**SPECMAC: THE SPECIE MACHINE**

*Dissertation report submitted*

*in partial fulfillment of the requirements for the degree of*

**Master of Computer Applications (Integrated)**

By

**Bhawna Singla**

**(AU124060202001P)**

**Sonali Chawla**

**(AU124060202002P)**

**Nisha**

**(AU124060202003P)**

###### Under the Supervision of

### Mrs. Shikha Gupta



**2012-2017**

**ANSAL UNIVERSITY**

**GURGAON, HARAYANA**

**Date: 31 May 2017**

**CERTIFICATE**

This is to certify that the project titled **SPECMAC: THE SPECIE MACHINE** submitted by **Bhawna Singla**, **Sonali Chawla** and **Nisha** in partial fulfillment for the award of degree of Master of Computer Applications of Ansal University, Gurgaon has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree.

Signature in full of Supervisor:

Name in Capital block letters: MRS. SHIKHA GUPTA

Designation: Assistant Professor (SET)

**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Acknowledgments | | | **i** |
|  | Dissertation APPROVAL | | | **ii** |
|  | DECLARATION | | | **iii** |
|  | Abstract | | | **iv** |
|  | LIST OF TABLES | | | **v** |
|  | [LIST](#_Toc157209473) OF FIGURES | | | **vi** |
|  |  | | |  |
| **1.** | **Introduction** | | | **1** |
|  | 1.1. | Project Description | | 2 |
|  | 1.2. | Objective and Scope of Work | | 2 |
| **2.** | **Literature Review** | | | **4** |
|  | 2.1. | Machine Learning | | 5 |
|  | 2.2. | Deep Learning | | 10 |
|  | 2.3. | Neural Network, Perceptron and Backpropogation | | 12 |
|  | 2.4. | Convolutional Neural Network | | 22 |
|  | 2.5. | Transfer Learning | | 24 |
|  | 2.6. | Inception of GoogLeNet | | 26 |
| **3.** | **Project Methodology** | | | **35** |
|  | 3.1. | Methodology Used | | 36 |
|  | 3.2. | Technology Used | | 37 |
|  |  | 3.2.1. | Android Studio | 37 |
|  |  | 3.2.2. | Python | 37 |
|  |  | 3.2.3. | TensorFlow | 38 |
|  |  | 3.2.4. | Docker Toolbox | 38 |
|  |  | 3.2.5. | Anaconda | 38 |
|  | 3.3. | Feasibility Study | | 39 |
|  |  | 3.3.1. | Economical Feasibility | 39 |
|  |  | 3.3.2. | Technical Feasibility | 39 |
|  |  | 3.3.3. | Legal Feasibility | 39 |
| **4.** | **Theoretical Framework** | | | **40** |
|  | 4.1. | Project Structure | | 41 |
|  | 4.2. | Flow Chart | | 42 |
| **5.** | **Experimental Procedure** | | | **43** |
|  | 5.1. | Collecting Specie Images | | 44 |
|  | 5.2. | Working with Tensorflow | | 44 |
|  | 5.3. | Working with Android Studio | | 50 |
| **6.** | **Snapshot of Application** | | | **103** |
|  | 6.1. | App Icon | | 104 |
|  | 6.2. | Scanning Specie | | 104 |
|  | 6.3. | Detecting Specie | | 105 |
|  | 6.4. | Specie Information | | 105 |
| **7.** | **Conclusion** | | | **106** |
|  | | | | |
| Suggestions for Further Study or Further Improvement | | | | 108 |
| References | | | | 110 |
|  | | | |  |
| [LIST OF PUBLICATIONS AND PRESENTATIONS](#_Toc157209553) | | | | 112 |
| Bibliography | | | | 112 |

**ACKNOWLEDGMENTS**

We take this opportunity to express our profound gratitude and deep regards to our guide Ms. Shikha Gupta for her exemplary guidance, monitoring and constant encouragement throughout the development of the project. Her aspiring guidance, invaluably constructive criticism and friendly advice during project development are appreciated.The blessing, help and guidance given by her time to time will carry us a long way in the journey of life on which we were about to embark. We are also sincerely grateful to our family and friends for sharing their truthful and illuminating views on a number of issues related to the project.

**Dissertation Approval**

This project report entitled “***SPECMAC: THE SPECIE MACHINE*”** by **SONALI CHAWLA, BHAWNA SINGLA and NISHA** is approved for the degree of Master of Computer Applications (Integrated). MCA (Integrated) is an integrated five years prestigious program. BCA Course is a three year undergraduate program of semester pattern where students are exposed to various areas of computer applications. The course provides sound practical skills which is required for the latest development of industries. After three years students can be admitted for MCA which is conducted for next two years.The main objective of the Dual Degree programme is to prepare students to face the challenges in software industry and academia by providing an excellent teaching, practices and research oriented environment.

Examiners

……………………………..

Supervisor(s)

Mrs. Shikha Gupta

Date: 31 May 2017

Place: Ansal University, Gurgaon

**Declaration**

I declare that this report represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Bhawla Singla

(AU124060202001P)

Sonali Chawla

(AU124060202002P)

Nisha

(AU124060202003P)

Date: 31 May 2017

**ABSTRACT**

Knowledge of species distribution is critical to ecological management and conservation biology. Effective management requires the detection of populations, which can sometimes be at low densities and is usually based on visual detection and counting. Image classification plays an important role in many applicable fields in our life, such as image analysis, remote sensing, and pattern recognition. It can be defined as the process of sorting all the pixels in an image into a finite number of individual classes. There are many types of techniques which can be used to classify and recognize different types of objects in images. For conventional statistical approaches for land cover classification, they only use the gray values of the image to detect and classify objects. They lead to misclassification due to the strictly convex boundaries. The textural features can be included for better classification but they are inconvenient for the conventional methods. On the other hand, artificial neural networks (ANNs) can handle non-convex decisions. The uses of textural features help to resolve misclassification. However, the image processing and the process of translating an image into a statistical distribution of low-level features is not an easy task. These tasks are complicated since the acquired image data often noisy, and target objects are influenced by lighting, intensity or illumination. In the case of flower classification, image processing is a crucial step for computer-aided plant species identification. Object detection and recognition based on image processing is vastly concentrating field in research. The project, SPECMAC: the Specie Machine aims at developing an android application build over deep learning platform using TensorFlow library. The applicationwould scan the specie with camera of a smartphone and give details and information about that scanned specie. The app used inception model made by Google and retrain it with a huge set of images collected over the internet.

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table Caption** | **Page No.** |
| Table 5.1 | Data collected | 44 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure Caption** | **Page No.** |
| Figure 2.1 | Unsupervised Learning | 8 |
| Figure 2.2 | Supervised Learning | 9 |
| Figure 2.3 | Reinforcement Learning | 10 |
| Figure 2.4 | Artificial Neural Network | 13 |
| Figure 2.5 | Single Layer Perceptron | 18 |
| Figure 2.6 | Multilayer Perceptron | 18 |
| Figure 2.7 | Transfer Learning | 25 |
| Figure 2.8 | Three ways to improve learning | 25 |
| Figure 4.1 | Project Structure | 41 |
| Figure 4.2 | Flow Chart | 42 |
| Figure 5.1 | 112.jpg | 50 |
| Figure 6.1 | App Icon Screen | 104 |
| Figure 6.2 | Scanning Specie Screen | 104 |
| Figure 6.3 | Detecting Specie Screen | 105 |
| Figure 6.4 | Specie Information Display Screen | 105 |

**1. INTRODUCTION**

* 1. **Project Description**

The project ***SPECMAC: the Specie Machine*** is a project made for android users to provide information to the android users about the specie scanned from their smart phone.

In today’s world, when technology has become a vital part of everyone’s daily life, there is a need to bridge the gap between technology and knowledge. To fill the gap between these two most important aspects, Specmac uses technology and provides basic knowledge and facts about the various species. The idea is to develop a framework which would allow everyone to get instant information about specie in front of their smart phone camera.

The framework ***Specmac*** is an inspiration from cartoon series Pokémon and will be designed to provide some basic information and facts about the living specie using an android smart phone. In this, the android user has to scan the specie from his/her smart phone. The algorithm developed will identify all the features of the specie and match it with the training set to identify the specie. The program will then call out all the details about the identified specie.

Machine Learning, a process to make the machine learn all the working is used to make the framework. Around 500 pictures per specie will be collected and used as the training set for training the algorithm. The framework will be prepared in Android Studio to make an app for android phones. For making an algorithm to identify the specie and do all the statistical work, Python is used.

* 1. **Objective and Scope of Work**

The main objective of the making ***Specmac*** is to build an android application to be used in zoology area. The people in zoological research area can use the application while researching on wildlife to know their behavior, culture, etc. Some of the specific objectives are observed as below:

* + **Providing aninteractive learning experience**: The framework is designed in a way that it will provide interactive experience to all the users. The interface will be user-friendly and easy to operate.
  + **Bridge the gap between Technology and Knowledge**: The framework use the smart phone to scan the specie and provide the knowledge about it. Thus, it fills the gap between two most important aspects of life, i.e., Technology and Knowledge.
  + **Transmission of knowledge with fun**: Along with the knowledge, the framework will also be fun to operate. The user can challenge their knowledge about various species on the earth.
  + **E-learning in Schools**: According to science, people can learn more from visuals and audios. The app can also be used in schools as a mode of e-learning. This will help students to learn more and in a better way.

**2. LITERATURE REVIEW**

**2.1. Machine Learning**

Programs can be automatically learned from data by machine learning systems. Instead of manually constructing them, this is an attractive alternative. Over the last few decades, the use of machine learning has been widely spread in every part of computer science and beyond. Machine learning means algorithms are designed which allows a computer to learn. It is not necessary that learning involves consciousness but learning is a kind of medium of detecting statistical regularities and other patterns in data. Applications like Web search, Ad Placements, Fraud Detection, Spam Filters, Credit Scoring, Drug Design, Recommender Systems, and Stock Trading etc use machine learning. There are various types of machine learning, this paper emphasizes on most widely used Supervised Learning, Unsupervised Learning and Reinforcement Learning[1].

## 2.1.1. Supervised Learning

The learning here is performed with the help of a teacher. Each input vector in the neural network needs a corresponding target vector, which indicates the desired output. At the time of training, the input vector is offered to the network, the result is an output vector. The output vector is the actual output vector. Then, comparison of actual output vector and the desired output vector is done. If any kind of difference is found, then the network generates an error signal. Adjustment of weights is done through this error signal until desired output and the actual output matches. In this kind of learning, error minimization is done by a supervisor or a teacher.

The most common form of machine learning, deep or not, is supervised learning. Imagine that we want to build a system that can classify images as containing, say, a house, a car, a person or a pet. We first collect a large data set of images of houses, cars, people and pets, each labeled with its category. During training, the machine is shown an image and produces an output in the form of a vector of scores, one for each category. We want the desired category to have the highest score of all categories, but this is unlikely to happen before training. We compute an objective function that measures the error (or distance) between the output scores and the desired pattern of scores. The machine then modifies its internal adjustable parameters to reduce this error. These adjustable parameters, often called weights, are real numbers that can be seen as ‘knobs’ that define the input–output function of the machine. In a typical deep-learning system, there may be hundreds of millions of these adjustable weights, and hundreds of millions of labeled examples with which to train the machine.

To properly adjust the weight vector, the learning algorithm computes a gradient vector that, for each weight, indicates by what amount the error would increase or decrease if the weight were increased by a tiny amount. The weight vector is then adjusted in the opposite direction to the gradient vector. The objective function, averaged over all the training examples, can be seen as a kind of hilly landscape in the high-dimensional space of weight values. The negative gradient vector indicates the direction of steepest descent in this landscape, taking it closer to a minimum, where the output error is low on average.

In practice, most practitioners use a procedure called stochastic gradient descent (SGD). This consists of showing the input vector for a few examples, computing the outputs and the errors, computing the average gradient for those examples, and adjusting the weights accordingly. The process is repeated for many small sets of examples from the training set until the average of the objective function stops decreasing. It is called stochastic because each small set of examples gives a noisy estimate of the average gradient over all examples. This simple procedure usually finds a good set of weights surprisingly quickly when compared with far more elaborate optimization techniques[2]. After training, the performance of the system is measured on a different set of examples called a test set. This serves to test the generalization ability of the machine — its ability to produce sensible answers on new inputs that it has never seen during training.

Many of the current practical applications of machine learning use linear classifiers on top of hand-engineered features. A two-class linear classifier computes a weighted sum of the feature vector components. If the weighted sum is above a threshold, the input is classified as belonging to a particular category.

Since the 1960s we have known that linear classifiers can only carve their input space into very simple regions, namely half-spaces separated by a hyperplane[3]. But problems such as image and speech recognition require the input–output function to be insensitive to irrelevant variations of the input, such as variations in position, orientation or illumination of an object, or variations in the pitch or accent of speech, while being very sensitive to particular minute variations (for example, the difference between a white wolf and a breed of wolf-like white dog called a Samoyed). At the pixel level, images of two Samoyeds in different poses and in different environments may be very different from each other, whereas two images of a Samoyed and a wolf in the same position and on similar backgrounds may be very similar to each other. A linear classifier or any other ‘shallow’ classifier operating on raw pixels could not possibly distinguish the latter two, while putting the former two in the same category. This is why shallow classifiers require a good feature extractor that solves the selectivity–invariance dilemma — one that produces representations that are selective to the aspects of the image that are important for discrimination, but that are invariant to irrelevant aspects such as the pose of the animal. To make classifiers more powerful, one can use generic non-linear features, as with kernel methods[4], but generic features such as those arising with the Gaussian kernel do not allow the learner to generalize well far from the training examples[5]. The conventional option is to hand design good feature extractors, which requires a considerable amount of engineering skill and domain expertise. But this can all be avoided if good features can be learned automatically using a general-purpose learning procedure. This is the key advantage of deep learning.

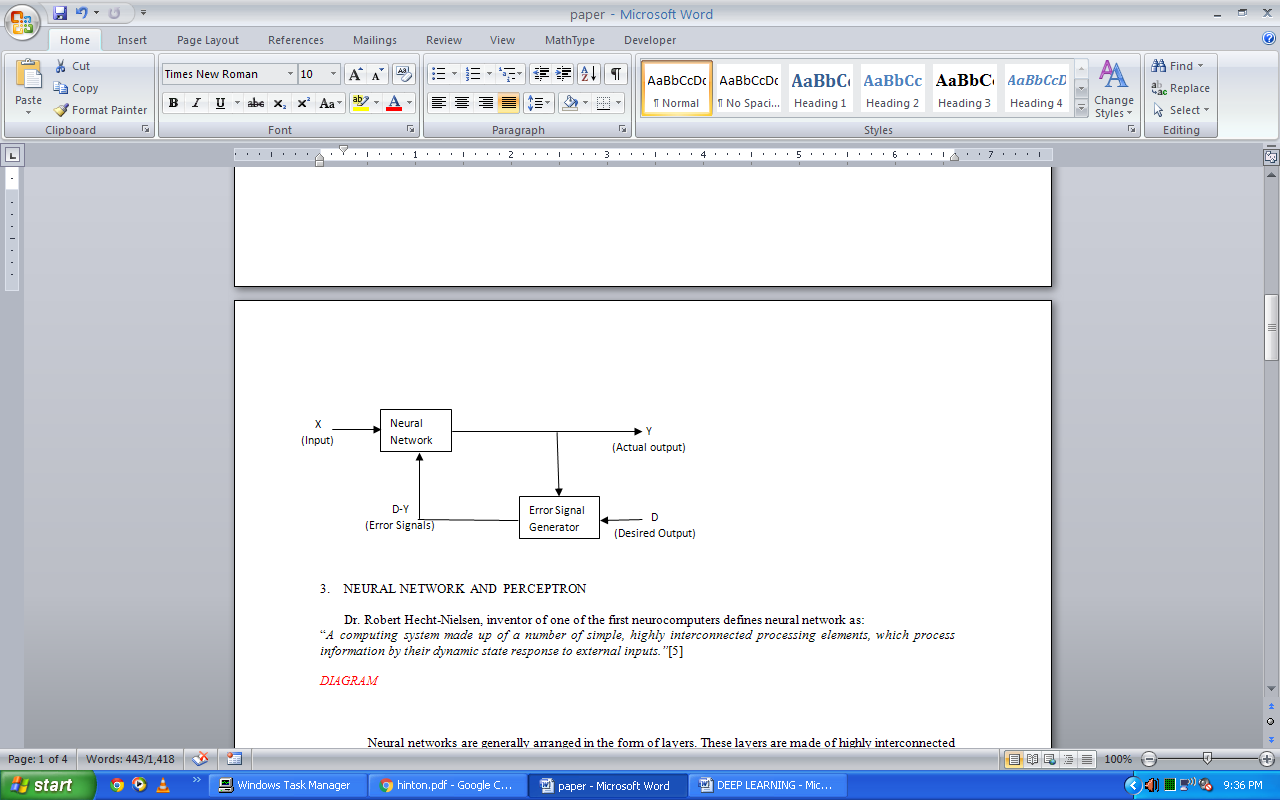


Figure 2.1 Supervised Learning

## A deep-learning architecture is a multilayer stack of simple modules, all (or most) of which are subject to learning, and many of which compute non-linear input–output mappings. Each module in the stack transforms its input to increase both the selectivity and the invariance of the representation. With multiple non-linear layers, say a depth of 5 to 20, a system can implement extremely intricate functions of its inputs that are simultaneously sensitive to minute details — distinguishing Samoyeds from white wolves — and insensitive to large irrelevant variations such as the background, pose, lighting and surrounding objects.

## 2.1.2. Unsupervised Learning

This type of learning is performed without the help of a teacher. It is an independent process and no supervision is done by a teacher. Similar types of input vectors are clustered together without the requirement of training data to specify how each group’s member looks. Input patterns are presented to the network and it arranges these patterns to form clusters. When the network receives a new input pattern, an output response is given by the neural network which indicates to which class the input pattern belongs. Suppose for an input, no pattern class is found, then it leads to generation of a new class.

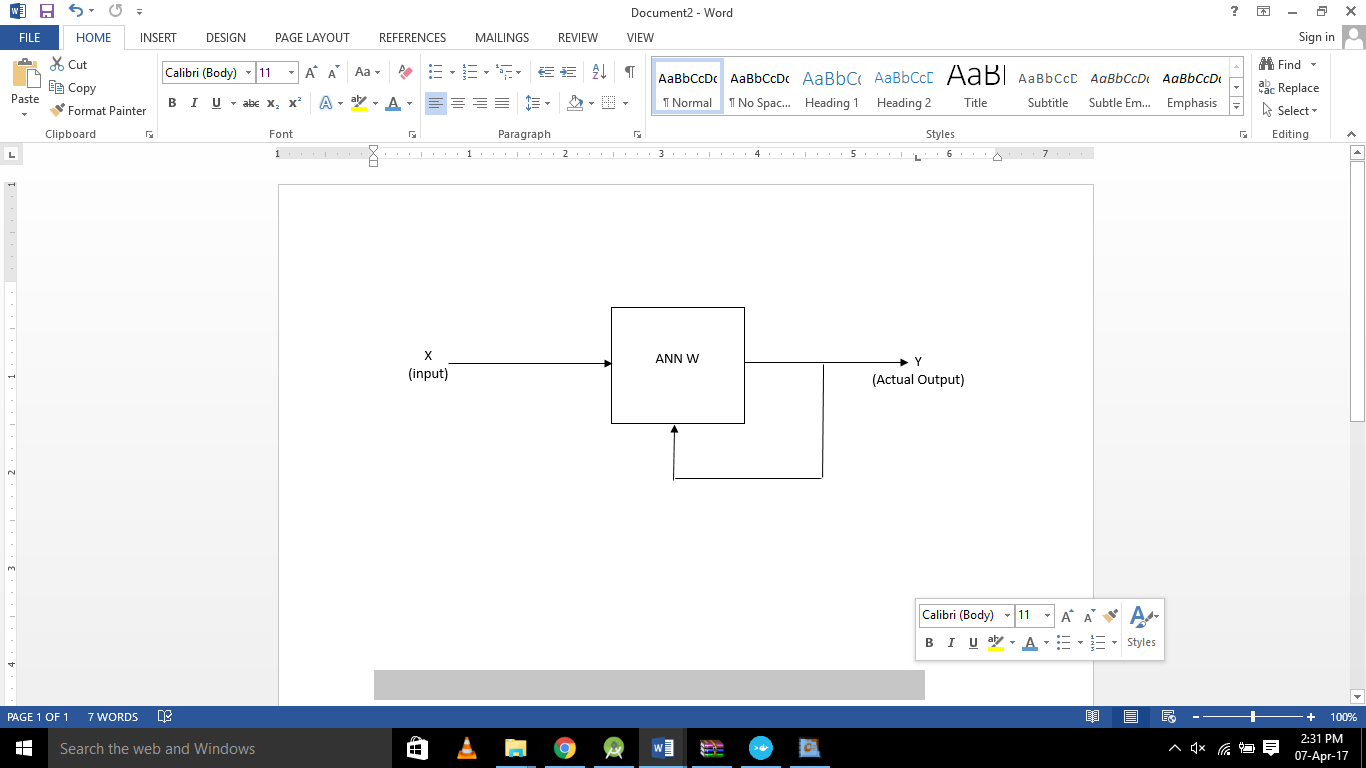


Figure 2.2 Unsupervised Learning

## 2.1.3. Reinforcement Learning

It is quite similar to supervised learning. In supervised learning, for each input pattern, the right target output values are already known. In Reinforcement learning, less amount of information is available. Only the critic info is available and not the exactly the correct information. Learning which is based on this critic is known as Reinforcement Learning and the feedback sent back is called as Reinforcement signal. Since the network obtains some feedback from its environment, therefore reinforcement learning is called as a form of supervised learning. The feedback received here is not instructive, it is only evaluative.

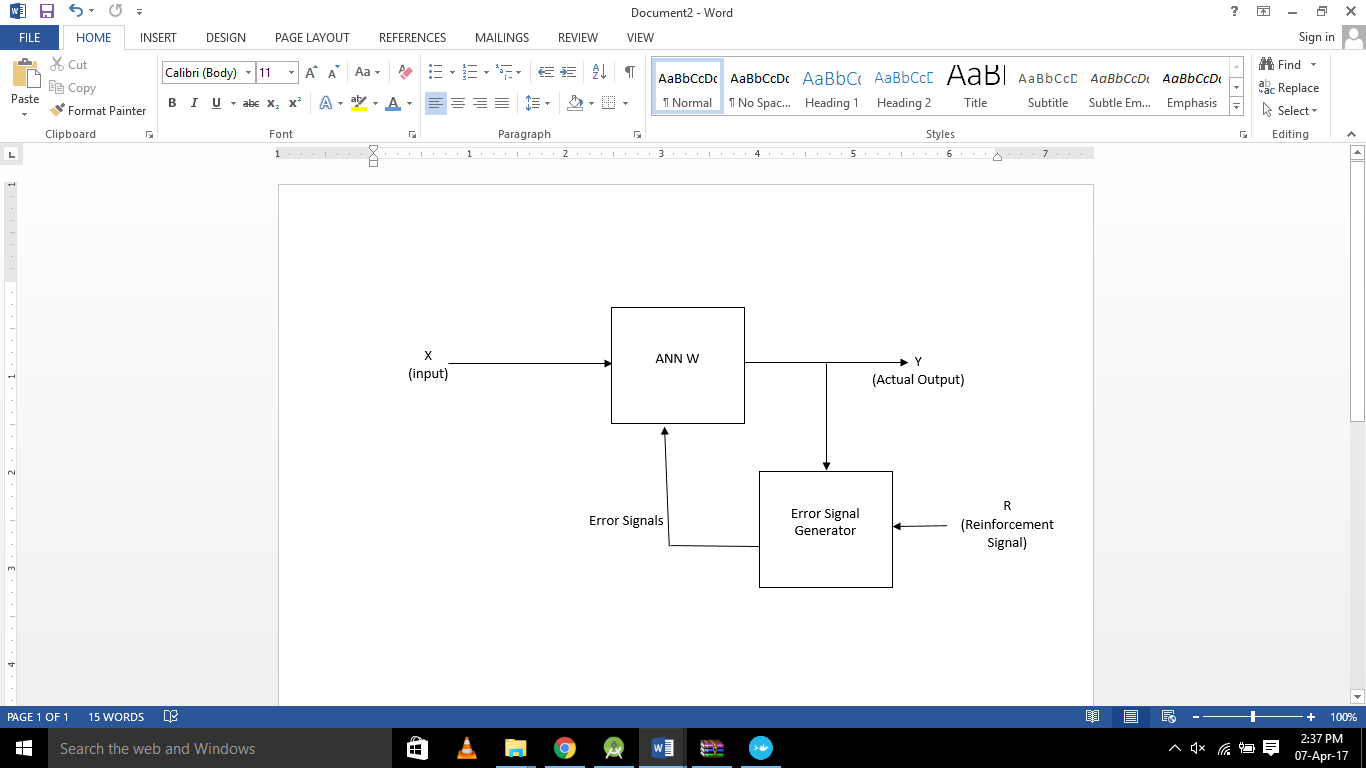


Figure 2.3 Reinforcement Learning

**2.2. Deep Learning**

Intelligent machines observe learning as their parameter. Deep understanding encourages in proceeding with the decisions in a more optimized form and also helps to perform in most productive method. As an alternative to bulky machines with additional programming, now various other algorithms are coming which will assist the machine for better understanding of the virtual environment and based upon it decisions will be taken. This can further help to make the machine independent and reduce the programming concepts. For different types of machines and their decisions, different types of algorithms are being introduced.

Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smart phones. Machine-learning systems are used to identify objects in images, transcribe speech into text, match news items, posts or products with users’ interests, and select relevant results of search. Increasingly, these applications make use of a class of techniques called deep learning.

Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, constructing a pattern-recognition or machine-learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns in the input.

Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification. Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transforms the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned. For classification tasks, higher layers of representation amplify aspects of the input that are important for discrimination and suppress irrelevant variations. An image, for example, comes in the form of an array of pixel values, and the learned features in the first layer of representation typically represent the presence or absence of edges at particular orientations and locations in the image. The second layer typically detects motifs by spotting particular arrangements of edges, regardless of small variations in the edge positions. The third layer may assemble motifs into larger combinations that correspond to parts of familiar objects, and subsequent layers would detect objects as combinations of these parts. The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure.

Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years. It has turned out to be very good at discovering intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. In addition to beating records in image recognition[6][7][8][9] and speech recognition [10][11][12], it has beaten other machine-learning techniques at predicting the activity of potential drug molecules[13], analyzing particle accelerator data[14][15], reconstructing brain circuits[16], and predicting the effects of mutations in non-coding DNA on gene expression and disease[17][18]. Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding[19], particularly topic classification, sentiment analysis, question answering[20] and language translation [21][22].

We think that deep learning will have many more successes in the near future because it requires very little engineering by hand, so it can easily take advantage of increases in the amount of available computation and data. New learning algorithms and architectures that are currently being developed for deep neural networks will only accelerate this progress.

**2.3. Neural Network, Perceptron and Backpropogation**

Dr. Robert Hecht-Nielsen, inventor of one of the first neuro-computers defines neural network as:

“A computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.”[23]

Neural networks are generally arranged in the form of layers. These layers are made of highly interconnected “nodes” which further contains an “activation function”. The ‘input vector’ is the pattern represented to the neural network, it further communicates to one more ‘hidden layers’ in the network. Actual processing of weights is done and hidden layers are linked to an ‘output layer’ as shown in the diagram[24]. Each link or connection is associated with weights containing info about input signal. Some of the basic applications of neural network are Speech Recognition, Face Identification, Data Mining, Robotics, Medical Applications, Medical Diagnosis, Pattern Recognition, Photos and Fingerprint Recognition, Handwriting Recognition, Text Translation, Computer Vision and many more.

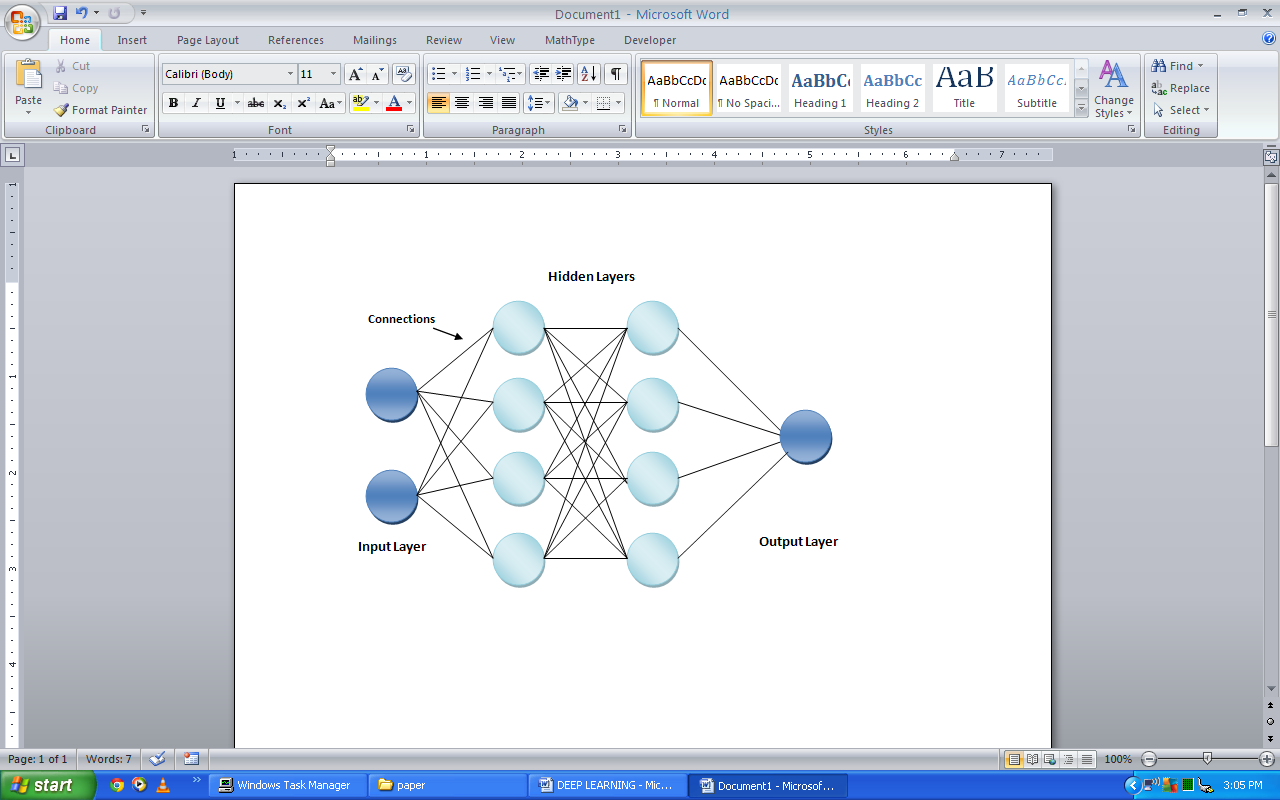


Figure 2.4 Artificial Neural Network

Artificial Neural Networks have been created to try to solve this problem. Zhang (2000) provided an overview of existing work in Artificial Neural Networks (ANNs). A multi-layer neural network consists of large number of units (neurons) joined together in a pattern of connections (Fig. 2.4). Units in a net are usually segregated into three classes: input units, which receive information to be processed; output units, where the results of the processing are found; and units in between known as hidden units. Feed-forward ANNs (Fig. 2.4) allow signals to travel one way only, from input to output. Generally, properly determining the size of the hidden layer is a problem, because an underestimate of the number of neurons can lead to poor approximation and generalization capabilities, while excessive nodes can result in over fitting and eventually make the search for the global optimum more difficult. An excellent argument regarding this topic can be found in [25]. [26]also studied the minimum amount of neurons and the number of instances necessary to program a given task into feed-forward neural networks.

ANN depends upon three fundamental aspects, input and activation functions of the unit, network architecture and the weight of each input connection. Given that the first two aspects are fixed, the behavior of the ANN is defined by the current values of the weights. The weights of the net to be trained are initially set to random values, and then instances of the training set are repeatedly exposed to the net. The values for the input of an instance are placed on the input units and the output of the net is compared with the desired output for this instance. Then, all the weights in the net are adjusted slightly in the direction that would bring the output values of the net closer to the values for the desired output. There are several algorithms with which a network can be trained [27]. However, the most well-known and widely used learning algorithm to estimate the values of the weights is the Back Propagation (BP) algorithm. The general rule for updating weights is: Wji = ηδ j Oi where:

* η is a positive number (called learning rate), which determines the step size in the gradient descent search. A large value enables back propagation to move faster to the target weight configuration but it also increases the chance of its never reaching this target.
* Oi is the output computed by neuron i
* δ j = Oj(1 − Oj)(Tj − Oj) for the output neurons, where Tj the wanted output for the neuron j and
* δ j = Oj(1 − Oj) k δkWkj for the internal (hidden) neurons During classification the signal at the input units propagates all the way through the net to determine the activation values at all the output units.

Each input unit has an activation value that represents some feature external to the net. Then, every input unit sends its activation value to each of the hidden units to which it is connected. Each of these hidden units calculates its own activation value and this signal are then passed on to output units. The activation value for each receiving unit is calculated according to a simple activation function. The function sums together the contributions of all sending units, where the contribution of a unit is defined as the weight of the connection between the sending and receiving units multiplied by the sending unit’s activation value. This sum is usually then further modified, for example, by adjusting the activation sum to a value between 0 and 1 and/or by setting the activation value to zero unless a threshold level for that sum is reached.

Feed-forward neural networks are usually trained by the original back propagation algorithm or by some variant. Their greatest problem is that they are too slow for most applications. One of the approaches to speed up the training rate is to estimate optimal initial weights [28]. Genetic algorithms have been used to train the weights of neural networks [29] and to find the architecture of neural networks [30]. There are also Bayesian methods in existence which attempt to train neural networks. [31] compare two Bayesian methods for training neural networks. A number of other techniques have emerged recently which attempt to improve ANNs training algorithms by changing the architecture of the networks as training proceeds. These techniques include pruning useless nodes or weights [32] and constructive algorithms, where extra nodes are added as required [33].

ANN learning can be achieved, among others, through (i) synaptic weight modification, (ii) network structure modifications (creating or deleting neurons or synaptic connections), (iii) use of suitable attractors or other suitable stable state points, (iv)appropriate choice of activation functions. Since back-propagation training is a gradient descending process, it may get stuck in local minima in this weight-space. It is because of this possibility that neural network models are characterized by high variance and unsteadiness. Radial Basis Function (RBF) networks have been also widely applied in many science and engineering fields [34]. An RBF network is a three-layer feedback network, in which each hidden unit implements a radial activation function and each output unit implements a weighted sum of hidden units outputs. Its training procedure is usually divided into two stages. First, the centers and widths of the hidden layer are determined by clustering algorithms. Second, the weights connecting the hidden layer with the output layer are determined by Singular Value Decomposition (SVD) or Least Mean Squared (LMS) algorithms. The problem of selecting the appropriate number of basic functions remains a critical issue for RBF networks. The number of basic functions controls the complexity and the generalization ability of RBF networks. RBF networks with too few basic functions cannot fit the training data adequately due to limited flexibility. On the other hand, those with too many basis functions yield poor generalization abilities since they are too flexible and erroneously fit the noise in the training data.

In 1957, Frank Rosenblatt invented a Perceptron at the Cornell Aeronautical Laboratory. The simplest neural network possible is called a perceptron. It is the most basic form of a neural network. It consists of one or more inputs, a processor and a single output. Each individual cell is known as a perceptron or a node. “Feed-forward” model is followed by a perceptron.

Other well-known algorithms are based on the notion of perceptron. Perceptron can be briefly described as: If x1 through xn are input feature values and w1 through wn are connection weights/prediction vector (typically real numbers in the interval [−1, 1]), then perceptron computes the sum of weighted inputs: i xiwi and output goes through an adjustable threshold: if the sum is above threshold, output is 1; else it is 0. The most common way the perceptron algorithm is used for learning from a batch of training instances is to run the algorithm repeatedly through the training set until it finds a prediction vector which is correct on all of the training set. This prediction rule is then used for predicting the labels on the test set. WINNOW [35] is based on the perceptron idea. It was experimentally proved [36] that WINNOW can adapt rapidly to changes in the target function (concept drift). A target function (such as user preferences) is not static in time. In order to enable, for example, a decision tree algorithm to respond to changes, it is necessary to decide which old training instances could be deleted. A number of algorithms similar to WINNOW have been developed, such as those by [37].

[38]created a newer algorithm, called voted-perceptron, which stores more information during training and then uses this elaborate information to generate better predictions about the test data. The information it maintains during training is the list of all prediction vectors that were generated after each and every mistake. For each such vector, it counts the number of iterations it “survives” until the next mistake is made; [38] refer to this count as the “weight” of the prediction vector. To calculate a prediction the algorithm computes the binary prediction of each one of the prediction vectors and combines all these predictions by means of a weighted majority vote. The weights used are the survival times described above.

There are two types of perceptrons in a neural network:

1. Single Layer Perceptron: It consists of a single layer of nodes or perceptrons between the input and output.
2. Multi Layer Perceptron: It has similar kind of structure as compared to single layer perceptron with one or more hidden layers. It consists of several layers of nodes or perceptrons, piled up on top of each other between the input and output layers of neural network. They are more powerful than single layer perceptron.

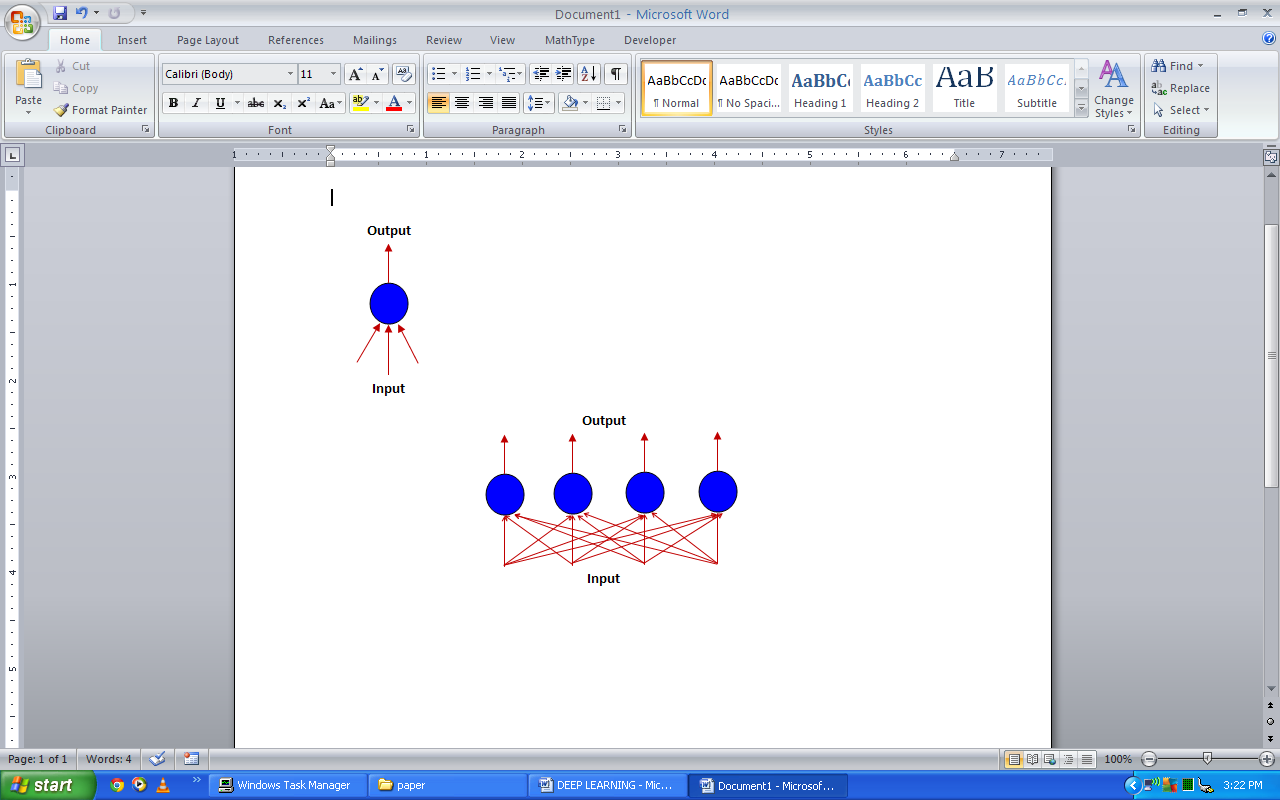
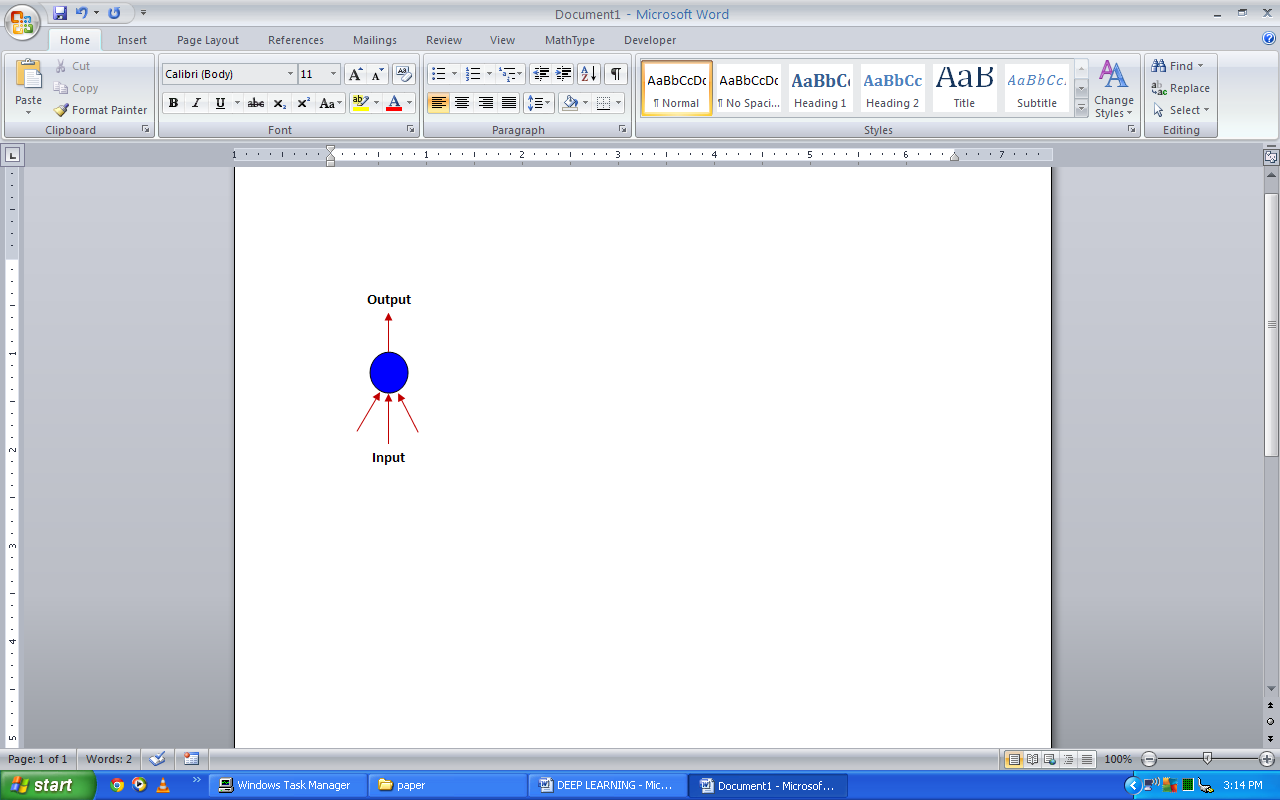


Figure 2.5 Single layer Perceptron

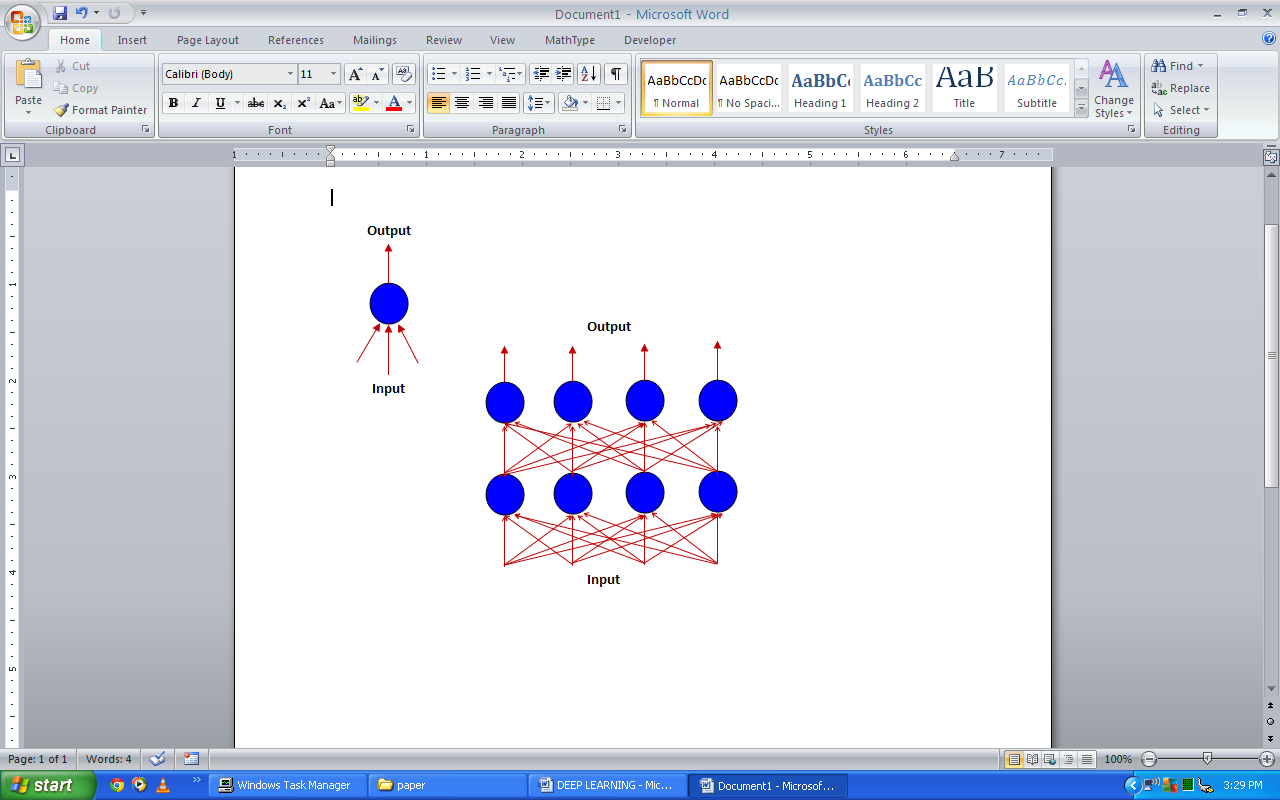


Figure2.6 Multilayer Perceptron

From the earliest days of pattern recognition, the aim of researchers has been to replace hand-engineered features with trainable multilayer networks, but despite its simplicity, the solution was not widely understood until the mid-1980s. As it turns out, multilayer architectures can be trained by simple stochastic gradient descent. As long as the modules are relatively smooth functions of their inputs and of their internal weights, one can compute gradients using the back propagation procedure. The idea that this could be done, and that it worked, was discovered independently by several different groups during the 1970s and 1980s.

The back propagation procedure to compute the gradient of an objective function with respect to the weights of a multilayer stack of modules is nothing more than a practical application of the chain rule for derivatives. The key insight is that the derivative (or gradient) of the objective with respect to the input of a module can be computed by working backwards from the gradient with respect to the output of that module (or the input of the subsequent module). The back propagation equation can be applied repeatedly to propagate gradients through all modules, starting from the output at the top (where the network produces its prediction) all the way to the bottom (where the external input is fed). Once these gradients have been computed, it is straightforward to compute the gradients with respect to the weights of each module.

Many applications of deep learning use feed forward neural network architectures, which learn to map a fixed-size input (for example, an image) to a fixed-size output (for example, a probability for each of several categories). To go from one layer to the next, a set of units compute a weighted sum of their inputs from the previous layer and pass the result through a non-linear function. At present, the most popular non-linear function is the rectified linear unit (ReLU), which is simply the half-wave rectifier f(z)= max(z, 0). In past decades, neural nets used smoother non-linearities, such as tanh(z) or 1/(1+exp(−z)), but the ReLU typically learns much faster in networks with many layers, allowing training of a deep supervised network without unsupervised pre-training. Units that are not in the input or output layer are conventionally called hidden units. The hidden layers can be seen as distorting the input in a non-linear way so that categories become linearly separable by the last layer.

