MINI-PROJECT REPORT

IMAGE CAPTIONING

May 23, 2021

GROUP CODE : **SMAH**
Hruthik Chevuri(IMT2018507)
Madhav Pasumarthy(IMT2018524)
Akshath Kaushal(IMT2018501)
Shanthan Reddy(IMT2018522)

1 Abstract

This report briefly describes and summarizes our mini project image captioning. We have designed a CNN-LSTM system for generating captions for images. This report contains our approach towards the model, BLEU score evaluation on the test dataset and the captions that are genearated for the given subjective images by our model .

2 Introduction

This neural network based image caption generator is implemented upon the Keras library. We have used Convolutional Neural Network(CNN) as an encoder and an LSTM-based Recurrent Neural Network as a decoder. Our implementation has achieved a decent accuracy.

Training machine learning models to automatically generate captions for images is a very challenging task in the fields of Computer Vision and Machine learning. This task is a combination of Image understanding, feature extraction and conversion of visual representation into natural languages. With many advancements in Neural networks, CNNs and RNNs can be effectively used to perform this task to achieve accurate results. This project can greatly help visually impaired people to better understand the context of any image on the internet.

2.1 Dataset

The Flickr8k dataset has 8000 images with each image being labelled with 5 different descriptions. The provided dataset comes with a lemmatized version of text data along with train,validation and test subsets.
Since we are using the pre-trained ResNet model, first our input images are converted into 3 channel RGB images of shape (3 x 224 x 224). After resizing the images, we construct a dictionary of words i.e a corpus of most frequently occurred words in captions. Then all these words are vectorized.

3 Architecture

We have used the Flickr8k dataset for training our model. We have used the ResNet pre-trained Convolutional Neural Network model as an encoder to extract and encode the image features into a higher dimensional feature space. An LSTM-based Recurrent Neural Network is used as a decode to convert the encoded features.

3.1 CNN encoder(Inception v3)

A Convolutional Neural Network is a Deep Learning algorithm that can take in an input, calculate weights and biases for various objects in the image and be able to differentiate one from another.Inception-v3 is widely used model and it ha attained more than 78% accuracy on Image net dataset.

Convolutional Layer: This layer requires a few components which are input data, filter or a feature detector and a feature map. The filter size is typically a 3x3 matrix. The filter is then applied to an area of the image, and a dot product is calculated between the input pixels and the filter. This dot product is then fed into an output array. Afterwards, the filter shifts and repeats the process until the kernel has swept across the entire image. The final output from the series of dot products from the input and the filter is known as a feature map, activation map, or a convolved feature.

Pooling Layer: Pooling layer conducts dimensional reduction. Similar to the convolutional layer, the pooling operation sweeps a filter across the entire input, but the difference is that this filter does not have any weights. In Max pooling, as the filter moves across the input, it selects the pixel with the maximum value to send to the output array. In Average Pooling, as the filter moves across the input, it calculates the average value within the receptive field to send to the output array.

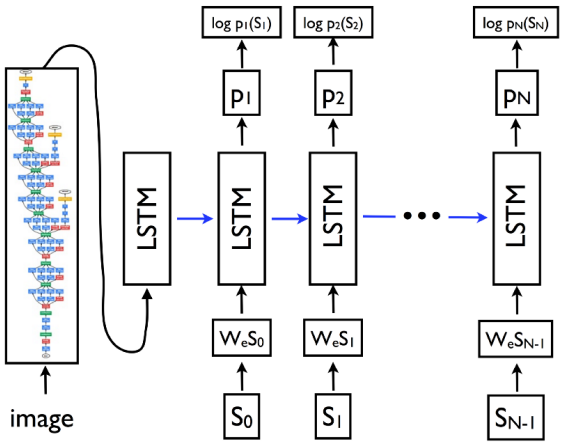
Fully connected Layer: This layer performs the task of classification based on the features extracted through the previous layers and their different filters. While convolutional and pooling layers tend to use ReLu functions, FC layers usually use a softmax activation function to classify inputs appropriately, producing a probability from 0 to 1. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object.

For our task, the encoder needs to extract image features of various sizes and encode them into vector space which will be sent to a RNN in later stages. Since our CNN need not classify images but encode features, we removed the fully connected layers and the max pool layers at the end of the network. As we can see from the above picture, the circled layers are removed from our architecture.

