Image Captioning using Deep Learning

1. Exploratory Data Analysis:

Data pre-processing and cleaning is an important part of the whole model building process. Understanding the data helps us to build more accurate models.

Flilckr8K (**Flickr8k_Dataset.zip** and **Flickr8k_text.zip**) contains 8,000 images that are each paired with five different captions which provide clear descriptions of the salient entities and events. The images were chosen from six different Flickr groups, and tend not to contain any well-known people or locations, but were manually selected to depict a variety of scenes and situations.

We find below when we extract the Flicker8K ZIP files.

Flickr8k_Dataset: Contains a total of 8092 images in JPEG format with different shapes and sizes. Of which 6000 are used for training, 1000 for test and 1000 for validation

Flickr8k_text: Contains below text files

- Train set (Flickr_8k.trainImages) contains image ids 6000 images in Flickr8k_Dataset
- Validation set (Flickr_8k.devImages) contains image ids 1000 images in Flickr8k_Dataset
- Test set (Flickr_8k.testImages) contains image ids 1000 images in Flickr8k_Dataset
- Flickr8k.token.txt contains 5 captions for each image i.e. total 40460 captions.

1000268201_693b08cb0e.jpg#0	A child in a pink dress is climbing up a set of stairs in an entry way.
1000268201_693b08cb0e.jpg#1	A girl going into a wooden building.
1000268201_693b08cb0e.jpg#2	A little girl climbing into a wooden playhouse.
1000268201_693b08cb0e.jpg#3	A little girl climbing the stairs to her playhouse.
1000268201_693b08cb0e.jpg#4	A little girl in a pink dress going into a wooden cabin.

- Flickr8k.lemma.token.txt contains 5 lemmatized captions for each image i.e. total 40460 captions.

1000268201_693b08cb0e.jpg#0	A child in a pink dress be climb up a set of stair in an entry way.
1000268201_693b08cb0e.jpg#1	A girl go into a wooden building.
1000268201_693b08cb0e.jpg#2	A little girl climb into a wooden playhouse.
1000268201_693b08cb0e.jpg#3	A little girl climb the stair to her playhouse.
1000268201_693b08cb0e.jpg#4	A little girl in a pink dress go into a wooden cabin.

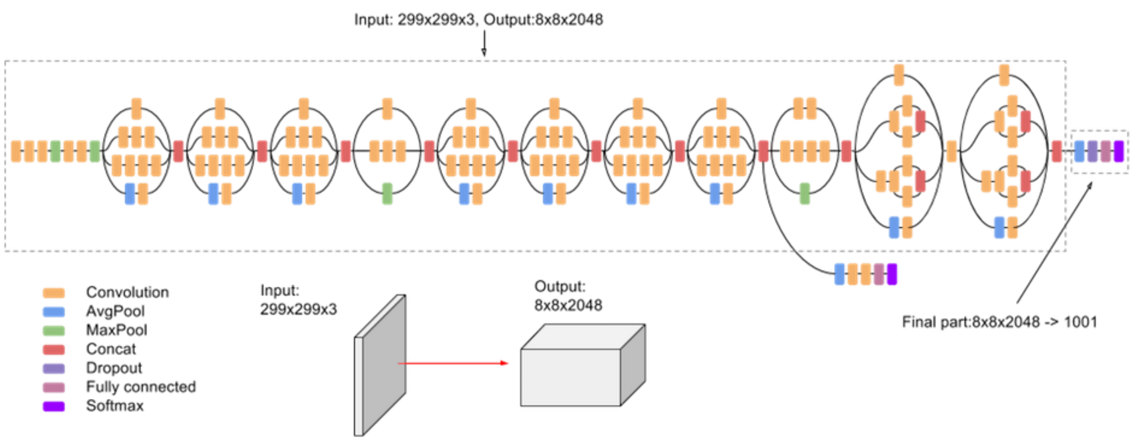
1a. Image Pre-processing:

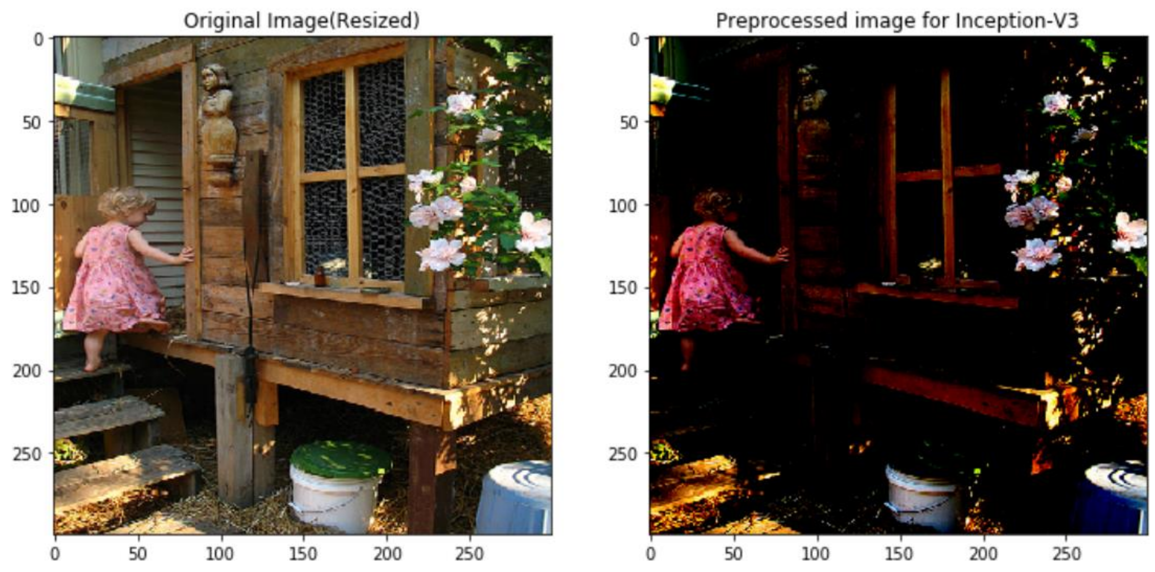
Images in Flicker8K_Dataset is one of the inputs for our use case. Our problem is to generate captions for the input Images and not to just classify those. We need to extract features vectors from images, so for this purpose we use pre-trained Convolutional Neural Network model Google’s Inception V3 by removing final softmax layer. This will reduce the development and training time to extract image vectors.

