PUNYASHLOK AHILYADEVI HOLKAR SOLAPUR UNIVERSITY, SOLAPUR

A

PROJECT REPORT

on

"TEXT RECOGNITION USING MACHINE LEARNING"

In partial fulfillment of the requirements for the Degree in Bachelor of Computer Science & Engineering is being submitted to Punyashlok Ahilyadevi Holkar Solapur University, Solapur.

by

Student's name	Exam seat number
Yakkaldevi Prathmesh Ashok	1726385
Manekari Akshay Shriniwas	1726286
Patel Saitarun Venkateshwarrao	1726521
Shaikh Shahrukh Amulal	1726372
Gullapalli Shrikant Ganesh	1726422

under the guidance of

Prof. Maindargi L. C.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

N B NAVALE SINHGAD COLLEGE OF ENGINEERING, SOLAPUR

(2020-2021)

SPSPM'S

N B NAVALE SINHGAD COLLEGE OF ENGINEERING, SOLAPUR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Certificate

This is to certify that,

Student's name	Exam seat number
Yakkaldevi Prathmesh Ashok	1726385
Manekari Akshay Shriniwas	1726286
Patel Saitarun Venkateshwarrao	1726521
Shaikh Shahrukh Amulal	1726372
Gullapalli Shrikant Ganesh	1726422

have satisfactorily completed the project and submitted its report entitled

TEXT RECOGNITION USING MACHINE LEARNING

This work is being submitted in partial fulfillment of the degree of Bachelor of Computer Science & Engineering by Punyashlok Ahilyadevi Holkar Solapur University, Solapur in the academic year 2020-21.

Prof. Maindargi L. C.	Prof. A. A. Phatak	Dr. S. D. Nawale
GUIDE	**H.O.D**	**PRINCIPAL**

SPSPM'S

N B NAVALE SINHGAD COLLEGE OF ENGINEERING, SOLAPUR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PROJECT APPROVAL SHEET

PUNYASHLOK AHILYADEVI HOLKAR SOLAPUR UNIVERSITY, SOLAPUR

Project entitled

TEXT RECOGNITION USING MACHINE LEARNING

Submitted by

Student's name	Exam seat number
Yakkaldevi Prathmesh Ashok	1726385
Manekari Akshay Shriniwas	1726286
Patel Saitarun Venkateshwarrao	1726521
Shaikh Shahrukh Amulal	1726372
Gullapalli Shrikant Ganesh	1726422

is approved in partial fulfillment of the Degree of Bachelor of Computer Science & Engineering by Punyashlok Ahilyadevi Holkar Solapur University, Solapur.

Prof.---------
Guide/Internal Examiner

Prof.----
External Examiner

Date: 28th May 2021

Place: N.B Navale Sinhgad College of Engineering, Solapur

Acknowledgement

We express our sincere gratitude and indebtedness to our guide Prof. Maindargi L. C. for his valuable guidance, encouragement and affection for the successful completion of this project. His sincere sympathy and kind attitude always encouraged us to carry out the project at a steady pace.

We would also like to thank all teaching faculty for their kind support in our project.

Abstract

As we know, mankind has always tried to find a way to pass their knowledge to their future generations by recording their knowledge on books. But these books weather over time. As we move more towards the digital age, it would be more efficient if we store the recorded knowledge in digital form, as it will last longer than physical books.

We decided to compartmentalize the problem among our team members and started our work individually on each part.

Shahrukh was tasked with the creation of GUI, Shrikant was tasked with Cropping of the image, Akshay was tasked with B/W conversion of images, Prathmesh was tasked with cropping of each character from the B/W image, and Saitarun was tasked with creating a Classification model and storing the output in a text file.

We ran into many problems ranging from character cropping not doing an enough adequate job to trying different models with various characters being mispredicted, etc. Some of these problems still exist but at a minimal level.

We made 4 classification models with varying accuracy in addition to the introduction of threshold to the models. And we got an accuracy of 86.238% on manual testing on images which was by far the best among all the models in terms of readability and understanding provided the image was taken in a well-lit environment.

.

Contents

I. Introduction

Any software, despite how resourceful and faster it is at executing the desired task, will be in the little interest of the End User if that software is difficult and not convenient to use. This phenomenon can be observed with wide popularity of Windows operating system over the Linux based operating systems. The user's convenience can be fulfilled by the equipment of the software with an easy to use Graphical User Interface (GUI). Hence we have created a simple and easy to use GUI for End User to use our software/project who can be of any background and not necessarily from an IT background.

As we know, cropping is one of the most basic photo manipulation processes, and it is carried out to remove an unwanted object or irrelevant noise from the periphery of a photograph, to change its aspect ratio, or to improve the overall composition.
The image may contain unwanted sections, through the process of cutting we can eliminate this section.
The unwanted sections in the image can hamper the process of black and white image conversion which is the next step therefore it's better to eliminate those sections in the initial phase itself.

Furthermore, the next phase in the model is related to black and white image conversion. The process of predicting characters from the images becomes a lot easier if the images are in black and white instead of colour. The subsequent phase of character cutting embeds a technique that only works with B/W images.
This phase is directly correlated to all the other phases because the B/W image that is generated through this phase acts as input to all the next phases.

The next phase is to recognize characters from images which consist of text. To locate a character we can go by scanning images and locating characters

The final phase uses a classification model with 26 classes of small alphabets ‘a’ to ‘z’. For training, we used a dataset containing ~26k images of 26 classes.
This classification model classifies each image created by the previous phase and saves it in a document.

II. Literature Review

Artificial neural networks (**ANNs**), usually simply called **Neural Networks** (**NNs**), are computing systems vaguely inspired by the biological neural network that constitute animal brains.

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called *edges*. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

Components:

Neurons:

ANNs are composed of artificial neurons which are conceptually derived from biological neurons. Each artificial neuron has inputs and produces a single output which can be sent to multiple other neurons. The inputs can be the feature values of a sample of external data, such as images or documents, or they can be the outputs of other neurons. The outputs of the final *output neurons* of the neural net accomplish the task, such as recognizing an object in an image.

To find the output of the neuron, first we take the weighted sum of all the inputs, weighted by the *weights* of the *connections* from the inputs to the neuron. We add a *bias* term to this sum. This weighted sum is sometimes called the *activation*. This weighted sum is then passed through a (usually nonlinear) activation function to produce the output. The initial inputs are external data, such as images and documents. The ultimate outputs accomplish the task, such as recognizing an object in an image

Connections and weights:

The network consists of connections, each connection providing the output of one neuron as an input to another neuron. Each connection is assigned a weight that represents its relative importance. A given neuron can have multiple input and output connections.

Propagation function:

The *propagation function* computes the input to a neuron from the outputs of its predecessor neurons and their connections as a weighted sum. A *bias* term can be added to the result of the propagation

Neural Networks and Deep Learning:

Logistic Regression: The learning algorithm used when the output label *y* in supervised learning problems are either 0 or 1.

- Cost function: For calculating the difference between the output of the model and output label *y*.
- Gradient Descent: Minimizes the cost by updating *weight* and *bias*.

