Individual Final Project Report

Hemanth Koganti

Introduction

This project aims to achieve Image Captioning using Deep Learning Techniques, mainly Convolutional Neural Networks and Long Short-Term Memory. The dataset used for this project is the Flickr8k dataset, found easily on Kaggle. Flickr8k, and others like it, such as the COCO dataset, have created the benchmark for sentence-based image captioning.

The scope of the project incorporates different domains under Artificial Intelligence such as Computer Vision, Deep Learning, and Natural Language Processing. Applications and business value of this project include:

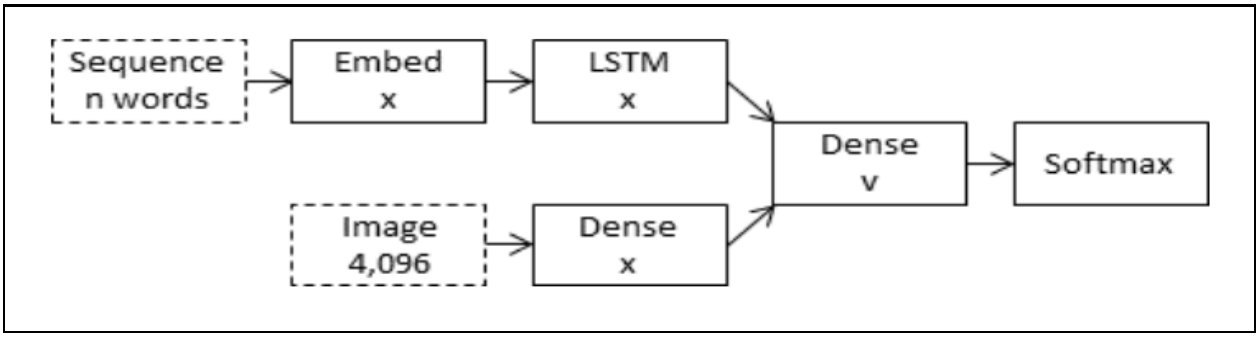
- Image Search in Search Engines
- Image Segregation and Classification
- Text-to-Speech to aid visually impaired individuals
- Automatic Image annotation in Facial Recognition, E-commerce, etc.

The project is divided into 5 main components:

1. Data Splitting – generating the train and test ids of the images
2. Text Preprocessing – preprocessing the captions before being fed into the LSTM model
3. Image Preprocessing – generating image features using CNN models
4. LSTM – Caption generator – generates captions
5. Model Evaluation – using various performance metrics

Brief Explanation of my work

Experimental Setup



We tackled this problem using an Encoder-Decoder model. Here our encoder model will combine both the encoded form of the image and the encoded form of the text caption and feed to the decoder. Our model will treat CNN as the 'image model' and the RNN/LSTM as the 'language model' to encode the text sequences of varying length. The vectors resulting from both the encodings are then merged and processed by a Dense layer to make a final prediction.

To encode our image features, we made use of transfer learning. We used VGG-16, InceptionV3, ResNet pre-trained models. Out of these, VGG-16 gave better results compared to others.

To encode our text sequence we will map every word to a 200-dimensional vector. For this will use a pre-trained Glove model. This mapping will be done in a separate layer after the input layer called the embedding layer. Glove Embedding basically maps words to a vector space, where similar words are clustered together and different words are separated. The advantage of using Glove over Word2Vec is that GloVe does not just rely on the local context of words but it incorporates global word co-occurrence to obtain word vectors. For our model, we will map all the words in our 31-word long caption to a 200-dimension vector using Glove.

We created a Merge model where we combined the image vector and the partial caption. Therefore our model will have 3 major steps:

1. Processing the sequence from the text
2. Extracting the feature vector from the image
3. Decoding the output using softmax by concatenating the above two layers

Summary of our model is attached below:

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	[(None, 31)]	0	
input_1 (InputLayer)	[(None, 4096)]	0	
embedding (Embedding)	(None, 31, 200)	345200	input_2[0][0]
dropout (Dropout)	(None, 4096)	0	input_1[0][0]
dropout_1 (Dropout)	(None, 31, 200)	0	embedding[0][0]
dense (Dense)	(None, 256)	1048832	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, 1726)	443582	dense_1[0][0]

Total params: 2,371,374
Trainable params: 2,026,174
Non-trainable params: 345,200

Input_2 is the partial caption of max length 31 which is fed into the embedding layer. This is where the words are mapped to the 200-d Glove embedding. It is followed by a dropout of 0.5 to avoid overfitting. This is then fed into the LSTM for processing the sequence.