Inception v3 is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. Also, it has smaller weight file (around 95MB) and it is faster to train.

Every training image, we are resizing it to 299x299 and then passing it to Inception for feature extraction. Inception gives us 2048-dimensional feature extracted vector for given input image.

A high-level diagram of the model is shown below:



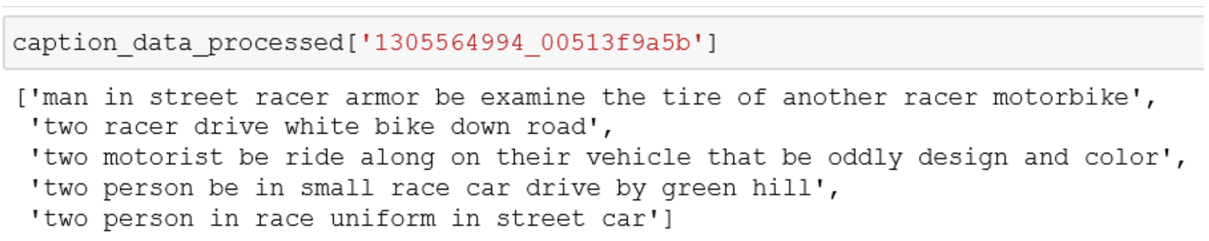


1b. Caption Pre-Processing:

Every caption line contains name of the image, caption number (0 to 4) and the actual caption.

1000268201_693b08cb0e.jpg#0	A child in a pink dress be climb up a set of stair in an entry way.
1000268201_693b08cb0e.jpg#1	A girl go into a wooden building.
1000268201_693b08cb0e.jpg#2	A little girl climb into a wooden playhouse.
1000268201_693b08cb0e.jpg#3	A little girl climb the stair to her playhouse.
1000268201_693b08cb0e.jpg#4	A little girl in a pink dress go into a wooden cabin.

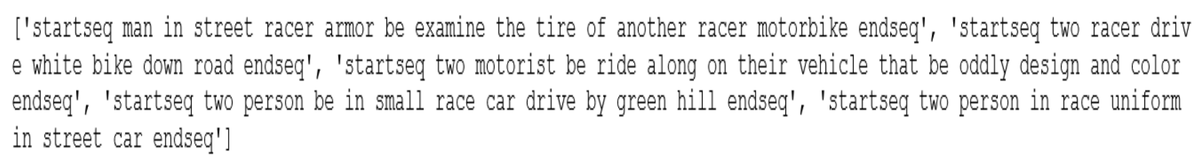
We create a dictionary, which contains the name of the image as keys and a list of the 5 captions for the corresponding image as values.



```
caption_data_processed['1305564994_00513f9a5b']
```

```
['man in street racer armor be examine the tire of another racer motorbike',
 'two racer drive white bike down road',
 'two motorist be ride along on their vehicle that be oddly design and color',
 'two person be in small race car drive by green hill',
 'two person in race uniform in street car']
```

