B. Tech Software Engineering

Aakash Gupta

2020

Smarter Charts

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF TECHNOLOGY

IN

SOFTWARE ENGINEERING

Submitted by:

Aakash Gupta

2K17/SE/002

Under the supervision of

Dr. Anil Singh Parihar

Department of Computer Science & Engineering

Delhi Technological University

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

Department of Computer Science & Engineering

Delhi Technological University

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, AAKASH GUPTA, Roll No. 2K17/SE/002 of B. Tech. Software Engineering, hereby declare that the project Dissertation titled "SMARTER CHARTS" which is submitted by me to the Department of Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology. The project is not an original concept and has been implemented before but the implementation done is by me.

Place: Noida **Aakash Gupta**

Date: 22/07/2020

Delhi Technological University

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I, hereby certify that Project Dissertation titled "SMARTER CHARTS" which is submitted by AAKASH GUPTA, Roll No. 2K17/SE/002 of Department of Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology, is a record of the project work (implementation) carried out by the student under my supervision. The project is not an original concept and many other similar tools and applications are available. However, to the best of my knowledge, the implementation done here is by the student.

Place: DELHI **Dr. Anil Singh Parihar**

Date: 22/07/2020 **SUPERVISOR**

Acknowledgment

I, AAKASH GUPTA, Roll No. 2K17/SE/002 of B. Tech. Software Engineering, would like to express my thanks of gratitude to my supervisor DR. ANIL SINGH PARIHAR and our, Delhi Technological University, Delhi who presented me with this golden opportunity to do this project. It was a great learning experience especially with a mentor such as my supervisor. I would also like to thank them in their constant motivation due to which I could complete this project in the stipulated timeline.

Aakash Gupta

(2k17/SE/002)

Abstract

Often while reading business or even academic literature we come across several charts and graphs (of various different types) which convey useful information that we want to use and manipulate digitally but converting the hard copies into digital format is a tedious task and thus Smarter Charts aims to provide a Web-based interface where one could simply upload an image containing charts and graphs and get data in stylesheets. The project is already implemented and available on the internet called WebPlotDigitizer which does the bar chart but we need to select the type of graph that we are doing using our deep learning model also we need to select the axis points manually while for Pie Chart which uses different gradient calculation to find the slices and further process the image thus it is not an original idea but an unique implementation. We Start by browsing and subsequently uploading our image in .PNG (**P**ortable **N**etwork **G**raphics) format and providing the name of output file on the Streamlit based Web Application, the CNN (**C**onvolutional **N**eural **N**etwork) based Deep Learning model predicts the class of Graph the image contains, i.e., Pie Chart or Bar Graph. Depending upon the category of the graph, the image belongs to, the algorithm for calculating the values depicted by different colors in the graphs are executed. The output image containing edge markings detected by the algorithm also, in .PNG format is saved for being attached in the output Stylesheet and values along with their corresponding colors are then used as depicted in Fig[1] to create a new .XLSX file (MS Excel Spreadsheet Format), and the output of our algorithm is then exported to the same with the output image. The .XLSX file with the name provided is then opened and available for viewing for the User who can further manipulate the data according to his/her requirements of the data. After the User has Downloaded/Saved the .XLSX File he/she then provides a command that the viewing is completing and the .XLSX file along with the output Image is removed from the Server Storage for the safety of User data

and Space constraints along with Scalability. The whole project is coded in Python 3.6 and has low Processing requirements along with very little memory requirements

CONTENTS

List of Figures

List of Tables

Introduction

We use graphs to express the mathematical and statistical Data more visually so that it is easier to understand and also makes the process of explaining the data faster. We often encounter graphs such as Pie Charts and Bar Graphs in Research Papers, Textbooks, Business reports, and various other documents that we read in our research process or leisure reading also. Pie Charts are used to show the share of a category in the total, like, for example, market share of a particular brand or the consumption of one type of item out of the total consumption. On the other hand, Bar Graphs are used to show quantities of the categories depicted in the graph. Both ways of depicting data are quite useful and often serve the purpose well. However, there are times that the data provided by theses graphs are needed so that they could be further processed and manipulated to generate the results or sometimes produce more related graphics at those instances of time we need the data in a flexible format so that we can easily and quickly use stylesheets tools such as Microsoft Excel and complete our goal with the data. The project is not a completely original idea and nor it is claimed but the implementation is Unique with a interface based upon Streamlit.

Often, the graphs are just images that were generated; thus, they cannot be directly used to extract data and work our way further. Thus, we need a tool that could convert the images often .PNG (**P**ortable **N**etwork **G**raphics) format and.JPG (**J**oint **P**rofessional Expert **G**roup) a format that needs to be processed to give data in a soft format and arranged in Stylesheets accordingly, which is inconvenient for the user. Extracting the data from images in this manner when the underlying data is not directly available is useful in cases where data extraction has to be applied on images from web pages, research publications, or slides. Data extraction systems [5] have been developed using deep learning-based classifiers that use semi-automatic

methods to detect the chart type and apply data extraction algorithms on the identified charts. A survey of transfer learning methods [1] revealed that models that are trained on data from different domains can be used effectively for newer applications. This served as a motivation to develop an application using existing model to fully automate the process of chart type classification.

Convolutional Neural Networks have been used for chart data extraction that work on both 2D and 3D charts [6] which are perceptually more useful to the users. Such applications However, the application developed in the present work primarily processes 2D images more effectively.

Existing free software like WebPlotDigitizer [4] claim to detect various plot types and are also able to extract data from them. It has also been used to extract polarimetric data from images received from Hubble Space Telescope which indicates the scope of usage of data extraction applications.

In order to make it easier for the user to extract data from images containing quantitative data, an interface is required. Existing applications (as mentioned in the product survey chapter) adopt the same principle of user friendliness. Generally, a chart image is taken as input [3] and the graphical components are analysed and the application further identifies the chart type and applies category specific methods to extract the underlying data.

“Smarter Charts” aims to be the tool that could perform this task efficiently without needing much computational power that could be deployed even on basic hardware and still run and give results in under 10 seconds while still being extremely convenient for the User. The User is provided with an intuitive and easy to use Web Application, as shown in Fig. [2] based on Python, where they could simply upload their image or even drag and drop the image, and after a few clicks, they could download their Microsoft Excel file containing the data. After the file is processed and their stylesheet file is downloaded their image along with their Output Image

and their Stylesheet file is deleted to conserve memory and also to ensure that their data may not be Misused by anyone as it is no longer available anywhere on the server-side. The tool could currently process Bar Graph and Pie Charts only.

