Problem Statement - To develop a tool A.I. based image colorizer to color old black and white images.

Problem Description - The project provides the user with an image colorizer that is a software application that uses deep learning techniques to automatically add colors to grayscale images. The project has two pre-trained neural network models that have been trained on vast amounts of colored images to predict the most likely colors for a grayscale image.The application takes a grayscale image(i.e. black and white) as input and processes it using the selected colorization model.The AI-based image colorizer project has a wide range of potential applications, from restoring old black-and-white photographs to adding colors to medical images for enhanced diagnosis. Overall, this project provides a user-friendly interface for adding colors to grayscale images using state-of-the-art deep learning techniques, offering a simple and efficient solution for various applications.

Repository Link - https://github.com/nikhildman/AIProject

Literature Survey -

1. "Colorful Image Colorization" by Richard Zhang, Phillip Isola, and Alexei A. Efros (2016):

This paper proposed a deep learning-based approach for image colorization that uses a convolutional neural network to predict the chrominance values of grayscale input images. The authors demonstrated that their approach outperforms previous methods in terms of visual quality and accuracy.

2. "Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2" by Suyog Jadhav and Om Deshmukh (2018):

This paper proposed a modified version of the Inception-ResNet-v2 architecture for image colorization. The authors used a combination of convolutional and deconvolutional layers to upsample the input image and predict the chrominance values. They demonstrated that their approach produces high-quality colorized images with fewer artifacts than previous methods.

3. "Instance Normalization for Deep Learning-based Image Colorization" by Jaewoo Kang and Wonjun Kim (2019):
This paper proposed a modified version of the U-Net architecture for image colorization that uses instance normalization instead of batch normalization. The authors demonstrated that their approach produces more realistic and accurate colorized images than previous methods.

4. "Deep Colorization via Multimodal Fusion with Prior Guidance" by Rui Cai, Xiaodan Liang, Xiaohui Shen, and Jiashi Feng (2019):
This paper proposed a multimodal fusion approach for image colorization that combines multiple sources of information, including color histograms, texture features, and object segmentation maps. The authors demonstrated that their approach produces high-quality colorized images with better color consistency than previous methods.

Code and Explanation -
Step-by-step approach to colorize a picture using an AI-based image colorization model:

1)Choose an image that you want to colorize.
2)Preprocess the image by converting it to grayscale using a suitable software or library, such as OpenCV or Pillow.
3)Load the pre-trained AI-based image colorization model that you want to use for colorization.
4)Feed the grayscale image to the model for colorization. The model will output a colorized image in RGB format.
5)Postprocess the colorized image by converting it to the desired format, such as JPEG or PNG, and saving it to a file.

Description of Modules used -

1) **argparse:** This module provides a convenient way to parse command line arguments in Python.

2) **matplotlib.pyplot:** This module provides a collection of functions for creating and manipulating figures and plots.

3) **colorizers:** This is a custom module that contains the definitions of two deep learning models for colorizing grayscale images: eccv16 and siggraph17.

Dataset Used - Our project doesn't explicitly use a dataset, as it is based on pre-trained colorization models that have been trained on large datasets of colored images.The pre-trained models used in the code, namely "eccv16" and "siggraph17", have been trained on different datasets.

1) **eccv16 -** ECCV16 refers to the European Conference on Computer Vision 2016, which is a prestigious conference in the field of computer vision. The ECCV16 model is a deep learning model designed for semantic segmentation, which is the process of labeling each pixel in an image with its corresponding object class. This model uses a fully convolutional neural network (FCN) architecture and has achieved state-of-the-art performance on several benchmark datasets.

2) **siggraph17 -** SIGGRAPH17 refers to the Special Interest Group on Computer Graphics and Interactive Techniques conference held in 2017. The SIGGRAPH17 model is a deep learning model designed for image synthesis, which is the process of generating images from scratch. This model uses a generative adversarial network (GAN) architecture and has demonstrated impressive results in generating realistic images of human faces, animals, and landscapes.

Code -

Section 1:

```
import argparse
import matplotlib.pyplot as plt

from colorizers import *
```

In this section, the necessary libraries and modules are imported for the program. Specifically, argparse is used to parse command line arguments, matplotlib.pyplot is used for plotting images, and colorizers is a custom module containing pre-trained models for image colorization.

Section 2: Parsing Command Line Arguments

```
parser = argparse.ArgumentParser()
parser.add_argument('-i','--img_path', type=str, default='imgs/2.jpg')
parser.add_argument('--use_gpu', action='store_true', help='whether to use GPU')
parser.add_argument('-o','--save_prefix', type=str, default='saved', help='will save into this file with {eccv16.png, siggraph17.png} suffixes')
opt = parser.parse_args()
```

In this section, command line arguments are parsed using argparse. Three arguments are defined: -i or --img_path for the path to the input image, --use_gpu for whether to use GPU for computation, and -o or --save_prefix for the prefix to save output images. The default values are provided for each argument. The parsed arguments are stored in the opt variable.

