Image Captioning

Debangshu Bhattacharya (MDS201910)
Nilanjan Debnath (MDS201919)

February 2021

Abstract

The task of automatically generating captions from an image is a primary task in the domain of image and scene understanding. With the surge in the prowess of neural network, there has been a huge growth in the field of classification and automatic language generation. However, an image captioning system goes beyond that. It not only has to detect objects in the image but also has to collate them together into a descriptive sentence. This is why it has been a challenging problem in the domain of machine learning as it aims to mimic human's ability to understand visual information and put it into a descriptive language. In this project, we are going to explore, from the very basics, the steps to build an image captioning model. For our experiments, we have used the flickr8k dataset extensively.

1 Introduction

Image classification has been a very pertinent sphere to work on in the field of computer vision. A much more difficult problem is image captioning where the algorithm has to understand the objects in the image and not only report it, but also make a semantically engaging sentence describing the contents of the image.
Image captioning not only helps in problems like automatic captioning of large datasets, but in real life helps in activities like helping blind people interpret an image without external help. Some of the domains where image captioning has found its applications are as follows:

- Self Driving cars for scene understanding.
- Aid to the blind
- Google image search
- Cctv Security

In this project, we first start with the steps to build a basic image captioning model. The idea of an image captioning system is to combine a vision system

(to detect the components of the image) with a language system (generating a descriptive comment based on the components detected by the vision system). Most image captioning systems consist of a encoder-decoder architecture where the encoder is a CNN based model to encode the image into a feature vector. The decoder is a RNN based model to generate a caption based on the feature vector. The decoder model can be a sequence to sequence model where it generates the whole caption in one shot or it can be a sequence to vector model where it generates the next word based on the partial sequence and feature vector as input. For our experiments, we have used the sequence to vector architecture where we generate a word at a time. For choosing words based on model output probability distribution, we have explored both greedy search and beam search algorithms. Finally, we have touched upon the Attention mechanism which are outperforming all other generative models for language generation tasks.

Our project report is divided into the following sections:

- **Background and Literature Review**: This section describes the papers that we looked into and which helped us in solving our problem.
- **Building an Image Captioning System**: This section describes the various components of an Image captioning model. This model is similar to the model in the paper "Show and tell: A neural image caption generator" [4] and is the model which has performed the best in our experiments. This section also talks about how an image captioning model is evaluated along with how BLEU score is computed.
- **Image Captioning with visual attention**: This section describes how visual attention can be incorporated in the model. The model is similar to the model in the paper "Show, attend and tell: Neural image caption generation with visual attention" [1]
- **Experiments and Results**: This section shows the various experiments and results we obtained.
- **Conclusion and Future scope**: This section concludes our report with some areas we can improve our model.
- **Appendix** This section displays some further results and contains our code links.

2 Background and Literature Review

In history of image captioning, most work has been done on co-embedding image and text in the same vector space. Similarly, A lot of work has been done on ranking the objects in the images based on its level of significance. These methods however, fail to address unseen compositions of objects in images, even though the objects might have been seen individually in the training images.

We first looked at the paper "Show and tell: A neural Image Caption Generator" [4]. This gives a very nice and elaborate introduction to the process of building an image captioning system. Like most neural network based architectures for image captioning, the authors use a encoder decoder architecture where the encoder is the vision model and the decoder is the language model. The CNN based image encoder is first pre-trained for the image classification task and then it's weights are frozen. For a new image, the output produced by the last hidden layer of the classification model is the feature vector describing the attributes of the image and this feature vector is provided to the RNN based decoder for caption generation.

We also looked into how attention mechanism is incorporated in these architectures to boost performance. [5][6] If we don't use attention, we get a static encoding of the image as a vector representation whereas attention allows a dynamic representation of the image as and when required. For the attention mechanism, we looked into the paper, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention" [1]. They designed an attention model which automatically learns to describe the content of the images. They introduce two attention based image caption generation methods under a common framework. Firstly a deterministic attention mechanism trainable bystandard back-propagation method and secondly, a stochastic attention mechanism trainable by maximizing an approximate variational lower bound. They used VGGNet weights to train the CNN layers, after which they used an attention model to focus on parts of the image the RNN needs to focus on to get the next word.

3 Building an Image captioning system

The purpose of our project is to understand, from the very basics, the various steps and components that are necessary for building a basic image captioning system. Further enhancements like adding an attention mechanism can be applied on top of this structure but these steps form the base of most image captioning systems.

3.1 Preprocess captions

Since an integral part of the image captioning model is the decoder which is a language model, preprocessing the training captions is an important step for a better language model.

We performed the following preprocessing for our experiments:

- Tokenize words based on NLTK's word tokenize module.
- Lower case folding.

- Remove punctuations.
- Remove all characters but alphabets. This removes all special characters, digits, etc.

An example of preprocessed caption:
Original caption: A child in a pink dress is climbing up a set of stairs in an entry way .
Preprocessed caption: a child in a pink dress is climbing up a set of stairs in an entry way

3.2 Preprocess images

The encoder, being a vision model, requires the images to be preprocessed in a certain way to ensure a good performance. As for our experiments, we are using Transfer Learning with the Inception pre-trained network as our encoder model we perform the same steps as preprocessing that is required by the Inception model, i.e, resize to target size of 299 x 299 and normalize to the range -1 to 1.

3.3 Define model architecture

Most image captioning systems follow an encoder decoder architecture with a structure as given in Figure 1. The ϕ and ψ field boxes indicate attention mechanisms which is a add on functionality. Similarly, the attribute vector A_t may or may not be present in the model we choose. For our purpose, let's explore from start the basics behind these encoder decoder architectures.

