Image Captioning

Industrial Training Report

submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

in

Information Technology

By

KESHAV – 40796303117

Maharaja Surajmal Institute of Technology

(Affiliated to Guru Gobind Singh Indraprastha University)

Janakpuri, New Delhi- 58

October 2019

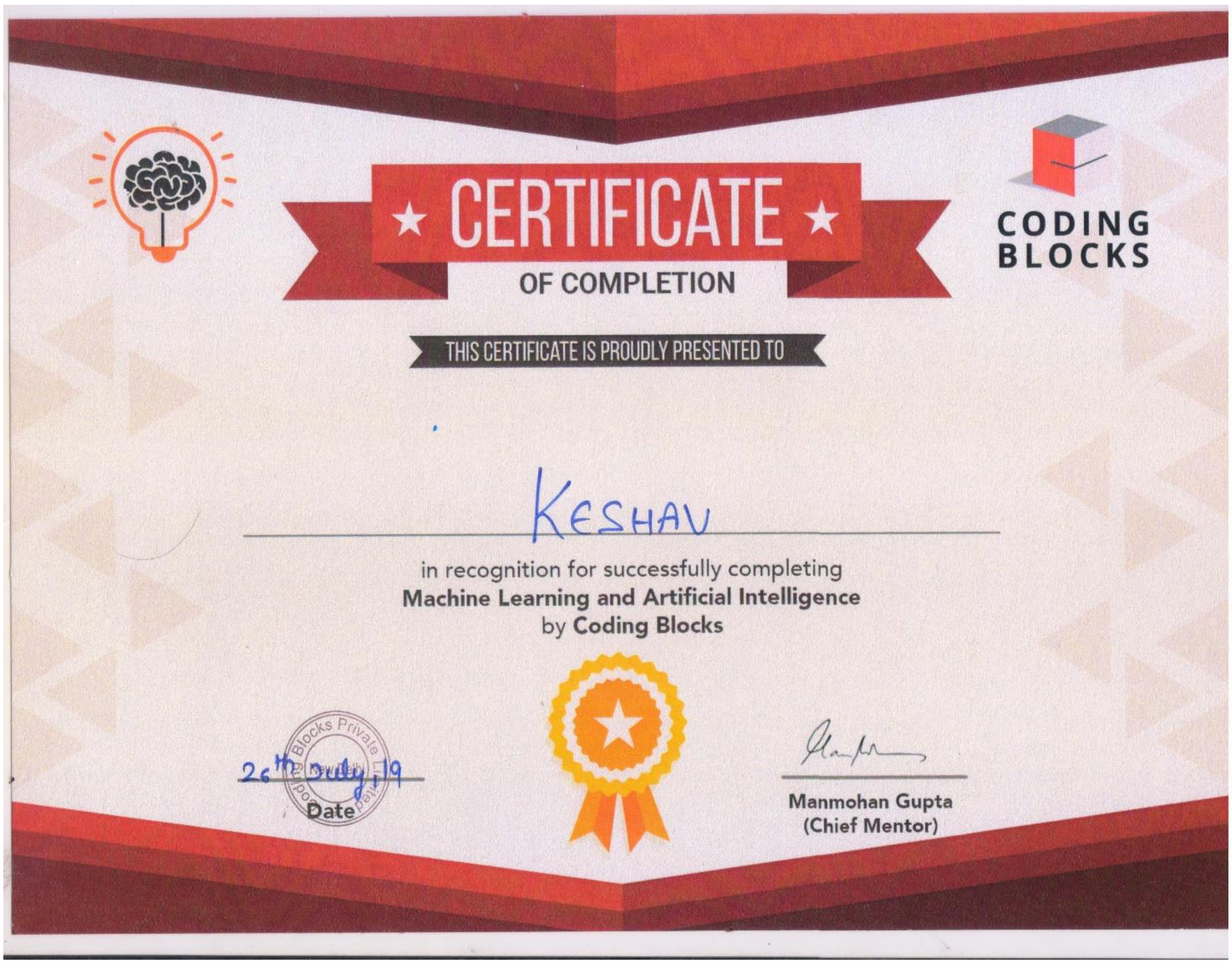
CANDIDATE'S DECLARATION

I, **KESHAV** , Roll No. 40796303117, B.Tech (Semester- 5th) of the Maharaja Surajmal Institute of Technology, New Delhi hereby declare that the Training Report entitled **"IMAGE CAPTIONING"** is an original work and data provided in the study is authentic to the best of my knowledge. This report has not been submitted to any other Institute for the award of any other degree.

Keshav

Enrollment No. 40796303117

CERTIFICATE



CERTIFICATE

OF COMPLETION

CODING BLOCKS

THIS CERTIFICATE IS PROUDLY PRESENTED TO

KESHAV

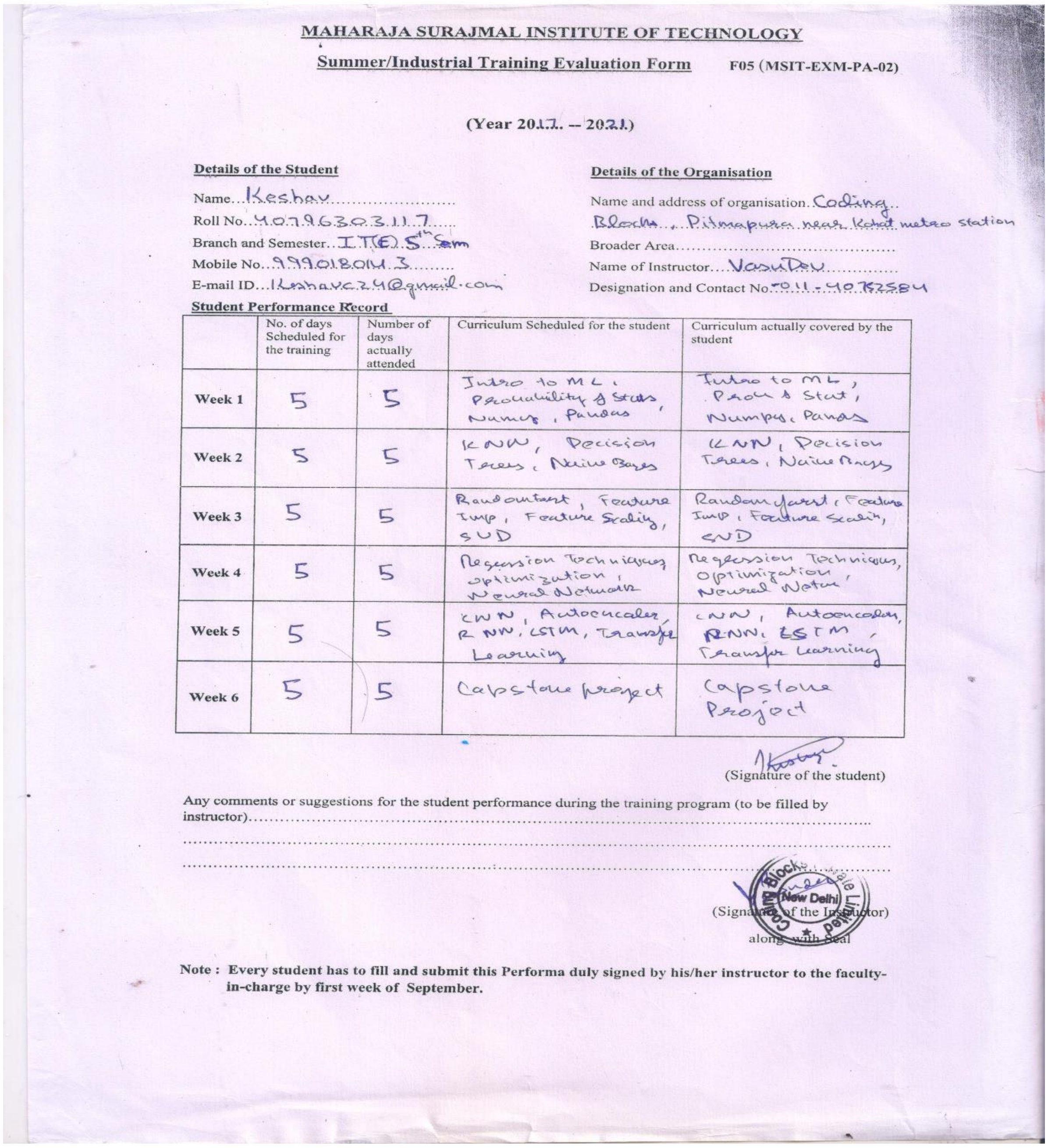
in recognition for successfully completing
Machine Learning and Artificial Intelligence
by **Coding Blocks**