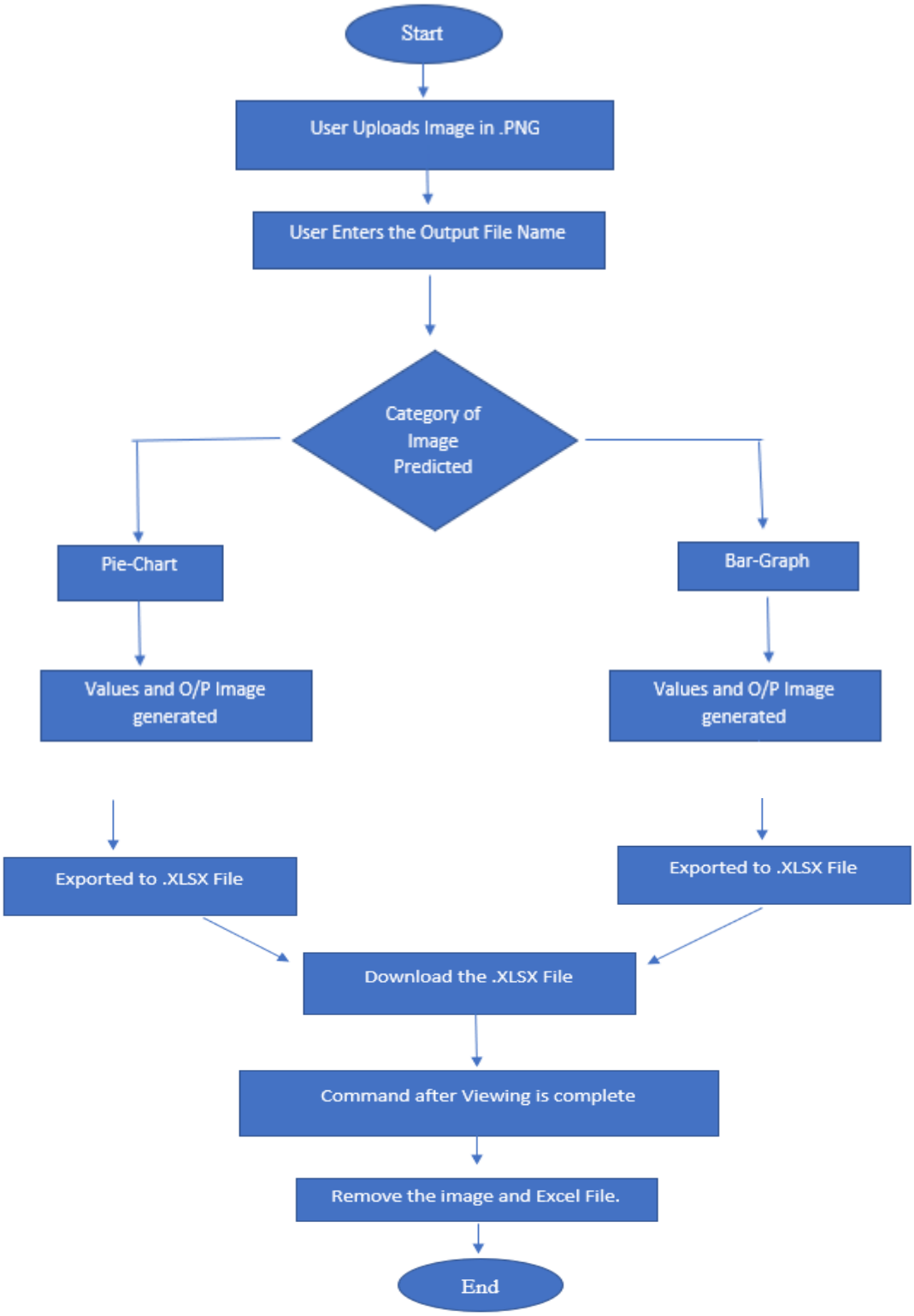

Fig. [1] :The Flowchart of the project.

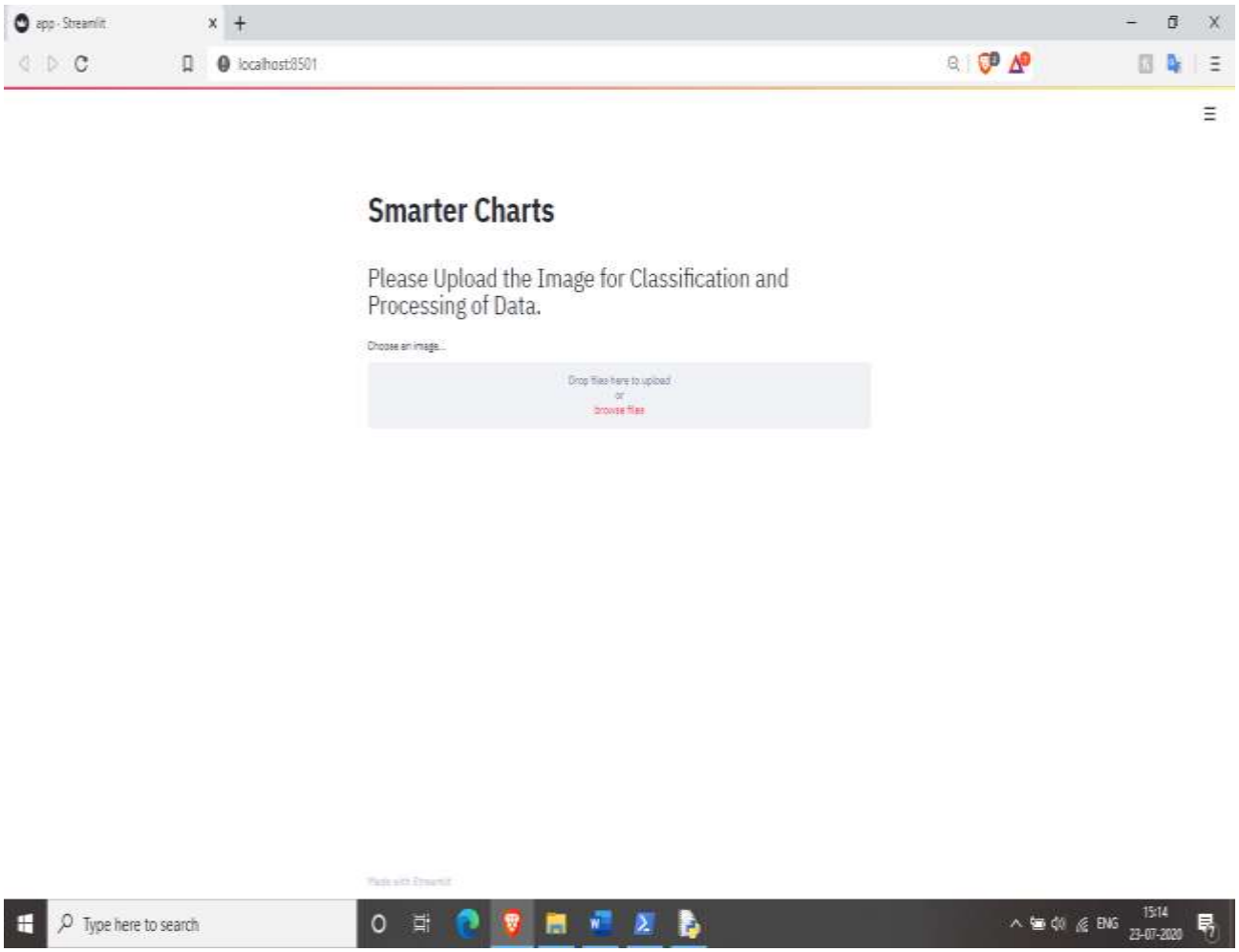

Fig. [2a] : The Web Application based on Streamlit[1] .