3.3.1 Encoder

The purpose of the encoder is to represent all the components of the image as a feature vector. There are two broad categories to which this encoder model may belong to:

1. **Parametric model:** Here, there is going to be a supervised learning algorithm to learn the features of the image as a feature vector. The most prominent architecture used is the Convolutional Neural Network based architectures (CNNs).
2. **Non Parametric model:** Here, the feature vector is attained in generally an unsupervised fashion. For example, using nearest neighbor search from another image database having rich tags and captions.

For our encoder model, we are using the pre-trained Inception model (parametric model). Even though we are not training our parametric model from scratch we give a summary of the steps that can be performed if one desires to train one:

1. Store the most common tokens of the train set as labels for the whole database.

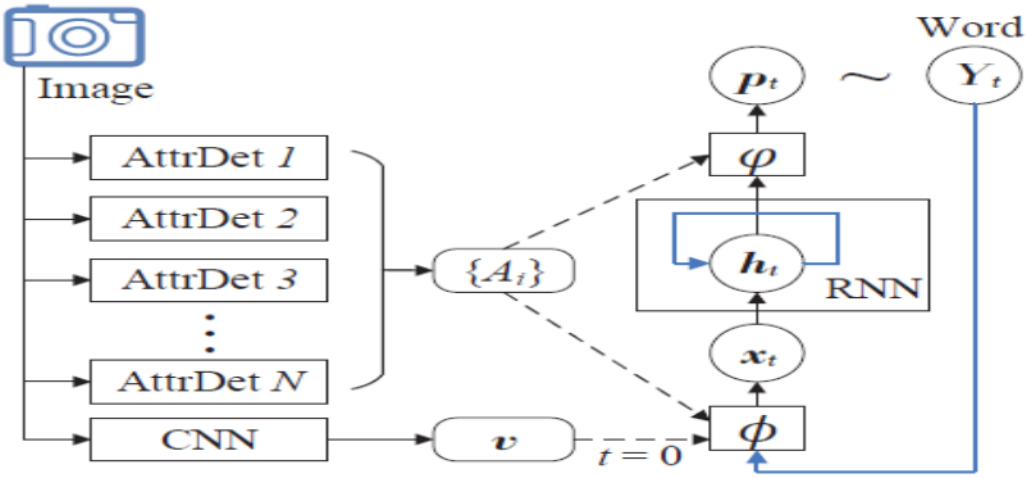


Figure 1: Overview of an Image Caption model architecture

2. Filter out the stopwords from the labels.
3. For each image, find out which labels are present in the true caption.
4. The problem reduces to a multi-label classification problem. Set the model parameters accordingly and train from scratch.

The Inception model is already trained on the ImageNet database consisting of about 1.5 million images with 1000 object categories. The architecture of the Inception model is shown in Figure 3. The example in the figure shows classification for only 10 classes whereas the model has the softmax output of 1000 classes as the final output. As we are only concerned with the feature vector of the image, we do not include the last layer and take the output of the second last layer which is a vector of length 2048, as our feature vector. When we consider Attention, we have to consider local features as well and hence a static feature representation is not sufficient. Hence, there, we take the output of third last layer which is a tensor of shape 8 x 8 x 2048 as our feature representation.

After we have obtained our feature vector representation, we further train a fully connected layer with 256 neurons to adjust the representation to our dataset. Our final encoder architecture is shown in Figure 2 which shows the various components of our vision model. So, to summarize our vision model the following steps occur:

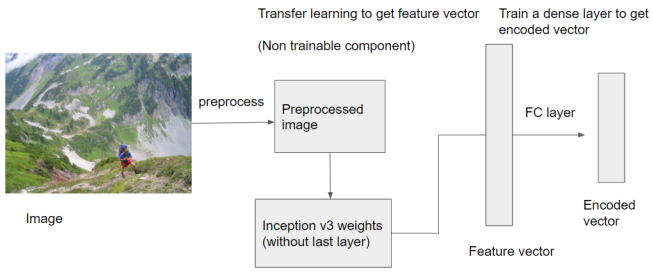


Figure 2: Encoder

- Preprocess the image to get our preprocessed image.
- The preprocessed image is passed to the pre-trained inception model to obtain a feature vector of length 2048.
- The feature vector is further passed to a fully connected layer with 256 neurons to get a encoded vector of length 256. The weights of this fully connected layer is trained based on the dataset for which we want to get our image captioning model. This adjusts the features learnt by the inception model to the dataset we are considering.

