VISUAL RECOGNITION

Mini Project - 2 Technical Report

Dev Patel IMT2018021 Dev.Patel@iiitb.or g

Ishmeet Singh IMT2018030 Ishmeet.Singh@iiit b.org

Prateek Kamboj IMT2018057 Prateek.Kamboj@i iitb.org

Ishaan Sachdeva IMT2018508 Ishaan.Sachdeva@ iiitb.org

Introduction

When we, humans see an image our brain can easily tell what the image is about but can the same thing be done by a computer which takes an image as input and produces relevant outputs? Our project is to build an image captioning application which will use deep learning techniques like Convolutional Neural Network(CNN) and Long Short Term Memory(LSTM) to recognize the context in the image.

Before talking about our approach for the project, we would like to talk about how important this problem is to real world scenarios.

1) Aid to blind - We can create a product for blind which can convert the scene into text and then voice.
2) Automatic Captioning - This can help make Google Image search as good as Google search ie every image could be converted into a caption and search can performed based on caption.

Dataset

We have used the Flickr 8k dataset. It consists of 8000 unique images and each image is mapped to 5 different sentences which describe the image.

Algorithm

1) The first step is to load the descriptions. First we have loaded the file and read the content inside the file into a string then we have created a function which maps the image with a list of its 5 captions.
2) **Cleaning Text:** In the next part we have used different NLP techniques to clean the text. We have removed punctuations, numbers and converted all the text to lowercase
3) **Extracting feature vector from all images:** In this part we have converted every image to a fixed size vector and fed it to a pre-trained neural network. We have used **Xception** model for this which can classify 1000 different classes. We have removed the last layer from this pre-trained model inorder to get fixed length information vector(2048 length vector) for each image.
4) **Loading the images:** In the next part we have loaded the images. First we have created a list of training images. Next we have created a dictionary which maps the image with the feature vectors which were extracted in the previous part. We have added unique words at the beginning and ending of each caption so that the LSTM can identify the starting and ending of every caption
5) **Tokenization of vocabulary**: In this part we have created vocabulary of unique words present across all the 40000 captions and mapped to a unique index value. Our vocabulary has 7577 words
6) **Data generator**: We have to train our model on 6000 images and each image will contain a 2048 length feature vector and the caption is also represented as numbers. This data is not held into memory, we use a generator method that will yield batches.
7) **Defining the CNN-LSTM model**: In this part we have defined the model. Since our input consists of two parts, image vector and caption we have used two different models and later have merged the output of both of them to make the final prediction.

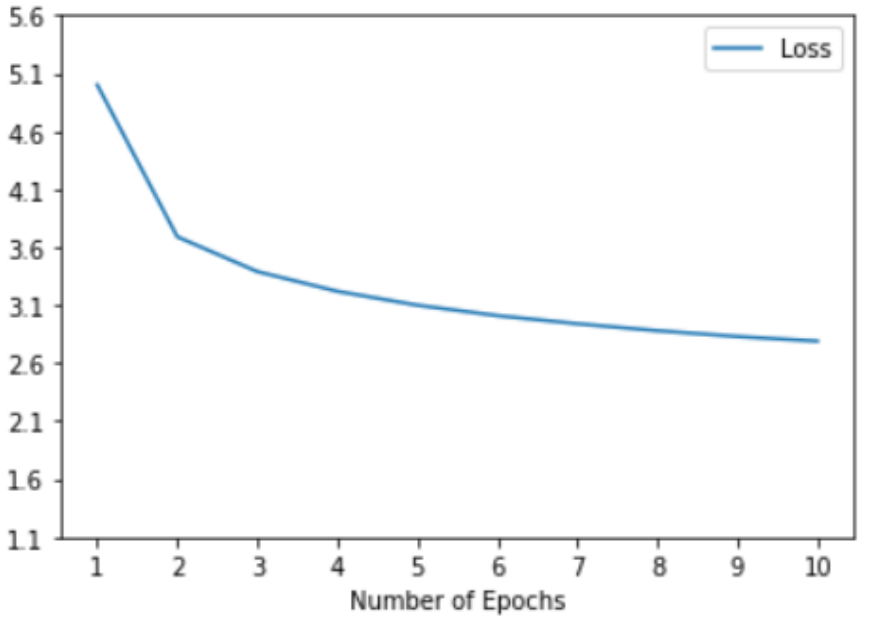
Training, Testing and Observations:

