

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590018, Karnataka



Project Report on

"HOUSE HOLD OBJECT RECOGNITION USING CONVOLUTION NEURAL NETWORK"

Submitted in partial fulfillment of the requirements for the award of the
degree of Bachelor of Engineering
in
Information Science & Engineering

Submitted by

USN
1BI15IS015
1BI16IS074
1BI16IS080
1BI17IS410

Name
CHANDRAKALA V
VARUNI A J
PRIYANKA J
SAHANA K V

Under the Guidance of
Mrs. M CHETHANA
Assistant Professor
Department of IS&E,
BIT Bengaluru-560004



**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING
BANGALORE INSTITUTE OF TECHNOLOGY**

K.R.Road, V.V.Pura, Bengaluru-560 004

2019-20



VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY

Bengaluru-560 004



DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

Certificate

This is to certify that the Project Work Phase II (15CSP85) work entitled "**HOUSE HOLD OBJECT RECOGNITION USING CONVOLUTION NEURAL NETWORK**" has been successfully completed by **Chandrakala V(1BI15IS015)**, **Varuni A J (1BI16IS074)**, **Priyanka J (1BI16IS080)** and **Sahana K V (1BI17IS410)** of VIII semester for the fulfillment of the requirements for the Bachelor of Engineering Degree in Information Science & Engineering of the VISVESVARAYA TECHNOLOGICAL UNIVERSITY during the academic year 2019-2020.

Mrs. M Chethana

Internal Guide

Assistant Professor

Department of IS&E

Bangalore Institute of Technology

Bangalore - 560004.

Dr. J Prakash

Professor and Head

Dept. of Information Science & Engineering

Bangalore Institute of Technology

K. R. Road, V. V. Puram

Bangalore - 560004.

Examiners: 1.

2.



ACKNOWLEDGEMENT

The knowledge & satisfaction that accompany the successful completion of any task would be incomplete without mention of people who made it possible, whose guidance and encouragement crowned our effort with success. We would like to thank all and acknowledge the help received to carry out this project.

We would like to express our sincere gratitude to **Dr.Aswath M U**, Principal, BIT for providing us a congenial environment and surrounding to work in.

We take this opportunity to also thank **Dr. J Prakash**, Professor and Head of the Department of Information Science and Engineering, BIT for giving us the privilege of working on such an interesting topic and for the help and guidance throughout the academic semester.

We would not forget to remember **Prof. K C Anupama**, Assistant Professor and Project Coordinator, for her encouragement and more over for her timely support and guidance till the completion of Project.

We are most humbled to mention the enthusiastic influence provided by our guide **Asst prof .M Chethana**, for her ideas, time to time suggestions and co-operation showed during the venture and helping make our project a success.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all Teaching and non-teaching staffs of ISE department which helped us in successfully completing the project work. We would also take this opportunity to thank our friends and family for their constant support and help.

Finally, we wish to thank all those who have directly or indirectly helped us in the completion of the project.

VARUNI A J
PRIYANKA J
SAHANA K V
CHANDRAKALA V

ABSTRACT

An object detection system finds objects of the real world present either in a digital image or a video, where the object can belong to house hold objects namely chairs, mug, etc. In order to detect an object in an image or a video the system needs to have a few components in order to complete the task of detecting an object, they are a model database, a feature detector, a hypothesiser and a hypothesiser verifier. This project presents a review of the various techniques that are used to detect an object, localise an object, categorise an object, extract features, appearance information, and many more, in images and videos. The comments are drawn based on the studied literature and key issues are also identified relevant to the object detection. Information about the source codes and online datasets is provided to facilitate the new researcher in object detection area. An idea about the possible solution for the multi class object detection is also presented.

TABLE OF CONTENTS

CHAPTER 1 – INTRODUCTION

CHAPTER 2 - REVIEW OF LITERATURE

2.1	Rich Feature Hierarchies for Accurate Object Detection Segmentation ,IEEE , Jun. 2014[1].	3
2.2	Faster R-CNN: Towards Real Time Object Detection with Region Proposal Networks,2017[1]	4
2.3	Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, IEEE 2016[3]	6
2.4	“YOLO9000: Better, Faster, Stronger,” J. Redmon and A. Farhadi, IEEE Press, Jul. 2017[3].	7

CHAPTER 3 – PROBLEM STATEMENT

3.1	Aim	9
3.2	Objectives	9

CHAPTER 4 – SYSTEM REQUIREMENTS 11

4.1	Functional Requirements	13
4.2	Non-Functional Requirements	

CHAPTER 5 – ARCHITECTURE 15

5.1	System Architecture	20
5.2	Module Description	

CHAPTER 6 – DESIGN

6.1	Data Flow Diagram	26
6.2	Flow Chart	28
6.3	Use case Diagram	29

CHAPTER 7 – IMPLEMENTATION

7.1	Implementation support	30
7.2	Views.py	37
7.3	Prediction.html	47
7.4	Recognition.html	48

7.5	Model.py	50
7.6	Prediction.py	52
CHAPTER 8 – PERFORMANCE ANALYSIS		
8.1	Testing	55
8.2	Results	59
CHAPTER 9 – CONCLUSION		
9.1	Conclusion	
9.2	Future Work	63
9.3	Application	63
BIBLIOGRAPHY		66

LIST OF FIGURES

Figure No.	Description	Page No.
5.1	Proposed Architecture	24
5.2	Overall project presentation	27
5.3	Web services	29
5.4	User profile	30
5.5	End point profile	31
6.1	Data flow diagram	34
6.2	Level data flow diagram	35
6.3	Flow chart	36
6.4	Use case diagram	37
8.1	Main screen	66
8.2	Registration block	67
8.3	Login page	68
8.4	Model configuration	69
8.5	Choosing image	70
8.6	Object recognition	70
8.7	Download image	71
8.8	Inputs from User	71

LIST OF TABLES

Table No.	Description	Page No.
8.1	Testing along with Results	63
8.2	Performance evaluation and Analysis	65

Chapter 1:

INTRODUCTION

Radar detection of moving objects is vulnerable to external environment. By introducing the object confirmation algorithm, the intelligent radar perimeter security system's false alarm rate can be reduced significantly. The object confirmation algorithm is essentially an object detection algorithm. Due to the poor generalization of artificial feature extraction algorithms, use the deep convolution neural network to extract deep features automatically for object confirmation.

In order to meet the real-time requirement in engineering practices, our algorithm use the YOLOv2 system as a basis, and selects anchor boxes which meet object scales of our training data set by k-means++ clustering. To improve the YOLOv2 network structure, low-layer deep features which denote the texture information and high-layer deep features which denote the semantic information are combined layer by layer to make object detection more accurate.

The experimental results show that the false alarm rate of the intelligent radar perimeter security system is further reduced by introducing the object confirmation algorithm. Especially for extreme weather, false alarms of radar detection greatly increase. But most of them are eliminated after running object confirmation algorithm. Therefore the warning accuracy of the entire system can be guaranteed. The detection speed of the object confirmation algorithm is 33FPS, which meets the real time requirement of engineering practices.

Machine Learning is one of the most hot research topics in computer science and engineering, which is applicable in many disciplines. It provides a collection of algorithms, methods and tools able to embody some kind of intelligence to machines. The power of ML is the provided modeling tools, which are able to be trained, via a learning procedure, with a set of data describing a certain problem and to respond to similar unseen data with a common way.

Machine Learning is an idea to learn from examples and experience, without being explicitly programmed. Instead of writing code, you feed data to the generic algorithm, and it builds logic based on the data given. The process of training an ML model involves providing an ML algorithm with training data to learn from. But most of them are eliminated after running object confirmation algorithm. Therefore the warning accuracy of the entire system can be guaranteed. The detection speed of the object confirmation algorithm is 33FPS, which meets the real time requirement of engineering practices.

CHAPTER 2

LITERATURE REVIEW

A literature review or narrative review is a type of review article. A literature review includes the current knowledge including substantive findings, as well as theoretical and methodological contributions to a particular topic. Literature reviews are secondary sources, and do not report new or original experimental work. Most often associated with academic oriented literature, such reviews are found in academic journals, and are not to be confused with book reviews that may also appear in the same publication. Literature reviews are a basis for research in nearly every academic field. A narrow-scope literature review may be included as part of a peer reviewed journal article presenting new research, serving to situate the current study within the body of the relevant literature and to provide context for the reader. In such a case, the review usually precedes the methodology and results sections of the work.

2.1 “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”

R. Girshick, J. Donahue, T. Darrell and J. Malik ,IEEE Press, Jun. 2014[1].

Object detection algorithms based on deep convolutional neural networks have achieved a qualitative improvement in performance. In 2014, deep features are applied to object detection for the first time in R-CNN [1]. Since then, object detection with deep learning based on Region Proposal was pioneered. SPP-net [2], Fast R-CNN [3], Faster R-CNN [4] and R-FCN [5] all were improvement of this kind of method. With the optimization of deep convolutional neural network, the above methods achieved higher accuracy and faster detection speed. But all of them didn't achieve real-time. In engineering practices, real-time is very important.

Methodology

R-CNN

To bypass the problem of selecting a huge number of regions, Ross Girshick et proposed a method where we use selective search to extract just 2000 regions from the image and he called them region proposals. Therefore, now, instead of trying to classify a huge number of regions, you can just work with 2000 regions.

The CNN acts as a feature extractor and the output dense layer consists of the features extracted from the image and the extracted features are fed into an SVM to classify the presence of the object within that candidate region proposal. In addition to predicting the presence of an object within the region proposals, the algorithm also predicts four values which are offset values to increase the precision of the bounding box. For example, given a region proposal, the algorithm would have predicted the presence of a person but the face of that person within that region proposal could've been cut in half. Therefore, the offset values help in adjusting the bounding box of the region proposal.

Problems with R-CNN

- It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.
- It cannot be implemented real time as it takes around 47 seconds for each test image.
- The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.

2.2 “Faster R-CNN: Towards RealTime Object Detection with Region Proposal Networks,” S Ren, K. He, R. Girshick and J. Sun, IEEE 2017[2].

Object detection is an important, yet challenging vision task. It is a critical part in many applications such as image search, image auto-annotation and scene understanding, object tracking. Moving object tracking of video image sequences was one of the most important subjects in computer vision. It had already been applied in many computer vision field, such as smart video surveillance (Arun Hampapur 2005), artificial intelligence, military guidance, safety detection and robot navigation, medical and biological application. In recent years, a number of successful single-object tracking system appeared, but in the presence of several objects, object detection becomes difficult and when objects are fully or partially occluded, they are obtruded from human vision which further increases the problem of detection.

Methodology

Fast R-CNN

The approach is similar to the R-CNN algorithm. But, instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map. From the convolutional feature map, we identify the region of proposals and warp them into squares and by using a RoI pooling layer we reshape them into a fixed size so that it can be fed into a fully connected layer. From the RoI feature vector, we use a softmax layer to predict the class of the proposed region and also the offset values for the bounding box.

The reason “Fast R-CNN” is faster than R-CNN is because you don’t have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image and a feature map is generated from it. Both of the above algorithms (R-CNN & Fast R-CNN) use selective search to find out the region proposals. Selective search is a slow and time-consuming process affecting the performance of the network. Therefore, Shaoqing Ren et al. came up with an object detection algorithm that eliminates the selective search algorithm and lets the network learn the region proposals.

Similar to Fast R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.

Limitations

- It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.
- It cannot be implemented real time as it takes around 47 seconds for each test image.

2.3 “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,” K. He, X. Zhang, S Ren and J. Sun ,IEEE 2015[3].

The ImageNet challenge has been crucial in demonstrating the effectiveness of deep CNNs. The problem is to recognize object categories in typical imagery that one might find on the Internet. The 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) classification task is to classify imagery obtained from Flickr and other search engines into the correct one of 1000 possible object category classes.

Visual recognition methods not based on deep CNNs hit a plateau in performance on this benchmark. The “top-5 error” is the percentage of times that the target label does not appear among the 5 highest-probability predictions, and many methods cannot get below 25%.

Template Matching

Template Matching is a technique for finding small parts of an image which match a template image. It slides the template from the top left to the bottom right of the image and compare for the best match with the template. The template dimension should be equal or smaller than the reference image. It recognizes a segment with the highest correlation as a target. Given an image S and an image T, where the dimension of S was both larger than T, output whether S contains a subset image I where I and T are suitably similar in pattern and if such I exists, output the location of I in S as in Hager and Bellhumeur.

Visual tracking methods can be roughly categorized in two ways namely, feature based and region based method as proposed by Ken Ito and Shigeyuki Sakane (2001). Feature based approach estimates the 3D pose of a target object to fit to the image features the edges, given a 3D geometrical model of an object. This method requires much computational cost. Region based can be classified into two categories namely, parametric method and view based method. The parametric method assumes a parametric model of the images in the target image and calculates optimal fitting of the model to pixel data in a region. View based method was used to find the best match of a region in a search area given the reference template. This has the

advantage that it does not require much computational complexity as in the feature based approach.

Disadvantages

- The best-known disadvantage of neural networks is their “black box” nature.
- Neural networks are also more computationally expensive than traditional algorithms.
- Duration of development.

2.4 “YOLO9000: Better, Faster, Stronger,” J. Redmon and A. Farhadi, IEEE Press, Jul. 2017[3].

YOLO is orders of magnitude faster(45 frames per second) than other object detection algorithms. The limitation of YOLO algorithm is that it struggles with small objects within the image, for example it might have difficulties in detecting a flock of birds. This is due to the spatial constraints of the algorithm.

