Machine Learning Engineer Nanodegree

Capstone Project

Yash Aggarwal March 14, 2019

I. Definition

Project Overview

The problem that i am solving in this project is image captioning.Image captioning has always been an easy task for human even a 5 year old can do this with utmost ease.

Just prior to the recent development of Deep Neural Networks this problem was inconceivable even by the most advanced researchers in Computer Vision. But with the advent of Deep Learning this problem can be solved very easily if we have the required dataset.

This problem was well researched by andrej karpathy in his Phd at Stanford as cited below.For solving this problem i will be using Flickr8k dataset. Captioning the images manually is slow but doing it using computers will reduce the time required for the task exponentially.

We must first understand how important this problem is to real world scenarios. Let's see few applications where a solution to this problem can be very useful.

- **Self driving cars — Automatic driving is one of the biggest challenges and if we can properly caption the scene around the car, it can give a boost to the self driving system.**
- **Aid to the blind — We can create a product for the blind which will guide them travelling on the roads without the support of anyone else. We can do this by first converting the scene into text and then the text to voice. Both are now famous applications of Deep Learning.**
- **CCTV cameras are everywhere today, but along with viewing the world, if we can also generate relevant captions, then we can raise alarms as soon as there is some malicious activity going on somewhere. This could probably help reduce some crime and/or accidents.**
- **Automatic Captioning can help, make Google Image Search as good as Google Search, as then every image could be first converted into a caption and then search can be performed based on the caption.**

Problem Statement

The problem is given the image i have to generate a caption most appropriate for the image.

This problem is a supervised learning problem thus i will be requiring a data which contains images and its relative captions.the complete details about the data will be discussed in next section.

As for solution i will be using cnn(inception model) to convert this image into a vector and then use rnn(Lstm) to generate sequence of words .the rnn will take in image vector and the previous word to generate the next word and this will be continued until endseq is encountered or predicted by rnn(endseq will be inserted by me in the end of caption).

I realise my solution might not be completely clear but i will be describing it in more detail in later sections.

The model can be evaluated using bleu score which is used to quantify relevance

between two sentences.So I will evaluate my model using BLEU score between predicted and given output and take its average.

Metrics

In this section I will discuss about the BLEU score which i will be using to evaluate my 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 score was developed for evaluating the predictions made by automatic machine translation systems. It is not perfect, but does offer 5 compelling benefits:
● It is quick and inexpensive to calculate.
● It is easy to understand.
● It is language independent.
● It correlates highly with human evaluation.
● It has been widely adopted.
The BLEU score was proposed by Kishore Papineni, et al. in their 2002 paper " BLEU: a Method for Automatic Evaluation of Machine Translation ".
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.

So i will be predicting the caption for each test image and will get bleu score for each

caption provided for that image and take the one with highest accuracy after doing this for entire test set i will take the average of the score to get the result.

BLEU

- N-gram overlap between machine translation output and reference translation
- Compute precision for n-grams of size 1 to 4
- Add brevity penalty (for too short translations)

$$\text{BLEU} = \min\left(1, \frac{\textit{output-length}}{\textit{reference-length}}\right) \left(\prod_{i=1}^{4} \textit{precision}_i\right)^{\frac{1}{4}}$$

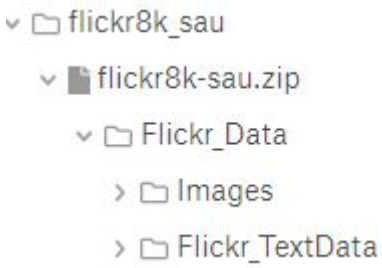
- Typically computed over the entire corpus, not single sentences

In this case I will be using 1gram Bleu score i will be comparing the caption predicted by my model and comparing it with all 5 possible captions and give it the score which is best of the five. I will be implementing it using nltk Bleu corpus function as it directly compared the generated sentence with all the sentences in the corpus.
This metric is relevant here because it calculates the relevance of my generated captions to the actual captions and is easy to implement as shown above.

II. Analysis

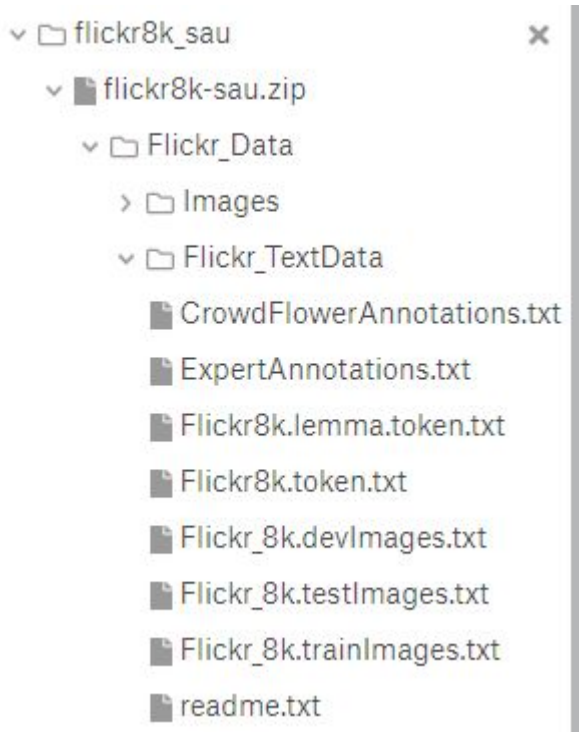
Data Exploration

The Flickr8k data consists of two files one is image file and other is file containing text so i will be talking about them separately :



Images File:
There are a total of 8000 images in image file.
The image files consists of all the images irrespective of whether they belong to test set or train set. Each image has an unique code associated with it so that it can be accessed using the set given in train and test token text files
Size of each image is 500 x 375 and all images have three channels rgb.

