# Indian Currency Recognition System

*a dissertation submitted in accordance*
*with the requirements of the degree of*

**Master of Technology**

*in*

## Data Analytics

*by*

**Chinmay Sathe**
**19MT0119**

Under the guidance of

**Dr. Subhashis Chatterjee**

Associate Professor


MATHEMATICS AND COMPUTING

INDIAN INSTITUTE OF TECHNOLOGY
(INDIAN SCHOOL OF MINES), DHANBAD

May 2021

DEPARTMENT OF MATHEMATICS
AND COMPUTING,
INDIAN INSTITUTE OF
TECHNOLOGY (INDIAN SCHOOL OF
MINES), DHANBAD
DHANBAD - 826004, INDIA

---

# CERTIFICATE

This is to certify that **Mr. Chinmay Sathe** (Roll Number:19MT0119), a student of **M.tech.(Data Analytics), Department of MATHEMATICS AND COMPUTING**, Indian Institute of Technology (Indian School of Mines), Dhanbad has worked under my guidance and completed his Dissertation entitled **"Indian Currency Recognition System"** in partial fulfillment of the requirement for award of degree of M.Tech. in Data Analytics from Department of Mathematics And Computing, Indian Institute of Technology (Indian School of Mines), Dhanbad.

This work has not been submitted for any other degree, award, or distinction elsewhere to the best of my knowledge and belief. They are solely responsible for the technical data and information provided in this work.

Supervisor:
**(Dr. Subhashis Chatterjee)**
Associate Professor,
Department of Mathematics and Computing,
IIT (ISM), Dhanbad

Forwarded by:
**(Prof. Garib Nath Singh)**
Head of Department,
Department of Mathematics and Computing,
IIT (ISM), Dhanbad

# Declaration

The Dissertation titled "Indian Currency Recognition System" is a presentation of my original research work and is not copied or reproduced or imitated from any other person's published or unpublished work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions, as may be applicable. Every effort is made to give proper citation to the published/unpublished work of others, if it is referred to in the Dissertation.

To eliminate the scope of academic misconduct and plagiarism, I declare that I have read and understood the UGC (Promotion of Academic Integrity and Prevention of Plagiarism in Higher Education Institutions) Regulations, 2018. These Regulations have been notified in the Official Gazette of India on 31st July, 2018.

I confirm that this Dissertation has been checked with the online plagiarism detector tool Turnitin (http://www.turnitin.com) provided by IIT (ISM) Dhanbad and a copy of the summary report/report, showing Similarities in content and its potential source (if any), generated online through Turnitin is enclosed at the end of the Dissertation. I hereby declare that the Dissertation shows less than 10% similarity as per the report generated by Turnitin and meets the standards as per MHRD/UGC Regulations and rules of the Institute regarding plagiarism.

I further state that no part of the Dissertation and its data will be published without the consent of my guide. I also confirm that this Dissertation work, carried out underthe guidance of Dr. Subhashis Chatterjee, Associate Professor, Department of Mathematics and Computing, has not been previously submitted for assessment for the purpose of award of a Degree either at IIT (ISM) Dhanbad or elsewhere to the best of my knowledge and belief.

Chinmay Sathe
M.Tech (Data Analytics)
Mathematics and Computing
Admission No.: 19MT0119

Forwarded by:
Dr. Subhashis Chatterjee
Associate Professor
Dept. of Mathematics and Computing

# Acknowledgement

I would like to express my thanks to the people who have helped me the most throughout my project.

I am very grateful to my academic research advisor, Dr. Subhashis Chatterjee, Associate Professor, Department of Mathematics and Computing, IIT Dhanbad for the encouragement to explore various research fields throughout the college time. I am extremely grateful to him for supporting and guiding me throughout the project.

I would also like to thank Prof. Garib Nath Singh, HOD, Department of Mathematics and Computing, for providing me all the facilities to complete the project.

I wish to thank all my batch mates, they have been a great companion during the literature survey and also throughout the duration of my Mtech.

Finally, I wish to thank my parents, my friends, and faculty of Department of Mathematics and Computing, IIT ISM Dhanbad for their kind support and assistance.

**Chinmay Sathe**
Place : IIT Dhanbad
Date:

# Abstract

While recognizing any country's currency, there seems to be a huge gap between visually impaired and a normal visioned person. The idea is to reduce this gap for Indian currency by proposing a deep learning based currency recognition system.

There have been many researches on the similar problem statement, some used Single Shot MultiBox Detector (SSD), a CNN based model, some have used simple CNNs.This thesis brings the best of both worlds, with added advantage of latest hardware accelerated GPU. This thesis, also differs in the Data set as well as the manipulation of the data (Data Augmentation) part.

By collecting image data from various sources, a convolution neural network will be trained to extract features from the images. Later, an object detection algorithm mainly YOLO/Tiny-YOLO, will take earlier extracted features into consideration while detecting a denomination.

**Keywords**: Indian Currency Recognition, Convolution Neural Network, CNN, You Only Look Once, YOLO, tiny-Yolo,Data Augmentation, Deep Neural Network.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

*This chapter introduces us to whys and hows of the thesis. It consists of three parts. First part explains the motivation and inspiration behind the project.Also, how the past researchers have done projects on the same topic. The second part mainly covers who will benefit from the project and what is the objective. The third and final part covers the content outline of the complete thesis in few lines.*

## 1.1 Motivation

Money is important to everyone, if someone speaks otherwise then either they are lying or they are rich. A thing of such importance needs to be evaluated correctly by a person. Imagine it takes you a lot of efforts to procure money, every single time, just because you simply are unable to recognize the denomination as quickly as others. That is exactly what visually impaired people go through DAILY.

While recognizing any country's currency, there seems to be a huge gap between visually impaired and a normal visioned person. The idea is to reduce this gap for Indian currency by proposing a deep learning based currency recognition system.There have been many researches on the similar problem statement, some used Single Shot MultiBox Detector (SSD), a CNN based model, some have used simple CNNs.This thesis brings the best of both worlds, with added advantage of latest hardware accelerated GPU. This thesis, also differs in the Data set as well as the manipulation of the data (Data Augmentation) part.

By collecting image data from various sources, a convolution neural network will be trained to extract features from the images. Later, an object detection algorithm mainly YOLO/Tiny-YOLO, will take earlier extracted features into consideration while detecting a denomination.

As Engineers and Data Scientists it is our sole duty to make people's life easier, hence the thesis.

## 1.2 Objective

The main objective of this thesis is to provide a well defined and highly accurate Indian Currency Recognition System, which will be specially beneficial for :

- Blind or Visually Impaired People
- Old People with common eye disorders
- Foreigners visiting India

## 1.3 Contents Outline

The thesis content outline is as follows:

- Chapter 2 : Covers background literature portion of the thesis. We will have basic understanding of how the project is going to work after this. It also contains detailed

explanations of Deep Neural Network. Then, we move towards Computer Vision which include CNN and YOLO models. Data Augmentation is covered then, and finally we add related work that has already been done in this field.

- Chapter 3 : Covers implementation portion of the thesis. It mainly covers the Data collection portion and the main content, i.e., the YOLO Model used.
- Chapter 4 :Covers experiments and analysis portion of the thesis. Includes analysis of CNN, YOLO, tiny-YOLO Models. It ends with the results of our thesis.
- Chapter 5 : Covers conclusion and future work portion of the thesis.

# Chapter 2

# Background Literature

*This capter mainly imparts some background knowledge of Indian Currency, their dimensions and color. Then we dive deep into Convolution Neural Network.The the our show stopper YOLO is explained. Finally data augmentation is understood.*

## 2.1 Indian Currency

**Indian Rupee** is the official currency of India. It is notable that India was one of the first few countries to have issues their own official coins. Though, in today word, coins are considered a burden because of their weight as well as their low denomination value. Also, they have a very significant difference in their shapes and sizes, thus, they are easily recognizable by even a blind person.

Here,we will mainly focus on currency denominations from Rs.10 up to Rs.2000, since they are the most used currencies nowadays. Since, demonetisation of old 500 and 1000 notes in 2016, there have been new issues of nearly every denomination. All the bank denominations we have used for the project are:

