Analysis of Recurrent Neural Networks For Image Captioning Systems

A DISSERTATION

Submitted in the partial fulfilment
Of the requirements for the award of the degree of

Bachelor & Master of Technology in Information Technology
With Specialization in Software Engineering

By:

Swapnil Jain
(ISM2013004)

Under the Guidance of:
Prof. U.S. Tiwary
Professor
IIIT-Allahabad

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD

(A UNIVERSITY ESTABLISHED UNDER SEC.3 OF UGC ACT, 1956 VIDE NOTIFICATION NO.F.9-4/99-U.3 DATED 04.08.2000 OF THE GOVT. OF INDIA)

A CENTRE OF EXCELLENCE IN INFORMATION TECHNOLOGY ESTABLISHED BY Ministry of H.R.D., GOVT. OF INDIA

Candidate Declaration

I do hereby declare that the work presented in this thesis entitled “Analysis of Recurrent Neural Networks in Image Captioning Systems”, submitted in the partial fulfilment of the degree of Bachelor of Technology & Master of Technology with specialization in Software Engineering at the Indian Institute of Information Technology, Allahabad is an authentic record of my original work carried out under the guidance of Prof. U.S. Tiwary due acknowledgements have been made in the text of the thesis to all other material user. This thesis work was done in full compliance with the requirements and constraints of the prescribed curriculum.

Place: Swapnil Jain
Date: ISM2013004

Certificate from Supervisor

I do hereby recommend that the thesis work prepared under my supervision by Swapnil Jain titled “Analysis of Recurrent Neural Networks in Image Captioning Systems” be accepted in the partial fulfilment of the requirements of the degree of Bachelor of Technology & Master of Technology with specialization in Software Engineering.

Place:	Prof. U.S. Tiwary
Date:	IIIT Allahabad

Certificate of Approval

The forgoing thesis is hereby approved as a credible study in the area of Information Technology and its allied areas carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it is submitted.

Signature & Name of the Committee Members:

Plagiarism Report

(Coming Soon)

Acknowledgements

The satisfaction and euphoria that accompany the successful completion of any project work would be impossible without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts. This project was not only a endeavour but also an interesting learning experience for me and it bears the imprint of a number of persons who directly or indirectly were a source of help and constant encouragement.

I would like to express my sincere and heartily thanks to my mentor Prof. U.S. Tiwary for his continuous motivation and guidance. His valuable suggestions, comments and support were an immense help for me. I am grateful to him for taking out time from his busy schedule and being very supportive in guiding my work. I am glad to be a part of his SILP Lab, a place that offers excellent environment for research.

Special thanks to my parents & my seniors Sudhakar Mishra, Rohit Mishra & Shirkant Malviya for their efforts & cooperation. The interesting and informative discussions we had together, greatly contributes to the completion of this work.

Swapnil Jain

Abstract

Recurrent neural networks are usually seen as a generation module for image captioning systems. In this work, two views of a recurrent neural network are proposed. First view is the same as in most literature where an image is put in recurrent neural network with the textual features to generate the caption for that particular image whereas second view suggests that recurrent neural network instead of generation can be used to encode the text and then that encoded form of text can be put in with visual features to generate the caption at a later stage. On comparing both architectures, it is found out that second view outperforms the first view from which it can be concluded that recurrent neural networks should be viewed better as encoders instead of being used as generation module.

In this work, a model is also presented in which two visual features extractors are combined together to encode an image. Both views of recurrent neural networks are considered to perform the experiments. On experimenting, it is observed that even on combining two visual encoders, results are not better.

Recent works have shown that attention model can be viewed as a state of the art for the image captioning systems. In this work, rather than generating state of the art architecture for image captioning systems a discussion is performed on visualization of feature maps that attention model used and how it works.

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

1.1. <u>Overview</u>

To generate short and concise textual description from an image automatically is a rudimentary problem in machine learning that attracts concepts of both natural language processing & computer vision. Human eye just by mere glance of scenery can describe the minute details of that scenery. But for a computer, task is rather much difficult and challenging. Visual to caption generation not only include extraction of the objects in an image but also the actions and attributes pertaining to those selected and then comes the science of construction of a sentence that describes the connections of those objects, their attributes and actions in an image. In another terms, it can be said that it is a problem of translating a matrix of pixels as source to the English as target language. Being difficult on one side, it is also quite an interesting problem on another.

1.2. <u>Motivation</u>

In most literature, Image Captioning Systems depends on neural networks, where image is fed to the convolutional neural networks to extract the visual features of an image and textual descriptions are generated using recurrent neural networks like long short term memory networks (LSTM). These systems use visual feature extractors in such a way that recurrent neural networks are biased towards generating words from the given vocabulary in such a way that their sequence provides a caption pertaining to that image. In short, recurrent neural networks in these models are being used as generators and visual features extracted from convolution neural networks condition the generation of best textual description for that image. However, rather than using recurrent networks as generators, other modes can be exploited in the way recurrent and convolutional networks are combined, in which the role of recurrent neural networks can be viewed as encoding of sequences as primary component not the generation.

1.3. <u>Problem Definition</u>

The main objective of this work is to experiment on the modes in which recurrent neural networks can be combined with the convolutional neural networks. In particular, two modes are being considered as part of the research.

- <u>Mode 1</u>: where the image is fed into the same recurrent neural network where the words are processed. This mode is named “Implant Mode”, that is image is implanted into the recurrent neural network.

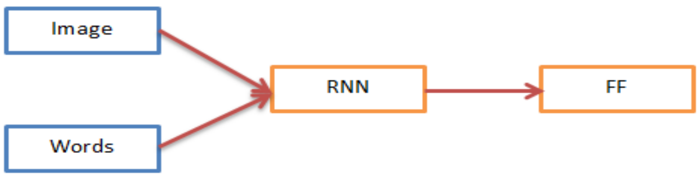


Figure 1.1: Implant Mode

- Mode 2: where image is merged with the encoded words obtained from recurrent neural network into a multimodal layer. This mode is named “Merge Mode”, that is image features are merged with encoded text.

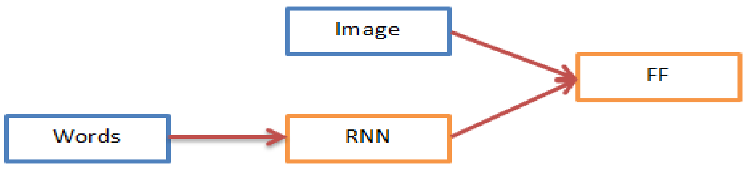


Figure 1.2: Merge Mode

1.4. Applications

Several applications can be availed on solving this problem. One of the applications is searching of images where captions can be generated and then image search techniques from information retrieval can be used to perform the required task. Another is to tell stories from album uploads like a story teller, it can be an attractive task to perform experiments on. Visually impaired people can get help from the image captioning systems that can make them understand the web better like some sort of website descriptor. More applications will be available as more research continues in this works.

1.5. Organization of Thesis

The work has been presented in seven chapters. The detailed description and summary of these chapters are as follows.

Chapter 1. Introduction

Chapter 2. Literature Review

This chapter includes data about all the previous work done in this research field and builds a foundation for the work presented ahead.

Chapter 3. Requirements

This chapter includes the description of the dataset used in the research work. Tools and software that are used in this work are also explained.

Chapter 4. Methodology

The basic procedure of how the proposed work is achieved and all its modules are explained in this chapter. A short discussion on feature map visualization and attention model is also performed in this chapter.

Chapter 5. Experiments

As the name suggests, this chapter describes the experiments that are performed. Experiments include comparison between merge and implant architecture, exploiting recurrent network behaviour on increasing number of captions per image, embedded word vector length and the combined feature extractors' model.

Chapter 6. Results & Analysis

After experimenting through the different architectures of the image captioning systems, results obtained are presented in this chapter. These results are further analysed and inferences are obtained.

Chapter 7. Discussion

The study of what is done and what can be done builds the base of this chapter. A discussion is performed on the conclusion and future scope of the work.

2. LITERATURE REVIEW

2.1. Approaches

The process of generating textual description from visuals (Bernardi et al. 2016) has been raised as an important test suite for the solutions of the problem of importing symbolic or linguistic information in conceptual data (Roy, 2005). Mostly captioning systems focus on concrete textual descriptions that is to generate captions that describe what is within the image strictly. Recently, researchers are moving ahead, by analysing visual question answering (Antol et al., 2015) and image based narration like story teller (Huang et al., 2016).

There are three main types of image captioning techniques or approaches (Bernardi et al. 2016).

- Template Based: Image captioning learning models that are based on algorithms of computer vision to detect the objects and extract the visual features from the image. Further these extracted objects and features are used in caption generation stage (Kulkarni et al., 2011; Mitchell et al., 2012). The caption generation stage is somehow related to the modules used in well-known natural language generation pipeline architecture.

- Retrieval Based: Some systems used the task of image captioning as a retrieval problem. In these systems, captions or its parts are computed by analysing the proximity or the relevance of sentences or sub-sentences in the training data for an image. This is attained by using either a multiple modal (Socher et al., 2011) or single modal (Ordonez et al., 2011) space. Many such information retrieval techniques depend on neural networks to handle both visual and textual features.

- Neural-Network Based: Third approach is the main approach that also relates to this work. In this approach, image captioning systems depends on generation of novel captions by exploiting recurrent neural networks. This approach uses convolutional neural network to extract the visual features. These features are further fed into the recurrent neural network to allow the network to sample terms from the obtained vocabulary such that it generates a sequence of words which fits the description of the image.

Third Approach is the main focus of this work. Convolutional Neural Network is mainly used to extract image features which conditions the generation of the best textual description for an image. Rather than using recurrent networks as generators, other modes can be exploited in the way recurrent and convolutional networks are combined, in which the role of recurrent neural networks can be viewed as encoding of sequences as primary component not the generation.

2.2. Related Work

Automatic caption generation of an image has attracted a lot of attention in both computer vision and natural language domain connecting to artificial intelligence. This task may seem difficult but recent researches in the area of neural networks subdomain of artificial intelligence, image processing and natural language processing has made a way to describe the contents of image like objects, their related actions and attributes accurately. Several state of the art techniques like long short term memory networks, convolutional neural networks are being leveraged now a day. Several datasets of images and their concise textual description are also available to achieve the same.

Image captioning systems based on involvement of neural network are also classified in two types.

- Pipeline Architecture
  In this type, both the language model and visual detectors works separately. First the visual detectors do the task of extracting visual features and then sentence is inferred from the language model.

- End-to-End Architecture
  Everything from language to visual model is encapsulated in one neural network. It is typically used in most literature.

A decent literature survey for image captioning is performed on the papers related to this work. This is shown as below with each paper & its corresponding methodology, metrics etc.

- Show and Tell: A Neural Image Caption Generator
  The model described in this paper is the most basic and probably the first which can be trained from end-to-end. The log likelihood of the probability distribution of the sequence of words conditioned on the image is being maximized as shown in below equation.

$$\theta^* = \arg\max_{\theta} \sum_{(I,S)} log p(S|I;\theta)$$

The motivation arises from the recent researches in the area of machine translation. In machine translation, recurrent neural network is used both as encoder and decoder. Encoder recurrent neural network is replaced by features extracted by convolutional neural network. Start marker is used as an input. This model is known as Neural Image Caption. They also used beam search. BLEU evaluation metrics are used to evaluate and compare the result and the model for tuning the model selection and its hyper parameters.

- <u>Deep Visual-Semantic Alignments for Generating Image Descriptions</u>
  Have a large corpus of sentences but which part of the sentence corresponds to which part of image is not mentioned. Karpathy and Fei-Fei 2015 tried to learn these relations and extract the image description. By embedding both image and sentence to a common space alignment was inferred. A multimodal RNN was proposed which extract captions from images, train model on inference based data and test on region level annotation.

  Alignment of caption parts to the corresponding objects or scenery in the image, a dependency tree based relations were inscribed into regions of an image with a rank-based formula. To obtain the word representations in the caption, bidirectional recurrent neural network was used. This neural network maps word representations and regions of the image such that similar concepts occupy nearby regions of the image.

- <u>DenseCap: Fully Convolutional Localization Networks for Dense Captioning</u>
  Johnson et al. 2015 provide a dense captioning task where set of explanations is generated for each detected image region. Henceforth, this process can be named as object detection task on detecting object with mapping to one word and image captioning when image region is detected.

  Main contribution includes dense localization layer which finds out interested regions in image and extractions of activations of those region via bilinear interpolation.

  VGG-16 convolutional neural network is used to extract the image features for the generation of captions. Final pooling layer is removed from it. So providing image as input, 512 feature maps or uniformly sampled regions are obtained which will be fed to the localization layer. After that these features are passed through a long short term memory network that conditioned the image feature vector. The loss function used is categorical cross entropy. METEOR evaluation metrics was used in this work, since it highly correlates with human judgements.