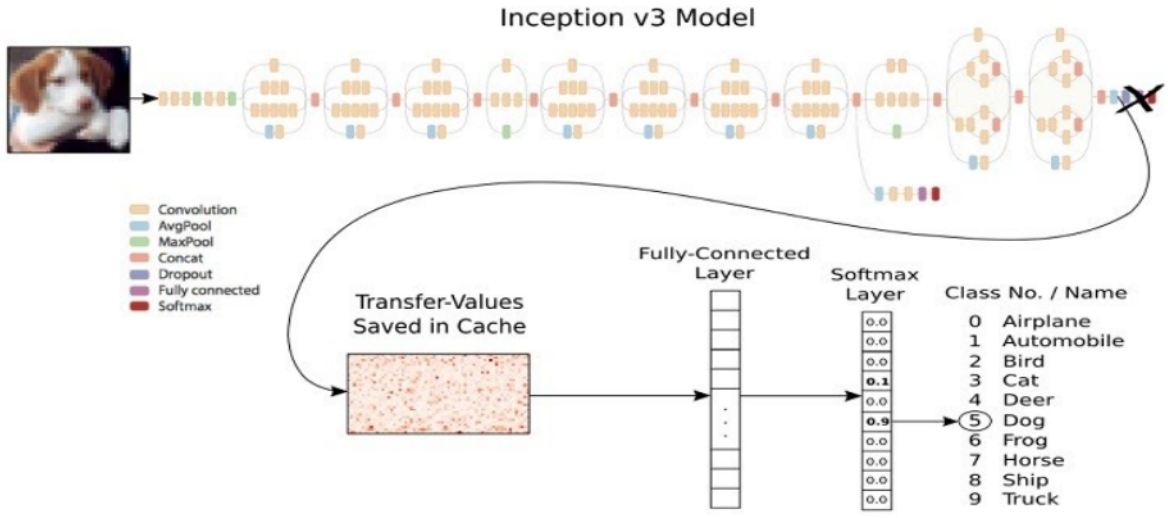


Figure 3: Inception model architecture

3.3.2 Decoder

The architecture of the decoder model is shown in Figure 4. Notice that the input to the decoder model is the CNN encoded vector representation of the image and a partial sequence vector. To understand what partial sequence vectors are, let's revise on how texts are represented as vectors and build up to partial sequence vectors.

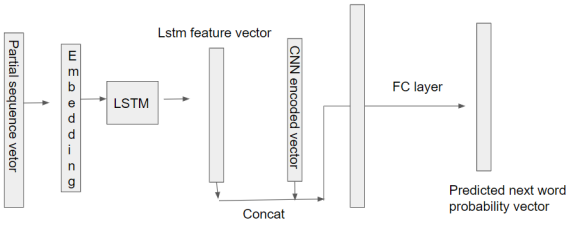


Figure 4: Decoder

Vector Representation of texts: For vector representation of texts, we need a vocabulary. This vocabulary is built from the train set. Generally, the most common tokens are considered in the vocabulary along with the pad token (<pad>), start sentence token (<s>) and end sentence token(</s>). Further, we need to calculate the maximum number of tokens possible in a text. Let's say that this value is d. So, our vector representation of a text will be a vector of length d where each token in the text is replaced by it's index in the vocabulary and padded with the index of the pad token to obtain our required vector. For example, let our vocabulary be {"<pad>", "hello", "how", "is" ,"are", "you"....} a set of size 1000 and let our maximum possible text length be 5. Now, the text "how are you" will be represented by the vector [2,4,5,0,0].

Partial Sequence Vector: Since our decoder architecture is a sequence to vector architecture, the decoder will predict one word at a time. So, a given caption sequence has to be divided into partial caption sequences and only the next word given the sequence till that "time" will be predicted. This forms the basis behind partial sequence vectors. Suppose, the text is " <s> hello how are you </s>". This text is divided into the following partial sequences: "<s>", "<s> hello", "<s> hello how", "<s> hello how are", "<s> hello how are you", "<s> hello how are you </s>". The corresponding target words for each partial sequence respectively will be "hello", "how", "are", "you", "</s>", "</s>. The vector representation of these partial sequences form the partial sequence vectors. Similarly, the vector representation of the target words form the target vector.

Word Embedding: Note that the vector representation we discussed till now just represents each word of the text by it's corresponding vocabulary in-

dex. However, a further level of embedding is required since it is a categorical variable. The simplest representation is a one hot vector representation of each vocabulary index, i.e, a vector of length same as the vocabulary size with all zeros except the element at the vocabulary index set to 1. This representation however does not capture the relationship among words and hence is not a good word to vector representation. So, here lies another component which could be trained from scratch. But for our experiments we are going to use the pre-trained GloVe model [3] which returns a vector representation of a word.

So, for each token in the partial sequence vector, we get a Glove Embedding and the resultant matrix is fed into a LSTM model which provides us a LSTM feature vector. The CNN encoded image feature vector and the LSTM feature vector are concatenated to form our final feature vector and is then fed into our output layer which is a fully connected layer with the number of neurons being the size of the vocabulary and softmax function as the output activation. This ensures that the decoder outputs a probability distribution over the vocabulary.

3.4 Generation strategy

The decoder model gives the probability distribution of the next word for a given partial sequence and feature vector as input(for each word in vocabulary, we get a conditional probability value of that word given the partial sequence and feature vector). We need to choose a word based on these probability values. The simplest strategy is to select the word which has the highest probability value and this is exactly the **Greedy Search** algorithm. Another alternative is **Beam Search** algorithm. Here, we have a parameter Beam width (B). At each 'timestep' we select B words instead of the top word (highest probability) and add it to the partial sequence candidates. At the next 'timestep' for each of the partial sequence candidates, we branch again to generate B more candidates. At the final step we take the top B sequences whose joint probability is the highest. An example showing the comparison of caption generated by greedy search and beam search is shown in figure 5 which shows beam search is doing slightly better than greedy search. We also compared the performance of these two algorithms based on the BLEU and METEOR scores for the first 200 test set images and the results we obtained are shown in Table 1.

Algorithm	**BLEU-1**	**BLEU-2**	**METEOR**
Greedy Search	0.549	0.35	0.353
Beam Search	0.553	0.34	0.36

Table 1: Comparison of greedy and beam search

Figure 5: Comparison of caption generated by greedy and beam search. Beam width 3 is used. The score is the log likelihood of the caption generated which is written in brackets after generated caption. Note how the beam search is performing better in finding out the color of the dog and also displays the variety of options of the water body.

3.5 Model Evaluation

3.5.1 Bi-Lingual Evaluation Understudy(BLEU)

The BLEU score [2] is an evaluation metric which evaluates how 'similar' the generated (candidate) sentence is compared to a list of ground truth references. It is a weighted geometric mean of the modified precision score at different n-grams. The score is always a number between 0 to 1 with 1 being a perfect generation.

