Automatic Image Captioning

Topics of Deep Learning (UE17CS338)

Project Report

Submitted by,

Abhinandan Singla	PES1201701128
Arpan Ghoshal	PES1201700240
Abhijeet Murthy	PES1201700139

Project Abstract and Scope:

Image Captioning is the process of generating a textual description of an image. It can be divided into two modules logically – one is an image-based model – which extracts the features and nuances out of our image, and the other is a language-based model – which translates the features and objects given by our image-based model to a natural sentence.
It can be used for **aid to the blind**, **CCTV cameras**, in **self-driving cars** and in **reverse image search**. This problem was well researched by Andrej Karapathy in his PhD thesis [1]. We tried to implement methods described in the paper with some other references combined for better results.

Solution Architecture:

Let, two images (img_1,img_2) are taken into consideration for the training set and one image as the testing set (img_3) with captions respectively(cap_1, cap_2 and cap_3).

First, we have converted the training images as a feature vector which will further feed to our neural network. We did this by transfer learning. We used the InceptionV3 model to get fixed length information of each image for automatic feature extraction.
Here, we have removed the last softmax layer to extract the image vector which was used to perform 1000-class classification as a 2048 length vector.

Now, we built the vocabulary of the caption. In the original training dataset, we have 8763 words but we have to choose the words which appear more than 10 times which makes around 1652 words. This helps the model become more robust to outliers and make fewer mistakes.

We then added two tokens <start_seq> and <end_seq> in the captions. So our img_1 and img_2 now have respective captions and vocabulary.

cap_1 -> “start_seq the black cat sat on grass end_seq”
cap_2 -> “start_seq the white cat is walking on road end_seq”
vocab = {black, cat, end_seq, grass, is, on, road, sat, start_seq, the, walking, white}

We gave an index to each word. So now our vocab becomes:

vocab = black -1, cat -2, end_seq -3, grass -4, is -5, on -6, road -7, sat -8, start_seq -9, the -10, walking -11, white -12

Now we framed it as a supervised learning problem where we have a set of data points D = {Xi, Yi}, where Xi is the feature vector of data point ‘i’ and Yi is the corresponding target variable. [2] Let’s take the first image vector img_1 and its corresponding caption:
“start_seq the black cat sat on grass end_seq”.

Recall that, Image vector is the input and the caption is what we need to predict. For the first time, we provide the image vector and the first word as input and try to predict the second word and this process is continued till we get the end_seq. [3] But we are passing indexes of the words with zero paddings, which will help in batch processing. Now, we mapped each index to a 200-long vector using a pre-trained GLOVE word embedding model.

Let’s compute the size of the data matrix.

No. of data points = No. of the actual image training dataset * Each having 5 captions * Words in each caption (avg) = 6000 * 5000 * 7 = **210000**

Length of each data point = Length of image vector(2048) + Length of partial caption (34 indices * 200 glove vector) = 2048 + 6800 = 8848

Finally, size of data matrix= 210000 * 8848= 1858080000 blocks. Assuming each block as 2 bytes, then to store data 3GB of the main memory is required.

So, we have used Stochastic Gradient Descent where we calculate the loss on a batch of data points to update the gradients with the help of generators.

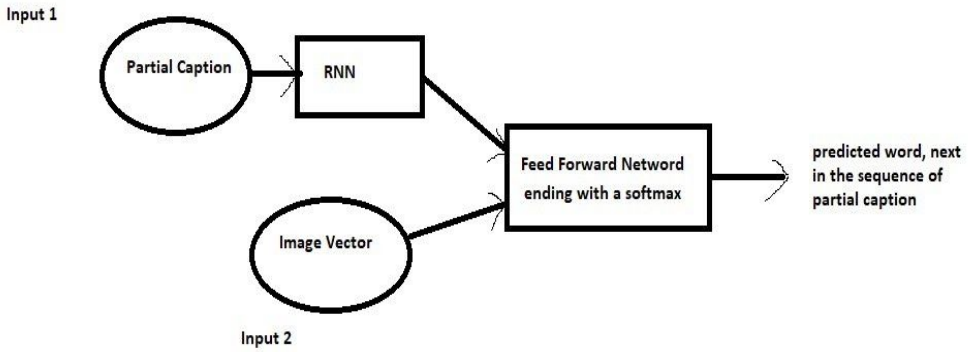
Finally, our matrix will become like this:

