Project 3- Automatic Image Captioning

1. Introduction

In this labwork, a RNN network is implemented in order to tackle the problem of image captioning. In the dataset there are 8000 training and test images along with the text files which contains the filename of an image in the dataset and the corresponding 5 possible captions for that image. In order to frame this problem as supervised learning problem the data points will be as such shown in Figure 1.

In every data point the images' feature vector (obtained from a pretrained CNN model) and a partial caption are inputted and we try to predict the next word in the caption. Since we deal with sequence in this task, employing LSTM is a good choice of model.

	Xi		Yi	
i	Image feature vector	Partial Caption	Target word	
1	Image_1	startseq	the	data points corresponding to image 1 and its caption
2	Image_1	startseq the	black	
3	Image_1	startseq the black	cat	
4	Image_1	startseq the black cat	sat	
5	Image_1	startseq the black cat sat	on	
6	Image_1	startseq the black cat sat on	grass	
7	Image_1	startseq the black cat sat on grass	endseq	
8	Image_2	startseq	the	data points corresponding to image 2 and its caption
9	Image_2	startseq the	white	
10	Image_2	startseq the white	cat	
11	Image_2	startseq the white cat	is	
12	Image_2	startseq the white cat is	walking	
13	Image_2	startseq the white cat is walking	on	
14	Image_2	startseq the white cat is walking on	road	
15	Image_2	startseq the white cat is walking on road	endseq	

Figure 1. The data matrix

2. Approaches

2.1. Preprocess the text data

Firstly from the text file a dictionary containing the mapping of filename and the 5 possible captions is extracted. Then the captions in the dictionary are cleaned by removing punctuation in addition to words such as 'a' and 's. Then we create a vocabulary out of the captions which contains the number of unique words in all captions, which is a total of 8763. However since the model requires to be robust to outliers, the vocabulary is updated to contain only the words that appear 10 or more times in the training data captions. We also save the training captions into a list where each list item(a possible caption for a particular image) starts and ends with identifier words.

```
train_des
  'startseq person repelling on the side of cliff face endseq',
  'startseq person wearing black with white helmet is going down mountainside with water below endseq',
  'startseq extreme climbing over ocean endseq',
  'startseq looking down on someone climbing cliff over water endseq'],
 '2255685792_f70474c6db': ['startseq three individuals are posing on skis behind no skiing sign endseq'
  'startseq three people in skiing gear are standing behind no skiing sign endseq',
  'startseq three people on skis are standing behind no skiing sign endseq',
  'startseq three people standing in the snow behind no skiing sign endseq',
  'startseq three people stand ready to ski next to no skiing sign endseq'],
 '2256138896_3e24b0b28d': ['startseq man does technical rock climbing endseq',
  'startseq man dressed in all black is climbing down mountain supported by cords endseq',
  'startseq man is climbing the side of cliff endseq',
```

Figure 2. The preprocessed captions in training set.

2.2. Preprocess image data

In order to extract the features of images we use pre trained Inception V3 and remove the last layer of softmax from it. We end up with 2048 lengthed feature vectors for each image in both test and training sets. This will be the first part of the input to model.

```
# Do something with the object
print(encoding_train)

{'381239475_044cbffa2b.jpg': array([0.0711422 , 0.2007759 , 0.18541265, ..., 0.24293034, 0.6908936 ,
       0.05522677], dtype=float32), '397982550_cf9f5cdb74.jpg': array([0.2215686 , 0.00874935, 0.2773999
       0.6534449 ], dtype=float32), '398413603_166896900f.jpg': array([0.29930645, 0.4964278 , 0.3204766
       0.12039228], dtype=float32), '3730457171_e66dde8c91.jpg': array([0.41620636, 0.711523  , 0.240669
       0.15430038], dtype=float32), '3747543364_bf5b548527.jpg': array([0.8761105 , 0.08396828, 0.760423
       0.24651234], dtype=float32), '404216567_75b50b5a36.jpg': array([0.13802844, 0.13139889, 0.1445530
       0.4047786 ], dtype=float32), '399679638_d3036da331.jpg': array([0.2643094 , 0.06401033, 0.2412200
       0.37543747], dtype=float32), '384465370_9918873f9a.jpg': array([0.1164654 , 0.2629016 , 0.5503247
       0.404966  ], dtype=float32), '396179143_e1511336e1.jpg': array([0.34354213, 0.22912237, 0.2654663
       0.1477849 ], dtype=float32), '390986651_c801db91a0.jpg': array([0.11959165, 0.2121415 , 0.2734688
       0.34850344], dtype=float32), '405331006_4e94e07698.jpg': array([0.1785418 , 0.5837573 , 0.0990508
       0.04577155], dtype=float32), '378453580_21d688748e.jpg': array([0.18748859, 0.30366   , 0.0982145
       0.2849999 ], dtype=float32), '400562847_e15aba0aac.jpg': array([0.5448617 , 0.6204397 , 0.3130540
```

Figure 3. The word embeddings of captions training set.

2.3. Preprocess the captions

In order to extract the features of images we use pre-trained Inception V3 and remove the last layer of softmax from it. We end up with 2048 lengthed feature vectors for each image in both test and training sets. This will be the first part of the input to model.

2.4. Data Preparation with Generator Function

Since the data is sequential, loading the whole dataset into memory takes a lot of space and using Stochastic Gradient Descent for these kinds of models is preferred. So that at a time a small batch of data can be loaded and processed. In order to do that generator functions are used since they generate a sequence of data at a time rather than all at once. In the generator function used, the partial captions in Figure 9 are replaced with the index correspondence in the vocabulary dictionary created. Then they are padded to the maximum caption length so all are equal sized integer vectors.

2.5. Word Embeddings

In order for the model to process the words they should be embedded to descriptive vectors. For this purpose pre-trained GLOVE model is used. A matrix for caption is created in the form of Figure 1, whereas the words are replaced with the indexes in wordtoid dictionary and ready to input into the model. The data matrix stores word embeddings in the corresponding index.

2.6. The Model

The design of model constructed is shown in Figure 4. The captions are embedded and dropout apple and then fed

to LSTM model. The output is fed to Feed forward neural network aliong with th image feature extracted previously and in one iteration the model predicts the next word in caption.

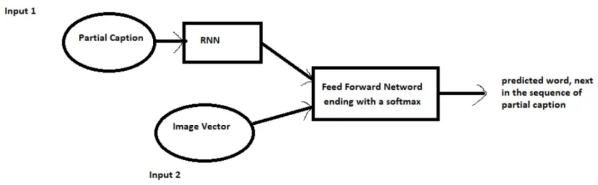

Figure 4. The model structure.

3. Experimental Results

The summary of the model is shown in Figure 5.

```
Model: "model_2"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_6 (InputLayer)            [(None, 34)]         0           []

input_5 (InputLayer)            [(None, 2048)]       0           []

embedding_1 (Embedding)         (None, 34, 200)      330400      ['input_6[0][0]']

dropout_3 (Dropout)             (None, 2048)         0           ['input_5[0][0]']

dropout_4 (Dropout)             (None, 34, 200)      0           ['embedding_1[0][0]']

dense_4 (Dense)                 (None, 256)          524544      ['dropout_3[0][0]']

lstm_1 (LSTM)                   (None, 256)          467968      ['dropout_4[0][0]']

add_1 (Add)                     (None, 256)          0           ['dense_4[0][0]',
                                                                 'lstm_1[0][0]']

dense_5 (Dense)                 (None, 256)          65792       ['add_1[0][0]']

dense_6 (Dense)                 (None, 1651)         424307      ['dense_5[0][0]']

==================================================================================================
Total params: 1,813,011
Trainable params: 1,813,011
Non-trainable params: 0
```

Figure 5. The model summary.

4. Discussion

The model requires two inputs, which are the image vector and LSTM for partial captioning. The images in training dataset are retrieved and fed to a pretrained CNN model named InceptionV3. InceptionV3 does classification hence the last layer of the model is removed in order to obtain the feature vectors for each image in training dataset. For the LSTM part the words in the captions needed to be mapped to their word embeddings. Here GLOVE is used which already has trained on english words. So we select the ones in our vocabularies and map the words to embeddidngs and fed dthem into LSTM model. The output of LSTM along with the feature vector obtained from image are fed into the NN model. The output is the next predicted word in the caption. Hence in order to get the full caption for an image, model has to be trained multiple times. The model in each iteration outputs a vector shaped(none,1652) which is the probability for each word in the vocabulary. The epoch is chosen as 3 and the loss of model in each iteration on

```
<ipython-input-51-e2dca6ee8a15>:3: UserWarning: `Model.fit_generator` is deprecated and
  model.fit_generator(generator, epochs=1, steps_per_epoch=steps, verbose=1)
2000/2000 [==============================] - 1486s 743ms/step - loss: 1.9626
  1/2000 [..............................] - ETA: 13:15 - loss: 2.5311<ipython-input-51-
  model.fit_generator(generator, epochs=1, steps_per_epoch=steps, verbose=1)
2000/2000 [==============================] - 1455s 727ms/step - loss: 1.4616
  1/2000 [..............................] - ETA: 26:08 - loss: 2.5069<ipython-input-51-
  model.fit_generator(generator, epochs=1, steps_per_epoch=steps, verbose=1)
2000/2000 [==============================] - 1275s 637ms/step - loss: 1.1584
<ipython-input-51-e2dca6ee8a15>:3: UserWarning: `Model.fit_generator` is deprecated and
  model.fit_generator(generator, epochs=1, steps_per_epoch=steps, verbose=1)
2000/2000 [==============================] - 722s 361ms/step - loss: 0.9518
2000/2000 [==============================] - 748s 374ms/step - loss: 0.8123
```

Figure 6. The losses in each iteration.

training set is shown in Figure 6. A data generator function is used to train model due to huge number of datapoints. Since the model processes sequential data, it requires multiple iteration where output is appended to input and so on. With data generator the model can be trained with batches of data instead of whole dataset all at once. In order to evaluate the performance of the model, the greedy(image) function is used, which outputs the caption until endseq is reached or the model is compiled at a maximum number of iterations. In Figure 10, it is showed the test image and the corresponding predicted caption for it.

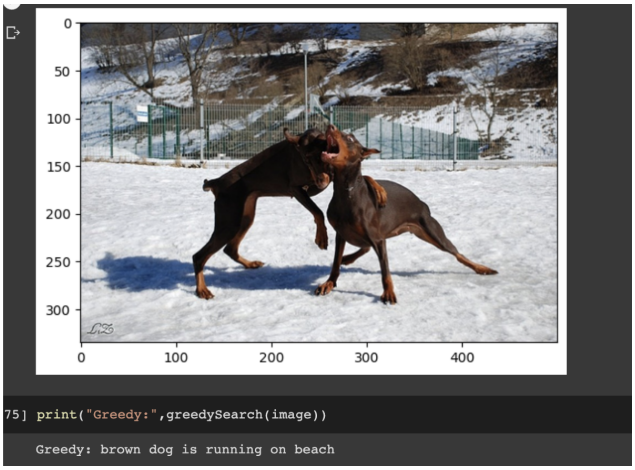

Figure 7. The automatic captioning for the test image.

5. References

1. https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8

2. https://github.com/hlamba28/Automatic-Image-Captioning/blob/master/Automatic