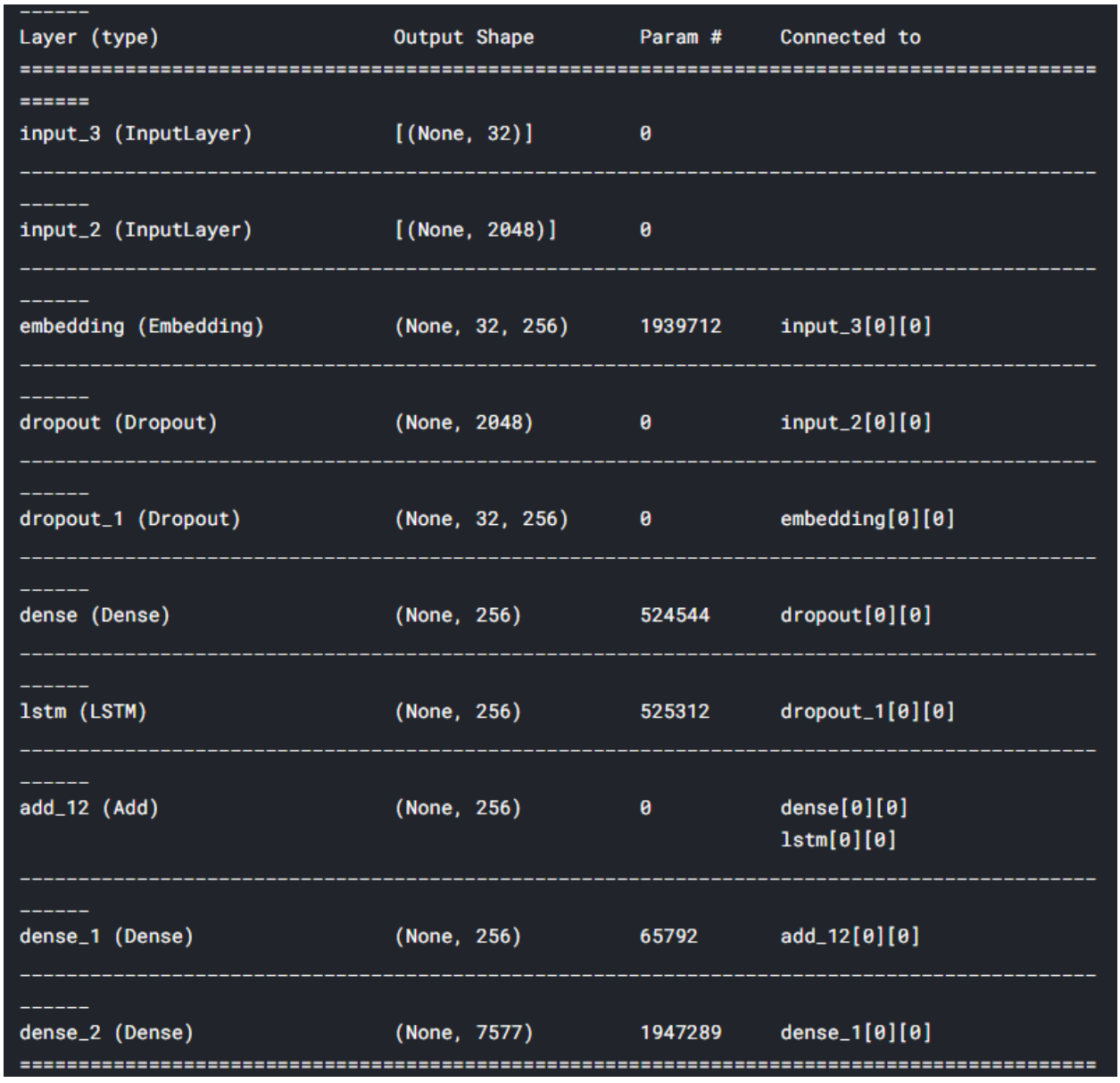
- We took 6000 training images by generating the input and output sequences in batches and fitting them to the model using model.fit_generator() method. Then the model was saved as an output.
- Now the thing to know is the experimentations with the training, we changed layers and activations methods and chose the best output from them and also we performed training on 5 epochs first with epochs in model.fit_generator() taken to be 1. It trained for about 70-80 minutes but the captions generated were small and nowhere near to the expected description.
- Then our best model which was trained for 10 epochs took around 200 minutes to run and generated better results. The train images, train descriptions and train features were all of size 6000. The Vocabulary size was 7577 and the description length was 32.
- As the number of epochs increased, the loss was decreasing but for example the model produced at 20 or 30 epochs didn't seem as good as the model at 10 epochs which we suspected was due to overfitting. Overall our highest running time for a model was at 6 hours 15 minutes which had other variation but the best results we got was at 200 minutes.
- After the model was saved, we loaded the model and generated predictions on the given subjective images. The predictions contain the max length of the index values so we used the same tokenizer.p pickle file to get the words from their index values.

