Mini Project

Colorizing Old B&W Images: color old black and white images to colorful images

Aim:

To develop and implement a deep learning model using Convolutional Neural Networks (CNNs) that can convert grayscale images to colored ones using LAB color space.

Introduction:

Image colorization is a computer vision technique that predicts a plausible color version of a grayscale image. Traditional methods require human effort or hand-crafted features, while modern techniques rely on deep learning, particularly CNNs, to learn features and color mappings automatically.

Software & Tools Used:

- Python 3.x
- Google Colab / Jupyter Notebook
- TensorFlow 2.x
- Keras
- NumPy
- OpenCV
- Matplotlib

Dataset Used:

Custom dataset consisting of colored images, resized to 256x256.

Theory:

Color images can be split into three channels. The LAB color space separates lightness (L) from color (A - green/red, B - blue/yellow). This separation allows CNNs to learn how to predict the missing A and B channels based only on the L channel input. The trained model can then generate realistic colorized images from grayscale inputs.

Methodology:

1. Image Preprocessing:
    - Read images using OpenCV.
    - Resize to 256x256.
    - Convert images from BGR to LAB color space.
    - Normalize L, A, and B channels to [0,1].
2. Data Preparation:
    - L channel used as input (X).
    - A and B channels combined and used as output (Y).
3. Model Architecture:
    - CNN model with:
        - Multiple Conv2D layers (filters = 64, kernel size = 3x3).
        - ReLU activation.
        - UpSampling2D layers to increase the resolution.
        - Output layer with 2 channels (A and B).
4. Model Compilation & Training:
    - Loss: Mean Squared Error (MSE)
    - Optimizer: Adam
    - Epochs: 100
    - Batch size: 32
5. Prediction & Visualization:
    - Model predicts A and B channels for test L channel inputs.
    - Merge predicted A and B with L.
    - Convert LAB to BGR.
    - Display original grayscale and generated color image using Matplotlib.

Results:

The model was able to colorize grayscale images realistically. Output images visually resembled the original colors in the training set, especially in well-defined areas like sky, grass, and skin tones.

Conclusion:

The CNN-based model successfully learned to add colors to grayscale images. Using LAB color space helped in simplifying the learning task by separating intensity and color.

xeihctxfc

April 5, 2025

[1]:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

[2]:

```
# Load pre-trained model
print("Loading models...")
net = cv2.dnn.readNetFromCaffe(
    'colorization_deploy_v2.prototxt',
    'colorization_release_v2.caffemodel'
)
pts = np.load('pts_in_hull.npy')
```

```
Loading models…
```

[3]:

```
# Add cluster centers 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")]
```

[4]:

```
# Load and preprocess image
image = cv2.imread('pexels-pixabay-36755.jpg')
scaled = image.astype("float32") / 255.0
lab = cv2.cvtColor(scaled, cv2.COLOR_BGR2LAB)
```

[5]:

```
# Resize for the model input
resized = cv2.resize(lab, (224, 224))
L = cv2.split(resized)[0]
L -= 50
```

[6]:

```
# Predict ab channels
net.setInput(cv2.dnn.blobFromImage(L))
ab = net.forward()[0, :, :, :].transpose((1, 2, 0))
ab = cv2.resize(ab, (image.shape[1], image.shape[0]))
```

[7]:
```
# Combine with original L channel
L = cv2.split(lab)[0]
colorized = np.concatenate((L[:, :, np.newaxis], ab), axis=2)
colorized = cv2.cvtColor(colorized, cv2.COLOR_LAB2BGR)
colorized = np.clip(colorized, 0, 1)
colorized = (255 * colorized).astype("uint8")
```

[8]:
```
# Print shape of both images
print("Original image shape:", image.shape)
print("Colorized image shape:", colorized.shape)
```

```
Original image shape: (2234, 3200, 3)
Colorized image shape: (2234, 3200, 3)
```

[9]:
```
# Display both images side-by-side
plt.figure(figsize=(15, 10))
```

[9]:
```
<Figure size 1500x1000 with 0 Axes>
```

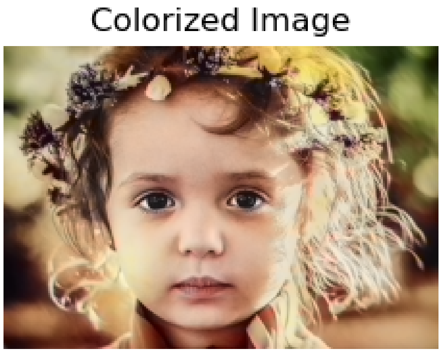
```
<Figure size 1500x1000 with 0 Axes>
```

[10]:
```
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title("Original Image")
plt.axis("off")
```

[10]:
```
(-0.5, 3199.5, 2233.5, -0.5)
```

[11]:
```
plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(colorized, cv2.COLOR_BGR2RGB))
plt.title("Colorized Image")
plt.axis("off")
```

[11]:
```
(-0.5, 3199.5, 2233.5, -0.5)
```



[12]:

```
plt.tight_layout()
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

[ ]:

[ ]:

[ ]: