Project Report

On

Shopping Assistance for Visually Impaired People

Jaypee Institute of Information Technology, Noida

(2019-2020)

Submitted By:

(Group 4)

Gargi Gupta M. Tech CSE

Stuti Gupta M. Tech DA

Kumar Satyajeet M. Tech CSE

Submitted To:

Dr. Monali Manavi

Abstract

Recognition of grocery products in store shelves poses peculiar challenges. Firstly, the task mandates the recognition of an extremely high number of different items, in the order of several thousand for medium-small shops, with many of them featuring small inter and intra class variability. Then, available product databases usually include just one or a few studio-quality images per product (referred to herein as reference images), whilst at test time recognition is performed on pictures displaying a portion of a shelf containing several products and taken in the store by cheap cameras (referred to as query images). Moreover, as the items on sale in a store as well as their appearance change frequently over time, a practical recognition system should handle seamlessly new products/packages. Inspired by recent advances in object detection and image retrieval, we propose to leverage on state-of-the-art image classification based on deep learning. In the study conducted our objective is to use this technology to help blind people around the world, which will help them to shop easily to identity the grocery product on the shelves. For image classification various CNN based image classification algorithms have been applied in this study. Our system is computationally expensive at training time but can perform recognition rapidly and accurately at test time.

Objective

The main objective of this project, is to detect real-time product image gathered from mobile video in grocery stores. The intent is to enable a blind user to scan a scene with a camera and have detected items on his shopping list.

The problem of using pictures of objects captured under ideal imaging conditions (here referred to as in vitro) to recognize objects in natural environments (in situ) is an emerging area of interest in computer vision and pattern recognition. In this project we will learn that how we can classify image belonging to a particular product category.

Division of Work

Module 1	Topic selection and Feasibility Study
Module 2	Background study
Module 3	Dataset Selection
Module 4	Code Implementation
Module 5	Result Analysis

Background Study

Today, images and video are everywhere. Online photo sharing sites and social networks have them in the billions. The field of vision research has been dominated by machine learning and statistics. Images and videos are used to detect, classify, and track objects or events in order to" understand" a real-world scene. Programming a computer and designing algorithms for understanding what is in these images is the field of computer vision. Computer vision powers applications like image search, robot navigation, medical image analysis, photo management and many more. From a computer vision point of view, the image is a scene consisting of objects of interest and a background represented by everything else in the image. The relations and interactions among these objects are the key factors for scene understanding. Object recognition identifies the object class in the training database, to which the object belongs to. Object detection determines the presence of an object and/or its scope, and locations in the image. It can be treated as a two-class object recognition, where one class represents the object class and another class represents non-object class. It can be further divided into soft detection, which only detects the presence of an object, and hard detection, which detects both the presence and location of the object.

The image classification is achieved by differentiating the image into the prescribed category based on the content of the vision. Image classification is the task of classifying a given image into one of the pre-defined categories. Traditional pipeline for image classification involves two modules: viz. feature extraction and classification. This is useful when there is a single class in the image and is distinctly visible in the image. Canonical example is classification of cat vs dog.

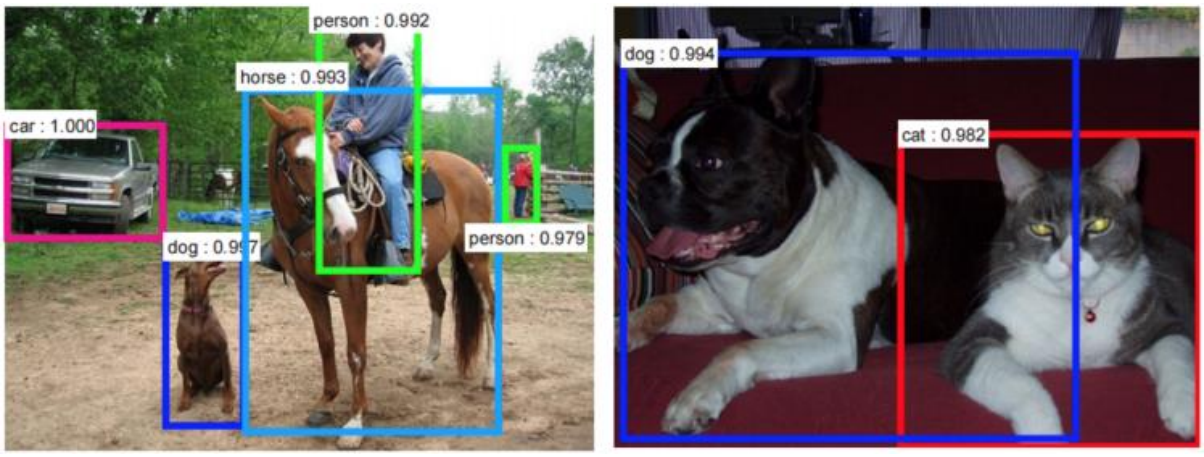


Figure 1 shows an example of ConvNets being used for recognizing everyday objects, humans and animals

In this project, we will use image classification-based algorithms to classify objects that belong to a particular product category from the Gro-zip dataset. CNN and its various other architectures like LeNet and AlexNet have been implemented in this project. So it is important to understand what CNN is.

A Convolutional Neural Network **(**CNN**,** or ConvNet**)** are a special kind of multi-layer neural networks, designed to recognize visual patterns directly from pixel images with

minimal pre-processing. ConvNets have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self-driving cars. Various network architectures are: LeNet, AlexNet, VGG, Inception, ResNet, ResNeXt, DenseNet. In this project we have implemented the classical single layer CNN and LeNet based CNN architecture.

