ABSTRACT

Computer vision has become universal in our society, with applications in various fields. In this project, we focus on one of the visual recognition facets of computer vision, i.e. image captioning. The problem of generating language descriptions for visual data has been studied from a long time but in the field of videos. In the recent few years emphasis has been lead on still image description with natural text. Due to the recent growth in the field of object recognition, the errand of scene depiction in a picture has turned out to be simpler.

The point of the venture was to train convolutional neural systems with a few several hyperparameters and apply it on a tremendous dataset of pictures (Image-Net), and join the consequences of this picture classifier with an intermittent neural system to produce an inscription for the grouped picture. In this report we present the definite engineering of the model utilized by us. We accomplished a BLEU score of 56 on the Flickr8k dataset while the cutting-edge results lay at 66 on the dataset.

TABLE OF CONTENTS

Chapter No.	Topics	Page No.
	Certificate	ii
	Declaration	iii
	Abstract	iv
	Acknowledgement	v
	List of Figures	vii
Chapter-1	1.1 Introduction	1
	1.2 Work from previous semester	3
	1.3 Deep Learning	6
	1.4 Software Used	7
	1.5 Libraries Used	9
Chapter-2	2.1 Dataset	11
	2.2 Architecture	14
	2.3 Building the Model	26
	2.4 LSTM	19
	2.5 Prediction with image as initial state	20
	2.6 Captions as a search problem	20
	2.7 Beam Search	21
Chapter-3	3.1 Results	23
	Conclusion & Future Scope	27
	References	28
	Plagiarism Certificate	29

LIST OF FIGURES

Label **Page No.**

1.1 Flowchart of the Speech to Text Procedure …… 5

2.1 An Image from MNIST Dataset ……. 11

2.2 A Snapshot from MNIST Dataset ……..…. 12

2.3 Predicted Results of the Dataset …….....…….. 12

2.4 Flickr30k Dataset Entities ……... 19

2.5 A CNN-LSTM Image Caption Architecture ……...……..... 14

2.6 Micro-Architecture of VGG16 Model ……...…. 15

2.7 Mapping input to embedding source ……... 18

2.8 The repeating module in an LSTM contains four interacting layers 19

2.9 Beam Search ……. 21

3.1 Image 1, Epoch 10 …… 23

3.2 Image 1, Epoch 20 …… 23

3.3 Image 1, Epoch 35 …… 24

3.4 Image 2, Epoch 10 …… 24

3.5 Image 2, Epoch 20 …… 24

3.6 Image 2, Epoch 35 …… 25

3.7 Result from Model Trained So far …… 25

3.8 Final Result on a Real Time Image from Model Trained So far ... 26

CHAPTER-1

1.1 INTRODUCTION

Image captioning is the process by which we can get a textual description of an image. It uses both Natural Language Processing and Computer Vision to develop the captions or the descriptions of the image. Automatically defining the content of an image is an intrinsic problem in artificial intelligence that NLP (connects natural language processing) and computer vision. Here, we present a generative model that can be used to generate natural sentences describing an image or a picture, based on a deep recurrent architecture that combines machine translation and recent advances in computer vision. The model is trained to maximize the probability of the target description sentence given the training image. Experiments on various datasets show how accurate the model is and the fluency of the language it learns purely from image descriptions. Our model is often quite exact, which we verify both quantitatively and qualitatively.

Being able to automatically define the content of an image using well-formed English sentences is a very tough task, but it could have great impact, for instance by helping to better understand the content of images on the web by visually impaired people. This task is undoubtedly harder, than the well-studied object recognition or image classification tasks, which has been a main focus in the computer vision community. Indeed, a characterization must express how these objects relate to each other as well as their traits and the activities they are indulged in and not only capture the objects contained in an image. Moreover, a language model is needed in addition to visual understanding since the above linguistic knowledge has to be expressed in a natural language.

Most past endeavors have proposed to line together existing arrangements of the above sub-issues, so as to go from a picture to its portrayal. Interestingly, we might want to exhibit in this work a solitary joint model that accepts a picture I as info, and is prepared to augment the probability of delivering an objective succession of words S = {S1, S2, . . .} where each word St originates from a given lexicon, that depicts the picture enough. The primary motivation of our work originates from late advances in machine interpretation, where the undertaking is to change a sentence S written in a source language, into its interpretation T in the objective language, by boosting p(T|S). For a long time, machine interpretation was likewise accomplished by a progression of isolated undertakings (deciphering words separately,

adjusting words, reordering, and so on), however late work has demonstrated that interpretation should be possible in a lot more straightforward way utilizing Recurrent Neural Networks (RNNs) and still achieve cutting edge execution. An "encoder" RNN peruses the source sentence and changes it into a rich fixed-length vector portrayal, which thus in utilized as the underlying shrouded condition of a "decoder" RNN that produces the objective sentence. Here, we propose to pursue this rich formula, supplanting the encoder RNN by a profound convolution neural system (CNN). In the course of the most recent couple of years it has been convincingly demonstrated that CNNs can deliver a rich portrayal of the information picture by installing it to a fixed-length vector, to such an extent that this portrayal can be utilized for an assortment of vision errands. Henceforth, it is normal to utilize a CNN as a picture "encoder", by first pre-preparing it for a picture order assignment and utilizing the last shrouded layer as a contribution to the RNN decoder that produces sentences. We consider this model the Neural Image Descriptor, or NID [1].

1.2 WORK FROM PREVIOUS SEMESTER

Time has been the proof of our usage of deep learning. The key is, Deep Learning is Data and Computing power together. Speech recognition is invading our lives. It's built in our phones, game consoles, smart watches, etc. Not only this it's even automating our homes. But why is it just hitting the mainstream now when it has been around for decades? The reason is "deep learning" which finally made speech recognition accurate enough to be used outside the carefully controlled environments. Our work was simply based on feeding sound into a neural network and training it to produce text.

Speech recognition basically is the inter-disciplinary sub-field of computational semantic that develops methodologies and technologies that allows the recognition and translation of spoken language into text by computers. It is also known as STT (speech to text). It incorporates knowledge and research in the computer science, and electrical engineering fields. Some speech recognition systems require to be trained where an individual speaker reads text into the system. The system recognizes the person's specific voice and uses it to fine-tune the recognition of that person's speech, resulting in increased precision. Systems that do not use training are called "speaker independent" systems and the one that use training are called "speaker dependent". The applications of speech recognition include voice user interfaces such as voice dialing, call routing, domestic appliance control, search, simple data entry, preparation of structured documents, speech-to-text processing.

Examples of Speech recognition applications includes voice user interfaces such as voice dialing (e.g. "call home"), call routing (e.g. "I would like to make a collect call"), search (e.g. find a podcast where particular words were spoken), simple data entry (e.g., entering a credit card number), structured documents preparation(e.g. a radiology report), Speaker Characteristics determination speech-to-text processing (e.g., word processors or emails), and aircraft (usually termed direct voice input).

The term voice recognition or speaker identification is not recognizing what the speaker is saying instead it rather refers to identifying the speaker. Recognizing the speaker simplifies the task of translating speech in systems that has been trained on some specific person's voice or it can be used to verify the identity of the speaker as a part of a security process.