Text Files:
Text files are located in Flickr_textData subfile as shown below:



The text files that are required for the project are:

1. Token.txt:This file contains all the captions for the images .Each image has 5 captions associated with it. In the form as mentioned below.

```
101654506_8eb26cfb60.jpg#0      A brown and white dog is running through the snow .
101654506_8eb26cfb60.jpg#1      A dog is running in the snow
101654506_8eb26cfb60.jpg#2      A dog running through snow .
101654506_8eb26cfb60.jpg#3      a white and brown dog is running through a snow covered field .
101654506_8eb26cfb60.jpg#4      The white and brown dog is running over the surface of the snow .
```

   As you can see the first part before the hash teall the key for which image the following caption is and part after hash tells what is the caption number.It is separated from captions by a tab.
   This file contains caption for all 8000 images since there is 5 caption for each image there are total of 40000 captions in this file.

2. Trainimages.txt:This file contains list of all the images that are to be considered in the training set.for flickr8k data there is 6k images provided for training so this file contains the key for all the 6000 training images.

3. Testimages.txt:This file contains list of all the images that are to be considered in the testing set.for flickr8k data there is 1k images provided for training so this file contains the key for all the 1000 testing images.

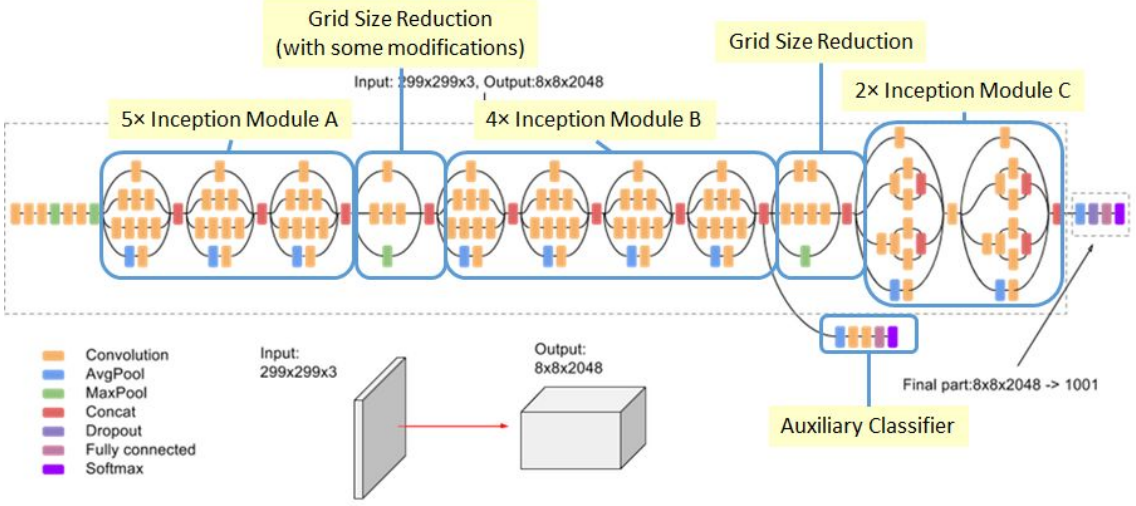
The training images captions consist of total 8.5k unique words.

So to segregate the images and captions for training and testing we need to store the test and train keys from these files and then match these keys and store the captions and images in different dictionaries.As you could see in my notebook.