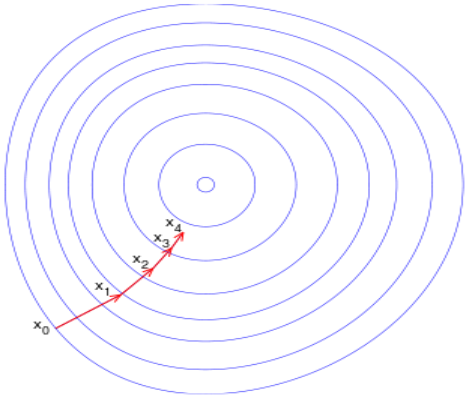


fig : Minimization of cost

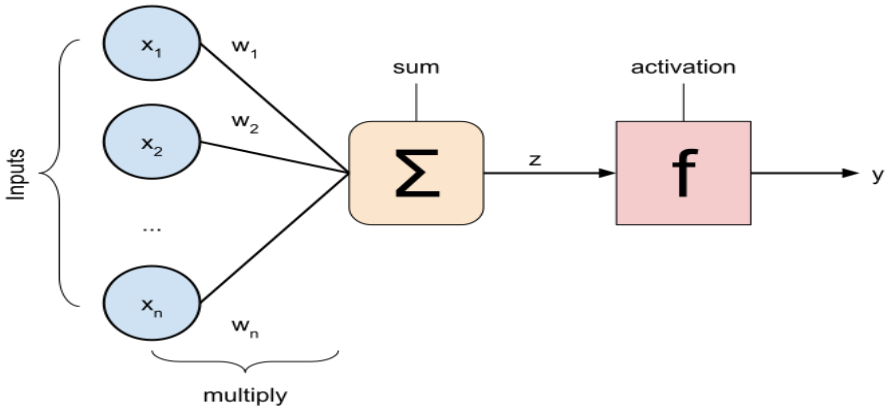


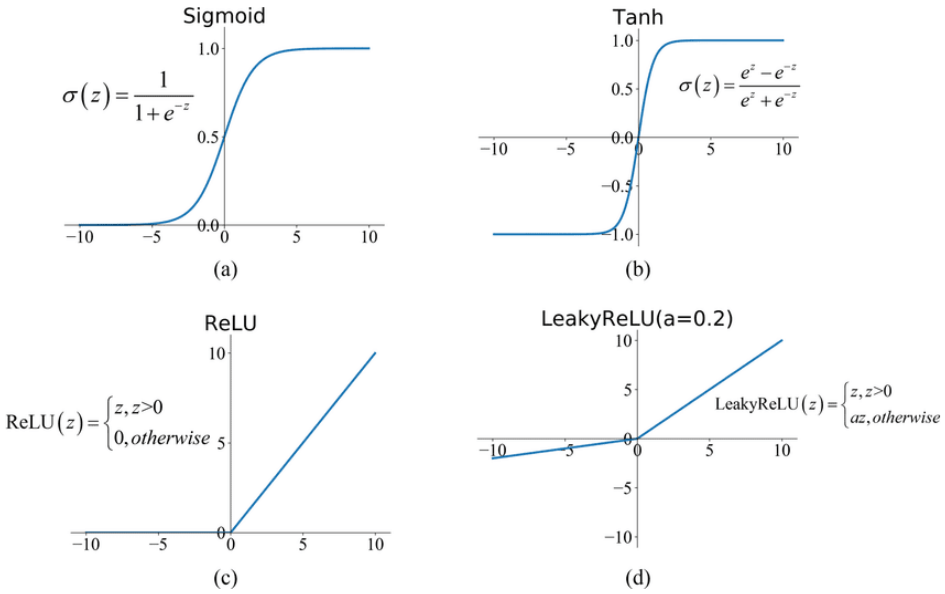
fig : Representation of a single Neuron

$$z = (weights(T) * input) + bias$$

$$a = y = \sigma(z)$$

Activation functions:

- Sigmoid : range 0 to 1
- tanH : range -1 to +1
- Relu : range 0 to +∞
- `Leaky Relu : range −∞ to +∞`



Vanishing and Exploding gradients:

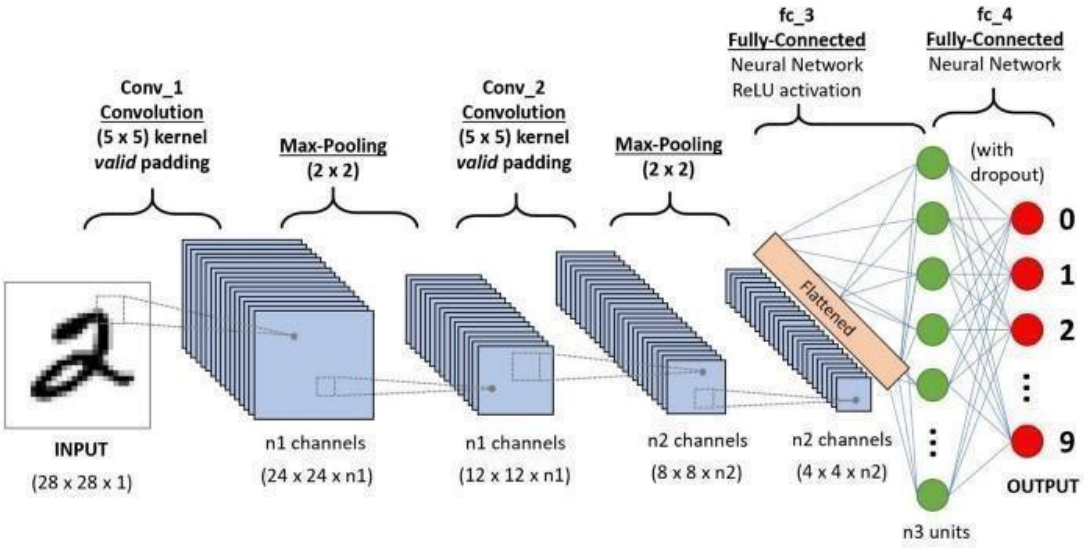
- One of the problems with training deep neural networks is that the derivatives or slopes can sometimes get either very large or very small.
- Solution to reduce Vanishing and Exploding gradients:
  - Reducing the number of Layers.
  - Random Initialization of weights.

Learning rate decay:

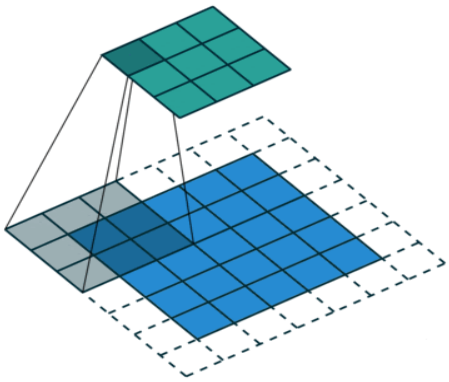
Learning rate decay is a technique for training modern neural networks. It starts with a large learning rate and then decays it multiple times. It is empirically observed to help both optimization and generalization.

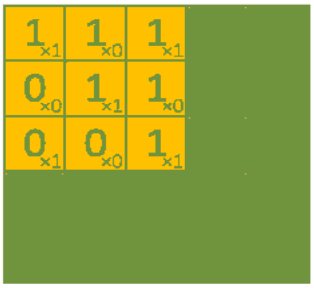
Convolutional Neural Networks:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

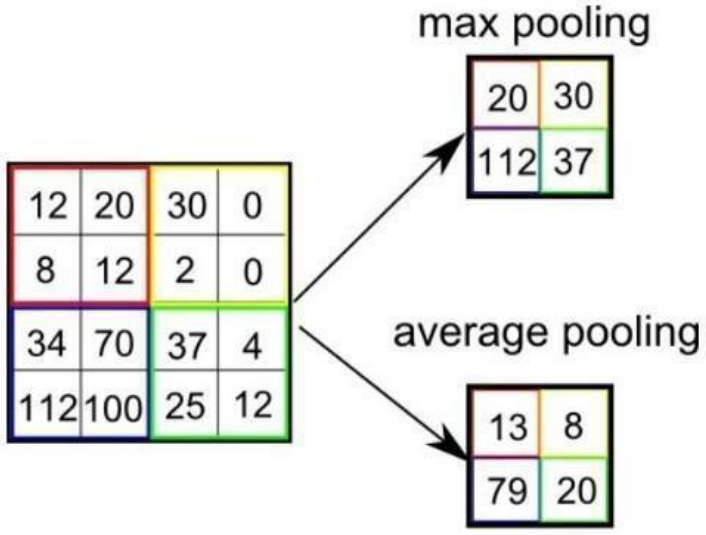


- Convolution layer: The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image





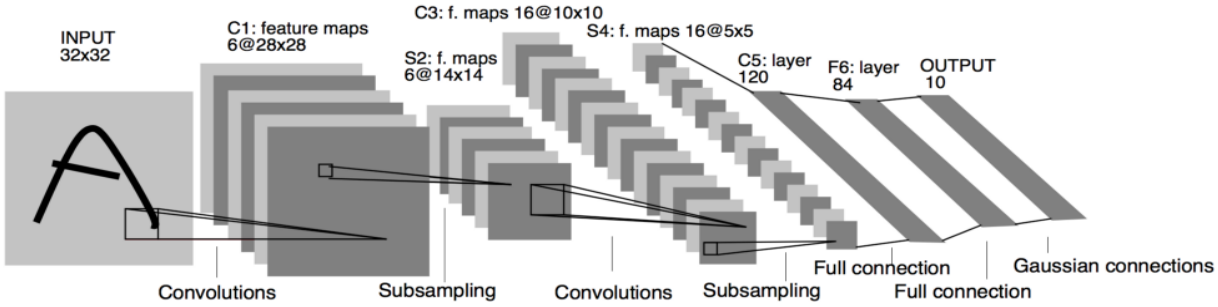
- Pooling layer: Similar to the Convolutional layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction.