From technology perspective, speech recognition with several waves of major innovations has a long history. Recently, this field has been benefited from advances in deep learning and big

data. The advances are evidenced by the worldwide industry adoption of various deep learning methods in designing speech recognition systems and not only by the surge of academic papers published in the field. These speech industry players include Google, Microsoft, IBM, Baidu, Apple, Amazon, Nuance, Sound Hound many of which have publicized the core technology in their speech recognition systems.

Siri, Google home and all the other services provided to us today are based on Neural Network. And, now, with the help of Machine Learning, TensorFlow, we build a deep Neural network which could recognize spoken numbers and words. To explain 'Tensor', it is just a multidimensional data. To see if our data was being recognized, we included, downloading the Dataset, building a Neural Network, training it on that data and testing it out. Since the input was a sequence of soundwaves a Python code was used to solve many problems as.

Breaking it down, to offer state of the art speech recognition we used LSTM (Long Short-Term Memory) network and TFLearn which helped us build neural networks faster. Moreover, Hyperparameters like the learning rate, was balanced between Time and Accuracy, which gave it a touch of perfection.

Our work was a two-way process based on simply feeding sound recordings into a neural network and train it to produce the output text. The problem was the variable speed of speech. One person might say “hello!” very fast while another might say “heeeeeellllllllooooo!” too slow, producing a much longer sound file with much more data. Both the sound files should be recognized exactly as the same text – “hello!”. It turns out to be pretty hard to automatically align audio files of various lengths to a fixed length piece of text. To work around this used some special and extra processing in addition to a deep neural network.

1.2.1 PROCEDURE FOLLOWED

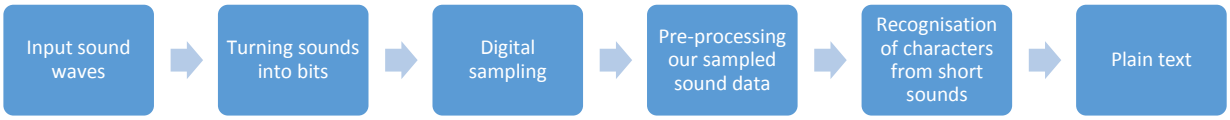

Figure 1.1: Flowchart of the Speech to Text Procedure

1.2.2 OUTPUT

We prepared two models. One with the same data set with which the model was trained and the other model is based on the real-time dataset. The model was trained successfully and was giving an output with the accuracy of 98% when the input given was the voice with which the model was trained while in case of real time it was giving an accuracy of 22%. The accuracy was low in case of voices other than the voices with which the model was trained due to low computing power because of which we cannot take big data sets.

1.3 DEEP LEARNING

Deep learning (also known as **deep structured learning** or **hierarchical learning**) is part of a broader family of machine learning methods based on the layers used in artificial neural networks. Learning can be supervised, semi-supervised or unsupervised [2][3][4].

Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases superior to human experts [5][6].

Neural networks were originally inspired by information processing and distributed communication nodes in biological systems synaptic structures yet have various differences from the structural and functional properties of biological brains, which make them incompatible with the neurological evidence. Specifically, Neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plasticity) and analog.

1.4 SOFTWARE USED

1.4.1 PYTHON

Python is an interpreted high-level programming language which can be used as a server to create web applications. It allows you to work more quickly and integrate system efficiently. First released in 1991, it was created by Guido van Rossum. Python's design philosophy highlight code readability with its remarkable utilization of noteworthy whitespace. Its language develops and object-arranged methodology plans to enable software engineers to compose clear, coherent code for little and substantial scale ventures.

Python is progressively composed and garbage-collected. It underpins various programming ideal models, including procedural, object-arranged, and useful programming. Python is frequently described as a "batteries included" language because of its complete standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council [7].

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages Python and CPython.

1.4.2 ANACONDA

To perform Python/R data science and machine learning on Windows, Linux and Mac OS X, the easiest way is the open-source Anaconda Distribution. Enabling individual data scientists to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with Conda
- Develop and train machine learning and deep learning models with scikit-learn, TensorFlow, and Theano
- Analyze data with scalability and performance with Dask, NumPy, pandas, and Numba
- Visualize results with Matplotlib, Bokeh, Datashader, and Holoviews

it is the industry standard for testing, developing, and training on a single machine, with about 11 million users worldwide.

1.5 LIBRARIES USED

1.5.1 TENSORFLOW

TensorFlow is a free open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015.

1.5.2 NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Num-array into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

1.5.3 PANDAS

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals

1.5.4 NLTK

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania. NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit, plus a cookbook.

NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. There are 32 universities in the US and 25 countries using NLTK in their courses. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.

CHAPTER-2

2.1 DATASET

2.1.1 Fashion-MNIST

Fashion-MNIST is a dataset of Zalando's article pictures, comprising of a preparation train set of 60,000 images and a test set of 10,000 precedents. Every precedent is a 28x28 grayscale picture, related with a mark from 10 classes. Zalando expects Fashion-MNIST to fill in as an immediate drop-in swap for the first MNIST dataset for benchmarking AI calculations. It has a similar picture size and structure of preparing and testing parts.

Each picture is 28 pixels in stature and 28 pixels in width, for a sum of 784 pixels altogether. Every pixel has a solitary pixel-esteem related with it, showing the softness or dimness of that pixel, with higher numbers meaning darker. This pixel-esteem is a number somewhere in the range of 0 and 255. The preparation and test informational collections have 785 segments. The main section comprises of the class names, and speaks to the vestment. The remainder of the segments contain the pixel-estimations of the related picture.

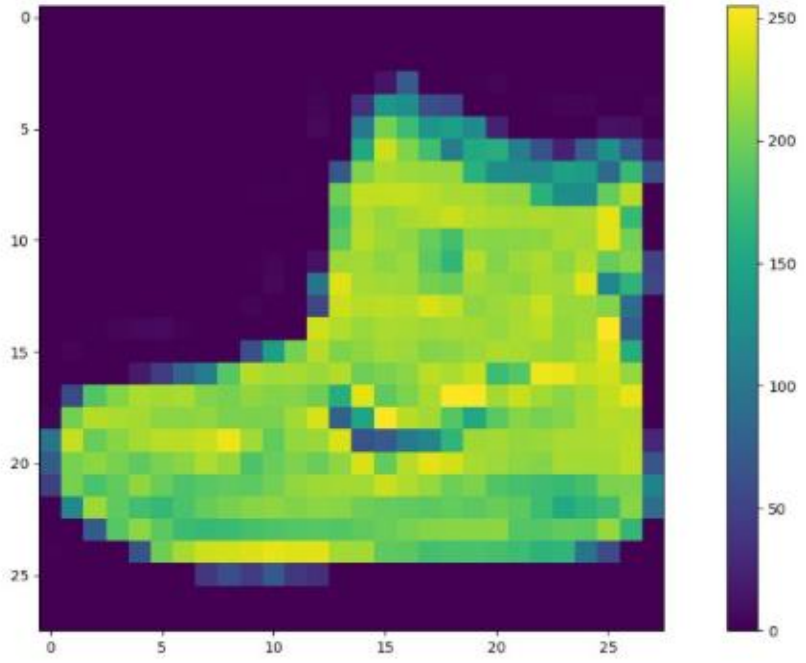

Figure 2.1: An Image from MNIST Dataset

Figure 2.2: A Snapshot from MNIST Dataset

Figure 2.3: Predicted Results of the Dataset

2.1.2 Flickr30k

The Flickr30k dataset has become a regular benchmark for sentence-based image description. Flickr30k Entities augments the 158,915 captions from Flickr30k with 244k grammatical relation chains, linking mentions of constant entities across totally different captions for constant image, and associating them with 275,775 manually annotated bounding boxes. Such annotations square measure essential for continuing progress in automatic image description and grounded language understanding. They permit U.S.A. to outline a replacement benchmark for localization of matter entity mentions in a picture. Flickr30k dataset features:

- 31783 Images
- 8.7 Objects per Image
- 44518 Object Categories
- 6.2 Objects per Category
- 5 Sentences per Image
- 16.6 Expressions per Image

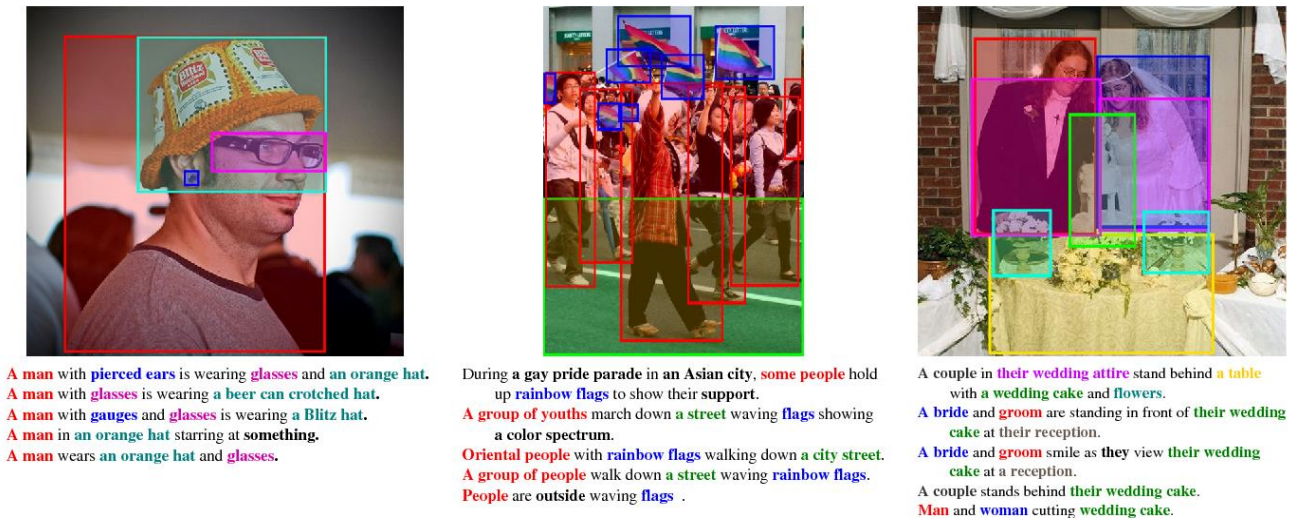

Figure 2.4: Flickr30k Dataset Entities

2.2 Architecture

The image captioning model is an example of encoder-decoder neural network, as it works by first “encoding” an image or picture into a fixed length vector followed by “decoding” the representation into a description in a natural language.

The image encoder is a convolutional neural network. The convolutional neural network is largely used for image tasks and is state-of-the-art for detection and image recognition currently. The VGG16 image recognition model pretrained on the Flickr30k image classification dataset is our choice of network.

The decoder on the other side is LSTM network. LSTM is widely used for sequence modeling tasks such as machine translation. In image captioning model, the LSTM network is trained as a language model.

Words in the descriptions are represented with an embedding model. Every word in the vocabulary is associated with a fixed-length vector representation that is learned during training. The following diagram illustrates the model architecture:

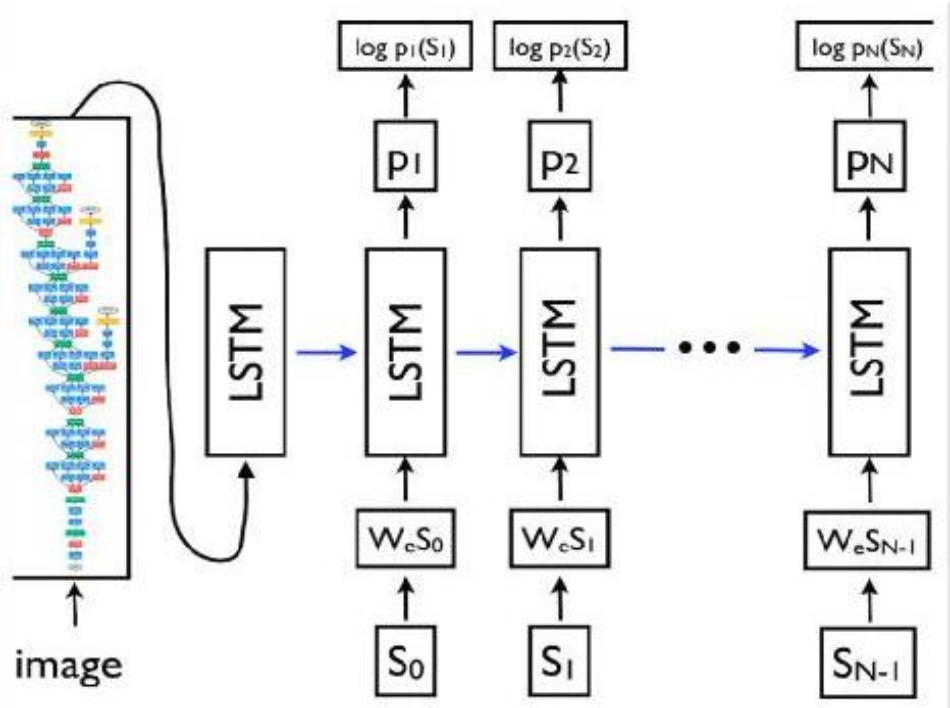

Figure 2.5: A CNN-LSTM Image Caption Architecture [1]

In Fig. 2.5, {S0, S1, ..., SN-1} are the words of the caption or the description and {WeS0, WeS1, ..., WeSN-1} are their corresponding word embedding vectors. Probability distributions generated by the model for the next word in the sentence are the outputs {p1, p2, ..., pN} of the

Long short-term memory. The terms {log p1(s1), log p2(s2), ..., log pN(sN)} are the log likelihoods of the correct word at each step. Minimization objective of the model is the negated sum of these terms.

The parameters of the Inception v3 model are kept fixed during the first phase of training: it simply is a static image encoder function. To change the image embedding into the word embedding vector space, a single trainable layer is added above the Inception v3 model. The model is trained according to the parameters of the word embeddings, the parameters of the layer on top of VGG16 and the parameters of the LSTM. In the second part of training, all parameters including the parameters of VGG16 are trained to jointly fine the image encoder and the LSTM.

