1. INTRODUCTION

1.1 Introduction:

Technology is growing very fast these days. Consequently the banking sector is also getting modern day by day. This brings a deep need of automatic fake currency detection in automatic teller machine and automatic goods seller machine. Many researchers have been encouraged to develop robust and efficient automatic currency detection machines. Automatic machine which can detect banknotes are now widely used in dispensers of modern products like candies, soft drinks bottle to bus or railway tickets. The technology of currency recognition basically aims for identifying and extracting visible and invisible features of currency notes. Until now, many techniques have been proposed to identify the currency note. But the best way is to use the visible features of the note. For example, colour and size.

This way is not helpful if the note is dirty or torn. If a note is dirty, its colour characteristics are changed widely. So it is important that how we extract the features of the image of the currency note and apply proper algorithm to improve accuracy to recognize the note. The evolution of technology, has lead to an increase in the ways in which fake forms of currencies are created and thus the counterfeit currency notes in the Indian market have increased. These fake or counterfeit notes have various ill-effect on society. They harm the lay man as there is no definitive way to be absolutely sure if the currency is original or fake. This report focuses on the comparison and analysis of different image processing algorithms to find the most efficient algorithm to identify and classify real and counterfeit currency notes and to find the denomination of the currency.

The comparative study for various image processing algorithms was conducted to identify and select the one which will be able to extract more prominent features, and is also better in terms of processing time, outlier rejection, and efficiency in computation and in feature matching. The rapid advancement of technology, there is also an increase in the production of counterfeit currencies, which is also making it difficult to distinguish between the real and the fake ones. The proposed system will be able to recognize the currencies and classify them as real or fake. In this approach, we will classify the fake notes using an image processing algorithm. Automatic recognition of fake Indian currency is very important in major domains like banking nowadays. This system is used to detect whether the currency is fake or original through the auto mated system which is through convolution neural network and recurrent neural network.

Deep learning excels in the task of recognition and classification of images over a large data sets, which is also primarily used in object category recognition. In the recent demonetization drive may be a step towards eradication of corruption and black money, but it fails to address the problem of counterfeit currency.

1.2 Problem Statement:

A dataset containing features extracted after wavelet transformation of images of currency notes, use the features available in the dataset to design a supervised learning model which can identify whether a currency note is real or fake. We begin our investigation by analyzing the dataset for any missing values or any outliers. Since, the four features that we are going to use to train our models are continuous in nature; we will normalize them in the range of 0 to 1. Normalization or scaling ensures that all features are treated equally when applying supervised learners. We would also look if any feature is skewed or has outliers, i.e., it has a sample with vastly different value than those in other samples. In case any feature is skewed we can apply log transform before scaling the feature. The next step will be to establish a benchmark model that we can use to make performance comparisons. For this problem, we define our benchmark model to be a naïve classifier, which classifies all banknotes as fake.

We will define few metrics to evaluate each model including our naïve classifier. We aim that our designed model should perform better than the naïve predictor. Since, this naïve prediction model does not consider any information to substantiate its claim; it helps establish a benchmark for whether a model is performing well. That been said, using a naïve prediction would be pointless as will predict all notes counterfeit and would not identify any note as genuine. Once our data is in good form, we will split our data into training and test sets. Thereafter, we will investigate different supervised learning algorithms and determine which is best at modelling the data. We will compare each algorithm with naïve predictor to evaluate its performance. Further, we will select one algorithm and fine-tune the chosen model on few of its important parameters using grid search. This will give us our final optimized and fine-tuned model.

2. LITERATURE SURVEY

Currently the major methods used for the classification of currency notes be it Indian or any other are either neural networks or image processing or a mixture of both. Different papers have taken various different approaches to do this task and each of these existing system give us a perspective into the challenges of detecting currency as real or fake. The many image processing algorithms that have been used each have their merits and demerits and in our paper we have analysed each of these in more detail so that we may achieve an optimal process for a real time application to do the task of identification and detection. The following papers were referred to get background on the system requirements.

1) The Paper Indian Banknote Recognition using Convolutional Neural Network Shubham Mittal, Shiva Mittal took a Deep learning based approach for the purpose of identification of denominations of Indian currency. And while that method worked efficiently in real time it is to be considered that the Convolution neural networks used are computationally expensive. Neural Networks require huge amount of training data to train deep learning model from scratch.
2) The paper Neural Network Approach for Indian Currency Recognition by Jayant kumar Nayak, Chaitan Majhi, Apurva Kumar Srivastav, Ajaya Kumar Das - 2015 uses Neural networks to recognize and then count currency notes of different denominations in a bundle for a ROI(Region of interest) . This paper had a drawback that Counterfeit currency detection was not incorporated.
3) Identification of Fake Notes and Denomination Recognition a paper by Archana M , Kalpitha C, Prajwal S, Pratiksha N. Explained how notes could be classified based on color,dimension and identification marks(mentioned by RBI)using computer vision.
4) Vipin Kumar Jain [2019] have proposed a paper Recognition of Fake currency detection done by image processing technique. This paper based on Fake Currency Identification by Android Mobile Phone Using Digital Image Processing. This is convenient use of Hand-Written Character Feature Extraction, the cell phone gadget is utilized to capture image, it is accessible wherever the image captured by android cell phone live in type of video and it is spared there, again it is changed over in gray scale design, binaries image and some morphological task performed on it, finally we found block of each character from complete images.
5) Amol A. Shirsat, S.D. Bharkat [2013] have proposed a Paper Currency Recognition System. This system mainly consists of three parts. The image of interest is first

processed and extracting the feature by applying toolbox MATLAB. The second part is currency recognition where classifier such as neural network is used. And finally, the result is displayed on pc.

6) S. S. Veling, Miss. Janhavi P. Sawal, Miss. Siddhi [2019] have proposed a paper Fake Indian Currency Recognition System by using MATLAB proposed work is to propose a currency note recognition system under hyperspectral imaging mode with different lights under different wavelengths and the comparison of features by using image processing algorithms.
7) Prof. Madhav Thigale, Lina Ladhane [2017] have proposed a paper Note to Coin converter using Digital Image Processing. A user will place the note and then with the help of image processing the Indian currency note will be identified. We are using MATLAB algorithm for detection of the value of note and we have implemented a fake note detection unit using UV LED and photodiode.
8) Deborah. Soniya Prathap [2014] has proposed a paper Detection of Fake currency using Image Processing. Choose the image and apply preprocessing. In pre-processing the image to be crop, smooth and adjust. Convert the image into gray color. After conversion apply the image segmentation. The features are extracting and reduce. Finally compare the image into original or forgery. However it had certain demerits like Lots of pre-processing steps involved and No portable application for smartphones. Hence each of these papers offer us an insight on the varying approaches and challenges.

In our paper we have analysed each of these in more detail so that we may achieve an optimal process for a real time application to do the task of identification and detection. In this paper the focus is on the comparison of the different image processing algorithms, their features and capabilities to analyse and judge which of them would be the most efficient in terms of working in a real time environment. While judging the efficiency of algorithms we take into account their accuracy, computational cost, feature matching speed, scale in variance and total image matching time. It also analyses the feasibility of the deployment of the image processing algorithm in a real time environment. This report focuses on finding the most optimal process for the task of identification of real and counterfeit currency

3. SYSTEM ANALYSIS

3.1 Existing System:

In this a methodology for counterfeit currency location extricates the general traits of latent pictures and distinguishing ID mark from the image of money. Properties from images of currency notes can include the extraction of some noticeable and undetectable highlights of Indian currency. After demonetization 500 and 2000 are the high esteemed cash notes existing till date so there is a most extreme likelihood that this notes can be duplicated so as to maintain a strategic distance from this they use programming to identify the fake notes utilizing picture handling procedure. Proposed Recognition of fake currency note using CNN and RNN. The Automatic Fake Currency Recognition System (AFCRS) is intended to identify the fake paper money to check whether it is fake or original.

3.1.1 Disadvantages of existing system:

- Time taken process.
- Requires more memory Space.
- Required hardware components.
- It requires 50 or 60 images to detect notes.
- It doesn’t find current Indian currency notes.

3.2 Proposed system:

The proposed solution provides a fake currency detection system that understands the various features off currency like watermark, shape, alignment, picture of Mahatma Gandhi, a translucent feel and a security thread. The objective of system is to provide a better detection based on recorded information of currency. These systems use deep learning technique Convolution Neural Network (CNN) and Recurrent Neural Network (RNN) to process information and provide the user with potentially more relevant prediction. We will be building a convolutional neural network according to proposed algorithm which will be trained on the given fake and original currency data set, and later be able to predict whether the given currency image is fake or original. In this AFCRS we will be solving an image classification problem, where our goal will be to tell which class the input image belongs to.

The way we are going to achieve it is by training an artificial neural network on image data set of currency and make the RNN (Recurrent Neural Network) to predict which class the image belongs to, when it sees an image having fake note or original note the next time. Convolutional neural networks (CNN’s) are nowadays widely used in pattern-recognition and image-recognition problems. They have many advantages compared to other techniques. Typically, Convolution neural networks use approximately 5 to 25 distinct layers of pattern recognition. They take raw data, without the need for an initial separate pre-processing or feature extraction stage: in a CNN, the feature extraction and classification occur naturally within a single framework. This is a major advantage when compared to other image processing techniques, while they need lot of computations only for pre-processing step.

3.2.1 Advantages of proposed system:

- Less time taken.
- Doesn’t require hardware Machines.
- It requires 15 or 20 images to detect notes.
- It takes less memory space compared to existing system.

3.3 System Architecture:

System 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. The purpose of system architecture activities is to define a comprehensive solution based on principles, concepts, and properties logically related to and consistent with each other. The solution architecture has features, properties, and characteristics which satisfy, as far as possible, the problem or opportunity expressed by a set of system requirements.

System Architecture is abstract, conceptualization-oriented, global, and focused to achieve the mission and life cycle concepts of the system. It also focuses on high-level structure in systems and system elements. It addresses the architectural principles, concepts, properties, and characteristics of the system-of-interest. It may also be applied to more than one system, in some cases forming the common structure, pattern, and set of requirements for classes or families of similar or related systems.

