**The Project Report on**

# ANIMAL CLASSIFICATION using cnn

**Submitted To**

**Acharya Nagarjuna University**

**A project report submitted in the partial fulfillment of the requirements for the**

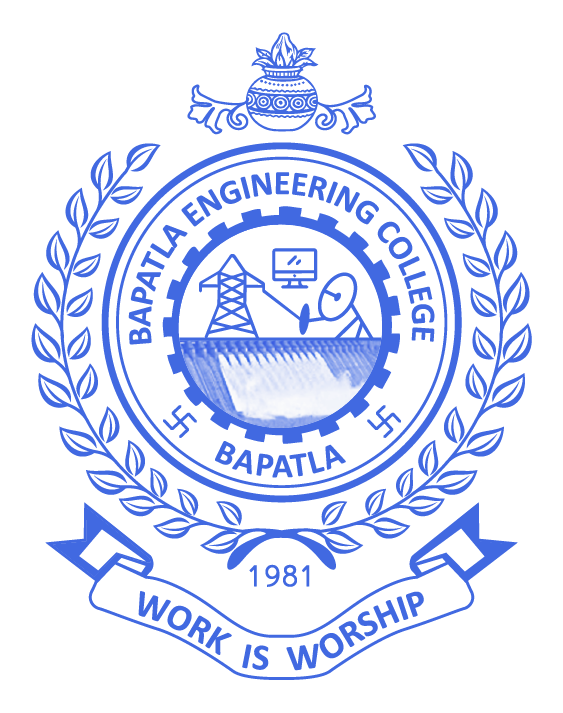
**Award of the Degree of**

**MASTER OF COMPUTER APPLICATIONS**

**Submitted By**

**ARUNURU NAVEEN**

**Reg.No: L20MC23014**

****

**Under The Esteemed Guidance Of**

**Sri N. KIRAN KUMAR M. Tech**

**Assistant Professor**

**DEPARTMENT OF MCA**

**BAPATLA ENGINEERING COLLEGE**

**(Approved by A.I.C.T.E)**

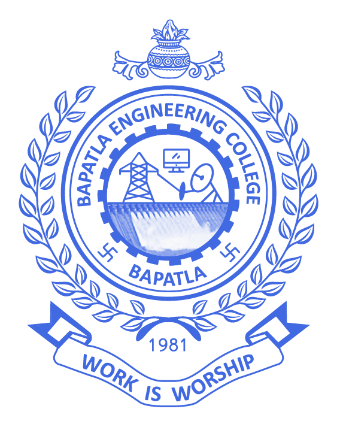
**(Affiliated to ACHARYA NAGARJUNA UNIVERSITY)**

**2019-2021**

**DEPARTMENT OF MCA**

**BAPATLA ENGINEERING COLLEGE**

**BAPATLA-522101**

****

**CERTIFICATE**

This is to certify that this project work entitled “**Secure Cloud Storage based on RLWE Problem”** is the bonafide work carried out by **ARUNURU NAVEEN**, **Reg.No: L20MC23014** submitted in Partial fulfillment of the requirement for the Award of Degree of “**Master of** **Computer Applications**”, during the academic year 2019-2021.

The results submitted in this project have been verified and are found to be satisfactory. The results embodied in this thesis have not been submitted to any other university for the award of the any other degree/diploma.

**Project Guide: Head of the Department:**

**Sri N. Kiran Kumar M. Tech Sri K.N. Prasad M. Tech**

**Assistant Professor Associate Professor & Head**

**Dept. of MCA Dept. of MCA**

**Bapatla Engineering College Bapatla Engineering College**

**Bapatla. Bapatla.**

**Signature of External Examiner**

# ACKNOWLEDGEMENT

I sincerely thank the following distinguished personalities who have given their advice and support for successful completion of this work.

I would like to thank **Dr. V. Damodara Naidu**, Principal, Bapatla Engineering College, Bapatla for providing the online resources and allowing me to carry out this work.

I extended my sincere thanks to **Sri K.N. Prasad**, Head of the Department for extending their cooperation and providing the required resources.

I am deeply indebted to most respect and my project guide **Sri N. Kiran Kumar**, Assistant Professor Dept. of MCA for his valuable and inspiring guidance, comments, suggestions and encouragements.

I extended my sincere thanks to all other teaching and non-teaching staff of the department who helped directly or indirectly for their cooperation and encouragement.

**ARUNURU NAVEEN**

**(L20MC23014)**

**DECLARATION**

This is to declare that the project **“Secure Cloud Storage based on RLWE Problem”** at Bapatla Engineering College has been presented by me during the academic year **2019-2021** in partial fulfillment of the requirements for the **“Master of Computer Application”**.

I also declare that this project is the result of my own efforts and that it has not been submitted to any other universities for the award of degree or diploma.

**ARUNURU NAVEEN**

**(L20MC23014)**

Table of Contents

[ACKNOWLEDGEMENT iii](#_Toc80180563)

[ABSTRACT viii](#_Toc80180564)

[CHAPTER-1 1](#_Toc80180565)

[INTRODUCTION 1](#_Toc80180566)

[1.1 INTRODUCTION 1](#_Toc80180567)

[CHAPTER- 2 5](#_Toc80180568)

[LITERATURE SURVEY 5](#_Toc80180569)

[2.1 LITERATURE REVIEW 5](#_Toc80180570)

[CHAPTER-3 7](#_Toc80180571)

[THEORETICAL BACKGROUND 7](#_Toc80180572)

[3.1 INTRODUCTION: 7](#_Toc80180573)

[3.2 INTRODUCTION TO PYTHON 12](#_Toc80180574)

[3.3 BENFITS OF PYTHON 21](#_Toc80180575)

[CHAPTER-4 30](#_Toc80180576)

[SYSTEM ANALYSIS 30](#_Toc80180577)

[4.1 EXISTING SYSTEM: 30](#_Toc80180578)

[4.1.1 DISADVANTAGES OF EXISTING SYSTEM: 30](#_Toc80180579)

[4.2 PROPOSED SYSTEM: 30](#_Toc80180580)

[4.2.1 ADVANTAGES OF PROPOSED SYSTEM: 31](#_Toc80180581)

[CHAPTER- 5 32](#_Toc80180582)

[SYSTEM DESIGN 32](#_Toc80180583)

[5.1 INTRODUCTION 32](#_Toc80180584)

[5.2 MODULES 32](#_Toc80180585)

[5.2.1 DATASET: 32](#_Toc80180586)

[5.2.2 PREPROCESSING: 32](#_Toc80180587)

[5.2.3 GRAPHS: 32](#_Toc80180588)

[5.2.4 PREDICTION: 32](#_Toc80180589)

[5.3 SYSTEM ARCHITECTURE 33](#_Toc80180590)

[5.4 UML DAIGRAMS 34](#_Toc80180591)

[5.4.1 CONSTRUCTION OF USE CASE DIAGRAMS: 37](#_Toc80180592)

[5.4.2 SEQUENCE DIAGRAMS: 40](#_Toc80180593)

[5.4.3 CLASS DIAGRAM: 42](#_Toc80180594)

[5.4.4 ACTIVITY DIAGRAM: 42](#_Toc80180595)

[CHAPTER-6 44](#_Toc80180596)

[SYSTEM REQUIREMENTS 44](#_Toc80180597)

[6.1 SYSTEM REQUIREMENTS 44](#_Toc80180598)

[6.1.1 HARDWARE REQUIREMENTS: 44](#_Toc80180599)

[6.1.2 SOFTWARE REQUIREMENTS: 44](#_Toc80180600)

[CHAPTER-7 45](#_Toc80180601)

[SYSTEM IMPLEMENTATION 45](#_Toc80180602)

[7.1 INPUT AND OUTPUT DESIGNS 45](#_Toc80180603)

[7.1.1 LOGICAL DESIGN 45](#_Toc80180604)

[7.1.2 PHYSICAL DESIGN 45](#_Toc80180605)

[7.2 INPUT & OUTPUT REPRESENTATION 46](#_Toc80180606)

[7.2.1 INPUT DESIGN 46](#_Toc80180607)

[7.2.2 OBJECTIVES 47](#_Toc80180608)

[7.2.3 OUTPUT DESIGN 47](#_Toc80180609)

[CHAPTER-8 69](#_Toc80180610)

[SYSTEM TESTING 69](#_Toc80180611)

[8.1 INTRODUCTION: 69](#_Toc80180612)

