

# Black And White Image Colorizer

Name: Nairit Das

Reg No: 24MAI0097

## **Exp 1:** Image Colorization Using Pre-Trained Deep Learning Models

**Description:** This experiment demonstrates the process of colorizing grayscale images using a pre-trained deep learning model based on the Caffe framework. The model utilizes a pre-trained architecture that predicts the chromatic (a and b) channels of the LAB color space while using the input grayscale image as the lightness (L channel). The key components of the process include:

1. **Model Initialization:** The pre-trained model is loaded along with cluster centers (pts\_in\_hull), which are embedded into the network as convolutional kernels.
2. **Input Preparation:** The grayscale image is normalized, resized to 224x224, and mean-centered for compatibility with the model's input requirements.
3. **Inference:** The network predicts the chromatic components, which are then resized back to the original image dimensions.
4. **Post-Processing:** The predicted chromatic channels are combined with the original grayscale L channel, and the result is converted from LAB to RGB color space to produce the final colorized image.

This pipeline leverages the LAB color space to separate luminance from chrominance, allowing the model to focus on reconstructing realistic color tones while preserving the structural details of the original image

## Code:

```
[42] from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount=True).

```
[43] # IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.
#!/bin/bash
!kaggle datasets download ashishpatel26/colorise-image
```

Dataset URL: <https://www.kaggle.com/datasets/ashishpatel26/colorise-image>  
 License(s): copyright-authors  
 colorise-image.zip: Skipping, found more recently modified local copy (use --force to force download)

```
[44] import zipfile
import os

# Define the path to your downloaded zip file
zip_file_path = '/content/colorise-image.zip' # Change this to the actual path of your zip file

# Define the destination folder for extraction
extraction_folder = '/content/drive/MyDrive/DATA/' # Change this if you want to extract to a different folder

# Check if the destination folder exists, if not, create it
if not os.path.exists(extraction_folder):
    os.makedirs(extraction_folder)
```

```
[45] import cv2 # opencv 3.4.2+ required
import os
import numpy as np
import matplotlib.pyplot as plt
```

```
[46] print(cv2.__version__)
```

4.11.0

### ✓ Prepare Model

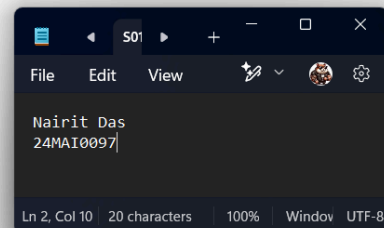
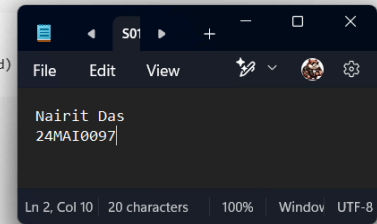
- You need to download models from Zhang's server
- Run "get\_models.sh" to get it

```
[47] proto = '/content/drive/MyDrive/DATA/colorization_deploy_v2.prototxt.txt'
weights = '/content/drive/MyDrive/DATA/colorization_release_v2_norebal.caffemodel'
# colorization_release_v2_norebal.caffemodel is trained with a classification loss with no class re-balancing term.
# The results are duller but "safer" colorizations
# weights = './models/colorization_release_v2_norebal.caffemodel'

# load cluster centers
pts_in_hull = np.load('/content/drive/MyDrive/DATA/pts_in_hull.npy')
pts_in_hull = pts_in_hull.transpose().reshape(2, 313, 1, 1).astype(np.float32)

# load model
net = cv2.dnn.readNetFromCaffe(proto, weights)
# net.getLayerNames()

# populate cluster centers as 1x1 convolution kernel
```



```

✓ [47] net.getLayer(net.getLayerId('class8_ab')).blobs = [pts_in_hull]
0s # scale layer doesn't look work in OpenCV dnn module, we need to fill 2.606 to conv8_313_rh layer manually
net.getLayer(net.getLayerId('conv8_313_rh')).blobs = [np.full((1, 313), 2.606, np.float32)]

```

```

✓ [48] img_path = '/content/drive/MyDrive/DATA/sample_2.jpg'
1s img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
img_input = img.copy()

