ILLINOIS INSTITUTE OF TECHNOLOGY

CS 512 Project Report

Student 1: Neil Nemi Shah

Student 1 Id: A20500775

Student 2: Nandish Pratikbhai Bhagat

Student 2 Id: A20490179

Illinois Institute of Technology

April 22, 2022

1) Problem Statement

In this research paper the authors aim to consistently and accurately compose appropriate phrases describing the image using the generated keywords.

2) Proposed Solution

The solution can easily be imagined while considering the process we need to follow to achieve the food classification output. The process is as follows:
However before we move forward there are certain things that we need to take care of before we move forward.

a) Setting up the environment or virtual environment for GPU usage.
b) DataSet (also narrowing down to the exactly right one). Hence I am going to give the link to it https://drive.google.com/drive/folders/1mWnyImQtU6uPXYtltmrwvraxAlPtIbEz?usp=sharing

Step 1:
We take the image dataset and we do image processing by controlling some parameters. We also take a pre trained VGG 16 model to understand the image data set
Step 2:
We have a dataset for testing which has 1000 images on which we train and obtain bleu score.
Step 3:
We configure the neural network and start the CNN Process for generating vocabulary.
Step 4:
Then we form RNN and use LSTM to accurately structure meaningful sentences using the generated keywords.
Step 5:
Decoder- The model's final step combines the information from the Image extractor and sequence processor phases.
Step 6:
Training Phase.
Step 7:

Implementation Phase generating new captions for a random image.
Step 8:
Optimising and resolving errors.