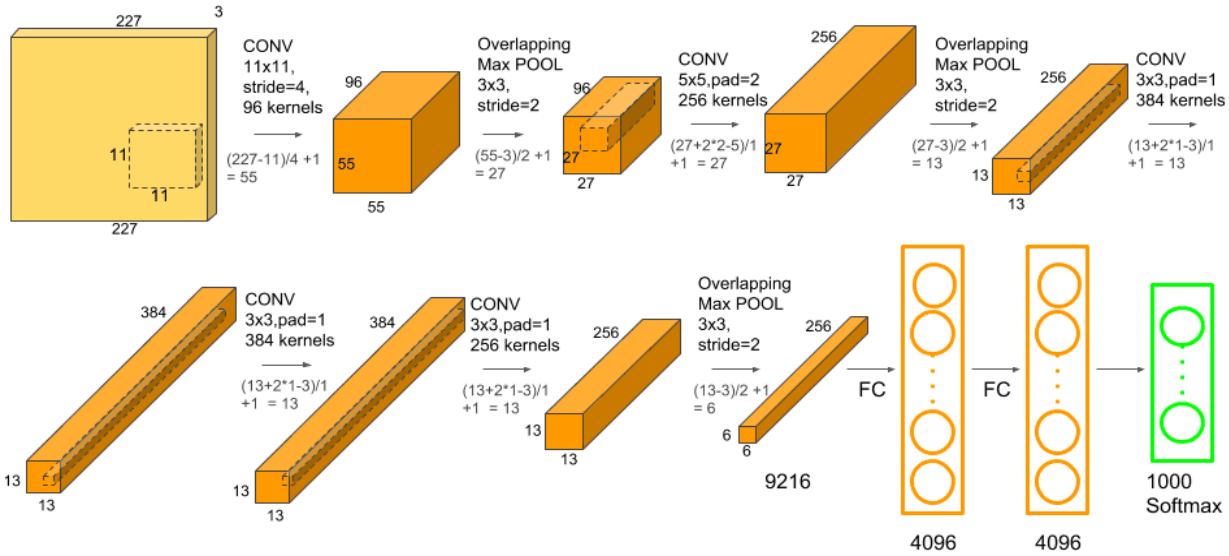
- Fully Connected layer(FC): Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

Case studies:

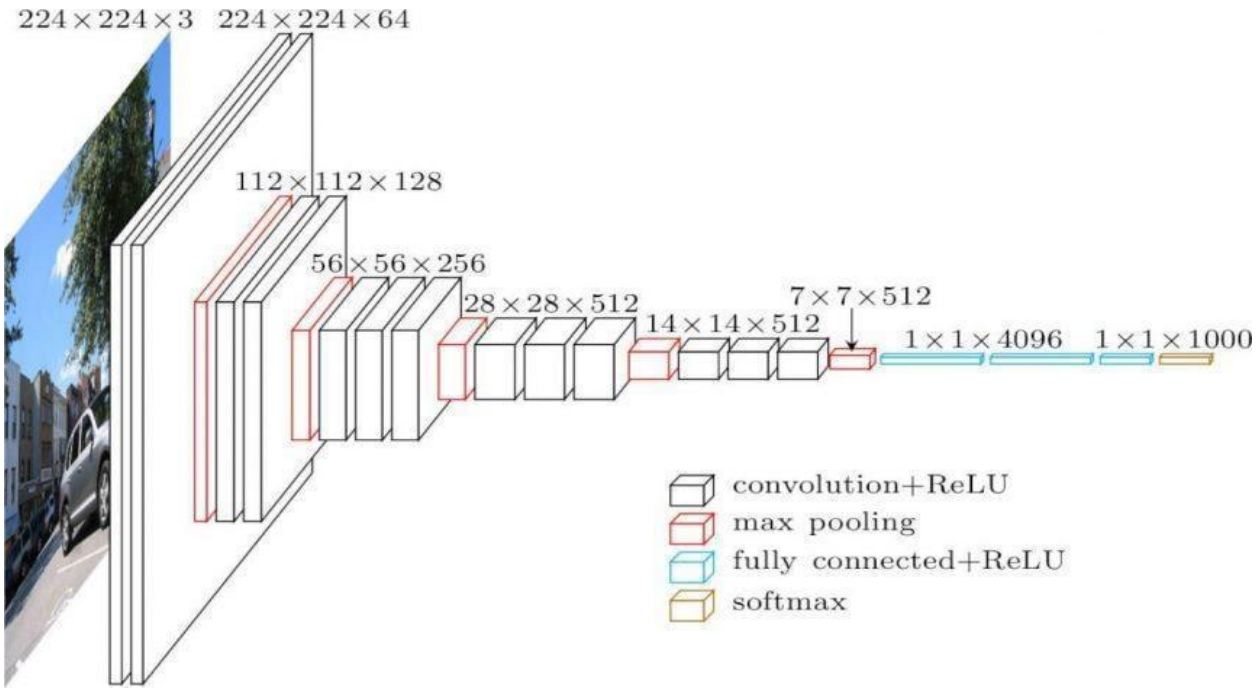
- **LeNet-5:** LeNet-5 CNN architecture is made up of 7 layers. The layer composition consists of three convolutional layers, two subsampling layers and two fully connected layers.
  - The goal of LeNet-5 was to detect handwritten digits.
  - It used Sigmoid activation for classification and tanH activation for previous layers.
  - It has about ~60k parameters.



- **AlexNet:** The architecture consists of eight layers: five convolutional layers and three fully-connected layers.
  - It uses the Softmax layer for classification and Relu activation for previous layers.
  - It has about ~60 million parameters.



- **VGG-16:** The architecture consists of sixteen layers: thirteen convolutional layers and three fully-connected layers.
    - It has about ~138 million parameters.

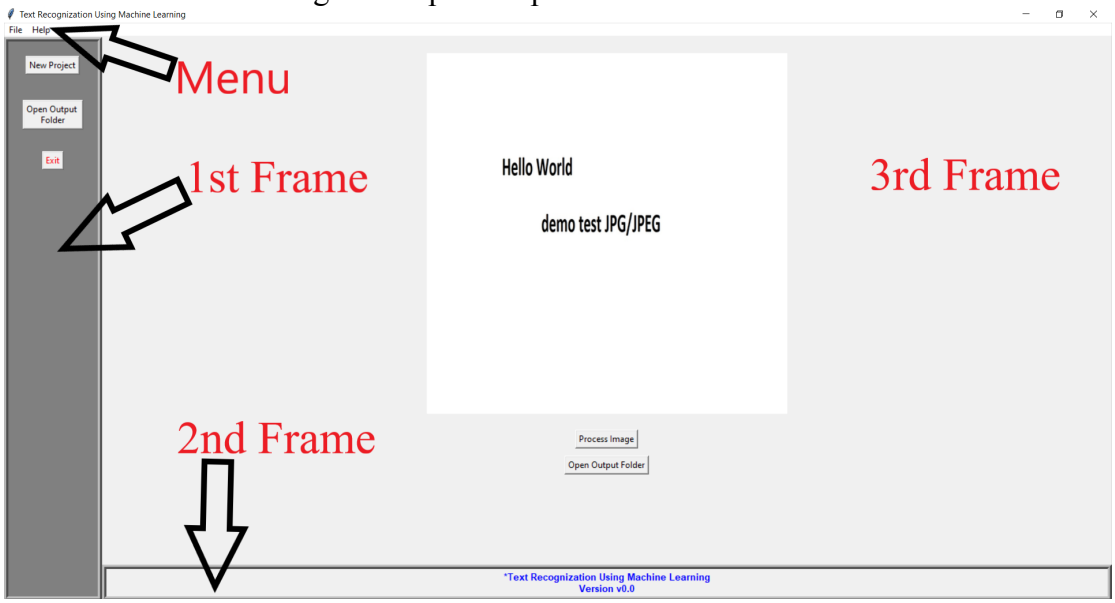


III. Methodology

In order to create a GUI for our project we searched for tools and platforms. The right set of tools and platforms were found in Python programming Language and its wide range of built-in and third party libraries.

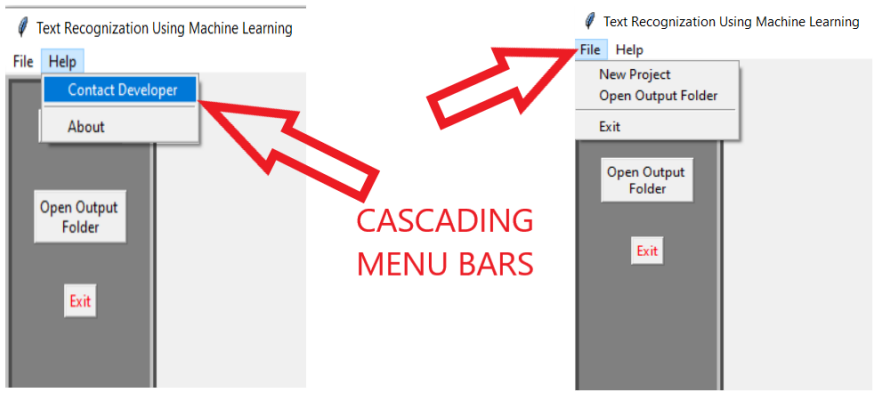
To develop our GUI we used python libraries such as Tkinter, OS, Subprocess and third party libraries such as Python Image Library(PIL). We have divided our GUI in Three Frames and Cascading Menu bar with Sub-menus. Following are the frames with their unique motive:

- Frame 1 : It is embedded with buttons viz. New Project, Open Output Folder and Exit.
- Frame 2 : It displays the software Version and required Details
- Frame 3 : On this Frame we can Import Images and we are successively presented with buttons viz. Process Image and Open Output Folder in accordance with user actions.



