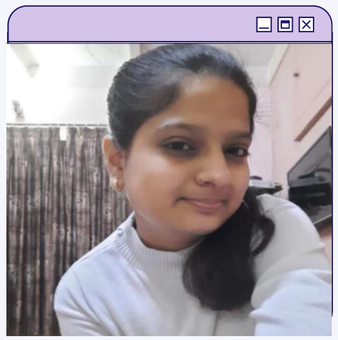
GLA UNIVERSITY
MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC

IBM
Innovation Centre for Education
B.Tech (CSE) programs
in collaboration with
GLA University, Mathura

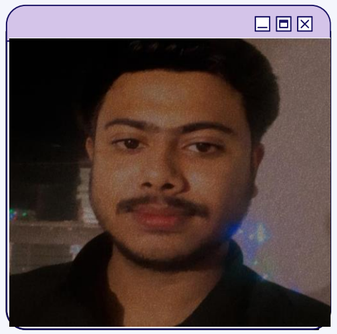
Cartoonify image with ML

Supervision by: Ankur Chaturvedi



Etisha Jain

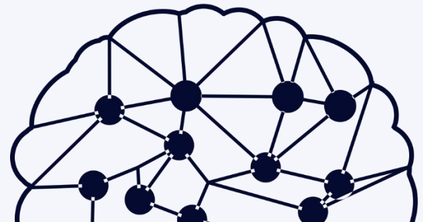
Harshit Garg



Arnav Srivastava

Vishu Goyal

Index of Presentation

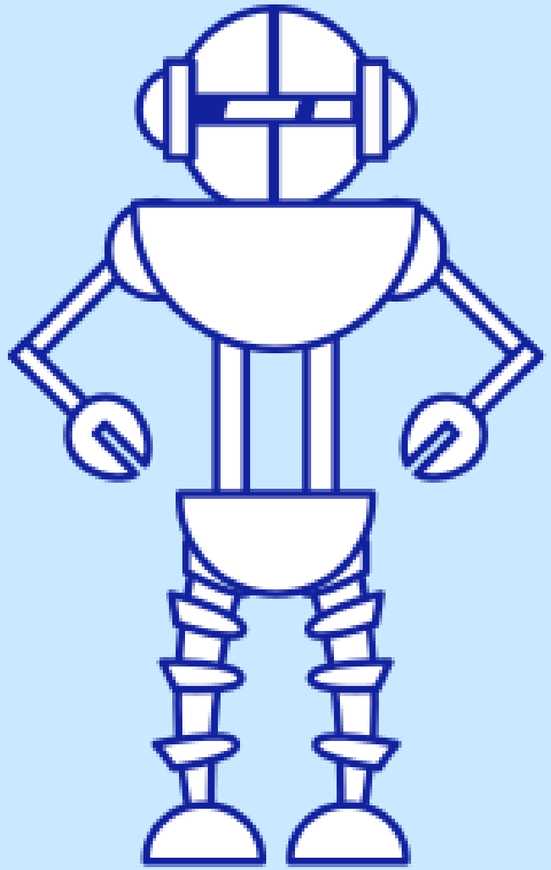


Introduction

Uses or Applications of Topic in Real Life Scenario

SnapChat

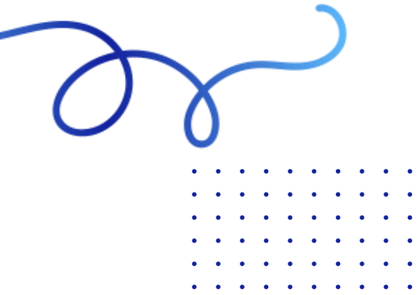
Selfie



Instagram

Animation

This robot icon can be replaced based on your theme of the paper



Steps to create the Cartoonifier app
Step 1: Importing the required modules

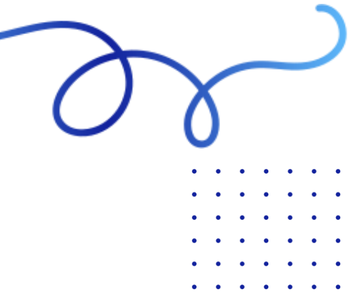
```
# CODE:

import tkinter as tk  # graphical user interface toolkit
from tkinter import *
import easygui  # to open the filebox
import cv2  # for image processing
import matplotlib.pyplot as plt

import os  # to read and save path
import sys
```

Step 2: Making the GUI main window

```
# CODE:

top = tk.Tk()
top.geometry('400x400')
top.title('Cartoonify Your Image !')
top.configure(background='blue')
label = Label(top, background='#CDCDCD', font=('calibri', 20, 'bold'))
```