Fig. [2b]: The Web Application shows the image that is Uploaded after the file is uploaded

1. Product Survey

Existing Product	Existing Features	Comparison with Smarter Charts
WebPlotDigitizer	1. Works with 1.1. Bar Plot 1.2. XY plots 1.3. Polar plot 1.4. Ternary plot etc. 2. Measures distances and angles between various features 3. Open-Source, Free to use and cross-platform 4. Semi-automated with sub-pixel resolution algorithms. 5. Complex web application	1. Works with 1.1. Bar plot 1.2. Pie Charts 2. Measures the vertical length of the bar and angles subtended by sectors at the center 3. Cross-Platform. 4. Fully automated from the detection of the image to result generation. 5. Simple Web application
Dexter: Data Extractor for scanned graphs	1. Java-based application 2. Basic functionality includes the selection of the area of interest of the image and then the coordinate axes and points. 3. Advanced functionalities include automatic point finding, coordinate axes, and line tracing 4. The web application requires a web browser. 5. Used for ADS(NASA) data.	1. Python-based tool. 2. Completely automatic data extraction for Bar graphs and pie charts. 3. Requires a web browser to run the streamlit based web application.
Graph Grabber	1. Works only on windows 10 PC with .NET framework. 2. Includes both manual and automated workflow options. 3. Free to use 4. Outputs rendered image.CSV File or third-party spreadsheet.	1. Works with any device which has a web browser and MS Excel. 2. Only automatic workflow option. 3. The output is an MS Excel spreadsheet with the rendered image along with the data.
Enguage Digitizer	1. Supports all major image formats.	1. Supports.PNG and.JPG

	2. Supports only coordinate axes based plots. 3. Automated workflow with manual fine tuning 4. Works only on Linux OS. It is a free to use tool.	2. Supports bar graph and Pie charts. 3. Fully automated workflow with no manual intervention, 4. Works with any device which has a web browser and MS Excel.
GetData Graph Digitizer	1. User needs to select the axes in the scientific plot and then he/she can manually trace the plot. Or use the automatic tracing. 2. Easy to use interface. 3. Outputs data in variety of formats including .TXT file, .XLS file or autocad file 4. Supports Scientific plots 5. Compatible only with Windows OS.. It is free to use.	1. User needs to upload the image and then download the output Excel file, workflow is automated. 2. Interface is designed for user friendliness. 3. Output is only given in MS Excel file. 4. Supports bar graph and pie chart. 5. Works with any device which has a web browser and MS Excel.
ChartSense	1. It is s proprietary software owned by Microsoft. 2. It is fully automated but user can adjust manually the coordinate axes that are detected to increase accuracy. 3. Output is given in MS Excel. 4. The classification of image is done using deep learning. 5. Interface is very user friendly and it supports wide variety of charts including bar chart, pie chart, tables, area chart etc	1. It is fully automated but we cannot manually adjust. 2. Output is in MS Excel. 3. The classification is done using deep learning. 4. Smarter charts support Bar graph and Pie Chart. 5. Very user friendly interface.

Table [1]: Product Survey

3. The Web Application

3.1 Streamlit Framework

The Web Application, as shown in Fig. [2] is developed using Streamlit[1]. Streamlit is an open source and free to use Python-based app framework that has seen high adoption among Data Scientists and Machine Learning enthusiasts. The reason for the success of Streamlit has been the 13 Streamlit calls-based development that allows the users to develop fully interactive web applications in a matter of hours. The new tool is efficient and even supports the caching of data and easy to learn and fully compatible with pandas Data Frame. The developer just needs to know the 13 calls, and he/she can develop a fully functional and interactive web application. As it is quite a new open-source tool, there are certain limitations and bugs in the framework, but the community support is good, and the framework is under constant update.

3.2 Smarter Chart Web Application

Smarter Chart being pure Python, integrating Streamlit was a simple task. Being a simple tool and straight forward tool, only 6 of the 13 Streamlit calls were used in making the Web application which are as follows:

- Streamlit.title()
- Streamlit.header()
- Streamlit.write()
- Streamlit.image()
- Streamlit.button()
- Streamlit.Text_Input()

