A Project Report

on

OBJECT DETECTION USING CONVOLUTIONAL NEURAL NETWORKS FOR DISASTER RECOVERY

Submitted in partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

B.KALPANA (16FE1A0515)

A.RAMANA KUMAR (16FE1A0512)

G.BRAHMAIAH (16FE1A0536)

A.MALLIKARJUN (16FE1A0509)

Under the guidance of

Mr.S.DEVA KUMAR M.Tech, (Ph.D)
Assistant Professor
Department of Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VIGNAN'S LARA INSTITUTE OF TECHNOLOGY AND SCIENCE
(Affiliated to Jawaharlal Nehru Technological University Kakinada, Kakinada)
(An ISO 9001:2015 Certified Institution, Approved by AICTE)
Vadlamudi, Guntur Dist, Andhra Pradesh-522213

April - 2020.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VIGNAN'S LARA INSTITUTE OF TECHNOLOGY AND SCIENCE
(Affiliated to Jawaharlal Nehru Technological University Kakinada, Kakinada)
(An ISO 9001:2015 Certified Institution, Approved by AICTE)
Vadlamudi-522213

CERTIFICATE

This is to certify that the project report entitled "**OBJECT DETECTION USING CONVOLUTIONAL NEURAL NETWORKS FOR DISASTER RECOVERY**" is a bonafide work done by **B.KALPANA (16FE1A0515)**, **A.RAMANA KUMAR (16FE1A0512)**, **G.BRAHMAIAH (16FE1A0536)**, **A.MALLIKARJUN (16FE1A0509)** under my guidance and submitted in fulfillment of the requirements for the award of the degree of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING** from **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA, KAKINADA**. The work embodied in this project report is not submitted to any University or Institution for the award of any Degree or diploma.

Project Guide
Mr. S.DEVA KUMAR M.Tech,(Ph.D)
Assistant Professor

Head of the Department
Mr. T.V.VAMSI KRISHNA M.Tech,(Ph.D)
Assistant Professor

External Examiner

DECLARATION

We hereby declare that the project report entitled **"OBJECT DETECTION USING CONVOLUTIONAL NEURAL NETWORKS FOR DISASTER RECOVERY"** is a record of an original work done by us under the guidance of **Mr. S. DEVA KUMAR,** Assistant Professor of Computer Science and Engineering and this project report is submitted in the fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering. The results embodied in this project report are not submitted to any other University or Institute for the award of any Degree or Diploma.

	Project Members	**Signature**
Place: Vadlamudi	**B.Kalpana(**16FE1A0515)	_____
Date:	**A.Ramana Kumar(**16FE1A0512)	_____
	G.Brahmaiah(16FE1A0536)	_____
	A.Mallikarjun(16FE1A0509)	_____

ACKNOWLEDGEMENT

The satisfaction that accompanies with the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

We are grateful to **Mr. S. DEVA KUMAR**, Assistant professor, Department of Computer Science and Engineering for guiding through this project and for encouraging right from the beginning of the project till successful completion of the project. Every interaction with him was an inspiration.

We thank **Mr. T. V. VAMSI KRISHNA,** Assistant Professor & HOD**,** Department of Computer Science and Engineering for support and Valuable suggestions.

We also express our thanks to **Dr. K. PHANEENDRA KUMAR,** Principal, Vignan's Lara Institute of Technology & Science for providing the resources to carry out the project.

We also express our sincere thanks to our beloved **Chairman Dr. LAVU RATHAIAH** for providing support and stimulating environment for developing the project.

We also place our floral gratitude to all other teaching and lab technicians for their constant support and advice throughout the project.

Project Members

B.Kalpana (16FE1A0515)

A.Ramana Kumar (16FE1A0512)

G.Brahmaiah (16FE1A0536)

A.Mallikarjun (16FE1A0509)

TABLE OF CONTENTS

ABSTRACT

Accurate and timely access to data describing disaster impact and extent of damage is key to successful disaster management (a process that includes prevention, mitigation, preparedness, response, and recovery). Airborne data acquisition using helicopter and unmanned aerial vehicle (UAV) helps obtain a bird's-eye view of disaster affected areas. However, a major challenge to this approach is robustly processing a large amount of data to identify and map objects of interest on the ground in real-time. The current process is resource-intensive (must be carried out manually)and requires offline computing(through post-processing of aerial videos).This research introduces and evaluates a series of convolutional neural network (CNN) models for ground object detection from aerial views of disaster's aftermath. These models are capable of recognizing critical ground assets including building roofs (both damaged and undamaged), vehicles, vegetation, debris, and flooded areas. The dataset consists images of floods and earthquakes each up to thousand images. The output of the project shows the accuracy of the model.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

NN	Neural Network
CNN	Convolutional Neural Network
VGG	Visual Geometry Group
RELU	Rectified Linear Units
DL	Deep Learning

CHAPTER 1
INTRODUCTION

1.1 NATURAL DISASTER

A Natural Disaster is a major bad Event caused by the natural processes of the Earth. Examples of some natural disasters are Floods, Earthquakes, Tsunamis, Cyclones, Volcanic eruptions and some other Geological Processes. Once a natural disaster occurs it causes loss of lives, property damage and leaves some economic damage.

With the changing climate, the frequency and severity of natural disasters are also on the rise requiring people and governments to be better prepared and equipped to cope with the effects of such catastrophes. One cannot prevent a natural disaster but once it has occurred if we take immediate actions we can save number of human lives.

Timely retrieval and integration of disaster information is critical for effective disaster management. Previous research findings show that disaster detection systems have a few major problems, which includes observing occurrence of disaster in limited range. Real-time spatial information about disaster damage and risk is of paramount importance to designing appropriate mitigation strategies and response plans.

According to the United Nations Office for Disaster Risk Reduction (UNISDR),in the 10-year period ending in 2014, natural disasters have affected 1.7 billion people, claimed 700,000 lives, and cost 1.4 trillion Dollars in damages. On June 2013 Uttarakhand received heavy rainfall, massive Landslides due to the large flashfloods, it suffered maximum damage of houses and structures, killing more than 1000 people, sources claimed the death toll could be rise up to 5000. Besides, identifying and locating objects of interest is a central issue in urban search and rescue (USAR) missions.

Disaster detection has been one of the most active research areas in remote sensing today because saving human lives is our priority once a disaster occurred. Researchers have studied the effect of changes occurred due to disaster using sensors and simple image processing techniques such as image algebra. Previous research findings show that disaster detection systems have a few major problems, which includes observing occurrence of disaster in limited range.

EXAMPLES OF NATURAL DISASTERS

There are several Examples of natural disasters due to which many living beings and non-living beings are affected. Out of all some of them are discussed below which are as follows:

FLOOD

A Flood is an Overflow of water on normally dry ground. This is most commonly due to an overflowing river, a dam break, snowmelt, or heavy rainfall. The Kerala Flood in India was an example of flood that has destroyed people's houses. Floods often cause damage to homes and businesses if they are in the natural flood plains of rivers. While riverine flood damage can be eliminated by moving away from rivers and other bodies of water, people have traditionally lived and worked by rivers because the land is usually flat and fertile and because rivers provide easy travel and access to commerce and industry. Flooding may occur as an overflow of water from water bodies, such as a river, lake, or ocean, in which the water overtops or breaks levees, resulting in some of that water escaping its usual boundaries.

EARTHQUAKE

An earthquake is the sudden movement of the Earth's tectonic plates, resulting in shaking of the ground. This shaking can result in the damage of various structures such as buildings and further breakdown of the Earth's surface. The study of earthquakes is called seismology. Earthquakes are usually quite brief, but there may be many over a short time frame. The sudden release of tension in the tectonic plates sends waves of energy that travel through the Earth. Seismology studies the cause, frequency, type and size of earthquakes. There are large earthquakes and small earthquakes. Large earthquakes can take down buildings and cause death and injury. Earthquakes are measured using observations from seismographs.

VOLCANIC ERUPTION

A volcanic eruption occurs when hot materials from the Earth's interior are thrown out of a volcano. Lava, rocks, dust, and gas compounds are some of these "ejecta". Eruptions can come from side branches or from the top of the volcano. Some eruptions are terrible explosions that throw out huge amounts of rock and volcanic ash and can kill many people. Some are quiet outflows of hot lava. Several more complex types of volcanic eruptions have been described by volcanologists. These are often named after famous volcanoes where that type of eruption has been seen. Some volcanoes may show only one type of eruption during a period of activity, while others may show a range of types in a series.

HURRICANES

A hurricane is a large rotating storm with high speed winds that forms over warm waters in tropical areas. Hurricanes have sustained winds of at least 74 miles per hour and an area of low air pressure in the center called the eye. The scientific name for a hurricane is a tropical cyclone. Tropical cyclones go by different names in different places. In North America and the Caribbean they are called "hurricanes", in the Indian Ocean they are called "cyclones", and in Southeast Asia they are called "typhoons." Hurricanes form over the warm ocean water of the tropics. When warm moist air over the water rises, it is replaced by cooler air. The cooler air will then warm and start to rise. This cycle causes huge storm clouds to form.

TSUNAMIS

Tsunamis are giant waves caused by earthquakes or volcanic eruptions under the sea. Out in the depths of the ocean, tsunami waves do not dramatically increase in height. But as the waves travel inland, they build up to higher and higher heights as the depth of the ocean decreases. The speed of tsunami waves depends on ocean depth rather than the distance from the source of the wave. Tsunami waves may travel as fast as jet planes over deep waters, only slowing down when reaching shallow waters. While tsunamis are often referred to as tidal waves, this name is discouraged by oceanographers because tides have little to do with these giant waves. The most damaging tsunami on record before 2004 was the one that killed an estimated 40,000 people in 1782 following an earthquake in the South China Sea.

AVALANCHE

An avalanche (also called a snow slide) is an event that occurs when a cohesive slab of snow lying upon a weaker layer of snow fractures and slides down a steep slope. Avalanches are typically triggered in a starting zone from a mechanical failure in the snowpack (slab avalanche) when the forces of the snow exceed its strength but sometimes only with gradual widening (loose snow avalanche). After initiation, avalanches usually accelerate rapidly and grow in mass and volume as they entrain more snow. If the avalanche moves fast enough, some of the snow may mix with the air forming a powder snow avalanche, which is a type of gravity current. Slides of rocks or debris, behaving in a similar way to snow, are also referred to as avalanches (see rockslide). The remainder of this article refers to snow avalanches.

1.2 CLASSIFICATION

Classification is a technique where we categorize data into a given number of classes. The main goal of a classification problem is to identify the category/class to which a new data will fall under.

Image Classification is the task of assigning an input image one label from a fixed set of categories. This is one of the core problems in Computer Vision that, despite its simplicity, has a large variety of practical applications. Many other seemingly distinct Computer Vision tasks (such as object detection, segmentation) can be reduced to image classification.

TERMINOLOGIES IN CLASSIFICATION

- Classifier: An algorithm that maps the input data to a specific category.
- Classification model: A Classification model tries to draw some conclusion from the input values given for training. It will predict the class labels/categories for the new data.
- Feature: A feature is an individual measurable property of a phenomenon being observed.
- Binary Classification: Classification task with two possible outcomes. Eg: Gender Classification (Male/Female).
- Multi class classification: Classification with more than two classes. In multi class classification each sample is assigned to one and only one target label. Eg: An animal can be cat or dog but not both at the same time.
- Multi label classification: Classification task where each sample is mapped to a set of target labels (more than one class). Eg: A news article can be about sports, a person, and location at the same time.

The following are the steps involved in building a classification model:

- Initialize: To initialize the classifier to be used.
- Train the classifier: All classifiers in scikit-learn uses a fit (X, y) method to fit the model (training) for the given train data X and train label y.
- Predict the target: Given an unlabeled observation X, the predict (X) returns the predicted label y.
- Evaluate the classifier model.

1.3 CLASSIFICATION ALGORITHMS

The 7 most commonly used classification algorithms along with the python code: Logistic Regression, Naïve Bayes, Stochastic Gradient Descent, K-Nearest Neighbours, Decision Tree, Random Forest, and Support Vector Machine.

LOGISTIC REGRESSION

Logistic regression is a machine learning algorithm for classification. In this algorithm, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

NAÏVE BAYES

Naive Bayes algorithm based on Bayes' theorem with the assumption of independence between every pair of features. Naive Bayes classifiers work well in many real-world situations such as document classification and spam filtering.

STOCHASTIC GRADIENT DESCENT

Stochastic gradient descent is a simple and very efficient approach to fit linear models. It is particularly useful when the number of samples is very large. It supports different loss functions and penalties for classification.

Advantages: Efficiency and ease of implementation.

Disadvantages: Requires a number of hyper-parameters and it is sensitive to feature scaling

K-NEAREST NEIGHBOURS

Neighbours based classification is a type of lazy learning as it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the k nearest neighbours of each point.

DECISION TREE

Given a data of attributes together with its classes, a decision tree produces a sequence of rules that can be used to classify the data.

RANDOM FOREST

Random forest classifier is a meta-estimator that fits a number of decision trees on various sub-samples of datasets and uses average to improve the predictive accuracy of the model and controls over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement.

SUPPORT VECTOR MACHINE

Support vector machine is a representation of the training data as points in space separated into categories by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

1.4 DEEP LEARNING

A deep learning technology is based on artificial neural networks (ANNs). These ANNs constantly receive learning algorithms and continuously growing amounts of data to increase the efficiency of training processes. The larger data volumes are, the more efficient this process is. The training process is called deep, because, with the time passing, a neural network covers a growing number of levels. The deeper this network penetrates, the higher its productivity is. DL algorithms can create new tasks to solve current ones.

ADVANTAGES OF DEEP LEARNING

Creating New Features

One of the main benefits of deep learning over various machine learning algorithms is its ability to generate new features from a limited series of features located in the training dataset. Therefore, deep learning algorithms can create new tasks to solve current ones. What does it mean for data scientists working in technological startups? Since deep learning can create features without a human intervention, data scientists can save much time on working with big data and relying on this technology. It allows them to use more complex sets of features in comparison with traditional machine learning software.

Advanced Analysis

Due to its improved data processing models, deep learning generates actionable results when solving data science tasks. While machine learning works only with labeled data, deep learning supports unsupervised learning techniques that allow the system to become smarter on its own. The capacity to determine the most important features allows deep learning to efficiently provide data scientists with concise and reliable analysis results.

DEEP LEARNING CHALLENGES

Deep learning is an approach that models human abstract thinking (or at least represents an attempt to approach it) rather than using it. However, this technology has a set of significant disadvantages despite all its benefits.

Continuous Input Data Management

In deep learning, a training process is based on analyzing large amounts of data. Although, fast-moving and streaming input data provide little time for ensuring an efficient training process. That is why data scientists have to adapt their deep learning algorithms in the way neural networks can handle large amounts of continuous input data.

Ensuring Conclusion Transparency

Another important disadvantage of deep learning software is that it is incapable of providing arguments why it has reached a certain conclusion. Unlike in case of traditional machine learning, you cannot follow an algorithm to find out why your system has decided that it is a cat on a picture, not a dog. To correct errors in DL algorithms, you have to revise the whole algorithm.

Resource-Demanding Technology

Deep learning is a quite resource-demanding technology. It requires more powerful GPUs, high-performance graphics processing units, large amounts of storage to train the models, etc. Furthermore, this technology needs more time to train in comparison with traditional machine learning.

Lack of flexibility

Despite the occasional warnings of AI taking over the world, deep learning algorithms are pretty simple in their nature. In order to solve a given problem, a deep learning network needs to be provided with data describing that specific problem, thus rendering the algorithm ineffective to solve any other problems. This is true no matter how similar they are to the original problem.

1.5 APPLICATIONS OF DEEP LEARNING

Automatic speech recognition

Large-scale automatic speech recognition is the first and most convincing successful case of deep learning.

LSTM RNNs can learn "Very Deep Learning" tasks that involve multi-second intervals containing speech events separated by thousands of discrete time steps, where one time step corresponds to about 10 ms. LSTM with forget gates is competitive with traditional speech recognizers on certain tasks.