Given a trained model and an image to generate captions for that image, we use beam search. Captions are generated word by word, where at each step t, to generate a new set of sentences with length t, we use the set of sentences already generated with length t - 1. We keep only the top k candidates at each step. The hyperparameter k is called the beam size. The best performance is found with k = 3.

2.2.1 VGG16 Model

Proposed by K. Simonyan in the paper "Very Deep Convolutional Networks for Large- Scale Image Recognition", VGG is a convolutional neural network model. With 92.7% the model achieves top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.

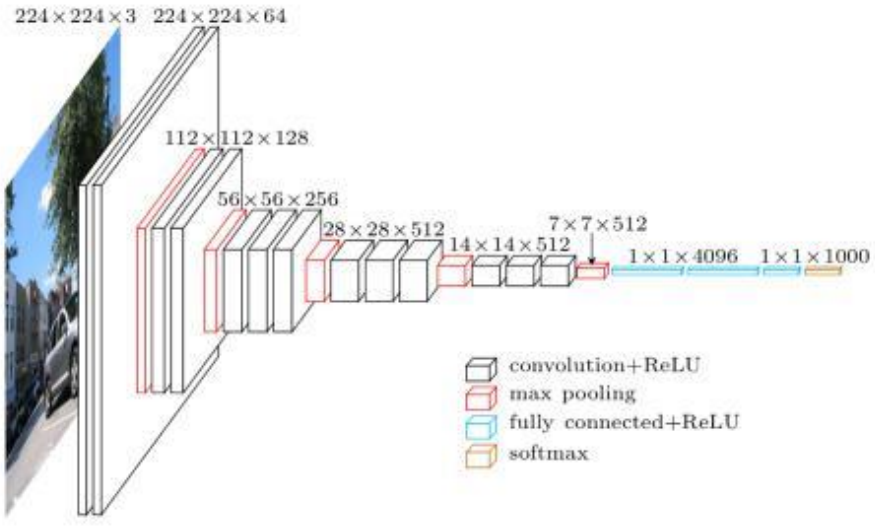

Figure 2.6: Micro-Architecture of VGG16 Model

2.3 Building the Model

2.3.1 Convolution Layer

The core building block that does most of the computational heavy lifting of a Convolutional Network is the Convolution layer.

The Convolution layer's parameters consist of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. During the forward pass, we slide (or convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the width and height of the input volume, we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some colour on the first layer, or eventually entire honeycomb or wheel-like patterns on higher layers of the network. Now, we will have an entire set of filters in each Convolution layer and each of them will produce a separate 2-dimensional activation map. We will stack these activation maps along the depth dimension and produce the output volume.

Three parameters control the size of the output volume:

- **Depth:** It corresponds to the number of filters we would like to use, each learning to look for something different in the input.
- **Stride:** The stride with which we slide the filter. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2 then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially.
- **Zero-padding:** Sometimes it is convenient to pad the input volume with zeros around the borders. It allows us to control the spatial size of the output volumes. Padding actually improves performance by keeping information at the borders.

We can compute the spatial size of the output volume as a function of the input volume size (W), the receptive field size of the Conv Layer neurons (F), the stride with which they are applied (S), and the amount of zero padding used (P) on the border. You can convince yourself that the correct formula for calculating how many neurons "fit" is given by (W−F+2P) *S+1.

2.3.2 Pooling Layer

It is common to periodically insert a Pooling layer in-between successive Convolution layers. Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the max or average operation.

2.3.3 Normalization Layer

The idea is to normalize the inputs of each layer in such a way that they have a mean output activation of zero and standard deviation of one. This is analogous to how the inputs to networks are standardized. This is the step when you apply an activation function, the most used function here is ReLu (Rectified linear unit). A rectified linear unit has output 0 if the input is less than 0, and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input.

2.3.4 Fully-Connected Layer

After feature extraction we need to classify the data into various classes, this can be done using a fully connected (FC) neural network. This layer basically takes an input volume (whatever the output is of the conv or ReLu or pool layer preceding it) and outputs an N dimensional vector where N is the number of classes that the program has to choose from. Each number in this N dimensional vector represents the probability of a certain class. The way this fully connected layer works is that it looks at the output of the previous layer and determines which features most correlate to a particular class.

2.3.5 Using a CNN for Image Embedding

A convolutional neural network can be used to create a dense feature vector. This dense vector, also called an embedding, can be used as feature input into other algorithms or networks. For an image caption model, this embedding becomes a dense representation of the image and will be used as the initial state of the LSTM.

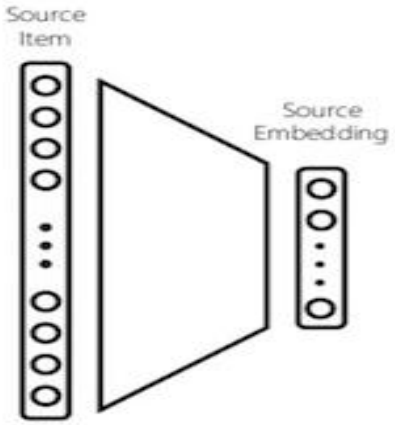

Figure 2.7: Mapping input to embedding source.

2.4 LSTM

Long Short-Term Memory networks are a special kind of RNN, that are capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber in 1997, and were refined by many people in following work. They work extremely well on a large variety of problems, and are used widely now.

LSTMs are especially designed to avoid the long-term dependency problem. Their default behavior is remembering information for long periods of time, not something they struggle to learn!

All intermittent neural systems have the type of a chain of rehashing modules of neural system. In standard RNNs, this rehashing module will have a basic structure, for example, a solitary tanh () layer.