[8.2 LEVELS OF TESTING 69](#_Toc80180613)

[8.2.1 BLACK BOX TESTING 70](#_Toc80180614)

[8.2.2 WHITE BOX TESTING 72](#_Toc80180615)

[CHAPTER-9 74](#_Toc80180616)

[OUTPUT SCREENS 74](#_Toc80180617)

[CONCLUSION 75](#_Toc80180618)

[REFERENCES 76](#_Toc80180619)

**List of Figures**

**Name of the figure Pg.no**

[Figure 3. 1 Structure of cloud computing 7](#_Toc80179715)

[Figure 3. 2 Characteristics of cloud computing 9](#_Toc80179716)

[Figure 3. 3 Structure of service models 10](#_Toc80179717)

[Figure 5. 1 System Architecture 33](#_Toc80179735)

[Figure 5. 2 Use Case Diagram 39](#_Toc80179736)

[Figure 5. 3 Sequence diagram 41](#_Toc80179737)

[Figure 5. 4 Class Diagram 42](#_Toc80179738)

[Figure 5. 5 Activity Diagram 43](#_Toc80179739)

# ABSTRACT

Classification and identification of wild animals for tracking and protection purposes has become increasingly important with the deterioration of the environment, and technology is the agent of change which augments this process with novel solutions. Computer vision is one such technology which uses the abilities of artificial intelligence and machine learning models on visual inputs. Convolution neural networks (CNNs) have multiple layers which have different weights for the purpose of prediction of a particular input. The precedent for classification, however, is set by the image processing techniques which provide nearly ideal input images that produce optimal results. Image segmentation is one such widely used image processing method which provides a clear demarcation of the areas of interest in the image, be it regions or objects. The Efficiency of CNN can be related to the preprocessing done before training. Further, it is a well-established fact that heterogeneity in image sources is detrimental to the performance of CNNs. Thus, the added functionality of heterogeneity elimination is performed by the image processing techniques, introducing a level of consistency that sets the tone for the excellent feature extraction and eventually in classification.

.

**PROBLEM STATEMENT:**

* The expansion of urban areas in modern times has resulted in widespread displacement of habitats in forested areas. As a result, wild animals are forced to venture into the human settlements that often infringe on their routine activities. More often than not, food is the primary motivator for such peregrinations. It is at this point that there is tangible danger to any humans that inadvertently cross the path of these animals when they are at their most ferocious predispositions. Hence, a need arises for the detection of wild animals at the border of human settlements close to wild habitats.

.

**OBJECTIVE:**

* we are performing classification of wild animals listed in the IUCN Red List of Threatened Species. The dataset for the same has been sourced from Kaggle [1]. The images are captured in diverse lighting conditions and environments thereby making our algorithm more capable of real world deployment in conservation setups.

# CHAPTER-1

# INTRODUCTION

## Introduction

CNN (convolutional neural network) is one category of deep learning neutral networks. It is a kind of multilayer neural network with artificial neurons. The multilayer mianly consists of three basic types: input layer, hidden layer and output layer. The kernel existed in first hidden layer can gain the convolved feature from input images and then the concolved feature can be transmitted to the next layer. During the process, the feature obtained becomes more complex as layer is deeper. With different image classes, CNN can extract picture features respectively and proceed identification. Because of its working principle, it has a wide range of applications in image classification. For example, it can be found in Facebook’s photo tagging and in self-driving areas as the core technology [1]. Although some portion of CNN image recognition technologies are established well in certain areas, the higher level of CNN image recognition is still in infancy, such as the identification for road obstacles which is challenging under the complicated lane conditions. The mian drawback of CNN is the requirement for large amount of data (e.g. same image with different angles and different lighting conditions) and failed to understand image contents. In this paper, CNN model are applied for similar rare creature image classification. High similarities exist in various species of the same class. Hence, it is hard to distinguish them by eyes in real time. This model can help develop animal conservation because it can fast recognize endangered animals remotely. Then, taking further actions to protect them from catch and hunt. The model is trained to classify images for snub-nosed monkeys, a kind of rare specials in monkey group, and normal monkeys which have high similarity to snub-nosed monkey. The CNN model extracts image features with input data to identify other similar images [2]. Training process enables CNN to identify differences among images. In test stage, it is able to classify images into different groups. Database construction and data preprocess are also contained in this article. Additionally, accuracy obtained in test data can be used to evaluate the model reliability. However, the model is a simple 2D convolution neural network, which means limitation applications existed in this model. Although some improvements are feasible on model reliability, it only can process 2D problems. This paper aims to investigate the similar species discrimination according to fine distinctions in animal groups. Taking monkey group as an example, the neural network can distinguish between the snub-nosed monkey and normal monkey. This study based on CNN algorithms is quite constructive in rare animal conservation aera, which can be further utilized in other rare animal groups..

.

# CHAPTER- 2

# literature survey

## 2.1 literature review

Wildlife protection has had a significant number of approaches which integrate myriad technology stacks to solve niche issues. A system that combines deep learning classification with dynamic background modelling to evolve a swift and precise method for human and animal detection from highly cluttered camera trap pictures. Background modelling helps generate region proposals for foreground objects, which are then classified using the DCNN, resulting in improved efficiency and increased accuracy. The proposed system achieves 82% accuracy in segmenting images into human, animal and background patches [2]. A.V Sayagavi et al. [3] used a network of cameras, connected to PIR motion sensors, so that image capture is triggered only when some movement is detected. The images captured through these cameras are processed to detect the presence of wild animals using YOLO, and if an animal is found, identify the species. Once identified, the animals are tracked for a suitable time using CSRT in order to determine their intent – such as to find whether they are moving across the village, or into it. In the latter case, alerts are generated and local authorities are notified through proper channels. The models at present can detect 5 types of animals namely (elephant, zebra, giraffe, lion and cheetah). A comparative study on 4 different algorithms based on deep neural networks has also been proposed in [4]. Two variants of single shot multibox detector (SSD) and two variants of faster region-based CNN (Faster R-CNN) have been compared. Two different activation functions are also compared. The SSD variants outperform all faster R-CNN variants and provide more precise detection compared to the latter. Non-intrusive monitoring of animals was explored by A.G Villa et al. [5]. They generated huge volumes of data as they made use of multiple camera trapping networks. To analyse the data, they used a very deep CNN framework and chose 26 species from the Snapshot Serengeti (SSe) dataset. The proposed model achieved an accuracy of 88.9%. A comparison with other techniques was also carried out which showcased that their model outperformed previous approaches. An automated wildlife monitoring system which leverages state-of-the-art deep CNN architecture has been devised [6]. The model achieved 90.4% accuracy for 3 animal classes. A single labelled dataset was used for training purposes, and a focus is placed on filtering animal images. A comparative study between the bag of visual words and deep learning CNN techniques for wild animal recognition [7]. The comparison is done for grey level as well as colour information. The features extracted by the BOW models were combined with a regularised L2 support vector machine for classification. This study suggests that there is a clear performance gulf between CNN methods and BOW. In [8] YOLOv3, a CNN architecture as a pre-trained model through transfer learning technique is used. Fine tuning was subsequently performed using an amalgamation of self-shot and crowdsourced images. The model locates the object detected and adds a bounding box upon it

# Chapter-3

# Theoretical background

## 3.1 Introduction:

The intrinsic nature of any classification technique is the ability to accurately identify the major features of the target that it aims to predict. It is a well-documented fact that Deep Learning models and frameworks provide enhanced accuracy when the inputs have decreased source-induced heterogeneity. Since raw real-world feeds do not generate ideal or optimal images for classification, the onus is on the application of image processing techniques to act as the liaison between the classifier and the inputs. Some of the most influential challenges to classifier performance with regards to light intensity, unavailability of high-quality night-vision training images, noisy or element-rich backgrounds of captured images and luminance problems because of shadow effect need to be mitigated. The goal is to enable the classifier to extract features optimally from the images in order to assign a class to them with minimal loss and maximum accuracy. Ergo, the preprocessing needs to be performed specifically to ensure that the features are as distinctly visible as possible.

## 3.2 Introduction to PYTHON

**Python Technology**

Python technology is both a programming language and a platform.

**The python Programming Language**

THE PYTHON PROGRAMMING LANGUAGE IS A HIGH-LEVEL LANGUAGE THAT CAN BE CHARACTERIZED BY ALL OF THE FOLLOWING BUZZWORDS:

* + - Simple
    - Architecture neutral
    - Object oriented
    - Portable
    - Distributed
    - High performance
    - Interpreted
    - Multithreaded
    - Robust
    - Dynamic
    - Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Python programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Python byte codes —the platform-independent codes interpreted by the interpreter on the Python platform. The interpreter parses and runs each Python byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

FEATURES OF MACHINE LEARNING

• It is nothing but automating the Automation.

• Getting computers to program themselves.

• Writing Software is bottleneck.

• Machine leaning models involves machines learning from data without the help of humans or any kind of human intervention.

• Machine Learning is the science of making of making the computers learn and act like humans by feeding data and information without being explicitly programmed.

• Machine Learning is totally different from traditionally programming, here data and output is given to the computer and in return it gives us the program which provides solution to the various problems. Below is the figure.

**Traditional Programming vs Machine Learning**

• Machine Learning is a combination of Algorithms, Datasets, and Programs.

• There are Many Algorithms in Machine Learning through which we will provide us the exact solution in predicting the disease of the patients.

• How Does Machine Learning Works?

• Solution to the above question is Machine learning works by taking in data, finding relationships within that data and then giving the output.

**Machine Learning Model**

• There are various applications in which machine learning is implemented such as Web search, computing biology, finance, e-commerce, space exploration, robotics, social networks, debugging and much more.

• There are 3 types of machine learning supervised, unsupervised, and reinforcement.

**BENEFITS OF PYTHON**

• Presence of Third-Party Modules

• Extensive Support Libraries

• Open Source and Community Development

• Learning Ease and Support Available

• User-friendly Data Structures

• Productivity and Speed

• Highly Extensible and Easily Readable Language.

**Python**

Python is high level language and it is also integrated version of the program. Python is an object-oriented approach and its main aim to help programmers to write the code clearly, logical code for small and large scale of project.

Pytrhon is dynamically typed and garbage collected it also support multiple programming and it is both procedure and object oriented and also functional programming. And structural programming also supported. It has many built in function it also supports filter, map and reduce function. All the machine learning algorithm and the libraries are being supported by the python programming language. Python also support list, dict, sets and other generators. Python code can be run in different platform such as anaconda, PyCharm etc.

The main goal of this programing language is as follows:

• Python is simple, object-oriented programming language.

• The language and implementation should provide support for software engineering principles such as strong type library preset for different machine learning algorithm, and all other algorithm in simple manner.

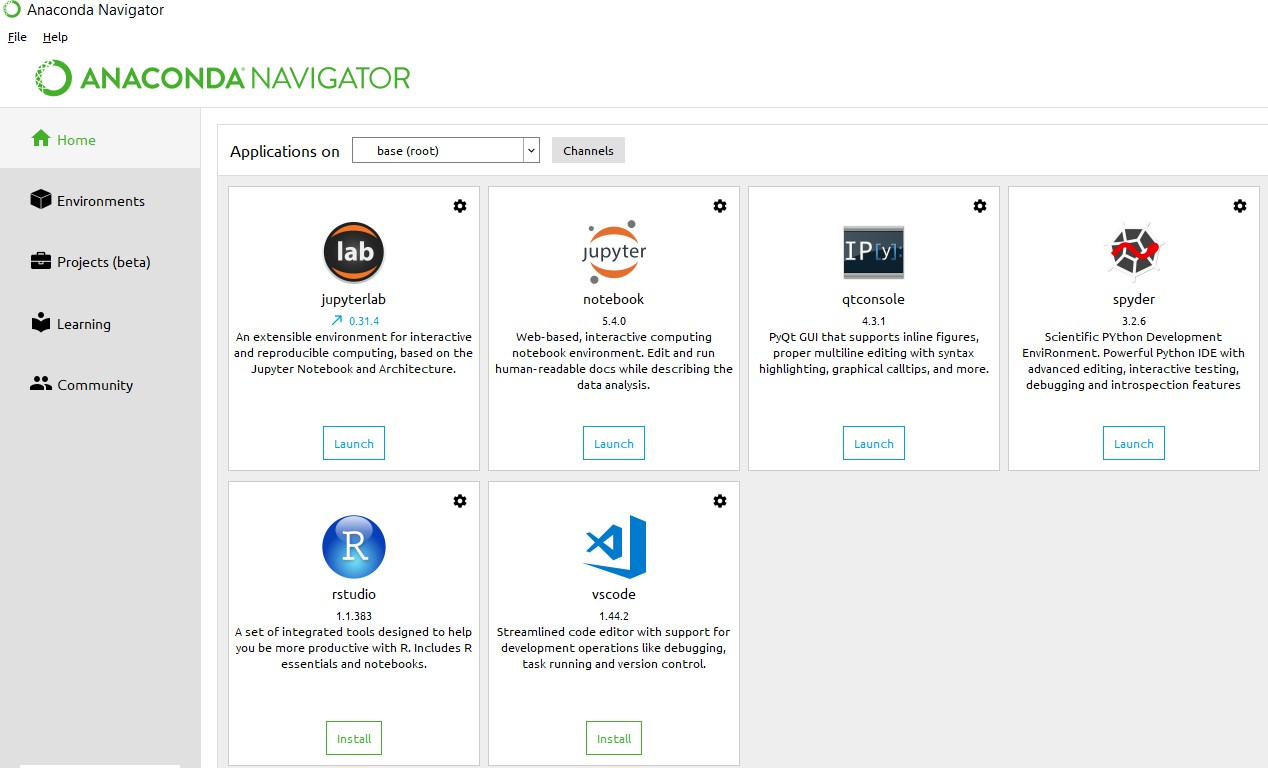
• Coding will be smooth in python and the data analysis can be easily done in python.

This is so much so to the point where we now have modules and APIs at our disposal, and you can engage in machine learning very easily without almost any knowledge at all of how it works. With the defaults from Scikit-learn, you can get 90-95% accuracy on many tasks right out of the gate. Machine learning is a lot like a car, you do not need to know much about how it works in order to get an incredible amount of utility from it.

Despite the apparent age and maturity of machine learning, I would say there's no better time than now to learn it, since you can actually use it. Machines are quite powerful, the one you are working on can probably do most of this series quickly. Data is also very plentiful lately.

**Anaconda**

Anaconda is free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine Learning applications, Large- scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. It is developed and maintained by Anaconda, Inc. The distribution incudes data-science packages suitable for Windows, Linux, and macOS. Packaged versions are required and are managed by the package management system anaconda. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for other things than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depends on, and a small number of other packages.



**Anaconda Console**

**Jupyter notebook**

Jupiter Notebook or so called IPython Notebook is an interactive web based computational mean for starting with Jupiter Notebook documents. The term notebook itself is a huge entity to represent the integration with different entity sets. JSON is the main document form from the same for the execution which follows the brief on the schema and the input and output means. It has high integration with several language set and has various flexibilities with the choices. The extension used for the same is “.ipynb” which runs in this platform. It’s an open-source software package with interactive communication means. It has it’s open standards for the same. It’s an open community best for budding programmers . The flexibility of the same is phenomenon and splendidly done the configuration and integration of the same is simplest and easy on hold so that no prior distortion is generated and the efficiency of the same is measured through out any system of choice.

It’s the best software sets that been used across cross for designing and developing of the products and support wide help support. Not only to that, it provides scalability in the code and the deployment of the same. Various Language can be changed and the project can be undertaken on the same. The created notebook files can be shared and stored in various means for further utilization. It supports cultivated and interactive output sets. Easily crossed over for graphing, plotting and visualizing of the elements. Data Integration of the same is to it’s best. The integration of big data and it can process chunks of values in an approx. time which gives a better performance and the higher computational means. Various works on data like cleaning, cleansing, transforming modeling and visualizing can be done by the same

Machine learning is the ability that gives the computer to learn without being explicitly programmed. There are two types of machine learning:

Supervised Learning: supervised learning is the learning of the labelled data. It is the types of machine learning that maps the input and output based on the examples input-output pairs. In supervised learning each training data having pairs of input and desired outputs values. Supervised learning algorithm analyzes the training data and produces a function which can be used for mapping of new data.

Fig 2.1 Supervised Learning The output to solve the supervised learning algorithm are as:

• Determine the types of data, before doing anything else the user should understand which types of data set is to be used for training the data.

• Gathered the training data sets either in form of human experts or from measurements.

• Determine the feature of inputs from the learned data and depends on the inputs it changed into feature vector; number of features should not be large but should contains enough information to accurately predict the outputs.

