Jyothi Hills, Panjal Road,
Vettikattiri PO, Cheruthuruthy , Thrissur,
Kerala 679531

Jyothi Engineering College
NAAC Accredited College with NBA Accredited Programmes*
Approved by AICTE & affiliated to APJ Abdul Kalam Technological University
A CENTRE OF EXCELLENCE IN SCIENCE & TECHNOLOGY BY THE CATHOLIC ARCHDIOCESE OF TRICHUR
JYOTHI HILLS, VETTIKATTIRI P.O, CHERUTHURUTHY, THRISSUR. PIN-679531 PH : +91- 4884-259000, 274423 FAX : 04884-274777
NBA accredited B.Tech Programmes in Computer Science & Engineering, Electronics & Communication Engineering, Electrical & Electronics Engineering and Mechanical Engineering valid for the academic years 2016-2022. NBA accredited B.Tech Programme in Civil Engineering valid for the academic years 2019-2022.

Snakebite Detection & Identification With Snakebite Mark Using Machine Learning Approach

MAIN PROJECT REPORT

MARY JOSE (JEC17CS064)

SARANYA K (JEC17CS090)

SIJIN K (JEC17CS096)

YASHIF V S (JEC17CS104)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY (B.Tech)

in

COMPUTER SCIENCE & ENGINEERING

of

A P J ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Under the guidance of

Dr. ASWATHY S U

CREATING TECHNOLOGY LEADERS OF TOMORROW

JUNE 2021

Department of Computer Science & Engineering

DECLARATION

We the undersigned hereby declare that the project report "Snakebite Detection Identification With Snakebite Mark Using Machine Learning Approach", submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Dr. Aswathy S U. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in this submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously used by anybody as a basis for the award of any degree, diploma or similar title of any other University.

Name of Students **Signature**

MARY JOSE (JEC17CS064)

SARANYA K (JEC17CS090)

SIJIN K (JEC17CS096)

YASHIF V S (JEC17CS104)

Place:

Date:

Department of Computer Science and Engineering

JYOTHI ENGINEERING COLLEGE, CHERUTHURUTHY

THRISSUR 679 531

JUNE 2021

BONAFIDE CERTIFICATE

This is to certify that the main project report entitled **Snakebite Detection & Identification With Snakebite Mark Using Machine Learning Approach** submitted by **Mary Jose (JEC17CS064), Saranya K (JEC17CS090), Sijin K (JEC17CS096) and Yashif V S (JEC17CS106)** in partial fulfillment of the requirements for the award of **Bachelor of Technology** degree in **Computer Science and Engineering** of **A P J Abdul Kalam Technological University** is the bonafide work carried out by them under our supervision and guidance.

Dr. Aswathy S U	**Mr. Shaiju Paul**	**Dr Saju P John**
Project Guide	Project Coordinator	Head of The Dept
Professor	Assistant Professor	Professor
Dept. of CSE	Dept. of CSE	Dept. of CSE

DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

COLLEGE VISION

Creating eminent and ethical leaders through quality professional education with emphasis on holistic excellence.

COLLEGE MISSION

- To emerge as an institution par excellence of global standards by imparting quality engineering and other professional programmes with state-of- the-art facilities.
- To equip the students with appropriate skills for a meaningful career in the global scenario.
- To inculcate ethical values among students and ignite their passion for holistic excellence through social initiatives.
- To participate in the development of society through technology incubation, entrepreneurship and industry interaction.

DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

DEPARTMENT VISION

Creating eminent and ethical leaders in the domain of computational sciences through quality professional education with a focus on holistic learning and excellence.

DEPARTMENT MISSION

- To create technically competent and ethically conscious graduates in the field of Computer Science & Engineering by encouraging holistic learning and excellence.
- To prepare students for careers in Industry, Academia and the Government.
- To instill Entrepreneurial Orientation and research motivation among the students of the department.
- To emerge as a leader in education in the region by encouraging teaching, learning, industry and societal connect

PROGRAMME OUTCOMES (POs)

1. **Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem Analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/Development of Solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct Investigations of Complex Problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern Tool Usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The Engineer and Society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and Team Work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project Management and Finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-Long Learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

1. The graduates shall have sound knowledge of Mathematics, Science, Engineering and Management to be able to offer practical software and hardware solutions for the problems of industry and society at large.

2. The graduates shall be able to establish themselves as practising professionals, researchers or Entrepreneurs in computer science or allied areas and shall also be able to pursue higher education in reputed institutes.

3. The graduates shall be able to communicate effectively and work in multidisciplinary teams with team spirit demonstrating value driven and ethical leadership.

PROGRAMME SPECIFIC OUTCOMES (PSOs)

1. An ability to apply knowledge of data structures and algorithms appropriate to computational problems.

2. An ability to apply knowledge of operating systems, programming languages, data management, or networking principles to computational assignments.

3. An ability to apply design, development, maintenance or evaluation of software engineering principles in the construction of computer and software systems of varying complexity and quality.

4. An ability to understand concepts involved in modeling and design of computer science applications in a way that demonstrates comprehension of the fundamentals and trade-offs involved in design choices.

COURSE OUTCOMES (COs)

C410.1 The students will be able to analyse a current topic of professional interest and present it before an audience.

C410.2 Students will be able to identify an engineering problem, analyse it and propose a work plan to solve it.

C410.3 Students will have gained thorough knowledge in design, implementations and execution of Computer science related projects.

C410.4 Students will have attained the practical knowledge of what they learned in theory subjects.

C410.5 Students will become familiar with usage of modern tools.

C410.6 Students will have ability to plan and work in a team.

ACKNOWLEDGEMENT

We take this opportunity to express our heartfelt gratitude to all respected personalities who had guided, inspired and helped us in the successful completion of this project work. First and foremost, we express my thanks to **The Lord Almighty** for guiding us in this endeavour and making it a success.

We take immense pleasure in thanking the **Management** of Jyothi Engineering College and **Dr Sunny Joseph Kalayathankal**, Principal, Jyothi Engineering College for having permitted us to carry out this seminar. Our sincere thanks to **Dr Saju P John**, Head of the Department of Computer Science and Engineering for permitting us to make use of the facilities available in the department to carry out the project work successfully.

We express our sincere gratitude to **Mr. Shaiju Paul & Mrs. Swapna B Sasi**, Project Coordinators for their invaluable supervision and timely suggestions. We are very happy to express our deepest gratitude to our mentor **Dr. Aswathy S U**, Professor, Department of Computer Science and Engineering, Jyothi Engineering College for her able guidance and continuous encouragement.

Last but not least, we extend our gratefulness to all teaching and non-teaching staff who directly or indirectly involved in the successful completion of this project work and to all friends who have patiently extended all sorts of help for accomplishing this undertaking.

ABSTRACT

Making inaccurate assumptions of snakebite from just observation of visual features is a dangerous thing. If the assumption turns out to be wrong, it would cost a person's life. Snakebite envenomation has a huge impact on India, as the subcontinent is home to a multitude of venomous snakes. In India, there are 270 types of venomous snakes, of which 60 are considered venomous and medically relevant with various levels of toxicity. Under such a circumstance, if a person comes in contact with a snakebite, they panic and because most people won't be familiar with what to do, it may lead to panic-driven heart attacks or such extreme cases. Even a doctor can be wrong in identifying snake bites. This system helps a person determine venomous and non-venomous snakes based on their bite pattern image using two methods i.e, image processing and machine learning.The process of identification and recognition of distinct snake bites at the earliest requires convolutional neural networks to help in understanding the image which can help in giving more accurate results. Even doctors can use our system to identify the snake and start administering medication which can lead to a substantial decrease in the death rate.

Keywords: Convolutional Neural Network, Image processing, Machine learning, Detection, Identification, Deep learning.

CONTENTS

List of Figures

List of Abbreviations

CNN	: *Convolutional Neural Network*
AI	: *Artificial Intelligence*
BPNN	: *Back Propagation Neural Network*
PIL	: *Plot Image Learning*
KNN	: *K Nearest Neighbour*
LBP	: *Linear Binary Pattern*
GA	: *Genetic Algorithm*
KWS	: *Keyword Spotting*
JSW	: *Joint Space Width*
SDLC	: *Software Development Life Cycle*
PCL	: *Passive Coherent Location*
DCT	: *Discrete Cosine Transformation*
UAV	: *Unmanned Aerial Vehicle*
FFT	: *Fast Fourier Transformation*
CT	: *Computed Tomography*
OA	: *Osteo Arthritis*
SVM	: *Support Vector Machine*
RBF	: *Radial Basis Function*
CFAR	: *Constant False Alarm Rate*
DoA	: *Direction of Arrival*
MUSIC	: *Multiple Signal Classification*
HHO	: *Harris – Hawks Optimization*
MTH	: *Multilevel Threshold*
MCET	: *Minimum Cross Entropy Thresholding*
DCT	: *Discrete Cosine Transformation*
EKF	: *Extended Kalman Filter*
MRI	: *Magnetic Resonance Imaging*

CHAPTER 1
INTRODUCTION

1.1 Overview

Snakes are dangerous reptiles. They are divided into 2 groups, which are venomous and non-venomous snakes. If a person gets bitten by a poisonous snake, it can cause local poisoning, hypertension, paralysis, blood clotting disorders, and even sudden death. Currently, a syndromic approach is widely used for the treatment but, this strategy has limitations. Antivenom is the only sure therapy for handling snakebites. However, one type of antivenom that is given cannot neutralize all the toxins, so this creates a risk to randomly treat it, making the identification of snakes a critical problem. Snakes can be recognized by looking at the shape of their head, scales, and teeth. They can also be identified from the bite pattern. It is difficult to identify snakes by observing visual features directly, so we wanted to build a system that eradicates these limitations and supports the current syndromic approach. Our system is "Snakebite Detection and Identification with Snakebite Mark Using Machine Learning Approach". The system uses a set of snakebite images as data. Image processing and a Machine Learning approach are used to classify the bite marks. Based on research that has been done using the snake bite images, we are creating a system that classifies snakes based on the image of snakes bite patterns. Capturing the snake-bitten area and forming a digital image of that area leads to the classification. The image is processed using the Adaptive Gaussian Thresholding segmentation method. Then it is classified using the inception v3 algorithm. As a result, the snake can be identified easily.

1.2 Objectives

The prime objective of the system is to identify the snakebite from the bite image, and giving them out as result. It is so that it can help the person to get faster medical aid and altogether decrease the snakebite envenoming deaths to a certain length.

1.3 Data Description

As the data for this project is not publicly available the data for this project is taken from Internet and the data set is created. As it is the case with a usual deep learning problem, we would be training the model using training dataset.

1.4 Organization of the project

The report is organised as follow:

- **Chapter 1:Introduction** Gives an introduction to "Snakebite Detection and Identification with Snakebite Mark Using Machine Learning Approach".
- **Chapter 2:Literature Survey** Summarizes the various existing techniques that helps in achieving the desired result.
- **Chapter 3: Problem Statement** Discusses about the need for the proposed system
- **Chapter 4:Project Management** Contains the effective project management model to be used for the project.
- **Chapter 5:Methodology** Methods which are used in this project.
- **Chapter 6:Results** Results and Discussion needed for the implementation.
- **Chapter 7:Conclusion and Future Works** Concludes with the future scope of implementation.
- **Chapter 8:Summary of the Results**
- **Chapter 9:Screenshots** Includes the Screenshots of the Implementation code.
- **References:** Includes the references for the project.

CHAPTER 2
LITERATURE SURVEY

2.1 Image processing for snake identification based on bite using Local Binary Pattern and Support Vector Machine method

Snakes are one of dangerous reptiles that are divided into 2 groups, namely venomous and nonvenomous snakes. If bitten by a poisonous snake, it can cause local poisoning, hypotension, paralysis, blood clotting disorders, and even death. Antivenom is the only sure therapy for handling snakebites . However, in one type of antivenom that is given cannot neutralize all the toxins, so this creates a further risk to human health, making identification of snakes an important problem. Snakes can be identified by looking at the shape of the head, scales and teeth . To distinguish venomous and non-venomous snakes can be seen from the bite pattern. Figure 2.1 is example of picture bite pattern.

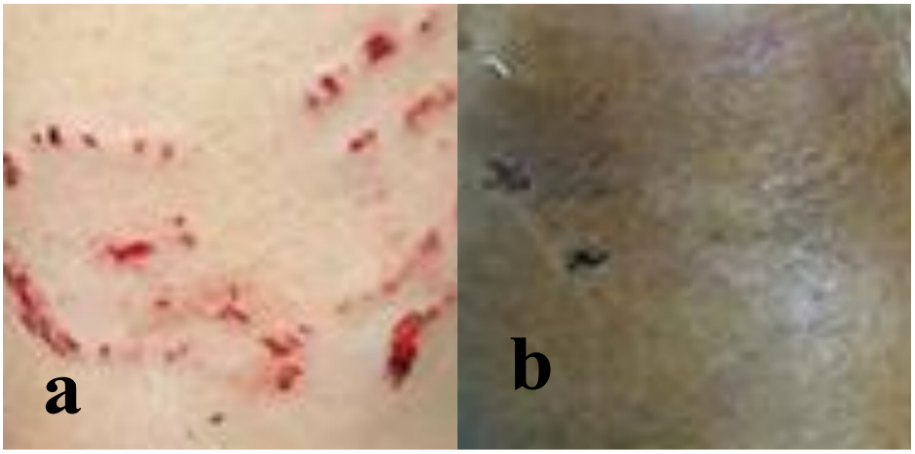


Figure 2.1: Example of picture (a) bite of snake not venomous and (b) venomous

It is difficult to identify snakes by observing visual features directly.The system classifies snakes based on snake body image seen from 4 perspectives: upper, side, bottom and body. The system built demonstrates the use of taxonomic features in the classification of snakes with the nearest neighbor classification, the system uses a snake image database and is converted to extract the taxonomic base features of snakes.