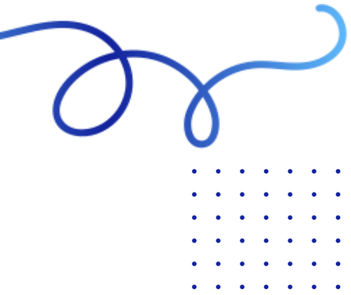
Step 3: Create a File Box to choose a particular file

```
# CODE:

""" fileopenbox opens the box to choose file
       and help us store file path as string"""

def upload():
   image_path = easygui.fileopenbox()
   cartoonify(image_path)
```



Step 4: Making the Cartoonify button in the GUI main window

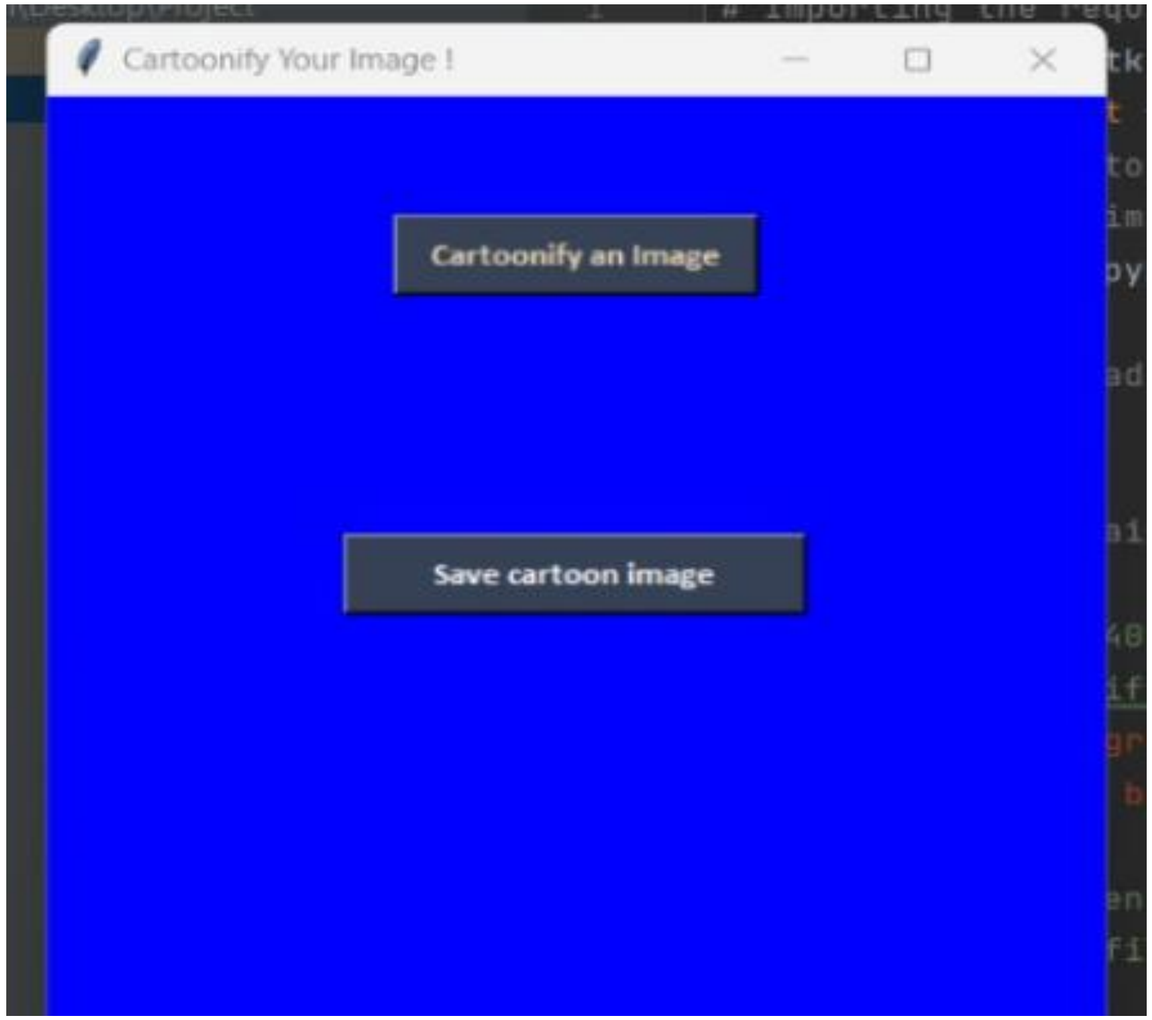
```
# CODE:

upload = Button(top, text="Cartoonify an Image", command=upload,
padx=10, pady=5)
upload.configure(background="#374256", foreground="wheat", font=
('calibri', 10, 'bold'))
upload.pack(side=TOP, pady=50)
```



