Spring 2018

Deep Learning Algorithm Recommender

Rajat Kabra
San Jose State University

Writing Project

Deep Learning Algorithm Recommender

Final Report

Author
Rajat Kabra
CS 298
May 2018

Advisor
Dr. Chris Tseng

A Writing Project Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment of the Requirements for the Degree: Master of Science

The Designated Committee Approves the Master's Project Titled

Deep Learning Algorithm Recommender

By

Rajat Kabra

Approved for the Department of Computer Science
San José State University
May 2018

Acknowledgements

I would like to express my gratitude to my project advisor, Dr. Chris Tseng, for his guidance, support, encouragement and time. I would also like to thank my committee members, Dr. Thomas Austin, and Dr. Jon Pearce, for their valuable input and suggestions during the course of the completion of my master project. I would also thank the faculty of Department of Computer Science for the knowledge they are imparting to the students and for the staff for their efforts in maintain the classes and helping students.

ABSTRACT

Deep learning contains a set of algorithms that are based on the functioning of human brain i.e. neural networks. These algorithms require a lot of computation power and time along with complex setup to get good results. The project contains several artificial neural network implementation for a variety of tasks like data classification, image classification, natural language processing and more. The project contains an exploratory analysis of hyperparameters of deep learning algorithms in domain of deep learning applications to prove that it is possible to achieve a good accuracy with less resources.

Table of Contents

Tables of Figures

Table of Tables

1 Project Description

1.1 Introduction

Machine learning is the science of getting computers to act without being explicitly programmed. [1] It is the practice of parsing data, learning from it and then making decisions based on what the algorithm has learned from the data. In all the machine learning algorithms, artificial neural networks have formed their own place and are categorized as deep learning, where deep is the number of layers in the artificial neural network.

Deep Learning is generally used to do a task for which standard machine learning is not an optimal choice like problems associated with computer vision, speech recognition, artificial intelligence and many more. In this project, we will be using multiple artificial neural networks like Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Deep Neural Network (DNN/NN) and Long Short Term Memory (LSTM) and will analyze their performance on several applications. These algorithms will be used in the project for data science problem like classification, image classification, object detection, and natural language processing. Each of the application that will be covered uses a different artificial neural network and multiple layers and activation functions.

Every artificial neural network is designed for a particular type of data problem. Deep learning models have a higher level of complexity associated with them when compared to standard machine learning. The number of parameters the model has and the ways in which a model can be created for the same problem far exceeds the complexity of standard machine learning problems. In the project, we will discuss these models, their architecture, hyperparameters, performance, implementation, applications and drawbacks.

1.2 Literature Review

The applications of deep learning are researched extensively and are a big part of literature. The technology is applied to several problems and new use cases are developed on a regular basis. Each type of artificial neural network has been researched extensively and has a very different set of properties when compared to other. These neural networks are made to complement each other for complicated tasks as every type of neural network has a different set of properties based on the task it could be used for.

1.2.1 Artificial Neural Networks

The motive behind creating artificial neural network was to create learning models that would imitate the working of a human brain [3]. These models are capable of learning any mapping function like a human brain. The building block of an artificial neural network, just like a biological neural network, is neuron. Neurons are computational units with weighted input signals and an activation function associated with it which produces an output signal. Each neuron has a bias which is an input that starts with value 1 and is weighted. If a neuron has four input, then it will have five weights, four for input and one for bias [4].

The initial weights for the model could be simple random values or they can be computed using complex techniques. For activation of neurons, the inputs are summed and passed through the neuron's activation function. An activation function is often termed as a transfer function because it starts the transfer of data from one neuron to the next. This function is a simple mapping of weighted input to the output. The reason it is called an activation function is that a neuron is activated only when the input reaches a particular threshold value [5].

Components of an artificial neural network:

- Neurons: A neural network is made up of neurons, which is the basic unit of any neural network. Every neuron takes in input and produces an output based on the input and weight associated with those inputs. If the output is greater than threshold, the neuron gets activated. In theory, it is a mathematical function [5].

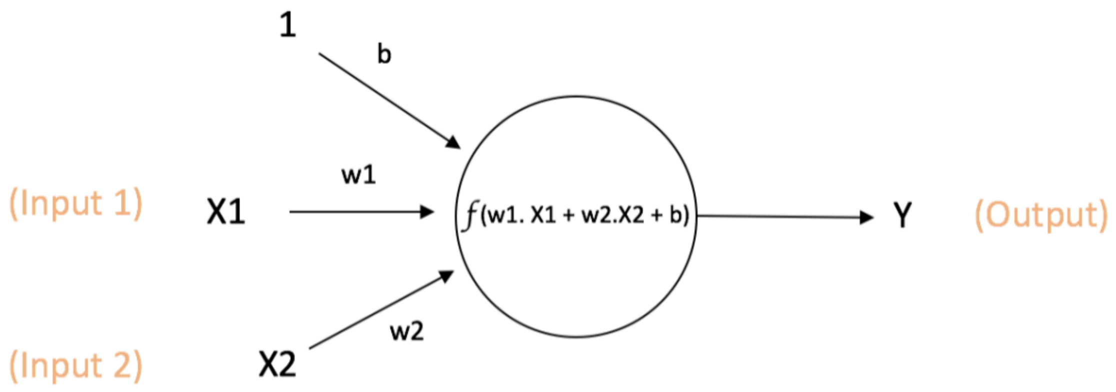


Fig. 1 Neuron Architecture

- Connection and Weights: The links between neurons that are used to send data are called connections and every connection has a weight assigned to it, which comes with the data on that connection [24]. The weight gives the input the importance it needs to make an impact. Some inputs are given more weight than others because they might be more relevant to the end result.

- Summation Function: The inputs and weights are vectors that are input to a neuron and the total input is the dot product of these two vectors which ends up in a single number. The summation function could be more complex than a simple dot product [25]. The exact summation function depends on the neural network and the model.
- Learning Rule: Every connection has a weight associated with it which has to change based on the input and what model has learned. If the weights are never changed, then the model will never improve and will be stuck at the same level. The functions to change weights are utilized during backward propagation in the neural network. These functions are also called as adaptation functions.. There are two types of learning rule, supervised learning and unsupervised learning. In supervised learning, the neural network is trained with a labeled dataset and the performance is measured by checking the output from neural network with the original result. Based on the performance the model is tweaked and tested again. The process is repeated to reduce the error. In supervised learning, the model must be trained before it can be useful. The training data needs to be fairly large and also needs to contain all necessary information to make sure that all important features are learned. Unsupervised learning, on the other hand, is limited to academic research and does not have a widespread use right now.

Layers

When neurons are arranged into a network, a single row of neurons in that network are termed as a layer. A neural network can be divided into three layers [15].

- Input Layer: This is the first layer in the neural network which takes raw input. These inputs are normalized in a given range using normalization techniques for better working of the neural network. Since all the operations in the neural network are mathematical, normalization of data is very important before it is fed to the input layer.
- Hidden Layer: This layer takes the output of the input layer as input. The neurons present in this layer extract information and relevant features from the input data. There can be multiple hidden layers in a neural network and all the computation is done in these layers. They cannot be accessed from the outside world, hence the name is hidden layer. A

neural network must always have an input layer and an output layer, but this layer can be removed from a simple neural network [4].

- Output Layer: This is the last layer in a neural network which takes the computation from the hidden layer and produces an output based on that data. The choice of activation function matters a lot in the output layer as it decides whether the output is for a regression problem or for a classification problem [17].

Activation Function

In the above explanation, there is a constant use of the term activation functions which are present in the neurons and are responsible for their firing up and computation. The role of activation function is to perform computation on the input based on the weights, impose a bias and decide whether the result has reached the threshold for activating the neuron or not. For example, in a neuron

$$Y = \sum (weight * input) + bias$$

Here, the value of Y could range from negative infinity to positive infinity. Now, neuron does not know what value it should be activated for. To make this decision, activation functions are added to the neural network. There are several types of activation functions [6][7].

- Step Function: This is the simplest form of activation function which uses a binary 0 or 1 decision based on the value of threshold and value of neuron. If the value of neuron is greater than the threshold, the neuron is activated [25].

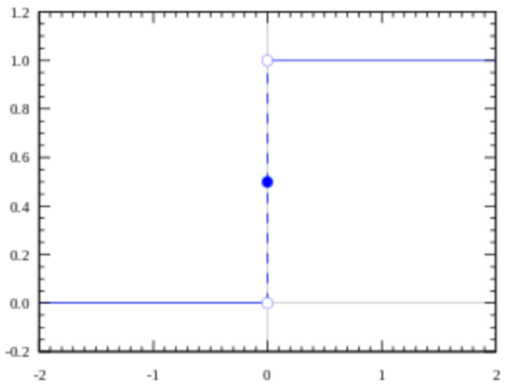


Fig. 2 Step Function

The output is 1 or true for activation and the output is 0 or false when the value is smaller than the threshold. The biggest drawback of this activation function is that it works great for a binary classification but fails when it comes to multi-class classification. In binary classification, one class will be 0 and the other one will be 1 but in multi-class classification, several class might get activated. To avoid this, we need an activation function that would give out an analog value for the activation rather than binary value.

- Linear Function: This is the best option to solve the drawbacks of the step function. The linear function is a straight line function where activation is proportional to the input. So, if more than one neuron gets activated, the final neuron can be decided based on the one with the highest value. But there is a problem with this method. If all the layers have a linear activation function, the output of one layer becomes input for the next layer and so on till the final layer. This way the activation function at the final layer will be nothing but the linear function output of the first layer. This means that we are losing the ability to stack up layers in the neural network [8].
- Sigmoid Function: Sigmoid function is a non-linear function which means that combining multiple sigmoid functions will yield a non-linear function in the end. The function gives analog output too, so solves the problem of step functions.

$$A = \frac{1}{1+e^{-x}}$$

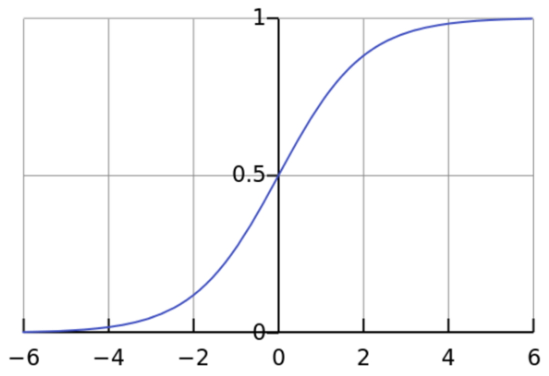


Fig. 3 Sigmoid Function

The biggest advantage of the sigmoid function is that it can make a clear differentiation on the prediction. Being a non-linear activation function [18], the output for this is always going to be in the range of 0 to 1, which is a big improvement over negative infinity to positive infinity for the linear function. This is great because it will never result in a crash in the activation function. One drawback of this activation function is that the model slows down the learning after a certain point as shown in Fig. 3 for -6 on x-axis.

- Tanh Function: This activation function is very similar to the sigmoid activation function.

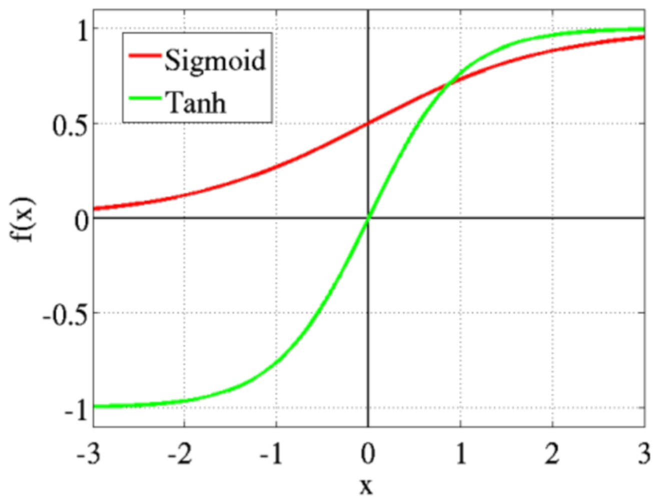


Fig. 4 Tanh Function

It can be said that is a scaled sigmoid function.

tanh(x) = 2 sigmoid (2x) - 1

Instead of being limited to 0 to 1 range, like the sigmoid activation function, it has a range from -1 to 1. One difference in tanh and sigmoid is that gradient is stronger in the former but it still suffers from vanishing gradient problem [25].

- ReLu (Rectified Linear Unit): It is a very simple activation function with formula F(x) = max(0,x). So the function will return x if the value of x is greater than 0, else it will return 0. ReLu is a non-linear function so stacking of layers is possible but ReLu does not have any bound on the positive side so the value could be positive infinity which can blow up. ReLu is not computationally as expensive as sigmoid or tanh. In sigmoid and tanh, the activation functions are all analog so almost every neuron will fire up which becomes computationally expensive. In ReLu, the activations are sparse which puts less load on the neural network.

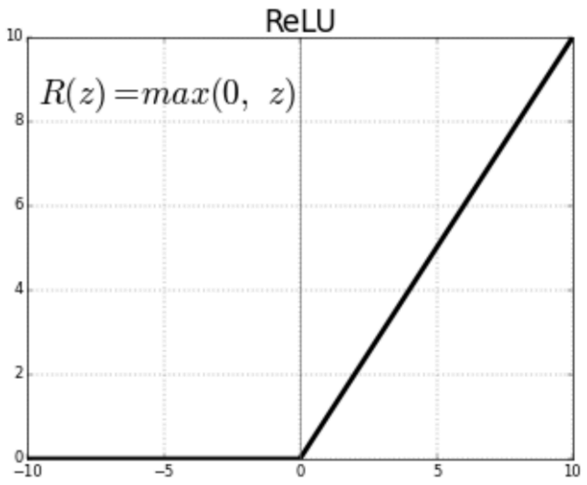


Fig. 5 ReLu Function

Relu suffers from a major problem in which the gradient can become 0. For activations in negative region, the gradient will be 0 so those neurons will stop learning as the error is also 0.

Types of neural network:

Multi-Layer Feed Forward Neural Network

This artificial neural network is often referred to as neural network and it is one of the most commonly used basic neural networks. It is a feed-forward neural network in which signal only goes in one direction, that is from the input layer to the output layer. Once a signal passes through a layer, it will never come back to that layer and all the neurons from one layer are connected to all the neurons at the next layer. Being a feed-forward neural network, deep neural networks cannot be used for a variety of complex applications which require knowledge of what happened at an earlier layer or if some data needs to be transferred to a previous layer. This model of the neural network is also sometimes referred to as Multi-Layer Perceptron (MLP) where perceptron is a single layer neural network.