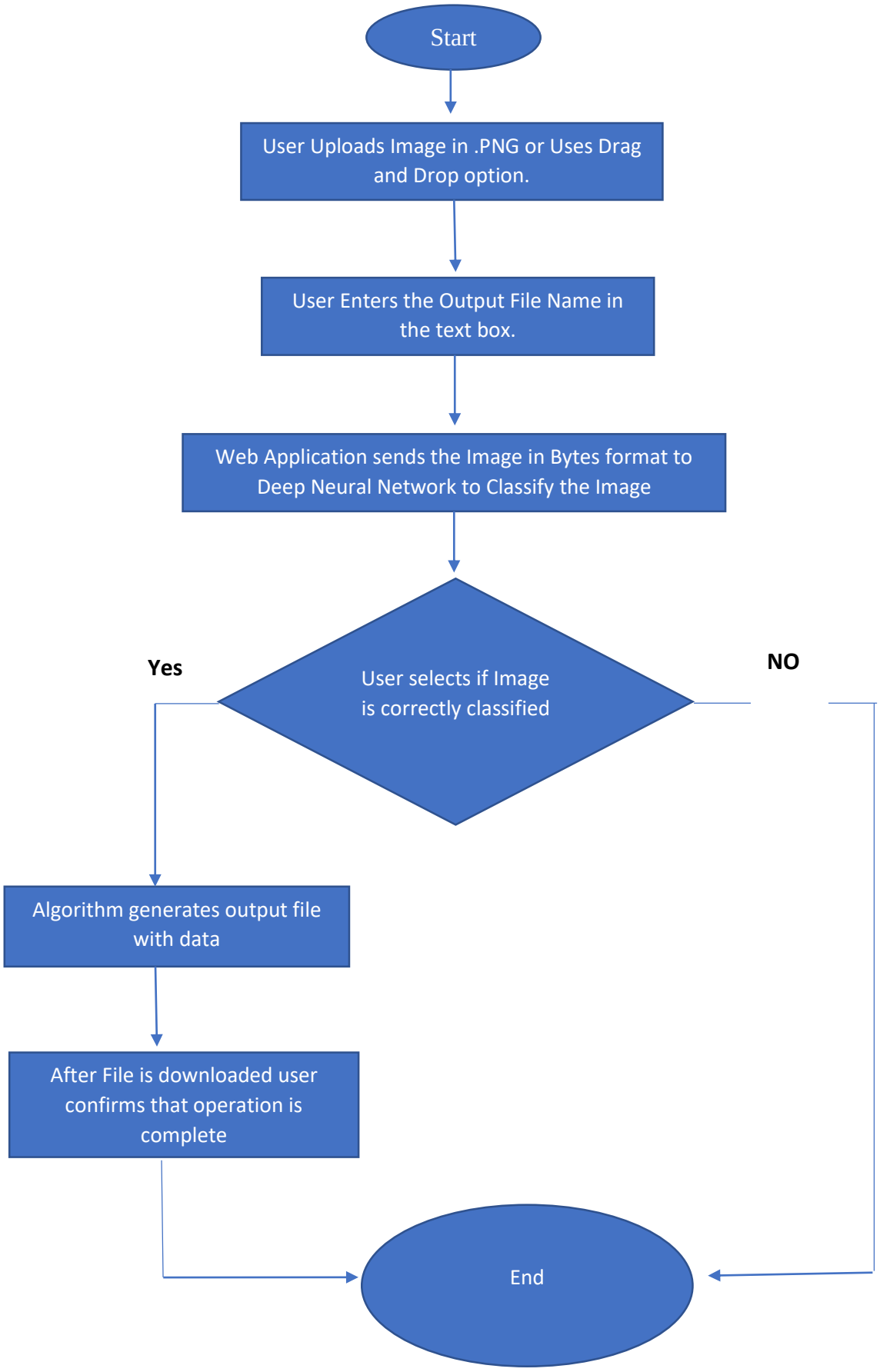

Fig. [3]: The Flow Chart depicting the working of the Web Application

4. Deep Neural network

4.1 Introduction

Deep Learning is a relatively new concept under the vast field of AI (**A**rtificial **I**ntelligence). The concept is that we have Weights, Activations, and Affine functions. The weights need to be initialized using Xavier's Initialization[2] or Kaiming Initialization[2], which ensures that standard deviations of the weights remain around one and the weights of the later layers do not converge to 0. After the weights are initialized, we train the model which could be either supervised or unsupervised learning. During the training of the model, we input a matrix (input is usually a matrix of number) it is multiplied with a weight to give out activations which are then given as input to an activation function such as ReLU[2] (**R**ectified **L**inear **U**nits) which are then again given as inputs to the next layer of weights and so on till the last layer of the neural network is reached where the output is used to calculate according to the loss function defined in the model, for example, MSE (Mean Square Error) which calculates the error as follows:

$$\text{Loss} = \sum((y-(y_i))\text{^}2)/n$$

Where,

y = Output

y_i = Target or the actual output

n = Total Observations

Eq [1] : Mean Square Error

Thus, after calculating the loss which would be high if the prediction is not correct else will be small, we use an optimizer like SGD (**S**tochastic **G**radient **D**escent) to adjust the weight

accordingly, so that the error is reduced and the model learns from the training data. To make a deep learning model completely, we require to define the three most critical aspect of it:

1. The architecture of the model
2. Loss Function
3. Dataset

4.2 Transfer Learning

We have used the existing ResNet34 and used FastAI library to add layers in the end which makes the model suitable for a 2 label classification. The FastAi makes this addition of layer process automatic and I have not added the layer manually on my own also the detection using Neural Network. The training of a model is the most computationally requiring and tedious task. It requires a Large dataset to train the model to acceptable accuracy levels, which are generally considered as >80% of accuracy. Often, the kind of models we require are already trained and made available on the Web, which can be simply downloaded and then modified to suit our requirements. Generally, what we do is we search for the model that gives reasonable accuracy , and then we add few layers at the end of the models then we first train a few epochs (or cycles) with the complete and then train last few (the layers that we appended) layers with higher learning rates as the earlier layers are already optimized, but the new layers that we added will not be; thus, we use a higher training rate and freeze the existing layers while training. This process is called Transfer learning, and it solves a few of the drawbacks of deep learning that we earlier listed as now we would not require a very large dataset to train the model and also the time and computation required to train the model are also reduced. The model used in Smarter Charts is developed using Transfer Learning. ResNet34[4] model for Computer Vision, as shown in Fig. [4], which won the ImageNet competition, was added with

few extra layers, in the end, to classify the image into two categories, i.e., Pie Chart or Bar Graph. Click here for visiting the notebook on Kaggle.com. The training of the model was easily 344+42 = 386 Image files, out of which 344 were the Images that were used to train the model while 42 were used for validation.

```
learn = cnn_learner(data, models.resnet34, metrics=error_rate,model_dir = '/kaggle/worki
ng/')
```

```
Downloading: "https://download.pytorch.org/models/resnet34-333f7ec4.pth" to /root/.ca
che/torch/checkpoints/resnet34-333f7ec4.pth
```

100% 83.3M/83.3M [00:17<00:00, 4.87MB/s]

Fig. [4]: Resnet34 Model being imported during Transfer Learning Process

In [11]:
```
lr=1e-3
```

In [12]:
```
learn.fit_one_cycle(5,slice(lr))
```

epoch	train_loss	valid_loss	error_rate	time
0	1.346404	0.870174	0.357143	00:13
1	0.765477	0.007358	0.000000	00:11
2	0.515315	0.000275	0.000000	00:11
3	0.374717	0.000158	0.000000	00:11
4	0.296736	0.000279	0.000000	00:11

Fig. [5]: Training of our model with learning rate = 10^{-3}

As can be seen in Fig. [5] we have trained our model with a learning rate 10^{-3} and that the model was trained in 67 secs for the five epochs and we have used discriminatory learning rates using the slice (LR) function which even distributes the learning rates from 0 to LR from the first layer to the last layers that we have added. The reason behind the discriminatory rates is that during transfer learning, we train the layers that we have added over and above the existing model needs more training than the existing layers of the model thus the earlier layers are trained with a low learning rate while later layers are trained with a higher learning rate.

4.2.1. Architecture of the Network

We input the image which is then converted into a matrix of dimension Row x Column. The matrix is served as an input to ResNet34 which is a 34 layers residual neural network which was developed for the ImageNet competition and was a winner in image classification. The ImageNet competition provided the dataset and so the ResNet34 is trained on those images thus before inputting the image to ResNet34 we normalize the image to ImageNet statistics. The ResNet34 outputs a 512x1 matrix thus to suit our purpose we add a linear layer at the end that outputs the 2 class labels i.e., Pie Chart and Bar Graph where we have set the Dropout as 0.5 and bias = True.