- Image Captioning with Deep Bidirectional LSTMs
  Wang et al. 2016 proposed two novel deep bidirectional models. The depth of non-linear transition was increased in different ways. It learnt hierarchical visual-language embedding. Performance of this model was on par with state of the art even without using techniques like attention or object detection. For prevention of over fitting in training deep models, data augmentation methods like multi scaling, multi cropping and vertical mirror are used.

  Convolutional neural network like Alexnet and vgg-16 is used to learn image features. Bidirectional LSTM to learn sentence features. Motivation to use deep bidirectional LSTM comes from the working of cognitive mind.

  According to this paper, LSTM are deep but the depth is horizontal that weights are used again and again which inhibits the learning of representative features. To achieve the vertical depth many LSTM layers can be stacked together or a multi-layer perceptron can be put in between.

  With being trainable as end-to-end, this model's loss is sum of loss obtained from both forward and backward LSTMs. To generate the final caption most probable word was taken from both backward and forward prediction.

- Show, Attend and Tell: Neural Image Caption Generation with Visual Attention
  Attention is preferred. One vector representation of whole image is not good to understand the scene in an image, its actions and attributes. Attention allows dynamically using features when needed. Till now, last layer features from a convolutional neural networks were used which describes the most important part of the image. It is one way but not the best to have more concise and explained textual description of an image. So there is requirement of low level features. To handle these features, two types of attention are proposed by Xu et al. 2015, hard and soft attention.

  Low level features were extracted from convolutional neural network. These features are termed as annotation vectors. Each vector responds to a part of the image.

  In decoder module, LSTM is used. As input embedding of the word, the vector of the previous time step and a context vector are fed into the LSTM. This context vector is responsible to frame the visual information corresponding to a particular image location provided as input. This attention model performs better than most of the described architectures and considered as state of the art method.

From study of all these, it can be inferred that one way to use recurrent neural network to predict image captions is to implant both visual and language features directly in recurrent neural network. Several architectures are presented to show this view. However a different and a peculiar view of recurrent neural network is also considered where these networks are typically used as an encoding model or language model used to encode the sequence of words of varying length. The encoded form of sequence obtained from this model can be then combined with the visual features ahead in a multimodal layer. This concludes to view recurrent neural network as an encoder to encode the text rather than a generator. This architecture is also included in literature but in lesser details than the previous architecture (Mao et al., 2015; You et al., 2016).

3. Requirements

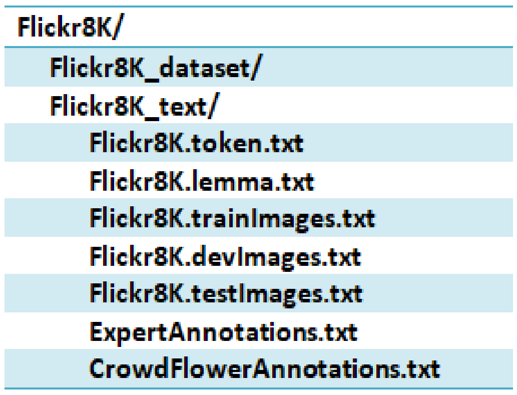
3.1. Dataset Description

FLICKR 8K [1] is the dataset that is used for experimentation in this work. This dataset includes images that are collected from Flickr website. Flickr is an image hosting service and video hosting service. It was created by Ludicorp in 2004.

University of Illinois, Urbana, Champaign has the sole link of this vast dataset. These images do not hold any public figure or place therefore entire image caption generation depends solely on the different objects present in the image.

This collection consists of 8000 images that are each mapped with five different captions which offer quite clear explanations of the salient attributes and actions taking place in the images.

The directory structure of FLICKR 8K dataset is given below:



```
Flickr8K/
    Flickr8K_dataset/
    Flickr8K_text/
        Flickr8K.token.txt
        Flickr8K.lemma.txt
        Flickr8K.trainImages.txt
        Flickr8K.devImages.txt
        Flickr8K.testImages.txt
        ExpertAnnotations.txt
        CrowdFlowerAnnotations.txt
```

- Flickr8K_dataset/
  This folder contains all the images collected from the Flickr website. There are total of 8000 images in this dataset.

- Flickr8k.token.txt
  The images' captions of the Flickr8k Dataset are written in this file. The first column is the ID of the image concatenated with ".jpg# [caption_number]" and then the caption itself.
  Example: *"1000268201_693b08cb0e.jpg#1 A girl going into a wooden building."*

- Flickr8k.lemma.txt
  If one wants to use or apply their learning model on the lemmatized captions, then this file comes into existence. It includes lemmatized version of the captions given in Flickr8K.token.txt.

Example: "*1305564994_00513f9a5b.jpg#4 Two people in race uniform in a street car.*"

- Flickr_8k.trainImages.txt
  FLICKR 8K already provides the division of images in training, validation and testing sets. This file includes total of 6000 image ids that should be used for training purposes.

- Flickr_8k.devImages.txt
  To develop or validate the model FLICKR 8K provides us the development/validation set of images. It consists of total of 1000 image ids that should be used for validation purposes.

- Flickr_8k.testImages.txt
  To test the model, the test set of images is also provided. This file fulfils that requirement. There are a total of 1000 image ids that should be used for testing purposes.

- ExpertAnnotations.txt
  The expert judgments are included in this file. Image & Caption IDs are the first two columns. Caption IDs are in the format <image file name>#<0-4>. The next three columns provide the expert judgments for that particular image-caption pair.

  Scores are ranged in [1, 4] discretely. The rating of 1 indicates that the caption is not able to provide the explanation of the image in any way; a rating of 2 indicates that the caption gives some aspects about the image but can't provide the explanation of the image; a rating of 3 indicates that the caption almost provide the explanation of the image but can't describe it properly, and a rating of 4 indicates that the caption fits the description of the image.

- CrowdFlowerAnnotations.txt
  CrowdFlower is a human-in-the-loop machine learning and artificial intelligence company based in San Francisco. It exploits intelligence of human to do simple tasks such as writing text or providing explanation of images to train machine learning algorithms.

  This file possesses the CrowdFlower judgments. Image & Caption IDs are the first two columns. The 3rd column is the percentage of Yes, the 4th column is the sum of Yes, and the 5th column is the sum of No. A Yes means that the caption fits the description of an image with probable minor mistakes while a No means that the caption is not able to provide the explanation for that image. Three or more judgements are attached to each image-caption pair.

In figure 3.1, some of the sample images from FLICKR 8K are represented. These images are in different dimensions but to show these images, all are fit together in a box of same frame.

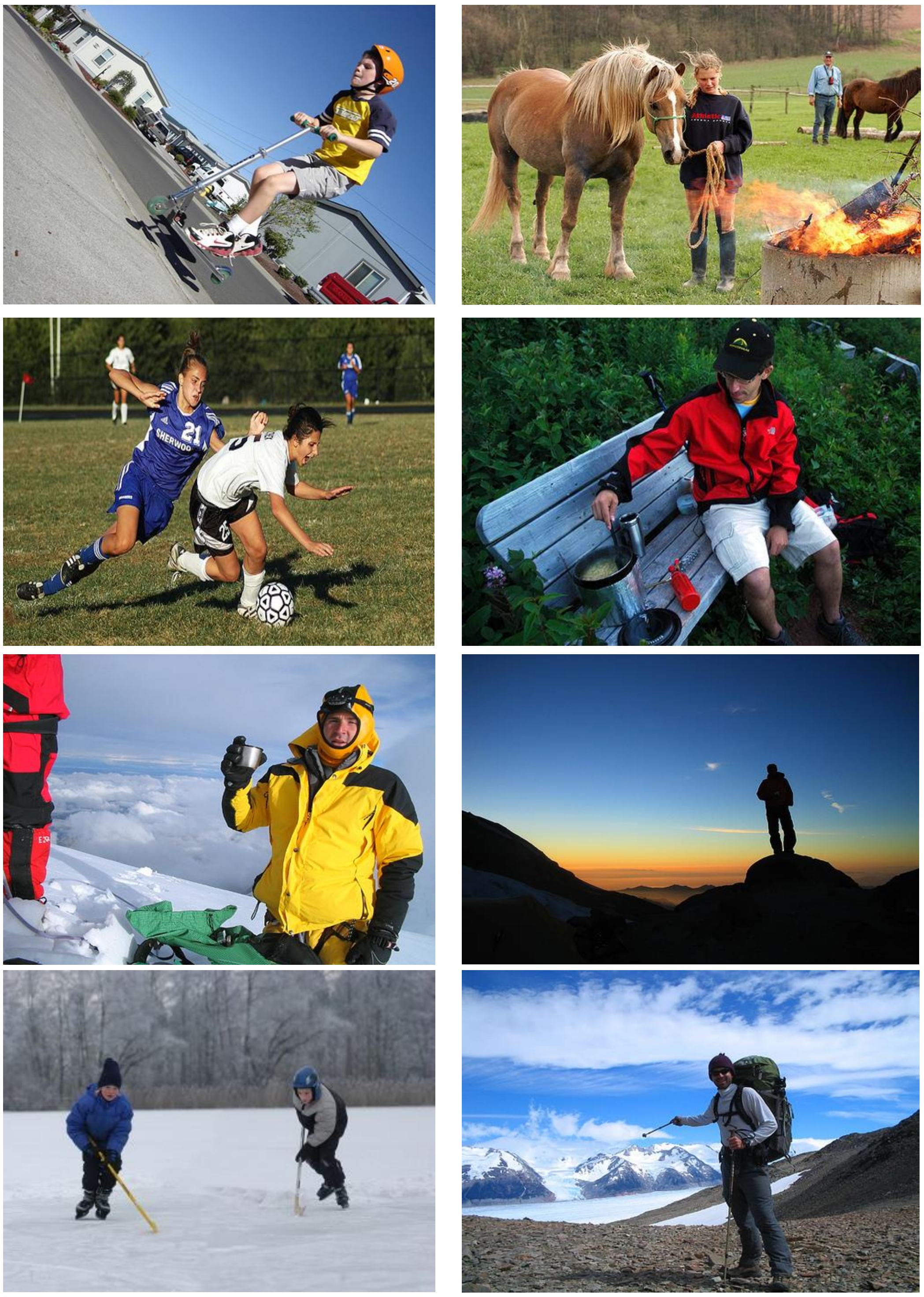


Figure 3.1: Images from FLICKR 8K.

3.2. <u>Software & Hardware Requirements</u>

To keep work flow in fast pace, some already existed machine learning libraries are exploited in the proposed work. These libraries that are availed in this work and their hardware requirements are described in detail as follows.

- <u>NVidia GPU</u>

  A graphics processing unit (GPU) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. Several applications of GPU are in mobiles, laptops, embedded circuits and play stations. Modern GPUs are very efficient at manipulating computer graphics and image processing.

  This work also requires GPU for fast processing and computation. Specifications of this GPU are given in detail as such.

Name	Persistence-M
Version	Quadro K6000
Cores	Sixteen
Memory	257838 MB

Table 3.1: Specs of GPU

- <u>TensorFlow</u>

  To do dataflow programming over a variety of tasks, a library was introduced by Google Brain team for Google's personal use. This library is known as TensorFlow. On November 9, 2015, TensorFlow was released under the Apache 2.0 open source license. This library is a symbolic math library which has spans its domain for applications used in machine learning such as neural networks.

- <u>Keras</u>

  For the purpose of built-in neural networks, a library written in Python called Keras was developed by Google Engineer Francois Chollet. This library is open source. This library has its backend in Theano, TensorFlow as well as CNTK. It contains several implementations of typical neural networks' modules like layers, activation functions, optimization algorithms etc. It also allows training on Graphical Processing Units.

- <u>Anaconda</u>

  Anaconda is an open source and free distribution of the R and Python programming languages for applications of data science and machine learning, predictive analytics and scientific computing that simplifies the package management and deployment by using package management utility *conda*.

- NLTK

  Natural Language Tool Kit (NLTK) library is used in this work. It is built in Python and avails the methods of natural language processing. The main purpose of this library is to pre-process the captions before feeding them into the neural networks.

4. METHODOLOGY

4.1. Introduction

4.1.1. Word Embedding Model

"Apple is tasty, how do we know it's a fruit not a company". Word embeddings are words converted into vectors. The representations of these vectors may be different depending on how they are generated.

Several machine learning algorithms are unable to process strings directly in their raw form. These algorithms need numbers to do any sort of task. To summarize, many real world applications need knowledge extracted from textual data. Some of the examples are sentiment analysis, recommendation based on reviews by Wallmart, Amazon.

Word embeddings are mapping of word to a vector using a dictionary. So it can be anything from one-hot encoding to a vector of real numbers. But in real world applications we need a standard version. This standard version was introduced by Mikolov et. al. [19]. It was named word2vec model.