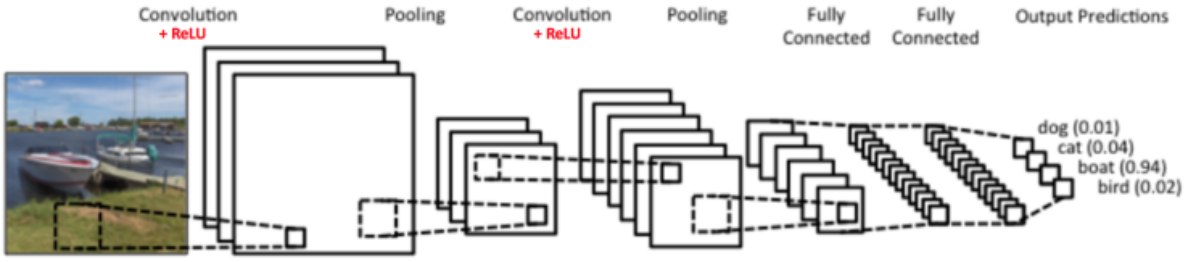


Figure 2 Convolution Neural Network

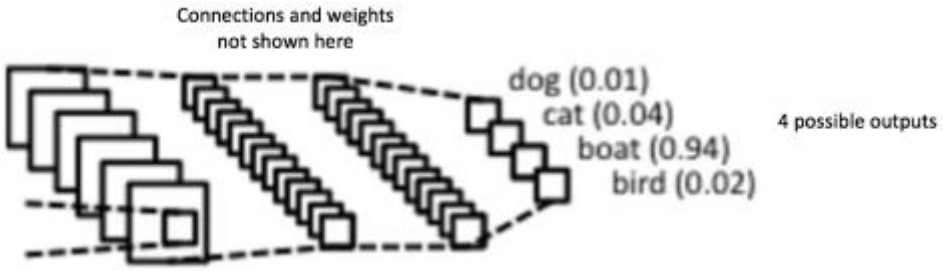
Three basic components to define a basic convolutional network are as follows:

The convolutional layer, ConvNets derive their name from the "convolution" operator. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. every image can be considered as a matrix of pixel values., element wise multiplication is computed (between the two matrices i.e. image and filter matrix) the multiplication outputs are added to get the final integer which forms a single element of the output matrix. Matrix. A '**filter** 'or 'kernel' or 'feature detector' and the matrix formed by sliding the filter over the image and computing the dot product is called the 'Convolved Feature' or 'Activation Map' or the '**Feature MapI**t is important to note that filters acts as feature detectors from the original input image an additional operation called ReLU has been used after every Convolution operation. ReLU stands for Rectified Linear Unit and is a non-linear operation. ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet. The Pooling layer (optional)- Spatial Pooling (also called subsampling or down sampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc. The function of Pooling is to progressively reduce the spatial size of the input representation. In particular, pooling

- makes the input representations (feature dimension) smaller and more manageable
- reduces the number of parameters and computations in the network, therefore, controlling overfitting
- makes the network invariant to small transformations, distortions and translations in the input image (a small distortion in input will not change the output of Pooling – since we take the maximum / average value in a local neighborhood).

- helps us arrive at an almost scale invariant representation of our image (the exact term is “equivariant”). This is very powerful since we can detect objects in an image no matter where they are located.

The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset. For example, the image classification task we set out to perform has four possible outputs as shown in figure below.



The sum of output probabilities from the Fully Connected Layer is 1. This is ensured by using the Softmax as the activation function in the output layer of the Fully Connected Layer. The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one. Now, the Convolution + Pooling layers’ act as Feature Extractors from the input image while Fully Connected layer acts as a classifier.

The overall training process of the Convolution Network may be summarized as below:

- **Step1:** We initialize all filters and parameters / weights with random values

- **Step2:** The network takes a training image as input, goes through the forward propagation step (convolution, ReLU and pooling operations along with forward propagation in the Fully Connected layer) and finds the output probabilities for each class.
    - Let’s say the output probabilities for the boat image above are [0.2, 0.4, 0.1, 0.3]
    - Since weights are randomly assigned for the first training example, output probabilities are also random.

- **Step3:** Calculate the total error at the output layer (summation over all 4 classes)
    - **Total Error = $\sum$ ½ (target probability – output probability) ^{2}**

- **Step4:** Use Backpropagation to calculate the *gradients* of the error with respect to all weights in the network and use *gradient descent* to update all filter values / weights and parameter values to minimize the output error.
    - The weights are adjusted in proportion to their contribution to the total error.
    - When the same image is input again, output probabilities might now be [0.1, 0.1, 0.7, 0.1], which is closer to the target vector [0, 0, 1, 0].
    - This means that the network has *learnt* to classify this particular image correctly by adjusting its weights / filters such that the output error is reduced.
    - Parameters like number of filters, filter sizes, architecture of the network etc. have all been fixed before Step 1 and do not change during training process – only the values of the filter matrix and connection weights get updated.

- **Step5:** Repeat steps 2-4 with all images in the training set.

The above steps *train* the ConvNet – this essentially means that all the weights and parameters of the ConvNet have now been optimized to correctly classify images from the training set.

When a new (unseen) image is input into the ConvNet, the network would go through the forward propagation step and output a probability for each class (for a new image, the output probabilities are calculated using the weights which have been optimized to correctly classify all the previous training examples). If our training set is large enough, the network will (hopefully) generalize well to new images and classify them into correct categories.

There are several architectures in the field of Convolutional Networks that have a name. The most common are:

- **LeNet**. The first successful applications of Convolutional Networks were developed by Yann LeCun in 1990's. Of these, the best known is the LeNet architecture that was used to read zip codes, digits, etc.

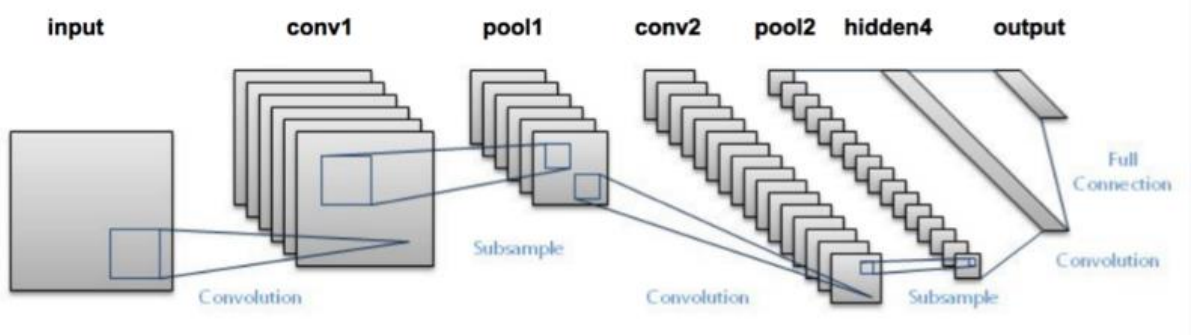


Figure 3 LeNet architecture

There are various versions of LeNet available. i.e. LeNet-5. LeNet-1, LeNet-4 and Boosted LeNet-4 are usually ignored.