The below figure is actual screenshot of GUI and describes the Main Menu which is further divided into following Cascading Sub-menus:

- File Menu (with options New Project, Open Output Folder, Exit)
- Help Menu (with options Contact Developer, About)



Buttons and Menu Description :
File menu options and Buttons in frame one perform the same operations as theirs names suggest.

New Project – This Enables users to create a new project with its own workspace.

Open Output Folder – This Enables users to open the output folder to check for the result produced by the software.

Exit – It allows the user to quit the software with a confirmation prompt.

Help Menu – It has two options :

- Developers Contact – It provides users with the medium to contact the developer for any queries.
- About – This option provides users with information about the software.

Now moving on to the next part i.e. B/W conversion, as we know that images are nothing but collections of pixels.These pixels can be addressed by R,G,B values.The variation in the intensity of each R,G,B values can generate a new colour.
We took the R,G,B values as the heart of the conversion process.Also figured out that brightness is directly proportional to R,G,B values as a whole.

$$Brightness \propto BGR\ values$$

The technique works on the "comparison-difference" method. For this we took an image as ‘Base’ image and computed the average of all pixels.The averages of ‘Base’ image help us to categorize the newer input images into the 3 different brightness intensity sections:
Namely,
low light.
medium light.
bright light.

In the case of low light images; average results are very smaller R,G,B values(typically less than 50); for such images there is a need to push the average values higher to some extend.What we observed is; if we don't do like this; image loses details in the process of conversion.

In the case of bright light images;average results are very larger R,G,B values(typically greater than 180);for such images there is a need to push the average values lower to some extend.What we observed is; if we don't do like this; image unnecessarily generates noise(black pixels) in the process of conversion.

OpenCV is used for opening the image,saving the image during the process of B/W conversion.

Then we compare each pixel value with alpha values(discussed in the design calculation section) and make a decision on whether to activate that pixel or not.

At the end we get the B/W image.
Black and white images consist of lines of text, generally all lines are horizontal. So we can scan the horizontal lines of pixels and can determine if any particular horizontal line of pixels are in line of text or not. By cropping it we can get an image of each line of text. And now using the same technique, by scanning a vertical line of Pixels, the character can be separated from a line of text, the distance between each character can be used to determine if characters belongs to same word or different.

The threshold should be used, if the distance between characters exceeds threshold, then its beginning of a new word. By these we are able to first separate line of texts then words and character from image. The characters are converted to 128*128 dimensions and sharpened, because neural network will be trained for this dimension

In some cases where 2 characters overlap each other can be considered as one by above technique, for a characters which overlapped each other space without overlapping each other, we can scan the character image, onces we encounter Black Pixel, we can recursively move it to another blank image using flood fill algorithm. By this two characters can be separated. This technique will not work if both characters overlap each other.

By the end of this procedure, we will get images of all characters present in the input image.

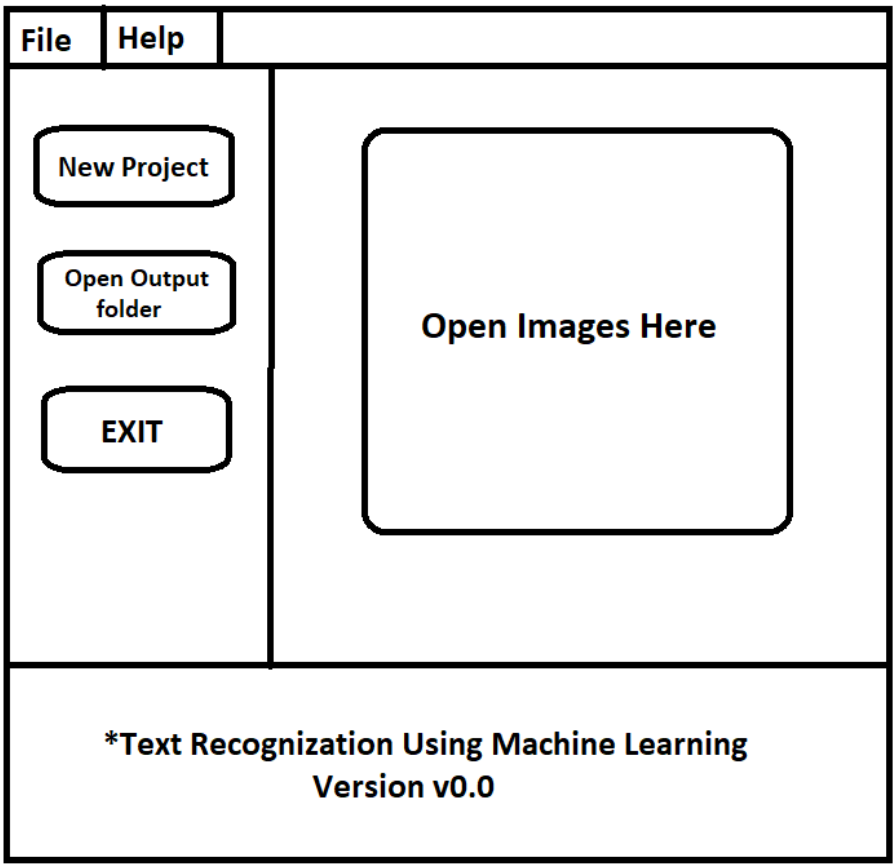
After the images are produced by the previous phase, these images are sent as input to the classification model one after another in sequence.

The classification model consists of 3 Convolutional Layers, 2 Dense Layers and an Output Layer with 26 outputs(1 each for each character). We use a dictionary variable called *word_dict* which maps each index to a character. The output of the Output Layer consists of probability/prediction of the character being present in the image. The index of the maximum value of the Output Layer is then used in combination with *word_dict* to get the character and save it in the output file.

This process is done for *n* images produced by the previous phase.

IV. Design Calculations

In order to create software that ensures the successful fulfillment of desired goals we need to go through the process of designing our software. The designing process enables us to map down our requirements, create future references and act in accordance. At the initial stage we thoroughly worked on how we wanted our GUI to be and focused on providing users with greater convenience and multiple ways for doing the desired task. We made use of drawing tools and came up with following GUI design:



Above figure represents the basic GUI we wanted to implement and was further the point of reference while actual implementation.

Tkinter is Python's standard GUI (graphical user interface) package. It is the most commonly used toolkit for GUI programming in Python.
Python when combined with Tkinter provides a fast and easy way to create GUI applications.
Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.we use two libraries here, namely TKinter and OpenCV.

OpenCV provides a facility to use the mouse as a paint brush or a drawing tool. Whenever any mouse event occurs on the window screen, it can draw anything. Mouse events can be left-button down, left-button up, double-click, etc. By using these coordinates, we can draw whatever we want.

Now moving on to the B/W conversion, manually after calculating the average values of R,G,B in the base image; it turned out to be (142,138,148).These values are further used to create formulas.

If we input a user image(it's the image which the user inputs to this program to get black and white image), the first step that we are doing is to calculate the delta values(Δ). An Intermediate image is an image which we used just for the purpose of calculations.

Delta values help us to see what's the difference between the avg. BGR values of user image and 'Base' image.For this base image, alphas can be (100,100,100). It means that the 'Base' image gets perfectly converted if alphas of B,G,R =(100,100,100).

$$\Delta B = \textit{average of B in User image} - \textit{average of B in Base image}$$
$$\Delta G = \textit{average of G in User image} - \textit{average of G in Base image}$$
$$\Delta R = \textit{average of R in User image} - \textit{average of R in Base image}$$

A subtle observation we made during this process is that if the average of *B* in the intermediary image increases by 20 then we need to increase the alpha value of *B* to "alpha value of *B* in base image + 20".
...(i)

In the case of *G*,if the average of *G* in the intermediary image increases by 34 then we need to increase the alpha value of *G* to "alpha value of *G* in base image + 22".
...(ii)

In the case of *R*,if the average of *R* in the intermediary image increases by 30 then we need to increase the alpha value of *R* to "alpha value of *R* in base image + 50".
...(iii)

The process now can be formulated by "If 20 increase is found in intermediary's Blue then we are increasing the alpha value by 40 and if the delta *B* is 50 in user image then how much should the alpha value be increased in the user image".

Formulating the above statement:

$$xB = \frac{40 * \Delta B}{20}$$

From such analogy we can derive xG, xR values.

$$xG = \frac{22 * \Delta G}{34} \qquad\qquad xR = \frac{50 * \Delta R}{30}$$

These values(xB, xG, xR) can either be positive or negative.