26th July, 19
Date

Coding Blocks Private Limited
New Delhi

[illegible]
Manmohan Gupta
(Chief Mentor)

EVALUATION FORM



MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY

Summer/Industrial Training Evaluation Form **F05 (MSIT-EXM-PA-02)**

(Year 2017 -- 2021)

Details of the Student

Name: Keshav
Roll No.: 40796303117
Branch and Semester: IT(E) 5th Sem
Mobile No.: 9990180143
E-mail ID: lkeshavcz4@gmail.com

Details of the Organisation

Name and address of organisation: Coding Blocks, Pitampura near Kohat metro station
Broader Area: ……
Name of Instructor: VasuDev
Designation and Contact No.: 011-40752584

Student Performance Record

	No. of days Scheduled for the training	Number of days actually attended	Curriculum Scheduled for the student	Curriculum actually covered by the student
Week 1	5	5	Intro to ML, Probability & Stats, Numpy, Pandas	Intro to ML, Prob & Stat, Numpy, Panas
Week 2	5	5	KNN, Decision Trees, Naive Bayes	KNN, Decision Trees, Naive Bayes
Week 3	5	5	Randomforest, Feature Imp, Feature Scaling, SVD	Random forest, Feature Imp, Feature Scalin, SVD
Week 4	5	5	Regression Techniques, optimization, Neural Network	Regression Techniques, Optimization, Neural Netw
Week 5	5	5	CNN, Autoencoder, RNN, LSTM, Transfer Learning	CNN, Autoencoder, RNN, LSTM, Transfer Learning
Week 6	5	5	Capstone project	Capstone Project

[illegible]
(Signature of the student)

Any comments or suggestions for the student performance during the training program (to be filled by instructor)……

……

……

Coding Blocks Private Limited
New Delhi

[illegible]
(Signature of the Instructor)
along with Seal

Note : Every student has to fill and submit this Performa duly signed by his/her instructor to the faculty-in-charge by first week of September.

ACKNOWLEDGEMENT

The success and outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I respect and thank **Mr. Vasu Dev**, for providing me an opportunity to do the project work in Coding Blocks and giving us all support and guidance which made me complete the project duly. I owe my deep gratitude towards him, who took keen interest on our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

I express my gratitude to **Mr. Manoj Malik (**HOD, IT) for motivating and allowing me to complete the summer training in good schedule.

Also I would like to thank **Mrs. Tripti Rathi** (Proctor, Information Technology Engineering Evening) for her immense help regarding the project.

I would also thank all the Information Technology Department faculty members of Maharaja Surajmal Institute of Technology, C-4 Janakpuri for their critical advice and guidance without which this summer training would not have been possible.

Last but not least I place a deep sense of gratitude to my family members and my friends who have been constant source of inspiration during the training period.

LIST OF FIGURES

TABLE OF CONTENTS

ABSTRACT

Caption generation is a challenging artificial intelligence problem where a textual description must be generated for a given photograph.

It requires both methods from computer vision to understand the content of the image and a language model from the field of natural language processing to turn the understanding of the image into words in the right order. Recently, deep learning methods have achieved state-of-the-art results on examples of this problem.

Deep learning methods have demonstrated state-of-the-art results on caption generation problems. What is most impressive about these methods is a single end-to-end model can be defined to predict a caption, given a photo, instead of requiring sophisticated data preparation or a pipeline of specifically designed models.

Chapter One

Company Profile

1. ABOUT THE COMPANY

CODING BLOCKS
Code Your Way To Success

Fig. 1.1 Coding Blocks Logo

Coding Blocks was founded in 2014 with a mission to create skilled Software Engineers for our country and the world. We are here to bridge the gap between the quality of skills demanded by industry and the quality of skills imparted by conventional institutes. At Coding Blocks, we strive to increase student interest by providing hands on practical training on every concept taught in the classroom.

We at Coding Blocks strongly believe that with the right guidance and perfect determination, any student willing to learn programming can become a master of coding. All team members at Coding Blocks are aces of their respective field and the share the highest level of commitment towards quality teaching and student success and satisfaction. The present structure of computer education in colleges and universities is not aligned to the needs of the IT Industry. Students have no place to go and bridge this huge gap. It was this realization that prompted me to address this very real need of the student community. Coding Blocks was started as a learning centre where we teach fundamentals of programming to college students. The courses here are designed to help students with their curriculum and give them a real feel of the IT industry. In the last three years, we have helped more than 7200 students achieve their goal and made them the darling of the industry.

1.2 COMPANY VISION

Coding Blocks has a vibrant yet straightforward purpose – To Develop Better.

We breathe and live the philosophy that every client matters and the results are everything.

We aim to compete with established IT firms in the market by delivering beneficial strategies and successful business practices. Our belief is client satisfaction is the long-term way for our business to grow.

1.3 COMPANY VALUES

- Passion for technology
- Constructive self-criticism, self-improvement
- Pursuit of Excellence
- Discovering advanced ways to meet & exceed expectations
- Openness & respect
- Integrity & honesty

1.4 COURSES

- Summer Industrial Internship Program
- Winter Industrial Training Program
- Regular Training

1.5 CONTACT DETAILS

Address

1st Floor ,47,Nishant Kunj , Main Pitampura Road, New Delhi, Near Kohat Enclave Metro Station(Exit from gate #2) (Opposite Pillar 337)

Phone No.

+91-9251494002

+91-9251094002

+91-9599586446

011-47052584

011-27354126

Online Support

https://codingblocks.com

1.6 About Team

The backbone of any training institute is the team of talented and enthusiastic mentors, and the story is no different for Coding Blocks. We take pride in the fact that we have the most talented team of mentors, who have experienced the education and corporate industries with equal measures. All our mentors have been well performing students at the best of colleges, have worked at industry giants such as Sony, Cyanogen, SanDisk, Facebook, Barclays, Practo, Harman, DRDO, Cadence, American Express, Nagarro and many more. We also have GSoCers, GSoC Mentors and Google Code-in mentors in our team, in addition to people who've been published in international journals. With great reviews encouraging us, and the not so great ones making us push our limits, our team certainly has the right blend of people for every technology and language.

Figure 1.2

Manmohan Gupta (Munna Bhaiya), an IIT-Delhi graduate, is an ace programmer, technocrat, an entrepreneurial doyen and a mathematician. He has co-founded Software giant, **Nagarro** and **Vidyamandir Classes(VMC)**. Having seen both the industries from up close, he could identify a key gap between college education and industry needs and has now set out to carve the path that will turn even ordinary students into industry-ready coders.

Figure 1.3

Passionate about teaching, Prateek is a CS graduate from DTU. He has previously worked with SanDisk, HackerEarth. He has also won various hackathons including Google's Code For India, Smart City Hackathon, qualified ACM-ICPC regionals and published papers in International Journals. His interactive CV (www.prateeknarang.com) is also popular in 120+ countries.

CHAPTER 2

TECHNOLOGY USED

2.2 MACHINE LEARNING

2.2.1 INTRODUCTION TO MACHINE LEARNING

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers.

Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

In this tutorial, we‘ll look into the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbor algorithm, decision tree learning, and deep learning. We‘ll explore which programming languages are most used in machine learning, providing you with some of the positive and negative attributes of each. Additionally, we‘ll discuss biases that are perpetuated by machine learning algorithms, and consider what can be kept in mind to prevent these biases when building algorithms.

It is associated with a machine learning algorithm (Artificial Neural Network, ANN) which uses the concept of human brain to facilitate the modelling of arbitrary functions. ANN requires a vast amount of data and this algorithm is highly flexible when it comes to model multiple outputs simultaneously. ANN is more complex topic

When solving a problem using traditional machine learning algorithm, it is generally recommended to break the problem down into different parts, solve them individually and combine them to get the result. Deep learning in contrast advocates to solve the problem end-to-end.

The most important difference between deep learning and traditional machine learning is its performance as the scale of data increases. When the data is small, deep learning algorithms don‘t perform that well. This is because deep learning algorithms need a large amount of data to understand it perfectly. On the other hand, traditional machine learning algorithms with their handcrafted rules prevail in this scenario.

2.2.2 TYPES OF LEARNING

2.2.2.1 SUPERVISED LEARNING

It is also called predictive model. As the name suggests is used to predict the future outcome based on the historical data. Predictive models are normally given clear instructions right from the beginning as in what needs to be learnt and how it needs to be learnt. These class of learning algorithms are termed as Supervised Learning.

For example: Supervised Learning is used when a marketing company is trying to find out which customers are likely to churn. We can also use it to predict the likelihood of occurrence of perils like earthquakes, tornadoes etc. with an aim to determine the Total Insurance Value.