• Check the learned function and the learned algorithm for example we use support vector machines or decisions tree.

• Complete the design and run the trained data sets.

• Analyzed the output and verify the data sets to get the accurate outputs.

Unsupervised Learning:

Unsupervised learning is a type of machine learning that helps in finding the previously unknown patterns in the data set without any known labels. It is known as self- organization and allows modelling probability densities of given inputs.

Fig 2.2 unsupervised Learning Some of the algorithm used in unsupervised learning are:

• Clustering

• Anomaly detection

• Neural networks

• Approach for learning latent variable models

• Non labelled data

Semi Supervised Machine Learning algorithm: It’s like the middle man which have some labeled data and some unlabeled which can be prosed by the both the structured and unsupervised learning.

The algorithms have been compared based upon the parameters: Size of the dataset and Number of technical indicators used. Accuracy and F-measure values have been computed for each algorithm. Long term model has been used to compute the accuracy and F-measure.

Reinforcement Learning: This type of learning is used to reinforce or strengthen the network based on critic information. That is, a network being trained under reinforcement learning, receives some feedback from the environment. However, the feedback is evaluative and not instructive as in the case of supervised learning. Based on this feedback, the network performs the adjustments of the weights to obtain better critic information in future.

This learning process is similar to supervised learning but we might have very less information. The following figure gives the block diagram of reinforcement learning:

**import numpy as np**

* NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.
* At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:
  + NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original.
  + The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
  + NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences.
  + A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today’s scientific/mathematical Python-based software, just knowing how to use Python’s built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

**import time**

This module provides various time-related functions. For related functionality, see also the datetime and calendar modules.

Although this module is always available, not all functions are available on all platforms. Most of the functions defined in this module call platform C library functions with the same name. It may sometimes be helpful to consult the platform documentation, because the semantics of these functions varies among platforms.

An explanation of some terminology and conventions is in order.

The epoch is the point where the time starts, and is platform dependent. For Unix, the epoch is January 1, 1970, 00:00:00 (UTC). To find out what the epoch is on a given platform, look at time.gmtime(0).

The term seconds since the epoch refers to the total number of elapsed seconds since the epoch, typically excluding leap seconds. Leap seconds are excluded from this total on all POSIX-compliant platforms.

The functions in this module may not handle dates and times before the epoch or far in the future. The cut-off point in the future is determined by the C library; for 32-bit systems, it is typically in 2038.

Function strptime() can parse 2-digit years when given %y format code. When 2-digit years are parsed, they are converted according to the POSIX and ISO C standards: values 69–99 are mapped to 1969–1999, and values 0–68 are mapped to 2000–2068.

UTC is Coordinated Universal Time (formerly known as Greenwich Mean Time, or GMT). The acronym UTC is not a mistake but a compromise between English and French.

DST is Daylight Saving Time, an adjustment of the timezone by (usually) one hour during part of the year. DST rules are magic (determined by local law) and can change from year to year. The C library has a table containing the local rules (often it is read from a system file for flexibility) and is the only source of True Wisdom in this respect.

The precision of the various real-time functions may be less than suggested by the units in which their value or argument is expressed. E.g. on most Unix systems, the clock “ticks” only 50 or 100 times a second.

On the other hand, the precision of time() and sleep() is better than their Unix equivalents: times are expressed as floating point numbers, time() returns the most accurate time available (using Unix gettimeofday() where available), and sleep() will accept a time with a nonzero fraction (Unix select() is used to implement this, where available).

The time value as returned by gmtime(), localtime(), and strptime(), and accepted by asctime(), mktime() and strftime(), is a sequence of 9 integers. The return values of gmtime(), localtime(), and strptime() also offer attribute names for individual fields.

See struct\_time for a description of these objects.

Changed in version 3.3: The struct\_time type was extended to provide the tm\_gmtoff and tm\_zone attributes when platform supports corresponding struct tm members.

Changed in version 3.6: The struct\_time attributes tm\_gmtoff and tm\_zone are now available on all platforms.

**import os**

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the fileinput module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module.

Notes on the availability of these functions:

The design of all built-in operating system dependent modules of Python is such that as long as the same functionality is available, it uses the same interface; for example, the function os.stat(path) returns stat information about path in the same format (which happens to have originated with the POSIX interface).

Extensions peculiar to a particular operating system are also available through the os module, but using them is of course a threat to portability.

All functions accepting path or file names accept both bytes and string objects, and result in an object of the same type, if a path or file name is returned.

On VxWorks, os.popen, os.fork, os.execv and os.spawn\*p\* are not supported.

# 

# Chapter-4

# System analysis

## 4.1 EXISTING SYSTEM:

A.V Sayagavi et al. [3] used a network of cameras, connected to PIR motion sensors, so that image capture is triggered only when some movement is detected. The images captured through these cameras are processed to detect the presence of wild animals using YOLO, and if an animal is found, identify the species. Once identified, the animals are tracked for a suitable time using CSRT in order to determine their intent – such as to find whether they are moving across the village, or into it. In the latter case, alerts are generated and local authorities are notified through proper channels. The models at present can detect 5 types of animals namely (elephant, zebra, giraffe, lion and cheetah).

.

### 4.1.1 DISADVANTAGES OF EXISTING SYSTEM:

* The proposed model achieved an accuracy of 88.9%. A comparison with other techniques was also carried out which showcased that their model outperformed previous approaches.
* Existing system accuracy of the model is low and time taken for prediction is high.
* .

## 4.2 PROPOSED SYSTEM:

Here, we present a system for the arrangement of fauna pictures, which will help biologist and researchers to additional review or potentially further develop territory, natural and eradication designs. Figure 1 shows the proposed plan of model for Fauna Image Order utilizing Convolutional Neural Network. The Aminal-10 dataset[6] utilized for the grouping is taken from Kaggle. We are utilizing Convolutional Neural Network with Broken ReLU initiation capacity and VGG16 design for our model. The underlying advance focuses on formation of elements with VGG16 model. Utilization of Image Processing along with Loading, Testing, Training, and Validating the dataset before the preparation step makes a difference to eliminate the noise, obstructions, smoothness and dust from the pictures. The subsequent stage employments Convolutional Neural Network alongside Leaky ReLU to train the model to accurately and definitively group animals classes..

.

### 4.2.1 ADVANTAGES OF PROPOSED SYSTEM:

Accuracy of the model is high compare to existing system

Time taken for prediction is low

Can train as many animals as possible and automates prediction process.

# CHAPTER- 5

# SYSTEM design

## 5.1 introduction

System Design Introduction:

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

## 5.2 modules

### 5.2.1 Data COLLECTION:

IN this project we use animal dataset collected from Kaggle which has three categories of 12 categoreis. Pixel values form images are taken as input and labels are used as output and each folder has 50 images which are used for training.

### 5.2.2 Pre-processing:

Pre-processing is a procedure adopted to enhance the quality of images and increase visualization. In animal imaging, image processing is a crucial phase that helps to improve the images quality. This can be one of the most critical factors in achieving good results and accuracy in next phases of proposed methodology. wound images may contain a different issue that may lead to poor and low visualization of the image. If the images are poor or of low quality, it may lead to unsatisfactory results. During preprocessing phase, we performed background elimination, elimination of non-essential blood supplies, image enhancement, and noise removal.

### 5.2.3 Train-Test Split and Model FITTING:

Now, we divide our dataset into training and testing data. Our objective for doing this split is to assess the performance of our model on unseen data and to determine how well our model has generalized on training data. This is followed by a model fitting which is an essential step in the model building process.

**Model Evaluation and Predictions:**

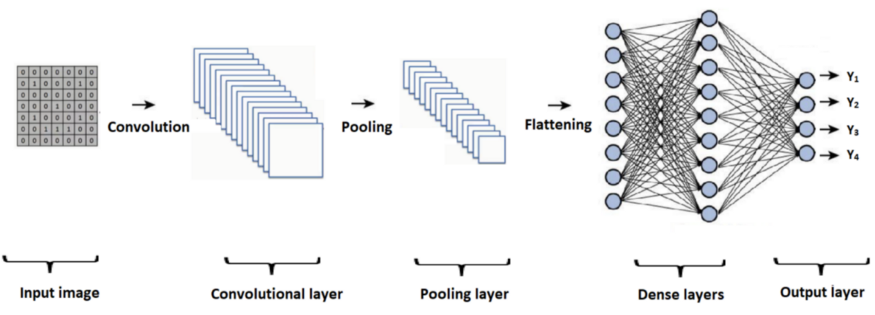
This is the final step, in which we assess how well our model has performed on testing data using certain scoring metrics, I have used 'accuracy score' to evaluate my model. First, we create a model instance, this is followed by fitting the training data on the model using a fit method and then we will use the predict method to make predictions on x\_test or the testing data, these predictions will be stored in a variable called y\_test\_hat. For model evaluation, we will feed the y\_test and y\_test\_hat into the accuracy\_score function and store it in a variable called test\_accuracy, a variable that will hold the testing accuracy of our model. We followed these steps for a variety of classification algorithm models and obtained corresponding test accuracy scores.