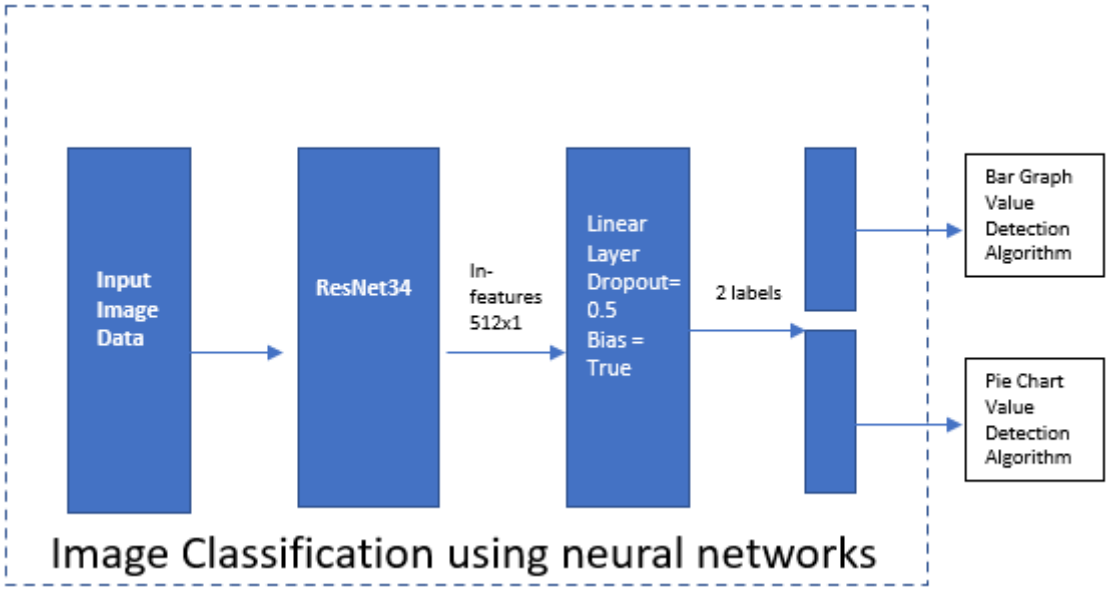

Fig. [6]: Architecture of the network

5. Algorithm for Bar Graph

5.1 Introduction to Bar Graphs

Bar Graphs are a pictorial representation of categorical data. In the Bar graph, rectangular boxes are drawn vertically to represent the proportion of the data that it represents. They are very commonly used to represent mathematical or statistical Data. A typical Bar Graphs looks like Fig. [7].

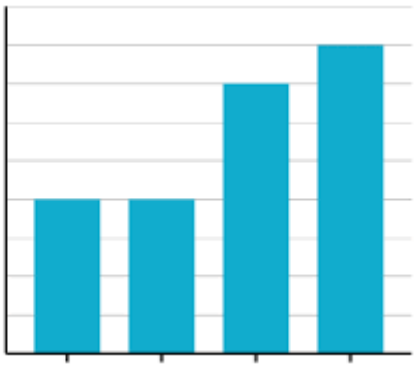

Fig. [7]: A typical Bar Graph. Credit Khan Academy

Bar Graphs are often confused with Histograms. Histograms do contain rectangular boxes like Bar Graph but they are used to compare data within a category for example Bar Graph may depict sales of different models of car from a manufacturer and Histogram may show the comparison between the sales of the model of the car YoY (**Y**ear **o**n **Y**ear) for all the different models that the manufacturer sells.

5.2 Steps for Value Calculations in a Bar Graph

5.2.1 Step 0: Corner Detection and sorting

The First step is to detect corners and then sort them into two arrays, in the first array one we store the corner points detected by the GoodFeaturesToTrack[4] function of the Open-CV[3] library sorted according to their x coordinates, and in the second list we store the corners detected by the same function of the Open-CV library sorted according to their y coordinates. Also, the image is converted into grayscale before being passed on to GoodFeaturesToTrack function.

5.2.2 Step 1: Detecting the X-Axis in the Image

We take every point in the second array(sorted for x-coordinate) and check for other points lying horizontally within a margin of the y coordinate of that point and then make the corresponding list of every point thus whichever point has the maximum horizontal neighbours is considered a point on the x-axis, and the x-axis is detected.

5.2.3. Step 2: Detecting a point on a bar above X-axis

We take a point on the X-axis and then check if there is a point in the first array where x-coordinate is the same or within a margin but have different y-coordinate and store it a verification point.

5.2.4 Step 3: Horizontal Scanning of X-axis

We scan the along the X-axis to detect the various bars and store the coordinates of the 2 ends of a horizontal gap with the same color but different from background color encountered while horizontal scanning along with the thickness of the gap in a dictionary and the color is stored with the same index in a separate list.

5.2.5 Step 4: Calculating Width of a Bar and to compute all the bars accordingly

We check the thickness of gaps within a margin, and the gaps with the most occurring thickness are registered as bars, and the thickness is stored along with the index of the gaps in so that we can extract their coordinates from the earlier stored dictionary. Thus the bars can be detected.

5.2.6 Step 5: Calculating the Vertical heights of the detected bars

Now, we calculate the vertical heights of the bars by taking all the stored key and value pairs in the dictionary created in step 3 and check for the thickness, if it matches the width/thickness calculated in step 4 then we calculate the vertical height by taking the point on the x-axis from the verification point stored in step 2 and the color stored in the list in step 3 and then running a loop till we reach a vertical height where color is not equal to the color of the bar. Finally, a fresh dictionary is generated with 3 items copied from the previous dictionary and a 4^{th} item being added, which is the height. Click here for visiting the source code.

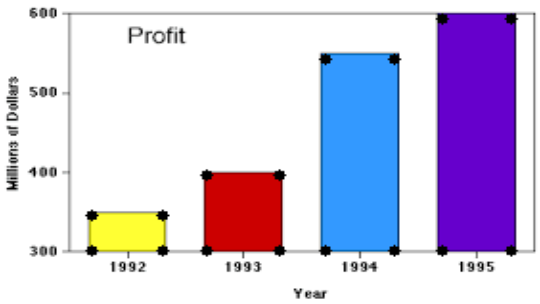

Fig. [8]: The Final Image generated showing the Bars detected by this Algorithm

6. Algorithm for Pie Charts

6.1 Introduction to Pie Charts

Pie Charts are also used for a pictorial representation of a mathematical or statistical data. Here, the difference is that the value representation is very different, and it is not used to represent the absolute values of a category but just the share of it in percentages. The percentage share is calculated by calculating the total of all the values across the category and then dividing each value and then multiplying it with 360 Degrees. The value received is the angle that the arc, corresponding to the value, subtends at the centre. Typical Pie Chart looks like in Fig. [9].