Computing the BLEU score: Let our reference sentences be "the cat is on the mat" and "there is a cat on the mat" and suppose our model generated the sentence "the the the the the the the". We know that the precision score is

just the fraction of tokens in candidate which are present in references. BLEU uses modified precision score instead. This is because if we considered just the unigram precision score, it would have been 1 indicating perfect caption generation which is clearly not the case. The modified precision score clips the count of a particular word to the maximum number of times it is present in a reference sentence. So, as "the" appears a maximum of twice in a reference sentence, the clipped count of "the" is 2 and the modified precision score changes to 2/7. The modified precision score of n-gram can be formulated as

$$p_n = \frac{\sum_{C \in Candidates} \sum_{ngram \in C} Count_{clip}(ngram)}{\sum_{C' \in Candidates} \sum_{ngram' \in C'} Count(ngram')}$$

Example: In this example, we show the computation of modified bigram precision score for the candidate sentence "the cat the cat on the mat" and for the same reference sentences as above. Table 2 shows the bigram counts and clipped counts for the candidate sentence. The modified precision score is thus $p_2 = \frac{1+0+1+1+1}{2+1+1+1+1} = 4/6 = 0.67$

Bigram	**Count**	$Count_{clip}$
the cat	2	1
cat the	1	0
cat on	1	1
on the	1	1
the mat	1	1

Table 2: bigrams and counts for the given example

Modified precision score cannot penalize shorter captions however. Suppose the model generates a two word caption which is present in the references but is not a descriptive sentence. In this scenario, the modified precision score will still be high. So a further penalty is imposed on the modified precision score called the **Brevity Penalty**. The Brevity Penalty (BP) is defined as follows:

$$BP = \begin{cases} 1, & \text{if } length(candidate) > 1 \\ \exp\left(1 - \frac{length(reference)}{length(candidate)}\right), & \text{otherwise} \end{cases}$$

Note how the Brevity penalty is penalizing only those captions whose length is lower than the length of a reference sentence. If the length of the candidate was higher then the expected modified precision score itself will drag down the BLEU score.

Finally, the BLEU score is computed as:

$$BLEU = BP * \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

where w_n is the weight of each n-gram that we want to consider. For example, if we only want to consider unigrams and bigrams with equal weightage then N = 2 and $w_1 = w_2 = 0.5$. For our experiments, we considered only unigrams as BLEU-1 and both unigrams and bigrams with equal weightage as BLEU-2 scores respectively.

3.5.2 Metric for Evaluation of Translation with Explicit Ordering (METEOR)

Metric is calculated using harmonic mean of unigram precision and unigram recall with recall being weighted higher than precision(9 times by default). Matching by performing stemming and synonym matching is also performed. Literature study results presented indicate METEOR score has a correlation of upto 0.964 with human judgement whereas BLEU has a correlation of upto 0.817. (Source: https://en.wikipedia.org/wiki/METEOR)

4 Image Captioning with visual attention

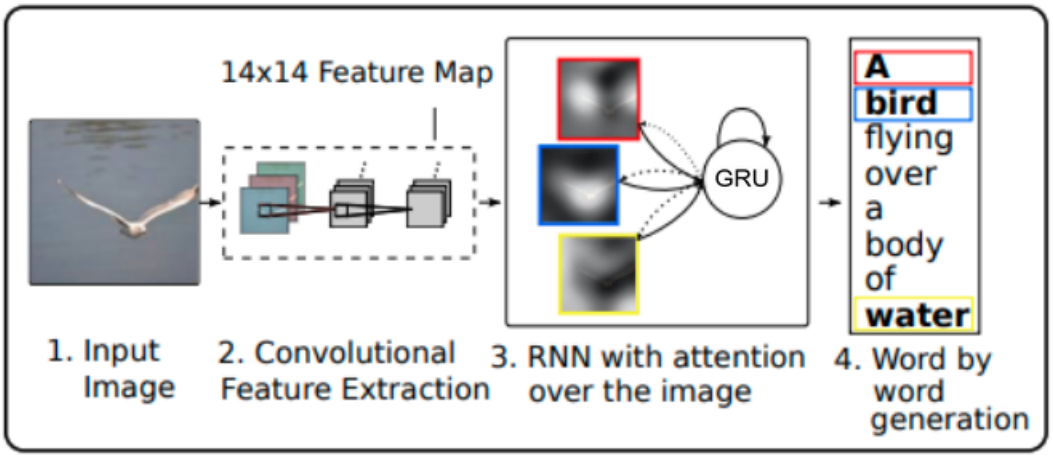


Figure 6: Basic structure of the attention model

The main structure of the model is similar to the last model shown, except for the fact that instead of using a RNN for decoding the image, we use a RNN with attention mechanism to decode the image. The basic structure of the model is as shown in Fig. 6.
In this model, since we use visual attention, through the Convolutional feature extraction, the image is first divided into n parts and with our CNN model with Inception V3 architecture, we compute the representation of each part in d dimensional vectors.

So when the RNN is generating a new word, the attention mechanism focuses on that relevant part of the image, so that the decoder only uses specific parts of the image to compute the next word.