Word2vec model is a prediction based embedding model which provides the probabilities to the words. It has been proven as state of the art for domains of word analogies and word similarities. Word2vec is a combination of two separate techniques, CBOW and Skip-gram. Both of these models are not deep that is they are shallow networks which perform the mapping of word(s) to a target word.

CBOW predicts the most probable word given the context. The context may be a single word or a group of words. It is a probabilistic model so works better than deterministic methods of embedding generation. It also needs low memory for computation. The disadvantage of CBOW is that it takes average of the contexts of a word. For example, CBOW will place apple in the cluster of fruits and companies.

Skip-Gram model predicts the context given the word. It is flip version of CBOW. It is able to pick out two semantics for a single word. For example it will have two separate vector representations for apple as a fruit and as a company.

4.1.2. Transfer Learning

Science of reusing an already learnt model as a starting point for a different task is known as transfer learning. Pre-trained models are the models that are already trained on some data. Now these models have already initialized weights taken from their previous training. Pre-trained models are used because enough time has already been put in learning the features and tuning the parameters.

Transfer learning is related to several domains including multi-tasking, concept drift etc. It is not exclusively related to the domain of deep learning. However the concept of transfer learning is quite popular in deep learning as well. There are several ways to apply transfer learning on deep learning.

Convolutional Networks can be used a fixed feature extractor. A convolutional network pre-trained on ImageNet is taken. Its last fully connected layer is removed since it is a softmax layer to do the classification and has no real use. Rest is treated as a feature extractor for the new dataset. In this work, this concept is applied because this work requires feature representation of the image. Fine tuning the network doesn't seem viable in the case of image captioning systems.

Another way is to use the pre-trained convolutional network but this time, fine-tune it on the new dataset. To either fine-tune all the layers of convolutional network or fine tuning the last layers keeping the early one fixed to prevent over fitting is also possible. It is observed that earlier layers of convolutional neural networks perform general task of image processing like edge detectors, colour blob detectors while later layers are responsible for solving classification problem of the trained dataset.

4.1.3. <u>Convolutional Neural Networks</u>

In neural networks, a single vector is taken as input which is transformed through a series of hidden layers. Every neural layer is a set of neurons where each neuron is linked to all neurons in a previous layer. Each layer neurons has weights. Each layer has an activation function. Neurons of same layer are independent of each other. Last layer is called output layer.

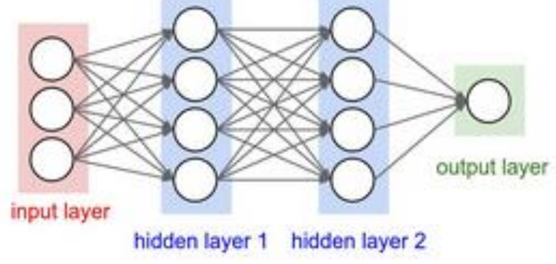


Figure 4.1: Regular Neural Network

Simple neural networks are not beneficial in terms of scaling for images. Consider images of shape 200x200x3 RGB, so if a single neuron of input layer of neural network is considered then it would have total of 200*200*3 i.e. 120,000 weights. Since there are several neurons in a single layer, so there will be more parameters. Therefore regular neural networks are not good choice for image learning tasks.

Convolutional neural network provides the solution. Layers of convolutional network are arranged in 3D. Depth of network refers to third dimension of a layer not to the number of layers. Neurons of convolutional network layer will only be connected to a local region not the whole region. This type of architecture frames the full image into a single feature vector.

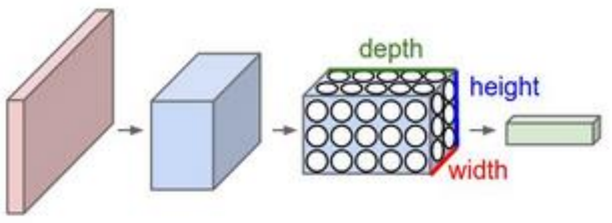


Figure 4.2: Convolutional Neural Network

Basically there are five types of layers in a convolutional network. The description of each layer is given below.

- Input Layer
  This layer hold the input image, a raw pixel image can be of any dimension.
- Conv Layer
  It is the core layer of a convolutional network. In this layer, activation value of neurons which are connected to the local regions of an image is computed. Computation is generally the dot product between the local region and the neuron weights.
- RELU Layer
  This layer is responsible for applying element wise activation function. Choice of activation function depends on the designer of the neural net.
- POOL Layer
  This layer is used to perform down-sampling in a convolutional network, reducing the spatial dimension.
- FC Layer
  Fully connected layer of a convolutional network is just like a fully connected layer of a regular neural network. Each neuron in this layer is connected to every element of previous layer.

The working of a simple convolutional neural network is shown in figure 4.3. First input image is passed through convolutional layer then activations are applied using RELU layer. To down sample the data pooling layer is used. After using some combinations input is finally passed to the fully connected layer which provides us the final classification scores.

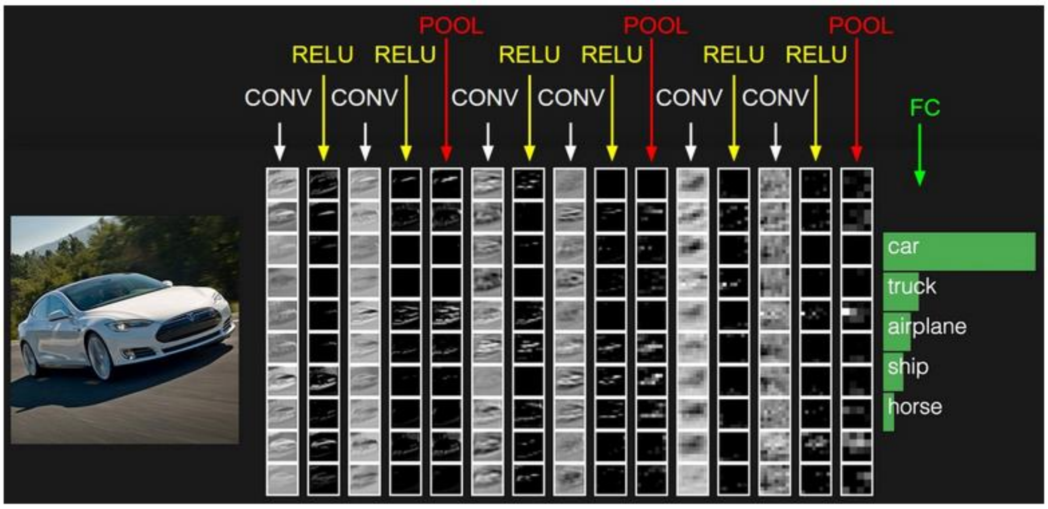


Figure 4.3: Training in CNN

4.1.4. <u>Recurrent Neural Networks</u>

Unlike neural networks, recurrent neural networks not only perceive the input but also that it have seen previously in time. So, these networks have two types of inputs first is the sample itself and second is the previous state. Recurrent networks are known to have memory. This memory is information stated previously in the input which is used by the recurrent networks in order to predict the current output. This information is preserved in recurrent network's hidden state which is increased at each time step.

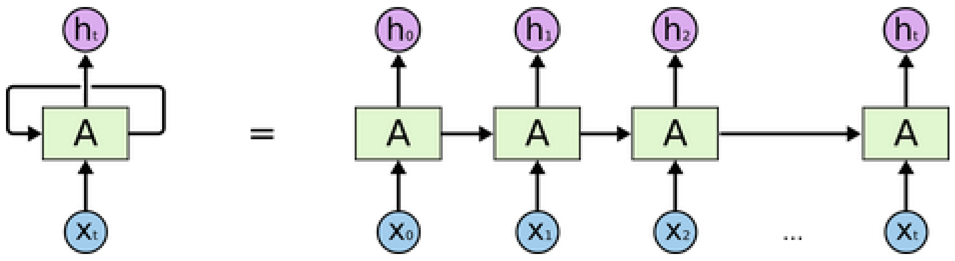


Figure 4.4: Unfolded Recurrent Neural Network

In a recurrent neuron, working flow is as such. First a single time step of the input in this case x_t is fed to the network. Current state h_t of the network is computed using both previous state and the current input. Now current hidden state h_t becomes h_{t-1} for the next time step. The function f is the activation function.

$$h_t = f\,(h_{t-1}, x_t)$$

Assume function f to be the tanh activation. Then W_{hh} are the weights of recurrent neuron and W_{xh} are the weights of input neuron.

$$h_t = \tanh\,(W_{hh}h_{t-1} + W_{xh}x_t)$$

When all time steps are completed, output y_t is computed using last hidden state.

$$y_t = W_{hy}h_t$$

Once the output is generated it is compared to the actual output of the data and then it generates the error. This error is then back-propagated to the network to update the weights of network's neurons. This working is shown in the figure 4.5.

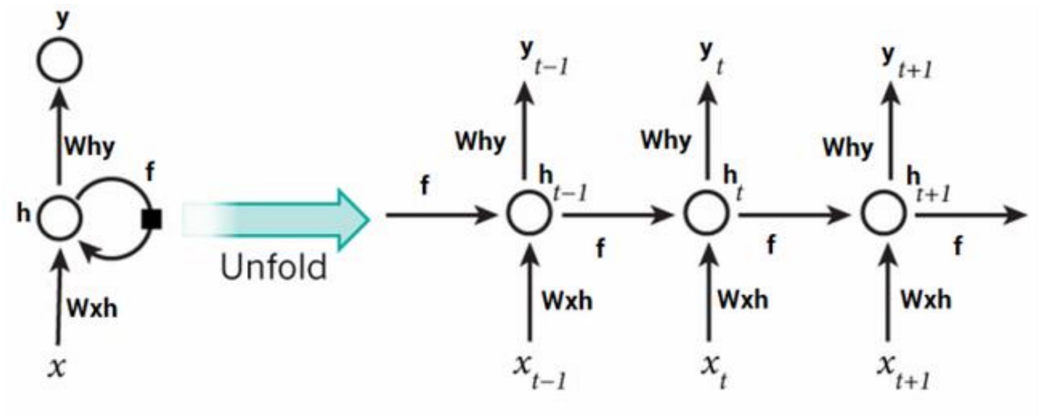


Figure 4.5: Working of Recurrent Neural Network

4.1.5. Long Short Term Memory Networks

"I grew up in India… I speak fluent *Hindi*." In this sentence, it is clear that the language will be Hindi but to know the answer context of India is needed. It is possible there is a large gap between the given sentence and the context of India which is used to find the answer. As this gap grows, recurrent neural networks become unable to learn to extract the information.

In theory, recurrent neural networks are perfectly able to handle long term dependencies. In practice, it was discovered by Hochreiter (1991) and Bengio et. al. (1994) that RNN are not able to learn these dependencies. Fortunately, LSTMs however are able to learn these dependencies successfully.

Long Short Term Memory Networks work well on a great number of problems. These are specifically designed to handle long term dependencies. Their default behaviour is to remember information for long intervals of time. LSTM like recurrent networks also follows chain like structure. But unlike recurrent networks having single neural network layer LSTMs have four linked to each other in a special way.