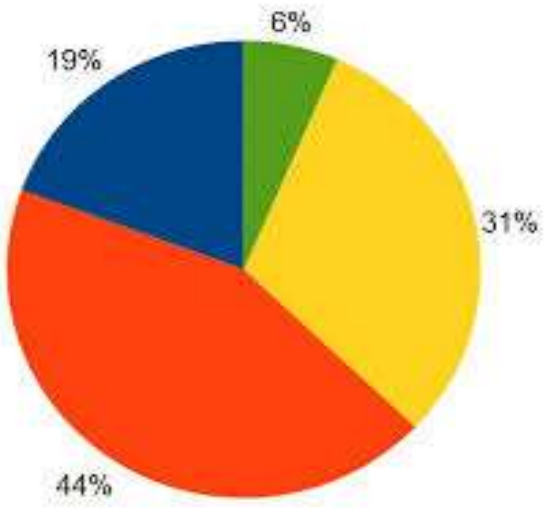

Fig. [9] : A typical Pie Chart. Credit : Chartio

The total of all the angles is 360 Degrees and generally the different Sectors are represented with different colours to make pie chart more legible.

6.2 Steps for Value Calculations in a Pie Chart

Just like in the Bar Graph, we proceed step by step for the pie chart as well.

6.2.1 Step 1: Detection of the Centre of the Pie Chart

The First step is to detect corners and store the corner points detected by the GoodFeaturesToTrack function of the Open-CV library. The basic concept is that the center of the Pie chart is surrounded by the maximum number of colors. For each point in the detected list of corners, we first make a circle around the point with radius **r** (defined on basis of the size of the image), then we select x = x+2, and x=x+r-1 and calculate the points on the circle defined with these values of x. We get 8 points around the center and we check the number of unique colors. The point with the maximum number of unique colors is detected as the center point.

6.2.2 Step 2: Calculating The radius of the Pie Chart

We again define a circle around the center point detected in the previous step with a radius r. Moreover, for x=r-1, we get 4 points on that circle. These 4 points are then checked for color and if None of them is background color, we start incrementing the radius of the circle defined and keep on checking the same 4 points for their color. When the color of all the 4 points change to the background color, we stop the incrementation and define that distance as the radius of our pie chart.

6.2.3 Step 3: Detecting all the points on the edge of the Pie Chart

We again define a circle around the center point and using x=r-1 again calculate the 4 points Around the center point. Then we transverse along each edge of the square to detect an edge,

If we encounter an edge along our path, we register the point and its color in a separate list. After all the edge points are detected by horizontally or vertically comparing around the points on the square, using the Centre Point and the point on edge, we define an equation of a line using two-point method. Then using the radius that we calculated in the previous step, we calculate the coordinate of the point on the line and having a distance equal to the radius from the center. Thus, this gives us output such as in Fig. [10].

6.2.4 Step 4: Calculating the Angles Subtended by the Various Sectors

We pass the Centre Point, the points lying on the edge of the pie chart, and a reference vector to a function to calculate the Angles subtended by the various sectors. The reference vector is used to sort the points on the edge of the circle by taking the dot product and the difference product between the point on the edge and reference vector, and thus the points are sorted in a clockwise order. After the points are sorted, we calculate the distance between the center and the various points on the edge of the pie chart(radius) and average out the distance for better accuracy of the data. Using the properties of the triangle, we take the Sin^{-1} of half distance between 2 consecutive points of the sorted list divided by the average radius calculated to receive half-angle, which is then multiplied by 2 to receive the angle subtended by the sector at the center. Thus, a list is made of the color of the sector, and the angle subtended at the center.

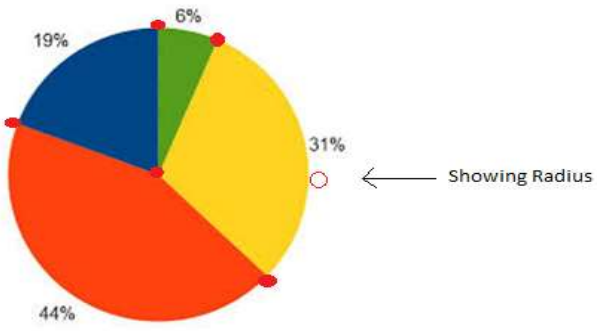

Fig. [10]: Output File generated by the Algorithm

8 Arranging Data in a StyleSheet

8.1 Xlsxwriter

To export the data generated by the Bar graph or pie chart algorithm we use Xlsxwriter library Of Python. It is open-source and free to use the tool. Using this tool, we can write text, merge Cells, export images and it even has the support for Pandas so that one could directly putTheir dataframe to an excel file. Though we cannot read and export to an existing excel file still has more features than similar other libraries available and also has more elaborate Documentation. We can not only write text in a cell, but we can also specify its formatting all this in One line of code. The code used to initialize the export of data generated by Smarter Charts are shown in Fig. [11].

```
workbook = xlsxwriter.Workbook(Output_Name)
worksheet = workbook.add_worksheet()
worksheet.set_column('C:C', 30)
worksheet.set_column('A:A', 15)
worksheet.set_column('B:B', 15)
```

Fig. [11]: Xlsx Code in the Bar Graph Module which is almost same as the Pie Chart Module

Further, we have formatted the headings to bold to increase legibility, as shown in the. The output generated by this library is speedy and has a very high fidelity degree, i.e., The file generated by this library is almost or sometimes completely like generated by excel itself.

The output is shown in Fig. [12].