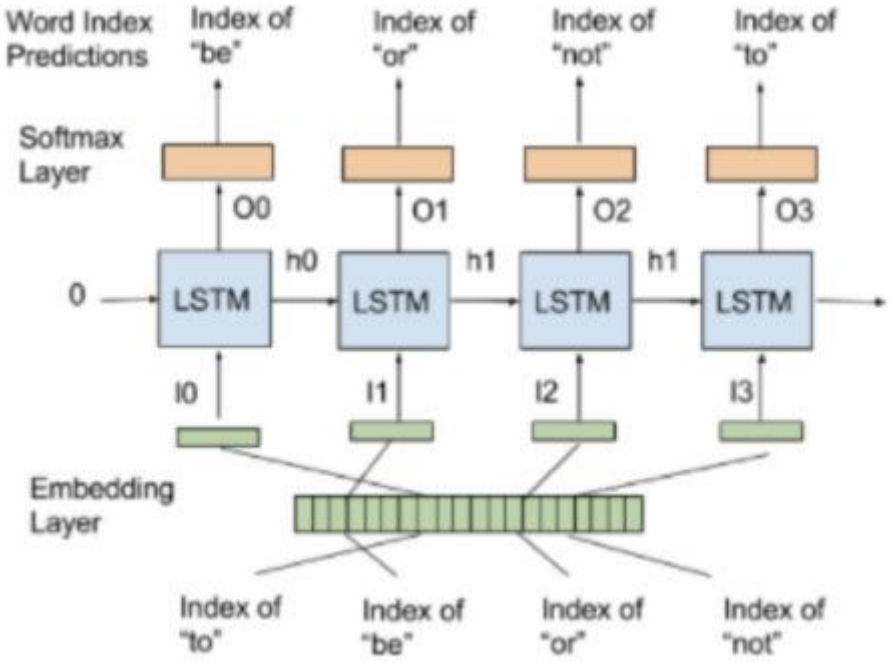
Algorithms and Techniques

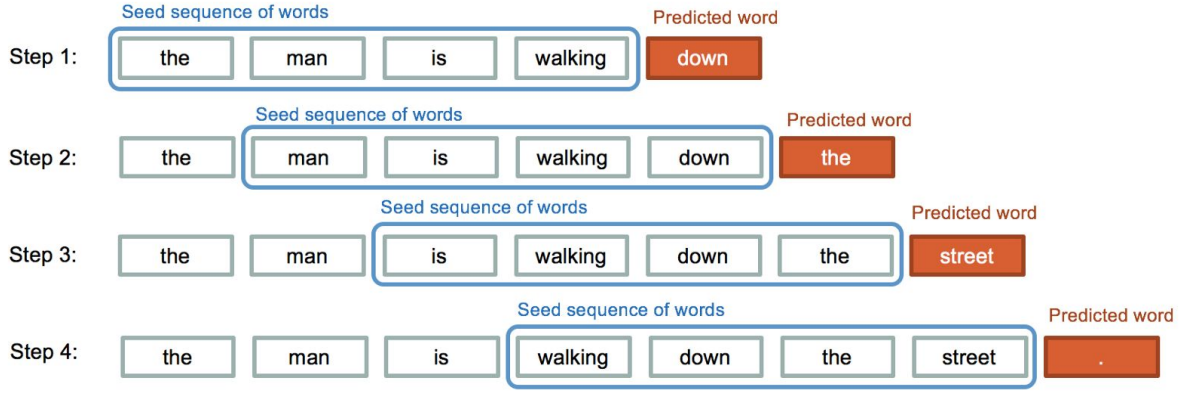
As it is clear that dow to solve the problem i need to handle two aspects one image and other text.

So to handle the image i will be using pre trained CNN structure Inception V3 model to be particular .I will be initializing the model using imagenet weights that is the weight of the network as it were after training on imagenet dataset as Imagenet is one of the largest dataset using its weight will allow my model to correctly vectorize large variety of images.
So inception v3 model will be used to vectorize images to be fed into the network.
This approach is known as transfer learning as i am using the model trained on some other dataset and then applying it on my dataset.



To handle the text files i will be using Lstm as Lstm is the most effective model to handle natural languages.They even take the temporal component into account that is where each word comes in the caption.Also as i will be requiring my model to predict word on the basis of image and the previous predicted word/words Lstm is the right choice for the job at hand.





The combined output of these two models will be used to predict the captions of the image word by word.

The main complication that occurs is during the entire project is training the model as the model is not a normal deep learning model i had to write separate data generator function to train it.It was my first time doing it thats why i found this task to be really difficult.

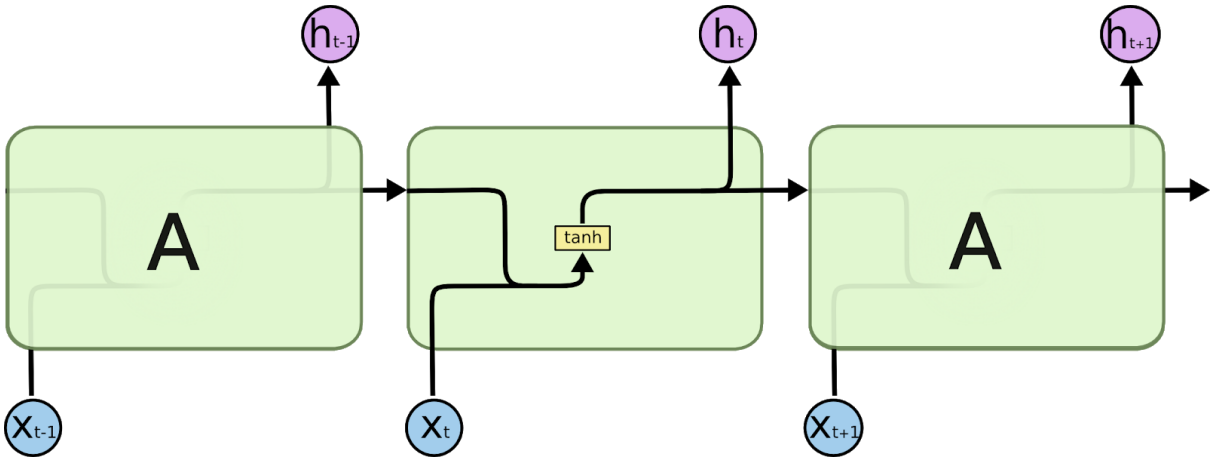
In above section I have discussed how the CNN and Lstm are used in this model Now I will be discussing them separately to give some clear understanding:

LSTM:

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work.[1] They work tremendously well on a large variety of problems, and are now widely used.