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

$Y = f(X)$

The goal is to approximate the mapping function so well that when you have new input data

that you can predict the output variables (Y) for that data.

It is called supervised learning because the process of algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers; the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

Supervised learning problems can be further grouped into regression and classification problems.

1. **Classification**: A classification problem is when the output variable is a category, such as —red‖ or —blue‖ or —disease‖ and —no disease‖.

2. **Regression**: A regression problem is when the output variable is a real value, such as

—dollars‖ or —weight‖.

Some common types of problems built on top of classification and regression include recommendation and time series prediction respectively.

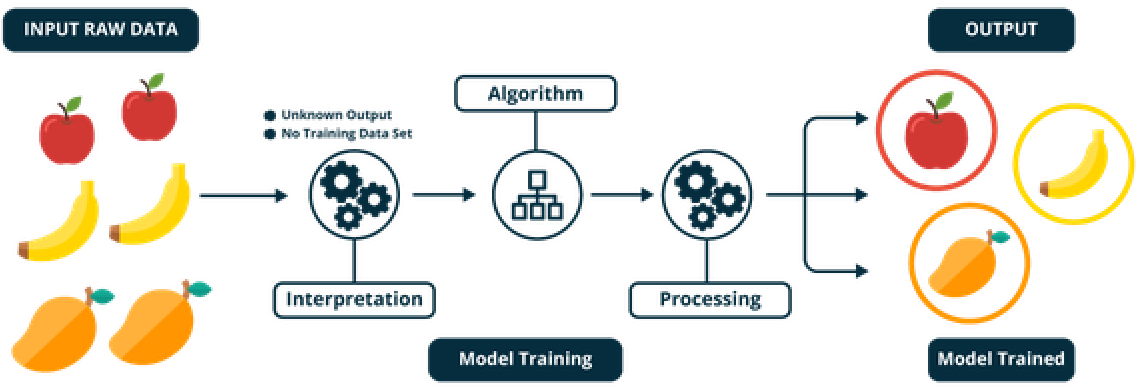


Figure 2.1

2.2.2.2 UNSUPERVISED LEARNING

Unsupervised learning is used to train descriptive models where no target is set and no single feature is important than the other. The case of unsupervised learning can be: When a retailer wishes to find out what are the combination of products, customers tends to buy more frequently. Furthermore, in pharmaceutical industry, unsupervised learning may be used to predict which diseases are likely to occur along with diabetes.

Unsupervised learning is where you only have input data (X) and no corresponding output variables.

The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data.

Unsupervised learning problems can be further grouped into clustering and association problems.

1. Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behaviour.

2. Association: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization.

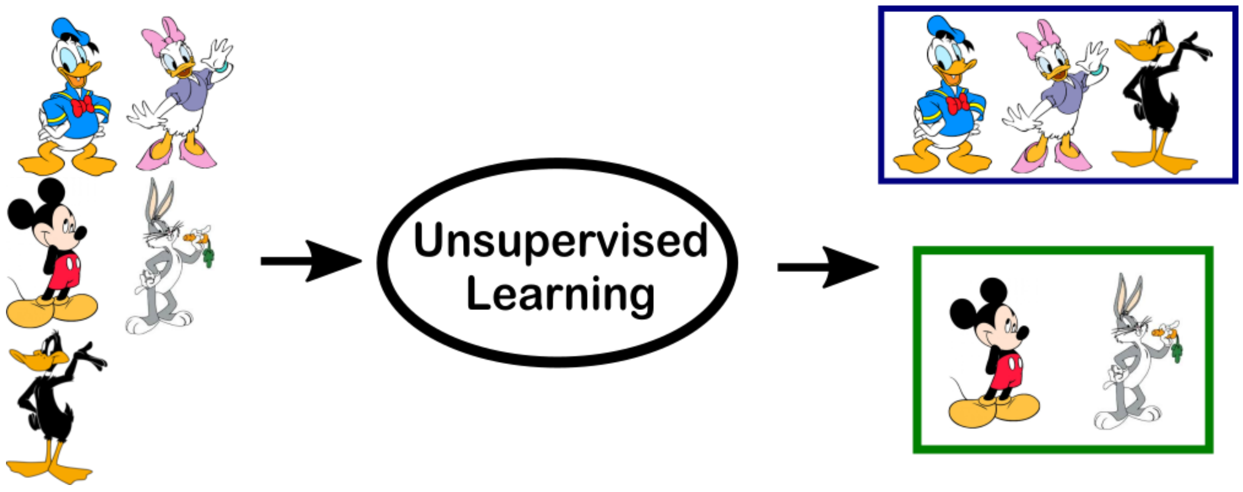


Figure 2.2

2.2.2.3 REINFORCEMENT LEARNING

It is an example of machine learning where the machine is trained to take specific decisions based on the business requirement with the sole motto to maximize efficiency (performance). The idea involved in reinforcement learning is: The machine/ software agent trains itself on a continual basis based on the environment it is exposed to, and applies its enriched knowledge to solve business problems. This continual learning process ensures less involvement of human expertise which in turn saves a lot of time! An example of algorithm used in RL is Markov Decision Process.

There is a subtle difference between Supervised Learning and Reinforcement Learning (RL). RL essentially involves learning by interacting with an environment. An RL agent learns from its past experience, rather from its continual trial and error learning process as against supervised learning where an external supervisor provides examples.

A good example to understand the difference is self-driving cars. Self-driving cars use Reinforcement learning to make decisions continuously – which route to take? What speed to drive on? These are some of the questions which are decided after interacting with the environment. A simple manifestation for supervised learning would be to predict fare from a cab going from one place to another

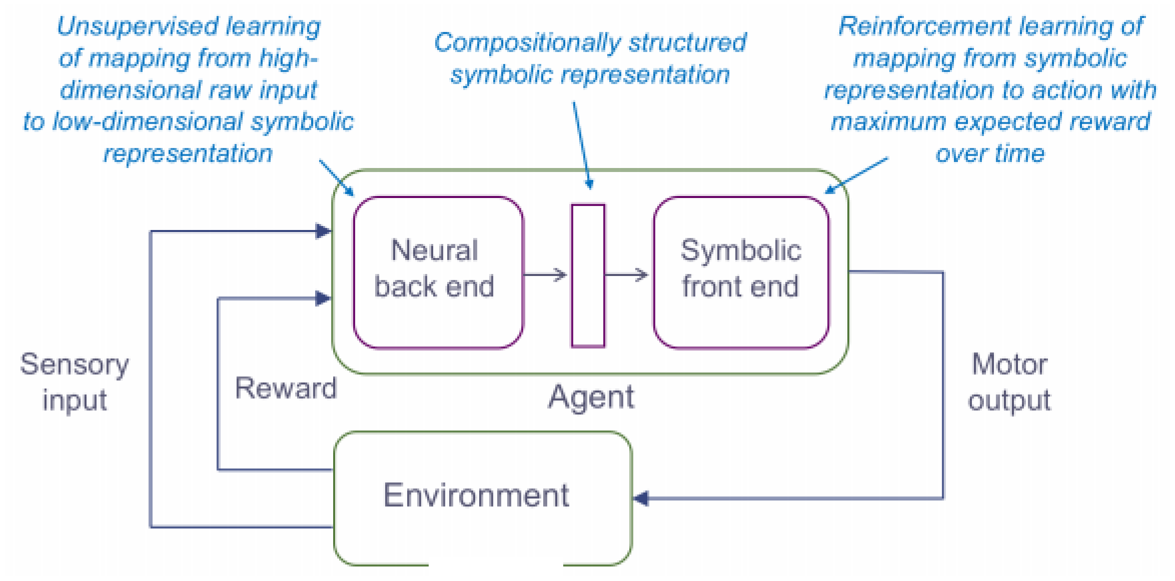

Unsupervised learning of mapping from high-dimensional raw input to low-dimensional symbolic representation
Compositionally structured symbolic representation
Reinforcement learning of mapping from symbolic representation to action with maximum expected reward over time
Neural back end
Symbolic front end
Sensory input
Reward
Agent
Motor output
Environment

Figure 2.3

2.2.3 APPLICATIONS OF MACHINE LEARNING

Applications for machine learning include:

1. Automated theorem proving
2. Adaptive websites
3. Affective computing
4. Bioinformatics
5. Brain–machine interfaces
6. Cheminformatics
7. Classifying DNA sequences
8. Computational anatomy
9. Computer vision, including object recognition
10. Detecting credit-card fraud
11. General game playing
12. Information retrieval
13. Internet fraud detection
14. Linguistics
15. Marketing

16. Machine learning control

17. Machine perception

18. Medical diagnosis

19. Economics

20. Insurance

21. Natural language processing

22. Natural language understanding

23. Optimization and metaheuristic

24. Online advertising

25. Recommender systems

26. Robot locomotion

27. Search engines

28. Sentiment analysis (or opinion mining)

29. Sequence mining

30. Software engineering

31. Speech and handwriting recognition

32. Financial market analysis

33. Structural health monitoring

34. Syntactic pattern recognition

35. Time series forecasting

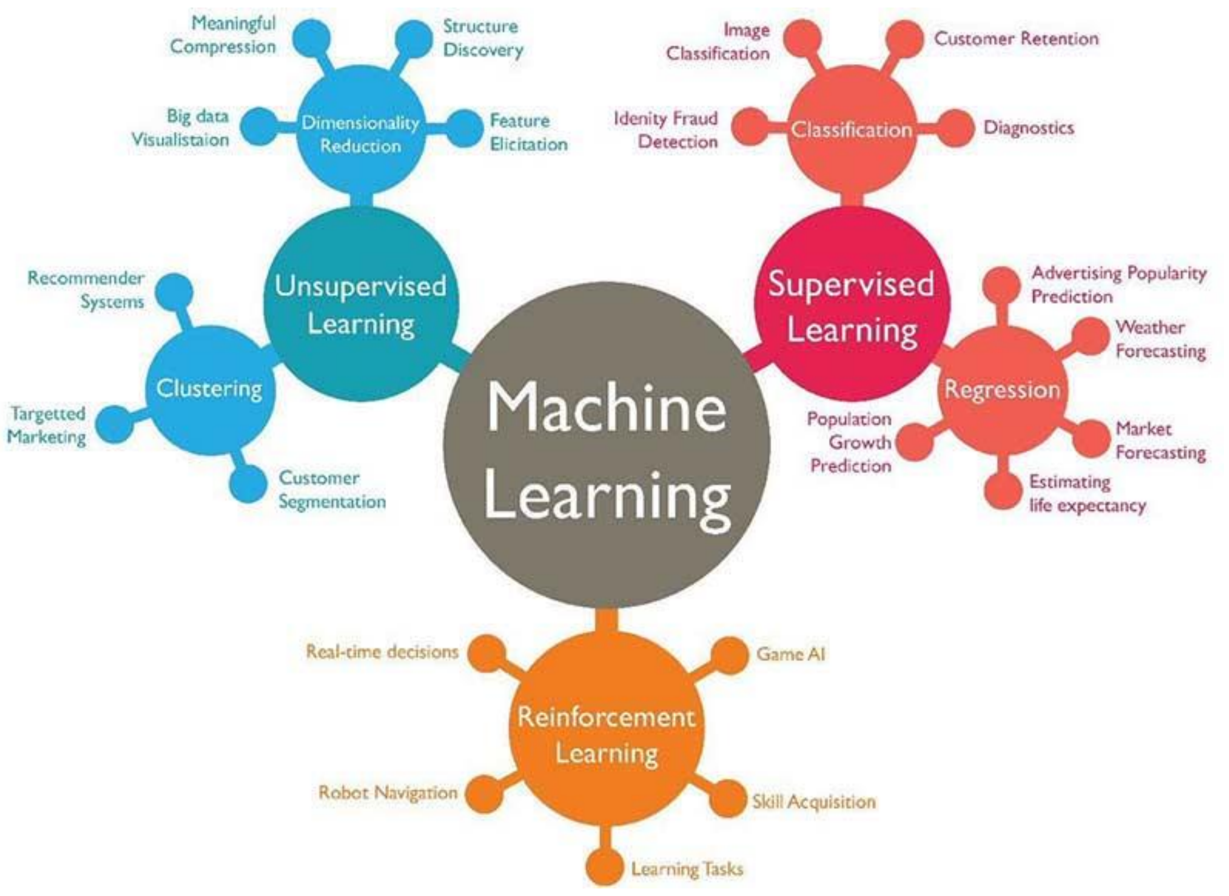


Figure 2.4

2.3 PYTHON FOR ML

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

1. Python is interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

2. Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter

directly to write your programs.

3. Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

4. Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

2.3.1 HISTORY

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include

1. Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

2. Easy-to-read: Python code is more clearly defined and visible to the eyes.

3. Easy-to-maintain: Python's source code is fairly easy-to-maintain.

4. A broad standard library: Python's bulk of the library is very portable and cross- platform compatible on UNIX, Windows, and Macintosh.

5. Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

6. Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

   Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

1. It supports functional and structured programming methods as well as OOP.

2. It can be used as a scripting language or can be compiled to byte-code for building large applications.

3. It provides very high-level dynamic data types and supports dynamic type checking.

4. It supports automatic garbage collection.

Figure 2.5

2.3.2 ANACONDA

Anaconda is the world‘s most popular Python data science platform. Anaconda, Inc. continues to lead open source projects like Anaconda, NumPy and SciPy that form the foundation of modern data science. Anaconda‘s flagship product, Anaconda Enterprise, allows organizations to secure, govern, scale and extend Anaconda to deliver actionable insights that drive businesses and industries forward.

Figure 2.6

Whether it's deploying into IT-managed infrastructure, or sharing code among a small team - getting the right packages into the right place is a common frustration among data scientists and IT staff. That is why we created conda, the leading package and environment manager for data science. Conda works with both R and Python packages, allowing you to easily manage and switch between separate environments built with different versions of R, Python, and their associated packages. Our open source enterprise-ready Python platform, Anaconda, includes conda plus over 330 of the most popular Python packages for science, math, engineering, and data analysis. Additionally, conda allows you to install over 300 R packages from Anaconda.org. Anaconda allows you to develop on Windows or Mac, and confidently deploy to Linux.

2.3.3 Keras

Figure 2.7

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that:

Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).

Supports both convolutional networks and recurrent networks, as well as combinations of the two.

Runs seamlessly on CPU and GPU.

Read the documentation at Keras.io.

Keras is compatible with: **Python 2.7-3.6**.

CHAPTER 3

THE PROJECT

3.1 OVERVIEW OF THE PROJECT

What do you see in the below picture?

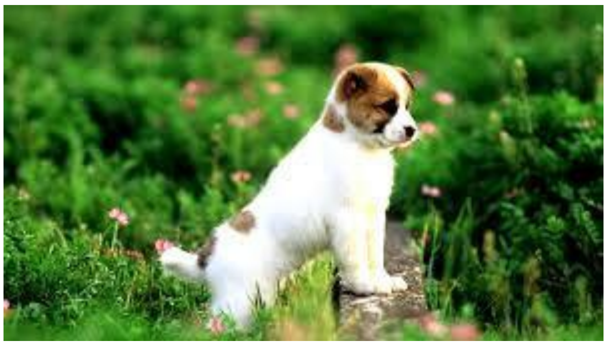


Figure 3.1

Can you write a caption?

Well some of you might say “**A white dog in a grassy area**”, some may say “**White dog with brown spots**” and yet some others might say “**A dog on grass and some pink flowers**”.

Definitely all of these captions are relevant for this image and there may be some others also. But the point I want to make is; it’s so easy for us, as human beings, to just have a glance at a picture and describe it in an appropriate language. Even a 5 year old could do this with ease. But, can you write a computer program that takes an image as input and produces a relevant caption as output?

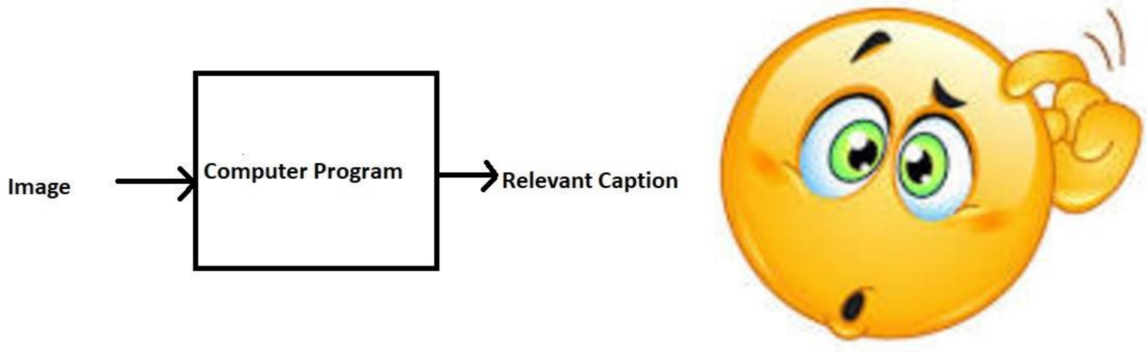


Figure 3.2

The Problem

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

This problem was well researched by Andrej in his PhD thesis at Stanford [1], who is also now the **Director of AI at Tesla.**

The purpose of this blog post is to explain (in as simple words as possible) that how Deep Learning can be used to solve this problem of generating a caption for a given image, hence the name **Image Captioning.**

To get a better feel of this problem, I strongly recommend to use this state-of-the-art system created by Microsoft called as Caption Bot. Just go to this link and try uploading any picture you want; this system will generate a caption for it.