These xB,xG,xR values give us the change in proportions w.r.t the user image. We add these freshly generated values to the capping value of the ‘Base’ image. This is how we are finally left with alpha values.

We are adding here 100 to each alpha so that the alpha values get adjusted according to the alpha values of the base image.

$$alphaB = 100 + xB$$
$$alphaG = 100 + xG$$
$$alphaR = 100 + xR$$

We go pixel by pixel in the user image and decide whether to activate(turn black) the pixel or not as described in the below method.

```
if(B<=alphaB and G<=alphaG and R<=alphaR) :  #then set pixel to black
                                    sliced_image[i][j]=(0,0,0)
else:
                                    sliced_image[i][j]=(255,255,255)
```

Note:
B,G,R are values of an individual pixel.
sliced_image is an image which we are newly generating.
B,G,R=(0,0,0) means Black.
B,G,R=(255,255,255) means White.

Now to remove noise from images we need to do following Calculations

- Find object for shapes of smaller size
- Remove them.

For this we will get (5, 5) and (2, 2) size kernel (cv2.getStructuringElement) that can find (cv2.morphologyEx) the object of these sizes. And remove these objects.

To scale up the image,
Image of small size need to scale up to 128*128 dimensions. Images should not be distorted so we will determine the ratio between height and width of the image and then determine the width to scale up by dividing 128 by ratio we found earlier.

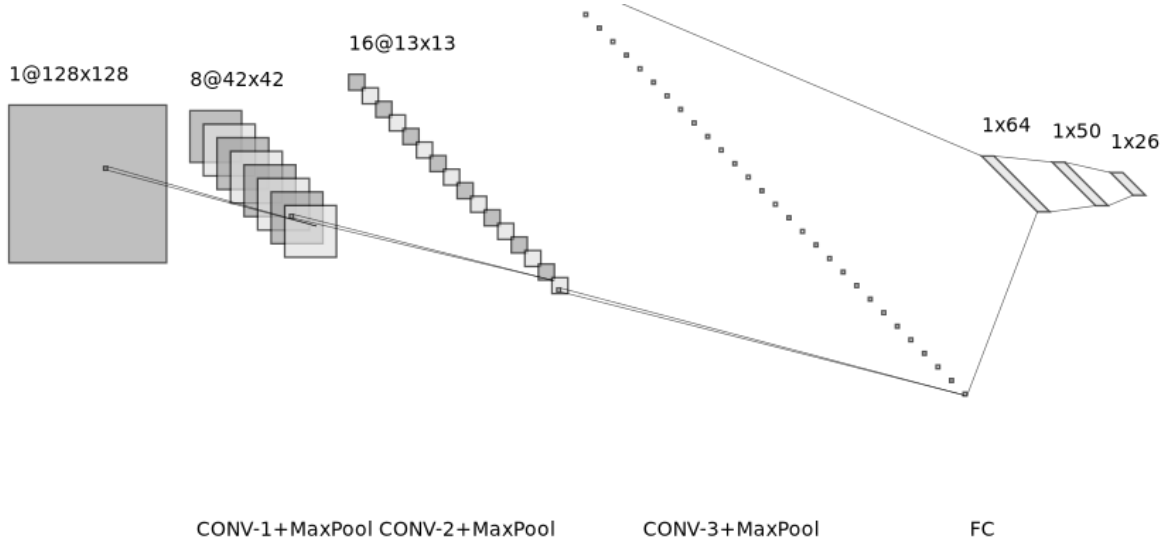
We will get height to be 128 but width will not be 128,so we simply add white Pixel to both side and make it 128*128 dimension image.

Now moving on to the classification model. We designed a classification model that takes input shape as (128, 128, 1)

ie (height, width, number of channels = 1)

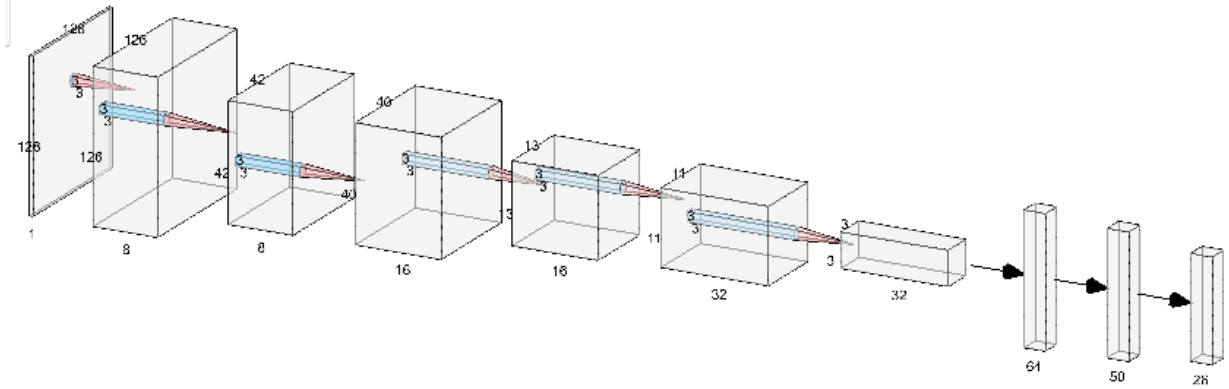
Here, *number of channels = 1*
Using Grayscale image instead of an RGB was efficient since the number of parameters (*weights*) were drastically reduced due to decrease in channels and an added benefit of this was increased speed of execution.

The structure of the classification model is,



(Fig: Architecture of Classification model)

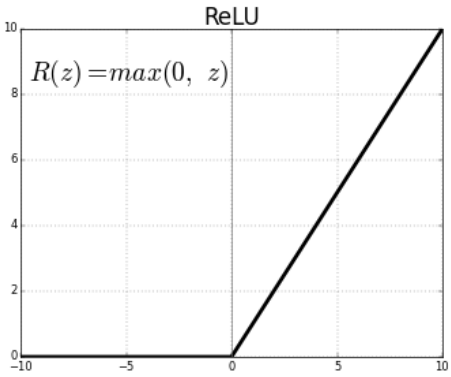
-



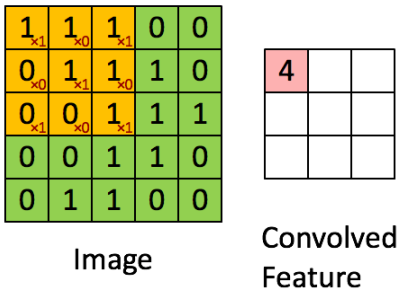
(Fig: Detailed Architecture of Classification Model)

Input Layer ->
Conv -> BatchNorm -> Activation(RELU) -> MaxPooling ->
Conv -> BatchNorm -> Activation(RELU) -> MaxPooling ->
Conv -> BatchNorm -> Activation(RELU) -> MaxPooling ->
Flatten ->
Dense ->
Dense ->
Output Layer

The first Conv layer takes input shape (128, 128, 1) & has 8 filters of size (3 x 3) which convolve over the input values to form an output of shape (126, 126, 8).
Followed by Batch Normalization to normalize the outputs.
Followed by Relu activation being applied to the output.



Followed by reducing the shape of output by using MaxPooling using a (3 x 3) filter to (42, 42, 8).



The second Conv layer takes input shape (42, 42, 8) & has 16 filters of size (3 x 3) which convolve over the input values to form an output of shape (40, 40, 16).
Followed by Batch Normalization to normalize the outputs.
Followed by Relu activation being applied to the output.
Followed by reducing the shape of output by using MaxPooling using a (3 x 3) filter to (13, 13, 16).

The Third Conv layer takes input shape (13, 13, 16) & has 32 filters of size (3 x 3) which convolve over the input values to form an output of shape (11, 11, 32).
Followed by Batch Normalization to normalize the outputs.
Followed by Relu activation being applied to the output.
Followed by reducing the shape of output by using MaxPooling using a (3 x 3) filter to (3, 3, 32).

Then the output of the MaxPool layer is flattened into a vector and passed to the first Fully Connected Layer(Dense Layer) which consists of 64 hidden units with Relu activation.
Followed by another Fully Connected Layer which consists of 50 hidden units with Relu activation.

Finally the output is passed to the final Output Layer which outputs a list containing 26 values, each a probability for individual characters ranging from *a* to *z*.

***Note**:* We created 4 models with varying epochs to test each model's accuracy and choose the best of those 4. All 4 models had training and test accuracy in the range of ~0.92. So choosing the model was down to checking which model was more accurate for the images produced by the previous phase.