Object detection is one of the classical problems in computer vision where you work to recognize what and where — specifically what objects are inside a given image and also where they are in the image. The problem of object detection is more complex than classification, which also can recognize objects but doesn’t indicate where the object is located in the image. In addition, classification doesn’t work on images containing more than one object. YOLO uses a totally different approach. YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

Object Tracking

A pattern for tracking moving objects efficiently was surveyed by Yilmaz et al (2006), through mining group movement. The work focuses on multiple group tracking based on local relationship and moving pattern. In this approach, similar object moving pattern could be efficiently grouped but different moving object could not be grouped and thus tracking failed in

such scenarios leading to energy waste. An appropriate solution adopted here is to use individual trackers for the objects in the video under analysis.

Filter Based Tracking

Jin Wang (2010), presented a model that improved the existing Kalman Filter to track objects even in the presence of occlusions. This work was a combination of appearance and motion information to track objects. The advantages of the method were that it was successful in locating objects accurately even when they were partially or fully occluded and was able to easily distinguish dissimilarly colored objects. The disadvantage of the method was its unreliability and less efficient along with wrong correspondence when different objects have similar appearance. An effective solution to overcome these problems are provided in the proposed system by selecting motion parameters along with feature extraction and formulating the distance based bus topology framework.

Advantages of Yolo

- YOLO is extremely fast.
- YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance.
- YOLO learns generalizable representations of objects so that when trained on natural images and tested on artwork, the algorithm outperforms other top detection methods.

CHAPTER 3

PROBLEM STATEMENT

3.1 Aim: “To develop a machine learning model to recognize the objects and find its accuracy”.

This convolutional neural network obtained state-of-the-art performance at object recognition on the CIFAR-10 image dataset in 2015. We will build this model using Keras, a high-level neural network application programming interface (API) that supports both Theano and Tensorflow backend. You can use either backend; however, using Theano.

3.2 Objectives

- Implement a Convolutional Neural Network using Python’s Machine Learning libraries sci-kit learn for recognizing objects in the input image.
- Split the input datasets into training data and testing data
- Train the model using the training data
- Test the model using the testing data and calculate the accuracy.
- Provide data visualization tools for the users for better user experience
- Deploy the solution over cloud and make it available to the public for use in as-a-service model.

3.2.1 Proposed System

In this project, we will be deploying a convolutional neural network (CNN) for object recognition. More specifically, we will be using the All-CNN network published in the 2015 ICLR paper, "Striving For Simplicity: The All Convolutional Net".

This convolutional neural network obtained state-of-the-art performance at object recognition on the CIFAR-10 image dataset in 2015. We will build this model using Keras, a high-level neural network application programming interface (API) that supports both Theano and Tensorflow backend. You can use either backend; however, we will be using Theano.

In this project, we will be performing the below tasks:

- Import datasets from Keras
- Use one-hot vectors for categorical labels
- Add layers to a Keras model
- Load pre-trained weights
- Make predictions using a trained Keras model

The dataset we will be using is the CIFAR-10 dataset, which consists of 60,000 32x32 color images in 10 classes, with 6000 images per class. There will be 50000 training images and 10000 test images.

3.2.2 proposed system outcome



Fig3.1 system outcome

CHAPTER 4

SYSTEM REQUIREMENTS

A Software Requirements Specification (SRS) is a description of a software system to be developed. It is modeled after business requirements specification also known as a stakeholder requirements specification (SRS). The software requirements specification lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide.

SRS establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

The SRS document lists sufficient and necessary requirements for the project development. To derive the requirements, the developer needs to have clear and thorough understanding of the products under development. This is achieved through detailed and continuous communications with the project team and customer throughout the software development process. The SRS may be one of a contract deliverable Data Item Descriptions or have other forms of organizationally-mandated content

.

4.1 Functional Requirements

A function of software system is defined in functional requirement and the behaviour of the system is evaluated when presented with specific inputs or conditions which may include calculations, data manipulation and processing and other specific functionality. The functional requirements of the project are one of the most important aspects in terms of entire mechanism of modules.

Product Perspective

- Accepting each registration input from the user and creating them in MySQL DB.
- Ability for the product admin to grant access or remove the access to the users from accessing the web services
- Ability for the users to run the model over the cloud making use of the RESTful web services exposed.
- To design an efficient user interface module to the admin to perform initial configuration
- To design and implement the machine learning model to recognize objects from the inputted images.
- To design and develop the Notification Module which alerts the user in case of an undesired condition.

Product functions

- The product must be independent from the underlying data node architecture (it should work on any number of data nodes)
- The product will be deployed on a fully functional application server like Apache Tomcat or WildFly Application Server
- All the modules will be implemented as a dynamic web application.

User Characteristics

- In this project, as need a small amount of configuration work to be done in the Eclipse box to set up the application
- User also needs to install Apache Tomcat/ WildFly server to host the all the user portals
- The input provided by the users should be from the web browsers where the users will be the patients personal information and the patient health record
- The appropriate portal must be accessed only the appropriate users belonging the correct role

4.2 Non-Functional Requirements

Non-functional requirements describe how a system must behave and establish constraints of its functionality. This type of requirements is also known as the system's quality attributes. Attributes such as performance, security, usability, compatibility are not the feature of the system, they are a required characteristic. They are "developing" properties that emerge from the whole arrangement and hence we can't compose a particular line of code to execute them. Any attributes required by the customer are described by the specification. We must include only those requirements that are appropriate for our project.

Reliability: The structure must be reliable and strong in giving the functionalities. The movements must be made unmistakable by the structure when a customer has revealed a couple of enhancements. The progressions made by the Programmer must be Project pioneer and in addition the Test designer.

Maintainability: The system watching and upkeep should be fundamental and focus in its approach. There should not be an excess of occupations running on diverse machines such that it gets hard to screen whether the employments are running without lapses.

Performance: Given the high computational requirements of proof-of-work, performance is crucial for good user experience. The structure should not capitulate when various customers would use everything the while. It should allow brisk accessibility to each and every piece of its customers. For instance, if two test specialists are all the while attempting to report the vicinity of a bug, then there ought not to be any irregularity at the same time.

Portability: The framework should to be effectively versatile to another framework. This is obliged when the web server, which s facilitating the framework gets adhered because of a few issues, which requires the framework to be taken to another framework.

Scalability: The framework should be sufficiently adaptable to include new functionalities at a later stage. The platform is a fixed cost liability and therefore an increase in scale translates to improved revenues. Flexibility: Flexibility is the capacity of a framework to adjust to changing

situations and circumstances, and to adapt to changes to business approaches and rules. An adaptable framework is one that is anything but difficult to reconfigure or adjust because of diverse client and framework prerequisites. The deliberate division of concerns between the trough and motor parts helps adaptability as just a little bit of the framework is influenced when strategies or principles change.

4.3 HARDWARE REQUIREMENTS

- Processor: Intel core i5 or AMD FX 8 core series
- RAM: 2 gigabyte (GB) (32-bit) or 4 GB
- Display: XGA(1024*768)pixels or higher resolution monitor with 32 bit color settings.
- Hard Disk: 50 GB and above.

4.4 SOFTWARE REQUIREMENTS

- Operating system: Windows 7 / 8/10,linux or cent OS
- Coding Language: Python 3.0,Core Java, Advanced Java ,J2EE,MVC framework.
- Frontend: Bootstrap framework ,HTML, CSS, JavaScript, Ajax, JQuery
- Application server :Apache Tomcat v9.0
- Database: HDFS

CHAPTER 5

ARCHITECTURE

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages.

5.1 System Architecture

The proposed system uses the data scraped from the kagle website and applies machine learning to develop a machine learning model of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

The system architecture is made up of:

Data Access Layer

Data access layer is the one which exposes all the possible operations on the data base to the outside world. It will contain the DAO classes, DAO interfaces, POJOs, and Utils as the internal components. All the other modules of this project will be communicating with the DAO layer for their data access needs.

Account Operations

Account operations module provides the following functionalities to the end users of our project.

- Register a new seller/ buyer account
- Login to an existing account
- Logout from the session

- Edit the existing Profile
- Change Password for security issues
- Forgot Password and receive the current password over an email
- Delete an existing Account

Account operations module will be re-using the DAO layer to provide the above functionalities.

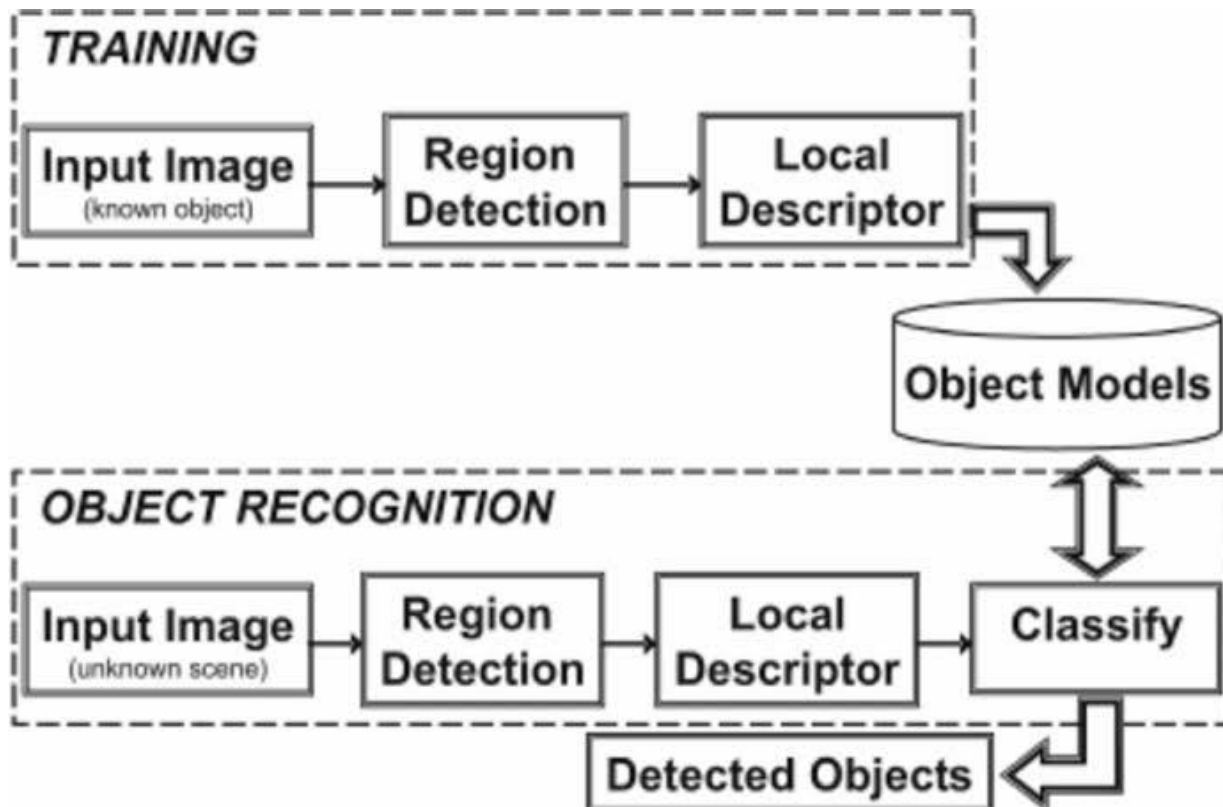


Figure 5.1: Proposed Architecture

CNN algorithm for object Recognition

Here, the core algorithm will be implemented to recognize the objects in the input image. In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

Training and Testing the model for accuracy

Here, the model will be trained using the datasets and tested for finding the accuracy of the model. Optimization will be done to improve the accuracy if needed. In machine learning, a common task is the study and construction of algorithms that can learn from and make predictions on data. Such algorithms work by making data-driven predictions or decisions, through building a mathematical model from input data.

The data used to build the final model usually comes from multiple datasets. In particular, three data sets are commonly used in different stages of the creation of the model. The model is initially fit on a training dataset, that is a set of examples used to fit the parameters (e.g. weights of connections between neurons in artificial neural networks) of the model. The model (e.g. a neural net or a naive Bayes classifier) is trained on the training dataset using a supervised learning method (e.g. gradient descent or stochastic gradient descent). In practice, the training dataset often consist of pairs of an input vector (or scalar) and the corresponding output vector (or scalar), which is commonly denoted as the target (or label). The current model is run with the training dataset and produces a result, which is then compared with the target, for each input

vector in the training dataset. Based on the result of the comparison and the specific learning algorithm being used, the parameters of the model are adjusted. The model fitting can include both variable selection .

Successively, the fitted model is used to predict the responses for the observations in a second dataset called the validation dataset. The validation dataset provides an unbiased evaluation of a model fit on the training dataset while tuning the model's hyper parameters (e.g. the number of hidden units in a neural network). Validation datasets can be used for regularization by early stopping: stop training when the error on the validation dataset increases, as this is a sign of over fitting to the training dataset. This simple procedure is complicated in practice by the fact that the validation dataset's error may fluctuate during training, producing multiple local minima. This complication has led to the creation of many ad-hoc rules for deciding when over fitting has truly begun.

Finally, the test dataset is a dataset used to provide an unbiased evaluation of a final model fit on the training dataset. If the data in the test dataset has never been used in training (for example in cross-validation), the test dataset is also called a holdout dataset.

Implementation of REST APIs for exposing the model to other apps/clients

Here, the APIs will be developed so that the existing applications can re-use the model we developed in the second module. Representational state transfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services. Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the Internet. RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations. Other kinds of Web services, such as SOAP Web services, expose their own arbitrary sets of operations.

"Web resources" were first defined on the World Wide Web as documents or files identified by their URLs. However, today they have a much more generic and abstract definition that encompasses everything or entity that can be identified, named, addressed, or handled, in any way whatsoever, on the Web. In a RESTful Web service, requests made to a resource's URI will

elicit a response with a payload formatted in HTML, XML, JSON, or some other format. The response can confirm that some alteration has been made to the stored resource, and the response can provide hypertext links to other related resources or collections of resources. When HTTP is used, as is most common, the operations (HTTP methods) available are GET, HEAD, POST, PUT, PATCH, DELETE, CONNECT, OPTIONS and TRACE.

User Interface design for the model

Here, the front end interface will be designed so that the end users can interact with the model with ease. User interface design (UI) or user interface engineering is the design of user interfaces for machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing usability and the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centered design).

Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to itself. Graphic design and typography are utilized to support its usability, influencing how the user performs certain interactions and improving the aesthetic appeal of the design; design aesthetics may enhance or detract from the ability of users to use the functions of the interface. The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

Cloud based deployment process of the model

Here, the model will be deployed on a cloud server to make the solution accessible across the geographical areas. For the cloud deployment process, we use either of Amazon web service or the Google Cloud.

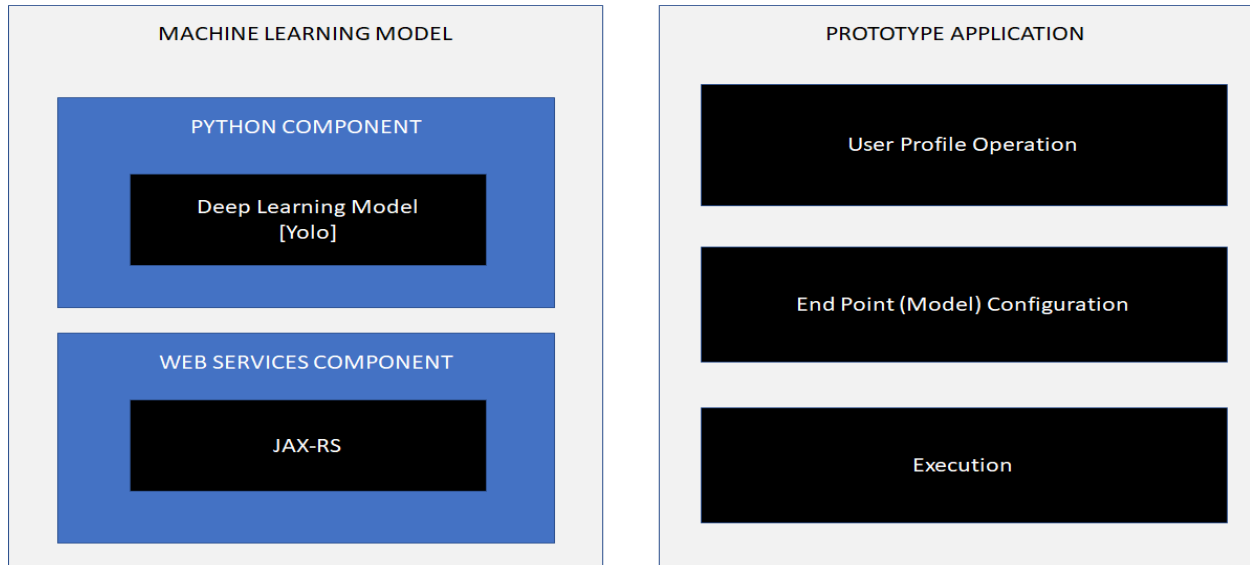


FIGURE 5.2 Here is the overall representation of the project

5.2 MODULE DESCRIPTION

The System has the following modules:

5.2.1 Object Recognition Model

In this module, we implement a Machine learning Model to recognize numerous objects from an input image. To do this, we make use Yolo Algorithm which is based on Convolutional Neural Network. Yolo is an algorithm that uses convolutional neural networks for object detection.

In comparison to recognition algorithms, a detection algorithm does not only predict class labels, but detects locations of objects as well. To build Yolo we're going to need Tensorflow (deep learning), NumPy (numerical computation) and Pillow (image processing) libraries. Also we're going to use seaborn's color palette for bounding boxes colors. Finally, let's import IPython function display() to display images in the notebook.

Module 2: Web services

In this module, we implement the web services to expose the model to the outside world. We expose an HTTP post API against which the user can upload a input image and request for executing the model. The web service API upon receiving the request from the client, will store

the uploaded image inside the '/root/or/input' location of the cloud machine and then it invokes the ObjectRecognition.py program by specifying this input file. The output image will be stored inside '/root/or/result' location inside the cloud server. To download this image into user's machine, the web service will provide another URL as a response by clicking on which the image gets downloaded to the client's machine.

Module2

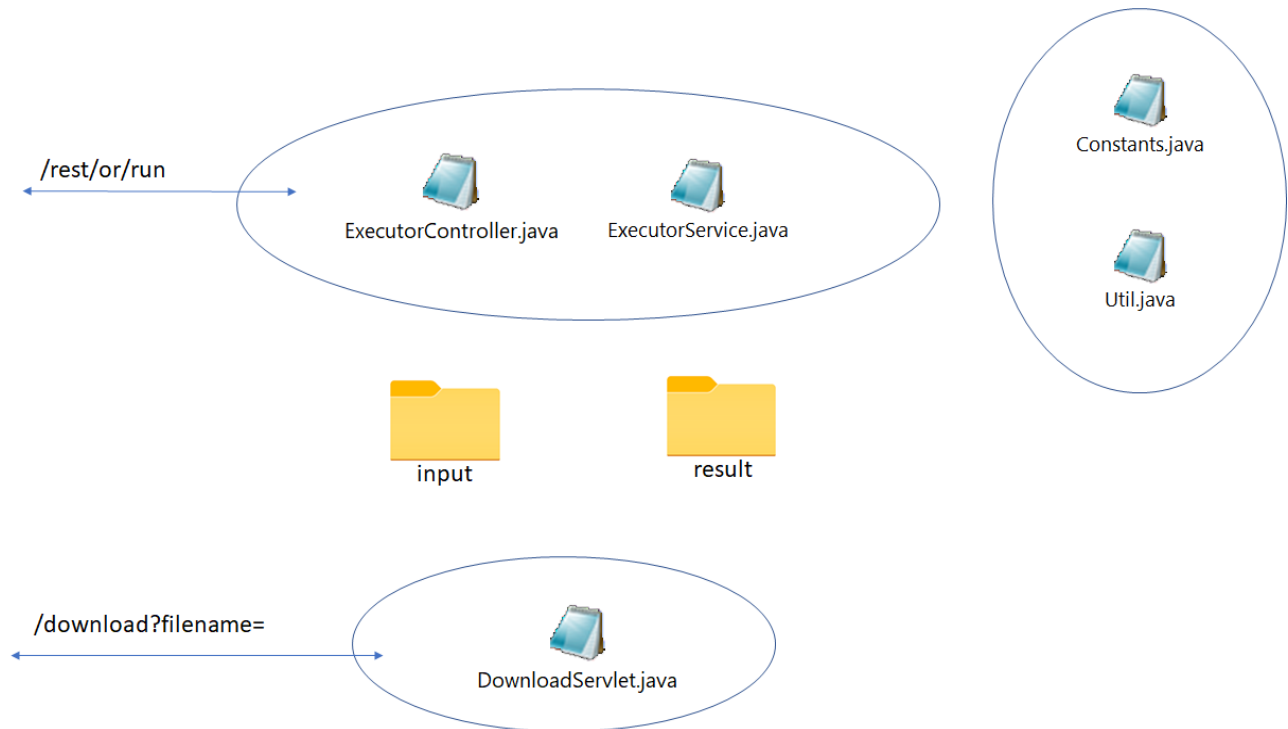


Figure 5.3 Web services

In this module, we implement the web services to expose the model to the outside world. We expose an HTTP post API against which the user can upload a input image and request for executing the model. The web service API upon receiving the request from the client, will store the uploaded image inside the '/root/or/input' location of the cloud machine and then it invokes the ObjectRecognition.py program by specifying this input file. The output image will be stored inside '/root/or/result' location inside the cloud server. To download this image into user's machine, the web service will provide another URL as a response by clicking on which the image gets downloaded to the client's machine.

Module 3: User Profile Operations

This module implements the basic user profile operations on the prototype application. The user profile operations include creating a new account, logging in to the existing account, logging out, editing the profile, changing the password, and deleting the profile if not needed anymore.

This application is also deployed on the cloud server so that this can be accessed by anyone across the globe using the IP address of this cloud server. The implementation is done using the J2EE architecture and for the database needs we have used SQLITE3.

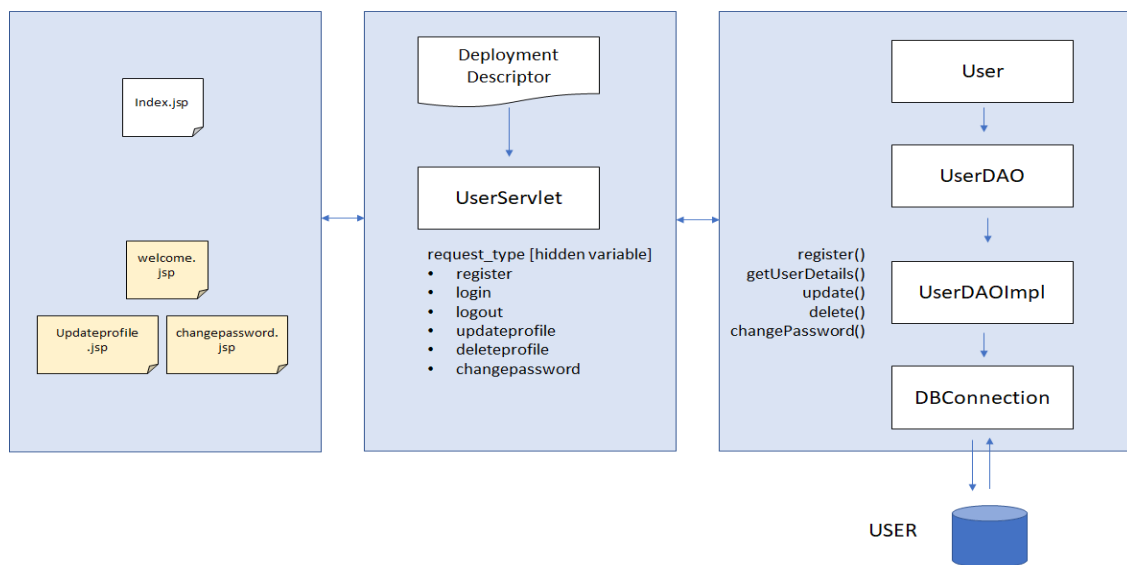


FIGURE 5.4 User Profile Operations

The user profile operations include creating a new account, logging in to the existing account, logging out, editing the profile, changing the password, and deleting the profile if not needed anymore.

This application is also deployed on the cloud server so that this can be accessed by anyone across the globe using the IP address of this cloud server. The implementation is done using the J2EE architecture and for the database needs we have used SQLITE3.

Module 4: Endpoint Configuration and Execution

In this module, the user can configure the model web service so that the prototype application will be capable of communicating with the model without having any issues. For configuring the model, the user will have to provide the Host name, Port number, Application Name and the Context root of the Model Web services. After configuring the endpoint, the user can then execute the algorithm by uploading a sample input image in the Execution component of the project.

This prototype application will receive the image and then forwards it to the configured web service and invokes the algorithm. Soon after that, the prototype application continues polling every 5 seconds to check if the results are available or not. Once available, it renders the result as an image in the HTML screen. The application will also allow the user to download the image into their system.

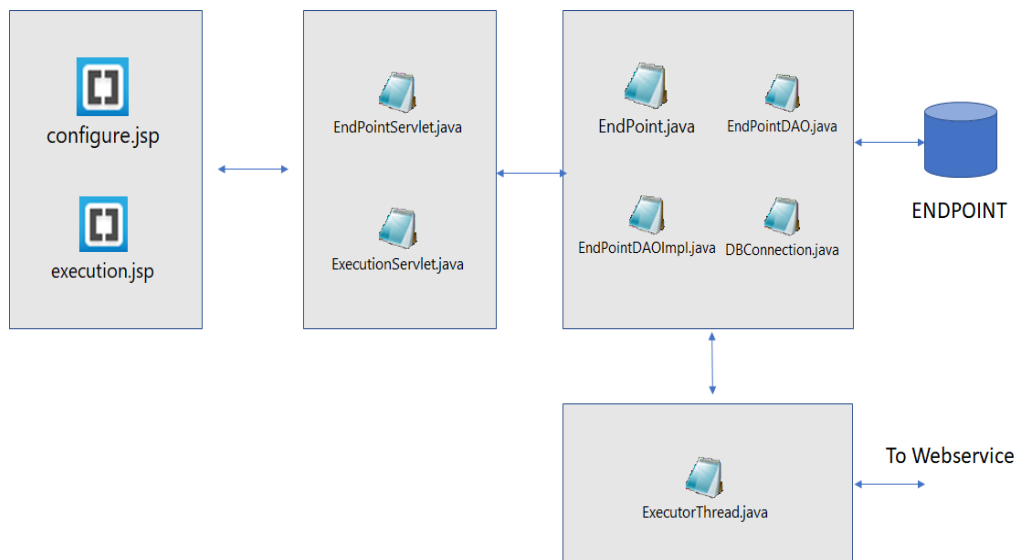


Figure 5.5 Endpoint Configuration and Execution

In this module, the user can configure the model web service so that the prototype application will be capable of communicating with the model without having any issues. For configuring the model, the user will have to provide the Host name, Port number, Application Name and the Context root of the Model Web services. After configuring the endpoint, the user can then execute the algorithm by uploading a sample input image in the Execution component of project.

5.2.2 Mode of operation of a System

The system is capable of operating in two different modes

<ul style="list-style-type: none">• Users Profile Operations	Here, the end users can perform various operations on their profiles. Firstly, the users can register a new account and thus getting an access to the portal. And then the users can login to their accounts using the registered email ID and password to access various other divisions in the portal. The users can then choose to update their profile by providing the new values to the fields they have provided during the registration phase, or the user can wish to change their password by providing their old password and new password. The user can also opt to delete their accounts in case they wish to no longer access our portal. The user can also logout from the portal to make sure the session created for them during login is terminated
<ul style="list-style-type: none">• Blockchain implementation	Here, we implement the core Distributed Ledger network (Blockchain Architecture). We also create an interface to the users where they can setup the blockchain node by entering its IP address. Users can add as many nodes as they want. More the nodes, better the security.
<ul style="list-style-type: none">• Surveillance Application	Here, we implement the application which communicates with CCTV/ Web Camera to capture the video frames. This can be set-up by the users by providing the IP address of the node where CCTV is pushing the video streams.
<ul style="list-style-type: none">• Blockchain Service Implementation	Here, we provide couple of services w.r.t blockchain. The first service is called 'Video Write' service which will be used by the surveillance application to write the videos to

	blockchain network. The second service is called 'Video Read' service which allows the authorized users to download the video frames from the blockchain network.
<ul style="list-style-type: none"> • Surveillance data Access implementation 	Here, we implement the Authorization mechanism to the Blockchain data. The authorized users can then read the video frames from Blockchain network using the previous module.

Table 5.1 Mode of operations

5.2.3 User operations

<ul style="list-style-type: none"> • Account Operations 	User must be creating a new account in our portal to get access to the rest of the modules. The user can also perform various other account operations like retrieving the forgotten password, login, logout, delete profile, change password, and edit profile
<ul style="list-style-type: none"> • Blockchain implementation 	Here the admin can perform the nodes addition or removal operations
<ul style="list-style-type: none"> • Surveillance Application 	Here the end users can access the recorded frames providing his/her credentials
<ul style="list-style-type: none"> • Blockchain Service Implementation 	Here the end users will be exposed to the webservices provided by the blockchain
<ul style="list-style-type: none"> • Surveillance data Access implementation 	Here the end users can manage their accounts.

Table 5.2 User operations

CHAPTER 6:

SYSTEM DESIGN

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

6.1 Data Flow Diagram

A Data Flow Diagram (DFD) is the graphical representation of the “flow” of data through an information system, modeling its process aspects. A DFD is very useful in understanding a system and can be efficiently used during analysis. A DFD shows the flow of data through a system. It views a system as a function that transforms the inputs into desired outputs. Any complex systems will not perform this transformation in a single step and a data will typically undergo a series of transformations before it becomes the output.

DFD – 0

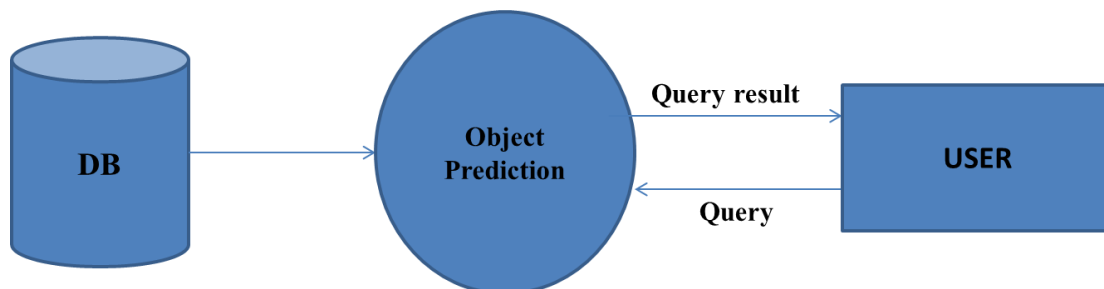


Figure 6.1: Level 0 Data Flow Diagram

DFD – 1

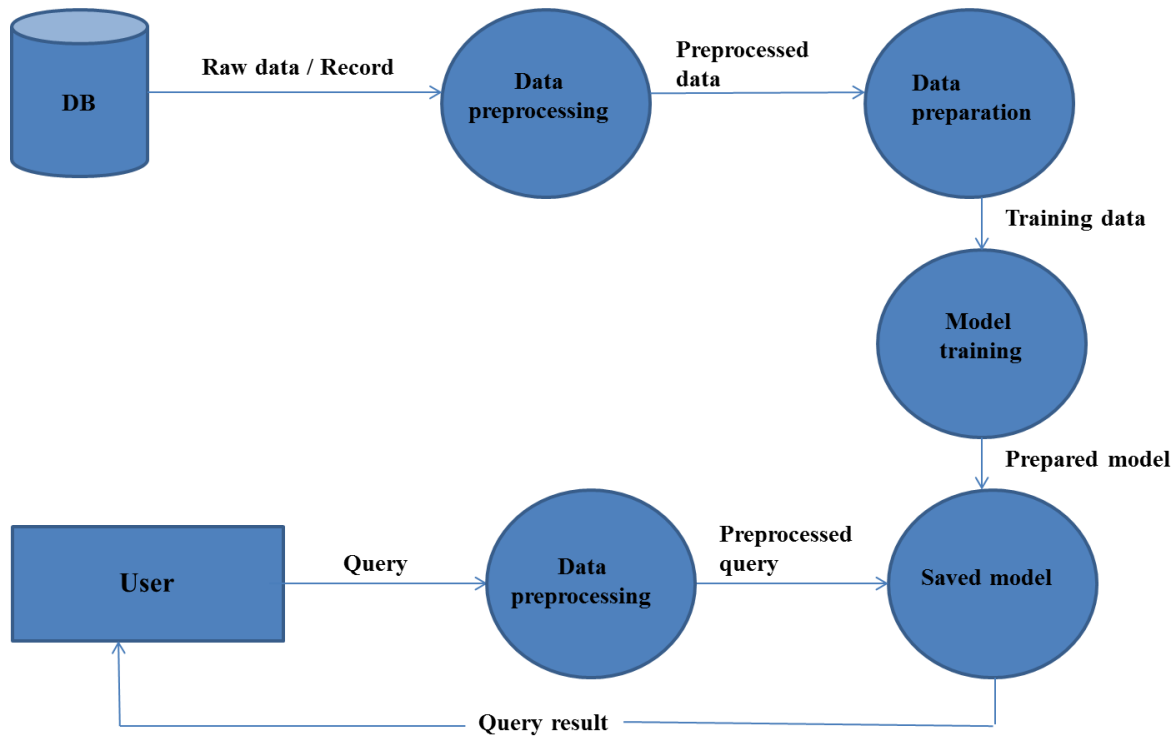


Figure 6.2: Level Data Flow Diagram

6.2 Flow Chart Diagram

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

Flowcharts are used in designing and documenting simple processes or programs. Like other types of diagrams, they help visualize what is going on and thereby help understand a process, and perhaps also find less-obvious features within the process, like flaws and bottlenecks. There are different types of flowcharts: each type has its own set of boxes and notations.

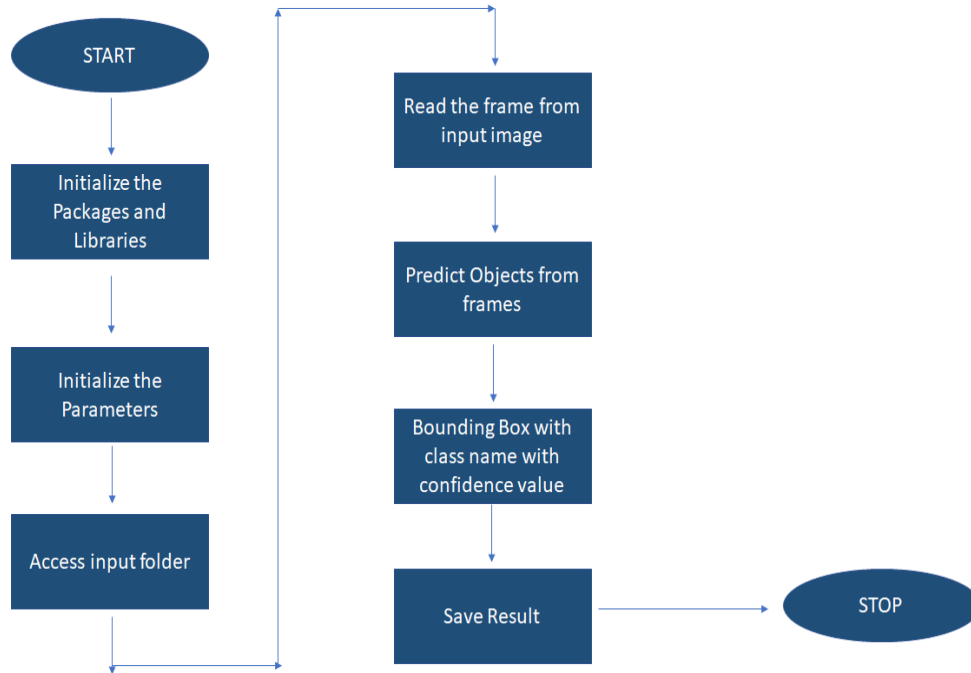


Figure 6.3: Flow Chart Diagram

6.3 Use Case Diagram

The use case diagram was used to identify the primary elements and processes that make up the system. The key elements are referred to as "actors" and the processes are called "use cases." This type of diagram serves to identify the primary elements and processes that form the system. "The use case diagram can be taken as a simple scene which describes what the user expects from the system. Each use case diagram represents a discrete task which involves external interactions with the system.

The use case diagram also captures the functional aspects of a system. More specifically, it captures the business processes carried out in the system. By discussing the functionality and processes of the system, you discover the important features of the system you design. The use case diagram also captures the functional aspects of a system. More specifically, it captures the processes of work performed in the system. By discussing the functionality and processes of the system, you discover the important features of the system you design.

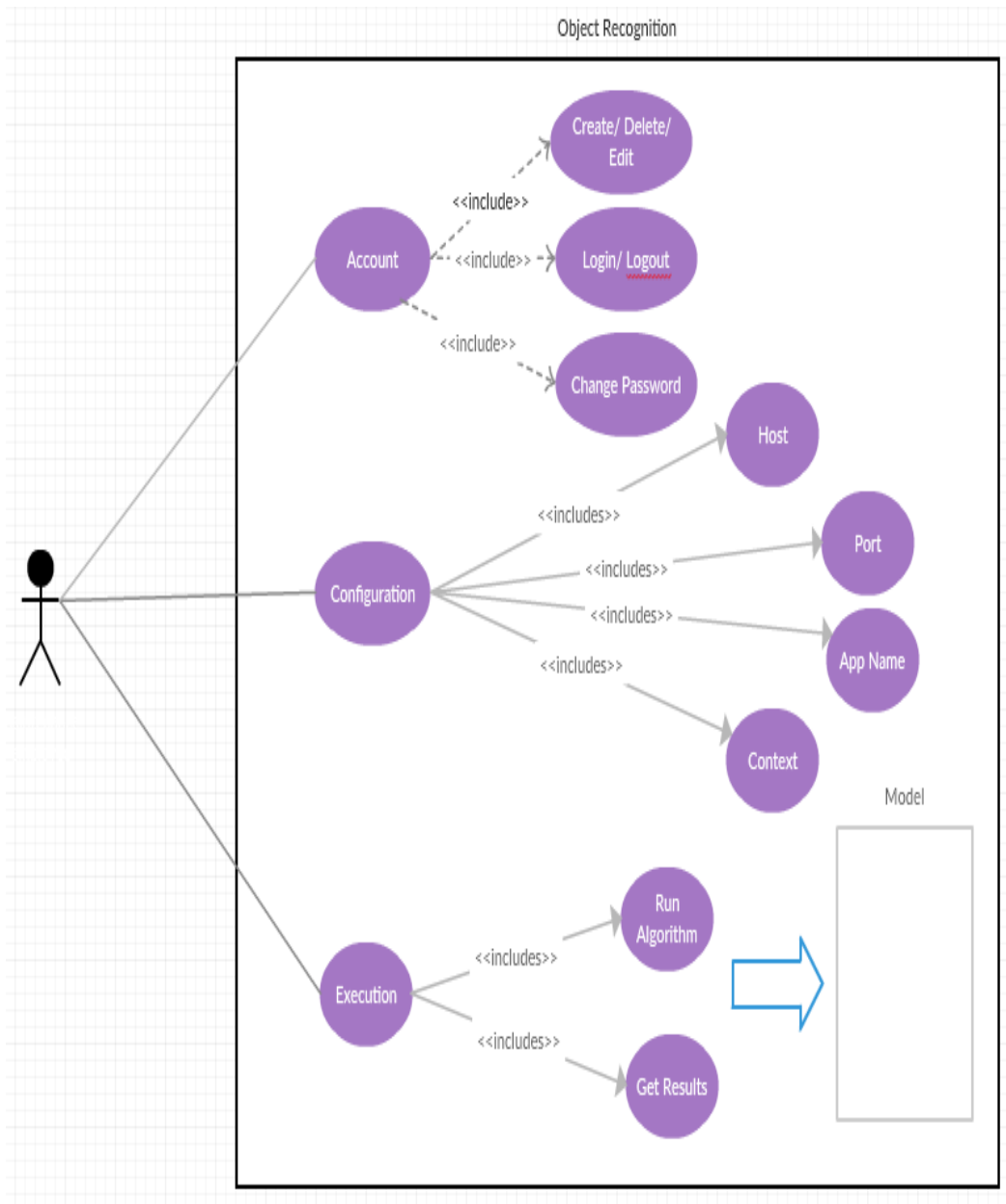


Figure 6.4: Use Case Diagram

Chapter 7:

IMPLEMENTATION

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy. In other words, an implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through programming and deployment. Many implementations may exist for a given specification or standard.

Implementation is one of the most important phases of the Software Development Life Cycle (SDLC). It encompasses all the processes involved in getting new software or hardware operating properly in its environment, including installation, configuration, and running, testing, and making necessary changes. Specifically, it involves coding the system using a particular programming language and transferring the design into an actual working system.

This phase of the system is conducted with the idea that whatever is designed should be implemented; keeping in mind that it fulfills user requirements, objective and scope of the system. The implementation phase produces the solution to the user problem.

7.1 Implementation Support

7.1.1 Installation of Eclipse

The following steps should be followed to install eclipse:

- Installation of JVM: Regardless of the operating system, some Java virtual machine (JVM) has to be installed. A Java Runtime Environment (JRE), or a Java Development Kit (JDK) can be installed depending on what is to be done with Eclipse. If Eclipse is intended for Java development, then a JDK (the JDK includes--among other useful things--the source code for the standard Java libraries)has to be installed.
- Download Eclipse from the Eclipse Downloads Page.

- The download will be delivered as a compressed (i.e. a ".zip", or ".tar.gz") file. Decompress this file into the directory of your choice (e.g. "c:\Program Files\Eclipse Indigo" on Windows). You can optionally create a shortcut of the executable file ("eclipse.exe" on Windows, or "eclipse" on Linux).

7.1.2 Installation of Apache Tomcat Server

The following steps should be followed to install Apache Tomcat in Eclipse:

- If Apache Tomcat is not present on the machine, it has to be downloaded and unzipped.
- Start the Eclipse WTP workbench.
- Open Window -> Preferences -> Server -> Installed Run times to create a Tomcat installed runtime.
- Click on Add to open the New Server Runtime dialog, then select your runtime under Apache (Apache Tomcat v7.0 in this project).
- Ensure the selected JRE is a full JDK and is of a version that will satisfy Apache Tomcat (this scenario was written using SUN JDK 1.6.029). If necessary, you can click on Installed JREs to add JDKs to Eclipse.
- Click Finish. Click Next, and fill in your Tomcat installation directory.

7.1.3 Install Python Environment on Ubuntu

To see which version of Python 3 you have installed, open a command prompt and run

```
$ python3 --version
```

If you are using Ubuntu 16.10 or newer, then you can easily install Python 3.6 with the following commands:

```
$ sudo apt-get update
```

```
$ sudo apt-get install python3.6
```

If you're using another version of Ubuntu (e.g. the latest LTS release), we recommend using the dead snakes PPA to install Python 3.6:

```
$ sudo apt-get install software-properties-common
```

```
$ sudo add-apt-repository ppa:deadsnakes/ppa
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install python3.6
```

If you are using other Linux distribution, chances are you already have Python 3 pre-installed as

well. If not, use your distribution's package manager. For example on Fedora, you would use dnf:

```
$ sudo dnf install python3
```

Note that if the version of the python3 package is not recent enough for you, there may be ways of installing more recent versions as well, depending on your distribution. For example installing the python36 package on Fedora 25 to get Python 3.6. If you are a Fedora user, you might want to read about multiple Python versions available in Fedora.

7.1.4 Installing Jupyter Notebook

While Jupyter runs code in many programming languages, Python is a requirement (Python 3.3 or greater, or Python 2.7) for installing the Jupyter Notebook. For new users, we highly recommend installing Anaconda. Anaconda conveniently installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

Use the following installation steps:

- Download Anaconda. We recommend downloading Anaconda's latest Python 3 version (currently Python 3.7).
- Install the version of Anaconda which you downloaded, following the instructions on the download page.

7.2 PSEUDOCODE

Yolo is an algorithm that uses convolutional neural networks for object detection. So what's great about object detection? In comparison to recognition algorithms, a detection algorithm does not only predict class labels, but detects locations of objects as well.

To build Yolo we're going to need Tensorflow (deep learning), NumPy (numerical computation) and Pillow (image processing) libraries. Also we're going to use seaborn's colour palette for bounding boxes colours. Finally, let's import IPython function `display()` to display images in the notebook.

```
Import tensorflow as tf
```

```
Import numpy as np
```



```
from PIL import Image, ImageDraw, ImageFont
from IPython.display import display
from seaborn import color_palette
import cv2
```

Next, some configurations for Yolo.

```
_BATCH_NORM_DECAY = 0.9
_BATCH_NORM_EPSILON = 1e-05
_LEAKY_RELU = 0.1
_ANCHORS = [(10, 13), (16, 30), (33, 23),
             (30, 61), (62, 45), (59, 119),
             (116, 90), (156, 198), (373, 326)]
_MODEL_SIZE = (416, 416)
```

`_MODEL_SIZE` (refers to the input size of the model.)

Batch normalization

Almost every convolutional layer in Yolo has batch normalization after it. It helps the model train faster and reduces variance between units (and total variance as well). Batch normalization is defined as follows.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$
$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$
$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

Batch norm and fixed padding

It's useful to define `batch_norm` function since the model uses batch norms with shared parameters heavily. Also, same as ResNet, Yolo uses convolution with fixed padding, which means that padding is defined only by the size of the kernel.

```
def batch_norm(inputs, training, data_format):
```

```
    """Performs a batch normalization using a standard set of parameters."""
```

```
    return tf.layers.batch_normalization(
```

```
        inputs=inputs, axis=1 if data_format == 'channels_first' else 3,
```

```
        momentum=_BATCH_NORM_DECAY, epsilon=_BATCH_NORM_EPSILON,
```

```
        scale=True, training=training)
```

```
def fixed_padding(inputs, kernel_size, data_format):
```

```
    """ResNet implementation of fixed padding.
```

```
    Pads the input along the spatial dimensions independently of input size.
```

```
    Args:
        inputs: Tensor input to be padded.
        kernel_size: The kernel to be used in the conv2d or max_
pool2d.
        data_format: The input format.
    Returns:
        A tensor with the same format as the input.
    """
    pad_total = kernel_size - 1
    pad_beg = pad_total // 2
    pad_end = pad_total - pad_beg

    if data_format == 'channels_first':
        padded_inputs = tf.pad(inputs, [[0, 0], [0, 0],
                                         [pad_beg, pad_end],
                                         [pad_beg, pad_end]])
    else:
        padded_inputs = tf.pad(inputs, [[0, 0], [pad_beg, pad_end],
                                         [pad_beg, pad_end], [0, 0]])
    return padded_inputs

def conv2d_fixed_padding(inputs, filters, kernel_size, data_format, strides=1):
    """Strided 2-D convolution with explicit padding."""
    if strides > 1:
        inputs = fixed_padding(inputs, kernel_size, data_format)

    return tf.layers.conv2d(
        inputs=inputs, filters=filters, kernel_size=kernel_size,
        strides=strides, padding=('SAME' if strides == 1 else 'VALID'),
```

```
use_bias=False, data_format=data_format)
```

7.2.2 Feature extraction: Darknet-53

For feature extraction Yolo uses Darknet-53 neural net pretrained on ImageNet. Same as ResNet, Darknet-53 has shortcut (residual) connections, which help information from earlier layers flow further. We omit the last 3 layers (Avgpool, Connected and Softmax) since we only need the features.

In []:

```
def darknet53_residual_block(inputs, filters, training, data_format,
                             strides=1):
    """Creates a residual block for Darknet."""
    shortcut = inputs

    inputs = conv2d_fixed_padding(
        inputs, filters=filters, kernel_size=1, strides=strides,
        data_format=data_format)
    inputs = batch_norm(inputs, training=training, data_format=data_format)
    inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

    inputs = conv2d_fixed_padding(
        inputs, filters=2 * filters, kernel_size=3, strides=strides,
        data_format=data_format)
    inputs = batch_norm(inputs, training=training, data_format=data_format)
    inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

    inputs += shortcut

    return inputs

def darknet53(inputs, training, data_format):
```

```
"""Creates Darknet53 model for feature extraction."""
inputs = conv2d_fixed_padding(inputs, filters=32, kernel_size=3,
                              data_format=data_format)
inputs = batch_norm(inputs, training=training, data_format=data_format)
inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)
inputs = conv2d_fixed_padding(inputs, filters=64, kernel_size=3,
                              strides=2, data_format=data_format)
inputs = batch_norm(inputs, training=training, data_format=data_format)
inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

inputs = darknet53_residual_block(inputs, filters=32, training=training,
                                  data_format=data_format)

inputs = conv2d_fixed_padding(inputs, filters=128, kernel_size=3,
                              strides=2, data_format=data_format)
inputs = batch_norm(inputs, training=training, data_format=data_format)
inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

for _ in range(2):
    inputs = darknet53_residual_block(inputs, filters=64,
                                      training=training,
                                      data_format=data_format)

inputs = conv2d_fixed_padding(inputs, filters=256, kernel_size=3,
                              strides=2, data_format=data_format)
inputs = batch_norm(inputs, training=training, data_format=data_format)
inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

for _ in range(8):
    inputs = darknet53_residual_block(inputs, filters=128,
                                      training=training,
```

```
        data_format=data_format)

route1 = inputs

inputs = conv2d_fixed_padding(inputs, filters=512, kernel_size=3,
                               strides=2, data_format=data_format)
inputs = batch_norm(inputs, training=training, data_format=data_format)
inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

for _ in range(8):
    inputs = darknet53_residual_block(inputs, filters=256,
                                       training=training,
                                       data_format=data_format)

route2 = inputs

inputs = conv2d_fixed_padding(inputs, filters=1024, kernel_size=3,
                               strides=2, data_format=data_format)
inputs = batch_norm(inputs, training=training, data_format=data_format)
inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

for _ in range(4):
    inputs = darknet53_residual_block(inputs, filters=512,
                                       training=training,
                                       data_format=data_format)

return route1, route2, inputs
```

7.2.3 Convolution layers

Yolo has a large number of convolutional layers. It's useful to group them in blocks.

In []:

```
def yolo_convolution_block(inputs, filters, training, data_format):
    """Creates convolution operations layer used after Darknet."""
    inputs = conv2d_fixed_padding(inputs, filters=filters, kernel_size=1,
                                   data_format=data_format)
    inputs = batch_norm(inputs, training=training, data_format=data_format)
    inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

    inputs = conv2d_fixed_padding(inputs, filters=2 * filters, kernel_size=3,
                                   data_format=data_format)
    inputs = batch_norm(inputs, training=training, data_format=data_format)
    inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

    inputs = conv2d_fixed_padding(inputs, filters=filters, kernel_size=1,
                                   data_format=data_format)
    inputs = batch_norm(inputs, training=training, data_format=data_format)
    inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

    inputs = conv2d_fixed_padding(inputs, filters=2 * filters, kernel_size=3,
                                   data_format=data_format)
    inputs = batch_norm(inputs, training=training, data_format=data_format)
    inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

    inputs = conv2d_fixed_padding(inputs, filters=filters, kernel_size=1,
                                   data_format=data_format)
    inputs = batch_norm(inputs, training=training, data_format=data_format)
    inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

    route = inputs
```

```
inputs = conv2d_fixed_padding(inputs, filters=2 * filters, kernel_size=3,
                              data_format=data_format)
inputs = batch_norm(inputs, training=training, data_format=data_format)
inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)

return route, inputs
```

7.2.4 Detection layers

Yolo has 3 detection layers, that detect on 3 different scales using respective anchors. For each cell in the feature map the detection layer predicts $n_anchors * (5 + n_classes)$ values using 1x1 convolution. For each scale we have $n_anchors = 3.5 + n_classes$ means that respectively to each of 3 anchors we are going to predict 4 coordinates of the box, its confidence score (the probability of containing an object) and class probabilities.

In []:

```
def yolo_layer(inputs, n_classes, anchors, img_size, data_format):
    """Creates Yolo final detection layer.

    Detects boxes with respect to anchors.

    Args:
        inputs: Tensor input.
        n_classes: Number of labels.
        anchors: A list of anchor sizes.
        img_size: The input size of the model.
        data_format: The input format.

    Returns:
        Tensor output.
    """
```



```
n_anchors = len(anchors)

inputs = tf.layers.conv2d(inputs, filters=n_anchors * (5 + n_classes),
                          kernel_size=1, strides=1, use_bias=True,
                          data_format=data_format)

shape = inputs.get_shape().as_list()
grid_shape = shape[2:4] if data_format == 'channels_first' else shape[1:3]
if data_format == 'channels_first':
    inputs = tf.transpose(inputs, [0, 2, 3, 1])
inputs = tf.reshape(inputs, [-1, n_anchors * grid_shape[0] * grid_shape[1],
                             5 + n_classes])

strides = (img_size[0] // grid_shape[0], img_size[1] // grid_shape[1])

box_centers, box_shapes, confidence, classes = \
    tf.split(inputs, [2, 2, 1, n_classes], axis=-1)

x = tf.range(grid_shape[0], dtype=tf.float32)
y = tf.range(grid_shape[1], dtype=tf.float32)
x_offset, y_offset = tf.meshgrid(x, y)
x_offset = tf.reshape(x_offset, (-1, 1))
y_offset = tf.reshape(y_offset, (-1, 1))
x_y_offset = tf.concat([x_offset, y_offset], axis=-1)
x_y_offset = tf.tile(x_y_offset, [1, n_anchors])
x_y_offset = tf.reshape(x_y_offset, [1, -1, 2])
box_centers = tf.nn.sigmoid(box_centers)
box_centers = (box_centers + x_y_offset) * strides

anchors = tf.tile(anchors, [grid_shape[0] * grid_shape[1], 1])
box_shapes = tf.exp(box_shapes) * tf.to_float(anchors)
```

```
confidence = tf.nn.sigmoid(confidence)

classes = tf.nn.sigmoid(classes)

inputs = tf.concat([box_centers, box_shapes,
                   confidence, classes], axis=-1)

return inputs
```

7.2.5 Upsample layer

In order to concatenate with shortcut outputs from Darknet-53 before applying detection on a different scale, we are going to upsample the feature map using nearest neighbor interpolation.

In []:

```
def upsample(inputs, out_shape, data_format):
    """Upsamples to `out_shape` using nearest neighbor interpolation."""
    if data_format == 'channels_first':
        inputs = tf.transpose(inputs, [0, 2, 3, 1])
        new_height = out_shape[3]
        new_width = out_shape[2]
    else:
        new_height = out_shape[2]
        new_width = out_shape[1]

    inputs = tf.image.resize_nearest_neighbor(inputs, (new_height, new_width))

    if data_format == 'channels_first':
        inputs = tf.transpose(inputs, [0, 3, 1, 2])
```

```
return inputs
```

7.2.6 Non-max suppression

The model is going to produce a lot of boxes, so we need a way to discard the boxes with low confidence scores. Also, to avoid having multiple boxes for one object, we will discard the boxes with high overlap as well using non-max suppression for each class.

In []:

```
def build_boxes(inputs):
    """Computes top left and bottom right points of the boxes."""
    center_x, center_y, width, height, confidence, classes = \
        tf.split(inputs, [1, 1, 1, 1, 1, -1], axis=-1)

    top_left_x = center_x - width / 2
    top_left_y = center_y - height / 2
    bottom_right_x = center_x + width / 2
    bottom_right_y = center_y + height / 2

    boxes = tf.concat([top_left_x, top_left_y,
                       bottom_right_x, bottom_right_y,
                       confidence, classes], axis=-1)

    return boxes


def non_max_suppression(inputs, n_classes, max_output_size, iou_threshold,
                       confidence_threshold):
    """Performs non-max suppression separately for each class.

    Args:
        inputs: Tensor input.
        n_classes: Number of classes.
```

```
max_output_size: Max number of boxes to be selected for
each class.

iou_threshold: Threshold for the IOU.

confidence_threshold: Threshold for the confidence score
.

Returns:
    A list containing class-to-boxes dictionaries
    for each sample in the batch.
    """
batch = tf.unstack(inputs)
boxes_dicts = []
for boxes in batch:
    boxes = tf.boolean_mask(boxes, boxes[:, 4] > confidence_threshold)
    classes = tf.argmax(boxes[:, 5:], axis=-1)
    classes = tf.expand_dims(tf.to_float(classes), axis=-1)
    boxes = tf.concat([boxes[:, :5], classes], axis=-1)

boxes_dict = dict()
for cls in range(n_classes):
    mask = tf.equal(boxes[:, 5], cls)
    mask_shape = mask.get_shape()
    if mask_shape.ndims != 0:
        class_boxes = tf.boolean_mask(boxes, mask)
        boxes_coords, boxes_conf_scores, _ = tf.split(class_boxes,
                                                    [4, 1, -1],
                                                    axis=-1)
        boxes_conf_scores = tf.reshape(boxes_conf_scores, [-1])
        indices = tf.image.non_max_suppression(boxes_coords,
                                                boxes_conf_scores,
                                                max_output_size,
                                                iou_threshold)
```

```
class_boxes = tf.gather(class_boxes, indices)
boxes_dict[cls] = class_boxes[:, :5]

boxes_dicts.append(boxes_dict)

return boxes_dicts
```

7.2.7 Final model class

Finally, let's define the model class using all of the layers described previously.

In []:

```
class Yolo_v3:
    """Yolo v3 model class."""

    def __init__(self, n_classes, model_size, max_output_size, iou_threshold,
                  confidence_threshold, data_format=None):
        """Creates the model.

        Args:
            n_classes: Number of class labels.
            model_size: The input size of the model.
            max_output_size: Max number of boxes to be selected
for each class.
            iou_threshold: Threshold for the IOU.
            confidence_threshold: Threshold for the confidence s
core.

            data_format: The input format.

        Returns:
            None.

        """
        if not data_format:
```

```

    if tf.test.is_built_with_cuda():
        data_format = 'channels_first'
    else:
        data_format = 'channels_last'

self.n_classes = n_classes
self.model_size = model_size
self.max_output_size = max_output_size
self.iou_threshold = iou_threshold
self.confidence_threshold = confidence_threshold
self.data_format = data_format

def __call__(self, inputs, training):
    """Add operations to detect boxes for a batch of input images
    .

    Args:
        inputs: A Tensor representing a batch of input image
s.

        training: A boolean, whether to use in training or i
nference mode.

    Returns:
        A list containing class-to-boxes dictionaries
        for each sample in the batch.
    """
    with tf.variable_scope('yolo_v3_model'):
        if self.data_format == 'channels_first':
            inputs = tf.transpose(inputs, [0, 3, 1, 2])

        inputs = inputs / 255

```

```
route1, route2, inputs = darknet53(inputs, training=training,
                                   data_format=self.data_format)

route, inputs = yolo_convolution_block(
    inputs, filters=512, training=training,
    data_format=self.data_format)
detect1 = yolo_layer(inputs, n_classes=self.n_classes,
                    anchors=_ANCHORS[6:9],
                    img_size=self.model_size,
                    data_format=self.data_format)

inputs = conv2d_fixed_padding(route, filters=256, kernel_size=1,
                              data_format=self.data_format)
inputs = batch_norm(inputs, training=training,
                   data_format=self.data_format)
inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)
upsample_size = route2.get_shape().as_list()
inputs = upsample(inputs, out_shape=upsample_size,
                 data_format=self.data_format)
axis = 1 if self.data_format == 'channels_first' else 3
inputs = tf.concat([inputs, route2], axis=axis)
route, inputs = yolo_convolution_block(
    inputs, filters=256, training=training,
    data_format=self.data_format)
detect2 = yolo_layer(inputs, n_classes=self.n_classes,
                    anchors=_ANCHORS[3:6],
                    img_size=self.model_size,
                    data_format=self.data_format)

inputs = conv2d_fixed_padding(route, filters=128, kernel_size=1,
```

```
        data_format=self.data_format)
    inputs = batch_norm(inputs, training=training,
        data_format=self.data_format)
    inputs = tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)
    upsample_size = route1.get_shape().as_list()
    inputs = upsample(inputs, out_shape=upsample_size,
        data_format=self.data_format)
    inputs = tf.concat([inputs, route1], axis=axis)
    route, inputs = yolo_convolution_block(
        inputs, filters=128, training=training,
        data_format=self.data_format)
    detect3 = yolo_layer(inputs, n_classes=self.n_classes,
        anchors=_ANCHORS[0:3],
        img_size=self.model_size,
        data_format=self.data_format)

    inputs = tf.concat([detect1, detect2, detect3], axis=1)

    inputs = build_boxes(inputs)

    boxes_dicts = non_max_suppression(
        inputs, n_classes=self.n_classes,
        max_output_size=self.max_output_size,
        iou_threshold=self.iou_threshold,
        confidence_threshold=self.confidence_threshold)

    return boxes_dicts
```

7.2.8 Utility functions

Here are some utility functions that will help us load images as NumPy arrays, load class names from the official file and draw the predicted boxes.

In []:

```
def load_images(img_names, model_size):
    """Loads images in a 4D array.

    Args:
        img_names: A list of images names.
        model_size: The input size of the model.
        data_format: A format for the array returned
                     ('channels_first' or 'channels_last').

    Returns:
        A 4D NumPy array.
    """
    imgs = []

    for img_name in img_names:
        img = Image.open(img_name)
        img = img.resize(size=model_size)
        img = np.array(img, dtype=np.float32)
        img = np.expand_dims(img, axis=0)
        imgs.append(img)

    imgs = np.concatenate(imgs)

    return imgs


def load_class_names(file_name):
    """Returns a list of class names read from `file_name`."""
    with open(file_name, 'r') as f:
        class_names = f.read().splitlines()
```

```
return class_names
```

```
def draw_boxes(img_names, boxes_dicts, class_names, model_size):
```

```
    """Draws detected boxes.
```

```
    Args:
```

```
        img_names: A list of input images names.
```

```
        boxes_dict: A class-to-boxes dictionary.
```

```
        class_names: A class names list.
```

```
        model_size: The input size of the model.
```

```
    Returns:
```

```
        None.
```

```
    """
```

```
    colors = ((np.array(color_palette("hls", 80)) * 255)).astype(np.uint8)
```

```
    for num, img_name, boxes_dict in zip(range(len(img_names)), img_names,
                                         boxes_dicts):
```

```
        img = Image.open(img_name)
```

```
        draw = ImageDraw.Draw(img)
```

```
        font = ImageFont.truetype(font='../input/futur.ttf,
```

```
                                   size=(img.size[0] + img.size[1]) // 100)
```

```
        resize_factor = \
```

```
            (img.size[0] / model_size[0], img.size[1] / model_size[1])
```

```
        for cls in range(len(class_names)):
```

```
            boxes = boxes_dict[cls]
```

```
            if np.size(boxes) != 0:
```

```
                color = colors[cls]
```

```
                for box in boxes:
```

```
                    xy, confidence = box[:4], box[4]
```

```
                    xy = [xy[i] * resize_factor[i % 2] for i in range(4)]
```

```
x0, y0 = xy[0], xy[1]
thickness = (img.size[0] + img.size[1]) // 200
for t in np.linspace(0, 1, thickness):
    xy[0], xy[1] = xy[0] + t, xy[1] + t
    xy[2], xy[3] = xy[2] - t, xy[3] - t
    draw.rectangle(xy, outline=tuple(color))
text = '{ } { : . 1 f } %'.format(class_names[cls],
                                   confidence * 100)
text_size = draw.textsize(text, font=font)
draw.rectangle(
    [x0, y0 - text_size[1], x0 + text_size[0], y0],
    fill=tuple(color))
draw.text((x0, y0 - text_size[1]), text, fill='black',
          font=font)

display(img)
```

7.2.9 Converting weights to Tensorflow format

Now it's time to load the official weights. We are going to iterate through the file and gradually create `tf.assign` operations.

In []:

```
def load_weights(variables, file_name):
    """Reshapes and loads official pretrained Yolo weights.

    Args:
        variables: A list of tf.Variable to be assigned.
        file_name: A name of a file containing weights.

    Returns:
        A list of assign operations.
```

```
"""  
  
with open(file_name, "rb") as f:  
    # Skip first 5 values containing irrelevant info  
    np.fromfile(f, dtype=np.int32, count=5)  
    weights = np.fromfile(f, dtype=np.float32)  
  
    assign_ops = []  
    ptr = 0  
  
    # Load weights for Darknet part.  
    # Each convolution layer has batch normalization.  
    for i in range(52):  
        conv_var = variables[5 * i]  
        gamma, beta, mean, variance = variables[5 * i + 1:5 * i + 5]  
        batch_norm_vars = [beta, gamma, mean, variance]  
  
        for var in batch_norm_vars:  
            shape = var.shape.as_list()  
            num_params = np.prod(shape)  
            var_weights = weights[ptr:ptr + num_params].reshape(shape)  
            ptr += num_params  
            assign_ops.append(tf.assign(var, var_weights))  
  
        shape = conv_var.shape.as_list()  
        num_params = np.prod(shape)  
        var_weights = weights[ptr:ptr + num_params].reshape(  
            (shape[3], shape[2], shape[0], shape[1]))  
        var_weights = np.transpose(var_weights, (2, 3, 1, 0))  
        ptr += num_params  
        assign_ops.append(tf.assign(conv_var, var_weights))
```

```
# Loading weights for Yolo part.
# 7th, 15th and 23rd convolution layer has biases and no batch norm.
ranges = [range(0, 6), range(6, 13), range(13, 20)]
unnormlized = [6, 13, 20]
for j in range(3):
    for i in ranges[j]:
        current = 52 * 5 + 5 * i + j * 2
        conv_var = variables[current]
        gamma, beta, mean, variance = \
            variables[current + 1:current + 5]
        batch_norm_vars = [beta, gamma, mean, variance]

        for var in batch_norm_vars:
            shape = var.shape.as_list()
            num_params = np.prod(shape)
            var_weights = weights[ptr:ptr + num_params].reshape(shape)
            ptr += num_params
            assign_ops.append(tf.assign(var, var_weights))

        shape = conv_var.shape.as_list()
        num_params = np.prod(shape)
        var_weights = weights[ptr:ptr + num_params].reshape(
            (shape[3], shape[2], shape[0], shape[1]))
        var_weights = np.transpose(var_weights, (2, 3, 1, 0))
        ptr += num_params
        assign_ops.append(tf.assign(conv_var, var_weights))

    bias = variables[52 * 5 + unnormlized[j] * 5 + j * 2 + 1]
    shape = bias.shape.as_list()
    num_params = np.prod(shape)
```

```
var_weights = weights[ptr:ptr + num_params].reshape(shape)
ptr += num_params
assign_ops.append(tf.assign(bias, var_weights))
```

```
conv_var = variables[52 * 5 + unnormalized[j] * 5 + j * 2]
shape = conv_var.shape.as_list()
num_params = np.prod(shape)
var_weights = weights[ptr:ptr + num_params].reshape(
    (shape[3], shape[2], shape[0], shape[1]))
var_weights = np.transpose(var_weights, (2, 3, 1, 0))
ptr += num_params
assign_ops.append(tf.assign(conv_var, var_weights))
```

```
return assign_ops
```

7.2.10 Running the model

Now we can run the model against the user input image

In []:

```
imgnm = sys.argv[1]

img_names = [imgnm]

tf.reset_default_graph()
batch_size = len(img_names)
batch = load_images(img_names, model_size=_MODEL_SIZE)
class_names = load_class_names('/root/or/ObjectRecognition/coco.names')
n_classes = len(class_names)
max_output_size = 10
iou_threshold = 0.5
confidence_threshold = 0.5
```

```
model = Yolo_v3(n_classes=n_classes, model_size=_MODEL_SIZE,
               max_output_size=max_output_size,
               iou_threshold=iou_threshold,
               confidence_threshold=confidence_threshold)

inputs = tf.placeholder(tf.float32, [batch_size, 416, 416, 3])
detections = model(inputs, training=False)
model_vars = tf.global_variables(scope='yolo_v3_model')
assign_ops = load_weights(model_vars, '/root/or/ObjectRecognition/yolov3.weights')
with tf.Session() as sess:
    sess.run(assign_ops)
    detection_result = sess.run(detections, feed_dict={inputs: batch})

draw_boxes(img_names, detection_result, class_names, _MODEL_SIZE)
```

CHAPTER 8 :

PERFORMANCE ANALYSIS

8.1 Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. The system has been verified and validated by running the test data and live data.

Steps	Test Action	Results
Step 1	Enter the URL http://hostname:8080/OR/index.jsp	Index page loaded successfully
Step 2	Click on Register	Register page loaded successfully
Step 3	Create a new account	Account Creation Successful
Step 4	Click on login	Login page loaded
Step 5	Login to an existing account	Login Successful
Step 6	Click on Configuration	Configuration Page Loaded Successfully
Step 7	Enter the details of the web service	Details entered correctly
Step 8	Click on Add button	End Point added successfully
Step 9	Click on Edit	Edit Overlay opened
Step 10	Update the details	Update successful
Step 11	Click on Delete	Deletion of End point Successful
Step 12	Add the end point details again	End point added
Step 13	Click on Execution	Execution page loaded successfully

Step 14	Click on Browse and upload the input image	Image image upload successful
Step 15	Click on Run	The algorithm started running
Step 16	Wait for some time and check the results	Results appeared correct

Table 8.1: Test cases for the project

Steps to perform integration testing:

Step 1: Create a Test Plan

Step 2: Create Test Cases and Test Data

Step 3: Once the components have been integrated execute the test cases

Step 4: Fix the bugs if any and re test the code

Step 5: Repeat the test cycle until the components have been successfully integrated

Name of the Test	Integration testing
Test plan	To check whether the system works properly when all the modules are integrated.
Test Data	Sample image files

Table 8.2: Test cases for integration testing

System testing

Ultimately, software is included with other system components and the set of system validation and integration tests are performed. System testing is a series of different tests whose main aim is to fully exercise the computer-based system. Although each test has a different role all work should verify that all system elements are properly integrated and formed allocated functions.

Name of the Test	System Testing
Item being tested	Over all functioning of GUI with all functions properly linked.
Sample Input	Sample image files
Expected Output	All the modules like working as expected
Actual Output	Application reacts to user inputs in expected manner.
Remarks	Successful

Table 8.3: Test cases for Input-Output

Snapshots

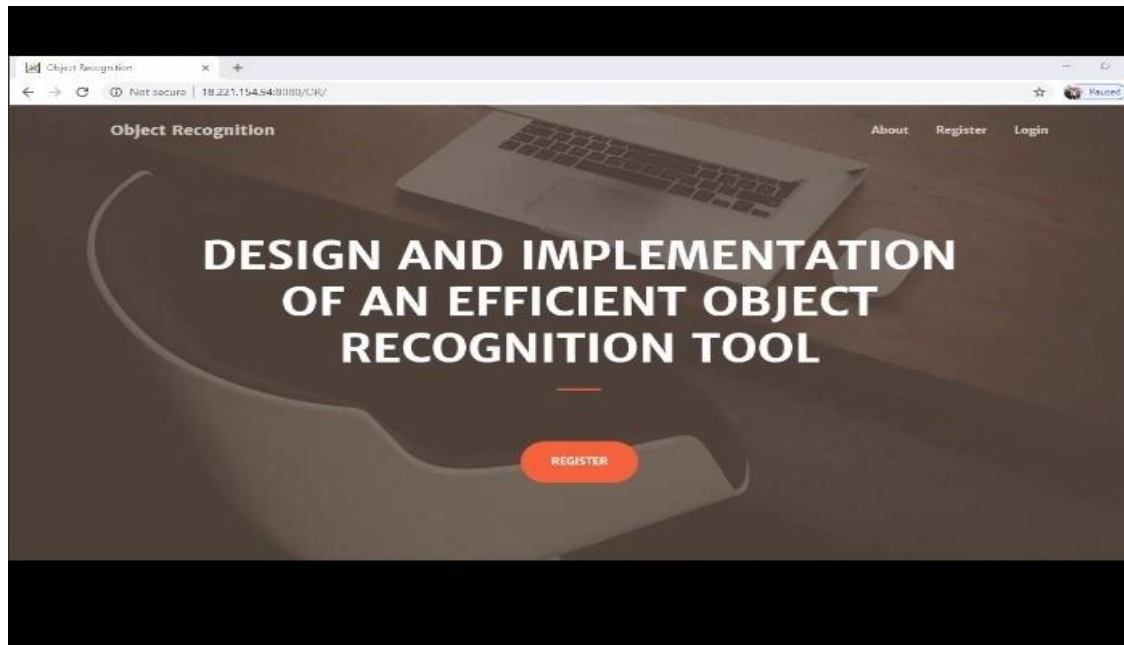


FIGURE 8.1: Main screen

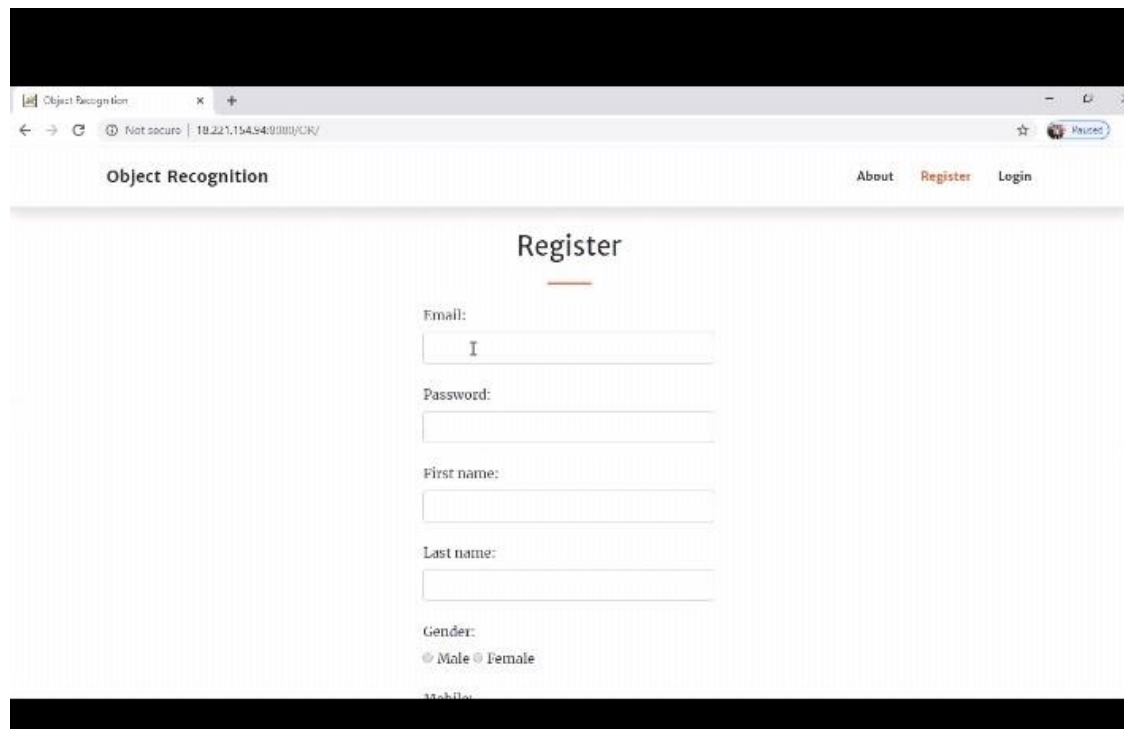
A screenshot of the 'Register' page of the 'Object Recognition' application. The browser's address bar shows '19.221.154.54:8080/CNN/'. The page has a light gray background. The title 'Object Recognition' is in the top left. Navigation links 'About', 'Register', and 'Login' are in the top right. The main heading is 'Register'. Below it are several input fields: 'Email:' with a text box containing 'I', 'Password:' with a text box, 'First name:' with a text box, 'Last name:' with a text box, and 'Gender:' with radio buttons for 'Male' and 'Female'. There is also a 'Submit' button at the bottom.

Figure 8.2: Registration block

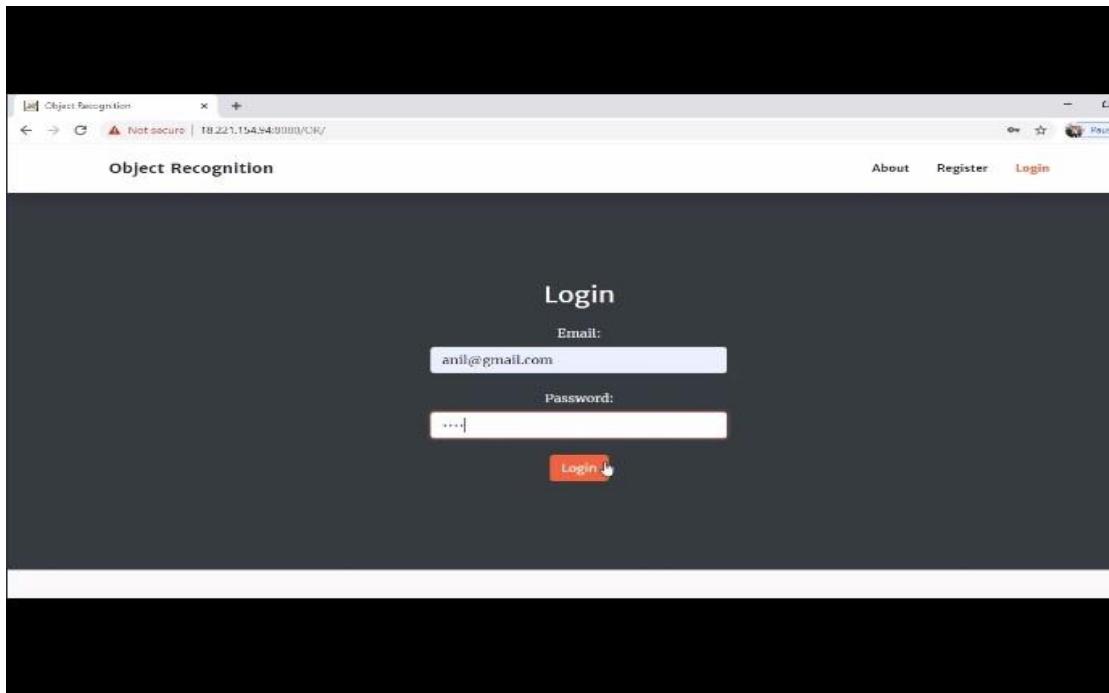


Figure 8.3: Login page

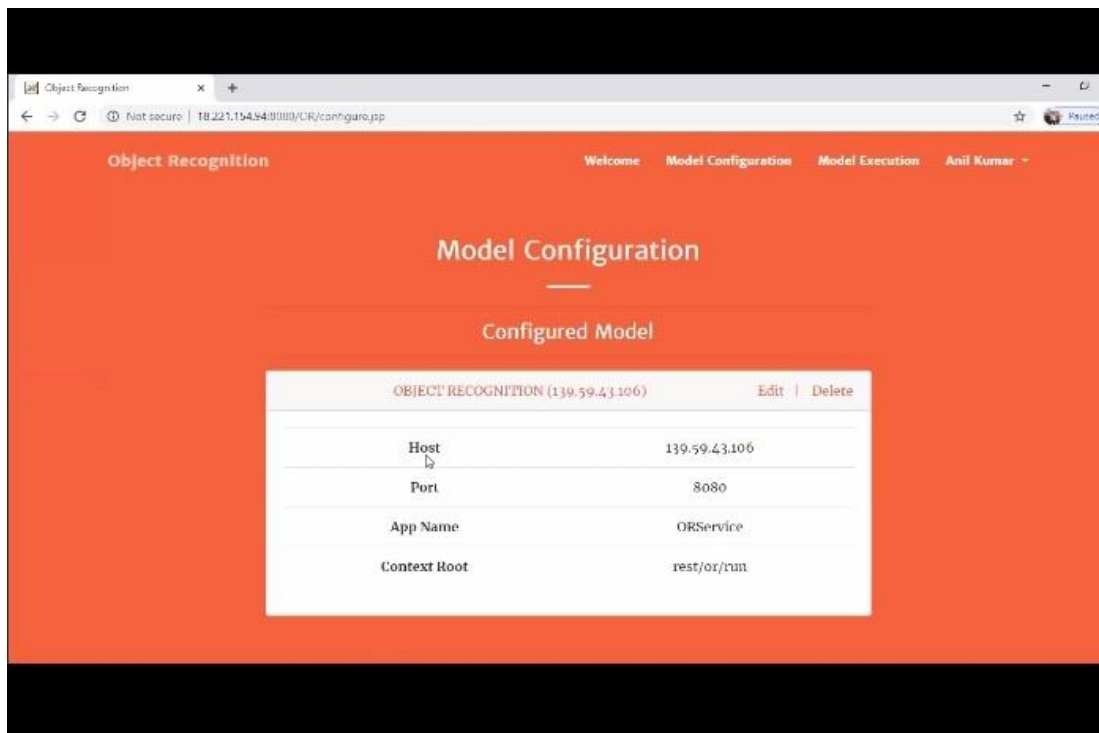


Figure 8.4: Model configuration

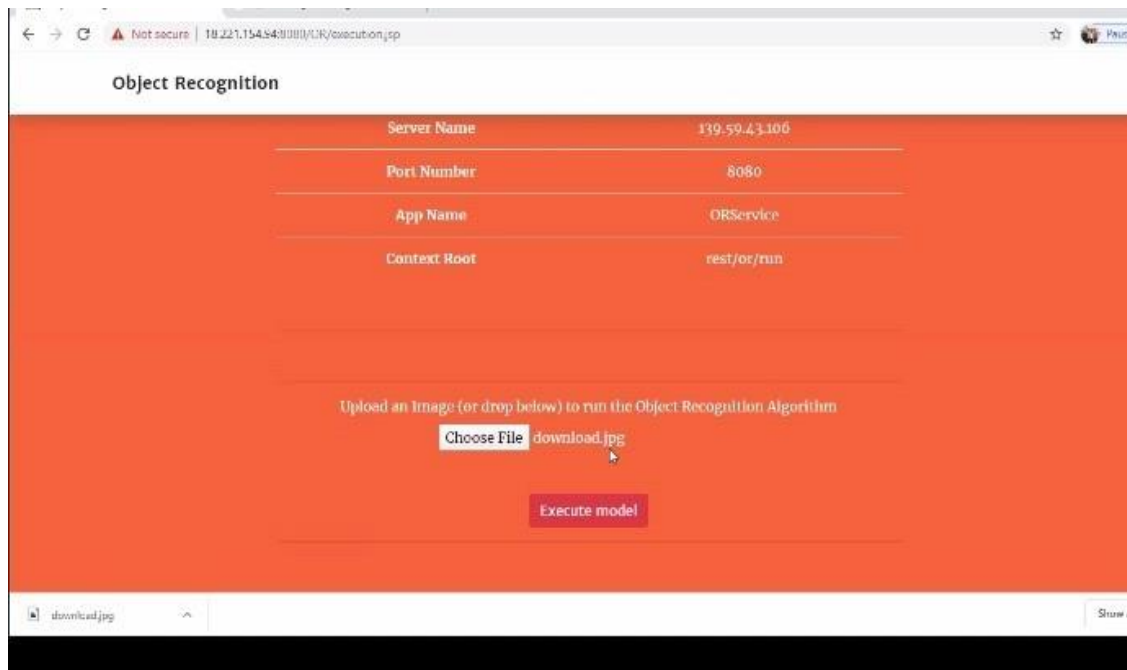


Figure 8.5: Choosing image

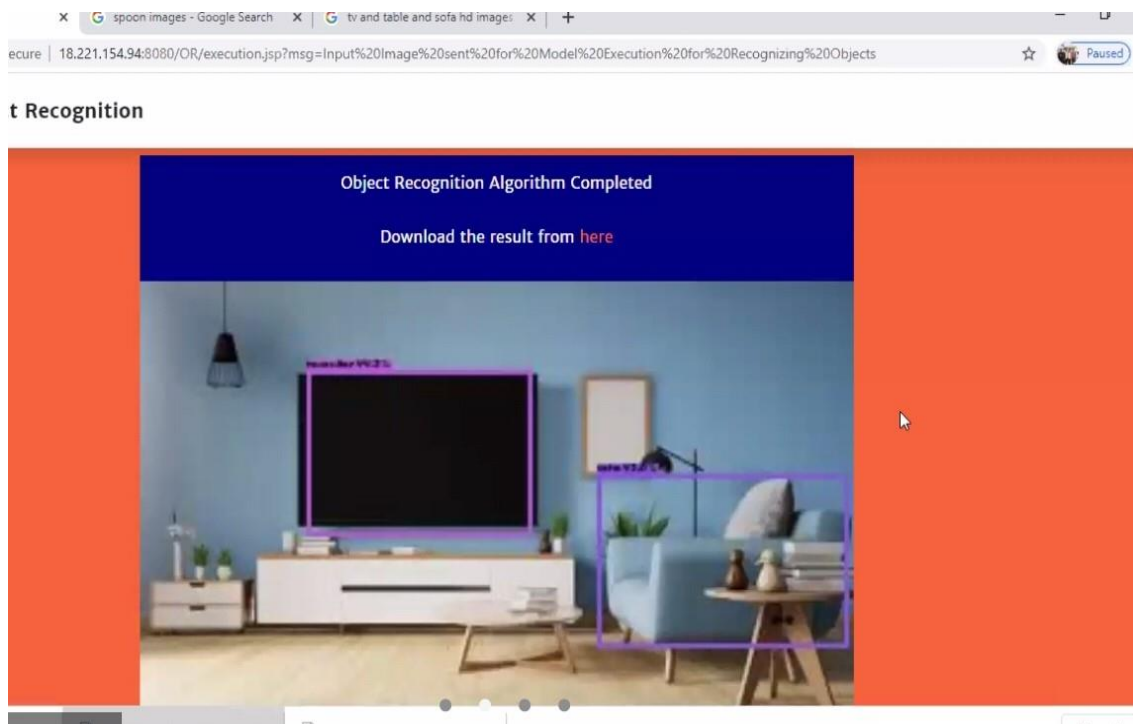


Figure 8.6: Object recognition

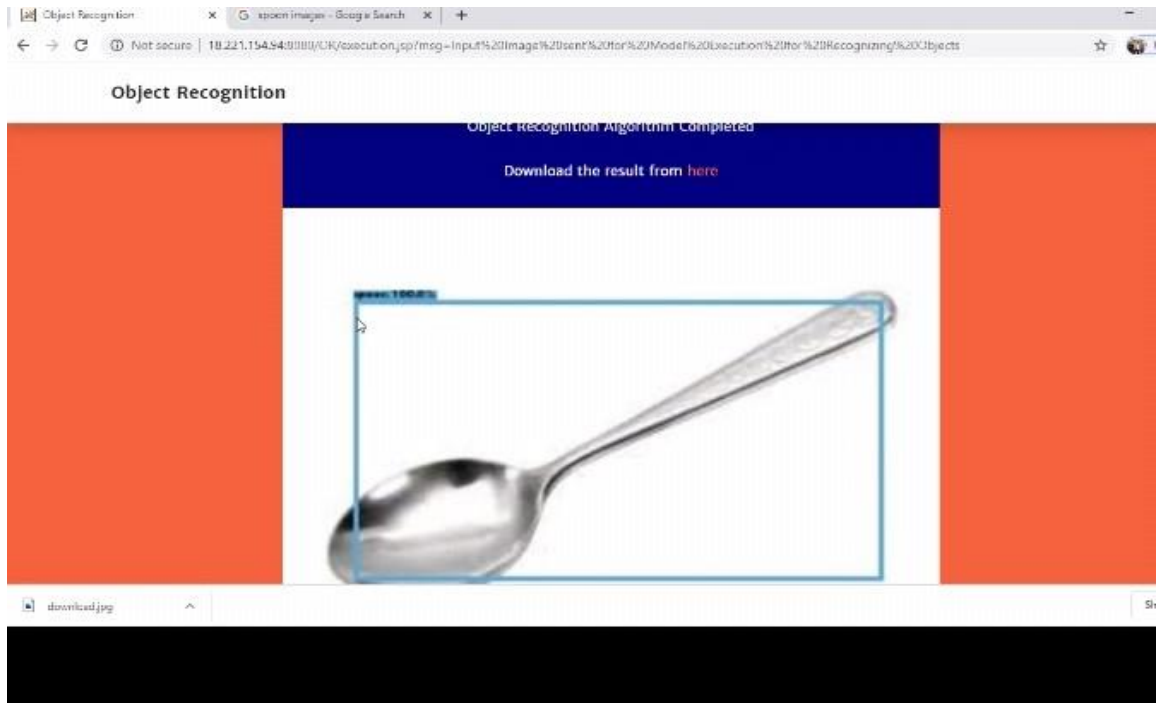


Figure 8.7 Download image

Chapter 9

CONCLUSION AND FUTURE WORK

9.1 Conclusion

By introducing the real-time object confirmation algorithm based on deep convolution neural network, the intelligent radar perimeter security system incorporates the high sensitivity of radar detection and the high accuracy of object confirmation. Due to breaking the limitation of single technology, the system's false alarm rate is reduced while the system's accuracy is improved. Using the object confirmation algorithm, the intelligent radar perimeter security system significantly reduces the dependence on the weather environment. It can still achieve effective warning under adverse weather conditions

9.2 Future work

In future, we aim to extend our solution to videos so that numerous objects can be recognized in a video stream as well

9.3 Applications

Decoding Facial Recognition

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

- Identifying every face in the picture
- Focusing on each face despite external factors, such as light, angle, pose, etc.
- Identifying unique features
- Comparing all the collected data with already existing data in the database to match a face with a name.
- A similar process is followed for scene labeling as well.

Analyzing Documents

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

Historic and Environmental Collections

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

Understanding Climate

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

Grey Areas

Introduction of the grey area into CNNs is posed to provide a much more realistic picture of the real world. Currently, CNNs largely function exactly like a machine, seeing a true and false value for every question. However, as humans, we understand that the real world plays out in a thousand shades of grey. Allowing the machine to understand and process fuzzier logic will help it understand the grey area us humans live in and strive to work against. This will help CNNs get a more holistic view of what human sees.

Other Interesting Fields

CNNs are poised to be the future with their introduction into driverless cars, robots that can mimic human behavior, aides to human genome mapping projects, predicting earthquakes and natural disasters, and maybe even self-diagnoses of medical problems. So, you wouldn't even have to drive down to a clinic or schedule an appointment with a doctor to ensure your sneezing attack or high fever is just the simple flu and not symptoms of some rare disease. One problem that researchers are working on with CNNs is brain cancer detection. The earlier detection of brain cancer can prove to be a big step in saving more lives affected by this illness.

BIBLIOGRAPHY

- [1] R. Girshick, J. Donahue, T. Darrell and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR 14), IEEE Press, Jun. 2014, pp. 580-587, doi: 10.1109/CVPR.2014.81.
- [2] K. He, X. Zhang, S Ren and J. Sun, “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, Sept. 2015, pp. 1904-1916, doi: 10.1109/TPAMI.2015.2389824.
- [3] R. Girshick, “Fast R-CNN,” Proc. IEEE Conf. Computer Vision (ICCV 15), IEEE Press, Dec. 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169.
- [4] S Ren, K. He, R. Girshick and J. Sun, “Faster R-CNN: Towards RealTime Object Detection with Region Proposal Networks,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, Jun. 2017, pp. 1137-1149, doi: 10.1109/TPAMI.2016.2577031.
- [5] J. Dai, Y. Li, K. He and J. Sun, “R-FCN: Object Detection via Region-based Fully Convolutional Networks,” arXiv:1605.06409, Jun. 2016, unpublished.
- [6] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR 16), IEEE Press, Jun. 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.
- [7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed et al., “SSD: Single Shot MultiBox Detector,” Proc. 14th European Conference. European Conference on Computer Vision (ECCV 2016), Springer Press, Sept. 2016, pp. 21-37, doi: 10.1007/978-3-319-46448-0_2.
- [8] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR 17), IEEE Press, Jul. 2017, pp. 6517-6525, doi: 10.1109/CVPR.2017.690.
- [9] M. Lin, Q. Chen and S. Yan, “Network In Network,” Proc. International Conference on Learning Representations(ILCR 2014), Apr. 2014.