Image Caption Generation using LSTMs and Convolutional Neural Networks

Project Team Number: 21
Teaching Assistant: Christopher Gomes
Team Members: Mrinalini Injeti Ramakanth, Omkar Reddy Gojala

Problem Description

Image Caption Generation involves predicting a sequence of words(sentence) from an input image. It is essentially an advanced **image classification** problem with a lot of potential applications like:

- Aiding visually impaired people who rely on sounds and texts to describe a scene
- Targeted marketing on social media applications like Facebook and Instagram
- A slightly (not-so) long term use case would be explaining what happens in a video, frame by frame

We were motivated to select this project after reading through the work of Andrej Karpathy[0] and Marc Tanti[1]. The idea of generating a description given an image was fascinating to us.

Dataset

We are using Flickr8K dataset for image caption generation available in Kaggle[2]. Flickr8K.zip comprises of two folders:

- Flickr8K_Images: Contains a total of 8000 jpeg images of different shapes and sizes. We are using 6000 images for training, 1000 images for testing and 1000 images for validation
- Flickr8K_TextData: Contains text files describing the images in the train, validation and the test sets. Each image has a total of 5 captions i.e. a total of 40000 captions

Captions:
a beagle and a golden retriever wrestling in the grass
Two dogs are wrestling in the grass
Two puppies are playing in the green grass.
two puppies playing around in the grass
Two puppies play in the grass

Figure 1

The dataset consists of images gathered from different Flickr groups, so the Flickr8K dataset contains a variety of images depicting scenes and situations. The images are currently of different dimensions i.e. 415x502, 500x332, 333x500 etc.

Approach and Methodology

In this project we deal with two types of data namely text and images. Various pre-processing steps are applied to the captions and images as discussed below:

Text pre-processing

- Tokenized each caption and converted all the words to lower-case so that the model would not treat words like “Hello” and “hello” differently
- Removed alpha-numeric characters as they won’t hold much significance in the data
- Removed punctuation marks from the description, as predicting even the punctuation correctly would be a deviation from the main motive of this project and would bring in linguistic complexities which can be treated as a different project
- Hanging letters are eliminated after removing punctuation marks like apostrophe, for example “The building’s height” becomes “The building s height”
- Added two tokens namely startseq and endseq as the prefix and postfix for each caption respectively, to help our model recognize the start and end of each description during training and predicting phases
- Filtered out unique words from the training captions and represented each word by a unique integer

Since all descriptions are of variable lengths, we calculated the maximum length description which was 34 in our case. While creating a word vector for each description we map each word with its corresponding integer value and pad zeros to the remaining length of vector to convert it into a fixed length vector of size 34.

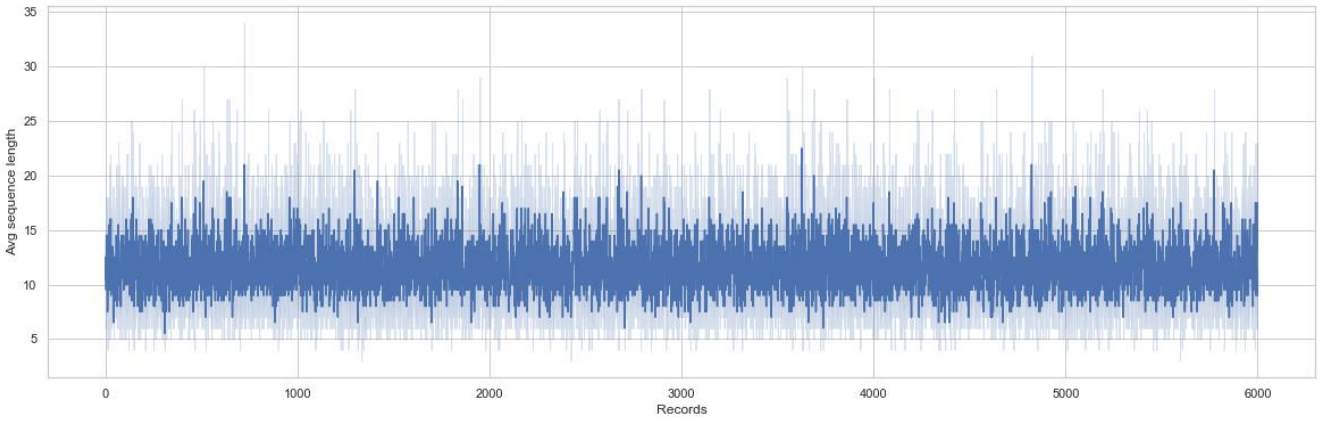


Figure 2: Average description length for each image

Image pre-processing and feature vector generation

Since the input to the neural network should be an image vector, we first resized all the images to a fixed dimension of 299x299x3 using OpenCV, and then converted each image into a fixed-length vector by employing transfer learning. We used a pre-trained InceptionV3 CNN model and VGG16 CNN model for comparison purposes.