Input_1 is the image vector extracted by our VGG-16 network. It is followed by a dropout of 0.5 to avoid overfitting and then fed into a Fully Connected layer.

Both the Image model and the Language model are then concatenated by adding and fed into another Fully Connected layer. The layer is a softmax layer that provides probabilities to our 1726 word vocabulary.

We compiled the model using Categorical_Crossentropy as the Loss function and Adam as the optimizer with a learning rate of 0.001. We tried different optimizers with different learning rates and Adam with a learning rate of 0.001 gave best results. Since our dataset has 6468 images and 32340 captions we created a function that can train the data in batches.

Next, we trained our model for 15 epochs with a batch size of 3 and we saved the model at each epoch. Initially, we trained the model using conventional batch sizes of 16,32,64,128 and then we trained with much smaller batch sizes such as 1,3,5. Smaller batches yielded better results and a batch size of 3 gave best results.

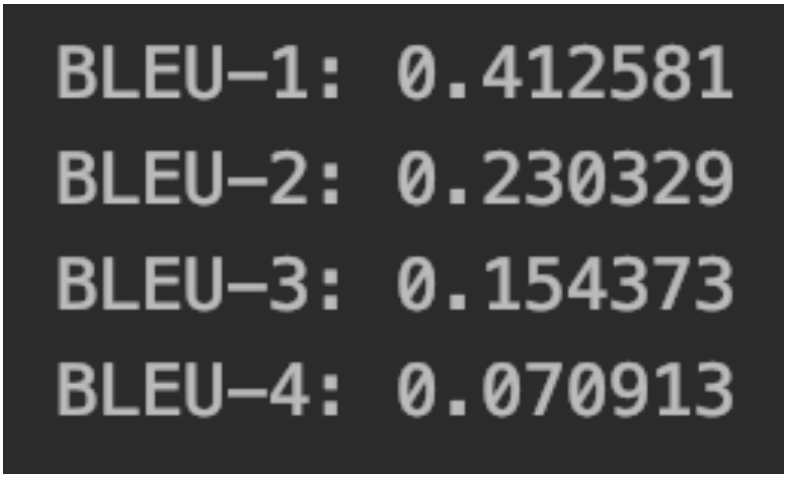
Results

As the model generates a 1725 long vector with a probability distribution across all the words in the vocabulary we greedily pick the word with the highest probability to get the next word prediction. This method is called Greedy Search. Below are some of the captions generated by our model for different images.

dog is jumping into the air to catch a fr

	man in red shirt is climbing up waterf
	A surfer in blue shorts surfs over wa

BLEU score metric was used to evaluate performance of our model. The Bilingual Evaluation Understudy Score, or BLEU for short, is a metric for evaluating a generated sentence to a reference sentence. A perfect match results in a score of 1.0, whereas a perfect mismatch results in a score of 0.0. The approach works by counting matching n-grams in the candidate translation to n-grams in the reference text, where 1-gram or unigram would be each token and a bigram comparison would be each word pair. The comparison is made regardless of word order. The counting of matching n-grams is modified to ensure that it takes the occurrence of the words in the reference text into account, not rewarding a model generated text that generates an abundance of reasonable words. Below are the cumulative BLEU scores when predicted on our test set.



Summary & Conclusion

Our model gave a decent BLEU score of 0.41 and got pretty good captions for certain images. While doing this project, I learned how to incorporate the field of Computer Vision and Natural Language Processing together to make predictions. And, also I have learned architectures of different pre-trained models. One of the main improvements that can be made to make the model more robust is by adding additional images and captions by trying the Flickr30K dataset which has a lot of images and captions. And also, we can make use of the beam search instead of greedy search at the output softmax. Beam Search is where we take top k predictions, feed them again to the model and then sort them using the probabilities returned by the model.

Codes Used / Modified percentage – 85 lines of code (found online), 12 lines of code (modified), 30 lines of code (added) Total = 76%

References

1. Learning CNN-LSTM Architectures for Image Caption Generation
2. Image Captioning
3. Show and Tell: A Neural Image Caption Generator
4. A Hierarchical Approach for Generating Descriptive Image Paragraphs
5. VizSeq: A Visual Analysis Toolkit for Text Generation Tasks
6. https://neurohive.io/en/popular-networks/vgg16/
7. https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8
8. https://cloud.google.com/tpu/docs/inception-v3-advanced
9. Recurrent Neural Network Regularization
10. Image Captioning - A Deep Learning Approach
11. https://hagan.okstate.edu/NNDesign.pdf
12. https://kharshit.github.io/blog/2019/01/11/image-captioning-using-encoder-decoder
13. https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2
14. https://colah.github.io/posts/2015-08-Understanding-LSTMs/