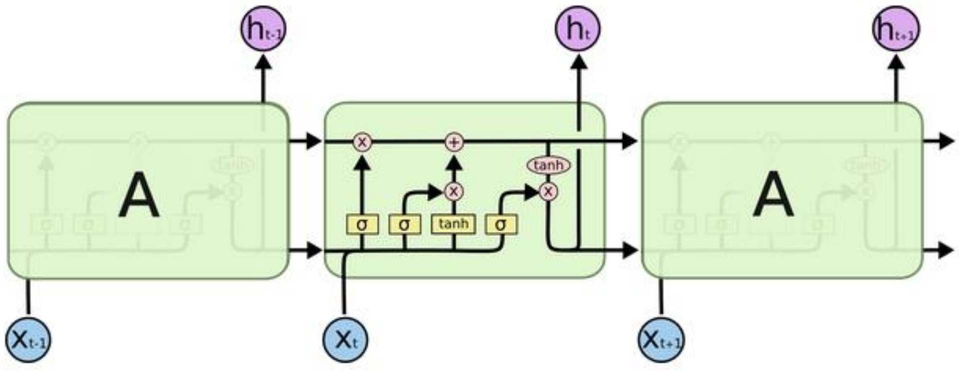


Figure 4.6: Basic Structure of LSTM Network

The key player in LSTMs is cell state, the horizontal line at the top of the figure. It is like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. Information just flow along it unchanged. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

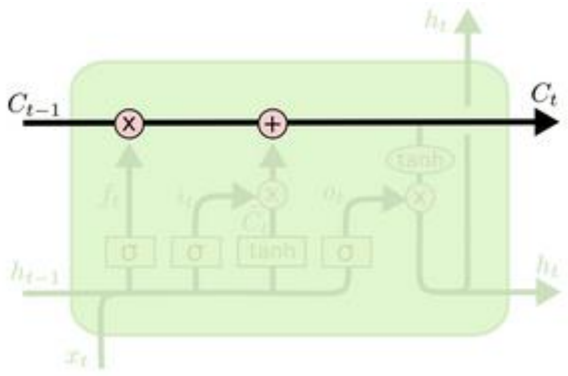
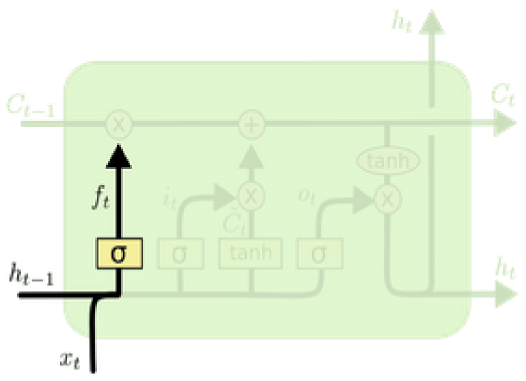


Figure 4.7: Cell State in LSTM

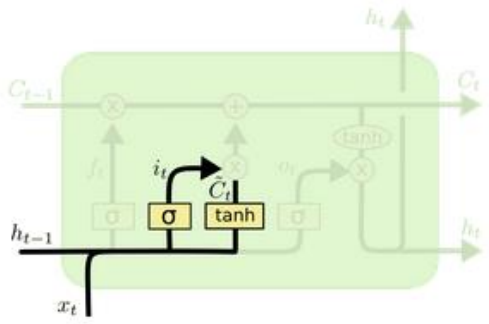
Step one of LSTM is deciding what to throw away from the cell state. Forget gate layer make this decision. By looking at values of h_{t-1} and x_t, it outputs a number b/w 0 and 1 for each value in cell state C_{t-1}. 1 mean “keep this completely” and 0 means “get rid of this completely”.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Figure 4.8: Forget Gate Layer in LSTM

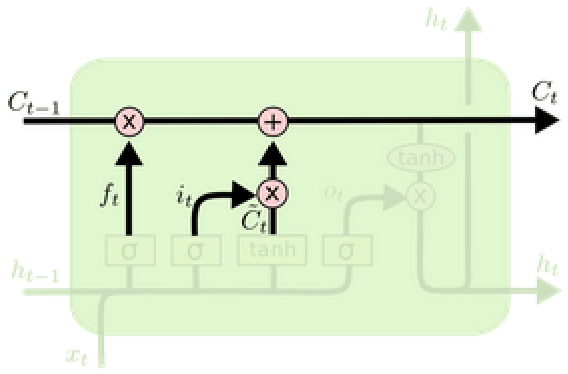
Step two of LSTM is deciding what to store in cell state. It consists of two parts. First is input gate layer is responsible for updating values. Then a *tanh* layer generates a new vector of values *C~t* that could be added to the state.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Figure 4.9: Input Gate and Tanh layer in LSTM

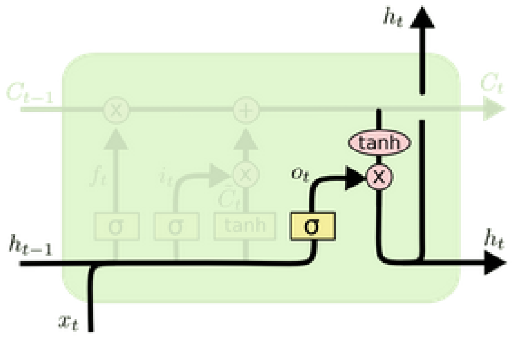
Step three is to update the old cell state, C_{t-1} into state C_t. Old state is multiplied by f_t then it i_t*$C_{\sim t}$ is added. In the case of the language model, this is where we'd actually drop the information about the old subject's gender and add the new information, as we decided in the previous steps.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 4.10: Updating Cell State in LSTM

At last, output based on cell state is calculated. First a sigmoid layer that decides what part of cell should contribute to output then tanh to constraint the values between -1 and 1 and then finally multiplication by the output of sigmoid gate.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 4.11: Output Gate Layer in LSTM

4.1.6. BLEU-N Evaluation Metric

Bilingual Evaluation Understudy or BLEU is a score which is able to compare target translation of text to one or more given reference translations. In image captioning systems, this evaluation metric is used to compare the target caption with given other captions.

If sentences are matched perfectly then the BLEU score is of 1.0 where a complete mismatch results in 0.0. However it is not a perfect evaluation metric but has some advantages. It is fast, almost inexpensive in computation, independent of the source language, correlates highly with evaluation of humans and is widely adopted.

The main task of BLEU score is to perform the comparison of n-grams of target translation with the n-grams of reference translations and count the number of matches. These matches are independent of the positions. More matches results in better score. The number of n-grams is generally taken up to 4. Therefore in this work, four evaluation metrics are considered, BLEU-1, BLEU-2, BLEU-3 and BLEU-4.

The Natural Language Toolkit Library (NLTK) of Python provides the implementation of BLEU scores. Besides machine translation, other application of BLEU score would be text summarization, speech recognition, image captioning systems, language generation etc.

4.2. Flow Chart

A picture is better than a thousand words. The flow chart that represents the methodology proposed in this work is presented in this section. Further sections provide the detailed explanation of each component shown in this flow chart.

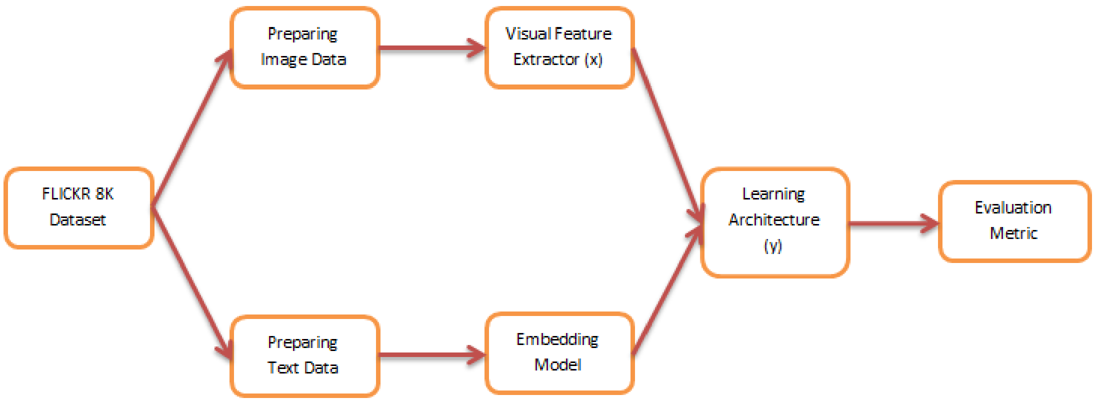


Figure 4.12: Flow Chart of Proposed Methodology

4.3. Preparing Image Data

FLICKR 8K is a collection of a total of 8000 images. These images are RGB images but possess variations in width and height. Since every visual feature extractor needs some kind of pre-processing of image therefore before feeding these images into the desired visual feature extractor or convolutional neural networks, these images are pre-processed. Technique used to pre-process the image is same in every type of convolutional neural networks.

First the source image is converted into the shape of the target image required by the network. For example, if VGG-16 is the convolutional neural network used then target image shape would be [224, 224] keeping the RGB channel.

Second is to feature-wise zero centre every sample with specified mean. In case of VGG-16, the specified mean set for R-G-B is [123.68, 116.779, 103.939] respectively. This specified set of means comes from the training data of Image Net competition, where mean was calculated as by summing up the intensities of the training data and dividing by the total number of pixels for each channel.

Henceforth, a pre-processed image becomes available for layers ahead.

4.4. Visual Feature Extractor

Now to extract the visual information from the images, convolutional neural networks are exploited. Basically two types of convolutional networks are used in the experiments. These networks are discussed below.

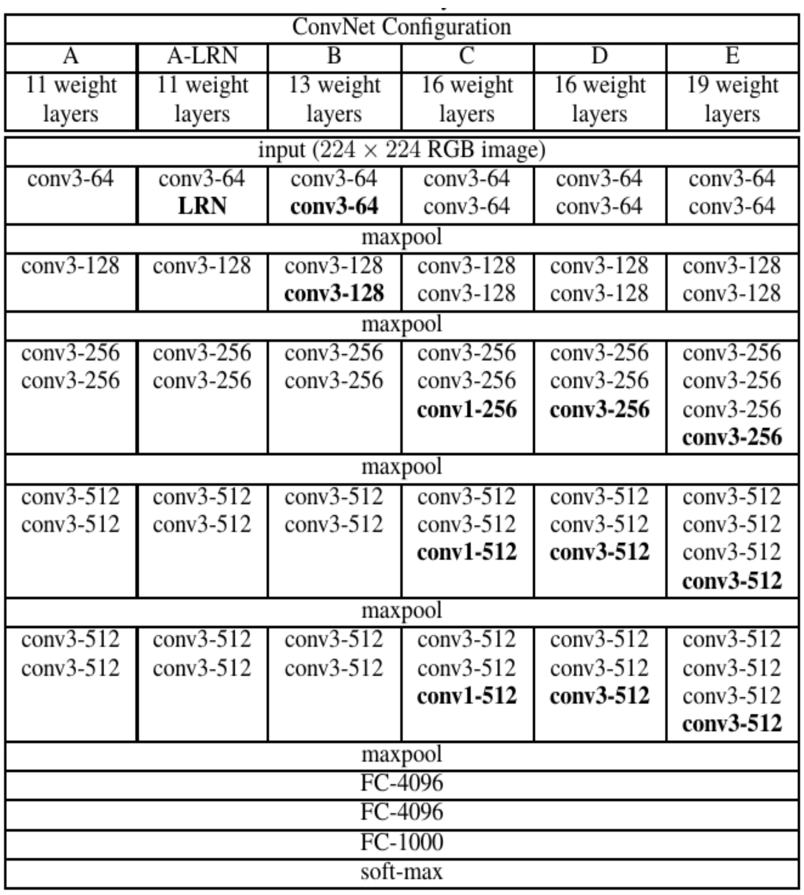
4.4.1. VGG

- Introduction

  VGG-Architectures were built by Visual Geometry Group [16]. It is a convolutional network proposed by K.Simonyan and A. Zisserman in their paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. Given an image as input, the model was built to detect the object in that image. The input image is of shape [224*224*3]. It is R-G-B image. The model size is 528MB. The network attains 70.5% top-1 accuracy and 92.7% top-5 test accuracy in ImageNet competition. ImageNet is a vast dataset that is a collection of over 14 million images mapped to 1000 classes.

- Architecture

  VGG-Architectures have several variations. These invariants are described in the below table. The column D indicates the VGG-16 convolutional neural network whereas column E indicates the invariant of VGG-16 known as VGG-19 network.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 4.13: VGG Architectures

In the shown figure 4.13 , "conv3-512" is abbreviation of *"512 Convolutional Filters of Size 3x3"*, "FC-4096" is abbreviation of *"Fully Connected Feed Forward Layer of 4096 Neurons"*, "maxpool" indicates a "Max Pooling Layer" and "soft-max" points to the "Soft-Max Layer of A Convolutional Neural Network".

The concise model of VGG-16 convolutional neural network can be seen in the given figure 4.14. A thing to notice is that there is a pre-processing layer which taken the RGB image as input in the range [0,255] and then perform feature-wise zero centring.

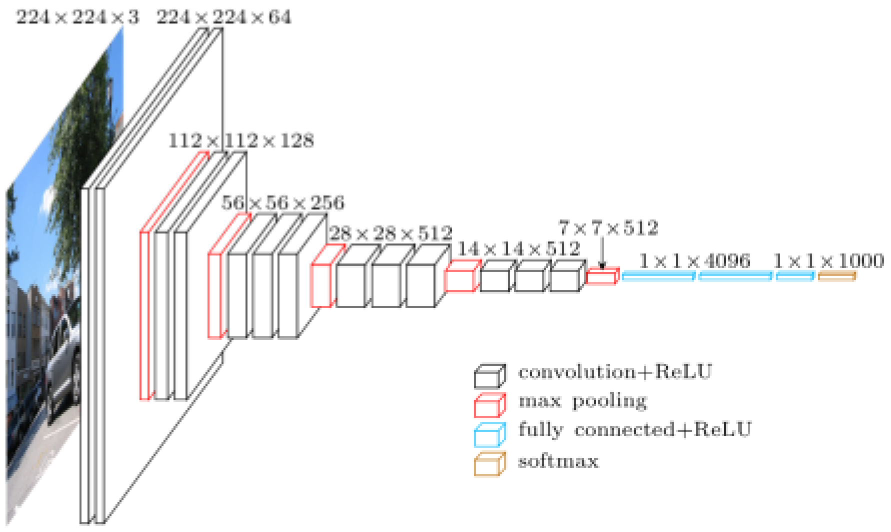


Figure 4.14: Visual Representation of VGG-16 Layer Network

- Application
  In this work, VGG-16 is used as one of the visual feature extractor. The pre-processed input image as mention in section 4.3 is forwarded to the network. Now the last layer of VGG-16 is a softmax layer so this layer was removed from the network resulting into fully connected layer 4096 neurons as the last layer. Thus after passing image into the VGG-16 network, we get a fixed-size feature vector of 4096 activations as shown in figure 4.15.

