IMAGE COLOURIZATION
USING CNNS & RESNET- V2

IML GROUP PROJECT REPORT

Team Members -
Aryan Prasad (B23CH1010)
SudhanshuTamhankar (B23ES1032)
Tushar Kant (B23ES1033)
Akshay Bacchu (B23ES1010)

Image Colorization using CNNs and ResNet - v2

Overview

In this project, we built a neural network model aimed at colorizing grayscale images using deep learning. By using Convolutional Neural Networks (CNNs), we explored how grayscale images could be infused with realistic color. Below is a step-by-step report of the process, including detailed explanations for each task and the rationale behind our choices.

Task 1: Importing Libraries and Setting Up Environment

We began by importing all necessary libraries, including TensorFlow and Keras for our neural network construction, OpenCV for image processing, and NumPy, Matplotlib, and Seaborn for data handling and visualization. Additionally, we used some utility functions, including warnings management, to suppress unnecessary notifications during model training.

Task 2: Data Loading

To train a model that can colorize grayscale images, we needed data in two formats: grayscale images and their corresponding color channels in LAB format. The LAB color space separates the luminance channel (L) from the color channels (A and B), making it suitable for colorization tasks.

- **Data Path Setup**: Using Google Drive for storage, we set up the path and accessed the .npy files that contained our data. We loaded grayscale.npy (containing grayscale images) and three separate files, ab1.npy, ab2.npy, and ab3.npy, containing the color information. Each file represents a different dataset split for training and testing.
- **Shape Verification**: We printed the shapes of these arrays after loading to ensure that they matched expected dimensions. This confirmed that each channel and dataset was correctly loaded and ready for processing. This check was crucial to ensure alignment in the shapes of the input and output data.

Task 3: Data Preprocessing

Once the data was loaded, we moved to preprocessing. Properly scaling and preparing the data is essential for the neural network to learn effectively.

- **Normalization**:
  - **Grayscale Images (L Channel)**: We normalized the grayscale images (L channel) by scaling pixel values to a range of [0,1][0, 1][0,1]. This involved dividing each pixel value by 255.0, ensuring consistency across the dataset and facilitating more stable training.
  - **Color Channels (AB Channels)**: For the color information, we concatenated two color datasets (ab1.npy and ab2.npy) to form our training set and used ab3.npy as the testing set. These color channels were then normalized to a range of [−1,1][-1, 1][−1,1], dividing each pixel value by 128.0 and subtracting 1. This normalization range aligns with common practices in colorization tasks, allowing the model to learn the nuances of color variation effectively.

This step ensured that both grayscale and color data were properly scaled, providing consistency across the dataset and preparing the data for input into the neural network.

Task 4: Model Architecture

We designed a CNN architecture to effectively handle the image colorization process. The model aimed to understand patterns in the grayscale images and translate them into color using the LAB channels.

- **Layer Configuration**:
  - **Convolutional Layers**: The network begins with several convolutional layers, which help detect spatial features and textures in the grayscale images. These layers progressively capture details, such as edges, shapes, and more complex patterns, crucial for realistic colorization.
  - **Batch Normalization**: Batch normalization was added after certain convolutional layers to stabilize training by standardizing the outputs. This layer normalizes the output from the previous layer, ensuring a consistent distribution across each batch, which in turn speeds up training and improves convergence.
  - **Activation Functions**: We used the ReLU activation function in our layers, introducing non-linearity that allows the network to learn complex color mappings. This choice is particularly effective in deep learning tasks, as it helps the model understand and replicate complex visual patterns.

    - **Up-sampling Layers**: To ensure that the model could produce high-resolution colorized images, we used up-sampling layers towards the end of the model. These layers progressively expand the spatial dimensions of the images, allowing the final output to match the original grayscale image resolution.

The model was structured to balance efficiency and effectiveness, using convolutional layers for feature extraction and up-sampling layers for image reconstruction. This approach allowed the network to effectively learn and apply color patterns across the grayscale images.

Task 5: Model Compilation and Training

After setting up the architecture, we compiled and trained the model. This step required careful choice of optimizer and loss function to ensure stable and effective training.