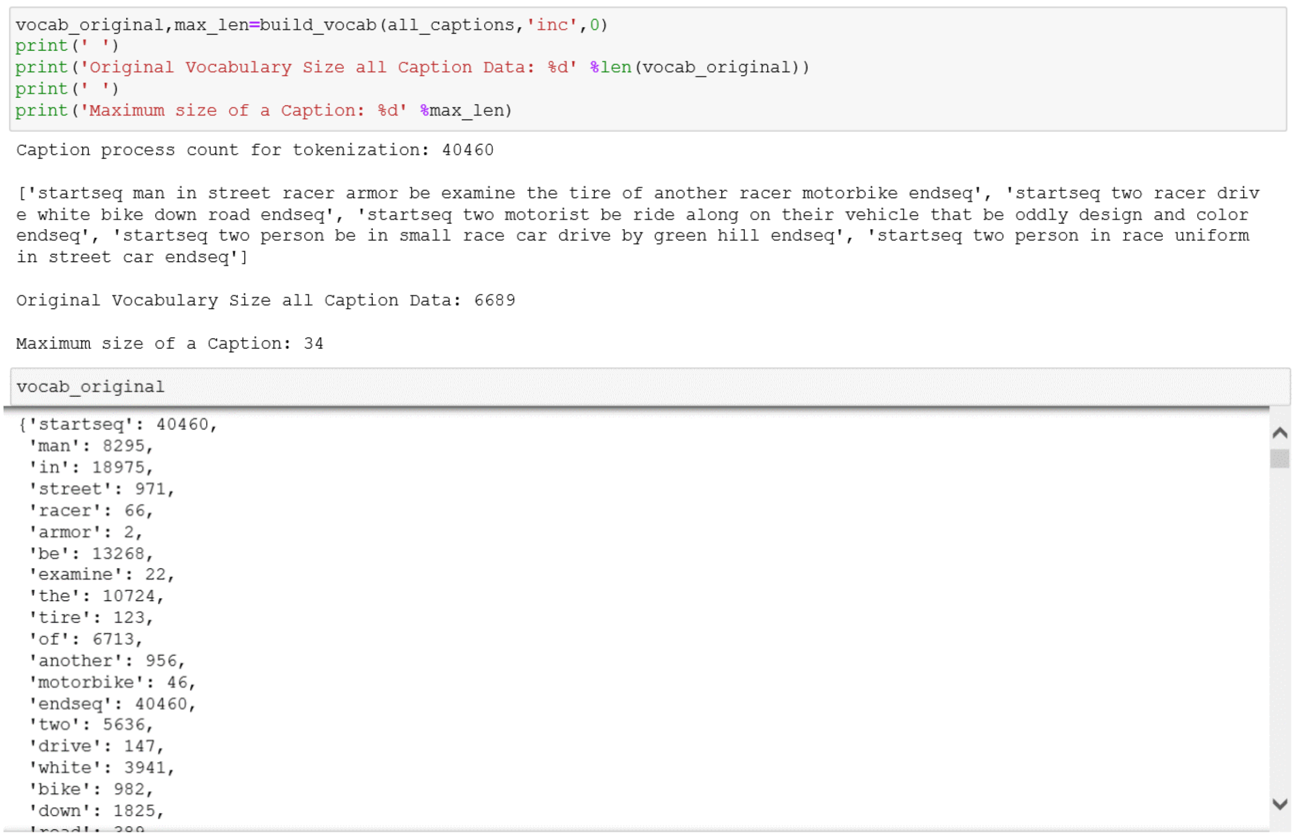
Append startseq & endseq tokens to every caption text, these tokens are needed for Seq2Seq modelling. One word will be generated at a time and which previously generated word/words becomes input for current word, so to start and stop the generation process these tokens will be used.



```
['startseq man in street racer armor be examine the tire of another racer motorbike endseq', 'startseq two racer driv
e white bike down road endseq', 'startseq two motorist be ride along on their vehicle that be oddly design and color
endseq', 'startseq two person be in small race car drive by green hill endseq', 'startseq two person in race uniform
in street car endseq']
```

Vocabulary set 1 of size 6689 with all the unique words in caption corpus is build, word index is created and each word in the vocabulary is mapped with pre-train GloVe word vectors of dimension 200, will be used to create word embedding matrix od size 6690x200.

All captions should be of same size for language model, max length of the caption is calculated and will be used for padding zeros to make all captions of same size. For our data max size of captions is 34.



```
vocab_original,max_len=build_vocab(all_captions,'inc',0)
print(' ')
print('Original Vocabulary Size all Caption Data: %d' %len(vocab_original))
print(' ')
print('Maximum size of a Caption: %d' %max_len)
```

```
Caption process count for tokenization: 40460

['startseq man in street racer armor be examine the tire of another racer motorbike endseq', 'startseq two racer driv
e white bike down road endseq', 'startseq two motorist be ride along on their vehicle that be oddly design and color
endseq', 'startseq two person be in small race car drive by green hill endseq', 'startseq two person in race uniform
in street car endseq']

Original Vocabulary Size all Caption Data: 6689

Maximum size of a Caption: 34
```

```
vocab_original
```

```
{'startseq': 40460,
 'man': 8295,
 'in': 18975,
 'street': 971,
 'racer': 66,
 'armor': 2,
 'be': 13268,
 'examine': 22,
 'the': 10724,
 'tire': 123,
 'of': 6713,
 'another': 956,
 'motorbike': 46,
 'endseq': 40460,
 'two': 5636,
 'drive': 147,
 'white': 3941,
 'bike': 982,
 'down': 1825,
vocab_original_wordtoidx
```

```
{'startseq': 1,
 'man': 2,
 'in': 3,
 'street': 4,
 'racer': 5,
 'armor': 6,
 'be': 7,
 'examine': 8,
 'the': 9,
 'tire': 10,
 'of': 11,
 'another': 12,
 'motorbike': 13,
 'endseq': 14,
 'two': 15,
 'drive': 16,
 'white': 17,
 'bike': 18,
 'down': 19,
 'road': 20
```

```
embedding_matrix_original.shape
```

```
(6690, 200)
```

Vocabulary set 2 of size 2993 with all the unique words in caption corpus is build, word index is created and each word in the vocabulary is mapped with pre-train GloVe word vectors of dimension 200, will be used to create word embedding matrix od size 2994x200.

```
vocab_filter,_=build_vocab(all_captions,'exc',3)
print(' ')
print('Filtered Vocabulary Size all Caption Data: %d' %len(vocab_filter))
```

```
Caption process count for tokenization: 40460

['startseq man in street racer armor be examine the tire of another racer motorbike endseq', 'startseq two racer driv
e white bike down road endseq', 'startseq two motorist be ride along on their vehicle that be oddly design and color
endseq', 'startseq two person be in small race car drive by green hill endseq', 'startseq two person in race uniform
in street car endseq']

Filtered Vocabulary Size all Caption Data: 2993
```

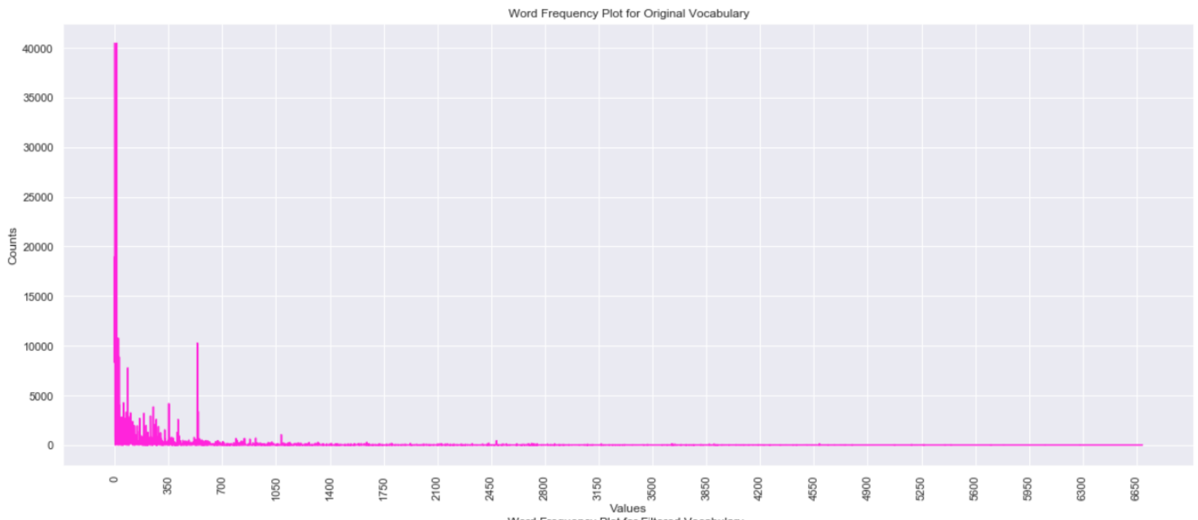
```
vocab_filter
```

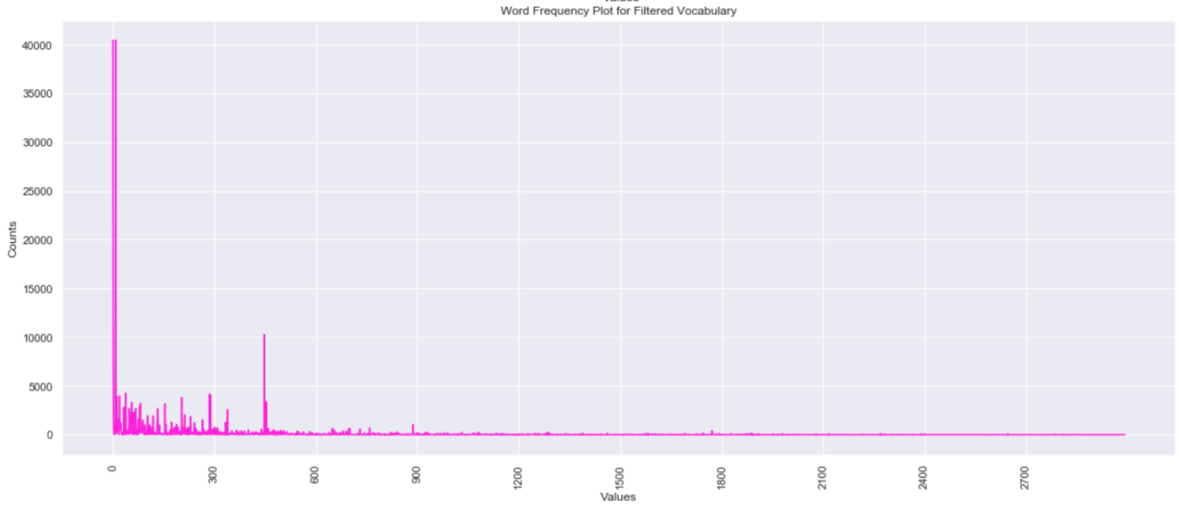
```
{'startseq': 40460,
 'man': 8295,
 'street': 971,
 'racer': 66,
 'examine': 22,
 'tire': 123,
 'another': 956,
 'motorbike': 46,
 'endseq': 40460,
 'two': 5636,
 'drive': 147,
 'white': 3941,
 'bike': 982,
 'road': 389,
 'ride': 1638,
 'along': 527,
 'vehicle': 101,
 'design': 6,
 'color': 56,
 'person': 3973
```

```
embedding_matrix_filter.shape
```