**Algorithms**

# ****CNN Architecture, Process & Inputs****

**Architecture:**

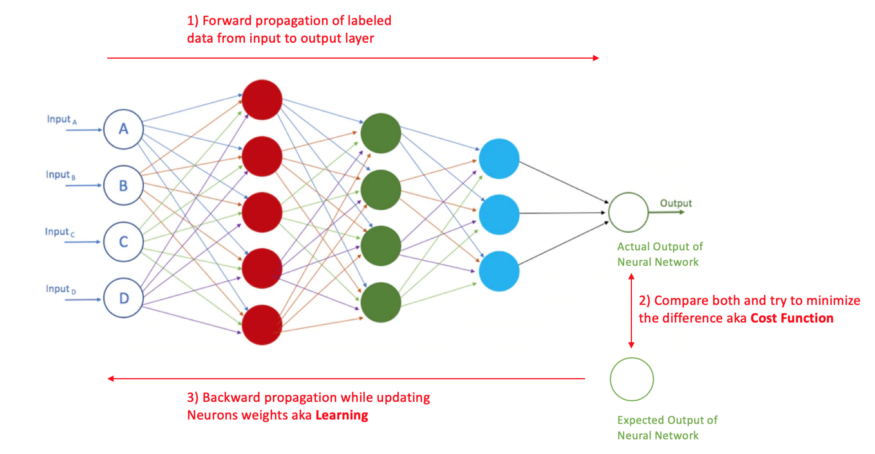
CNNs contain a combination of layers which transform an image into output the model can understand.



* Convolutional layer: creates a feature map by applying a filter that scans the image several pixels at a time
* Pooling layer: scales down the information generated by the convolutional layer to effectively store it
* Fully connected input layer: flattens the outputs into a single vector
* Fully connected layer: applies weights over the inputs generated by the feature analysis
* Fully connected output layer: generates final probabilities to determine the image class

**Process:**

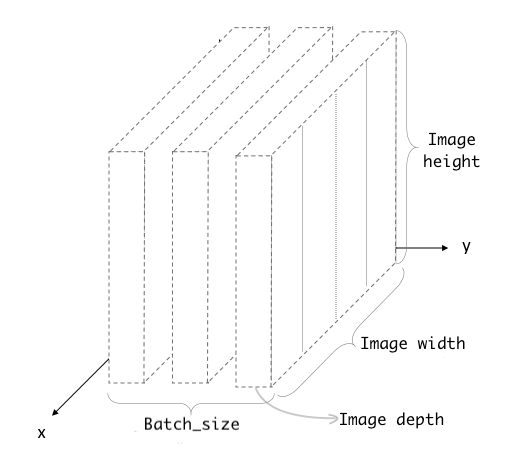
Forward and backward propagation iterate through all of the training samples in the network until the optimal weights are determined and only the most powerful and predictive neurons are activated to make a prediction



* The model trains throughout many epochs by taking one forward and one backward pass of all training samples each time
* Forward propagation calculates the loss and cost functions by comparing the difference between the actual and predicted target for each labeled image
* Backward propagation uses gradient descent to update the weights and bias for each neuron, attributing more impact on the neurons which have the most predictive power, until it arrives to an optimal activation combination
* As the model sees more examples, it learns to better predict the target causing the loss measure to decrease
* The cost function takes the average loss across all samples indicating overall performance

**Inputs:**

Model inputs always have to be in a 4D array consisting of (batch\_size, height, width, depth)



* Batch size: The number of training examples in one epoch (the higher the batch size, the more memory you’ll need)
* Height & Width: Pixel dimensions of your image
* Depth: Red, Green or Blue (3), or Black & White (1)

# My VGG19 Model

Below is an 8 step configuration of my best performing VGG19 model. VGG19 is an advanced CNN with pre-trained layers and a great understanding of what defines an image in terms of shape, color, and structure. VGG19 is very deep and has been trained on millions of diverse images with complex classification tasks. I didn’t train VGG19 any further, only froze its layers and appended a shallow 2 layer network on top of it to perform my classification task of identifying images with and without trees.

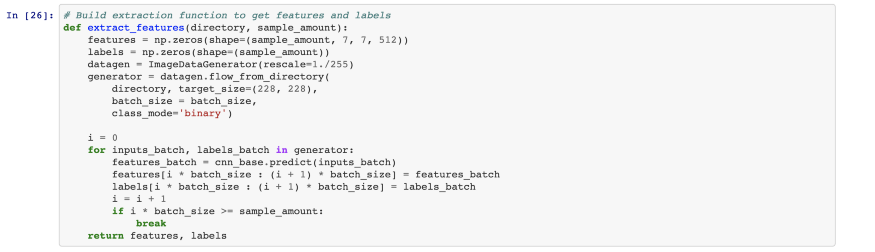
1. Load your model.



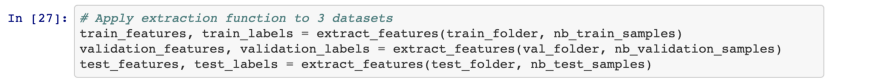
2. Load your data set size. In this case, the photos designated for training, testing and validation have already been randomly shuffled into different folders and manually separated between those with (target = 1) and without trees (target = 0).



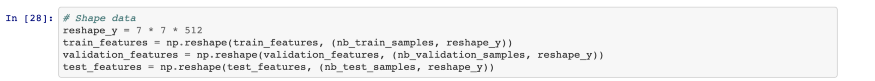
3. Set up a function to extract and freeze the VGG-19’s initial layers which process the features and labels of images under the hood. This will allow the model to apply transfer learning wherein it can recall its pre-training from millions of photos on the web.



4. Apply the function to your training, validation & test datasets so it extracts the features and labels from all of them.



5. Make sure data is in the right shape to reflect the dimensions of your datasets.



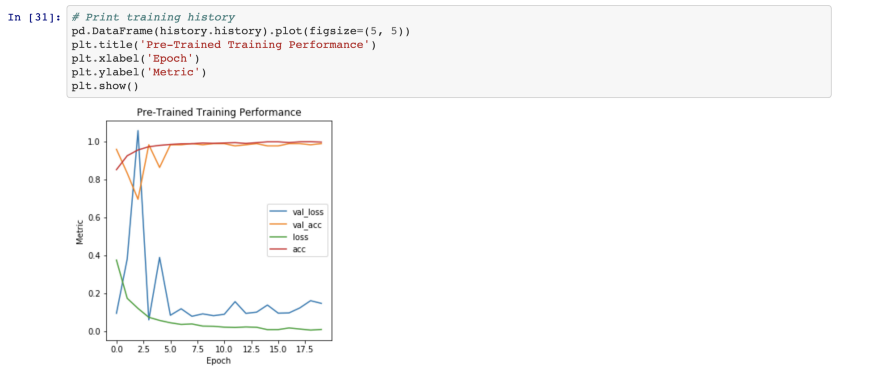
6. Save extracted features and labels into a ‘bottlenecked’ folder for your final classifying layer to refer to and conclude which binary category an image belongs to.



7. Build the image classifier final layer on top of the VGG-10 “brain” and put it to work.



8. To visualize how well your model learned using your accuracy and loss metrics, print your training history. As you can see, with each epoch (or iteration), our accuracy increase and loss decreased.



Beyond experimenting with different model hyper-parameters, here are some more steps you can take to improve your model:

**To improve model performance**, add adversity so your model can learn to recognize your target even in unlikely circumstances:

* Flip the image direction
* Incorporate images that resemble your target
* Add blurred and unsharpened versions

**To improve model efficiency,** reduce the model load:

* Perform max pooling in between the layers to reduce image dimensionality by compressing spatial size and parameters
* Try early stopping to prevent overfitting; if your model reaches a peak accuracy, it will stop looking for an even higher accuracy after a specific number of epochs
* Train on fewer epochs to cut the processing time
* Try a different activation function, like ReLU which only activates certain neurons, making it more efficient compared to sigmoid or tanh
* Try dropout so randomly selected neurons are ignored during training, thus creating less network computation
* Avoid large pixel images as adding more image clarity doesn’t improve learning much (224 by 224 pixels is standard)

## 5.3 system architecture

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system. Organized in a way that supports reasoning about the structures and behaviors of the system.