After having chosen a suitable classification model i.e.*SmallAZVersion2*, we just load in it the memory when the software executes.
We iterate through each and image and get the 26 value list (i.e. Prediction values from Output Layer) and use the index of the maximum value with *word_dict* and output the character into the output file.

word_dict = { 1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f', 7: 'g', 8: 'h', 9: 'i', 10: 'j',
11: 'k', 12: 'l', 13: 'm', 14: 'n', 15: 'o', 16: 'p', 17: 'q', 18: 'r', 19: 's', 20: 't',
21: 'u', 22: 'v', 23: 'w', 24: 'x', 25: 'y', 26: 'z' }

Note: We also experimented with *threshold* values ranging from 2.5 to.10.0. We used this variable because we were getting inaccurate predictions in cases of multiple characters in a single image, no character and incomplete character in the image, etc.
After experimenting with different threshold values, we decided to choose 5.0 as a threshold.

i.e. If the maximum value of the output list is not greater than the threshold, then the character will not be saved in the output file.

V. Simulations/Experiments carried out

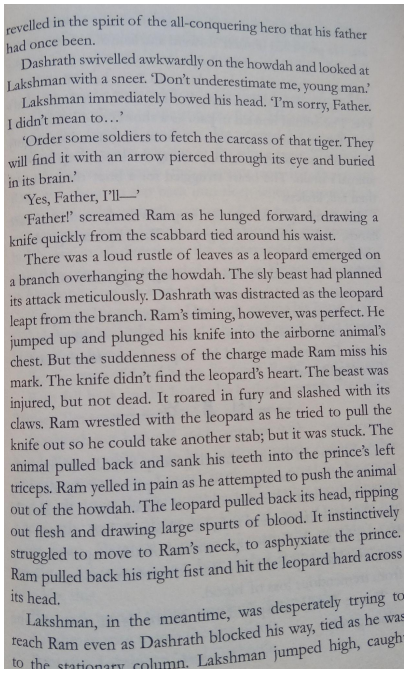
To process the mouse click events we define the

`click_and_crop`
The callback function is a mouse event that happens, OpenCV will relay the pertinent details to our `click_and_crop` function.

In order for our function to handle the relay, we need to accept 5 arguments:

- **event:** The event that took place (left mouse button pressed, left mouse button released, mouse movement, etc).
- **x:** The *x*-coordinate of the event.
- **y:** The *y*-coordinate of the event.
- **flags:** Any relevant flags passed by OpenCV.
- **params:** Any extra parameters supplied by OpenCV.

The experimentation of B/W code was carried through passing different images with different lighting conditions. The following are the results,



revelled in the spirit of the all-conquering hero that his father had once been.

Dashrath swivelled awkwardly on the howdah and looked at Lakshman with a sneer. 'Don't underestimate me, young man.'

Lakshman immediately bowed his head. 'I'm sorry, Father. I didn't mean to…'

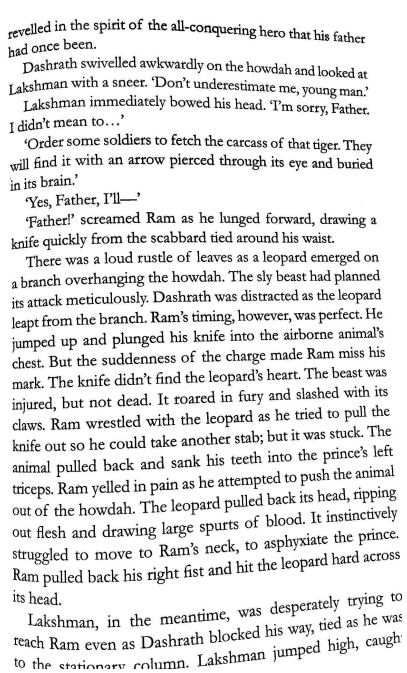
'Order some soldiers to fetch the carcass of that tiger. They will find it with an arrow pierced through its eye and buried in its brain.'

'Yes, Father, I'll—'

'Father!' screamed Ram as he lunged forward, drawing a knife quickly from the scabbard tied around his waist.

There was a loud rustle of leaves as a leopard emerged on a branch overhanging the howdah. The sly beast had planned its attack meticulously. Dashrath was distracted as the leopard leapt from the branch. Ram's timing, however, was perfect. He jumped up and plunged his knife into the airborne animal's chest. But the suddenness of the charge made Ram miss his mark. The knife didn't find the leopard's heart. The beast was injured, but not dead. It roared in fury and slashed with its claws. Ram wrestled with the leopard as he tried to pull the knife out so he could take another stab; but it was stuck. The animal pulled back and sank his teeth into the prince's left triceps. Ram yelled in pain as he attempted to push the animal out of the howdah. The leopard pulled back its head, ripping out flesh and drawing large spurts of blood. It instinctively struggled to move to Ram's neck, to asphyxiate the prince. Ram pulled back his right fist and hit the leopard hard across its head.

Lakshman, in the meantime, was desperately trying to reach Ram even as Dashrath blocked his way, tied as he was to the stationary column. Lakshman jumped high, caugh



revelled in the spirit of the all-conquering hero that his father had once been.

Dashrath swivelled awkwardly on the howdah and looked at Lakshman with a sneer. 'Don't underestimate me, young man.'

Lakshman immediately bowed his head. 'I'm sorry, Father. I didn't mean to…'

'Order some soldiers to fetch the carcass of that tiger. They will find it with an arrow pierced through its eye and buried in its brain.'

'Yes, Father, I'll—'

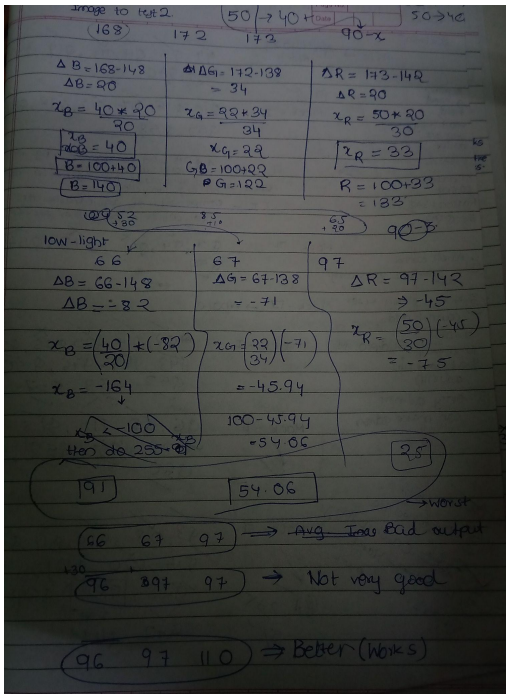
'Father!' screamed Ram as he lunged forward, drawing a knife quickly from the scabbard tied around his waist.

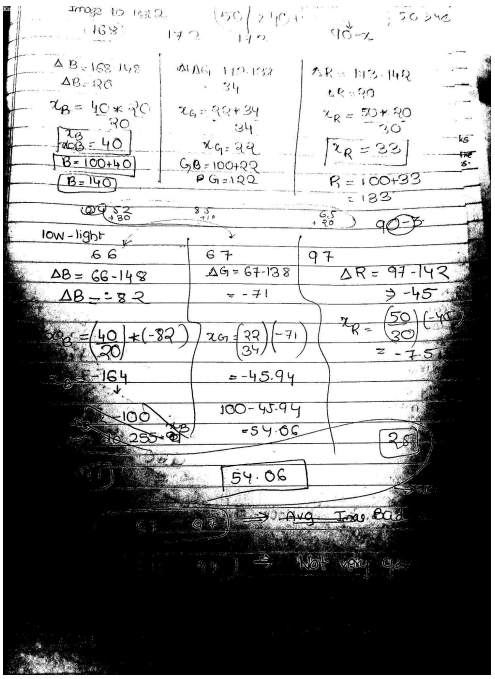
There was a loud rustle of leaves as a leopard emerged on a branch overhanging the howdah. The sly beast had planned its attack meticulously. Dashrath was distracted as the leopard leapt from the branch. Ram's timing, however, was perfect. He jumped up and plunged his knife into the airborne animal's chest. But the suddenness of the charge made Ram miss his mark. The knife didn't find the leopard's heart. The beast was injured, but not dead. It roared in fury and slashed with its claws. Ram wrestled with the leopard as he tried to pull the knife out so he could take another stab; but it was stuck. The animal pulled back and sank his teeth into the prince's left triceps. Ram yelled in pain as he attempted to push the animal out of the howdah. The leopard pulled back its head, ripping out flesh and drawing large spurts of blood. It instinctively struggled to move to Ram's neck, to asphyxiate the prince. Ram pulled back his right fist and hit the leopard hard across its head.

