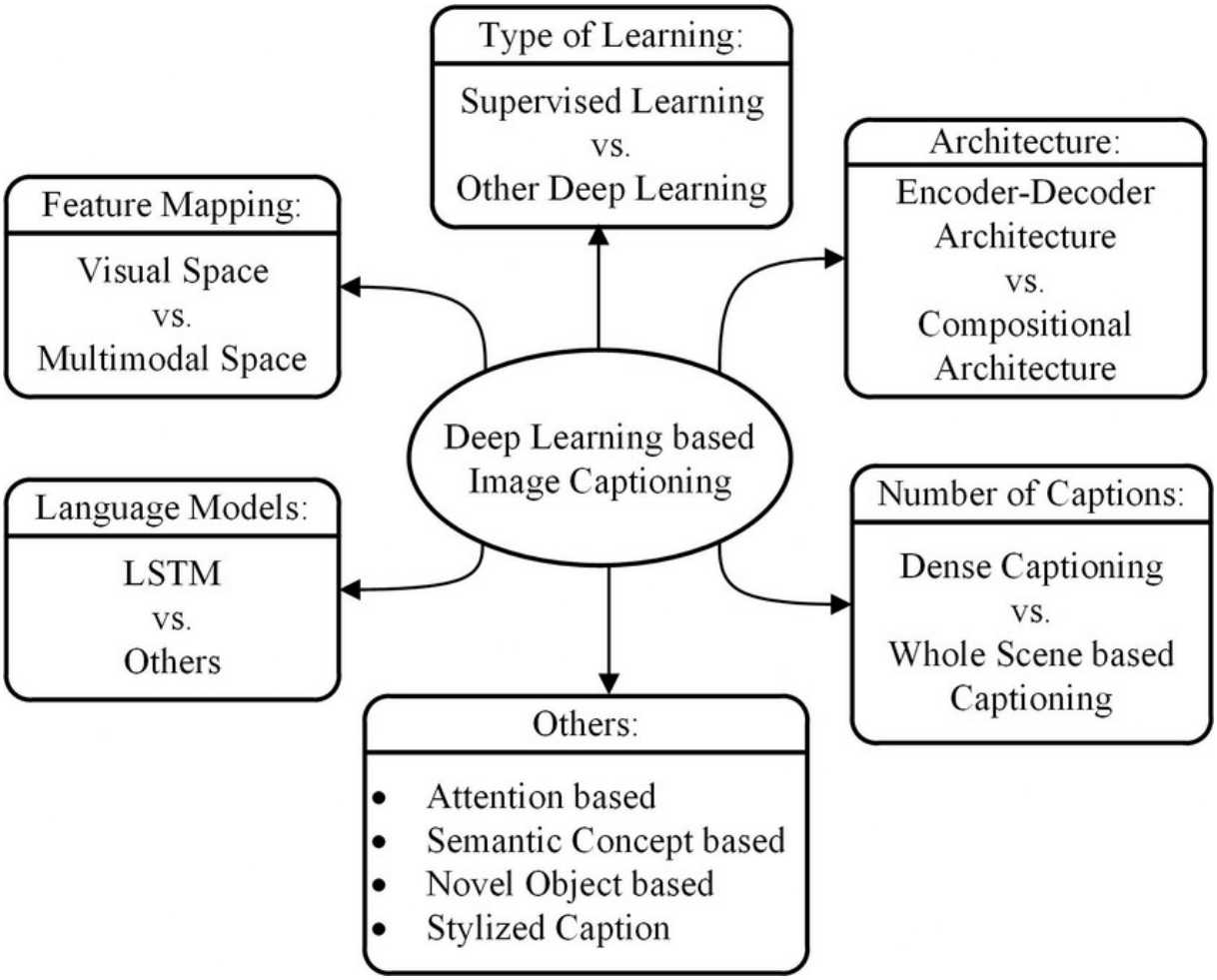
4.Experimental Investigations

Tensor flow: TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources.

Keras: Keras leverages various optimization techniques to make high-level neural network API easier and more performantresources.

Flask: Web framework used for building Web applications .

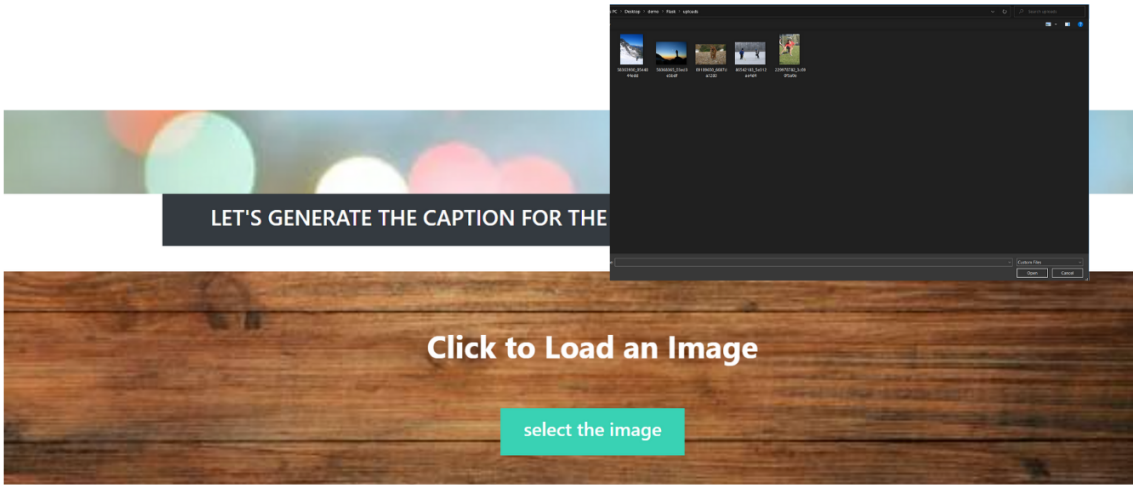
5.Flowchart



6.Result

- Firstly, we need to run the application in the bowser using local host

- Now we need to open anaconda prompt from the start menu.
- Navigate to the folder where your python script is.
- Now type “python app.py” command



7. Advantages & Disadvantages

Advantages:

- Each attribute corresponds to one entry of the used vocabulary.
- Complementary properties of RNN and LSTM.
- Properties of bottom-up & top-down captioning approaches.

Disadvantages:

- Template based framework is strictly constrained to image contents recognized by visual models.
- Retrieval based image captioning methods transfer well-formed human-written sentences or phrases for generating descriptions for query images.

8. Applications

- Recommendations in editing applications.
- Usage in virtual assistants.
- For image indexing.
- For visually impaired persons.
- for social media, and several other natural language processing applications.

9. Conclusion

- We have implemented a deep learning approach for the captioning of images. The sequential API of Keras

was used with Tensorflow as a backend to implement the deep learning architecture.

10. Future Scope

Image caption generator model is widening its boundaries from just taking images and generating captions. It is now widely been used to assist virual assistants like Google, Alexa, etc. It also helps designers and developers with recommendations in editing appliacations and also in developing applications.

It is also helpful in searching for the user need and land the user with the desired result.

11. Bibiliography

References used from previous works and websites referred for analysis about the project and their links:

- https://youtu.be/kE5QZ8G_78c - Supervised and Unsupervised learning
- https://youtu.be/DKSZHN7jftI - Artificial Neural Networks