Section 3: Loading and Configuring Models

```
colorizer_eccv16 = eccv16(pretrained=True).eval()
colorizer_siggraph17 = siggraph17(pretrained=True).eval()
if(opt.use_gpu):
    colorizer_eccv16.cuda()
    colorizer_siggraph17.cuda()
```

In this section, the pre-trained models for image colorization are loaded from the colorizers module. Specifically, the eccv16 and siggraph17 models are loaded and set to evaluation mode using the .eval() method. If the --use_gpu argument is provided, the models are also configured to use GPU for computation using the .cuda() method.

Section 4: Preprocessing the Input Image

```
img = load_img(opt.img_path)
(tens_l_orig, tens_l_rs) = preprocess_img(img, HW=(256,256))
if(opt.use_gpu):
    tens_l_rs = tens_l_rs.cuda()
```

In this section, the input image is loaded using the load_img() function from the colorizers module. The image is then preprocessed by extracting the luminance channel (L) in both original and resized resolutions using the preprocess_img() function. The resized L channel is converted to a tensor and, if GPU is used, moved to the GPU memory.

Section 5: Colorizing the Input Image

```
img_bw = postprocess_tens(tens_l_orig, torch.cat((0*tens_l_orig,0*tens_l_orig),dim=1))
out_img_eccv16 = postprocess_tens(tens_l_orig, colorizer_eccv16(tens_l_rs).cpu())
out_img_siggraph17 = postprocess_tens(tens_l_orig, colorizer_siggraph17(tens_l_rs).cpu())
```

In this section, the input image is colorized using both the eccv16 and siggraph17 models. The postprocess_tens() function is used to convert the output tensors to images by concatenating the original L channel with the predicted chrominance (ab) channels. The output images are stored in the out_img_eccv16 and out_img_siggraph17 variables.

Section 6:Displaying the result

```
plt.imsave('%s_eccv16.png'%opt.save_prefix, out_img_eccv16)
plt.imsave('%s_siggraph17.png'%opt.save_prefix, out_img_siggraph17)

plt.figure(figsize=(12,8))
plt.subplot(2,2,1)
plt.imshow(img)
plt.title('Original')
plt.axis('off')

plt.subplot(2,2,2)
plt.imshow(img_bw)
plt.title('Input')
plt.axis('off')

plt.subplot(2,2,3)
plt.imshow(out_img_eccv16)
plt.title('Output (ECCV 16)')
plt.axis('off')

plt.subplot(2,2,4)
plt.imshow(out_img_siggraph17)
plt.title('Output (SIGGRAPH 17)')
plt.axis('off')
plt.show()
```

The first two lines save the colorized output images in PNG format using the file name specified by the user with the save_prefix argument. Two versions of the output image are saved, one for each colorization model (eccv16 and siggraph17).

The following lines create a figure with four subplots, each displaying a different image. The first subplot displays the original input image (img), the second subplot shows the grayscale input image (img_bw) used as input for colorization, and the last two subplots show the colorized output images obtained using the two different colorization models (out_img_eccv16 and out_img_siggraph17).

The imshow() function is used to display the images, and the title() function is used to add titles to each subplot. The axis('off') function is used to remove the axes and ticks from each subplot. Finally, the show() function is called to display the figure on the screen.

Examples of Output->

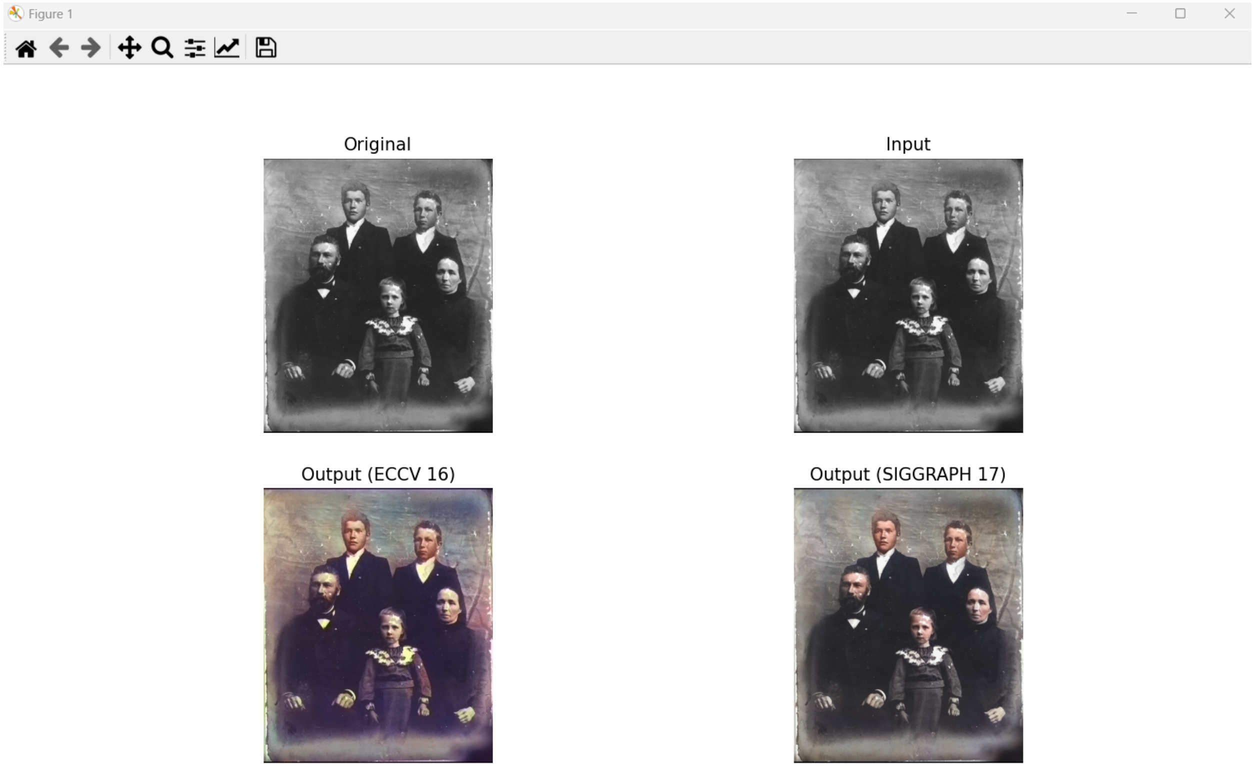
Figure 1
Original
Input
Output (ECCV 16)
Output (SIGGRAPH 17)