As in the image, the inception v3 model divides the image into n parts and

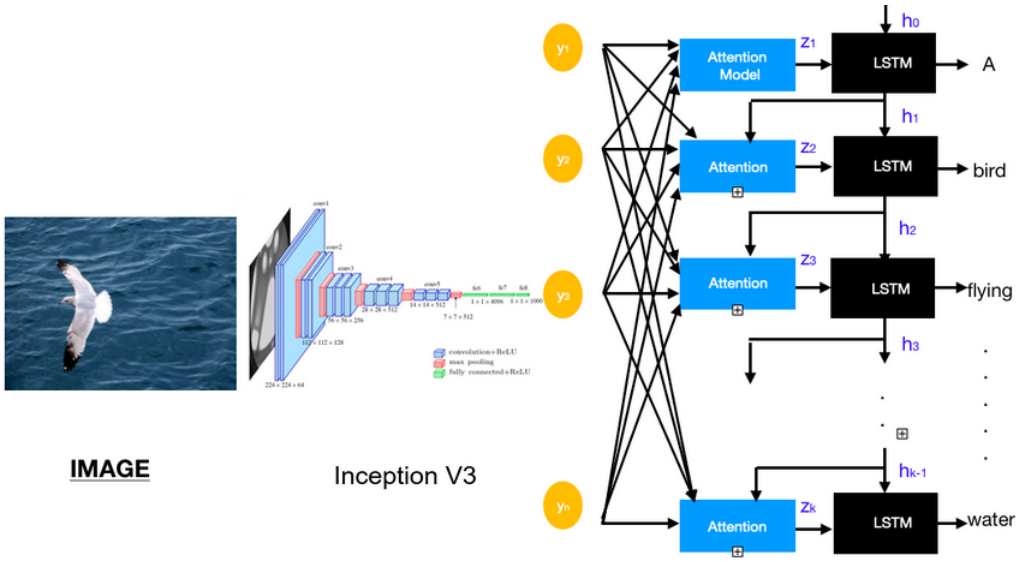


Figure 7: How the next word is generated

provides a feature representation for each of the parts. Upon each part we run an attention model to get weights α_i for each part. These weights will determine where the decoder focuses on for the next word.

Now, let the n, d dimensional feature vectors for n different parts of the images be denoted by $\alpha = \{\alpha_1, \alpha_2, \alpha_3 ... \alpha_n\}, \alpha_i \in R$.

The attention model takes in the hidden layer of the RNN and the feature vectors as input and provides weights for each feature vector. On the weights, we use softmax to get a probability distribution for each part of the image.

$$e_{ti} = f_{att}(a_i, h_{t-1})$$

$$\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{k=1}^{n} exp(e_{tk})}$$

In the above equations, e_{ti} is the weight attributed to part i of the image at timestep t. h_{t-1} is the hidden layer of the RNN, and a_i is the feature vector for i-th part of the image.

$$f_{ATT} = V_{attn}^T * tanh(U_{attn} * h_{t-1} + W_{attn} * a_i)$$

The attention model is really just a simple feed forward neural network which takes the inputs, does a linear transformation over it, applies a non linearity

(a) Young boy with a black tshirt off

(b) How attention works

Figure 8: The highlighted parts are the ones which at a particular timestep, the attention model deems important to calculate the next word from

and one more linear transformation.

$$z_t = \sum_{i=1}^{n} \alpha_{ti} a_i$$

So now that we have all the weights for each part, we calculate the weighted sum and pass this weighted sum z_t, the previous state of the decoder and the previous word generated by the RNN through the recurrent neural network to generate the next word. This process is repeated till we hit our end marker.

We can see an example of this happening in figure 8. As we can see, the different blocks are all the parts the image has been separated into. We use our CNN over all these to give the n feature vectors. Now, using the attention model, we

calculate the weights for each part of the image. The highlighted parts are the ones which at a particular timestep(Figure 8b). given the previous word and state of the decoder are deemed to be more important by the attention model. Now, as we said earlier, we take the weighted sum to create the context vector z_t and pass the context vector, the previous state and the last generated word through the RNN to generate the next word of the sentence till we hit the end marker.

5 Experiments and Results

5.1 Dataset

We used the flickr8k dataset (https://www.kaggle.com/adityajn105/flickr8k) extensively for our project. The dataset consists of 8091 images with each image annotated by 5 captions. We performed a 70-30 train test split on the dataset resulting in 5663 images in train set and 2428 images in test set.

5.2 Training

We trained the models on batches with each batch consisting of 30 images. Figure 9 shows the training curve for the best model we obtained where we trained the model for 10 epochs. The loss function used was categorical cross-entropy loss and the optimizer used was Adam with a learning rate of 0.001.

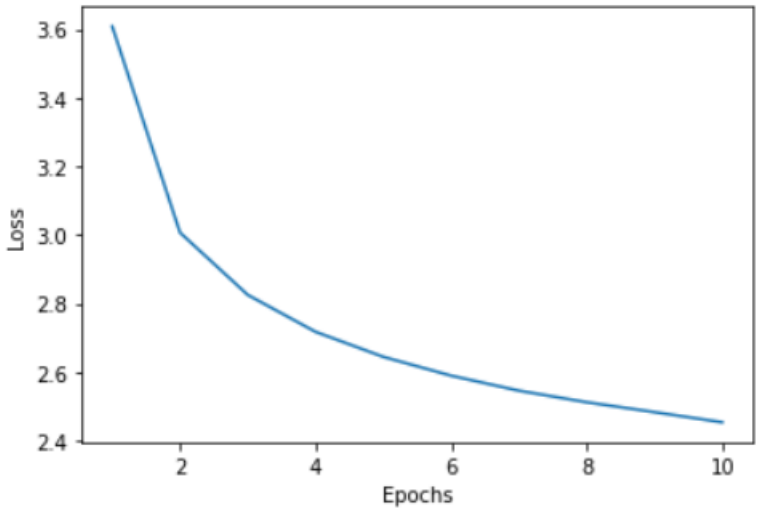


Figure 9: Training curve of best model observed

The training curve for the best model we observed is shown in Figure

5.3 Results

5.3.1 Evaluation

We evaluated our model performance on the 30% test set we had split earlier. In table 3, we note down our best model's performance on our test set (which contains 2428 images) along with the performance of the model given in [1]. Note that the authors have considered a different test set than ours (they have considered 1000 images in the test set while we considered 2428 test set images) and also has not considered Brevity Penalty. So we cannot compare the metric values as a means of comparing the model but we display the results here anyways as it gives a good idea about the best results that has been observed for image captioning on Flickr8k dataset.

	BLEU-1	**BLEU-2**	**METEOR**
Our best model (on 2428 test images)	0.54	0.33	0.34
Show, Attend and Tell model (1000 test images)	0.67	0.457	0.20

Table 3: Performance of our best model and the model given in the paper Show, Attend and Tell (different test sets)

5.3.2 Some examples

Figure 10 shows two examples where our model is performing very well and Figure 11 shows two examples where our model is not performing very well.

6 Conclusion and Future Scope

We have been successful in getting familiar with the various steps and components in building an image captioning system from scratch. We observed results which are not perfect but can be still considered decent given the challenging nature of the problem. We also touched upon how visual attention is incorporated in the image captioning model.

For further enhancements in model performance, we can consider the following steps which we consider as our future scope in this project:

- Using semantic attention on top of visual attention.
- Training more hidden layers in the encoder architecture to learn a better feature representation of the image.
- Experiment with other pre-trained architecures like VGGNet and Resnet as the encoder architectures.

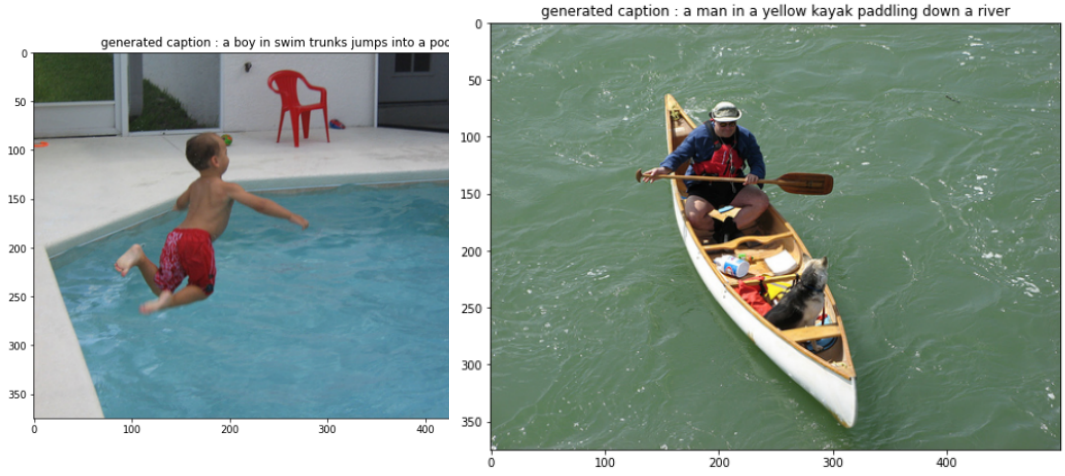


Figure 10: Two test set examples where model is performing very well. Subfigure 1(left image) shows the image of a boy jumping into a pool in his swimming trunks and the subfigure 2(right image) is of a man kayaking in river. The captions generated are identical to the true captions.

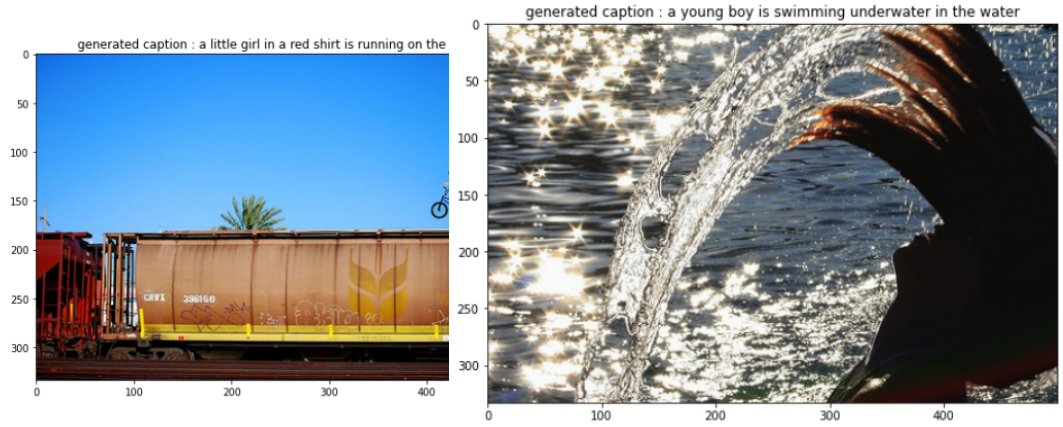
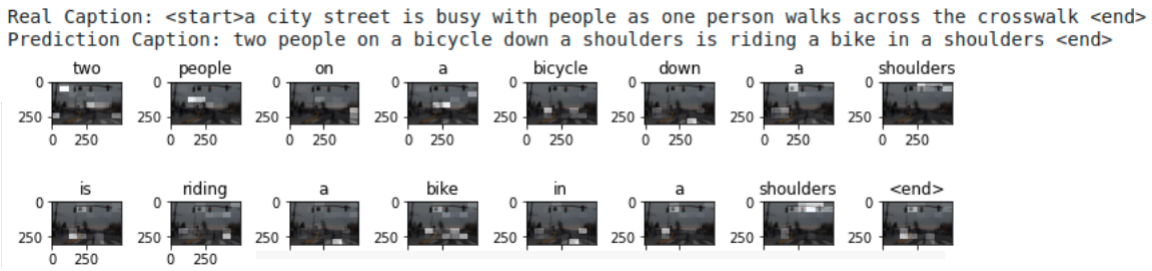


Figure 11: Two examples where our model is failing to caption well. Subfigure 1(left image) captions the image as " a little girl in red shirt is running on the grass" which is absolutely incorrect. Subfigure 2(right image) captions the image as "a young boy swimming underwater in the water" is incorrect in the sense that is not a young boy but a woman and "underwater in the water" is a wrong semantically but it still captures the essence of the image.