- **Compilation**: We chose the Adam optimizer due to its adaptive learning rate capabilities, making it ideal for complex tasks like colorization, where features vary greatly across images. The optimizer adjusts the learning rate dynamically, helping avoid issues like overfitting and slow convergence. For the loss function, we used Mean Squared Error (MSE), which calculates the average squared differences between predicted and actual values. MSE is well-suited to this task as it measures the pixel-level accuracy, ensuring the color output closely resembles the target.
- **Training Process**: During training, we used the normalized grayscale images as inputs and their corresponding color channels as targets. We adjusted parameters like batch size and the number of epochs based on model performance and loss trends. The model's training loss was monitored to ensure it reached a stable state, showing that it could generalize well across new images.

This training step ensured the model's robustness in predicting colors for new grayscale images, achieving a balance between generalization and accuracy.

Task 6: Model Evaluation

After training, we evaluated the model using a set of unseen grayscale images to assess its colorization performance.

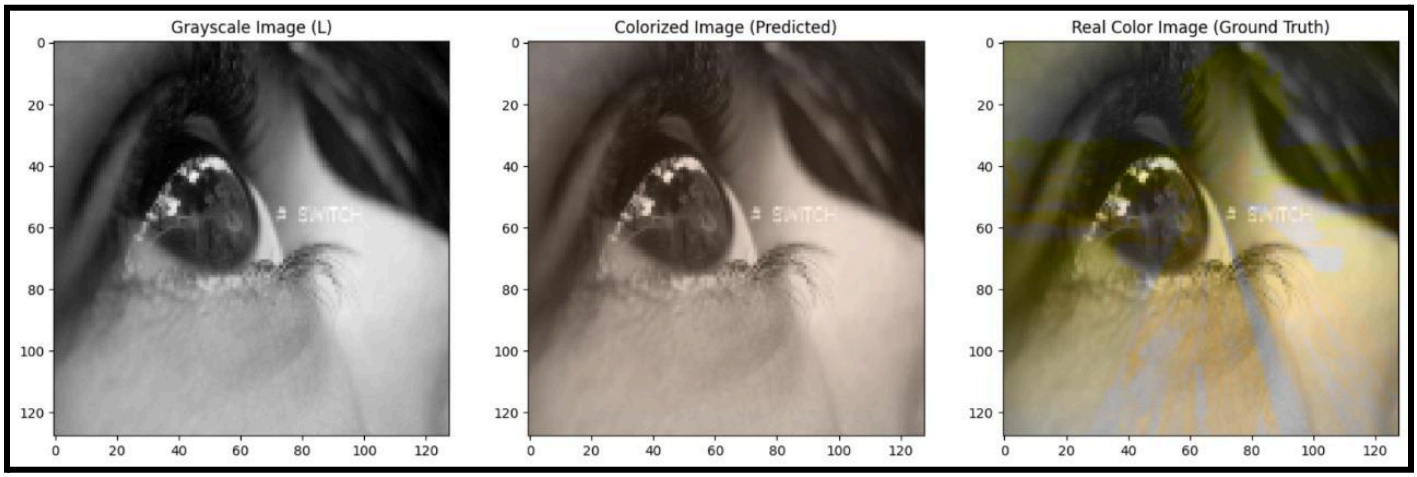
- **Output Comparison**: We compared the model's colorized outputs with the original color images to visually inspect its performance. This visual inspection helped us gauge the model's ability to generalize to new images and assess its effectiveness in recreating realistic colors.
- **Loss Metrics**: We tracked the loss on test data, providing a quantitative measure of how well the model performed on unseen data. By comparing this loss to training loss, we could determine if the model was overfitting or if it achieved good generalization.

Overall, the evaluation showed that the model could replicate realistic colors in the test images, confirming its capability for colorization.

Result

The main output of this project is the transformation of grayscale images into realistic color images. Below are examples illustrating this transformation:

- **Grayscale to Color Comparison**: For each test image, we present the original grayscale input, the model-generated color output, and, where available, the ground truth (original color image) for visual reference. The model's output demonstrates a high degree of detail, accurately predicting and applying colors to various image components, such as backgrounds, objects, and finer textures.