3.2 PROBLEM STATEMENT

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

- Self driving cars — Automatic driving is one of the biggest challenges and if we can properly caption the scene around the car, it can give a boost to the self driving system.
- Aid to the blind — We can create a product for the blind which will guide them travelling on the roads without the support of anyone else. We can do this by first converting the scene into text and then the text to voice. Both are now famous applications of Deep

Learning. Refer this where its shown how Nvidia research is trying to create such a product.

- CCTV cameras are everywhere today, but along with viewing the world, if we can also generate relevant captions, then we can raise alarms as soon as there is some malicious activity going on somewhere. This could probably help reduce some crime and/or accidents.

- Automatic Captioning can help, make Google Image Search as good as Google Search, as then every image could be first converted into a caption and then search can be performed based on the caption.

3.3 Prerequisites

- Deep Learning
- Muli Layerd Perceptron
- Convolutional Neural Network
- Recurrent Neural Network
- Transfer Learning
- Backpropagation
- Overfitting
- Text Processing
- Python Syntax
- Data Structures
- Keras

3.4 Data Collection

There are many open source datasets available for this problem, like Flickr 8k (containing8k images), Flickr 30k (containing 30k images), MS COCO (containing 180k images), etc.

But for the purpose of this case study, I have used the Flickr 8k dataset which you can download by filling the form provided by the University of Illinois at Urbana-Champaign. Also training a model with large number of images may not be feasible on a system which is not a very high end PC/Laptop.

This dataset contains 8000 images each with 5 captions (as we have already seen in the Introduction section that an image can have multiple captions, all being relevant simultaneously).

These images are bifurcated as follows:

- Training Set — 6000 images
- Dev Set — 1000 images
- Test Set — 1000 images

3.5 SOFTWARE REQUIREMENT SPECIFICATION (SRS)

A **software requirements specification** (**SRS**) is a description of a software system to be developed. It is modeled after business requirements specification (CONOPS), also known as a stakeholder requirements specification (StRS).

The software requirements specification lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.Used appropriately, software requirements specifications can help prevent software project failure.

3.5.1 FUNCTIONAL REQUIREMENTS

In software engineering and systems engineering, a functional requirement defines a function of a system or its component, where a function is described as a specification of behaviour between outputs and inputs.

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish.

The System must provide following functionalities:

- User can easily update the database with the photo
- Caption should appear
- User can easily edit the settings
- How the system meets applicable regulatory requirements

3.5.2 NON FUNCTIONAL REQUIREMENTS

In systems engineering and requirements engineering, a **non-functional requirement** (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. They are contrasted with functional requirements that define specific behaviour or functions. The plan for implementing *functional* requirements is detailed in the system *design*. The plan for implementing *non-functional* requirements is detailed in the system *architecture*, because they are usually Architecturally Significant Requirements.

Following Non-Functional requirement should be there

- Secure access of confidential data
- 24*7 availability
- Better component design to get better performance at peak time
- Flexible service based architecture will be highly desirable for future extension. Various other Non-Functional requirements are:

1. Security
2. Reliability

3. Maintainability
4. Portability
5. Extensibility
6. Reusability

3.5.3 SYSTEM REQUIREMENT SPECIFICATION

The project was made and tests on:

- Operating System- Ubuntu
- IDE- Jupyter Notebook
- Text Editor- Jupyter Notebook

CHAPTER FOUR

SCREENSHOTS OF THE PROJECT

4.1 Data collection

First, I collected data from Kaggle.

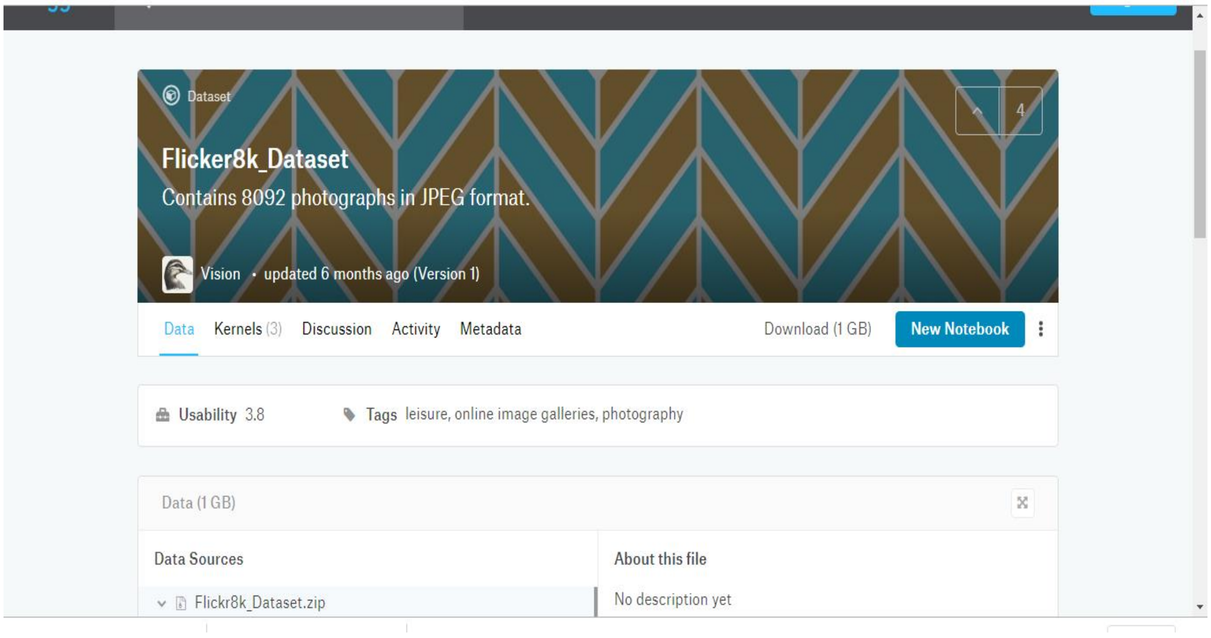
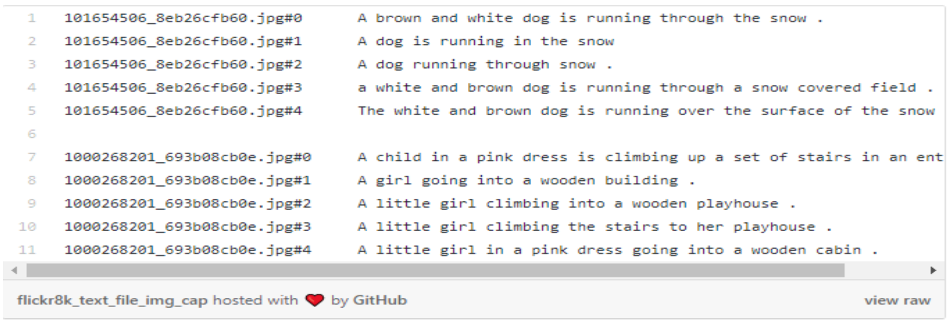


Figure 4.1

4.2 Understanding Data

One of the files is “Flickr8k.token.txt” which contains the name of each image along with its 5 captions. We can read this file as follows:



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

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

Figure4.2

Thus every line contains the <image name>#i <caption>, where 0≤i≤4

i.e. the name of the image, caption number (0 to 4) and the actual caption.

Now, we create a dictionary named “descriptions” which contains the name of the image (without the .jpg extension) as keys and a list of the 5 captions for the corresponding image as values.

```
descriptions['101654506_8eb26cfb60'] = ['A brown and white dog is
running through the snow .', 'A dog is running in the snow', 'A dog
running through snow .', 'a white and brown dog is running through a
snow covered field .', 'The white and brown dog is running over the
surface of the snow .']
```

Figure 4.3

4.3 Data Cleaning

When we deal with text, we generally perform some basic cleaning like lower-casing all the words (otherwise“hello” and “Hello” will be regarded as two separate words), removing special tokens (like ‘%’, ‘$’, ‘#’, etc.), eliminating words which contain numbers (like ‘hey199’, etc.).

We have 8763 unique words across all the 40000 image captions. We write all these captions along with their image names in a new file namely, “*descriptions.txt”* and save it on the disk.

However, if we think about it, many of these words will occur very few times, say 1, 2 or 3 times. Since we are creating a predictive model, we would not like to have all the words present in our vocabulary but the words which are more likely to occur or which are common. This helps the model become more **robust to outliers** and make less mistakes.