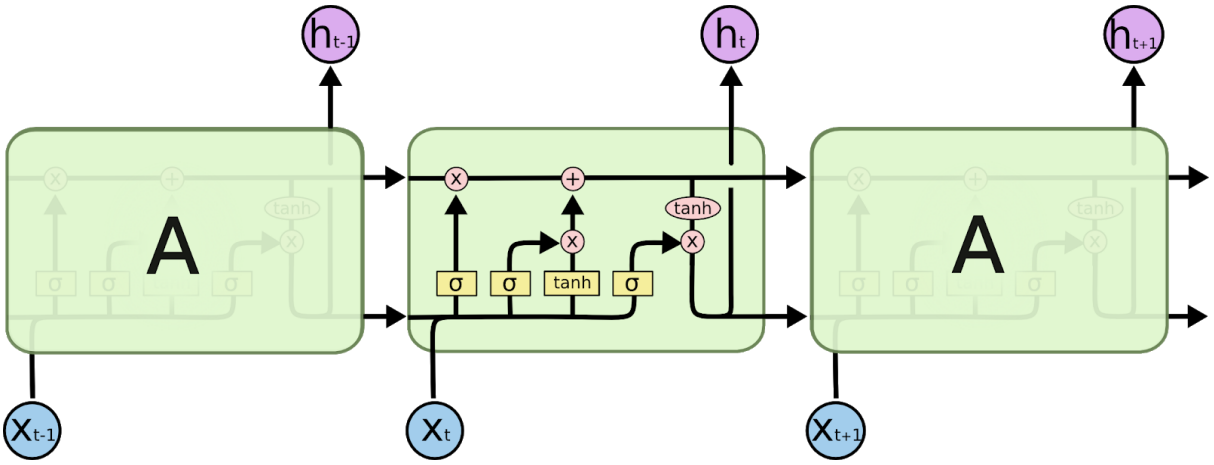
LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.



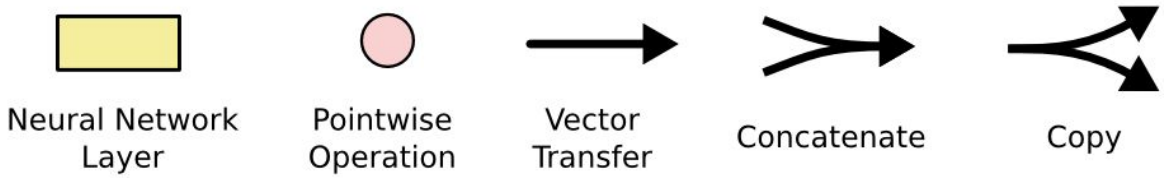
The repeating module in a standard RNN contains a single layer.

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.



The repeating module in an LSTM contains four interacting layers.

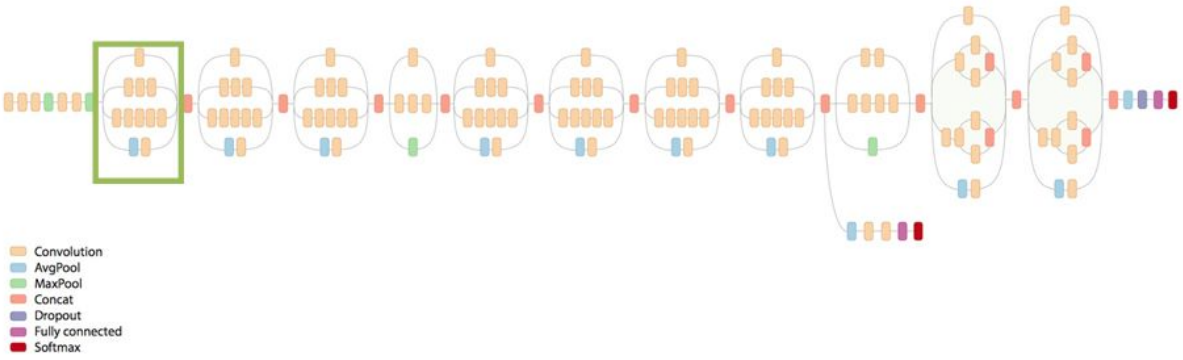
Don’t worry about the details of what’s going on. We’ll walk through the LSTM diagram step by step later. For now, let’s just try to get comfortable with the notation we’ll be using.