	Xi		Yi
i	Image feature vector	Partial Caption	Target word
1	Image_1	[9, 0, 0 …., 0]	10
2	Image_1	[9, 10, 0, 0 …., 0]	1
3	Image_1	[9, 10, 1, 0, 0 …., 0]	2
4	Image_1	[9, 10, 1, 2, 0, 0 …., 0]	8
5	Image_1	[9, 10, 1, 2, 8, 0, 0 …., 0]	6
6	Image_1	[9, 10, 1, 2, 8, 6, 0, 0 …., 0]	4
7	Image_1	[9, 10, 1, 2, 8, 6, 4, 0, 0 …., 0]	3
8	Image_2	[9, 0, 0 …., 0]	10
9	Image_2	[9, 10, 0, 0 …., 0]	12
10	Image_2	[9, 10, 12, 0, 0 …., 0]	2
11	Image_2	[9, 10, 12, 2, 0, 0 …., 0]	5
12	Image_2	[9, 10, 12, 2, 5, 0, 0 …., 0]	11
13	Image_2	[9, 10, 12, 2, 5, 11, 0, 0 …., 0]	6
14	Image_2	[9, 10, 12, 2, 5, 11, 6, 0, 0 …., 0]	7
15	Image_2	[9, 10, 12, 2, 5, 11, 6, 7, 0, 0 …., 0]	3

Model Architecture:

We created merge models while freezing some layers and finally optimizing it with adam.

This the brief architecture which contains the high-level sub-modules:



This is the model summary:

```
model.summary()
```

Layer (type)	Output Shape	Param #	Connected to
input_4 (InputLayer)	(None, 34)	0	
input_3 (InputLayer)	(None, 2048)	0	
embedding_2 (Embedding)	(None, 34, 200)	330400	input_4[0][0]
dropout_3 (Dropout)	(None, 2048)	0	input_3[0][0]
dropout_4 (Dropout)	(None, 34, 200)	0	embedding_2[0][0]
dense_2 (Dense)	(None, 256)	524544	dropout_3[0][0]
lstm_2 (LSTM)	(None, 256)	467968	dropout_4[0][0]
add_2 (Add)	(None, 256)	0	dense_2[0][0] lstm_2[0][0]
dense_3 (Dense)	(None, 256)	65792	add_2[0][0]
dense_4 (Dense)	(None, 1652)	424564	dense_3[0][0]

```
Total params: 1,813,268
Trainable params: 1,813,268
Non-trainable params: 0
```

Where the inputs are Partial Caption and Image feature vector and output is an appropriate word, next in the sequence of partial caption provided in the inputs. [4]

Then lower the learning rate and increased the batch size as SGD was used. Lastly, we implement a greedy search in the interface and we get the desired outputs with the maximum probability of the words after iterating to the model till we encounter <end_seq> and when we get the maximum threshold of words generated.

Constraints, Assumptions & Dependencies:

We must understand that the images used for testing must be semantically related to those used for training the model. For example, if we train our model on the images of cats, dogs, etc. we must not test it on images of aeroplanes, waterfalls, etc. If the distribution of train and test sets are different the model will not give the desired performance.

Future Work Plan:

- Using a larger dataset.
- Changing the model architecture, e.g. include an attention module.
- Doing more hyperparameter tuning (learning rate, batch size, number of layers, number of units, dropout rate, batch normalization etc.).
- Use the cross-validation set to understand overfitting.
- Using Beam Search instead of Greedy Search during Inference.
- Using BLEU Score to evaluate and measure the performance of the model.

References

[1] A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 664-676, 1 April 2017.

[2] Tanti, Marc & Gatt, Albert & Camilleri, Kenneth. (2017). What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?. 10.18653/v1/W17-3506.

[3] TANTI, M., GATT, A., & CAMILLERI, K, "Where to put the image in an image caption generator," in Natural Language Engineering, 467-489, 12 March 2018

[4] O. Vinyals, A. Toshev, S. Bengio and D. Erhan, "Show and tell: A neural image caption generator," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, 2015, pp. 3156-3164.