3.2 LSTM decoder

Long short-term memory networks are an extension for recurrent neural networks, which basically extends the memory.LSTMs enable RNNs to remember inputs over a long period of time. This memory can be seen as a gated cell. In an LSTM you have three gates: input, forget and output gate. These gates determine whether or not to let new input in (input gate), delete the information because it isn't important (forget gate), or let it impact the output at the current timestep (output gate).

For our task, the decoder needs to generate image captions word by word using LSTMs. The input for the decoder is the encoded image feature vectors from CNN and the encoded image captions produced in the data preprocessing stage. In each iteration of the LSTM network, we concatenate the embedded captions of all previous words and the encoded images and feed them to the LSTM to get the next state of LSTM. Then, fully connected layers which uses the softmax activation function can predict the probabilities of current word embedding based on the current state and append it to the word embedding prediction matrix.



Loss function The nature of our RNN output is a series of words' occurrences, and in order to increase the quality of the RNN output, we use Cross Entropy Loss. This is the most effective measurement for the performance of a classification model whose output is a probability value between 0 and 1. We have used **Adam optimizer** which is an adaptive learning rate optimization algorithm that has been designed specifically for training deep neural networks.

3.3 Word embeddings

Here we need to convert the captions for training images into embedded captions, for that we need to create word embeddings. we can create either by using the corpus of data we have as captions or either we can use pretrained word embeddings like glove. Here we tried our image captioning using **glove embeddings**.

4 Beam search

Decoding the most likely output sequence involves searching through all the possible output sequences based on their likelihood. A popular approach in such a case is to use Beam Search. The algorithm is a best-first search algorithm which iteratively considers the set of the k best sentences up to time t as candidates to generate sentences of size t + 1, and keep only the resulting best k of them, because this better approximates the probability of getting the global maximum.

We tried a few Beam search sizes and the BLEU score came out to be better under a size of 5. Hence we set our Beam search size to 3 and also 5.

5 Model size constraint

We have initially sent the image into inception v3 model and then the word embeddings are taken from pretrained glove. we also added dropout layer to avoid overfitting and then to dense layer. The output of the dense layer is connected to the LSTM. and added two linear dense layers at the end which gives the captions for the input image.

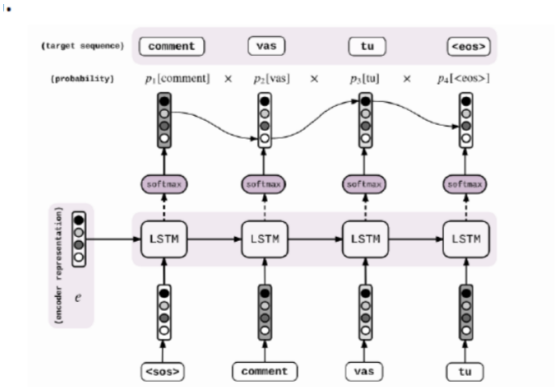
input_3 (InputLayer)	[(None, 38)]	0	
input_2 (InputLayer)	[(None, 2048)]	0	
embedding (Embedding)	(None, 38, 200)	270200	input_3[0][0]
dropout (Dropout)	(None, 2048)	0	input_2[0][0]
dropout_1 (Dropout)	(None, 38, 200)	0	embedding[0][0]
dense (Dense)	(None, 256)	524544	dropout[0][0]
lstm (LSTM)	(None, 256)	467968	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, 1351)	347207	dense_1[0][0]

Total params: 1,675,711
Trainable params: 1,675,711
Non-trainable params: 0

The model has approximately 1.7 million parameters and the wights take upto 7MB space.

6 Training the model and experiments

We have given the train image and test image names in txt files. so according to the names in the files, those images from taken from the images folder and encoded them and saved the encoded images into train and test image encodings respectively.We saved these encodings into .pkl files which will be easier to load them for next time. We gave these training image encodings to our described model for training and glove word embeddings were used.



We trained our model for 30 epochs with categorical loss entropy as loss function and adam optimizer with a batch size of 3. We tried out these parameters and save the model which will be used to run the captioning next time without starting from training, we can load the model weights next time.

we then experimented then for image captioning with **beam search prediction**. After using beam search, the scores are improved to an extent. We tried out beam searcgh for different values of 'k'. we got improved results when we have taken k = 3 or 5. So beam search is preferred over argmax function at the end.

7 Evaluation of test data using BLEU

Full form of BLEU is bilingual evaluation understudy. BlEU's output will be a number between 0 and 1. The score indicates how similar is the given text with respect to the reference text with values given closer to 1 representing more similar texts.A perfect score is not possible in practice as a translation would have to match the reference exactly. This is not even possible by human translators.NLTK provides the sentence-bleu() function for evaluating a candidate sentence against one or more reference sentences.

we have 5 different captions for each test image provided. so these 5 captions are given as a reference array and the caption given by the system has been passed into this function to evaluate bleu score.