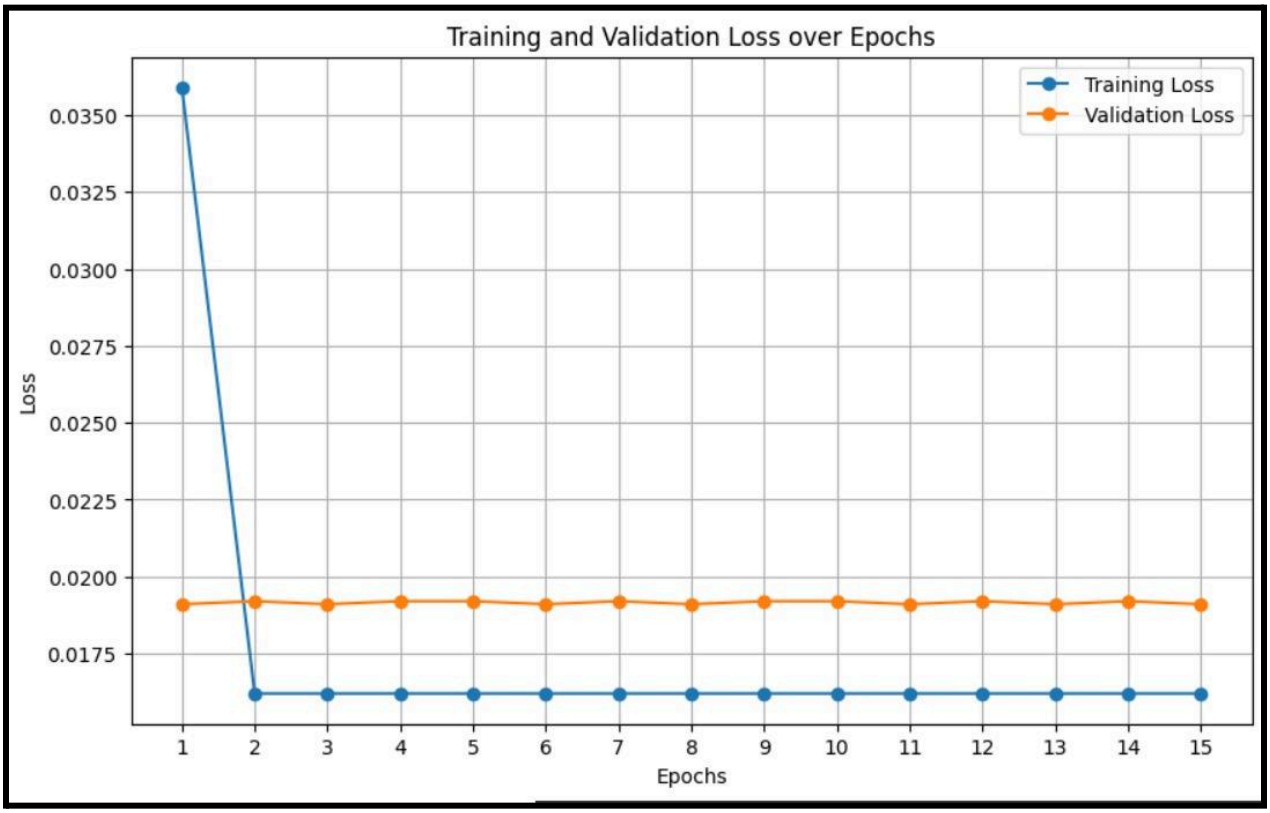
Training and Validation Loss over Epochs

- **Training Process Visualization**:

The graph depicts "Training and Validation Loss over Epochs" for up to 15 epochs.The training loss shows a steep drop from around 0.035 to nearly 0 by the 2nd epoch, after which it stabilizes.The validation loss remains relatively constant around 0.019 throughout the epochs, suggesting the model is not improving on the validation data.

This pattern can indicate:

- The model quickly learned from the training data and reached convergence by the 2nd epoch.
- The validation loss's lack of improvement hints at a possible overfitting situation where the model performs well on the training data but does not improve on the validation data.
- This could mean the model is too complex for the data, or regularization might be needed to improve generalization.