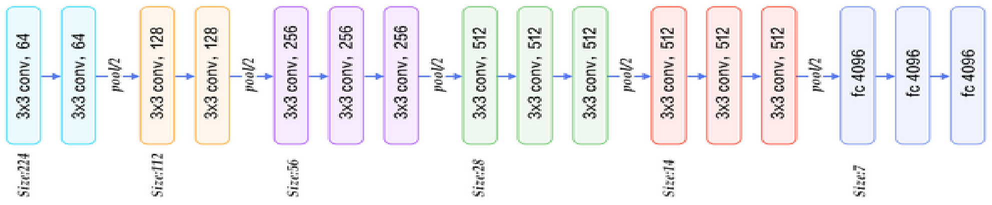


Figure 4.15: VGG-16 with Soft-max Removed

4.4.2. <u>InceptionV3</u>

- <u>Introduction</u>

  The InceptionV3 architecture was first proposed by Christian et. al. 2015 in their paper "Rethinking the Inception Architecture for Computer Vision". InceptionV3 architecture is benchmarked on ILSVRC 2012 classification challenge. The top-5 error rate of 3.6% and top-1 error rate of 17.3% on testing set were reported from this architecture.

- <u>Architecture</u>

  Figure 4.16 shows the architecture of a single inception module. Using this inception module we can get several combinations of convolutions. There are 1x1, 3x3 and 5x5 convolutions along with a 3x3 max pooling. To reduce the expense of computation larger convolutions were not used, so 1x1 convolutions were suggested by the reference paper to reduce dimensionality of feature maps.

  The performance of the network is improved when multiple features from multiple filters are exploited. There exists another fact that reasons for inception architecture being better than others. Convolutional Architecture before inceptionv3, extract the convolutions on the spatial and channel wise domain simultaneously. By allowing 1x1 convolutions, the inception module is exploiting cross-channel correlations, ignoring the spatial dimensions which are then followed by 3x3 and 5x5 convolutional filters to extract the cross-spatial and cross-channel correlations.

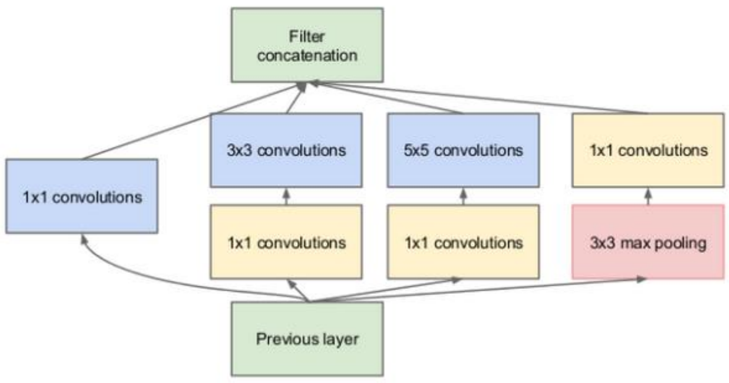


Figure 4.16: Inception Module

- <u>Application</u>

  In this work, one of the visual features' extractor is InceptionV3. It requires a target image of shape 299*299*3. Therefore pre-processed image of shape

299*299*3 was put into the InceptionV3 network. The resulting feature vector obtained after passing the image through InceptionV3 network and performing 2D global average pooling was a fixed-size vector of 2048 activations, as shown in figure 4.17.

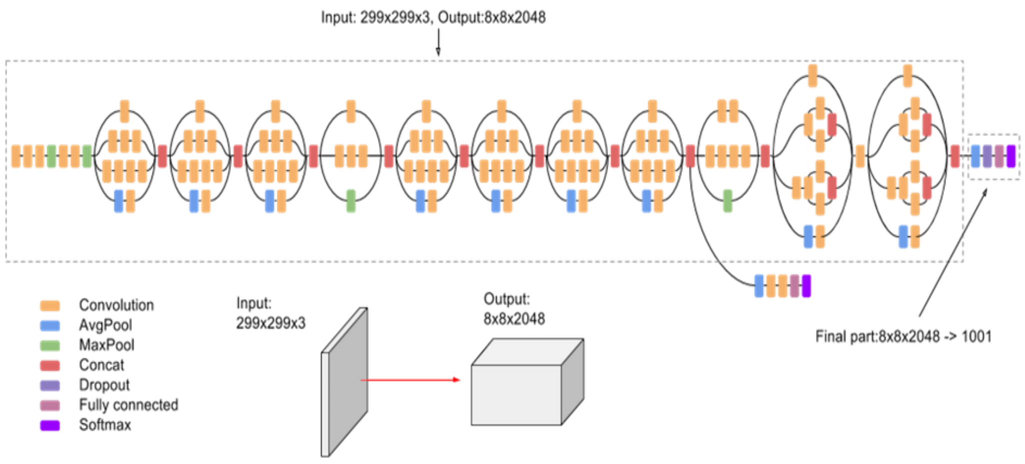


Figure 4.17: InceptionV3 Architecture

4.5. Preparing Text Data

Textual data is pre-processed before it is fed into the embedding model. Steps taken for pre-processing are described as such.

- Conversion of words to lower case
- Removal of punctuation
- Removal of words with frequency less than some threshold
- Removal of words with numbers like telephone number or street car plate number
- Stop-words are not removed since it have an important role in generating a fluent caption

After pre-processing the text it needs to be passed to long short term memory networks via embedding model. Now before it is fed into the LSTM, a starting tag and an ending tag is necessary to denote the start and end of the caption, because the first input word provided will be this starting tag.

For example,

- Original Caption: A man is riding on a “Caucasian horse”
- Pre-Processed Caption: a man is riding on a horse
- After Tagging: [startseq, a, man, is, riding, on, a, horse, endseq]

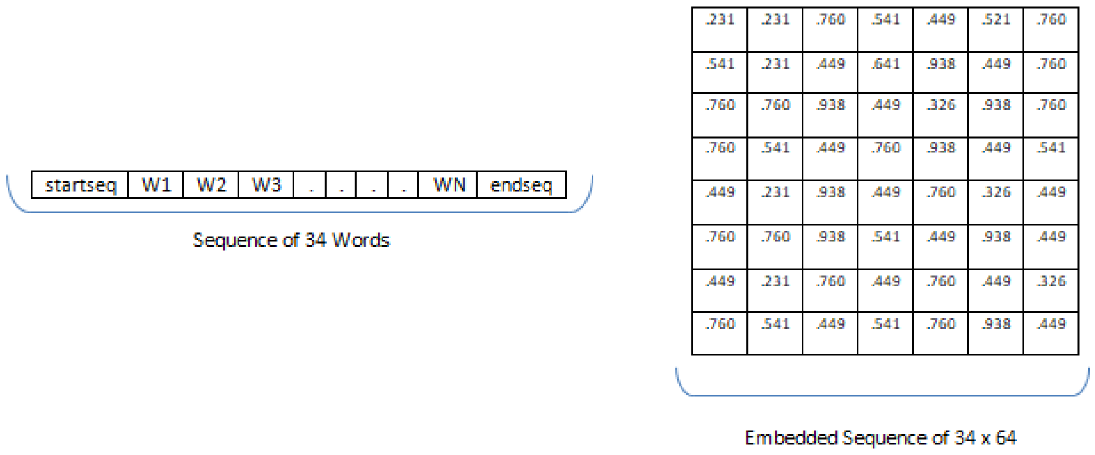
Now after all this, we will have startseq as the first input word to the LSTM then at each time step one word will be added to our input. But LSTM networks need a fixed size input vector. Therefore, input is padded to the maximum length of the caption for each sample. The maximum length of the caption is sum of Two [endseq + startseq] and the length of the caption containing maximum words. If every time step of the LSTM is considered then the input sequence and the output word will be as shown in figure 4.18.

INPUT	OUTPUT
startseq [pad maxlen-1 times]	a
startseq a [pad maxlen-2 times]	man
startseq a man [pad maxlen-3 times]	is
startseq a man is [pad maxlen-4 times]	riding
startseq a man is riding [pad maxlen-5 times]	on
startseq a man is riding on [pad maxlen-6 times]	a
startseq a man is riding on a [pad maxlen-7 times]	horse
startseq a man is riding on a horse [pad maxlen-8 times]	endseq

Figure 4.18: Example of LSTM Input & Output

4.6. <u>Embedding Model</u>

Word Embedding is a dense vector that represents a word. Inferences between words can be obtained using word embedding. In proposed work, embedding model is trained on the whole training data. Words are embedded with word vector length among [64, 128 and 256]. Hence, if an input sequence is of length 34 which is actual maximum length of a caption (2 inclusive) is passed through embedding model with embedding length of 64 than output generated will be a matrix of 34*64.



.231	.231	.760	.541	.449	.521	.760
.541	.231	.449	.641	.938	.449	.760
.760	.760	.938	.449	.326	.938	.760
.760	.541	.449	.760	.938	.449	.541
.449	.231	.938	.449	.760	.326	.449
.760	.760	.938	.541	.449	.938	.449
.449	.231	.760	.449	.760	.449	.326
.760	.541	.449	.541	.760	.938	.449

Figure 4.19: Embedded Word Vector Example

4.7. **Learning Architectures**

4.7.1. Encoder Decoder Architecture

In this work, image captioning systems follow standardized version of architecture also known as encoder-decoder architecture. As the name suggests, it has two components. Encoder, a network module which reads the input and encode it into some fixed length vector type representation. Decoder is a network module which reads the encoded input and generates the desired output.

4.7.2. Implant Architecture

In implant architecture, the visual information vector which is extracted from the activation values of neural layer of a visual feature extractor namely convolutional neural network discussed in Section 4.3 implanted into the recurrent neural network simultaneously with the text input. In this architecture, image is treated as a part of the prefix of the caption.

Implant architecture trains recurrent neural network to condition the generation of caption on both linguistic and visual features. In short, recurrent neural network is primarily held responsible for language generation conditioned on image. The basic architecture of implant view is shown in figure 4.20.

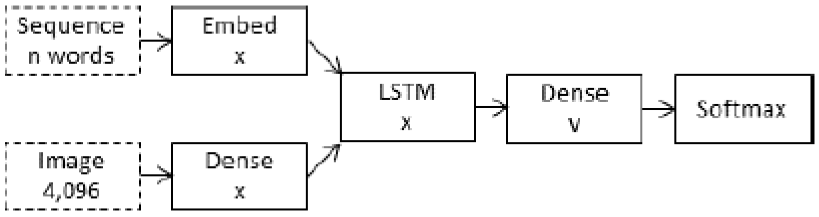


Figure 4.20: Implant Architecture

4.7.3. Merge Architecture

In merge architecture, recurrent neural network only have embedded text as input, visual features are kept away from the recurrent neural network and are merged at a later stage in the neural network. This way, caption prefix is of only text with no image.

Merge architecture allows recurrent neural network to only encode linguistic information without caring for the perceptual features which themselves are introduced at a prediction stage ahead. Only at this prediction stage, visual information conditions the generation of captions.

Several ways to combine two encoded inputs like addition, concatenation, multiplication etc. exist. The basic architecture of merge view is shown in figure 4.21.

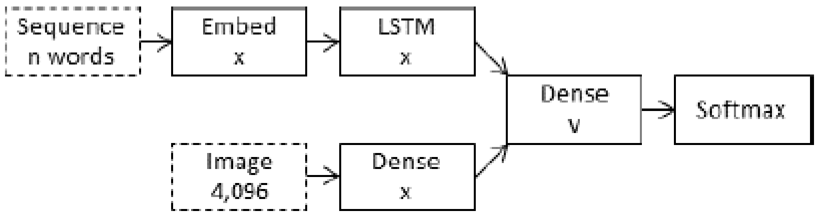


Figure 4.21: Merge Architecture

4.7.4. Ensemble Architecture

The work also proposes a model where two visual feature extractors are used to encode the image. This is done to see whether the use of two visual feature extractors could have any effect on the performance of model. The feature vectors extracted from both convolutional neural networks are merged after passing through a dense layer separately. Here also both views implant and merge are implemented to check the consistency of the behaviour of recurrent neural network.