Lakshman, in the meantime, was desperately trying to reach Ram even as Dashrath blocked his way, tied as he was to the stationary column. Lakshman jumped high, caugh

Input Output

The input(image with Low light lighting condition)



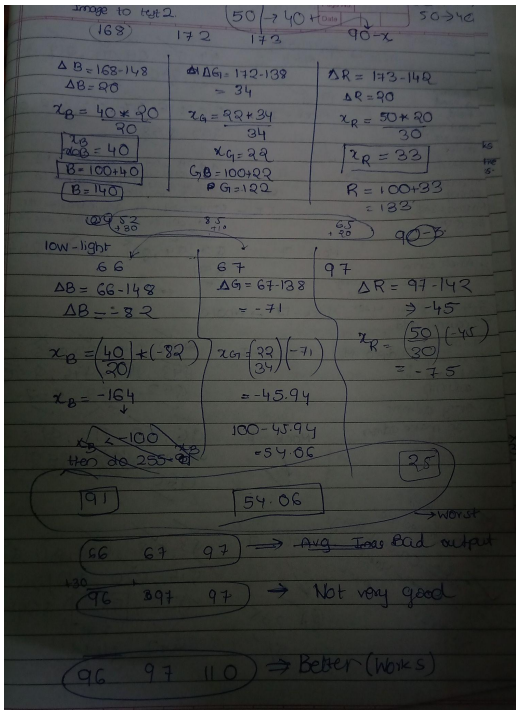


Input Output

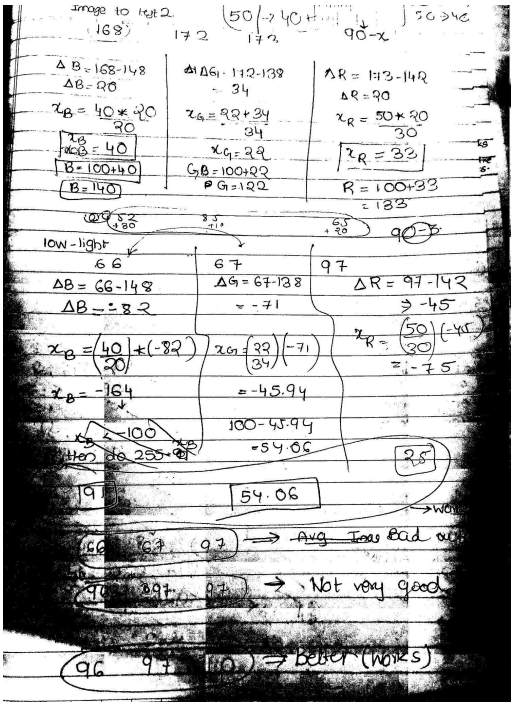
Unwanted black pixels do get resulted.This arises if alpha values go less than 0 or greater than 255.
A new modification to existing code was needed to push the negative values to the positive end.
The technique which we employed was,

1. Min(alphaB,alphaG,alphaR) is added equals to Max(alphaB,alphaG,alphaR)
2. Max(alphaB,alphaG,alphaR) is increased with the difference of (Max(alphaB,alphaG,alphaR)-Min(alphaB,alphaG,alphaR))/3
3. The in-between alpha value(the one which isn't max or min) is changed to average of max(alphaB,alphaG,alphaR) and min(alphaB,alphaG,alphaR)

Then after employing the previous step;the result improves to an extent.



Input



Output

Now the character cropping algorithm had some problems as well,
At first we tried to scale character image to 28*28 dimensions, and trained Neural Network for images of these dimensions. The Neural Network was not able to extract the useful features from the image, so we generated more dataset and trained our model on a bigger dataset, which was not enough.

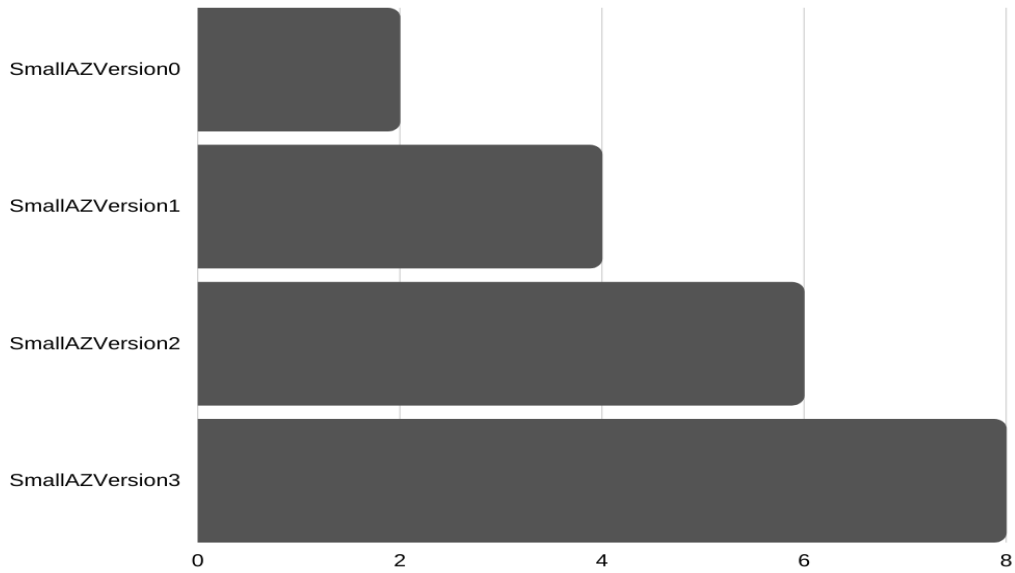
Then we trained Neural Network on 128*128 dimension image, and scaled the character image extracted from Input to 128*128 dimensions. While scaling up, we would lose data from it, the image had a Sharp corner and somewhat distorted at curves.
So after scaling, we need to Sharpen the image, we would first blur the image and then Sharpe it again. By doing this the character image was good enough for the classification model to be able to be recognizable..
But there was issue in some character images.in some images there was two character in 128*128 character image. We were separating characters from image of line of text by scanning it from left to right, we were looking if vertical line of Pixel was blank or not, so if two character overlap each other space then there won't be straight vertical blank line of pixels. Hence we get two characters in the same image. To tackle this issue we come up with the following idea.
Instead of straight vertical blank line of Pixel we can have flexible vertical blank line of pixel which can bend 2 or 3 Pixel to left or right, so if character overlap each's space by 2 or 3 Pixel width then we can separate it.
We wrote code for it. By this, we were able to separate two characters. But few characters were getting combined because the number pixel the flexible vertical blank line was bending. Some of the examples being the width of some characters like 'i', 'l', 'j'.

So we come up with another idea,
We can scan images from left to right and once we encounter Black Pixel, we can recursively move it to another blank image using the flood fill algorithm.
Using this, two characters can be separated.
This solution was good enough for tackling this issue. The disadvantage or issue with this idea was that if any character is broken then it would consider it as two separate characters

The solution was to use good technique for converting RGB input images to Black and white. Because while converting RGB images to black and white, some characters were breaking apart.

Classification model selection:

We made 4 models using the same training set for each model and trained each model for different amounts of epochs. Our aim for making 4 models was to find the most accurate model to be used in our project.



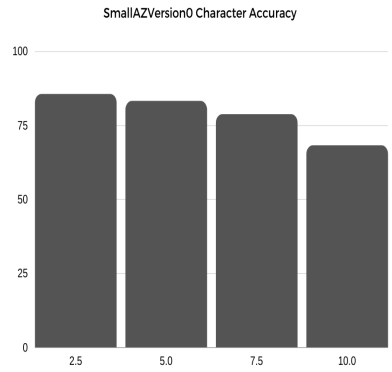
(fig.1)

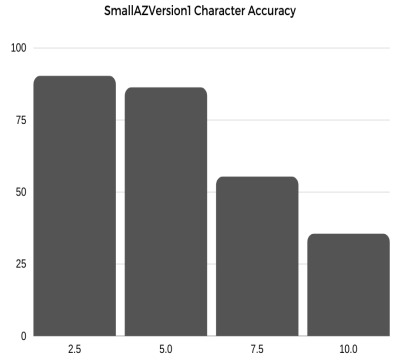
Here in fig.1,

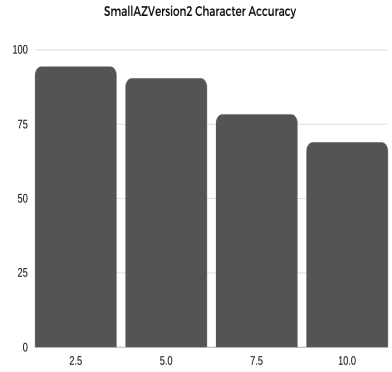
x axis represents the number of epochs trained.

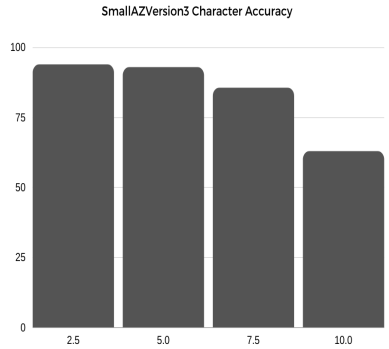
y axis represents the individual models.