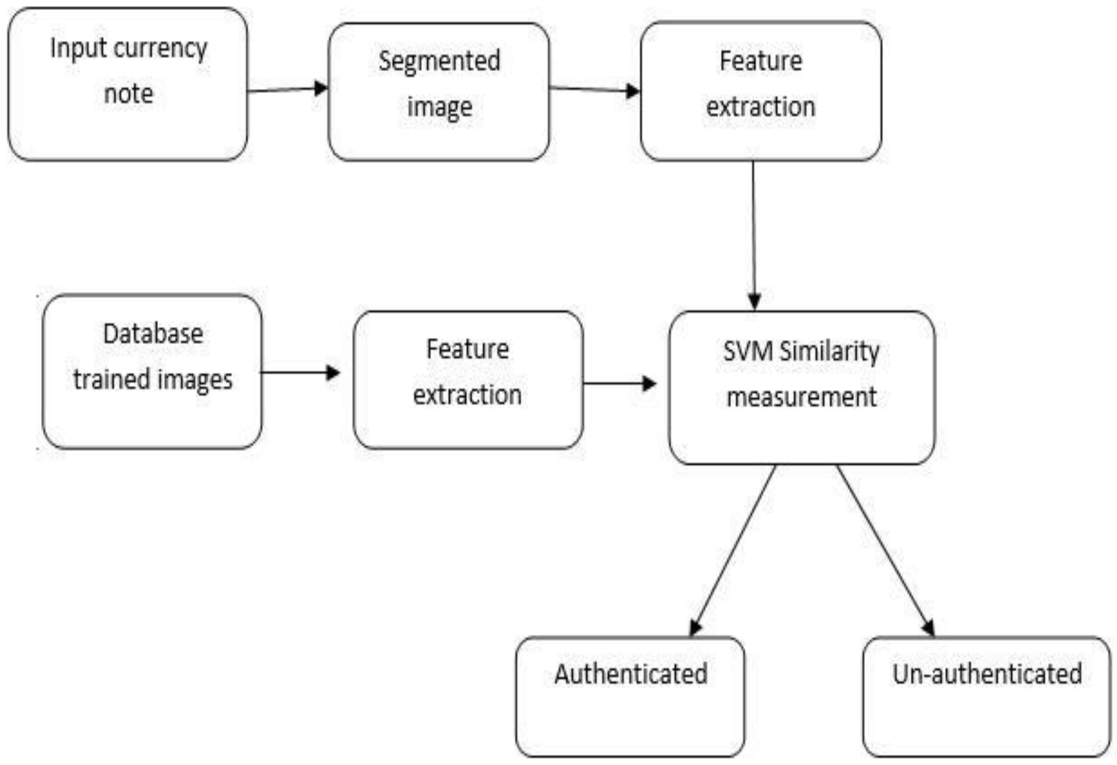


Fig 3.1: System Architecture

3.3.1 Input Currency note:

The image we get from scanner is formatted by JPEG. JPEG (Joint Photographic Experts Group) is a standard for destructive or loss compromising for digital images. When you save the image as JPEG, the image will lose some information, and this cannot be recovered.

3.3.2 Image Segmentation:

Image segmentation is the task of clustering parts of an image together that belong to the same object class. This process is also called pixel-level classification. In other words, it involves partitioning images (or video frames) into multiple segments or objects. New segment of image segmentation models with remarkable performance improvements. Deep Learning based image segmentation models often achieve the best accuracy rates on popular benchmarks, resulting in a paradigm shift in the field.

3.3.3 Feature Extraction:

Feature extraction for machine learning and deep learning. Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data. A type of dimensionality reduction where a large number of pixels of the image are efficiently represented in such a way that interesting parts of the image are captured effectively.

3.3.4 Database Trained Images:

An image database system organizes digital pictures into a central location for fast sharing and irretrievability. It is the storage element of digital asset management (DAM), giving DAM users extensive features. These features include tools that allow users to upload, search and share company graphics.

3.3.5 SVM (Support Vector Machine) Similarity:

"Support Vector Machine" (SVM) is a supervised machine learning that can be used for either classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

3.3.6 Authentication:

Authentication is the process of determining whether someone or something is, in fact, who or what it says it is. Authentication technology provides access control for systems by checking to see if a user's credentials match the credentials in a database of authorized users or in a data authentication server.

3.3.7 Un-authentication:

When we give a user or process the opportunity to interact with our database without supplying a set of credentials, we create the possibility for security issues.The process of determining whether someone or something is, in fact, who or what it says it is un-authentication.

4. SYSTEM REQUIREMENTS

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well-ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers. All computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements. A second meaning of the term of system requirements is a generalisation of this first definition, giving the requirements to be met in the design of a system or sub-system.

Software Requirements:

- Python
- TensorFlow
- Keras

Operating Systems Supported:

- Windows 10
- Linux
- Macos

Technologies and Languages used to develop:

- Python
- Machine Learning Algorithms
- Deep Learning Algorithms

4.1 System Specification:

4.1.1 Hardware Requirements:

Processor	**: i3 or i5**
RAM	**: 4GB**
Storage	**: 500GB HDD**
Web camera	

4.1.2 Software Requirements:

Operating System	**: Windows / Linux / MacOS**
Python 3+	**: Programming language**
Keras 2.0+	**: Library for neural network which uses TensorFlow**
TensorFlow 2.0+	**: Deep learning library in python**
NumPy 1.18.2	**: NumPy is the fundamental package for scientific computing**

5. SOFTWARE DESCRIPTION

5.1 PYTHON:

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management.

Interactive Mode Programming:

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

```
$ python

Python 2.4.3 (#1, Nov 11 2010, 13:34:43)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>>
```

Type the following text at the Python prompt and press the Enter −

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");.However in Python version 2.4.3, this produces the following result –Hello, Python!

Script Mode Programming:

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

```
$ python test.py
```

This produces the following result −

```
Hello, Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo

```
#!/usr/bin/python

print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

```
$ chmod +x test.py     # This is to make file executable

$./test.py
```

This produces the following result −

```
Hello, Python!
```

Python Identifiers:

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

Reserved Words:

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and	Exec	not
assert	Finally	or
break	For	pass
class	From	print
continue	Global	raise
def	If	return
del	Import	try
elif	In	while

else	Is	with
except	Lambda	yield

Lines and Indentation:

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

```
if True:
print "True"
else:
  print "False"
```

However, the following block generates an error −

```
if True:
print "Answer"
print "True"
else:
print "Answer"
print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces. #!/usr/bin/python

```
import sys
try:
    # open file stream
    file = open(file_name, "w")
except IOError:
    print "There was an error writing to", file_name
    sys.exit()
print "Enter '", file_finish,
print "' When finished"
while file_text != file_finish:
    file_text = raw_input("Enter text: ")
    if file_text == file_finish:
        # close the file
        file.close
        break
    file.write(file_text)
    file.write("\n")
file.close()
file_name = raw_input("Enter filename: ")
if len(file_name) == 0:
    print "Next time please enter something"
    sys.exit()
try:
    file = open(file_name, "r")
except IOError:
    print "There was an error reading file"
```

```
   sys.exit()
file_text = file.read()
file.close()
print file_text
```

5.1.6 Multi-Line Statements:

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For **example −**

```
total = item_one + \
        item_two + \
        item_three
```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

```
days = ['Monday', 'Tuesday', 'Wednesday',
```

5.1.7 Command Line Arguments:

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python -h
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
-c cmd : program passed in as string (terminates option list)
-d     : debug output from parser (also PYTHONDEBUG=x)
-E     : ignore environment variables (such as PYTHONPATH)
-h     : print this help message and exit
```

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

5.1.8 Python Lists:

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5];

list3 = ["a", "b", "c", "d"]

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5);

tup3 = "a", "b", "c", "d";

The empty tuple is written as two parentheses containing nothing −

tup1 = ();

To write a tuple containing a single value you have to include a comma, even though there is only one value −

tup1 = (50,);

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

Live Demo

```
#!/usr/bin/python

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5, 6, 7 );

print "tup1[0]: ", tup1[0];

print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result −

```
tup1[0]:  physics

tup2[1:5]:  [2, 3, 4, 5]
```

Updating Tuples

Accessing Values in Dictionary

Note − del() method is discussed in subsequent section.

Properties of Dictionary Keys:

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys −

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}

print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Manni
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example –

Live Demo

```
#!/usr/bin/python
dict = {['Name']: 'Zara', 'Age': 7}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
Traceback (most recent call last):
   File "test.py", line 3, in <module>
      dict = {['Name']: 'Zara', 'Age': 7};
TypeError: unhashable type: 'list'
```

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates −

Live Demo

```
#!/usr/bin/python
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');
# Following action is not valid for tuples
# tup1[0] = 100;
# So let's create a new tuple as follows
tup3 = tup1 + tup2;
print tup3;
```

When the above code is executed, it produces the following result −

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example −

Live Demo

```
#!/usr/bin/python

tup = ('physics', 'chemistry', 1997, 2000);

print tup;

del tup;

print "After deleting tup : ";

print tup;
```

This produces the following result. Note an exception raised, this is because after del tup tuple does not exist anymore −

```
('physics', 'chemistry', 1997, 2000)

After deleting tup :

Traceback (most recent call last):

  File "test.py", line 9, in <module>

    print tup;