- **AlexNet**. The first work that popularized Convolutional Networks in Computer Vision was the AlexNet developed by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton. The Network had a very similar architecture to LeNet, but was deeper, bigger, and featured Convolutional Layers stacked on top of each other (previously it was common to only have a single CONV layer always immediately followed by a POOL layer).

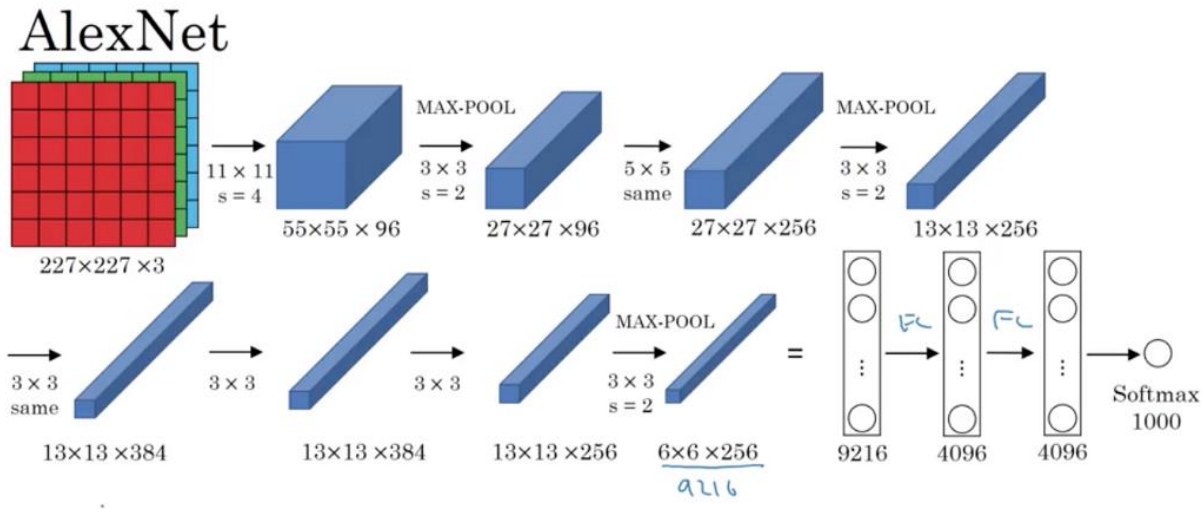


Figure 4 AlexNet architecture

..

- **VGGNet**. The runner-up in ILSVRC 2014 was the network from Karen Simonyan and Andrew Zisserman that became known as the VGGNet. Its main contribution was in showing that the depth of the network is a critical component for good performance. Their final best network contains 16 CONV/FC layers and, appealingly, features an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end. A downside of the VGGNet is that it is more expensive to evaluate and uses a lot more memory and parameters (140M). Most of these parameters are in the first fully connected layer, and it was since found that these FC layers can be removed with no performance downgrade, significantly reducing the number of necessary parameters.
- **ResNet** (developed by Kaiming) called as Residual Network. It features special *skip connections* and a heavy use of batch normalization.

Literature Review

In [1] Qiao S et al studied the problem of object suggestions in supermarket images and other natural images. They believed that the estimation of the object scale in the image helps to generate object suggestions, especially for supermarket images where the scale is usually in a small range. Therefore, it was recommend to estimate the object scale of the image before generating the object proposal. The proposed method of predicting object scale was called ScaleNet. To validate the effectiveness of ScaleNet, they built three supermarket datasets, two of which were real datasets for testing and the other were synthetic datasets for training. In [2] publicly available code and model of the OverFeat network to report a series of experiments for different recognition tasks that were trained to perform object

classification on ILSVRC 13. Features were extracted from the OverFeat network as general-purpose image representations to handle various object recognition tasks for object image classification, scene recognition, fine-grained recognition, attribute detection, and image retrieval applied to various data sets. These tasks and data sets were then moved away from the original tasks and data that OverFeat network training solved. It was reported that it gave consistently superior results compared to the most advanced systems that are highly tuned in all visual classification tasks in various data sets. Authors in [2] used the off-the-shelf CNN representation, OverFeat, to use a simple classifier to handle different recognition tasks. The learned CNN model was originally optimized for object classification tasks in the ILSVRC 2013 data set.

In [4] Tonioni A et al, studied recognition of a large number of different projects, for small and medium-sized stores, of the order of thousands, many of which have small internal and intra-class variability. The available product database then typically includes only one or several studio-quality images (referred to herein as reference images) for each product, while at the time of testing, a portion of the shelf containing several products is displayed and photographed in the store. Perform recognition on the picture. Through a cheap camera (called a query image). In addition, as the items for sale in the store and their appearance change frequently over time, the actual identification system should handle the new product/package seamlessly. Inspired by the latest advances in object detection and image retrieval, we recommend using state-of-the-art object detectors based on deep learning to obtain initial product-independent project detection. They then pursued product identification by searching for similarities between global descriptors computed on the reference and cropped query images. To maximize performance, they learned ad-hoc global descriptors by CNN trained on the reference image based on image embedding loss. Their system was computationally expensive at the time of training, but could perform identification quickly and accurately during testing.

Target User for the Project

Vision loss is one of ultimate obstacle in the lives of blind that prevent them to perform tasks on their own and self-reliantly. According to the internet sources, the estimated number of people visually impaired in the world is 285 million, out of which 39 million are blind and 246 million having low vision They are bound to depend upon on their family for *assistance*, who often may not afford quality time due to busy routine. So, for this project our goal is to help visually impaired people in their day to day task of shopping.