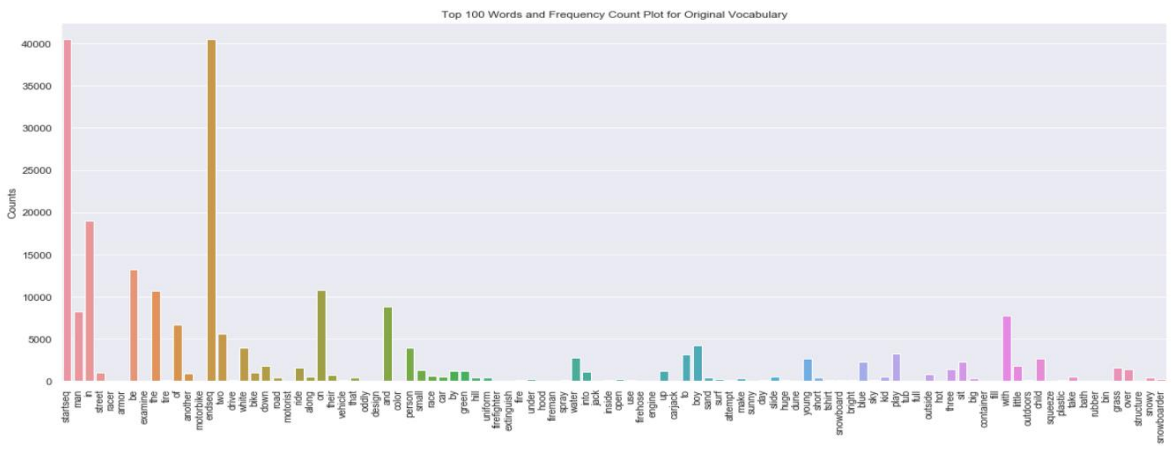
```
(2994, 200)
```

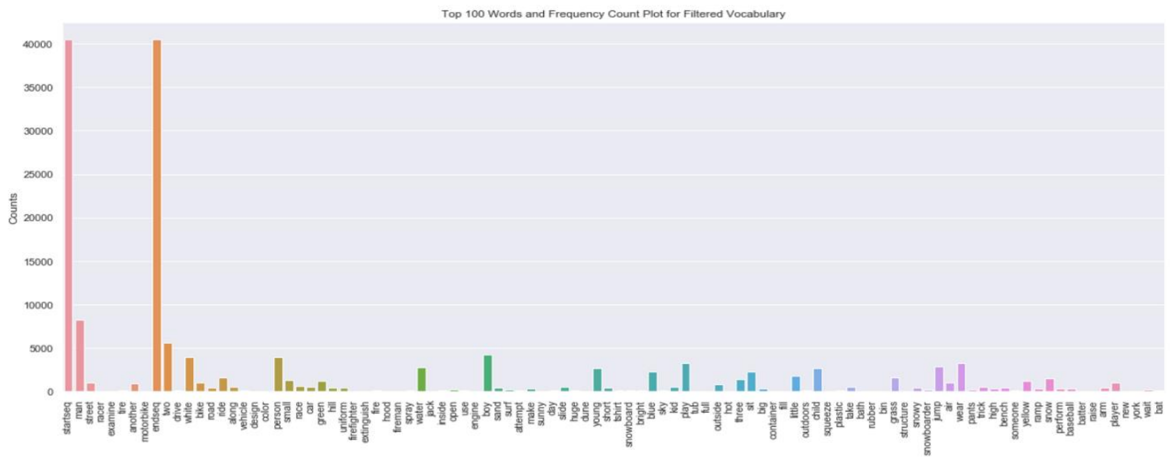
Word frequency distributions for Original and Filtered vocabularies are almost same.





Plotting top 100 words with frequency counts, show StopWords are most frequent in Original vocabulary.





1c. Generating Data Points (Xi's,yi – X is inputs and y is predicted output):
To convert our use case into a supervised learning problem, we've to prepare data points which becomes input to the merge model which generates caption text as output.

We have Image vectors, which becomes one of the inputs say 'X1'.
We build a padded word(embedded) sequence, which becomes another input say 'X2'.
The next word, which will be predicted (encoded word probability) is the output say 'y'.

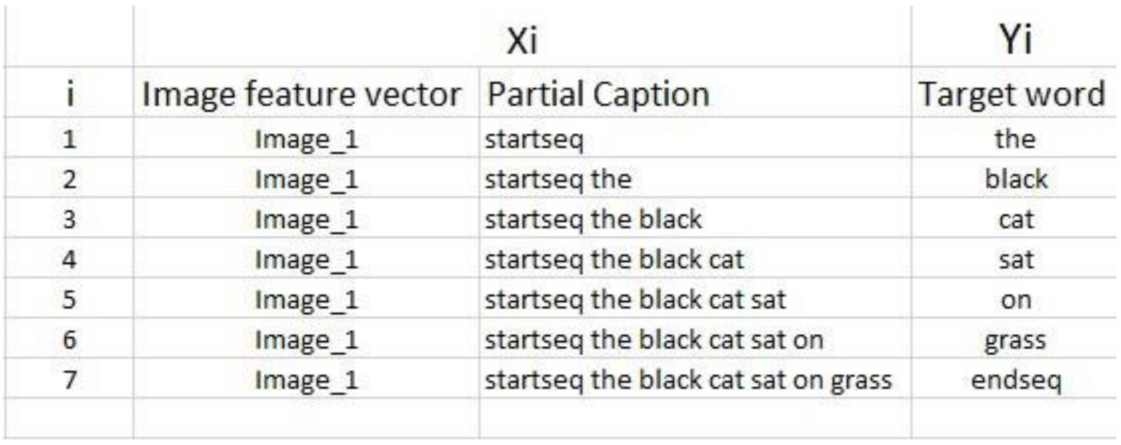
For a sample, we take a caption **startseq the black cat sat on grass endseq** for Image_1.