The initial success in speech recognition was based on small-scale recognition tasks base on TIMIT. The data set contains 630 speakers from eight major dialects of American English, where each speaker reads 10 sentences. Its small size lets many configurations be tried. More importantly, the TIMIT task concerns phone-sequence recognition, which, unlike word-sequence recognition, allows weak phone bigram language models. This lets the strength of the acoustic modeling aspects of speech recognition be more easily analyzed.

Image recognition

A common evaluation set for image classification is the MNIST database data set. MNIST is composed of handwritten digits and includes 60,000 training examples and 10,000 test examples. As with TIMIT, its small size lets users test multiple configurations. A comprehensive list of results on this set is available. Deep learning- based image recognition has become "superhuman", producing more accurate results than human contestants. This first occurred in 2011.

Visual art processing

Closely related to the progress that has been made in image recognition is the increasing application of deep learning techniques to various visual art tasks. DNNs have proven themselves capable, for example, of

- Identifying the style period of a given painting.
- Neural Style Transfer - capturing the style of a given artwork and applying it in a visually pleasing manner to an arbitrary photograph or video, and
- Generating striking imagery based on random visual input fields.

Military

The United States Department of Defense applied deep learning to train robots in new tasks through observation.

Bioinformatics

An auto encoder ANN was used in bioinformatics, to predict gene ontology annotations and gene-function relationships.

In medical informatics, deep learning was used to predict sleep quality based on data from wearables and predictions of health complications from electronic health record data. Deep learning has also showed efficacy in healthcare.

DL is used mostly for the prediction of the sleep quality based on data from the wearables of the people. It can also show the various health issues of the humans based on the electronic health record data.

Self-Driving cars

Deep Learning is the force that is bringing autonomous driving to life. A million sets of data are fed to a system to build a model, to train the machines to learn, and then test the results in a safe environment. Data from cameras, sensors, geo-mapping is helping create succinct and sophisticated models to navigate through traffic, identify paths, signage, pedestrian-only routes, and real-time elements like traffic volume and road blockages.

Fraud-Detection

Another domain benefitting from Deep Learning is the banking and financial sector that is plagued with the task of fraud detection with money transactions going digital. Auto encoders in Keras and Tensorflow are being developed to detect credit card frauds saving billions of dollars of cost in recovery and insurance for financial institutions. Fraud prevention and detection are done based on identifying patterns in customer transactions and credit scores, identifying anomalous behavior and outliers.

1.6 NEURAL NETWORKS

NN DEFINITION

Neural Networks form the backbone of deep learning. The goal of a neural network is to find an approximation of an unknown function. It is formed by interconnected neurons. These neurons have weights, and bias which is updated during the network training depending upon the error. The activation function puts a nonlinear transformation to the linear combination which then generates the output. The combinations of the activated neurons give the output.

BASICS OF NEURAL NETWORK

Neuron:- Just like a neuron forms the basic element of our brain, a neuron forms the basic structure of a neural network. Just think of what we do when we get new information. When we get the information, we process it and then we generate an output. Similarly, in case of a neural network, a neuron receives an input, processes it and generates an output which is either sent to other neurons for further processing or it is the final output.

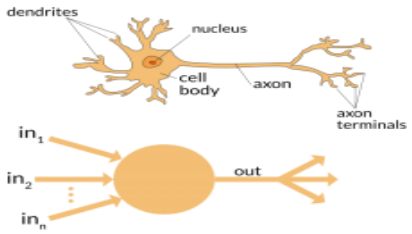

Fig 1.1 Neuron

Weights: - When input enters the neuron, it is multiplied by a weight. For example, if a neuron has two inputs, then each input will have has an associated weight assigned to it. We initialize the weights randomly and these weights are updated during the model training process. The neural network after training assigns a higher weight to the input it considers more important as compared to the ones which are considered less important. A weight of zero denotes that the particular feature is insignificant.

Bias: - In addition to the weights, another linear component is applied to the input, called as the bias. It is added to the result of weight multiplication to the input. The bias is basically added to change the range of the weight multiplied input. After adding the bias, the result would look like a*W1+bias. This is the final linear component of the input transformation.

Activation Function: - Once the linear component is applied to the input, a non- linear function is applied to it. This is done by applying the activation function to the linear combination. The activation function translates the input signals to output signals. The output after application of the activation function would look something like f (a*W1+b) where f() is the activation function.

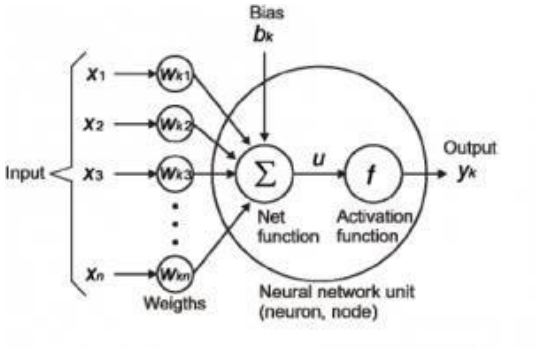

Fig 1.2 Neural Networks

Input / Output / Hidden Layer: - Simply as the name suggests the input layer is the one which receives the input and is essentially the first layer of the network. The output layer is the one which generates the output or is the final layer of the network. The processing layers are the hidden layers within the network. These hidden layers are the ones which perform specific tasks on the incoming data and pass on the output generated by them to the next layer. The input and output layers are the ones visible to us, while are the intermediate layers are hidden.

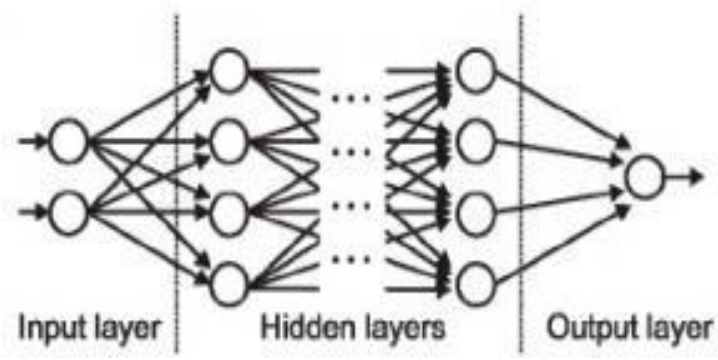

Fig 1.3 Layers in Neural Networks

Batch Normalization: - As a concept, batch normalization can be considered as a dam we have set as specific checkpoints in a river. This is done to ensure that distribution of data is the same as the next layer hoped to get. When we are training the neural network, the weights are changed after each step of gradient descent. This changes the how the shape of data is sent to the next layer.

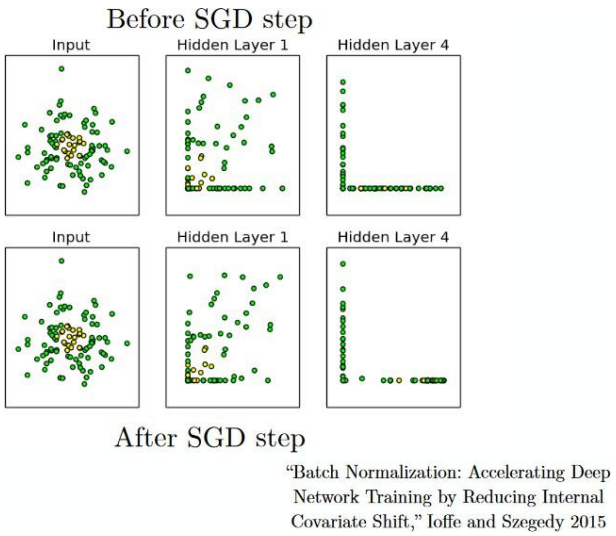

Fig 1.4 Batch Normalization

But the next layer was expecting the distribution similar to what it had previously seen. So we explicitly normalize the data before sending it.

Dropout: - Dropout is a regularization technique which prevents over-fitting of the network. As the name suggests, during training a certain number of neurons in the hidden layer is randomly dropped. This means that the training happens on several architectures of the neural network on different combinations of the neurons. You can think of drop out as an ensemble technique, where the output of multiple networks is then used to produce the final output.

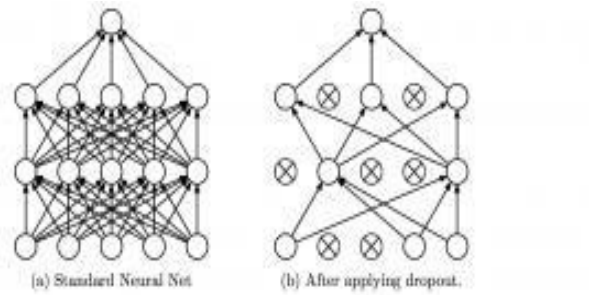

Fig 1.5 Dropouts in Neural Networks

PREDICTION LABELS

The created model will be trained with the dataset. The model extracts the features from the training dataset. Later it will predict the classes based on the given input. These Prediction labels play an important role in the deep neural networks.

Cost Function: - When we build a network, the network tries to predict the output as close as possible to the actual value. We measure this accuracy of the network using the cost/loss function. The cost or loss function tries to penalize the network when it makes errors.

- Our objective while running the network is to increase our prediction accuracy and to reduce the error, hence minimizing the cost function. The most optimized output is the one with least value of the cost or loss function.

- $C= 1/m \sum(y – a)^2$ where m is the number of training inputs, a is the predicted.

Learning Rate: - The learning rate is defined as the amount of minimization in the cost function in each iteration. In simple terms, the rate at which we descend towards the minima of the cost function is the learning rate. We should choose the learning rate very carefully since it should neither be very large that the optimal solution is missed and nor should be very low that it takes forever for the network to converge.

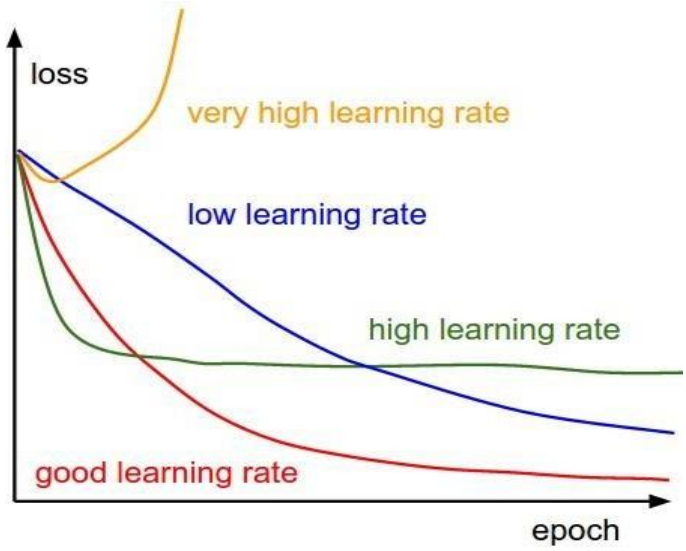

Fig 1.6 Learning Rate

1.7 TYPES OF NEURAL NETWORKS

Feed Forward Neural Network – Artificial Neuron:

This neural network is one of the simplest forms of ANN, where the data or the input travels in one direction. The data passes through the input nodes and exit on the output nodes. This neural network may or may not have the hidden layers. In simple words, it has a front propagated wave and no back propagation by using a classifying activation function usually. Below is a Single layer feed forward network. Here, the sum of the products of inputs and weights are calculated and fed to the output. The output is considered if it is above a certain value i.e. threshold (usually 0) and the neuron fires with an activated output (usually 1) and if it does not fire, the deactivated value is emitted (usually -1).

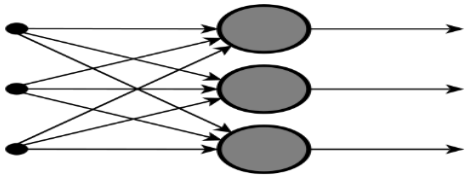

Fig 1.7 Feed Forward Neural Networks

Radial Basis Function Neural Network:

Radial basic functions consider the distance of a point with respect to the center. RBF functions have two layers, first where the features are combined with the Radial Basis Function in the inner layer and then the output of these features are taken into consideration while computing the same output in the next time-step which is basically a memory.

Below is a diagram which represents the distance calculating from the center to a point in the plane similar to a radius of the circle. Here, the distance measure used in Euclidean, other distance measures can also be used. The model depends on the maximum reach or the radius of the circle in classifying the points into different categories. If the point is in or around the radius, the likelihood of the new point begin classified into that class is high. There can be a transition while changing from one region to another and this can be controlled by the beta function.

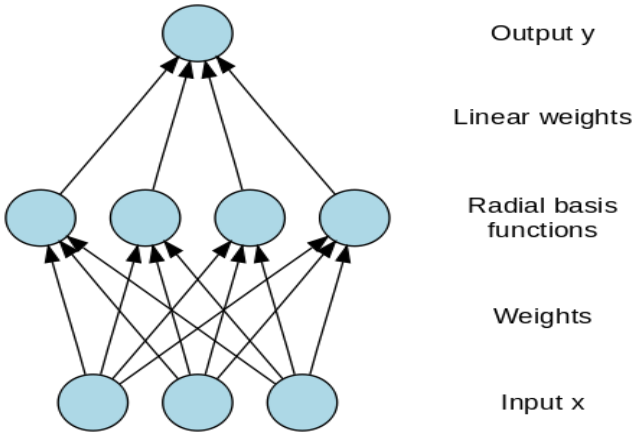

Fig 1.8 Radial Basis Function Neural Networks

Recurrent Neural Network (RNN) – Long Short Term Memory:

The Recurrent Neural Network works on the principle of saving the output of a layer and feeding this back to the input to help in predicting the outcome of the layer. Here, the first layer is formed similar to the feed forward neural network with the product of the sum of the weights and the features. The recurrent neural network process starts once this is computed, this means that from one time step to the next each neuron will remember some information it had in the previous time-step.

This makes each neuron act like a memory cell in performing computations. In this process, we need to let the neural network to work on the front propagation and remember what information it needs for later use. Here, if the prediction is wrong we use the learning rate or error correction to make small changes so that it will gradually work towards making the right prediction during the back propagation.

RNN have a **"**memory**"** which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

RNN converts the independent activations into dependent activations by providing the same weights and biases to all the layers, thus reducing the complexity of increasing parameters and memorizing each previous output by giving each output as input to the next hidden layer. This is how a basic Recurrent Neural Network looks like,

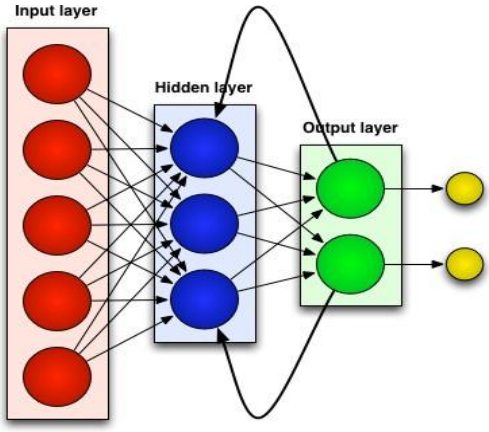

Fig 1.9 Recurrent Neural Networks

CNN (Convolutional neural network)

Convolutional neural networks are basically applied on image data. Suppose we have an input of size (28*28*3), If we use a normal neural network, there would be 2352(28*28*3) parameters. And as the size of the image increases the number of parameters becomes very large. We "convolve" the images to reduce the number of parameters (as shown above in filter definition). As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the output of that filter at every position. We will stack these activation maps along the depth dimension and produce the output volume.

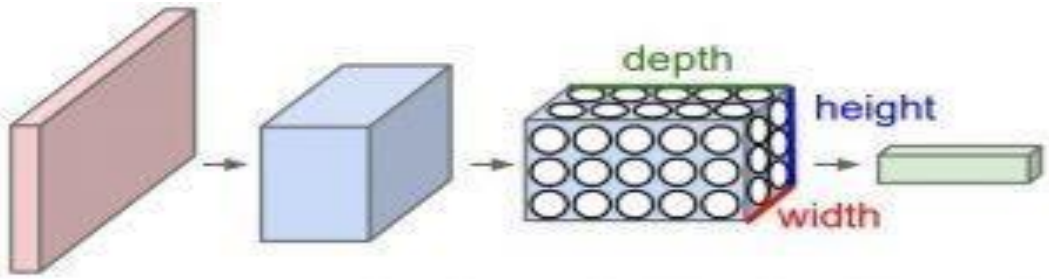

Fig 1.10 Convolutional Neural Networks

Modular Neural Network:

Modular Neural Networks have a collection of different networks working independently and contributing towards the output. Each neural network has a set of inputs which are unique compared to other networks constructing and performing sub-tasks. These networks do not interact or signal each other in accomplishing the tasks. The advantage of a modular neural network is that it breakdowns a large computational process into smaller components decreasing the complexity. This breakdown will help in decreasing the number of connections and negates the interaction of these networks with each other, which in turn will increase the computation speed. However, the processing time will depend on the number of neurons and their involvement in computing the results.

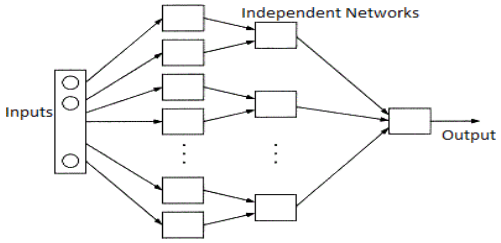

Fig 1.11 Modular Neural Networks

Multilayer Perceptron:

A multilayer perceptron has three or more layers. It is used to classify data that cannot be separated linearly. It is a type of artificial neural network that is fully connected. This is because every single node in a layer is connected to each node in the following layer.

A multilayer perceptron uses a nonlinear activation function (mainly hyperbolic tangent or logistic function). Here's what a multilayer perceptron looks like.

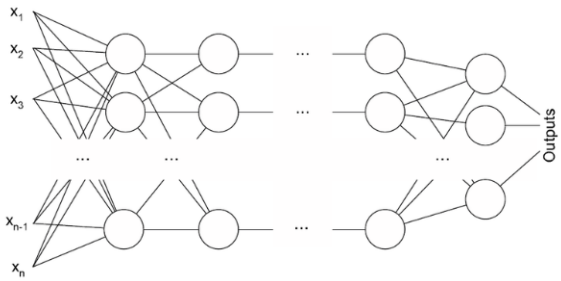

Fig 1.12 Multilayer Perceptron

1.8 CNN ARCHITECHTURES

A Convolutional Neural Network **(**CNN**,** or ConvNet**)** are a special kind of multi-layer neural networks, designed to recognize visual patterns directly from pixel images with minimal preprocessing.

LeNet-5

LeNet-5, a pioneering 7-level convolutional network by LeCun et al in 1998, that classifies digits, was applied by several banks to recognise hand-written numbers on checks (cheques) digitized in 32x32 pixel greyscale inputimages. The ability to process higher resolution images requires larger and more convolutional layers, so this technique is constrained by the availability of computing resources.

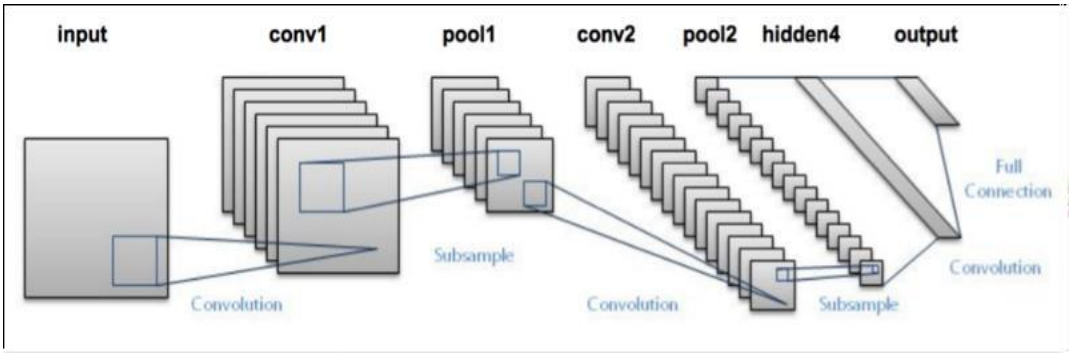

Fig 1.13 LeNet Architecture

AlexNet

The network had a very similar architecture as LeNet by Yann LeCun et al but was deeper, with more filters per layer, and with stacked convolutional layers. It consisted 11x11, 5x5,3x3, convolutions, max pooling, dropout, data augmentation, ReLU activations, SGD with momentum. It attached ReLU activations after every convolutional and fully-connected layer. AlexNet was trained for 6 days simultaneously on two Nvidia Geforce GTX 580 GPUs which is the reason for why their network is split into two pipelines. AlexNet was designed by the SuperVision group, consisting of Alex Krizhevsky, Geoffrey Hinton, and Ilya Sutskever. It contains 5 convolutional layers and 3 fully connected layers. Relu is applied after very convolutional and fully connected layer. Dropout is applied before the first and the second fully connected year. The image size in the following architecture chart should be 227 * 227 instead of 224 * 224.

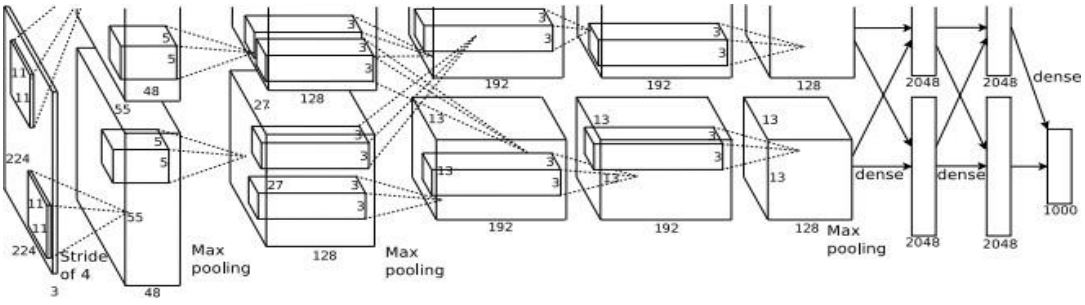

Fig 1.14 Alex Net Architecture

ZFNet

It achieved a top-5 error rate of 14.8% which is now already half of the prior mentioned non-neural error rate. It was mostly an achievement by tweaking the hyper- parameters of AlexNet while maintaining the same structure with additional Deep Learning elements as discussed earlier in this essay. It consisted of 5 shareable convolutional layers, max-pooling layers, dropout layers, and 3 fully connected layers. It used a 7×7 size filter and a decreased stride value in the first layer. The last layer of ZFNet is the softmax layer.

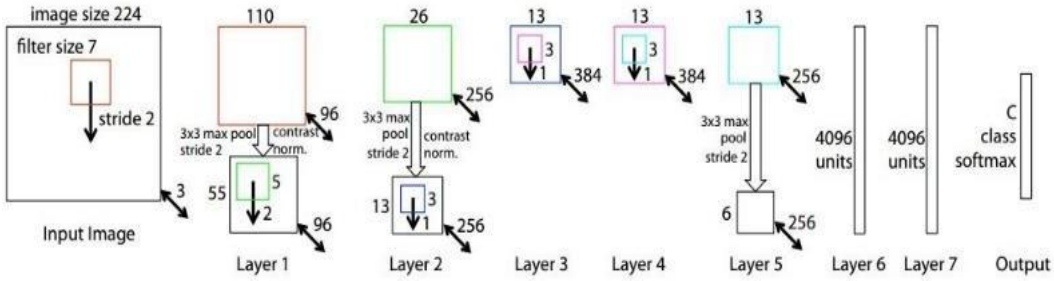

Fig 1.15 ZFNet Architecture

Google Net/Inception

It achieved a top-5 error rate of 6.67%! This was very close to human level performance which the organisers of the challenge were now forced to evaluate. As it turns out, this was actually rather hard to do and required some human training in order to beat Google Nets accuracy. After a few days of training, the human expert (Andrej Karpathy) was able to achieve a top-5 error rate of 5.1 %(single model) and 3.6 %(ensemble). The network used a CNN inspired by LeNet but implemented a novel element which is dubbed an inception module. It used batch normalization, image distortions and RMSprop. This module is based on several very small

convolutions in order to drastically reduce the number of parameters. Their architecture consisted of a 22 layer deep CNN but reduced the number of parameters from 60 million (AlexNet) to 4 million.

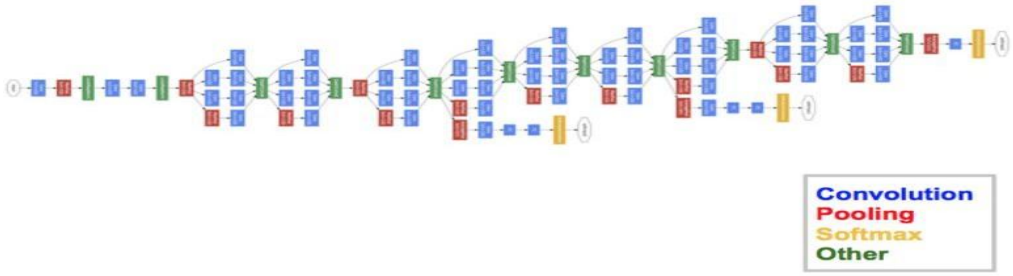

Fig 1.16 Google Net Architecture

VGGNet

VGGNet consists of 16 convolutional layers and is very appealing because of its very uniform architecture. Similar to AlexNet, only 3x3 convolutions, but lots of filters. Trained on 4 GPUs for 2–3 weeks. It is currently the most preferred choice in the community for extracting features from images. The weight configuration of the VGGNet is publicly available and has been used in many other applications and challenges as a baseline feature extractor.VGG stands for Visual Geometry Group. VGGNet is a neural network that performed very well in the Image Net Large Scale Visual Recognition Challenge (ILSVRC) in 2014. It scored first place on the image localization task and second place on the image classification task.

ADVANTAGES OF CNN

- Once trained, the predictions are pretty fast.
- With any number of inputs and layers, CNN can train.
- Neural networks work best with more data points.
- One of the powerful models in classification.

DISADVANTAGES OF CNN

- High Computational cost.
- They use to need a lot of Training data.

APPLICATIONS OF CNN

Image Recognition

CNNs are often used in image recognition systems. In 2012 an error rate of 0.23 percent on the MNIST database was reported. Another paper on using CNN for image classification reported that the learning process was "surprisingly fast"; in the same paper, the best published results as of 2011 were achieved in the MNIST database and the NORB database. Subsequently, a similar CNN called AlexNet won the Image Net Large Scale Visual Recognition Challenge 2012. Image recognition, in the context of machine vision, is the ability of software to identify objects, places, people, writing and actions in images. Computers can use machine vision technologies in combination with a camera and artificial intelligence software to achieve image recognition.

Video Analysis

Compared to image data domains, there is relatively little work on applying CNNs to video classification. Video is more complex than images since it has another (temporal) dimension. However, some extensions of CNNs into the video domain have been explored. One approach is to treat space and time as equivalent dimensions of the input and perform convolutions in both time and space. Another way is to fuse the features of two convolutional neural networks, one for the spatial and one for the temporal stream. Long short-term memory (LSTM) recurrent units are typically incorporated after the CNN to account for inter-frame or inter-clip dependencies. Unsupervised learning schemes for training spatio-temporal features have been introduced, based on Convolutional Gated Restricted Boltzmann Machines and Independent Subspace Analysis.

Decoding Facial Recognition

Facial recognition is broken down by a convolutional neural network into the following major components –

- Identifying every face in the picture.
- Focusing on each face despite external factors, such as light, angle, pose, etc.
- Identifying unique features.
- Comparing all the collected data with already existing data in the database to match a face with a name.

A similar process is followed for scene labeling as well.

Analyzing Documents

Convolutional neural networks can also be used for document analysis. This is not just useful for handwriting analysis, but also has a major stake in recognizers. For a machine to be able to scan an individual's writing, and then compare that to the wide database it has, it must execute almost a million commands a minute. It is said with the use of CNNs and newer models and algorithms, the error rate has been brought down to a minimum of 0.4% at a character level, though it's complete testing is yet to be widely seen.

Historic and Environmental Collections

CNNs are also used for more complex purposes such as natural history collections. These collections act as key players in documenting major parts of history such as biodiversity, evolution, habitat loss, biological invasion, and climate change.

Understanding Climate

CNNs can be used to play a major role in the fight against climate change, especially in understanding the reasons why we see such drastic changes and how we could experiment in curbing the effect. It is said that the data in such natural history collections can also provide greater social and scientific insights, but this would require skilled human resources such as researchers who can physically visit these types of repositories. There is a need for more manpower to carry out deeper experiments in this field.

CHAPTER 2
LITERATURE SURVEY

2.1 LITERATURE STUDY

Within the context of natural disasters, when communities participate in data collection and information exchange, new opportunities can emerge for better understanding of urban vulnerabilities, capacities, and risks, as well as creating data-driven methods for damage assessment and recovery planning. [1] Craglia et al used user-generated content (UGC) such as Facebook and Twitter to produce volunteered geographic information (VGI) maps. [2] To increase the accuracy of detection, machine learning is implemented to improve the efficiency of extracting feature. [3]

Object detection is being used in various other fields like defense, architecture, etc. But it is used the most for medical purposes. One of the examples is detecting tumor in the brain using deep neural network. [4] Cyclone track forecasting using artificial neural networks is shown.[5] Baker et al proposed a Monte-Carlo algorithm for using UAVs to conduct post disaster search, which was proven faster than the conventional sweeping search.[6] Evaluates the effectiveness of multilayer feed forward neural networks, radial basis neural networks, and Random Forests in earthquake damage.[7]

CNN can be trained as robust feature extractors from raw pixel values and at the same time, learn classifiers for object recognition tasks. [8] Radovic et al used transfer learning based on the you-only-look-once(YOLO)algorithm to detect airplanes on the ground from aerial views.[9]

One of the most famous practices in deep learning, CNN [10-12] is making detection getting a better result in a disaster such as an avalanche, earthquake, and landslide. NIPS paper on regional based convolution neural network is also referred for further comparison. [13]

Ren et al. [14] proposed Faster R-CNN, which adopts region proposal network (RPN) to reduce calculation time, and achieves 73.2% and 70.4% mean average precision (mAP) on PASCAL VOC [15] 2007 and 2012, respectively. CNN was initially studied for a handwritten character of the postal code.

Since the performance is promising and more efficient than prior study, CNN is then extensively used in image recognition such as object recognition tasks [16] from trained classifier of CNN which is robust to feature extractor from raw pixel value. Then, CNN is practiced for human pose estimation tasks [17].

Guirado et al employed Faster R-CNN to track and count the number of whales from satellite images. Previous work, however, has not systematically investigated and documented the problem of CNN-based aerial inspection in natural disasters.[18] In natural language processing, it has come to be applied to emotion analysis, text classification, translation and semantic segmentation task. [19]

Simonyan and Zisserman introduced VGGNet that has 11–19 layers and achieves better accuracy in image classification than AlexNet by increasing the number of network layers.[20] Szegedy et al proposed GoogLeNet, which includes inception modules that apply convolutional and max pooling at the same time, thus out performing VGGNet.[21]

A considerable amount of literature has been published on detection based on machine learning suggested hierarchical shape features in the bags-of-visual words setting to detect large-scale damage.[22] ResNet used residual network in its image classification architecture that can outperform an average human by 3.57%.[23]

The most common linear function which is used widely for the training of data is the logistic classifier. Logistic classifier consists of weight and biases which are used to tune the changes in the data. The weight decides the class of the point. [24] One cannot remove any layer arbitrarily. One has to keep in mind the architecture of the network [25].

Redmon et al. [26] introduced YOLO, which reaches 45 frames per second(FPS)with a 63.4% mAP on VOC2007.Liuetal.[27] proposed SSD based on Mobile Net achieving 76.8% mAP onVOC2007.Deep learning is an algorithm deepening the hierarchy of "Neural Network" which is one type of machine learning.

This neural network is an algorithm modelling nerve cells (neurons) of the brain of the living organism and has a long history of research starting from 1940, and Hubel and Wiesel publish a paper on cat's visual cortex in 1968 [28]. [29-31] are examples of neural inspired models.

2.2 EXISTING METHODS

Object recognition and classification is a rapidly growing field in the area of machine learning. In particular, object recognition is a key feature of image classification, and the commercial implications of this are vast. However, a major challenge to this approach is robustly processing a large amount of data to identify and map objects of interest on the ground in real-time. The current process is resource-intensive (must be carried out manually) and requires offline computing (through post-processing of aerial videos).

For instance, image classifiers will increasingly be used to:

- Replace passwords with facial recognition.
- Allow autonomous vehicles to detect obstructions.
- Identify geographical features from satellite imagery.

These are just a few of many examples of how image classification will ultimately shape the future of the world we live in.

Image classification requires vast amounts of supervised, labeled data. Acquiring this data requires human effort – meaning data is scarce and high- cost. To overcome this problem neural network came into existence. There are a lot of algorithms that people used for image classification before CNN became popular. People used to create features from images and then feed those features into some classification algorithm like SVM. Some algorithm also used the pixel level values of images as a feature vector too. To give an example, you could train a SVM with 784 features where each feature is the pixel value for a 28x28 image.

The problem of Image Classification goes like this: Given a set of images that are all labeled with a single category, we are asked to predict these categories for a novel set of test images and measure the accuracy of the predictions. There are a variety of challenges associated with this task, including viewpoint variation, scale variation, intra-class variation, image deformation, image occlusion, illumination conditions, background clutter etc.

2.3 LIMITATIONS OF EXISTING METHODS

- Since the final model is not so easy to see, we cannot do small calibrations to the model hence it’s tough to incorporate our business logic.
- A subtle issue ("disadvantage" if you like) with Naive-Bayes is that if you have no occurrences of a class label and a certain attribute value together (e.g. class="nice", shape="sphere") then the frequency-based probability estimate will be zero.
- Given Naive-Bayes' conditional independence assumption, when all the probabilities are multiplied you will get zero and this will affect the posterior probability estimate.
- Computer Vision researchers have come up with a data-driven approach to solve this.
- Instead of trying to specify what every one of the image categories of interest look like directly in code, they provide the computer with many examples of each image class and then develop learning algorithms that look at these examples and learn about the visual appearance of each class.
- In other words, they first accumulate a training dataset of labeled images, then feed it to the computer in order for it to get familiar with the data.
- This problem happens when we are drawing samples from a population and the drawn vectors are not fully representative of the population.
- Lagrange correction and other schemes have been proposed to avoid this undesirable situation.
- The accuracy is Low when trained on the large datasets.
- Among various methods of data collection, unmanned aerial vehicles (UAVs) or drones have drawn much attention due to their growing ubiquity, easy operation, large coverage area, and storage capacity.
- Compared to traditional methods of aerial data collection such as helicopter flyovers, UAV-based data collections more affordable, less resource-intensive, and can provide high-resolution imagery from difficult-to-reach places.
- The CNN models are trained on an in-house aerial video dataset (named Volan2018) that is created using web mining techniques.
- Volan2018 contains eight annotated aerial videos (65,580frames) collected by drone or helicopter from eight different locations in various hurricanes that struck the United States in 2017–2018.

CHAPTER 3

PROPOSED METHODOLOGY

3.1 INTRODUCTION

Artificial Neural Networks (ANN) is based on Biological Neural Networks. In ANN each connected node is referred as a Neuron. Here the ANN works same as a biological brain, it receives the signal, process it and can signal a neuron connected to it. Here as stated above each neuron process one output which is sent as input to the next neuron. A neuron can have multiple input and output connections. Each connection is assigned a weight. The neurons are typically organized into multiple layers, especially in deep learning.

CNNs can be thought of automatic feature extractors from the image. While if I use a algorithm with pixel vector I lose a lot of spatial interaction between pixels, a CNN effectively uses adjacent pixel information to effectively down sample the image first by convolution and then uses a prediction layer at the end.

3.2 BLOCK DIAGRAM

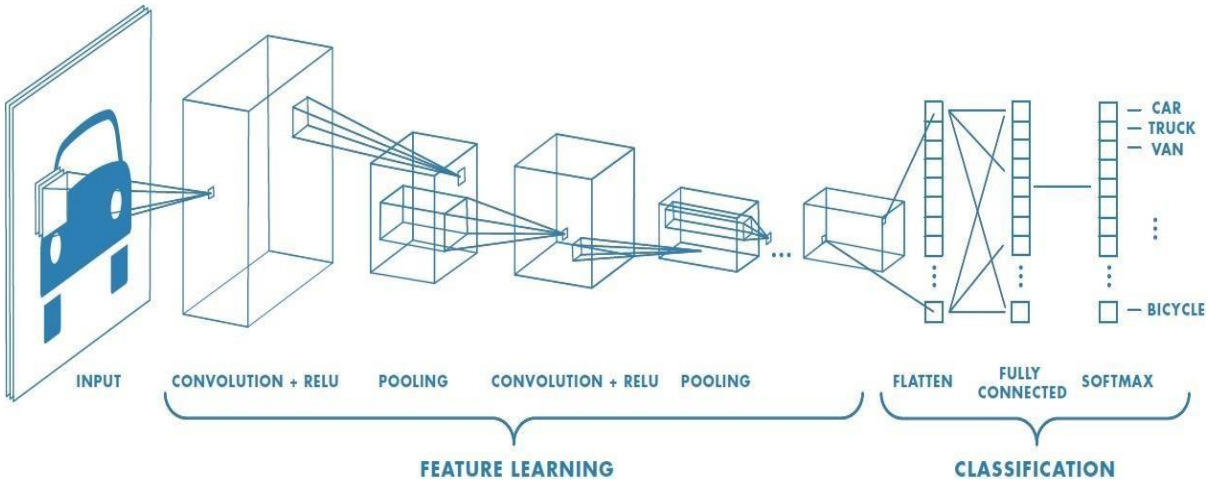

Fig 3.1 Block Diagram

As shown in the above figure, if you send an image as input it will undergo many steps and then send an output. Between input and output layers there are some other layers which are referred as hidden layers. Among all these Convolutional Neural Network (CNN) is the best algorithm used for implementing deep learning techniques. CNN is also a kind of ANN. CNN is

also known as ConvNet, consisting several layers named as convolutional layers, ReLu layer, pooling layer, and fully-connected layer.

3.3 PROPOSED SYSTEM

The Convolutional Neural Network model is trained using the Image dataset which can be done in two ways:

1. From Scratch
2. Using Pre-trained models

TRANSFER LEARNING

Transfer learning is a method where a model developed for a task is reused as the starting point for a model on a second task. Transfer learning makes use of the knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars can be used to some extent to recognize trucks.

PRE-TRAINING

When we train the network on a large dataset (for example: Image Net) , we train all the parameters of the neural network and therefore the model is learned. It may take hours on your GPU.

FINE TUNING

We can give the new dataset to fine tune the pre-trained CNN. Consider that the new dataset is almost similar to the original dataset used for pre-training. Since the new dataset is similar, the same weights can be used for extracting the features from the new dataset.

- If the new dataset is very small, it's better to train only the final layers of the network to avoid over fitting, keeping all other layers fixed. So remove the final layers of the pre-trained network. Add new layers . Retrain only the new layers.
- If the new dataset is very much large, retrain the whole network with initial weights from the pretrained model.

VGG-19

VGG stands for visual Geometry Group. VGG-19 is a convolutional neural network that is trained on more than a million images from the Image Net database. The network is 19 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.VGG-19 is a deep convolutional neural network consists of Convolutional layers, sub sampling layers and fully connected layers.

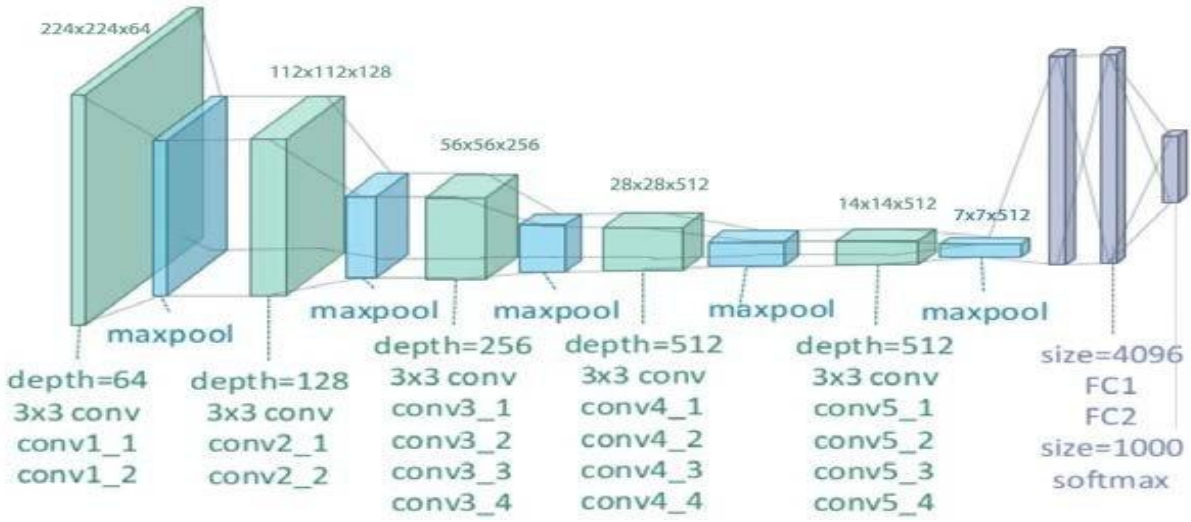

Fig 3.2 VGG-19 Architecture

VGG-16

VGGNet consists of 16 convolutional layers and is very appealing because of its very uniform architecture. Similar to AlexNet, only 3x3 convolutions, but lots of filters. Trained on 4 GPUs for 2–3 weeks. It is currently the most preferred choice in the community for extracting features from images. The weight configuration of the VGGNet is publicly available and has been used in many other applications and challenges as a baseline feature extractor.

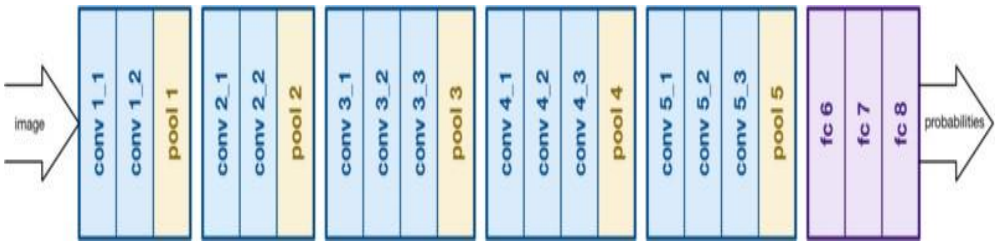

Fig 3.3 VGG-16 Architecture

RESNET-50

ResNet-50 is a convolutional neural network that is trained on more than a million images from the Image Net database. The network is 50 layers deep and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.

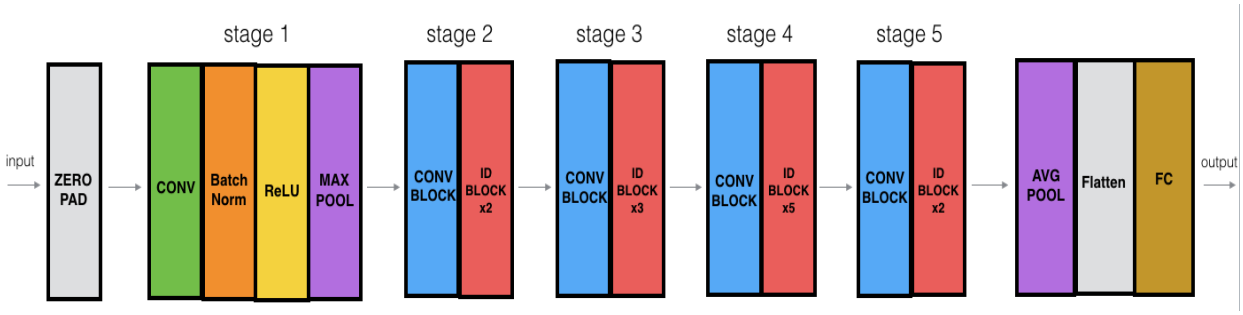

Fig 3.4 ResNet-50 Architecture

CONVOLUTIONAL LAYER

The convolution layer is the main building block of a convolutional neural network. The convolution layer comprises of a set of independent filters. Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

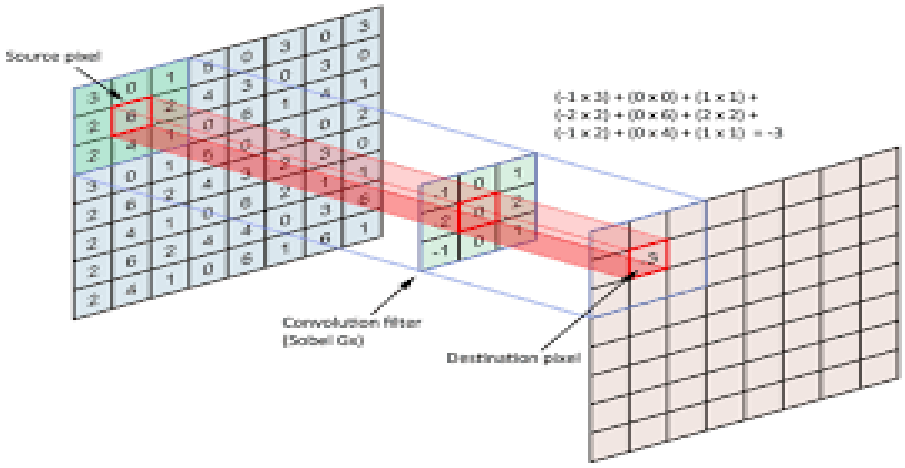

Fig 3.5 Convolution

FILTERS

A filter in a CNN is like a weight matrix with which we multiply a part of the input image to generate a convoluted output. Let's assume we have an image of size 28*28. We randomly assign a filter of size 3*3, which is then multiplied with different 3*3 sections of the image to form what is known as a convoluted output. The filter size is generally smaller than the original image size. The filter values are updated like weight values during back propagation for cost minimization. This filter is scrolled all over the image and the dot product is stored in a feature

map. Not only one filter is chosen similarly other filters are also taken from the input image and the same process is done to see the difference and the result.

Consider the below image. Here filter is a 3*3 matrix $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$ which is multiplied with

Each 3*3 section of the image to form the convolved feature.

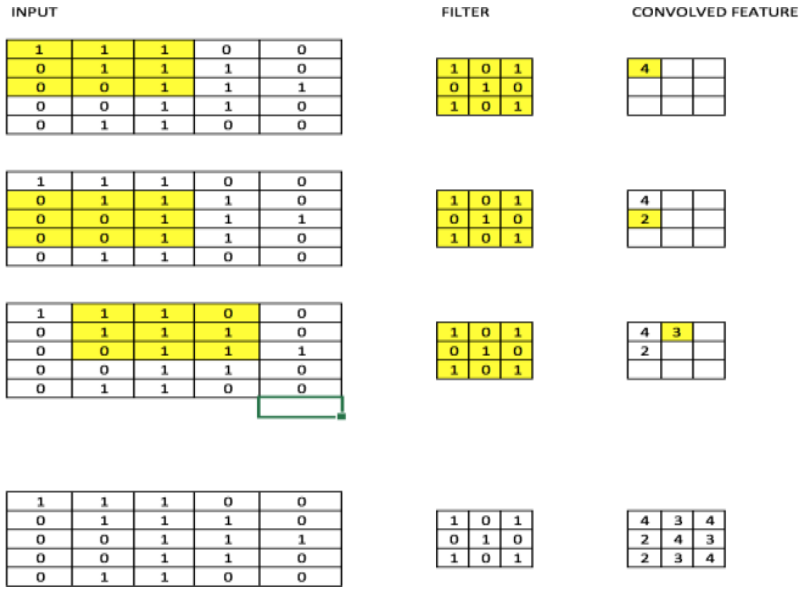

Fig 3.6 Convolution of image with filter

ACTIVATION FUNCTIONS

The most commonly applied activation functions are – Sigmoid, ReLU and softmax.