After training 4 models, we experimented with threshold values to increase the accuracy because if there were any images with multiple characters or any other symbol other than the characters trained to the model, in theory their prediction values won't be higher than the threshold value.









Here in above graphs,

x axis represents the threshold values.
y axis represents the accuracy percentage.

Each model varies in its accuracy when the *threshold* is applied. Therefore for choosing a model, we ran some sample examples through each model at different *thresholds*. And the accuracy is displayed in the aforementioned graphs.

We noted that after training for >6 epochs the model would lose accuracy on the test set, this may be due to overfitting the model to the training set. So *SmallAZversion2* was giving the best accuracy before applying the *threshold.*

But we were getting special characters and other characters that were not trained on the model. Therefore we need to introduce a *threshold* variable to ignore those characters whose prediction values would not be over the *threshold*.

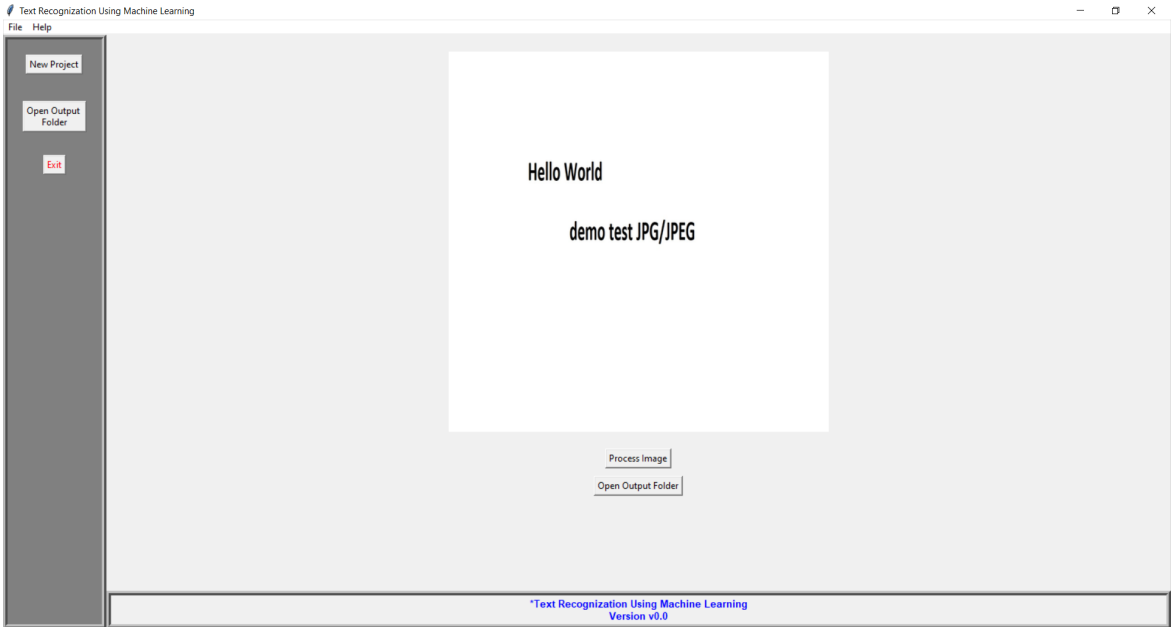
The *threshold* 2.5 and 5.0 and model *SmallAZVersion2* were giving consistent results compared to other models in regards to accuracy and readability of the output file.

After much consideration and discussion and cross checking output files of both threshold 2.5 and 5.0, we decided on using threshold 5.0.

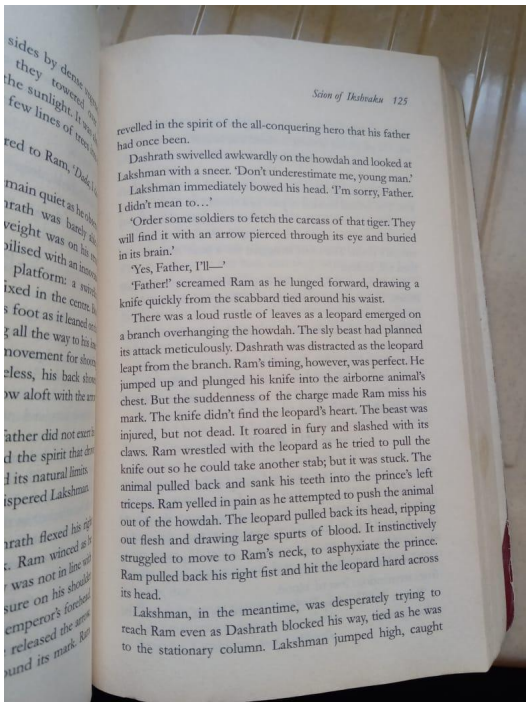
i.e. We decided to use SmallAZVersion2 with threshold=5.0.

VI. Result and Discussion

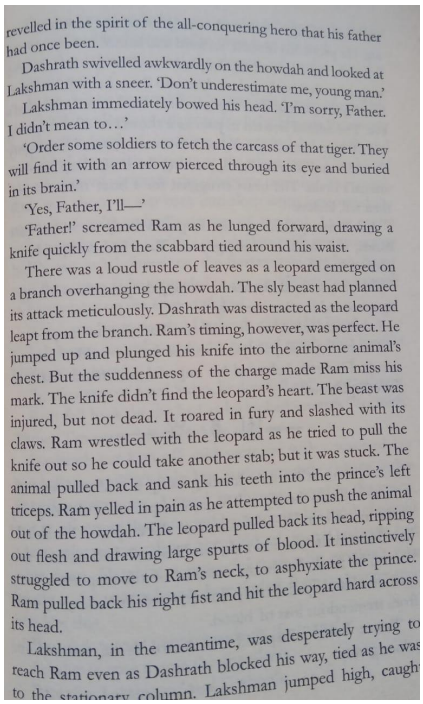
The result of our efforts of manufacturing a GUI for users' convenience was successful with the use of Python Programming Language as a means of development. The end product the GUI we Developed is best described by the figure below which is actual Screenshot of running the programmed GUI :



If we take a look at the performance of the cropping tool, it is competent enough to get the work done.



Input



Output

Character cropping wouldn't work if text is not horizontal or if the image contains black patches, to overcome this problem we can develop CNN to extract words from the image.

Classification Model *SmallAZVersion2* with combination with *threshold=5.0* seems to give the most effective blend between accuracy and readability of output files.

We can further improve this classification model by introducing more characters in its training sets and also testing with many more changes in hyperparameters during training.
Also it can be tested by adding more layers or reducing the layers and fiddling with the number of epochs trained.

VII. Conclusion

The convenience of using the software of the End User is of paramount importance which can be fulfilled with simple and easy to use Graphical User Interface (GUI). Although there are many ways for developing GUI the best/easiest way we found is using Python Programming Language which comes with an abundance of built-in and third party libraries.

The B/W image converter code has a hard time to deal with the “Low light and Bright light images”,So.the images need to be clicked in normal indoor lighting conditions and passed to the code.
Before giving an image to the code; we need to make sure that the image is properly cropped so that it helps to dampen the noise(unwanted black pixels).
From the technical front it can be concluded that the images with alpha values in the range (40 to 180) are good for the converter and can result in better outputs.

The classification model works well if proper images are provided to it, but still some inaccuracies may still persist while dealing with half visible characters, multiple characters in a single image, or plain inaccurate predictions like sometimes unable to distinguish between ‘b’ and ‘h’ or ‘e’ and ‘o’ when these characters are distorted in shape.

VIII. References / Bibliography

- Deep learning Specialization by AndrewNg：https://www.coursera.org/specializations/deep-learning
- Training set: http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/
- Grokking Deep Learning - Andrew Trask
- https://colab.research.google.com/
- http://alexlenail.me/NN-SVG/AlexNet.html
- http://alexlenail.me/NN-SVG/LeNet.html
- https://www.researchgate.net/figure/Commonly-used-activation-functions-a-Sigmoid-b-Tanh-c-ReLU-and-d-LReLU_fig3_335845675
- https://www.researchgate.net/figure/Commonly-used-activation-functions-a-Sigmoid-b-Tanh-c-ReLU-and-d-LReLU_fig3_335845675
- https://www.researchgate.net/figure/Commonly-used-activation-functions-a-Sigmoid-b-Tanh-c-ReLU-and-d-LReLU_fig3_335845675
- https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
- https://towardsdatascience.com/the-vanishing-exploding-gradient-problem-in-deep-neural-networks-191358470c11
- https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
- https://medium.com/@pechyonkin/key-deep-learning-architectures-lenet-5-6fc3c59e6f4
- https://neurohive.io/en/popular-networks/alexnet-imagenet-classification-with-deep-convolutional-neural-networks/
- https://neurohive.io/en/popular-networks/vgg16/