In the late 1990s, neural nets and back propagation were largely forsaken by the machine-learning community and ignored by the computer-vision and speech-recognition communities. It was widely thought that learning useful, multistage, feature extractors with little prior knowledge was infeasible. In particular, it was commonly thought that simple gradient descent would get trapped in poor local minima — weight configurations for which no small change would reduce the average error.

In practice, poor local minima are rarely a problem with large networks. Regardless of the initial conditions, the system nearly always reaches solutions of very similar quality. Recent theoretical and empirical results strongly suggest that local minima are not a serious issue in general. Instead, the landscape is packed with a combinatorially large number of saddle points where the gradient is zero, and the surface curves up in most dimensions and curves down in the remainder. The analysis seems to show that saddle points with only a few downward curving directions are present in very large numbers, but almost all of them have very similar values of the objective function. Hence, it does not much matter which of these saddle points the algorithm gets stuck at.

Interest in deep feed forward networks was revived around 2006 by a group of researchers brought together by the Canadian Institute for Advanced Research (CIFAR). The researchers introduced unsupervised learning procedures that could create layers of feature detectors without requiring labeled data. The objective in learning each layer of feature detectors was to be able to reconstruct or model the activities of feature detectors (or raw inputs) in the layer below. By ‘pre-training’ several layers of progressively more complex feature detectors using this reconstruction objective, the weights of a deep network could be initialized to sensible values. A final layer of output units could then be added to the top of the network and the whole deep system could be fine-tuned using standard back propagation. This worked remarkably well for recognizing handwritten digits or for detecting pedestrians, especially when the amount of labeled data was very limited.

The first major application of this pre-training approach was in speech recognition, and it was made possible by the advent of fast graphics processing units (GPUs) that were convenient to program and allowed researchers to train networks 10 or 20 times faster. In 2009, the approach was used to map short temporal windows of coefficients extracted from a sound wave to a set of probabilities for the various fragments of speech that might be represented by the frame in the centre of the window. It achieved record-breaking results on a standard speech recognition benchmark that used a small vocabulary and was quickly developed to give record-breaking results on a large vocabulary task. By 2012, versions of the deep net from 2009 were being developed by many of the major speech groups6 and were already being deployed in Android phones. For smaller data sets, unsupervised pre-training helps to prevent over fitting, leading to significantly better generalization when the number of labeled examples is small, or in a transfer setting where we have lots of examples for some ‘source’ tasks but very few for some ‘target’ tasks. Once deep learning had been rehabilitated, it turned out that the pre-training stage was only needed for small data sets.

There was, however, one particular type of deep, feed forward network that was much easier to train and generalized much better than networks with full connectivity between adjacent layers. This was the convolutional neural network (ConvNet). It achieved many practical successes during the period when neural networks were out of favor and it has recently been widely adopted by the computer vision community.

**2.4. Convolution Neural Network**

ConvNets are designed to process data that come in the form of multiple arrays, for example a color image composed of three 2D arrays containing pixel intensities in the three color channels. Many data modalities are in the form of multiple arrays: 1D for signals and sequences, including language; 2D for images or audio spectrograms; and 3D for video or volumetric images. There are four key ideas behind ConvNets that take advantage of the properties of natural signals: local connections, shared weights, pooling and the use of many layers.

The architecture of a typical ConvNet is structured as a series of stages. The first few stages are composed of two types of layers: convolutional layers and pooling layers. Units in a convolutional layer are organized in feature maps, within which each unit is connected to local patches in the feature maps of the previous layer through a set of weights called a filter bank. The result of this local weighted sum is then passed through a non-linearity such as a ReLU. All units in a feature map share the same filter bank. Different feature maps in a layer use different filter banks. The reason for this architecture is twofold. First, in array data such as images, local groups of values are often highly correlated, forming distinctive local motifs that are easily detected. Second, the local statistics of images and other signals are invariant to location. In other words, if a motif can appear in one part of the image, it could appear anywhere, hence the idea of units at different locations sharing the same weights and detecting the same pattern in different parts of the array. Mathematically, the filtering operation performed by a feature map is a discrete convolution, hence the name.

Although the role of the convolutional layer is to detect local conjunctions of features from the previous layer, the role of the pooling layer is to merge semantically similar features into one. Because the relative positions of the features forming a motif can vary somewhat, reliably detecting the motif can be done by coarse-graining the position of each feature. A typical pooling unit computes the maximum of a local patch of units in one feature map (or in a few feature maps). Neighboring pooling units take input from patches that are shifted by more than one row or column, thereby reducing the dimension of the representation and creating invariance to small shifts and distortions. Two or three stages of convolution, non-linearity and pooling are stacked, followed by more convolutional and fully-connected layers. Back propagating gradients through a ConvNet is as simple as through a regular deep network, allowing all the weights in all the filter banks to be trained.

Deep neural networks exploit the property that many natural signals are compositional hierarchies; in which higher-level features are obtained by composing lower-level ones. In images, local combinations of edges form motifs, motifs assemble into parts, and parts form objects. Similar hierarchies exist in speech and text from sounds to phones, phonemes, syllables, words and sentences. The pooling allows representations to vary very little when elements in the previous layer vary in position and appearance.

The convolutional and pooling layers in ConvNets are directly inspired by the classic notions of simple cells and complex cells in visual neuroscience, and the overall architecture is reminiscent of the LGN–V1–V2–V4–IT hierarchy in the visual cortex ventral pathway. When ConvNet models and monkeys are shown the same picture, the activations of high-level units in the ConvNet explains half of the variance of random sets of 160 neurons in the monkey’s infero-temporal cortex. ConvNets have their roots in the neocognitron, the architecture of which was somewhat similar, but did not have an end-to-end supervised-learning algorithm such as back propagation. A primitive 1D ConvNet called a time-delay neural net was used for the recognition of phonemes and simple words.

There have been numerous applications of convolutional networks going back to the early 1990s, starting with time-delay neural networks for speech recognition and document reading. The document reading system used a ConvNet trained jointly with a probabilistic model that implemented language constraints. By the late 1990s this system was reading over 10% of all the cheques in the United States. A number of ConvNet-based optical character recognition and handwriting recognition systems were later deployed by Microsoft. ConvNets were also experimented with in the early 1990s for object detection in natural images, including faces and hands, and for face recognition.

**2.5. Transfer Learning**

Transfer learning is a refinement of learning in a new task by transferring the knowledge from a related task which has already been learned. While most of the machine learning algorithms addresses a single task, an ongoing topic of interest in the community of machine learning is the development of algorithms which assists transfer learning.

Methods are being developed by transfer learning for transfer of learned knowledge in one or more source tasks and utilize it to enhance learning in a related target task. Methods that facilitate knowledge transfer represent development towards making machine learning as efficient as human learning.

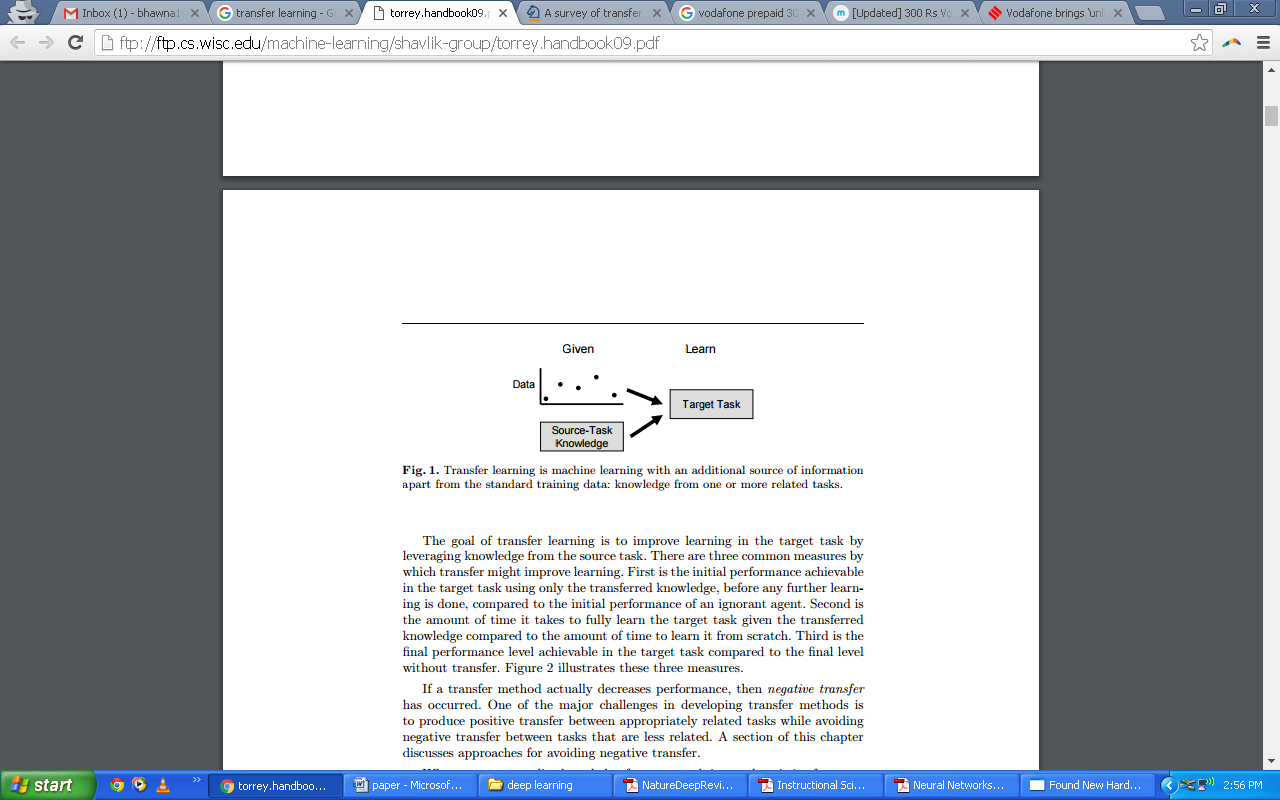


Figure 2.7 Transfer learning

Transfer learning is machine learning with an additional source of information apart from the standard training data: knowledge from one or more related tasks. Transfer techniques are likely to be extremely dependent on the machine learning algorithms which are being used to learn and understand the tasks, and can be considered as expansion of those algorithms.Objective of transfer learning is to improve learning from the source task in the desired task by leveraging knowledge. Following are three ways in which transfer can improve learning:

In the first measure, initial performance is achievable in the desired target task by use of only transferred knowledge, before any additional learning is done, compared to the initial performance of an ignorant agent.Second is the time taken to fully learn the target task when the transferred knowledge is given as compared to the time taken to learn it from scratch. Third measure is the final performance level attained in the target task as compared to the final level without transfer.

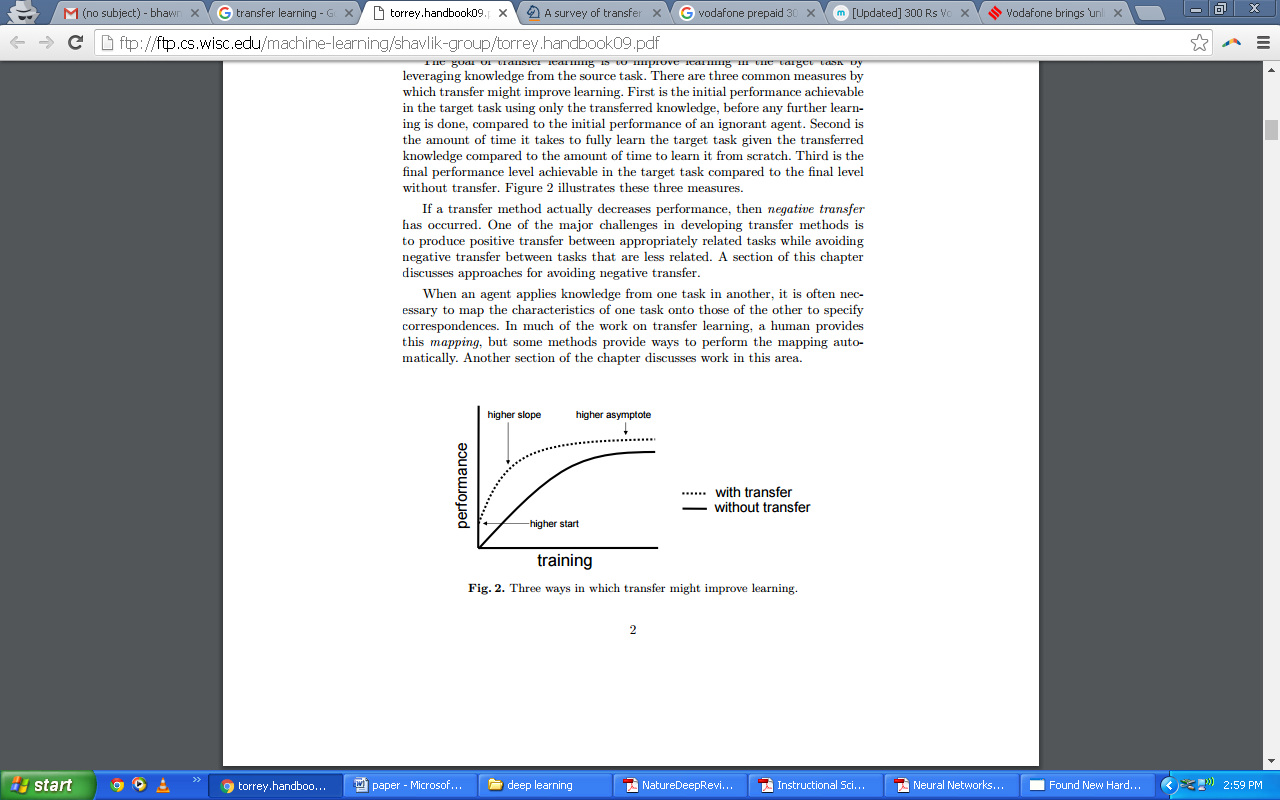


Figure 2.8 Three ways to improve learning

Negative transfer has happened if a transfer technique actually decreases performance. Producing positive transfer between almost related tasks while refraining negative transfer between less related tasks are one of the major challenges in development of transfer methods.

When knowledge is applied by agent from one task to another, mapping the characteristics of one task onto those of the other to define correspondences is often necessary. A human provides this mapping in much of the work on transfer learning, but mapping is performed automatically by some methods.

**2.6. Inception of GoogLeNets**

Artificial Neural Networks have spurred remarkable recent progress in image classification and speech recognition. But even though these are very useful tools based on well-known mathematical methods, we actually understand surprisingly little of why certain models work and others don’t.We train an artificial neural network by showing it millions of training examples and gradually adjusting the network parameters until it gives the classifications we want. The network typically consists of 10-30 stacked layers of artificial neurons. Each image is fed into the input layer, which then talks to the next layer, until eventually the “output” layer is reached. The network’s “answer” comes from this final output layer.

We know that after training, each layer progressively extracts higher and higher-level features of the image, until the final layer essentially makes a decision on what the image shows. For example, the first layer maybe looks for edges or corners. Intermediate layers interpret the basic features to look for overall shapes or components, like a door or a leaf. The final few layers assemble those into complete interpretations—these neurons activate in response to very complex things such as entire buildings or trees. One way to visualize what goes on is to turn the network upside down and ask it to enhance an input image in such a way as to elicit a particular interpretation. Say you want to know what sort of image would result in “Banana.” Start with an image full of random noise, then gradually tweak the image towards what the neural net considers a banana. By itself, that doesn’t work very well, but it does if we impose a prior constraint that the image should have similar statistics to natural images, such as neighboring pixels needing to be correlated.

Why is this important? Well, we train networks by simply showing them many examples of what we want them to learn, hoping they extract the essence of the matter at hand, and learn to ignore what doesn’t matter. But how do we check that the network has correctly learned the right features? It can help to visualize the network’s representation of a fork.Indeed, in some cases, this reveals that the neural net isn’t quite looking for the thing we thought it was.

Instead of exactly prescribing which feature we want the network to amplify, we can also let the network make that decision. In this case we simply feed the network an arbitrary image or photo and let the network analyze the picture. We then pick a layer and ask the network to enhance whatever it detected. Each layer of the network deals with features at a different level of abstraction, so the complexity of features we generate depends on which layer we choose to enhance.

If we choose higher-level layers, which identify more sophisticated features in images, complex features or even whole objects tend to emerge. Again, we just start with an existing image and give it to our neural net. We ask the network: “Whatever you see there, I want more of it!” This creates a feedback loop: if a cloud looks a little bit like a bird, the network will make it look more like a bird. This in turn will make the network recognize the bird even more strongly on the next pass and so forth, until a highly detailed bird appears, seemingly out of nowhere.

The results are intriguing—even a relatively simple neural network can be used to over-interpret an image, just like as children we enjoyed watching clouds and interpreting the random shapes. This network was trained mostly on images of animals, so naturally it tends to interpret shapes as animals. But because the data is stored at such a high abstraction, the results are an interesting remix of these learned features.

Of course, we can do more than cloud watching with this technique. We can apply it to any kind of image. The results vary quite a bit with the kind of image, because the features that are entered bias the network towards certain interpretations.

This technique gives us a qualitative sense of the level of abstraction that a particular layer has achieved in its understanding of images. This technique is known as “Inceptionism” in reference to the neural net architecture used.

If we apply the algorithm iteratively on its own outputs and apply some zooming after each iteration, we get an endless stream of new impressions, exploring the set of things the network knows about. We can even start this process from a random-noise image, so that the result becomes purely the result of the neural network. The techniques presented here help us understand and visualize how neural networks are able to carry out difficult classification tasks, improve network architecture, and check what the network has learned during training. It also makes us wonder whether neural networks could become a tool for artists—a new way to remix visual concepts—or perhaps even shed a little light on the roots of the creative process in general.

The computational cost of Inception is also much lower than VGGNet or its higher performing successors. This has made it feasible to utilize Inception networks in big-data scenarios, where huge amount of data needed to be processed at reasonable cost or scenarios where memory or computational capacity is inherently limited, for example in mobile vision settings. It is certainly possible to mitigate parts of these issues by applying specialized solutions to target memory use, or by optimizing the execution of certain operations via computational tricks. However, these methods add extra complexity. Furthermore, these methods could be applied to optimize the Inception architecture as well, widening the efficiency gap again.

Still, the complexity of the Inception architecture makes it more difficult to make changes to the network. If the architecture is scaled up naively, large parts of the computational gains can be immediately lost. Also, does not provide a clear description about the contributing factors that lead to the various design decisions of the GoogLeNet architecture. This makes it much harder to adapt it to new use-cases while maintaining its efficiency. For example, if it is deemed necessary to increase the capacity of some Inception-style model, the simple transformation of just doubling the number of all filter bank sizes will lead to a 4x increase in both computational cost and number of parameters. This might prove prohibitive or unreasonable in a lot of practical scenarios, especially if the associated gains are modest.

The main idea of the Inception architecture is to consider how an optimal local sparse structure of a convolutional vision network can be approximated and covered by readily available dense components. Note that assuming translation invariance means that our network will be built from convolutional building blocks. All we need is to find the optimal local construction and to repeat it spatially. [2]suggests a layer-by layer construction where one should analyze the correlation statistics of the last layer and cluster them into groups of units with high correlation. These clusters form the units of the next layer and are connected to the units in the previous layer. We assume that each unit from an earlier layer corresponds to some region of the input image and these units are grouped into filter banks. In the lower layers (the ones close to the input) correlated units would concentrate in local regions. Thus, we would end up with a lot of clusters concentrated in a single region and they can be covered by a layer of 1×1 convolutions in the next layer, as suggested in [12]. However, one can also expect that there will be a smaller number of more spatially spread out clusters that can be covered by convolutions over larger patches, and there will be a decreasing number of patches over larger and larger regions. In order to avoid patch-alignment issues, current incarnations of the Inception architecture are restricted to filter sizes 1×1, 3×3 and 5×5; this decision was based more on convenience rather than necessity. It also means that the suggested architecture is a combination of all those layers with their output filter banks concatenated into a single output vector forming the input of the next stage. Additionally, since pooling operations have been essential for the success of current convolutional networks, it suggests that adding an alternative parallel pooling path in each such stage should have additional beneficial effect, too.

As these “Inception modules” are stacked on top of each other, their output correlation statistics are bound to vary: as features of higher abstraction are captured by higher layers, their spatial concentration is expected to decrease. This suggests that the ratio of 3×3 and 5×5 convolutions should increase as we move to higher layers.

One big problem with the above modules, at least in this naive form, is that even a modest number of 5×5 convolutions can be prohibitively expensive on top of a convolutional layer with a large number of filters. This problem becomes even more pronounced once pooling units are added to the mix: the number of output filters equals to the number of filters in the previous stage. The merging of output of the pooling layer with outputs of the convolutional layers would lead to an inevitable increase in the number of outputs from stage to stage. While this architecture might cover the optimal sparse structure, it would do it very inefficiently, leading to a computational blow up within a few stages.

This leads to the second idea of the Inception architecture: judiciously reducing dimension wherever the computational requirements would increase too much otherwise. This is based on the success of embedding: even low dimensional embedding might contain a lot of information about a relatively large image patch. However, embedding represent information in a dense, compressed form and compressed information is harder to process. The representation should be kept sparse at most places (as required by the conditions of [2]) and compress the signals only whenever they have to be aggregated en masse. That is, 1×1 convolutions are used to compute reductions before the expensive 3×3 and 5×5 convolutions. Besides being used as reductions, they also include the use of rectified linear activation making them dual-purpose.

In general, an Inception network is a network consisting of modules of the above type stacked upon each other, with occasional max-pooling layers with stride 2 to halve the resolution of the grid. For technical reasons (memory efficiency during training), it seemed beneficial to start using Inception modules only at higher layers while keeping the lower layers in traditional convolutional fashion. This is not strictly necessary, simply reflecting some infrastructural inefficiencies in our current implementation.

A useful aspect of this architecture is that it allows for increasing the number of units at each stage significantly without an uncontrolled blow-up in computational complexity at later stages. This is achieved by the ubiquitous use of dimensionality reduction prior to expensive convolutions with larger patch sizes. Furthermore, the design follows the practical intuition that visual information should be processed at various scales and then aggregated so that the next stage can abstract features from the different scales simultaneously.

The improved use of computational resources allows for increasing both the width of each stage as well as the number of stages without getting into computational difficulties. One can utilize the Inception architecture to create slightly inferior, but computationally cheaper versions of it. We have found that all the available knobs and levers allow for a controlled balancing of computational resources resulting in networks that are 3 − 10× faster than similarly performing networks with non-Inception architecture, however this requires careful manual design at this point.

By the “GoogLeNet” name we refer to the particular incarnation of the Inception architecture used in our submission for the ILSVRC 2014 competition. We also used one deeper and wider Inception network with slightly superior quality, but adding it to the ensemble seemed to improve the results only marginally. We omit the details of that network, as empirical evidence suggests that the influence of the exact architectural parameters is relatively minor. This network (trained with different image patch sampling methods) was used for 6 out of the 7 models in our ensemble.

All the convolutions, including those inside the Inception modules, use rectified linear activation. The size of the receptive field in our network is 224×224 in the RGB color space with zero mean. “#3×3 reduce” and “#5×5 reduce” stands for the number of 1×1 filters in the reduction layer used before the 3×3 and 5×5 convolutions. One can see the number of 1×1 filters in the projection layer after the built-in max-pooling in the pool proj column. All these reduction/projection layers use rectified linear activation as well.

The network was designed with computational efficiency and practicality in mind, so that inference can be run on individual devices including even those with limited computational resources, especially with low-memory footprint.

The network is 22 layers deep when counting only layers with parameters (or 27 layers if we also count pooling). The overall number of layers (independent building blocks) used for the construction of the network is about 100. The exact number depends on how layers are counted by the machine learning infrastructure. The use of average pooling before the classifier is based on [12], although our implementation has an additional linear layer. The linear layer enables us to easily adapt our networks to other label sets, however it is used mostly for convenience and we do not expect it to have a major effect. We found that a move from fully connected layers to average pooling improved the top-1 accuracy by about 0.6%, however the use of dropout remained essential even after removing the fully connected layers.