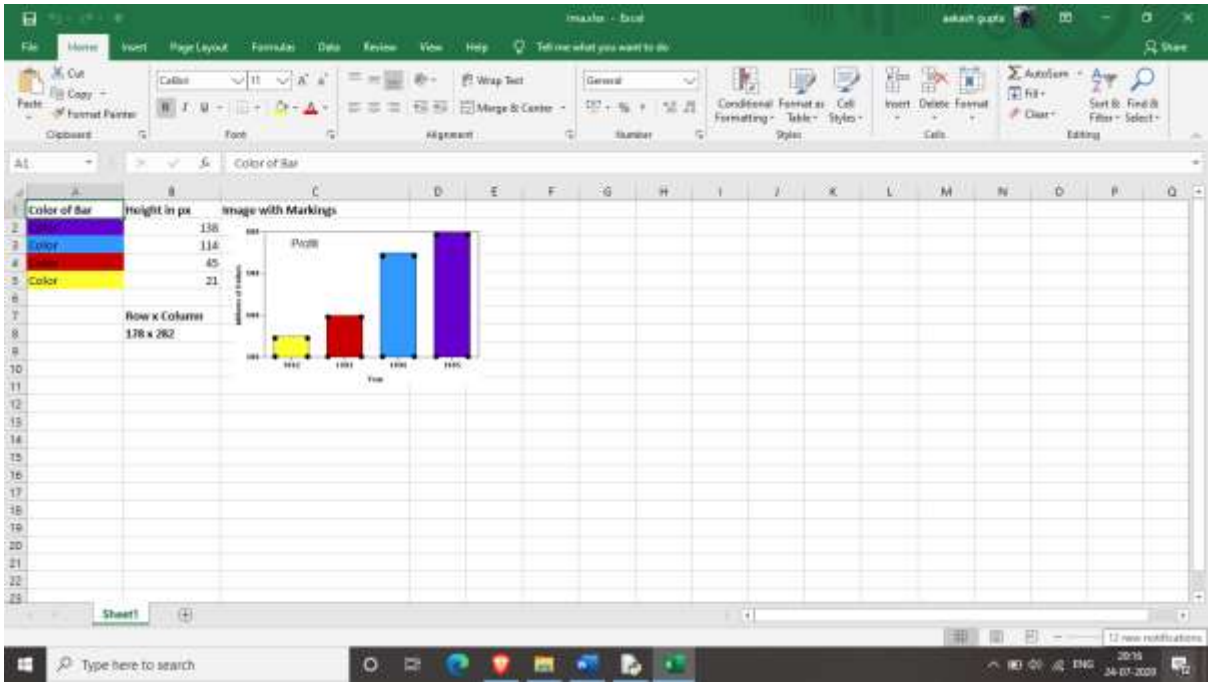

Fig. [12]: The output generated by the xlsxwriter library with the data from the Bar Graph Module.

9. Result

The result generated by each is of the module is analyzed in this chapter.

9.1 Results from the Bar Graph Module.

We shall consider the same example as Fig. [12]. The output Image is shown here in Fig. [13a]

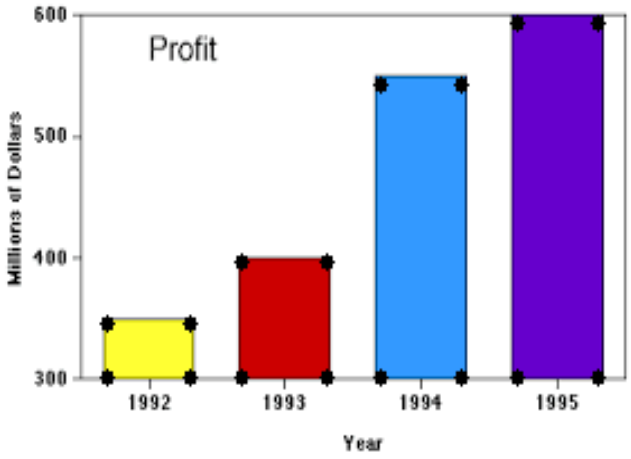

Fig. [13a] : The output image

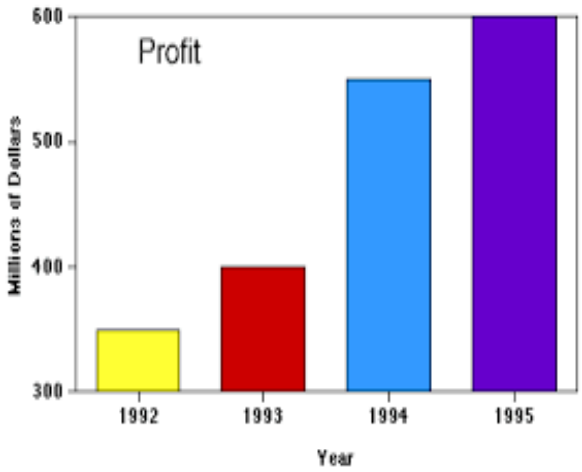

Fig. [13b]: The Image scanned by the module

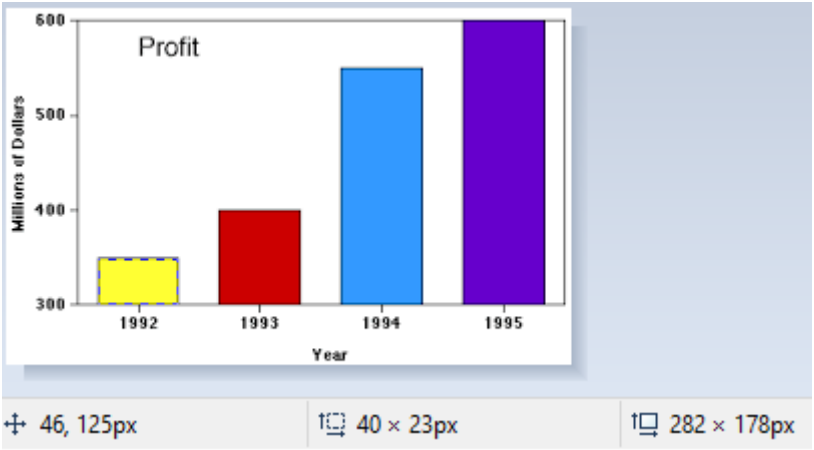

Fig. [14a] : Shows the dimensions for year 1992 by tracing are 40x23(Width x Height)

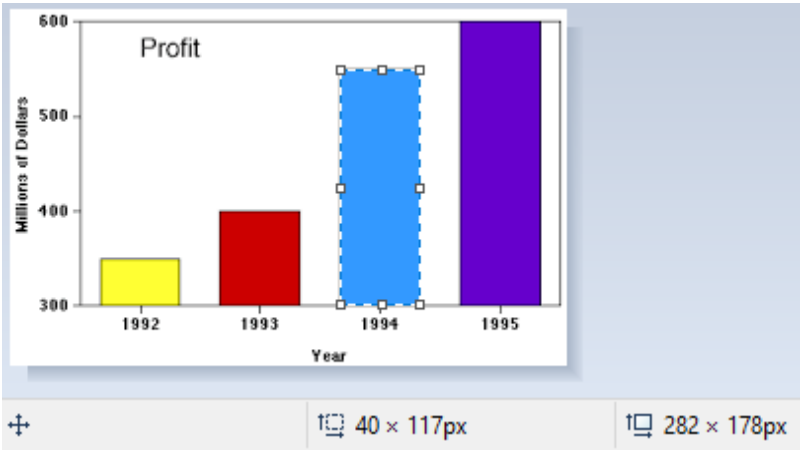

Fig. [14b] : Shows the dimensions for year 1992 by tracing are 40x117(Width x Height)

Categories	Actual value(Tracing)	Smarter Charts Result	Error
1992	23	21	8.69%
1993	47	45	4.25%
1994	117	114	2.56%
1995	141	138	2.13%

Table [2]: Error in the calculations of the heights by Smarter Charts

9.2 Results from the Pie Chart Module.

We shall consider the results, as shown in Fig. [15] for further calculations of the error in the predictions.

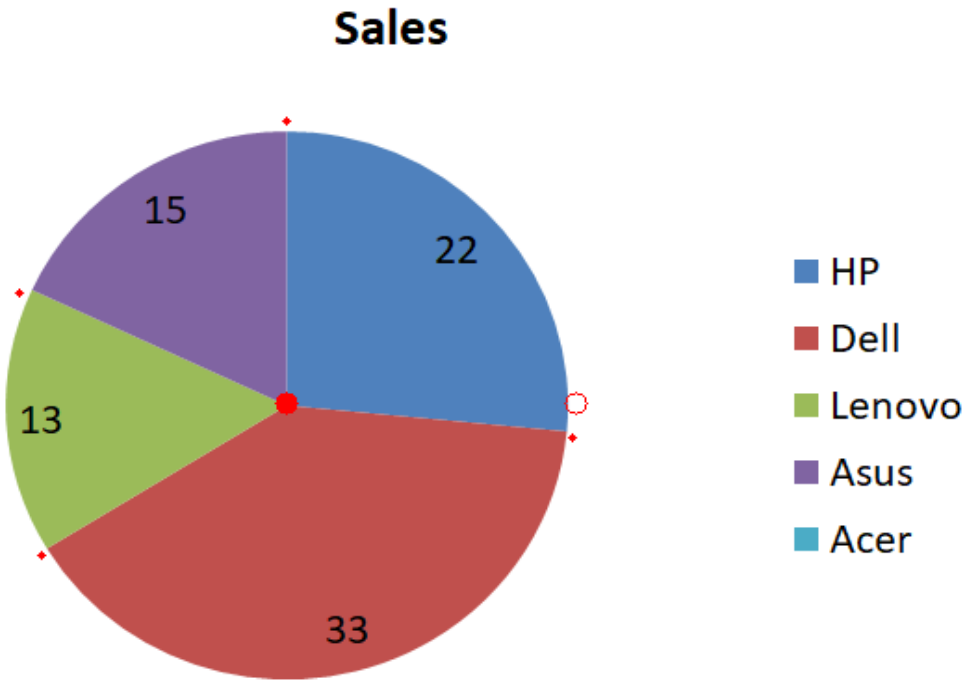

Fig. [15a] : The output image

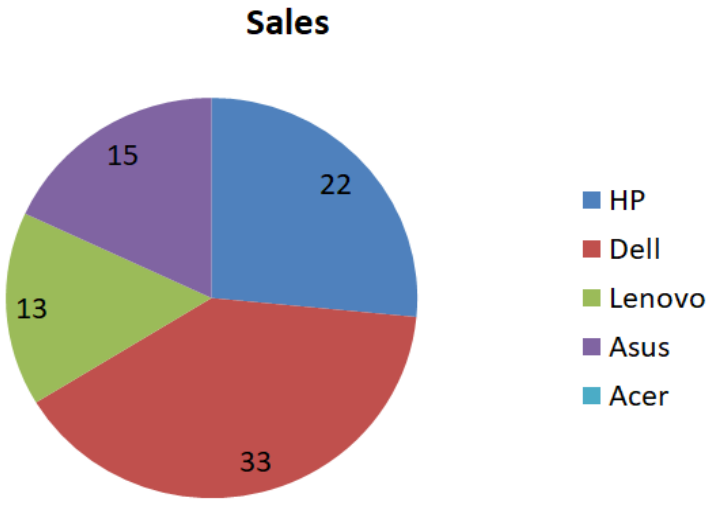

Fig. [15b] : The Image scanned by the module. Please note numbers in the image are not in percentage.

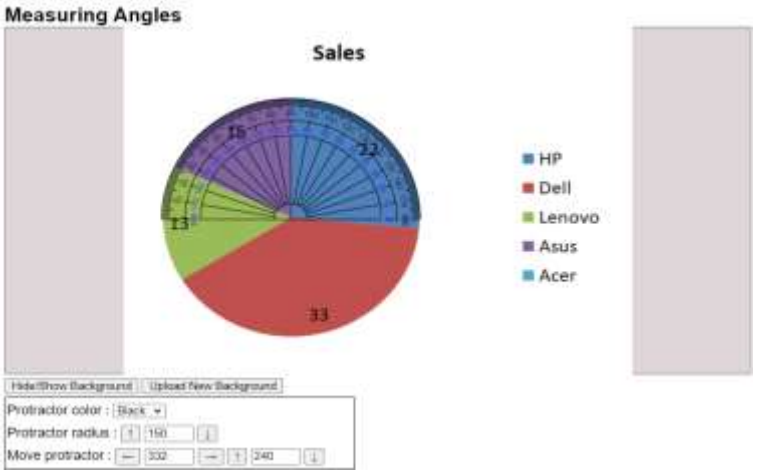

Fig. [16a]: Shows the angle subtended by Asus Category. Courtesy: Ursupplier.com

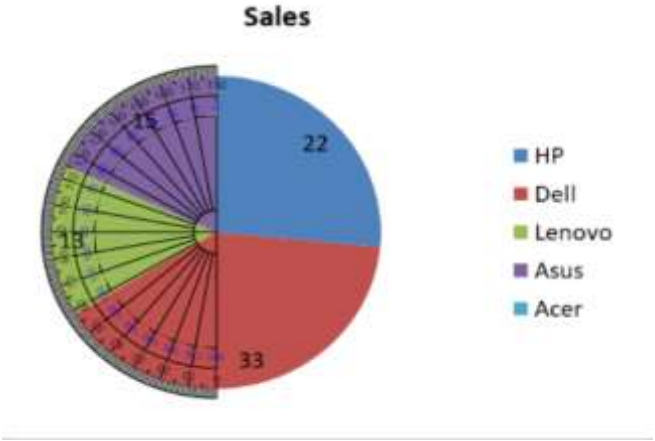

Fig. [16b]: Shows the angle subtended by Lenovo Category. Ursupplier.com

Categories	**Actual value(Tracing)**	**Smarter Charts Result**	**Error**
Asus	19.16%	18.68%	2.5%
Lenovo	15.83%	15.44%	2.52%
Dell	40%	38.84%	2.9%
HP	25.01%	27.04%	8.12%

Table [3]: Error in the calculations of the %age share by Smarter Charts

References

[1] K. Weiss et al., “A survey of transfer learning”, *Journal of Big Data* vol. 3, no. 9, May 2016.

[2] K. He et al., “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”, *2015 IEEE International Conference on Computer Vision*, Feb 2015.

[3] J. Gao et al., “Visual Information Extraction Widget for improving chart images accessibility”, *2012 19th IEEE International Conference on Image Processing*, Oct 2012.

[4] F. Marin et al., “WebPlotDigitizer, a polyvalent and free software to extract spectra from old astronomical publications: application to ultraviolet spectropolarimetry”, *French Society of Astronomy & Astrophysics*, Aug 2017.

[5] D. Jung et al., “ChartSense: Interactive Data Extraction from Chart Images”, *CHI Conference on Human Factors in Computing Systems*, May 2017.

[6] P De., “Automatic Data Extraction from 2D and 3D Pie Chart Images”, *2018 IEEE 8th International Advance Computing Conference (IACC)*, Dec 2018.