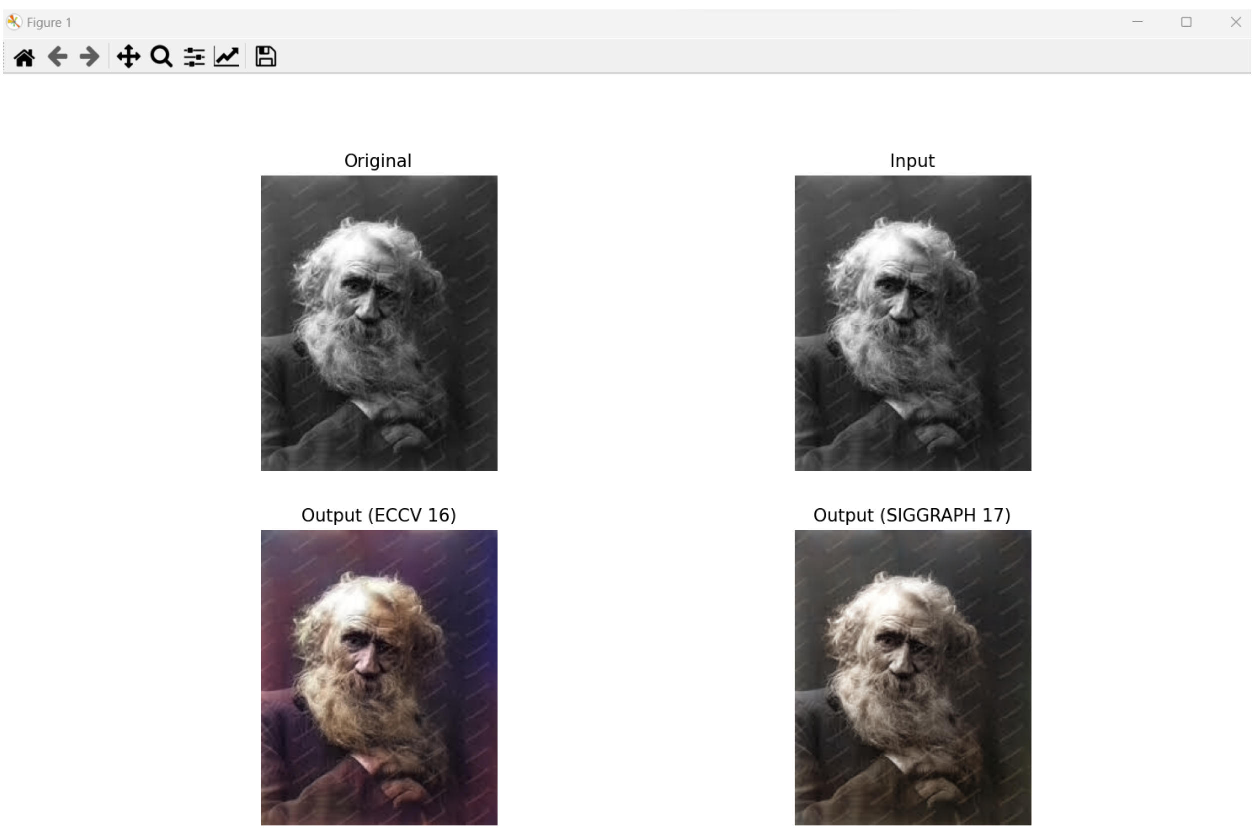
Figure 1
Original
Input
Output (ECCV 16)
Output (SIGGRAPH 17)

Scope of the project - The scope of the above code is to provide a way to colorize grayscale images using pre-trained deep learning models. This code can be useful in a variety of applications such as:

1)Colorizing old black and white photographs or movies.

2)Colorizing medical images to help doctors visualize the data better.

3)Enhancing images for computer vision tasks such as object detection or recognition.

4)Generating colorized images for artistic or creative purposes.