In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

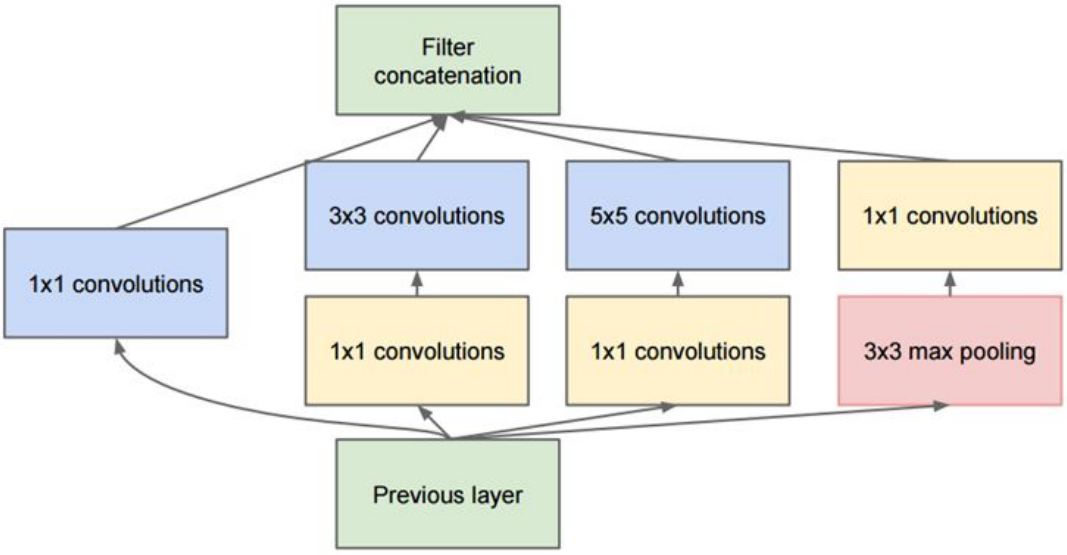
Inception V3:

When we first take a look at the structure of GoogLeNet, we notice immediately that not everything is happening sequentially, as seen in previous architectures. We have pieces of the network that are happening in parallel.



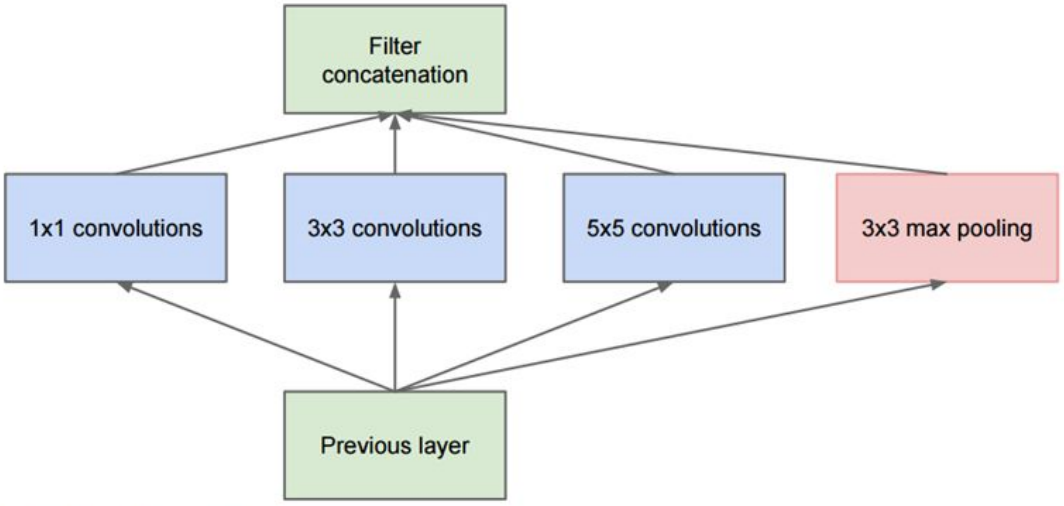
Green box shows parallel region of GoogLeNet

This box is called an Inception module. Let's take a closer look at what it's made of.



Full Inception module

The bottom green box is our input and the top one is the output of the model (Turning this picture right 90 degrees would let you visualize the model in relation to the last picture which shows the full network). Basically, at each layer of a traditional ConvNet, you have to make a choice of whether to have a pooling operation or a conv operation (there is also the choice of filter size). What an Inception module allows you to do is perform all of these operations in parallel. In fact, this was exactly the "naïve" idea that the authors came up with.



Naïve idea of an Inception module

Now, why doesn't this work? It would lead to **way** too many outputs. We would end up with an extremely large depth channel for the output volume. The way that the authors address this is by adding 1x1 conv operations before the 3x3 and 5x5 layers. The 1x1 convolutions (or network in network layer) provide a method of dimensionality reduction. For example, let's say you had an input volume of 100x100x60 (This isn't necessarily the dimensions of the image, just the input to any layer of the network). Applying 20 filters of 1x1 convolution would allow you to reduce the volume to 100x100x20. This means that the 3x3 and 5x5 convolutions won't have as large of a volume to deal with. This can be thought of as a "pooling of features" because we are reducing the depth of the volume, similar to how we reduce the dimensions of height and width with normal maxpooling layers. Another note is that these 1x1 conv layers are followed by ReLU units which definitely can't hurt . You may be asking yourself "How does this architecture help?". Well, you have a module that consists of a network in network layer, a medium sized filter convolution, a large sized filter convolution, and a pooling operation. The network in network conv is able to extract information about the very fine grain details in the volume, while the 5x5 filter is able to cover a large receptive field of the input, and thus able to extract its information as well. You also have a pooling operation that helps to reduce spatial sizes and combat overfitting. On top of all of that, you have ReLUs after each conv layer, which help improve the nonlinearity of the network. Basically, the network is able to perform the functions of these different operations while still remaining computationally

considerate. The paper does also give more of a high level reasoning that involves topics like sparsity and dense connections.