InceptionV3 model was trained on Imagenet dataset for the purpose of classification. Since our aim is to generate a fixed-length vector of size 2048, we removed the last softmax layer from the pretrained model and extracted a 2048 length vector for each image. Each image along with its corresponding feature vector is stored in a pickle file. We have maintained 3 different pickle files for training, validation and test sets, for computational efficiency.

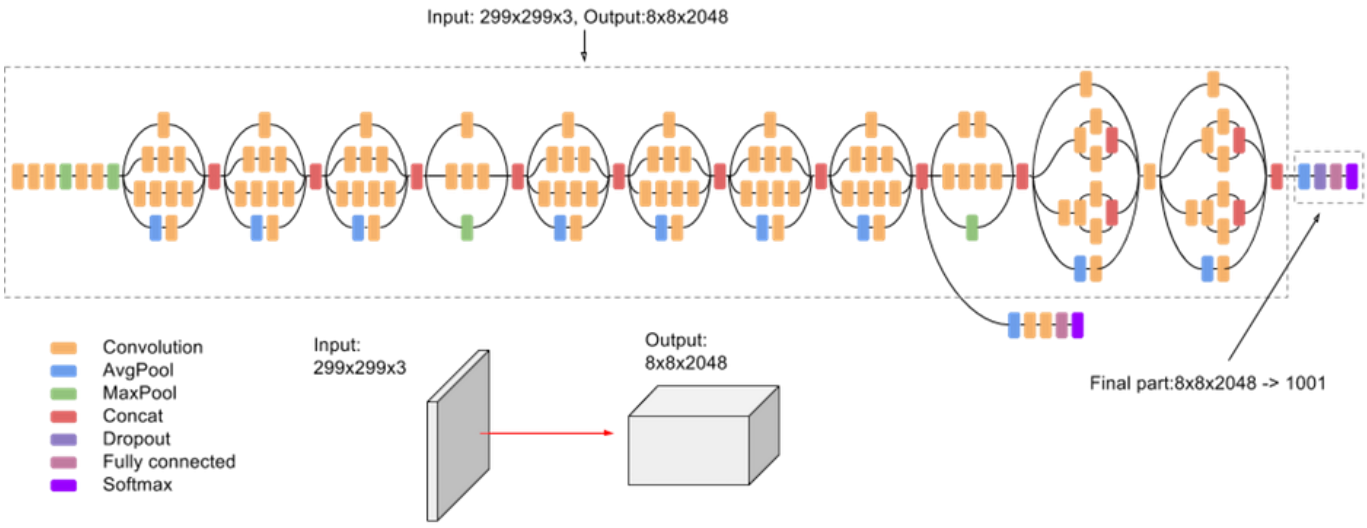


Figure 3: InceptionV3 architecture

Similarly, we generated a fixed-length vector of size 4096 using a pretrained VGG16 model and stored the results in pickle files.

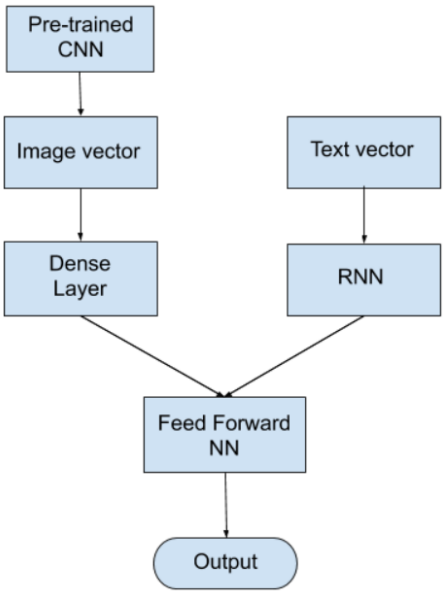
Sequential Caption Injection

Image	Partial Caption	Target Word
Image	startseq	a
Image	startseq a	young
Image	startseq a young	boy
.....	.....	.....
Image	startseq a young boy wearing a helmet and riding a bike in a park	endseq

Table 1: Training sequences

For each image we will train the model by temporally injecting incremental sequences of the description. We split each image-description pair into multiple such incremental sequences. The first sequence uses input description as 'startseq' and first word in the description as the target word. The target word acts as a label and helps training our model. The second sequence uses the description "startseq <first-word>" as the input description and second word in the description as the target word. This process is repeated until we reach the end of the description and assign 'endseq' as the target word.

Model Architecture



- In this model we are using an encoder-decoder architecture
- The 2048 image vector generated from the pretrained CNN model is fed into a dense layer with 256 nodes, to obtain a fixed length image vector of 256
- The 34-length word vector is fed into an embedding LSTM/GRU/SimpleRNN layer with 256 nodes to generate a 256 fixed length word vector
- The 256-length image and word vector are considered as the encoder outputs
- Decoder model adds both the encoder outputs and feeds it into dense 256 layer
- The output layer is a dense layer with 8763 nodes (size of the vocabulary) followed by a softmax layer, which will output probabilities for each word in the vocabulary. The maximum probability word will be predicted as the next word in the description
- There is a total of 5,611,323 trainable parameters and the model took 2 hours to run on a NVIDIA Tesla P100 GPU

```
Model: "model_3"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_5 (InputLayer)            (None, 34)           0
__________________________________________________________________________________________________
input_4 (InputLayer)            (None, 2048)         0
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 34, 256)      2243584     input_5[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 2048)         0           input_4[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 34, 256)      0           embedding_2[0][0]
__________________________________________________________________________________________________
dense_4 (Dense)                 (None, 256)          524544      dropout_3[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   (None, 256)          525312      dropout_4[0][0]
__________________________________________________________________________________________________
add_2 (Add)                     (None, 256)          0           dense_4[0][0]
                                                                 lstm_2[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 256)          65792       add_2[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 8763)         2252091     dense_5[0][0]
==================================================================================================
Total params: 5,611,323
Trainable params: 5,611,323
Non-trainable params: 0
__________________________________________________________________________________________________
```

Generating Captions

We leverage the strategy of sequentially injecting text iteratively into the network to generate a description word by word. The initial input to the model would be the term 'startseq'. In each iteration we append the prediction of the network to the existing sequence of strings and inject the sequence with the same image back into the network to predict the next word in the description.

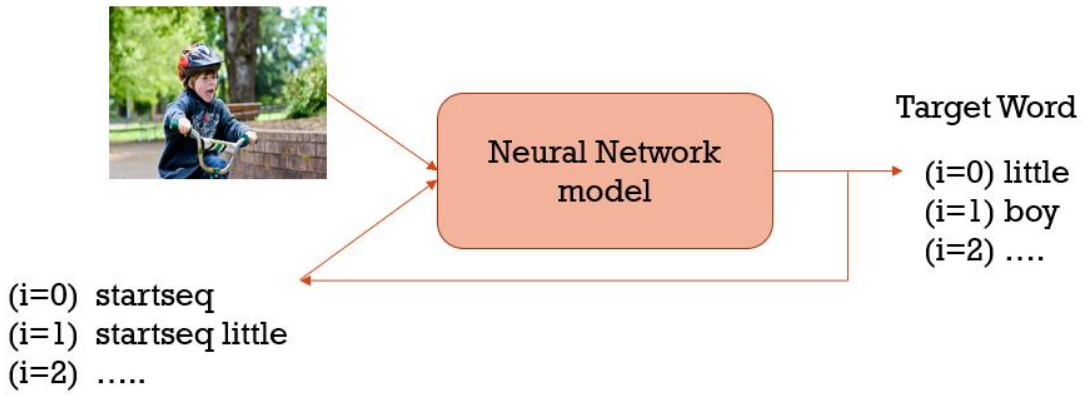


Figure 4: Model caption generation

As seen in Figure 4, in the first iteration we start the description as 'startseq' and send the equivalent word embedding and the image vector as the input to the model and the model predicts the word "little" as our next word in the description. We concatenate the word 'little' to the description, i.e "startseq little". We use this description and the image again as an input in the second iteration and predict "boy" as our next word. We iterate until the model predicts the next word in the description to be 'endseq' or we reach the maximum description length, 34 in our case.

Results

We used two different pre-trained CNN models to generate the image feature vectors, namely InceptionV3 and VGG16. We did not observe any significant difference in the model results, the reason being these networks were extensively trained on huge image datasets like ImageNet and have very less difference in performance.

We used three different recurrent neural networks to generate captions, namely Simple Recurrent Neural Network, Gated Recurrent Unit and Long Term-Short Term Memory. We observed that LSTM was performing the best out of all the networks. We also used a Feed Forward Neural Network (FFNN) out of curiosity and observed that the whole model behaved as a non-efficient image classifier rather than a generative model. This was observed because FFNN does not take previous seen inputs into consideration to make prediction, losing the temporal context in the data.