Scores:
we have generated the captions initially using the word embeddings created from the corpus of caption data and also with glove embedding.

using the embedding created —— Score is 0.42
using glove embedding —— Score is 0.391 approx

Then we experimented by comparing them with the scores obtained from beam search prediction

Argmax search —— Score is 0.391
Beam search with k = 3 —— Score is 0.46
Beam search with k = 5 —— Score is 0.452 approx

8 Conclusion

i)Here from the first comparision , we can see that the score obtained by using glove embedding is less compared to other one because glove embedding consists of entire wikipedia corpus.These test images will contain many unrelated words in that embedding. Here the test images are similar to that of training images. So using the corpus created from caption data performed better compared to glove.

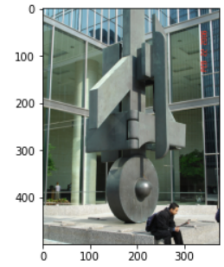
ii) But the model using the glove embeddings will produce better captions for all images in general. So it will give better results on subjective images.

iii) Using beam search prediction will help in generating captions slightly better.

iv) Due to memory constrained given, the model is restricted with performance. If we use VGG 16 model which will have approximately 134 M parameters, then it may provide better captions and BLEU score.

8.1 Subjective images

These are the captions generated for the 5 subjective images using argmax search, beam search with k =2,3,5.

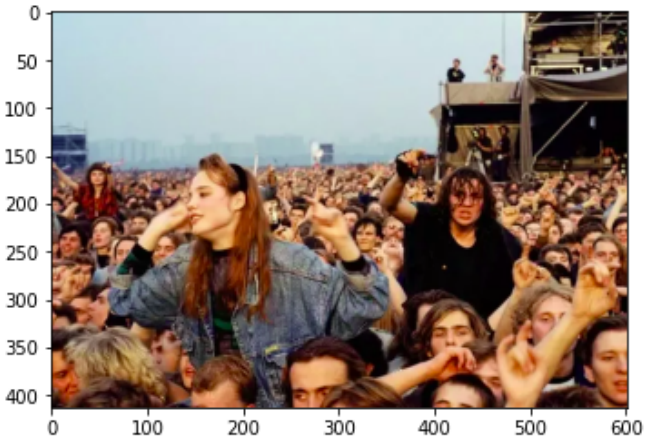


```
a boy in a red shirt be jump off a rock into a lake
Beam Search with K = 2: a boy in a red shirt be jump off a rock into a lake
Beam Search with K = 3: a boy in a red shirt be jump on a bed
Beam Search with K = 5: a little boy in a white shirt and blue jean be climb a rock wall
```

```
a group of person be stand in a street
Beam Search with K = 2: a group of child be play soccer in front of a building
Beam Search with K = 3: a group of person walk down a street
Beam Search with K = 5: a group of person walk down a street
```

```
a little girl in a white dress be run along a dirt road
Beam Search with K = 2: a little girl in a pink tutu jump on a trampoline
Beam Search with K = 3: a little girl in a white bathe suit dance in front of a kiddie pool
Beam Search with K = 5: a little girl in a white bathe suit dance in front of a pool
```

```
a little girl in a pink tutu be climb a rock wall
Beam Search with K = 2: a little girl in a white dress be blow bubble in a wooded area
Beam Search with K = 3: a little girl in an orange swimsuit be climb a rock wall
Beam Search with K = 5: a little girl in a yellow dress be climb a rock wall
```



```
group of kid be play in firework
Beam Search with K = 2: group of kid be stand in front of the sun
Beam Search with K = 3: group of kid be stand in front of large tree
Beam Search with K = 5: group of kid be pose for picture
```

9 References

[1] BLEU SCORE calculation. url: https://machinelearningmastery.com/calculate-bleu-score-for-text-python/

[2] Image Captioning in Deep Learning. url:https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2

[3] Inception v3 architecture. url:https://towardsdatascience.com/a-simple-guide-to-the-ver

[4] Beam search. url:https://machinelearningmastery.com/beam-search-decoder-natural-lan

[5] GloVe: Global Vectors for Word Representation Jeffrey Pennington, Richard Socher, Christopher D. Manning. url:https://nlp.stanford.edu/projects/glove/