- https://youtu.be/5ctbvkAMQO4 - Natural language processing

- https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c - VGG16 Reference

12. Appendix

a.Source Code

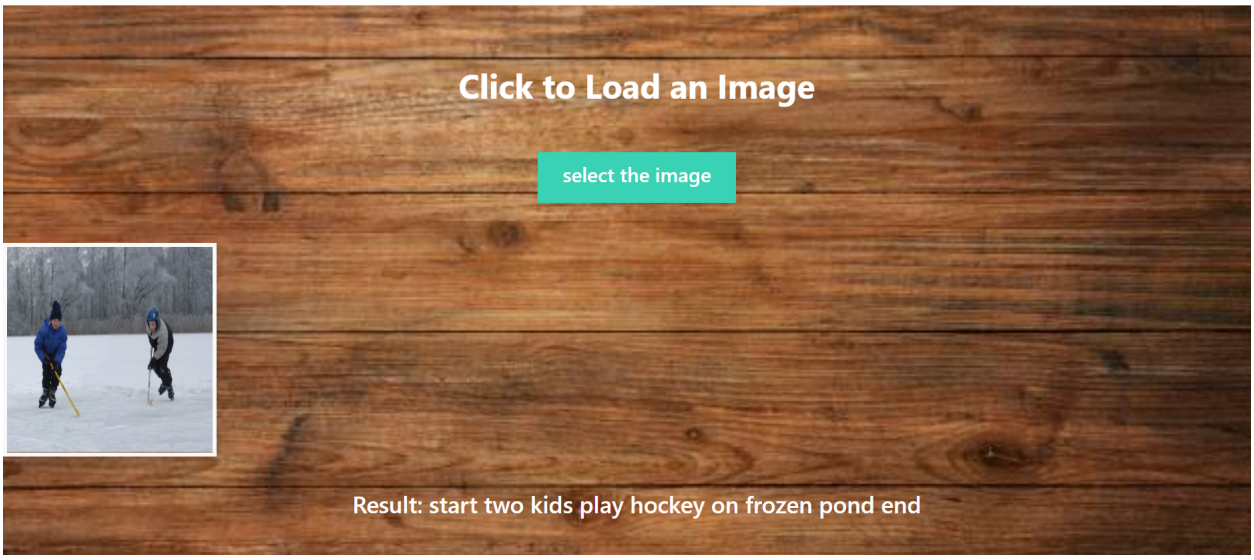
app.py

```python
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.applications.xception import Xception
from keras.models import load_model
from pickle import load
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import argparse
import os
from flask import Flask , request, render_template
from werkzeug.utils import secure_filename
#from gevent.pywsgi import WSGIServer
from gtts import gTTS

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')
@app.route('/predict',methods = ['GET','POST'])
def upload():
    if request.method == 'POST':
        f = request.files['image']
        print("current path")
        basepath = os.path.dirname(__file__)
        print("current path", basepath)
        filepath = os.path.join(basepath,'uploads',f.filename)
        print("upload folder is ", filepath)
        f.save(filepath)
        text = modelpredict(filepath)
        return text

def extract_features(filename, model):
    image = Image.open(filename)

    image = image.resize((299,299))
    image = np.array(image)
    # for images that has 4 channels, we convert them into 3 channels
    if image.shape[2] == 4:
        image = image[..., :3]
    image = np.expand_dims(image, axis=0)
```

app.py

```
47
48  def word_for_id(integer, tokenizer):
49   for word, index in tokenizer.word_index.items():
50       if index == integer:
51           return word
52   return None
53
54
55  def generate_desc(model, tokenizer, photo, max_length):
56      in_text = 'start'
57      for i in range(max_length):
58          sequence = tokenizer.texts_to_sequences([in_text])[0]
59          sequence = pad_sequences([sequence], maxlen=max_length)
60          pred = model.predict([photo,sequence], verbose=0)
61          pred = np.argmax(pred)
62          word = word_for_id(pred, tokenizer)
63          if word is None:
64              break
65          in_text += ' ' + word
66          if word == 'end':
67              break
68      return in_text
69
70  #path = 'Flicker8k_Dataset/111537222_07e56d5a30.jpg'
71  def modelpredict(filepath):
72      max_length = 32
73      tokenizer = load(open(r"./models/tokenizer.p","rb"))
74      model = load_model(r'./models/model_9.h5')
75      xception_model = Xception(include_top=False, pooling="avg")
76
77      photo = extract_features(filepath, xception_model)
78      #img = Image.open(img_path)
79      description = generate_desc(model, tokenizer, photo, max_length)
80      print(description)
81      myobj = gTTS(text=description , lang="es", slow=False)
82      myobj.save("welcome.mp3")
83      os.system("welcome.mp3")
84      return description
85
86  if __name__ == '__main__':
87      app.run(debug = True,threaded=False)
```

b. UI Output


Click to Load an Image
select the image
Result: start two kids play hockey on frozen pond end