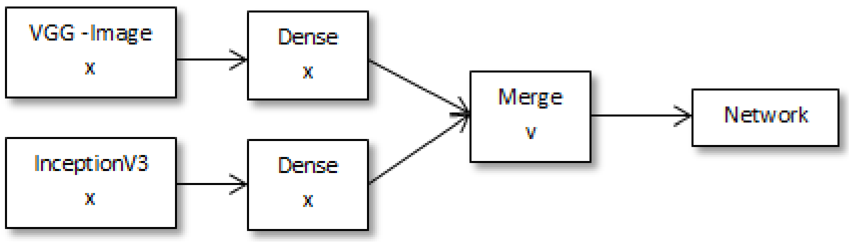


Figure 4.22: Ensemble Architecture

4.8. Features' Visualization & Attention Model

Learning model for an image captioning system in general is shown in figure 4.24. In this model, an image is encoded by a visual deep convolutional neural network which then gives us an encoded representation h. This representation h then is passed to the long short term memory network to generate the desired caption.

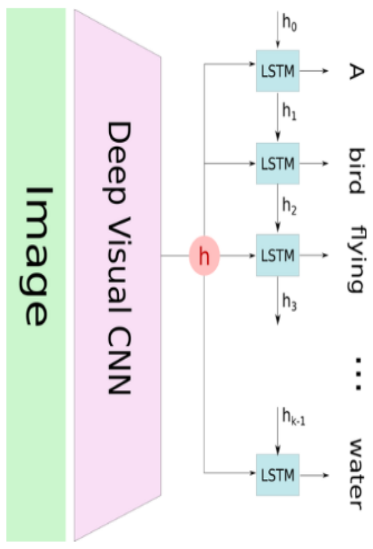


Figure 4.23: Image Captioning without Attention

The main problem with this architecture is that it tries to learn the caption generation based on the whole representation h of the image. Generally when model is trying to predict the next word of the caption only a part of the image is concerned. Therefore this model cannot produce words for different attributes of the image efficiently. Hence the attention mechanism comes into action. To apply the attention, image is divided into N parts and by using convolutional neural network representations [y_1, y_2, y_3 … y_N] are learnt. When LSTM tries to generate a new word, with the help of attention it is able to focus only on the relevant portion of the image, thus image caption generation is efficient.

As shown in figure 4.25, if i words have been predicted, the hidden state of LSTM is h_i. Relevant part of the image is selected using h_i as the context. The output z_i of the attention model is the representation of the filtered image where relevant scenes of the image are concerned and is used as an input to LSTM to predict the new word

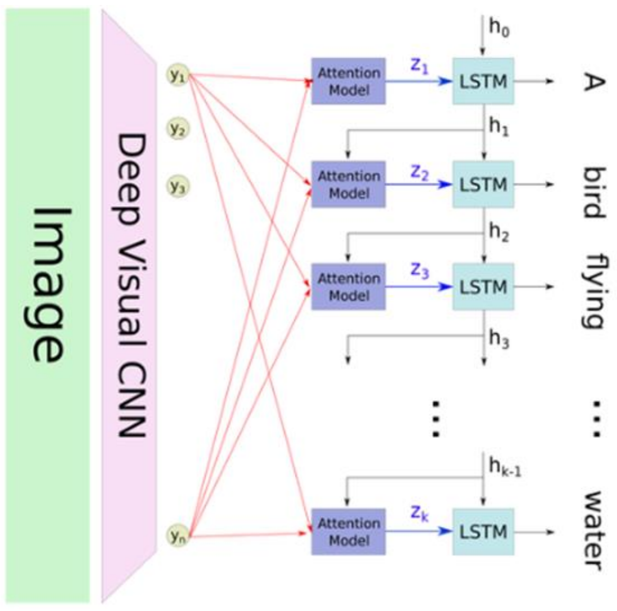
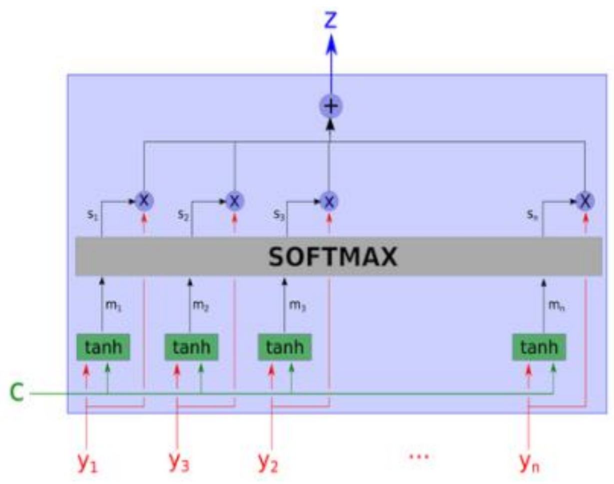


Figure 4.24: Image Captioning with Attention

and to return a new hidden state h_{i+1}. Attention module is shown in figure 4.26. Each representation of image [y_1, y_2, y_3 … y_N] is combined with the context vector c and then passed through tanh layer. Each m_i is computed independently without looking at the other y_j.



$m_i = tanh(w_{cm}.c + w_{ym}.y_i)$

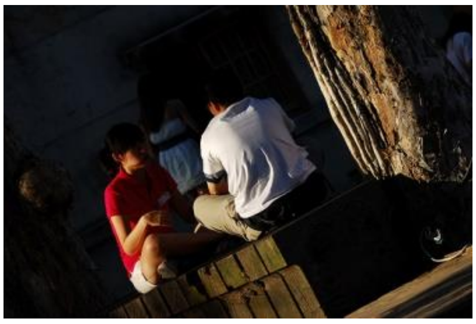
Figure 4.25: Attention Module

After that the computed value m_i are passed through softmax layer along with the image annotations. Here, the s_i is the softmax of the m_i projected on a learned direction. So the softmax can be considered as max of the « relevance » of the input variables, according to the context. Output z is weighted arithmetic mean of all y_i where the weights represent the relevance of each variable according to the context c.

$$z = \sum_i s_i y_i$$

To see what type of features attention model uses, we have given a try to visualize the features representations of the image. These feature maps are extracted from the VGG-16 convolutional neural network in which layers ahead of conv3 512 x 14 x 14 are removed. This visualization is shown in figure.

In figure 4.27, some of feature maps are taken into consideration since there were a total of 512 feature maps of 14x14. As it can be seen in the feature maps, every feature map corresponds to some portion of the image. This property of the features produced from the convolutional network is exploited by the attention mechanism to generate the caption.



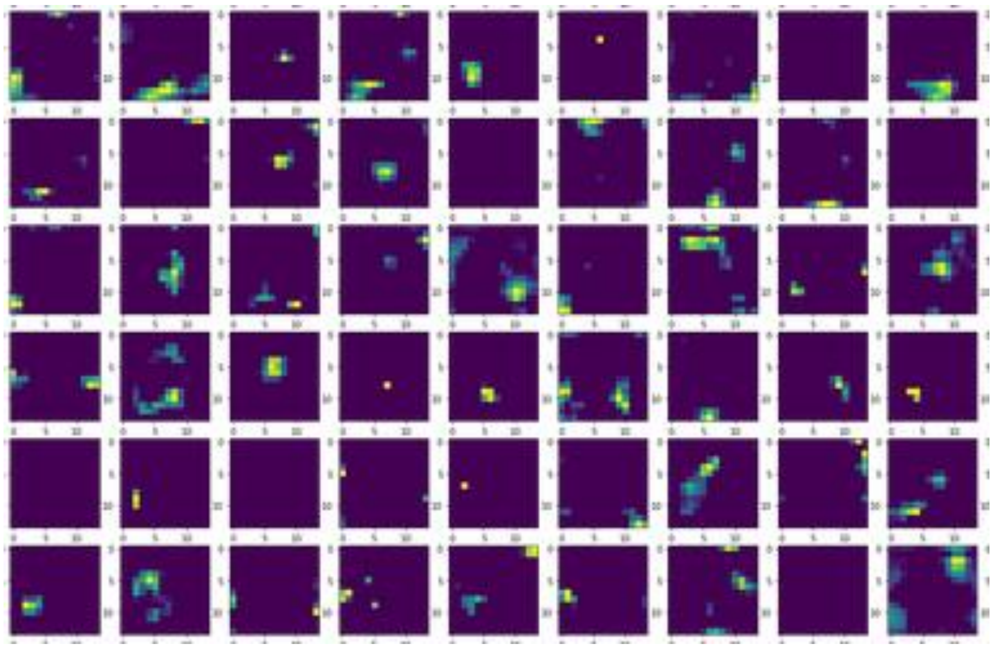


Figure 4.26: Feature Map Visualization

5. Experiments

In the proposed work, several experiments are performed to gain insight of captioning concepts. Sections ahead describe these experiments.

5.1. Implant versus Merge Architecture

Architecture of both views that is used to perform the experiments is very basic. There is no hyper-parameter tuning involved and the techniques such as regularization are not considered. This is done to avoid the bias from both architectures so that one finely tune architecture cannot overwhelm another. For testing purposes, these architectures are run 3 times and then mean of the score is noted.

Implant architecture is shown in figure 5.1. First two boxes at the top are input layers. One is sequence that we get from text pre-processing and another visual representation of image extracted from convolutional networks. The sequence goes to embedding model to dense layer and then a dropout is applied. Visual feature vector goes to dense layer to dropout. Then both the encodings of language and image are combined and implanted into the LSTM. After that a dense and then a softmax layer is introduced. Dropout rate is fifty percentage fixed.

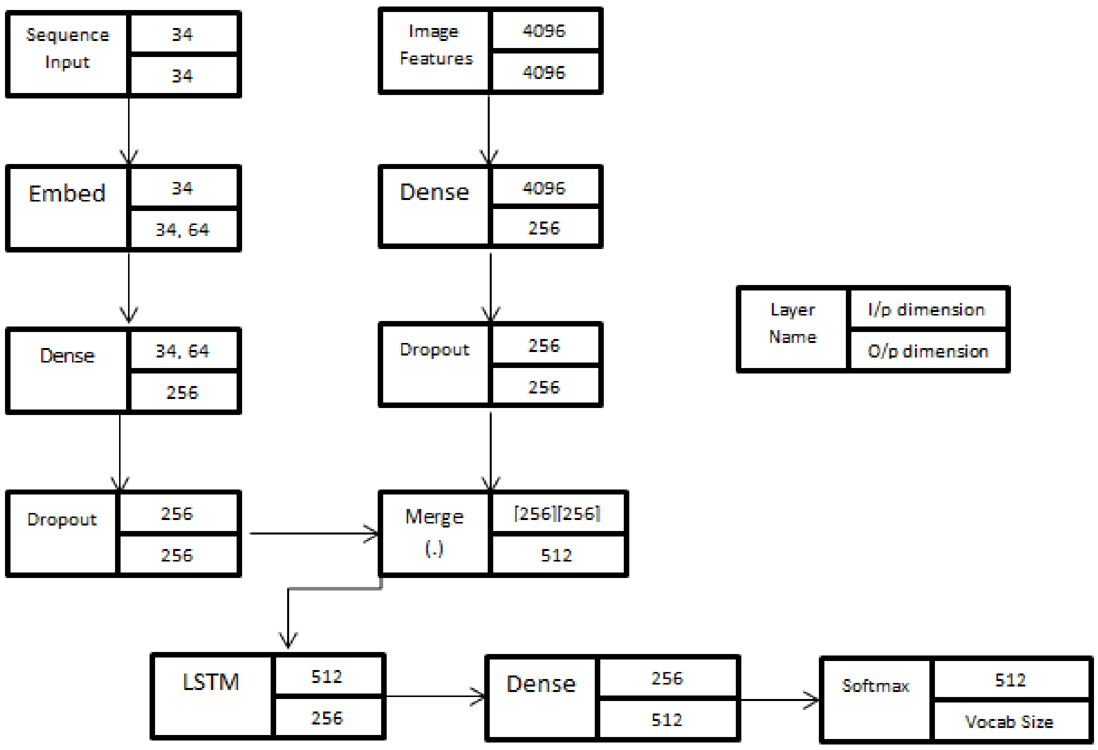


Figure 5.1: Implant Architecture

Neural Net

Merge architecture is shown in figure 5.2. Every box represents a neural layer. First two boxes at the top are input layers. One is sequence that we get from text pre-processing and another visual representation of image extracted from convolutional networks. The sequence goes to embed to dropout to LSTM. Image features goes to dense to dropout. After that both extracted features are merged together on concatenation. Then concatenated vector goes to dense layer and then to softmax layer. Softmax layer is a fully connected layer with softmax activation. Total number of neurons in softmax layer is equal to size of trained vocabulary. This layer will predict the most probable word.