X for Data point #1 would be **Image_1 Vector** + **startseq**
y for Data point #1 would be **the**

X for Data point #2 would be **Image_1 Vector** + **startseq the**
y for Data point #2 would be **black**

and so on till predicted output is **endseq**.

For our sample caption we'll generate 7 data points.

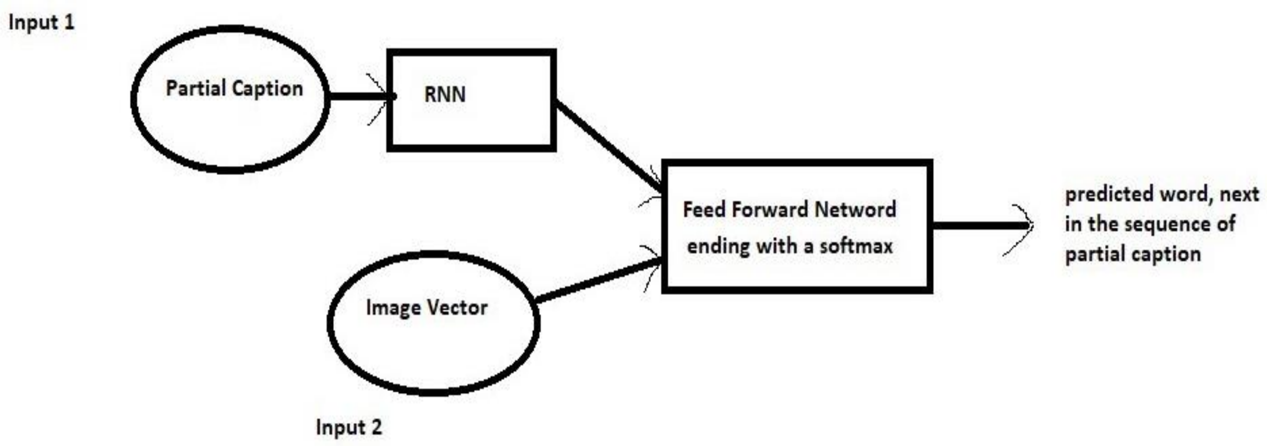


	Xi		Yi
i	Image feature vector	Partial Caption	Target word
1	Image_1	startseq	the
2	Image_1	startseq the	black
3	Image_1	startseq the black	cat
4	Image_1	startseq the black cat	sat
5	Image_1	startseq the black cat sat	on
6	Image_1	startseq the black cat sat on	grass
7	Image_1	startseq the black cat sat on grass	endseq

2. Base model & Architecture:

For our use case, we've to process two inputs Image vectors and Partial captions. We'll be using Functional model API to merge two sequential models.

