```
!rm -rf colorization  // --> Remove any existing 'colorization' folder

!pip install --upgrade torch torchvision scikit-image opencv-python matplotlib  // --> Install required libraries

!git clone --depth 1 https://github.com/richzhang/colorization.git  // --> Clone the colorization repo with shallow depth

%cd colorization  // --> Change directory to the cloned 'colorization' folder

!mkdir -p models  // --> Create 'models' directory if it doesn't exist

!wget https://colorizers.s3.us-east-2.amazonaws.com/colorization_release_v2-9b330a0b.pth -O models/colorization_release_v2.pth  // --> Download ECCV16 model weights

!wget https://colorizers.s3.us-east-2.amazonaws.com/colorization_release_v1-6d696e8a.pth -O models/colorization_release_v1.pth  // --> Download SIGGRAPH17 model weights


print("✅ Setup complete!")  // --> Confirm setup is done


from google.colab import files  // --> Import Colab files module for upload

import os  // --> Import OS module for file handling


uploaded = files.upload()  // --> Upload B&W image file

input_filename = next(iter(uploaded))  // --> Get the uploaded filename

os.rename(input_filename, f"imgs/{input_filename}")  // --> Move uploaded file to 'imgs' folder

print(f"✅ Uploaded: {input_filename}")  // --> Print uploaded filename


from PIL import Image  // --> Import PIL for image handling

import matplotlib.pyplot as plt  // --> Import matplotlib for plotting


img = Image.open(f"imgs/{input_filename}")  // --> Open uploaded image

plt.imshow(img)  // --> Display image using matplotlib

plt.title('Original B&W Image')  // --> Set image title

plt.axis('off')  // --> Hide axes

plt.show()  // --> Show the image plot


import sys  // --> Import sys for path modification
```

```
sys.path.append('.')  // --> Add current directory to Python path

from colorizers import *  // --> Import colorizer models

import torch  // --> Import PyTorch for deep learning


device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')  // --> Set device to GPU if available


try:

   colorizer_eccv16 = eccv16(pretrained=True).to(device).eval()  // --> Load and prepare ECCV16 model

   colorizer_siggraph17 = siggraph17(pretrained=True).to(device).eval()  // --> Load and prepare SIGGRAPH17 model

   print("✅ Models loaded successfully!")  // --> Print success message

except Exception as e:

   print(f"❌ Error loading models: {e}")  // --> Print error if model loading fails

   raise  // --> Raise exception to stop execution


def process_image(img_path):  // --> Define image processing function

   img = load_img(img_path)  // --> Load image

   (tens_l_orig, tens_l_rs) = preprocess_img(img, HW=(256,256))  // --> Preprocess image into tensors

   tens_l_rs = tens_l_rs.to(device)  // --> Move tensor to selected device


   with torch.no_grad():  // --> Disable gradient tracking

      out_img_eccv16 = postprocess_tens(tens_l_orig, colorizer_eccv16(tens_l_rs).cpu())  // --> Get ECCV16 model output

      out_img_siggraph17 = postprocess_tens(tens_l_orig, colorizer_siggraph17(tens_l_rs).cpu())  // --> Get SIGGRAPH17 model output

   return out_img_eccv16, out_img_siggraph17  // --> Return colorized images


eccv16_result, siggraph17_result = process_image(f"imgs/{input_filename}")  // --> Run colorization function
```

```
plt.figure(figsize=(20,10))  // --> Create figure with specified size
plt.subplot(1,3,1)  // --> First subplot
plt.imshow(img)  // --> Show original image
plt.title('Original')  // --> Set title
plt.axis('off')  // --> Hide axis

plt.subplot(1,3,2)  // --> Second subplot
plt.imshow(eccv16_result)  // --> Show ECCV16 result
plt.title('ECCV16')  // --> Set title
plt.axis('off')  // --> Hide axis

plt.subplot(1,3,3)  // --> Third subplot
plt.imshow(siggraph17_result)  // --> Show SIGGRAPH17 result
plt.title('SIGGRAPH17')  // --> Set title
plt.axis('off')  // --> Hide axis

plt.show()  // --> Display all subplots
```