NameError: name 'tup' is not defined
```

What Is A Script?

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode.

Scripts are reusable:

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable:

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor:

Just about any text editor will suffice for creating Python script files. You can use Microsoft Notepad, Microsoft WordPad, Microsoft Word.

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, where as the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions. The same program in its human-readable source code form, from which executable programs are derived.

What is Python ?

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Some of the libraries used in Python:

1**. Requests** - The most famous http library written by kennethreitz. It's a must have for every python developer.

2. Scrapy - If you are involved in webscraping then this is a must have library for you. After using this library you won't use any other.

3. wxPython - A gui toolkit for python. I have primarily used it in place of tkinter. You will really love it.

4. Pillow - A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

5. SQLAlchemy - A database library. Many love it and many hate it. The choice is yours.

6. BeautifulSoup - I know it's slow but this xml and html parsing library is very useful for beginners.

7. Twisted - The most important tool for any network application developer. It has a very beautiful api and is used by a lot of famous python developers.

8. NumPy - How can we leave this very important library ? It provides some advance math functionalities to python.

9. SciPy - When we talk about NumPy then we have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python. 10. matplotlib - A numerical plotting library. It is very useful for any data scientist or any data analyzer.

11. Pygame - Which developer does not like to play games and develop them ? This library will help you achieve your goal of 2d game development.

12. Pyglet - A 3d animation and game creation engine. This is the engine in which the famous python port of minecraft was made

13. pyQT - A GUI toolkit for python. It is my second choice after wxpython for developing GUI's for my python scripts.

14. pyGtk - Another python GUI library. It is the same library in which the famous Bittorrent client is created.

15. Scapy - A packet sniffer and analyzer for python made in python.

16. pywin32 - A python library which provides some useful methods and classes for interacting with windows.

17. nltk - Natural Language Toolkit – I realize most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings. But it's capacity is beyond that. 18. nose - A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. SymPy - SymPy can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

5.2 Teachable Machine:

Teachable Machine is a web-based tool that makes creating machine learning models fast, easy, and accessible to everyone. Educators, artists, students, innovators, makers of all kinds – really, anyone who has an idea they want to explore. No prerequisite machine learning knowledge required. You train a computer to recognize your images, sounds, and poses without writing any machine learning code. Then, use your model in your own projects, sites, apps, and more. We can export your model as a TensorFlow.js model and host it on Teachable Machine for free, so you can call it into any website or app. You can also convert it to TensorFlow and TensorFlow Lite and download it for local use.

Models made with Teachable Machine are TensorFlow.js models, so you can export them and use them anywhere Javascript runs — like on a website, on a server, or in some prototyping apps like Framer. Additionally, you can export your model to some other formats to use them in other platforms and environments. Learn more about exporting and using your model in the Teachable Machine Community.

5.3 TensorFlow:

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019. TensorFlow can be used in a wide variety of programming languages, most notably Python, as well as Javascript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors. TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017.While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

5.3.1 TensorFlow Lite:

In May 2017, Google announced a software stack specifically for mobile development, TensorFlow Lite. In January 2019, TensorFlow team released a developer preview of the mobile GPU inference engine with OpenGL ES 3.1 Compute Shaders on Android devices and Metal Compute Shaders on iOS devices.] In May 2019, Google announced that their TensorFlow Lite Micro (also known as TensorFlow Lite for Microcontrollers) and ARM's uTensor would be merging.

5.3.2 TensorFlow 2.0:

As TensorFlow's market share among research papers was declining to the advantage of PyTorch, the TensorFlow Team announced a release of a new major version of the library in September 2019. TensorFlow 2.0 introduced many changes, the most significant being TensorFlow eager, which changed the automatic differentiation scheme from the static computational graph, to the "Define-by-Run" scheme originally made popular by Chainer and later PyTorch. Other major changes included removal of old libraries, cross-compatibility between trained models on different versions of TensorFlow, and significant improvements to the performance on GPU.

5.4 Keras:

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML. As of version 2.4, only TensorFlow is supported. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet is also the author of the XCeption deep neural network model.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling. Keras allows users to productize deep models on smart phones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU).

5.5 Machine Learning:

Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behaviour. Artificial intelligence systems are used to perform complex tasks in a way that is similar to how humans solve problems. The goal of AI is to create computer models that exhibit “intelligent behaviours” like humans, according to Boris Katz, a principal research scientist and head of the InfoLab Group at CSAIL. This means machines that can recognize a visual scene, understand a text written in natural language, or perform an action in the physical world.

Machine learning is one way to use AI. It was defined in the 1950s by AI pioneer Arthur Samuel as “the field of study that gives computers the ability to learn without explicitly being programmed.”The definition holds true, according to Mikey Shulman, a lecturer at MIT Sloan and head of machine learning at Kensho, which specializes in artificial intelligence for the finance and U.S. intelligence communities. He compared the traditional way of programming computers, or “software 1.0,” to baking, where a recipe calls for precise amounts of ingredients and tells the baker to mix for an exact amount of time. Traditional programming similarly requires creating detailed instructions for the computer to follow.

Some cases, writing a program for the machine to follow is time-consuming or impossible, such as training a computer to recognize pictures of different people. While humans can do this task easily, it’s difficult to tell a computer how to do it. Machine learning takes the approach of letting computers learn to program themselves through experience.

5.6 Deep Learning:

Deep learning is a machine learning technique. It teaches a computer to filter inputs through layers to learn how to predict and classify information. Observations can be in the form of images, text, or sound. The inspiration for deep learning is the way that the human brain filters information. Its purpose is to mimic how the human brain works to create some real magic. In the human brain, there are about 100 billion neurons. Each neuron connects to about 100,000 of its neighbours. We’re kind of recreating that, but in a way and at a level that works for machines. In our brains, a neuron has a body, dendrites, and an axon. The signal from one neuron travels down the axon and transfers to the dendrites of the next neuron. That connection where the signal passes is called a synapse. Neurons by themselves are kind of useless. But when you have lots of them, they work together to create some serious magic. That’s the idea behind a deep learning algorithm! You get input from observation and you put your input into one layer. That layer creates an output which in turn becomes the input for the next layer, and so on. This happens over and over until your final output signal! The neuron (node) gets a signal or signals (input values), which pass through the neuron. That neuron delivers the output signal.