(a) a city street is busy with people as one person walks across the crosswalk



(b) How attention works for this image

Figure 12: An example where our model finds minute details within the image. This is a very interesting example which goes on to show the reach of attention mechanism. While the real caption a city street is busy with people as one person walks across the crosswalk is very different from the predicted caption, two people on a bicycle down a shoulders is riding a bike in a shoulders, which on first glance does not make much sense, but when we look closely to the middle left, we can see a person carrying a bicycle on his shoulders. Attention hence can be used to find out very minute details from images with high precision.

References

[1] Ryan Kiros Kyunghyun Cho Aaron Courville Ruslan Salakhutdinov Richard S. Zemel Yoshua Bengio Kelvin Xu, Jimmy Lei Ba. Show, attend and tell: Neural image caption generation with visual attention. In *ICML'15: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, page 2048–2057, 2015.

[2] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[3] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[4] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[5] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[6] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659, 2016.

7 Appendix

7.1 Code Links

1. **Best model so far** This contains the best model we got so far by training on flickr8k dataset (70% train and 30% test sets). For generation, we have used greedy search here.

2. **Image Captioning with Visual Attention** This notebook contains implementation of image captioning with visual attention

3. **Image Captioning comparison of beam and greedy search generations** This notebook includes our implementation of beam search and compares the performance of greedy and beam search in generation of sentence.

4. **Getting started link:** Here, we have implemented the model and trained on just 100 images from the flickr30k dataset. This was to test the working soundness of the architecture.

7.2 More Results

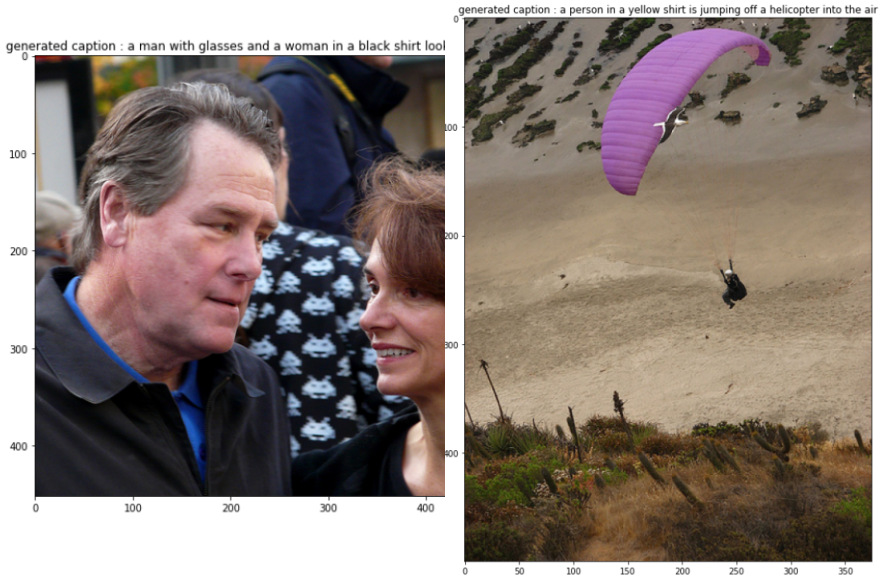
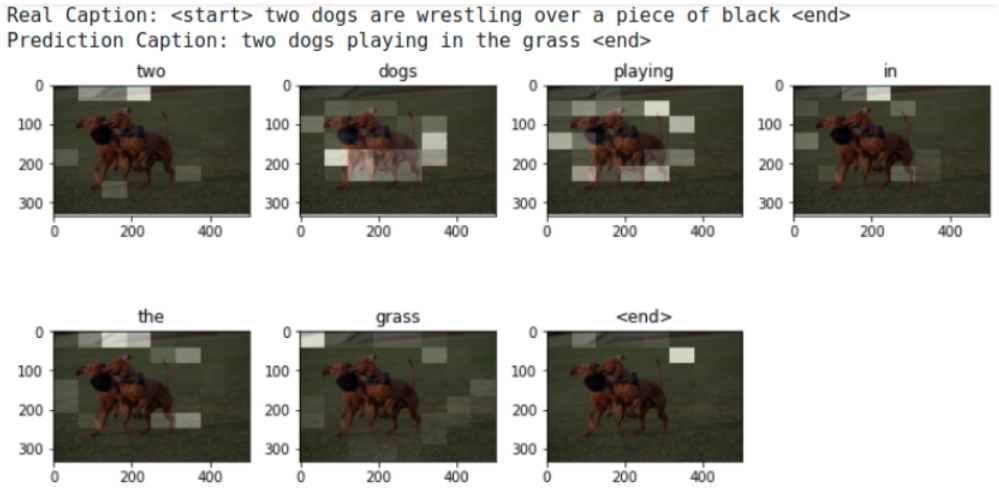


Figure 13: Some more examples of captions generated on test set images using our image captioning model. The caption generated for the left image is "a man with glasses and a woman in a black shirt is looking at something". The caption generated for the right image is "a person in a yellow shirt is jumping off a helicopter into the air". While for the first image the man does not have glasses and for the second image there is no helicopter and the person is not in a yellow shirt, the model still captures the essence of the image