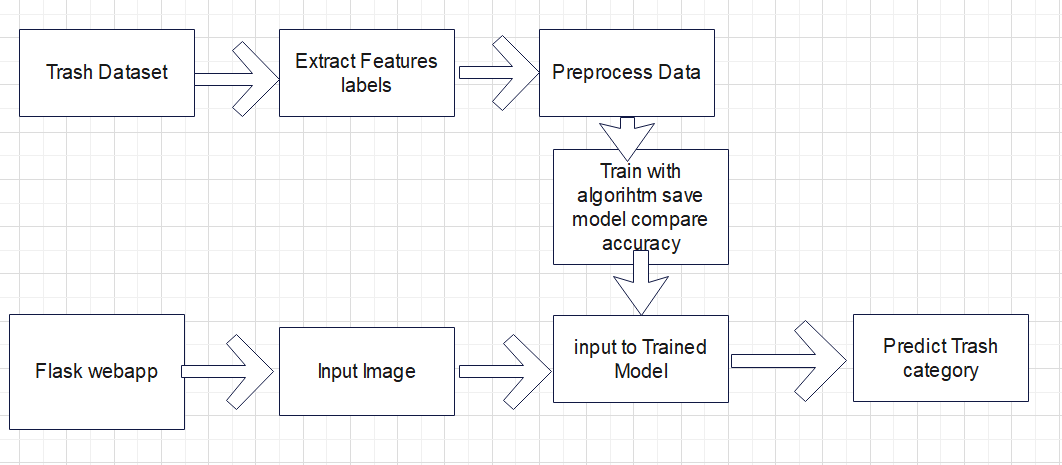


Figure 5. 1 System Architecture

3-Tier Architecture:

The three-tier software architecture (a three-layer architecture) emerged in the 1990s to overcome the limitations of the two-tier architecture. The third tier (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and can accommodate hundreds of users (as compared to only 100 users with the two tier architecture) by providing functions such as queuing, application execution, and database staging.

The three tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user. These characteristics have made three layer architectures a popular choice for Internet applications and net-centric information systems.

**Advantages of Three-Tier:**

* Separates functionality from presentation.
* Clear separation – better understanding.
* Changes limited to well define components.
* Can be running on WWW.
* Effective network performance.

## 5.4 UML DAIGRAMS

Global Use Case Diagrams:

Identification of actors:

Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation:



<<Actor name>>

An actor is someone or something that:

Interacts with or uses the system.

* Provides input to and receives information from the system.
* Is external to the system and has no control over the use cases.

Actors are discovered by examining:

* Who directly uses the system?
* Who is responsible for maintaining the system?
* External hardware used by the system.
* Other systems that need to interact with the system.

Questions to identify actors:

* + Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?
  + Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
  + Which external hardware or systems (if any) use the system to perform tasks?
  + What problems does this application solve (that is, for whom)?
  + And, finally, how do users use the system (use case)? What are they doing with the system?

The actors identified in this system are:

1. System Administrator
2. Customer
3. Customer Care

Identification of use cases:

Use case: A use case can be described as a specific way of using the system from a user’s (actor’s) perspective.

Graphical representation:



A more detailed description might characterize a use case as:

* Pattern of behavior the system exhibits
* A sequence of related transactions performed by an actor and the system
* Delivering something of value to the actor

Use cases provide a means to:

* capture system requirements
* communicate with the end users and domain experts
* test the system

Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

Guide lines for identifying use cases:

* For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a course of events that leads to clear goal
* Name the use cases.
* Describe the use cases briefly by applying terms with which the user is familiar.

This makes the description less ambiguous

Questions to identify use cases:

* What are the tasks of each actor?
* Will any actor create, store, change, remove or read information in the system?
* What use case will store, change, remove or read this information?
* Will any actor need to inform the system about sudden external changes?
* Does any actor need to inform about certain occurrences in the system?
* What usecases will support and maintains the system?

**1.2 Flow of Events**

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

* When and how the use case starts and ends
* Use case/actor interactions
* Data needed by the use case
* Normal sequence of events for the use case
* Alternate or exceptional flows

### 5.4.1 Construction of Use case diagrams:

FLOW CHART:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

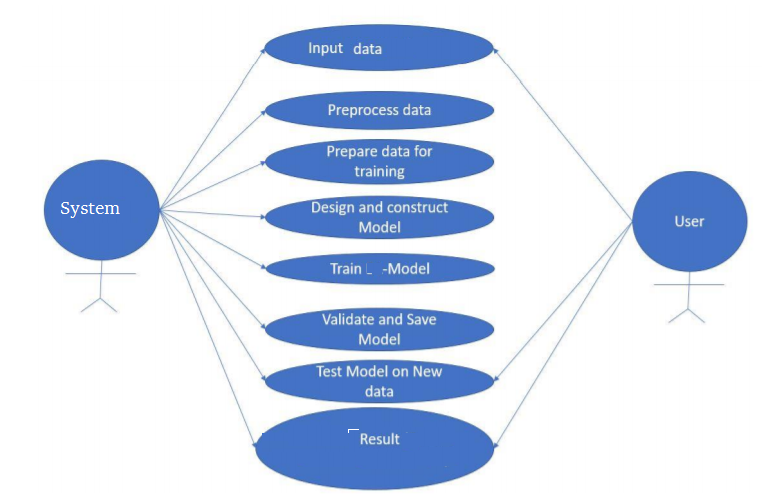


Figure 5. 2 Use Case Diagram

### 5.4.2 SEQUENCE DIAGRAMS:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

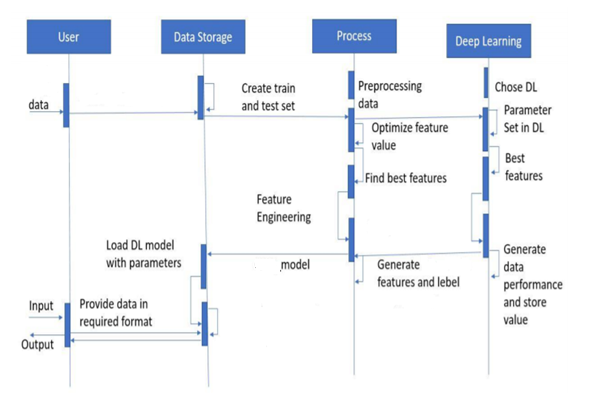


Figure 5. 3 Sequence diagram

### 5.4.3 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

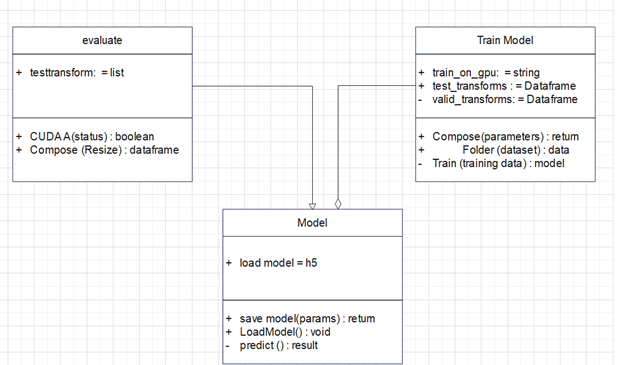


Figure 5. 4 Class Diagram

### 5.4.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

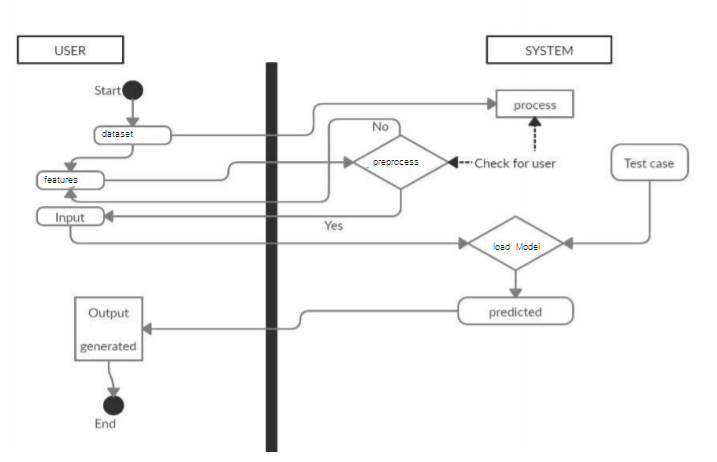
****

Figure 5. 5 Activity Diagram

# 

# CHAPTER-6

# system requirements

## 6.1 SYSTEM REQUIREMENTS

### 6.1.1 HARDWARE REQUIREMENTS:

* System : Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz
* Hard Disk : 1 TB.
* Input Devices : Keyboard, Mouse
* Ram : 4 GB.