1. Rs.10 Note: **Dimensions:** 123 mm × 63 mm, **Color:** Brown, **Issued:** 2018


(a) Rs.10 Front (b) Rs.10 Back

Figure 2.1: Rs.10 Denomination

2. Rs.20 Note: **Dimensions:** 129 mm × 63 mm, **Color:** Yellow, **Issued:** 2019


(a) Rs.20 Front (b) Rs.20 Back

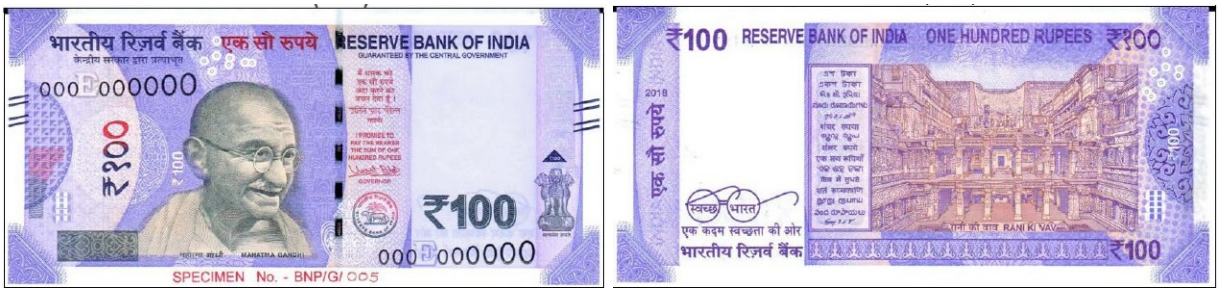
Figure 2.2: Rs.20 Denomination

3. Rs.50 Note: **Dimensions:** 135 mm × 66 mm, **Color:** Cyan, **Issued:** 2017


(a) Rs.50 Front (b) Rs.50 Back

Figure 2.3: Rs.50 Denomination

4. Rs.100 Note: **Dimensions:** 142 mm × 66 mm, **Color:** Lavender, **Issued:** 2018




(a) Rs.100 Front (b) Rs.100 Back

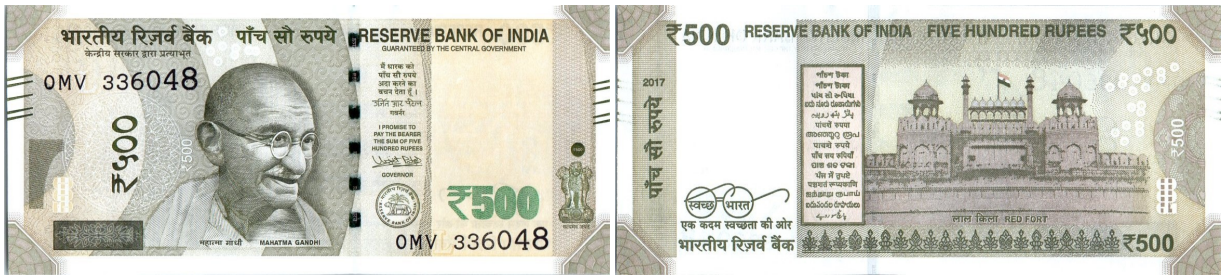
Figure 2.4: Rs.100 Denomination

5. Rs.200 Note: **Dimensions:** 146 mm × 66 mm, **Color:** Orange, **Issued:** 2017


(a) Rs.200 Front (b) Rs.200 Back

Figure 2.5: Rs.200 Denomination

6. Rs.500 Note: **Dimensions:** 150 mm × 66 mm, **Color:** Stone Grey, **Issued:** 2016




(a) Rs.500 Front (b) Rs.500 Back

Figure 2.6: Rs.500 Denomination

7. Rs.2000 Note: **Dimensions:** 166 mm × 66 mm, **Color:** Magenta, **Issued:** 2016


(a) Rs.2000 Front (b) Rs.2000 Back

Figure 2.7: Rs.2000 Denomination

## 2.2 Computer Vision

Computer vision is a field of AI that trains computers to interpret and understand the visual world. Using pictures from cameras and videos and deep learning models, machines can accurately identify and classify objects, basically act as close as human vision. CNN are used for Feature extraction while YOLO are later used for object detection.

### 2.2.1 Convolutional Neural Network

Similar to simple perceptron, Convolutional Neural Networks also have biological origins. Hubel and Wiesel studied the structure of eye of a mammal, hence winning Nobel in 1981. They revealed that neurons in eye had small receptive field as shown in Fig.2.8

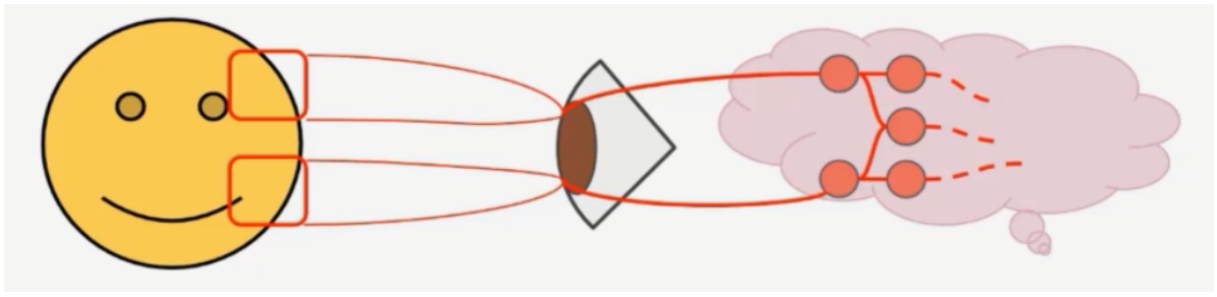


Figure 2.8: Receptor Field in Human Vision

This idea later inspired CNN models, implemented in 1998 paper by Yann LeCun et al. The LeNet-5 architecture was first used to classify MNIST dataset. The Basic Architecture of a CNN is shown in Fig.2.9.

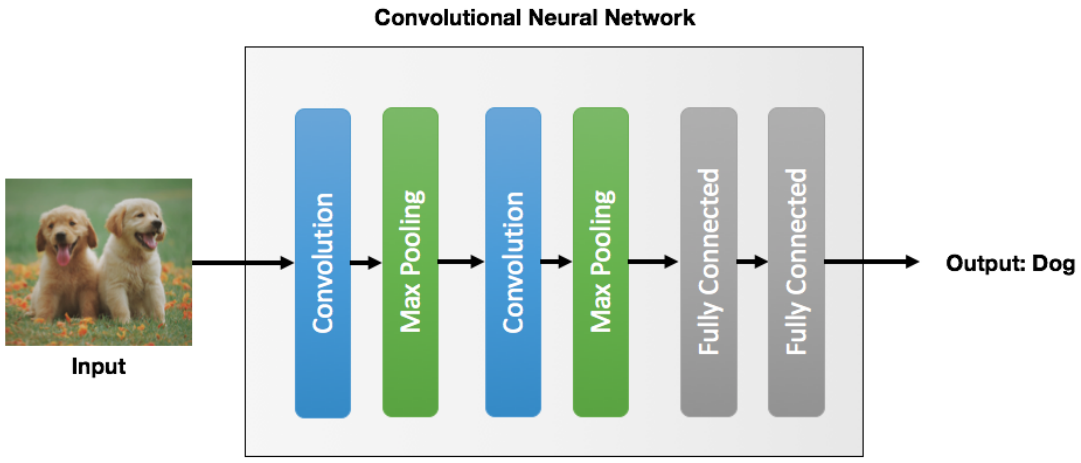



Figure 2.9: Basic CNN Architecture

The CNN architecture can be majorly divided into following parts:

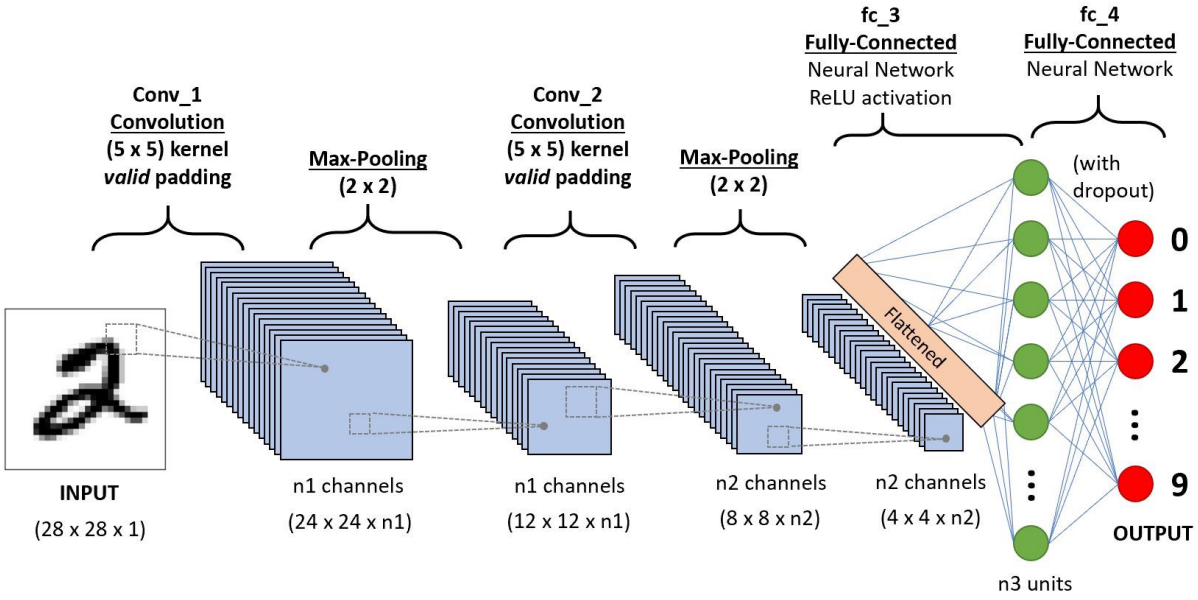



Figure 2.10: Deep CNN Architecture

1. **Input Tensor** : Tensors are N-D arrays. In CNN images are converted to N-Dimensional array, which then are fed as input to the first convolutional layer. The structure of tensors is in the form (I,H,W,C),where,

   - I denotes Image

- H denotes Height of Image in Pixels
- W denotes Width of Image in Pixels
- C denotes the color channel:1 for GrayScale and 3 for RGB

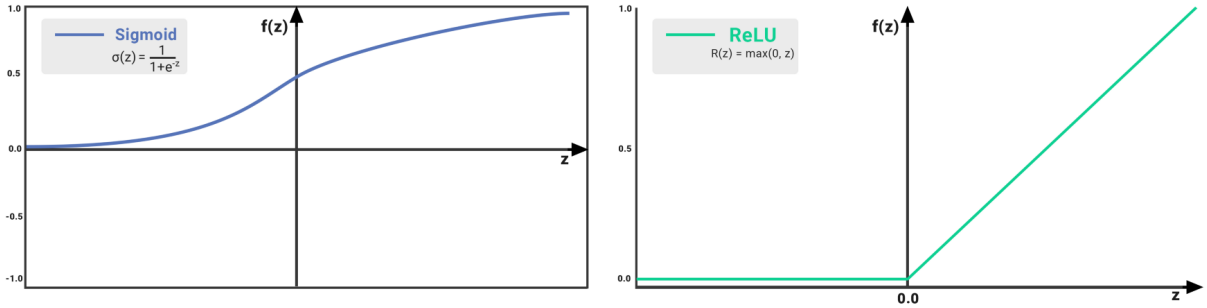
2. **Convolution Layer**: A convolution sweeps the window through images then calculates its input and filter dot product pixel values. This allows convolution to emphasize the relevant features.With this computation, you detect a particular feature from the input image and produce feature maps (convolved features) which emphasizes the important features.