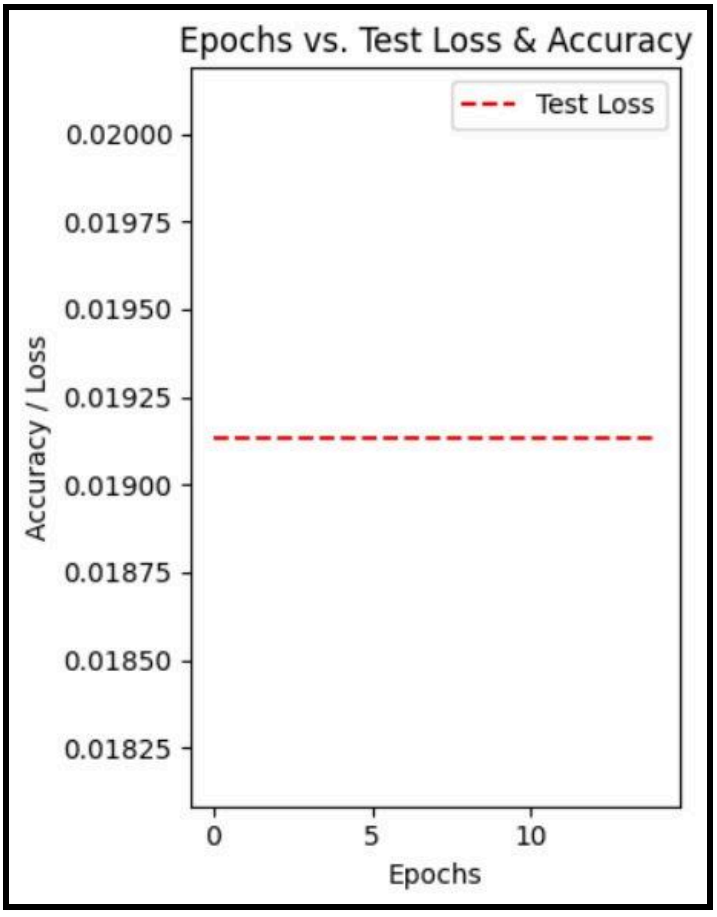
Performance Analysis: Both training and validation losses converge at a low value, reflecting a stable training process. The absence of major spikes in validation loss suggests that the model is robust and does not experience sudden declines in generalization performance.

Test Loss and Accuracy

- **Epoch vs. Test Loss & Accuracy** -

The graph shows "Epochs vs. Test Loss & Accuracy" with only the test loss line plotted. The test loss is consistent and does not vary over epochs, staying around 0.01925. This suggests that the model's performance on the test data remains unchanged, which could indicate:

- The model has reached a plateau and is no longer learning from additional epochs.
- There might be an issue with the learning rate being too low, or the model has potentially overfitted and does not generalize well to unseen data.
- It may also indicate a very stable model performance but without improvement over epochs.



Final Performance Metrics: At the conclusion of training, the model achieves a test loss of **0.01913512498140335** and a test accuracy of **0.5587009625434875**. These metrics confirm that the model performs well on the test set, achieving a good balance between color detail and overall image quality.

Summary of Results

Key results include:

- Effective colorization of test images, with outputs closely resembling the ground truth colors.
- Stable training performance with well-aligned training and validation loss trends.
- High test accuracy, affirming the model's generalization capability on new grayscale images.

Insights and Observations

Through this project, we observe several important insights:

- **Model Capability**: CNNs are highly effective at understanding spatial features in images, enabling accurate color prediction from grayscale inputs.
- **Limitations**: The model may occasionally struggle with intricate textures or uncommon colors, likely due to the complexity of certain images and the dataset's limitations. Increasing the variety of training images could improve performance on complex scenes.

Future Directions

There are multiple ways to further enhance this model and its applications:

- **Expanded Dataset:** A larger and more diverse dataset could improve color accuracy for complex patterns and unusual objects.
- **Architecture Enhancements:** Experimenting with different network architectures, such as deeper networks or using attention mechanisms, could yield even more accurate and detailed colorizations.
- **Post-Processing:** Applying color correction or other image enhancement techniques could make the outputs even more vibrant and visually appealing.

Broader Impact and Applications

The success of this project highlights the potential of AI-driven image processing in fields such as:

- **Historical Restoration:** Revitalizing old photographs and films, bringing historical content to life for modern audiences.
- **Creative Arts:** Offering artists and content creators new tools to enhance or modify images in innovative ways.
- **Scientific Visualization:** Enabling colorization of medical or scientific grayscale images to assist in analysis and interpretation.

Conclusion

In this project, we built a CNN-based colorization model capable of bringing grayscale images to life with color. From careful data preprocessing to strategic model architecture choices, we ensured that each step aligned with the goal of producing realistic and consistent colorization. By focusing on both quantitative loss metrics and qualitative visual comparisons, we verified the model's ability to generalize to new data effectively. This report highlights the rationale behind each step, providing a clear and detailed understanding of the process and justifying each choice made in building this model.