### 6.1.2 SOFTWARE REQUIREMENTS:

* Operating system : Windows XP/7/10.
* Coding Language : Python
* Tool : Anaconda
* Interface : OPENCV

# Chapter-7

# System implementation

To conduct studies and analyses of an operational and technological nature, and To promote the exchange and development of methods and tools for operational analysis as applied to defense problems.

## 7.1 input and output designs

### 7.1.1 Logical design

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design are included. Logical design includes ER Diagrams i.e. Entity Relationship Diagrams

### 7.1.2 Physical design

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified / authenticated, how it is processed, and how it is displayed as output. In Physical design, following requirements about the system are decided.

1. Input requirement,
2. Output requirements,
3. Storage requirements,
4. Processing Requirements,
5. System control and backup or recovery.

Put another way, the physical portion of systems design can generally be broken down into three sub-tasks:

1. User Interface Design
2. Data Design
3. Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor. The H/S personal specification is developed for the proposed system.

## 7.2 INPUT & OUTPUT REPRESENTATION

### 7.2.1 Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

### 7.2.2 Objectives

Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

### Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

* 1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
  2. Select methods for presenting information.
  3. Create document, report, or other formats that contain information produced by the system.

**Code**

from tensorflow.keras.layers import Input,Lambda,Dense,Flatten

from tensorflow.keras.models import Model

from tensorflow.keras.preprocessing import image

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

import numpy as np

from glob import glob

import matplotlib.pyplot as plt

IMAGE\_SIZE = [224,224]

train\_path = "cancer/"

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale=1./255,horizontal\_flip=True,zoom\_range=0.2,validation\_split=0.15)

training\_set = train\_datagen.flow\_from\_directory(

train\_path,target\_size=(224,224), batch\_size=32,class\_mode='categorical',

subset='training')

validation\_set = train\_datagen.flow\_from\_directory(

train\_path,target\_size=(224,224), batch\_size=32,class\_mode='categorical',shuffle = True,

subset='validation')

from tensorflow.keras.applications import VGG19

from tensorflow.keras.layers import GlobalAveragePooling2D,Dropout

## We are initialising the input shape with 3 channels rgb and weights as imagenet and include\_top as False will make to use our own custom inputs

mv = VGG19(input\_shape=IMAGE\_SIZE+[3],weights='imagenet',include\_top=False)

for layers in mv.layers:

layers.trainable = False

# if u want to add more folders and train then change number 4 to 5 or 6 based on folders u have to train

x = Flatten()(mv.output)

prediction = Dense(4,activation='softmax')(x)

# In[7]:

model = Model(inputs=mv.input,outputs=prediction)

model.summary()

import tensorflow as tf

class myCallback(tf.keras.callbacks.Callback):

def on\_epoch\_end(self,epoch,logs={}):

if(logs.get('loss')<=0.05):

print("\nEnding training")

self.model.stop\_training = True

# initiating the myCallback function

callbacks = myCallback()

## Let us compile the model with Adam optimizer and loss function categorical\_crossentropy and metrics as categorical\_accuracy

from tensorflow.keras.optimizers import Adam

model.compile(optimizer=Adam(lr=0.0001),loss='categorical\_crossentropy',metrics=['categorical\_accuracy'])

history = model.fit(training\_set,

validation\_data=validation\_set,

epochs=50,

verbose=1,

steps\_per\_epoch=len(training\_set),

validation\_steps=len(validation\_set),

callbacks = [callbacks]

)

acc = history.history['categorical\_accuracy']

val\_acc = history.history['val\_categorical\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(len(acc))

import matplotlib.pyplot as plt

plt.plot(epochs,acc)

plt.plot(epochs,val\_acc)

plt.title("Training and validation Accuracy")

plt.savefig('accuracytrain.png')

plt.plot(epochs,loss)

plt.plot(epochs,val\_loss)

plt.title("Training and validation Loss")

plt.savefig('validationaccuracy.png')

model.save("cancer.h5")

#from tensorflow.keras.models import load\_model

#from tensorflow.keras.preprocessing import image

#import numpy as np

# dimensions of our images

#img\_width, img\_height = 224,224

# load the model we saved

#model = load\_model('content.h5')

# predicting images

#img = image.load\_img('content/Train/Cars/C0.jpg', target\_size=(img\_width, img\_height))

#x = image.img\_to\_array(img)

#x = np.expand\_dims(x, axis=0)

#classes = model.predict(x)

#print (classes)

# Chapter-8

# System testing

## 8.1 INTRODUCTION:

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands.

Testing objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say, testing is a process of executing a program with intent of finding an error.

1. A successful test is one that uncovers an as yet undiscovered error.
2. A good test case is one that has probability of finding an error, if it exists.
3. The test is inadequate to detect possibly present errors.
4. The software more or less confirms to the quality and reliable standards.

## 8.2 Levels of Testing

Code testing:

This examines the logic of the program. For example, the logic for updating various sample data and with the sample files and directories were tested and verified.

Specification Testing:

Executing this specification starting what the program should do and how it should performed under various conditions. Test cases for various situation and combination of conditions in all the modules are tested.

Unit testing:

In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example, the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system.

Each Module can be tested using the following two Strategies:

1. Black Box Testing
2. White Box Testing

### 8.2.1 BLACK BOX TESTING

What is Black Box Testing?

Black box testing is a software testing techniques in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.

In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.



The above Black Box can be any software system you want to test. For example : an operating system like Windows, a website like Google ,a database like Oracle or even your own custom application. Under Black Box Testing , you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Black box testing - Steps

Here are the generic steps followed to carry out any type of Black Box Testing.

* Initially requirements and specifications of the system are examined.
* Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
* Tester determines expected outputs for all those inputs.
* Software tester constructs test cases with the selected inputs.
* The test cases are executed.
* Software tester compares the actual outputs with the expected outputs.
* Defects if any are fixed and re-tested.

Types of Black Box Testing

There are many types of Black Box Testing but following are the prominent ones -

* Functional testing – This black box testing type is related to functional requirements of a system; it is done by software testers.
* Non-functional testing – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements  such as performance, scalability, usability.
* Regression testing – Regression testing is done  after code fixes , upgrades or any other system maintenance to check the new code has not affected the existing code.

### 8.2.2 WHITE BOX TESTING

White Box Testing is the testing of a software solution's internal coding and infrastructure. It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability.White box testing is also known as clear, open, structural, and glass box testing.

It is one of two parts of the "box testing" approach of software testing. Its counter-part, blackbox testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing. The term "whitebox" was used because of the see-through box concept. The clear box or whitebox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "black box testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested

WHAT DO YOU VERIFY IN WHITE BOX TESTING?

White box testing involves the testing of the software code for the following:

* Internal security holes
* Broken or poorly structured paths in the coding processes
* The flow of specific inputs through the code
* Expected output
* The functionality of conditional loops
* Testing of each statement, object and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

HOW DO YOU PERFORM WHITE BOX TESTING?

  To give you a simplified explanation of white box testing, we have divided it into **two basic steps**. This is what testers do when testing an application using the white box testing technique:

**STEP 1) UNDERSTAND THE SOURCE CODE**

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

**Step 2) CREATE TEST CASES AND EXECUTE**

The second basic step to white box testing involves testing the application’s source code for proper flow and structure. One way is by writing more code to test the application’s source code. The tester will develop little tests for each process or series of processes in the application. This  method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include manual testing, trial and error testing and the use of testing tools as we will explain further on in this article.