Hence we consider only those words which **occur at least 10 times** in the entire corpus. So now we have only 1651 unique words in our vocabulary. However, we will append 0’s (zero padding explained later) and thus total words = 1651+1 = **1652** (one index for the 0).

4.4 Data Preprocessing(Images)

Images are nothing but input (X) to our model. As you may already know that any input to a model must be given in the form of a vector.

We need to convert every image into a fixed sized vector which can then be fed as input to the neural network. For this purpose, we opt for **transfer learning** by using the InceptionV3 model (Convolutional Neural Network) created by Google Research.

This model was trained on Imagenet dataset to perform image classification on 1000 different classes of images. However, our purpose here is not to classify the image but just get fixed-length informative vector for each image. This process is called **automatic feature engineering.**

Hence, we just remove the last softmax layer from the model and extract a 2048 length vector (**bottleneck features**) for every image as follows:

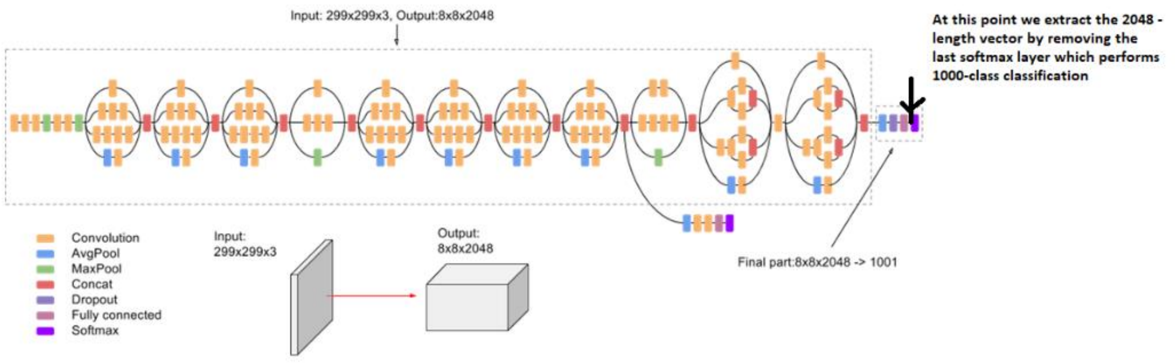

Input: 299x299x3, Output:8x8x2048
At this point we extract the 2048 - length vector by removing the last softmax layer which performs 1000-class classification
Convolution
AvgPool
MaxPool
Concat
Dropout
Fully connected
Softmax
Input: 299x299x3
Output: 8x8x2048
Final part:8x8x2048 -> 1001

Figure 4.4

We save all the bottleneck train features in a Python dictionary and save it on the disk using Pickle file, namely "**encoded_train_images.pkl**" whose keys are image names and values are corresponding 2048 length feature vector.

NOTE: This process might take an hour or two if you do not have a high end PC/laptop.

Similarly we encode all the test images and save them in the file "**encoded_test_images.pkl**".

4.5 Data Preprocessing(Captions)

We must note that captions are something that we want to predict. So during the training period, captions will be the target variables (Y) that the model is learning to predict.

But the prediction of the entire caption, given the image does not happen at once. We will predict the caption **word by word**. Thus, we need to encode each word into a fixed sized vector. However, this part will be seen later when we look at the model design, but for now we will create two Python Dictionaries namely “wordtoix” (pronounced — word to index) and “ixtoword” (pronounced — index to word).

Stating simply, we will represent every unique word in the vocabulary by an integer (index). As seen above, we have 1652 unique words in the corpus and thus each word will be represented by an integer index between 1 to 1652.

These two Python dictionaries can be used as follows:

wordtoix[‘abc’] -> returns index of the word ‘abc’

ixtoword[k] -> returns the word whose index is ‘k’

4.6 Data Preperation using generatorn function

This is one of the most important steps in this case study. Here we will understand how to prepare the data in a manner which will be convenient to be given as input to the deep learning model.

If we take two images, First we need to convert both the images to their corresponding 2048 length feature vector as discussed above. Let “**Image_1**” and “**Image_2**” be the feature vectors of the first two images respectively

Secondly, let’s build the vocabulary for the first two (train) captions by adding the two tokens “startseq” and “endseq” in both of them: (Assume we have already performed the basic cleaning steps)

Caption_1 -> “startseq the black cat sat on grass endseq”

Caption_2 -> “startseq the white cat is walking on road endseq”

vocab = {black, cat, endseq, grass, is, on, road, sat, startseq, the, walking, white}

Let’s give an index to each word in the vocabulary:

black -1, cat -2, endseq -3, grass -4, is -5, on -6, road -7, sat -8, startseq -9, the -10, walking -11, white -12

Now let’s try to frame it as a **supervised learning problem** where we have a set of data points D = {Xi, Yi}, where Xi is the feature vector of data point ‘i’ and Yi is the corresponding target variable.

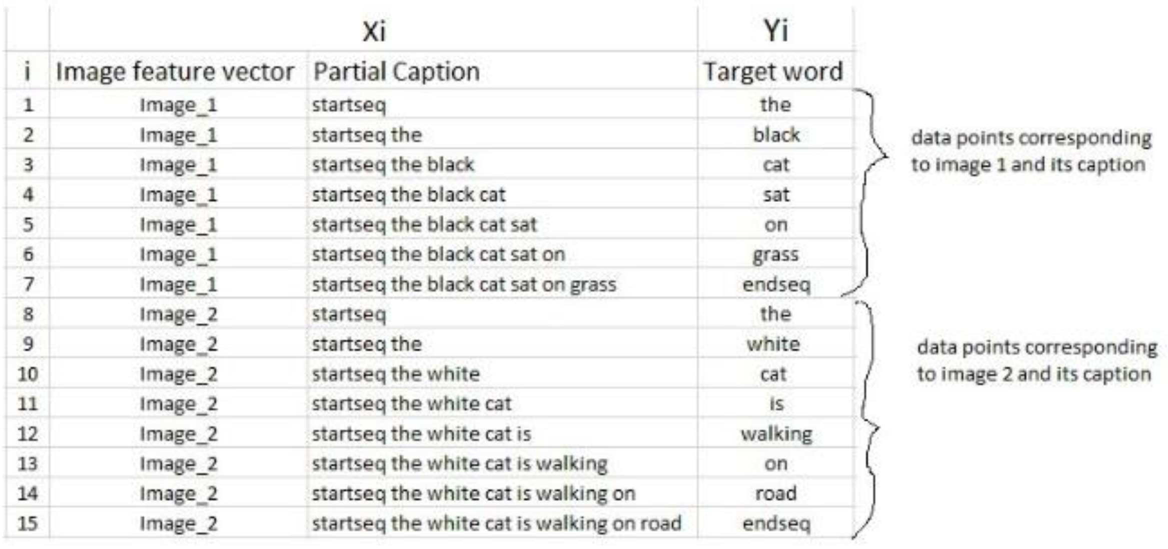
Let’s take the first image vector **Image_1** and its corresponding caption “**startseq the black cat sat on grass endseq**”. Recall that, Image vector is the input and the caption is what we need to predict. But the way we predict the caption is as follows:

For the first time, we provide the image vector and the first word as input and try to predict the second word, i.e.:

Input = Image_1 + ‘startseq’; Output = ‘the’

Then we provide image vector and the first two words as input and try to predict the third word, i.e.:

Input = Image_1 + ‘startseq the’; Output = ‘cat’