Results and Final Model

★ The final model ran for *12375* second i.e. about *3 hours and 26 minutes* (on 10 epoch)
★ The total trainable parameters were *5,002,649*
★ The model size is 57.3 mb
★ Bleu score 1st- 0.4609, 2nd- 0.2543, 3rd- 0.1402, 4th- 0.0653
★ Loss Values for each epoch:

No. of Epoch	Time for each step(in ms)	Loss
1	107	5.0096
2	111	3.6851
3	111	3.3997
4	112	3.2294
5	108	3.1059
6	111	3.0184
7	112	2.9458
8	109	2.8894
9	109	2.8393
10	108	2.7967





Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	[(None, 32)]	0	
input_2 (InputLayer)	[(None, 2048)]	0	
embedding (Embedding)	(None, 32, 256)	1939712	input_3[0][0]
dropout (Dropout)	(None, 2048)	0	input_2[0][0]
dropout_1 (Dropout)	(None, 32, 256)	0	embedding[0][0]
dense (Dense)	(None, 256)	524544	dropout[0][0]
lstm (LSTM)	(None, 256)	525312	dropout_1[0][0]
add_12 (Add)	(None, 256)	0	dense[0][0] lstm[0][0]
dense_1 (Dense)	(None, 256)	65792	add_12[0][0]
dense_2 (Dense)	(None, 7577)	1947289	dense_1[0][0]

This picture represents the structure of our model layers

❖ OUTPUT ON SUBJECTIVE IMAGES

The next part contains the various outputs that were obtained on the subjective images in order image 1 to 5 respectively.

1. Man in red shirt is standing in front of brick wall

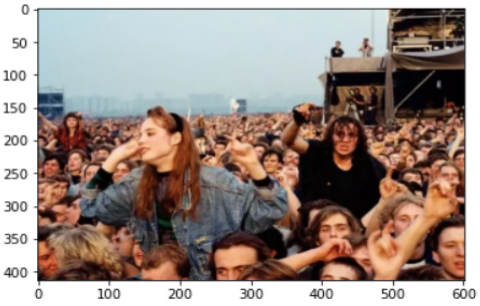
```
start man in red shirt is standing in front of brick wall end
[14]: <matplotlib.image.AxesImage at 0x7fa3e4f88550>
```

2. Man in red shirt is riding bike down the street

```
start man in red shirt is riding bike down the street end
[17]: <matplotlib.image.AxesImage at 0x7fa3dc0c25d0>
```

3. Man in red shirt is standing in front of crowd

```
start man in red shirt is standing in front of crowd end
[18]: <matplotlib.image.AxesImage at 0x7fa3c49211d0>
```



4. Two people are standing in front of the water

```
start two people are standing in front of the water end
[22]: <matplotlib.image.AxesImage at 0x7fa3961dcd10>
```

5. Brown dog is running through the water

```
start brown dog is running through the water end
[23]: <matplotlib.image.AxesImage at 0x7fa3c4acea90>
```