Bilingual Evaluation Understudy Score (BLEU)

- BLEU is a metric for evaluating a generated sentence to a reference sentence
- BLEU score lies between 0 and 1, where a perfect match results in a score of 1 and a perfect mismatch results in a score of 0
- It counts the matching n-grams in the generated and the reference description, where 1-gram or unigram would be each token and a bigram comparison would be each word pair
- The n-gram comparison is made regardless of word order
- BLEU ensures that the all the occurrences in the reference text is considered

Reported BLEU for unigram, bigram, trigram and 4-gram in the below table:

BLEU N-GRAM	SCORE		
	Simple RNN	GRU	LSTM
BLEU-1	0.364671	0.484472	0.572214
BLEU-2	0.181622	0.281942	0.339204
BLEU-3	0.104175	0.203185	0.237129
BLEU-4	0.085675	0.100058	0.116733

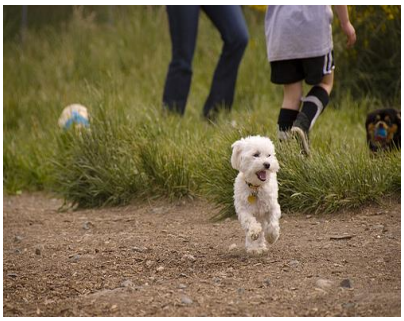
Table 2: N-gram BLEU metric for Simple RNN, GRU and LSTM

Captions generated by the model

a) Generated description which were a close match to the reference description

Actual Caption:
a boy with a blue helmet is riding a bike

Predicted Caption:
little boy rides bike with helmet



Actual Caption:
white fluffy dog running in dirt

Predicted Caption:
white dog runs across the sand

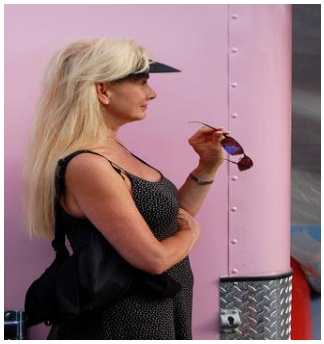
Actual Caption:
a boy dribbles a basketball in the gymnasium

Predicted Caption:
boy in white shirt is playing basketball

b) Humorous captions

Actual Caption:
man fly fishing in a small river with steam in the background

Predicted Caption:
man is swinging on a swing along a stream



Actual Caption:
a woman wearing a black and white outfit while holding her

Predicted Caption:
man in pink dress is holidng her her head

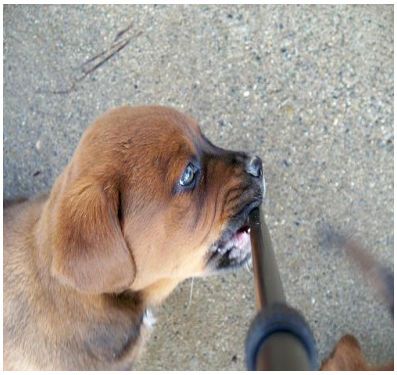
Actual Caption:
a group of different people are walking in all different directions

Predicted Caption:
group of people walking ocean

c) Inaccurate captions

Actual Caption:
a man wearing a red life jacket is holding a purple rope while skiing

Predicted Caption:
man in white and white and white shorts leash on swing



Actual Caption:
a dog is chewing on metal pole

Predicted Caption:
dog is standing in its mouth

Actual Caption:
a young hockey player playing in the ice rink

Predicted Caption:
chasing player in motorcycle is playing chasing

Discussions and Future Work

The idea of a neural network generating a sentence with proper grammar by looking at an image is utterly fascinating. We were successfully able to generate some good captions, some funny captions and some not so good captions as seen from the results. The main reason behind the funny captions was due to the model being confused by a predominant presence of a color or object in the image that sidelines the main subject and actions in the image. This can be fixed using visual attention techniques like soft visual attention or gaussian visual attention that help elevate the importance of the subject and actions in the image. The descriptions that we generated were very small in terms of length and we attribute this to the limited amount of data that we had for training. Generally generative models require huge amounts of data during training to be more efficient just like any other neural network.

One of the future works that we are interested to pursue is training the model with Flickr32k dataset consisting of 32k images and also creating an image to speech application where the user can take a snapshot of the surroundings and get a voice feedback of what is going on in that snapshot.

Contributions

Omkar worked on setting up the Google Cloud Platform and on pre-processing the text data. Mrinalini pre-processed the images and employed transfer learning using InceptionV3 to generate image vectors and stored them in pickle files. Omkar pre-processed the images and employed transfer learning using VGG16 to generate image vectors and stored them in pickle files. Omkar and Mrinalini performed EDA, wrote the feed forward neural network and Simple RNN using keras and trained the model. Mrinalini worked on the LSTM based model and Omkar worked on Gated Recurrent Unit based model. Both collectively picked the best results and presented them.

Code and Presentation

Code: https://github.com/Omkar20895/ImageCaptionGenerator

Presentation: https://www.slideshare.net/OmkarReddy7/image-caption-generation-using-convolutional-neural-network-and-lstm

References

[0] https://cs.stanford.edu/people/karpathy/cvpr2015.pdf

[1] https://arxiv.org/pdf/1703.09137.pdf

[2] https://arxiv.org/pdf/1411.4555.pdf

[3] https://keras.io/layers/about-keras-layers/

[4] https://www.kaggle.com/shadabhussain/flickr8k