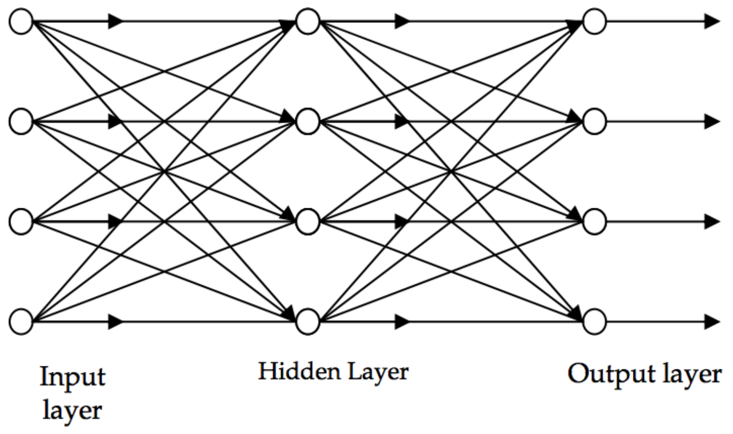


Fig. 6 Multi-Layered Feed Forward Neural Network

These neural networks have at least three layers out of which one layer is the input layer, one is the output layer and rest are all hidden layer. Hidden layers are responsible for dealing with the raw input from the input layer, perform computations on the data to extract high-feature statistics and transfer the data to the output layer. Each layer in these models learns to extract a different feature from the data, just like the human brain does [11].

The objective of creating a neural network is to make it learn the same way a human brain learns. Learning occurs in this neural network by altering the connection weights after every iteration of data processing. The weight is calculated based on the error that is in the output when compared to the actual result. This learning is achieved using backpropagation, a mean squares algorithm. The error in an output node *j* in the *n*th data point is represented by

$$e_j(n) = d_j(n) - y_j(n)$$

Where *d* is the target value and *j* is the value computed by the model. Based on these errors, new weights are calculated. The formula for this weight is given by

$$\mathcal{E}(n) = \frac{1}{2}\sum_j e_j^2(n)$$

The change in the weight can be calculated using gradient descent and is denoted by

$$\Delta w_{ji}(n) = -\eta \frac{\partial \mathcal{E}(n)}{\partial v_j(n)} y_i(n)$$

The weights are recomputed till the desired result is achieved in the neural network.

Recurrent Neural Network

The biggest drawback of feed-forward neural network is that once a signal passes a layer, it can never come back to it and the layer will never remember it. The original aim of creating artificial neural network was to imitate the working of the human brain, but human brain is capable of sending data back to a previous layer. So, the feed-forward neural network is not a correct imitation of the human brain. In a recurrent neural network, the connections between units form a directed circle and each neuron has its own internal memory which they use to process random sequence of input [12]. They are called recurrent because the same operation could be performed multiple times for a sequence of input. The memory captures the information about what has been calculated so far.

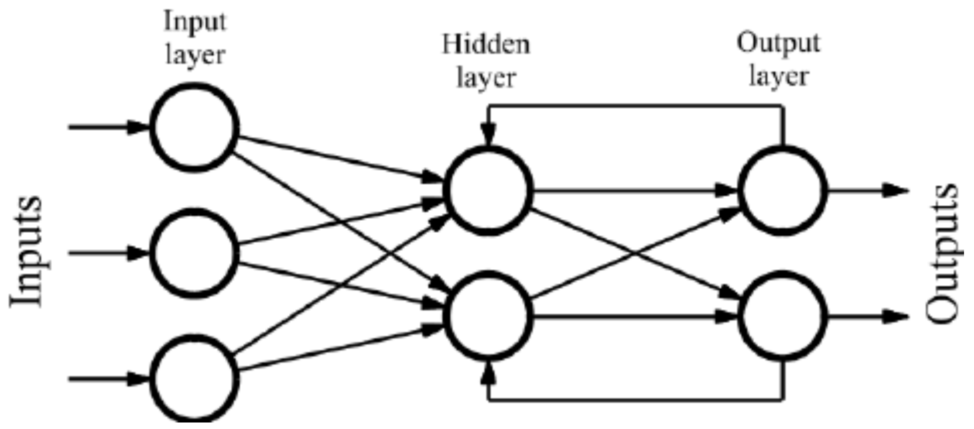


Fig. 7 Recurrent Neural Network

Recurrent neural network shows great accuracy and success in natural language processing. In the application domain of language modelling and text generation, we want to predict the probability of a word based on its previous predicted word. RNN is used for machine translation, speech recognition mostly. A very important concept in RNN is Backpropagation Through Time (BPTT). In a neural network, the flow of data is usually from input to output which is forward-propagation and it is checked if the output is correct or not. If the output is not correct, backpropagation is done, in which we go backwards in the network to find the partial error with respect to weights [13] [14].

Convolutional Neural Network

This class of neural network is used for visual analysis. CNN is a feed-forward neural network and uses a variety of layers. The reason why CNN is used for visual data is that it requires least preprocessing. Regular feed-forward neural networks cannot be used for visual data because they do not scale well. For example, for a 32x32x3 images with length and width as 32 and 3 as color channels, the total number of weight required in one neuron in the hidden layer will be 32*32*3 = 3072 weights. Now if the image is of size 200x200x3, then the total weights will be 120,000. This kind of input in one neuron is not easily manageable and can lead to network failure. To resolve this scaling issue, in CNN the neurons are placed in 3 dimensions.

CNN has different layers than a normal neural network or a recurrent neural network. In CNN, one layer transforms one activation volume to another using a differentiable function. Some of the following layers are:

- Input Layer: This layer takes in the raw pixel values of the image. So in case of the CIFAR10 dataset, the length is 32, width is 32 and three colors red, green, and blue.
- Convolution Layer: These layers are made of filters and feature maps. Filters are nothing but neurons which are present in this layer and have input weights and gives a value as output. These filters always have fixed size inputs. If the convolutional layer is the first layer in the model and has to consume the input, it will take the raw data but if it is a hidden layer, then it will only take feature map as an input. Feature map is the output of another filter from the previous layer.

  This layer does most of the heavy work in CNN. The parameters at this layer have learnable filters and every filter is small spatially but covers the entire depth of the input volume. So in case of CIFAR10 dataset, we assume that we have the first convolution layer of size 5x5x3, so this layer will convolve across the entire input volume and will calculate the dot product. As the window slides over the input, a 2D activation map is generated.

  A big part of the convolutional layer is parameter sharing to control the number of parameters. For example, if there are 290,400 neurons and each neuron has 364 weights including one bias. Together, this yields 105,705,600 parameters on the first layer which is too much. These parameters could be reduced by making an assumption that if one feature is useful for computation at (x,y), then it would also be useful at (x_2, y_2). By making this assumption, it is possible to use same set of weights and bias in a 2-dimensional slice of depth. This would reduce the number of parameters from 105,705,600 to 34,944 [15].

- ReLu Layer: This layer applies the activation function on the data and follows ReLu activation properties. This layer leaves the size of the input data volume unchanged.
- Pooling Layer: CNN highly depends on the data compression as the features obtained from images are huge. This compression of features is done by the pooling layer as they downsample the feature map obtained from previous layers. Pooling layers are not used after every convolutional layer but after a sequence of layers so that the data from these layers could be compressed down. The purpose is also to generalize the feature. So after this layer, a [32x32x12] input might become [16x16x12].
- Fully-connected Layer: These are simple neural network layer which resembles the layers present in any feed-forward neural network [16]. These layers might have a non-linear activation function to generate the output probabilities of predictions. They are used at the end after all the features are extracted and consolidated at the previous layers. The activation function at this layer is generally sigmoid it this layer is placed as the output layer, else it could be any. Thus, this layer computes the class score and reduces the volume like [1x1x10] where each 10 numbers will correspond to class score.

Mostly the layer stacking starts with multiple convolutional and relu layers, followed by pooling layer to compress their output. This pattern of layers is repeated till the images are reduced spatially to smaller size. It is also possible to transition to fully-connected layer in between these layers. The last fully-connected layer holds the output, which could be a score of various classes in the data [19].

Convolutional neural network has formed a base of several popular modern applications like self-driving cars, autonomous bots, facial recognition software and a lot more. CNN learns using small squares of input data over the input. By dividing the data into smaller chunks, the model uses less weights when compared to its other counterparts like deep neural network or artificial neural network.

Long Short Term Memory Network

LSTM neural network is a special branch of recurrent neural network which is capable of learning long-term dependencies in sequence based data. The capacity of RNN in predicting long-term dependencies is small. For example, RNN could be used to predict the last word in "water in the ocean is blue" because further context is not required and the sequence is small.

But if RNN is used to predict the last word of “I was in Spain for ten years, I speak fluent Spanish”, it will not be able to this. RNN can figure out that the next word will be a language, but it will not recognize which language as the gap between relevant information and the point where it is needed is large. This is also known as the vanishing gradient problem.
To solve this problem, LSTM networks are used. They are designed to learn long-term data dependencies and avoid the problems present in RNN. LSTM reduces the problem of gradient descent by using an internal memory state which is added to the processed input, rather than multiplying it. The architecture of LSTM cell is shown in Fig. 9.

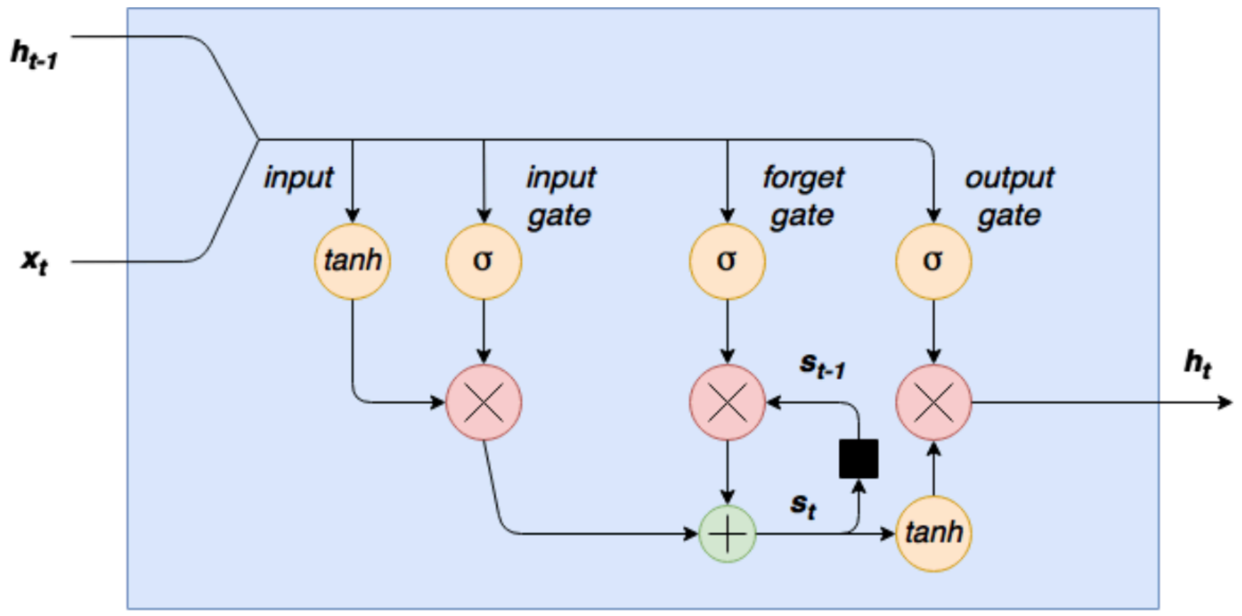


Fig. 8 Long Short Term Memory Network [11]

In Fig. 9, the flow of data is from left to right with the input $\boldsymbol{x_t}$ and the output of the previous cell as $\boldsymbol{h_{t-1}}$ concatenated and entering the top data path. The data then passes through gates which is what makes LSTM different from RNN [20].

- Input Gate: As discussed before, tanh activation function yields an output in the range (-1,1). The input gate is a hidden layer with sigmoid activation function with weighted input values. The output of input gate is a value between 0 and 1 which is used to determines which inputs are switched on and off.
- Forget Gate: The forget gate is a sigmoid activated set of nodes which is multiplied by internal state of the network at time t-1. This element-wise multiplication is used to decide which previous states to forget and which to remember.

- Output Gate: The output gate performs two functions, first two squash the input using a tanh function and an output sigmoid function. The output sigmoid activation function is multiplied with current state of network to determine the output of the cell for this state.

1.3 Problem Statement

Deep learning applications and algorithms require lot of computational power and time for training. Without providing these two things, it is really difficult to utilize their full potential. All major computer vision and natural language processing applications are completed by either big companies or research foundation with grants. These kind of resources are not available to a student or a freelancer looking to make a career in deep learning. The implementations given in papers and research work is not optimized to be executed on a normal laptop or a low power cloud platform.

1.4 Project Goal

The project goal is to perform exploratory analysis of deep learning algorithms by tuning the hyperparameters to reduce the required computation power and training time. Through this project, we will have a better understanding about the layers and activation functions in a neural network along with different types of neural network. The project will also involve stacking layers of different neural networks together to achieve success on cross domain problems between computer vision and natural language processing. The project also aims to use deep learning algorithms for data science problems like classification.

1.5 Dataset Description

Failed Cases Dataset

TABLE I

FAILED CASES DATASET DESCRIPTION

Input Features	9
Target Features	1
Type of features	Categorical, Numerical
Rows	124,494
Positive : Negative class ratio	106 : 124495
Challenges	Class Imbalance, No feature information

CIFAR10

The dataset contains 60,000 images of size 32x32. The dataset is divided in 10 classes with each class having 6,000 images. The dataset comes split in two parts for training and testing where 50,000 images are for training the model and 10,000 for testing the model. The training data and testing data are split randomly from the original dataset and the training data is divided into 5 batches with 10,000 images in each batch. It is possible that a training batch might contain more images from one class. The classes in the dataset are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. All the pictures in the dataset are unique. This dataset will be used for image classification.

CIFAR100

The dataset contains 60,000 images of size 32x32 but the dataset is divided into 100 classes with 600 images for each class. The dataset contains 500 training images for each class and 100 testing images. For the CIFAR100, the 100 classes are grouped in 20 superclasses. For example, subclasses beaver, dolphin, otter, seal and whale belongs to the superclass aquatic mammals. All the pictures in data are unique and have no overlap with any other class. Every superclass has 5 subclasses. This dataset will be used for image classification.

Flickr8K

It is a publically available dataset and is available from the University of Illinois at Urbana, Champaign. The dataset is made up of random images with each image having 5 captions associated with it. The dataset has no image of a popular person or a place. The dataset will be used for the automatic image captioning. The dataset contains a total of 8,000 images with each image of varying size. One part of the dataset is the images and one part is the text files which contain raw captions, lemmatized version of captions, training images, validation images, and test images.

2 Project Implementation

In this section, there will be detailed explanation on how the project was implemented along with instructions on installing the software on the local machine. The majority of the project was implemented on Google Collaboratory and Amazon AWS, which are cloud platform for high computation power, so there will be a guide to install and run the project on those platforms as well. The section will also describe about the steps taken to prepare the data for feeding into the neural network.

2.1 Software Installation and Setup

The project was entirely implemented in Python programming language version 3.6.3 because of the support in language for data analysis as well as deep learning libraries. The libraries that were used in the project are Pandas, NumPy, Keras, SciPy, Bokeh, MatPlotLib, Scikit-learn and TensorFlow. The Pandas library is used in Python for its data structure and data analysis tools. The library has several functions for data loading and data preparation. NumPy provides functions for scientific computing in Python, which in this project is required during the vector computation of the images and for reshaping the input data for CSV file problems. NumPy is also required by Keras on the backend. Keras is a high-level neural network API which uses TensorFlow or Theano on its backend. The library is written in Python and works on both GPU and CPU. TensorFlow is a high performance numerical computation library which uses data-flow graphs. The library has support for machine learning and deep learning on GPU, CPU and TPU.

The project requires graphical processing unit for computations because deep learning takes a lot of time and computational power which is not available on a normal computers central processing unit. The task that will take days on a CPU will be done in a few hours on GPU. Tensor Processing Unit (TPU) are special hardware which can do the same task in a few minutes, but they are not available in the market for public use.

2.1.1 Setup on Windows 7

The project was initially started on Windows 7, installed on a local machine with no GPU support. The GPU was not required for the data science part of the project at an early stage. The setup on the Windows platform was done using Anaconda, a repository for packages, notebooks and environment. Anaconda gives easy installation for libraries used in the project. The installation instructions can be found on https://anaconda.org/.

2.1.2 Setup on Google Collaboratory

Collaboratory is a Google research project which is free to use to help with machine learning, education, and research. Collaboratory uses a Jupyter Notebook environment which requires no setup to run or use Python version 2 or version 3. The cloud platform comes pre-installed with various Python libraries like Pandas, MatPlotLib and NumPy which are essential for machine learning and deep learning projects. The cloud platform also comes pre-installed with TensorFlow which can run libraries like Keras and Theano. Googles cloud platform was used in the project because of its Jupyter like interface and its free GPU access, which was essential for the project.

The Collaboratory platform is still in the development stage thus faces several issues like losing Python connection during the execution of the code. The virtual machine also has limited storage space available which was a problem in some parts of the project because of being memory intensive. All of the issues were acceptable as the platform is free to use and is a great effort from Google to make learning easier.

2.1.3 Setup on Amazon AWS

Due to the setbacks faced in Collaboratory, the project was switched to Amazon AWS cloud. AWS provides several pre-built Amazon Machine Image (AMI). There are AMI which comes with GPU computation support that are just built for deep learning applications. There are several versions of AMI, each with different level of computation power and memory. An AMI with GPU computation is charged more heftily than an AMI with less computation power.

To setup the machine on AWS, the procedure was followed on the AWS website.

Link: https://aws.amazon.com/blogs/machine-learning/get-started-with-deep-learning-using-the-aws-deep-learning-ami/ [22]

There is no need to change the environment in AWS from CPU to GPU as these AMI comes with GPU computation support by default. Using AWS require a basic understanding of networking concepts like ssh and scp to transfer files or access the cluster as the AMI does not come with a graphical interface [23].

2.2 Data Preprocessing

CIFAR10

The dataset comes for multiple programming languages. The dataset in the project was downloaded from https://www.cs.toronto.edu/~kriz/cifar.html. The dataset is also available within the Keras library. The dataset contains images in machine consumable byte format, thus no processing is required to convert the images into matrix format.

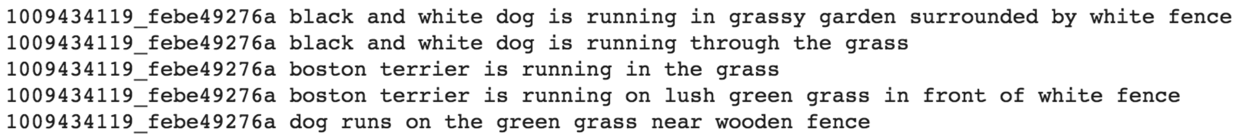
CIFAR100

The dataset comes for multiple programming languages. The dataset in the project was downloaded from https://www.cs.toronto.edu/~kriz/cifar.html. The dataset is also available within the Keras library. The dataset contains images in machine consumable byte format, thus no processing is required to convert the images into matrix format.

Flickr8K

The dataset is available only on University of Illinois at Urbana, Champaign. To download the dataset, a form is supposed to be filled over the university website and an a download link is emailed for the dataset. The images in the dataset are in JPG format. There are a total of 8000 images in the dataset. The dataset will be divided in the ratio of 6:1:1 for training, validation and testing respectively. To extract features from the image, the INCEPTIONV3 model was used. The model is freely available over the internet for extracting features from an image and is about 500 MB.

The dataset contains captions associated with every image, so the captions are to be associated with the image features and they require minimal cleaning. The photo ID is used to map the images to the descriptions. The data is converted into a dictionary with photo ID and associated text. To create a vocabulary for the model, the captions are cleaned to convert uppercase to lowercase and remove all the punctuations , stop words and numbers. Once the data is cleaned, the model will have a shorter vocabulary to learn. The dictionary is saved into a file with image ID and image captions, so 5 entries per file.



```
1009434119_febe49276a black and white dog is running in grassy garden surrounded by white fence
1009434119_febe49276a black and white dog is running through the grass
1009434119_febe49276a boston terrier is running in the grass
1009434119_febe49276a boston terrier is running on lush green grass in front of white fence
1009434119_febe49276a dog runs on the green grass near wooden fence
```

Fig. 9 Image ID with Caption sample

2.3 Model Training

2.3.1 Data Science

Deep Neural Network

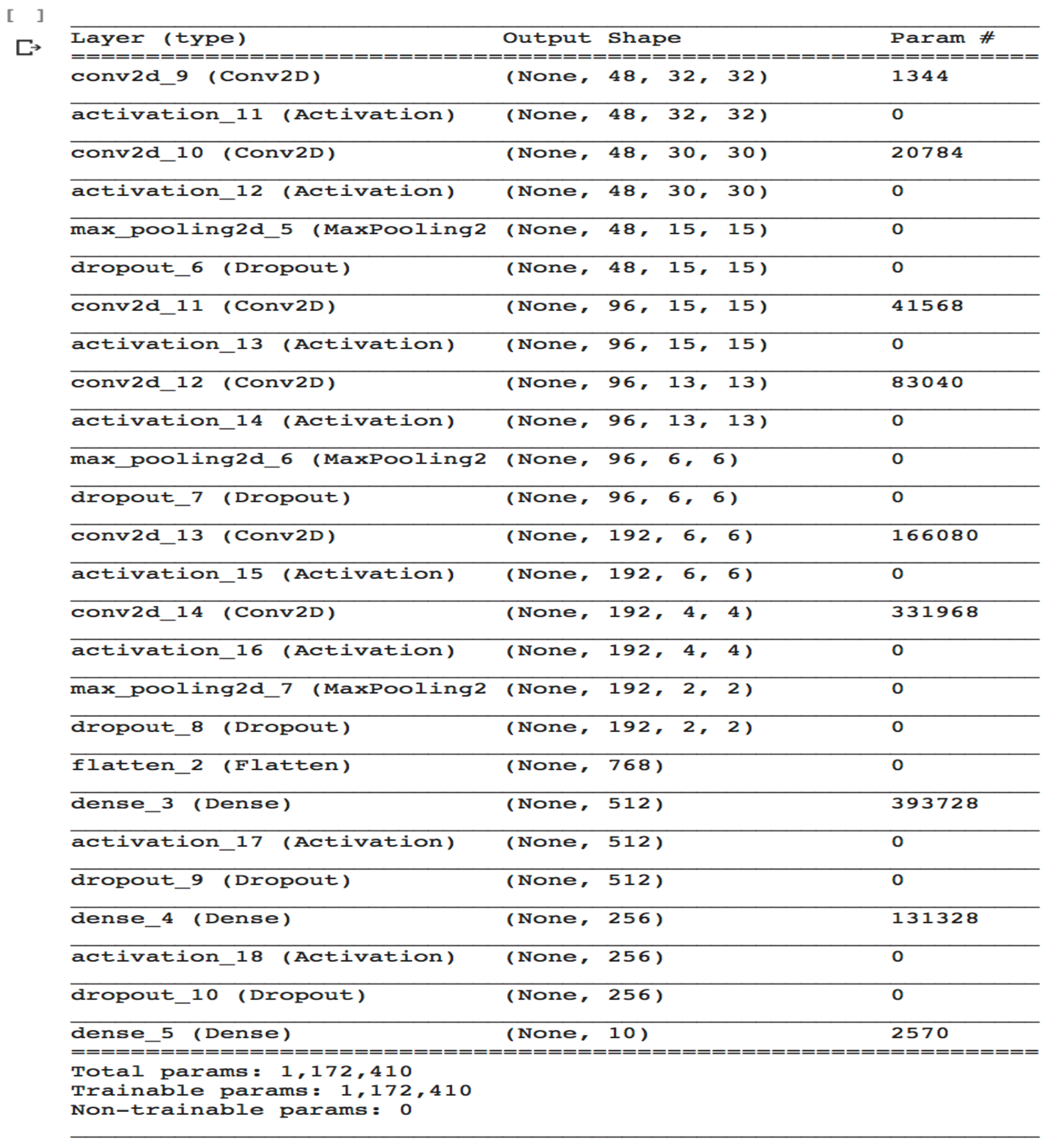
The training data prepared during the data processing part can be directly used to train the model in deep neural network. The constraint needs to be placed in the first layer of the deep neural network which should have an input shape which is same as the number of features in the training set. For example, if the dataset has 8 features, then the input shape should be (8,). The

last layer in the neural network should have 1 neuron if it is a binary classification and the activation function should be sigmoid [9]. Sigmoid gives an output of either 0 or 1.
The model is trained and tested on both the original data with the imbalance and the upsampled data with no class imbalance. The model is evaluated on the basis of accuracy, precision, recall, F1 score and support. [19]

Other Artificial Neural Networks
The training data prepared during the data processing needs to be reshaped before the data can be fed into the model [10]. These models only accept data in the form of a 3D matrix. For data science, the flow of the model needs to be sequential. The first layer of the models needs to be a model specific layer. The input data needs to be reshaped into a format where it should make the multiplication of the three dimension length makes up for the total features in the training set. So if the training data has 10 columns and 100 rows which makes a total of 1000 features, the data could be either reshaped into (100,5,2). The first dimension needs to be equal to the number of rows in the training set [2]. The first layer of the model will have the same number of neurons as the number of rows in the training set. The activation function used in the first layer in the model is relu for its advantages discussed in the literature review. The activation function could be replaced with other activation functions. There are other ways to construct the model with a different number of neurons but that requires data reshaping to be done accordingly. [15]
Once the input layer is placed, multiple layers of different neural networks like RNN, LSTM, DNN can be placed after it with reducing number or same number of neurons till the final layer is reached. The final layer is always a dense layer to make the classification output. Before the final dense layer of the neural network, a flattening layer needs to be added which flattens the output of convolution part of the neural network into a 1D vector. The reason to do this is that the final layer cannot work with a 3D matrix to generate a classification output. After the model is stacked up, it is compiled and fitted with training data. The fit command has several passable arguments but the arguments used in the project are epochs, batch_size, shuffle, and verbose.

2.3.2 CIFAR10 and CIFAR100

For the project, both CIFAR10 and CIFAR100 dataset were used to find the performance of stacked convolutional neural network and simple convolutional neural network. Multiple models were created and trained on CIFAR10 and CIFAR100 dataset to analyze the effects of the training data, activation function, layer stacking and data augmentation. The architecture of one of the complex model is given in Fig. 11.



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 48, 32, 32)        1344
_________________________________________________________________
activation_11 (Activation)   (None, 48, 32, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 48, 30, 30)        20784
_________________________________________________________________
activation_12 (Activation)   (None, 48, 30, 30)        0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 48, 15, 15)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 48, 15, 15)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 96, 15, 15)        41568
_________________________________________________________________
activation_13 (Activation)   (None, 96, 15, 15)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 96, 13, 13)        83040
_________________________________________________________________
activation_14 (Activation)   (None, 96, 13, 13)        0
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 96, 6, 6)          0
_________________________________________________________________
dropout_7 (Dropout)          (None, 96, 6, 6)          0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 192, 6, 6)         166080
_________________________________________________________________
activation_15 (Activation)   (None, 192, 6, 6)         0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 192, 4, 4)         331968
_________________________________________________________________
activation_16 (Activation)   (None, 192, 4, 4)         0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 192, 2, 2)         0
_________________________________________________________________
dropout_8 (Dropout)          (None, 192, 2, 2)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 768)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               393728
_________________________________________________________________
activation_17 (Activation)   (None, 512)               0
_________________________________________________________________
dropout_9 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 256)               131328
_________________________________________________________________
activation_18 (Activation)   (None, 256)               0
_________________________________________________________________
dropout_10 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_5 (Dense)              (None, 10)                2570
=================================================================
Total params: 1,172,410
Trainable params: 1,172,410
Non-trainable params: 0
_________________________________________________________________
```

Fig. 10 Model Summary

Several models were used with different stack of convolutional layers with different activation functions. In the model shown in Fig. 11, there are 6 convolutional layer with the model starting with 2 convolutional layer followed by one pooling layer to compress the output from the two convolution layers. After these layer, a dropout layer is added to the model to prevent the model from overfitting. At dropout layer, a fraction on input units is dropped at random to prevent the model from overfitting. After the dropout layer, the model is again followed by a two pairs of convolutional layer and pooling layer and the same concept is repeated. After the training, the output was flattened to pass through a dense layer with 512 neurons which is then reduced to 10 neurons, equal to the number of classes in CIFAR10 dataset.

The exact same models were trained on CIFAR100 dataset as well for measuring and comparing the performance of the model. Data augmentation techniques were used to generate more training data for both CIFAR10 and CIFAR100 dataset and the model was trained and tested on these augmented datasets as well.

2.3.3 Automatic Image Captioning

This part of the project is associated with creating a model that is a combination of convolutional neural network and LSTM network to detect objects in an image and describe that image with a caption. It requires methods from the domain of computer vision and from language modelling. The model is trained on the Flickr8K dataset which contains images and 5 captions associated with every image. The training of the model requires several steps once the data is processed.

- Load the processed data into the memory.
- From the data, load 6000 training images, 1000 validation images and 1000 testing images.
- Create a dictionary with key as image ID and value as a list of captions for training, testing and validation dataset.
- The images are compressed into size (299,299) for consistency before encoding and are converted into array.
- The images are encoded using the INCEPTIONV3 model. As discussed in literature review and other projects, the last layer is for classification in neural networks. The same applies to INCEPTIONV3 as well. The last layer is a softmax layer with 1000 neurons and it is removed. This makes the last layer a dense layer with 4096 neurons. The size of encoded images will be (1,2048).
- All the images are encoded using the modified INCEPTIONV3 model into shape (1,2048).
- The encoded data could be saved down to system to remove these time consuming steps when retraining the model.
- The same process is done with validation and test images.

- The vocabulary of the model is built using unique words from all the captions. There are a total of 8256 unique words in the model.
- All the unique words are mapped to indexes.
- All the captions in are appended with start and end sequence to mark the start and end of an caption.
- The caption and image ID are divided into a list with caption starting with <start> and ending in <end> and its image id. This yields a total of 30000 entries. Example:
  ('<start> Two girls playing in a park . <end>', '3421928157_26a325345f.jpg')
- Sample of how the words are mapped to the index.
  <start> => 5553 Two => 2666 girls => 4606 playing => 3779 in => 8156 a => 32 park => 1816 . => 7023 <end> => 5232
- Deep learning model
  - Image Mode: A sequential model with one dense layer because the images are already processed by the INCEPTIONV3 model.
  - Caption Model: To predict the next word based on the previous word. This requires the model to have return_sequences enabled. This is LSTM model.
  - Final_model: The image model and the caption model are concatenated together to yield the final model with the final softmax activation layer.

```
_________________________________________________________________________________________________
__
Layer (type)                     Output Shape          Param #     Connected to
=================================================================================================
==
dense_1 (Dense)                  (None, 300)           614700
_________________________________________________________________________________________________
__
repeatvector_1 (RepeatVector)    (None, 40, 300)       0
_________________________________________________________________________________________________
__
embedding_1 (Embedding)          (None, 40, 300)       2476800
_________________________________________________________________________________________________
__
lstm_1 (LSTM)                    (None, 40, 256)       570368
_________________________________________________________________________________________________
__
timedistributed_1 (TimeDistribut (None, 40, 300)       77100
_________________________________________________________________________________________________
__
bidirectional_1 (Bidirectional)  (None, 512)           1140736     merge_1[0][0]
_________________________________________________________________________________________________
__
dense_3 (Dense)                  (None, 8256)          4235328     bidirectional_1[0][0]
_________________________________________________________________________________________________
__
activation_1 (Activation)        (None, 8256)          0           dense_3[0][0]
=================================================================================================
==
Total params: 9,115,032
Trainable params: 9,115,032
Non-trainable params: 0
```

Fig. 11 Model Summary

- The model is compiled and trained using the data generator function which sends one word at a time for an image and keeps on appending words.
- The models weight and architecture is saved as an h5 file.

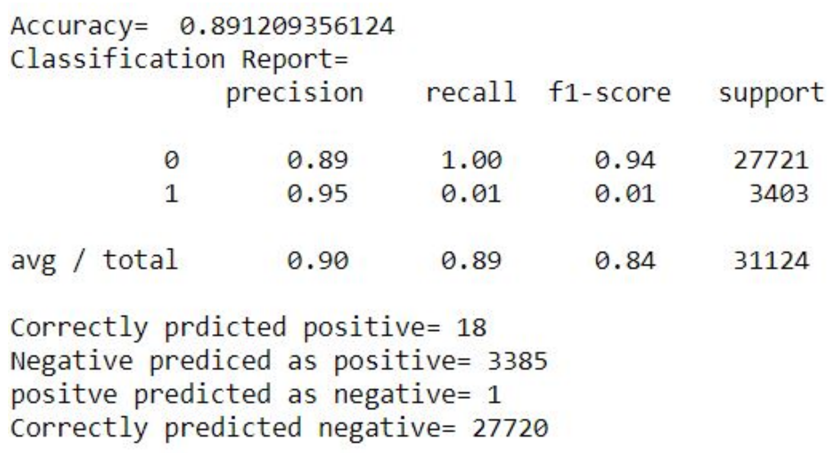
3 Results

The results from the project are summarized in this section. There are different evaluation matrices used for different task of the project because of the difference in their domain. The results are divided into multiple section, each section for a different part of the project.

3.1 Data Science

Predict Failure Dataset

Deep Neural Network
(Dense Layers)



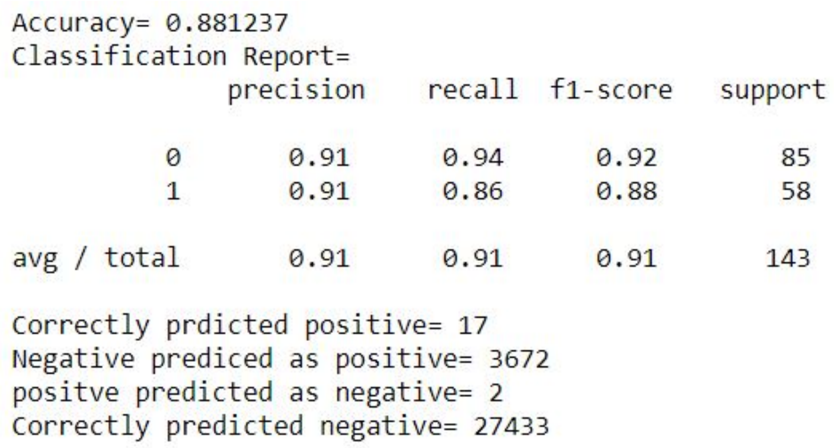
```
Accuracy=  0.891209356124
Classification Report=
             precision    recall  f1-score   support

          0       0.89      1.00      0.94     27721
          1       0.95      0.01      0.01      3403

avg / total       0.90      0.89      0.84     31124

Correctly prdicted positive= 18
Negative prediced as positive= 3385
positve predicted as negative= 1
Correctly predicted negative= 27720
```

Recurrent Neural Network
(SimpleRNN Layer)



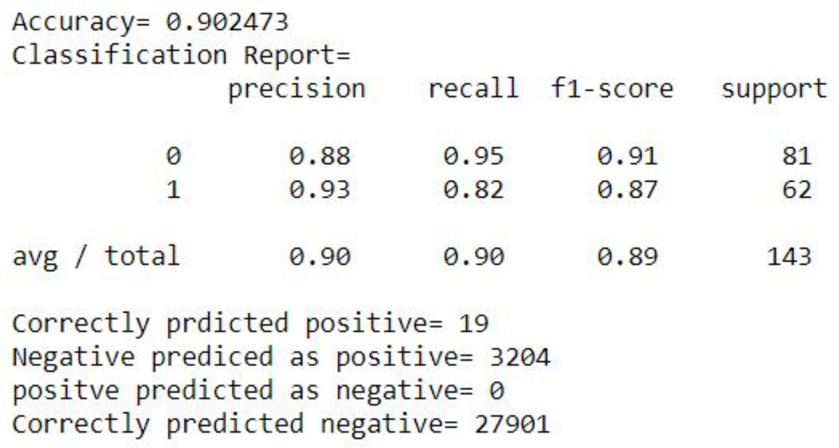
```
Accuracy= 0.881237
Classification Report=
             precision    recall  f1-score   support

          0       0.91      0.94      0.92        85
          1       0.91      0.86      0.88        58

avg / total       0.91      0.91      0.91       143

Correctly prdicted positive= 17
Negative prediced as positive= 3672
positve predicted as negative= 2
Correctly predicted negative= 27433
```

Long Short Term Memory
Network (LSTM Layer)



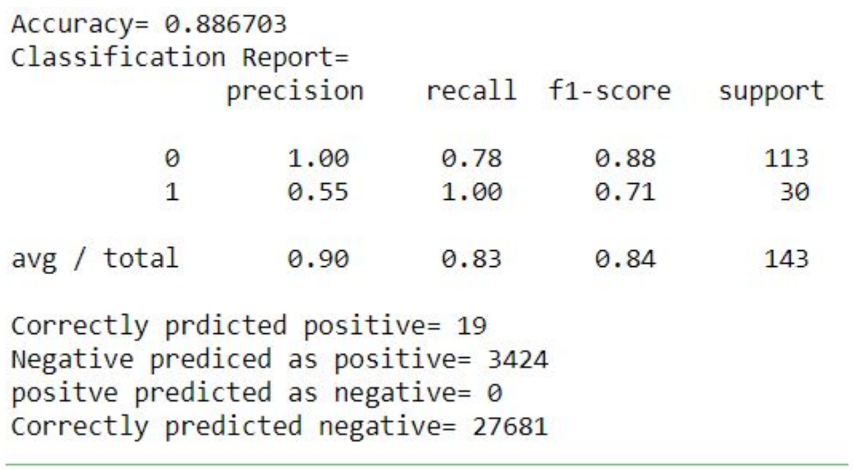
```
Accuracy= 0.902473
Classification Report=
             precision    recall  f1-score   support

          0       0.88      0.95      0.91        81
          1       0.93      0.82      0.87        62

avg / total       0.90      0.90      0.89       143

Correctly prdicted positive= 19
Negative prediced as positive= 3204
positve predicted as negative= 0
Correctly predicted negative= 27901
```

Convolutional Neural Network (Convolutional Layer)



```
Accuracy= 0.886703
Classification Report=
              precision    recall  f1-score   support

          0       1.00      0.78      0.88       113
          1       0.55      1.00      0.71        30

avg / total       0.90      0.83      0.84       143

Correctly prdicted positive= 19
Negative prediced as positive= 3424
positve predicted as negative= 0
Correctly predicted negative= 27681
```

3.2 CIFAR10 and CIFAR100

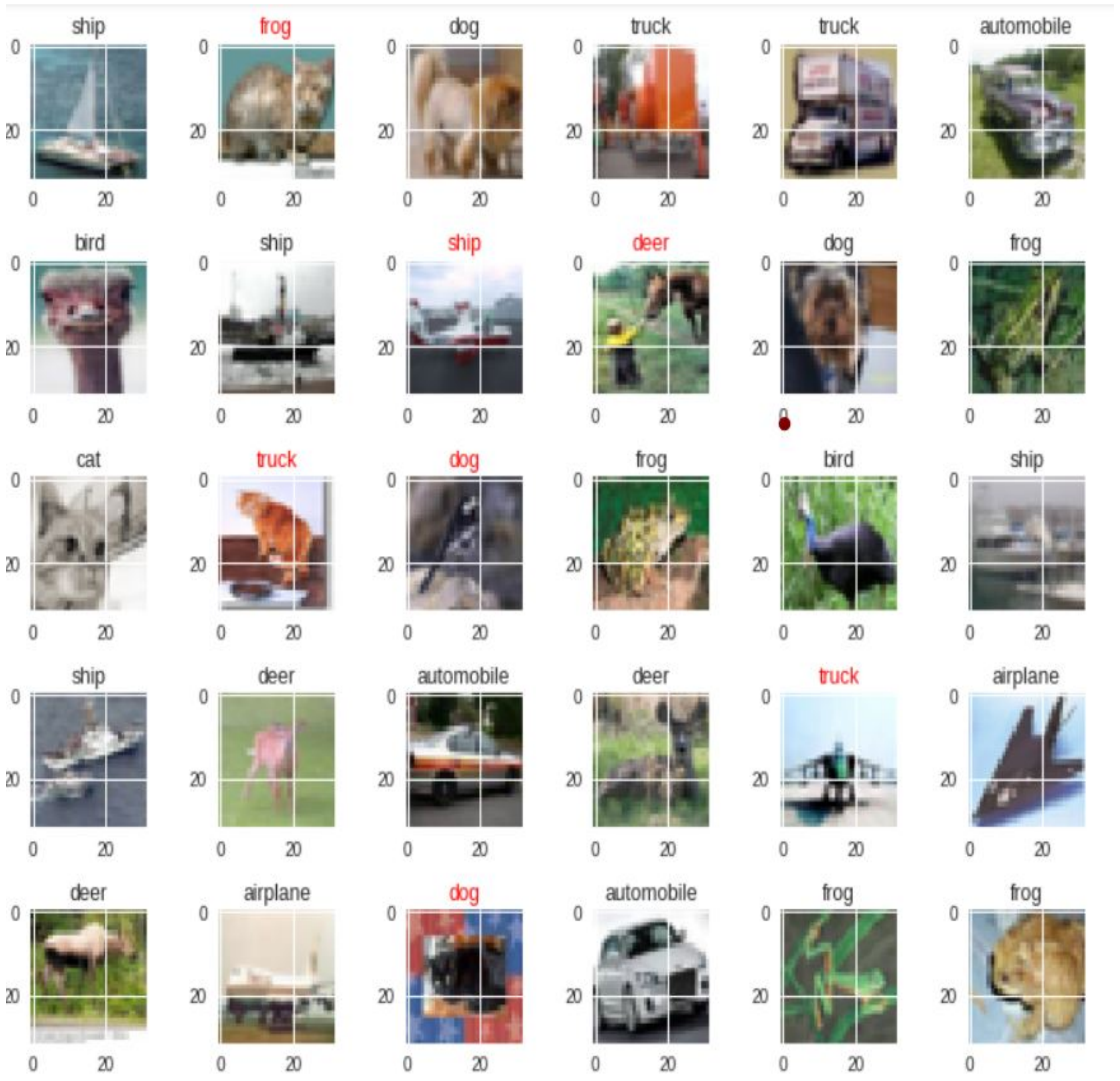
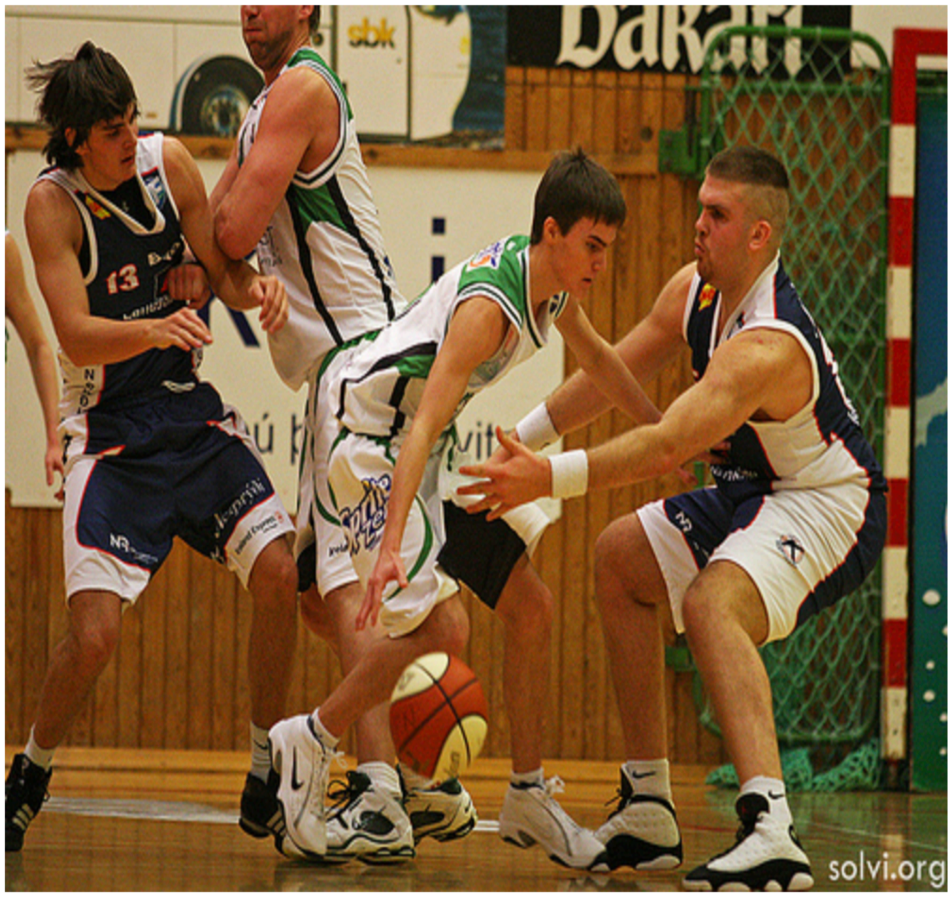
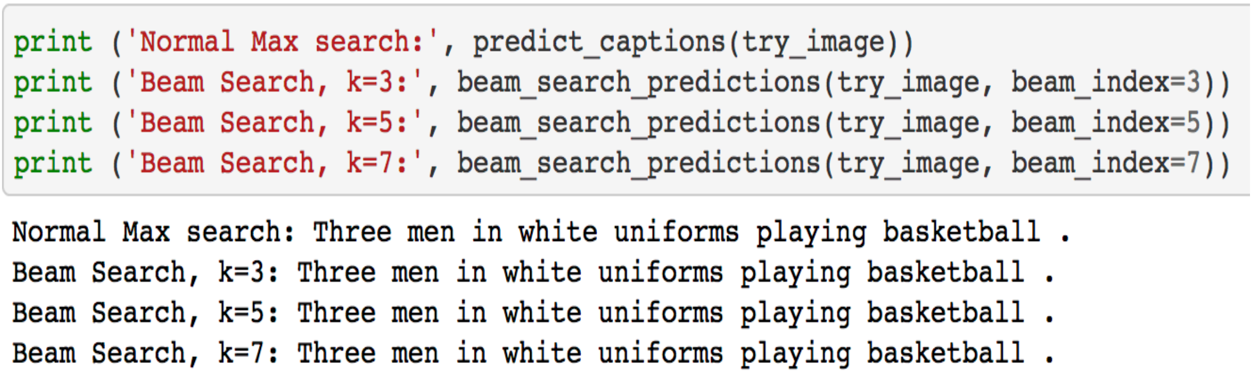


Fig. 12 Result from CIFAR10 Classification

3.3 Automatic Caption Generator





```
print ('Normal Max search:', predict_captions(try_image))
print ('Beam Search, k=3:', beam_search_predictions(try_image, beam_index=3))
print ('Beam Search, k=5:', beam_search_predictions(try_image, beam_index=5))
print ('Beam Search, k=7:', beam_search_predictions(try_image, beam_index=7))
```

```
Normal Max search: Three men in white uniforms playing basketball .
Beam Search, k=3: Three men in white uniforms playing basketball .
Beam Search, k=5: Three men in white uniforms playing basketball .
Beam Search, k=7: Three men in white uniforms playing basketball .
```

Fig. 13 Result of Caption Generator -I

```
Normal Max search: A skateboarder in the air in front of a red slide .
Beam Search, k=3: A skateboarder in the air in front of a blue building .
Beam Search, k=5: A skateboarder in the air in front of a blue building .
Beam Search, k=7: A male skateboarder is skating in a skate park .
```

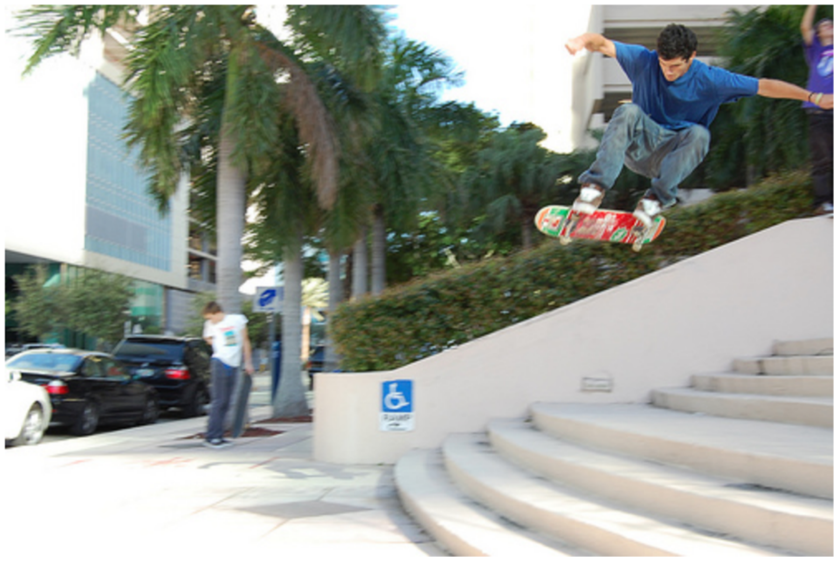


Fig. 14 Result of Caption Generator -II

```
Normal Max search: A person on a bike is coming up through the mud .
Beam Search, k=3: A guy is doing a trick on a bike .
Beam Search, k=5: A mountain biker jumps a rock on a mountain .
Beam Search, k=7: A mountain biker is riding on a line back at bushes .
```

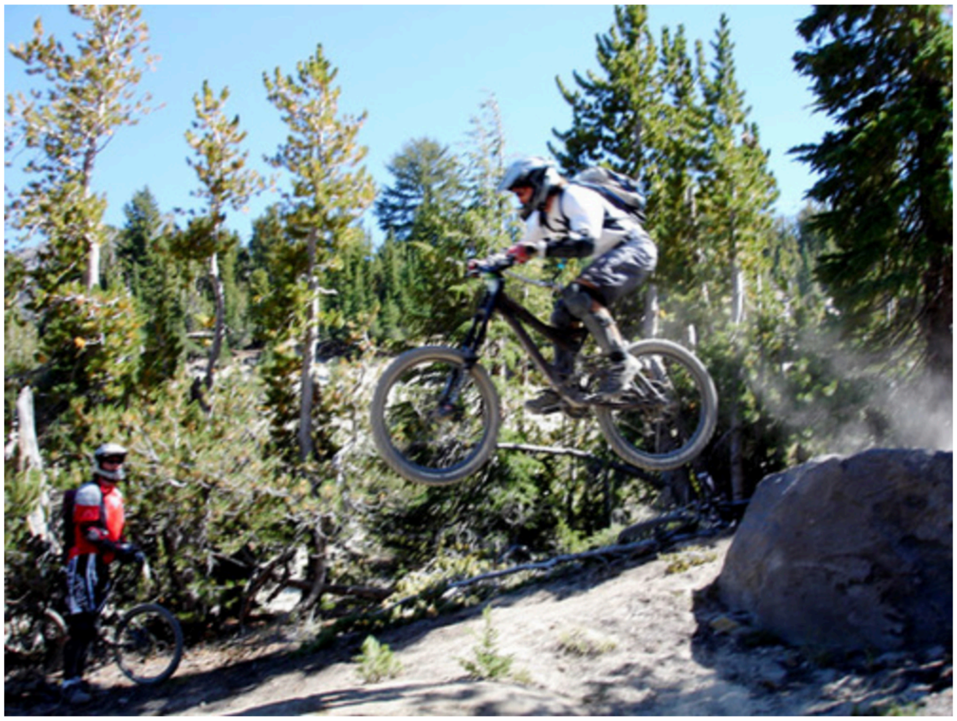


Fig. 15 Result of Caption Generator -III

```
Normal Max search: A dog is jumping in the air to catch something .
Beam Search, k=3: A brown dog is jumping in the air .
Beam Search, k=5: A dog is jumping in the air to catch something .
Beam Search, k=7: A dog in a harness holds a stick in his mouth while standing in the grass .
```

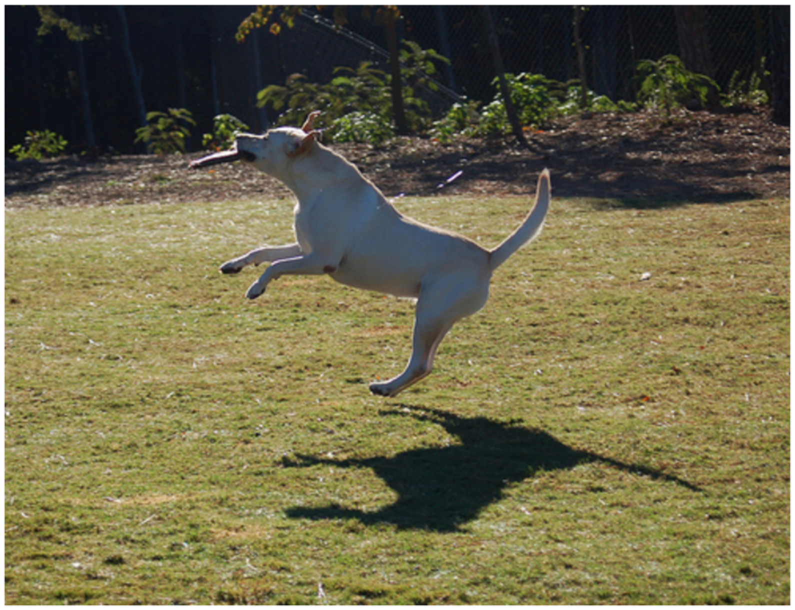


Fig. 16 Result of Caption Generator -IV

```
Normal Max search: A man rides a bicycle on a trail down a river .
Beam Search, k=3: A man is riding a bicycle on a trail through some trees .
Beam Search, k=5: A man rides a mountain bike down a slope in the woods .
Beam Search, k=7: A man rides a bicycle on a trail down a river .
```

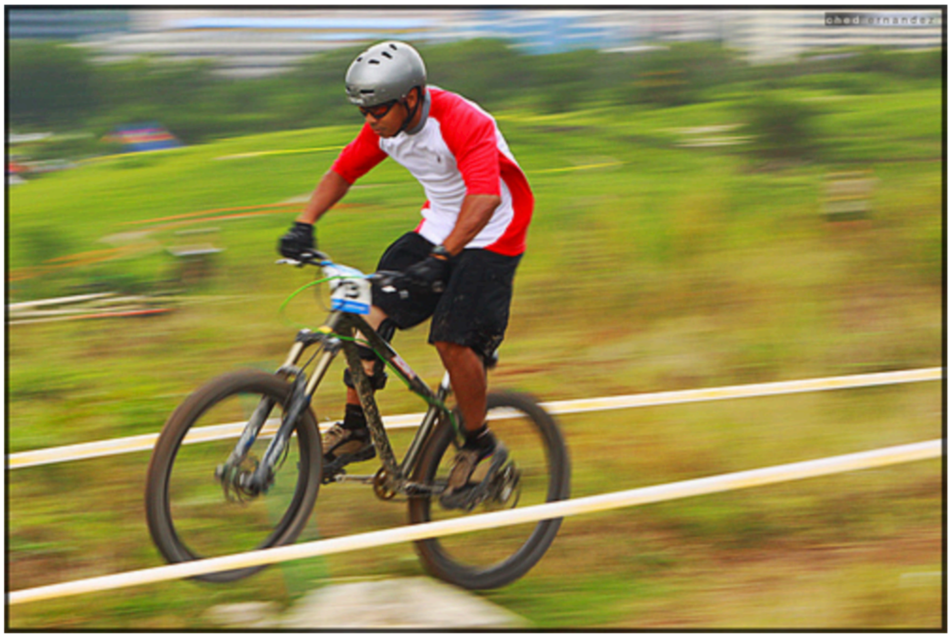


Fig. 17 Result of Caption Generator -V

4 Analysis

4.1 Data Science

Using deep learning is a highly complicated task when dealing with data science problems like classification or regression. The model trains itself slowly when compared to standard machine learning algorithms like random forest, decision tree or support vector machine. Deep learning algorithms are less prone to overfitting the training data unlike machine learning models but this advantage does not overcome the other drawbacks. The following inferences could be made by comparing the results:

- Reshaping the data is a challenging aspect which can greatly affect the performance of the model as the number of parameters the model will take depends on it.
- Stacking up layers with different properties can help reduce overfitting but increase computation time.
- Model requires GPU computation power and high memory space to complete training, unlike standard machine learning algorithms.
- The model creation requires extensive knowledge of the dataset and features.
- Data normalization is extremely important and can have a huge effect on the model.
- Models are less prone to class imbalance than many standard machine learning algorithms.
- Unlike machine learning, in deep learning the choice of model can have huge performance impact. RNN and LSTM work great on sequence-based data while CNN works best for data with higher dimensions.
- The model performs poorly than machine learning algorithms on most of the dataset. One reason for this could be the less number of layers that could be stacked due to computing limit.

4.2 CIFAR10 and CIFAR100

Both the dataset contains images of size 32x32 with 3 color dimensions. The difference between the two datasets is the huge difference in training samples for every class. In CIFAR10, there are 6000 images for every class while in CIFAR100 there are only 600 images per class. Based on the models generated on this dataset, the following are the findings.

- Data augmentation can significantly increase a models performance in terms of accuracy on the validation set but the same cannot be said for the training set.
- The type of activation function has minimal effect on the performance of the model in terms of accuracy when the dataset is of a limited size with fixed size images.
- The type of activation function has a good effect on the training time.
- The choice of activation function at the final layer affects the models as it decides the what the model should predict.
- The choice and number of layers can have a great impact on the performance of the model.
- For better training and to avoid overfitting, the model needs to have at least three to four convolutional layer along with dropout layers at regular intervals with a good range.
- The dropout should start with 25% and should go as high as 50% at the end. Dropout higher than 75% leads to important feature loss.
- The size of pooling layer depends on the size of the input feature maps. If the input and feature maps are of small size, a pool of size 2x2 works the best. By using a huge pool, the model can remove important features.
- The number of feature maps should start with a small number at starting layer and should go high based on the features in the data.
- Pooling layer should be placed after at least two convolutional layers to reduce their size. Pooling the output of every convolutional layer slows down the model and also does not utilize full potential of the pooling layers.

The results on the two models are as follows:

CIFAR10 Results:

TABLE II

CIFAR10 RESULTS

CL Layer	Architecture	Train Accuracy	Validation Accuracy	Train Loss	Validation loss
3	conv48, conv48, pool2x2, drop25, conv96, pool, flat, dense1024, drop25, dense10	78.25	76.54	0.53	0.61

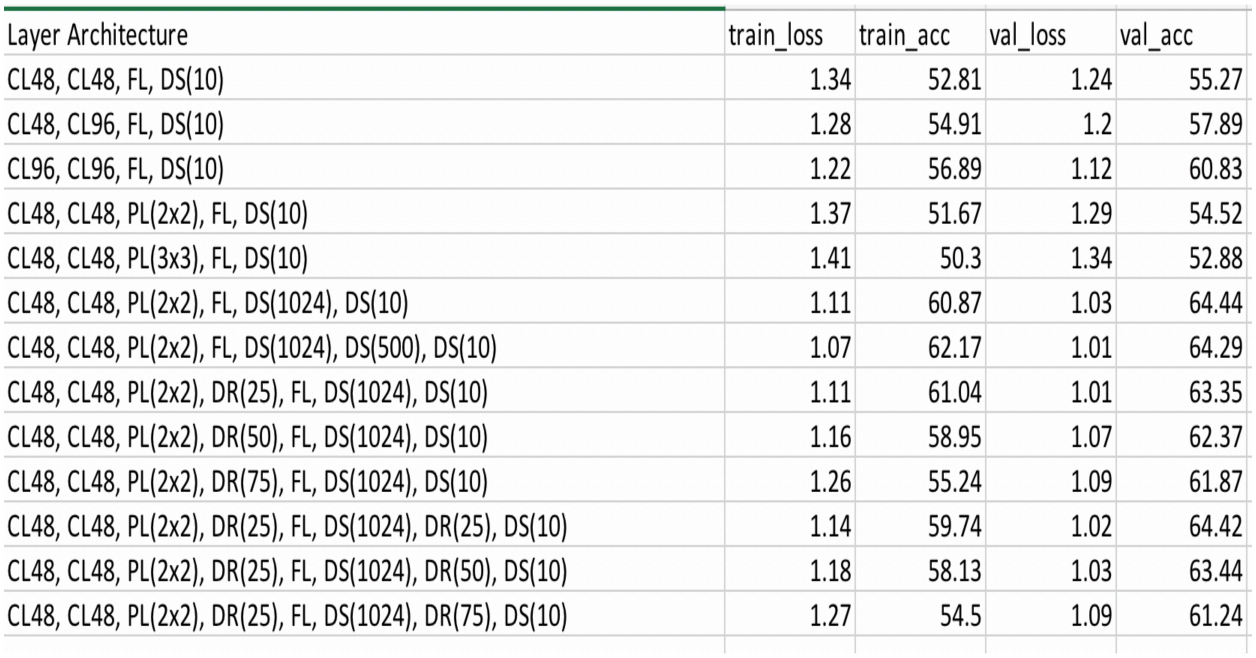
4	conv48, conv48, pool2x2, drop25, conv96, conv96, pool2x2, drop25, flat, dense 1024, drop50, dense10	83.74	79.41	0.47	0.76
5	conv48, conv48, pool2x2, drop25, conv96, conv96, pool2x2, drop25, conv192 flat, dense 1024, drop50, dense10	84.25	81.46	0.36	0.7
6	conv48, conv48, pool2x2, drop25, conv96, conv96, pool2x2, drop25, conv192, conv192, pool2x2, drop25 flat, dense512, drop25, dense256, drop50, dense10	91.78	85.07	0.22	0.67
6 with augmentation	conv48, conv48, pool2x2, drop25, conv96, conv96, pool2x2, drop25, conv192, conv192, pool2x2, drop25 flat, dense512, drop25, dense256, drop50, dense10	89.12	87.97	0.27	0.64

To recommend an appropriate model for the problem, several architectures were tried by adding layers and tuning the hyperparameters. In the following results, the representation will be

Convolutional Layer	CL(Input feature map size)
Dropout Layer	DR (dropout percentage)
Flatten Layer	FL
Dense Layer	DS (number of neurons)
Pooling Layer	PL (pool size)

Two Layer Models:

The first model was tried with a total of two convolutional layers and multiple other layers including pooling layer, dense layer, dropout layer and flatten layer. Every model was executed for a total of 10 epochs to compare the effects of the hyperparameter.



Layer Architecture	train_loss	train_acc	val_loss	val_acc
CL48, CL48, FL, DS(10)	1.34	52.81	1.24	55.27
CL48, CL96, FL, DS(10)	1.28	54.91	1.2	57.89
CL96, CL96, FL, DS(10)	1.22	56.89	1.12	60.83
CL48, CL48, PL(2x2), FL, DS(10)	1.37	51.67	1.29	54.52
CL48, CL48, PL(3x3), FL, DS(10)	1.41	50.3	1.34	52.88
CL48, CL48, PL(2x2), FL, DS(1024), DS(10)	1.11	60.87	1.03	64.44
CL48, CL48, PL(2x2), FL, DS(1024), DS(500), DS(10)	1.07	62.17	1.01	64.29
CL48, CL48, PL(2x2), DR(25), FL, DS(1024), DS(10)	1.11	61.04	1.01	63.35
CL48, CL48, PL(2x2), DR(50), FL, DS(1024), DS(10)	1.16	58.95	1.07	62.37
CL48, CL48, PL(2x2), DR(75), FL, DS(1024), DS(10)	1.26	55.24	1.09	61.87
CL48, CL48, PL(2x2), DR(25), FL, DS(1024), DR(25), DS(10)	1.14	59.74	1.02	64.42
CL48, CL48, PL(2x2), DR(25), FL, DS(1024), DR(50), DS(10)	1.18	58.13	1.03	63.44
CL48, CL48, PL(2x2), DR(25), FL, DS(1024), DR(75), DS(10)	1.27	54.5	1.09	61.24

Fig. 18 CIFAR10 Two Layer Model Results

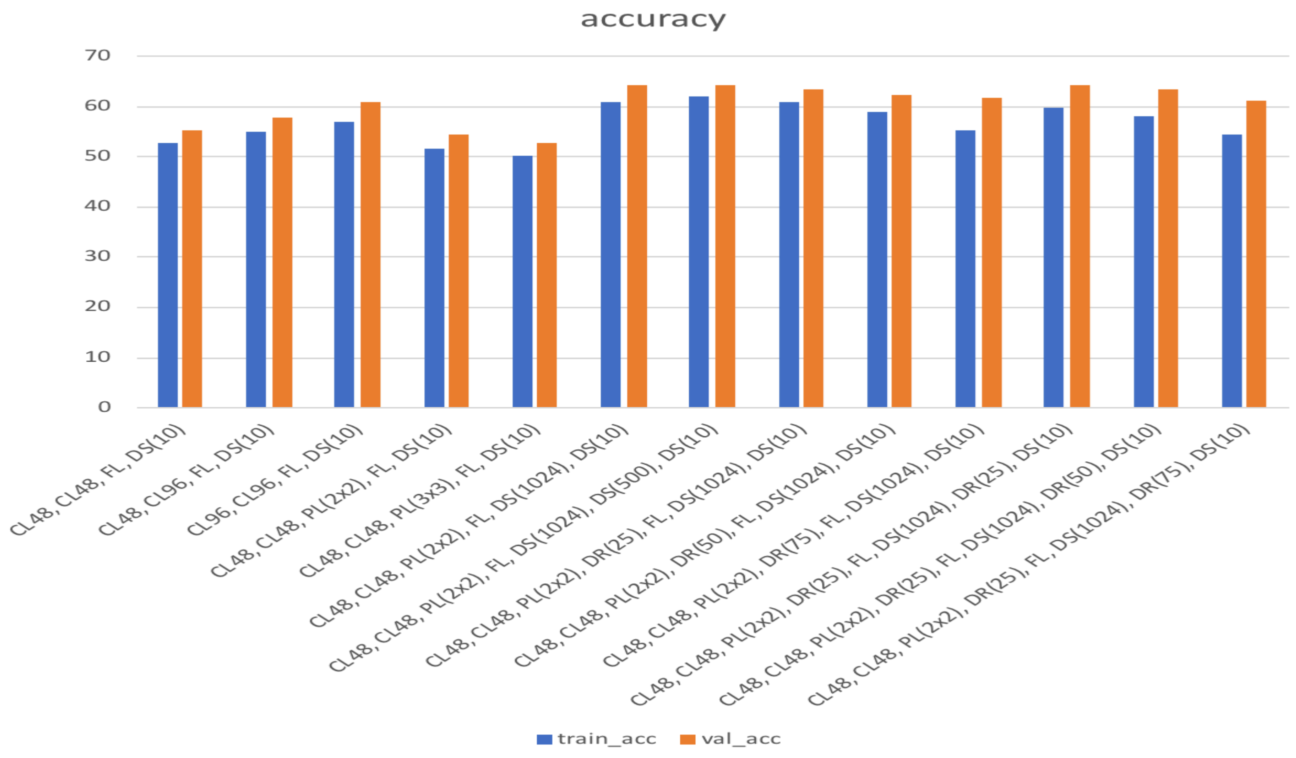


Fig. 19 CIFAR10 Two Layer Mode Accuracy Results

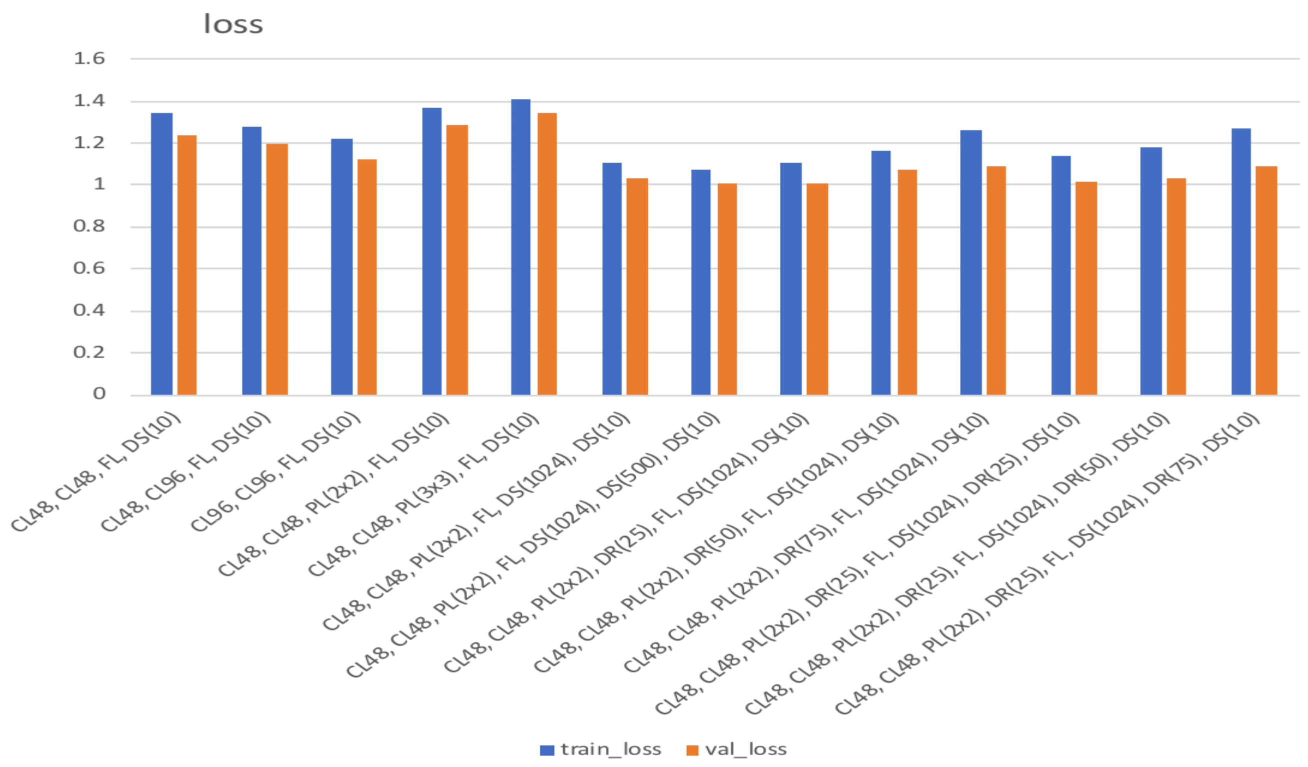


Fig. 20 CIFAR10 Two Layer Mode Loss Results

Three Layer Models:

Layer	train_loss	train_acc	val_loss	val_acc
CL, CL, CL(96), FL, DS(10)	1.23	56.61	1.13	60.61
CL(48), CL(48), PL(2x2) CL(96), FL, DS(10)	1.32	53.27	1.23	57
CL(48), CL(48), PL(2x2), DR(25), CL(96), FL, DS(10)	1.35	52.08	1.28	54.45
CL(48), CL(48), PL(2x2), DR(50), CL(96), FL, DS(10)	1.35	52.02	1.26	54.33
CL(48), CL(48), PL(2x2), DR(75), CL(96), FL, DS(10)	1.39	49.8	1.39	50.45
CL(48), CL(48), PL(2x2), DR(25), CL(96), PL(2x2), FL, DS(10)	1.41	49.25	1.36	51.38
CL(48), CL(48), PL(2x2), DR(25), CL(96), PL(2x2), FL, DS(1024), DS(10)	1.18	58.43	1.14	59.4
CL(48), CL(48), PL(2x2), DR(25), CL(96), PL(2x2), FL, DS(1024), DR(25), DS(10)	1.19	57.51	1.12	60.21
CL(48), CL(48), PL(2x2), DR(25), CL(96), PL(2x2), FL, DS(1024), DR(50), DS(10)	1.24	55.72	1.15	59.3
CL(48), CL(48), PL(2x2), DR(25), CL(96), PL(2x2), FL, DS(1024), DR(75), DS(10)	1.32	52.51	1.18	58.55
CL(48), CL(48), PL(2x2), DR(100), CL(96), PL(2x2), FL, DS(1024), DR(100), DS(10)	1.15	59.7	1.09	61.63

Fig. 21 CIFAR10 Three Layer Model Results

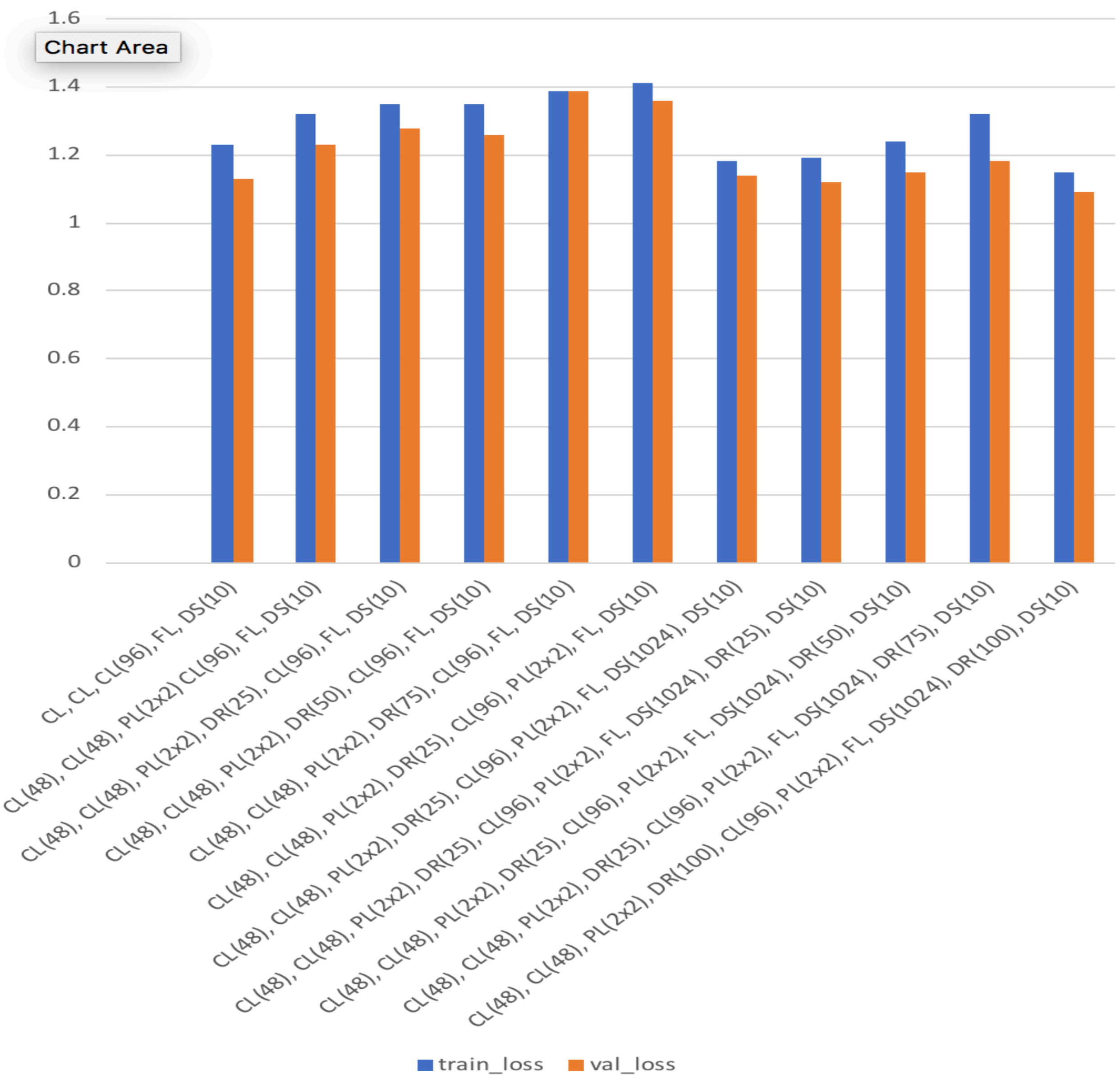


Fig. 22 CIFAR10 Three Layer Mode Loss Results

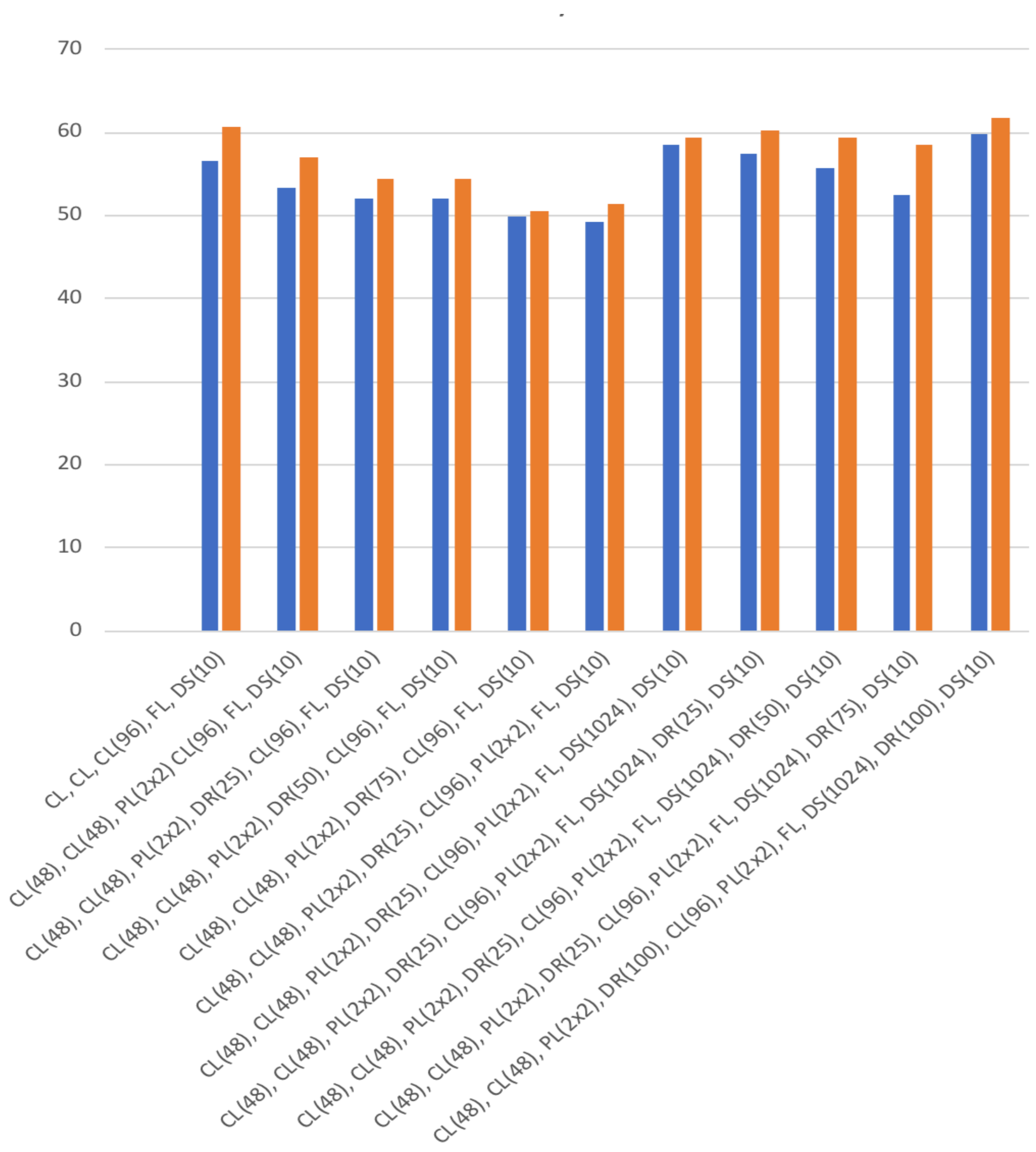
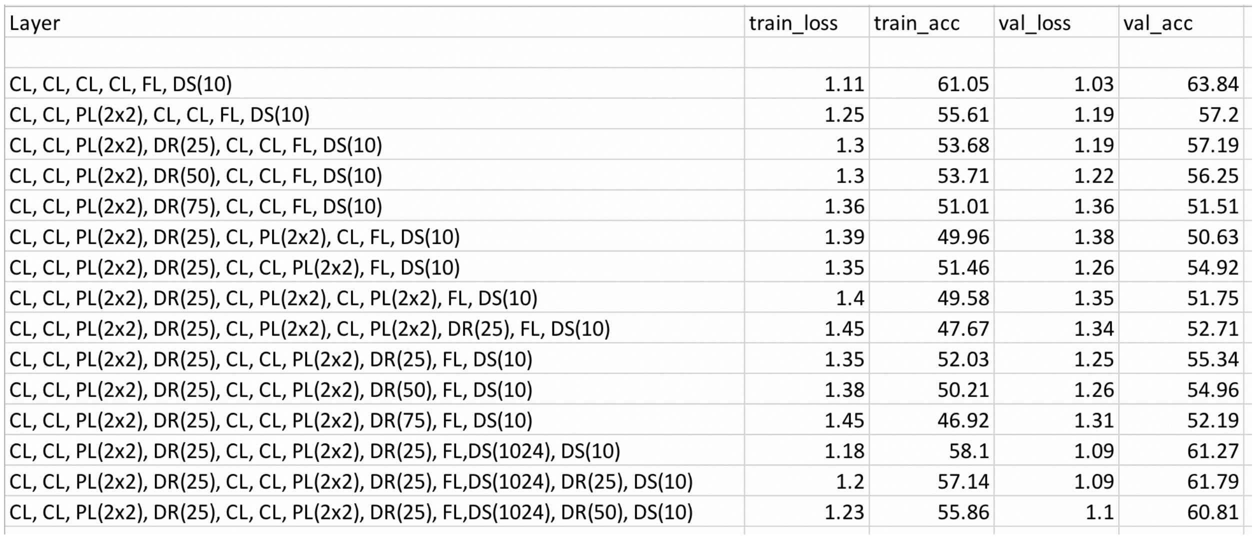


Fig. 23 CIFAR10 Three Layer Mode Accuracy Results

Four Layer Models:



Layer	train_loss	train_acc	val_loss	val_acc
CL, CL, CL, CL, FL, DS(10)	1.11	61.05	1.03	63.84
CL, CL, PL(2x2), CL, CL, FL, DS(10)	1.25	55.61	1.19	57.2
CL, CL, PL(2x2), DR(25), CL, CL, FL, DS(10)	1.3	53.68	1.19	57.19
CL, CL, PL(2x2), DR(50), CL, CL, FL, DS(10)	1.3	53.71	1.22	56.25
CL, CL, PL(2x2), DR(75), CL, CL, FL, DS(10)	1.36	51.01	1.36	51.51
CL, CL, PL(2x2), DR(25), CL, PL(2x2), CL, FL, DS(10)	1.39	49.96	1.38	50.63
CL, CL, PL(2x2), DR(25), CL, CL, PL(2x2), FL, DS(10)	1.35	51.46	1.26	54.92
CL, CL, PL(2x2), DR(25), CL, PL(2x2), CL, PL(2x2), FL, DS(10)	1.4	49.58	1.35	51.75
CL, CL, PL(2x2), DR(25), CL, PL(2x2), CL, PL(2x2), DR(25), FL, DS(10)	1.45	47.67	1.34	52.71
CL, CL, PL(2x2), DR(25), CL, CL, PL(2x2), DR(25), FL, DS(10)	1.35	52.03	1.25	55.34
CL, CL, PL(2x2), DR(25), CL, CL, PL(2x2), DR(50), FL, DS(10)	1.38	50.21	1.26	54.96
CL, CL, PL(2x2), DR(25), CL, CL, PL(2x2), DR(75), FL, DS(10)	1.45	46.92	1.31	52.19
CL, CL, PL(2x2), DR(25), CL, CL, PL(2x2), DR(25), FL,DS(1024), DS(10)	1.18	58.1	1.09	61.27
CL, CL, PL(2x2), DR(25), CL, CL, PL(2x2), DR(25), FL,DS(1024), DR(25), DS(10)	1.2	57.14	1.09	61.79
CL, CL, PL(2x2), DR(25), CL, CL, PL(2x2), DR(25), FL,DS(1024), DR(50), DS(10)	1.23	55.86	1.1	60.81

Fig. 24 CIFAR10 Four Layer Model Results

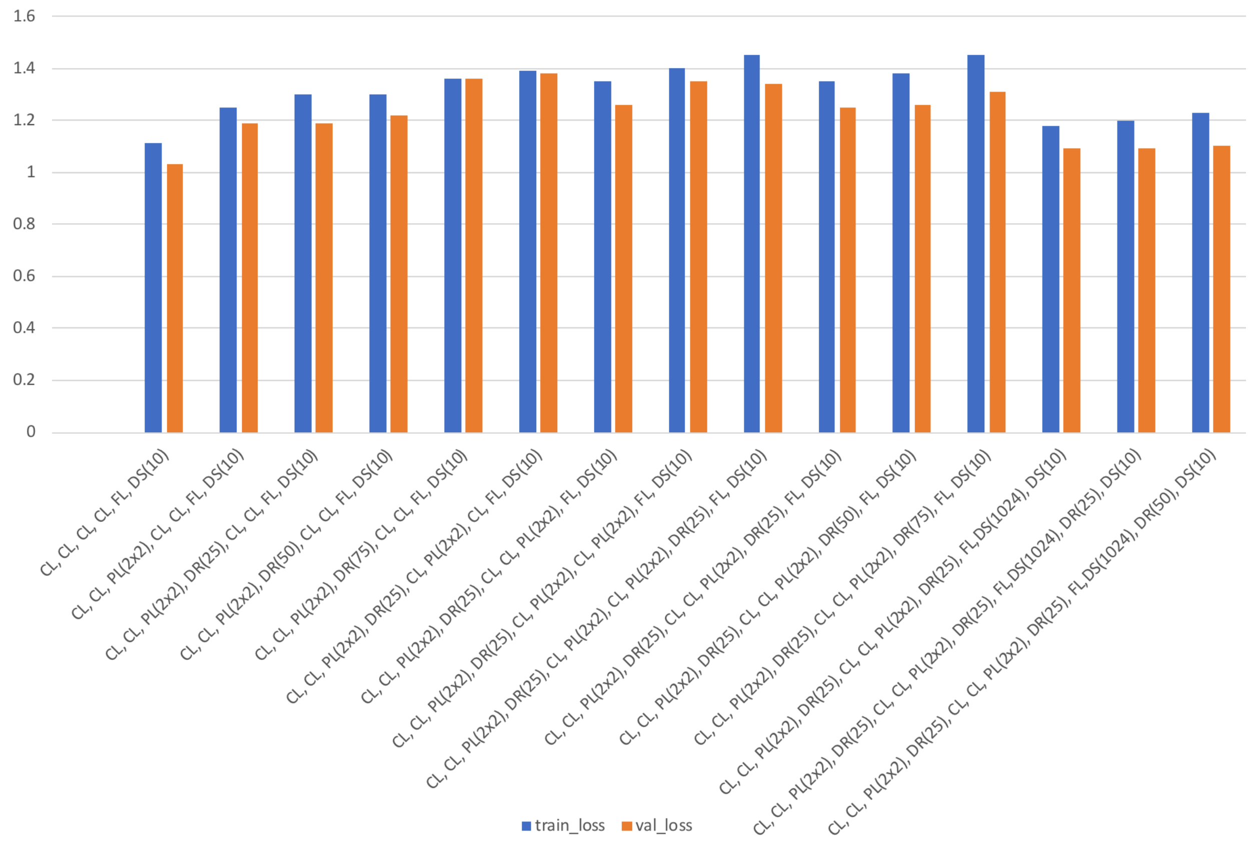


Fig. 25 CIFAR10 Four Layer Model Loss Results

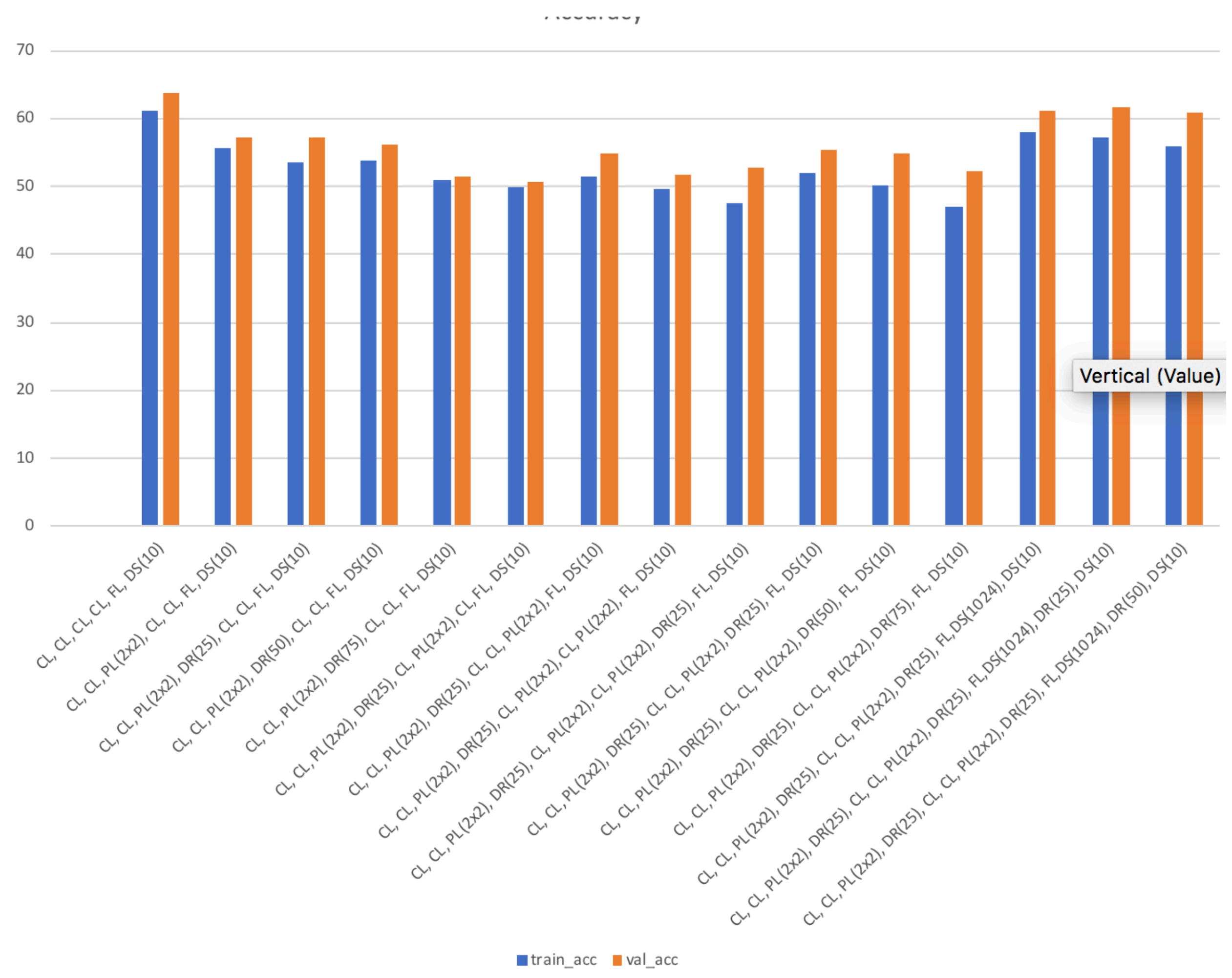


Fig. 26 Four Layer Model Accuracy Results

CIFAR100 Results:

TABLE III

CIFAR100 Results

CL Layer	Architecture	Train Accuracy	Validation Accuracy	Train Loss	Validation loss
3	conv48, conv48, pool2x2, drop25, conv96, pool, flat, dense1024, drop25, dense10	51.57	49.1	0.65	0.95
4	conv48, conv48, pool2x2, drop25, conv96, conv96, pool2x2, drop25, flat, dense 1024, drop50, dense10	60.14	55.35	0.59	0.87

5	conv48, conv48, pool2x2, drop25, conv96, conv96, pool2x2, drop25, conv192 flat, dense 1024, drop50, dense10	63.45	57.44	0.54	0.76
6	conv48, conv48, pool2x2, drop25, conv96, conv96, pool2x2, drop25, conv192, conv192, pool2x2, drop25 flat, dense512, drop25, dense256, drop50, dense10	67.11	60.31	0.47	0.74
6 with augmentation	conv48, conv48, pool2x2, drop25, conv96, conv96, pool2x2, drop25, conv192, conv192, pool2x2, drop25 flat, dense512, drop25, dense256, drop50, dense10	66.93	65.71	0.49	0.73

The same model stacks starting from basic two-layer models were repeated for CIFAR100 dataset as well with similar observations.

The performance of same models on CIFAR10 and CIFAR100 have huge differences because of the limited data available in CIFAR100. The data in CIFAR100 is also fine labeled which causes a problem when a shark can be categorized as a whale. The performance on CIFAR10 is significantly higher because the data is 10 times more than the data available in the CIFAR100 for every class. The results in Table IV and Table V proves the concept mentioned in the literature review that more layers mean different layer can concentrate on learning different features of an image which makes the model more efficient. Data augmentation has resulted in improvement in the validation accuracy for both the dataset but the training accuracy dropped. A few possible reasons for the drop in training accuracy is that augmentation makes training data more difficult then validation data. Augmentation also increases the size of training data by multiple times, thus the data becomes huge and the accuracy is checked for the entire dataset.

4.3 Automatic Image Captioning

There is no computerized way to detect the performance of this model as the captions generated are based on human inference. It is possible that the caption in the test dataset read "Thee men sitting in a park" and the model could predict the image as "Three man sitting on a bench in park". Both the statements are correct for a human but are different for a computer. One evaluation matric that was used in the training phase was the loss. The loss should have less value, unlike accuracy. The final model after a training of 24 hours, got a loss of 2.65 which is a great result. After 24 hours, the training was stopped because the improvement in the model was slow.

The model was able to generate perfect captions for many images as shown in Fig. 14 to Fig. 18. There were some minor errors as well in the prediction while some were totally off. For example in Fig. 18, the land is classified as water because of the blurry image that could be misinterpreted as water. In Fig. 15, the model is able to classify that a person is riding a skateboard but skateboard was mostly associated with a slide or a park in the training set, so the model generates the caption involving a slide. In this project, the output highly depends on what is present in the training set. For example, if a training set has never seen a whale, the model will never predict a caption for an image involving a whale. It will be able to predict the whale as a fish in the ocean as the training set will have fish.

There are several datasets available over the internet which could be used for the project but these datasets are so huge that they cannot be used for academic purpose. The model could be trained on multiple dataset too which will improve the model to a great extent as it would have data to learn from and more objects to classify.

5 Conclusion

We can say with proof that the deep learning models are not recommended for general data science classification or regression problems. The amount of resources that are needed for data preparation and training the model are far more costly than the advantages of using these neural networks. The neural networks are designed to deal with a level of complex data which cannot be solved using standard machine learning algorithms or have bad performance with these algorithms. Data science problems involving binary classification or regression can be better solved with models like random forest, decision tree, clustering, support vector machine or other machine learning models.

Convolutional neural networks are best recommended for high dimensional data like images and video. But the use cases of CNN are simply not limited to visual data as they can also be used for natural language processing, music recognition, acoustic modeling and several other applications. Convolutional networks are also not efficient for tasks other than image recognition when used without any other neural network. For example, in music recognition, CNN will require a recurrent network associated with it to remember the previous melodies. A recommended model for these problems is based on the type and size of data along with the output expected. If the data is huge and spatial, it is expected to have multiple pooling layers to extract the features of the data. It is beneficial to have pooling layer and dropout layer after at least two convolutional layers to make sure that they are used correctly and are beneficial. A dropout of 0.25 or 0.5 is the best-suited option in the model with 0.25 at the beginning for better results. A dropout higher than this could result in poor results. The size of the pool depends on the size of input data. For an input of 32x32 in CIFAR dataset, the pooling size of (2x2) is the best option to avoid data loss. These parameters are always inter-dependent on each other.