Based on research that has been done using snake body image, then in this study a system was built that classifies venomous and non-venomous snakes based on the image of snake(‘s) bite patterns. The method used is Local Binary Pattern (LBP) for feature extraction and uses the Support Vector Machine classification method or commonly abbreviated SVM.LBP is a method that labels pixels by threshing and considering the results as binary numbers. The nature of the LBP operator is a simple computing process and has a fast computing time. SVM is one of the classification methods with the advantage of being able to minimize errors in training sets, find the best hyperplane that separates 2 classes in feature space.In this research, an image processing system for bite identification based on bites using the LBP and SVM methods was built. The system is created using Matlab R2016a. Input to the system is an image with jpg format (* .jpg) measuring 96 × 96 pixels with a picture of someone who has bitten by a snake and the image has been cropped on the wound area. In this system only classifies venomous and non-venomous snakes without knowing the type of snake.

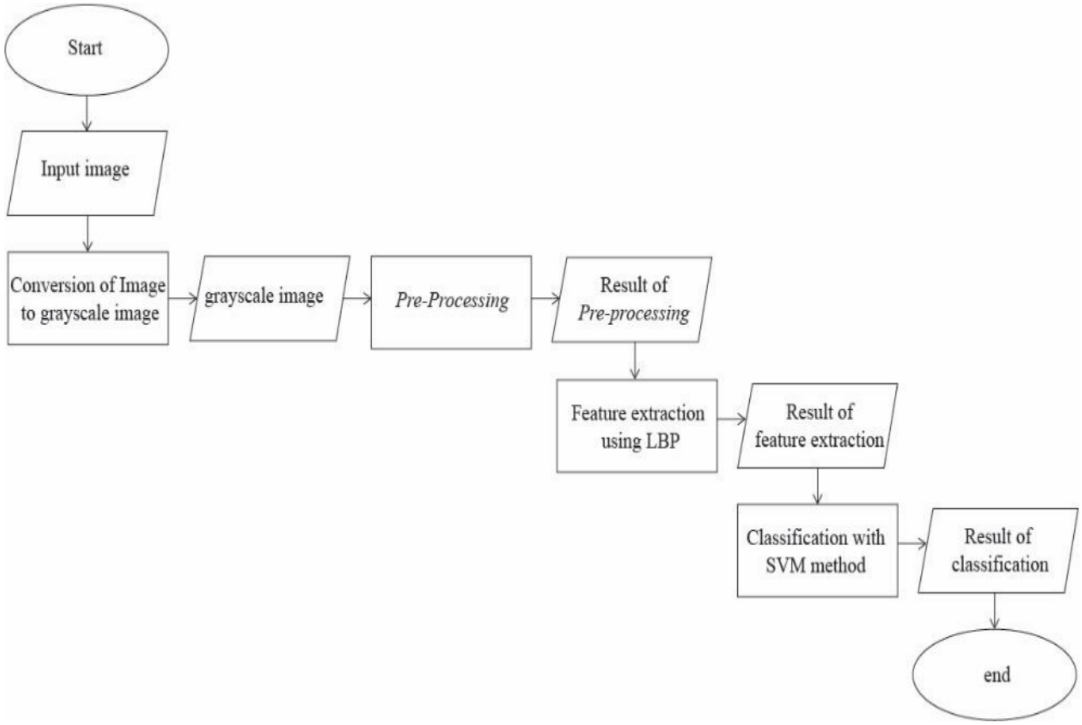


Figure 2.2: Flowchart

The thing that will be done on this system is starting with the image input, then the image is converted into a grayscale image, after that the image enters the pre-processing stage. In the pre-processing stage, morphology is carried out, namely erosion and dilation. The erosion process is done by reducing the pixels on the contour of the object with a 3 × 3 element

structure that has a value of 1 whose function is to eliminate the noise in the image, the noise is a pixel that is white but not an object of that image, this process causes pixels the object becomes thinner. Whereas dilation is processed by adding pixels to the object's contours according to the 3 × 3 element structure which is 1 to thick the pixels of After the pre-processing stage is carried out, the next is feature extraction using LBP, the end result of which is a characteristic of the image input in the form of a histogram, which is then represented as a 1 × n matrix. with the values generated in the normalized feature vector into the range 0-1. And finally the classification with the SVM method uses the RBF kernel. For more details, figure 2.2 is a flowchart or flow chart of this system.

Image data that has images with wrinkled or hairy skin or bruising on the bite or wound area that is still not dry and blood clots on the bite marks causes classification errors, errors in this classification because the system incorrectly detects snake bites, which should not snake bites but the system detects it is a snake bite resulting in poor accuracy. Conversely, if the image data does not have an image with the conditions previously explained, the system detects and classifies the image correctly. With 89% accuracy, it can be said that the system is good at recognizing the bite of a poisonous and nonvenomous snake. [5]

2.2 A Development of Snake Bite Identification System (N'viteR) using NEURO-GA

Globally, it is estimated that 5·4-5·5 million people are bitten by snakes each year resulting in about 400 000 amputations, 2.5 million envenoming and between 20 000 and 125 000 lost their life from snake bites. There are different types of bites even for venomous snakes and non venomous ones. Some bites from venomous ones may not be life threatening. It is crucial to differentiate between venomous and non-venomous snake whereby an immediate and effective medical care can be instituted to the victims. However, early identification is not easy. This was the reason to develop a snake bite identification system (N'viteR) to differentiate the snake using Neuro-GA technique. Based on 200 cases, this work had revealed that Neuro-GA has yield a high accuracy in identifying the snake. A number of experiments have been done which based on number of epoch, momentum, learning rate, number of generation, population and chromosome. This hybrid technique has achieved 97.6% of accuracy which enables early identification of snake and immediate specific antivenom can be administration. Hence, reduces the rate of morbidity and mortality. This paper proposed the snake bite identification system N'viteR based on the combination of two Artificial Intelligence (AI) techniques:

- Back Propagation Neural Network (BPNN)

- Genetic Algorithm (GA)

which coined the term Neuro-GA.

In order to develop N'viteR, there were multiple objectives to be considered the main ones among them were

1. To collect snake bite cases and identify the clinical effects of snake bites
2. To design and develop N'viteR using Neuro-GA
3. To evaluate N'viteR based on number of epoch, values of momentum and value of learning rate.

The Neuro-GA technique is proposed in N'viteR since it has an element of learning and optimization. The learning element owned by BPNN will possible the system to learn the data (cases) while the GA will help to chose the best (optimum) weight of cases to be tested with.

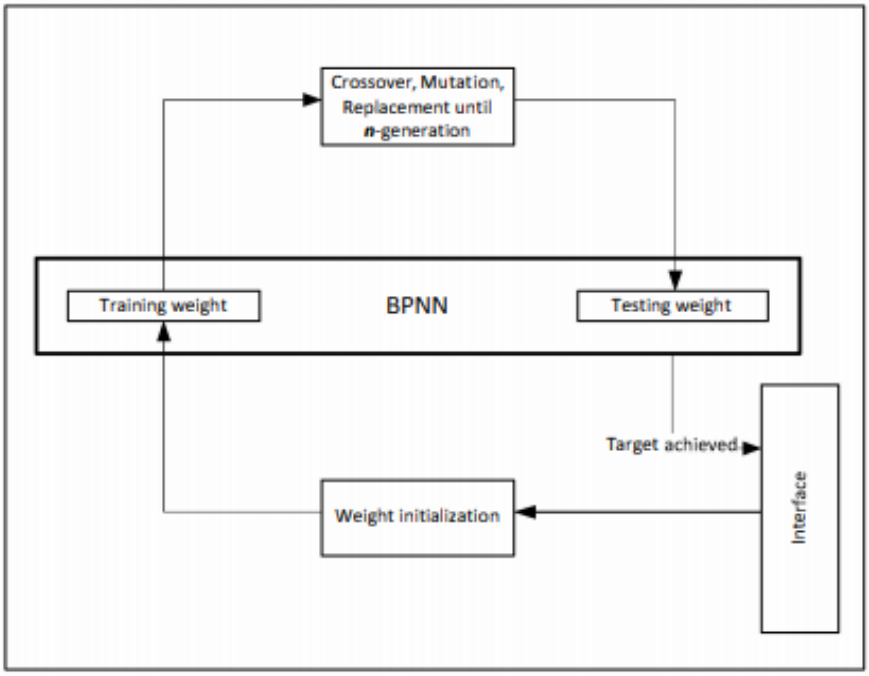


Figure 2: Architecture of N'viteR

Figure 2.3: N'viteR

The architecture of BPNN which consists of 32 input nodes which are the symptoms of snake bite and five hidden nodes that have been adjusted to produce a better result. The output node represents venomous and non-venomous snakes.

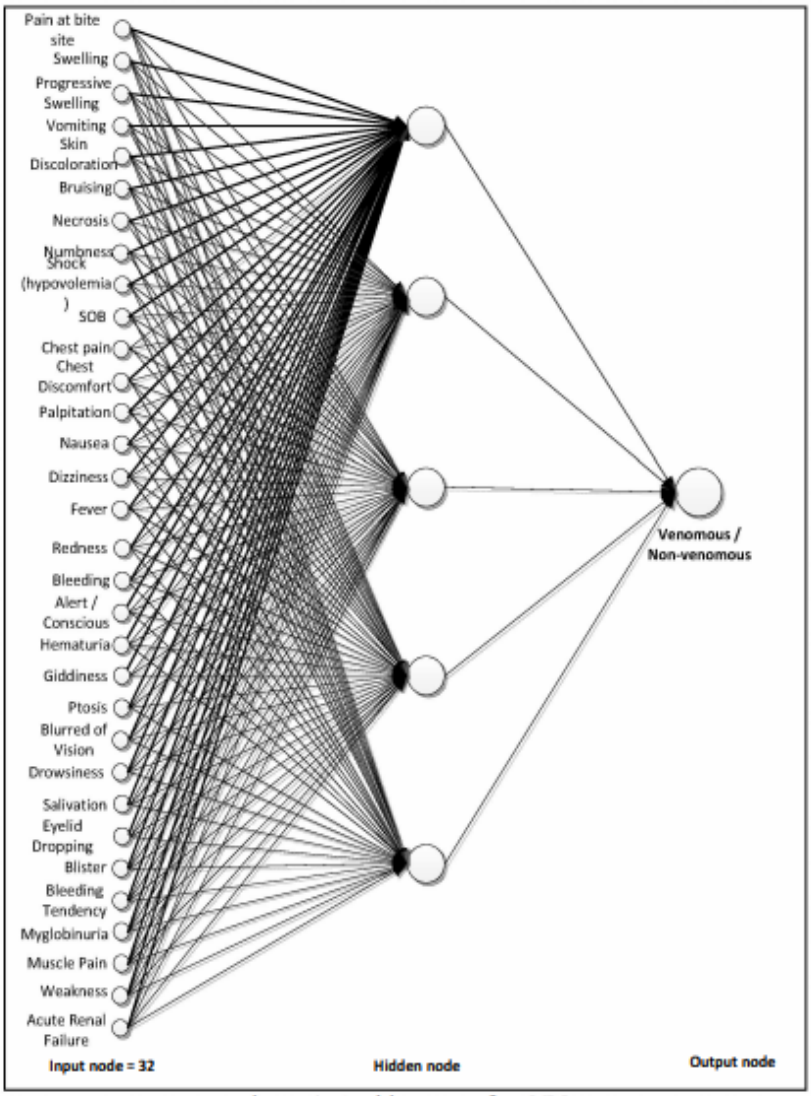


Figure 1: Architecture of BPNN

Figure 2.4: Back Propagation Neural Network

Even though BPNN is the best-known method to deal with classification problem through learning process, a combination with GA yields a high accuracy to identify a venomous and non-venomous snake based on cases provided. This hybrid technique may give higher accuracy if it involves large number of data (cases), generation and populations even it will take a longer time to finish the training process. [6]

2.3 Detection of Knee Osteoarthritis Using X-Ray

We describe a method to detect osteoarthritis (OA) from knee X-ray images. The detection is based on the thickness of cartilage in knee bone, which correspond to possibility of osteoarthritis. Using our approach better diagnosis treatment can be applied to the patient since a computed automated measurements leads to accurate values so the image segmentation and mathematical morphological operation is applied to extract the border of cartilage by covering the boundary of cartilage. If cartilage thickness is lesser than 1.69 mm the patient has OA, otherwise not for this analysis we considered dataset of 100 X-ray image both with and without OA. Human body has many bone joints which play a major role in physical working. Among them knee joint is one of the most important joints of our body, which allows the leg to bend, straighten, rotate and to carry the weight of body. In the stage infant stage of conventional knee cap is no well-developed. But as the human growing age the conventional knee cap developed with required cartilage. Knee is a complicated joint structure of lower leg with two important joints, one joint between femur and tibia and another joint is between femur and patella

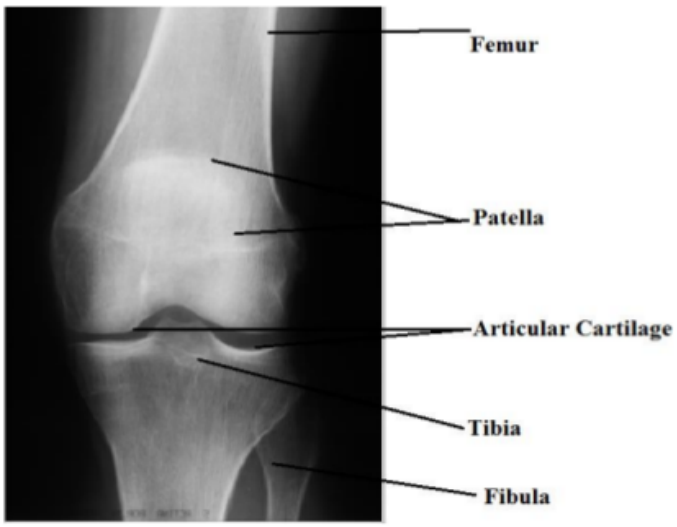


Figure 2.5: Anatomy of Knee

Osteoarthritis is a common disease in human under various joints such as knee, hips, hand and wrist. It is the stage of human bone in which the joints of human body become damaged and stop moving freely which causes pain. As the cartilage become thin and gap between the bones become narrows. The main causes of osteoarthritis are greater than age of 40 years, overweight, previous joint injury and by genetic hereditary. Knee osteoarthritis can make sitting and walking extremely painful. Knee osteoarthritis also varies between male

female genders as per the survey shown by Arthritis Research UK Primary care Centre, Keele University,some of those result are shown in table1.The survey shows that female has higher ratio to get osteoarthritis.

In clinical diagnosis, for a human body X-ray imaging is best tool for detecting abnormalities in bone painful, and deformed areas of bone. MRI(Magnetic Resonance Imaging) and CT(Computed Tomography) will give more details compared to X-ray, to determine the exact location of injured bone. X-ray is relatively less expensive compared to MRI and CT and the cartilage can be clearly seen. X-Rays used to examine broken bones and detect diseased cartilage. In recent years, due to the rapid development of computer technology, computer vision, image processing and pattern recognition, the related technologies have become increasingly important in medical image analysis. To solve our problem, image processing tool is one of the best tool for the detecting the disease .We proposed an automated system for osteoarthritis assessment, which is applied on X-ray images for identifying the characteristics of osteoarthritis. Image segmentation and edge detection method are applied to determine the thickness of the joint space.KL grades 0(normal), 1(doubtful) , 2(minimal) and 3(moderate).The KL method is is possible only when we have many X-ray image of knee of a patient with osteoarthritis and after. Unfortunately, this KL grade is not effective in the early period since X-rays show joint space narrowing. KL grade does not specify criteria for OA using cartilage thickness. In 1995 J Christopher Buckland-Wright et al. proposed a method of measurement of cartilage thickness using JSW (Joint Space Width) technique .This method used to approach such as plain film macro radiography and double contrast macroarthography for X-ray image generation then finally they compared with the sum of the tibial and femoral cartilage.For this analysis they used a hectic process is being done for osteoarthritis detection. Philipp Peloschek et al. given a RAQuantify software, this software use joint measurement in four steps and joint be selected at manually . This method that is measurement is based on web based technique ,for this they only measured the thickness with the RAQuantify and how to detection of knee osteoarthritis and how the identified this disease . Tati et al.in 2005 proposed a technique to automatically determine the region of interest needed for knee osteoarthritis assessment to horizontal and vertical translation to place the axis and locate the joint. Boundary detection between femur and joint space, for some image boundary appears discontinuously. Lior et al. in 2008 developed an automated approach for the detection of OA according to Kellegren-Lawrence method which show the probability of OA in different stage i.e. KL grade 0,1,2,3 .Their work on 20 preselected image ,each image is a 150*150 window of center of joint, then these image downgraded by factor of 10 into 15*15 images. Schmidt et al. proposed a semi-automated method of JSW in knee X-ray that identifies the femoral and tibial edges by first adjusting the image intensity. Then use of canny edge detection algorithm determine the distal edge of the medial and lateral femoral condyles. Define the extent of the medial and lateral compartments by the user. The

inner boundary was defined at that point where the slope of the tibial spine began to increase and the outer boundary was defined as the outer edge of the tibial plateau. The cortical bone interface of the tibia was found by determining the brightest pixels in each vertical scan line. This proposed method was replicated across the entire medial and lateral tibial compartment to detect femoral and tibial edges. [8]

2.4 An efficient Harris hawks-inspired image segmentation method

Image segmentation is an essential preprocessing step in computer vision, pattern recognition, and image processing in various fields of applications such as medical images Basically, image segmentation is the process of dividing an image into several non-overlapping regions or structures of interest based on grayscale, color, texture, shape, size, or position of image.Segmentation is a crucial phase in image processing because it simplifies the representation of an image and facilitates its analysis.The multilevel thresholding method is more efficient for segmenting digital mammograms compared to the classic bi-level thresholding since it uses a higher number of intensities to represent different regions in the image[9].

In this paper, an efficient methodology for multilevel segmentation is proposed using the Harris Hawks Optimization (HHO) algorithm and the minimum cross- entropy as a fitness function One of the most widespread techniques for image segmentation is the histogram thresholding, which is widely used due to its simplicity, high accuracy, and robustness against the other methods. It extracts the data of the histogram from an image and defines the optimal threshold values (th) to classify the pixels in different regions. Image thresholding techniques can be classified into two different types: bilevel and multilevel. If the objects of interest are clearly distinguished from the background of an image by a single threshold value, it is termed as bilevel thresholding, while dividing an image into sev- eral different regions by multiple threshold values is known as a multilevel threshold (MTH). Regarding the implementations, MTH is more accurate than the classic bi-level threshold method for the segmentation of digital mammograms due to its diverse number of intensities used to represent regions in the image.

This article introduces the HHO algorithm as an alternative method for multilevel image thresholding. The HHO algorithm is a swarm-based method to handle continuous optimization tasks. This method develops new exploratory and exploitative trends in the field based on the simulation of hawks and rabbits. The results initially presented reveal that the HHO is among the efficient methods proposed recently, and the solutions are of high quality. The HHO is selected due it has not been tested over real problems like image segmentation. Moreover, its use permits to handle with the drawback of MTH avoiding to fail in sub-optimal values. MTH then is a multidimensional problem that increases its complexity with the amount of th values.

Considering the above, the aim is to present an alternative efficient multilevel thresholding approach based on the HHO for digital images. Since HHO is new, it has not been extensively used or tested in many implementations. Finally, to verify the flexibility of the proposed MTH method based on HHO, it has been used to segment digital mammograms.The motivation of this paper then is related to two directions. The first is to test the HHO in an image processing application. Meanwhile, the second direction is to use the segmentation proposal on a real problem related to medical images. Considering such motivations, we develop an efficient tool that can be used as a preprocessing step in different image processing systems. The proposed approach is called MCET-HHO, where the HHO algorithm is used to find the best configuration of thresholds that segment an image by considering the Minimum Cross Entropy Thresholding (MCET) as a fitness function. In the HHO, each solution is a set of thresholds that is part of a population. Using the operators that mimics the behavior of Harris hawks, the algorithm evolves the solutions until finding the optimal.

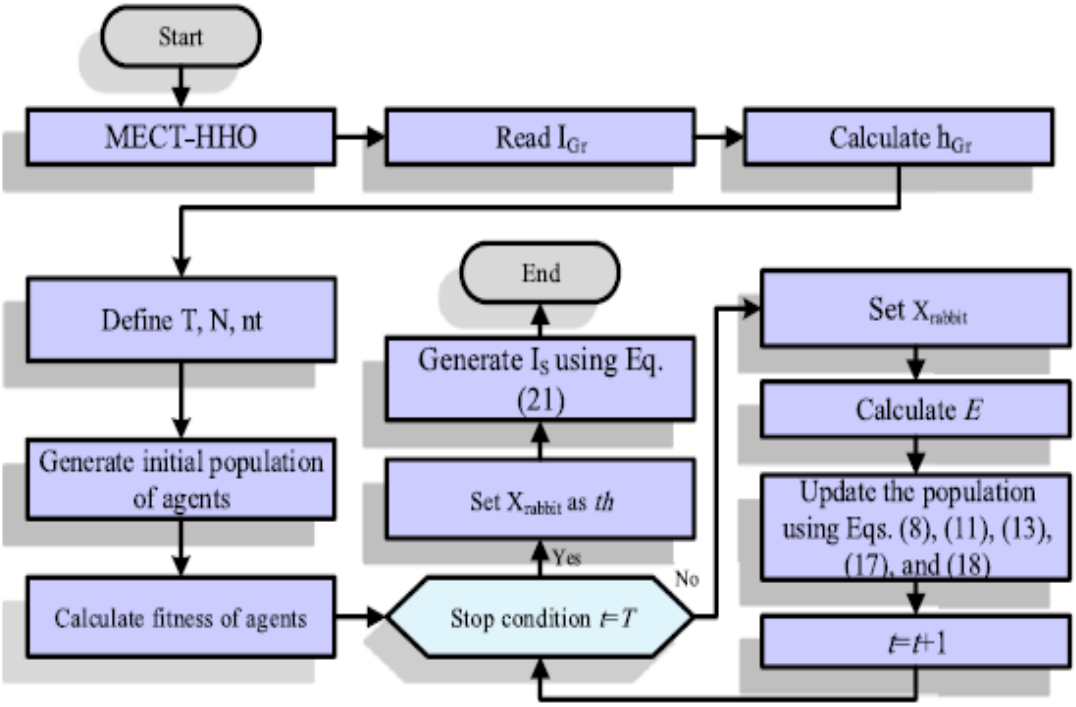


Figure 2.6: The flowchart of MCET-HHO.

The main contributions of this paper are summarized as follows:

- Substantiating the HHO in dealing with a real multidimensional application related to image processing.
- Proposing an efficient method for multilevel thresholding by using the minimum cross-

entropy.

- Applying the proposed technique to the segmentation of digital mammograms; this approach is utilized in CADx systems.

This work proposes the application of a new approach based on the HHO metaheuristic algorithm for multilevel segmentation called MCET-HHO. The proposed method optimizes the search for the best solution of a function inspired by the behavior of the Harris hawks, based on the technique of the minimization of cross- entropy. For the development of this proposal, a comparison of the multilevel segmentation of three benchmarks is performed, one that presents images with different characteristics in the intensities of their gray levels, another in the Berkeley segmentation database, and finally in Medical images of digital mammograms. In order to validate the behavior of the MCET-HHO algorithm, the segmentation of the images is carried out with four different levels, obtaining a series of metrics that allow measuring the quality of the segmentation for each of the levels. In addition, the multi-level segmentation of these images is also carried out with a series of different algorithms, allowing the results obtained in this work to be compared with those currently found in the literature.

Finally, the MCET-HHO algorithm shows an improvement in multilevel segmentation applied to images compared to the different algorithms reported in the literature included in this work, presenting a new approach that can be used for different applications as a complement to the different techniques that are currently used. The proposed approach is adequate to improve digital mammograms, as evidenced by the results. Since medical imaging is crucial for the diagnosis of many diseases

2.5 Deep Learning Model for Identifying Snakes by using Snakes' Bite Marks

Identifying snakes by using their bite marks may help the doctors to diagnose the victim with proper anti venoms for saving patients[3]. It is very important step for doctors to help the patients who suffered by snakes bites. Hence here a study was done on processing images to classify them as different family of snakes using CNN (Convolution Neural Network) model in Deep Learning techniques. The CNN model needs different snakes and their bite marks images to classify them as venomous and non-venomous snakes and by processing venomous snakes bite marks images it can able to find the venomous snakes family. To give accurate results the proposed Deep learning model has to be trained periodically with all possible different images of same snake's family and different snakes' families. The performance of the CNN model is

on its knowledge and finding patterns on the input images to find the family of the snakes. If the input images are huge in numbers and size then the system may take time to give results. That has to be considered to give results in less duration execution time.

The main problem in snake's bites is identifying the family of the snake which attacked the prey[6]. If survivors know the family of the snakes then easily doctors will take perfect antivenom and clinical procedures to cure the patient. But if the family of the snake is unknown then it will be a problem to doctors to give treatment to the survivor. Research works are going on in this area to find the snake is attacked the patient and what is the anti-venom have to be injected to the patient. Here, a Deep Learning based approach is proposed to find the family of the snake by using 'Convolution Neural Network (CNN)' model.

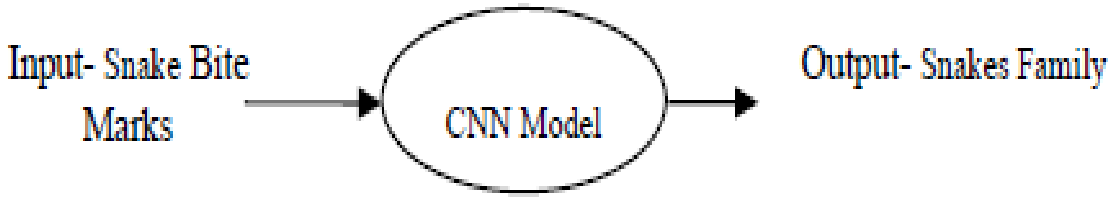


Figure 2.7: Context Flow Model of CNN on Snake Identification

The input to the CNN model is snake bite marks photos or images. The output of the model will be the family of the snake.This kind of bite marks pictures or data set are essential to train the CNN model to detect the family of the supplied snakes' bites photos.

The idea proposed here is based on 'Deep Learning' model to train the system periodically for processing the new input image to identify the kind of snake family to initiate proper treatment to the victim. The CNN model is specifically for processing images to find objects present in the input images. This model can help to find the appropriate snakes bites patterns when the system is trained already by providing sufficient set of input images about the snakes bite marks.

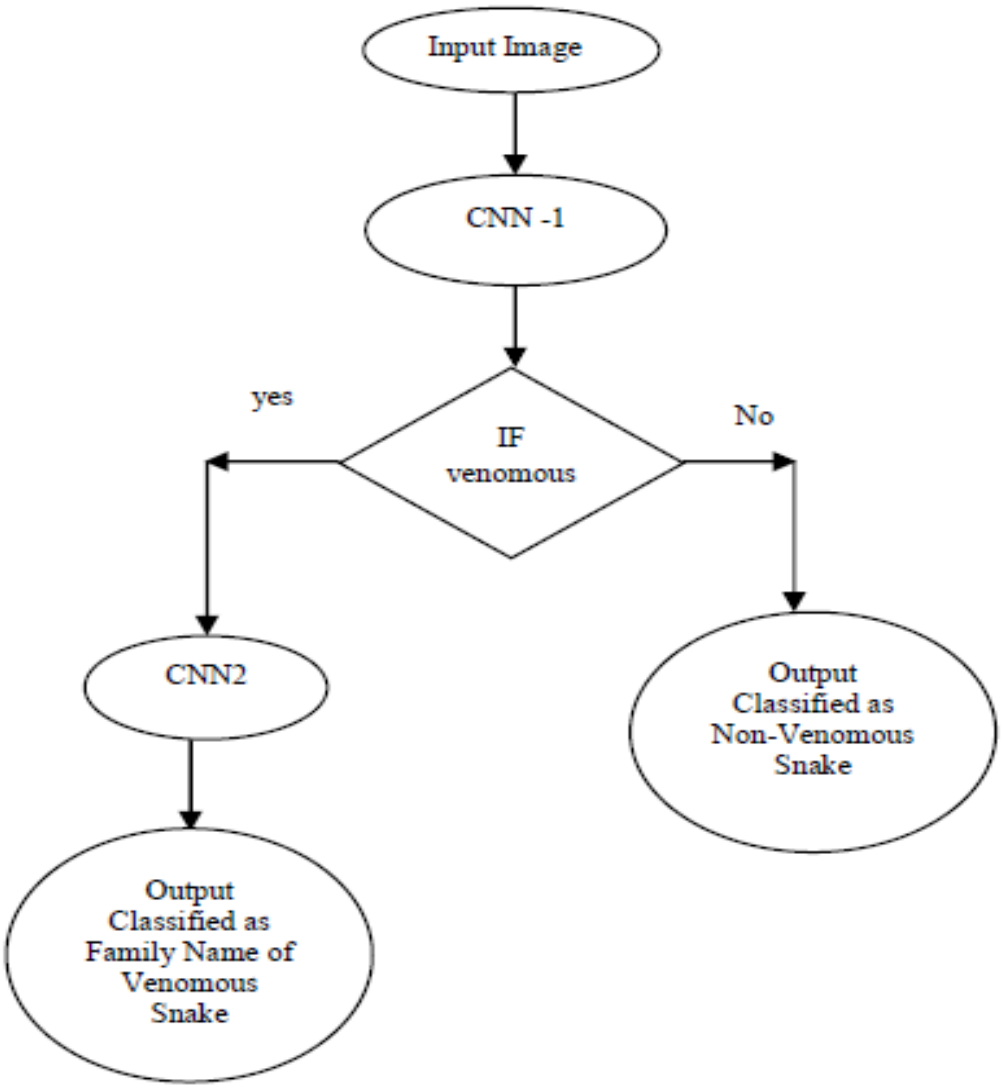


Figure 2.8: Actions flow the Proposed Model

CHAPTER 3

PROBLEM STATEMENT

The project "Snakebite Detection & Identification With Snakebite Mark Using Machine Learning Approach" aims at providing an efficient solution in the field of snakebite envenomation. In contrast to the previous systems, our motivation is the identification and recognition of distinct snake bites at the earliest, resulting in faster antivenom administration which in turn, narrows the mortality rate due to the envenomation by utilizing a more efficient and accurate machine learning approach. Classification is done by the convolutional neural network. One of the main advantages is the ability to identify the snake and provide faster medical aid Hence substantially reducing the death rate. This property distinguishes this methodology from the rest since this property helps the training model to extract all those relevant features which can lead to more accurate and precise classification. Thus, the image processing helps to get snakebite identification that can overcome the limitations of the existing system is implemented.

CHAPTER 4
PROJECT MANAGEMENT

4.1 Introduction

Project management is the discipline of planning, organizing, securing, managing, leading, and controlling resources to achieve specific goals. A project is a temporary endeavor with a defined beginning and end (usually time-constrained, and often constrained by funding or deliverables), undertaken to meet unique goals and objectives, typically to bring about beneficial change or added value. The temporary nature of projects stands in contrast with business as usual (or operations), which are repetitive, permanent, or semi-permanent functional activities to produce products or services. In practice, the management of these two systems is often quite different, and as such requires the development of distinct technical skills and management strategies.

In our project we are following the typical development phases of an engineering project

1. Initiation
2. Planning and Design
3. Execution and Construction
4. Monitoring and Controlling Systems
5. Completion

4.1.1 Initiation

The initiating processes determine the nature and scope of the project. The initiating stage should include a plan that encompasses the following areas :

1. Analysing the business needs/requirements in measurable goals
2. Reviewing of the current operations
3. Financial analysis of the costs and benefits including a budget
4. Stakeholder analysis, including users, and support personal for the project

5. Project charter including costs, tasks, deliverables, and schedule

4.1.2 Planing and design

After the initiation stage, the project is planned to an appropriate level of detail (see example of a flow-chart). The main purpose is to plan time, cost and resources adequately to estimate the work needed and to effectively manage risk during project execution. As with the initiation process, a failure to adequately plan greatly reduces the project's chances of successfully accomplishing its goals.

- Determining how to plan
- Developing the scope statement
- Selecting the planning team
- Identifying deliverables and creating the work breakdown structure
- Identifying the activities needed to complete those deliverables
- Developing the schedule
- Risk planning

4.1.3 Execution

Executing consists of the processes used to complete the work defined in the project plan to accomplish the project's requirements. The execution process involves coordinating people and resources, as well as integrating and performing the activities of the project in accordance with the project management plan. The deliverables are produced as outputs from the processes performed as defined in the project management plan and other frameworks that might be applicable to the type of project at hand.

4.1.4 Monitoring & controlling

Monitoring and controlling consists of those processes performed to observe project execution so that potential problems can be identified in a timely manner and corrective action can be taken, when necessary, to control the execution of the project. The key benefit is that project performance is observed and measured regularly to identify variances from the project management plan.

4.2 System Development Life Cycle

The Systems development life cycle (SDLC), or Software development process in systems engineering, information systems, and software engineering, is a process of creating or altering information systems, and the models and methodologies that people use to develop these systems. In software engineering, the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system.

The SDLC phases serve as a programmatic guide to project activity and provide a flexible but consistent way to conduct projects to a depth matching the scope of the project. Each of the SDLC phase objectives is described in this section with key deliverables, a description of recommended tasks, and a summary of related control objectives for effective management. The project manager must establish and monitor control objectives during each SDLC phase while executing projects. Control objectives help to provide a clear statement of the desired result or purpose and should be used throughout the entire SDLC process.

4.2.1 Spiral Model

We have used the Spiral model in our project. The Spiral model incorporates the best characteristics of both- waterfall and prototyping model. In addition, the Spiral model also contains a new component called Risk Analysis, which is not there in the waterfall and prototype model. In the Spiral model, the basic structure of the software product is developed first. After the basic structure is developed, new features such as user interface and data administration are added to the existing software product. This functionality of the Spiral model is similar to a spiral where the circles of the spiral increase in diameter. Each circle represents a more complete version of the software product. The spiral is a risk-reduction oriented model that breaks a software project up into main projects, each addressing one or major risks. After major risks have been addressed the spiral model terminates as a waterfall model. Spiral iteration involves six steps:

1. Determine objectives, alternatives and constraints.
2. Identify and resolve risks.
3. Evaluate alternatives.
4. Develop the deliverables for the iteration and verify that they are correct.
5. Plan the next iteration.

6. Commit to an approach for the next iteration.

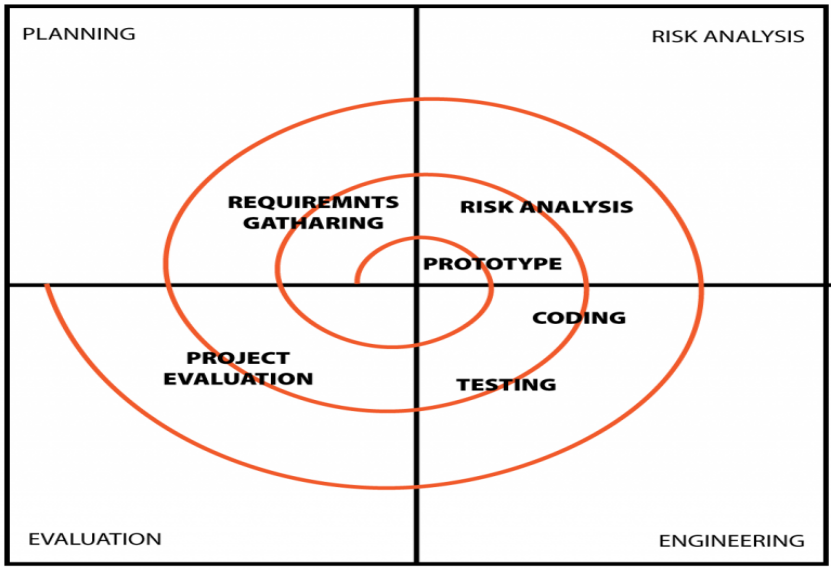


Figure 4.1: Spiral Model

CHAPTER 5
METHODOLOGY

5.1 System Requirements & Specifications

5.1.1 Google Colab

Google Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.To be precise, Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited. It supports multiple machine learning libraries.

5.1.2 TensorFlow

TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. TensorFlow was originally developed for large numerical computations without keeping deep learning in mind. However, it proved to be very useful for deep learning development as well, and therefore Google open-sourced it. TensorFlow works on the basis of data flow graphs that have nodes and edges. As the execution mechanism is in the form of graphs, it is much easier to execute TensorFlow code in a distributed manner across a cluster of computers while using GPUs.Graphical Processing Units (GPUs) are popular in the context of games, where you need the screen and image to be of high resolution. GPUs were originally designed for this purpose. However, they are being used for developing deep learning applications as well.

One of the major advantages of TensorFlow is that it supports GPUs, as well as CPUs. It also has a faster compilation time than other deep learning libraries, like Keras and Torch.

5.1.3 Windows 10

Windows 10 is a series of personal computer operating systems produced by Microsoft as part of its Windows NT family of operating systems. It is the successor to Windows 8.1 and was released to manufacturing on July 15, 2015, and to retail on July 29, 2015. Windows 10 receives new builds on an ongoing basis, which are available at no additional cost to users. Mainstream builds of Windows 10 are labeled version YYMM with YY representing the year and MM representing the month of release. For example, the latest mainstream build of Windows 10 is Version 1809. There are additional test builds of Windows 10 available to Windows Insiders. Devices in enterprise environments can receive these updates at a slower pace, or use long-term support milestones that only receive critical updates, such as security patches, over their ten-year lifespan of extended support.

5.1.4 Python 3.8.5

Python is a dynamic object-oriented programming language that can be used for many kinds of software development. It offers strong support for integration with other languages and tools, comes with extensive standard libraries, and can be learned in a few days. Many Python programmers report substantial productivity gains and feel the language encourages the development of higher quality, more maintainable code.

TensorFlow supports Python 3.5, 3.6 and 3.7 on Windows 10. Although TensorFlow 2.1 will be the final version of TensorFlow that will support Python 2 (regardless of OS).Python 3.8 support requires TensorFlow 2.2 or later.Python 3.9 support requires TensorFlow 2.5 or later.

Python runs on Windows, Linux/Unix, Mac OS X, OS/2, Amiga, Palm Handhelds, and Nokia mobile phones. Python has also been ported to the Java and .NET virtual machines. Python is distributed under an OSI-approved open source license that makes it free to use, even for commercial products.

5.1.5 OpenCV

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. As said earlier we are utilizing this library for our image processing. The library has more than 2500 optimized algorithms.These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce

3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

5.1.6 Jupyter Environment

JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones.

5.2 Proposed System

Modules

In this system, we are not just retrieving the bite image description as venomous or non venomous[7]. Whereas, we are also considering the further species recognition in here. Based on this we can divide this system into two modules which are :

- Data Acquisition and Pre-processing Module
- Detection and Identification Module

5.2.1 Data Acquisition and Pre-processing Module

As said earlier we have two modules which will do two different types procedures so that we get an accurate output. In the first module the input which are the bite images are collected. Data acquisition part being the most basic part where the data which is the snake bite mark has to be collected. The images of maximum quality available has to be collected to give the maximum accuracy to the system.The images which will be of different scales and pixels which will be inconvenient for the proceedings of the system. So, the data enters the pre-processing stage.

As the part of Image preprocessing, data will undergo image denoising for RGB images, which is to remove the noises and unwanted parts in the data collected which will give a clear focus on the bite mark. After that, the bite images are converted into a gray scale images to help in the segmentation process. Then the process is followed by Adaptive Gaussian Thresholding to generate the image. After the pre-processing stage is completed, the next step is the feature extraction.

In this stage the converted images are considered for all the uses which is to compare the image with the images in the database so that we get a close resembling outcome from the newly formed gray scale image[2].

5.2.2 Detection and Identification Module

In the second module, the obtained result from the previous module will be taken as the current input. The input for this module will be the snakebite image which has already undergone the segmentation process. This image will be focused on the snake bite mark.

Next, the images will be undergoing feature extraction which helps in the next step of

classification of the data. After the classification of images, they are taken into the training and testing parts which will help the system to give out the most accurate results.

The program applies Transfer Learning using Inception v3 to train it to classify a new set of images. Transfer learning is a machine learning method that utilizes a pre-trained neural network. Inception v3 architecture model trained on ImageNet images, and train a new top layer that can recognize other classes of images[10]. The image recognition model called Inception-v3 consists of two parts:

- Feature extraction part with a convolutional neural network
- Classification of the images from the dataset

The label for each image is taken from the name of the subfolder it's in. It is the third edition of the Inception CNN model by Google, originally instigated during the ImageNet Recognition Challenge.

NEURAL NETWORK IDENTIFIED

Neural network identified for classification is Convolutional Neural Network (CNN).

CNN

Convolutional Neural Networks or covnets are neural networks that share their parameters. Imagine an image which can be represented as a cuboid having its length, width (dimension of the image) and height (as image generally have red, green, and blue channels)[1].

A covnet is a sequence of layers, and every layer transforms one volume to another through differentiable function.