Think of the input layer as your senses: the things you see, smell, and feel, for example. These are independent variables for one single observation. This information is broken down into numbers and the bits of binary data that a computer can use. You'll need to either standardize or normalize these variables so that they're within the same range. They use many layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output of the previous layer for its input. What they learn forms a hierarchy of concepts. In this hierarchy, each level learns to transform its input data into a more and more abstract and composite representation. That means that for an image, for example, the input might be a matrix of pixels. The first layer might encode the edges and compose the pixels. The next layer might compose an arrangement of edges. The next layer might encode a nose and eyes. The next layer might recognize that the image contains a face, and so on.

What happens inside the neuron?

The input node takes in information in a numerical form. The information is presented as an activation value where each node is given a number. The higher the number, the greater the activation. Based on the connection strength (weights) and transfer function, the activation value passes to the next node. Each of the nodes sums the activation values that it receives (it calculates the weighted sum) and modifies that sum based on its transfer function. Next, it applies an activation function. An activation function is a function that's applied to this particular neuron. From that, the neuron understands if it needs to pass along a signal or not.

Each of the synapses gets assigned weights, which are crucial to Artificial Neural Networks (ANNs). Weights are how ANNs learn. By adjusting the weights, the ANN decides to what extent signals get passed along. When you're training your network, you're deciding how the weights are adjusted.

The activation runs through the network until it reaches the output nodes. The output nodes then give us the information in a way that we can understand. Your network will use a cost function to compare the output and the actual expected output. The model performance is evaluated by the cost function. It's expressed as the difference between the actual value and the predicted value.

There are many different cost functions you can use, you're looking at what the error you have in your network is. You're working to minimize loss function. (In essence, the lower the loss functions, the closer it is to your desired output). The information goes back, and the neural network begins to learn with the goal of minimizing the cost function by tweaking the weights. This process is called back propagation.

In forward propagation, information is entered into the input layer and propagates forward through the network to get our output values. We compare the values to our expected results. Next, we calculate the errors and propagate the info backward. This allows us to train the network and update the weights. (Back propagation allows us to adjust all the weights simultaneously). During this process, because of the way the algorithm is structured, you're able to adjust all of the weights simultaneously. This allows you to see which part of the error each of your weights in the neural network is responsible for.

5.7 CONVOLUTIONAL NEURAL NETWORKS:

Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity.

Convolutional Neural Networks (CNNs) are analogous to traditional ANNs in that they are comprised of neurons that self-optimise through learning. Each neuron will still receive an input and perform a operation (such as a scalar product followed by a non-linear function) - the basis of countless ANNs. From the input raw image vectors to the final output of the class score, the entire of the network will still express a single perceptive score function (the weight). The last layer will contain loss functions associated with the classes, and all of the regular tips and tricks developed for traditional ANNs still apply.

The only notable difference between CNNs and traditional ANNs is that CNNs are primarily used in the field of pattern recognition within images. This allows us to encode image-specific features into the architecture, making the network more suited for image-focused tasks - whilst further reducing the parameters required to set up the model. One of the largest limitations of traditional forms of ANN is that they tend to struggle with the computational complexity required to compute image data. Common machine learning benchmarking datasets such as the MNIST database of handwritten digits are suitable for most forms of ANN, due to its relatively small image dimensionality of just 28×28. With this dataset a single neuron in the first hidden layer will contain 784 weights (28×28×1 where 1 bare in mind that MNIST is normalised to just black and white values), which is manageable for most forms of ANN. If you consider a more substantial coloured image input of 64 × 64, the number of weights on just a single neuron of the first layer increases substantially to 12, 288. Also take into account that to deal with this scale of input, the network will also need to be a lot larger than one used to classify colour-normalised MNIST digits, then you will understand the drawbacks of using such models.

5.7.1 CNN ARCHITECTURE:

Deep Learning is a subset of Machine Learning which consists of algorithms that are inspired by the functioning of the human brain or the neural networks. These structures are called as Neural Networks. It teaches the computer to do what naturally comes to humans. Deep learning, there are several types of models such as the Artificial Neural Networks (ANN), Auto encoders, Recurrent Neural Networks (RNN) and Reinforcement Learning. But there has been one particular model that has contributed a lot in the field of computer vision and image analysis which is the Convolutional Neural Networks (CNN) or the ConvNets.

CNNs are a class of Deep Neural Networks that can recognize and classify particular features from images and are widely used for analyzing visual images. Their applications range from image and video recognition, image classification, medical image analysis, computer vision and natural language processing.

The term ‘Convolution” in CNN denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other. In simple terms, two images which can be represented as matrices are multiplied to give an output that is used to extract features from the image.

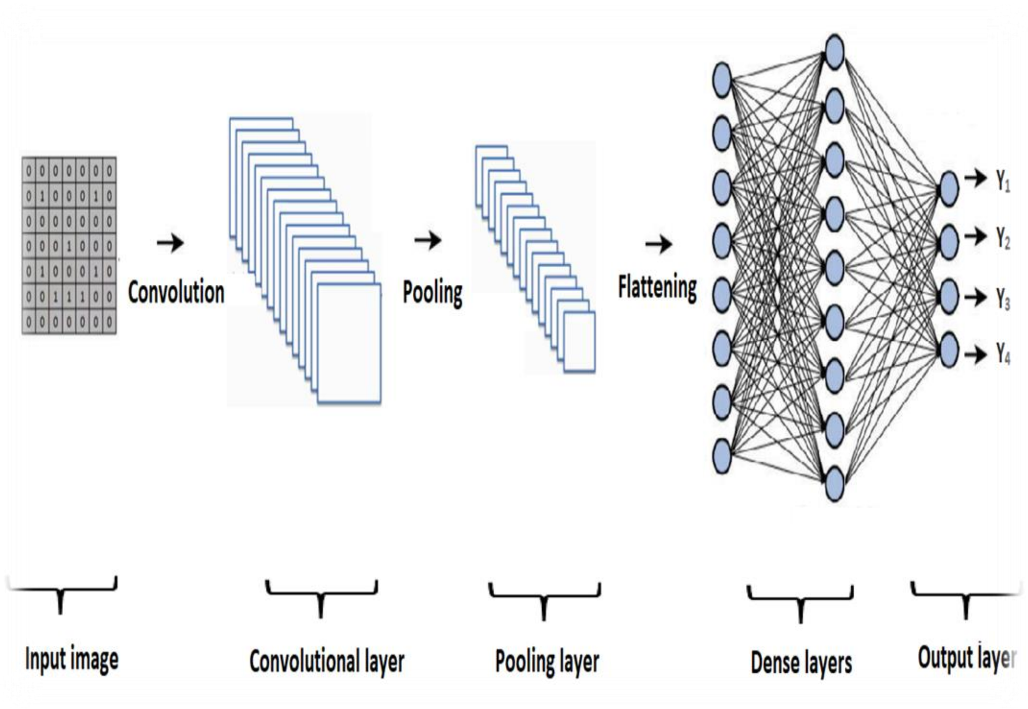


Fig 5.1: CNN Architecture

1. Convolutional Layer:

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

2. Pooling Layer:

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations.

In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer.

3. Flattening Layer:

Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector. The flattened matrix is fed as input to the fully connected layer to classify the image.

5.8 RECURRENT NEURAL NETWORK:

Recurrent Neural Network (RNN) is a type of Neural Network where the output from previous steps is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.

RNN have a "memory" which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

RNN converts the independent activations into dependent activations by providing the same weights and biases to all the layers, thus reducing the complexity of increasing parameters and memorizing each previous outputs by giving each output as input to the next hidden layer. Hence these three layers can be joined together such that the weights and bias of all the hidden layers is the same, into a single recurrent layer.

5.8.1 RNN ARCHITECTURE:

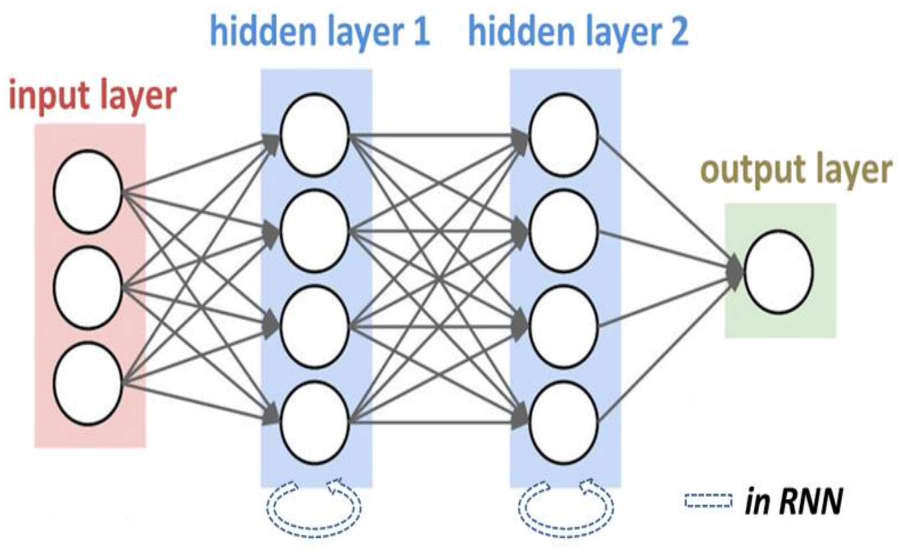


Fig 5.2: RNN Architecture

A Recurrent Neural Network is a type of neural network that contains loops, allowing information to be stored within the network. In short, Recurrent Neural Networks use their reasoning from previous experiences to inform the upcoming events. Recurrent models are valuable in their ability to sequence vectors, which opens up the API to performing more complicated tasks.

Recurrent Neural Networks can be thought of as a series of networks linked together. They often have a chain-like architecture, making them applicable for tasks such as speech recognition, language translation, etc. An RNN can be designed to operate across sequences of vectors in the input, output, or both. For example, a sequenced input may take a sentence as an input and output a positive or negative sentiment value. Alternatively, a sequenced output may take an image as an input, and produce a sentence as an output.

6. SYSTEM DESIGN

6.1 UML DIAGRAMS

6.1.1 Dataflow Diagram:

A data-flow diagram is a way of representing a flow of data through a process or a system of an information system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow there are no decision rules and no loops.

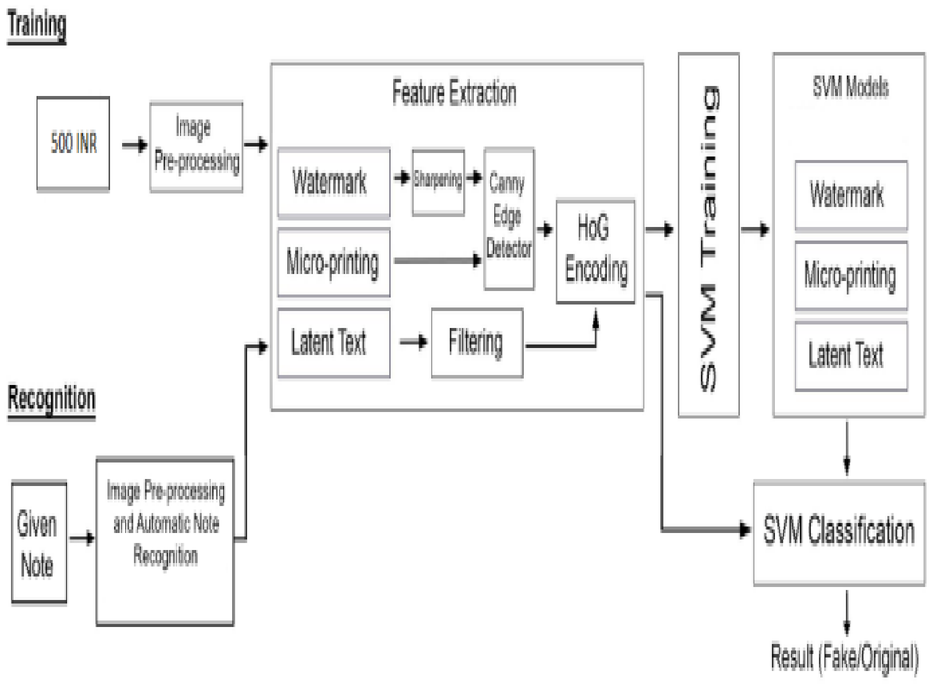


Fig 6.1: Dataflow Diagram

A. Security Thread:

It is a 3mm windowed security thread with inscriptions of India in Hindi, RBI and 2000/500 on banknotes with colour shift. Colour of the thread changes from green to blue when the note is tilted.