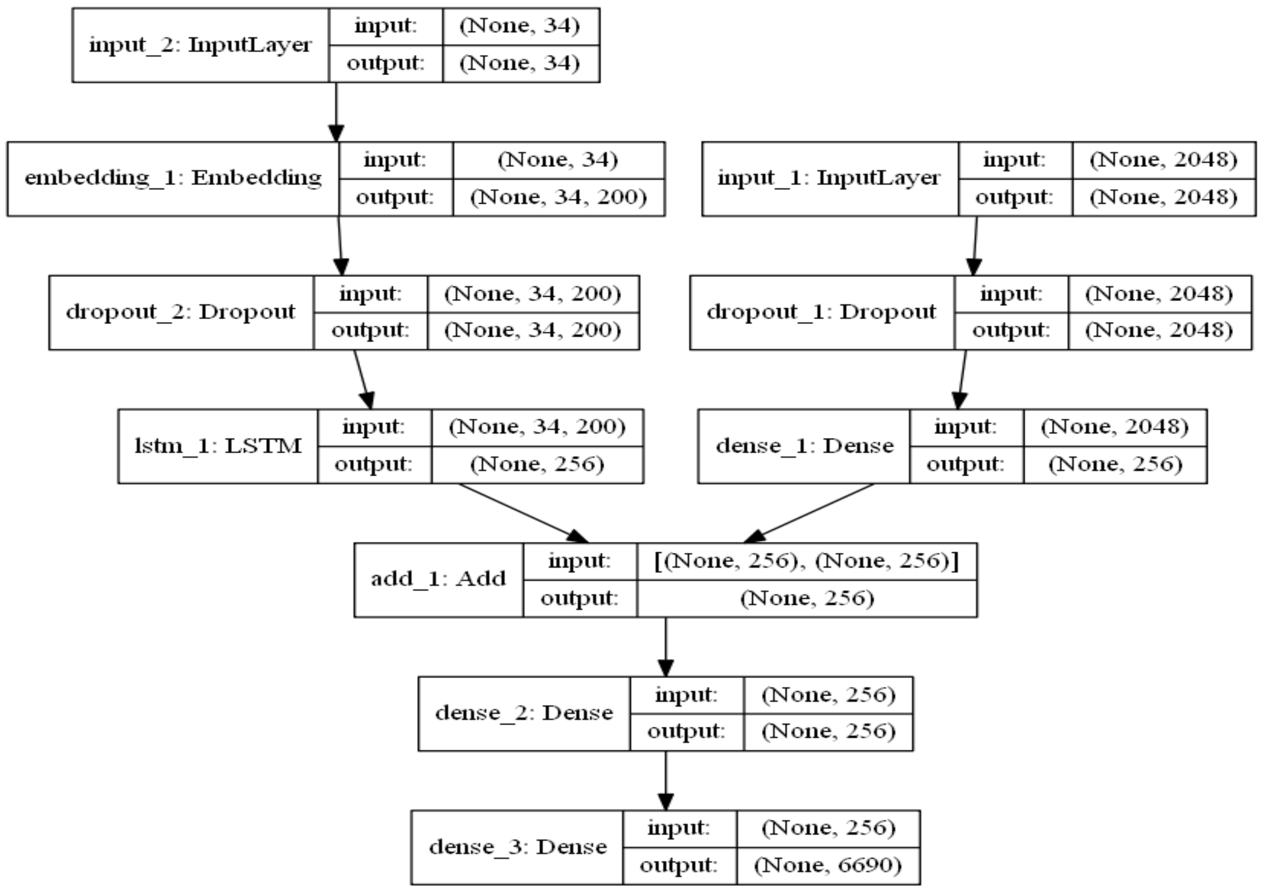
High level architecture of the base model would be like below


Input 1
Partial Caption
RNN
Feed Forward Netword ending with a softmax
predicted word, next in the sequence of partial caption
Image Vector
Input 2

We've two vocabulary set's, below is the summary and network structure for model defined with original vocabulary.

```
model1=define_model(vocab_original_size,max_len,'model_original')
```

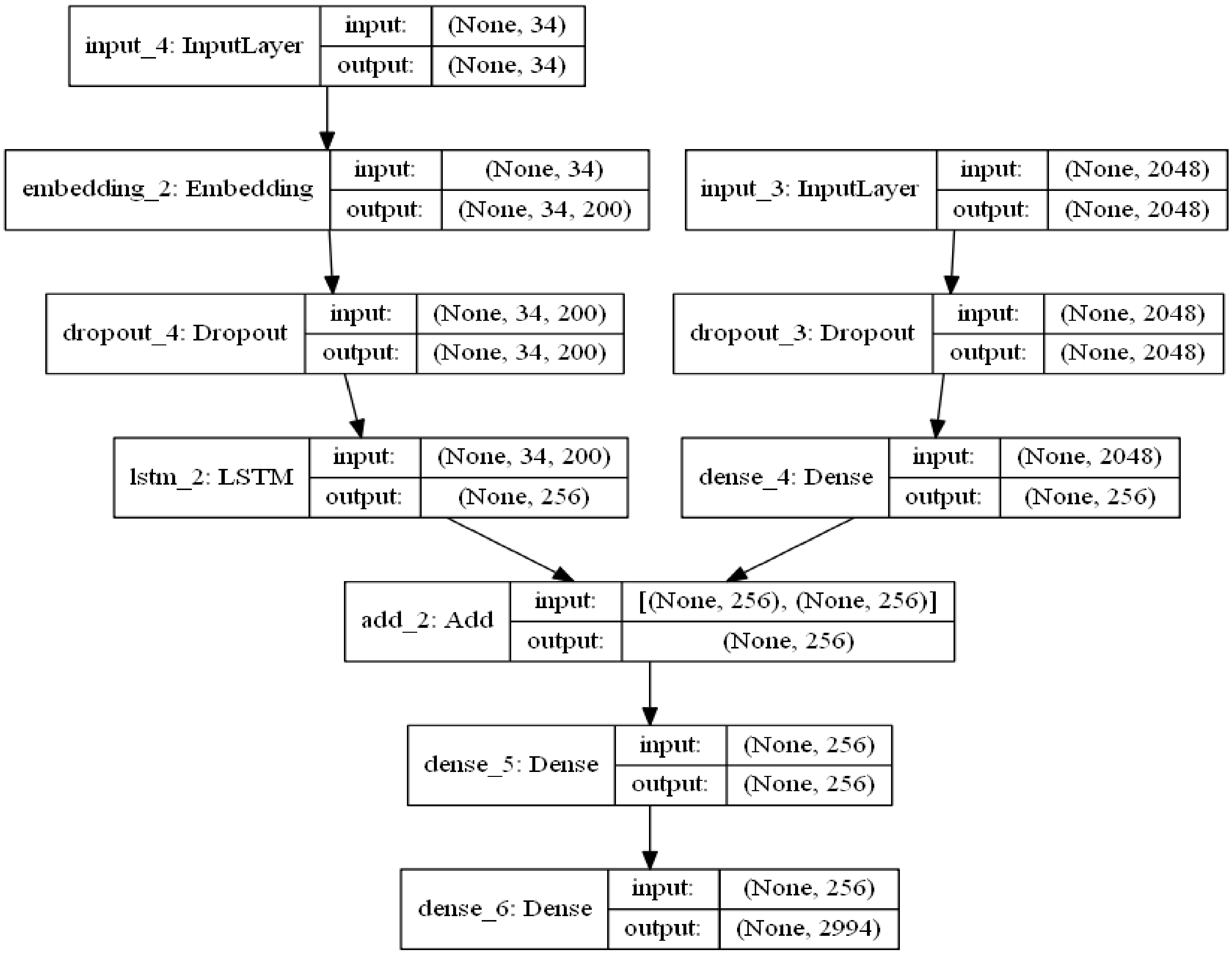
```
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            (None, 34)           0
__________________________________________________________________________________________________
input_1 (InputLayer)            (None, 2048)         0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 34, 200)      1338000     input_2[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 2048)         0           input_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 34, 200)      0           embedding_1[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 256)          524544      dropout_1[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 256)          467968      dropout_2[0][0]
__________________________________________________________________________________________________
add_1 (Add)                     (None, 256)          0           dense_1[0][0]
                                                                 lstm_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 256)          65792       add_1[0][0]
__________________________________________________________________________________________________
dense_3 (Dense)                 (None, 6690)         1719330     dense_2[0][0]
==================================================================================================
Total params: 4,115,634
Trainable params: 4,115,634
Non-trainable params: 0
```