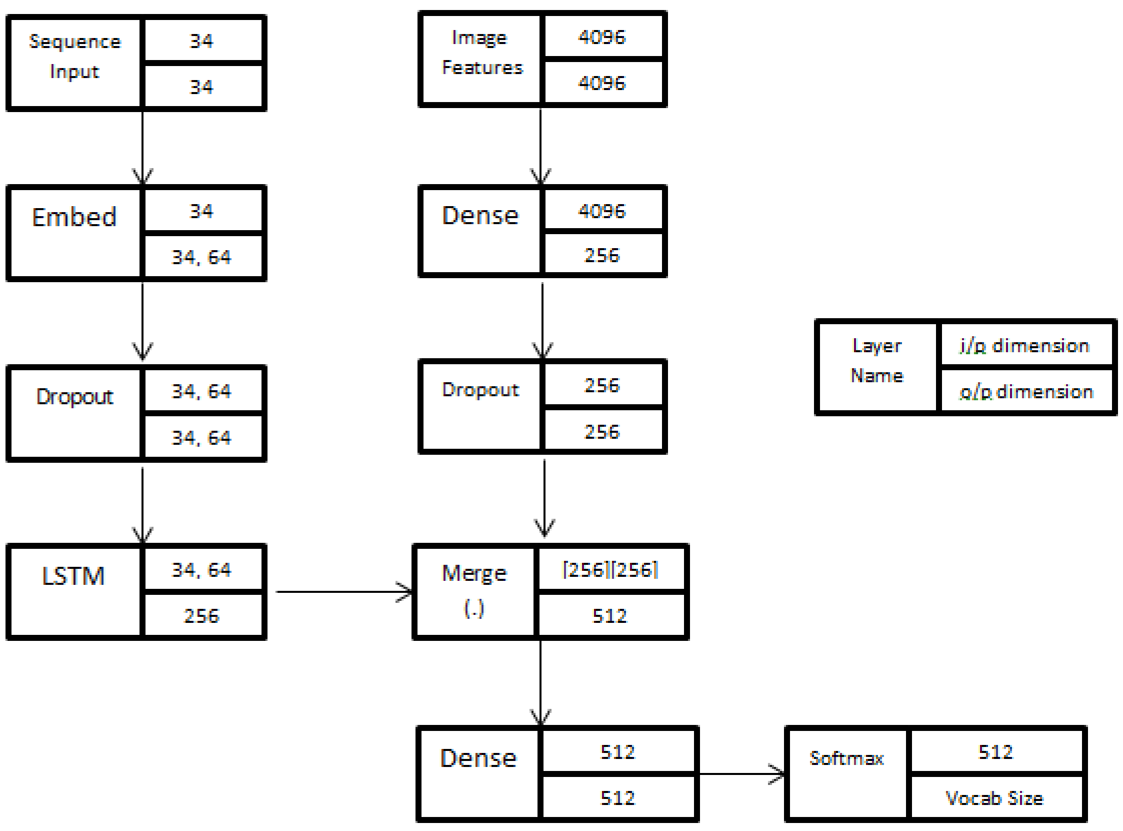


Figure 5.2: Merge Architecture Neural Net

On both architectures, data is train and tested. Parameters on which these experiments are performed are embedded word vector length and the size of LSTM state. The evaluation metrics score and their analysis is performed in "*Results and Analysis"* section.

5.2. <u>One vs. Three vs. Five Captions</u>

To understand the working of Long Short Term Memory Networks, an experiment is also performed where constraints are put on number of captions. By using same implant and merge architectures mentioned above, this experiment is performed. In this experiment, number of captions that are provided to the network is in the range of one, three and five.

5.3. Ensemble Models

This is the last experiment performed in this work. The ensemble model based on use of two visual feature extractors or convolutional neural networks is exploited. Two convolutional neural networks are VGG-16 and InceptionV3. The architecture of this model is shown in figure 5.3 and figure 5.4. It can be seen that feature vectors extracted from both models are first passed through dense layer separately and then they are merged using concatenation both. Rest architecture is the same as it is for previous implant and merge architectures.

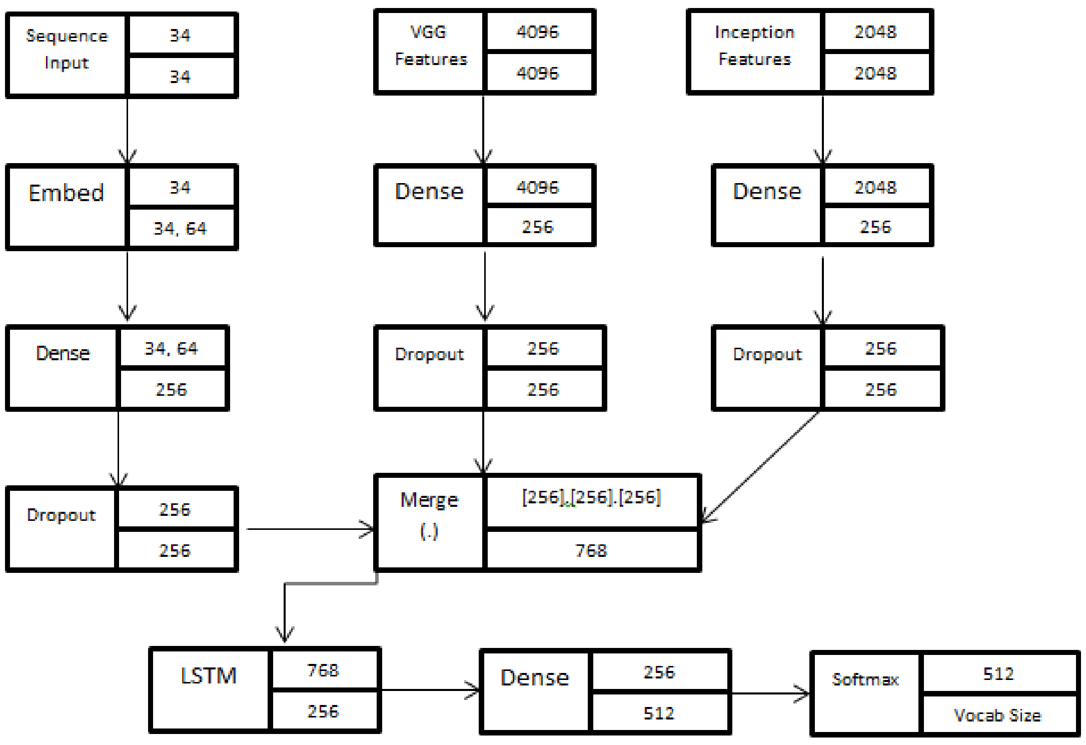


Figure 5.3: Ensemble Implant Architecture Neural Net

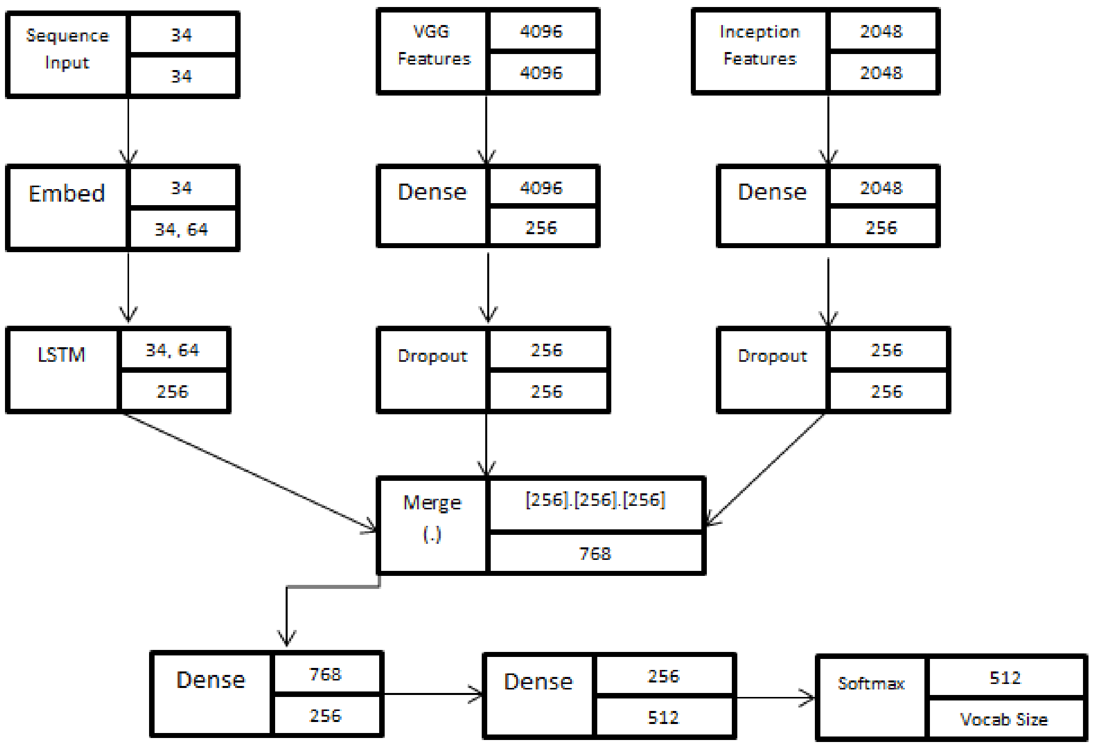


Figure 5.4: Ensemble Merge Architecture Neural Net

6. Results & Analysis

- **Implant vs. Merge**

  Both implant and merge architecture are tested. In order to keep the flow of experiments in fast pace, models are tested on one caption. Although architectures were evaluated three times each to keep the experimentation consistent and then the average of scores was taken. Parameter taken into consideration is the word vector embedding length. To maintain the consistency state size of 256 is fixed in these architectures. These scores are given in table 6.1. Abbreviations used are "I-E64" = Implant Architecture with Embedding Word Vector Length 64 and "M-E64" = Merge Architecture with Embedding Word Vector Length 64.

BLEU Scores on One Caption				
	BLEU-1	BLEU-2	BLEU-3	BLEU-4
I-E64	0.2568	0.1112	0.0763	0.0348
I-E128	**0.3379**	**0.1444**	**0.1002**	**0.0396**
I-E256	0.3060	0.1409	0.1033	0.0444
M-E64	**0.3559**	**0.1674**	**0.1197**	**0.0513**
M-E128	0.3510	0.1621	0.1148	0.0482
M-E256	0.3059	0.1363	0.0959	0.0384

Table 6.1: BLEU Scores noted on Single Caption

From the given architectures and table 6.1 various observations can be made which are pointed out as such.

1. <u>Consistent Advantage</u>

   Above table 6.1 suggests that late merging of visual features with the textual information is more beneficial. Although the deviation in results of implant and merge is really narrow yet it performs consistently during three separate runs and has an advantage over implant. Merge architecture performs better than implant architecture and can generate captions of good quality even with smaller layers.

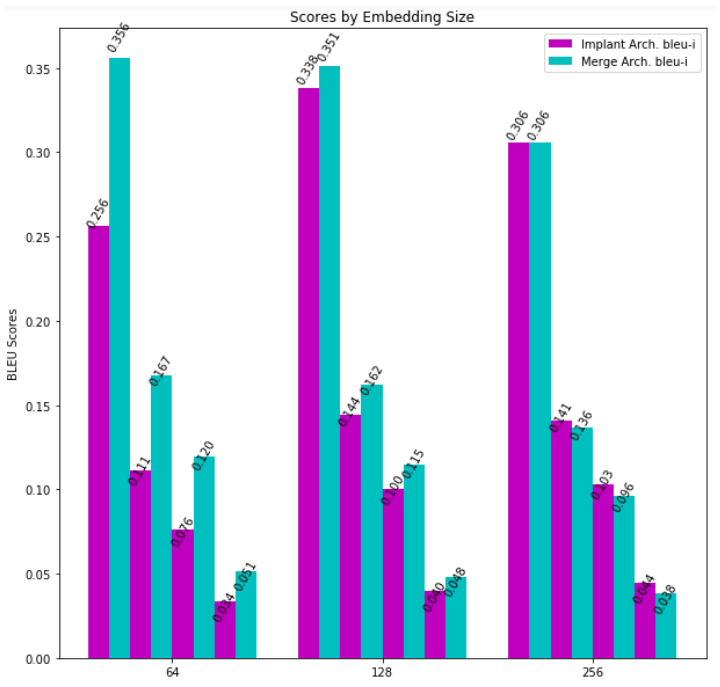


Figure 6.1: Graph representing BLEU scores.