Main Points

- Used 9 Inception modules in the whole architecture, with over 100 layers in total! Now that is deep…
- No use of fully connected layers! They use an average pool instead, to go from a 7x7x1024 volume to a 1x1x1024 volume. This saves a huge number of parameters.
- Uses 12x fewer parameters than AlexNet.
- During testing, multiple crops of the same image were created, fed into the network, and the softmax probabilities were averaged to give us the final solution.
- Utilized concepts from R-CNN (a paper we'll discuss later) for their detection model.
- There are updated versions to the Inception module (Versions 6 and 7).
- Trained on "a few high-end GPUs **within a week**".

Benchmark

	Flickr8K				Flickr30K				MSCOCO 2014					
Model	B-1	B-2	B-3	B-4	B-1	B-2	B-3	B-4	B-1	B-2	B-3	B-4	METEOR	CIDEr
Nearest Neighbor	—	—	—	—	—	—	—	—	48.0	28.1	16.6	10.0	15.7	38.3
Mao et al. [38]	58	28	23	—	55	24	20	—	—	—	—	—	—	—
Google NIC [54]	63	41	27	—	66.3	42.3	27.7	18.3	66.6	46.1	32.9	24.6	—	—
LRCN [8]	—	—	—	—	58.8	39.1	25.1	16.5	62.8	44.2	30.4	—	—	—
MS Research [12]	—	—	—	—	—	—	—	—	—	—	—	21.1	20.7	—
Chen and Zitnick [5]	—	—	—	14.1	—	—	—	12.6	—	—	—	19.0	20.4	—
Our model	57.9	38.3	24.5	16.0	57.3	36.9	24.0	15.7	62.5	45.0	32.1	23.0	19.5	66.0

Table 2. Evaluation of full image predictions on 1,000 test images. **B-n** is BLEU score that uses up to n-grams. High is good in all columns. For future comparisons, our METEOR/CIDEr Flickr8K scores are 16.7/31.8 and the Flickr30K scores are 15.3/24.7.

This image is taken from andrej karpathy's paper here he has compared different models scores on some standard image captioning dataset.Here the comparison has been made on the bases of BLEU score which already has been discussed in detail in previous sections higher the score better the result as u can see karpathy's model on average can attain 57 accuracy score which is pretty good compared to other models.I will be trying to get a score pretty close to this as it will be hard to get same score using less data.

Model	B-1	B-2	B-3	B-4
Human agreement	61.5	45.2	30.1	22.0
Nearest Neighbor	22.9	10.5	0.0	0.0
RNN: Fullframe model	14.2	6.0	2.2	0.0
RNN: Region level model	**35.2**	**23.0**	**16.1**	**14.8**

ble 3. BLEU score evaluation of image region annotations.

He also compared the score of region based image captioning against that done by Humans not surprisingly Humans are much better in captioning images as compared to our model but i am sure with development in field of computer vision and nlp machines will surpass this score.

III. Methodology

Data Preprocessing

Data preprocessing is the hardest and most crucial part of this project.

As mentioned earlier for this project there are two types of data Text and Image data.They will be processed separately.

Text Data:
The text data consists of the captions associated with the images.So first of all we will need to make a dictionary consisting of all the captions associated with image as value and image address as key as there are total of 5 captions for each image there will be 5 captions associated with each keys.
I then cleaned all the descriptions which included:

1. **Tokenizing the descriptions.**

2. **Converting all the letters to lower case.**
3. **Removing punctuations.**
4. **Removing numbers .**
5. **Removing hanging s and r.**
6. **Adding startseq and endseq in the beginning and end of the captions were added so we can teach network where the caption begins and how to end it.**

Once this is done we will load the image addresses of train and test images and create separate dictionaries of captions for each set.

After completing this task I created vocabulary containing all the unique words in the training captions and removed the words that appeared less than a certain threshold times in my case threshold was set 3.This reduced the size of vocabulary quite a bit.Also an additional index is added for 0.

After it was done i created two dictionaries one to convert word to indexes to be used to convert actual words into numbers so it can be used to train the model as all the models work on numbers not on words directly.
I also made a second dictionary to convert these indexes to the words which will be used at the time of testing as i will get output a list of indexes not the word.