B. Serial Number:

Serial number panel with banknote number growing from small to big on the top left side and bottom right side.

C. Latent image:

A vertical band on front side of denomination at right hand size. It contains latent image showing numeral of denomination when banknote is held horizontally at eye level.

D. Watermark:

The portrait of Mahatma Gandhi, and multidirectional lines and a mark showing the denominational numeral appear which can be viewed when held against light.

E. Identification Mark:

A mark with intaglio print which can be felt by touch helps blind person to identify the denomination. In 500 denominations the mark is of five lines while in 2000 line the mark is of seven lines. For 100 and 200 denominations the mark if of three lines.

F. Micro lettering:

This feature appears between the vertical band and Mahatma Gandhi portrait. It contains the word RBI in Rs.10. The notes of Rs .20 and above also contain the denominational value of the notes in micro letters. This feature can be seen well under a magnifying glass.

6.1.2 Use Case Diagram:

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

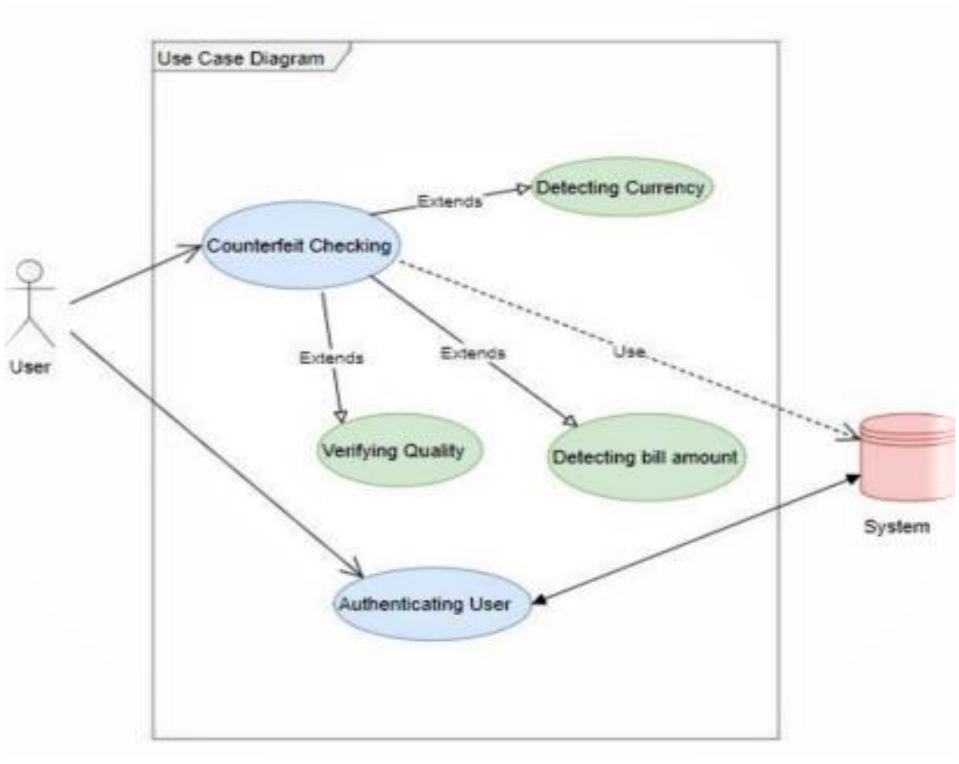


Fig 6.2: Use Case Diagram

Different components have been built to perform the verification of the currency note. Shows the basic structure of the Deep Money architecture, its actors, and the way they interact with each other. There is a single actor here, which is shown as the user. The user can input an image and request the authentication of the currency note. The system will respond through various other functions, as shown. External systems may be used to receive assistance from the sensors if necessary. As Deep Money progresses, other more appropriate features may be found and the use of these initial features may be minimized and other more versatile features may evolve. In this case, the 'input image' will remain the same but the dependent functionalities may change.

6.1.3 Class Diagram:

Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide. Class diagrams are useful in many stages of system design.

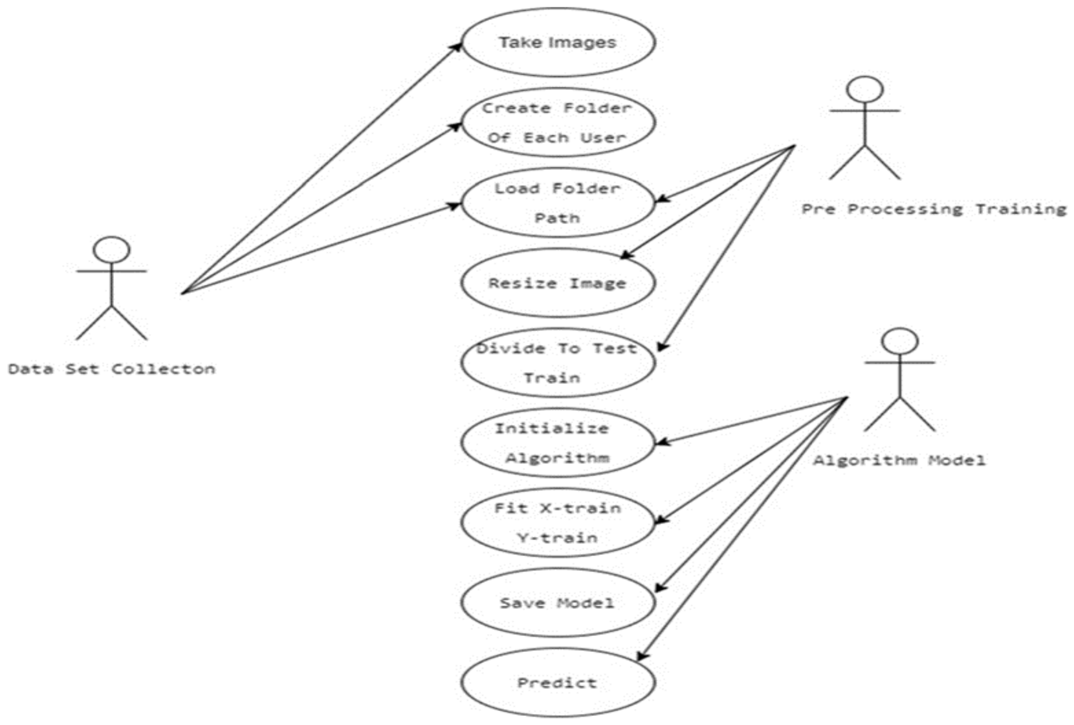


Fig 6.3: Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

6.1.4 Sequence Diagram:

A sequence diagram is a type of interaction diagram because it describes how—and in what order a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.

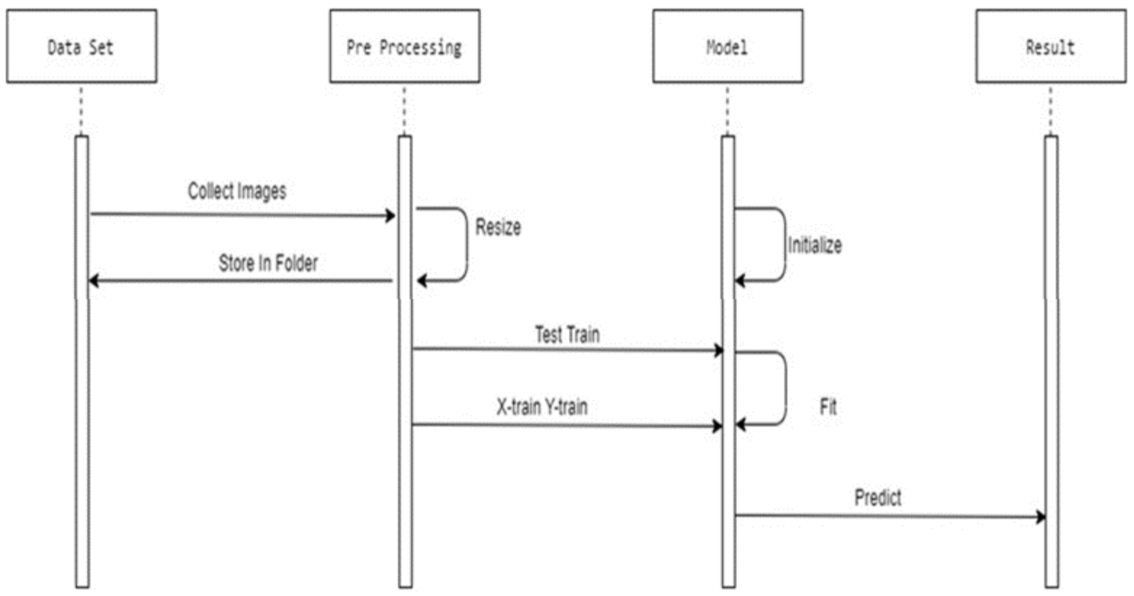


Fig 6.4: Sequence Diagram

7. IMPLEMENTATION

7.1 Dataset Collection:

Currency images data set can be collected manually while training data of dataset and store in each class. Minimum 15 to 20 images are collected of each and every denomination of 10, 20, 50, 100, 200, 500 and 2000 respectively. Store the images individually in each and every class by naming Real and Fake to classes to train a model.

7.1.1 Encoding Dataset:

In this stage face recognition, keras, tensorflow, sklearn libraries are initialized and each image is looped from the folder using batching and multiprocessing.

- Load the input image and convert It from BGR.
- Detect the (x,y)-coordinates of the bounding boxes.
- Corresponding to each face in the input image compute the facial embedding for the face.

7.2 Process of Implementation:

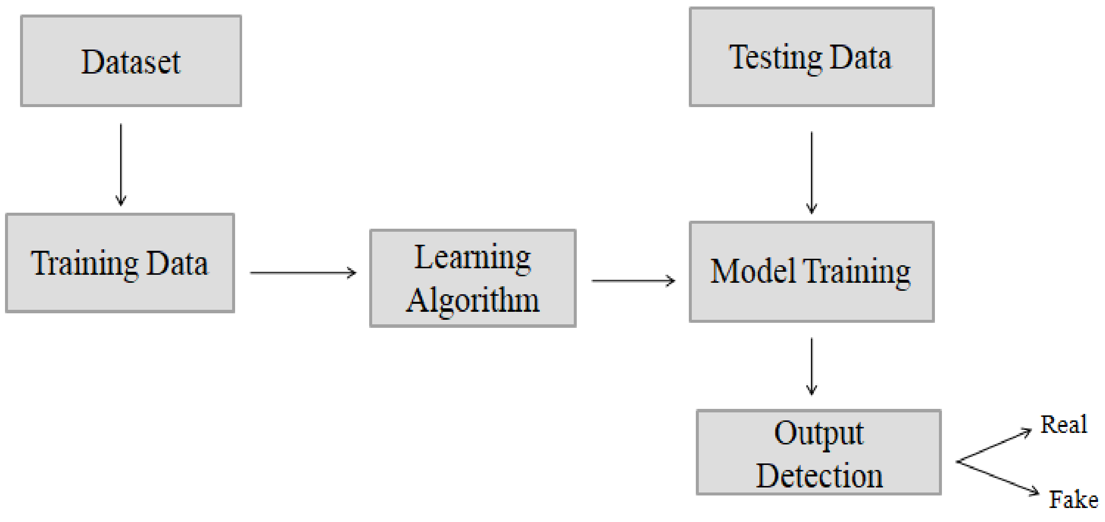


Fig 7.1: Process Of Implementation

7.2.1 Dataset:

New Indian currency denomination dataset is not available online so a new dataset is created using a 21MP camera. The size of all the currency images in the dataset which are captured in landscape mode is 5344×3006. The size of all the currency images in the dataset which are captured in portrait. A total of 150 images are captured to create the dataset. All the currency notes which are acceptable in the market are used like old and new 10 rupee notes, old 20 rupee notes, old and new 50 rupee notes, old and new 100 rupee notes, new 200, 500, and 2000 notes. To improve the dataset size data augmentation is applied to the currency note images. The different types of augmentations used are Zoom, Rotate 90, Rotate 270, Tilt, Distortion, and Flip. Dataset after data augmentation contains a total of 500 images.

7.2.2 Training Data:

The data will be trained through the dataset collection of a dataset which has been collected of each and every image of Indian rupees to detect whether fake or real note for Indian currency. The data will be trained according to denominations of Indian currency are training individual through the classes of images.

7.2.3 Learning Algorithms:

Here learning algorithms are CNN (Convolutional Neural Network) and RNN (Recurrent Neural Network) are used for training model, which are present in tensorflow and keras to train the model. Both CNN and RNN algorithms are used to reduce complexity and also increase accuracy to process flow.

7.2.4 Testing Data:

Data will be tested by checking each and every process before training the model of dataset. The data will be tested through the verify the process currency prediction of dataset by extracting various features and argumentation process and passes through the process of model training.

7.2.5 Model Training:

Model training process will be done after the process of testing dataset and data collection. Model will be trained by exporting images of Indian currency and training model through tensorflow and keras by using python language.

7.2.6 Output Detection:

Output will be detected whether real or fake, empty after training a model. When any Indian currency note is placed near camera then it will extract all features of currency notes. Then the note will be detected and counted by trained model of dataset collection whether real or fake.

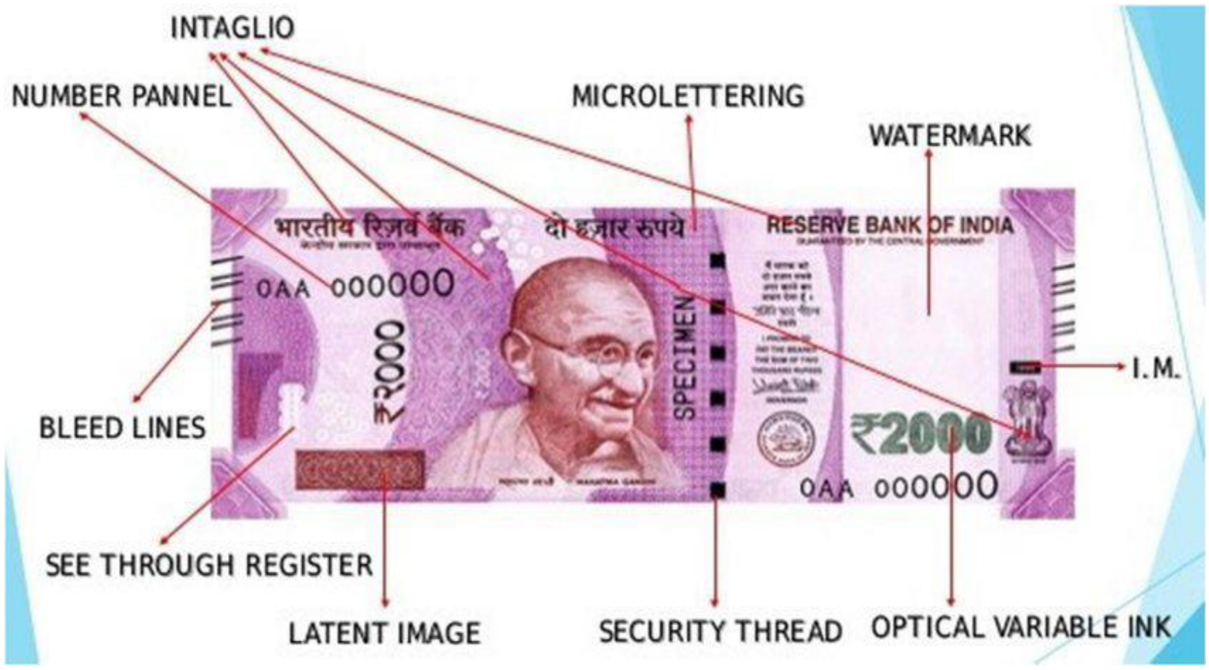


Fig 7.2: Features of Indian Currency note

7.3 Flow Chart for Process:

Process flows or workflows refer to a series of sequential tasks that are performed to achieve a certain goal. Each task in the process flow is governed by input, transformation, and output. A process flow represents the order in which tasks need to be executed in order to achieve goals. Process flow is a sequential representation of a process and its components, including operations, timelines, and people involved, and resources needed. The main objective of process flows is to help you standardize and optimize your processes and help your team better understand how your business works.

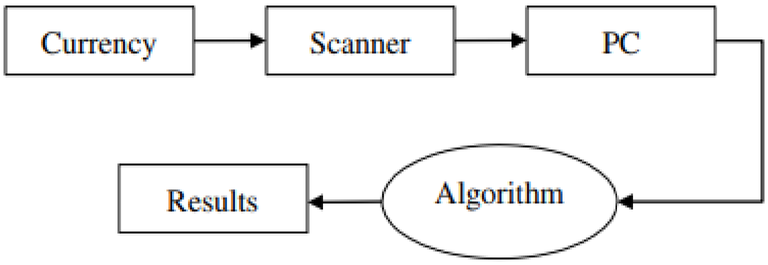


Fig 7.3: Flow chart process for Implementation

The system is based on scanner, PC, and algorithm. The aid of the algorithm is located in the unique figure, RGB to Gray, image binarization, noise elimination, segmentation, pattern matching, etc. We realize there by programming with Python. Flowchart of the system is described in Figure 7.3, Flowchart of the process. The system contains a complete user interface, the user just needs to open the image, and the result will be shown after processing. The most important part of this system is pattern matching. Based on edge detection algorithm, the system performs pattern matching. When some error occurs, the system will emerge some exception, which may cause the exceptions like "the image not complete", "failed recognition", etc.

7.4 System Lunch:

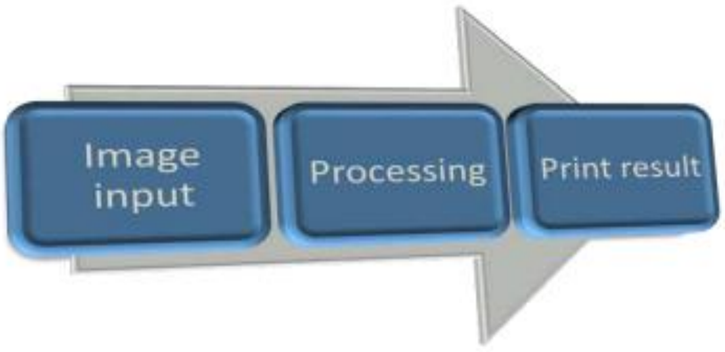


Fig 7.4: Flow Chart of System Lunch

7.4.1 User interface:

The user interface is for printing out the original image and special features.

7.4.2 Read in image:

The system can read not only JPEG (JPG) format but others. Our image was obtained from a scanner. As mentioned before, the resolution is set to 600 DPI. But this will make the image a big size. So after reading in the image, the system will reset the image to size 1024 by 768 pixels and this work will refer to image pre-processing.

7.4.3 Image pre-processing:

The aim of image pre-processing is to suppress undesired distortions or enhance some image features that are important for further processing or analysis. In our work, image pre-processing includes these parts: Image adjusting Image smoothening (removing noise). When we get the image from a scanner, the size of the image is so big. In order to reduce the calculation, we decrease the size to 1024×768 pixels. When using a digital camera or a scanner and perform image transfers, some noise will appear on the image. Image noise is the random variation of brightness in images.

Removing the noise is an important step when image processing is being performed. However noise may affect segmentation and pattern matching. shows the image polluted by noise. When performing smoothing process on a pixel, the neighbour of the pixel is used to do some transforming. After that a new value of the pixel is created. The neighbour of the pixel is consisting with some other pixels and they build up a matrix, the size of the matrix is odd number, the target pixel is located on the middle of the matrix.

7.5 Source Code:

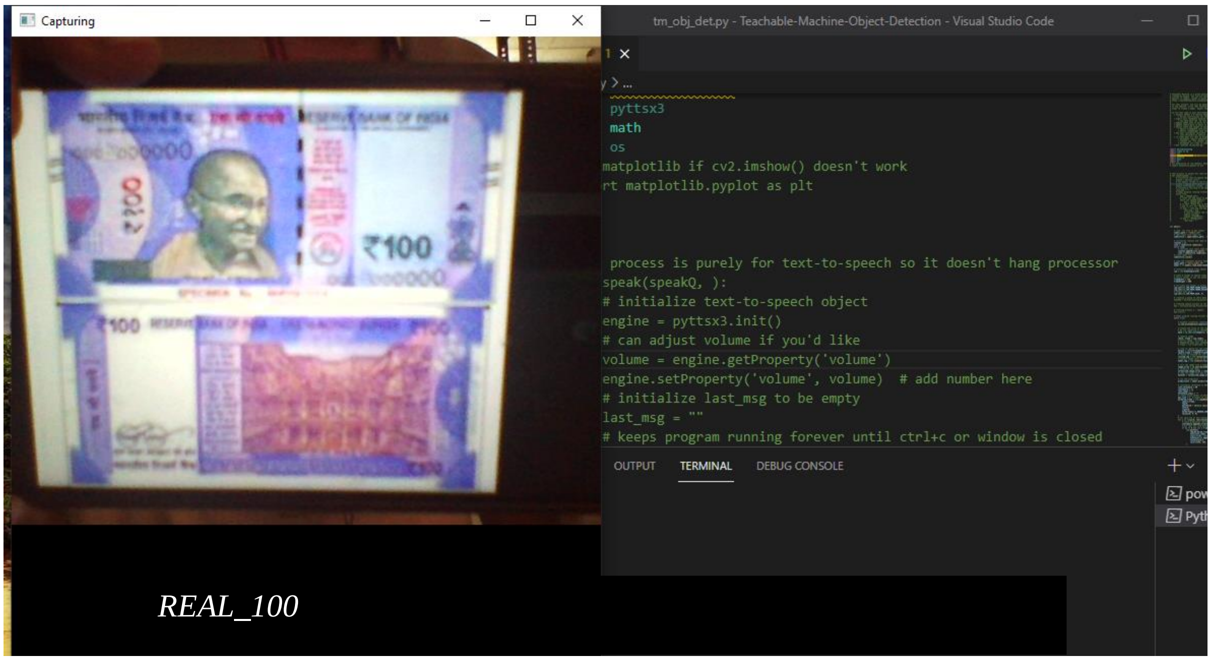
```
import multiprocessing

import numpy as np

import cv2

import tensorflow.keras as tf

import pyttsx3

import math

import os

def main():


    # read .txt file to get labels

    labels_path = "labels.txt"

    # open input file label.txt

    labelsfile = open(labels_path, 'r')


    # initialize classes and read in lines until there are no more

    classes = []

    line = labelsfile.readline()

    while line:

        # retrieve just class name and append to classes

        classes.append(line.split(' ', 1)[1].rstrip())

        line = labelsfile.readline()

    # close label file

    labelsfile.close()


    # load the teachable machine model

    model_path = "keras_model.h5"

    model = tf.models.load_model(model_path, compile=False)
```

```
# initialize webcam video object
cap = cv2.VideoCapture(0)

# width & height of webcam video in pixels -> adjust to your size
# adjust values if you see black bars on the sides of capture window
frameWidth = 720
frameHeight = 600

# set width and height in pixels
cap.set(cv2.CAP_PROP_FRAME_WIDTH, frameWidth)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, frameHeight)
# enable auto gain
cap.set(cv2.CAP_PROP_GAIN, 0)

# creating a queue to share data to speech process
#speakQ = multiprocessing.Queue()

# creating speech process to not hang processor
#p1 = multiprocessing.Process(target=speak, args=(speakQ, ), daemon="True")

# starting process 1 - speech
#p1.start()

# keeps program running forever until ctrl+c or window is closed
while True:

    # disable scientific notation for clarity
    np.set_printoptions(suppress=True)
```

```
# Create the array of the right shape to feed into the keras model.
# We are inputting 1x 224x224 pixel RGB image.
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

# capture image
check, frame = cap.read()
# mirror image - mirrored by default in Teachable Machine
# depending upon your computer/webcam, you may have to flip the video
# frame = cv2.flip(frame, 1)

# crop to square for use with TM model
margin = int(((frameWidth-frameHeight)/2))
square_frame = frame[0:frameHeight, margin:margin + frameHeight]
# resize to 224x224 for use with TM model
resized_img = cv2.resize(square_frame, (224, 224))
# convert image color to go to model
model_img = cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB)

# turn the image into a numpy array
image_array = np.asarray(model_img)
# normalize the image
normalized_image_array = (image_array.astype(np.float32) / 127.0) - 1
# load the image into the array
data[0] = normalized_image_array

# run the prediction
predictions = model.predict(data)

# confidence threshold is 90%.
conf_threshold = 90
```

```
confidence = []
conf_label = ""
threshold_class = ""
# create blach border at bottom for labels
per_line = 2  # number of classes per line of text
bordered_frame = cv2.copyMakeBorder(
    square_frame,
    top=0,
    bottom=30 + 15*math.ceil(len(classes)/per_line),
    left=0,
    right=0,
    borderType=cv2.BORDER_CONSTANT,
    value=[0, 0, 0]
)
# for each one of the classes
for i in range(0, len(classes)):
    # scale prediction confidence to % and apppend to 1-D list
    confidence.append(int(predictions[0][i]*100))
    # put text per line based on number of classes per line
    if (i != 0 and not i % per_line):
        cv2.putText(
            img=bordered_frame,
            text=conf_label,
            org=(int(0), int(frameHeight+25+15*math.ceil(i/per_line))),
            fontFace=cv2.FONT_HERSHEY_SIMPLEX,
            fontScale=0.5,
            color=(255, 255, 255)
        )
        conf_label = ""
    # appendclasses and confidences to text for label
    conf_label += classes[i] + ": " + str(confidence[i]) + "%; "
    # prints last line
    if (i == (len(classes)-1)):
        cv2.putText(
            img=bordered_frame,
            text=conf_label,
            org=(int(0), int(frameHeight+25+15*math.ceil((i+1)/per_line))),
            fontFace=cv2.FONT_HERSHEY_SIMPLEX,
            fontScale=0.5,
            color=(255, 255, 255)
        )
        conf_label = ""
    # if above confidence threshold, send to queue
    if confidence[i] > conf_threshold:
        print(classes[i])
        #speakQ.put(classes[i])
        threshold_class = classes[i]
# add label class above confidence threshold
cv2.putText(
    img=bordered_frame,
    text=threshold_class,
    org=(int(0), int(frameHeight+20)),
            fontFace=cv2.FONT_HERSHEY_SIMPLEX,

            fontScale=0.75,

            color=(255, 255, 255)

        )


        # original video feed implementation

        cv2.imshow("Capturing", bordered_frame)

        cv2.waitKey(10)


        # # if the above implementation doesn't work properly

        # # comment out two lines above and use the lines below

        # # will also need to import matplotlib at the top

        # plt_frame = cv2.cvtColor(bordered_frame, cv2.COLOR_BGR2RGB)

        # plt.imshow(plt_frame)
        # plt.draw()
        # plt.pause(.001)

    # terminate process 1
    p1.terminate()

if __name__ == '__main__':
    main()
```

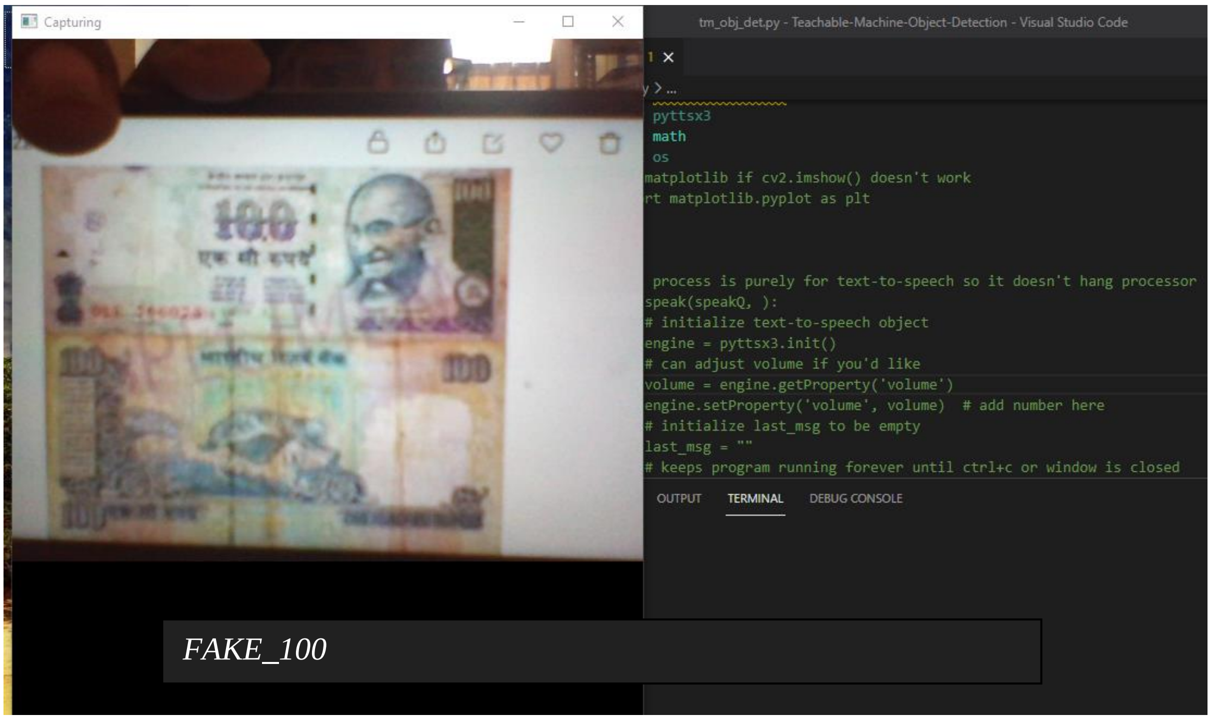
7.6 Result:


Capturing
tm_obj_det.py - Teachable-Machine-Object-Detection - Visual Studio Code
OUTPUT
TERMINAL
DEBUG CONSOLE

```
pyttsx3
math
os
matplotlib if cv2.imshow() doesn't work
rt matplotlib.pyplot as plt


 process is purely for text-to-speech so it doesn't hang processor
speak(speakQ, ):
# initialize text-to-speech object
engine = pyttsx3.init()
# can adjust volume if you'd like
volume = engine.getProperty('volume')
engine.setProperty('volume', volume)  # add number here
# initialize last_msg to be empty
last_msg = ""
# keeps program running forever until ctrl+c or window is closed
```

```
REAL_100
```

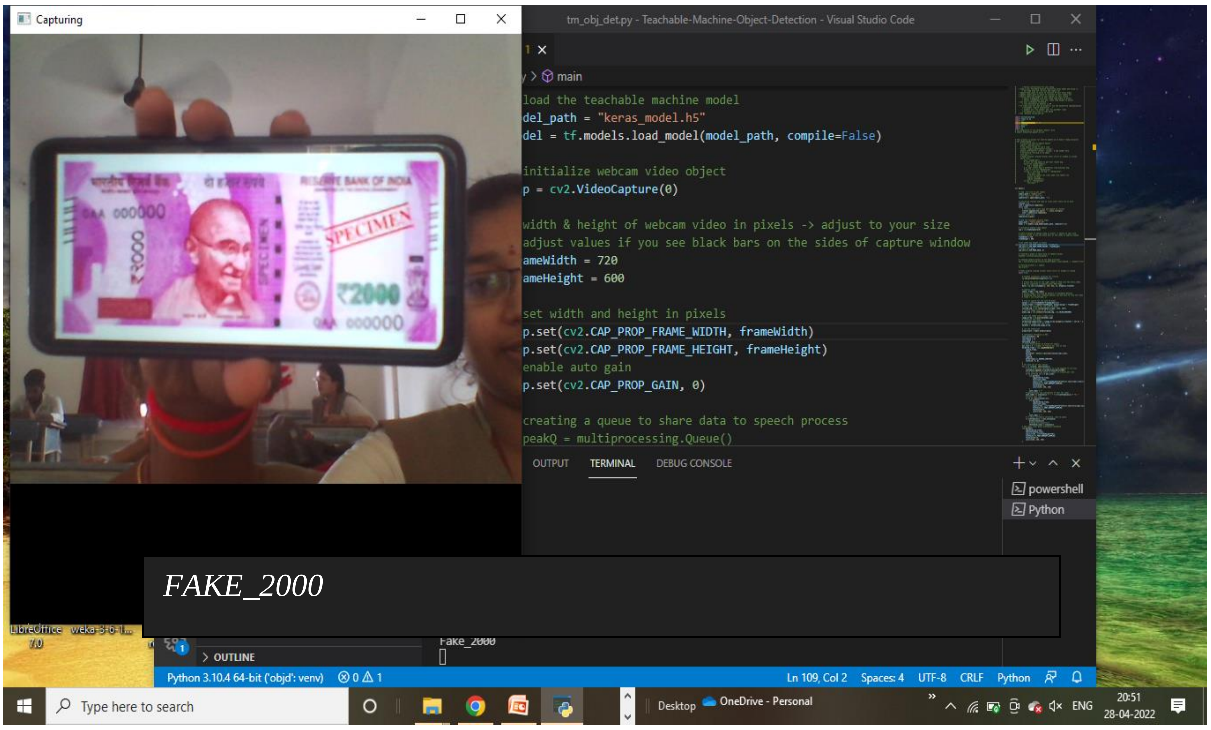
Fig 7.5: Real 100rs Currency Detection


Capturing
tm_obj_det.py - Teachable-Machine-Object-Detection - Visual Studio Code
OUTPUT
TERMINAL
DEBUG CONSOLE

```
pyttsx3
math
os
matplotlib if cv2.imshow() doesn't work
rt matplotlib.pyplot as plt


 process is purely for text-to-speech so it doesn't hang processor
speak(speakQ, ):
# initialize text-to-speech object
engine = pyttsx3.init()
# can adjust volume if you'd like
volume = engine.getProperty('volume')
engine.setProperty('volume', volume)  # add number here
# initialize last_msg to be empty
last_msg = ""
# keeps program running forever until ctrl+c or window is closed
```

```
FAKE_100
```

Fig 7.6 Fake 100rs Currency Detection



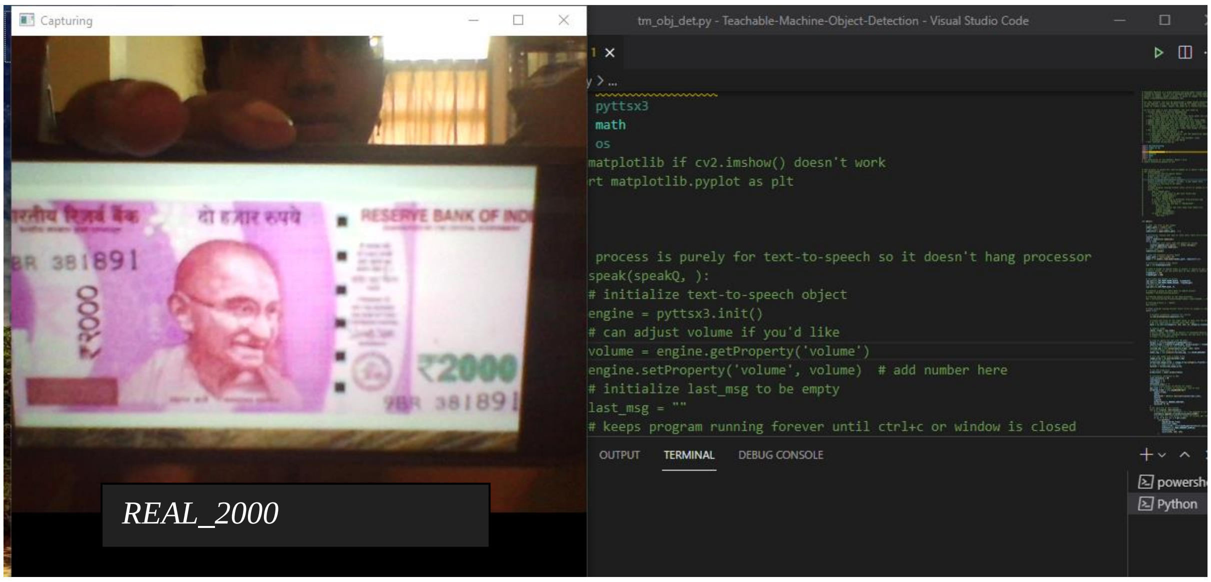
```
load the teachable machine model
del_path = "keras_model.h5"
del = tf.models.load_model(model_path, compile=False)

initialize webcam video object
p = cv2.VideoCapture(0)

width & height of webcam video in pixels -> adjust to your size
adjust values if you see black bars on the sides of capture window
ameWidth = 720
ameHeight = 600

set width and height in pixels
p.set(cv2.CAP_PROP_FRAME_WIDTH, frameWidth)
p.set(cv2.CAP_PROP_FRAME_HEIGHT, frameHeight)
enable auto gain
p.set(cv2.CAP_PROP_GAIN, 0)

creating a queue to share data to speech process
peakQ = multiprocessing.Queue()
```

```
FAKE_2000
```

```
Fake_2000
```

Capturing
tm_obj_det.py - Teachable-Machine-Object-Detection - Visual Studio Code
main
OUTPUT
TERMINAL
DEBUG CONSOLE
powershell
Python
OUTLINE
Python 3.10.4 64-bit ('objd': venv)
Ln 109, Col 2 Spaces: 4 UTF-8 CRLF Python
Type here to search
Desktop
OneDrive - Personal
ENG
20:51
28-04-2022

Fig 7.6: Fake 2000rs Currency Detection



```
pyttsx3
math
os
matplotlib if cv2.imshow() doesn't work
rt matplotlib.pyplot as plt


 process is purely for text-to-speech so it doesn't hang processor
speak(speakQ, ):
# initialize text-to-speech object
engine = pyttsx3.init()
# can adjust volume if you'd like
volume = engine.getProperty('volume')
engine.setProperty('volume', volume)  # add number here
# initialize last_msg to be empty
last_msg = ""
# keeps program running forever until ctrl+c or window is closed
```

```
REAL_2000
```

Capturing
tm_obj_det.py - Teachable-Machine-Object-Detection - Visual Studio Code
OUTPUT
TERMINAL
DEBUG CONSOLE
powersh
Python

Fig 7.7: Real 2000rs Currency Detection

8. SYSTEM 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. There are various types of test. Each test type addresses a specific testing requirement.

8.1 TYPES OF TESTS:

8.1.1 Unit Testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.1.2 Integration Testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

8.1.3 Functional Testing:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.1.4 System Testing:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.1.5 White Box Testing:

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.1.6 Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

8.1.7 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

TESTING CODE:

- ➢ Code is usually developed in a file using an editor.
- ➢ To test the code, import it into a Python session and try to run it.
- ➢ Usually there is an error, so you go back to the file, make a correction, and test again.
- ➢ This process is repeated until you are satisfied that the code works.
- ➢ The entire process is known as the development cycle.
- ➢ There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.
- ➢ This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

9. APPLICATIONS

An application program is a computer program designed to carry out a specific task other than one relating to the operation of the computer itself, typically to be used by end-users. Word processors, media players, and accounting software are examples. The collective noun refers to all applications collectively.

Some of the applications are:

- Currency Checking Machine.
- Fake currency detection App.
- Counting Machine.
- A Currency Recognition App.
- Platform independent application to recognize Indian currency notes using deep learning techniques
- A Currency Detector App for Visually Impaired.
- Indian Currency Note Detector And Voice Assistant Detection For Blind.

10. CONCLUSION

Our model is for performing the fake currency detection. The detection accuracy is most accurate since the currency characteristics features are learned through layer by layer. Here we have considered the whole currency image, but in future we will try to include all the security features of currency by employing suitable structural design and with suitable training data. Further, noise may be present in the captured image which has to be considered as a pre-processing step in currency detection process. The recognition and fake currency detection can also be extended by considering the patterns of currency surface as features for improving the detection accuracy. Deep learning techniques are crucial to currency note recognition. Based on deep learning literature review, most of the researchers using Convolutional Neural Networks for recognizing fake notes from the real notes. This technique can get more accuracy after comparing this technique with other previous works. Future researchers should build a system that can be used for currency recognition for all countries around the world because some of the models identify few different currencies. On the other hand, some of the current models cannot identify fake money. Finally, the researchers should crucially work on extracting security thread features.We proposed a Indian currency detection and counting method that uses CNN, RNN through teachable machine and banknote images captured by a camera under visible light conditions. Our method was designed to classify genuine and fake banknotes regardless of the denomination and the side of the banknote exposed to the camera. In addition, the use of a camera for capturing banknote images allows general users, including visually impaired people, to identify fake banknotes without using specialized devices.

11. FUTURE ENCHANCEMENT

Machines available today are not only fake note detector but they provide an extra facility of counting them. This feature can be added with our device that would make it as most reliable counterfeit currency detector along with counting feature that would be helpful for banking purpose. This project discussed a technique for verifying Indian paper currency. This project is an effort to suggest an approach for extracting characteristic of Indian paper currency. Approach suggested from the beginning of image acquisition to converting it to gray scale image and up to the word segmentation has been stated. The work will surely be very useful for minimizing the counterfeit currency. In Future, Mobile app can be developed which would be useful for normal as well as visually impaired persons, the same system can be developed for the remaining Indian currency notes and other country's currency notes. Also the app's interface can be further modified as per the user requirements. This will increase its utilization by increasing its user network since India is going to establish the largest digital network in the world in the coming years. Thus the application will be available in all android devices and IOS devices in future if worked upon.

Technology is growing on a large scale. The proposed system can be extended towards fake currency recognition and counting. Denomination 10, 20, 50, 100, 200, 500 and 2000 of other countries other than India can be added also comparison between them can be achieved. When an image is loaded from the outside into the training folder then it is not giving 100% accuracy. We can improve this problem by optimizing the system.

12. REFRENCES

[1]. VedasamhithaAbburu, Saumya Gupta, S. R. Rimitha, ManjunathMulimani, Shashidhar G. Koolagudi, Currency Recognition system using Image Processing, Tenth International Conference on Contemporary Computing (IC3), 10-12 August, 2017, Noida, India

[2]. GokulRamasamy, Sakthi Subramanian, Identification of currency denomination using image processing, International Journal of Advance Research, Ideas and Innovation In Technology, ISSN: 2454-132X, Impact factor: 4.295, (Volume 5, Issue 2).

[3]. Paper Currency Detection based Image Processing Technique : A review paper by Shaimaa H, Shaker and Mohammed GheniAlawan,Journal of AL-Qadisiyah for computer science and mathematics, Vol.10 No.1 Year 2018, ISSN (Print) : 2074 – 0204, ISSN (Online) : 2521 – 3504.

[4]. Ahmed Yousry , Mohamed Taha and MazenM.Selim, Currency Recognition System for Blind people using ORB Algorithm, International Arab Journal of e-Technology, Vol. 5, No. 1, January 2018. [5]. Chakraborty, K. ; Mukherjee, S. ; Dasgupta, D. ; Basumatary, J. ; Kalita, J. C. , "Robust Framework for Indian Currency Denomination Recognition for Visually Impaired," ADBU Journal of Engineering Technology (AJET),vol. 1, no. 1, pp. 7-11, 2014.

[6]. Aruna D H, ManpreetBagga, Dr.Baljit Singh "A Survey on Indian Currency Note Denomination Recognition System". International Journal of Advance Research in Science and Engineering, IJARSE, Vol. No.4, Special Issue (01), 2015.

[7]. Gouri Sanjay Tele, AkshayPrakashKathalkar, SnehaMahakalkar, Bharat Sahoo and VaishnaviDhamane, Detection of Fake Indian Currency, International Journal of Advance Research, Ideas and Innovation In Technology, ISSN: 2454-132X Impact factor: 4.295, (Volume 4, Issue 2).

[8]. Qian Zhang, Currency Rcognition using Deep Learing, Thesis submitted to Aukland University of Technology, 2018. No of Epochs Accu racy JOURNAL OF CRITICAL REVI.

[9]. M. Laavanya, V. Vijayaraghavan, Real Time Fake Currency Note Detection using Deep Learning, International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249 – 8958, Volume-9 Issue-1S5, December, 2019.

[10]. Shubham Mittal, Shiva Mittal, Indian Banknote Recognition using Convolutional Neural Network, 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU), Electronic ISBN: 978-1-5090-6785-5, Print on Demand(PoD) ISBN: 978-1-5386-3833-0, 05 November, 2018.

[11] .TuyenDanh Pham, Young Ho Park, Seung Yong Kwon, Kang RyoungRark, DaeSikJeong and Sungsoo Yoon, Efficient Banknote Recognition Based on Seection of Discriminative Regions, MDPI journal, Sensors(basel), Volume16, Issue 3, 10.3390/s16030328.

[12]. AnishShrestha, BhuwanKhadka, NimeshDahal, SarojBhattarai, cash recogniton system using convolution neural network with transfer learning, tribhuvan university institute of engineering purwanchal campus, November, 2018.

[13]. HanishAggarwal and Padam Kumar "Indian Currency Note Denomination Recognition in Color Images". International Journal on Advanced Computer Engineering and Communication Technology, Vol-1 Issue: 1, 2014.

[14]. Ali, T. et al. (2019) 'DeepMoney: Counterfeit money detection using generative adversarial networks', PeerJ Computer Science, 2019(9). doi: 10.7717/peerj-cs.216.

[15]. Chowdhury, U. R., Jana, S. and Parekh, R. (2020) 'Automated System for Indian Banknote Recognition using Image Processing and Deep Learning', in 2020 International Conference on Computer Science, Engineering and Applications, ICCSEA 2020, pp. 1–5. doi: 10.1109/ICCSEA49143.2020.9132850.

[16]. Dai, J. et al. (2016) 'R-FCN: Object detection via region-based fully convolutional networks', in Advances in Neural Information Processing Systems.

[17]. Goodfellow, I. et al. (2020) 'Generative adversarial networks', Communications of the ACM, 63(11). doi: 10.1145/3422622.