2. <u>Parameters Handled</u>
   There is one more observation about the parameter handling of both architectures. On one hand implant architecture uses recurrent network to train on both visual and linguistic encodings at the same time thus there is a great increase in the vocabulary handled by recurrent network whereas merge architecture uses only textual encodings for recurrent network providing it smaller vocabulary on expense of more parameters at the point of merging of image and text in the network.

3. <u>Training Rate</u>
   As shown in figure 6.2, the training rate of implant architecture is rather slow than merge architecture. Merge architecture learns much faster and better than the implant architectures.

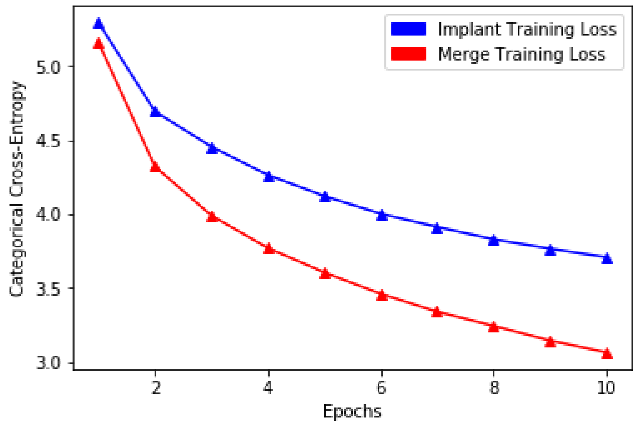


Figure 6.2: Training Rate

4. <u>Performance over Embedding Word Vector Length</u>
   Merge architecture tends to decrease in performance as embedding size grows. In this case embedding size of 64 is the best case scenario for merge whereas embedding size of 256 is the worst case scenario. On other side implant architecture shows inconsistent performance on variations in the length of embedded word vector.

5. <u>Training Vocabulary Used</u>
   On test captions, the amount of English words used for generation process is very low. Overall maximum vocabulary used is only 16% of the training data. It inferred that neural model finds it difficult to use infrequent words. It clearly suggests reducing the size of vocabulary will have minimal loss in performance.

6. <u>Overall Inference</u>
   The final observation is that late merging of image features with linguistic information gives better results than the one shown in implant architecture. Therefore, it concludes that if recurrent network should better be viewed as an encoder to encode the textual data rather than a generator in image captioning systems because if that was the case it would need an image in order to know what to extract from the image and generate as caption but that does not prove to be beneficial.

- **<u>One versus Three versus Five Captions</u>**
  As shown in experiments chapter, an effort to understand the working of LSTM is made. Here results are noted with variations in number of captions. These results are noted only for "M-E64" architecture since it performs better than the others and thus can tell us more about the way LSTM works. Table 6.2, represents the BLEU scores in the experiment performed. Abbreviation "M-E64-1" means merge architecture with word embedding length of 64 trained on one caption.

BLEU Scores				
	BLEU-1	BLEU-2	BLEU-3	BLEU-4
M-E64-1	0.3559	0.1674	0.1197	0.0513
M-E64-3	0.4038	0.2217	0.1539	0.0673
M-E64-5	**0.5878**	**0.3288**	**0.2401**	**0.1134**

Table 6.2: BLEU Scores noted on 1, 3 & 5 Captions

Table 6.2 clearly dictates that Long Short Term Memory Networks works better in terms of more number of captions. As we can see, results of model M-E64-5 are way better than the remaining two. One plausible reasoning for such a performance is long short term memory network are able to learn long term dependencies. More captions provide better context to the LSTM networks for the caption generation. Therefore such a score is observed.

- **Ensemble Models**

  From the experiment performed on ensemble model of visual feature extractors, it is observed that the model gives poor results. This model was indeed implemented in both views and evaluated on a single caption. The BLEU score observed on the ensemble model are shown in table 6.3. Abbreviation E-I-E64 means ensemble model in implant view trained with embedding word vector of length 64.

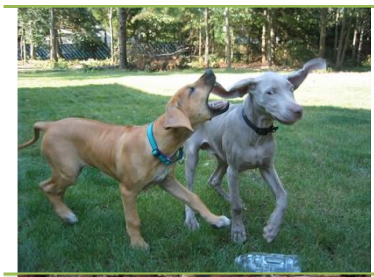
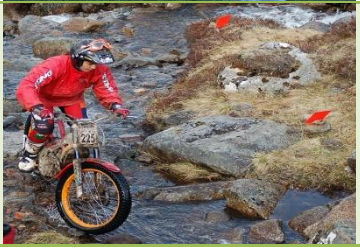
BLEU Scores				
	BLEU-1	BLEU-2	BLEU-3	BLEU-4
E-I-E64	0.3382	0.1407	0.0987	0.0413
E-M-E64	0.3460	0.1662	0.1199	0.0531

Table 6.3: BLEU Scores noted on Ensemble Models

As clearly visible in table 6.3, these results show a slight variation with predefined merge and implant architecture. One plausible explanation is that both the visual feature extractor VGG-16 and InceptionV3 are trained on a same dataset. In the ensemble model, their last layer feature vector which represents the most important feature of the image is exploited. So somehow both architectures learnt the same features and these same features if combined are just a copy of each other.

- **Tested Samples**

  To conclude this chapter, some samples are presented which are tested on the overall best model so far proposed in this work. These samples are shown in figure 6.3.

	**dogs are running on grass**
	**man in red shirt riding bike**

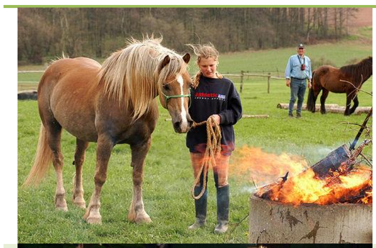
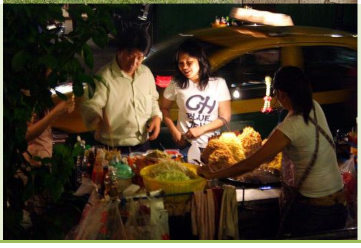
	**girl standing with horse**
	**a girl and a boy in white shirt buying fruits**

Figure 6.3: Tested Samples

7. Discussion

In this work, two views of recurrent neural network in an image captioning systems are proposed. These views consider recurrent neural network as a generator and encoder respectively.

Recurrent neural network if considered as a component to generate captions, image access is necessary so that recurrent neural network will know what to extract from the image and generate as caption. But this view of taking recurrent neural network as a generator component does not give beneficiary result, so this is not the case seemingly.

However if another view of recurrent neural network is considered which is to encode the textual data instead of being viewed as a generative component, it makes sense because implant architecture does suffer great loss in performance than its competitor merge architecture. One possible explanation of this happening can be the prefix size. When textual data is provided as input to the recurrent network then at each time step prefix size is increased where there is addition of one word in the corresponding prefix. Each such prefix is framed in a fixed size vector.

In implant architecture, to encode is much difficult because of image features that are included with the textual features. Indeed implant architecture follow the basic rule of caption generation where every word in the caption is concatenated with the visual features. There are two reasons for this difficulty. First is the requirement of compressing the prefixes of the captions with the visual features in a vector of limited size or fixed size. Second being growth. There is a huge increase in the vocabulary size handled by recurrent neural network because each prefix is a sum of image and word. In merge architecture, this problem is removed since recurrent network task is to encode text rather than encoding image and text. Although there is a load on the layer in which both the image and encoded textual features are merged and fed into.

In general, implant architecture shows worse performance than the merge architecture. Therefore, recurrent neural network should be better viewed as an encoder or a learner of language representations which can further fed into the neural network that is predicting the caption based on previous predictions. If recurrent neural network was the primary component of generation module of image captioning systems than it would need the visual features but this thinking cause performance loss.

So, to conclude everything if merging of features is needed in neural network architecture then it would be better to first encode both the representations and then feed into a multimodal layer rather than putting everything into the same recurrent neural network via separate input pipeline. To this point, best view of recurrent neural network will be to learn linguistic representations.

A quite different point of view is also experimented and proposed in this work. This view is the combination of two visual feature extractors where both of them are convolutional neural networks and winners of ImageNet Competition. First features are extracted from both of the feature extractors and then both features are compressed into a fixed size vector separately. By passing through a feed forward layer and then combining with recurrent neural network everything goes through a soft-max layer that samples the most probable word required in the caption conditioned on the previously generated sequence.

Now this architecture after extracting visual features also follow both views named implant and merge discussed previously. Although two feature extractors are combined together yet there is no significant increase in performance. One plausible explanation of this can be that both feature extractors are designed to learn the same thing that is both were generated based on ImageNet classification and visual features were extracted from the last layer of both the networks that is the layer which possess the most important features of the scene or image. That is why these architectures are not able to attain the satisfiable performance.

If both architectures are compared that is if comparison of implant and merge architecture is considered in terms of combined feature extractors' model, results are same. Merge still shows better performance even when two visual feature extractors are used.

One more thing to notice is that on increasing number of captions long short term memory networks that are preferred recurrent neural network in this work performs better. The performance of model is directly proportional to the increment in number of captions. The reason is to do with the working of long short term memory networks.

In this work, a discussion is also performed on the working of attention model and the visualization of feature vectors. This discussion shows that on applying attention to such type of learning architectures where focus on different regions is required model performance can be improvised. Attention brings out the activations of the regions other than the most important one thus playing a major role in such type of learning architectures. Not only that attention can also be applied in machine translation problems, sequence to sequence problems like question-answering, chat bots etc.

8. FUTURE SCOPE

The inferences discovered in this work give invitation to more research on the applicability of merge architecture in pool of different domains. The science of transfer learning can leverage from the merge view of image captioning systems where the recurrent neural network that is used for captioning can be replaced with the general corpus trained neural language model. However, implant architecture cannot have such advantage because it needs both the image as well as the text to do its learning. Future researches can be done to see how a transfer learning based neural language model trained on a general corpus if transferred to image captioning systems in place of the recurrent neural network that is used to perform the encoding of the linguistics performs in the caption generator.

References

[1] M. Hodosh, P. Young and J. Hockenmaier (2013) "Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics", Journal of Artificial Intelligence Research, Volume 47, pages 853-899

[2] Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. Collecting Image Annotations Using Amazon's Mechanical Turk. In Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk.

[3] Bernardi, Raffaella, et al. Automatic Description Generation from Images: A Survey of Models, Datasets, and Evaluation Measures. J. Artif. Intell. Res.(JAIR) 55 (2016): 409-442.

[4] Karpathy, Andrej. CONNECTING IMAGES AND NATURAL LANGUAGE. Diss. STANFORD UNIVERSITY, 2016.

[5] Karpathy A, Fei-Fei L. Deep visual-semantic alignments for generating image descriptions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015: 3128-3137.

[6] Vinyals, Oriol, et al. Show and tell: A neural image caption generator. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[7] Xu, Kelvin, et al. Show, attend and tell: Neural image caption generation with visual attention. International Conference on Machine Learning. 2015.

[8] Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. arXiv preprint arXiv:1601.06759 (2016).

[9] Kiros R, Salakhutdinov R, Zemel R S. Unifying visual-semantic embeddings with multimodal neural language models. arXiv preprint arXiv:1411.2539, 2014.

[10] Hao Fang, Saurabh Gupta, Rupesh K. Srivastava, Li Deng, “From Captions to Visual Concepts and Back”, Microsoft

[11] K. Simonyan, A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, arXiv:1409.1556, University of Oxford.

[12] Bryan A. Plummer et. al. “Flickr30k Entities: Collecting Region-to- Phrase Correspondences for Richer Image-to- Sentence Models”, arXiv:1505.04870 [cs.CV]

[13] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, “Deep Captioning With Multimodal Recurrent Neural Networks”, Baidu, I.2.6; I.2.7; I.2.10, ICLR 2015, https://arxiv.org/pdf/1412.6632

[14] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization
, volume 29, pages 65–72, 2005.

[15] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition , pages 4566–4575, 2015
[16] Visual Geometry Group, http://www.robots.ox.ac.uk/~vgg/

[17] ImageNet Competition, http://image-net.org/

[18] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, "Rethinking the Inception Architecture for Computer Vision", https://arxiv.org/abs/1512.00567v3

[19] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, "Efficient Estimation of Word Representations in Vector Space", https://arxiv.org/abs/1301.3781v3

[20] http://cs231n.github.io/transfer-learning/

[21] Kishore Papineni, Salim Roukos, Todd Ward, And Wei-Jing Zhu, "BLEU: a Method for Automatic Evaluation Of Machine Translation", IBM T.J. Watson Research Center, NY10598, USA, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp 311-318

[22]https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

[23] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Jauvin, "A Neural Probabilistic Language Model", Département d'Informatique et Recherche Opérationnelle, Centre de Recherche Mathématiques, Université de Montréal, Montréal, Québec, Canada, Journal of Machine Learning Research 3 (2003) 1137–1155

[24] http://colah.github.io/posts/2015-08-Understanding-LSTMs/