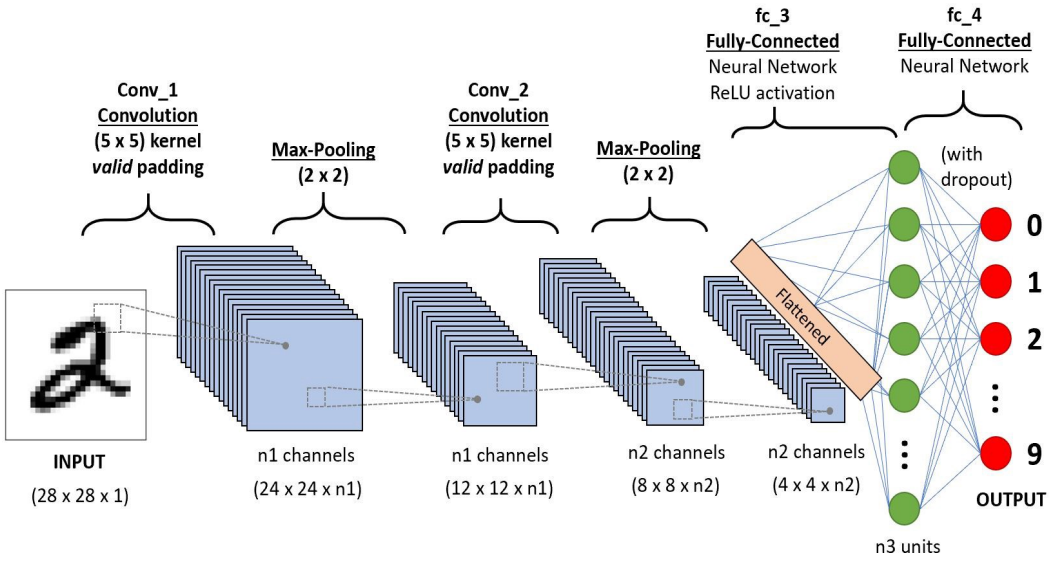


Figure 5.1: Convolutional Neural Network

Types of layers in CNN:

Consider a convolutional neural network on an image of dimension 32 x 32 x 3.

1. **Input Layer**: This layer holds the raw input of image with width 32, height 32 and depth 3.

2. **Convolution Layer**: This layer computes the output volume by computing dot product between all filters and image patch. Suppose we use total 12 filters for this layer we'll get output volume of dimension 32 x 32 x 12.

3. **Activation Function Layer**: This layer will apply element wise activation function to the output of convolution layer. Some common activation functions are RELU: max(0,

x), Sigmoid: 1/(1+e^-x), Tanh, etc.

4. **Pool Layer**: This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents from overfitting. Two common types of pooling layers are max pooling and average pooling. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

5. **Fully-Connected Layer**: This layer is regular neural network layer which takes input from the previous layer and computes the class scores and outputs the 1-D array of size equal to the number of classes. [4]

5.3 Data Flow Diagrams

5.3.1 Data Flow Diagram- Level 0

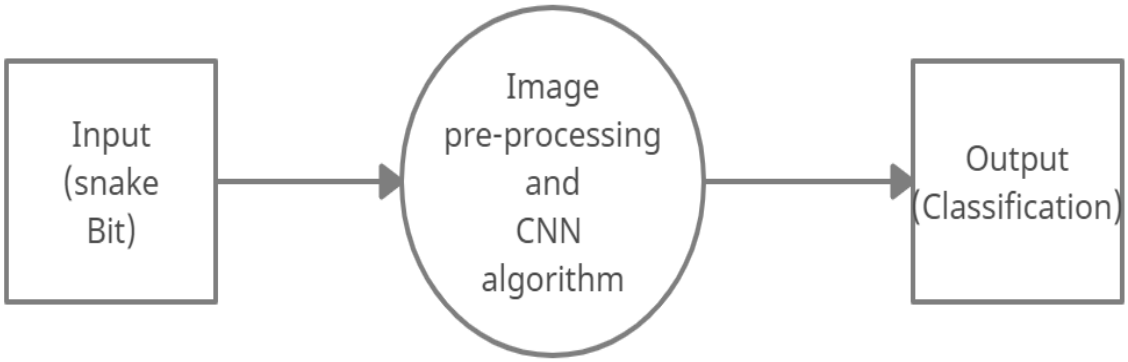


Figure 5.2: DFD- Level 0

5.3.2 Data Flow Diagram- Level 1

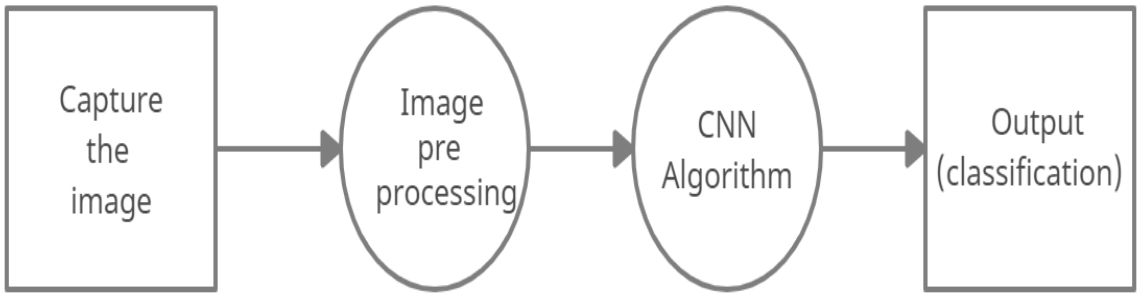


Figure 5.3: DFD Level-1

5.3.3 Data Flow Diagram- Level 2

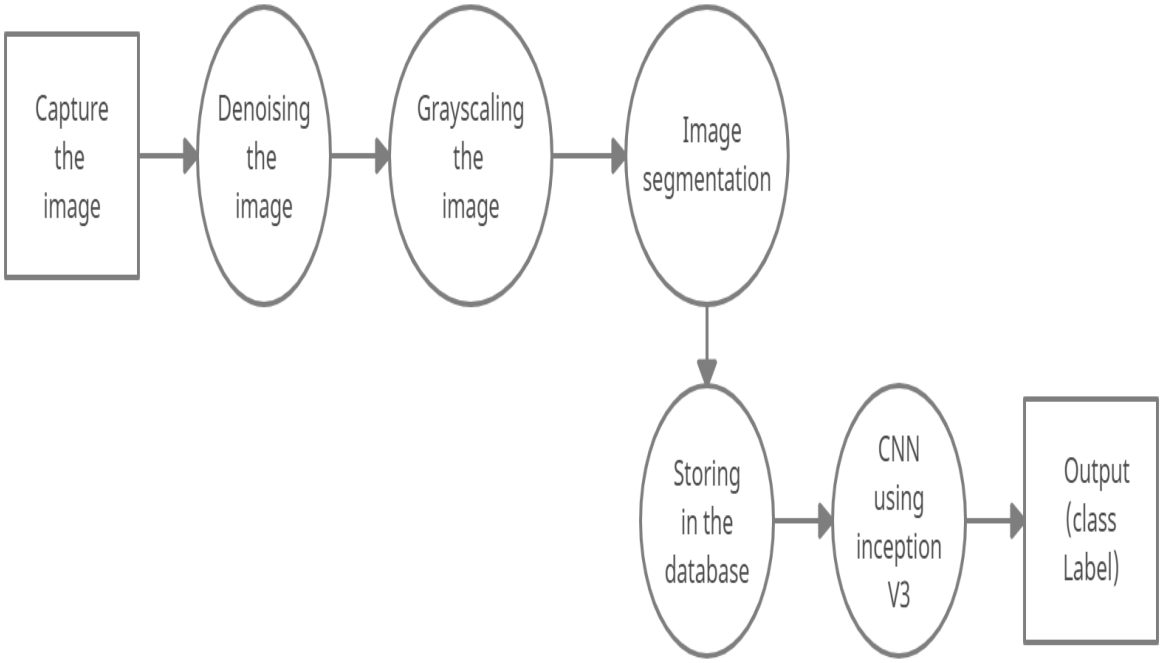


Figure 5.4: DFD- Level 2

5.4 Architectural Diagram

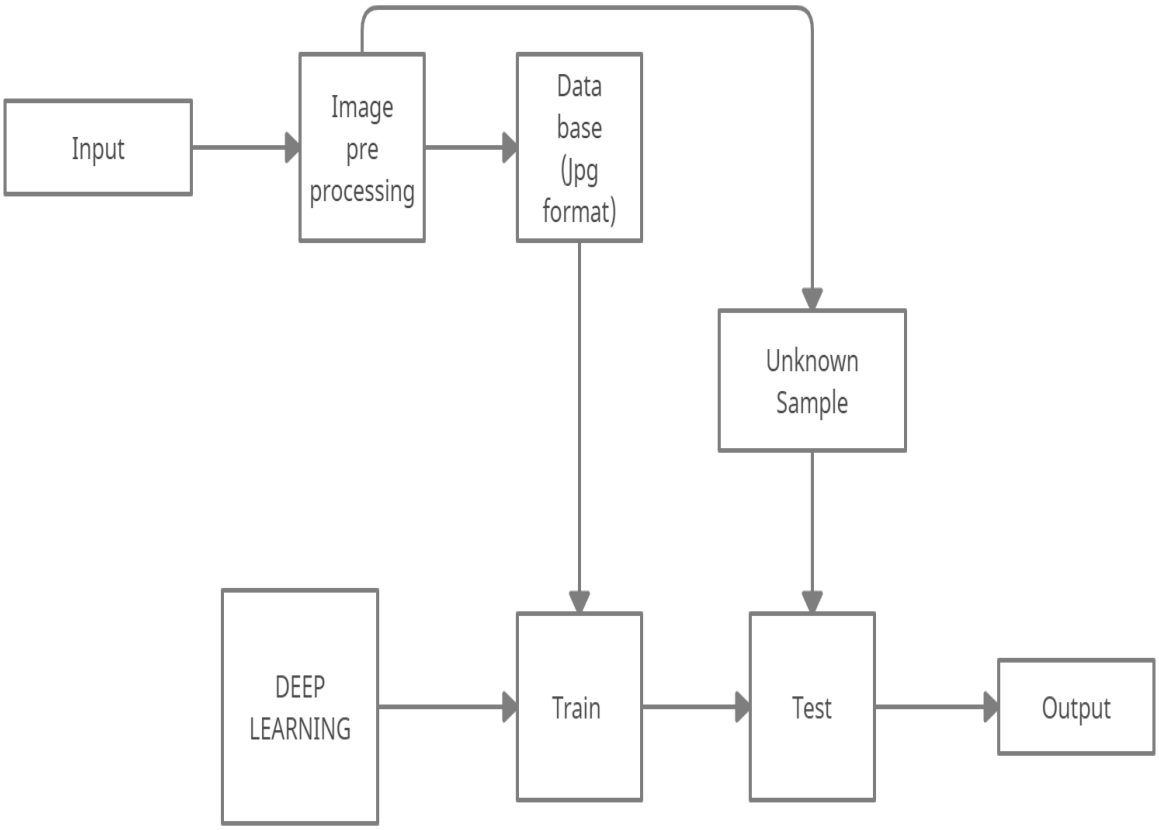


Figure 5.5: Architectural Diagram

5.5 Implementation

5.5.1 Start with Google Colab

Colab is ideal for everything from improving your Python coding skills to working with deep learning libraries, like PyTorch, Keras, TensorFlow, and OpenCV. One can create notebooks in Colab, upload notebooks, store notebooks, share notebooks, mount your Google Drive. One can import most of their favorite directories, upload personal Jupyter Notebooks, upload notebooks directly from GitHub, upload Kaggle files, download your notebooks, and do just about everything else

5.5.2 Import Libraries

Used libraries are listed below:

- os
- numpy
- cv2
- evaljs
- np utils
- resize
- matplotlib
- tensorflow

5.5.3 Data Pre-processing

Here the data that gets into the system can be of two types i.e, if the image is to be given as an input from the system we can just give it as :

```
from google.colab import files
uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.
        format( name=fn, length=len(uploaded[fn])))
```

The second way is to capture a live image of snake bite, which is done utilizing the webcamera.

```
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode

def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
   async function takePhoto(quality) {
```

```
    const div = document.createElement('div');
    const capture = document.createElement('button');
    capture.textContent = 'Capture';
    div.appendChild(capture);

    const video = document.createElement('video');
    video.style.display = 'block';
    const stream = await navigator.mediaDevices.getUserMedia
    ({video: true});

    document.body.appendChild(div);
    div.appendChild(video);
    video.srcObject = stream;
    await video.play();

// Resize the output to fit the video element.
    google.colab.output.setIframeHeight
    (document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)
  data = eval_js('takePhoto({})'.format(quality))
  binary = b64decode(data.split(',')[1])
  with open(filename, 'wb') as f:
    f.write(binary)
  return filename
```

Here we are utilizing the javascript and also this code alone won't work so we have to allow webcamera first then we will behaving another snippet of code so as to give a image capturing part.

```
from IPython.display import Image
try:
  filename = take_photo()
  print('Saved to {}'.format(filename))

  # Show the image which was just taken.
  display(Image(filename))
except Exception as err:
  # Errors will be thrown if the user does not have a webcam
or if they do not
  # grant the page permission to access it.
  print(str(err))
```

Most feasible way for the data is entering the data from the system for the training part. The image capturing method is for the user to enter the snake bite image to the system. After choosing the data entry we goes to the next part of the system that is the data preprocessing part. Here we are doing the needful so that the image quality can be the utmost.

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
import cv2 as cv
img = cv2.imread('#location')

#denoising from colored
dst = cv2.fastNlMeansDenoisingColored(img,None,10,10,7,21)
cv2.imwrite('denoisejust.png',dst)

#convert grayscale
gray_imgd = cv2.cvtColor(dst, cv2.COLOR_BGR2GRAY)

#save to new file
cv2.imwrite('newimg.png', gray_imgd)

#segmentation
```

```
img = cv2.imread('newimg.png',0)
img = cv2.medianBlur(img,5)
th = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
            cv2.THRESH_BINARY,11,12)

plt.subplot(121),plt.imshow(th,'gray')
cv2.imwrite('segjust.jpg',th)
plt.show('seg.png')
```

The whole preprocessing was compressed into a single code. here we are using the function **cv2.fastNlMeansDenoisingColored()** which denoises the RGB images. This step helps in the removal of unwanted noises from the image. Next part is the gray scaling on the denoised image to help in the segmentation part. The segmentation happening in the system was harder to work around so the utilization of 4 types of segmentations happened and the most accurate output was chosen so we took the Adaptive Gaussian Thresholding here as the most useful one as it gave the most clear bite image after working on multiple other segmentations.

5.5.4 Detection and Identification

Import Data

```
import os
import numpy as np
from PIL import Image
from imageio import imread
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import tf_slim as slim
from tf_slim.nets import inception
import tf_slim as slim
import cv2
import matplotlib.pyplot as plt
```

Data Loading

Setting up all the initial variables with default file locations and respective values.

```
ckpt_path = "/kaggle/input/inception_v3.ckpt"
```

```
images_path = "/kaggle/input/animals/*"
img_width = 299
img_height = 299
batch_size = 16
batch_shape = [batch_size, img_height, img_width, 3]
num_classes = 1001
predict_output = []
class_names_path = "/kaggle/input/imagenet_class_names.txt"
with open(class_names_path) as f:
    class_names = f.readlines()
```

Create Inception v3 model

```
X = tf.placeholder(tf.float32, shape=batch_shape)

with slim.arg_scope(inception.inception_v3_arg_scope()):
    logits, end_points = inception.inception_v3(
        X, num_classes=num_classes, is_training=False
    )

predictions = end_points["Predictions"]
saver = tf.train.Saver(slim.get_model_variables())
```

Define function for loading images and resizing for sending to model for evaluation in RGB mode.

```
def load_images(input_dir):
    global batch_shape
    images = np.zeros(batch_shape)
    filenames = []
    idx = 0
    batch_size = batch_shape[0]
    files = tf.gfile.Glob(input_dir)[:20]
    files.sort()
    for filepath in files:
        with tf.gfile.Open(filepath, "rb") as f:
            imgRaw = np.array(Image.fromarray(imread(
            f, as_gray=False, pilmode="RGB")).resize((299, 299))).
```

```
            astype(np.float) / 255.0
        images[idx, :, :, :] = imgRaw * 2.0 - 1.0
        filenames.append(os.path.basename(filepath))
        idx += 1
        if idx == batch_size:
            yield filenames, images
            filenames = []
            images = np.zeros(batch_shape)
            idx = 0
    if idx > 0:
        yield filenames, images
```

Training the model

Loading Pre-Trained Model and classifying images using model

```
session_creator = tf.train.ChiefSessionCreator(
        scaffold=tf.train.Scaffold(saver=saver),
        checkpoint_filename_with_path=ckpt_path,
        master='')
```

Above code snippet is used for Loading pre-trained model

```
with tf.train.MonitoredSession(session_creator=session_creator)
as sess:
    for filenames, images in load_images(images_path):
        labels = sess.run(predictions, feed_dict={X: images})
        for filename, label, image in zip(filenames, labels,images):
            predict_output.append([filename, label, image])
```

This snippet is to classify images using model

Testing or Predictions

```
for x in predict_output:
    out_list = list(x[1])
    topPredict = sorted(range(len(out_list)), key=lambda i:
    out_list[i], reverse=True)[:5]
    plt.imshow((((x[2]+1)/2)*255).astype(int))
```

```
plt.show()
print("Filename:",x[0])
print("Displaying the top 5 Predictions for above image:")
for p in topPredict:
    print(class_names[p-1].strip())
```

CHAPTER 6
RESULTS

6.1 Preprocessing

In the first experiment, the effectiveness of the model in detecting snake using the bite mark are examined.

6.1.1 captured image 6.1.2 denoised image 6.1.3 grayscaling

The above given images shows the image at different preprocessing phases. The system goes to the next part which is segmentation. Adaptive Gaussian thresholding gives us an output like the one given below.

Figure 6.1: Adaptive Gaussian Thresholding

The preprocessing steps will give theses images as the output for each steps respectively and the final output will be the segmented image.

6.2 Identification

This section works on the feature extraction and classification of the images that have undergone the preprocessing steps. The feature extraction part is done with a convolutional neural network. The classification part is done using a fully-connected and softmax layers. After the digital image is processed and classified. It undergoes traning and testing phase. After training and testing system will give out an output of score for each image which will be a value. The highest valued snake is the one whose bite image is given as input.

Figure 6.2: Final prediction

The programming language used for implementing the model is Python.The main goal is to identify the snakebite from the bite image.Thus helping the person to get faster medical aid and altogether decrease the snakebite envenoming deaths to a certain length.

	Classification
Accuracy	79.652
Precision	0.7961
Recall	0.825
F1 Score	0.80

Figure 6.3: Classification Results

The deep learning method is used for getting the most accurate result.

CHAPTER 7
CONCLUSION AND FUTURE WORKS

7.1 Conclusion

Deep learning techniques have been widely used in many applications such as healthcare, big data analysis, etc. We are using these deep learning techniques to detect and identify snake bite images, thus overcoming the limitations of the existing snakebite envenoming treatment approach. Our main motivation for designing such a system being, identification and recognition of distinct snake bite at the earliest, following in faster antivenom administration, which in turns narrow the mortality rate due to envenomation It uses the data for two major classifications. The first one involves detection, which aims at detecting the presence of a bite mark. The second one aims at classifying and identifying the bite that was detected. The system is all done by the deep neural network and transfer learning using inception V3 to get the utmost accuracy that can be provided. The system we designed possesses the capacity to detect the snake bite mark and then identify the snake species giving out minimum errors in the score that will be displayed as the output

7.2 Future Scope

We can add some more features like symptom prediction and location sharing feature to reach the nearest hospitals which have the proper antivenom for the snakebite.

CHAPTER 8

SUMMARY OF THE RESULTS

	Classification
Accuracy	79.652
Precision	0.7961
Recall	0.825
F1 Score	0.80

Figure 8.1: Summarization Table

CHAPTER 9

SCREENSHOTS

9.1 Google Colab Code

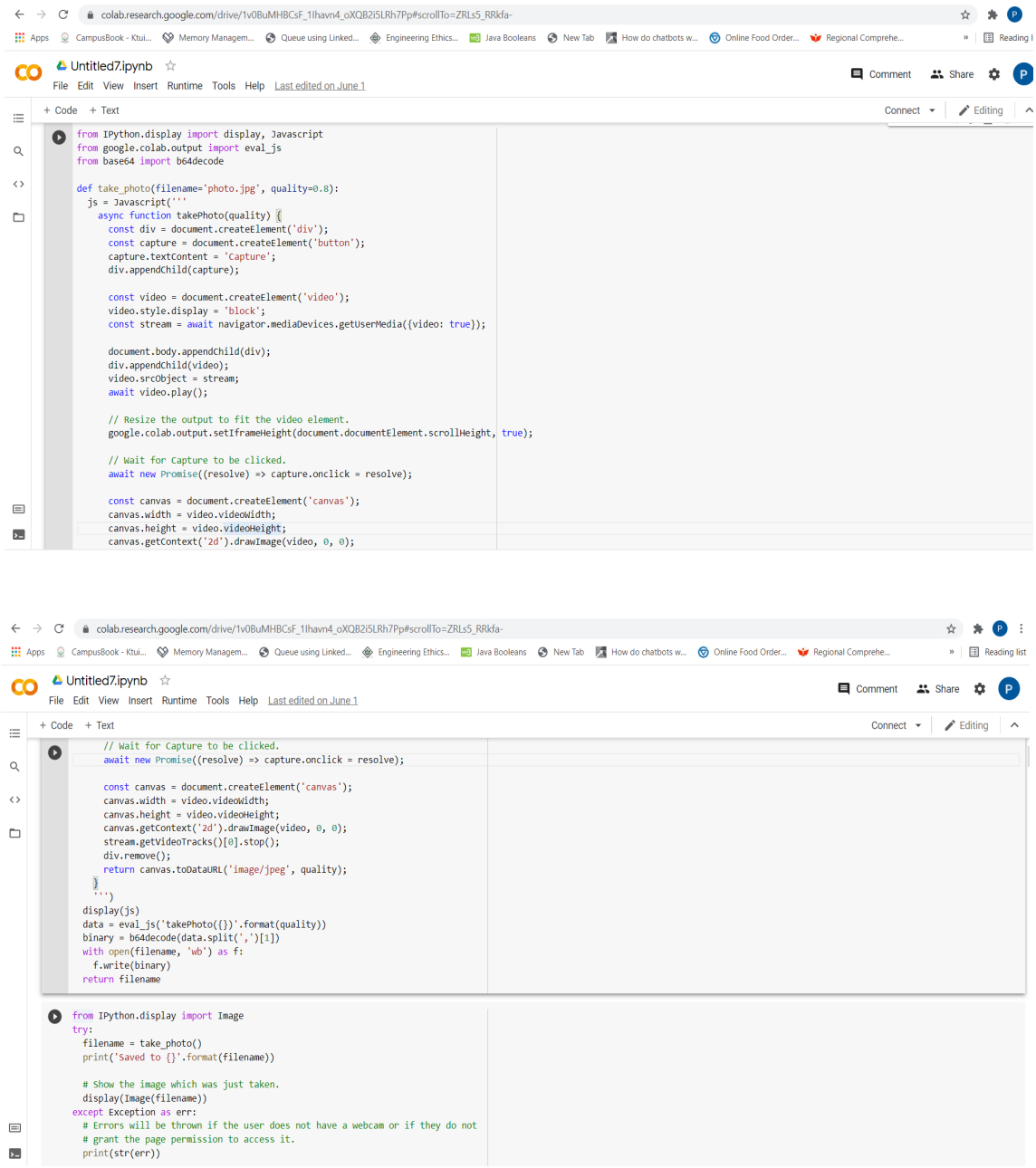
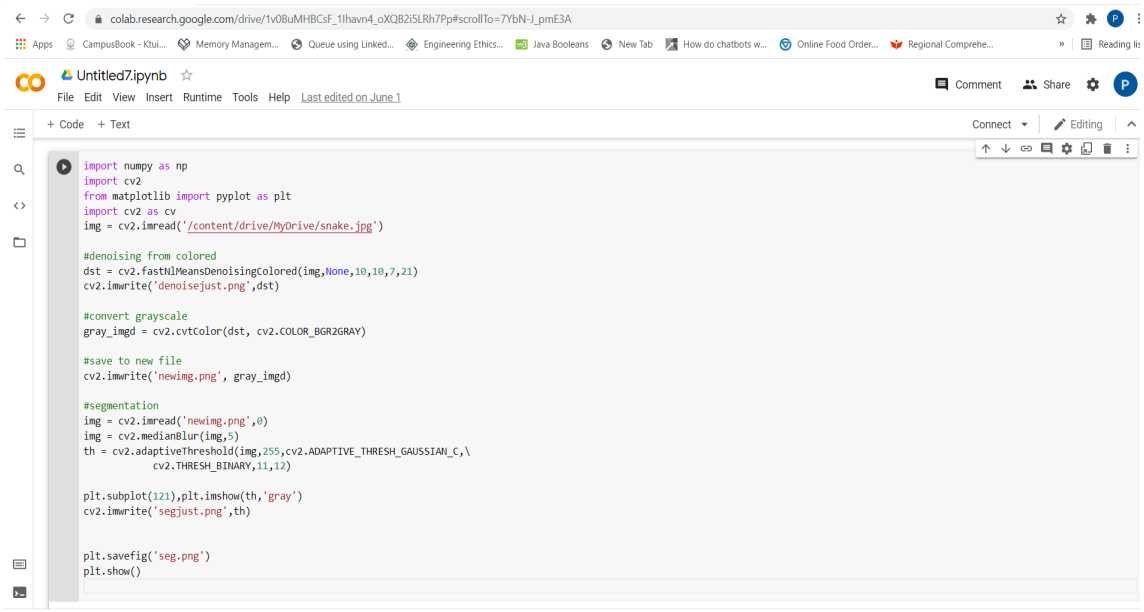
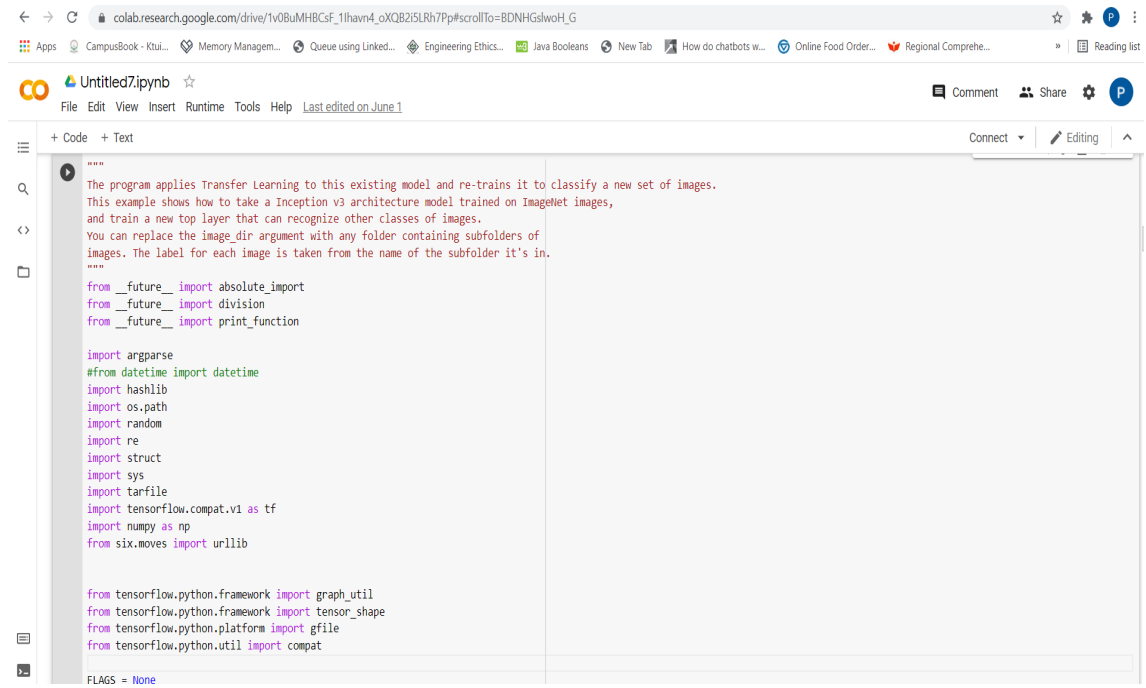


Figure 9.1: Image Capturing

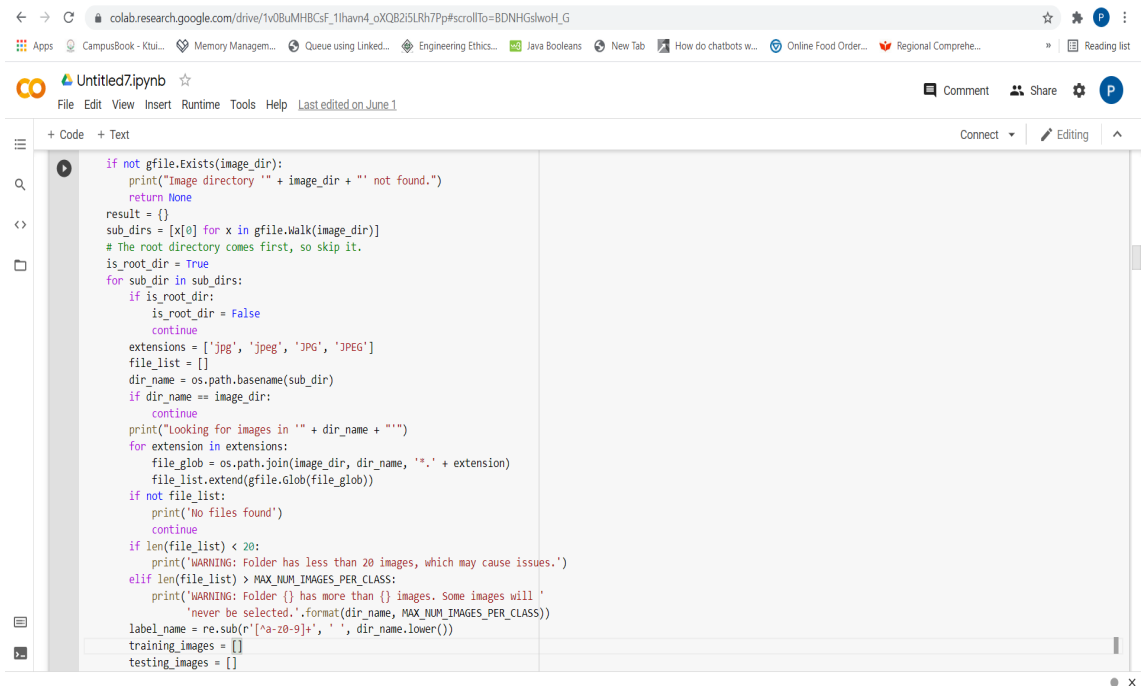


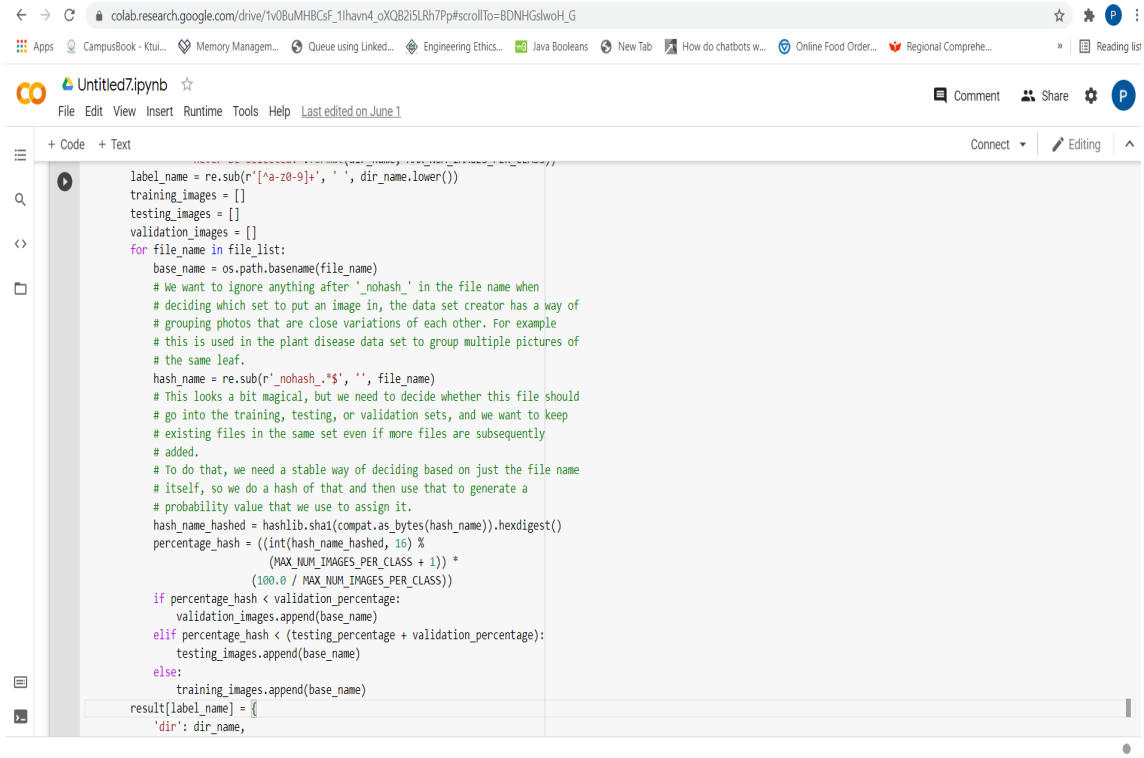
```
import numpy as np
import cv2
from matplotlib import pyplot as plt
import cv2 as cv
img = cv2.imread('/content/drive/MyDrive/snake.jpg')

#denoising from colored
dst = cv2.fastNlMeansDenoisingColored(img,None,10,10,7,21)
cv2.imwrite('denoisejust.png',dst)

#convert grayscale
gray_imgd = cv2.cvtColor(dst, cv2.COLOR_BGR2GRAY)

#save to new file
cv2.imwrite('newimg.png', gray_imgd)

#segmentation
img = cv2.imread('newimg.png',0)
img = cv2.medianBlur(img,5)
th = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
            cv2.THRESH_BINARY,11,12)

plt.subplot(121),plt.imshow(th,'gray')
cv2.imwrite('segjust.png',th)


plt.savefig('seg.png')
plt.show()
```

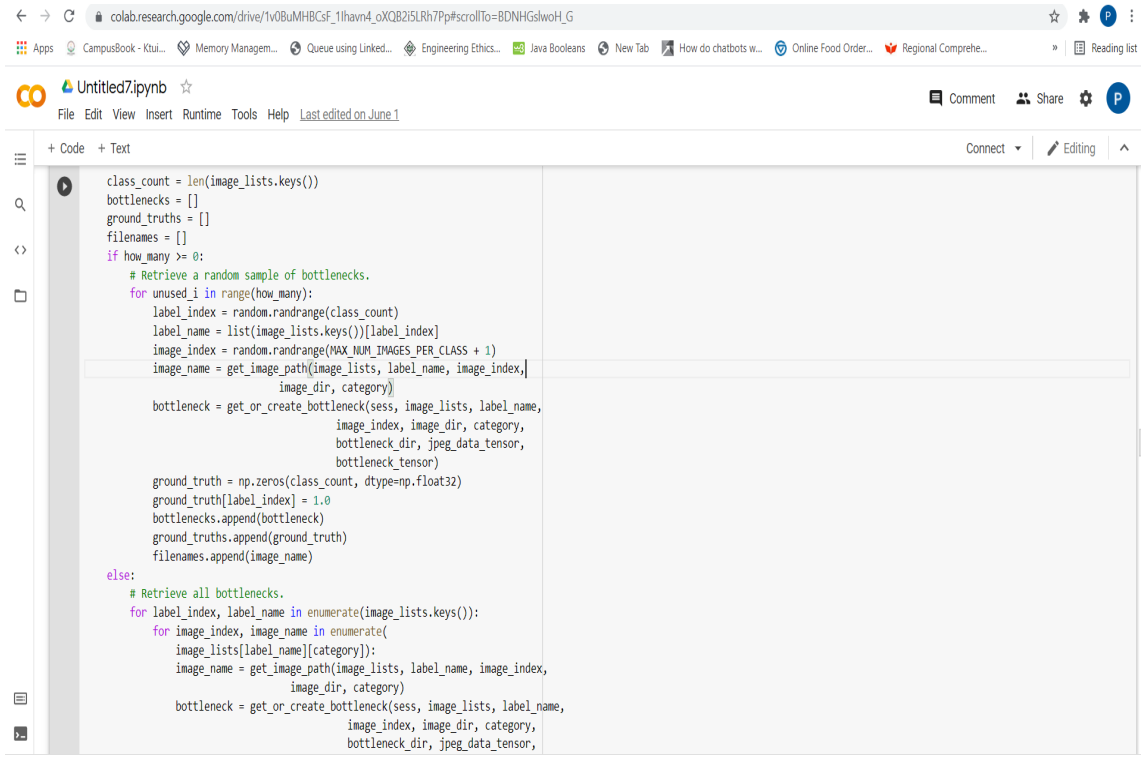
Figure 9.2: Image Preprocessing



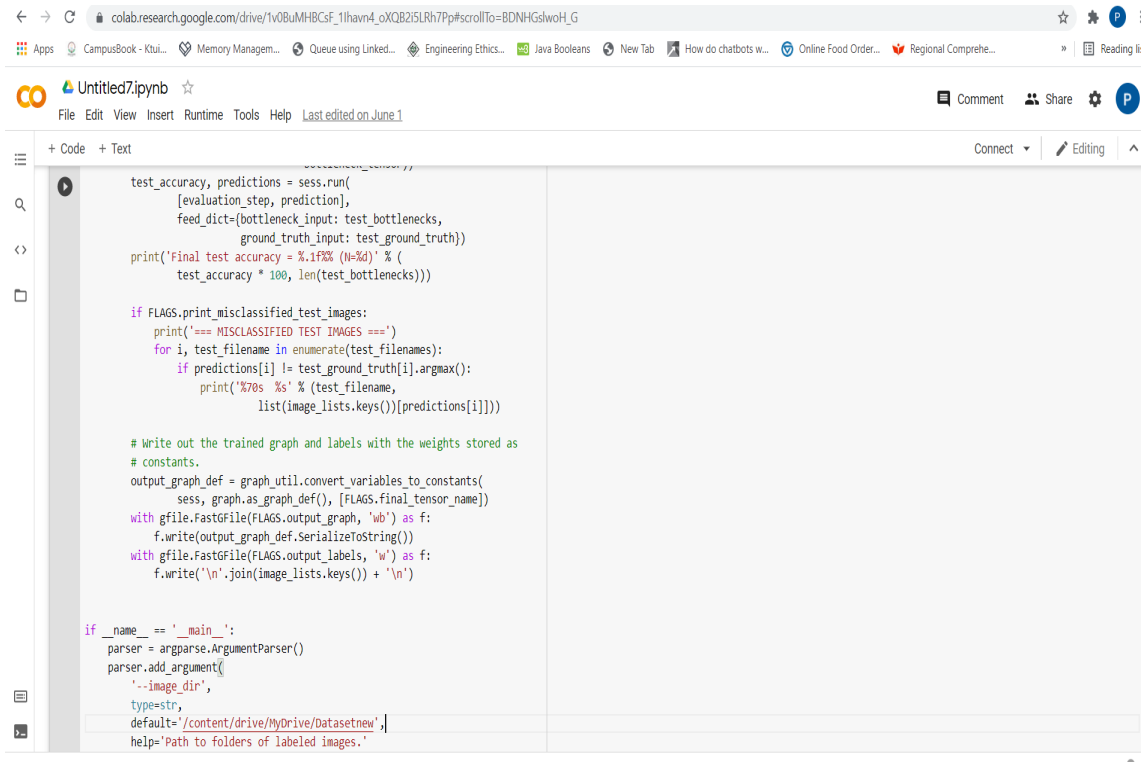
```
"""
The program applies Transfer Learning to this existing model and re-trains it to classify a new set of images.
This example shows how to take a Inception v3 architecture model trained on ImageNet images,
and train a new top layer that can recognize other classes of images.
You can replace the image_dir argument with any folder containing subfolders of
images. The label for each image is taken from the name of the subfolder it's in.
"""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import argparse
#from datetime import datetime
import hashlib
import os.path
import random
import re
import struct
import sys
import tarfile
import tensorflow.compat.v1 as tf
import numpy as np
from six.moves import urllib


from tensorflow.python.framework import graph_util
from tensorflow.python.framework import tensor_shape
from tensorflow.python.platform import gfile
from tensorflow.python.util import compat