Sigmoid:-One of the most common activation functions used is Sigmoid. It is defined as:

$$\textbf{Sigmoid}(x) = 1/ (1+e^{-x})$$

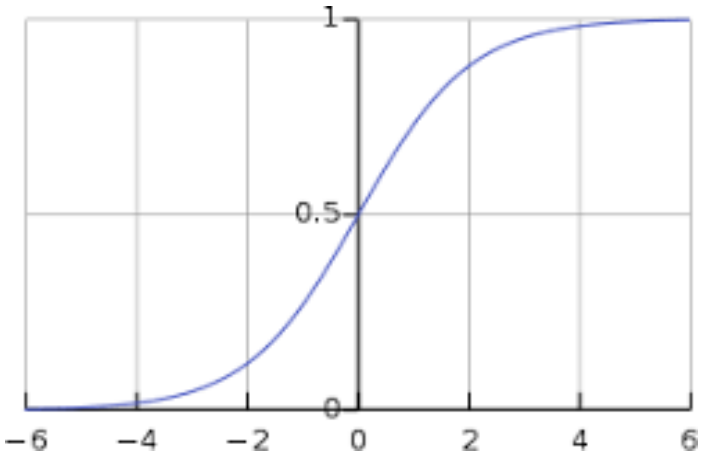

Fig 3.7 Sigmoid Activation Function

The sigmoid transformation generates a more smooth range of values between 0 and 1.We might need to observe the changes in the output with slight changes in the input values. Smooth curves allow us to do that and are hence preferred over step functions.

RELU (Rectified Linear Unit):-Instead of sigmoid, the recent networks prefer using ReLu activation functions for the hidden layers. The function is defined as:

$$F(x) = \max(x, 0).$$

The output of the function is X when X>0 and 0 for X<=0. The function looks like this:

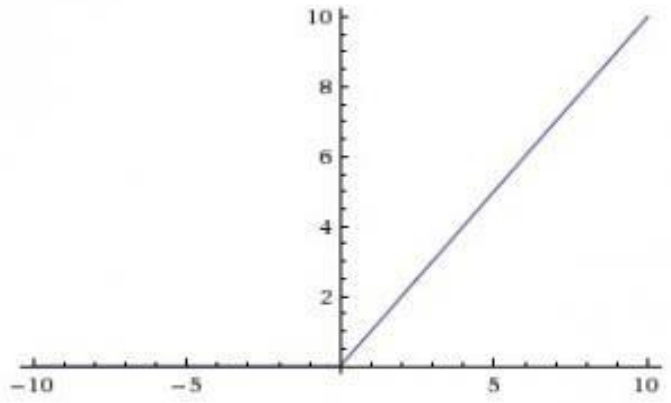

Fig 3.8 RELU Activation Function

The major benefit of using ReLU is that it has a constant derivative value for all inputs greater than 0. ReLU is linear (identity) for all positive values, and zero for all negative values. This means that:

- It's cheap to compute as there is no complicated math. The model can therefore take less time to train or run.
- It converges faster. Linearity means that the slope doesn't plateau, or "saturate," when *x* gets large. It doesn't have the vanishing gradient problem suffered by other activation functions like sigmoid or tanh.
- It's sparsely activated. Since ReLU is zero for all negative inputs, it's likely for any given unit to not activate at all. This is often desirable.

Softmax:-Softmax activation functions are normally used in the output layer for classification problems. It is similar to the sigmoid function, with the only difference being that the outputs are normalized to sum up to 1.

- The sigmoid function would work in case we have a binary output, however in case we have a multiclass classification problem, softmax makes it really easy to assign values to each class which can be easily interpreted as probabilities.
- It's very easy to see it this way – Suppose you're trying to identify a 6 which might also look a bit like 8. The function would assign values to each number as below. We can easily see that the highest probability is assigned to 6, with the next highest assigned to 8 and so on…

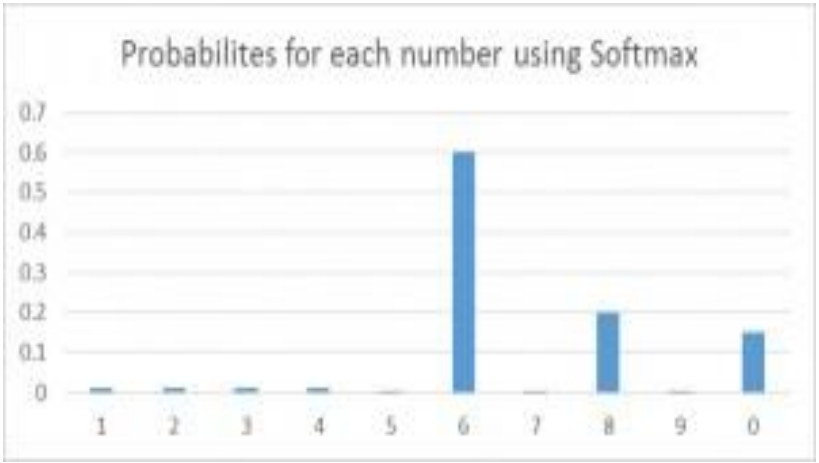

Fig 3.9 Softmax Activation Function

POOLING LAYER

It is common to periodically introduce pooling layers in between the convolution layers. This is basically done to reduce a number of parameters and prevent over-fitting. The most common type of pooling is a pooling layer of filter size (2, 2) using the MAX operation.

Pooling layer is also called as sub sampling layer. Pooling is of three types: Max. Pooling, Min. Pooling and Avg. Pooling. This pooling layer reduces the size of the image to perform Max. Pooling on the image firstly select one stride. Next scroll the stride over the filtered image and from that select one Max. Value and store it in the feature map. By performing this operation the dimensionality of the image reduces. Similarly to perform Min. or Avg. Pooling the same process is done.

- Pick a window size (usually 2 or 3).
- Pick a stride (usually 2).
- Walk your window across your filtered images.
- From each window, take the maximum value.

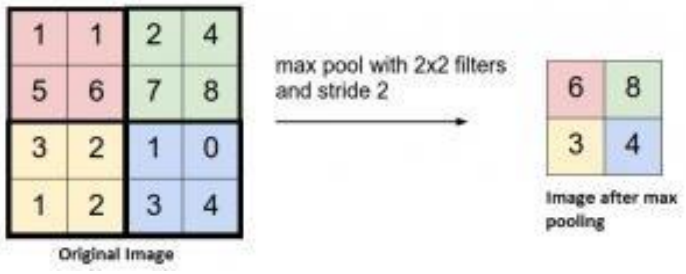

Fig 3.10 Pooling of an image

FULLY CONNECTED LAYER

After multiple layers of convolution and padding, we would need the output in the form of a class. The convolution and pooling layers would only be able to extract features and reduce the number of parameters from the original images. However, to generate the final output we need to apply a fully connected layer to generate an output equal to the number of classes we need. It becomes tough to reach that number with just the convolution layers. Convolution layers generate 3D activation maps while we just need the output as whether or not an image belongs to a particular class. The output layer has a loss function like categorical cross-entropy, to compute the error in prediction. Once the forward pass is complete the back propagation begins to update the weight and biases for error and loss reduction.

PUTTING IT ALL TOGETHER

- We pass an input image to the first convolutional layer. The convoluted output is obtained as an activation map. The filters applied in the convolution layer extract relevant features from the input image to pass further.
- Each filter shall give a different feature to aid the correct class prediction. In case we need to retain the size of the image, we use same padding (zero padding), otherwise valid padding is used since it helps to reduce the number of features.
- Pooling layers are then added to further reduce the number of parameters.
- Several convolution and pooling layers are added before the prediction is made. Convolutional layer help in extracting features. As we go deeper in the network more specific features are extracted as compared to a shallow network where the features extracted are more generic.
- The output layer in a CNN as mentioned previously is a fully connected layer, where the input from the other layers is flattened and sent so as the transform the output into the number of classes as desired by the network.
- The output is then generated through the output layer and is compared to the output layer for error generation. A loss function is defined in the fully connected output layer to compute the mean square loss. The gradient of error is then calculated.
- The error is then back propagated to update the filter(weights) and bias values.
- One training cycle is completed in a single forward and backward pass.

3.4 HARDWARE REQUIREMENTS

- 4GB RAM
- INTEL I3 PROCESSOR

3.5 SOFTWARE REQUIREMENTS

PYTHON

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It has a wide range of applications from Web development (like: Django and Bottle), Scientific and mathematical computing (Orange, SciPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D).

Reason for increasing popularity

- Emphasis on code readability**,** shorter codes, ease of writing
- Programmers can express logical concepts in fewer lines of code in comparison to languages such as C++ or Java.
- Python supports multiple programming paradigms, like object-oriented, imperative and functional programming or procedural.
- There exists an inbuilt function for almost all of the frequently used concepts.
- Philosophy is “Simplicity is the best”.

KERAS

Keras is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. It was developed by François Cholet, a Google engineer.

Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend. So Keras is high-level API wrapper for the low-level API, capable of running on top of Tensor Flow, CNTK, or Theano. Keras doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function. Keras doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

TENSORFLOW

Tensor Flow is an open source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. It is a standard expectation in the industry to have experience in Tensor Flow to work in machine learning.

Tensor Flow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open source license on November 9, 2015.

CHAPTER 4

SYSTEM DESIGN

4.1 DATA FLOW DIAGRAM

A data-flow diagram is a way of representing a flow of a data of a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow; there are no decision rules and no loops.

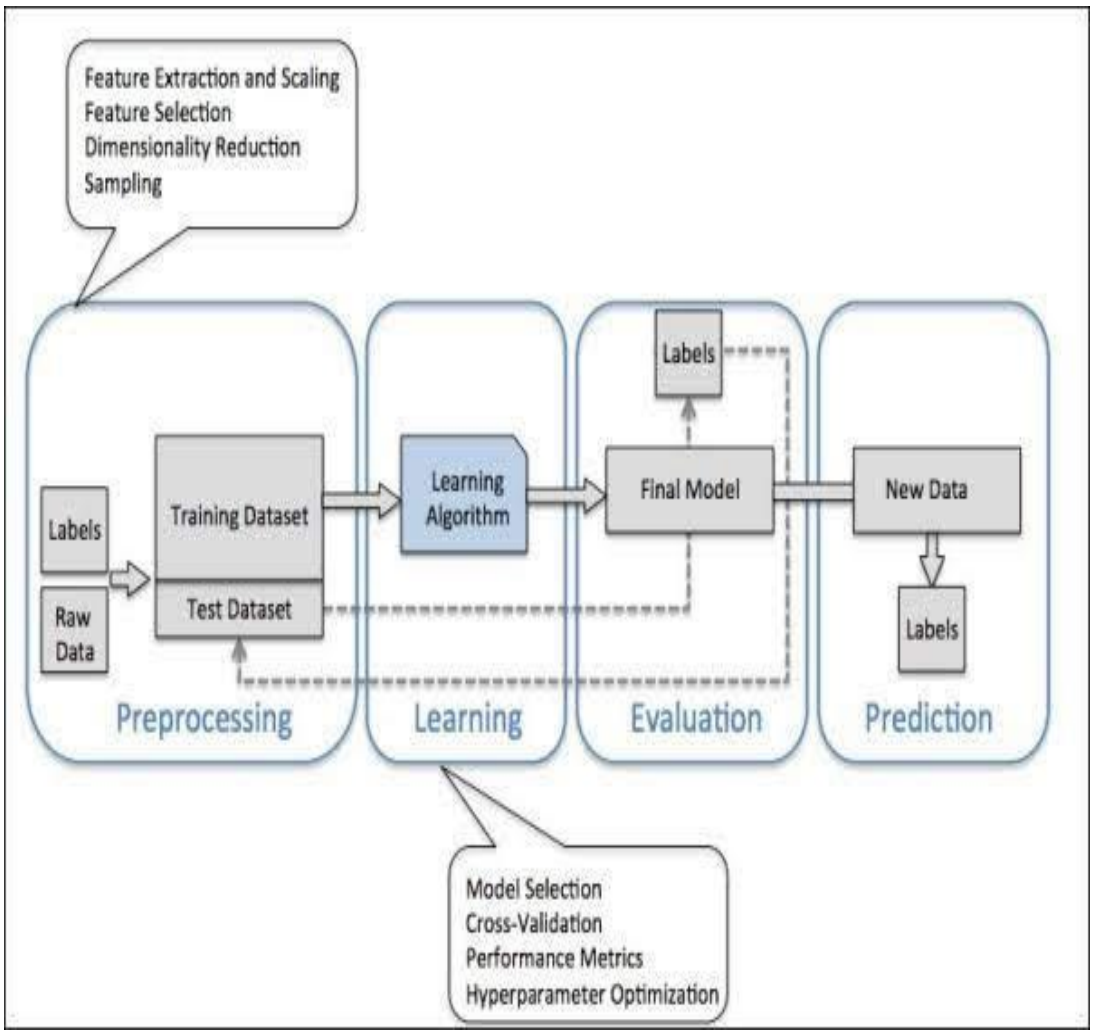

Fig 4.1 Data Flow Diagram

4.2 UML DIAGRAMS

4.2.1 CLASS DIAGRAM

A class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.