The recurrent neural network is recommended for natural language or linguistic data or sequence-based data. The architecture of these models depends on the prediction or the expected output. In the caption generation model, since previous words are used to predict the next word, the model returns sequences, which in this case are the previous words. The neural network also requires an embedding to convert the indexes into dense vectors. If the output of the RNN/LSTM has to go through a flatten and dense layer, it should also have an input_length, which is true in our case.

The project could be improved greatly but there are a few setbacks due to limited computation power availability. Several deep learning algorithms require a processing power that is not available on a normal laptop or a computer. A normal laptop with high-end configurations and GPU was only able to run one data science problem of all the problems and failed on others due to memory issues. One of the high-end machine on AWS also took more than 24 hours to train the model. This limited availability of resources stopped several improvements like batch processing, transfer learning, data stacking and bigger models. The possible future work in the project could be to improve on the model by increasing the neural network layer to be more precise to features in the data and to use more data. The data classification part of the project is an unexplored territory in the deep learning world and could have great potential. The CIFAR dataset becomes a base for several deep learning projects and could be used as a base for image captioning model too. The caption generation model has a huge scope for improvement. It would be an apt choice to use a combination of Flickr8K, Flickr30K and MS COCO dataset which together contains over a million images with a size of more than 16 GB.

6 References

[1] T. Kirubarajan, Y. Bar-Shalom, K. R. Pattipati and L. M. Loew, "Interacting segmentation and tracking of overlapping objects from an image sequence," in *Decision and Control, 1997., Proceedings of the 36th IEEE Conference on*, San Diego, 1997.

[2] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors.," 3 July 2012. [Online]. Available: http://arxiv.org/abs/1207.0580. [Accessed 15 November 2015].

[3] M. A. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.

[4] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, C. Suen, A. Coates, A. Maas, A. Hannun, B. Huval, T. Wang and S. Tandon, "UFLDL Tutorial," 7 January 2015. [Online]. Available: http://ufldl.stanford.edu/tutorial/. [Accessed 15 November 2015].

[5] I. Arel, D. Rose and T. Karnowski, "Deep Machine Learning - A New Frontier in Artificial Intelligence Research [Research Frontier]," *Computational Intelligence Magazine, IEEE,* vol. 5, no. 4, pp. 13-18, 2010.

[6] Y. LeCun, K. Kavukcuoglu and C. Farabet, "Convolutional Networks and Applications in Vision," *Computational Intelligence Magazine, IEEE ,* vol. 5, no. 4, pp. 13-18, 2010.

[7] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., 2012, pp. 1097-1105.

[8] T. Kang, "Using Neural Networks for Image Classification," 18 May 2015. [Online]. Available: http://scholarworks.sjsu.edu/etd_projects/395. [Accessed 15 November 2015].

[9] A. Krizhevsky, "The CIFAR-10 and CIFAR-100 datasets," 12 December 2013. [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html. [Accessed 2015 15 November].

[10] Numpy developers, "Numpy," Numpy developers, 23 October 2013. [Online]. Available: http://www.numpy.org/. [Accessed 15 November 2015].

[11] A. Dundar, "Convolutional Neural Networks," 13 January 2013. [Online]. Available: https://www.youtube.com/watch?v=n6hpQwq7Inw. [Accessed 15 November 2015].

[12] K. Murphy, Machine Learning: A Probabilistic Perspective, MIT, 2012.

[13] Y. Amit and P. Felzenszwalb, "Object Detection," in *Computer Vision, A Reference Guide*, Springer US, 2014, pp. 537-542.

[14] S. McCann and J. Reesman, "Object Detection using Convolutional Neural Networks," Stanford University, 2013.

[15] "Convolutional Layers - Keras Documentation", Keras.io, 2018. [Online]. Available: https://keras.io/layers/convolutional/. [Accessed: 09- May- 2018].

[16] "Image Preprocessing - Keras Documentation", Keras.io, 2018. [Online]. Available: https://keras.io/preprocessing/image/. [Accessed: 09- May- 2018].

[17] "Recurrent Layers - Keras Documentation", Keras.io, 2018. [Online]. Available at: https://keras.io/layers/recurrent/. [Accessed: 09- May- 2018].

[18] "Applications - Keras Documentation", Keras.io, 2018. [Online]. Available at: https://keras.io/applications/. [Accessed: 09- May- 2018].

[19] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks", *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673-2681, 1997.

[20] A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 664-676, 2017.

[21] "Recurrent neural networks and LSTM tutorial in Python and TensorFlow - Adventures in Machine Learning", *Adventures in Machine Learning*, 2018. [Online]. Available: http://adventuresinmachinelearning.com/recurrent-neural-networks-lstm-tutorial-tensorflow/. [Accessed: 09- May- 2018].

[22] "Get Started with Deep Learning Using the AWS Deep Learning AMI | Amazon Web Services*", Amazon Web Services*, 2018. [Online]. Available: https://aws.amazon.com/blogs/machine-learning/get-started-with-deep-learning-using-the-aws-deep-learning-ami/. [Accessed: 09- May- 2018].

[23] Scikit-learn.org. (2018). API Reference - *scikit-learn 0.19.1 documentation*. [online] Available at: http://scikit-learn.org/stable/modules/classes.html [Accessed 9 May 2018].

[24] Kim, W. Bukhari and M. Lee, "Feature Analysis of Unsupervised Learning for Multi-task Classification Using Convolutional Neural Network", *Neural Processing Letters*, 2017.

[25] A. Sharma, "Understanding Activation Functions in Neural Networks", *Medium*, 2017. [Online]. Available: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0. [Accessed: 09- May- 2018].

7 Appendix

7.1 Source Code

The source code for the project is available at https://github.com/RajatKabra.

7.2 Dataset

- The dataset for data science problems are available at https://github.com/RajatKabra.
- The CIFAR10 and CIFAR100 dataset are freely available at https://www.cs.toronto.edu/~kriz/cifar.html.
- The Flickr8K dataset can be requested from by filling a form http://nlp.cs.illinois.edu/HockenmaierGroup/Framing_Image_Description/KCCA.html.

7.2 Model Training and data preparation

The code and process is under the repository and section 2 of the report. The code needs access to Google Collaboratory and Amazon AWS account with GPU computation. A step by step guide to use AWS account for deep learning is available at https://aws.amazon.com/blogs/machine-learning/get-started-with-deep-learning-using-the-aws-deep-learning-ami/.

A complete guide to using Google Collaboratory is given here https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d.