# convert BGR to RGB
img = cv2.cvtColor(img, cv2.COLOR_GRAY2RGB)

img_rgb = img.copy()

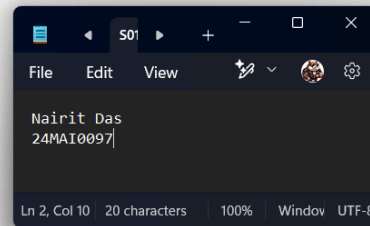
# normalize input
img_rgb = (img_rgb / 255.).astype(np.float32)

# convert RGB to LAB
img_lab = cv2.cvtColor(img_rgb, cv2.COLOR_RGB2Lab)
# only L channel to be used
img_l = img_lab[:, :, 0]

input_img = cv2.resize(img_l, (224, 224))
input_img -= 50 # subtract 50 for mean-centering

# plot images
# fig = plt.figure(figsize=(10, 5))
# fig.add_subplot(1, 2, 1)
# plt.imshow(img_rgb)
# fig.add_subplot(1, 2, 2)
# plt.axis('off')
plt.figure(figsize=(9,6))
plt.imshow(input_img, cmap='gray')

```

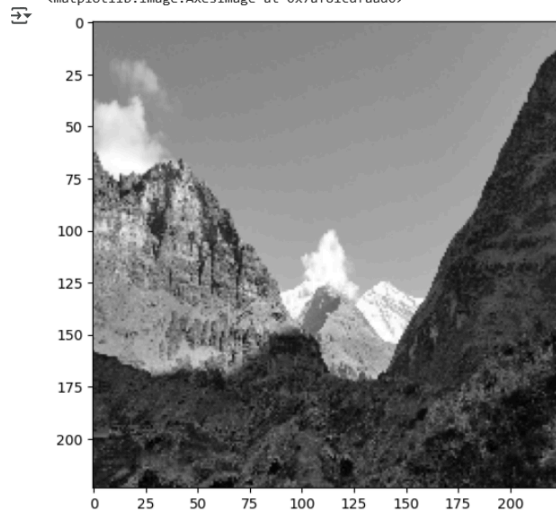
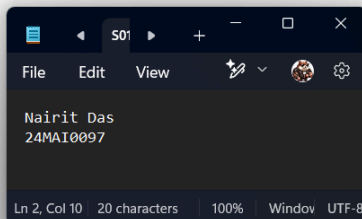


A small window titled "S01" with a menu bar (File, Edit, View) and a toolbar. The main content area displays the text "Nairit Das" and "24MAI0097". The status bar at the bottom shows "Ln 2, Col 10", "20 characters", "100%", "Window", and "UTF-8".

```

▶ <matplotlib.image.AxesImage at 0x7af81cdfaad0>

```

A small window titled "S01" with a menu bar (File, Edit, View) and a toolbar. The main content area displays the text "Nairit Das" and "24MAI0097". The status bar at the bottom shows "Ln 2, Col 10", "20 characters", "100%", "Window", and "UTF-8".

## ▼ Prediction

```

net.setInput(cv2.dnn.blobFromImage(input_img))
pred = net.forward()[0,:,:,:].transpose((1, 2, 0))

# resize to original image shape
pred_resize = cv2.resize(pred, (img.shape[1], img.shape[0]))

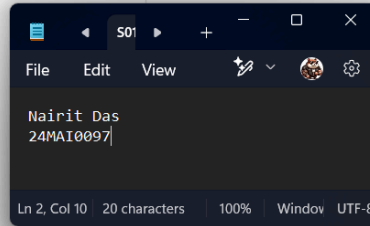
# concatenate with original image L
pred_lab = np.concatenate([img_l[:, :, np.newaxis], pred_resize], axis=2)

# convert LAB to RGB
pred_rgb = cv2.cvtColor(pred_lab, cv2.COLOR_Lab2RGB)
pred_rgb = np.clip(pred_rgb, 0, 1) * 255
pred_rgb = pred_rgb.astype(np.uint8)