FLAGS = None
```


colab.research.google.com/drive/1v0BuMHBCsF_1Ihavn4_oXQB2i5LRh7Pp#scrollTo=BDNHGslwoH_G
Untitled7.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 1
Comment
Share
+ Code + Text
Connect
Editing
if not gfile.Exists(image_dir):
print("Image directory '" + image_dir + "' not found.")
return None
result = {}
sub_dirs = [x[0] for x in gfile.Walk(image_dir)]
# The root directory comes first, so skip it.
is_root_dir = True
for sub_dir in sub_dirs:
if is_root_dir:
is_root_dir = False
continue
extensions = ['jpg', 'jpeg', 'JPG', 'JPEG']
file_list = []
dir_name = os.path.basename(sub_dir)
if dir_name == image_dir:
continue
print("Looking for images in '" + dir_name + "'")
for extension in extensions:
file_glob = os.path.join(image_dir, dir_name, '*.' + extension)
file_list.extend(gfile.Glob(file_glob))
if not file_list:
print('No files found')
continue
if len(file_list) < 20:
print('WARNING: Folder has less than 20 images, which may cause issues.')
elif len(file_list) > MAX_NUM_IMAGES_PER_CLASS:
print('WARNING: Folder {} has more than {} images. Some images will '
'never be selected.'.format(dir_name, MAX_NUM_IMAGES_PER_CLASS))
label_name = re.sub(r'[^a-z0-9]+', ' ', dir_name.lower())
training_images = []
testing_images = []


colab.research.google.com/drive/1v0BuMHBCsF_1Ihavn4_oXQB2i5LRh7Pp#scrollTo=BDNHGslwoH_G
Untitled7.ipynb
File Edit View Insert Runtime Tools Help Last edited on June 1
Comment
Share
+ Code + Text
Connect
Editing
label_name = re.sub(r'[^a-z0-9]+', ' ', dir_name.lower())
training_images = []
testing_images = []
validation_images = []
for file_name in file_list:
base_name = os.path.basename(file_name)
# We want to ignore anything after '_nohash_' in the file name when
# deciding which set to put an image in, the data set creator has a way of
# grouping photos that are close variations of each other. For example
# this is used in the plant disease data set to group multiple pictures of
# the same leaf.
hash_name = re.sub(r'_nohash_.*$', '', file_name)
# This looks a bit magical, but we need to decide whether this file should
# go into the training, testing, or validation sets, and we want to keep
# existing files in the same set even if more files are subsequently
# added.
# To do that, we need a stable way of deciding based on just the file name
# itself, so we do a hash of that and then use that to generate a
# probability value that we use to assign it.
hash_name_hashed = hashlib.sha1(compat.as_bytes(hash_name)).hexdigest()
percentage_hash = ((int(hash_name_hashed, 16) %
(MAX_NUM_IMAGES_PER_CLASS + 1)) *
(100.0 / MAX_NUM_IMAGES_PER_CLASS))
if percentage_hash < validation_percentage:
validation_images.append(base_name)
elif percentage_hash < (testing_percentage + validation_percentage):
testing_images.append(base_name)
else:
training_images.append(base_name)
result[label_name] = {
'dir': dir_name,



```
    class_count = len(image_lists.keys())
    bottlenecks = []
    ground_truths = []
    filenames = []
    if how_many >= 0:
        # Retrieve a random sample of bottlenecks.
        for unused_i in range(how_many):
            label_index = random.randrange(class_count)
            label_name = list(image_lists.keys())[label_index]
            image_index = random.randrange(MAX_NUM_IMAGES_PER_CLASS + 1)
            image_name = get_image_path(image_lists, label_name, image_index,
                                        image_dir, category)
            bottleneck = get_or_create_bottleneck(sess, image_lists, label_name,
                                                  image_index, image_dir, category,
                                                  bottleneck_dir, jpeg_data_tensor,
                                                  bottleneck_tensor)
            ground_truth = np.zeros(class_count, dtype=np.float32)
            ground_truth[label_index] = 1.0
            bottlenecks.append(bottleneck)
            ground_truths.append(ground_truth)
            filenames.append(image_name)
    else:
        # Retrieve all bottlenecks.
        for label_index, label_name in enumerate(image_lists.keys()):
            for image_index, image_name in enumerate(
                image_lists[label_name][category]):
                image_name = get_image_path(image_lists, label_name, image_index,
                                            image_dir, category)
                bottleneck = get_or_create_bottleneck(sess, image_lists, label_name,
                                                      image_index, image_dir, category,
                                                      bottleneck_dir, jpeg_data_tensor,
```



```
        test_accuracy, predictions = sess.run(
                [evaluation_step, prediction],
                feed_dict={bottleneck_input: test_bottlenecks,
                           ground_truth_input: test_ground_truth})
        print('Final test accuracy = %.1f%% (N=%d)' % (
                test_accuracy * 100, len(test_bottlenecks)))

        if FLAGS.print_misclassified_test_images:
            print('=== MISCLASSIFIED TEST IMAGES ===')
            for i, test_filename in enumerate(test_filenames):
                if predictions[i] != test_ground_truth[i].argmax():
                    print('%70s  %s' % (test_filename,
                                        list(image_lists.keys())[predictions[i]]))

        # Write out the trained graph and labels with the weights stored as
        # constants.
        output_graph_def = graph_util.convert_variables_to_constants(
                sess, graph.as_graph_def(), [FLAGS.final_tensor_name])
        with gfile.FastGFile(FLAGS.output_graph, 'wb') as f:
            f.write(output_graph_def.SerializeToString())
        with gfile.FastGFile(FLAGS.output_labels, 'w') as f:
            f.write('\n'.join(image_lists.keys()) + '\n')


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--image_dir',
        type=str,
        default='/content/drive/MyDrive/Datasetnew',
        help='Path to folders of labeled images.'
```

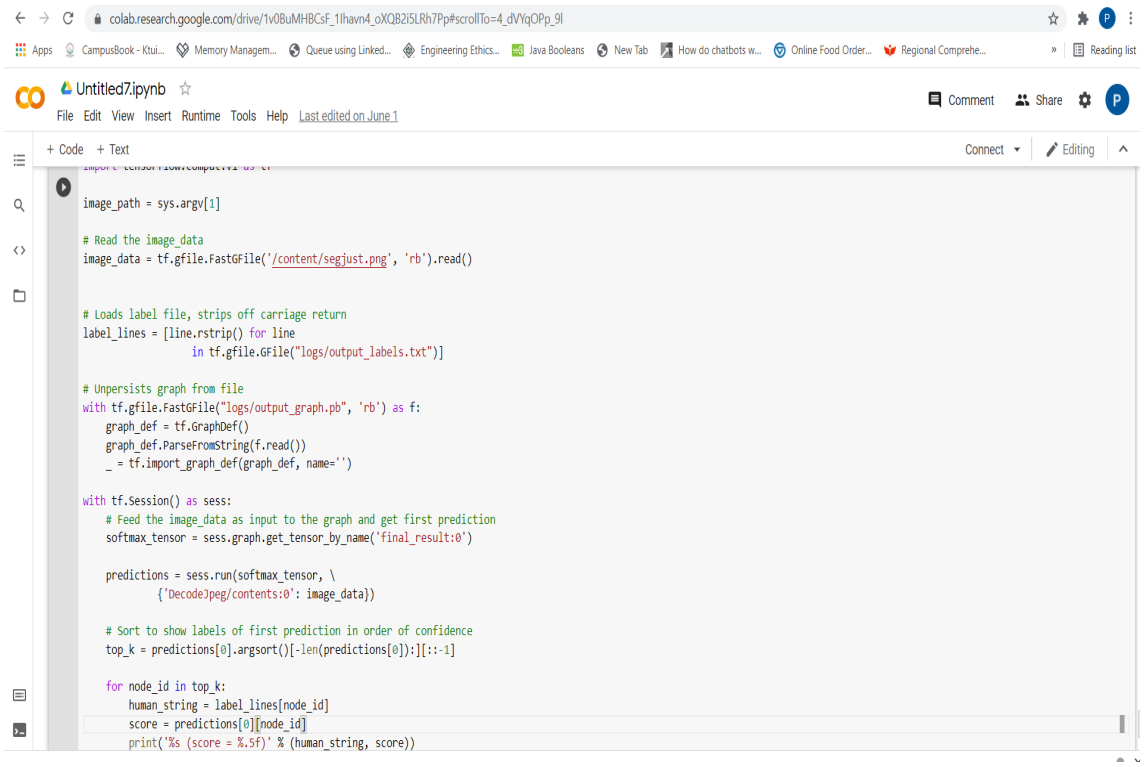


Figure 9.3: Testing the CNN Classification Model

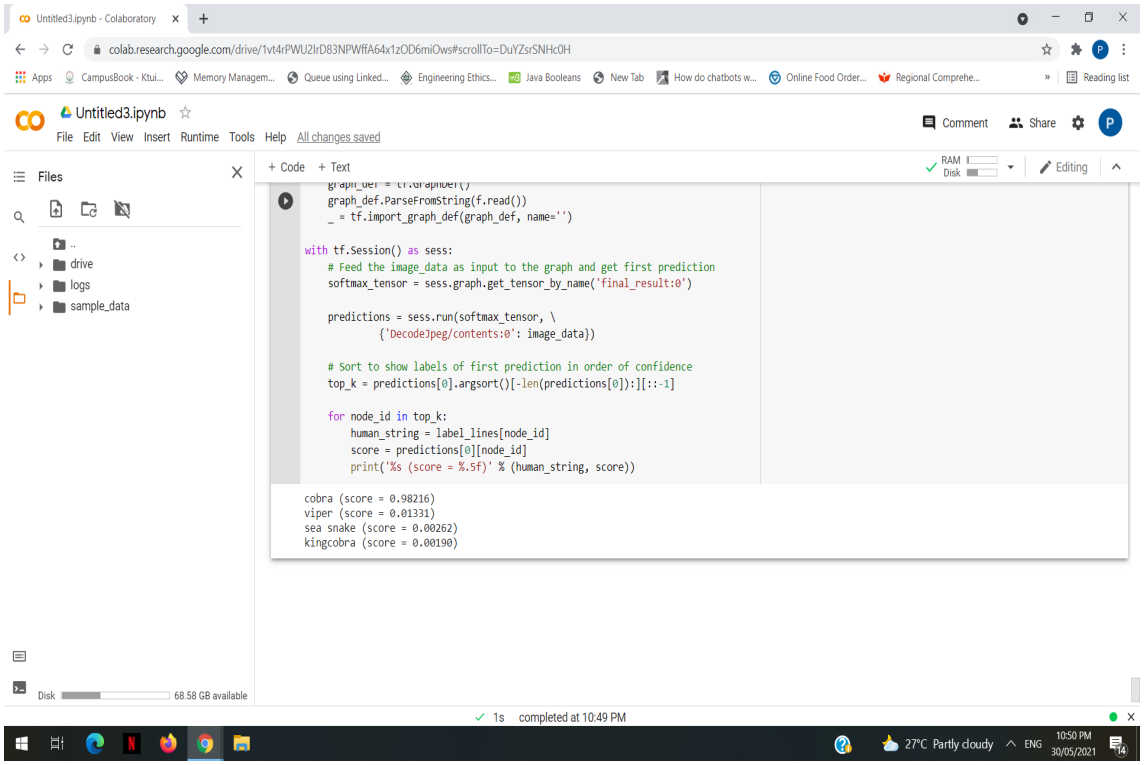


Figure 9.4: Final Result

REFERENCES

[1] Isa Setiawan Abdurrazaq, Suyanto Suyanto, and Dody Qori Utama. Image-based classification of snake species using convolutional neural network. In *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pages 97–102. IEEE, 2019.

[2] Isabelle Bolon, Andrew M Durso, Sara Botero Mesa, Nicolas Ray, Gabriel Alcoba, François Chappuis, and Rafael Ruiz de Castañeda. Identifying the snake: First scoping review on practices of communities and healthcare providers confronted with snakebite across the world. *PLoS one*, 15(3):e0229989, 2020.

[3] Rafael Ruiz de Castañeda, Andrew M Durso, Nicolas Ray, José Luis Fernández, David J Williams, Gabriel Alcoba, François Chappuis, Marcel Salathé, and Isabelle Bolon. Snakebite and snake identification: empowering neglected communities and health-care providers with ai. *THe Lancet Digital Health*, 1(5):e202–e203, 2019.

[4] Shamimi A Halim, Azlin Ahmad, Norzaidah Md Noh, Azliza Mohd Ali, Nurzeatul Hamimah Abdul Hamid, Siti Farah Diana Yusof, Rozianawaty Osman, and Rashidi Ahmad. A development of snake bite identification system (n'viter) using neuro-ga. In *2012 International Symposium on Information Technologies in Medicine and Education*, volume 1, pages 490–494. IEEE, 2012.

[5] NPAUD Hernawati, DQ Utama, et al. Image processing for snake indentification based on bite using local binary pattern and support vector machine method. In *Journal of Physics: Conference Series*, volume 1192, page 012007. IOP Publishing, 2019.

[6] R Kamalraj. Deep learning model for identifying snakes by using snakes' bite marks. In *2020 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–4. IEEE, 2020.

[7] Kalana P Maduwage, Margaret A O'Leary, Anjana Silva, and Geoffrey K Isbister. Detection of snake venom in post-antivenom samples by dissociation treatment followed by enzyme immunoassay. *Toxins*, 8(5):130, 2016.

[8] Mahima Shanker Pandey, SS Soam, and Surya Prakash Tripathi. Detection of knee osteoarthritis using x-ray. *International Journal of Computer Science and Information Technologies, ISSN*, pages 0975–9646, 2016.

[9] Erick Rodriguez-Esparza, Laura A Zanella-Calzada, Diego Oliva, Ali Asghar Heidari, Daniel Zaldivar, Marco Pérez-Cisneros, and Loke Kok Foong. An efficient harris hawks-inspired image segmentation method. *Expert Systems with Applications*, 155:113428, 2020.

[10] Xiaoling Xia, Cui Xu, and Bing Nan. Inception-v3 for flower classification. In *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, pages 783–787, 2017.