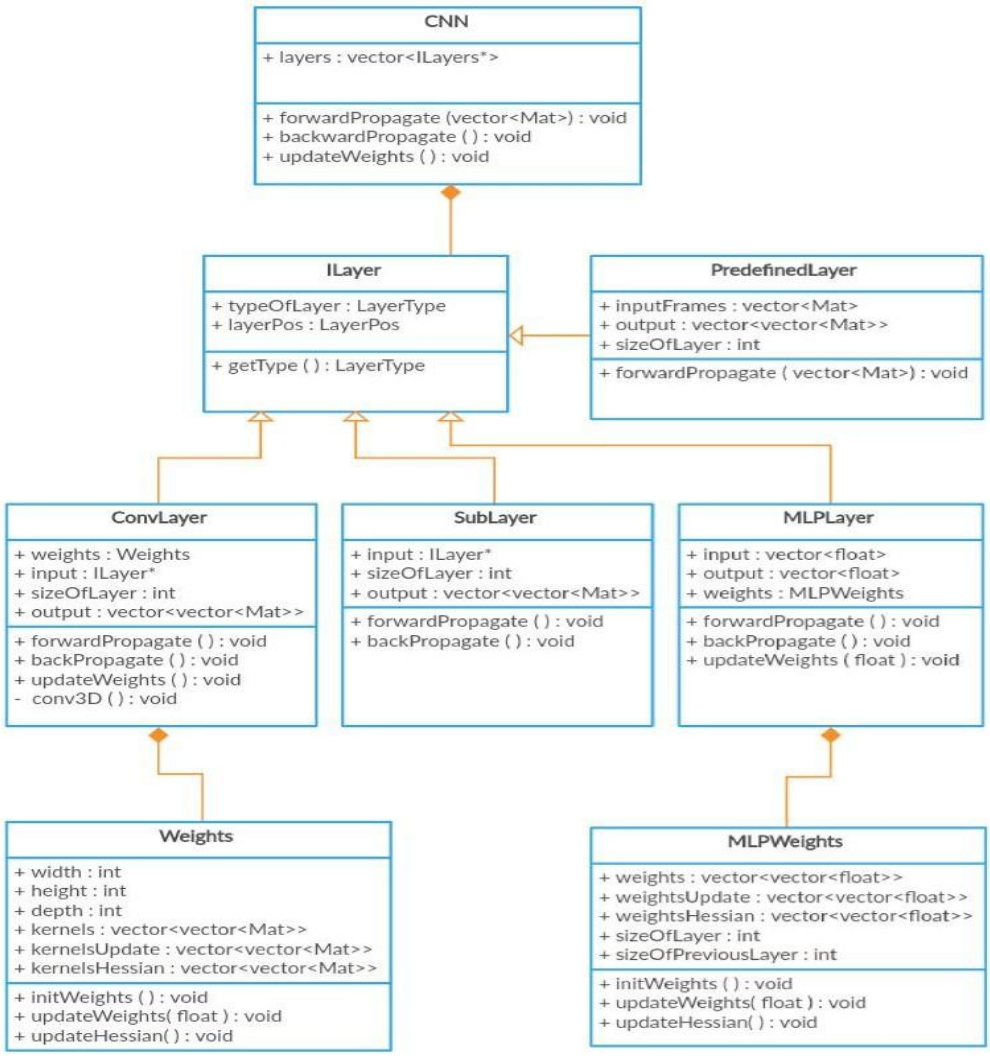

Fig 4.2.1 Class Diagram

4.2.2 USECASE DIAGRAM

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

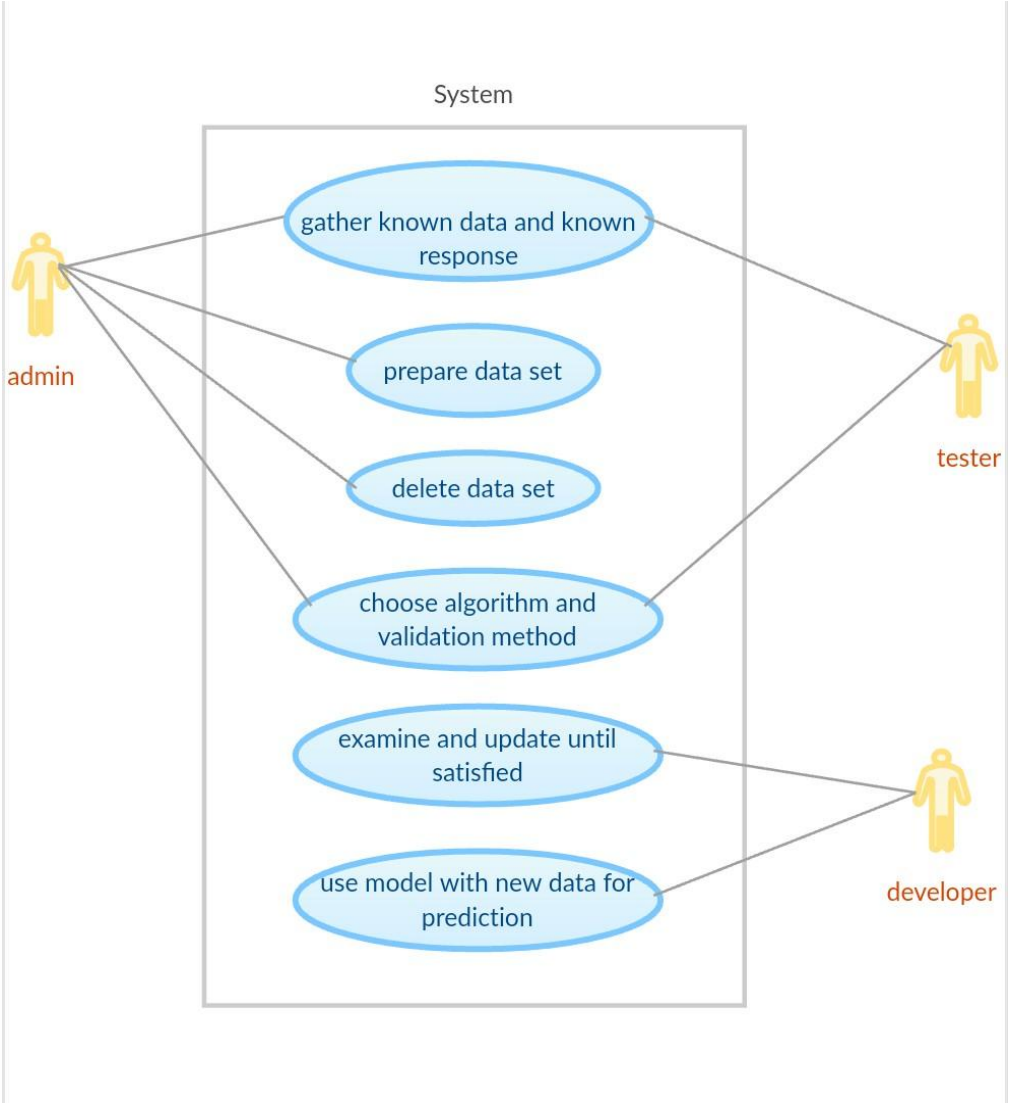

Fig 4.2.2 Use Case Diagram

4.2.3 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

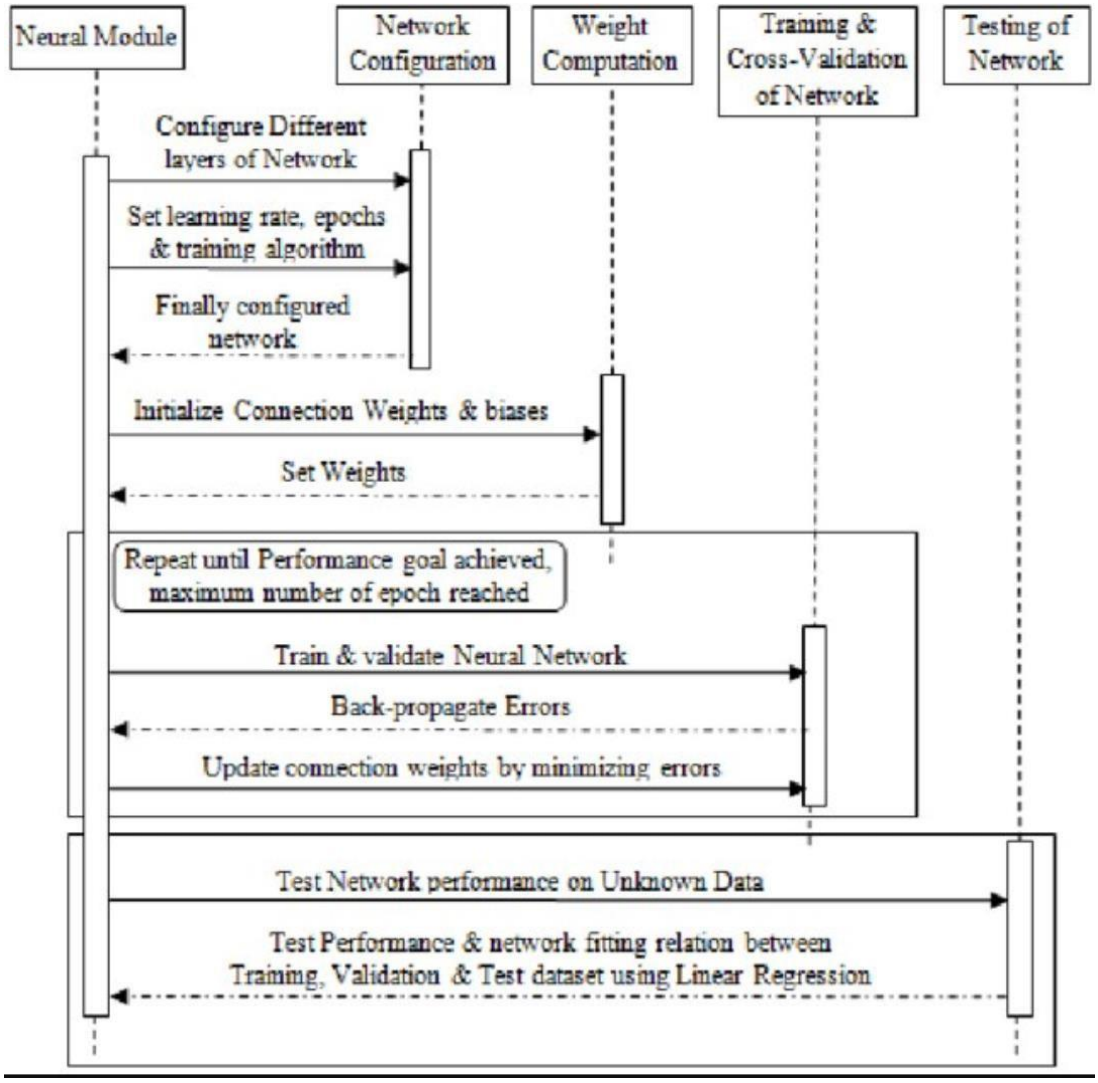

Fig 4.2.3 Sequence Diagram

4.2.4 STATE CHART DIAGRAM

A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction.

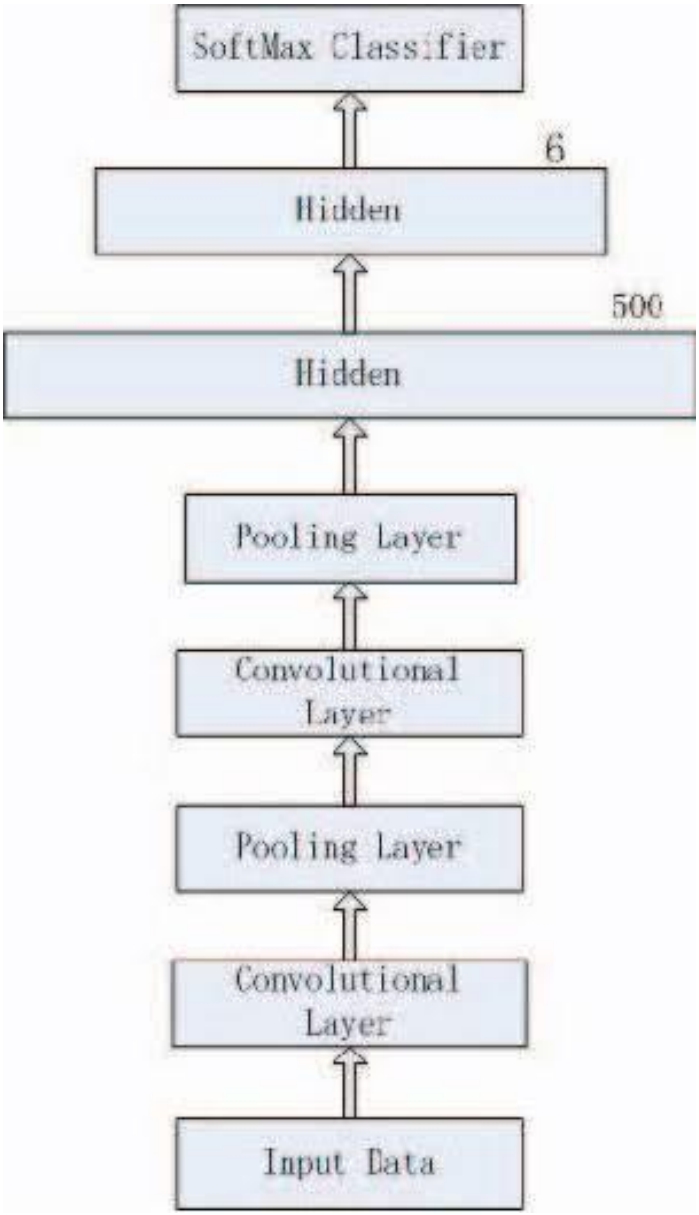

Fig 4.2.4 State Chart Diagram

4.2.5 ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

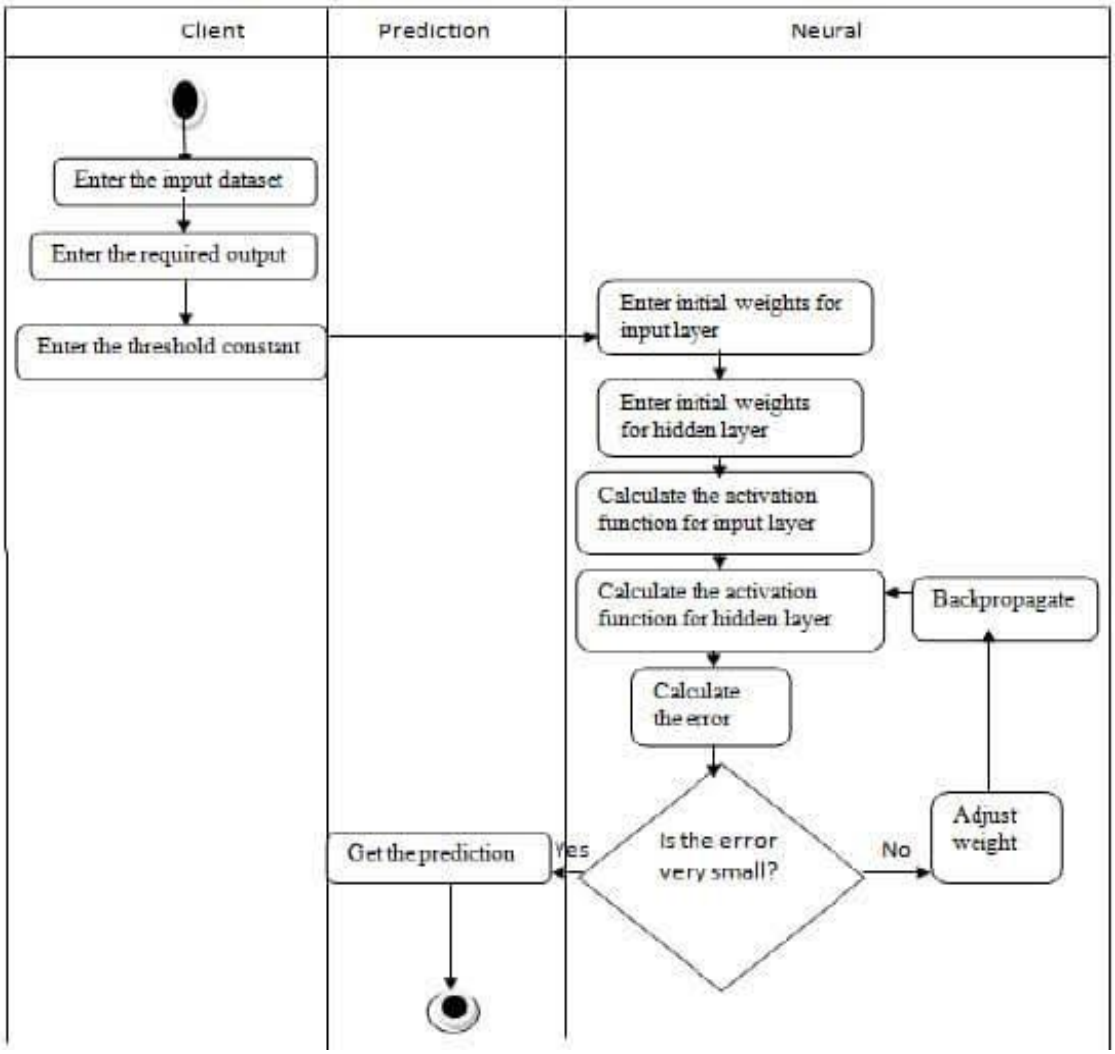

Fig 4.2.5 Activity Diagram

4.2.6 OBJECT DIAGRAM

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams.

Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment. Object diagrams are used to render a set of objects and their relationships as an instance.

The purposes of object diagrams are similar to class diagrams. The difference is that a class diagram represents an abstract model consisting of classes and their relationships. However, an object diagram represents an instance at a particular moment, which is concrete in nature.

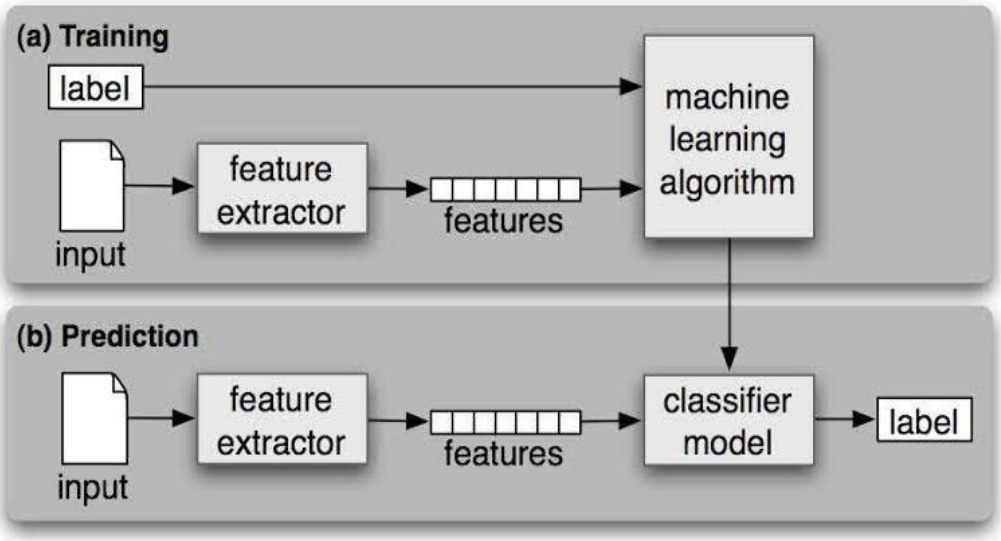

Fig 4.2.6 Object Diagram

CHAPTER 5
TESTING

Software Testing is a process of executing the application with intent to find any software bugs. It is used to check whether the application met its expectations and all the functionalities of the application are working. The final goal of testing is to check whether the application is behaving in the way it is supposed to under specified conditions. All aspects of the code are examined to check the quality of the application. The primary purpose of testing is to detect software failures so that defects may be uncovered and corrected. The test cases are designed in such way that scope of finding the bugs is maximum.