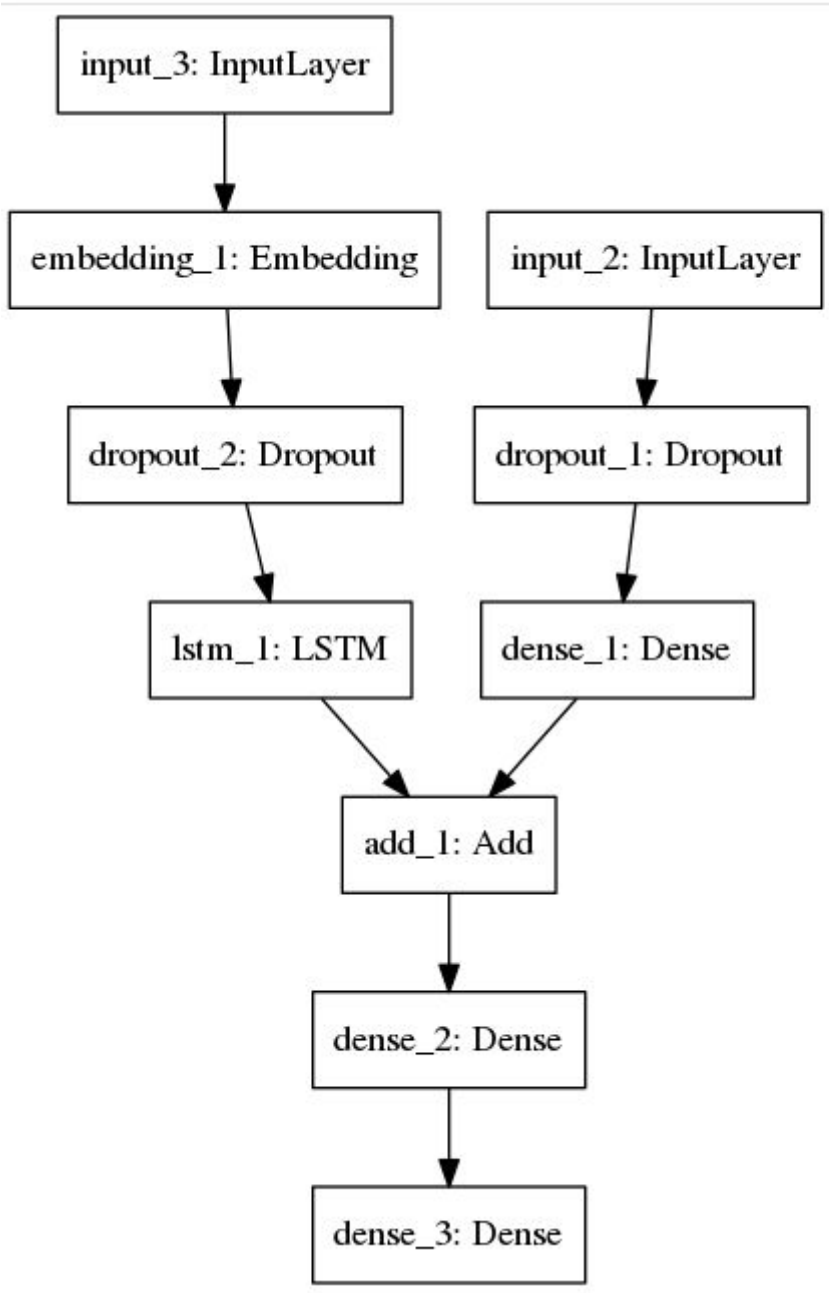
Preprocessing Image:
Fist i converted all the images into the size of 299X299 and then converted them to a numpy array to be fed into the network.
Then I loaded the inception v3 model along with its imagenet weight.
Then i passed all the images one by one through inception network and got the output of second layer which is a vector.In a way i converted a 3 dimensional images to a one (2048,) dimensional vector.
I did this for all the images and stored them in a dictionary with the image address as key and save the dictionary as a pickle file.
I did it for both test and train images separately.
The last part of preprocessing that is data generator function will be discussed in implementation section.

Implementation

Once all the data preprocessing is done then the next step is training the model.
I used keras functional apis to create the model my code for model is shown below:

```
inputs1 = Input(shape=(2048,))
fe1 = Dropout(0.5)(inputs1)
fe2 = Dense(256, activation='relu')(fe1)
inputs2 = Input(shape=(max_length,))
se1 = Embedding(vocab_size, embedding_dim, mask_zero=True)(inputs2)
se2 = Dropout(0.5)(se1)
se3 = LSTM(256)(se2)
decoder1 = add([fe2, se3])
decoder2 = Dense(256, activation='relu')(decoder1)
outputs = Dense(vocab_size, activation='softmax')(decoder2)
model = Model(inputs=[inputs1, inputs2], outputs=outputs)
```

As you can see my models consists of two networks that take in two separate inputs and and are later joined together the input1 take in the image vector and the second input takes in the word vector of size maxlength.



As it is clear that the model is not a normal deep learning model we need a special function to train these kind of models called Data generator function. So suppose we have an image like one given below:

The caption of this image is ‘startseq a white dog is running through the snow endseq’ .So for training we will need to break this sentence into words and vectorize it and then index each word .So first we will feed our network with the image and ‘startseq’ as input and ask it to predict the next word suppose the next word predicted is ‘a’ then we will join it with startseq and send it in the network to predict next word this process will go on till the endseq is reached.As shown in image below:

	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

Appending zeros to each sequence to make them all of same length 34

So that is the function of data generator function to help train our model.

The dropout layer is added to prevent the model from overfitting.Also Lstm layer is used to process the word vector and after the combination of both the layers a dense layer is added to allow the model to learn from both features simultaneously and the last layer is output layer it has same size as our vocabulary it consists of the probability of each word being the next word given previous words.

While testing we will be using greedy search algorithm to get the caption output given the image that is we will always choose the highest probable next word to get the caption output.

Refinement

Fist of all i needed to choose between various cnn models to be used for encoding images the accuracy for VGG was 50.6 ,for resnet was 55 and for inception model it is 56 percent so i chose the inception model.

Next step for refinement was to find right optimizer it was pretty close between Rmsprop and adam but the adam optimizer was able to converge more quickly and gave better results.for example it took only ten epochs for adam to get to 2.5 loss whereas it took rms 14 epochs for the same.