Unit testing:

|  |  |
| --- | --- |
| Sl # Test Case : ­ | UTC­1 |
| Name of Test: ­ | Load dataset |
| Items being tested: ­ | Dataset features and labels are displayed or not |
| Sample Input: ­ | Dataset IMAGE file |
| Expected output: ­ | All features and labels should be displayed |
| Actual output: ­ | Total data is displayed |
| **Remarks: ­** | **Pass.** |

|  |  |
| --- | --- |
| Sl # Test Case : ­ | UTC­2 |
| Name of Test: ­ | Split data |
| Items being tested: ­ | Data is divided in to train and test set |
| Sample Input: ­ | Test and train size |
| Expected output: ­ | Dataset is divided in to 2 parts |
| Actual output: ­ | Based on given test size data is divided and stored in train and test sets |
| Remarks: ­ | pass |

**Integration Testing:**

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. It occurs after unit testing and before validation testing. Integration testing can be done in two ways: Bottom­up integration testing and Top­down integration testing.

* + 1. **Bottom­up Integration**

This testing begins with unit testing, followed by tests of progressively higher­level combinations of units called modules or builds.

* + 1. **Top­down Integration**

In this testing, the highest­level modules are tested first and progressively, lower­level modules are tested thereafter.

In a comprehensive software development environment, bottom­up testing is usually done first, followed by top­down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic actual situations. Table 6.5 shows the test cases for integration testing and their results

|  |  |
| --- | --- |
| Sl # Test Case : ­ | ITC­1 |
| Name of Test: ­ | Train Model |
| Item being tested: ­ | Model fit is performed |
| Sample Input: ­ | Train x and train y |
| Expected output: ­ | Fit is performed |
| Actual output: ­ | Training is done and accuracy is displayed |
| Remarks: ­ | Pass. |

|  |  |
| --- | --- |
| Sl # Test Case : ­ | ITC­2 |
| Name of Test: ­ | Accuracy calculation |
| Item being tested: ­ | If accuracy of each algorithm is calculated |
| Sample Input: ­ | Test x and test y |
| Expected output: ­ | Accuracy of each algorithm |
| Actual output: ­ | Accuracy of each model |
| Remarks: ­ | Pass. |

**System testing**:

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black­box testing, and as such, should require no knowledge of the inner design of the code or logic. System testing is important because of the following reasons:

System testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.

The application is tested thoroughly to verify that it meets the functional and technical specifications.

The application is tested in an environment that is very close to the production environment where the application will be deployed.

System testing enables us to test, verify, and validate both the business requirements as well as the application architecture.

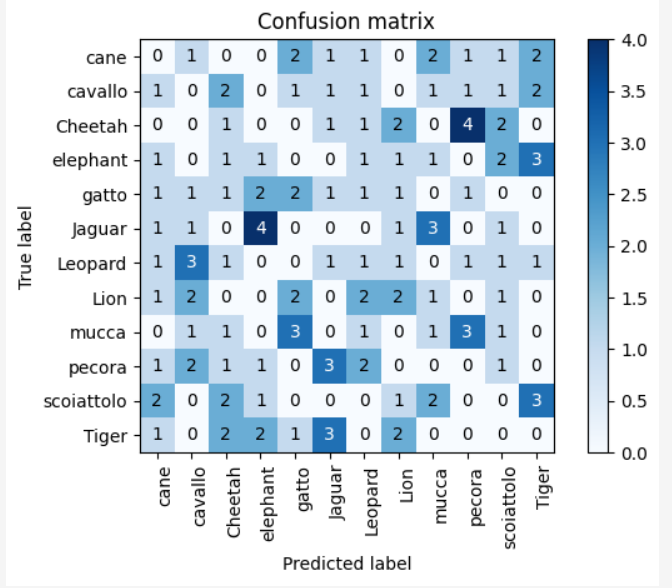
System Testing is shown in below tables

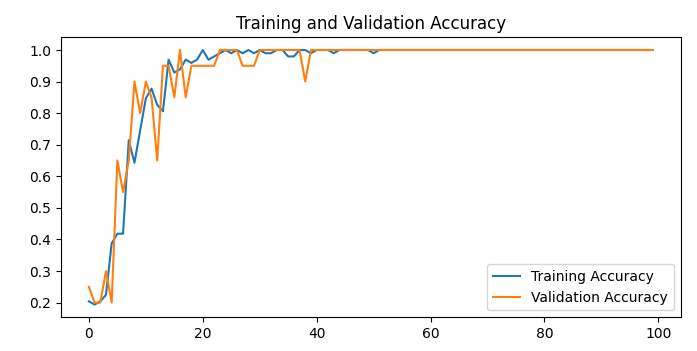
|  |  |
| --- | --- |
| Sl # Test Case : ­ | STC­1 |
| Name of Test: ­ | System testing in various versions of OS |
| Item being tested: ­ | OS compatibility. |
| Sample Input: ­ | Execute the program in windows XP/ Windows­7/8 |
| Expected output: ­ | Performance is better in windows­7 |
| Actual output: ­ | Same as expected output, performance is better in windows­7 |
| Remarks: ­ | Pass |

# CHAPTER-9

# Output Screens

## 9.1 Dataset SCREEN





# CONCLUSION

In conclusion, this model has accuracy around 96% after many attempts of optimization in terms of hyperparameters and model structures. With this model, it can distinguish normal animals and rare animals in a very short time. Despite striking similarities existed in biology, groups of animals do differ in some ways [6]. Thus, the model can be trained to help humans effectively protect rare specials of animals from other common creatures. On the other hand, the model is possible to find out new species with anomalous behavior, like much lower test accuracy. Nevertheless, the model has some limitations. The most obvious one is the dataset, which has only two categories. With this model, only the snub-nosed monkey can be identified in the group of monkeys. It does not work in other kinds of rare monkeys, including sapajou, Hainan gibbons, etc. [7]. Besides, the model cannot be used to identify a particular creature in multitype animals because of this limitation. One of the solutions is to enhance image numbers and image categories. It would be another difficulty for constructing this model because a large amount of training data is required. In addition, this binary classification in this CNN model can be improved to multiclass image classification. Thus, the model can identify more than one types of rare animals at the same time. Moreover, there are 2D convolutional kernels used in this model, which can utilize context across the width and height of single slice to make predictions [8]. However, compared with applying 3D convolutional kernels which can capture voxel information from adjacent slices (whole volume as inputs), 2D CNN (single slice as inputs) has lower accuracy, and it can only process 2D images. Although some limitations existed in this CNN model, there are many spaces for improvements and more applications in a wider range of area. The development of CNN image recognition technology is still on the road. With more complex convolution neural network, it has high computational cost and high configuration requirement [9]. Moreover, one of the core difficulties for complex neural network is associated with the training dataset as it is the foundation for neural network learning. Along with accomplishing these challenges, the accuracy for image classification will be improved further and the network will be more concise in the future [9].

# 

# REFERENCES

[1] W. Rawat and Z. Wang, Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review, in Neural Computation, vol. 29, no. 9, Sept. 2017, pp. 2352-2449.

[2] Kavish Sanghvi, Adwait Aralkar, Saurabh Sanghvi and Ishani Saha, Fauna Image Classification using Convolutional Neural Network, International Journal of Future Generation Communication and Networking Vol. 13, No. 1s, (2020), pp. 08-16.

[3] Gousia Habib, Shaima Qureshi, Optimization and acceleration of convolutional neural networks: A survey, Journal of King Saud University - Computer and Information Sciences, October 20, 2020.

[4] T. Sinha, B. Verma and A. Haidar, “Optimization of convolutional neural network parameters for image classification”, 2017 IEEE Symposium Series on Computational Intelligence (SSCI), 2017, pp. 1-7.

[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting”, The Journal of Machine Learning Research, vol. 15, no. 2014, pp. 1929-1958.

[6] S. Y. Chaganti, I. Nanda, K. R. Pandi, T. G. N. R. S. N. Prudhvith and N. Kumar, Image Classification using SVM and CNN, 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA), 2020, pp. 1-5.

[7] Yang Y, Tian Y, He, C, Huang Z, Dong S, Wang B, Xiao W. The Critically Endangered Myanmar snub-nosed monkey Rhinopithecus strykeri found in the Salween River Basin, China. Oryx, 2018, 52(1), 134-136

. [8] M. Zreik, T. Leiner, B. D. de Vos, R. W. van Hamersvelt, M. A. Viergever and I. Išgum, Automatic segmentation of the left ventricle in cardiac CT angiography using convolutional neural networks, 2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI), 2016, pp. 3-4.

[9] Jiaohua Qin, Wenyan Pan, Xuyu Xiang, Yun Tan, Guimin Hou, A biological image classification method based on improved CNN, Ecological Informatics, vol.58, July 2020, pp. 1-2.