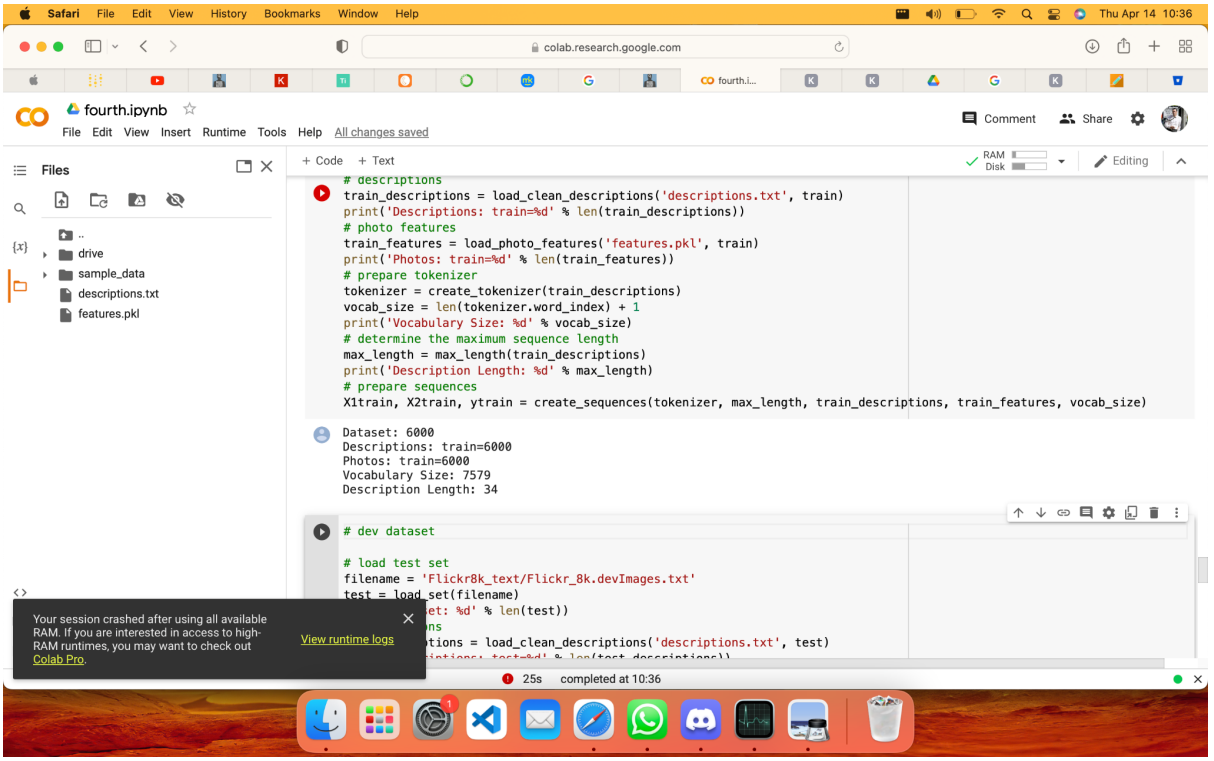
3) Implementation Details

i) Import all the necessary libraries and functions.
ii) Now as we know that this project will require intense model training we have implemented an environment where we install CUDA, cudnn and tensorflow, fixing bugs along the way, for GPU enabled functioning.
iii) We are going to use the VGG16 model to give the images some context however we are not going to use the last layer of this model as this is not a classification project.
iv) We will now preprocess the image and change the target size 224*224 and extract its features hence converting its pixels into a numpy array. Giving each image an id and storing its features in it.
v) We make a features.pkl and while writing with binary we write the data (features) about the images so we can fetch them in future. Now we will focus on the text data. We take the stored captions and preprocess them by removing the image id, removing their type and then we will finally map the image description with the image.
vi) Now we will focus on the text data. We take the stored captions and preprocess them by removing the image id, removing their type and then we will finally map the image description with the image.
vii)We make a file called descriptions.txt which stores all the descriptions. Then we make a start and end sequence for each description to clearly pinpoint the intended start and end.
ix) We now calculate the total vocabulary length.
x) Now we will convert the dictionary into a list of descriptions.
xi) we will now make the final model where we merge the text and image with a decoder model

x) we have created a function which will return sequence
xi)Now we make the data generator by using function generator 1.
x) We train the model with 50 epochs with the use of the above generator 1 function. We also save each model with the name saved_model[number].h5(i.e saved_model0.h5)
xi) Now we load the train images and get the tokens
xii)Now we read the descriptions.txt and do necessary processing on it. After that we prepare the tokenizer
xiii) Now to get the BLEU score(using test data) we take the test images and descriptions.txt for test images and do necessary processing on it.
xiv) Now we use Tokenizer() and do necessary processing and then generate a tokenizer.pkl file.
xv) Now we load two things 1) tokenizer and 2) the model using model.load()
xvi)Now using the function fun2extractf() we get the features of the given test image. i.e. Testimage1.jpeg
xvii) Now using these features and trained models we generate the new caption.

ERRORS Encountered (NOW RESOLVED)

1) RAM memory exceeded and we changed to Progressive Loading Approach



2) Processing took a lot of time and then we implemented GPU

```
#We check if the GPU is available or not
from tensorflow.python.client import device_lib
def get_available_devices():
    local_device_protos = device_lib.list_local_devices()
    return [x.name for x in local_device_protos]
print(get_available_devices())
# my output was => ['/device:CPU:0']

# good output must be => ['/device:CPU:0', '/device:GPU:0']
```

```
['/device:CPU:0', '/device:GPU:0']
```

TeamMate Responsibilities:

Sr no.	Responsibility	Task	Description

1	Student 1,student 2	Data gathering	Gathering the accurate data
2	Student 1	Data Preparation	Responsible for taking the dataset and performing image processing by controlling some parameters
3	Student 2	Data Preparation	Responsible for splitting data into train and test (80%- 20%)
4	Student 1	Modelling	Initialising the model
	Student 1,student 2	Modelling	CNN-For generating vocabulary
5	Student1	Modelling	Sequence processor for RNN-LSTM
6	Student 2	Modelling	Decoder-Development and deployment
7	Student 2	Modelling	Optimising the model
8	Student 1	Training	Training the model
9	Student 2	Implementation	Implementing the model
10	Student 1	Evaluation	Using test data to find the accuracy of the model(Bleu matric)
11	Student 2	Testing	Final project bug fixing and testing
12	Student 1 & Student 2	Documentation	Preparing the Report and Presentation

4) Results and Discussion

a) GPU running or not check filter

```
#We check if the GPU is available or not
from tensorflow.python.client import device_lib
def get_available_devices():
    local_device_protos = device_lib.list_local_devices()
    return [x.name for x in local_device_protos]
print(get_available_devices())
# my output was => ['/device:CPU:0']

# good output must be => ['/device:CPU:0', '/device:GPU:0']
```

```
['/device:CPU:0', '/device:GPU:0']
```

b) VGG model parameters

```
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
...
Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0
```

c) Features.pkl file generation

```
print('features extracted = ',len(features))
# save to a new file this has the image edited with changed parameters(cleaned image)
# we are making a pickle file
# Reference https://sites.pitt.edu/~naraehan/python3/pickling.html
dump(features, open('features.pkl', 'wb'))#wb=write binary
```

```
features extracted =  8091
```

d) Descriptions.txt file generation

```
lines=list()
for key, desc_list in descriptions.items():
    for desc in desc_list:
        lines.append(key+ ' ' +desc)
data='\n'.join(lines)
file=open('descriptions.txt','w')#opening descriptions.txt with wirte permission
file.write(data)
file.close()
```

e) Max length of characters

```
#print(len(d.split()))
print('Description Length =',lines)
```

```
Description Length = 34
```