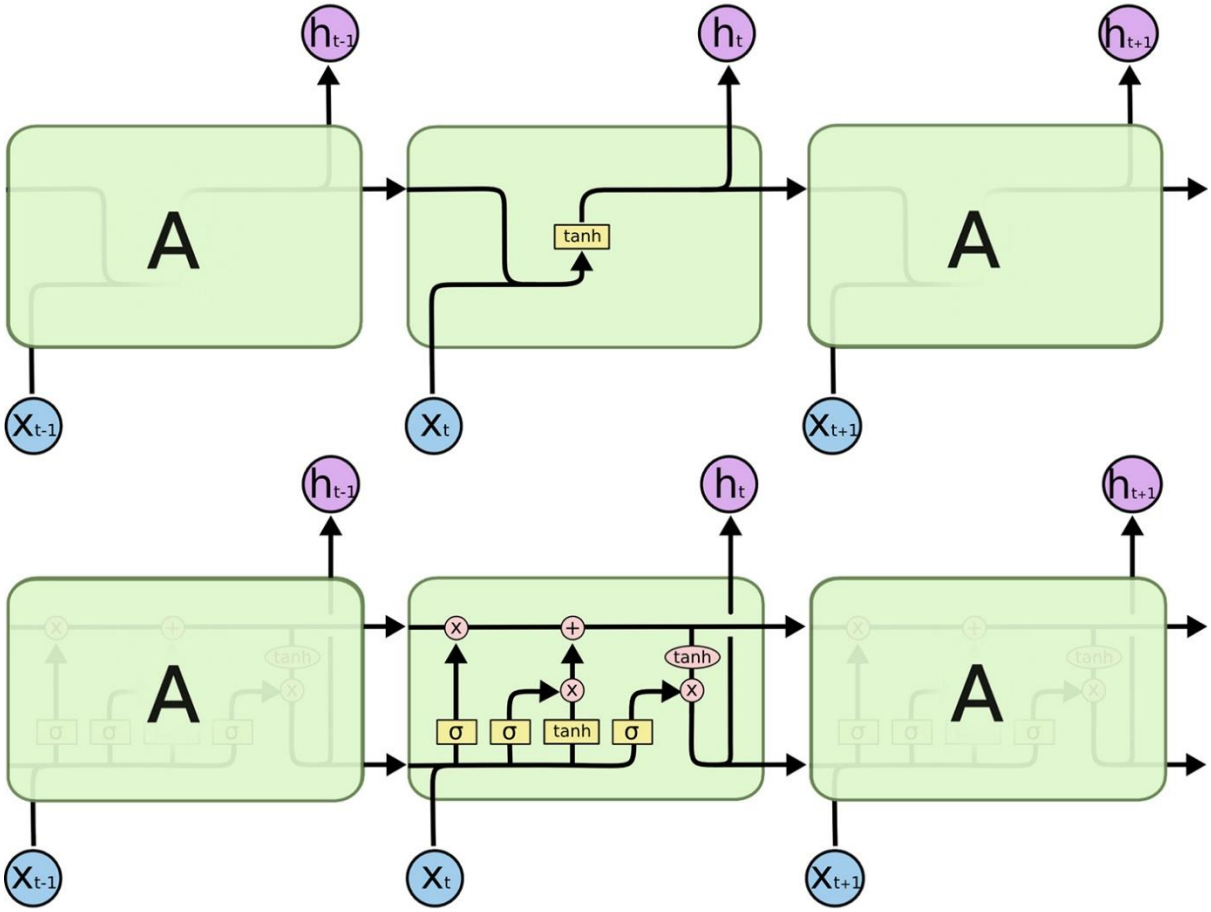

Figure 2.8: The repeating module in an LSTM contains four interacting layers.

2.5 Prediction with Image as Initial State

In a sentence language model, an LSTM estimates the next word in a sentence. Similarly, an LSTM attempts to predict the next character in a character language model, given the context of characters previously seen. You will create an image embedding in an image caption model. This embedding will then be fed into an LSTM as an initial state. This becomes the first prior state to the language model, influencing the next predicted words.

At each time-step, the previous cell state is considered by LSTM and a prediction for the most probable next value in the sequence is the output. This process is repeated until the sampling of end token, signaling the end of the caption.

2.6 Captions as a search problem

Generating a caption can be viewed as a graph search problem. Here, the nodes are words. The edges are the probability of moving from one node to another. Finding the optimal path involves maximizing the total probability of a sentence. Sampling and choosing the most probable next value is a greedy approach to generating a caption. It is computationally efficient, but can lead to a sub-optimal result. Given all possible words, it would not be computationally/space efficient to calculate all possible sentences and determine the optimal sentence. This rules out using a search algorithm such as Depth First Search or Breadth First Search to find the optimal path.

2.7 Beam Search

Beam search uses breadth-first search to build its search tree. At each level of the tree, it generates all successors of the states at the current level, sorting them in increasing order of heuristic cost. However, it only stores a predetermined number (β), of best states at each level (called the beam width). Only those states are expanded next. The greater the beam width, the fewer states are pruned. With an infinite beam width, no states are pruned and beam search is identical to breadth-first search. The beam width bounds the memory required to perform the search. Since a goal state could potentially be pruned, beam search sacrifices completeness (the guarantee that an algorithm will terminate with a solution, if one exists). Beam search is not optimal (that is, there is no guarantee that it will find the best solution).

In general, beam search returns the first solution found. Beam search for machine translation is a different case: once reaching the configured maximum search depth (i.e. translation length), the algorithm will evaluate the solutions found during search at various depths and return the best one (the one with the highest probability).

The beam width can either be fixed or variable. One approach that uses a variable beam width starts with the width at a minimum. If no solution is found, the beam is widened and the procedure is repeated.

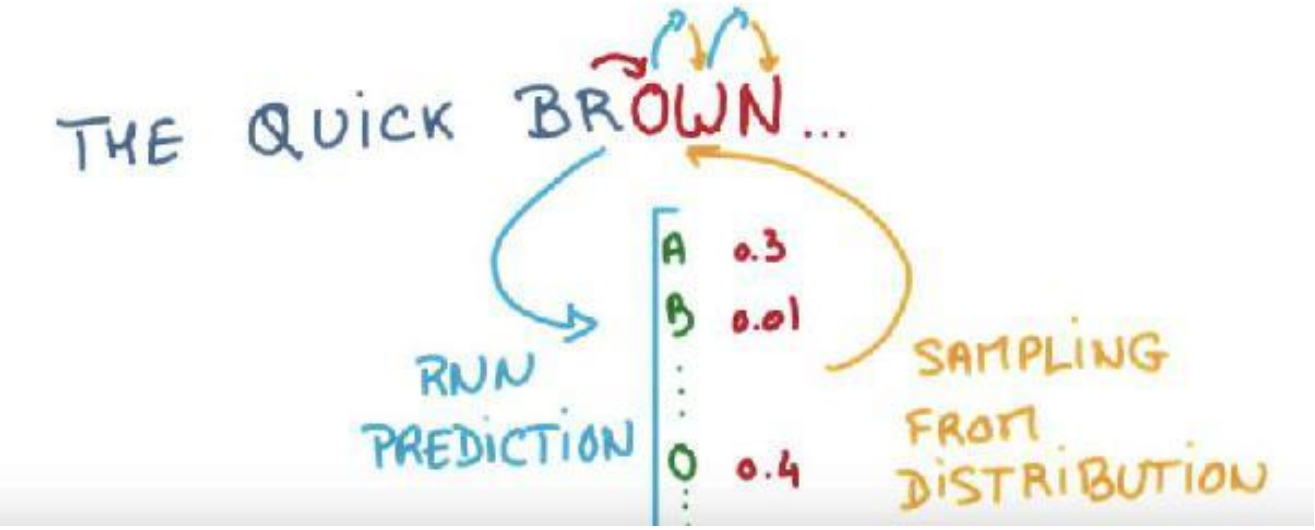

Figure 2.9: Beam Search.

A beam search is most often used to maintain tractability in large systems with insufficient amount of memory to store the entire search tree. For example, it is used in many machine translation systems. To select the best translation, each part is processed, and many different ways of translating the words appear. The top best translations according to their sentence structures are kept, and the rest are discarded. The translator then evaluates the translations according to a given criterion, choosing the translation which best keeps the goals. The first use of a beam search was in the Harpy Speech Recognition System, CMU 1976.

CHAPTER-3

3.1 Results & Discussion

We have presented NID (Neural Image Descriptor), an end-to-end neural network system that can automatically view an image and generate a reasonable description in plain English. NID is based on a convolution neural network that encodes an image into a compact representation, followed by a recurrent neural network that generates a corresponding sentence. The model is trained to maximize the likelihood of the sentence given the image.