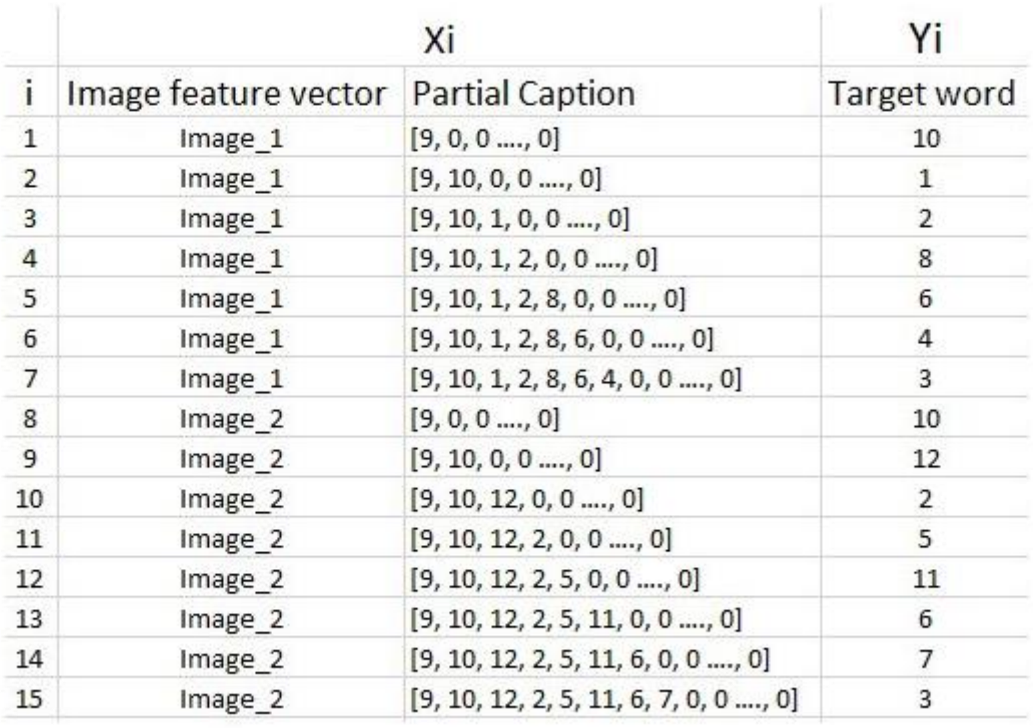
	Xi		Yi
i	Image feature vector	Partial Caption	Target word
1	Image_1	startseq	the
2	Image_1	startseq the	black
3	Image_1	startseq the black	cat
4	Image_1	startseq the black cat	sat
5	Image_1	startseq the black cat sat	on
6	Image_1	startseq the black cat sat on	grass
7	Image_1	startseq the black cat sat on grass	endseq
8	Image_2	startseq	the
9	Image_2	startseq the	white
10	Image_2	startseq the white	cat
11	Image_2	startseq the white cat	is
12	Image_2	startseq the white cat is	walking
13	Image_2	startseq the white cat is walking	on
14	Image_2	startseq the white cat is walking on	road
15	Image_2	startseq the white cat is walking on road	endseq

data points corresponding to image 1 and its caption
data points corresponding to image 2 and its caption

Data Matrix for both the images and captions

Figure 4.5

However, we have already discussed that we are not going to pass the actual English text of the caption, rather we are going to pass the sequence of indices where each index represents a unique word**.**



	Xi		Yi
i	Image feature vector	Partial Caption	Target word
1	Image_1	[9, 0, 0 …., 0]	10
2	Image_1	[9, 10, 0, 0 …., 0]	1
3	Image_1	[9, 10, 1, 0, 0 …., 0]	2
4	Image_1	[9, 10, 1, 2, 0, 0 …., 0]	8
5	Image_1	[9, 10, 1, 2, 8, 0, 0 …., 0]	6
6	Image_1	[9, 10, 1, 2, 8, 6, 0, 0 …., 0]	4
7	Image_1	[9, 10, 1, 2, 8, 6, 4, 0, 0 …., 0]	3
8	Image_2	[9, 0, 0 …., 0]	10
9	Image_2	[9, 10, 0, 0 …., 0]	12
10	Image_2	[9, 10, 12, 0, 0 …., 0]	2
11	Image_2	[9, 10, 12, 2, 0, 0 …., 0]	5
12	Image_2	[9, 10, 12, 2, 5, 0, 0 …., 0]	11
13	Image_2	[9, 10, 12, 2, 5, 11, 0, 0 …., 0]	6
14	Image_2	[9, 10, 12, 2, 5, 11, 6, 0, 0 …., 0]	7
15	Image_2	[9, 10, 12, 2, 5, 11, 6, 7, 0, 0 …., 0]	3

Figure 4.6

4.7 Word Embeddings

As already stated above, we will map the every word (index) to a 200-long vector and for this purpose, we will use a pre-trained GLOVE Model. Now, for all the 1652 unique words in our vocabulary, we create an embedding matrix which will be loaded into the model before training.

4.8 Model Architecture

Since the input consists of two parts, an image vector and a partial caption, we cannot use the Sequential API provided by the Keras library. For this reason, we use the Functional API which allows us to create Merge Models.

First, let's look at the brief architecture which contains the high level sub-modules:

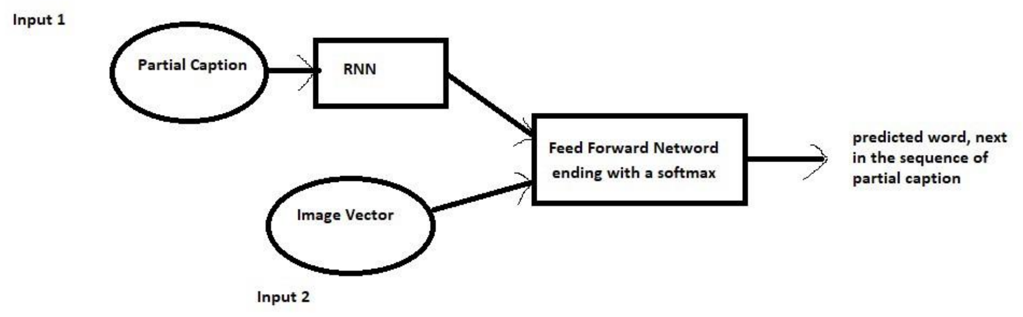
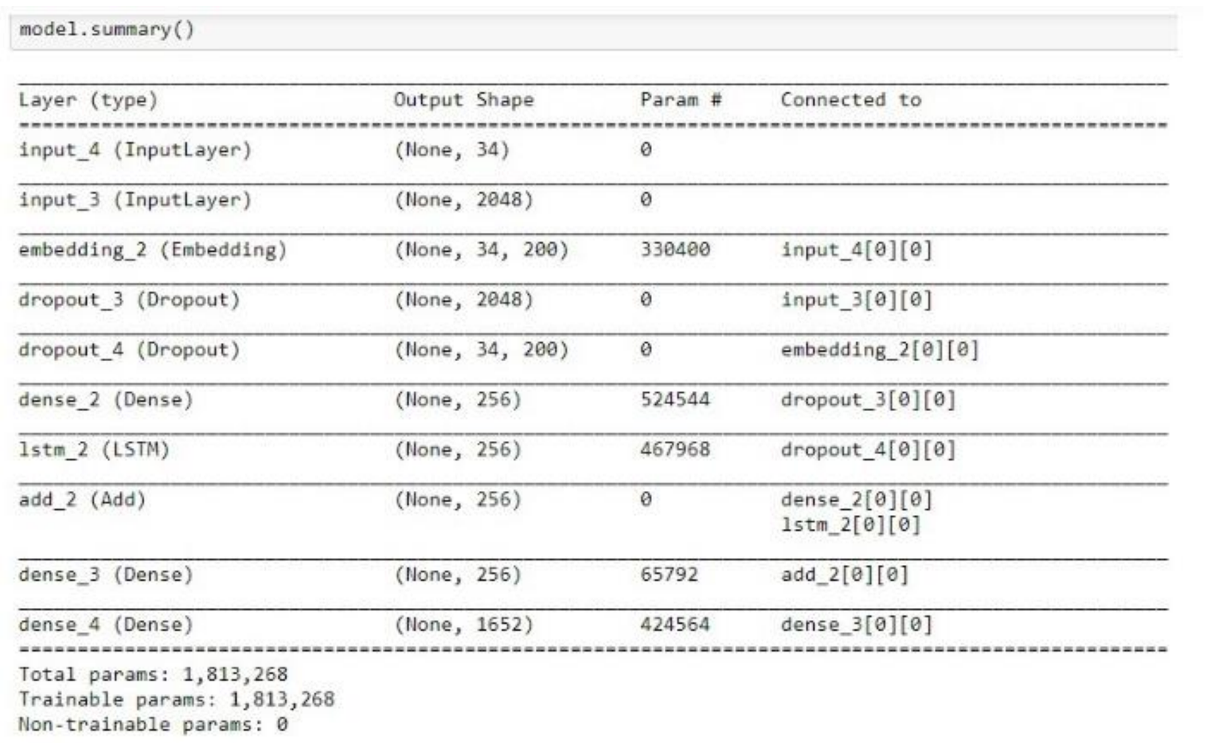

Input 1
Partial Caption
RNN
Feed Forward Netword ending with a softmax
predicted word, next in the sequence of partial caption
Image Vector
Input 2

