S	
E1	
E2	
For Office Use Only	

Masters Project Final Report

(MCS)

2019

Project Title	Solo traveler application with snake detection
Student Name	W.A.Isuru Mahesh Wickramasinghe
Registration No. & Index No.	2017/MCS/090 17440909
Supervisor's Name	Dr. G.D.S.P Wimalaratne

For Office Use ONLY

Solo traveler application with snake detection

A dissertation submitted for the Degree of Master of Computer Science

W.A.I.M.Wickramasinghe

University of Colombo School of Computing

2020

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: W.A.I.M.Wickramasinghe

Registration Number:2017/MCS/090

Index Number:17440909

_____

Signature: Date:

This is to certify that this thesis is based on the work of

Mr./Ms.

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name:Dr G.D.S.P. Wimalaratne

_____

Signature: Date:

Abstract

The snakes are not the most famous animals among all other animals. They are perceived as animals should be feared and killed. There are more than 3000 snake species all around the world except in Antarctica, Iceland, Ireland, New Zealand and Greenland. Among these species 600 are venomous and 200 species venomous enough to kill a man.

Snakebite is a critical medical emergency that requires quick medical treatment. To make medical treatment, identifying snake is the most vital task. People normally identify snakes based on visual features like body shape, eye shape and color patterns. The knowledge of identification snakes which is not ordinary for many people where only a few experts have this knowledge.

This study was focused on creating an automatic snake image classification system that supports mobile and web-based systems. To the best of my knowledge this is the best mobile application that works fully offline and give better accuracy. Convolutional Neural Network was applied to train the snake classification model because Convolutional Neural Network has been obtained great results in image classification.

Snake dataset was collected through AICrowd competition. Few Convolutional Neural Network algorithms had been applied on the snake dataset and identified the best algorithm for snake classification.

The mobile system was optimized to classify the snakes with and without internet. Therefore, TensorFlow Lite was used to convert the trained image classification model to mobile support format. TensorFlow and Keras was used as the framework for developing and testing the image classification model.

Acknowledgement

I would like to express my sincere gratitude to my supervisor Dr. G.D.S.P. Wimalaratne Senior Lecturer of University of Colombo School of Computing, for his invaluable guidance and support. He always responded to my questions so promptly and the valuable comments and advices that given by him helped me to write the thesis in a professional way.

And also, I would like to express my sincere gratitude to Dr. Kasun Karunanayaka and Mr Prabhash Kumarasinghe for the given support during the project work.

Table of Contents

List of Figures

List of Tables

1. Introduction

The snakes are not the most famous animals among all other animals they are perceived as animals should be feared and killed. There are more than 3000 snake species all around the world except in Antarctica, Iceland, Ireland, New Zealand and Greenland. Among these species 600 are venomous and 200 species venomous enough to kill a man [1].

Annually, at least 20,00 deaths and 421,000 envenomings occur because of snake bites worldwide. These statistics can be high as 94,000 deaths and 1,841,000 envenomings.

Snake bite is an ordinary medical problem among plantation workers and farmers in South East Asian region. Most of the younger generation are victims of snake bites [2]. These snake bites can be cause death or chronic disability therefore It has a significant impact at the economy of the country.

Sri Lanka boasts one of the highest rates of biological endemism in the world whether in plants or animals and is included among the top five biodiversity hotspots in the world. Therefore, Sri Lanka is a famous destination for travelers all over the world for exploring the biodiversity.

Travelers who visit Sri Lanka go into jungles to explore the biodiversity. Most of them do not aware what snake species live in that area. This application makes them aware of what snake species are generally found in that area. In their journey they may be get snake bites, without proper identification and first aids their life can be in danger.

Sri Lanka is a country which has highest snake bites in the world. Because of the highest biodiversity of the country many snake species can be found. Sri Lankan snake fauna comprise of 100 species belonging to 10 families. Out of these 100 species, 87 live on land, 14 live in the ocean, and the remaining one inhabits brackish water [3]. Nearly 49% (50 species) of the snake species found in Sri Lanka are endemic to the island, or do not occur naturally anywhere else in the world.

The snakes of Sri Lanka that live on land can be categorized into four groups, depending on the lethality of their venom. Accordingly, 5 species can be considered as deadly

venomous- Cobra, Russell's Viper, Common Krait, Sri Lankan Krait, Saw-scaled Viper; 5 species as mildly venomous and 12 species as mildly venomous. The remaining 61 species are non-venomous. This demonstrates that the majority of snake species (63%) are in fact, harmless.

Snakebite is a critical medical emergency that requires quick medical treatment. To make medical treatment, identifying snake is the most vital task. People normally identify snakes based on visual features like body shape, eye shape and color patterns [4]. The knowledge of identification snakes which is not ordinary for many people where only a few experts have this knowledge. This application helps to identify snakes that live in those areas and what first aids can be done after snake bite.

1.1 Problem

After a snake bite identifying the snake is the most vital task for giving antivenom to the patient. Antivenom is the only effective medicine for a snake bite. People is used to kill the snake after a snake bite and bring it to the hospital with the patient. People put their lives in more danger when trying to kill the snake and sometimes they kill the snake without identifying it is a nonvenomous one. Identifying the snake should be done by a practiced physician. If snake was not caught laboratory tests can be done to identify snake by its venom. It is a time-consuming task that causing delay of medical treatment. Most of time snake is identified after seeing the death snake or according to the details that described by the patient. This identification process can be led to highly inaccurate results. Snake is identified having a very lower accuracy of 53% after a snake bite [5].

There is no automated system to identify snakes using images. With an automated system it would be easy to identify snakes and it would be reduced the danger in human lives when they put trying to catch snakes. Identifying snakes with earliest possible will give a good chance to save the life of the patient.

Furthermore, due to the lack of understanding of snakes in Sri Lanka, they are frequently killed regardless of their identity. Therefore, the ability to identify snakes will save the lives of human beings and also harmless snakes.

Snakes identification is more challenging because some species have patterns that vary depending on their age, some species have patterns that vary depending on their location and two species might look very similar, with one being venomous and the other not [6].

1.2 Motivation

As mentioned above there is no automated framework to identify snakes. It would be very helpful to doctors and people to identify snakes easily. It would reduce the deaths of the people and also the snakes. Sri Lanka is a country which has a great snake fauna and one of the countries that has highest snake bites. Travelers all around the world visit Sri Lanka to study the biodiversity of Sri Lanka. Having an application to study snakes and identify biodiversity areas of Sri Lanka will be helpful to travelers. This automated framework can be converted to Sri Lankan context with few modifications. Taking the initial step to creating this snake identification framework and giving some useful application to the society is the main motivation.

1.3 The Exact Computer Science Problem

Human visual cortex that made up of 140 million neurons is one of the most difficult part of the brain to understand. Human visual context is responsible for processing and interpreting visual data to give perception and formulate memories. Humans can extract much information after seeing an image and tell the whole story behind that.

Computer vision is a field of computer science that deals with understanding the images and videos like humans do. Computers do not see images and videos like humans do. They see images as pixels, numeric values that represent color variations of red, blue and green.

Computers should determine the pattern of pixels to understand images and identify objects in the images. With the advancement of deep learning computer vision has gained a good progress for identifying objects in an image. To have a good accuracy using a deep learning algorithm, it should be trained with huge amount of data and require great computational power. Still computer vision is not good at understanding what is going on in the images. Understanding relationship between objects in the images needs common sense and prior knowledge. Human visual cortex has been trained for years but making a machine to interpret as human visual cortex always been a challenge.

The results of the snake identification framework should be very accurate because depending on the identified snake doctors decide the antivenom for the snake. Most of the snake bites happen in rural areas so internet connectivity cannot be guaranteed so application should work without internet.

Until now there is no automated classification system to identify snake species from images. Snakes classification is an ongoing research problem. This project is target on developing such an application which will capture images of snakes and classify them using image processing and deep learning technologies.

1.4 Aims and Objectives

The main objective of the project is to develop an automated framework to identify snakes using Image processing and machine learning techniques. Most snake bites are reported in rural areas therefore Internet cannot be guaranteed hence application should be worked with internet and without internet.

The following objectives can be found in this research

- Develop a mobile application to identify snakes in real time.
- Mobile application should produce accurate results with minimum computational time.
- Identify the best Machine learning algorithm to use for snake identification for mobile and server.

- Build and train Machine learning algorithm using snake images.
- Develop a web application to identify snake species.

1.5 Scope

This system identifies 10 snake species Common gartersnake,Dekay's brownsnake,Western diamondback rattlesnake,Black rat snake,Copperhead,Eastern racer,Rough green snake,Eastern hognose snake,Timber rattlesnake and Foxsnake. This system includes two applications a mobile and a web-based application. Mobile application is the main application that comes with many features and web application only comes with limited features. Mobile application runs on Android platform. It comes with features such as Identifying snakes on real time, display snake details and show nearest biodiversity areas and hospitals according to user's location. Users can identify snakes by taking a picture from camera or uploading an image in the gallery. Identifying snakes works with internet and without internet. To identify snakes without internet mobile supported machine learning model should be developed. When developing the mobile application, the size of the application, the performance, memory, computational power of the device and the accuracy of the results are considered.

First machine learning model is developed to identify above 10 snake species. Machine learning models need high computational power, memory and they are big in size. Therefore, developed machine leaning model should be converted to mobile support form. When optimizing a machine learning model to mobile support form, it degrades the accuracy. Therefore, optimized machine learning model is deployed with mobile application and other one is deployed in a powerful server. Hence mobile supported machine learning model gives less accuracy, mobile application is designed to get results from the server when user has internet connectivity. When user has no internet connectivity it will rely on the machine learning model that runs locally inside the application.

Web application only supports to identify snakes when an image uploads to the system. It directly uses the server deployed machine leaning models to retrieve results.

1.6 Novelty of the Project

The main outcome of the project is a stand-alone mobile application that works offline for identifying 10 snake species - Common gartersnake,Dekay's brownsnake,Western diamondback rattlesnake,Black rat snake,Copperhead,Eastern racer,Rough green snake,Eastern hognose snake,Timber rattlesnake and Foxsnake. People normally identify snakes based on visual features like eye shape, body shape and color patterns. The knowledge of identification snakes which is not ordinary for most people where only a few experts have this knowledge. It is proved that Computer vison can be achieved more accurate results than human vision. In this research Computer vison accuracy is evaluated against snake species context.

There are many Machine learning algorithms for Image classification. Their accuracy can be varied against the input. The best Machine learning algorithm is identified against snake species context.

The Machine learning models are required high computational power and more storage, mobile devices do not have high computation power and more storage so best mobile supportive Machine learning model is identified according to above requirements. Natively machine learning models do not support mobile devices, they should be converted to mobile support format. After this conversion, the accuracy of the model can be degraded, best mobile supporting Machine learning model is identified.

1.7 Structure of the Dissertation

The chapters of this dissertation describe how the project is planned, organized and implemented. The first chapter describes the problem that was addressed and the outcome of the project.

The second chapter describes the literature review that was done during the project. Literature review study described related work and methods that had been applied and used by other researchers in image classification domain.

The third chapter describes the selected methodology for snake classification after doing the literature review. This chapter also describes the architecture, design phase and implementation of the application. In this chapter there is a detailed description how dataset was collected, how dataset was preprocessed and how application was implemented.

The fourth chapter describes the results that obtained after project was completed. This chapter also describes what were the techniques that applied to evaluate the project work and the accuracy of applied methodologies in the project.

Final chapter describes conclusions that observed after project was completed.

2. Literature Review

2.1 Introduction

Snakes identification is more challenging because some species have patterns that vary depending on their age, some species have patterns that vary depending on their location and two species might look very similar, with one being venomous and the other not [2]. It is very important to do a literature survey for studying related work and finding approaches that have been taken by other researchers for snake classification and image processing related applications. After identifying their approaches some valid insights can be taken when selecting the most suitable approach for the current application. In this chapter related work is analyzed and discussed their outcomes.

2.2 Related work

Prakash M. Manikar et al. proposed an image-based plant leaf disease recognition by using the feedforward back propagation network (BPNN) [7]. Images of various leaves were collected using high resolution camera to get better results. The research analyzed RGB images of plant leaves and extract suitable features. The noise added during image acquisition should be removed. RGB images are device dependent so they are converted into independent color space which providing same color regardless of the device used to take pictures.

Image segmentation is done using k-means clustering to simplify the representation of image in a more meaningful way. K-means clustering has been selected over hierarchical clustering because K-means treats each observation in the data set as an object having a location in space. Mostly green colored pixels have been identified and removed because they were representing healthy areas.

The extracted features have been given as inputs to pre-trained neural network for automatic classification of diseases. Neural network has been selected as a classification algorithm because It is a recognized technique.

Mohini Niraj Sheth et el. proposed an image-based snake identification system by using Principal component analysis (PCA) algorithm. In their work total number of 85 images have been used. Forty images are of non-venomous snakes, twenty-seven images are of venomous snakes and 19 images are of semi venomous snakes.

Grayscale images have been used as they required less memory storage and image operations were easy. PCA algorithm had been used to extract feature vectors of images. PCA requires centralize data. Centralized data was obtained computing row mean and subtracted row mean from image data. Computed scattering matrix of centralized data and eigen value of scatter matrix has been found out. In order to get eigen vectors of larger eigen values sort the eigen value in descending order and computed feature vector after that plot feature vector and store this data. Euclidean distance has been calculated between feature vector of training image and test image. The image which has minimum Euclidean distance has been selected [8].

In their work average success rate of identifying snakes is 74%. It is a low score rate therefore PCA algorithm can't be considered as a good algorithm for identifying snakes.

In the work Amiza Amir et el. investigated the accuracy of five machine learning algorithms - naive Bayes, nearest neighbors, k-nearest neighbors (k-NN), backpropagation neural network, and decision tree J48 for image-based snake identification problem. Color and Edge Directivity Descriptor (CEDD) has been used to extract features of 349 images that belongs to 22 different snake species [9].

The Weka tool has been used to implement five machine learning algorithms using CEDD data set. In their work they got 75.54% with Naïve Bayes,87.63% Backpropagation neural network ,89.22% Nearest neighbor and 71.29% with Decision tree J48

Luz Jimenez et el. proposed image-based species identification system using Artificial Neural Network (ANN). In their work total of 11,198 images were tested belongs to fish, plants and butterflies. In Image preprocessing, Grabcut's algorithm has been used to

remove image background and converted images to grayscale. Different filters were added to remove the image noise. Feature extraction was done after image preprocessing,15 geometrical, morphological and texture features are extracted and used for pattern recognition.

A neural network has been trained using extracted features. Input neurons was determined from the extracted number of features which is 15. The success rate of this research is 91.65% for fish identification,92.87% for plants identification and 93.25% for butterfly identification [10].

Alejandro Arteaga et el. proposed image-based real time snake classification system using regional-based convolutional neural network(R-CNN) architectures [11]. In their work total number of 247 images were collected that belonged to 9 Pseudalsophis snake species. TensorFlow had been used to load pretrained R-CNN models such as: ResNet, Inception V2, VGG16 and MobileNet. Different size of training and test dataset had been used to train the deep learning models. Their work shows good results with accuracy of 75% ResNet,70% Inception V2,70% VGG16 and 10% MobileNet.

In the work Alex Pappachen James et el. described what features are most important to successfully identify a snake. The snake images for the experiment have been collected from forest across different parts of Kerala, India. Total number of 1299 snake images were collected that belongs to spectacled cobra, russel's viper, king cobra, common krait, saw scaled viper and hump nosed pit viper. Feature dataset has been created by extracting 38 physical characteristics from snake images.

To perform snake classification multiple classifiers have been used. The dataset was randomly split into 5% samples in training set and 95% in test set and performance evaluated on individual classifiers. The selection of features is performed on the training set. Selection and testing was repeated 100 times to ensure statistical correctness and got Naïve Bayes 77.69,Bayes net 78.81 and Multilayer Perceptron 86.85 accuracy [12]. The research has been proved that 15 characteristics enough to identify a snake.

Sathiesh Kumar proposed a system to identify flower species using Convolutional Neural Networks and Transfer learning. When recognizing flowers, the main features to be considered is color, shape and texture. The unpredictable variety of existence of above features in flowers, the feature extraction is difficult. In his work without applying different feature extract methods Convolutional Neural Networks were directly used.

Image dataset was collected from the Visual Geometry group of University of Oxford. Transfer leaning has been used to train the dataset therefore no need to train the CNN from scratch. OverFeat is a trained network on ImageNet and it has been used as the feature extractor by freezing all the pre-trained layers except Fully Connected layer in CNN. In this method over 90% of accuracy was obtained [13].

In the work of Rupali S.Zambre et el.proposed a method to identify cotton leaf disease using SVM. The RGB color of images of cotton leaf are collected and several feature extraction methods have been used. SVM has been used as the image classifier and obtained 97% of accuracy [14].

In the work of Neha Sharma et el.compared the accuracy of three Convolution Neural Networks AlexNet, GoogleNet and ResNet50.The architecture of the networks differs from each other by number of internal layers and techniques used therefore accuracy of the networks are differed. To identify the accuracy of the networks three datasets-CIFAR-10, CIFAR-100 and MNIST have been used. CIFAR-100 dataset has 50000 images and 10000 test images that belongs to 100 different classes. CIFAR-10 dataset has 5000 images and 10000 test images that belongs to 10 different classes.

In his work he has calculated average accuracy of identifying images for each dataset using above three CNN networks. GoogLeNet has given the best accuracy for CIFAR-100 dataset with 64.40% accuracy. Resnet50 has given the best accuracy for CIFAR-10 dataset with 78.10% accuracy [15].

2.3 Summary of Literature Review

Various methodologies were identified in the image classification context by going through many research papers. These methodologies use different techniques in image classification and they have different pros and cons.

The findings that obtain during this study was summarized and further studied.

2.3.1 Taxonomy of Image Classification Models

According above research papers following image classification methods were identified.

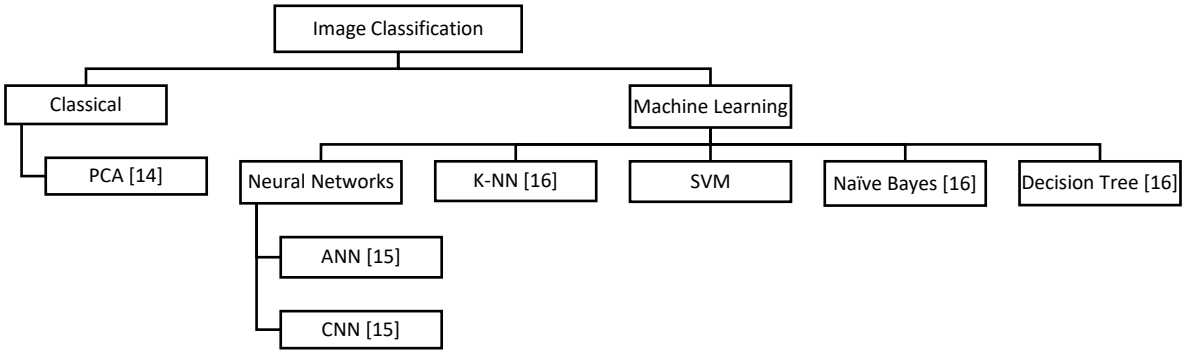


Figure 1: Image Classification Models

2.3.2 Comparison of Classifiers

Classifiers	Advantages	Disadvantages	References
PCA	In a dataset there can be more than thousands of features and some of those features can be corelated. PCA can efficiently identify those correlated features.	PCA components are less readable when comparing to original features.	[16]
	Improving the algorithm efficiency by reducing the correlated data set and reduces the overfitting of the algorithm.	Information can be loss because of high variance of feature set.	
ANN	Capable in distinguishing complex nonlinear relationship between independent and dependent variables.	Great tendency of data overfitting.	[17]
	Simplistic statistical training.	Bigger computational load.	[17]
CNN	Multiple features can be extracted simultaneously.	High computation level.	
	Robust to noise	No capable for generalization.	
KNN	No training needed.	Susceptible to noise.	[18]
	Simplest classifier.	Costly testing for each instance.	

SVM	Great generalization potential.	Speed and size constraint for both training and testing	
	Exceptionally robust.	Complex algorithm structure.	
		Slow training.	
Naïve Bayes	It requires short computational time for training.	The Naive Bayes classifier needs a huge dataset to obtain accurate results	[18]
	It increases the classification efficiency by removing the unrelated features.	Less accurate as compared to other classifiers on some datasets.	
Decision tree	Decision Trees are very simple and fast	It has long training time.	[18]
	It can also deal with noisy data.	Decision trees can have significantly more complex representation for some concepts due to replication problem	
	It supports incremental learning		

Table 1: Comparison of Classifiers

3. Methodology

3.1 Introduction

The methodology is the way that project is implemented with the findings and the knowledge gained through the literature review. This chapter provides a comprehensive overview of the image classification techniques that was used, the dataset details, how dataset preprocessed, overall project architecture and the implementation.

3.2 Problem Analysis

Snakes identification is more challenging because some species have patterns that vary depending on their age, some species have patterns that vary depending on their location and two species might look very similar, with one being venomous and the other not. Because of this variation it is very difficult to extract features manually.

Proper Image classification method should be chosen to address above problem. When ANN and CNN are compared, ANN is not suitable for image classification because these networks need more processing for images that have more pixels. Consider an image of size [100*100*3], ANN should have 300,000 weights in its first layer to receive this input vector [13].Feature engineering should be done by an expert for traditional Machine Learning algorithms but CNN can learn features by itself. CNN has shown high accurate results than traditional ML algorithms in image classification domain [19].

3.3 Convolution Neural Network

Convolution Neural Networks are going to use as the Image classification methodology. CNNs are specially used in Computer Vision Applications that involve Image Classification and Object recognition. Snakes recognition is a combination of both Image Classification and Object Recognition.

CNN consists of multiple convolutional layers that followed by one or more fully connected layers and one output layer. CNN can receive an image as input, apply some mathematical operations and classify it under certain categories. Features should be hand engineered in traditional classifiers but CNN has the ability to learn the features by itself. CNN receives an image as three-dimensional matrix. Three dimensions are height, width and depth of the image, where depth is the number of color channels. As an example, CNN takes a [64x64x3] color image, pass it through the layers while identifying various features and predict the class or label of the image [20].

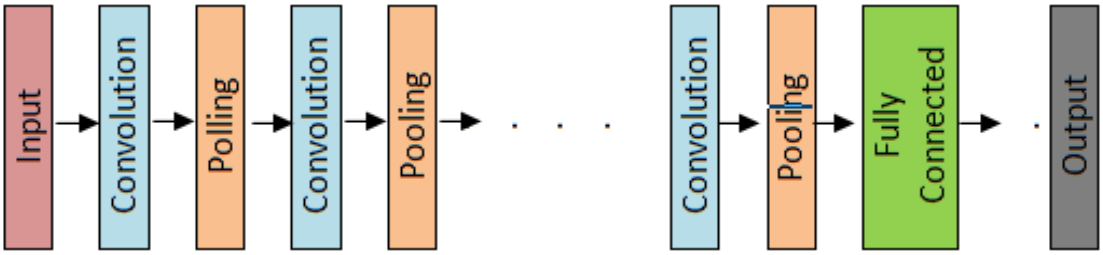


Figure 2: CNN Architecture

CNN has three types of layers as follows

1.Convolutional Layer

2.Pooling Layer

3.Fully-Connected Layer

3.3.1 Convolutional Layer

Convolutional layer is the brain of the CNN. It identifies the features of an image. Normally there are multiple convolutional layers. The first convolutional layer is responsible for identifying high level features of an image. Kernel is a matrix that has the same depth of the input image. For an example, Kernel dimensions can be 5x5x3 for an image size of 64x64x3. Kernel moves over the image and it does the matrix multiplication between Kernel values and original image values. These multiplications are summed up to create a

feature map. This process continues until full image is covered. To identify various features of an image, multiple filters are applied. The applied Kernel values are later modified by the backpropagation technique

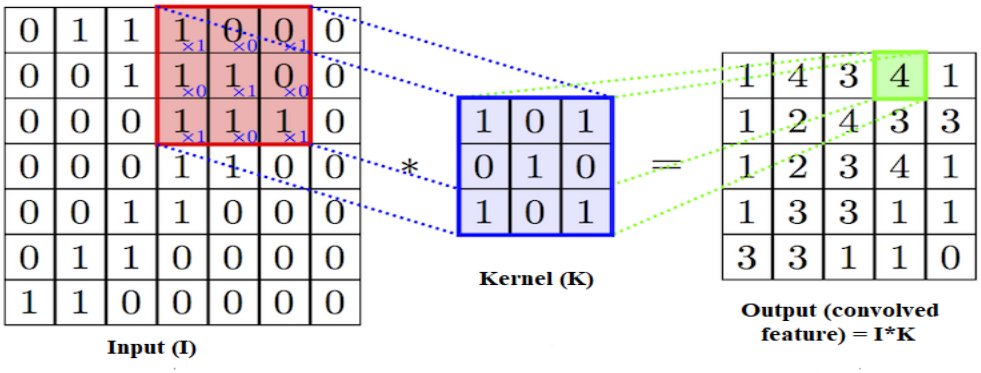


Figure 3: Convolutional Layer

3.3.2 Pooling Layer

It is common to add a pooling layer between two Convolutional layers in a CNN. The purpose of a pooling layer is to reduce the dimensionality of the feature maps generated in Convolutional layer. It helps to reduce the needed computational power and training time to train the network. There are multiple pooling techniques, among those most frequently use one is max pooling. Max pooling takes the largest value among the selected region as shown in figure 4.

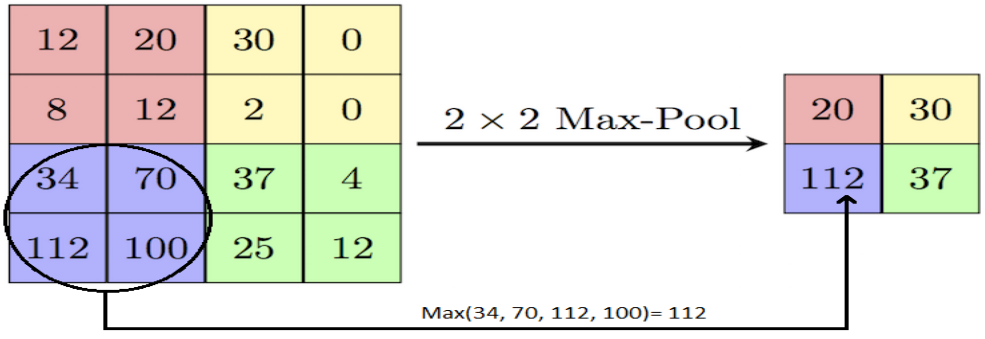


Figure 4: Pooling Layer

3.3.3 Fully-Connected Layer

In the fully-connected layers it takes all the input from previous layers and flattened into a feature vector to predict the output probabilities.

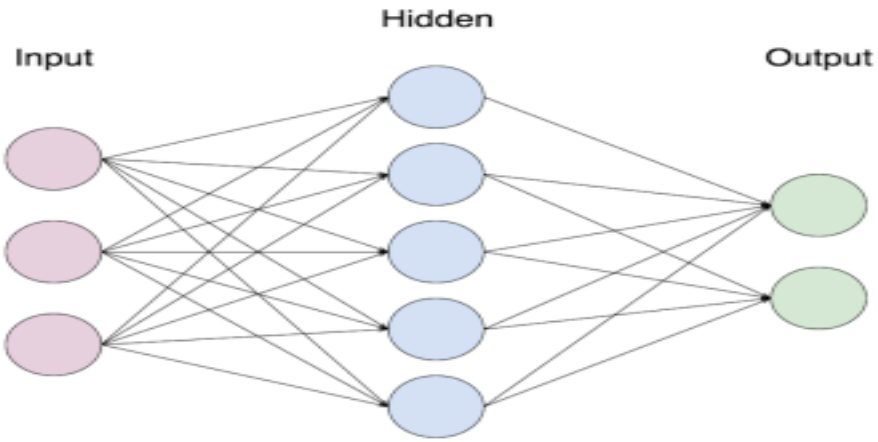


Figure 5: Fully Connected Layer

3.4 CNN Types

There are many CNNs are currently available**.** The architecture of the networks differs from each other by number of internal layers and techniques used therefore accuracy of the network is differed [21]. When selecting a CNN, it's accuracy, computation power, model size and mobile support should be considered. Therefore, multiple CNNs that has high accuracy such as:MobilenetV2, InceptionResNetV2 and DenseNet121 were evaluated and got the best CNN that suitable for mobile and the server [20].

3.4.1 MobileNetV2

MobileNetV2 is a lightweight, low-latency and low-power power CNN architecture that can be implemented in low-end devices like mobile devices. MobileNetV2 is an advancement of MobilnetV1 architecture. MobileNetV2 has used depth wise separable

convolutional layers in MobileNetV1 that can significantly reduce the model size and the complexity of the network. MobileNetV2 has two new features like linear bottle necks and shortcut connections between bottlenecks [22].

The architecture of MobileNetV2

Input	Operator	t	c	n	s
$224^2 * 3$	Conv2d	-	32	1	2
$112^2 * 32$	bottleneck	1	16	1	1
$112^2 * 16$	bottleneck	6	24	2	2
$56^2 * 24$	bottleneck	6	32	3	2
$28^2 * 32$	bottleneck	6	64	4	2
$14^2 * 64$	bottleneck	6	96	3	1
$14^2 * 96$	bottleneck	6	160	3	2
$7^2 * 160$	bottleneck	6	320	1	1
$7^2 * 320$	Conv2d 1*1	-	1280	1	1
$7^2 * 1280$	Avgpool 7*7	-	-	1	-
1 * 1* 1280	Conv2d 1*1	-	k	-	-

Table 2:MobileNetV2 Architecture

3.4.2 InceptionResNetV2

Traditionally deep learning networks have stacked of layers. When the network goes deeper accuracy gets saturated and then degrades rapidly. This is not causing by network is overfitting it is happened because layers suffer vanishing gradient problem. Traditional deep learning networks learning from data by passing each input through the model and Backpropagation. During the backpropagation weights are updated according to loss function. To calculate the gradient of loss function chain rule is applied. That means calculate the gradient of each layer and multiply the values calculated at layers before that. Because of every value is very small, multiplying them are close to zero. This is called the vanishing gradient problem.

To overcome this problem ResNet introduces residual blocks and skip connection [23]. In ResNet shortcuts are added to skip 2 or 3 layers and skip learning for some layers. This method solved the vanishing gradient problem.

Inception is an architecture that designed going wider rather than going deeper. Inception architecture uses different convolutional filters rather than using one [24]. It increases the ability to identify features of the images more accurately rather than other deep learning networks.

InceptionResNetV2 is an architecture that has designed based on ResNet and Inception taking best of the both worlds.

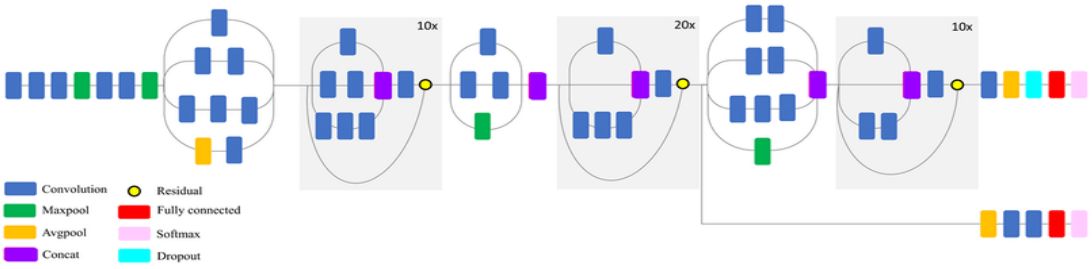


Figure 6:InceptionResNetV2 Architecture

3.4.3 DenseNet

The problem with traditional CNNs is the vanishing gradient problem when they go deeper. Traditional networks pass data from one layer to another. ResNet proved that some layers can be dropped randomly because some layers contribute very little [23]. ResNet has introduced residual blocks to address this issue. The problem with ResNet is, it has very large number of parameters.

In DenseNet the layers are densely connected together: Each layer receives all the inputs from previous layers and pass on its own feature maps to all subsequent layers [25]. DenseNet architecture has significant advantages when comparing to other CNN

architectures like resolving vanishing gradient problem, feature reuse and strengthen feature propagation.

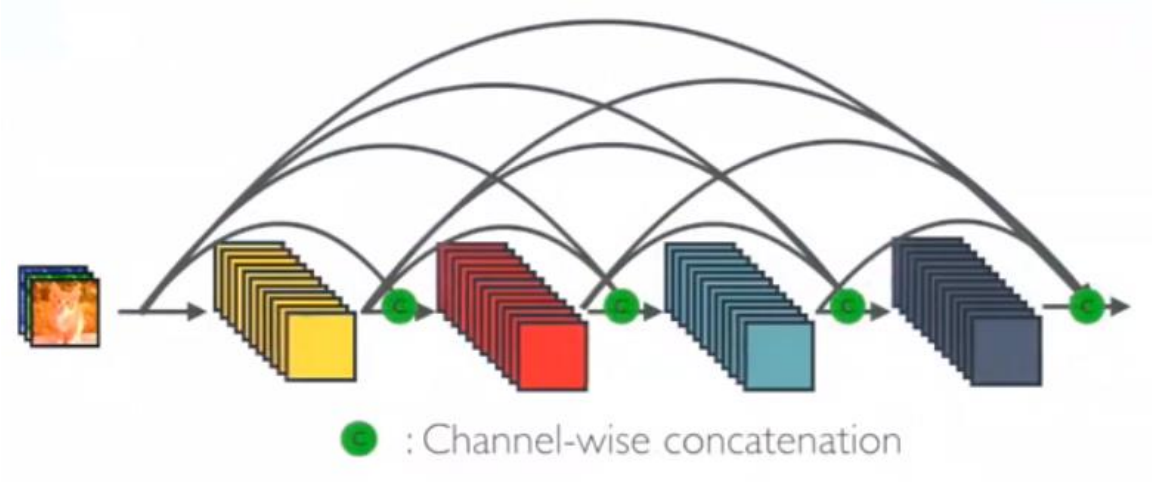


Figure 7: DenseNet Architecture

3.5 Transfer learning

To train a CNN from scratch, huge computational power and time is needed. CNN gives accuracy when there is huge amount of data [26]. If there is small amount of data, at the training time CNN would be overfitted and gives inaccurate results. To overcome these issues, transfer leaning is used. In transfer learning pre-trained models can be used which was trained on a huge dataset [11].

3.6 Dataset

Image dataset was collected through AICrowd competition. The dataset includes images belongs to 10 snake species which contain both test and train images. There are 36,025 train images and 1941 test images in the dataset.

The details of the dataset as follows

Snake Name	Train Images Count	Test Images Count
Timber rattlesnake	1983	142
Common garter snake	10554	517
Rough green snake	1918	144
Foxsnake	1283	88
Eastern hognose snake	1803	105
DeKay's brownsnake	4406	194
Black rat snake	5236	284
Copperhead	2373	164
Western diamondback rattlesnake	5266	249
Eastern racer	1381	54

Table 3: Original Dataset

3.6.1 Handling Class Imbalance

The dataset consists with imbalance classes. The imbalance classes effect the accuracy of the classification model [27]. The majority class dominates the parameters of the classification model. This reduces the error of majority class at the early stage of iterations and increases the error of minority class.

The new class distribution was determined by calculating the average number of images in the existing dataset.

Average images per class = Total Images/Number of classes

= 36025/10

= 3600

Following techniques were applied to handle the class imbalance problem.

Image Augmentation

Image Augmentation is a technique to generate more new images with existing few images [28]. Augmentation techniques such as zoom, horizontal flip, vertical flip, rotation was applied to randomly selected images from minority classes and generated new images.

Keras ImageGenerator method has been used to generate new images.

```
ImageDataGenerator(
                    width_shift_range=0.2,
                     height_shift_range=0.2,
                     zoom_range=0.2,
                     rotation_range=45,
                     horizontal_flip=True,
                     vertical_flip=True,
                     )
```

Following examples shows generated augmented images for Timer Rattle snake

Original Image

Width shift range

Height shift range

Horizontal flip

Vertical flip

Rotation

Zoom

Figure 8:Augmented Images

Random Under Sampling

Under sampling was used to randomly remove images from majority classes until reaching balance distribution

The details of the dataset after class balancing

Snake Name	**Train Images Count**	**Test Images Count**
Timber rattlesnake	3600	200
Common garter snake	3600	200
Rough green snake	3600	200
Foxsnake	3600	200
Eastern hognose snake	3600	200
DeKay's brownsnake	3600	200
Black rat snake	3600	200
Copperhead	3600	200
Western diamondback rattlesnake	3600	200
Eastern racer	3600	200

Table 4:Balanced Dataset

3.7 Tflite Models

TensorFlow Lite converts already trained models that was trained on high power machines to light weight mobile supported tflite models. Deep leaning models normally use 32-bit floating point data types with high precision. These models need high storage, high CPU power and high memory to run. Mobile phones may not have these requirements to run a deep learning model. TensorFlow Lite can convert 32-bit floating point models to 8-bit integer models. When converting to tflite model there is an accuracy degrade when comparing to original model but comes with other advantages such as: CPU operations are faster for 8-bit integers than 32-bit floating point numbers, module size is reduced 4x when moving from 32-bits to 8-bits and lower bit data means more data can be fit into memory therefore no need to access memory more often.

Floating point numbers has exponent and mantissa. The exponent allows for representing wide range of numbers and mantissa gives the precision. When converting 32-bit float 8-bit integer exponent is replaced by a fixed scaling factor and use integers to represent the value of a number relative to this constant.