Some of the results through different epochs are shown:

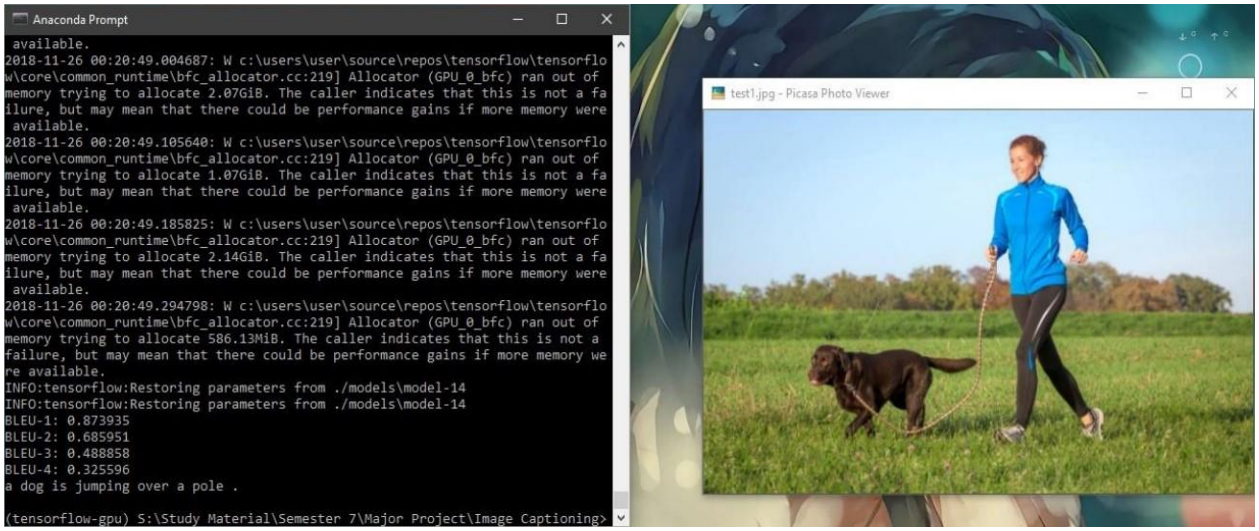

Figure 3.1: Image 1, Epoch 10.

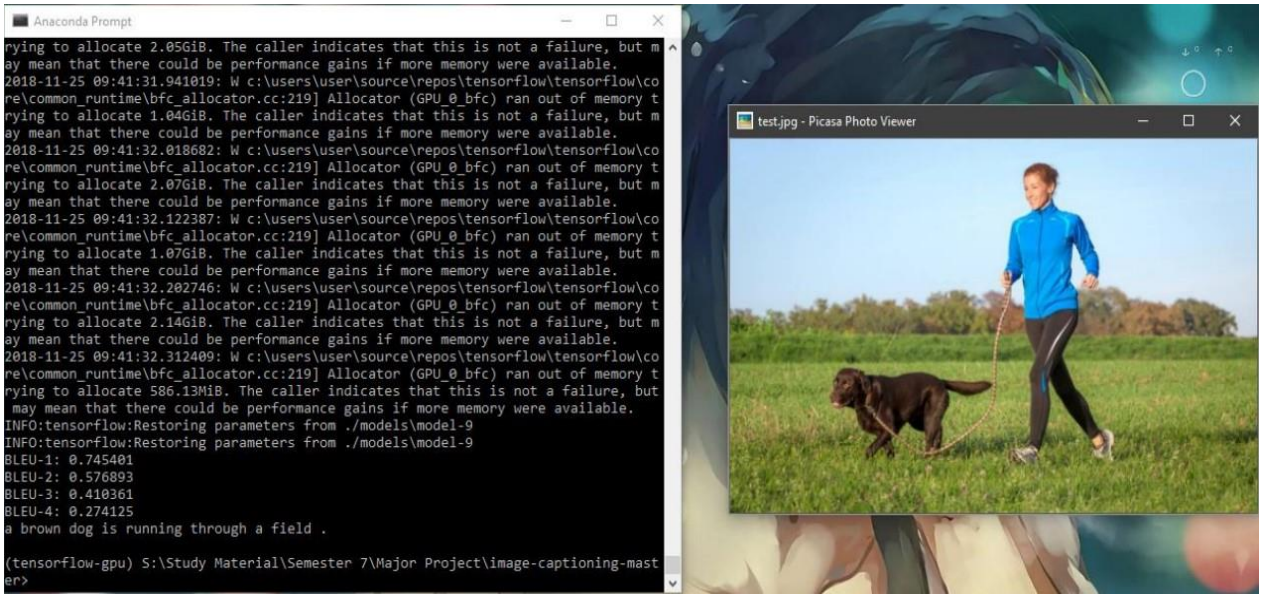

Figure 3.2: Image 1, Epoch 20.

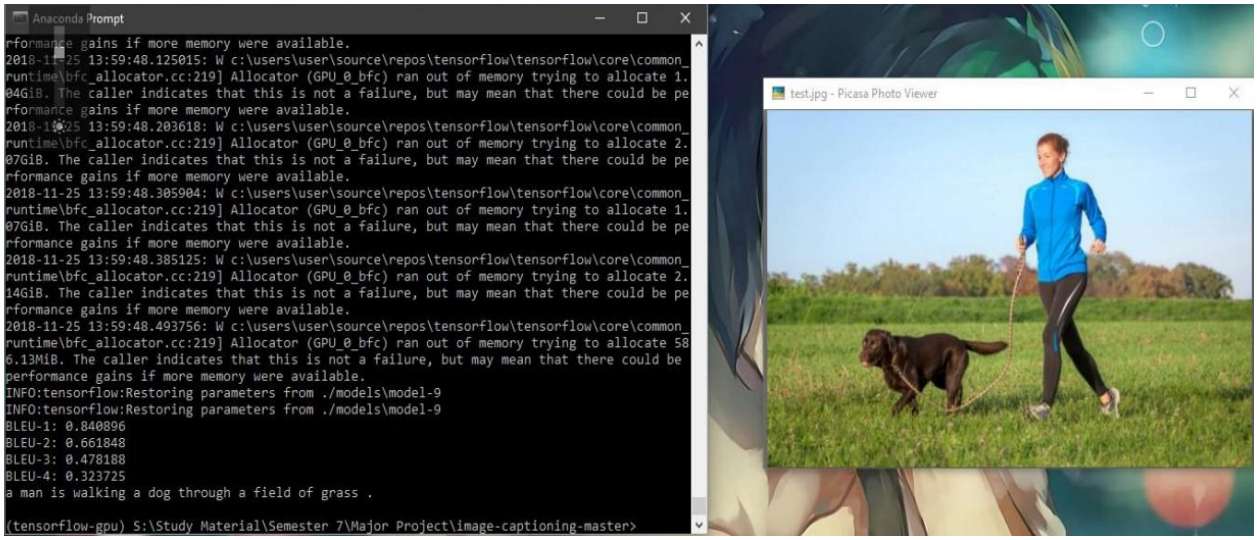

Figure 3.3: Image 1, Epoch 35.