The the final step was to choose right number of epochs but in this case as i can't use any predefined metrics in keras library i had to run each epoch till i get to an epoch after which the blue score on test set decreased for subsequent epochs thus the number of best epochs for my model is 10.
Also I needed to choose the appropriate threshold for the words to be considered into vocabulary i first tried 10 but the output came out to be too generalized ,then i reduced it to five still output got better but still there was not much difference ,when threshold was set three i got best results.

IV. Results

Model Evaluation and Validation

My model gives a final bleu score on test test of 56.17 which is pretty close to the state of the art results it is through the refinements above i was able to achieve this result.
What amazes me most is that all the captions no matter how off the mark they may be make perfect sense grammatically and syntatically.This score is much better than the lower benchmark set by knn by at least 30 percent and is only 2 percent short of state of the art results.
For example:

caption='black and white dog is running through field'

In this example my model is able to predict the caption perfectly it is even able to detect correct color of the dog along with its action and surrounding which is amazing.

Now showing one of the worst examples on test model is:

caption='man in red shirt is sitting near lake'

In this image my model got confused as the predominant color is red so it detected man in red shirt and since there are umbrella in the picture it predicted the place to be near beach or lake also it was not able to predict correct number of people in the image although number of such cases are small and i am sure given a larger dataset my model will be able to overcome this obstacle as well.

It is clear that my model is robust and is able to do a fine job captioning the images as both the images are from test data so my model is able to do sufficiently well on unseen data.

Justification

Now coming to compare my model to benchmark models as mentioned above:

	Flickr8K				Flickr30K				MSCOCO 2014					
Model	B-1	B-2	B-3	B-4	B-1	B-2	B-3	B-4	B-1	B-2	B-3	B-4	METEOR	CIDEr
Nearest Neighbor	—	—	—	—	—	—	—	—	48.0	28.1	16.6	10.0	15.7	38.3
Mao et al. [38]	58	28	23	—	55	24	20	—	—	—	—	—	—	—
Google NIC [54]	63	41	27	—	66.3	42.3	27.7	18.3	66.6	46.1	32.9	24.6	—	—
LRCN [8]	—	—	—	—	58.8	39.1	25.1	16.5	62.8	44.2	30.4	—	—	—
MS Research [12]	—	—	—	—	—	—	—	—	—	—	—	21.1	20.7	—
Chen and Zitnick [5]	—	—	—	14.1	—	—	—	12.6	—	—	—	19.0	20.4	—
Our model	57.9	38.3	24.5	16.0	57.3	36.9	24.0	15.7	62.5	45.0	32.1	23.0	19.5	66.0

Table 2. Evaluation of full image predictions on 1,000 test images. **B-n** is BLEU score that uses up to n-grams. High is good in all columns. For future comparisons, our METEOR/CIDEr Flickr8K scores are 16.7/31.8 and the Flickr30K scores are 15.3/24.7.

My model is able to do pretty well considering the amount of data i used it is only off by 2 percent compared to the best model available but when compared to lowes benchmark models such as Knn my model seems to do really well as in that case bleu score is 22.7 only as shown below:

Model	B-1	B-2	B-3	B-4
Human agreement	61.5	45.2	30.1	22.0
Nearest Neighbor	22.9	10.5	0.0	0.0
RNN: Fullframe model	14.2	6.0	2.2	0.0
RNN: Region level model	**35.2**	**23.0**	**16.1**	**14.8**

ble 3. BLEU score evaluation of image region annotations.

Though none of the state of the art results including my model can cross the human accuracy i am sure with the development in field of NLP and Cv we will be able to overcome it.

As compared to these benchmark models i am really happy with my model and will like to improve it further using latest tools.

V. Conclusion

Reflection

So now defining my complete end to end solution.First i downloaded the flickr 8k dataset and studied it completely after which i preprocessed the data both text and images I converted images to vector using inception model and cleaned all the text data and tokenized it.
My final model has two input layers one takes in word vector (caption)and other image as this model needs to be trained differently (as mentioned above)than the normal model i used data generator function for training it.After adjusting right parameters i was able to get satisfactory results that is bleu score of 56.16 on test set.
Also while finding the captions for the test images I used greedy function that is i chose the word most likely to appear next.

The most difficult part of this project is going to be the generator function it took quite a bit of effort to implement it and also to implement this project i had to apply both the knowledge of NLP and Computer vision.
As for the results I am really happy and surprised with them as i didn't expect them to be so accurate they really exceeded my expectation.Most of the captions in test images were well formed and fairly accurate as shown in my notebook.

Improvement

There are lots of improvement that i can make in model:

1. I can use a larger dataset to better train my model but was unable to do so duse to limitations in computational power.
2. Using attentive lstm as proposed by andrej in his paper I tried to implement it even wrote a class to make it compatible with keras but wasn't able to implement it correctly(as keras doesn't have attentive layer yet)
3. Also i can use different approach than greedy to select the next word in my caption such as beam search.

Soruces and citations:

1. https://cs.stanford.edu/people/karpathy/cvpr2015.pdf
2. https://arxiv.org/abs/1411.4555
3. https://arxiv.org/abs/1703.09137
4. https://arxiv.org/abs/1708.02043
5. https://machinelearningmastery.com/develop-a-deep-learning-caption-generation-model-in-python/
6. https://www.youtube.com/watch?v=yk6XDFm3J2c
7. https://www.appliedaicourse.com/