3.8 Model Design

3.8.1 Training Model

Following diagram describes the architecture design for training the data models to support server version and mobile version. TensorFlow and Keras is used to train the network and TensorFlow Lite is used to convert the models to mobile support version

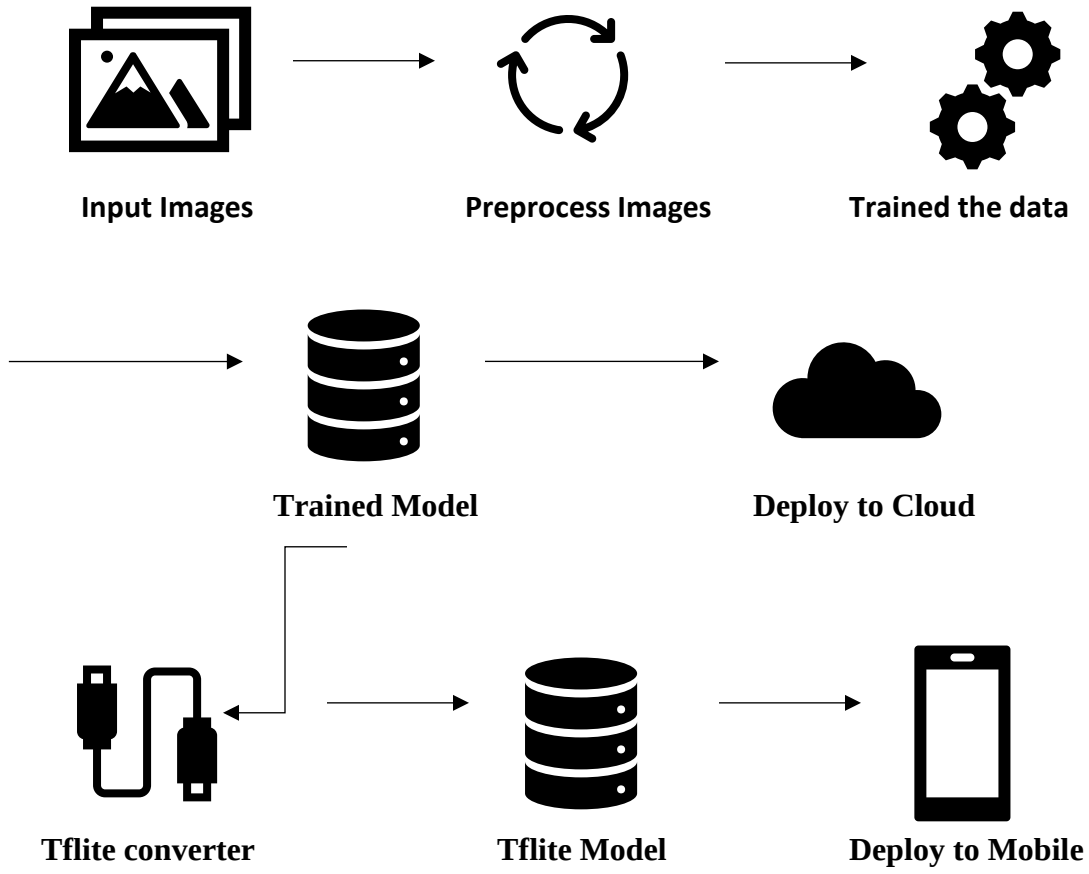


Figure 9:Training Model

3.8.2 Architecture

The following diagram describes the complete architecture of the application how mobile and server version trained machine learning models interact with each other. When there is connectivity mobile application receive the results by call the hosted Machine Learning (ML) model in the server to achieve better performance. When there is no connectivity It gets the results through the Machine Learning (ML) model deployed inside the mobile application.

There is a separate web application to get results from the server directly.

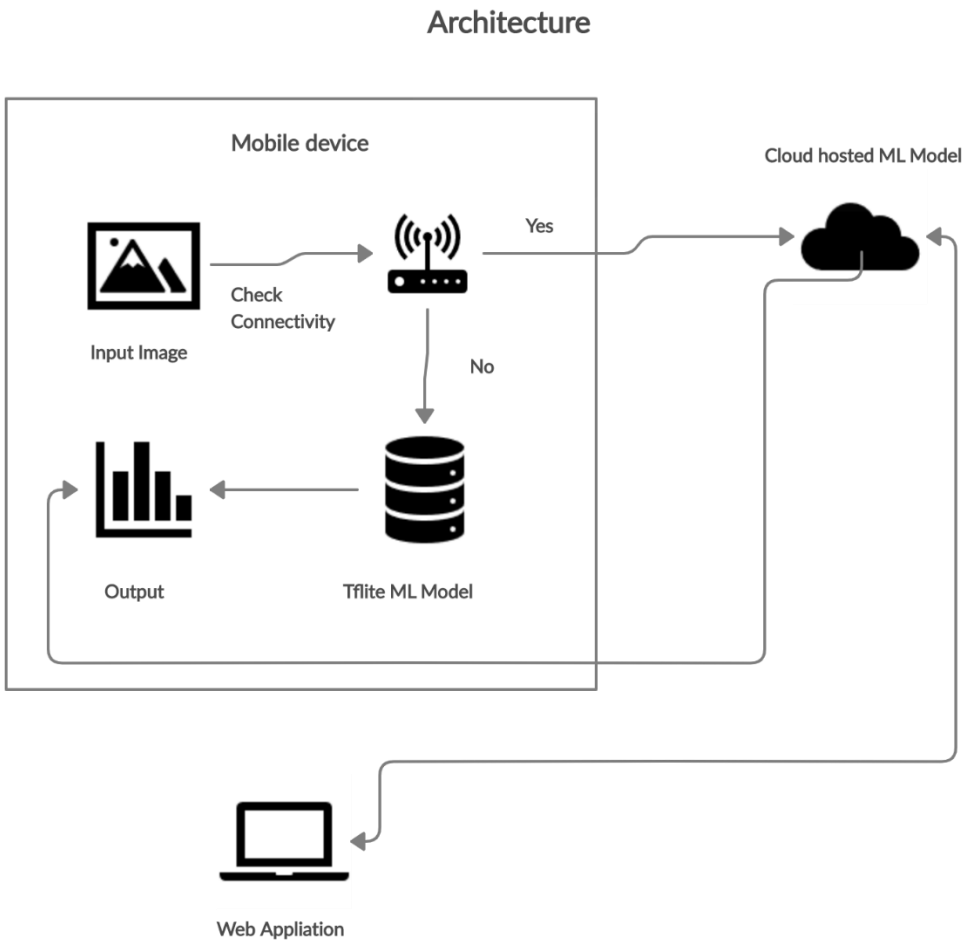


Figure 10: Architecture

3.8.3 Mobile Design

Following are the screens for capturing snake images and displaying results

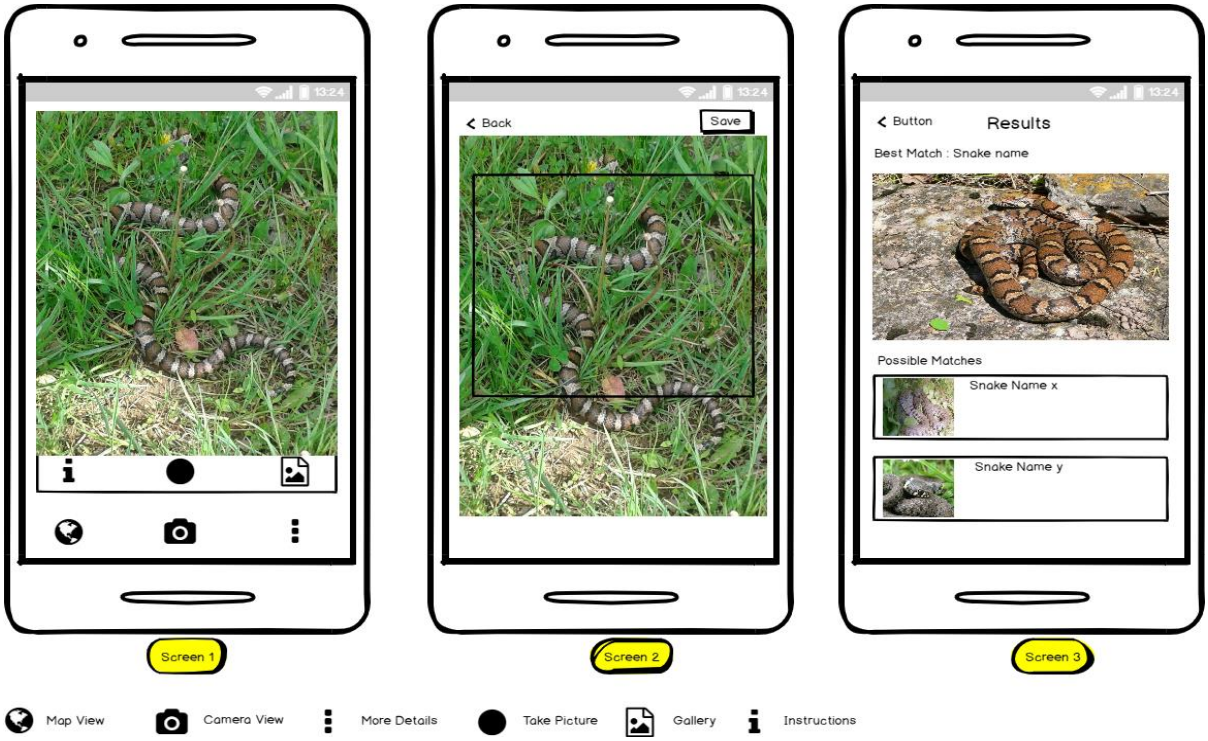


Figure 11:Camera view

In Camera view there are three main screens

Screen1: User can get an image using Take Picture button or select an existing Image using Gallery.

Screen2: After having the Image User can crop the Image

Screen3: After cropping the image, Image is analyzed using the trained model inside the app and possible results are displayed.

Following are the two screens to display snake details and the map view to display biodiversity rich areas near to user's location

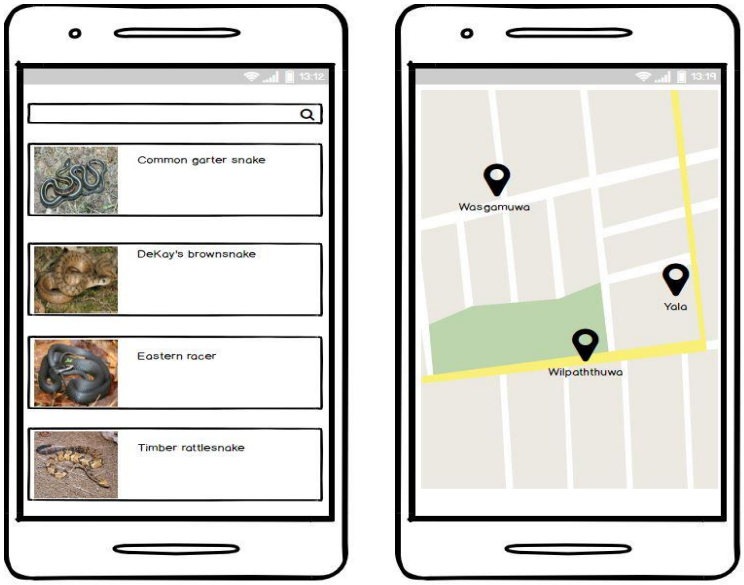


Figure 12:Map View

3.8.4 Web Design

Following is the screen for web application to upload a snake image and get the results directly through the server.

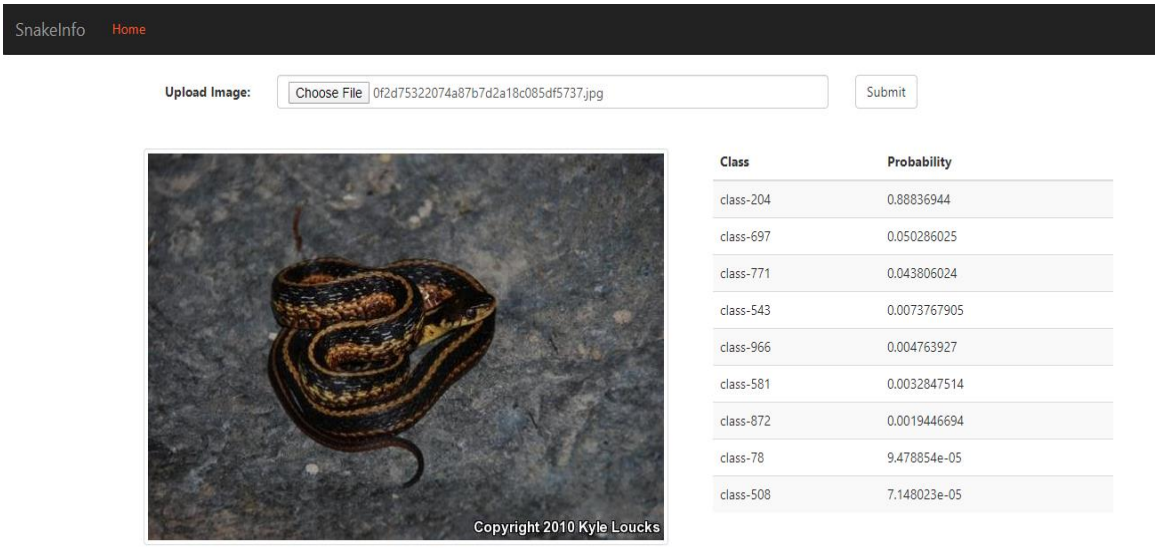


Figure 13:Web View

3.9 Implementation

TensorFlow and Keras was used to train the models and TensorFlow Lite was used to convert the models to mobile support version. Colab Notebook GPU with 25 GB RAM runtime version was used as development environment. ReactJs used for developing we application. Flask Framework used to develop the API and Heroku server was used to host the ML model.

3.9.1 Image Preprocessing

Image should be preprocessed before feeding to the network. Original Images came with different sizes. Image data set is converted to 128*128 images using OpenCV and feed to the CNN networks.

```
CATEGORIES=["class-78","class-204","class-508","class-543","class-
581","class-697","class-771","class-804","class-872","class-966"]

IMG_SIZE=128

training_data=[]

def create_training_data():

    for category in CATEGORIES:

        path=os.path.join(DATADIR,category)

        class_num=CATEGORIES.index(category)

        for img in os.listdir(path):

            try:

                snake_images_array=cv2.imread(os.path.join(path,img))

snake_resize_images_array=cv2.resize(snake_images_array,(IMG_SIZE,IMG_S
IZE))

training_data.append([snake_resize_images_array,[class_num]])

            except Exception as e:

                pass

create_training_data()
```

3.9.2 Training

Keras Api was used to load pretrained models that trained on ImageNet dataset. MobileNetV2, InceptionResNetV2 and DenseNet121 pretrained models were used with transfer learning. These models have been already trained using 14 million images that belongs to 1000 classes. Pretrained model was loaded without including the top classification layer. Convolutional layers of the model were freezed to prevent updating already calculated weights while training. Since pre-trained models were trained on 1000 classes, new output layer should be manually added to classify 10 classes in snake dataset. Before training the new model, it should be complied with loss function and optimizer. Finally, new model was trained with 30 epochs. Dataset was divided as 20% for validation and 80 % for training. Validation dataset was used to evaluate the performance of the dataset on each epoch.

Following sample code describes the training process using MobileNetV2

```
IMG_SIZE=128
IMG_SHAPE = (IMG_SIZE, IMG_SIZE, 3)
base_model=tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                                include_top=False,
                                                weights='imagenet’)
for layer in base_model.layers:
    layer.trainable=False

global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
prediction=Dense(10,activation='softmax')
model = tf.keras.Sequential([
  base_model,
  global_average_layer,
  prediction
])
base_learning_rate = 0.0001
model.compile(optimizer=tf.keras.optimizers.Adam(lr=base_learning_rate)
,
              loss="categorical_crossentropy",
              metrics=['accuracy'])

hist=model.fit(train_data,train_labels,batch_size=32,epochs=30,validati
on_split=0.2)
model.save("Snakes-MobileNetV2.h5")
```

3.9.3 Fine-Tuning

Once the model was trained on the new data, base model was unfreezed and train with newly added classifiers again. This allowed to fine-tune features in base model in order to make them relevant to training dataset features. Trained model was fine-tuned with a much lower learning rate to avoid overfitting the network. Model was fine-tuned 5 epochs.

```
base_model.trainable = True
base_learning_rate = 0.00001
model.compile(optimizer=tf.keras.optimizers.Adam(lr=base_learning_ra
te),
              loss="categorical_crossentropy",
              metrics=['accuracy'])

fine_tune_epochs = 5
total_epochs =  initial_epochs + fine_tune_epochs
history_fine=model.fit(train_data,train_labels_categorical,batch_siz
e=32,epochs=total_epochs,initial_epoch=history.epoch[-
1],validation_split=0.2)
```

3.9.4 Converting to tflite models

After saving the trained model it should be converted to a tflite model for supporting mobile devices. TensorFlow Lite was used to convert the already trained model as tflite model.

```
model=load_model("Snakes-MobileNetV2.h5")
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()
open("snake_mobilenet.tflite", "wb").write(tflite_model)
```

3.9.5 Load tflite models in Android

After adding TensorFlow Lite Interpreter to Android project tflite model can be loading using below code snippet

```
tflite = Interpreter(loadModelFile())

private fun loadModelFile(): MappedByteBuffer {

        val fileDescriptor =
this.getAssets().openFd("snake_mobilenet.tflite")

        val inputStream =
FileInputStream(fileDescriptor.fileDescriptor)

        val fileChannel: FileChannel = inputStream.getChannel()

        val startOffset = fileDescriptor.startOffset

        val declaredLength = fileDescriptor.declaredLength

        return fileChannel.map(FileChannel.MapMode.READ_ONLY,
startOffset, declaredLength)
```

5. Results and Evaluation

5.1 Results

The accuracy of the CNNs should be measured to identify suitable CNN for classification of snake species. While networks were training their validation accuracy and training accuracy was recorded. After training was completed models were saved and evaluate using Model.Evaluate() method in Keras Models API. Using test dataset clear idea can be obtained how trained ML models behave when they see new data.ML models were trained with different epoch sizes to get better results. Graphs were generated for every model to get clear idea how model accuracy behave while training. Python pyplot library was used to generate graphs. Training parameters were different for each ML model. To get a clear idea about trainable parameters Model.Summary() method used in Keras Models API.

Following are the results that obtain after training each ML model.

5.1.1 MobileNetV2

Following diagram describes trained layers of the network and trainable parameters

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_128 (Model)	(None, 4, 4, 1280)	2257984
global_average_pooling2d (Gl	(None, 1280)	0
dense (Dense)	(None, 10)	12810

Total params: 2,270,794
Trainable params: 12,810
Non-trainable params: 2,257,984

Figure 14:MobileNetV2 Model Summary

MobileNetV2 training models were evaluated using test dataset and following results were obtained. Model is evaluated before fine-tuning and after fine-tuning.

	30 epochs		35 epochs (Fine-Tuning)	
Model	Loss	Accuracy	Loss	Accuracy
MobileNetV2	0.9241	0.6745	0.5994	0.7990

Table 5:MobileNetV2 Training Results

MobileNetV2 training accuracy vs validation accuracy was recorded for each epoch to get a better idea how ML model behave while training.

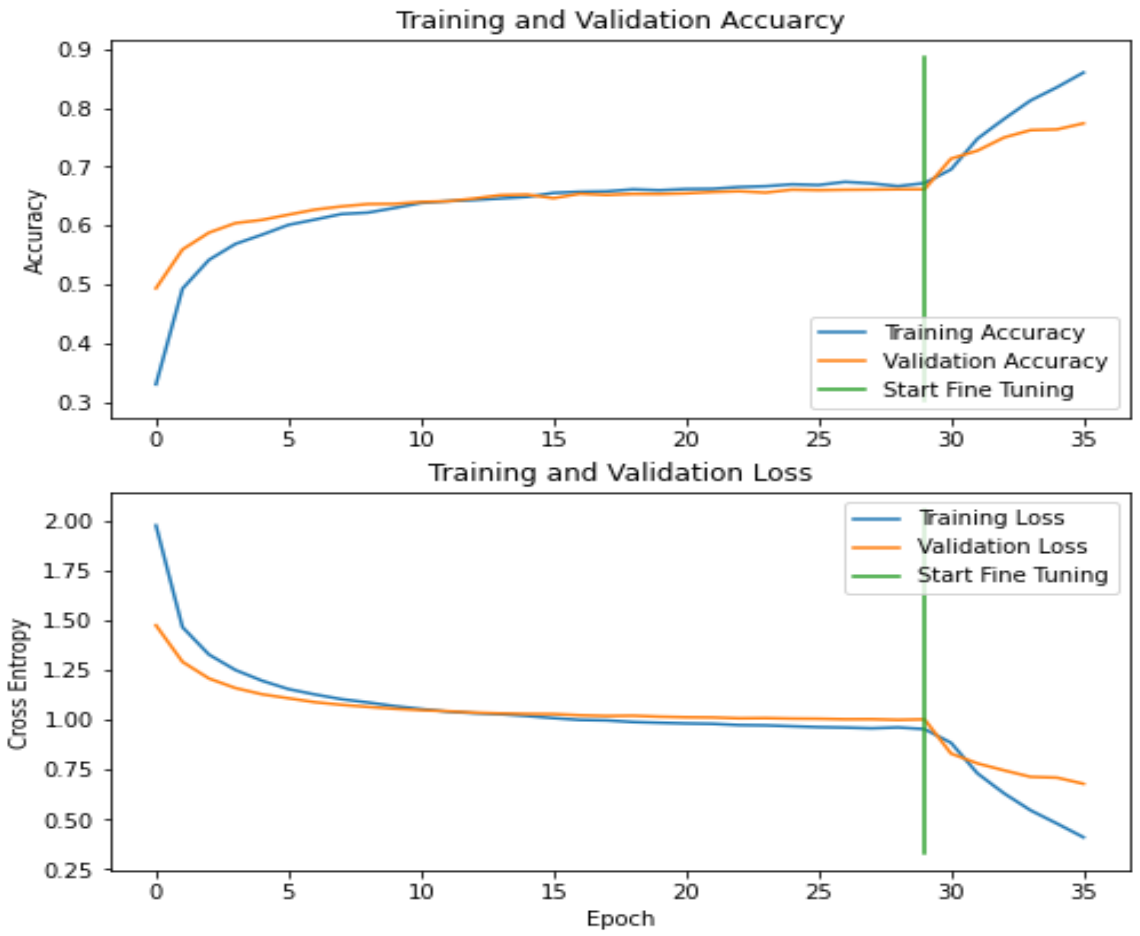


Figure 15:MobileNetV2 Training

5.1.2 InceptionResNetV2

Following diagram describes trained layers of the network and trainable parameters.

Layer (type)	Output Shape	Param #
inception_resnet_v2 (Model)	(None, 2, 2, 1536)	54336736
global_average_pooling2d (Gl	(None, 1536)	0
dense (Dense)	(None, 10)	15370

Total params: 54,352,106
Trainable params: 15,370
Non-trainable params: 54,336,736

Figure 16:InceptionResNetV2 Model Summary

InceptionResNetV2 training models were evaluated using test dataset and following results were obtained.

	30 epochs		35 epochs (Fine-Tuning)	
Model	Loss	Accuracy	Loss	Accuracy
InceptionResNetV2	1.0926	0.6180	0.8508	0.8175

Table 6:InceptionResNetV2 Training Results

InceptionResNetV2 training accuracy vs validation accuracy was recorded for each epoch to get a better idea how ML model behave while training.

Figure 17:InceptionResNetV2 Training

5.1.3 DenseNet121

Following diagram describes trained layers of the network and trainable parameters.

```
Layer (type)                 Output Shape              Param #
=================================================================
densenet121 (Model)          (None, 4, 4, 1024)        7037504
_________________________________________________________________
global_average_pooling2d (Gl (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 10)                10250
=================================================================
Total params: 7,047,754
Trainable params: 10,250
Non-trainable params: 7,037,504
```

Figure 18:DenseNet121 Model Summary

DenseNet121 training models were evaluated using test dataset and following results were obtained.

	30 epochs		35 epochs (Fine-Tuning)	
Model	Loss	Accuracy	Loss	Accuracy
DenseNet121	0.9235	0.6830	0.4867	0.8430

Table 7: DenseNet121 Training Results

DenseNet121 training accuracy vs validation accuracy was recorded for each epoch to get a better idea how ML model behave while training.

Figure 19:DenseNet121 Training

5.1.4 Tflite Models Size

ML models were converted to tflite mobile supporting format. The ML model size and Tflite model size were recorded.

Model name	Model Size	Tflite Model Size
MobileNetV2	27 MB	2.2 MB
InceptionResNetV2	638.9 MB	53.4 MB
DenseNet121	83.4 MB	7.1 MB

Table 8:ML Model sizes

5.1.5 Tflite Models Accuracy

Tflite model accuracy also observed because it is important while selecting the suitable model for mobile. The accuracy of those models was evaluated using the test dataset. Tensorflow.lite.Interpreter.Run() method was used to predict the results in the mobile application. The average time to predict the snake in the mobile application was measured because it is important when selecting the suitable model for mobile. Therefore, the average execution time to predict the snake was recorded using test dataset.

Model name	Accuracy %	Prediction Time(ms)
MobileNetV2	0.72	48.27
InceptionResNetV2	0.81	210
DenseNet121	0.83	151.50

Table 9:Tflite Models Accuracy

5.1.6 Compare Mobile Mode vs Server Mode

Selected test Images were manually tested and observed the results. All the image classes were manually tested with the server deployed ML model and mobile deployed ML tflite model. Following are the part of the results that gained three snake species Common garter snake, Timber rattlesnake and DeKay's brownsnake

Common garter snake Timber rattlesnake DeKay's brownsnake

Figure 20:Test Snake Image

Test results for Common garter snake, Timber rattlesnake and DeKay's brownsnake with MobileNetV2 tflite offline mode

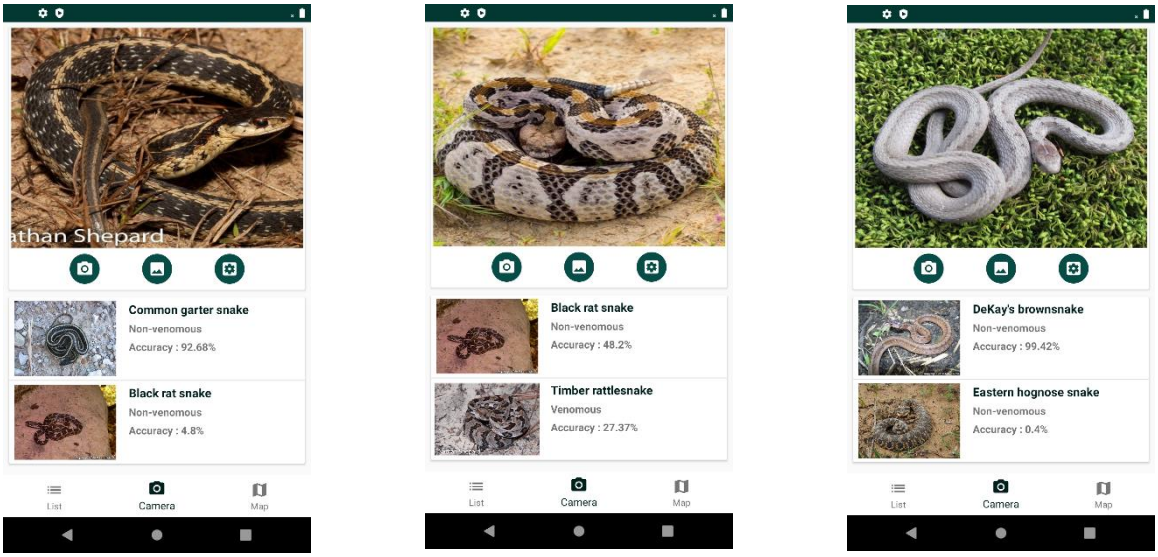


Figure 21:MobileNetV2 Offline Test Results

Test results for Common garter snake, Timber rattlesnake and DeKay's brownsnake with MobileNetV2 server online mode

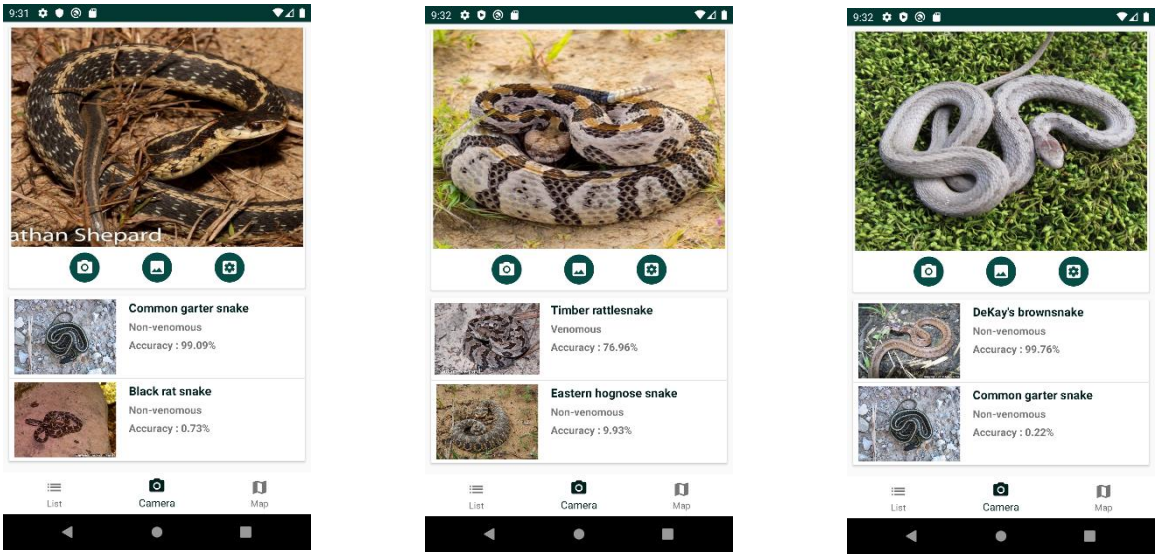


Figure 22:MobileNetV2 Online Test Results

Test results for Common garter snake, Timber rattlesnake and DeKay's brownsnake with DenseNet121 tflite offline mode

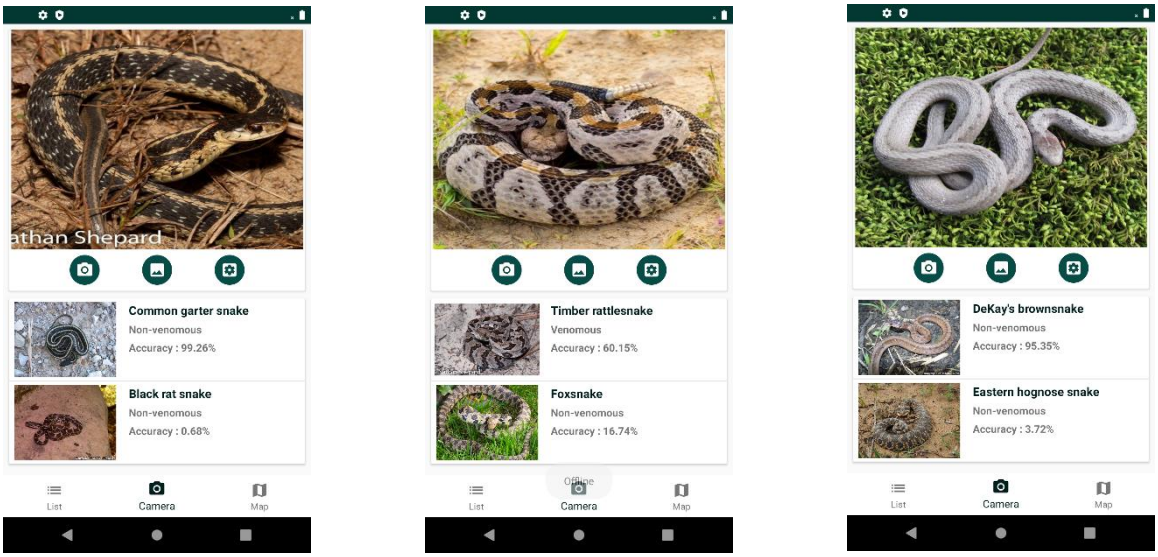


Figure 23:DenseNet121 Offline Test Results

Test results for Common garter snake, Timber rattlesnake and DeKay's brownsnake with DenseNet121 server online mode

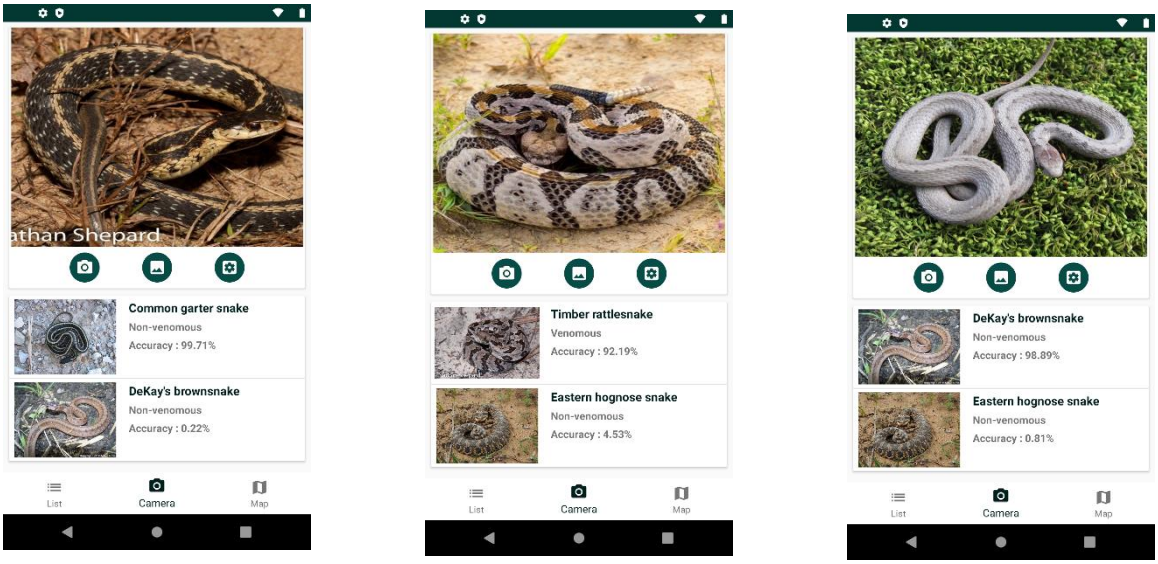


Figure 24:DenseNet121 Online Test Results

InceptionResNetV2 ML model is larger than 200MB.It was unable to deploy in web server because of server limitations. Test results are displayed for Common garter snake, Timber rattlesnake and DeKay's brownsnake with InceptionResNetV2 tflite offline mode

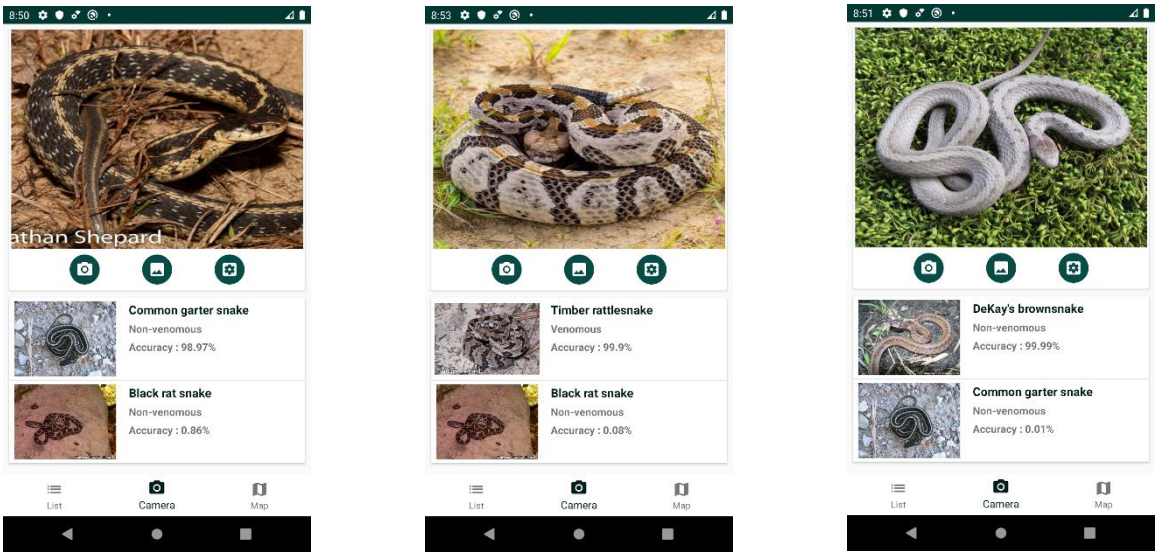


Figure 25:InceptionResNetV2 Offline Test Results

5.2 Evaluation

Three Image Classification algorithms were trained 30 epochs and fine-tuned for 5 epochs. All models were showed significant accuracy increase when fine-tuning. Before fine-tuning all models gain accuracy around 65%. After fine-tuning all models pass 80% accuracy. Finally, accuracy of the models is MobileNetV2-81%, InceptionResNetV2-82%, DensetNet121-83%. DenseNet121 has the better accuracy among three models.

5.2.1 Confusion Matrix

Confusion matrix gives more insights about the performance of the classification algorithm [28]. The matrix compares the actual values with predicted values by classification algorithm. It is generated using the test data. When there are multiple classes, calculating accuracy only misleads the performance of the classification algorithm. As an example, when algorithm gives accuracy of 80% for 10 classes. It is difficulty to identify that all classes gave 80% accuracy. It can be 9 classes have accuracy of 85% and 1 class has accuracy of 35%. It is very important to have a better accuracy for all classes in snake classification.

Sample confusion matrix for binary classification.

		Predicted	
		Negative	Positive
Actual	Negative	True Positive	False Positive
	Positive	False Negative	True Negative

True Positive (TP) - The predicted value by algorithm is positive and It is true. For example, algorithm predicted the snake image as Rattle snake and It is correct.

True Negative (TN) - The predicted value by algorithm is negative and It is true. For example, algorithm predicted the snake image is not Rattle snake and It is correct.

False Positive (FP) - The predicted value by algorithm is positive but it is actually negative. For example, algorithm predicted the snake image as Rattle snake and It is incorrect.

False Negative (FP) - The predicted value by algorithm is negative but it is actually positive. For example, algorithm predicted the snake image as not Rattle snake but it is actually Rattle snake.

Following matrices can be calculated using Confusion Matrix

Precision

Precision calculates how many of the predicted values by the model is actually positive

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Recall

Recall calculates how many of the actual positive values that predicted by the model

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

F1-Score

F1-score gives an insight of the balance between Precision and Recall

$$F1 = \frac{2 * (\mathrm{Precision} * \mathrm{Recall})}{\mathrm{Precision} + \mathrm{Recall}}$$

Confusion Matrix was generated for all models using test dataset. Classification report was also generated with Precision, Recall and, F1-Score values

In a multi-class classification, diagonal of confusion matrix shows right predictions for each class. Rough green snake has the highest Precision and Recall among all models. Eastern hangoose snake has the least Recall among all models. That can be happened because Eastern hangoose snake has wide range of color patterns.

MobilenetV2 and InceptionResNetV2 have mostly misclassified Foxsnake as Eastern hangoose snake most likely due to their color patterns seem equal.MobileNetV2 has misclassified Timber rattle snake as Diamondback rattle snake most likely due to they have same color pattern because they are belong to same family.

In DenseNet121 Black rat snake has the least precision that means other classes has misclassified them as Black rat snake. Timber rattlesnake,Foxsnake,Eastern hangoose snake and Eastern racer have been mostly misclassified.

MobileNetV2 Confusion Matrix

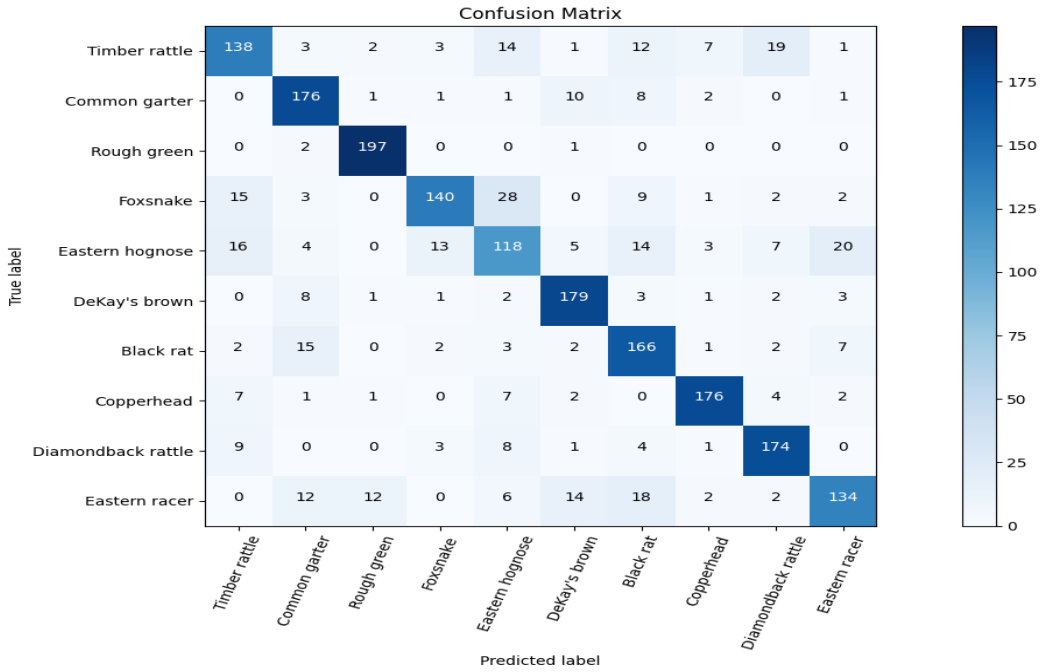


Figure 26:Confusion Matrix MobileNetV2

MobileNetV2 Classification Report

	precision	recall	f1-score	support
Timber rattlesnake	0.74	0.69	0.71	200
Common garter snake	0.79	0.88	0.83	200
Rough green snake	0.92	0.99	0.95	200
Foxsnake	0.86	0.70	0.77	200
Eastern hognose snake	0.63	0.59	0.61	200
DeKay's brown snake	0.83	0.90	0.86	200
Black rat snake	0.71	0.83	0.76	200
Copperhead	0.91	0.88	0.89	200
Diamondback rattle snake	0.82	0.87	0.84	200
Eastern racer	0.79	0.67	0.72	200
accuracy	0.80	0.80	0.80	0.80
macro avg	0.80	0.80	0.80	2000
weighted avg	0.80	0.80	0.80	2000

Table 10:MobileNetV2 Classification Report

InceptionResNetV2 Confusion Matrix

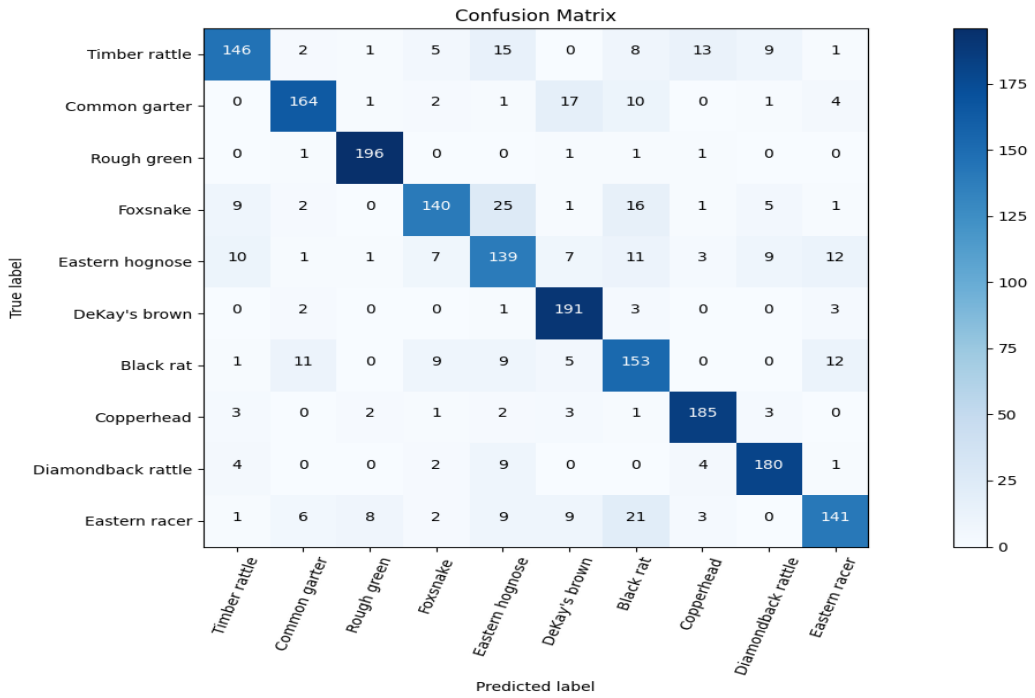


Figure 27:Confusion Matrix InceptionResNetV2

InceptionResNetV2 Classification Report

	precision	recall	f1-score	support
Timber rattlesnake	0.84	0.73	0.78	200
Common garter snake	0.87	0.82	0.84	200
Rough green snake	0.94	0.98	0.96	200
Foxsnake	0.83	0.70	0.76	200
Eastern hognose snake	0.66	0.70	0.68	200
DeKay's brown snake	0.82	0.96	0.88	200
Black rat snake	0.68	0.77	0.72	200
Copperhead	0.88	0.93	0.90	200
Diamondback rattle snake	0.87	0.90	0.88	200
Eastern racer	0.81	0.71	0.75	200
accuracy	0.82	0.82	0.82	0.82
macro avg	0.82	0.82	0.82	2000
weighted avg	0.82	0.82	0.82	2000

Table 11:Classification Report InceptionResNetV2

DenseNet121 Confusion Matrix

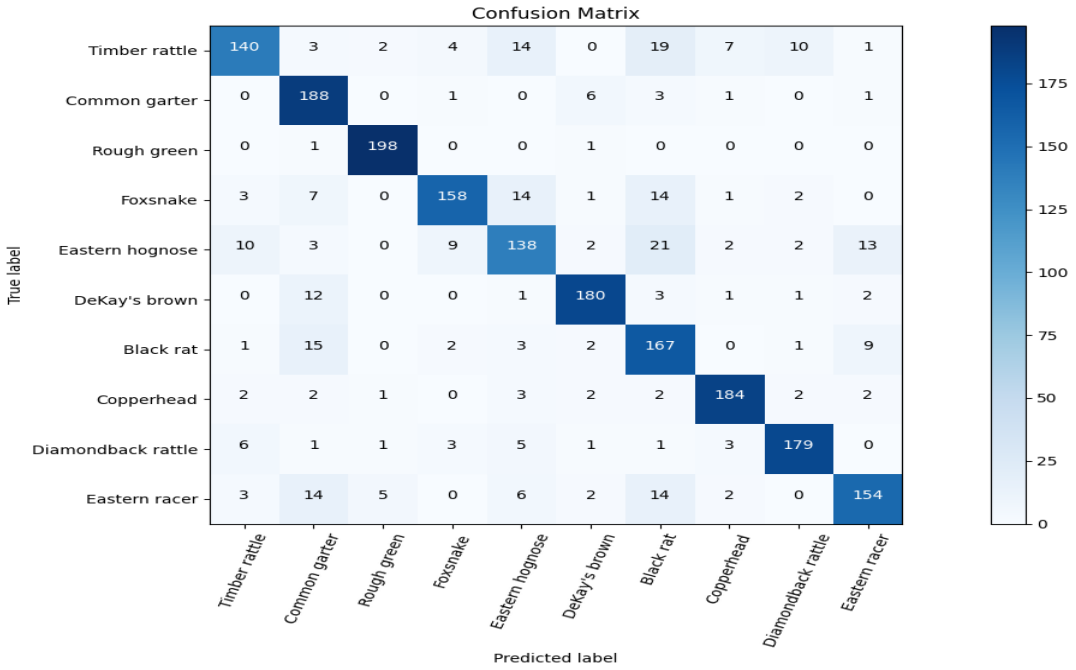


Figure 28:Confusion Matrix DenseNet121

DenseNet121 Classification Report

	precision	recall	f1-score	support
Timber rattlesnake	0.85	0.7	0.77	200
Common garter snake	0.76	0.94	0.84	200
Rough green snake	0.96	0.99	0.97	200
Foxsnake	0.89	0.79	0.84	200
Eastern hognose snake	0.75	0.69	0.72	200
DeKay's brown snake	0.91	0.9	0.91	200
Black rat snake	0.68	0.84	0.75	200
Copperhead	0.92	0.92	0.92	200
Diamondback rattle snake	0.91	0.90	0.90	200
Eastern racer	0.85	0.77	0.81	200
accuracy	0.84	0.84	0.84	0.84
macro avg	0.85	0.84	0.84	2000
weighted avg	0.85	0.84	0.84	2000

Table 12:Classification Report DenseNet121

5.2.2 Local Interpretable Model-Agnostic Explanations (LIME)

Deep Learning models can be very complex when it comes to understand how it predicts the output. If user can understand how the model predicts the output, it will increase the trust towards the model. Second, developers can find the errors and fine tune the features of the model.

LIME is a technique that highlights most relevant features of an image that affects for the prediction [29]. LIME divides the image into superpixels. They are clusters of similar features of the image. A range of perturbed images are created by randomly hiding some superpixels. These perturbed images are fed into the Deep Learning model and identifies most relevant superpixels.

Predictions of ML models were tested using LIME. Following results show what are most relevant features of Common garter snake, DeKay's brown snake and Foxsnake that affected for the prediction.

DeKay's brown snake | Foxsnake | Common garter snake

25 superpixels were generated for each image

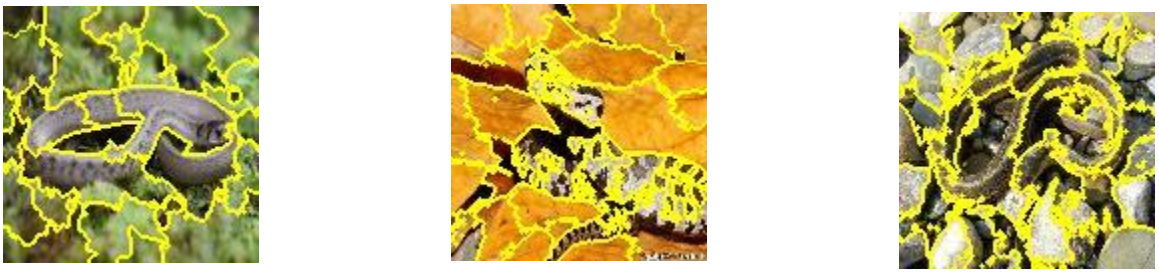


Figure 29:Images with superpixels

25 perturbed images were generated for an image and 3 of them are displayed.

Perturbed images of DeKay's brown snake

Figure 30:Dekay's brownsnake perturbed images

Perturbed images of Foxsnake

Figure 31:Foxsnake perturbed images

Perturbed images of Common garter snake

Figure 32:Common garter snake perturbed images

Identified most relevant superpixels for the prediction of Dekay's brown snake, Foxsnake and Common garter snake by MobileNetV2

Figure 33:Top features MobileNetV2

Identified most relevant superpixels for the prediction of Dekay's brown snake, Foxsnake and Common garter snake by InceptionResNetV2

Figure 34:Top features InceptionResNetV2

Identified most relevant superpixels for the prediction of Dekay's brown snake, Foxsnake and Common garter snake by DenseNet121

Figure 35:Top features DenseNet121

LIME gave a better insight about how models predict the results. DeneNet121 has identified the snake better than other models. It can be identified DeneNet121 has identified snake features accurately. Other models have been learned about the background pixels also.

After evaluating these results DenseNet121 was selected as the ML model to deploy in mobile and server. Execution time and size is also a significant requirement in mobile application but when comes to snake classification accuracy should be also considered. Therefore, DenseNet121 is the better option.

5. Conclusion and Future Work

5.1 Conclusion

The main purpose of the study was creating a stand-alone mobile application in snake classification that works offline with higher accuracy. When selecting a ML model that works in mobile for snake classification, parameters such as accuracy, performance and memory allocation should be considered. Therefore, the study analyzed three different CNNs such as: MobileNetV2, DenseNet121 and InceptionResNetV2 for selecting the most suitable one.

Transfer Learning was used to extract features from the dataset. Transfer learning was helped to extract features from the dataset using low computation power and less time. All ML models reached the accuracy level of 80% after fine-tuning. When comparing to MobileNetV2 and DenseNet121, InceptionResNetV2 has more layers and depth. But It did not achieve the highest accuracy. It was noted DenseNet121 achieved highest accuracy comparing to other models. After converting ML models to tflite models there is a significant accuracy degradation in MobileNetV2.

5.2 Future Work

This study can be carried out to gain more accuracy in snake identification by using a large image size for training and can be trained on a Deep Learning algorithm such as NasNetMobile. Furthermore, more images can be collected for each snake specie and try out the accuracy of above Convolutional Neural Networks.

This study was focused on snake species in different countries. Sri Lanka is a country which has a great snake fauna so this study can be carried out in Sri Lankan context. In Sri Lanka many snake bites are recorded annually. This application can be modified to identify snakes in Sri Lanka and that would help many people in the country.

List of References

[1] A. Kasturiratne *et al.*, "The Global Burden of Snakebite: A Literature Analysis and Modelling Based on Regional Estimates of Envenoming and Deaths," *PLoS Med*, vol. 5, no. 11, p. e218, Nov. 2008, doi: 10.1371/journal.pmed.0050218.

[2] D. A. Warrell, "Guidelines for the Management of Snake-Bites," p. 162.

[3] "Karunarathna - 2009 - Neglected snake fauna of Sri Lanka (text in Sinhal.pdf." .

[4] H. De Silva, S. Gunatilake, S. Kularatne, and K. Sellahewa, "Anti-venom for snakebite in Sri Lanka," *Ceylon Med. J.*, vol. 47, no. 2, p. 43, Aug. 2011, doi: 10.4038/cmj.v47i2.3449.

[5] I. Bolon *et al.*, "Identifying the snake: First scoping review on practices of communities and healthcare providers confronted with snakebite across the world," *PLoS ONE*, vol. 15, no. 3, p. e0229989, Mar. 2020, doi: 10.1371/journal.pone.0229989.

[6] A. James, "Snake classification from images," PeerJ Preprints, preprint, Mar. 2017. doi: 10.7287/peerj.preprints.2867v1.

[7] P. M. Mainkar, S. Ghorpade, and M. Adawadkar, "Plant Leaf Disease Detection and Classification Using Image Processing Techniques," *International Journal of Innovative and Emerging Research in Engineering*, vol. 2, no. 4, p. 6, 2015.

[8] M. N. Sheth, S. L. Nalbalwar, and A. B. Nandgaonkar, "IDENTIFICATION OF SNAKE TYPE FROM IMAGE," p. 4.

[9] A. Amir, N. A. H. Zahri, N. Yaakob, and R. B. Ahmad, “Image Classification for Snake Species Using Machine Learning Techniques,” in *Computational Intelligence in Information Systems*, vol. 532, S. Phon-Amnuaisuk, T.-W. Au, and S. Omar, Eds. Cham: Springer International Publishing, 2017, pp. 52–59.

[10] A. Hernández-Serna and L. F. Jiménez-Segura, “Automatic identification of species with neural networks,” *PeerJ*, vol. 2, p. e563, Nov. 2014, doi: 10.7717/peerj.563.

[11] A. Patel, L. Cheung, N. Khatod, I. Matijosaitiene, A. Arteaga, and J. W. Gilkey, “Revealing the Unknown: Real-Time Recognition of Galápagos Snake Species Using Deep Learning,” *Animals*, vol. 10, no. 5, p. 806, May 2020, doi: 10.3390/ani10050806.

[12] A. P. James, B. Mathews, S. Sugathan, and D. K. Raveendran, “Discriminative histogram taxonomy features for snake species identification,” *Hum. Cent. Comput. Inf. Sci.*, vol. 4, no. 1, p. 3, Dec. 2014, doi: 10.1186/s13673-014-0003-0.

[13] I. Gogul and V. S. Kumar, “Flower species recognition system using convolution neural networks and transfer learning,” in *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, Chennai, India, Mar. 2017, pp. 1–6, doi: 10.1109/ICSCN.2017.8085675.

[14] S. P. Patil and R. S. Zambre, “Classification of Cotton Leaf Spot Disease Using Support Vector Machine,” vol. 4, no. 5, p. 6, 2014.

[15] N. Sharma, V. Jain, and A. Mishra, “An Analysis Of Convolutional Neural Networks For Image Classification,” *Procedia Computer Science*, vol. 132, pp. 377–384, 2018, doi: 10.1016/j.procs.2018.05.198.

[16] S. Karamizadeh, S. M. Abdullah, A. A. Manaf, M. Zamani, and A. Hooman, “An Overview of Principal Component Analysis,” *JSIP*, vol. 04, no. 03, pp. 173–175, 2013, doi: 10.4236/jsip.2013.43B031.

[17] “Review_on_Techniques_for_Plant_Leaf_Classification.pdf.” .

[18] “Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques,” *IJSR*, vol. 5, no. 1, pp. 1842–1845, Jan. 2016, doi: 10.21275/v5i1.NOV153131.

[19] K. Hoang, “Image Classification with Fashion-MNIST and CIFAR-10,” p. 5.

[20] F. Sultana, A. Sufian, and P. Dutta, “Advancements in Image Classification using Convolutional Neural Network,” *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pp. 122–129, Nov. 2018, doi: 10.1109/ICRCICN.2018.8718718.

[21] K. Nguyen, C. Fookes, A. Ross, and S. Sridharan, “Iris Recognition With Off-the-Shelf CNN Features: A Deep Learning Perspective,” *IEEE Access*, vol. 6, pp. 18848–18855, 2018, doi: 10.1109/ACCESS.2017.2784352.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, Jun. 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.

[23] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *arXiv:1512.03385 [cs]*, Dec. 2015, Accessed: May 16, 2020. [Online]. Available: http://arxiv.org/abs/1512.03385.

[24] L. D. Nguyen, D. Lin, Z. Lin, and J. Cao, “Deep CNNs for microscopic image classification by exploiting transfer learning and feature concatenation,” in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, Florence, 2018, pp. 1–5, doi: 10.1109/ISCAS.2018.8351550.

[25] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” *arXiv:1608.06993 [cs]*, Jan. 2018, Accessed: Jun. 21, 2020. [Online]. Available: http://arxiv.org/abs/1608.06993.

[26] M. Hussain, J. J. Bird, and D. R. Faria, “A Study on CNN Transfer Learning for Image Classification,” in *Advances in Computational Intelligence Systems*, vol. 840, A. Lotfi, H. Bouchachia, A. Gegov, C. Langensiepen, and M. McGinnity, Eds. Cham: Springer International Publishing, 2019, pp. 191–202.

[27] M. Buda, A. Maki, and M. A. Mazurowski, “A systematic study of the class imbalance problem in convolutional neural networks,” *Neural Networks*, vol. 106, pp. 249–259, Oct. 2018, doi: 10.1016/j.neunet.2018.07.011.

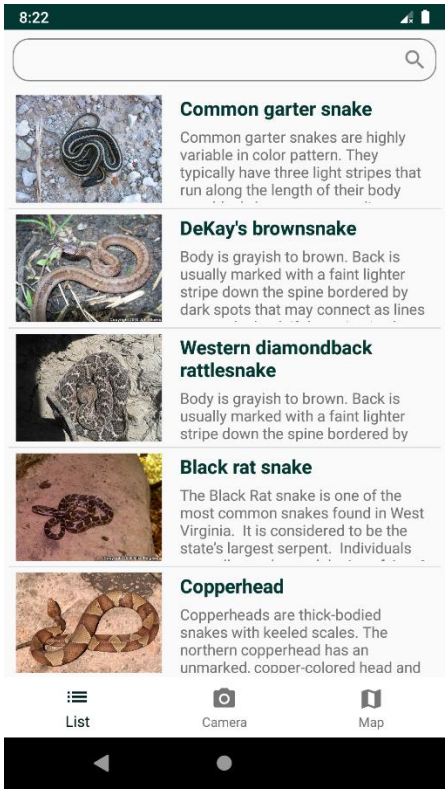
[28] A. Patel, L. Cheung, N. Khatod, I. Matijosaitiene, A. Arteaga, and J. W. Gilkey, “Revealing the Unknown: Real-Time Recognition of Galápagos Snake Species Using Deep Learning,” *Animals*, vol. 10, no. 5, p. 806, May 2020, doi: 10.3390/ani10050806.

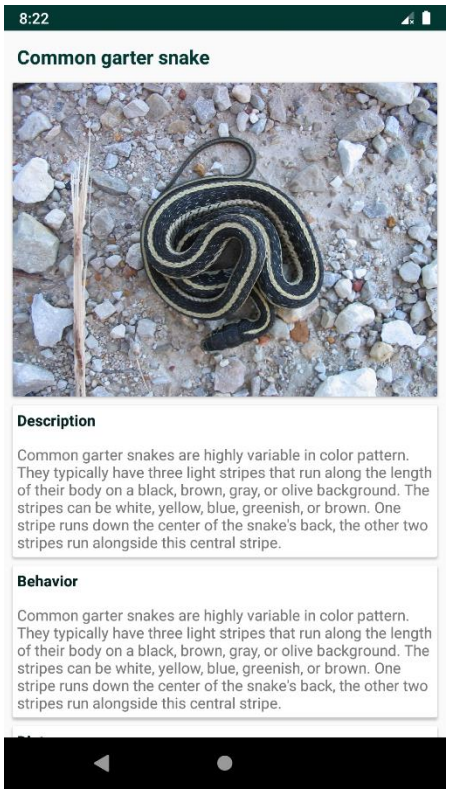
[29] I. Palatnik de Sousa, M. Maria Bernardes Rebuzzi Vellasco, and E. Costa da Silva, “Local Interpretable Model-Agnostic Explanations for Classification of Lymph Node Metastases,” *Sensors*, vol. 19, no. 13, p. 2969, Jul. 2019, doi: 10.3390/s19132969.

Appendix

User Manual to use Mobile application

Following screens are the Home page and the Snake details page of the application. In the home page, we have search option top of the page, using that we can easily find a snake. We can see more details of a snake by going to snake details page by click a snake card in the home page



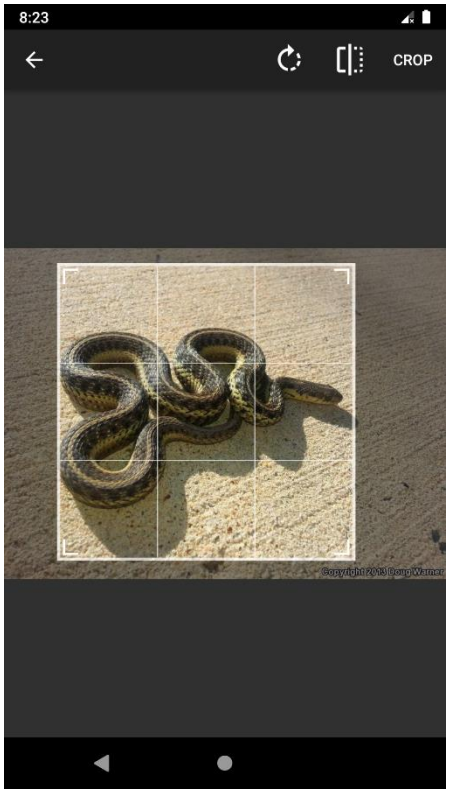


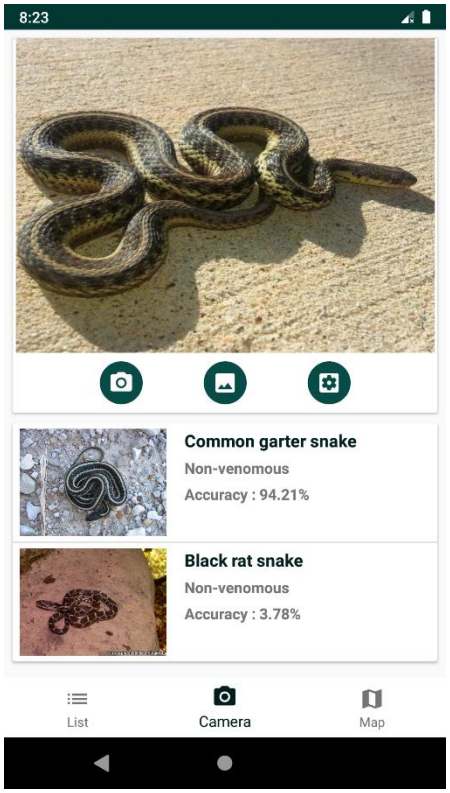
Using bottom navigation bar, we can navigate between main screens of the mobile application. There are three buttons in the bottom navigation bar List, Camera and Map

How to Identify snakes using mobile application

Go to Camera view clicking by camera icon in bottom navigation bar. There are three buttons in this view camera button, gallery button and predict button. Using both of camera button and galley button we can capture a snake image. After capturing a snake image, we can use crop option to reduce the noise. After cropping an image use predict button to generate results.

Following two screens are Crop Image screen and Camera view





Using the map icon in the bottom navigation bar we can move to Map View. In this view nearby hospitals and rich biodiversity areas are displayed