5.1 TESTING LEVELS

There are various testing levels based on the specificity of the test.

Unit Testing: - Unit testing refers to tests conducted on a section of code in order to verify the functionality of that piece of code. This is done at the function level.

Integration Testing: - Integration testing is any type of software testing that seeks to verify the interfaces between components of a software design. Its primary purpose is to expose the defects associated with the interfacing of modules.

System Testing: - System testing tests a completely integrated system to verify that the system meets its requirements.

Acceptance Testing: - Acceptance testing tests the readiness of application, satisfying all requirements.

Performance Testing: - Performance testing is the process of determining the speed or effectiveness of a computer, network, software program or devices such as response time or millions of instructions per second etc.

5.2 SYSTEM TEST CASES

A test case is a set of test data, preconditions, expected results and post conditions, developed for a test scenario to verify compliance with a specific requirement. I have designed and executed a few test cases to check if the project meets the functional requirements.

TEST CONDITION	INPUT SPECIFICATION	OUTPUT SPECIFICATION	PASS/FAIL
Testing with the images in project dataset	Image of flood/earthquake	Predication label of input image and accuracy	PASS

Table 5.1 Test case for images present in dataset

TEST CONDITION	INPUT SPECIFICATION	OUTPUT SPECIFICATION	PASS/FAIL
Testing with other images	Image not present in dataset	Not flood/earthquake	PASS

Table 5.2 Test case for images not present in dataset

The testers need to focus on for the following:-

- Test with new data, rather than the original training data. If necessary, split your training set into two groups: one that does training, and one that does testing. Better, obtain and use fresh data if you are able.
- Understand the architecture of the network as a part of the testing process. Testers won't necessarily understand how the neural network was constructed, but need to understand whether it meets requirements. And based on the measurements that they are testing, they may have to recommend a radically different approach, or admit the software is just not capable of doing what it was asked to do with confidence.

CHAPTER 6
RESULTS

INPUT IMAGE

The CNN model is trained with the dataset. The dataset we considered for the training of CNN model consists of two classes. The class labels are flood and earthquake. Each class consists of up to thousand images.

The dataset used is image dataset which consists of real-time images of disaster's aftermath. This image dataset consists of 2423 images which are of size 224x224x3.The image dataset is divided into two classes:

1. Flood
2. Earthquake

Out of all 2423 images 1073 images belongs to flood and 1350 images belongs to Earthquake.

Fig 6.1 Flood Input Image

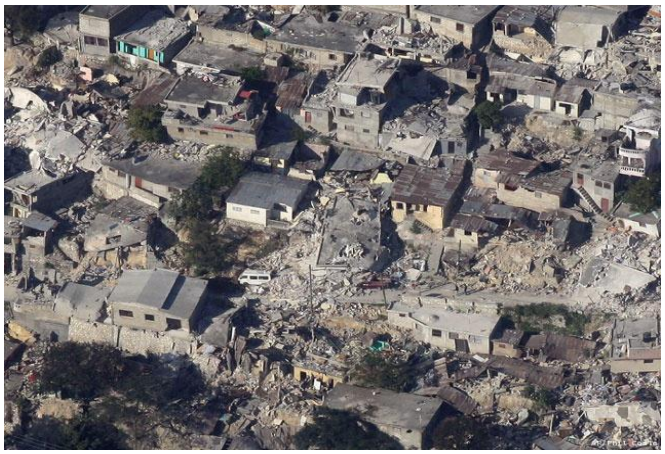

Fig 6.2 Earthquake Input Image

PREDICTION LABELS

The outcome of the model will be the prediction label of the input image and the accuracy of the model.

The accuracy can be defined as the percentage of correctly classified instances (TP + TN)/ (TP + TN + FP + FN). Where TP, FN, FP and TN represent the number of true positives, false negatives, false positives and true negatives, respectively.

ACCURACY/LOSS OF MODEL

VGG-19

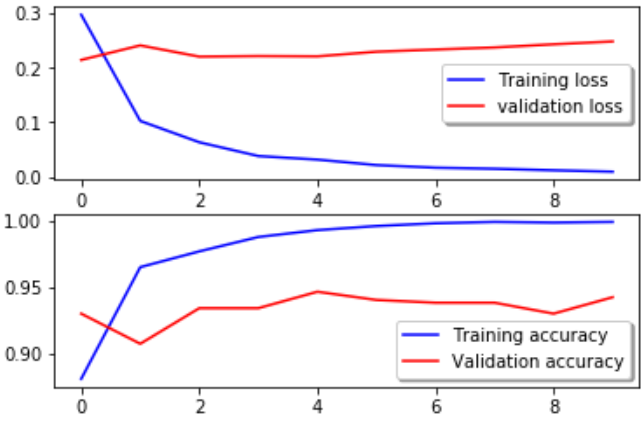

Fig 6.3 Accuracy/Loss of Vgg-19 model

VGG-16

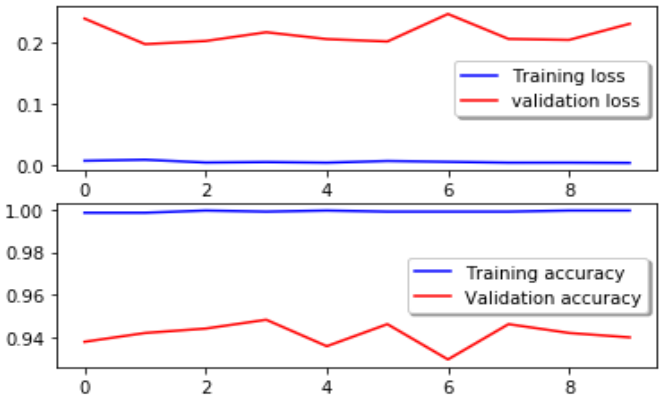

Fig 6.4 Accuracy/Loss of Vgg-16 model

RESNET-50

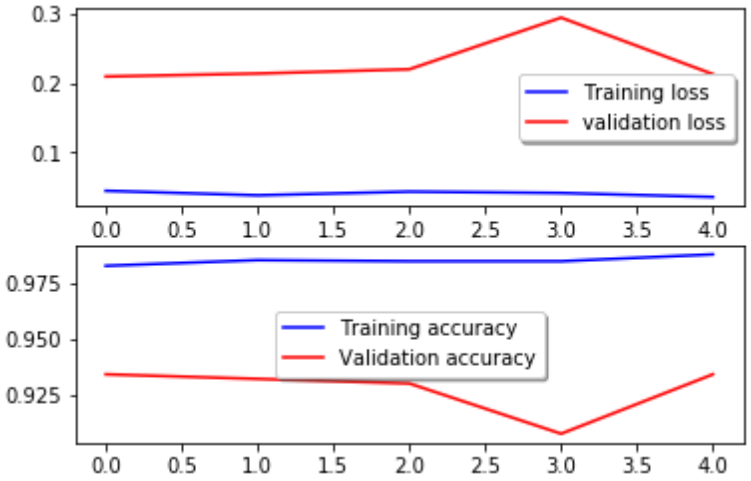

Fig 6.5 Accuracy/Loss of ResNet-50 model

SCRATCH

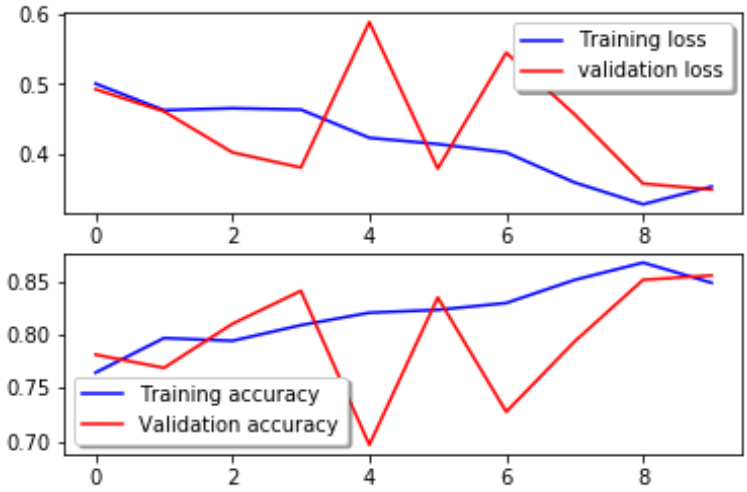

Fig 6.6 Accuracy/Loss of model

CONFUSION MATRIX

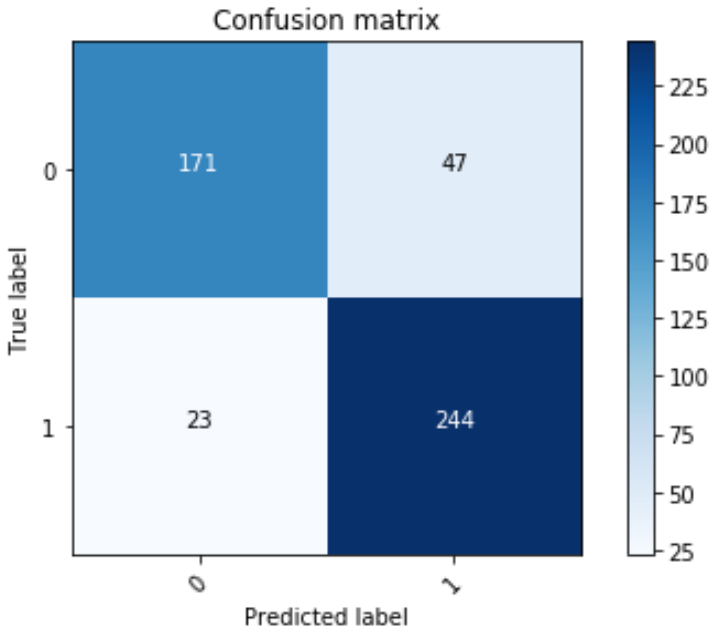

Fig 6.7 Confusion matrix

SCREENSHOTS OF OUTPUT

VGG-19

```
Train on 1938 samples, validate on 485 samples
Epoch 1/5
1938/1938 [==============================] - 26s 14ms/step - loss: 0.2192 - acc: 0.9164 - val_loss: 0.1967 - val_acc: 0.9443
Epoch 2/5
1938/1938 [==============================] - 17s 9ms/step - loss: 0.0961 - acc: 0.9639 - val_loss: 0.2110 - val_acc: 0.9464
Epoch 3/5
1938/1938 [==============================] - 17s 9ms/step - loss: 0.0466 - acc: 0.9871 - val_loss: 0.2273 - val_acc: 0.9505
Epoch 4/5
1938/1938 [==============================] - 17s 9ms/step - loss: 0.0467 - acc: 0.9861 - val_loss: 0.2211 - val_acc: 0.9340
Epoch 5/5
1938/1938 [==============================] - 17s 9ms/step - loss: 0.0257 - acc: 0.9928 - val_loss: 0.2237 - val_acc: 0.9299
Training time: -100.11394095420837
485/485 [==============================] - 5s 11ms/step
[INFO] loss=0.2237, accuracy: 92.9897%
```

Fig 6.8 Output of VGG-19 model

VGG-16

```
Train on 1938 samples, validate on 485 samples
Epoch 1/5
1938/1938 [==============================] - 14s 7ms/step - loss: 0.3028 - acc: 0.8875 - val_loss: 0.1659 - val_acc: 0.9320
Epoch 2/5
1938/1938 [==============================] - 6s 3ms/step - loss: 0.1171 - acc: 0.9551 - val_loss: 0.1972 - val_acc: 0.9278
Epoch 3/5
1938/1938 [==============================] - 6s 3ms/step - loss: 0.0664 - acc: 0.9773 - val_loss: 0.1597 - val_acc: 0.9402
Epoch 4/5
1938/1938 [==============================] - 6s 3ms/step - loss: 0.0497 - acc: 0.9850 - val_loss: 0.1677 - val_acc: 0.9361
Epoch 5/5
1938/1938 [==============================] - 6s 3ms/step - loss: 0.0352 - acc: 0.9912 - val_loss: 0.1650 - val_acc: 0.9361
Training time: -41.823803424835205
485/485 [==============================] - 2s 4ms/step
[INFO] loss=0.1650, accuracy: 93.6082%
```

Fig 6.9 Output of VGG-16 model

RESNET-50

```
Train on 1938 samples, validate on 485 samples
Epoch 1/5
1938/1938 [==============================] - 14s 7ms/step - loss: 0.2647 - acc: 0.8767 - val_loss: 0.4033 - val_acc: 0.8680
Epoch 2/5
1938/1938 [==============================] - 5s 3ms/step - loss: 0.1365 - acc: 0.9458 - val_loss: 0.2299 - val_acc: 0.9093
Epoch 3/5
1938/1938 [==============================] - 5s 3ms/step - loss: 0.1058 - acc: 0.9649 - val_loss: 0.2139 - val_acc: 0.9155
Epoch 4/5
1938/1938 [==============================] - 5s 3ms/step - loss: 0.0896 - acc: 0.9690 - val_loss: 0.2048 - val_acc: 0.9196
Epoch 5/5
1938/1938 [==============================] - 5s 3ms/step - loss: 0.0881 - acc: 0.9680 - val_loss: 0.2225 - val_acc: 0.9155
Training time: -39.91569495201111
485/485 [==============================] - 2s 3ms/step
[INFO] loss=0.2225, accuracy: 91.5464%
```

Fig 6.10 Output of ResNet-50 model

SCRATCH

```
Epoch 1/10
 - 1s - loss: 0.4980 - acc: 0.7667 - val_loss: 0.4926 - val_acc: 0.7814
Epoch 2/10
 - 1s - loss: 0.4633 - acc: 0.7971 - val_loss: 0.4608 - val_acc: 0.7691
Epoch 3/10
 - 1s - loss: 0.4654 - acc: 0.7944 - val_loss: 0.4019 - val_acc: 0.8103
Epoch 4/10
 - 1s - loss: 0.4559 - acc: 0.8119 - val_loss: 0.3801 - val_acc: 0.8412
Epoch 5/10
 - 1s - loss: 0.4211 - acc: 0.8217 - val_loss: 0.5892 - val_acc: 0.6969
Epoch 6/10
 - 1s - loss: 0.4143 - acc: 0.8244 - val_loss: 0.3786 - val_acc: 0.8351
Epoch 7/10
 - 1s - loss: 0.4019 - acc: 0.8298 - val_loss: 0.5452 - val_acc: 0.7278

Epoch 00007: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
Epoch 8/10
 - 1s - loss: 0.3573 - acc: 0.8514 - val_loss: 0.4566 - val_acc: 0.7938
Epoch 9/10
 - 1s - loss: 0.3267 - acc: 0.8682 - val_loss: 0.3571 - val_acc: 0.8515
Epoch 10/10
 - 1s - loss: 0.3487 - acc: 0.8514 - val_loss: 0.3486 - val_acc: 0.8557
```

Fig 6.11 Output of model

ACCURACY TABLE

Model	Accuracy
VGG-19	92.98%
VGG-16	93.60%
RESNET-50	91.54%
SCRATCH	85.57%

Fig 6.12 Accuracy Table

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

Two classes of data floods and earthquake are chosen for testing and validation of image classification using deep learning. The convolutional neural network is used in VGG architecture for classification purpose. From the experiments, it is observed that the images are classified correctly and shows the effectiveness of deep learning algorithm. This data sets used both and training and testing purpose using CNN. It provides the accuracy rate 85%.