3. **Pooling Layer**: The pooling operation involves sliding a two-dimensional filter over each channel of feature map and summarising the features lying within the region covered by the filter. For a feature map having dimensions h x w x c, the dimensions of output obtained after a pooling layer is: (h - f + 1) / s x (w - f + 1)/s x c

4. **Fully Connected Layer**: The second last layer Flattens all input and gives 1D tensor, which is then used to evaluate output.

5. **Output Tensor**: This is the last layer which directly provides the output of the model.

6. **Activation Functions**: After each convolution or pooling layer we add one activation function so as to train our model for non-linear data as well.Rectified Linear Unit (ReLu) is generally used here. For final output Sigmoid or softmax functions are used, based on output requirements.Fig shows these three activation functions.




(a) Sigmoid Function (b) ReLU Function

Figure 2.11: Activation Functions

### 2.2.2 YOLO and Tiny-YOLO

YOLO or You Only Live Once are real time object detection models, which basically work on extracted features from any CNN model, they later use some maths to identify and

detect specific objects. Since, first being released in 2016, Yolo has been updated 4 times. Here, we will be explaining YOLOv3.

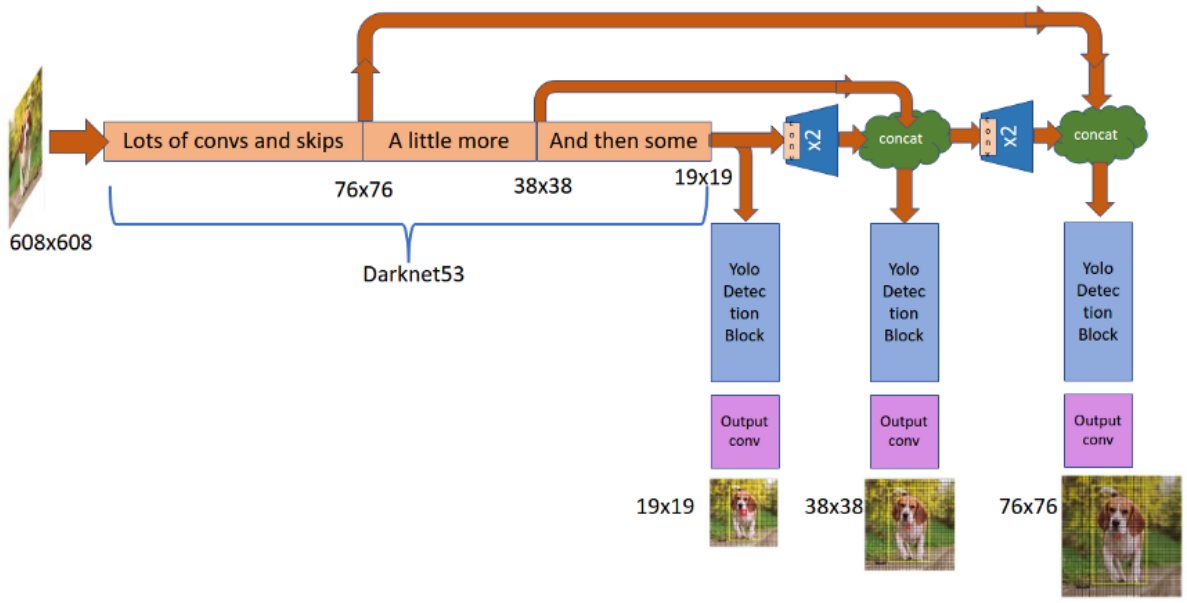



Figure 2.12: YOLOv3 Architecture

YOLO basically works in 3 major steps:

1. It first takes an input image

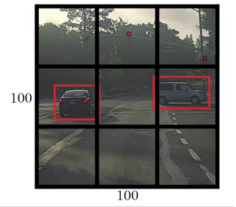



Figure 2.13: Step 1

2. The framework then divides the input image into grids (generally of 3x3 size).classification and localization are applied on each grid.

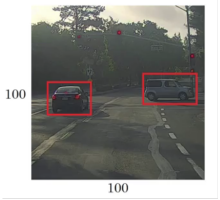



Figure 2.14: Step 2

3. It then predicts the bounding boxes and their corresponding class probabilities for objects.

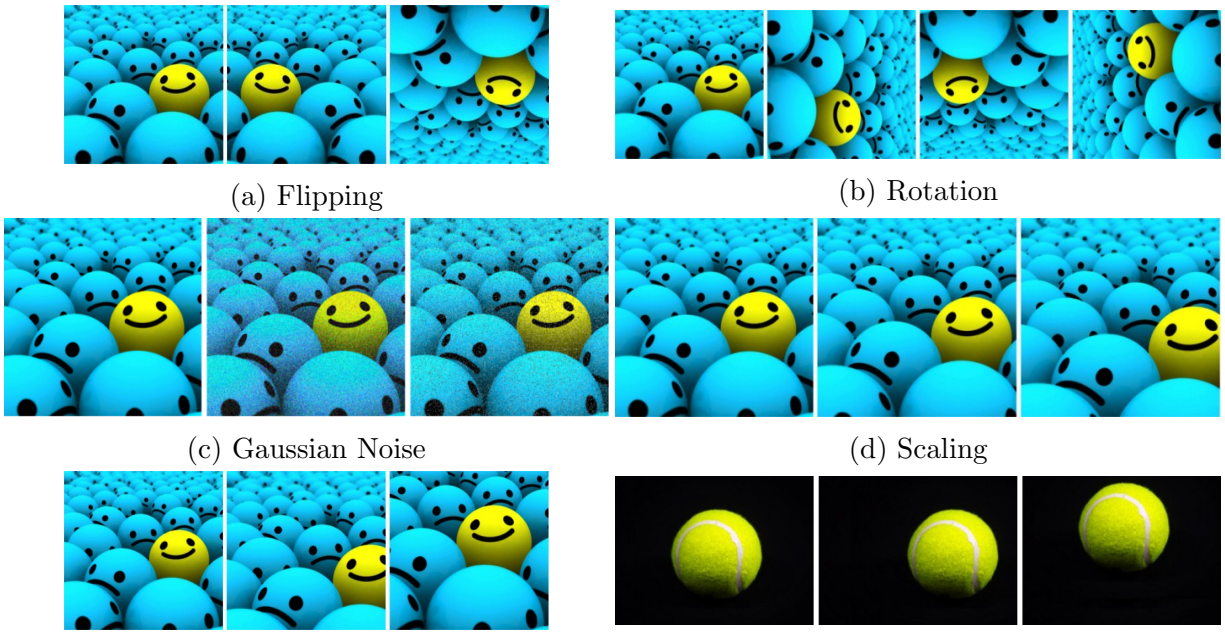
**Tiny-YOLO** is 442 times faster and much smaller brother of YOLO model. Unfortunately, it also causes the accuracy to lower than any other model. But, given that other model take 1000x time, it is better to use Tiny-YOLO whenever high accuracy is not demanded. The following Figure shows the architecture of a Tiny-YOLO used in Google Colaboratory.

```
yolov3-tiny_custom
layer     filters    size              input                output
    0 conv     16  3 x 3 / 1   416 x 416 x   3   ->   416 x 416 x  16  0.150 BFLOPs
    1 max          2 x 2 / 2   416 x 416 x  16   ->   208 x 208 x  16
    2 conv     32  3 x 3 / 1   208 x 208 x  16   ->   208 x 208 x  32  0.399 BFLOPs
    3 max          2 x 2 / 2   208 x 208 x  32   ->   104 x 104 x  32
    4 conv     64  3 x 3 / 1   104 x 104 x  32   ->   104 x 104 x  64  0.399 BFLOPs
    5 max          2 x 2 / 2   104 x 104 x  64   ->    52 x  52 x  64
    6 conv    128  3 x 3 / 1    52 x  52 x  64   ->    52 x  52 x 128  0.399 BFLOPs
    7 max          2 x 2 / 2    52 x  52 x 128   ->    26 x  26 x 128
    8 conv    256  3 x 3 / 1    26 x  26 x 128   ->    26 x  26 x 256  0.399 BFLOPs
    9 max          2 x 2 / 2    26 x  26 x 256   ->    13 x  13 x 256
   10 conv    512  3 x 3 / 1    13 x  13 x 256   ->    13 x  13 x 512  0.399 BFLOPs
   11 max          2 x 2 / 1    13 x  13 x 512   ->    13 x  13 x 512
   12 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   13 conv    256  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 256  0.089 BFLOPs
   14 conv    512  3 x 3 / 1    13 x  13 x 256   ->    13 x  13 x 512  0.399 BFLOPs
   15 conv     36  1 x 1 / 1    13 x  13 x 512   ->    13 x  13 x  36  0.006 BFLOPs
   16 yolo
   17 route  13
   18 conv    128  1 x 1 / 1    13 x  13 x 256   ->    13 x  13 x 128  0.011 BFLOPs
   19 upsample           2x    13 x  13 x 128   ->    26 x  26 x 128
   20 route  19 8
   21 conv    256  3 x 3 / 1    26 x  26 x 384   ->    26 x  26 x 256  1.196 BFLOPs
   22 conv     36  1 x 1 / 1    26 x  26 x 256   ->    26 x  26 x  36  0.012 BFLOPs
   23 yolo
```

Figure 2.15: Tiny-YOLO Architecture

## 2.3 Data Augmentation

Whenever, we have lesser amount of data it is better to create some more data out of the original data so that model can cover all possible corners of a data set. This also includes all possible unseen Data. Some of the basic augmentation techniques are explained in the following set of figures.