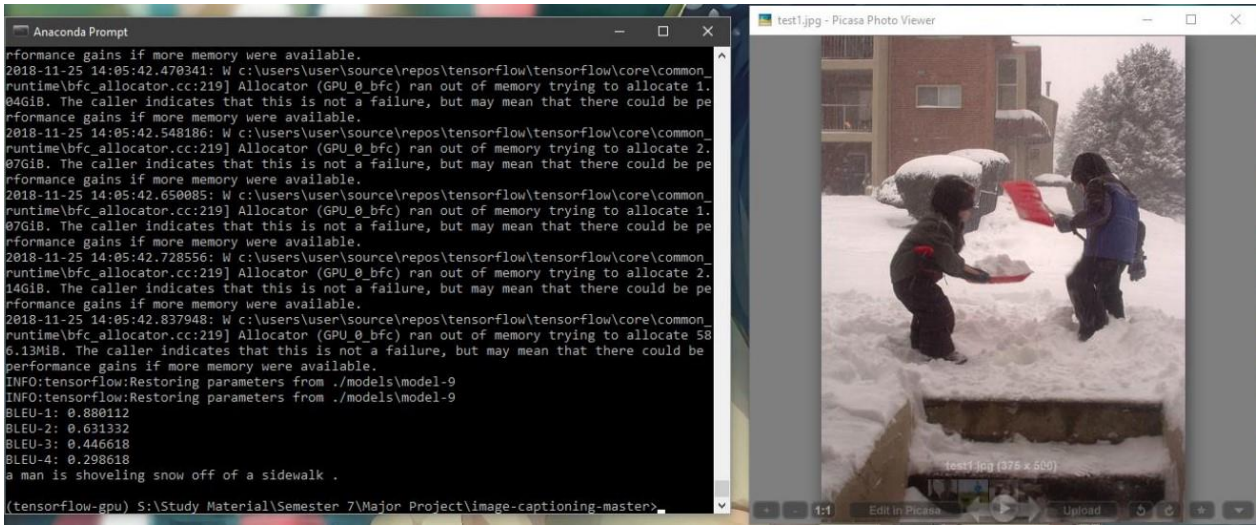

Figure 3.4: Image 2, Epoch 10.

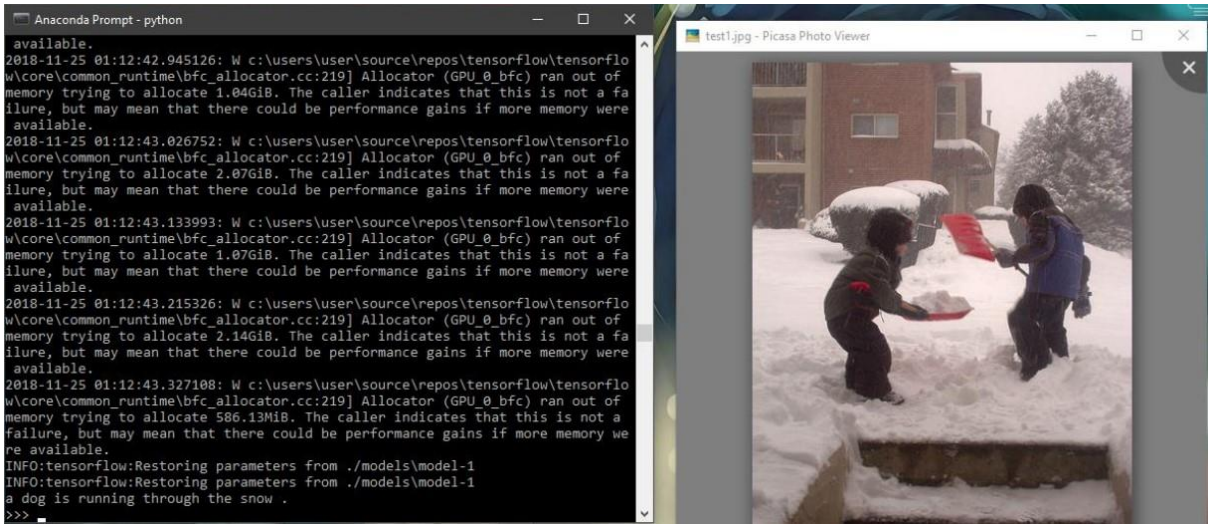

Figure 3.5: Image 2, Epoch 20.

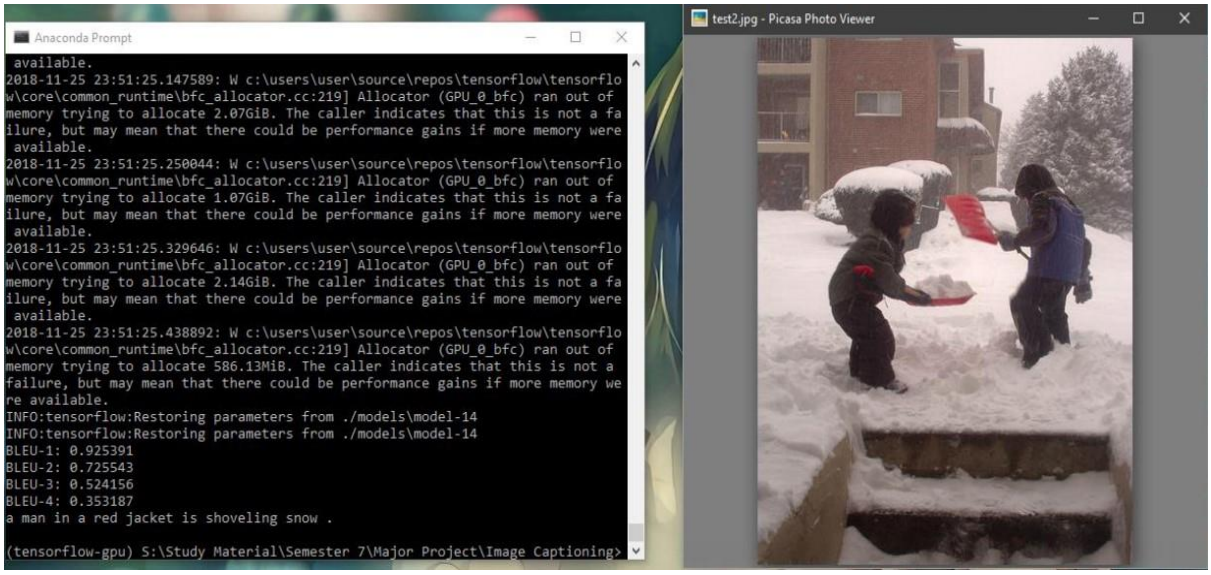

Figure 3.6: Image 2, Epoch 35.