Figure 4.7

Summary

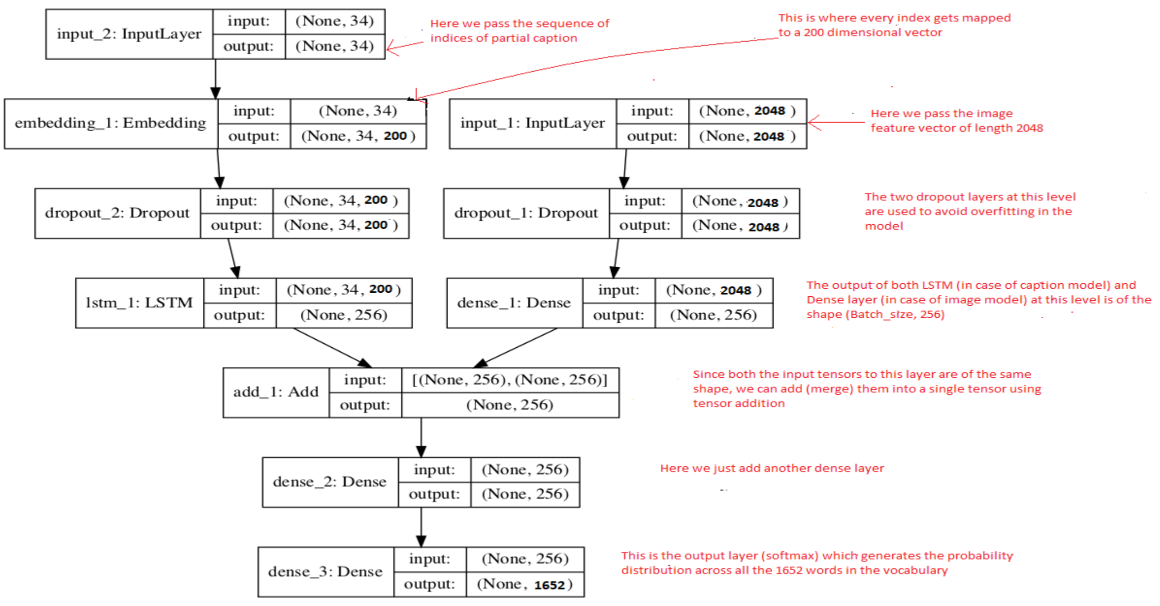


```
model.summary()
```

```
________________________________________________________________________________
Layer (type)                 Output Shape         Param #     Connected to
================================================================================
input_4 (InputLayer)         (None, 34)           0
________________________________________________________________________________
input_3 (InputLayer)         (None, 2048)         0
________________________________________________________________________________
embedding_2 (Embedding)      (None, 34, 200)      330400      input_4[0][0]
________________________________________________________________________________
dropout_3 (Dropout)          (None, 2048)         0           input_3[0][0]
________________________________________________________________________________
dropout_4 (Dropout)          (None, 34, 200)      0           embedding_2[0][0]
________________________________________________________________________________
dense_2 (Dense)              (None, 256)          524544      dropout_3[0][0]
________________________________________________________________________________
lstm_2 (LSTM)                (None, 256)          467968      dropout_4[0][0]
________________________________________________________________________________
add_2 (Add)                  (None, 256)          0           dense_2[0][0]
                                                              lstm_2[0][0]
________________________________________________________________________________
dense_3 (Dense)              (None, 256)          65792       add_2[0][0]
________________________________________________________________________________
dense_4 (Dense)              (None, 1652)         424564      dense_3[0][0]
================================================================================
Total params: 1,813,268
Trainable params: 1,813,268
Non-trainable params: 0
```

Figure 4.8

The below plot helps to visualize the structure of the network and better understand the two streams of input:



input_2: InputLayer | input: (None, 34) | output: (None, 34)
Here we pass the sequence of indices of partial caption
This is where every index gets mapped to a 200 dimensional vector
embedding_1: Embedding | input: (None, 34) | output: (None, 34, 200)
input_1: InputLayer | input: (None, 2048) | output: (None, 2048)
Here we pass the image feature vector of length 2048
dropout_2: Dropout | input: (None, 34, 200) | output: (None, 34, 200)
dropout_1: Dropout | input: (None, 2048) | output: (None, 2048)
The two dropout layers at this level are used to avoid overfitting in the model
lstm_1: LSTM | input: (None, 34, 200) | output: (None, 256)
dense_1: Dense | input: (None, 2048) | output: (None, 256)
The output of both LSTM (in case of caption model) and Dense layer (in case of image model) at this level is of the shape (Batch_size, 256)
add_1: Add | input: [(None, 256), (None, 256)] | output: (None, 256)
Since both the input tensors to this layer are of the same shape, we can add (merge) them into a single tensor using tensor addition
dense_2: Dense | input: (None, 256) | output: (None, 256)
Here we just add another dense layer
dense_3: Dense | input: (None, 256) | output: (None, 1652)
This is the output layer (softmax) which generates the probability distribution across all the 1652 words in the vocabulary

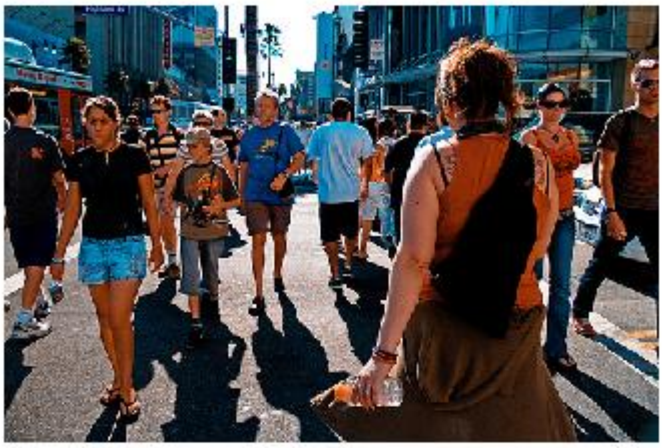
Figure 4.9

CHAPTER FIVE

CONCLUSION

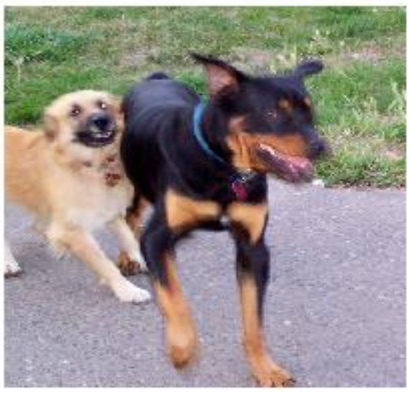
5.1 Final Results

At the end of our summer training ,we have made a prototype of a Image captioning system with a aim to help people . Some of the great results are:



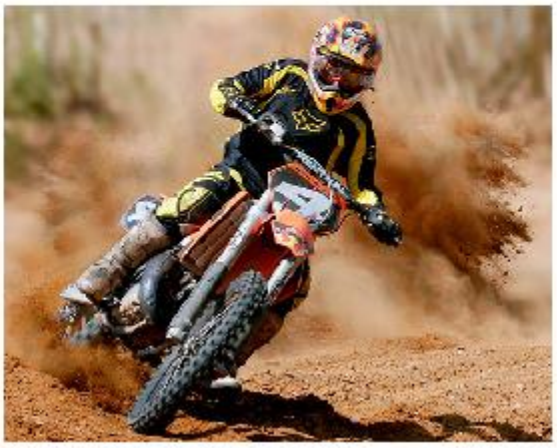
```
many people are standing around crowded street
```

Figure 5.1



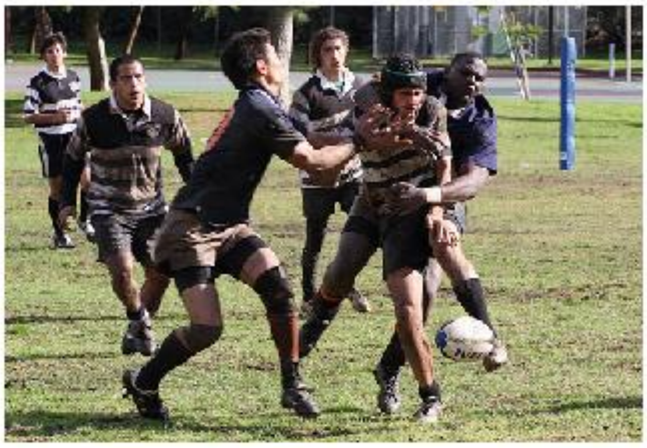
```
two dogs are playing with each other in the grass
```

Figure 5.2



man on motorcycle is driving on dirt track

Figure 5.3



two men in sports uniforms are playing soccer

Figure 5.4

BIBLIOGRAPHY

- https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-using-deep-learning/
- https://www.appliedaicourse.com
- https://www.kaggle.com/ming666/flicker8k-dataset
- https://stackoverflow.com/
- https://towardsdatascience.com/image-captioning-with-keras-teaching-computers-to-describe-pictures-c88a46a311b8
- https://online.codingblocks.com/app/courses