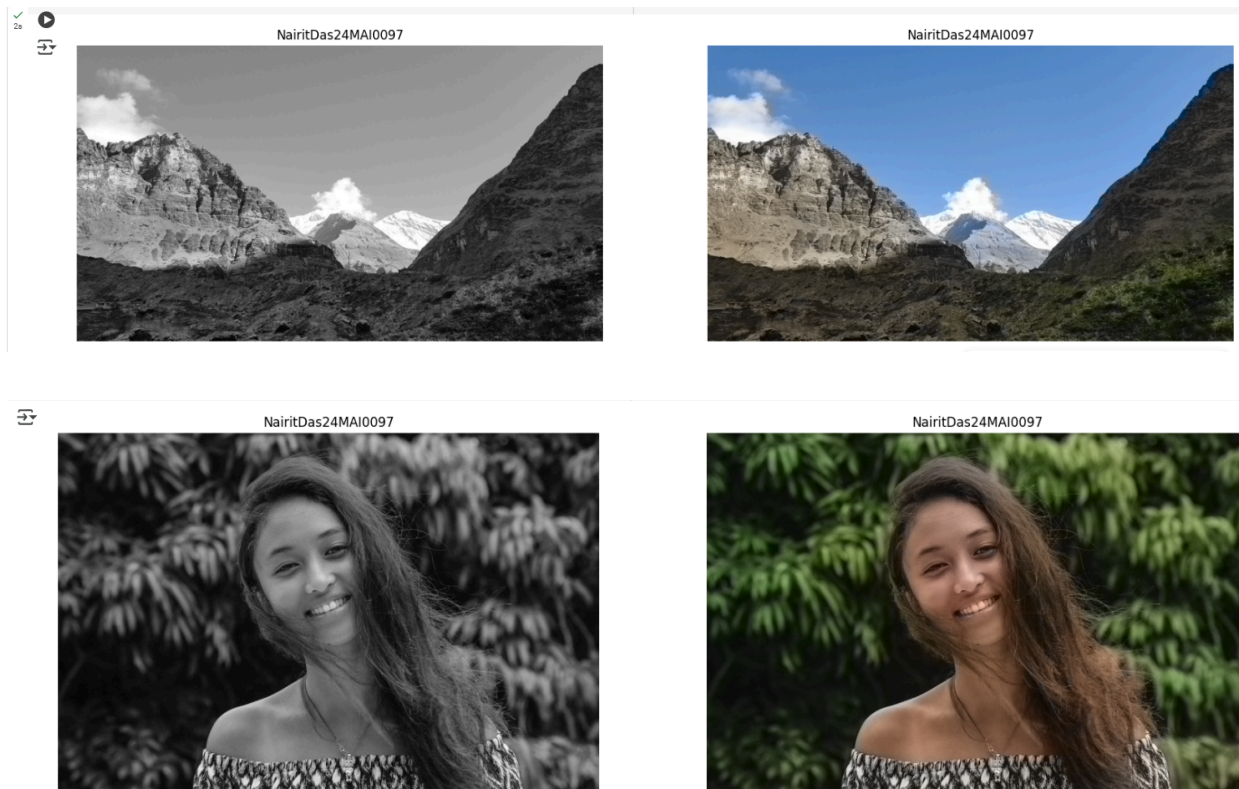
# plot prediction result
fig = plt.figure(figsize=(20, 10))
fig.add_subplot(1, 2, 1).axis('off')
plt.imshow(img_l, cmap='gray')
fig.add_subplot(1, 2, 2).axis('off')
plt.imshow(pred_rgb)
# plt.savefig(output_filename)

# save result image file
filename, ext = os.path.splitext(img_path)
# input_filename = '%s_input%s' % (filename, ext)
# output_filename = '%s_output%s' % (filename, ext)

```



## Snapshots of Results:





NairitDas24MAI0097



NairitDas24MAI0097



**Conclusion:** The experiment successfully colorized the grayscale image, generating a vibrant and natural-looking output. The pre-trained model demonstrated strong generalization by producing plausible color tones, even without additional training. While the results are impressive, further improvements could be achieved by fine-tuning the model on specific datasets to better match desired colorization styles or domain-specific requirements.

---