f) Actual model generated and its parameters

```
 Layer (type)                Output Shape              Param #     Connected to
==================================================================================================
 input_3 (InputLayer)        [(None, 34)]              0           []

 input_2 (InputLayer)        [(None, 4096)]            0           []

 embedding (Embedding)       (None, 34, 256)           1940224     ['input_3[0][0]']

 dropout (Dropout)           (None, 4096)              0           ['input_2[0][0]']

 dropout_1 (Dropout)         (None, 34, 256)           0           ['embedding[0][0]']

 dense (Dense)               (None, 256)               1048832     ['dropout[0][0]']

 lstm (LSTM)                 (None, 256)               525312      ['dropout_1[0][0]']

 add (Add)                   (None, 256)               0           ['dense[0][0]',
                                                                    'lstm[0][0]']

 dense_1 (Dense)             (None, 256)               65792       ['add[0][0]']

 dense_2 (Dense)             (None, 7579)              1947803     ['dense_1[0][0]']

...
Trainable params: 5,527,963
Non-trainable params: 0
```

g) Final epoch result

```
6000/6000 [==============================] - 409s 68ms/step - loss: 3.3063 - accuracy: 0.3124
6000/6000 [==============================] - 410s 68ms/step - loss: 3.2623 - accuracy: 0.3169
6000/6000 [==============================] - 413s 69ms/step - loss: 3.2100 - accuracy: 0.3206
6000/6000 [==============================] - 413s 69ms/step - loss: 3.1808 - accuracy: 0.3229
6000/6000 [==============================] - 417s 69ms/step - loss: 3.1536 - accuracy: 0.3247
6000/6000 [==============================] - 416s 69ms/step - loss: 3.1351 - accuracy: 0.3270
6000/6000 [==============================] - 416s 69ms/step - loss: 3.1156 - accuracy: 0.3292
6000/6000 [==============================] - 423s 71ms/step - loss: 3.1023 - accuracy: 0.3294
6000/6000 [==============================] - 417s 69ms/step - loss: 3.0950 - accuracy: 0.3307
6000/6000 [==============================] - 427s 71ms/step - loss: 3.0803 - accuracy: 0.3321
6000/6000 [==============================] - 413s 69ms/step - loss: 3.0741 - accuracy: 0.3328
6000/6000 [==============================] - 410s 68ms/step - loss: 3.0686 - accuracy: 0.3332
6000/6000 [==============================] - 418s 70ms/step - loss: 3.0590 - accuracy: 0.3346
6000/6000 [==============================] - 419s 70ms/step - loss: 3.0586 - accuracy: 0.3350
6000/6000 [==============================] - 420s 70ms/step - loss: 3.0557 - accuracy: 0.3354
6000/6000 [==============================] - 421s 70ms/step - loss: 3.0504 - accuracy: 0.3353
6000/6000 [==============================] - 428s 71ms/step - loss: 3.0511 - accuracy: 0.3361
6000/6000 [==============================] - 430s 72ms/step - loss: 3.0468 - accuracy: 0.3370
6000/6000 [==============================] - 437s 73ms/step - loss: 3.0459 - accuracy: 0.3363
6000/6000 [==============================] - 428s 71ms/step - loss: 3.0448 - accuracy: 0.3363
...
6000/6000 [==============================] - 419s 70ms/step - loss: 3.0965 - accuracy: 0.3344
6000/6000 [==============================] - 421s 70ms/step - loss: 3.0984 - accuracy: 0.3351
6000/6000 [==============================] - 420s 70ms/step - loss: 3.1000 - accuracy: 0.3345
6000/6000 [==============================] - 420s 70ms/step - loss: 3.1054 - accuracy: 0.3345
```

f) bleu score

```
BLEU Score 1 =  0.5527533767828469
BLEU Score 2 =  0.26931755251042905
BLEU Score 3 =  0.16980789028605708
BLEU Score 4 =  0.06915533578431837
```

g) New caption

```
1/1 [==============================] - 0s 85ms/step
1/1 [==============================] - 1s 673ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 24ms/step
Generated Caption is : startseq  dog is running on the beach endseq
```

For the image

```
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
Generated Caption is : startseq  two dogs are playing in the grass endseq
```

For the image

```
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
Generated Caption is : startseq  dog is running through the grass endseq
```

For the image

I have attached saved models of 50 run epochs.

5) References

i) http://stackoverflow.com
ii) http://pyimagesearch.com
iii) https://www.educba.com/python-dump/

iv)https://www.tensorflow.org/api_docs/python/tf/keras/utils/load_img
v) https://www.tensorflow.org/api_docs/python/tf/keras/utils/img_to_array
vi) https://sites.pitt.edu/~naraehan/python3/pickling.html
vii) https://towardsdatascience.com/image-feature-extraction-traditional-and-deep-learning-techniques-ccc059195d04
viii) https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8
ix) https://keras.io/examples/vision/image_captioning/
x) https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy
xi) https://keras.io/api/optimizers/
xii) https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/sequence/pad_sequences
xiii) https://www.geeksforgeeks.org/use-yield-keyword-instead-return-keyword-python/

xiv)

xv) [PDF] thecvf.com

xvi) [PDF] thecvf.com