Given relatively large depth of the network, the ability to propagate gradients back through all the layers in an effective manner was a concern. The strong performance of shallower networks on this task suggests that the features produced by the layers in the middle of the network should be very discriminative. By adding auxiliary classifiers connected to these intermediate layers, discrimination in the lower stages in the classifier was expected. This was thought to combat the vanishing gradient problem while providing regularization. These classifiers take the form of smaller convolutional networks put on top of the output of the Inception (4a) and (4d) modules. During training, their loss gets added to the total loss of the network with a discount weight (the losses of the auxiliary classifiers were weighted by 0.3). At inference time, these auxiliary networks are discarded. Later control experiments have shown that the effect of the auxiliary networks is relatively minor (around 0.5%) and that it required only one of them to achieve the same effect.

The older Inception models used to be trained in a partitioned manner, where each replica was partitioned into a multiple sub-networks in order to be able to fit the whole model in memory. However, the Inception architecture is highly tunable, meaning that there are a lot of possible changes to the number of filters in the various layers that do not affect the quality of the fully trained network. In order to optimize the training speed, we used to tune the layer sizes carefully in order to balance the computation between the various model sub-networks. In contrast, with the introduction of TensorFlow our most recent models can be trained without partitioning the replicas. This is enabled in part by recent optimizations of memory used by backpropagation, achieved by carefully considering what tensors are needed for gradient computation and structuring the computation to reduce the number of such tensors. Historically, we have been relatively conservative about changing the architectural choices and restricted our experiments to varying isolated network components while keeping the rest of the network stable. Not simplifying earlier choices resulted in networks that looked more complicated that they needed to be.

**3. PROJECT METHODOLOGY**

**3.1. Methodology Used**

The methodology of this project provides a detailed overview of all the guidelines, practices, procedures and working methods that will be followed in the development of the framework at various stages.

* + **Conceptualizing the Framework:**The entire framework will conceptualized in forms of Data Flow Diagrams (DFDs). This helps to know the complete inside-out of the project easily and thus, project can be made more efficiently.
  + **Software**: Appropriate software are chosen to make the project concept a reality. Python will be used to develop a machine learning algorithm to identify the scanned specie. Android Studio will be used to make the application having the algorithm embedded for android phones. SQLite is used to preserve the historical data and new data in the phones.
  + **Feasibility study**: A feasibility study was conducted on various aspects to check whether our project is feasible or not. This study led to a conclusion that the product is technically, economically and legally feasible.
  + **Collecting Training Data:** To make algorithm work, training dataset needs to be built. Around 500 pictures per specie are collected and used as the training data. The algorithm will work on the basis of this data.
  + **Developing Identification Algorithm:** The identification algorithm will be developed using machine learning technology. Python language is used for this purpose. The algorithm is to identify the specie scanned from the smart phone.
  + **Preparing Interfaces:** All the interfaces needed in the application will be made in Android Studio. The interfaces will be made to provide user-friendly and easy to use experience to users.
  + **Connecting all elements:**This stage includes connecting all the elements i.e. algorithm, interfaces, and images. This makes the application ready to use.

**3.2. Technology Used**

A wide range of software will be used for development of this framework like software for designing interfaces, conceptualizing the entire framework and for database connectivity. The software that will be used are listed below:

* + **Android Studio:**

Android Studio is the official integrated development environment (IDE) for Android platform development. It provides the fastest tools for building apps on every type of Android device. World-class code editing, debugging, performance tooling, a flexible build system and an instant build/deploy system are all the features of Android Studio used to build unique and high quality apps.

* + **Python:**

Python is a widely used high-level programming language used for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

* + **TensorFlow:**

TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

* + **Docker Toolbox:**

The Docker Toolbox is an installer to quickly and easily install and setup a Docker environment on computer. Docker Machine is a tool that lets you install Docker Engine on virtual hosts, and manage the hosts with docker-machine commands. You can use Machine to create Docker hosts on your local Mac or Windows box, on your company network, in your data center, or on cloud providers like AWS or Digital Ocean.

* + **Anaconda**

Anaconda is a freemium open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment. Its package management system is conda. Anaconda is the leading open data science platform powered by Python. The open source version of Anaconda is a high performance distribution of Python and R and includes over 1100 of the most popular Python, R and Scala packages for data science. Additionally, it gives access to over 720 packages that can easily be installed with conda, renowned package, dependency and environment manager that is included in Anaconda.

**3.3. Feasibility Study**

* **Economical Feasibility**

The project includes the use of open source software such as Python, Tensorflow and Android Studio. The framework can be used on any version of Android with Graphics Driver and Audio Driver. Therefore, this project is economically feasible.

* **Technical Feasibility**

All the inventories like interfaces and scripts will be made by using software such as Android Studio, Tensorflow and Python. The software are open source software and are easy to use. Therefore, this project is technically feasible.

* **Legal Feasibility**

The project ***Specmac*** is legally feasible. It uses only the open source software and easily available hardware such as Windows Operating System and Android Operating System. If there is any kind similarity to any person living or dead,it is purely coincidental. The development of this framework doesn’t use the copyright of others. Thus, the project is legally feasible.The project doesn’t conflicts any legal requirements which make it.

**4. THEORETICAL FRAMEWORK**

**4.1. Project Structure**

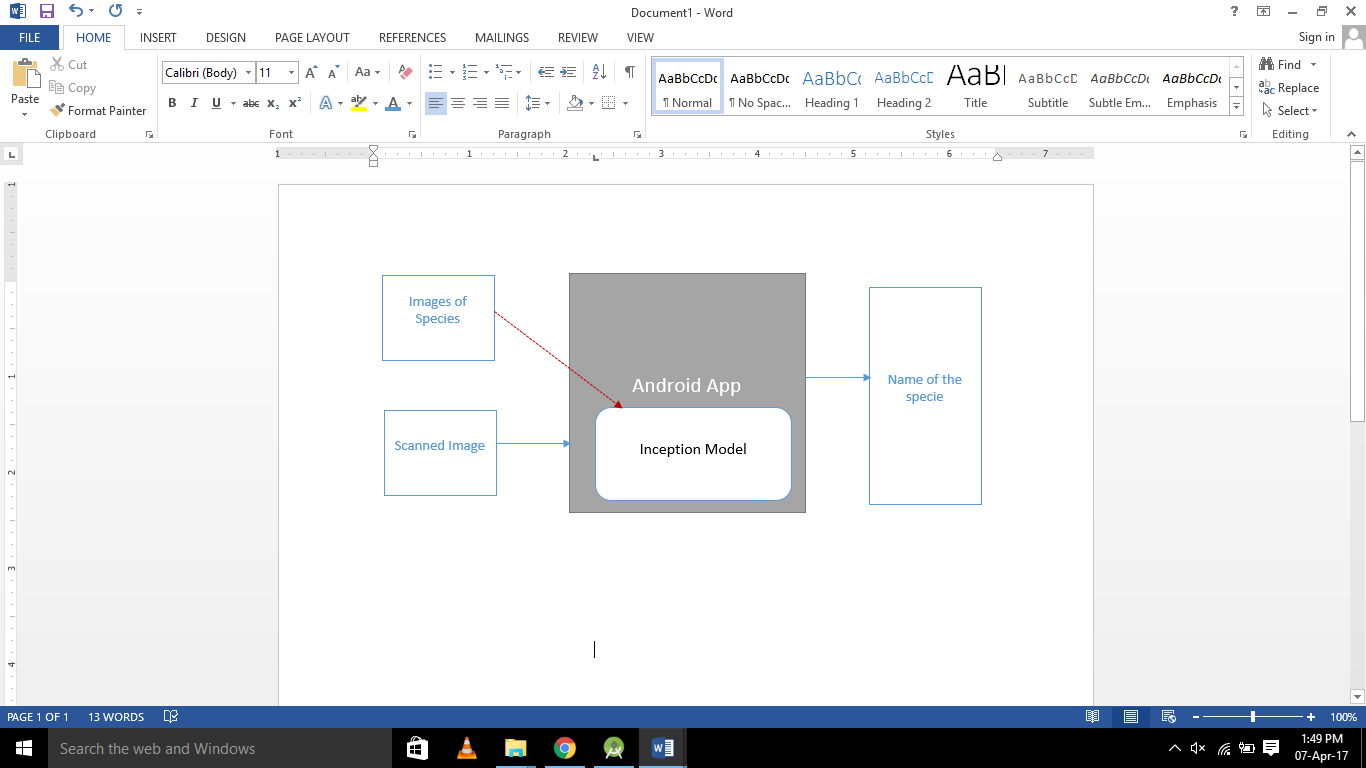


Figure 4.1 Structure of the Project

**4.2. Flow Chart**

Display Name and information

Display More Information

Press More?

Close Camera

Close Camera?

Inception Model

Training Set of Images

Launch App

Scan Specie

Calculate Probability

Open Camera

Yes

No

No

Yes

**5. EXPERIMENTAL PROCEDURE**

**5.1. Collecting Specie Images**

The project Specmac requires over 100 images for every specie to detect the scanned specie. For this purpose, around 500 images of each specie has been collected. The following table display the name of the specie and number of images collected.

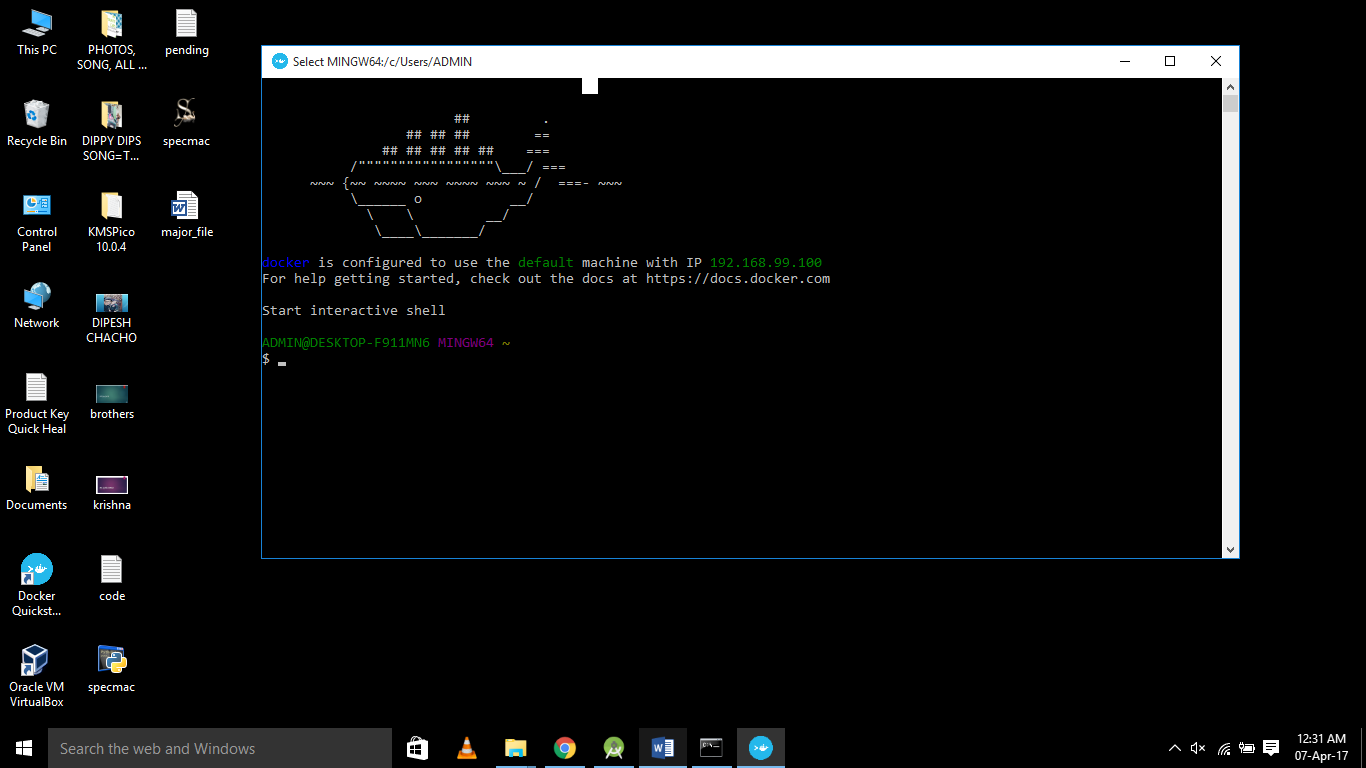
Table 5.1 Data Collected

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Name of the Specie** | **# of Images Collected** |
| 1. | Ant | 472 |
| 2. | Buffalo | 372 |
| 3. | Butterfly | 603 |
| 4. | Cat | 674 |
| 5. | Cow | 617 |
| 6. | Crow | 485 |
| 7. | Dog | 691 |
| 8. | Honey Bee | 617 |
| 9. | Horse | 793 |
| 10. | House Fly | 467 |
| 11. | Human | 1375 |
| 12 | Mosquito | 261 |
| 13. | Parrot | 501 |
| 14. | Pig | 401 |
| 15. | Pigeon | 496 |
| 16. | Rabbit | 654 |
| 17. | Rat | 413 |
| 18. | Sparrow | 616 |

The above collected images would serve as the training data for the model.

**5.2. Working with Tensorflow**

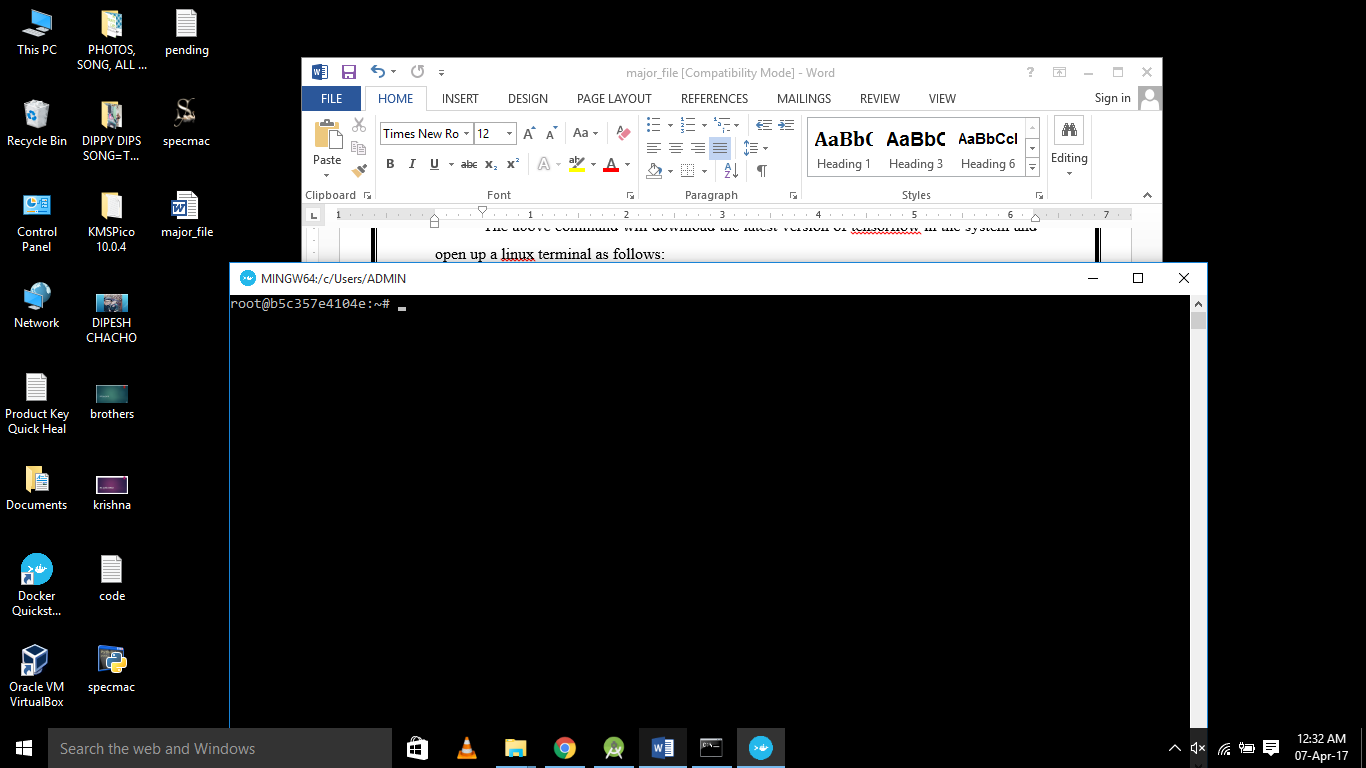
Tensorflow which is a library which works on Inception model made by Google for the tasks of image classification. This library is used with Python to make a model to classify between different images of species. This step is starting with the installation of docker toolbox which will serve as linux terminal on virtual machine. The docker is opened as:



After docker has been opened, tensorflow is installed using the following command:

docker run –it gcr.io/tensorflow/tensorflow:latest-devel

The above command will download the latest version of tensorflow in the system and open up a linux terminal at root directory as follows:



Run

exit

Second step is to install bazel in order to use tensorflow in android studio. For this purpose, chocolatey needs to be downloaded and install. To install chocolatey and then bazel, run command prompt as administrator and paste the following command:

@powershell -NoProfile -ExecutionPolicy Bypass -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"

choco install bazel

Next step is to download tensorflow library in the root folder which can be done using the following command in docker terminal:

git clone https://github.com/tensorflow/tenosrflow.git

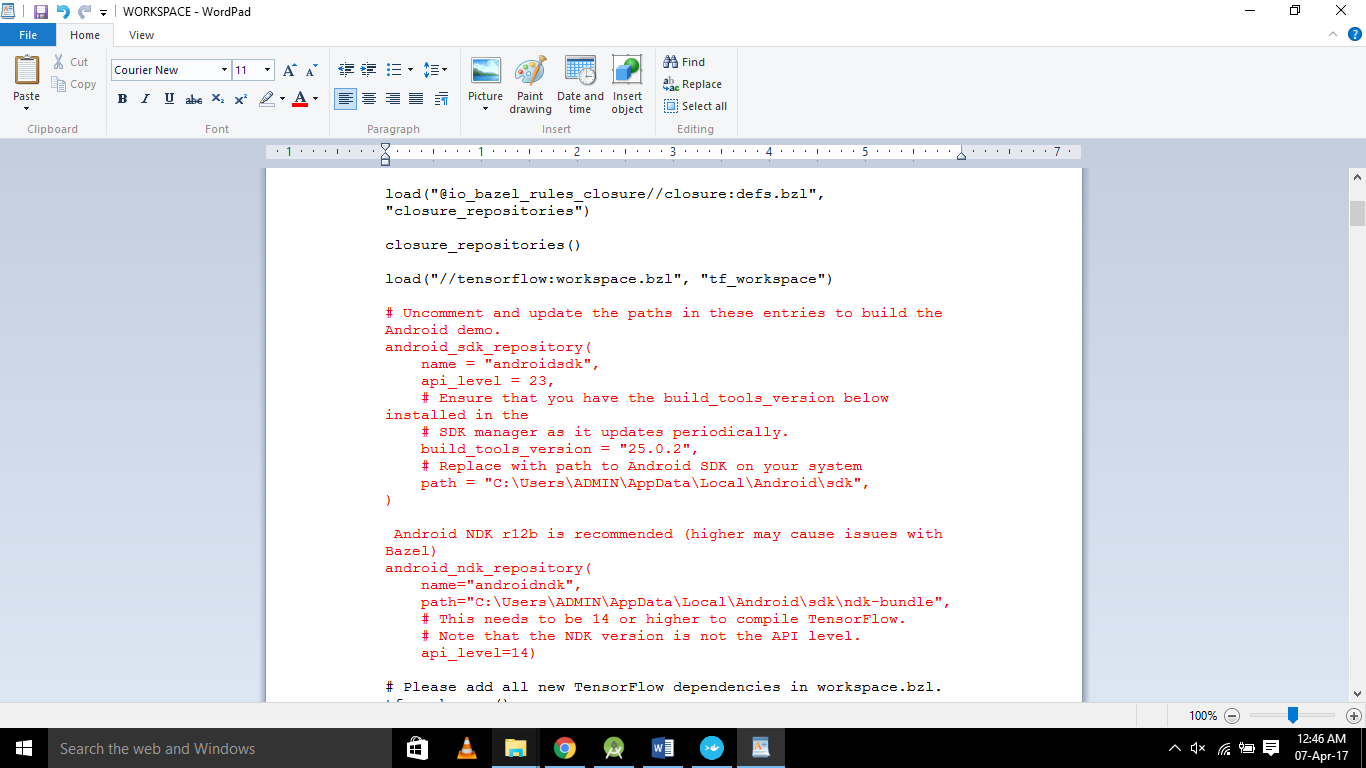
The next step requires python to be used. Install Anaconda with Python 3.5. Run the following command in docker terminal:

cd ternsorflow

./configure

The above command with ask for the python path. Provide the path for python.exe in Anaconda3 folder.

Now it’s time to prepare the bazel files. Find the file named WORKSPACE in /tensorflow root directory. The file would look like:



Uncomment the highlighted text by removing ‘#’ from the front. Update the path to sdk and ndk bundle in this. Now we need to download inception model to run the library with the following command:

curl –L https://storage.googleapis.com/download.tensorflow.org/models/inception5h.zip -o /tmp/inception5h.zip

unzip /tmp/inception5h.zip –d tensorflow/exmaples/android/assets/

Open android studio and open tensorflow/tensorflow/examples/android. Run the following command in docker:

cd $HOME

mkdir tf\_files

cd tf\_files

mkdir specie

cd $HOME

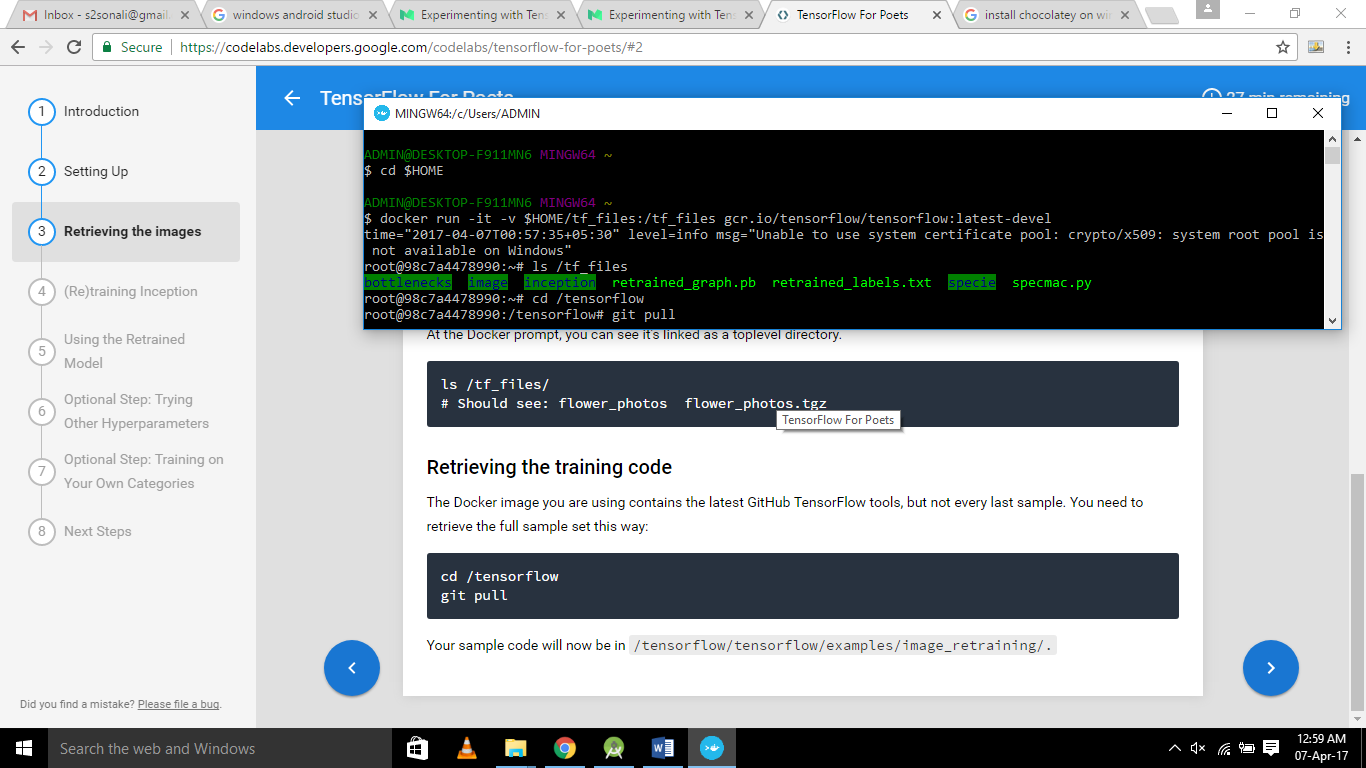
docker run –it –v $HOME/tf\_files:/tf\_files gcr.io/tensorflow/tensorflow:latest-devel

ls /tf\_files

cd /tensorflow

git pull

This command will structure the pointer at the tensorflow directory, pull all the downloaded resources to this folder



Now we need to train our model using the images of species we have collected. For this run the following command:

python tensorflow/examples/image\_retraining/retrain.py \

--bottleneck\_dir=/tf\_files/bottlenecks \

--how\_many\_training\_steps 500 \

--model\_dir=/tf\_files/inception \

--output\_graph=/tf\_files/retrained\_graph.pb \

--output\_labels=/tf\_files/retrained\_labels.txt \

--image\_dir /tf\_files/specie

exit

Bottlenecks would be created. Each bottle neck represent the layer of network. Around 6500 bottlenecks are created in this model. Next step is to classify the image for which one python file needs to be written as follows:

**Specmac.py**

import tensorflow as tf, sys

image\_path=sys.argv[1]

# Read in the image\_data

image\_data = tf.gfile.FastGFile(image\_path, 'rb').read()

# Load label file, strips off carraige return

label\_lines = [line.rstrip() for line

in tf.gfile.GFile("/tf\_files/retrained\_labels.txt")]

#Unpersists graph from file

with tf.gfile.FastGFile("/tf\_files/retrained\_graph.pb", 'rb') as f:

graph\_def = tf.GraphDef()

graph\_def.ParseFromString(f.read())

\_ = tf.import\_graph\_def(graph\_def, name='')

with tf.Session() as sess:

# Feed the image\_data as input to the graph and get first prediction

softmax\_tensor = sess.graph.get\_tensor\_by\_name('final\_result:0')

predictions = sess.run(softmax\_tensor, \

{'DecodeJpeg/contents:0':image\_data})

top\_k = predictions[0].argsort()[-len(predictions[0]):][::-1]

for node\_id in top\_k:

human\_string = label\_lines[node\_id]

score = predictions[0][node\_id]

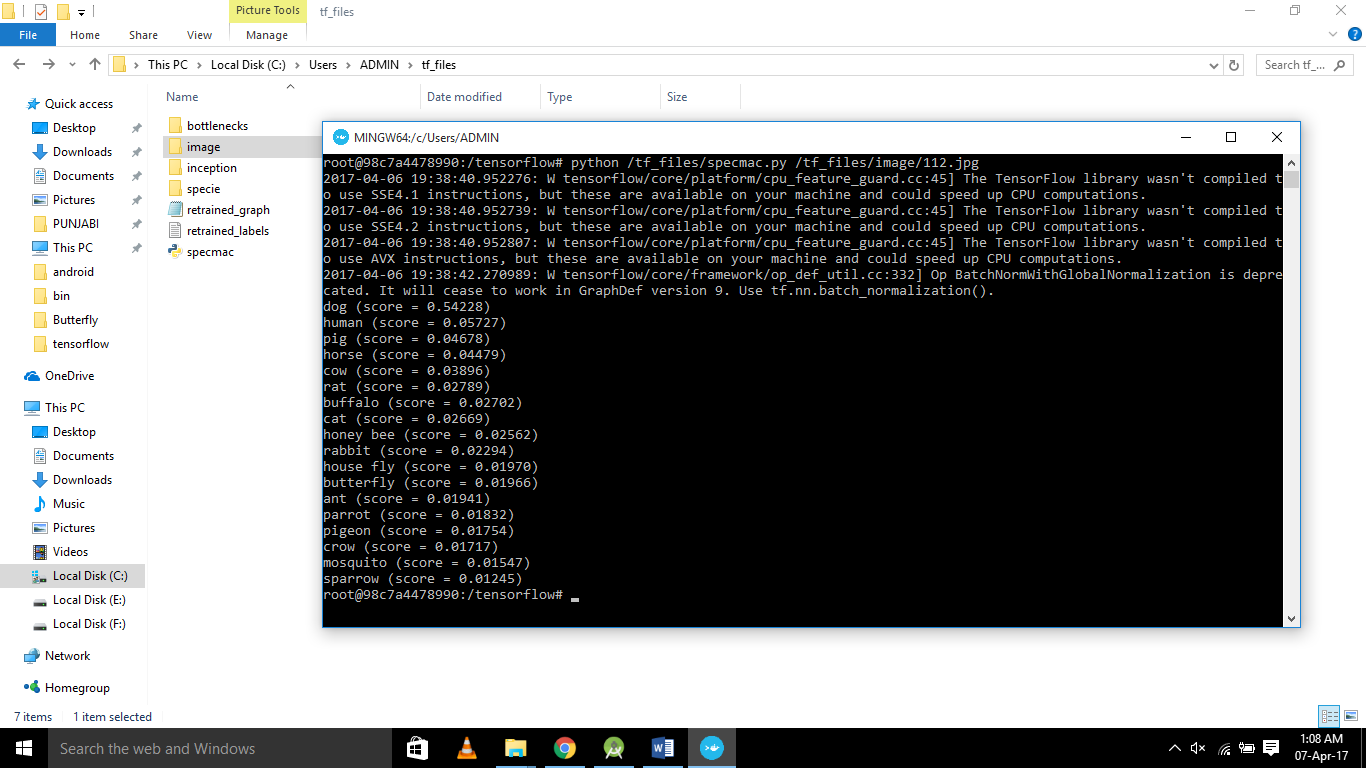
print('%s (score = %.5f)' % (human\_string,score))

Put this file in folder tf\_files. Run the following command in docker terminal to deploy this file in prepared model and test the model.

python /tf\_files/specmac.py /tf\_files/image/112.jpg



Figure 5.1: 112.jpg



**5.3. Working with Android Studio**

After we are done with retraining the inception model, we import it to Android Studio and code the interface around it. Following files contain the code used to make interfaces:

**AndroidMenifest.xml**

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="org.tensorflow.demo">  
  
<uses-sdk  
 android:minSdkVersion="21"  
 android:targetSdkVersion="25" />  
  
<uses-permission android:name="android.permission.CAMERA" />  
  
<uses-feature android:name="android.hardware.camera" />  
<uses-feature android:name="android.hardware.camera.autofocus" />  
  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />  
<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
  
<application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/title\_activity\_more"  
 android:theme="@style/AppTheme">  
<activity  
 android:name=".ClassifierActivity"  
 android:label="@string/title\_activity\_more"  
 android:theme="@style/MaterialTheme"  
 android:screenOrientation="portrait">  
<intent-filter>  
<action android:name="android.intent.action.MAIN" />  
  
<category android:name="android.intent.category.LAUNCHER" />  
</intent-filter>  
</activity>  
  
<activity  
 android:name=".Display"  
 android:label="@string/title\_activity\_more"  
 android:theme="@style/AppTheme"  
 android:screenOrientation="portrait"  
 android:parentActivityName="org.tensorflow.demo.ClassifierActivity">  
<meta-data android:name="android.support.PARENT\_ACTIVITY"  
 android:value="org.tensorflow.demo.ClassifierActivity"/>  
</activity>  
  
<activity  
 android:name=".More"  
 android:label="@string/title\_activity\_more"  
 android:screenOrientation="portrait"  
 android:theme="@style/AppTheme"  
 android:parentActivityName="org.tensorflow.demo.Display">  
<meta-data android:name="android.support.PARENT\_ACTIVITY"  
 android:value="org.tensorflow.demo.Display"/>  
</activity>  
</application>  
  
</manifest>

**activity\_camera.xml**

*<?*xml version="1.0" encoding="utf-8"*?>*<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/container"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#000"  
 tools:context="org.tensorflow.demo.CameraActivity" />

activity\_display.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context="org.tensorflow.demo.Display"  
 android:orientation="horizontal">  
<TextView  
 android:id="@+id/result\_name"  
 android:layout\_width="match\_parent"  
 android:layout\_height="72dp"  
 android:layout\_weight="1"  
 android:text="Name"  
 android:layout\_alignParentTop="true"  
 android:background="@android:color/holo\_purple"  
 android:textColor="@android:color/white"  
 android:textAllCaps="true"  
 android:textSize="35dp"  
 android:textStyle="bold"  
 android:gravity="center"  
 tools:layout\_editor\_absoluteY="0dp"  
 tools:layout\_editor\_absoluteX="8dp" />  
  
<TextView  
 android:id="@+id/result\_desc"  
 android:layout\_width="match\_parent"  
 android:layout\_height="250dp"  
 android:layout\_alignParentBottom="true"  
 android:layout\_weight="1"  
 android:text="Desc"  
 android:background="@android:color/white"  
 android:textColor="@android:color/black"  
 android:textSize="20dp"  
 android:gravity="left"  
 android:paddingLeft="@dimen/margin\_medium"  
 tools:layout\_editor\_absoluteY="0dp"  
 tools:layout\_editor\_absoluteX="8dp" />  
  
<ImageView  
 android:id="@+id/result\_img"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_above="@+id/result\_desc"  
 android:layout\_alignParentStart="true"  
 android:layout\_below="@+id/result\_name"  
 android:layout\_weight="1"  
 android:src="@drawable/ic\_launcher"  
 tools:layout\_editor\_absoluteX="8dp"  
 tools:layout\_editor\_absoluteY="0dp" />  
  
<Button  
 android:id="@+id/more"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:onClick="dispMsg"  
 android:text="More"  
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentEnd="true" />  
  
</RelativeLayout>

**activity\_more.xml**

*<?*xml version="1.0" encoding="utf-8"*?>*<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="horizontal"  
 tools:context="org.tensorflow.demo.More">  
  
<TextView  
 android:id="@+id/more\_name"  
 android:layout\_width="match\_parent"  
 android:layout\_height="72dp"  
 android:layout\_weight="1"  
 android:text="Name"  
 android:layout\_alignParentTop="true"  
 android:background="@android:color/holo\_purple"  
 android:textColor="@android:color/white"  
 android:textAllCaps="true"  
 android:textSize="35dp"  
 android:textStyle="bold"  
 android:gravity="center"  
 tools:layout\_editor\_absoluteY="0dp"  
 tools:layout\_editor\_absoluteX="8dp" />  
  
<ScrollView  
 android:id="@+id/SCROLLER\_ID"  
 android:layout\_width="match\_parent"  
 android:layout\_height="575dp"  
 android:scrollbars="vertical"  
 android:layout\_alignParentBottom="true"  
 android:layout\_below="@+id/more\_name"  
 android:fillViewport="true">  
  
<TextView  
 android:id="@+id/more\_desc"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_alignParentBottom="true"  
 android:layout\_weight="1"  
 android:text="Desc"  
 android:background="@android:color/white"  
 android:textColor="@android:color/black"  
 android:textSize="20dp"  
 android:gravity="left"  
 android:paddingLeft="@dimen/margin\_medium"  
 android:paddingTop="16dp"  
 tools:layout\_editor\_absoluteY="0dp"  
 tools:layout\_editor\_absoluteX="8dp" />  
</ScrollView>  
  
</RelativeLayout>

**camera\_connection\_fragment.xml**

*<?*xml version="1.0" encoding="utf-8"*?>*<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
<org.tensorflow.demo.AutoFitTextureView  
 android:id="@+id/texture"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_alignParentBottom="true" />  
  
<org.tensorflow.demo.RecognitionScoreView  
 android:id="@+id/results"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_alignParentTop="true" />  
  
<org.tensorflow.demo.OverlayView  
 android:id="@+id/debug\_overlay"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_alignParentBottom="true" />  
  
</RelativeLayout>

**camera\_connection\_fragment\_stylize.xml**

*<?*xml version="1.0" encoding="utf-8"*?>*<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
<org.tensorflow.demo.AutoFitTextureView  
 android:id="@+id/texture"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"/>  
  
<org.tensorflow.demo.OverlayView  
 android:id="@+id/tracking\_overlay"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"/>  
  
<org.tensorflow.demo.OverlayView  
 android:id="@+id/debug\_overlay"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"/>  
</FrameLayout>

**camera\_connection\_fragment\_tracking.xml**

*<?*xml version="1.0" encoding="utf-8"*?>*<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
<org.tensorflow.demo.AutoFitTextureView  
 android:id="@+id/texture"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true" />  
  
<RelativeLayout  
 android:id="@+id/black"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#FF000000" />  
  
<GridView  
 android:id="@+id/grid\_layout"  
 android:numColumns="7"  
 android:stretchMode="columnWidth"  
 android:layout\_alignParentBottom="true"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />  
  
<org.tensorflow.demo.OverlayView  
 android:id="@+id/overlay"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_alignParentTop="true" />  
  
<org.tensorflow.demo.OverlayView  
 android:id="@+id/debug\_overlay"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_alignParentTop="true" />  
</RelativeLayout>

**string.xml**

*<?*xml version="1.0" encoding="utf-8"*?>*<resources>  
<string name="description\_info">Info</string>  
<string name="request\_permission">This sample needs camera permission.</string>  
<string name="camera\_error">This device doesn\'t support Camera2 API.</string>  
<string name="title\_activity\_display">Display</string>  
<string name="title\_activity\_more">SPECMAC</string>  
</resources>

**style.xml**

*<?*xml version="1.0" encoding="utf-8"*?>*<resources>  
<style name="MaterialTheme" parent="android:Theme.Material.Light" >  
<item name = "android:windowActionBar">false</item>  
<item name = "android:windowNoTitle">true</item>  
</style>  
</resources>

**AutoFitTextureView.java**

package org.tensorflow.demo;  
  
import android.content.Context;  
import android.util.AttributeSet;  
import android.view.TextureView;  
  
public class AutoFitTextureView extends TextureView {  
private int ratioWidth = 0;  
private int ratioHeight = 0;  
  
public AutoFitTextureView(final Context context) {  
this(context, null);  
 }  
  
public AutoFitTextureView(final Context context, final AttributeSet attrs) {  
this(context, attrs, 0);  
 }  
  
public AutoFitTextureView(final Context context, final AttributeSet attrs, final int defStyle) {  
super(context, attrs, defStyle);  
 }  
  
public void setAspectRatio(final int width, final int height) {  
if (width < 0 || height < 0) {  
throw new IllegalArgumentException("Size cannot be negative.");  
 }  
ratioWidth = width;  
ratioHeight = height;  
 requestLayout();  
 }  
  
 @Override  
protected void onMeasure(final int widthMeasureSpec, final int heightMeasureSpec) {  
super.onMeasure(widthMeasureSpec, heightMeasureSpec);  
final int width = MeasureSpec.*getSize*(widthMeasureSpec);  
final int height = MeasureSpec.*getSize*(heightMeasureSpec);  
if (0 == ratioWidth || 0 == ratioHeight) {  
 setMeasuredDimension(width, height);  
 } else {  
if (width < height \* ratioWidth / ratioHeight) {  
 setMeasuredDimension(width, width \* ratioHeight / ratioWidth);  
 } else {  
 setMeasuredDimension(height \* ratioWidth / ratioHeight, height);  
 }  
 }  
 }  
}

**CameraActivity.java**

package org.tensorflow.demo;  
  
import android.Manifest;  
import android.app.Activity;  
import android.app.Fragment;  
import android.content.pm.PackageManager;  
import android.media.Image.Plane;  
import android.media.ImageReader.OnImageAvailableListener;  
import android.os.Build;  
import android.os.Bundle;  
import android.os.Handler;  
import android.os.HandlerThread;  
import android.util.Size;  
import android.view.KeyEvent;  
import android.view.WindowManager;  
import android.widget.Toast;  
import java.nio.ByteBuffer;  
import org.tensorflow.demo.env.Logger;  
import org.tensorflow.demo.R;  
  
public abstract class CameraActivity extends Activity implements OnImageAvailableListener {  
private static final Logger *LOGGER* = new Logger();  
  
private static final int *PERMISSIONS\_REQUEST* = 1;  
  
private static final String *PERMISSION\_CAMERA* = Manifest.permission.*CAMERA*;  
private static final String *PERMISSION\_STORAGE* = Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*;  
  
private boolean debug = false;  
  
private Handler handler;  
private HandlerThread handlerThread;  
  
 @Override  
protected void onCreate(final Bundle savedInstanceState) {  
*LOGGER*.d("onCreate " + this);  
super.onCreate(null);  
 getWindow().addFlags(WindowManager.LayoutParams.*FLAG\_KEEP\_SCREEN\_ON*);  
  
 setContentView(R.layout.*activity\_camera*);  
  
if (hasPermission()) {  
 setFragment();  
 } else {  
 requestPermission();  
 }  
 }  
  
 @Override  
public synchronized void onStart() {  
*LOGGER*.d("onStart " + this);  
super.onStart();  
 }  
  
 @Override  
public synchronized void onResume() {  
*LOGGER*.d("onResume " + this);  
super.onResume();  
  
handlerThread = new HandlerThread("inference");  
handlerThread.start();  
handler = new Handler(handlerThread.getLooper());  
 }  
  
 @Override  
public synchronized void onPause() {  
*LOGGER*.d("onPause " + this);  
  
if (!isFinishing()) {  
*LOGGER*.d("Requesting finish");  
 finish();  
 }  
  
handlerThread.quitSafely();  
try {  
handlerThread.join();  
handlerThread = null;  
handler = null;  
 } catch (final InterruptedException e) {  
*LOGGER*.e(e, "Exception!");  
 }  
  
super.onPause();  
 }  
  
 @Override  
public synchronized void onStop() {  
*LOGGER*.d("onStop " + this);  
super.onStop();  
 }  
  
 @Override  
public synchronized void onDestroy() {  
*LOGGER*.d("onDestroy " + this);  
super.onDestroy();  
 }  
  
protected synchronized void runInBackground(final Runnable r) {  
if (handler != null) {  
handler.post(r);  
 }  
 }  
  
 @Override  
public void onRequestPermissionsResult(  
final int requestCode, final String[] permissions, final int[] grantResults) {  
switch (requestCode) {  
case *PERMISSIONS\_REQUEST*: {  
if (grantResults.length > 0  
&& grantResults[0] == PackageManager.*PERMISSION\_GRANTED*&& grantResults[1] == PackageManager.*PERMISSION\_GRANTED*) {  
 setFragment();  
 } else {  
 requestPermission();  
 }  
 }  
 }  
 }  
  
private boolean hasPermission() {  
if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  
return checkSelfPermission(*PERMISSION\_CAMERA*) == PackageManager.*PERMISSION\_GRANTED* && checkSelfPermission(*PERMISSION\_STORAGE*) == PackageManager.*PERMISSION\_GRANTED*;  
 } else {  
return true;  
 }  
 }  
  
private void requestPermission() {  
if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  
if (shouldShowRequestPermissionRationale(*PERMISSION\_CAMERA*) || shouldShowRequestPermissionRationale(*PERMISSION\_STORAGE*)) {  
 Toast.*makeText*(CameraActivity.this, "Camera AND storage permission are required", Toast.*LENGTH\_LONG*).show();  
 }  
 requestPermissions(new String[] {*PERMISSION\_CAMERA*, *PERMISSION\_STORAGE*}, *PERMISSIONS\_REQUEST*);  
 }  
 }  
  
protected void setFragment() {  
final Fragment fragment =  
 CameraConnectionFragment.*newInstance*(  
new CameraConnectionFragment.ConnectionCallback() {  
 @Override  
public void onPreviewSizeChosen(final Size size, final int rotation) {  
 CameraActivity.this.onPreviewSizeChosen(size, rotation);  
 }  
 },  
this,  
 getLayoutId(),  
 getDesiredPreviewFrameSize());  
  
 getFragmentManager()  
 .beginTransaction()  
 .replace(R.id.*container*, fragment)  
 .commit();  
 }  
  
protected void fillBytes(final Plane[] planes, final byte[][] yuvBytes) {  
for (int i = 0; i < planes.length; ++i) {  
final ByteBuffer buffer = planes[i].getBuffer();  
if (yuvBytes[i] == null) {  
*LOGGER*.d("Initializing buffer %d at size %d", i, buffer.capacity());  
 yuvBytes[i] = new byte[buffer.capacity()];  
 }  
 buffer.get(yuvBytes[i]);  
 }  
 }  
  
public boolean isDebug() {  
return debug;  
 }  
  
public void requestRender() {  
final OverlayView overlay = (OverlayView) findViewById(R.id.*debug\_overlay*);  
if (overlay != null) {  
 overlay.postInvalidate();  
 }  
 }  
  
public void addCallback(final OverlayView.DrawCallback callback) {  
final OverlayView overlay = (OverlayView) findViewById(R.id.*debug\_overlay*);  
if (overlay != null) {  
 overlay.addCallback(callback);  
 }  
 }  
  
public void onSetDebug(final boolean debug) {}  
  
 @Override  
public boolean onKeyDown(final int keyCode, final KeyEvent event) {  
if (keyCode == KeyEvent.*KEYCODE\_VOLUME\_DOWN* || keyCode == KeyEvent.*KEYCODE\_VOLUME\_UP*) {  
debug = !debug;  
 requestRender();  
 onSetDebug(debug);  
return true;  
 }  
return super.onKeyDown(keyCode, event);  
 }  
  
protected abstract void onPreviewSizeChosen(final Size size, final int rotation);  
protected abstract int getLayoutId();  
protected abstract Size getDesiredPreviewFrameSize();  
}

**CameraConnectionFragment.java**

package org.tensorflow.demo;  
  
import android.app.Activity;  
import android.app.AlertDialog;  
import android.app.Dialog;  
import android.app.DialogFragment;  
import android.app.Fragment;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.res.Configuration;  
import android.graphics.ImageFormat;  
import android.graphics.Matrix;  
import android.graphics.RectF;  
import android.graphics.SurfaceTexture;  
import android.hardware.camera2.CameraAccessException;  
import android.hardware.camera2.CameraCaptureSession;  
import android.hardware.camera2.CameraCharacteristics;  
import android.hardware.camera2.CameraDevice;  
import android.hardware.camera2.CameraManager;  
import android.hardware.camera2.CaptureRequest;  
import android.hardware.camera2.CaptureResult;  
import android.hardware.camera2.TotalCaptureResult;  
import android.hardware.camera2.params.StreamConfigurationMap;  
import android.media.ImageReader;  
import android.media.ImageReader.OnImageAvailableListener;  
import android.os.Bundle;  
import android.os.Handler;  
import android.os.HandlerThread;  
import android.text.TextUtils;  
import android.util.Size;  
import android.util.SparseIntArray;  
import android.view.LayoutInflater;  
import android.view.Surface;  
import android.view.TextureView;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.Toast;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.Collections;  
import java.util.Comparator;  
import java.util.List;  
import java.util.concurrent.Semaphore;  
import java.util.concurrent.TimeUnit;  
import org.tensorflow.demo.env.Logger;  
import org.tensorflow.demo.R;  
  
public class CameraConnectionFragment extends Fragment {  
private static final Logger *LOGGER* = new Logger();  
  
private static final int *MINIMUM\_PREVIEW\_SIZE* = 320;  
private static final SparseIntArray *ORIENTATIONS* = new SparseIntArray();  
private static final String *FRAGMENT\_DIALOG* = "dialog";  
  
static {  
*ORIENTATIONS*.append(Surface.*ROTATION\_0*, 90);  
*ORIENTATIONS*.append(Surface.*ROTATION\_90*, 0);  
*ORIENTATIONS*.append(Surface.*ROTATION\_180*, 270);  
*ORIENTATIONS*.append(Surface.*ROTATION\_270*, 180);  
 }  
  
private final TextureView.SurfaceTextureListener surfaceTextureListener =  
new TextureView.SurfaceTextureListener() {  
 @Override  
public void onSurfaceTextureAvailable(  
final SurfaceTexture texture, final int width, final int height) {  
 openCamera(width, height);  
 }  
  
 @Override  
public void onSurfaceTextureSizeChanged(  
final SurfaceTexture texture, final int width, final int height) {  
 configureTransform(width, height);  
 }  
  
 @Override  
public boolean onSurfaceTextureDestroyed(final SurfaceTexture texture) {  
return true;  
 }  
  
 @Override  
public void onSurfaceTextureUpdated(final SurfaceTexture texture) {}  
 };  
  
public interface ConnectionCallback {  
void onPreviewSizeChosen(Size size, int cameraRotation);  
 }  
  
private String cameraId;  
private AutoFitTextureView textureView;  
private CameraCaptureSession captureSession;  
private CameraDevice cameraDevice;  
private Integer sensorOrientation;  
private Size previewSize;  
private final CameraDevice.StateCallback stateCallback =  
new CameraDevice.StateCallback() {  
 @Override  
public void onOpened(final CameraDevice cd) {  
cameraOpenCloseLock.release();  
cameraDevice = cd;  
 createCameraPreviewSession();  
 }  
  
 @Override  
public void onDisconnected(final CameraDevice cd) {  
cameraOpenCloseLock.release();  
 cd.close();  
cameraDevice = null;  
 }  
  
 @Override  
public void onError(final CameraDevice cd, final int error) {  
cameraOpenCloseLock.release();  
 cd.close();  
cameraDevice = null;  
final Activity activity = getActivity();  
if (null != activity) {  
 activity.finish();  
 }  
 }  
 };  
  
private HandlerThread backgroundThread;  
private Handler backgroundHandler;  
private ImageReader previewReader;  
private CaptureRequest.Builder previewRequestBuilder;  
private CaptureRequest previewRequest;  
private final Semaphore cameraOpenCloseLock = new Semaphore(1);  
private final OnImageAvailableListener imageListener;  
private final Size inputSize;  
private final int layout;  
private final ConnectionCallback cameraConnectionCallback;  
  
private CameraConnectionFragment(  
final ConnectionCallback connectionCallback,  
final OnImageAvailableListener imageListener,  
final int layout,  
final Size inputSize) {  
this.cameraConnectionCallback = connectionCallback;  
this.imageListener = imageListener;  
this.layout = layout;  
this.inputSize = inputSize;  
 }  
  
private void showToast(final String text) {  
final Activity activity = getActivity();  
if (activity != null) {  
 activity.runOnUiThread(  
new Runnable() {  
 @Override  
public void run() {  
 Toast.*makeText*(activity, text, Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
 }  
  
private static Size chooseOptimalSize(final Size[] choices, final int width, final int height) {  
final int minSize = Math.*max*(Math.*min*(width, height), *MINIMUM\_PREVIEW\_SIZE*);  
final Size desiredSize = new Size(width, height);  
  
boolean exactSizeFound = false;  
final List<Size> bigEnough = new ArrayList<Size>();  
final List<Size> tooSmall = new ArrayList<Size>();  
for (final Size option : choices) {  
if (option.equals(desiredSize)) {  
exactSizeFound = true;  
 }  
  
if (option.getHeight() >= minSize && option.getWidth() >= minSize) {  
 bigEnough.add(option);  
 } else {  
 tooSmall.add(option);  
 }  
 }  
  
*LOGGER*.i("Desired size: " + desiredSize + ", min size: " + minSize + "x" + minSize);  
*LOGGER*.i("Valid preview sizes: [" + TextUtils.*join*(", ", bigEnough) + "]");  
*LOGGER*.i("Rejected preview sizes: [" + TextUtils.*join*(", ", tooSmall) + "]");  
  
if (exactSizeFound) {  
*LOGGER*.i("Exact size match found.");  
return desiredSize;  
 }  
  
if (bigEnough.size() > 0) {  
final Size chosenSize = Collections.*min*(bigEnough, new CompareSizesByArea());  
*LOGGER*.i("Chosen size: " + chosenSize.getWidth() + "x" + chosenSize.getHeight());  
return chosenSize;  
 } else {  
*LOGGER*.e("Couldn't find any suitable preview size");  
return choices[0];  
 }  
 }  
  
public static CameraConnectionFragment newInstance(  
final ConnectionCallback callback,  
final OnImageAvailableListener imageListener,  
final int layout,  
final Size inputSize) {  
return new CameraConnectionFragment(callback, imageListener, layout, inputSize);  
 }  
  
 @Override  
public View onCreateView(  
final LayoutInflater inflater, final ViewGroup container, final Bundle savedInstanceState) {  
return inflater.inflate(layout, container, false);  
 }  
  
 @Override  
public void onViewCreated(final View view, final Bundle savedInstanceState) {  
textureView = (AutoFitTextureView) view.findViewById(R.id.*texture*);  
 }  
  
 @Override  
public void onActivityCreated(final Bundle savedInstanceState) {  
super.onActivityCreated(savedInstanceState);  
 }  
  
 @Override  
public void onResume() {  
super.onResume();  
 startBackgroundThread();  
  
if (textureView.isAvailable()) {  
 openCamera(textureView.getWidth(), textureView.getHeight());  
 } else {  
textureView.setSurfaceTextureListener(surfaceTextureListener);  
 }  
 }  
  
 @Override  
public void onPause() {  
 closeCamera();  
 stopBackgroundThread();  
super.onPause();  
 }  
  
private void setUpCameraOutputs(final int width, final int height) {  
final Activity activity = getActivity();  
final CameraManager manager = (CameraManager) activity.getSystemService(Context.*CAMERA\_SERVICE*);  
try {  
for (final String cameraId : manager.getCameraIdList()) {  
final CameraCharacteristics characteristics = manager.getCameraCharacteristics(cameraId);  
  
final Integer facing = characteristics.get(CameraCharacteristics.*LENS\_FACING*);  
if (facing != null && facing == CameraCharacteristics.*LENS\_FACING\_FRONT*) {  
continue;  
 }  
  
final StreamConfigurationMap map =  
 characteristics.get(CameraCharacteristics.*SCALER\_STREAM\_CONFIGURATION\_MAP*);  
  
if (map == null) {  
continue;  
 }  
  
final Size largest =  
 Collections.*max*(  
 Arrays.*asList*(map.getOutputSizes(ImageFormat.*YUV\_420\_888*)),  
new CompareSizesByArea());  
  
sensorOrientation = characteristics.get(CameraCharacteristics.*SENSOR\_ORIENTATION*);  
  
previewSize =  
*chooseOptimalSize*(  
 map.getOutputSizes(SurfaceTexture.class),  
inputSize.getWidth(),  
inputSize.getHeight());  
  
final int orientation = getResources().getConfiguration().orientation;  
if (orientation == Configuration.*ORIENTATION\_LANDSCAPE*) {  
textureView.setAspectRatio(previewSize.getWidth(), previewSize.getHeight());  
 } else {  
textureView.setAspectRatio(previewSize.getHeight(), previewSize.getWidth());  
 }  
  
 CameraConnectionFragment.this.cameraId = cameraId;  
 }  
 } catch (final CameraAccessException e) {  
*LOGGER*.e(e, "Exception!");  
 } catch (final NullPointerException e) {  
ErrorDialog.*newInstance*(getString(R.string.*camera\_error*))  
 .show(getChildFragmentManager(), *FRAGMENT\_DIALOG*);  
throw new RuntimeException(getString(R.string.*camera\_error*));  
 }  
  
cameraConnectionCallback.onPreviewSizeChosen(previewSize, sensorOrientation);  
 }  
  
private void openCamera(final int width, final int height) {  
 setUpCameraOutputs(width, height);  
 configureTransform(width, height);  
final Activity activity = getActivity();  
final CameraManager manager = (CameraManager) activity.getSystemService(Context.*CAMERA\_SERVICE*);  
try {  
if (!cameraOpenCloseLock.tryAcquire(2500, TimeUnit.*MILLISECONDS*)) {  
throw new RuntimeException("Time out waiting to lock camera opening.");  
 }  
 manager.openCamera(cameraId, stateCallback, backgroundHandler);  
 } catch (final CameraAccessException e) {  
*LOGGER*.e(e, "Exception!");  
 } catch (final InterruptedException e) {  
throw new RuntimeException("Interrupted while trying to lock camera opening.", e);  
 }  
 }  
  
private void closeCamera() {  
try {  
cameraOpenCloseLock.acquire();  
if (null != captureSession) {  
captureSession.close();  
captureSession = null;  
 }  
if (null != cameraDevice) {  
cameraDevice.close();  
cameraDevice = null;  
 }  
if (null != previewReader) {  
previewReader.close();  
previewReader = null;  
 }  
 } catch (final InterruptedException e) {  
throw new RuntimeException("Interrupted while trying to lock camera closing.", e);  
 } finally {  
cameraOpenCloseLock.release();  
 }  
 }  
  
private void startBackgroundThread() {  
backgroundThread = new HandlerThread("ImageListener");  
backgroundThread.start();  
backgroundHandler = new Handler(backgroundThread.getLooper());  
 }  
  
private void stopBackgroundThread() {  
backgroundThread.quitSafely();  
try {  
backgroundThread.join();  
backgroundThread = null;  
backgroundHandler = null;  
 } catch (final InterruptedException e) {  
*LOGGER*.e(e, "Exception!");  
 }  
 }  
  
private final CameraCaptureSession.CaptureCallback captureCallback =  
new CameraCaptureSession.CaptureCallback() {  
 @Override  
public void onCaptureProgressed(  
final CameraCaptureSession session,  
final CaptureRequest request,  
final CaptureResult partialResult) {}  
  
 @Override  
public void onCaptureCompleted(  
final CameraCaptureSession session,  
final CaptureRequest request,  
final TotalCaptureResult result) {}  
 };  
  
private void createCameraPreviewSession() {  
try {  
final SurfaceTexture texture = textureView.getSurfaceTexture();  
assert texture != null;  
  
texture.setDefaultBufferSize(previewSize.getWidth(), previewSize.getHeight());  
  
final Surface surface = new Surface(texture);  
  
previewRequestBuilder = cameraDevice.createCaptureRequest(CameraDevice.*TEMPLATE\_PREVIEW*);  
previewRequestBuilder.addTarget(surface);  
  
*LOGGER*.i("Opening camera preview: " + previewSize.getWidth() + "x" + previewSize.getHeight());  
  
previewReader =  
 ImageReader.*newInstance*(  
previewSize.getWidth(), previewSize.getHeight(), ImageFormat.*YUV\_420\_888*, 2);  
  
previewReader.setOnImageAvailableListener(imageListener, backgroundHandler);  
previewRequestBuilder.addTarget(previewReader.getSurface());  
  
cameraDevice.createCaptureSession(  
 Arrays.*asList*(surface, previewReader.getSurface()),  
new CameraCaptureSession.StateCallback() {  
  
 @Override  
public void onConfigured(final CameraCaptureSession cameraCaptureSession) {  
if (null == cameraDevice) {  
return;  
 }  
  
captureSession = cameraCaptureSession;  
try {  
previewRequestBuilder.set(  
 CaptureRequest.*CONTROL\_AF\_MODE*,  
 CaptureRequest.*CONTROL\_AF\_MODE\_CONTINUOUS\_PICTURE*);  
previewRequestBuilder.set(  
 CaptureRequest.*CONTROL\_AE\_MODE*, CaptureRequest.*CONTROL\_AE\_MODE\_ON\_AUTO\_FLASH*);  
  
previewRequest = previewRequestBuilder.build();  
captureSession.setRepeatingRequest(  
previewRequest, captureCallback, backgroundHandler);  
 } catch (final CameraAccessException e) {  
*LOGGER*.e(e, "Exception!");  
 }  
 }  
  
 @Override  
public void onConfigureFailed(final CameraCaptureSession cameraCaptureSession) {  
 showToast("Failed");  
 }  
 },  
null);  
 } catch (final CameraAccessException e) {  
*LOGGER*.e(e, "Exception!");  
 }  
 }  
  
private void configureTransform(final int viewWidth, final int viewHeight) {  
final Activity activity = getActivity();  
if (null == textureView || null == previewSize || null == activity) {  
return;  
 }  
final int rotation = activity.getWindowManager().getDefaultDisplay().getRotation();  
final Matrix matrix = new Matrix();  
final RectF viewRect = new RectF(0, 0, viewWidth, viewHeight);  
final RectF bufferRect = new RectF(0, 0, previewSize.getHeight(), previewSize.getWidth());  
final float centerX = viewRect.centerX();  
final float centerY = viewRect.centerY();  
if (Surface.*ROTATION\_90* == rotation || Surface.*ROTATION\_270* == rotation) {  
 bufferRect.offset(centerX - bufferRect.centerX(), centerY - bufferRect.centerY());  
 matrix.setRectToRect(viewRect, bufferRect, Matrix.ScaleToFit.*FILL*);  
final float scale =  
 Math.*max*(  
 (float) viewHeight / previewSize.getHeight(),  
 (float) viewWidth / previewSize.getWidth());  
 matrix.postScale(scale, scale, centerX, centerY);  
 matrix.postRotate(90 \* (rotation - 2), centerX, centerY);  
 } else if (Surface.*ROTATION\_180* == rotation) {  
 matrix.postRotate(180, centerX, centerY);  
 }  
textureView.setTransform(matrix);  
 }  
  
static class CompareSizesByArea implements Comparator<Size> {  
 @Override  
public int compare(final Size lhs, final Size rhs) {  
return Long.*signum*(  
 (long) lhs.getWidth() \* lhs.getHeight() - (long) rhs.getWidth() \* rhs.getHeight());  
 }  
 }  
  
public static class ErrorDialog extends DialogFragment {  
private static final String *ARG\_MESSAGE* = "message";  
  
public static ErrorDialog newInstance(final String message) {  
final ErrorDialog dialog = new ErrorDialog();  
final Bundle args = new Bundle();  
 args.putString(*ARG\_MESSAGE*, message);  
 dialog.setArguments(args);  
return dialog;  
 }  
  
 @Override  
public Dialog onCreateDialog(final Bundle savedInstanceState) {  
final Activity activity = getActivity();  
return new AlertDialog.Builder(activity)  
 .setMessage(getArguments().getString(*ARG\_MESSAGE*))  
 .setPositiveButton(  
 android.R.string.*ok*,  
new DialogInterface.OnClickListener() {  
 @Override  
public void onClick(final DialogInterface dialogInterface, final int i) {  
 activity.finish();  
 }  
 })  
 .create();  
 }  
 }  
}

**Classifier.java**

package org.tensorflow.demo;  
  
import android.graphics.Bitmap;  
import android.graphics.RectF;  
import java.util.List;  
  
public interface Classifier {  
public class Recognition {  
private final String id;  
private final String title;  
private final Float confidence;  
private RectF location;  
  
public Recognition(  
final String id, final String title, final Float confidence, final RectF location) {  
this.id = id;  
this.title = title;  
this.confidence = confidence;  
this.location = location;  
 }  
  
public String getId() {  
return id;  
 }  
  
public String getTitle() {  
return title;  
 }  
  
public Float getConfidence() {  
return confidence;  
 }  
  
public RectF getLocation() {  
return new RectF(location);  
 }  
  
public void setLocation(RectF location) {  
this.location = location;  
 }  
  
 @Override  
public String toString() {  
 String resultString = "";  
if (id != null) {  
 resultString += "[" + id + "] ";  
 }  
  
if (title != null) {  
 resultString += title + " ";  
 }  
  
if (confidence != null) {  
 resultString += String.*format*("(%.1f%%) ", confidence \* 100.0f);  
 }  
  
if (location != null) {  
 resultString += location + " ";  
 }  
  
return resultString.trim();  
 }  
 }  
  
 List<Recognition> recognizeImage(Bitmap bitmap);  
  
void enableStatLogging(final boolean debug);  
  
 String getStatString();  
  
void close();  
}

**ClassifierActivity.java**

package org.tensorflow.demo;  
  
import android.graphics.Bitmap;  
import android.graphics.Bitmap.Config;  
import android.graphics.Canvas;  
import android.graphics.Matrix;  
import android.graphics.Paint;  
import android.graphics.Typeface;  
import android.media.Image;  
import android.media.Image.Plane;  
import android.media.ImageReader;  
import android.media.ImageReader.OnImageAvailableListener;  
import android.os.SystemClock;  
import android.os.Trace;  
import android.util.Size;  
import android.util.TypedValue;  
import android.view.Display;  
import java.util.List;  
import java.util.Vector;  
import org.tensorflow.demo.OverlayView.DrawCallback;  
import org.tensorflow.demo.env.BorderedText;  
import org.tensorflow.demo.env.ImageUtils;  
import org.tensorflow.demo.env.Logger;  
import org.tensorflow.demo.R;  
  
public class ClassifierActivity extends CameraActivity implements OnImageAvailableListener {  
private static final Logger *LOGGER* = new Logger();  
  
private static final int *INPUT\_SIZE* = 299;  
private static final int *IMAGE\_MEAN* = 128;  
private static final float *IMAGE\_STD* = 128.0f;  
private static final String *INPUT\_NAME* = "Mul:0";  
private static final String *OUTPUT\_NAME* = "final\_result";  
  
private static final String *MODEL\_FILE* = "file:///android\_asset/rounded\_graph.pb";  
private static final String *LABEL\_FILE* = "file:///android\_asset/retrained\_labels.txt";  
  
private static final boolean *SAVE\_PREVIEW\_BITMAP* = false;  
  
private static final boolean *MAINTAIN\_ASPECT* = true;  
  
private static final Size *DESIRED\_PREVIEW\_SIZE* = new Size(640, 480);  
  
private Classifier classifier;  
  
private Integer sensorOrientation;  
  
private int previewWidth = 0;  
private int previewHeight = 0;  
private byte[][] yuvBytes;  
private int[] rgbBytes = null;  
private Bitmap rgbFrameBitmap = null;  
private Bitmap croppedBitmap = null;  
  
private Bitmap cropCopyBitmap;  
  
private boolean computing = false;  
  
private Matrix frameToCropTransform;  
private Matrix cropToFrameTransform;  
  
private ResultsView resultsView;  
  
private BorderedText borderedText;  
  
private long lastProcessingTimeMs;  
  
 @Override  
protected int getLayoutId() {  
return R.layout.*camera\_connection\_fragment*;  
 }  
  
 @Override  
protected Size getDesiredPreviewFrameSize() {  
return *DESIRED\_PREVIEW\_SIZE*;  
 }  
  
private static final float *TEXT\_SIZE\_DIP* = 10;  
  
 @Override  
public void onPreviewSizeChosen(final Size size, final int rotation) {  
final float textSizePx =  
 TypedValue.*applyDimension*(  
 TypedValue.*COMPLEX\_UNIT\_DIP*, *TEXT\_SIZE\_DIP*, getResources().getDisplayMetrics());  
borderedText = new BorderedText(textSizePx);  
borderedText.setTypeface(Typeface.*MONOSPACE*);  
  
classifier =  
 TensorFlowImageClassifier.*create*(  
 getAssets(),  
*MODEL\_FILE*,  
*LABEL\_FILE*,  
*INPUT\_SIZE*,  
*IMAGE\_MEAN*,  
*IMAGE\_STD*,  
*INPUT\_NAME*,  
*OUTPUT\_NAME*);  
  
resultsView = (ResultsView) findViewById(R.id.*results*);  
previewWidth = size.getWidth();  
previewHeight = size.getHeight();  
  
final Display display = getWindowManager().getDefaultDisplay();  
final int screenOrientation = display.getRotation();  
  
*LOGGER*.i("Sensor orientation: %d, Screen orientation: %d", rotation, screenOrientation);  
  
sensorOrientation = rotation + screenOrientation;  
  
*LOGGER*.i("Initializing at size %dx%d", previewWidth, previewHeight);  
rgbBytes = new int[previewWidth \* previewHeight];  
rgbFrameBitmap = Bitmap.*createBitmap*(previewWidth, previewHeight, Config.*ARGB\_8888*);  
croppedBitmap = Bitmap.*createBitmap*(*INPUT\_SIZE*, *INPUT\_SIZE*, Config.*ARGB\_8888*);  
  
frameToCropTransform =  
 ImageUtils.*getTransformationMatrix*(  
previewWidth, previewHeight,  
*INPUT\_SIZE*, *INPUT\_SIZE*,  
sensorOrientation, *MAINTAIN\_ASPECT*);  
  
cropToFrameTransform = new Matrix();  
frameToCropTransform.invert(cropToFrameTransform);  
  
yuvBytes = new byte[3][];  
  
 addCallback(  
new DrawCallback() {  
 @Override  
public void drawCallback(final Canvas canvas) {  
 renderDebug(canvas);  
 }  
 });  
 }  
  
 @Override  
public void onImageAvailable(final ImageReader reader) {  
 Image image = null;  
  
try {  
 image = reader.acquireLatestImage();  
  
if (image == null) {  
return;  
 }  
  
if (computing) {  
 image.close();  
return;  
 }  
computing = true;  
  
 Trace.*beginSection*("imageAvailable");  
  
final Plane[] planes = image.getPlanes();  
 fillBytes(planes, yuvBytes);  
  
final int yRowStride = planes[0].getRowStride();  
final int uvRowStride = planes[1].getRowStride();  
final int uvPixelStride = planes[1].getPixelStride();  
 ImageUtils.*convertYUV420ToARGB8888*(  
yuvBytes[0],  
yuvBytes[1],  
yuvBytes[2],  
previewWidth,  
previewHeight,  
 yRowStride,  
 uvRowStride,  
 uvPixelStride,  
rgbBytes);  
  
 image.close();  
 } catch (final Exception e) {  
if (image != null) {  
 image.close();  
 }  
*LOGGER*.e(e, "Exception!");  
 Trace.*endSection*();  
return;  
 }  
  
rgbFrameBitmap.setPixels(rgbBytes, 0, previewWidth, 0, 0, previewWidth, previewHeight);  
final Canvas canvas = new Canvas(croppedBitmap);  
 canvas.drawBitmap(rgbFrameBitmap, frameToCropTransform, null);  
  
if (*SAVE\_PREVIEW\_BITMAP*) {  
 ImageUtils.*saveBitmap*(croppedBitmap);  
 }  
  
 runInBackground(  
new Runnable() {  
 @Override  
public void run() {  
final long startTime = SystemClock.*uptimeMillis*();  
final List<Classifier.Recognition> results = classifier.recognizeImage(croppedBitmap);  
lastProcessingTimeMs = SystemClock.*uptimeMillis*() - startTime;  
  
cropCopyBitmap = Bitmap.*createBitmap*(croppedBitmap);  
resultsView.setResults(results);  
 requestRender();  
computing = false;  
 }  
 });  
  
 Trace.*endSection*();  
 }  
  
 @Override  
public void onSetDebug(boolean debug) {  
classifier.enableStatLogging(debug);  
 }  
  
private void renderDebug(final Canvas canvas) {  
if (!isDebug()) {  
return;  
 }  
final Bitmap copy = cropCopyBitmap;  
if (copy != null) {  
final Matrix matrix = new Matrix();  
final float scaleFactor = 2;  
 matrix.postScale(scaleFactor, scaleFactor);  
 matrix.postTranslate(  
 canvas.getWidth() - copy.getWidth() \* scaleFactor,  
 canvas.getHeight() - copy.getHeight() \* scaleFactor);  
 canvas.drawBitmap(copy, matrix, new Paint());  
  
final Vector<String> lines = new Vector<String>();  
if (classifier != null) {  
 String statString = classifier.getStatString();  
 String[] statLines = statString.split("\n");  
for (String line : statLines) {  
 lines.add(line);  
 }  
 }  
  
 lines.add("Frame: " + previewWidth + "x" + previewHeight);  
 lines.add("Crop: " + copy.getWidth() + "x" + copy.getHeight());  
 lines.add("View: " + canvas.getWidth() + "x" + canvas.getHeight());  
 lines.add("Rotation: " + sensorOrientation);  
 lines.add("Inference time: " + lastProcessingTimeMs + "ms");  
  
borderedText.drawLines(canvas, 10, canvas.getHeight() - 10, lines);  
 }  
 }  
}

**Display.java**

package org.tensorflow.demo;  
  
import android.content.Context;  
import android.content.Intent;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.os.Bundle;  
import android.app.Activity;  
import android.text.Html;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.ImageView;  
import android.widget.TextView;  
import android.widget.Toolbar;  
  
import org.tensorflow.demo.R;  
import org.tensorflow.demo.RecognitionScoreView;  
  
import static java.security.AccessController.*getContext*;  
  
public class Display extends Activity {  
public static String *name1* = "none";  
  
 @Override  
protected void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_display*);  
  
 getActionBar().setDisplayHomeAsUpEnabled(true);  
  
 Intent intent=getIntent();  
 String name = intent.getStringExtra("name");  
*name1* = name;  
  
 TextView result\_name = (TextView) findViewById(R.id.*result\_name*);  
 result\_name.setText(name);  
  
 String desc = Desc(name);  
  
 TextView result\_desc = (TextView) findViewById(R.id.*result\_desc*);  
 result\_desc.setText(Html.*fromHtml*(desc));  
  
int bmp = img(name);  
 ImageView result\_img = (ImageView) findViewById(R.id.*result\_img*);  
 result\_img.setImageResource(bmp);  
 }  
  
 @Override  
public boolean onOptionsItemSelected(MenuItem item) {  
switch (item.getItemId()) {  
case android.R.id.*home*:  
this.onBackPressed();  
return true;  
default:  
return super.onOptionsItemSelected(item);  
 }  
 }  
  
public void dispMsg(View v){  
 Intent intent1=new Intent(this, More.class);  
 intent1.putExtra("name",*name1*);  
 startActivity(intent1);  
 }  
  
private String Desc(String name){  
 String desc = "";  
if(name.equals("dog"))  
 desc = "<b>Scientific Name:</b> Canis Lupus Familiaris<br><b>Genus:</b> Canis<br><b>Family:</b> Canidae<br><b>Order:</b> Carnivora<br><b>Class:</b> Mammalia<br><b>Phylum:</b> Chordata<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 10-13 years";  
else if(name.equals("human"))  
 desc = "<b>Scientific Name:</b> Homo Sapiens<br><b>Genus:</b> Homo<br><b>Family:</b> Hominidae<br><b>Order:</b> Primates<br><b>Class:</b> Mammalia<br><b>Phylum:</b> Chordata<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 79-100 years";  
else if(name.equals("ant"))  
 desc = "<b>Scientific Name:</b> Iridomyrmex Purpureus<br><b>Genus:</b> Iridomyrmex<br><b>Family:</b> Formicidae<br><b>Order:</b> Hymenoptera<br><b>Class:</b> Insecta<br><b>Phylum:</b> Arthropoda<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 4-15 years";  
else if(name.equals("rat"))  
 desc = "<b>Scientific Name:</b> Rattus Rattus / Norvegicus<br><b>Genus:</b>Rattus<br><b>Family:</b> Muridae<br><b>Order:</b> Rodentia<br><b>Class:</b> Mammalia<br><b>Phylum:</b> Chordata<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 2 years";  
else if(name.equals("crow"))  
 desc = "<b>Scientific Name:</b> Corvus Brachyrhynchos<br><b>Genus:</b> Corvus<br><b>Family:</b> Corvidae<br><b>Order:</b> Passeriformes<br><b>Class:</b> Aves<br><b>Phylum:</b> Chordata<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 10-15 years";  
else if(name.equals("cow"))  
 desc = "<b>Scientific Name:</b> Bos Taurus<br><b>Genus:</b> Bos<br><b>Family:</b> Bovidae<br><b>Order:</b> Artiodactyla<br><b>Class:</b> Mammalia<br><b>Phylum:</b> Chordata<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 18-22 years";  
else if(name.equals("sparrow"))  
 desc = "<b>Scientific Name:</b> Passer Domesticus<br><b>Genus:</b> Passer<br><b>Family:</b> Passeridae<br><b>Order:</b> Passeriformes<br><b>Class:</b> Aves<br><b>Phylum:</b> Chordata<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 3 years";  
else if(name.equals("honey bee"))  
 desc = "<b>Scientific Name:</b> Apis Nearctica<br><b>Genus:</b> Apis<br><b>Family:</b> Apidae<br><b>Order:</b> Hymenoptera<br><b>Class:</b> Insecta<br><b>Phylum:</b> Arthropoda<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 122-152 days";  
else if(name.equals("horse"))  
 desc = "<b>Scientific Name:</b> Equus Ferus Caballus<br><b>Genus:</b> Equus<br><b>Family:</b> Equidae<br><b>Order:</b> Perissodactyla<br><b>Class:</b> Mammalia<br><b>Phylum:</b> Chordata<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 25-30 years";  
else if(name.equals("mosquito"))  
 desc = "<b>Scientific Name:</b> Aedes Aegypti<br><b>Genus:</b> Aedes<br><b>Family:</b> Culicidae<br><b>Order:</b> Diptera<br><b>Class:</b> Insecta<br><b>Phylum:</b> Arthropoda<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 42-56 years";  
else if(name.equals("rabbit"))  
 desc = "<b>Scientific Name:</b> Oryctolagus Cuniculus<br><b>Genus:</b> Oryctolagus<br><b>Family:</b> Leporidae<br><b>Order:</b> Lagomorpha<br><b>Class:</b> Mammalia<br><b>Phylum:</b> Chordata<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 3 years";  
else if(name.equals("cat"))  
 desc = "<b>Scientific Name:</b> Felis Catus<br><b>Genus:</b> Felis<br><b>Family:</b> Felidae<br><b>Order:</b> Carnivora<br><b>Class:</b> Mammalia<br><b>Phylum:</b> Chordata<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 4-5 years";  
else if(name.equals("buffalo"))  
 desc = "<b>Scientific Name:</b> Syncerus Caffer<br><b>Genus:</b> Syncerus<br><b>Family:</b> Bovidae<br><b>Order:</b> Artiodactyla<br><b>Class:</b> Mammalia<br><b>Phylum:</b> Chordata<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 12 years";  
else if(name.equals("butterfly"))  
 desc = "<b>Scientific Name:</b> Danaus Plexippus<br><b>Genus:</b> Danaus<br><b>Family:</b> Hesperioidae<br><b>Order:</b> Lepidoptera<br><b>Class:</b> Insecta<br><b>Phylum:</b> Arthropoda<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 12 months";  
else if(name.equals("house fly"))  
 desc = "<b>Scientific Name:</b> Musca Domestica<br><b>Genus:</b> Musca<br><b>Family:</b> Muscidae<br><b>Order:</b> Diptera<br><b>Class:</b> Insecta<br><b>Phylum:</b> Arthropoda<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 28 days";  
else if(name.equals("parrot"))  
 desc = "<b>Scientific Name:</b> Micropsitta Pusio<br><b>Genus:</b> Micropsitta<br><b>Family:</b> Psittaculidae<br><b>Order:</b> Psittaciformes<br><b>Class:</b> Aves<br><b>Phylum:</b> Chordata<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 50-95 years";  
else if(name.equals("pig"))  
 desc = "<b>Scientific Name:</b> Sus Scrofa Domesticus<br><b>Genus:</b> Sus<br><b>Family:</b> Suidae<br><b>Order:</b> Artiodactyla<br><b>Class:</b> Mammalia<br><b>Phylum:</b> Chordata<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 8 years";  
else if(name.equals("pigeon"))  
 desc = "<b>Scientific Name:</b> Columba Livia Domestica<br><b>Genus:</b> Columba<br><b>Family:</b> Columbidae<br><b>Order:</b> Columbiformes<br><b>Class:</b> Aves<br><b>Phylum:</b> Chordata<br><b>Kingdom:</b> Animalia<br><b>Lifespan:</b> 6 years";  
else  
desc = "New Specie Encountered";  
return desc;  
 }  
  
private int img(String name){  
int bmp = 0;  
if(name.equals("dog"))  
 bmp = R.mipmap.*dog*;  
else if(name.equals("human"))  
 bmp = R.mipmap.*human*;  
else if(name.equals("ant"))  
 bmp = R.mipmap.*ant*;  
else if(name.equals("rat"))  
 bmp = R.mipmap.*rat*;  
else if(name.equals("crow"))  
 bmp = R.mipmap.*crow*;  
else if(name.equals("cow"))  
 bmp = R.mipmap.*cow*;  
else if(name.equals("sparrow"))  
 bmp = R.mipmap.*sparrow*;  
else if(name.equals("honey bee"))  
 bmp = R.mipmap.*honey\_bee*;  
else if(name.equals("horse"))  
 bmp = R.mipmap.*horse*;  
else if(name.equals("mosquito"))  
 bmp = R.mipmap.*mosquito*;  
else if(name.equals("rabbit"))  
 bmp = R.mipmap.*rabbit*;  
else if(name.equals("cat"))  
 bmp = R.mipmap.*cat*;  
else if(name.equals("buffalo"))  
 bmp = R.mipmap.*buffalo*;  
else if(name.equals("butterfly"))  
 bmp = R.mipmap.*butterfly*;  
else if(name.equals("house fly"))  
 bmp = R.mipmap.*house\_fly*;  
else if(name.equals("parrot"))  
 bmp = R.mipmap.*parrot*;  
else if(name.equals("pig"))  
 bmp = R.mipmap.*pig*;  
else if(name.equals("pigeon"))  
 bmp = R.mipmap.*pigeon*;  
else  
bmp = R.drawable.*ic\_action\_info*;  
return bmp;  
 }  
}

**More.java**

package org.tensorflow.demo;  
  
import android.app.ActionBar;  
import android.content.Intent;  
import android.os.Bundle;  
import android.app.Activity;  
import android.text.Html;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.TextView;  
import android.widget.Toolbar;  
  
public class More extends Activity {  
  
private final String name1 = "none";  
  
 @Override  
protected void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_more*);  
  
 getActionBar().setDisplayHomeAsUpEnabled(true);  
  
 Intent intent = getIntent();  
 String name = intent.getStringExtra("name");  
  
 TextView more\_name = (TextView) findViewById(R.id.*more\_name*);  
 more\_name.setText(name);  
  
 String desc = Desc(name);  
  
 TextView more\_desc = (TextView) findViewById(R.id.*more\_desc*);  
 more\_desc.setText(Html.*fromHtml*(desc));  
 }  
  
 @Override  
public boolean onOptionsItemSelected(MenuItem item) {  
switch (item.getItemId()) {  
case android.R.id.*home*:  
this.onBackPressed();  
return true;  
default:  
return super.onOptionsItemSelected(item);  
 }  
 }  
  
private String Desc(String name){  
 String desc = "";  
if(name.equals("dog"))  
 desc = "- Dogs are not color blind.<br>- Most popular breed of dog is Labrador.<br>- Dog have superior hearing than humans.<br>- Dogs have a remarkable sense of smell.<br>- Dogs communicate with each other using chemical cues, body language and vocalisations.<br>- Dogs have three eyelids to protect the eye from dirt and dust.<br>- Dogs show empathy for others.<br>- Dogs can be as smart as a 2-years old human child.";  
else if(name.equals("human"))  
 desc = "- Every human have unique tongue print.<br>- The total length of all the nerves in human body is 75 km.<br>- A human makes approximately 20,000 breaths per day.<br>- A human eye can distinguish up to 10 million different colors.<br>- Human ears keep on growing throughout the lives with almmost unbelievable speed.<br>- Human heart beats 35 millions times a year.<br>- Men have fewer taste buds on the surface of tongues than women.<br>- Around 10,000 chemical reactions occur every second in the human brain.";  
else if(name.equals("ant"))  
 desc = "- Ants are capable of carrying objects 50 times their own body weight with their mandibles.<br>- Soldier ants use their heads to plug the entrances to their nests and keep intruders from gaining access.<br>- Certain ant species defend plants in exchange for food and shelter.<br>- The total biomass of all the ants on earth is roughly equal to the totalbiomass of all the people on earth.<br>- Ants sometimes herd or tend to insects of other species, like aphids or leafhoppers.<br>- Ants will enslave other ants, keeping them captive and making them do work for the colony.<br>- Ants lived alongside the dinosaurs.<br>- Ants started farming long before humans.<br>- Some ants form 'super-colonies', massive communities of ants that can stretch for thousands of miles.<br>- Ants follow scent trails laid by scout ants to gather food.";  
else if(name.equals("rat"))  
 desc = "- Rats take care of injured and sick rats in their group.<br>- Without companionship rats tend to become lonely and depressed.<br>- Rats have excellent memories. Once they learn a navigation route, they won't forget it.<br>- When happy, rats have been observed to chatter or grid their teeth or vibrating eyes.<br>- Rats make happy 'laughter' sounds when they play.<br>- Rats succumb to peer-pressure, just like humans.<br>- Although very curious animals, rats are also shy.<br>- A rat can go longer than a camel without water.<br>- Rats' tails help them to balance, communicate and regulate their body temperature.<br>- The rat is the first of the twelve animals of the Chinese zodiac.<br>- Rats are recognised at the vehicle of Lord Ganesh in Indian tradition.";  
else if(name.equals("crow"))  
 desc = "- There are about 45 species of crow worldwide known by a variety of names, including treepies, corbies, nutcrackers, bushpies, choughs and pica pica.<br>- Mating crows will often remain together for years and some until by death.<br>- Corvides are absolutely fearless, particularly when chasing bald or golden eagles.<br>- Almost all corvids have been observed using tools, and the Raven can be taught to speak basic human language.<br>- Crows are emotional animals, too.<br>- Crows are considered song-birds and posses a deep repertoire of melodies.<br>- Crows have an excellent memory.<br>- Magpies, Choughs and Nutcrackers are all basically modified crows.<br>- Crows, rooks, Ravens and Jackdaws are the most successfull members of the group except in Central and Southern America where only Jays have reached.";  
else if(name.equals("cow"))  
 desc = "- Like humans, cows form close friendships and choose to spend much of their time with 2-4 preferred individuals.<br>- Cows display emotions and have been shown to produce more milk when they are treated better and as individuals.<br>- Cows get excited when they solve problems.<br>- Cows show their excitement when let out into a field after long periods confined indoors.<br>- Cows like to sleep close to their families and sleeping arrangements are determined by individuals' rank in the social hierarchy.<br>- Cows are devotional mothers and are known to walk fro miles to find their calves.<br>- Cows are extremely curious and inquisitive animals which investigate everything.<br>- Like many other grazing animals cows have one stomach which is divided into four compartments or chambers.";  
else if(name.equals("sparrow"))  
 desc = "- Sparrow is a very small bird. It can reach 4 to 8 inches in length and 0.8 to 1.4 ounches in weight.<br>- Sparrow has stout body, covered with brown, black and white feathers. It wings are rounded.<br>- Males and Females can be distinguished by the feather coloration: males have reddish back and black bib, while females have brown blacks with strips.<br>- Sparrows are very social and they in colonies called flocks.<br>- Sparrows are carnivores by nature.<br>- Sparrow usually fly at the speed of 24 miles per hour.<br>- Although sparrows do not belong to the group of water birds, they can swim very fast to escape from the predators.<br>- Main predators of sparrows are dogs, cats, foxes and snakes.<br>- Sparrows are not territorial animals, but they will aggressively protect their nest from other sparrows.<br>- Sparrows usually build nest under the roofs, under bridges and in tree hollows.<br>- Male is responsible for building of the nest.<br>- Sparrows are allegedly monogamous.<br>- Sparrows have several broods each year.<br>- Sparrows can survive between 4-5 years in wild.";  
else if(name.equals("honey bee"))  
 desc = "- Honey bees can fly at speeds of up to 15 miles per hour.<br>- A honey bee colony can contain up to 60,000 bees at its peak.<br>- A single honey bee worker produces about 1/12th of a teaspoon of honey in her lifetime.<br>- A queen honey bee stores a lifetime supply of sperm.<br>- The queen honey bee lays up to 1,500 eggs per day, and may lay up to 1 million in her lifetime.<br>- The honey bee uses the most complex symbolic language of any animal on earth outside of the primate family.<br>- Drones, the only male honey bees, bie immediately after mating.<br>- Honey bees maintain a constant temperature of about 93 degree F within the hive year-round.<br>- Honey bees produce beeswax from special glans on their abdomens.<br>- An industrious worker bee may visit 2,000 flowers per day.";  
else if(name.equals("horse"))  
 desc = "- Horses can sleep both lying down and standing up.<br>- Horses can run shortly after birth.<br>- Domestic horses have a lifespan of around 25 years.<br>- A 19th century horse names 'Old Billy' is said to have lived 62 years.<br>- Horses have around 205 bones in their skeleton.<br>- Horses have been domesticated for iver 5000 years.<br>- Horses are herbivores.<br>- Horses have bigger eyes than any other mammal that lives on land.<br>- Because horse’s eyes are on the side of their head they are capable of seeing nearly 360 degrees at one time.<br>- Horses gallop at around 44 kph (27 mph).<br>- The fastest recorded sprinting speed of a horse was 88 kph (55 mph).<br>- Estimates suggest that there are around 60 million horses in the world.<br>- Scientists believe that horses have evolved over the past 50 million years from much smaller creatures.<br>- A male horse is called a stallion.<br>- A female horse is called a mare.<br>- A young male horse is called a colt.<br>- A young female horse is called a filly.<br>- Ponies are small horses.";  
else if(name.equals("mosquito"))  
 desc = "- Mosquito are the deadliest animals on earth.<br>- Only female mosquitoes bite humans and animals; male feed on flower nectar.<br>- Some mosquitos don't bite humans, preferring other hosts like amphibians or birds.<br>- Mosquitoes fly at speeds between 1-1.5 miles per hour.<br>- A mosquito's wings beat 300-600 times per second.<br>- Mosquito mates synchronize their wing beats to perform a lover's duet.<br>- Salt marsh mosquitoes may travel up to 100 miles from their larval breeding habitat.<br>- All mosquitoes require water to breed.<br>- An adult mosquito may live 5-6 months.<br>- Mosquitoes can detect carbon dioxide from 75 feet away.";  
else if(name.equals("rabbit"))  
 desc = "- A female rabbit is called a doe.<br>- A male rabbit is called a buck.<br>- A young rabbit is called a kit (or kitten).<br>- Rabbits live in groups.<br>- The European rabbit lives underground, in burrows. A group of burrows is known as a warren.<br>- More than half of the world’s rabbits live in North America.<br>- Rabbits have long ears which can be as long as 10 cm (4 in).<br>- Rabbits have a lifespan of around 10 years.<br>- Rabbits are herbivores (plant eaters).<br>- Pet rabbits that live inside are often referred to as ‘house rabbits’.<br>- Rabbits reproduce very quickly. This can be a major headache for people living in agricultural areas where rabbits are seen as pests.<br>- Rabbits are born with their eyes closed and without fur.";  
else if(name.equals("cat"))  
 desc = "- Cats are one of, if not the most, popular pet in the world.<br>- There are over 500 million domestic cats in the world.<br>- Cats and humans have been associated for nearly 10000 years.<br>- Cats conserve energy by sleeping for an average of 13 to14 hours a day.<br>- Cats have flexible bodies and teeth adapted for hunting small animals such as mice and rats.<br>- A group of cats is called a clowder, a male cat is called a tom, a female cat is called a molly or queen while young cats are called kittens.<br>- Domestic cats usually weigh around 4 kilograms (8 lb 13 oz) to 5 kilograms (11 lb 0 oz).<br>- The heaviest domestic cat on record is 21.297 kilograms (46 lb 15.2 oz).<br>- Cats can be lethal hunters and very sneaky, when they walk their back paws step almost exactly in the same place as the front paws did beforehand, this keeps noise to a minimum and limits visible tracks.<br>- Cats have powerful night vision, allowing them to see at light levels six times lower than what a human needs in order to see.<br>- Cats also have excellent hearing and a powerful sense of smell.<br>- Older cats can at times act aggressively towards kittens.<br>- Domestic cats love to play, this is especially true with kittens who love to chase toys and play fight. Play fighting among kittens may be a way for them to practice and learn skills for hunting and fighting.<br>- On average cats live for around 12 to 15 years.<br>- Cats spend a large amount of time licking their coats to keep them clean.<br>- Feral cats are often seen as pests and threats to native animals.";  
else if(name.equals("buffalo"))  
 desc = "- Buffalo are the largest animals found in north America and can grow to 6-7 feet long, weighing up to 2,000lbs.<br>- Buffalo live mostly on the plains, but some are known to inhabit river valleys as well as forests.<br>- Plains Indians used every part of the buffalo to help them survive.<br>- Buffalo can be 6 to 7 feet long, stand from 5 to 6 feet tall and weigh up to 2,000 pounds.<br>- Buffalo’s tail can be almost three feet long, and they use it to swap at pesky bugs.<br>- Thick brown fur keeps them warm on the frigid and frozen plains during the winter months.<br>- Buffalo also has a large hump on its back.<br>- Buffalo are herbivores and feed on about 60 pounds of plant material and grasses a day.<br>- Buffalo are considered to be an adult when they reach 3 years old.";  
else if(name.equals("butterfly"))  
 desc = "- Butterflies are insects.<br>- A butterfly’s lifecycle is made up of four parts, egg, larva (caterpillars), pupa (chrysalis) and adult.<br>- Butterflies attach their eggs to leaves with a special glue.<br>- Most caterpillars are plant eaters (herbivores).<br>- Fully grown caterpillars attach themselves to a suitable twig or leaf before shedding their outside layer of skin to reveal a hard skin underneath known as a chrysalis.<br>- An adult butterfly will eventually emerge from the chrysalis where it will wait a few hours for its wings to fill with blood and dry, before flying for the first time.<br>- Butterflies can live in the adult stage from anywhere between a week and a year, depending on the species.<br>- Butterflies have four wings.<br>- Butterflies often have brightly coloured wings with unique patterns made up of tiny scales.<br>- Most butterflies feed on nectar from flowers.<br>- Butterflies have taste receptors on their feet.<br>- Scientists estimate that there are between 15000 and 20000 different species of butterfly.<br>- Birdwing butterflies have large, angular wings and fly in a similar way to birds.<br>- Monarch butterflies are known for their long migration.";  
else if(name.equals("house fly"))  
 desc = "- House flies live almost everywhere there are people.<br>- House flies are relatively young insects on the evolutionary timeline.<br>- House flies multiply quickly.<br>- House flies don't travel very far, and take their time getting where they're going.<br>- House flies make their living in filth.<br>- House flies are on an all-liquid diet.<br>- House flies taste with their feet.<br>- House flies transmit a lot of diseases.<br>- House flies can walk upside down.<br>- House flies poop a lot.";  
else if(name.equals("parrot"))  
 desc = "- There are around 372 different parrot species.<br>- Most parrots live in tropical areas.<br>- Parrots have curved bills (beaks), strong legs and clawed feet.<br>- Parrots are often brightly coloured.<br>- Parrots are believed to be one of the most intelligent bird species.<br>- Some species are known for imitating human voices.<br>- Most parrot species rely on seeds as food. Others may eat fruit, nectar, flowers or small insects.<br>- Parrots such as the budgerigar (budgie) and cockatiel are popular as pets.<br>- Some parrot species can live for over 80 years.<br>- There are 21 different species of cockatoo.<br>- Cockatoos usually have black, grey or white plumage (feathers).<br>- New Zealand is home to some very unique parrots including the kea, kaka and kakapo.<br>- Keas are large, intelligent parrots that live in alpine areas of New Zealand’s South Island. They are the world’s only alpine parrot and are known for their curious and sometimes cheeky behaviour near ski fields where they like to investigate bags, steal small items and damage cars.<br>- Kakapos are critically endangered flightless parrots, as of 2010 only around 130 are known to exist. They are active at night (nocturnal) and feed on a range of seeds, fruit, plants and pollen. Kakapos are also the world’s heaviest parrot.<br>- The flag of Dominica features the sisserou parrot.";  
else if(name.equals("pig"))  
 desc = "- Pigs are intelligent animals.<br>- Like humans, pigs are omnivores, meaning they eat both plants and other animals.<br>- A pig’s snout is an important tool for finding food in the ground and sensing the world around them.<br>- Pigs have an excellent sense of smell.<br>- There are around 2 billion pigs in the world.<br>- Humans farm pigs for meat such as pork, bacon and ham.<br>- Some people like to keep pigs as pets.<br>- Wild pigs (boar) are often hunted in the wild.<br>- In some areas of the world, wild boars are the main source of food for tigers.<br>- Feral pigs that have been introduced into new areas can be a threat to the local ecosystem.<br>- Pigs can pass on a variety of diseases to humans.<br>- Relative to their body size, pigs have small lungs.";  
else if(name.equals("pigeon"))  
 desc = "- Pigeons are incredibly complex and intelligent animals.<br>- Pigeons are renowned for their outstanding navigational abilities.<br>- Pigeons are highly sociable animals. They will often be seen in flocks of 20-30 birds.<br>- Pigeons mate for life, and tend to raise two chicks at the same time.<br>- Both female and male pigeons share responsibility of caring for and raising young.<br>- Pigeons have excellent hearing abilities.<br>- Despite the social perception as dirty and disease-ridden, pigeons are actually very clean animals and there is very little evidence to suggest that they are significant transmitters of disease.<br>- Pigeons and humans have lived in close proximity for thousands of years.<br>- Although pigeon droppings are seen by some as a problem in modern society, a few centuries ago pigeon guano was seen as extremely valuable.<br>- Pigeons can fly at altitudes up to and beyond 6000 feet, and at an average speed of 77.6 mph. The fastest recorded speed is 92.5 mph.<br>- Pigeons are fed by many members of different religions including Muslims, Hindus and Sikhs for spiritual reasons.";  
else  
desc = "New Specie Encountered";  
return desc;  
 }  
}

**OverlayView.java**

package org.tensorflow.demo;  
  
import android.content.Context;  
import android.graphics.Canvas;  
import android.util.AttributeSet;  
import android.view.View;  
import java.util.LinkedList;  
import java.util.List;  
  
public class OverlayView extends View {  
private final List<DrawCallback>callbacks = new LinkedList<DrawCallback>();  
  
public OverlayView(final Context context, final AttributeSet attrs) {  
super(context, attrs);  
 }  
  
public interface DrawCallback {  
public void drawCallback(final Canvas canvas);  
 }  
  
public void addCallback(final DrawCallback callback) {  
callbacks.add(callback);  
 }  
  
 @Override  
public synchronized void draw(final Canvas canvas) {  
for (final DrawCallback callback : callbacks) {  
 callback.drawCallback(canvas);  
 }  
 }  
}

**RecognitionScoreView.java**

package org.tensorflow.demo;  
  
import android.app.Activity;  
import android.content.Context;  
import android.content.Intent;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.graphics.Canvas;  
import android.graphics.Paint;  
import android.graphics.Typeface;  
import android.util.AttributeSet;  
import android.util.TypedValue;  
import android.view.View;  
  
import org.tensorflow.demo.Classifier.Recognition;  
  
import java.io.ByteArrayOutputStream;  
import java.util.List;  
  
public class RecognitionScoreView extends View implements ResultsView {  
private static final float *TEXT\_SIZE\_DIP* = 54; private List<Recognition>results;  
private final float textSizePx;  
private final Paint fgPaint;  
private final Paint bgPaint;  
private final Context context1;  
  
  
public RecognitionScoreView(final Context context, final AttributeSet set) {  
super(context, set);  
context1=context;  
textSizePx =  
 TypedValue.*applyDimension*(  
 TypedValue.*COMPLEX\_UNIT\_DIP*, *TEXT\_SIZE\_DIP*, getResources().getDisplayMetrics());  
fgPaint = new Paint();  
fgPaint.setTextSize(textSizePx);  
fgPaint.setColor(0xcc9932cc);  
fgPaint.setTypeface(Typeface.*DEFAULT\_BOLD*);  
fgPaint.setTextAlign(Paint.Align.*CENTER*);  
  
bgPaint = new Paint();  
bgPaint.setColor(0xccffffff);}  
  
 @Override  
public void setResults(final List<Recognition> results) {  
this.results = results;  
 postInvalidate();  
 }  
  
 @Override  
public void onDraw(final Canvas canvas) {*;*int y = (int) (fgPaint.getTextSize() \* 1.5f);  
  
 canvas.drawPaint(bgPaint);  
  
if (results != null) {  
  
for (final Recognition recog : results) {  
 String name = recog.getTitle();Intent intent=new Intent(context1, Display.class);  
 intent.putExtra("name",name);  
context1.startActivity(intent);  
break;  
 }  
 }  
 }  
}

**ResultsView.java**

packageorg.tensorflow.demo;  
import org.tensorflow.demo.Classifier.Recognition;  
import java.util.List;  
  
public interface ResultsView {  
public void setResults(final List<Recognition> results);  
}

**TensorFlowImageClassifier.java**

package org.tensorflow.demo;  
  
import android.content.res.AssetManager;  
import android.graphics.Bitmap;  
import android.os.Trace;  
import android.util.Log;  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.util.ArrayList;  
import java.util.Comparator;  
import java.util.List;  
import java.util.PriorityQueue;  
import java.util.Vector;  
import org.tensorflow.Operation;  
import org.tensorflow.contrib.android.TensorFlowInferenceInterface;  
  
public class TensorFlowImageClassifier implements Classifier {  
private static final String *TAG* = "TensorFlowImageClassifier";  
  
private static final int *MAX\_RESULTS* = 3;  
private static final float *THRESHOLD* = 0.1f;  
  
private String inputName;  
private String outputName;  
private int inputSize;  
private int imageMean;  
private float imageStd;  
  
private Vector<String>labels = new Vector<String>();  
private int[] intValues;  
private float[] floatValues;  
private float[] outputs;  
private String[] outputNames;  
  
private boolean logStats = false;  
private TensorFlowInferenceInterface inferenceInterface;  
private TensorFlowImageClassifier() {}  
  
public static Classifier create(  
 AssetManager assetManager,  
 String modelFilename,  
 String labelFilename,  
int inputSize,  
int imageMean,  
float imageStd,  
 String inputName,  
 String outputName) {  
 TensorFlowImageClassifier c = new TensorFlowImageClassifier();  
 c.inputName = inputName;  
 c.outputName = outputName;  
  
String actualFilename = labelFilename.split("file:///android\_asset/")[1];  
 Log.*i*(*TAG*, "Reading labels from: " + actualFilename);  
 BufferedReader br = null;  
try {  
 br = new BufferedReader(new InputStreamReader(assetManager.open(actualFilename)));  
 String line;  
while ((line = br.readLine()) != null) {  
 c.labels.add(line);  
 }  
 br.close();  
 } catch (IOException e) {  
throw new RuntimeException("Problem reading label file!" , e);  
 }  
  
 c.inferenceInterface = new TensorFlowInferenceInterface(assetManager, modelFilename);  
  
final Operation operation = c.inferenceInterface.graphOperation(outputName);  
final int numClasses = (int) operation.output(0).shape().size(1);  
 Log.*i*(*TAG*, "Read " + c.labels.size() + "labels, output layer size is " + numClasses);  
  
c.inputSize = inputSize;  
 c.imageMean = imageMean;  
 c.imageStd = imageStd;  
  
c.outputNames = new String[] {outputName};  
 c.intValues = new int[inputSize \* inputSize];  
 c.floatValues = new float[inputSize \* inputSize \* 3];  
 c.outputs = new float[numClasses];  
  
return c;  
 }  
  
 @Override  
public List<Recognition> recognizeImage(final Bitmap bitmap) {  
Trace.*beginSection*("recognizeImage");  
  
 Trace.*beginSection*("preprocessBitmap");  
bitmap.getPixels(intValues, 0, bitmap.getWidth(), 0, 0, bitmap.getWidth(), bitmap.getHeight());  
for (int i = 0; i <intValues.length; ++i) {  
final int val = intValues[i];  
floatValues[i \* 3 + 0] = (((val >> 16) & 0xFF) - imageMean) / imageStd;  
floatValues[i \* 3 + 1] = (((val >> 8) & 0xFF) - imageMean) / imageStd;  
floatValues[i \* 3 + 2] = ((val & 0xFF) - imageMean) / imageStd;  
 }  
 Trace.*endSection*();  
  
Trace.*beginSection*("feed");  
inferenceInterface.feed(inputName, floatValues, 1, inputSize, inputSize, 3);  
 Trace.*endSection*();  
  
Trace.*beginSection*("run");  
inferenceInterface.run(outputNames, logStats);  
 Trace.*endSection*();  
  
Trace.*beginSection*("fetch");  
inferenceInterface.fetch(outputName, outputs);  
 Trace.*endSection*();  
  
PriorityQueue<Recognition> pq =  
new PriorityQueue<Recognition>(  
 3,  
new Comparator<Recognition>() {  
 @Override  
public int compare(Recognition lhs, Recognition rhs) {  
return Float.*compare*(rhs.getConfidence(), lhs.getConfidence());  
 }  
 });  
for (int i = 0; i <outputs.length; ++i) {  
if (outputs[i] >*THRESHOLD*) {  
 pq.add(  
new Recognition(  
"" + i, labels.size() > i ? labels.get(i) : "unknown", outputs[i], null));  
 }  
 }  
final ArrayList<Recognition> recognitions = new ArrayList<Recognition>();  
int recognitionsSize = Math.*min*(pq.size(), *MAX\_RESULTS*);  
for (int i = 0; i< recognitionsSize; ++i) {  
 recognitions.add(pq.poll());  
 }  
 Trace.*endSection*();return recognitions;  
 }  
  
 @Override  
public void enableStatLogging(boolean logStats) {  
this.logStats = logStats;  
 }  
  
 @Override  
public String getStatString() {  
return inferenceInterface.getStatString();  
 }  
  
 @Override  
public void close() {  
inferenceInterface.close();  
 }  
}

**6. SNAPSHOTS OF APP**

**6.1. App Icon**

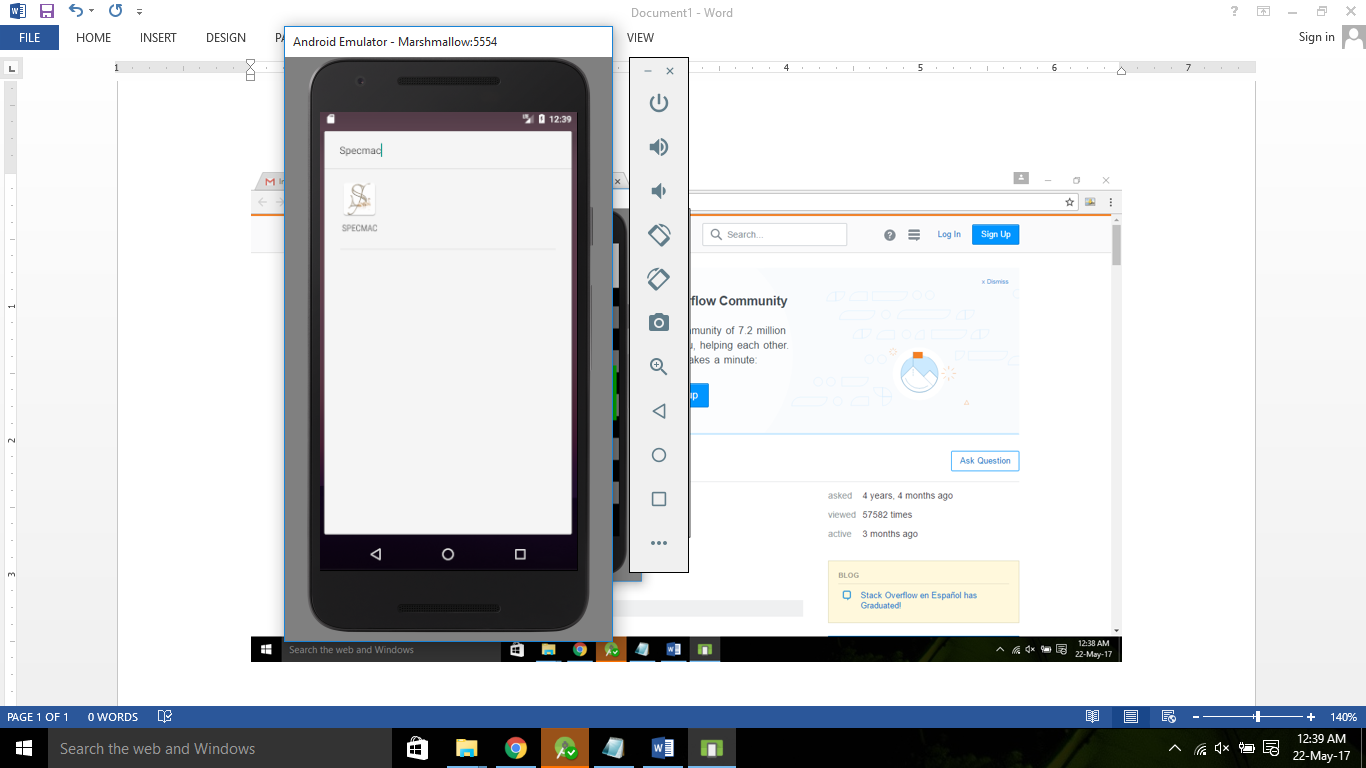


Figure 6.1: App Icon Screen

**6.2. Scanning Specie**

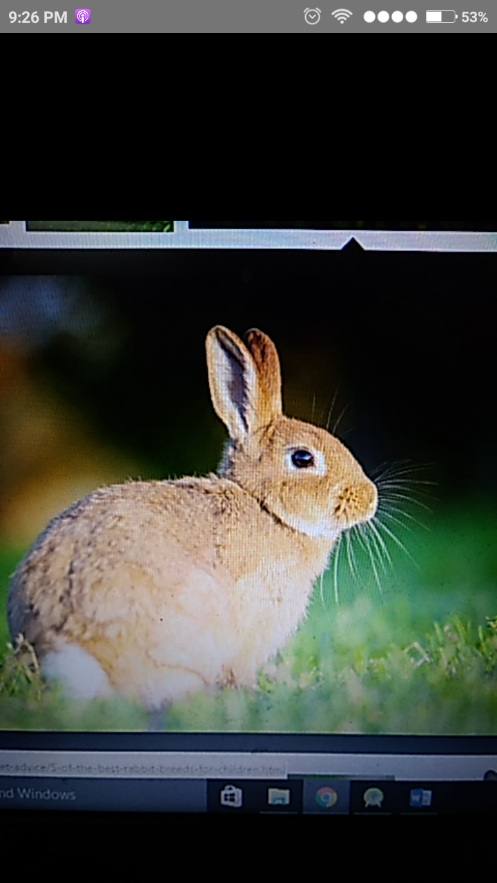
****

Figure 6.2: Scanning Specie Screen

**6.3. Detecting Specie**

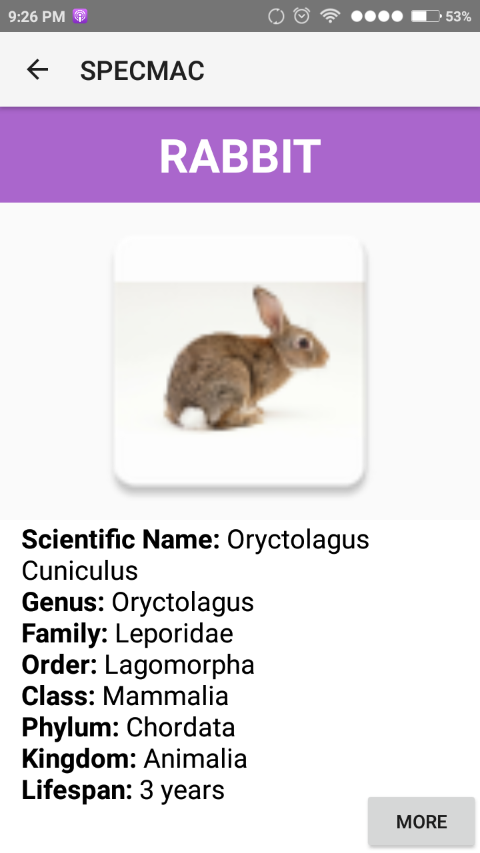
****

Figure 6.3: Detecting Specie Screen

**6.4. Specie Information**

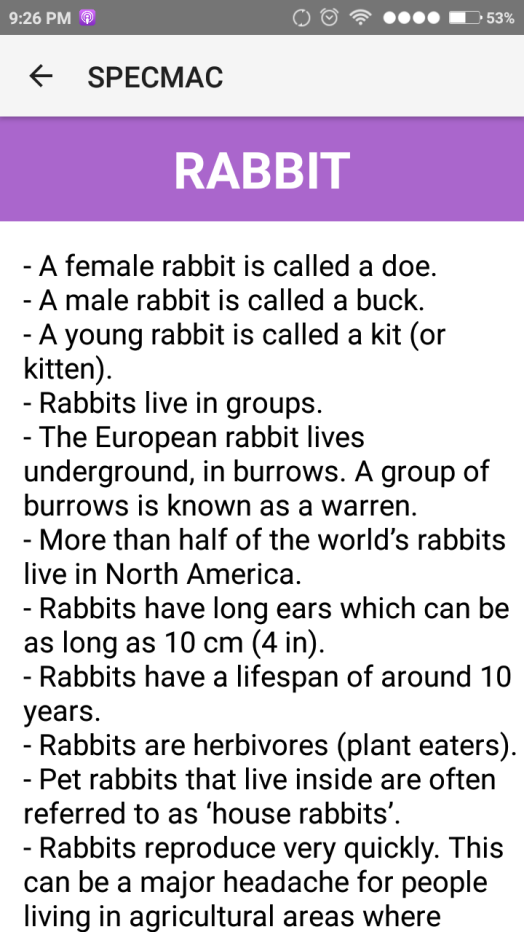
****

Figure 6.4: Specie Information Display Screen

**CONCLUSION**

For concluding the report, all the perspective of current system using in detection of the species are studied and then a plot is thought for a new system depending upon the latest technology. The proposed system can be used in replacement of current system giving better results. It also helps in giving information about the species detecting through the device.

For this purpose, the new approach of machine learning and deep learning is used tensorflow library and inception model. For making statistical work easy and effective, Python is chosen as workbenches. After plotting everything, feasibility study has been done. Data is collected as huge set of images, imported in inception model and this model is retrained using the images collected. This would help new zoologers to detect and learn about the specie on the Earth at any point through the portable smartphone.

**SUGGESTIONS FOR FURTHER STUDY OR FURTHER IMPROVEMENT**

Since the project ***Specmac: the Specie Machine*** is based on machine learning technology, it is not needed to update the algorithm as this will be done directly by machine self-learning. Apart from this, there are many possibilities as to how the project can be taken forward.

* The algorithm can further be developed for the iOS platform using swift.
* The data is collected as over 500 images for each species. More images can be collected for the better and effective accuracy of the model.
* The newly collected data can be stored in the form of table through SQLite which can be updated in the future after discovering them.
* The system can be further made as an expert system which can also give reply against the random questions asked by the user about the species.

# REFERENCES

|  |  |
| --- | --- |
| [1] | AK Jain, MN Murty, and P Flynn, "Data clustering: a review," *ACM Comput Surveys*, vol. 31, no. 3, pp. 264-323, 1999. |
| [2] | L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Proc. Advances in Neural Information Processing Systems vol. 20*, 2007, p. 161/168. |
| [3] | R. O. Duda and P. E. Hart, *Pattern Classiﬁcation and Scene Analysis*.: Wiley, 1973. |
| [4] | B. Schölkopf and Smola, "A. Learning with Kernels," MIT, 2002. |
| [5] | Y. Bengio, O. Delalleau, and N. Roux Le, "The curse of highly variable functions for local kernel machines," in *Proc. Advances in Neural Information Processing Systems*, 2005, pp. 107-114. |
| [6] | A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Advances in Neural Information Processing Systems*, 2012, pp. 1090-1098. |
| [7] | C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling.," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 35, pp. 1915-1929, 2013. |
| [8] | J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Proc. Advances in Neural Information Processing Systems*, 2014, pp. 1799-1807. |
| [9] | C. et al Szegedy. (2014) Going deeper with convolutions. [Online]. HYPERLINK "http://arxiv.org/abs/1409.4842" http://arxiv.org/abs/1409.4842 |
| [10] | T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Cernocky, "Strategies for training large scale neural network language models," in *Proc. Automatic Speech Recognition and Understanding*, 2011, pp. 196-201. |
| [11] | G. et al. Hinton, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, pp. 82-97, 2012. |
| [12] | T. Sainath, A.-R. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Proc. Acoustics, Speech and Signal Processing*, 2013, pp. 8614-8616. |
| [13] | T. Ciodaro, D. Deva, J. de Seixas, and D. Damazio, "Online particle detection with neural networks based on topological calorimetry information," in *J. Phys. Conf. Series 368*, 2012. |
| [14] | Kaggle. (2014) Higgs boson machine learning challenge. [Online]. HYPERLINK "https://www.kaggle.com/c/higgs-boson" https://www.kaggle.com/c/higgs-boson |
| [15] | J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, "Deep neural nets as a method for quantitative structure-activity relationships," *J. Chem. Inf. Model.*, vol. 55, pp. 263-274, 2015. |
| [16] | M. et al. Helmstaedter, "Connectomic reconstruction of the inner plexiform layer in the mouse retina," vol. 500, pp. 168-174, 2013. |
| [17] | M. K. Leung, H. Y. Xiong, L. J. Lee, and B. J. Frey, "Deep learning of the tissueregulated splicing code," *Bioinformatics*, vol. 30, pp. 121-129, 2014. |
| [18] | H. Y. et al. Xiong, "The human splicing code reveals new insights into the genetic determinants of disease," *Science 347*, p. 6218, 2015. |
| [19] | R., et al. Collobert, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res*, vol. 12, pp. 2493-2537, 2011. |
| [20] | A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," in *Proc. Empirical Methods in Natural Language Processing*, 2014. |
| [21] | S. Jean, K. Cho, R. Memisevic, and Y. Bengio. (2015) On using very large target vocabulary for neural machine translation. [Online]. HYPERLINK "http://arxiv.org/abs/1412.2007" http://arxiv.org/abs/1412.2007 |
| [22] | I. Sutskever, O. Vinyals, and Le. Q. V., "Sequence to sequence learning with neural networks," in *Proc. Advances in Neural Information Processing Systems*, 2014, pp. 3104-3112. |
| [23] | T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Cernocky, "Strategies for training large scale neural network language models," in *Proc. Automatic Speech Recognition and Understanding*, 2011, pp. 196-201. |
| [24] | L. S. Camargo and T. Yoneyama, "Specification of Training Sets and the Number of Hidden Neurons for Multilayer Perceptrons," in *Neural Computation*, 2001, pp. 2673-2680. |
| [25] | M. Kon and L. Plaskota, "Information complexity of neural networks," *Neural Networks*, vol. 13, pp. 365-375, 2000. |
| [26] | C. Neocleous and C. Schizas, "Artificial Neural Network Learning: A Comparative Review," *Springer*, vol. LNAI 2308, pp. 300–313, 2002. |
| [27] | J. Yam and W. Chow, "Feedforward Networks Training Speed Enhancement by Optimal Initialization of the Synaptic Coefficients," *IEEE Transactions on Neural Networks*, vol. 12, pp. 430-434, 2001. |
| [28] | M. N. H. Siddique and M. O. Tokhi, "Training Neural Networks: Backpropagation vs. Genetic Algorithms," in *IEEE International Joint Conference on Neural Networks, Vol. 4*, 2001, pp. 2673–2678. |
| [29] | G. G. Yen and H. Lu, "Hierarchical genetic algorithm based neural network design," in *IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, 2000, pp. 168–175. |
| [30] | F. Vivarelli and C. Williams, "Comparing Bayesian neural network algorithms for classifying segmented outdoor images," in *Neural Networks 14*, 2001, pp. 427-437. |
| [31] | G. Castellano, A. Fanelli, and M. Pelillo, "An iterative pruning algorithm for feedforward neural networks," in *IEEE Transactions on Neural Networks 8*, 1997, pp. 519–531. |
| [32] | R. Parekh, J. Yang, and V. Honavar, "Constructive Neural Network Learning Algorithms for Pattern Classification," *IEEE Transactions on Neural Networks*, vol. 11, no. 2, pp. 436-451, 2000. |
| [33] | J. Robert and Howlett L.C.J., "Radial Basis Function Networks 2: New Advances in Design," 2001. |
| [34] | N. Littlestone and M. Warmuth, "The weighted majority algorithm," *Information and Computation*, vol. 108, no. 2, pp. 212–261, 1994. |
| [35] | A. Blum, "Empirical Support for Winnow and Weighted-Majority Algorithms: Results on a Calendar Scheduling Domain," *Machine Learning*, vol. 26, no. 1, pp. 5-23, 1997. |
| [36] | Auer P. and Warmuth M., "Tracking the Best Disjunction," *Machine Learning 32*, pp. 127–150, 1998. |
| [37] | Y. Freund and R. Schapire, "Large Margin Classification Using the Perceptron Algorithm," *Machine Learning 37*, pp. 277–296, 1999. |

**LIST OF PUBLICATIONS AND PRESENTATIONS**

1. Chawla S., SinglaB., Nisha, Gupta S., “Modeling Specie Detection using Deep Learning”, International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE), Vol. 4, Issue 10, Oct 2016.

**BIBLIOGRAPHY**

1. https://developer.android.com/studio/index.html
2. https://www.programiz.com/python-programming
3. https://www.tutorialspoint.com/android/android\_studio.htm
4. https://www.tensorflow.org/
5. https://www.tutorialspoint.com/sqlite/sqlite\_overview.htm
6. http://stackoverflow.com/questions/101754/is-there- a-way- to-run- python-on- android
7. https://www.quora.com/Can-I- make-an- Android-app- with-Python
8. https://wiki.python.org/moin/Android