Below is the summary and network structure for model defined with filtered vocabulary.

```
model2=define_model(vocab_filter_size,max_len,'model_filter')
```

```
Model: "model_2"
__________________________________________________________________________________________________
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)      598800      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_4 (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_4[0][0]
                                                                 lstm_2[0][0]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 256)          65792       add_2[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 2994)         769458      dense_5[0][0]
==================================================================================================
Total params: 2,426,562
Trainable params: 2,426,562
Non-trainable params: 0
```



3. Early Results

3a. For Image inputs, I've used Googles pre-trained CNN model Inception V3 with trained weights on imagenet.

```
model_inceptv3=InceptionV3(weights='imagenet')
```

```
model_inceptv3.summary()
```

```
concatenate_2 (Concatenate)       (None, 8, 8, 768)    0           activation_92[0][0]
                                                                   activation_93[0][0]
__________________________________________________________________________________________________
activation_94 (Activation)        (None, 8, 8, 192)    0           batch_normalization_94[0][0]
__________________________________________________________________________________________________
mixed10 (Concatenate)             (None, 8, 8, 2048)   0           activation_86[0][0]
                                                                   mixed9_1[0][0]
                                                                   concatenate_2[0][0]
                                                                   activation_94[0][0]
__________________________________________________________________________________________________
avg_pool (GlobalAveragePooling2 (None, 2048)           0           mixed10[0][0]
__________________________________________________________________________________________________
predictions (Dense)               (None, 1000)         2049000     avg_pool[0][0]
==================================================================================================
Total params: 23,851,784
Trainable params: 23,817,352
Non-trainable params: 34,432
__________________________________________________________________________________________________
```

And remove last SoftMax/Prediction layer, to extract feature maps/image vectors.

```
model_inceptv3_new.summary()
```

```
mixed9_1 (Concatenate)            (None, 8, 8, 768)    0           activation_88[0][0]
                                                                   activation_89[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)       (None, 8, 8, 768)    0           activation_92[0][0]
                                                                   activation_93[0][0]
__________________________________________________________________________________________________
activation_94 (Activation)        (None, 8, 8, 192)    0           batch_normalization_94[0][0]
__________________________________________________________________________________________________
mixed10 (Concatenate)             (None, 8, 8, 2048)   0           activation_86[0][0]
                                                                   mixed9_1[0][0]
                                                                   concatenate_2[0][0]
                                                                   activation_94[0][0]
__________________________________________________________________________________________________
avg_pool (GlobalAveragePooling2 (None, 2048)           0           mixed10[0][0]
==================================================================================================
Total params: 21,802,784
Trainable params: 21,768,352
Non-trainable params: 34,432
__________________________________________________________________________________________________
```

3b. For Caption inputs, processed the caption texts, build vocabularies, word to index, index to word mappings.

Used pre-trained GloVe vectors of 200 dimensions and created word embedding matrix.

```
embedding_matrix_original.shape
```

```
(6690, 200)
```

```
embedding_matrix_filter.shape
```

```
(2994, 200)
```

3c. Created Data Points generator function, which will create features and target for supervised learning (refer section 1c).

3d. Created base merge models with both vocabulary sets (refer section 2).

3e. Hyper Param Tuning, training the models with train set and validation set in progress.

4. Tentative algorithms

Currently using Pre-trained CNN Inception V3 for image embeddings, GloVe for word embeddings and LSTM for language processing.

Will be using LSTM with Attentions to enhance the model.

5. References

1. Guiding the Long-Short Term Memory model for Image Caption Generation
2. Mind's Eye: A Recurrent Visual Representation for Image Caption Generation
3. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention
4. Image Captioning with Semantic Attention
5. Boosting Image Captioning with Attributes