(a) Flipping

(b) Rotation

(c) Gaussian Noise

(d) Scaling

(e) Cropping

(f) Transforming

Figure 2.16: Data Augmentation Techniques

# Chapter 3

# Implementation

*This Chapter goes through the journey of data collection and annotation part. Since in every machine learning project data is a crucial entity, a chapter devoted to it seems legit.*

## 3.1 Project Architecture

The following process flow is self-explanatory:

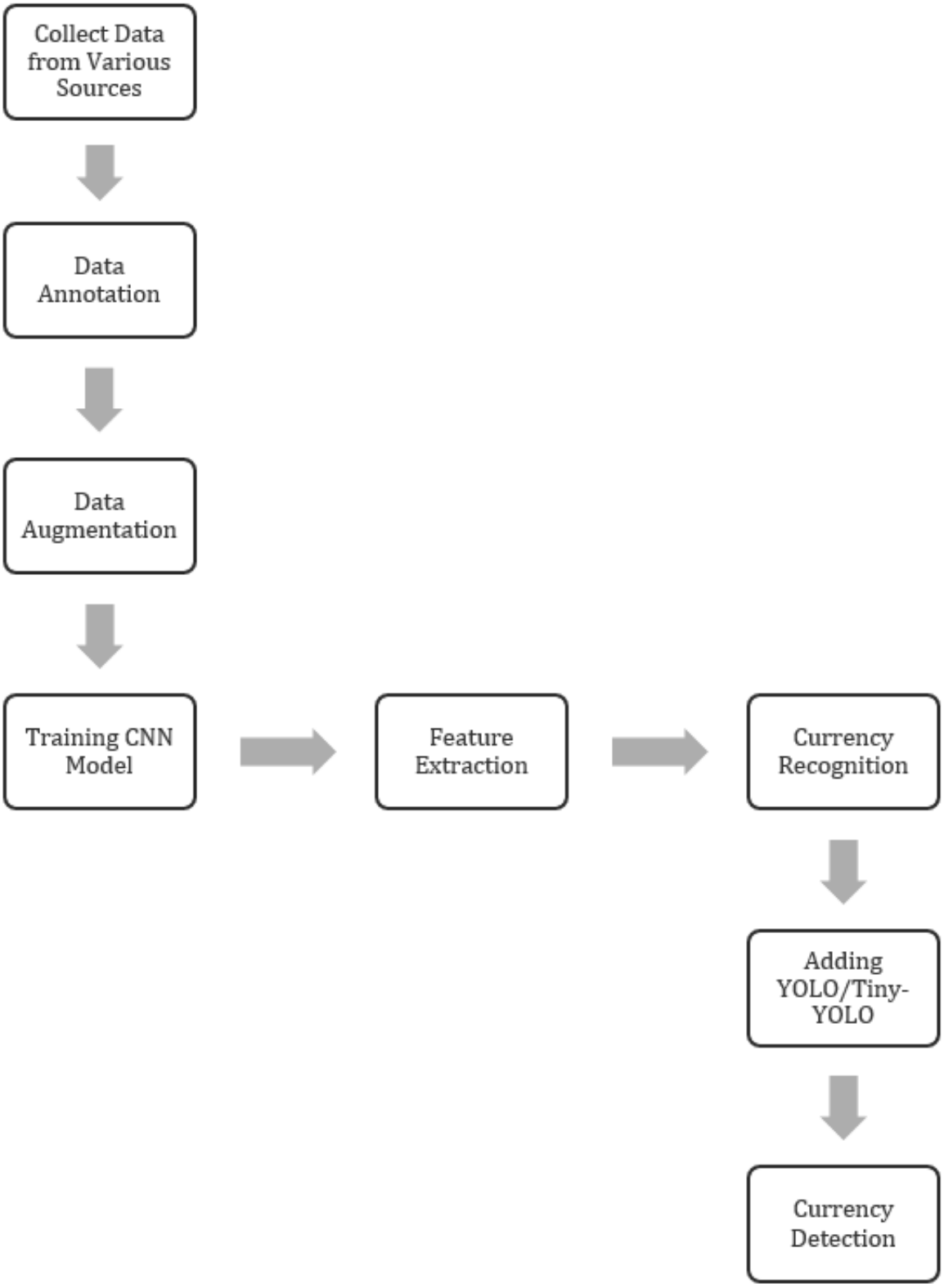



Figure 3.1: The Complete Project Process

## 3.2 Data Preparation

Data collection and annotation was the most cuber-some part of this project. Since, not many people have done project on this topic and if they had, they didn't share the data they used. It is understandable, since data is the new oil. But it makes it very difficult for someone like me to procure appropriate amount of data for this project. Data Preparation took around 4-5 months of hard work in which, most of the time was spent on annotation part.

## 3.3 Getting Dataset

Other than Kaggle, being the majority data provider for this project, there were other ways like clicking pictures of the currency from different angle and under various light shades, to collect data. The data set includes various mixtures of new & old Indian currencies, with around 450 samples of each denomination.

Around 450 images containing abstract background are also included, to make the model more efficient. Part of the data is collected from Kaggle, Google search and a small portion of high quality images are added from my side, just to make the data set more wholesome. The images varied from lower side of quality 150x150 to high quality images of 4000x4000 pixel sizes. They vary from 5 KB to around 4000 KB at the max. The reason for collecting such variety of image was to bring robustness to the model.

Following table shows exact numbers of images of respective denominations: Also, here

| Denomination | Training Count | Validation Count |
|---|---|---|
| Rs.10 | 448 | 45 |
| Rs.20 | 449 | 45 |
| Rs.50 | 450 | 45 |
| Rs.100 | 450 | 45 |
| Rs.200 | 500 | 45 |
| Rs.500 | 437 | 45 |
| Rs.2000 | 438 | 45 |

Table 3.1: Count of Denomination for Training and Validation

are some sample images of the data collected :

Figure 3.2: Sample of Rs.10 Denomination

Figure 3.3: Sample of Rs.20 Denomination

Figure 3.4: Sample of Rs.50 Denomination

Figure 3.5: Sample of Rs.100 Denomination

Figure 3.6: Sample of Rs.200 Denomination

Figure 3.7: Sample of Rs.500 Denomination


Figure 3.8: Sample of Rs.2000 Denomination

## 3.4 Marking Data

After the data has been divided into Training and Testing portion, following is the directory view of whole Data Set:

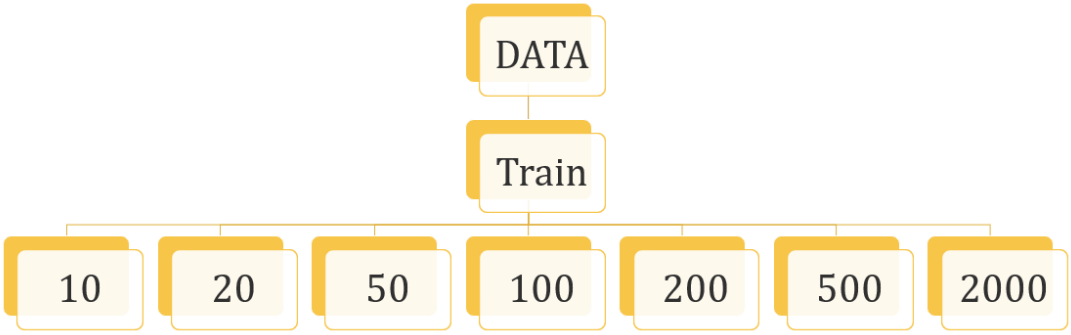



Figure 3.9: Directory View of Training Data

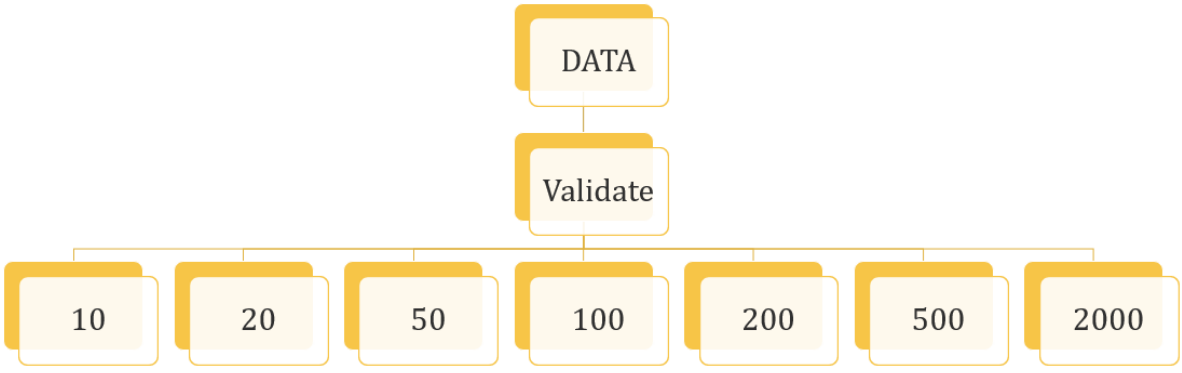



Figure 3.10: Directory View of Testing Data

### 3.4.1 LabelImg Tool

LabelImg is a graphical image annotation tool.It is written in Python and uses Qt for its graphical interface. Following below steps LabelImg tool was used for the annonation of each image:

1. Open LabelInmg and then open Directory in which data is kept

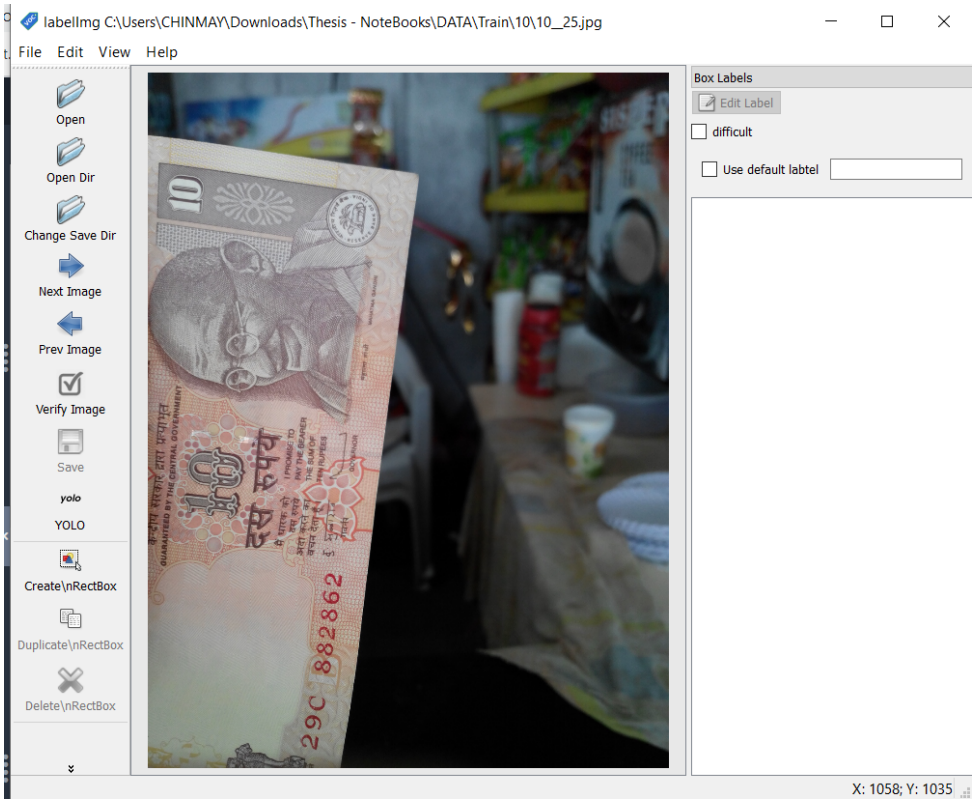



Figure 3.11: Annotation - Step 1

2. Create a rectanlge box around the denomination in the image and select its class. In this case Rs.10.

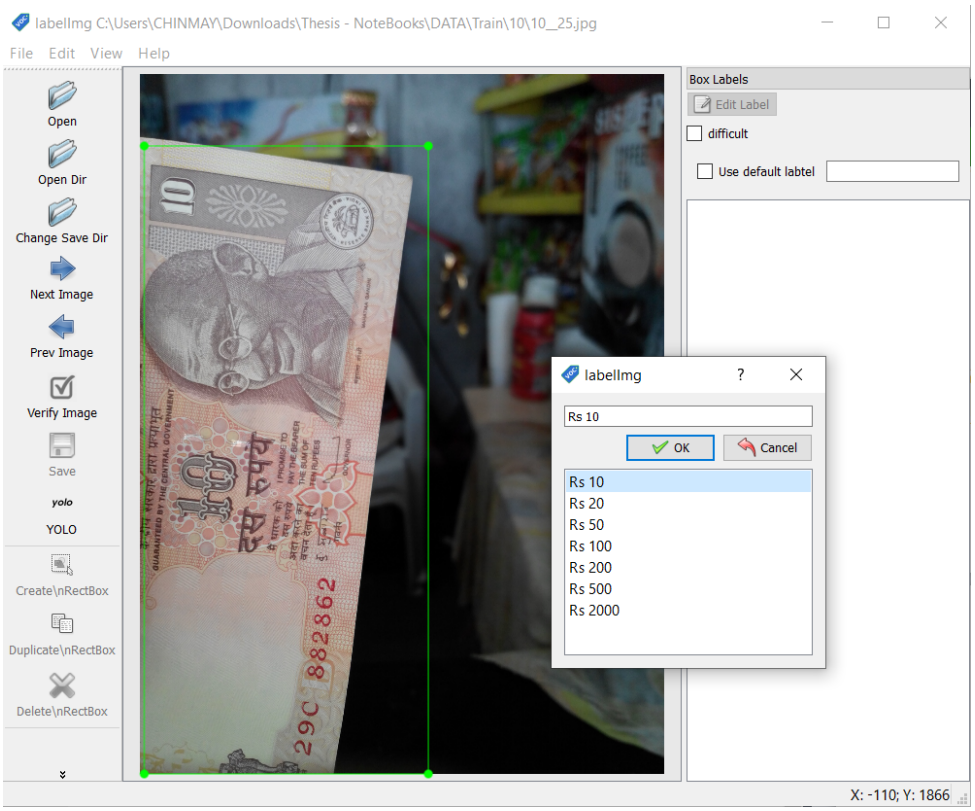



Figure 3.12: Annotation - Step 2

3. Select the save director same as the data directory.

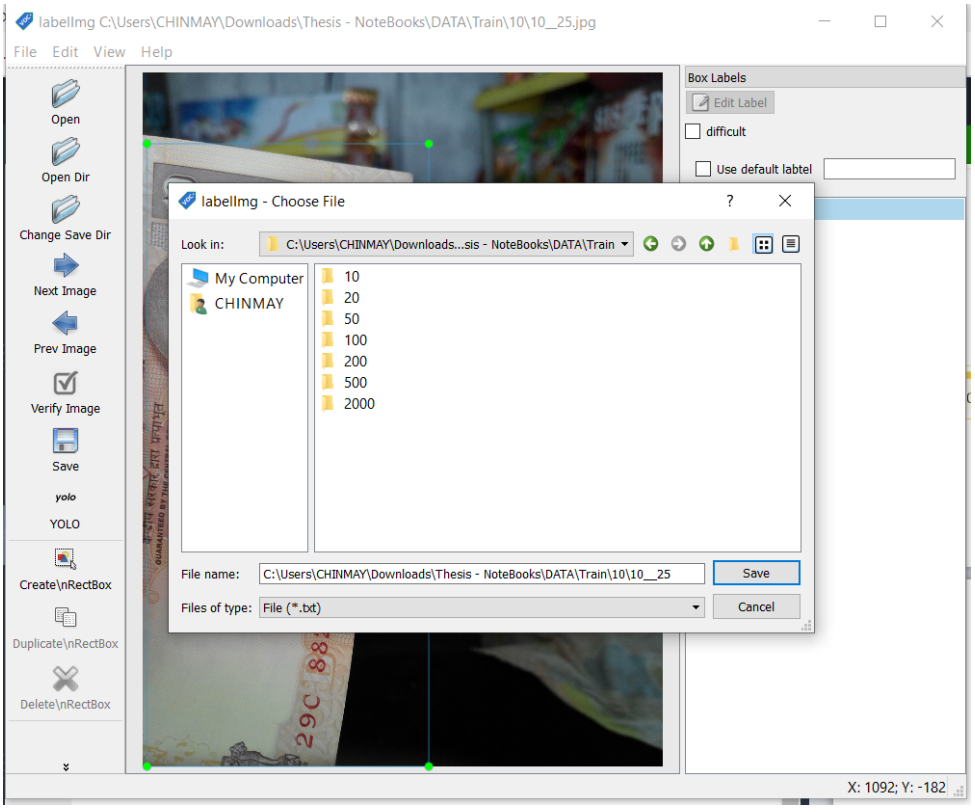



Figure 3.13: Annotation - Step 3

4. Resulting Annotation as in LabelIMG interface.

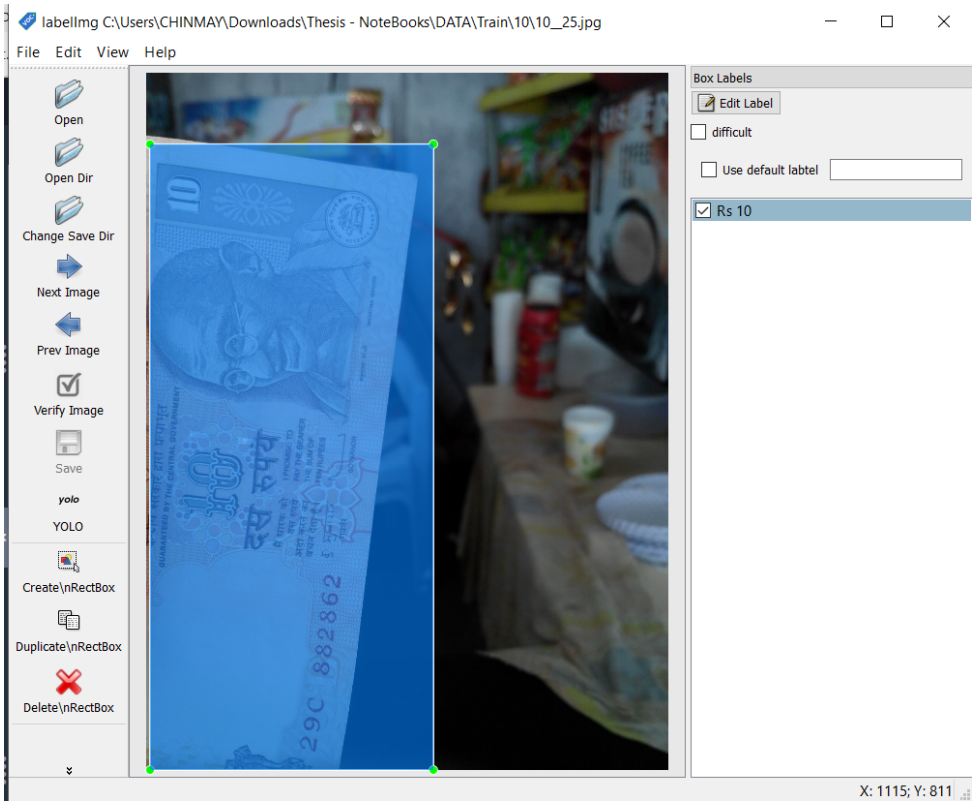



Figure 3.14: Annotation - Step 4

5. The directory will finally contain two files now. First being the original image file and other being a text file containing heightX, heightY, widthX, widthY of the denomination in the image.

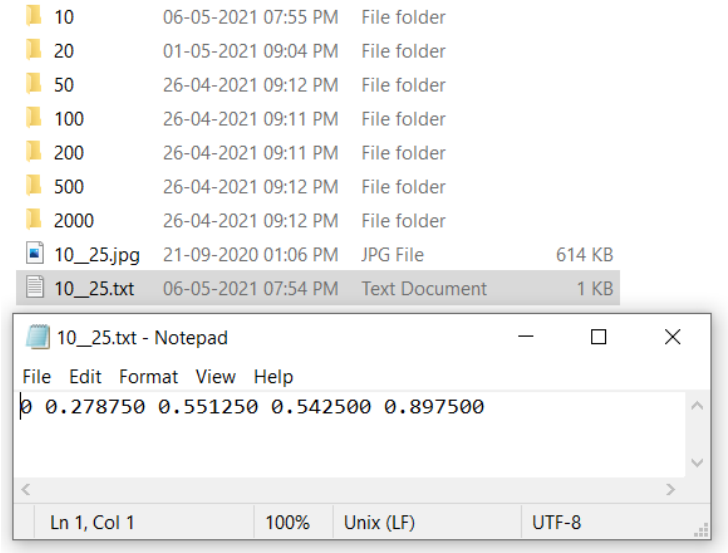



Figure 3.15: Annotation - Step 5

Here, annotation was done for 3172 images (Training) and 315 images(Test).

## 3.5 Data Augmentation

Since, in real life, the denominations are not as crisp as needed by the model, Data Augmentation becomes a necessity. Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning models. Using Adobe Photoshop, and Keras, images are rotated on various angles, and some are even cropped, to make the training process more robust.

Although, our data set is sufficiently large, still more the data more accurate the model is going to be. Hence, we move on towards data augmentation procedure. Re-scaling of images are done to make them normalized. Thus, each image is divided by 255 (here, it is understood that images are actually tensors of numerical data ranging from 0-255). Other than this, no further augmentations were performed since, model was efficient enough with the given images.

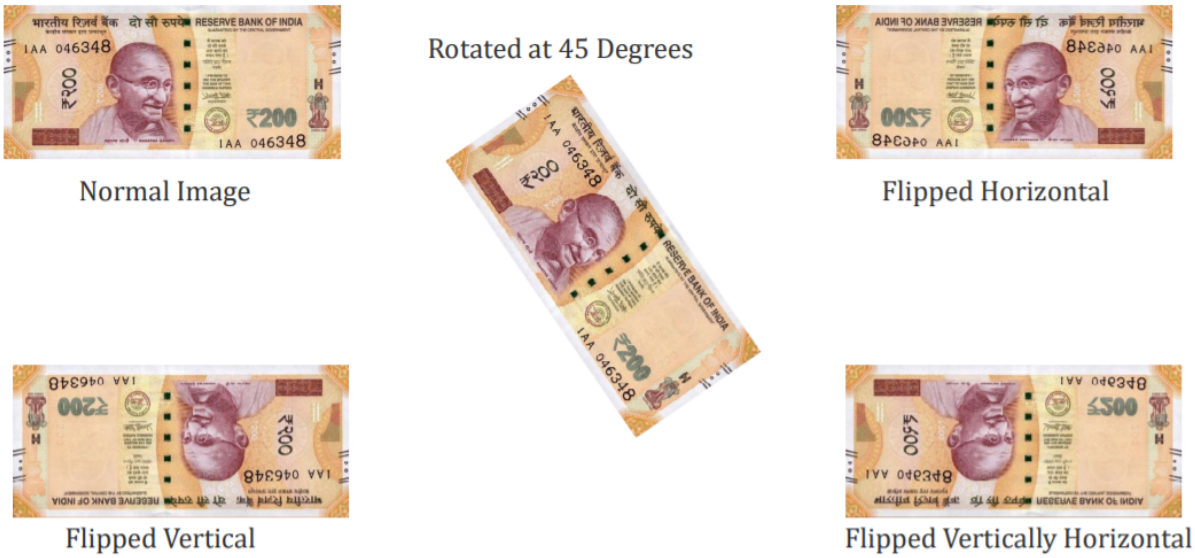



Figure 3.16: Rs.200 sample image augmented

# Chapter 4

# Experiments and Analysis

*In this chapter, we move towards model implementation portion. This majorly includes training CNN Model for feature extraction and then using those features for training of a YOLO model or better Tiny-YOLO model. The chapter ends with the result declaration of the whole process.*

## 4.1 Experiments

### 4.1.1 CNN Model

Training a convolutional neural network requires a hug amount of data, since extracting specific features from thousands of pixels is not an easy task. Here, experimentation was done on majorly two types of CNN models. The two models vary namely on the basis of their input dimensions as well as size. Other than that both the models are exactly the same in terms of number of Convolution layers etc. Later for time saving purposes **early stopping** and **checkpoints** was used as well. Fig 4.1 shows the architecture of CNN used for the project.

```
model1 = Sequential()
model1.add(Conv2D(32, (3,3), activation="relu", input_shape=(inp_x,inp_y,3)))
model1.add(MaxPooling2D(2,2))
model1.add(Conv2D(32, (3,3), activation="relu"))
model1.add(MaxPooling2D(2,2))
model1.add(Dropout(0.2))
model1.add(Conv2D(64, (3,3), activation="relu"))
model1.add(MaxPooling2D(2,2))
model1.add(Conv2D(64, (3,3), activation="relu"))
model1.add(MaxPooling2D(2,2))
model1.add(Dropout(0.3))
model1.add(Flatten())
model1.add(Dense(128, activation="relu"))
model1.add(Dense(7, activation="softmax"))
model1.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])
model1.summary()
```

Figure 4.1: CNN Architecture Used

- First, two layers are having inputs as 32x32 convolution layers with a 3x3 filter and activation function as ReLU. After each conv2d layer there is one Max Pooling layer of 2x2 size.
- After first two sets of Convolution and Max Pooling layers, a Dropout layer is added with 20% dropout ratio.
- Then, again two layers are having inputs as 64x64 convolution layers with a 3x3 filter and activation function as ReLU. After each conv2d layer there is one Max Pooling layer of 2x2 size.
- Another Dropout layer is added with 30% dropout ratio.
- Flatten Layer is added to convert all 3D tensors to 1D tensor.
- Now a dense layer with input=128 is added, with ReLU activation function.
- Now a dense layer with input=7 is added, this layer has soft-max activation function as this is the Output layer of the model.
- Finally, the model is compiled with Adam optimizer,categorical cross entropy loss function and accuracy as metrics.

- Early Stopping was used which monitored validation loss, with aim of finding minimum validation loss after at least 5 Epochs were resulting in no change.
- Model checkpoint was used to save the best model after every epoch.
- The same architecture is used for both type of models, with varying params.

1. **The 150 CNN Model:**

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 128)               401536
_________________________________________________________________
dense_1 (Dense)              (None, 7)                 903
=================================================================
Total params: 468,007
Trainable params: 468,007
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.2: 150x150 Model Summary

The Model has input as (150,150,3), where 150x150 is the new image dimensions and 3 represents RGB color coding. Rest everything else is same as explained earlier.

2. **The 224 CNN Model:**

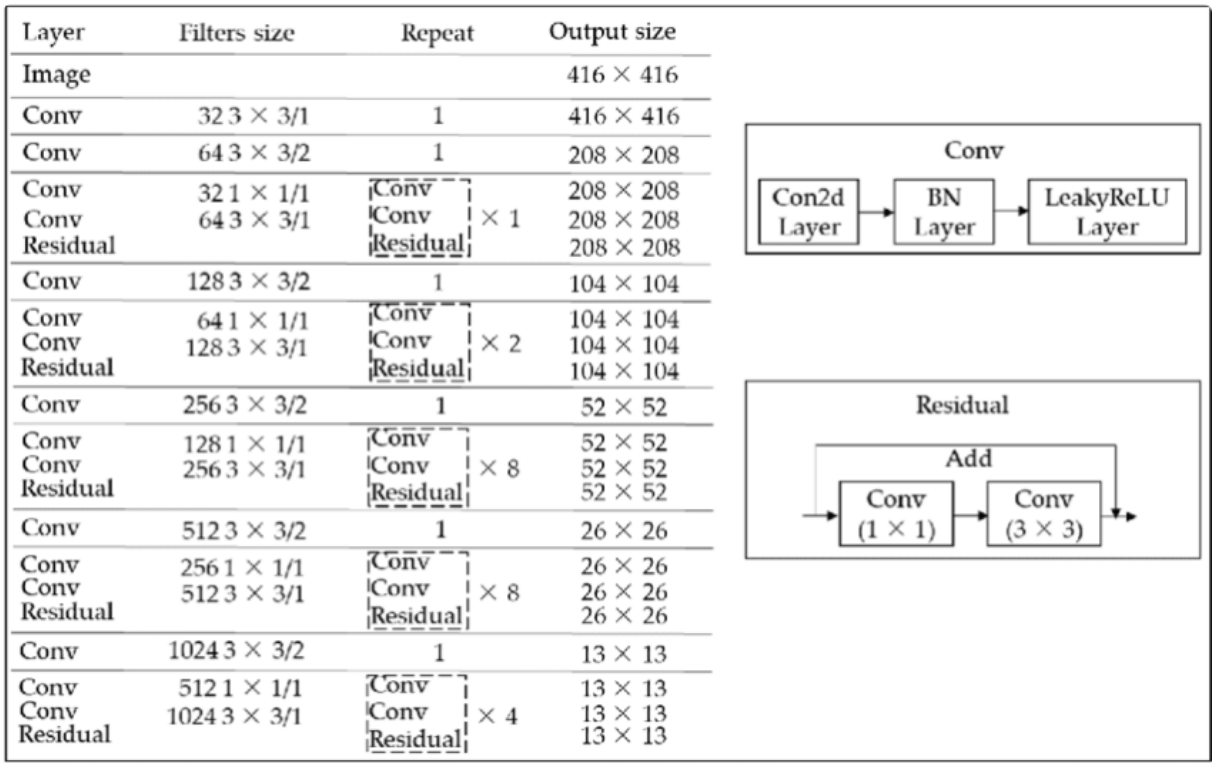
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 222, 222, 32)      896
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 109, 109, 32)      9248
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 54, 54, 32)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 54, 54, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 52, 52, 64)        18496
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 26, 26, 64)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 24, 24, 64)        36928
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 12, 12, 64)        0
_________________________________________________________________
dropout_5 (Dropout)          (None, 12, 12, 64)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 9216)              0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               1179776
_________________________________________________________________
dense_5 (Dense)              (None, 7)                 903
=================================================================
Total params: 1,246,247
Trainable params: 1,246,247
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.3: 224x224 Model Summary

The Model has input as (224,224,3), where 224x224 is the new image dimensions and 3 represents RGB color coding. Rest everything else is same as explained earlier.

### 4.1.2 YOLO Object Detection Model

YOLO although very fast, still took around 6-7 hours to be fully train on a GPU for just 70 images of training data set. Thus, it was practically impossible for the complete data set to be used.



| Layer | Filters size | Repeat | Output size |
|---|---|---|---|
| Image | | | 416 × 416 |
| Conv | 32 3 × 3/1 | 1 | 416 × 416 |
| Conv | 64 3 × 3/2 | 1 | 208 × 208 |
| Conv<br>Conv<br>Residual | 32 1 × 1/1<br>64 3 × 3/1 | Conv<br>Conv<br>Residual × 1 | 208 × 208<br>208 × 208<br>208 × 208 |
| Conv | 128 3 × 3/2 | 1 | 104 × 104 |
| Conv<br>Conv<br>Residual | 64 1 × 1/1<br>128 3 × 3/1 | Conv<br>Conv<br>Residual × 2 | 104 × 104<br>104 × 104<br>104 × 104 |
| Conv | 256 3 × 3/2 | 1 | 52 × 52 |
| Conv<br>Conv<br>Residual | 128 1 × 1/1<br>256 3 × 3/1 | Conv<br>Conv<br>Residual × 8 | 52 × 52<br>52 × 52<br>52 × 52 |
| Conv | 512 3 × 3/2 | 1 | 26 × 26 |
| Conv<br>Conv<br>Residual | 256 1 × 1/1<br>512 3 × 3/1 | Conv<br>Conv<br>Residual × 8 | 26 × 26<br>26 × 26<br>26 × 26 |
| Conv | 1024 3 × 3/2 | 1 | 13 × 13 |
| Conv<br>Conv<br>Residual | 512 1 × 1/1<br>1024 3 × 3/1 | Conv<br>Conv<br>Residual × 4 | 13 × 13<br>13 × 13<br>13 × 13 |



Figure 4.4: General YOLO Architecture

The architectures uses DarkNet-53, which is a very robust and deeper feature extraction architecture of CNN. It has 53 convol layers with Leaky ReLU and batch normalization layers. No form of pooling is included, but striding of 2 is used instead. Later predicitons are done based on logistic regression.

Altough successful, YOLO was still slow and did most of the unnecessary calculations for almost every image. Thus, for saving time and calculations another bettter model was approached.

### 4.1.3 Tiny-YOLO Model

With just 70 images, YOLO proved why it is considered to be the best one available today. It still took 7 hours for complete execution, for which Tiny-YOLO was used instead of FULL YOLO.

It has 24 convolutional layers and in the end 2 FC layers. The convolutional layers are pretrained on the ImageNet classification task at half the resolution (224 × 224 input image) and then the resolution is doubled for the detection training.

```
yolov3-tiny_custom
layer     filters    size              input                output
    0 conv     16  3 x 3 / 1   416 x 416 x   3   ->   416 x 416 x  16  0.150 BFLOPs
    1 max          2 x 2 / 2   416 x 416 x  16   ->   208 x 208 x  16
    2 conv     32  3 x 3 / 1   208 x 208 x  16   ->   208 x 208 x  32  0.399 BFLOPs
    3 max          2 x 2 / 2   208 x 208 x  32   ->   104 x 104 x  32
    4 conv     64  3 x 3 / 1   104 x 104 x  32   ->   104 x 104 x  64  0.399 BFLOPs
    5 max          2 x 2 / 2   104 x 104 x  64   ->    52 x  52 x  64
    6 conv    128  3 x 3 / 1    52 x  52 x  64   ->    52 x  52 x 128  0.399 BFLOPs
    7 max          2 x 2 / 2    52 x  52 x 128   ->    26 x  26 x 128
    8 conv    256  3 x 3 / 1    26 x  26 x 128   ->    26 x  26 x 256  0.399 BFLOPs
    9 max          2 x 2 / 2    26 x  26 x 256   ->    13 x  13 x 256
   10 conv    512  3 x 3 / 1    13 x  13 x 256   ->    13 x  13 x 512  0.399 BFLOPs
   11 max          2 x 2 / 1    13 x  13 x 512   ->    13 x  13 x 512
   12 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   13 conv    256  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 256  0.089 BFLOPs
   14 conv    512  3 x 3 / 1    13 x  13 x 256   ->    13 x  13 x 512  0.399 BFLOPs
   15 conv     36  1 x 1 / 1    13 x  13 x 512   ->    13 x  13 x  36  0.006 BFLOPs
   16 yolo
   17 route  13
   18 conv    128  1 x 1 / 1    13 x  13 x 256   ->    13 x  13 x 128  0.011 BFLOPs
   19 upsample           2x    13 x  13 x 128   ->    26 x  26 x 128
   20 route  19 8
   21 conv    256  3 x 3 / 1    26 x  26 x 384   ->    26 x  26 x 256  1.196 BFLOPs
   22 conv     36  1 x 1 / 1    26 x  26 x 256   ->    26 x  26 x  36  0.012 BFLOPs
   23 yolo
```

Figure 4.5: Tiny-Yolo Architecture used in project

Results showed that there was not much significant difference between accuracy's produced by both the models.

## 4.2 Results

This section contains the final results of all the hardwork done till now. From collecting data to the limitations of the experiments. Lets see how it all turned out.

### 4.2.1 Data Collection and Experimental Environment

**Data Collection** count is shown in Table 3.1, which was used to train CNN model. But later due to YOLO and Tiny-YOLO taking nearly impractical time to be executed, the data set was reduced to 10 images per class for training and 3 images for testing.

**Experimental Environment** used is shown in the below table.

| Hardware | Configuration |
|---|---|
| Processor | 2 |
| Vendor ID | GenuineIntel |
| CPU Family | 4 |
| Mooel | 63 |
| Model Name | Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz |
| Cache size | 46080 KB |
| Cpu cores | 4 |
| Clflush size | 64 |
| Cache Alignment | 64 |
| Address Sizes | 46 bits physical, 48 bits virtual |
| Thread(s) per core | 2 |
| GPU 0 | NVIDIA GeForce MX250 |
| CUDA Version | 11.2 |

Table 4.1: System Configuration of local machine

| Hardware | Configuration |
|---|---|
| Processor | 2 |
| Vendor ID | GenuineIntel |
| CPU Family | 6 |
| Mooel | 63 |
| Model Name | Intel(R) Xeon(R) CPU @ 2.30GHz |
| Cache size | 46080 KB |
| Cpu cores | 1 |
| Clflush size | 64 |
| Cache Alignment | 64 |
| Address Sizes | 46 bits physical, 48 bits virtual |
| Thread(s) per core | 2 |
| GPU 0 | Tesla P100-PCIE-16GB |
| CUDA Version | 11.2 |

Table 4.2: System Configuration of Google Colab

### 4.2.2 Feature Extraction

For a total of 3172 images belonging to 7 classes for training and 315 images belonging to 7 classes for testing, two types of models were trained. Both the model had same architecture, they differed in the input dimension (one had 150x150 and the other had 224x224). Further details of both the models have been already discussion in section 4.1.1. Following two tables show the results after running sufficient epochs.

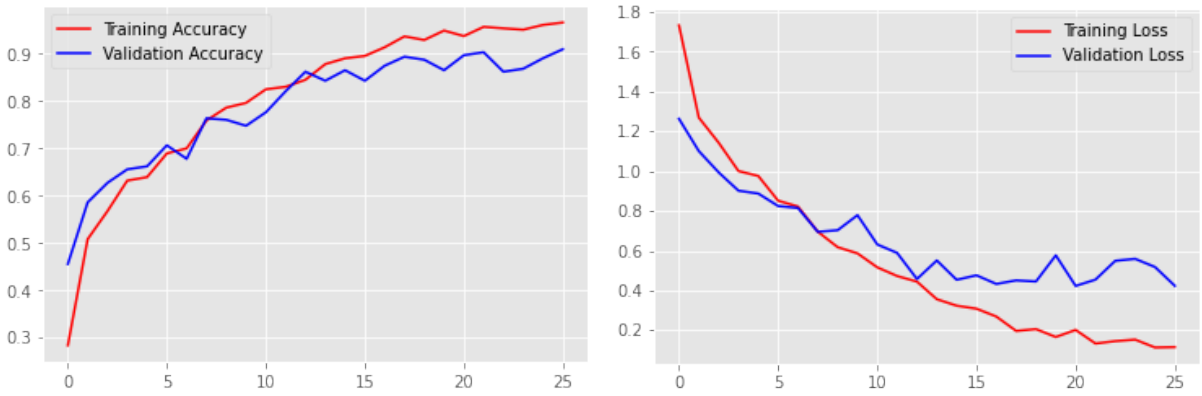
| | |
|---|---|
| Designated Epochs | 100 |
| Epoch Ran | 25 |
| Time | 50.89 min |
| Accuracy | 96.270 |
| Validation Accuracy | 92.206 |
| Loss | 0.10448 |
| Validation Loss | 0.28608 |

Table 4.3: Results from 150 CNN Model

| | |
|---|---|
| Designated Epochs | 100 |
| Epoch Ran | 21 |
| Time | 61.81 min |
| Accuracy | 95.870 |
| Validation Accuracy | 89.206 |
| Loss | 0.18448 |
| Validation Loss | 0.38608 |

Table 4.4: Results from 224 CNN Model

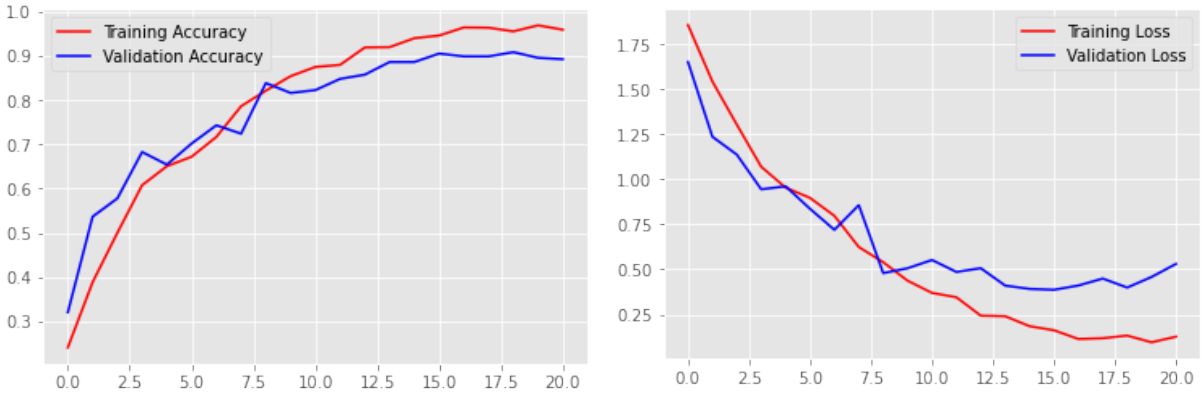
Early stopping was used here, hence model ran till 25 and 21 epochs respectively, and stopped if for 5 or more epochs validation loss did not change.




(a) 150 Model Accuracy

(b) 150 Model Loss

Figure 4.6: 150 Model Results




(a) 224 Model Accuracy

(b) 224 Model Loss

Figure 4.7: 224 Model Results

Based on above results, 224 model was chosen as it proved better results in all forms of metric.

### 4.2.3 Currency Detection

Using pre-trained CNN model as YOLOs and tiny YOLO's front end, 70 images were trained with the following parameters and their initialised values.

| | |
|---|---|
| Batch | 64 |
| Subdivisions | 16 |
| Width | 608 |
| Height | 608 |
| Channels | 3 |
| Momentum | 0.9 |
| Decay | 0.0005 |
| Saturation | 1.5 |
| Exposure | 1.5 |
| Hue | 0.1 |
| Learning rate | 0.001 |
| Max batches | 14000 |
| Steps | 11200,16800 |

Table 4.5: Hyper parameters for YOLO Model

| | |
|---|---|
| Batch | 64 |
| Subdivisions | 16 |
| Width | 416 |
| Height | 416 |
| Channels | 3 |
| Momentum | 0.9 |
| Decay | 0.0005 |
| Angle | 0 |
| Saturation | 1.5 |
| Exposure | 1.5 |
| Hue | 0.1 |
| Learning rate | 0.001 |
| Max batches | 14000 |
| Steps | 11200,16800 |

Table 4.6: Hyper parameters for Tiny-YOLO Model

The **tiny-YOLO model** secured an accuracy of **99.34** along as well as training loss of **0.3675**. While YOLO was taking impractical time, thus that approach was not continued further.

# Chapter 5

# Conclusion and Future Work

In this thesis, the main focus was object detection, the object in this case being denominations from Indian Currency. As seen by the results Tiny-YOLO performed way better than expectation for a small data set of 70 images only. From getting a data set of around 4000 images, then extracting features from those images using customized CNN. Later, training a Tiny-YOLO based on that CNN architecture took real hard work and has helped me gain enormous amount of knowledge in the field of Object Recognition and Object Detection.

## 5.1 Future Work

- More data is not always significantly beneficial. As seen in this case, although the data set was sufficiently large, due to computational restrictions, it could not be completely utilized. In future, if the computational advancements allow, complete or at the very least 50% of data could be used.
- CNN Model can be further trained with different sets of parameters that include number of hidden layers as well. The input can be varied for faster computation as well.
- The trained model from the final Tiny-YOLOs result can be used in developing a mobile or computer application, which can be used by people for easily detecting Indian currency. It won't require much of training, and hence, for a professional developer it is an easy task.

# Bibliography


[1] Deep Learning Based Indian Currency Detection for Visually Challenged using VGG16 - Nijil Raj N, Anandu S Ram, Aneeta Binoo Joseph, Shabna S, (International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-9 Issue-2, July 2020 969 Published By: Blue Eyes Intelligence Engineering & Sciences Publication Retrieval Number: B3955079220/2020©BEIESP DOI:10.35940/ijrte.B3955.079220 )

[2] Indian Currency Recognition from Live Video Using Deep Learning - Kushal Bhavsar, Keyurbhai Jani , and Rakeshkumar Vanzara (U. V. Patel College of Engineering, Ganpat University, Mehsana 384012, Gujarat, India)

[3] Indian Currency Recognition using Neural Network Pattern Recognition Tool - Mr.Viranchi N Patel1, Dr.Udesang K Jaliya2 and Mr.Keyur N Brahmbhatt (BVM Engineering College, V.V.Nagar, India )

[4] Currency Recognition Using Deep Learning - Zhang Qian,Yan, Weiqi (Auckland University of Technology, Master of Computer and Information Sciences,2018-11-18T03:45:35Z)

[5] Komal Vora, Ami Shah, Jay Mehta, —A Review Paper on Currency Recognition System‖ International Journal of Computer Applications (0975 – 8887), Volume 115 – No. 20, April 2015

[6] Chinmay Bhurke, Meghana Sirdeshmukh, Prof. Mrs. M.S.Kanitkar, —Currency Recognition Using Image Processing‖ International Journal of Innovative Research in Computer and Communication Engineering, Vol. 3, Issue 5, May 2015

[7] G. Trupti Pathrabe, Mrs.Swapnili Karmore, A Novel Approach of Embedded System for Indian Paper Currency Recognition, International Journal of Computer Trends and Technology, May to June Issue 2011, ISSN: 2231-2803.

[8] EbtesamAlthafiri, Muhammad Sarfraz, Muhannad Alfarras, "Bahraini Paper Currency Recognition", Journal of Advanced Computer Science and Technology Research, June 2012, Vol. 2 No.2, Pp. 104-115.

[9] Rubeena Mirza, Vinti Nanda, Characteristic Extraction Parameters for Genuine Paper Currency Verification Based on Image Processing, IFRSA International Journal of Computing, Volume 2, Issue 2, April 2012.

[10] Pathrabe T, Bawane N.G, Feature Extraction Parameters for Genuine Paper Currency Recognition & Verification, International Journal of Advanced Engineering Sciences and Technologies, Volume 2, 85-89, 2011.

[11] Masato Aoba, Tetsuo Kikuchi, YoshiyasuTakefuji, "Euro banknote recognition system using a three-layer perceptron and RBF networks", IPSJ Transaction on Mathematical Modeling and Its Application, Vol 44, No. SIG 7 (TOM 8), May 2003, Pp. 99-109.

[12] Kalyan Kumar Debnath, Sultan Uddin Ahmed, Md. Shahjahan, "A Paper Currency Recognition System. Using Negatively Correlated Neural Network Ensemble", Journal Of Multimedia, December 2010, Vol. 5, No. 6, Pp. 560-567.

[13] Yifeng Liu, Lin Zeng Haar-SVM for Real-time Banknotes Recognition Journal of Information & Computational Science 11:12 (2014) 4031–4039 August 10, 2014.

[14] Faiz M. Hasanuzzaman, Xiaodong Yang, and Ying Li Tian, Senior Member, IEEE Robust and Effective Component-b