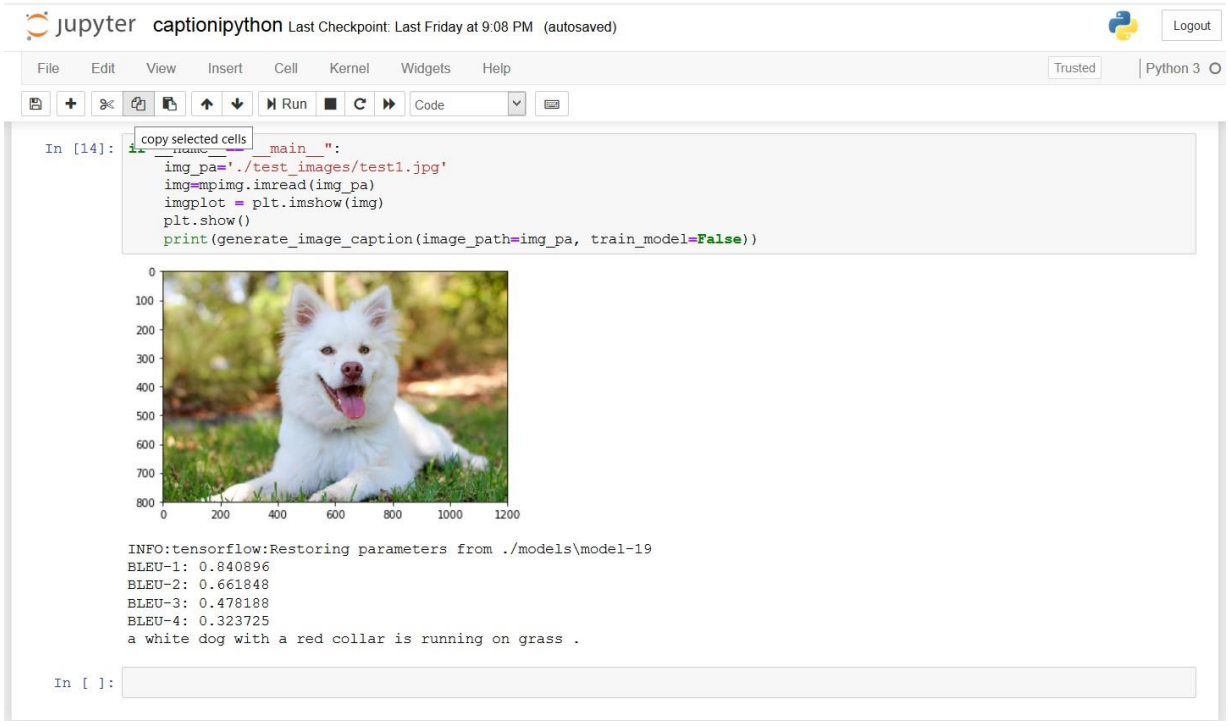

Figure 3.7: Result from Model Trained So far.

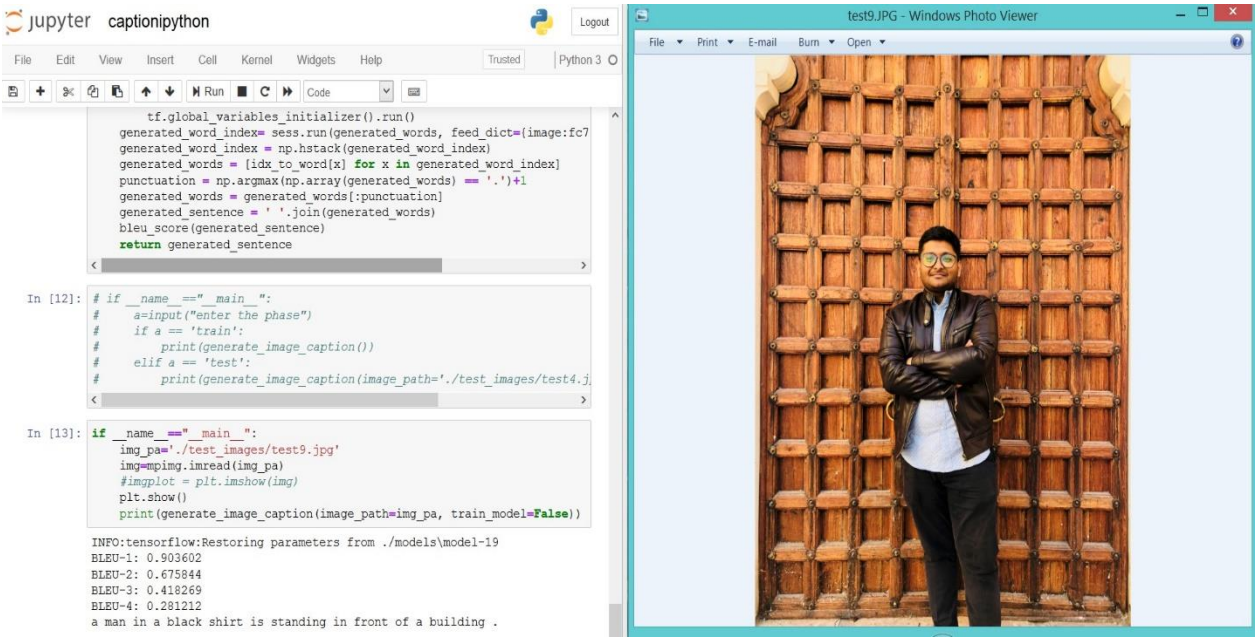

Figure 3.8: Final Result on a Real Time Image from Model Trained So far.

CONCLUSION & FUTURE SCOPE

The presented NID (Neural Image Descriptor), which is capable of viewing an image itself and generating an appropriate description in simple English. It is an end-to-end neural network system. NID encoding an image into a compact representation is based on a convolution neural network, followed by a recurrent neural network generating a corresponding appropriate sentence. The model is trained to maximize the accuracy of the sentence for the image. It is clear that as the size of the available image description datasets increases, so will the performance approaches towards NID.

Some Future Scope regarding the field include(s):

- A slightly long-term use of case would for sure be explaining what is happening in a video, frame by frame.
- Of great help for visually impaired people. A lot of applications can be developed in that space.
- Social Media. Platforms like Instagram can infer directly from the images, where you are present (beach, cafe etc.), what are you wearing (color, type of clothes) and most importantly what are you doing (playing, standing, etc.)
- Google Photos: Classifies your photo into animals, humans, hills, sea etc.
- Tesla/Google Self Drive Cars: All the self-driven cars are using image/video processing with neural network to achieve their goal.

REFERENCES

[1] Vinyals O., Alexander T., Samy B. and Dumitru E. "Show and tell: A neural image caption generator." *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015): 3156-3164.

[2] Bengio, Y., Courville, A.C., & Vincent, P. (2013). Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*, 1798-1828.

[3] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks: the official journal of the International Neural Network Society, 61*, 85-117.

[4] Goodfellow, I.J., Bengio, Y., & Courville, A.C. (2015). Deep Learning. *Nature, 521*, 436-444.

[5] Ciresan, D.C., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 3642-3649.

[6] Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM, 60*, 84-90.

[7] https://en.wikipedia.org/wiki/Python_(programming_language)

[8] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE transactions on pattern analysis and machine intelligence, 37 9*, 1904-16.

[9] Peng, Z., Dai, Y., Tang, Q., Cui, X., & Guo, S. (2018). Show and Tell: A Neural Image Caption Generator.

[10] Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., & Woo, W. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. *NIPS*.

[11] Udacity, "Beam Search," Internet: https://www.youtube.com/watch?v=UXW6Cs82UKo

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv:1409.0473, 2014.