Step 5: Making a Save button in the GUI main window

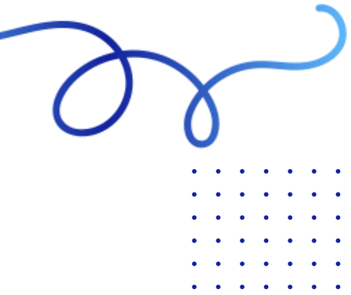
```
# CODE:

savel = Button(top, text="Save cartoon image", command=lambda:
save(resize_image6, image_path), padx=30, pady=5)
savel.configure(background='#364156', foreground='white', font=
('calibri', 10, 'bold'))
savel.pack(side=TOP, pady=50)
```

Step 6: Main function to build the GUI window



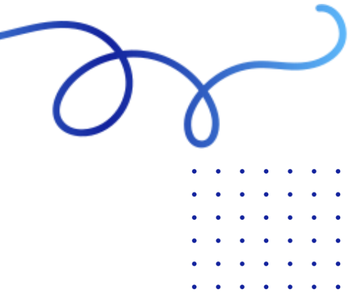
Step 7: How is an Image stored?

```
# CODE:

def cartoonify(image_path):
        # read image
   original_image = cv2.imread(image_path)
   original_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)
   print(original_image) # this will be stored in form of number
# to confirm it is image that was chosing
   if original_image is None:
      print("Can't find any image. Choose appropriate file")
      sys.exit()

      #resize the image after each transformation
   resize_image1 = cv2.resize(original_image, (960, 540))
   plt.imshow(resize_image1, cmap='gray')
```



Step 8: Image transformation to grayscale

```
# CODE:

# converting an image to grayscale
 grayscale_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2GRAY)

 resize_image2 = cv2.resize(grayscale_image, (960, 540))
 plt.imshow(resize_image2, cmap="gray")
```



Step 9: Smoothening a grayscale image

```
# CODE:

# applying median blur to smoothen an image
smooth_grayscale_image = cv2.medianBlur(grayscale_image, 5)

resize_image3 = cv2.resize(smooth_grayscale_image, (960, 540))
plt.imshow(resize_image3, cmap='gray')
```

Step 10: Extracting the edges in the image (Highlighted Edges)

```
# CODE:

# retrieving the edges for cartoon effect
# by using thresholding technique
get_edge = cv2.adaptiveThreshold(smooth_grayscale_image, 255,
cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 9, 9)

resize_image4 = cv2.resize(get_edge, (960, 540))
plt.imshow(resize_image4, cmap='gray')
```

Step 10b: Cartoon effect specialities:- Smooth colours

```
# CODE:

# applying bilateral filter to remove noise
# and keep edge sharp as required
color_image = cv2.bilateralFilter(original_image, 9, 300, 300)

resize_image5 = cv2.resize(color_image, (960, 540))
plt.imshow(resize_image5, cmap="gray")
```

Step 11: Giving a Cartoon Effect

```
# CODE:

# masking edged image with our “BEAUTIFY” image
 cartoon_image = cv2.bitwise_and(color_image, color_image,
mask=get_edge)

 resize_image6 = cv2.resize(cartoon_image, (960, 540))
 plt.imshow(resize_image6, cmap='gray')
```

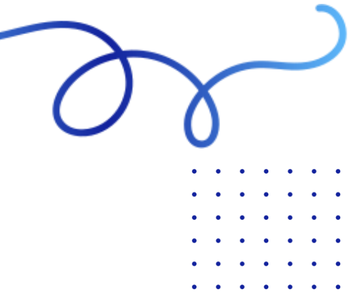
Step 12: Plotting all the transitions together

```
# CODE:

# Plotting the whole transition
 images = [resize_image1, resize_image2, resize_image3, resize_image4,
resize_image5, resize_image6]
 fig, axes = plt.subplots(3, 2, figsize=(8, 8), subplot_kw =
{'xticks': [], 'yticks': []}, gridspec_kw=dict(hspace=0.1,
wspace=0.1))
 for i, ax in enumerate(axes.flat):
    ax.imshow(images[i], cmap='gray')

# save button code
plt.show()
```

Step 12: Plotting all the transitions together





Step 13: Enhancing the save button

```
# CODE:

def save(resize_image6, image_path):
 # saving an image using imwrite function
   new_name = “cartoonified_Image”
   path1 = os.path.dirname(image_path)
   extension = os.path.splitext(image_path)[1]
   path = os.path.join(path1, new_name + extension)
   cv2.imwrite(path, cv2.cvtColor(resize_image6, cv2.COLOR_RGB2BGR))
   I = “Image saved by name “ + new_name + “ at “ + path
   tk.messagebox.showinfo(title=None, message=I)
```

Experiments and Results

Conclusion and Future Work

Summary

We have successfully developed Image Cartoonifier with OpenCV in Python. This is the magic of OpenCV which let us do wonder and it is just a piece of what openCV can offer.

Future Work

References

[1] Youtube

[2] medium.com

[3] StackOverFlow

[4] Python.org

[5] openCV.org

Any Queries

Thank You