When we processed our dataset on VGG-19, we got an accuracy of 92.98%, this model consists of 19 layers and classify the images into categories. This model has an input size of 224×224. Here we had taken 2423 images of DISASTER dataset, but the VGG-19 has an ability to be processed on millions of images. Here we had taken 20 epochs to classify the images and run the process on a single CPU system.

When we processed our dataset on VGG-16, we got an accuracy of 93.60%, this model is also referred as Oxford Net and consists of 16 layers. This model has an input size of 224×224. Here we had taken 2423 images of DISASTER dataset, but the VGG-16 has an ability to be processed on millions of images. Here we had taken 20 epochs to classify the images and run the process on a single CPU system.

When we processed our dataset on ResNet50, we got an accuracy of 91.54%, This model has an input size of 224×224. Here we had taken 2423 images of DISASTER dataset, but the ResNet50 has an ability to be processed on millions of images. Here we had taken 20 epochs to classify the images and run the process on a single CPU system.

Based on all the models performance ResNet50 consumes a very less time compared to others, as we designed our model to help the people who are frequently affected by disasters, so that they can take necessary preventive measures to reduce the severity and damage.

FUTURE SCOPE

In future, we will run our model on Aerial Imagery dataset which contains more than thousands of images in order to improve the accuracy. The outcomes in this research are based on results that involve only sample datasets. It is necessary that additional datasets should be considered for the evaluation of different classification problems as the information growth in the recent technology is extending to heights beyond assumptions. Recent field of technology is growing and data are by nature dynamic.

Hence, further classification of the entire system needs to be implemented right from the scratch since the results from the old process have become obsolete. The scope of future work can deal with Incremental learning, which stores the existing model and processes the new incoming data more efficiently.

The image classification model can be enhanced in future, by including more low-level features such as shape and spatial location features apart from optimizing the weights and learning rate of the neural network. In this research work, while evaluating the fitness of an individual, the consideration is given only to the occurrence frequencies of an image in retrieval result and not the location of it.

So, the location of the image in retrieval result should be taken into account when evaluating the fitness of an individual in future works. When these enhancements are incorporated in the classification system, it would help further improve the performance and be useful for applications meant for the explicit classification system.

REFERENCES

[1]Yalong Pia, Nipun D.Nathb, Amir H.Behzadana, Convolutional neural networks for object detection in aerial imagery for disaster response and recovery.

[2]M.Craglia, F.Ostermann, L.Spinsanti, Digital Earth from vision to practice: making sense citizen-generated content, Int.J.Digital Earth 5 (5) (2012)398-416.

[3]D. Sulla-menashe, R. E. Kennedy, Z. Yang, J. Braaten, O. N. Krankina, and M. A. Friedl, "Remote Sensing of Environment Detecting forest disturbance in the Pacific Northwest from MODIS time series using temporal segmentation," Remote Sens. Environ., vol. 151, pp. 114–123, 2014.

[4]Rupal R. Agravat, and Mehul S. Raval, "Brain Tumor Segmentation,"Computer Society of India , December 2016, p. 31.

[5]R. Kovordanyi and C. Roy, "Cyclone track forecasting based on satellite images using artificial neural networks," ISPRS J. Photogramm. Remote Sens., vol. 64, pp. 513–521, 2009.

[6]C.A.B.Baker, S.Ramchurn, W.T.Teacy, N.R.Jennings, Planning search and rescue missions for UAV teams, Proceedings of the Twenty-second European Conference on ArtificialIntelligence, Netherlands, IOSPress, 2016.

[7]A. J. Cooner, Y. Shao, and J. B. Campbell, "Detection of Urban Damage Using Remote Sensing and Machine Learning Algorithms : Revisiting the 2010 Haiti Earthquake," Remote Sens., no. Ml, 2016.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems, 2012, pp. 1097–1105.

[9]M.Radovic,O.Adarkwa,Q.Wang,Object Recognition in Aerial Images Using Convolutional Neural Networks,J.Imag.3(2)(2017)21.

[10]M. B. Bejiga, A. Zeggada, and F. Melgani, "Convolutional neural networks for near real-time object detection from UAV imagery in avalanche search and rescue operations," 2016 IEEE Int. Geosci. Remote Sens. Symp., pp. 693–696, 2016.

[11]V. Q. Nguyen, H.-J. Yang, K. Kim, and A.-R. Oh, "Real-Time Earthquake Detection Using Convolutional Neural Network and Social Data," 2017 IEEE Third Int. Conf. Multimedia. Big Data, pp. 154–157, 2017.

[12]S. Li, H. Hua, and L. Song, "Automatic Recognition of landslides based on CNN and Texture Change Detection," 31st Youth Acad. Annu. Conf. Chinese Assoc. Autom., vol. 7384, no. Subsection C, p. 73842E–73842E–6, 2016.

[13]Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, "Rich Feature H ierarchies for Accurate Object Detection and Semantic Segmentation," The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 580-587

[14]S.Ren,K.He,R.Girshick,J.Sun,Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,Proceeding of Advances in Neural Information Processing Systems,Vancouver,Canada,NIPS,2015.

[15]M.Everingham,Luc Van Gool,Christopher K. I. Williams,John WinnAndrew Zisserman,The Pascal Visual Object Classes (VOC) Challenge,Int.J.Comput.Vision 88 (2)(2010)303-338.

[16]A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Proc. the Int. Conf. Neural Inf. Process. Syst., pp. 1097–1105, 2012.

[17]A. Toshev and C. Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks," CVPR, pp. 1653–1660, 2014.
[18]E.Guirado, S.Tabik, M.L.Rivas, D.Alcaraz-Segura, F.Herrera, Automatic whale counting in satellite images with deeplearning, bioRxiv, 2018.

[19]C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning Hierarchical Features for Scene Labeling," Pattern Anal. Mach. Intell. IEEE Trans., vol. 35, no. 8, pp. 1915–1929, 2013.

[20]K.Simonyan, A.Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXivpreprintar Xiv: 14091556.

[21]C.Szegedy,W.Liu,Y.Jia,P.Sermanet,S.Reed,D.Anguelov,D.Erhan,V.Vanhoucke,A.Rabinovi ch,Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, IEEE, 2015.

[22]L. Gueguen and R. Hamid, "Large-scale damage detection using satellite imagery," 2015 IEEE Conf. Comput. Vis. Pattern Recognit., pp. 1321–1328, 2015.

[23]K.He, X.Zhang, S.Ren, J.Sun, Deep residual learning for image recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, LasVegas, NV, USA, IEEE, 2016.

[24]Udacity course, "Deep learning," retrieved from https://www.udacity.com/course/deep-learning--ud730 last accessed on 20 April 2017

[25] Stanford course, "Transfer Learning", retrieved from http://cs231n.github.io/transfer-learning/ last accessed on 18 April 2017

[26]J.Redmon,S.Divvala,R.Girshick,A.Farhadi,Youonlylookonce:Unified,real time object detection,Proceedings of the IEEE Conference on ComputerVision and PatternRecognition,LasVegas,NV,USA,IEEE,2016.

[27]W.Liu,D.Anguelov,D.Erhan,C.Szegedy,S.Reed,C.Y.Fu,A.C.Berg,SSD:Single shot multi box detector,European Conference on ComputerVision,Amsterdam, Netherlands,Springer,2016.

[28]T. Hubel, D. and Wiesel, "Receptive fields and functional architecture of monkey striate cortex," J. Physiol., no. 195, pp. 215– 243, 1968.

[29]Y. LECUN, L.´ON BOTTOU, Y. BENGIO, and P. HAFFNER, "Gradient-Based Learning Applied to Document Recognition," Proc. IEEE, vol. 86, no. 11, 1998.

[30]T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, “Robust object recognition with cortex-like mechanisms,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 3, pp. 411–426, 2007.

[31]K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” Biol. Cybern., vol. 36, no. 4, pp. 193–202, 1980.

.

BIBLIOGRAPHY

- DEEP LEARNING FOR COMPUTER VISION – (Rajalingappaa Shanmugamani)
- DEEP LEARNING WITH PYTHON – (Francois Chollet)
- NEURAL NETWORKS AND DEEP LEARNING – (Charu C.Agarwal)

URL'S

- https://www.analyticsvidhya.com/blog/2017/05/25-must-know-terms- concepts-for-beginners-in-deep-learning/
- https://www.researchgate.net/publication/325116934_Image_classification_using_ Deep_learning
- https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/
- https://www.analyticsindiamag.com/7-types-classification-algorithms/
- https://www.pyimagesearch.com/2018/09/10/keras-tutorial-how-to-get- started-with-keras-deep-learning-and-python/
- https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg- googlenet-resnet-and-more-666091488df5
- https://towardsdatascience.com/the-4-convolutional-neural-network-models- that-can-classify-your-fashion-images-9fe7f3e5399d

APPENDIX

MAIN CODE:

```
# import the necessary packages from
keras.models import Sequential
from keras.layers.normalization import BatchNormalization from
keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D from
keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dropout
from keras.layers.core import Dense
from keras import backend as K

class SmallVGGNet:
    @staticmethod
    def build(width, height, depth, classes):
        # initialize the model along with the input shape to be #
        "channels last" and the channels dimension itself model
        = Sequential()
        inputShape = (height, width, depth) chanDim =
        -1
        # if we are using "channels first", update the input shape #
        and channels dimension
        if K.image_data_format() == "channels_first":
            inputShape = (depth, height, width) chanDim =
            1
        # CONV => RELU => POOL layer set
        model.add(Conv2D(32, (3, 3), padding="same",
        input_shape=inputShape))
        model.add(Activation("relu"))
        model.add(BatchNormalization(axis=chanDim))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.25))
```

```
# (CONV => RELU) * 2 => POOL layer set
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
# (CONV => RELU) * 3 => POOL layer set
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
# first (and only) set of FC => RELU layers
model.add(Flatten()) model.add(Dense(512))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
# softmax classifier model.add(Dense(classes))
model.add(Activation("softmax"))
# return the constructed network architecture
return model
```

TRAINING THE VGG MODEL:

USAGE

```
# python train_vgg.py --dataset animals --model output/smallvggnet.model --label-bin
output/smallvggnet_lb.pickle --plot output/smallvggnet_plot.png

# set the matplotlib backend so figures can be saved in the background
import matplotlib
matplotlib.use("Agg")

# import the necessary packages
from pyimagesearch.smallvggnet import SmallVGGNet from
sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import SGD
from imutils import paths import
matplotlib.pyplot as plt import
numpy as np
import argparse
import random
import pickle
import cv2 import
os

# construct the argument parser and parse the arguments ap
= argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True, help="path
   to input dataset of images")
ap.add_argument("-m", "--model", required=True, help="path
   to output trained model")
ap.add_argument("-l", "--label-bin", required=True,
   help="path to output label binarizer")
```

```
ap.add_argument("-p", "--plot", required=True,
   help="path to output accuracy/loss plot")
args = vars(ap.parse_args())

# initialize the data and labels
print("[INFO] loading images...")
data = []
labels = []

# grab the image paths and randomly shuffle them imagePaths
= sorted(list(paths.list_images(args["dataset"])))
random.seed(42)
random.shuffle(imagePaths)

# loop over the input images for
imagePath in imagePaths:
   # load the image, resize it to 64x64 pixels (the required input
   # spatial dimensions of SmallVGGNet), and store the image in the #
   data list
   image = cv2.imread(imagePath) image
   = cv2.resize(image, (64, 64))
   data.append(image)
   # extract the class label from the image path and update the #
   labels list
   label = imagePath.split(os.path.sep)[-2]
   labels.append(label)
# scale the raw pixel intensities to the range [0, 1] data
= np.array(data, dtype="float") / 255.0
labels = np.array(labels)

# partition the data into training and testing splits using 75% of #
the data for training and the remaining 25% for testing (trainX,
testX, trainY, testY) = train_test_split(data,
   labels, test_size=0.25, random_state=42)
```

```
# convert the labels from integers to vectors (for 2-class, binary #
classification you should use Keras' to_categorical function
# instead as the scikit-learn's LabelBinarizer will not return a #
vector)
lb = LabelBinarizer()
trainY = lb.fit_transform(trainY) testY
= lb.transform(testY)

# construct the image generator for data augmentation
aug = ImageDataGenerator(rotation_range=30, width_shift_range=0.1,
    height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
    horizontal_flip=True, fill_mode="nearest")

# initialize our VGG-like Convolutional Neural Network model
= SmallVGGNet.build(width=64, height=64, depth=3,
    classes=len(lb.classes_))

# initialize our initial learning rate, # of epochs to train for, #
and batch size
INIT_LR = 0.01
EPOCHS = 75
BS = 32

# initialize the model and optimizer (you'll want to use #
binary_crossentropy for 2-class classification)
print("[INFO] training network...")
opt = SGD(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="categorical_crossentropy", optimizer=opt,
    metrics=["accuracy"])

# train the network
H = model.fit_generator(aug.flow(trainX, trainY, batch_size=BS),
    validation_data=(testX, testY), steps_per_epoch=len(trainX) // BS, epochs=EPOCHS)
```

```
# evaluate the network
print("[INFO] evaluating network...")
predictions = model.predict(testX, batch_size=32)
print(classification_report(testY.argmax(axis=1),
predictions.argmax(axis=1), target_names=lb.classes_))

# plot the training loss and accuracy N
= np.arange(0, EPOCHS)
plt.style.use("ggplot")
plt.figure()
plt.plot(N, H.history["loss"], label="train_loss") plt.plot(N,
H.history["val_loss"], label="val_loss") plt.plot(N,
H.history["acc"], label="train_acc") plt.plot(N,
H.history["val_acc"], label="val_acc") plt.title("Training
Loss and Accuracy (SmallVGGNet)") plt.xlabel("Epoch
#")
plt.ylabel("Loss/Accuracy")
plt.legend()
plt.savefig(args["plot"])

# save the model and label binarizer to disk print("[INFO]
serializing network and label binarizer...")
model.save(args["model"])
f = open(args["label_bin"], "wb")
f.write(pickle.dumps(lb)) f.close()
```

PREDICTING THE MODEL:

USAGE

python predict.py --image images/dog.jpg --model output/simple_nn.model --label-bin output/simple_nn_lb.pickle --width 32 --height 32 --flatten 1

python predict.py --image images/dog.jpg --model output/smallvggnet.model --label-bin output/smallvggnet_lb.pickle --width 64 --height 64

```
# import the necessary packages
from keras.models import load_model
import argparse
import pickle
import cv2

# construct the argument parser and parse the arguments ap
= argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True, help="path
   to input image we are going to classify")
ap.add_argument("-m", "--model", required=True, help="path
   to trained Keras model")
ap.add_argument("-l", "--label-bin", required=True,
   help="path to label binarizer")
ap.add_argument("-w", "--width", type=int, default=28,
   help="target spatial dimension width")
ap.add_argument("-e", "--height", type=int, default=28,
   help="target spatial dimension height")
ap.add_argument("-f", "--flatten", type=int, default=-1,
   help="whether or not we should flatten the image")
args = vars(ap.parse_args())

# load the input image and resize it to the target spatial dimensions image =
cv2.imread(args["image"])
output = image.copy()
image = cv2.resize(image, (args["width"], args["height"]))
```

```
# scale the pixel values to [0, 1]
image=image.astype("float") / 255.0

# check to see if we should flatten the image and add a batch #
dimension
if args["flatten"] > 0: image =
    image.flatten()
    image = image.reshape((1, image.shape[0]))

# otherwise, we must be working with a CNN -- don't flatten the #
image, simply add the batch dimension
else:
    image = image.reshape((1, image.shape[0], image.shape[1],
        image.shape[2]))

# load the model and label binarizer
print("[INFO] loading network and label binarizer...")
model = load_model(args["model"])
lb = pickle.loads(open(args["label_bin"], "rb").read())

# make a prediction on the image preds
= model.predict(image)

# find the class label index with the largest corresponding #
probability
i = preds.argmax (axis=1)[0] label
= lb.classes_[i]
# draw the class label + probability on the output image text =
"{}: {:.2f} %".format (label, preds [0][i] * 100)
cv2.putText (output, text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
    (0, 0, 255), 2)
# show the output image
cv2.imshow("Image", output)
cv2.waitKey(0)
```