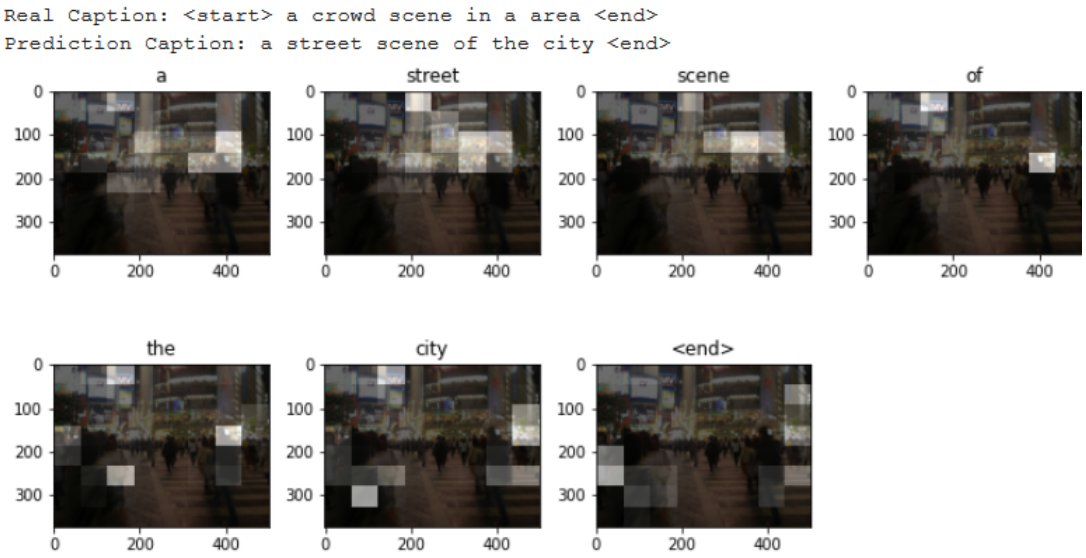
(a) Two dogs are wrestling over a piece of black



(b) How attention works

Figure 14: The original image and true caption is shown in subfigure 14a whereas subfigure 14b shows how the attention model is working on this image and generates the caption "two dogs playing in the grass" which is not bad

(a) A crowd scene in an area



(b) How attention works for this image

Figure 15: The real caption a crowd scene in a area and our predicted caption a street scene of the city are nearly similar

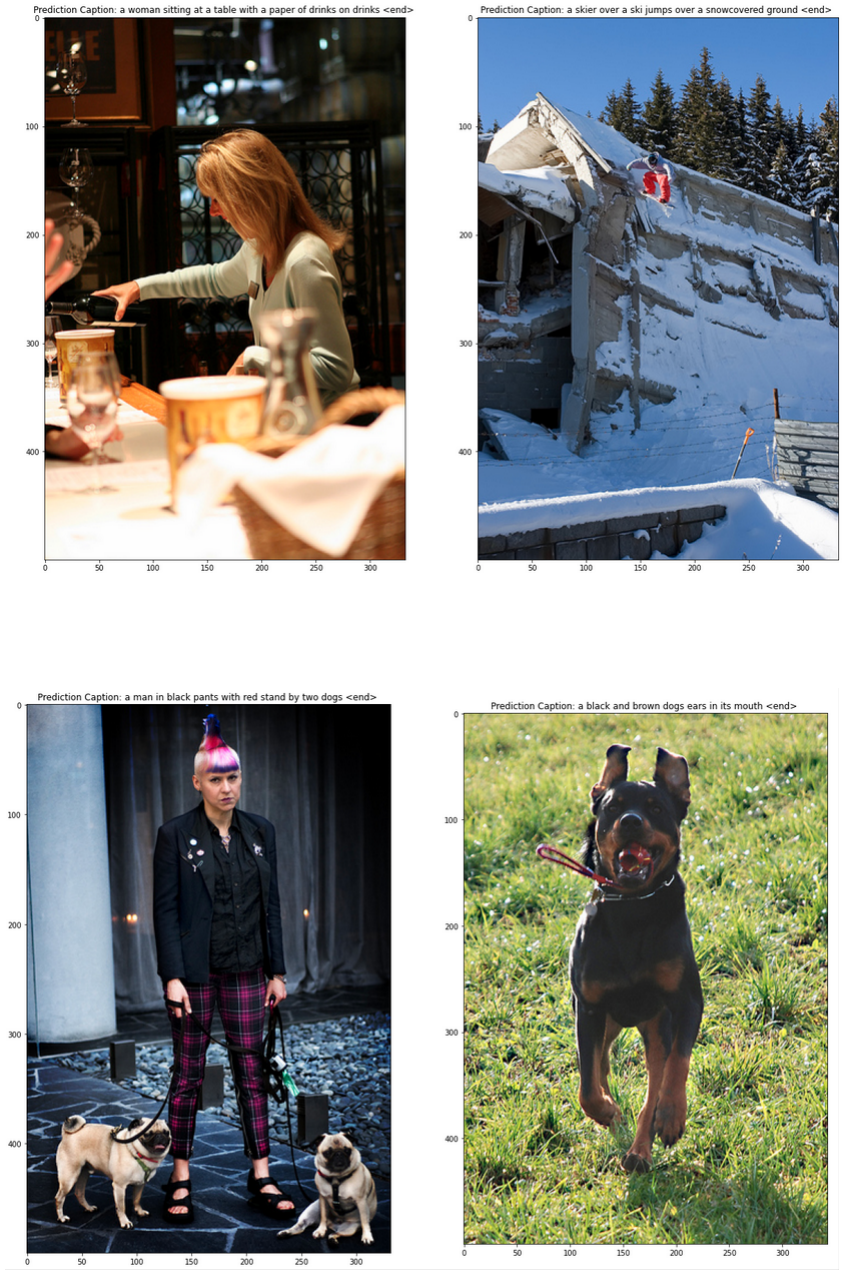


Figure 16: Some more examples