Tools and Platform Used

Python is general purpose and interpreted, high- level programming language and dynamically typed and collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. It was developed by Guido Van Rossum. Popular IDE's for python are Spyder, IPython Notebook. Some popular Python libraries are pandas, SciPy, NumPy, sckikit-learn (for machine learning), matplotlib. In this project python version 3.6 has been used. In this project following libraries have been used:

- **Pandas:** pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. It developed by Wes McKinney in 2008

- **SciPy:** SciPy is a source Python library used for scientific and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering. SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries

- **NumPy:** Numerical Python extensions(NumPy) is a library for the Python programming adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers.

- **Scikit-learn**: Scikit-learn is a free software machine learning library for the Python programming language features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries. Scikit-learn are largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

- **Matplotlib:** Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+,Matplotlib was originally written by John D. Hunter, Matplotlib is designed to be as usable as MATLAB, with the ability to use Python, and the advantage of being free and open-source.
- **TensorFlow:** It is a free open source software library for data flow and distinguishable programming across a range of tasks. It is a symbolic math library that is also used in machine learning applications such as neural networks

DATASET

Dataset used in the project: GroZi-120

The GroZi-120 is a multimedia database of 120 grocery products. The objects belonging to it vary in colour, size, opacity, shape and rigidity. Every product has two different representations in the database: one captured in vitro and another in situ.

Figure 5 snapshot of the dataset website

The in vitro images are isolated and captured under ideal imaging conditions (e.g. stock photography studio or a lab). On the other hand, in situ representations come from natural environments (real world). The dataset providers shot 29 videos on the same day. Such videos include every product present in the in vitro part of the dataset. These images were selected every 5 frames and were stored together with their location in the video. These images present variations in scale, illumination, reflectance, colour, pose and rotation, while the video frames provide a cluttered background.

For situ-based dataset we have 120 folders with situ images along with videos.

For vitro-based dataset we have 120 folders with vitro images.

This dataset is provided by The University California, San Diego (Computer Vision Laboratory).

To source of the dataset is mentioned in the link stated below:

http://grozi.calit2.net/grozi.html

Project Module Diagram

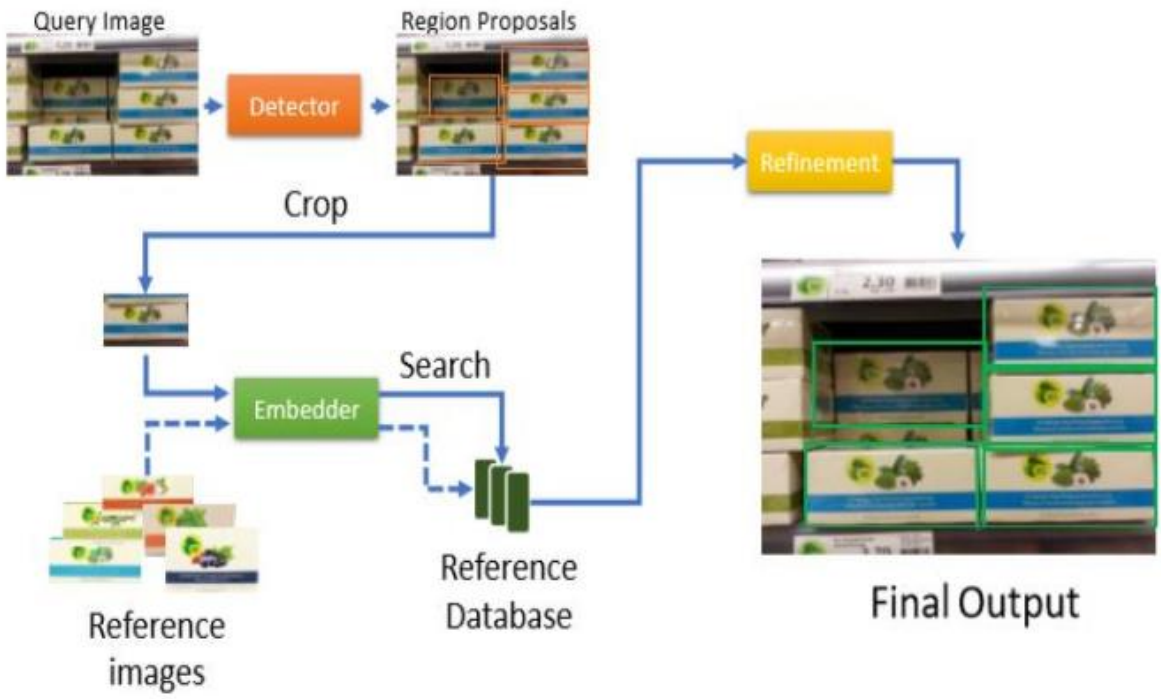


Figure 6 Workflow diagram

The workflow diagram shown above shows the detailed workflow of the project.

The Gro-Zip dataset used in this project contains vitro and in situ images of the 120 grocery items. So according to the workflow firstly the object will be detected with bounding boxes and then using image classification algorithms we aim to classify the products belonging to a particular product category.

In this project classical CNN, LeNet and AlexNet is applied on the Gro-zip dataset for image classification purpose. After applying all the three algorithms we observe that AlexNet gave the highest image classification accuracy results.

Design Complexity

Level 0:

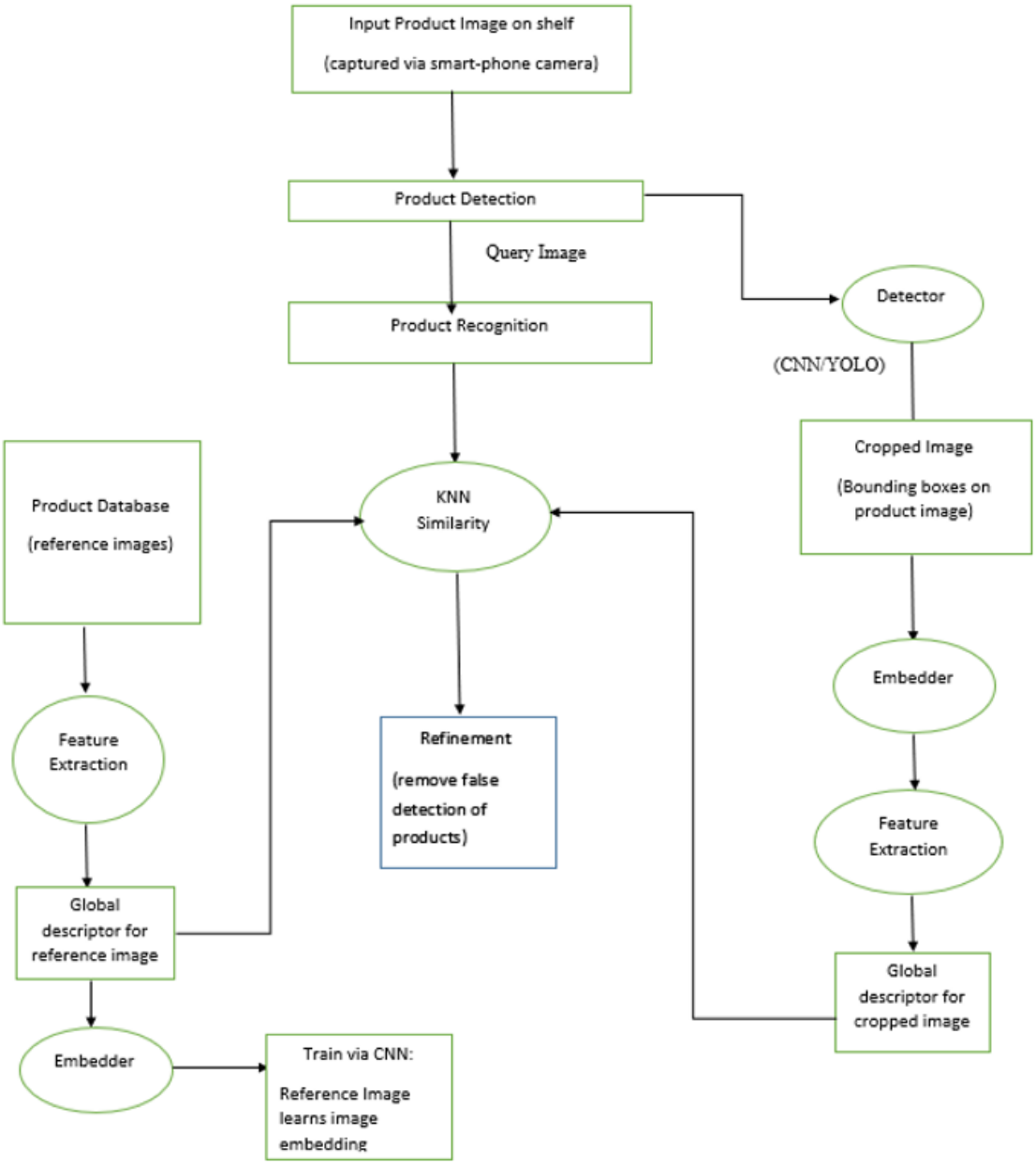


Figure 7 Level 0 DFD diagram

Level 1: Detection

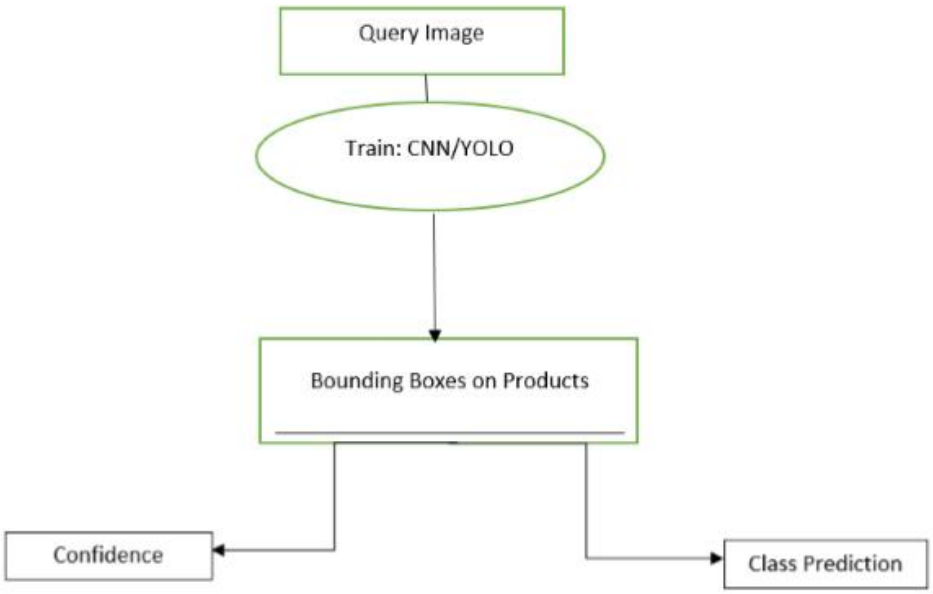


Figure 8 Level 1 DFD diagram

Level 2: Recognition

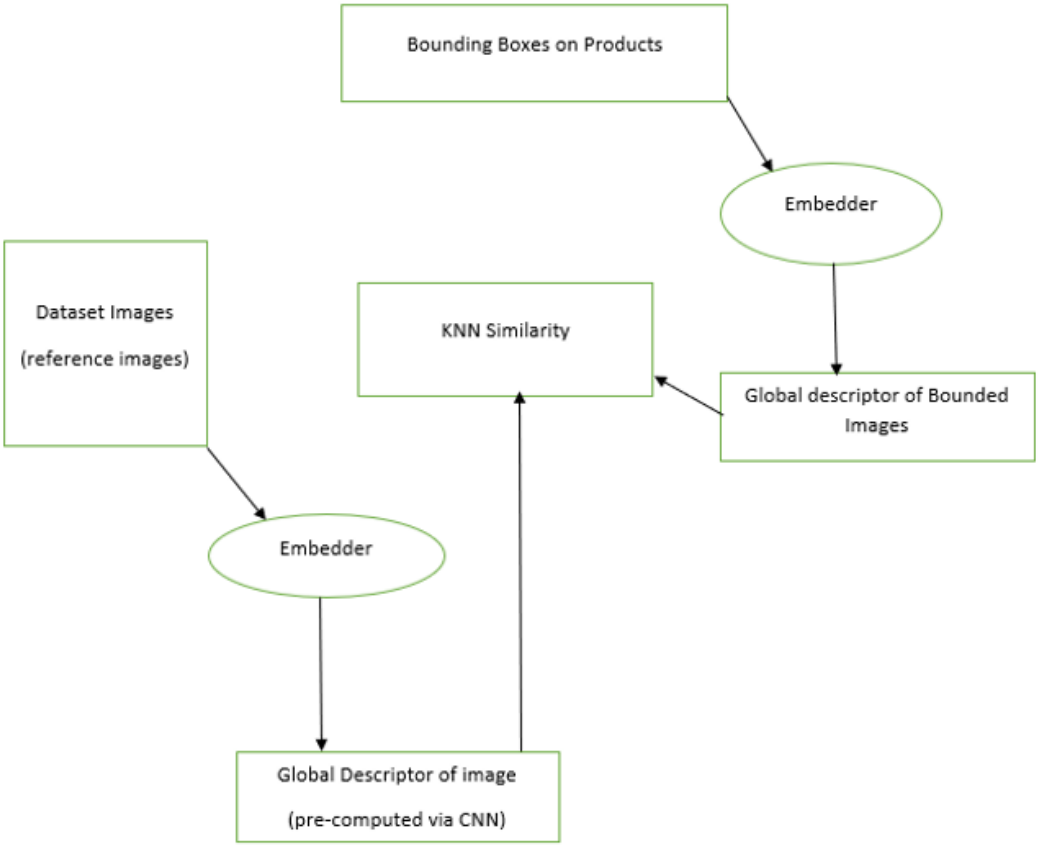


Figure 9 Level 3 DFD diagram

Use case diagram:

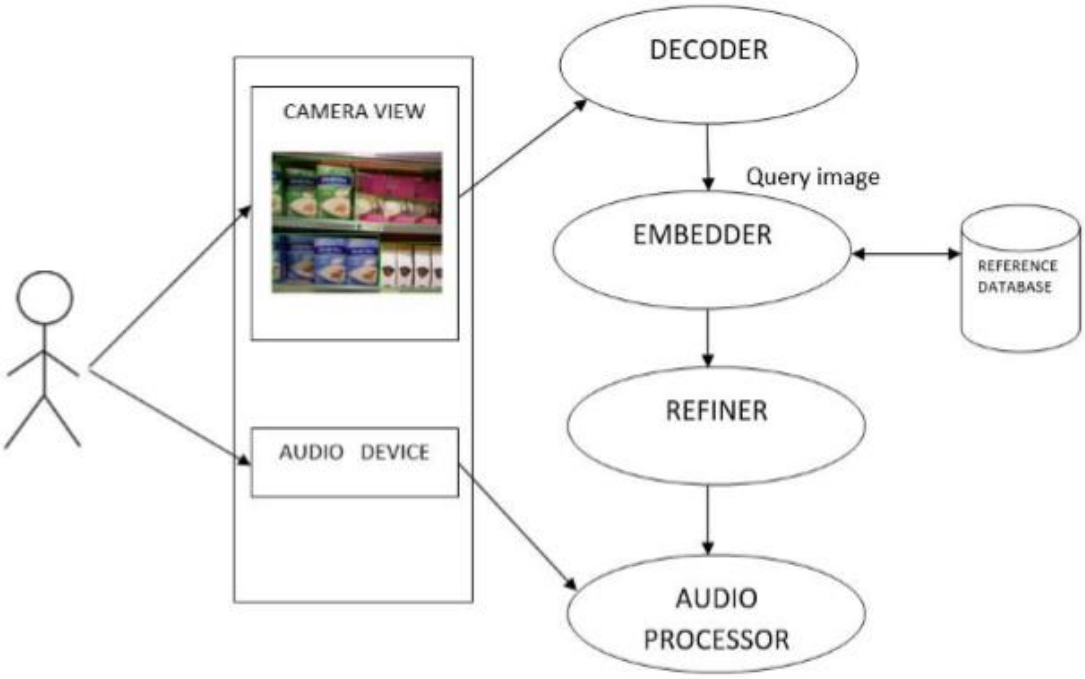


Figure 10 Use case diagram of the project

Project Limitations

The limitations of our project are:

- Our model for this project is trained only for 120 products i.e. the products belonging to only GroZi-120 dataset. Henceforth, we cannot classify any other products.
- Since the images of the product obtained from video camera were very low-quality product images therefore, it was very difficult to classify them properly.
- In this project we have implemented CNN, LeNet and Alexnet (CNN based architecture) this resulted in very heavy computation and high memory consumption of the system.
- Hardware like camera and sensors are needed to bring whole system into real life model.
- Proper simulation tools are required to make it work in real world.

Future Work

In future, this project can be explored further to give better image classification accuracy results. More advanced CNN architectures like VGG, ResNet, DenseNet can be implemented by a researcher. Since, our project is limited to classification of 120 grocery product only, these algorithms can be implemented on a dataset containing more than 120 grocery products.

Project Code

A. CNN

```
from sklearn.model_selection import train_test_split
from keras.preprocessing import image
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
import os.path
import matplotlib.pyplot as plt

labels = pd.read_csv("D:\\index2\\UPC_index.txt",sep="\t")
print(labels.shape)
print(labels.head(3))

tag=[]
train_image = []
for i in (range(1,121)):
    j=1
    while(j!=0):
        file_path='D:\\inSitu\\'+str(i)+'\\video\\video'+str(j)+'.png'
        if os.path.isfile(file_path):
            img = image.load_img(file_path, target_size=(28,28,1), grayscale=False)
            img = image.img_to_array(img)
            img = img/255
            train_image.append(img)
            tag.append(i-1)
            j=j+1
```

```
        else:
            j=0

print("tag length:",len(tag))
print("Total Images: ",len(train_image))
list_of_tuples=list(zip(train_image,tag))
df = pd.DataFrame(list_of_tuples, columns = ['img', 'label'])
print(df.shape)

X = np.array(train_image)
y=pd.get_dummies(df['label']).values

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.2)

print("Train Images:",X_train.shape[0])
print("Test Images:",X_test.shape[0])

model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),activation='relu',input_shape=(28,28,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(120, activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])

history=model.fit(X_train, y_train, epochs=5)

scores = model.evaluate(X_test, y_test, verbose=0)
print("Test Accuracy:"+"%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
y_pred=model.predict_classes(X_test)
for i in (range(0,5)):
    j=y_pred[i]
    classname=labels['product_name'][j]
    print("Predicted class for image {} are: {}".format(i,classname))



import cv2
tag1=[]
test_image = []
nemos_friends = []
for i in (range(1,11)):
    plt.subplot(1, 2, 1)
    file_path='D:\\New Folder\\p'+str(i)+'.png'
    img = image.load_img(file_path, target_size=(28,28,1), grayscale=False)
    img1 = image.img_to_array(img)
    img1 = img1/255
    test_image.append(img1)
    friend = cv2.cvtColor(cv2.imread(file_path), cv2.COLOR_BGR2RGB)
    nemos_friends.append(friend)
    plt.subplot(1, 2, 1)
    plt.imshow(nemos_friends[i-1])
    plt.show()
test = np.array(test_image)
y_pred=model.predict_classes(test)
for i in (range(0,10)):
    j=y_pred[i]
    print("Predicted class for image {} are: {}".format(i,labels['product_name'][j]))
```

2. LeNet

```
from sklearn.model_selection import train_test_split
from keras.preprocessing import image
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
import os.path
import matplotlib.pyplot as plt

labels = pd.read_csv("D:\\index2\\UPC_index.txt",sep="\t")
print(labels.shape)
print(labels.head(3))

tag=[]
train_image = []
for i in (range(1,121)):
    j=1
    while(j!=0):
        file_path='D:\\inSitu\\'+str(i)+'\\video\\video'+str(j)+'.png'
        if os.path.isfile(file_path):
            img = image.load_img(file_path, target_size=(28,28,1), grayscale=False)
            img = image.img_to_array(img)
            img = img/255
            train_image.append(img)
            tag.append(i-1)
            j=j+1
```

```
        else:
            j=0

print("tag length:",len(tag))
print("Total Images: ",len(train_image))
list_of_tuples=list(zip(train_image,tag))
df = pd.DataFrame(list_of_tuples, columns = ['img', 'label'])
print(df.shape)

X = np.array(train_image)
y=pd.get_dummies(df['label']).values

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.2)

print("Train Images:",X_train.shape[0])
print("Test Images:",X_test.shape[0])

model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),activation='relu',input_shape=(28,28,3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(500, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(120, activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])

history=model.fit(X_train, y_train, epochs=5)
```

```
scores = model.evaluate(X_test, y_test, verbose=0)
print("Test Accuracy:"+"%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))

y_pred=model.predict_classes(X_test)
for i in (range(0,5)):
    j=y_pred[i]
    classname=labels['product_name'][j]
    print("Predicted class for image {} are: {}".format(i,classname))



import cv2
tag1=[]
test_image = []
nemos_friends = []
for i in (range(1,11)):
    plt.subplot(1, 2, 1)
    file_path='D:\\New Folder\\p'+str(i)+'.png'
    img = image.load_img(file_path, target_size=(28,28,1), grayscale=False)
    img1 = image.img_to_array(img)
    img1 = img1/255
    test_image.append(img1)
    friend = cv2.cvtColor(cv2.imread(file_path), cv2.COLOR_BGR2RGB)
    nemos_friends.append(friend)
    plt.subplot(1, 2, 1)
    plt.imshow(nemos_friends[i-1])
    plt.show()
test = np.array(test_image)
y_pred=model.predict_classes(test)
```

```
for i in (range(0,10)):
    j=y_pred[i]
    print("Predicted class for image {} are: {}".format(i,labels['product_name'][j]))
```

Results and Findings

CNN

```
tag length: 11139
Total Images:  11139
(11139, 2)
Train Images: 8911
Test Images: 2228
Epoch 1/5
8911/8911 [==============================] - 4s 486us/step - loss: 1.5441 - acc: 0.6937
Epoch 2/5
8911/8911 [==============================] - 3s 373us/step - loss: 0.1530 - acc: 0.9697
Epoch 3/5
8911/8911 [==============================] - 3s 354us/step - loss: 0.0681 - acc: 0.9874
Epoch 4/5
8911/8911 [==============================] - 4s 394us/step - loss: 0.0396 - acc: 0.9936
Epoch 5/5
8911/8911 [==============================] - 4s 426us/step - loss: 0.0253 - acc: 0.9948
Test Accuracy:acc: 94.85%
Predicted class for image 0 are: Haribo Gold-Bears Gummi Candy
Predicted class for image 1 are: big red gum
Predicted class for image 2 are: Campbell's Tomato Soup - Microwavable bowl
Predicted class for image 3 are: Jif Creamy Peanut Butter
```

Figure11 Shows the accuracy results for CNN

LeNet

```
tag length: 11139
Total Images:  11139
(11139, 2)
Train Images: 8911
Test Images: 2228
Epoch 1/5
8911/8911 [==============================] - 5s 526us/step - loss: 2.3280 - acc: 0.4788
Epoch 2/5
8911/8911 [==============================] - 4s 423us/step - loss: 0.4529 - acc: 0.8767
Epoch 3/5
8911/8911 [==============================] - 4s 425us/step - loss: 0.2384 - acc: 0.9351
Epoch 4/5
8911/8911 [==============================] - 4s 420us/step - loss: 0.1529 - acc: 0.9578
Epoch 5/5
8911/8911 [==============================] - 4s 423us/step - loss: 0.0974 - acc: 0.9736
Test Accuracy:acc: 97.26%
Predicted class for image 0 are: Haribo Gold-Bears Gummi Candy
Predicted class for image 1 are: big red gum
Predicted class for image 2 are: Campbell's Tomato Soup - Microwavable bowl
Predicted class for image 3 are: Jif Creamy Peanut Butter
```

Figure 12 Shows the accuracy results for LeNet

AlexNet

```
tag length: 11139
Total Images:  11139
(11139, 2)
Train Images: 8911
Test Images: 2228
Epoch 1/5
8911/8911 [==============================] - 5s 530us/step - loss: 2.4127 - acc: 0.4543
Epoch 2/5
8911/8911 [==============================] - 4s 424us/step - loss: 0.4377 - acc: 0.8818
Epoch 3/5
8911/8911 [==============================] - 4s 444us/step - loss: 0.1978 - acc: 0.9469
Epoch 4/5
8911/8911 [==============================] - 4s 437us/step - loss: 0.1224 - acc: 0.9663
Epoch 5/5
8911/8911 [==============================] - 4s 442us/step - loss: 0.0758 - acc: 0.9788
Test Accuracy:acc: 98.26%
Predicted class for image 0 are: Haribo Gold-Bears Gummi Candy
Predicted class for image 1 are: big red gum
Predicted class for image 2 are: Campbell's Tomato Soup - Microwavable bowl
Predicted class for image 3 are: Jif Creamy Peanut Butter
```

Figure 13 Shows the accuracy results for AlexNet

Observations

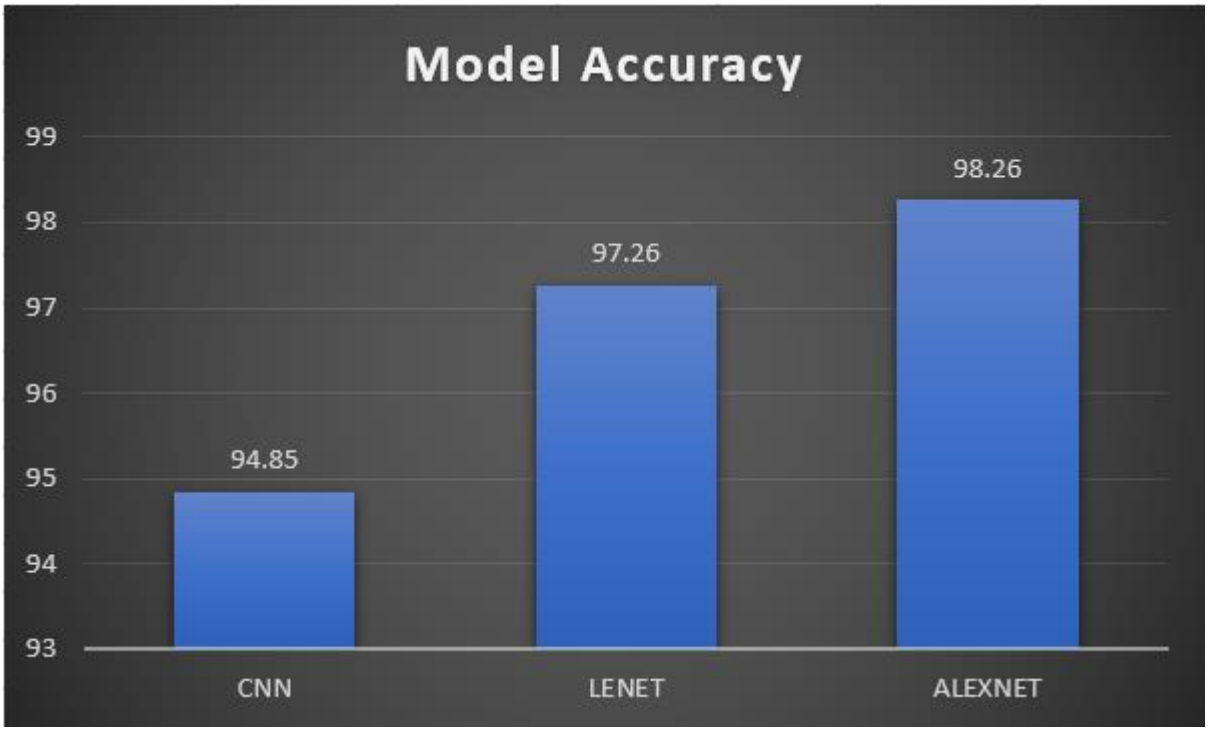


Figure 14 accuracy results bar plot

From figure 4 we can clearly observe that AlexNet gave the highest accuracy results i.e. 98.26 % as compared to CNN and LeNet.

References:

[1] Qiao S, Shen W, Qiu W, Liu C, Yuille A. Scalenet: Guiding object proposal generation in supermarkets and beyond. InProceedings of the IEEE International Conference on Computer Vision 2017 (pp. 1791-1800).

[2] Sharif Razavian A, Azizpour H, Sullivan J, Carlsson S. CNN features off-the-shelf: an astounding baseline for recognition. InProceedings of the IEEE conference on computer vision and pattern recognition workshops 2014 (pp. 806-813).

[3] Li J, Wang X, Su H. Supermarket commodity identification using convolutional neural networks. In2016 2nd International Conference on Cloud Computing and Internet of Things (CCIOT) 2016 Oct 22 (pp. 115-119). IEEE.

[4] Tonioni A, Serro E, Di Stefano L. A deep learning pipeline for product recognition in store shelves. arXiv preprint arXiv:1810.01733. 2018 Oct 3.

[5] Kumar R, Meher S. A novel method for visually impaired using object recognition. In2015 International Conference on Communications and Signal Processing (ICCSP) 2015 Apr 2 (pp. 0772-0776). IEEE.

Contents