**Practical Assignment**

**Objective: - Coloring old Black&White photos**

Automated image colorization of B&W images has been a hot topic of exploration in the field of computer vision.

**Dataset Link: -**

Use anyone of your choice.

**Task: -** Create a Web Application using FASTAPI. Use the end user should be able to upload an B&W image and get color photos .

**Assignment Submission: -** Only submit the GitHub Link. Create a proper Readme documentation.

#!/usr/bin/env python

# coding: utf-8

# In[6]:

# import the necessary packages

import numpy as np

import cv2

import streamlit as st

from PIL import Image

def colorizer(img):

img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

img = cv2.cvtColor(img, cv2.COLOR\_GRAY2RGB)

# load our serialized black and white colorizer model and cluster

# center points from disk

#Note: Please take in account the directories of your local system.

prototxt = r"C:\Users\dhananjayan\projects\Colorizer\models\models\_colorization\_deploy\_v2.prototxt"

model = r"C:\Users\dhananjayan\projects\Colorizer\models\colorization\_release\_v2.caffemodel"

points = r"C:\Users\dhananjayan\projects\Colorizer\models\pts\_in\_hull.npy"

net = cv2.dnn.readNetFromCaffe(prototxt, model)

pts = np.load(points)

# add the cluster centers as 1x1 convolutions to the model

class8 = net.getLayerId("class8\_ab")

conv8 = net.getLayerId("conv8\_313\_rh")

pts = pts.transpose().reshape(2, 313, 1, 1)

net.getLayer(class8).blobs = [pts.astype("float32")]

net.getLayer(conv8).blobs = [np.full([1, 313], 2.606, dtype="float32")]

# scale the pixel intensities to the range [0, 1], and then convert the image from the BGR to Lab color space

scaled = img.astype("float32") / 255.0

lab = cv2.cvtColor(scaled, cv2.COLOR\_RGB2LAB)

# resize the Lab image to 224x224 (the dimensions the colorization

#network accepts), split channels, extract the 'L' channel, and then perform mean centering

resized = cv2.resize(lab, (224, 224))

L = cv2.split(resized)[0]

L -= 50

# pass the L channel through the network which will \*predict\* the 'a' and 'b' channel values

net.setInput(cv2.dnn.blobFromImage(L))

ab = net.forward()[0, :, :, :].transpose((1, 2, 0))

# resize the predicted 'ab' volume to the same dimensions as our input image

ab = cv2.resize(ab, (img.shape[1], img.shape[0]))

# grab the 'L' channel from the \*original\* input image (not the

# resized one) and concatenate the original 'L' channel with the predicted 'ab' channels

L = cv2.split(lab)[0]

colorized = np.concatenate((L[:, :, np.newaxis], ab), axis=2)

# convert the output image from the Lab color space to RGB, then clip any values that fall outside the range [0, 1]

colorized = cv2.cvtColor(colorized, cv2.COLOR\_LAB2RGB)

colorized = np.clip(colorized, 0, 1)

# the current colorized image is represented as a floating point

# data type in the range [0, 1] -- let's convert to an unsigned 8-bit integer representation in the range [0, 255]

colorized = (255 \* colorized).astype("uint8")

# Return the colorized images

return colorized

##########################################################################################################

st.write("""

# Colorize your Black and white image

"""

)

st.write("This is an app to turn Colorize your B&W images.")

st.write("Created on Thursday, 12 November 2020 (IST) \n @author: Dhananjayan")

file = st.sidebar.file\_uploader("Please upload an image file", type=["jpg", "png"])

if file is None:

st.text("You haven't uploaded an image file")

else:

image = Image.open(file)

img = np.array(image)

st.text("Your original image")

st.image(image, use\_column\_width=True)

st.text("Your colorized image")

color = colorizer(img)

st.image(color, use\_column\_width=True)

print("done!")

# In[ ]: