Colourization of Grayscale Images

Dishan D.M.T.
Department of Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
e20069@eng.pdn.ac.lk

Karunarathne K.N.P.
Department of Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
e20189@eng.pdn.ac.lk

Rathnaweera R.V.C.
Department of Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
e20328@eng.pdn.ac.lk

Dishan W.M.N.
Department of Computer Engineering
University of Peradeniya
Peradeniya, Sri Lanka
e20455@eng.pdn.ac.lk

***Abstract*—Image colorization is an important and challenging problem in computer vision with applications in media restoration, medical imaging, and remote sensing. The process involves assigning realistic colors to grayscale images, traditionally performed through manual painting, which is time-consuming and subjective. Recent advancements in deep learning have enabled automated approaches, but achieving high-quality colorization remains difficult due to the complexity of learning natural color distributions.**

This study explores convolutional neural networks (CNNs) and U-Net architectures for grayscale image colorization. Initially, a baseline CNN model was trained on CIFAR-10, aiming to learn color mapping for small-scale images. To improve performance, we experimented with U-Net, leveraging its encoder-decoder structure for better spatial feature extraction. Furthermore, we investigated the impact of larger datasets, incorporating ImageNet and Places365 to enhance generalization and diversity in color prediction. Despite these efforts, our results did not demonstrate significant improvements, suggesting inherent limitations in conventional deep learning approaches for colorization.

Several challenges were identified during this study, including dataset scalability issues, computational constraints, and the inability of CNN-based models to capture global semantic relationships necessary for realistic colorization. Our findings indicate that alternative techniques, such as Generative Adversarial Networks (GANs) or self-supervised learning, may offer better solutions by incorporating adversarial loss or leveraging large-scale unlabeled data.

This research provides insights into the effectiveness of CNN-based and U-Net-based colorization methods and highlights the challenges associated with large-scale dataset handling. The study underscores the need for advanced models capable of capturing global context and color semantics more effectively. Future work will explore GAN-based approaches and self-supervised learning strategies to achieve more realistic and visually appealing colorization results.

***Index Terms*—Image colorization Convolutional neural networks (CNNs) Deep learning U-Net architecture Self-supervised learning Computer vision Image processing Color restoration LAB color space Neural networks Dataset scalability Machine learning Feature extraction Artificial intelligence**

I. Introduction

Gray Image colorization is a fundamental problem in computer vision with significant applications in media restoration, healthcare, and remote sensing. Traditional colorization techniques rely on manual editing, which is both labor-intensive and highly subjective. With advances in deep learning, automated approaches have gained attention, yet generating realistic and context-aware colorized images remains challenging.

Deep learning-based colorization typically involves convolutional neural networks (CNNs) trained to predict color information from grayscale images. However, CNN-based methods often struggle with global semantic understanding, leading to desaturated or unrealistic outputs. To address this, architectures such as U-Net have been explored, leveraging encoder-decoder structures for improved spatial feature extraction. Furthermore, large-scale datasets like ImageNet and Places365 have been introduced to enhance generalization and diversity in colorization models. Despite these efforts, achieving significant improvements in output quality remains difficult due to computational constraints and the inherent ambiguity in colorization tasks.

This study explores CNN-based and U-Net-based architectures for grayscale image colorization, analyzing their effectiveness when trained on CIFAR-10, ImageNet, and Places365 datasets. We examine the limitations of these models and discuss the challenges associated with large-scale dataset handling. The results suggest that traditional deep learning models struggle to capture realistic color distributions, emphasizing the need for alternative approaches such as Generative Adversarial Networks (GANs) or self-supervised learning. This research provides insights into the current state of deep learning-based colorization and highlights potential future directions for improving colorization fidelity.

II. Approach

A. LAB Color Space Representation

Rather than using the standard RGB color space, we adopted the LAB color space, which better separates luminance from chromatic information. The grayscale input image corresponds to the L (lightness) channel, while the model predicts the A and B (chromatic) channels. This representation allows the model to focus on learning color distributions while preserving image structure.

Given an RGB image $I_{\text{RGB}} \in \mathbb{R}^{H \times W \times 3}$, we transform it into LAB space:

$$I_{\text{LAB}} = f_{\text{LAB}}(I_{\text{RGB}}), \tag{1}$$

where $f_{\text{LAB}}(\cdot)$ represents the nonlinear transformation to the LAB color space. The model takes only the L channel as input:

$$I_L = \text{ExtractChannel}(I_{\text{LAB}}, L), \tag{2}$$

and predicts the chromaticity components:

$$(\hat{I}_a, \hat{I}_b) = f_\theta(I_L), \tag{3}$$

where f_θ represents the trainable deep learning model. The predicted $(\hat{I}_a, \hat{I}_b)$ components are then combined with I_L and converted back to RGB:

$$I_{\text{RGB}}^{\text{pred}} = f_{\text{RGB}}(I_L, \hat{I}_a, \hat{I}_b). \tag{4}$$

The network is trained by minimizing the Mean Squared Error (MSE) loss between the predicted and ground truth chrominance values:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} \left((I_a^{(i)} - \hat{I}_a^{(i)})^2 + (I_b^{(i)} - \hat{I}_b^{(i)})^2 \right). \tag{5}$$

Additionally, perceptual loss functions may be introduced to improve the quality of predicted colors.

B. Model Architectures

*1) **CNN (Baseline Model)**:* In this study, we implemented an 8-layer convolutional neural network (CNN) as the baseline model. The network consists of convolutional layers with ReLU activations, followed by batch normalization and pooling layers. Each convolutional operation is defined as:

$$x_{l+1} = \sigma(W_l * x_l + b_l), \tag{6}$$

where:

- x_l is the feature map at layer l,
- W_l and b_l are the trainable weight and bias parameters,
- $\sigma(\cdot)$ represents the ReLU activation function,
- $*$ denotes the convolution operation.

The final output layer uses a `tanh` activation to constrain predictions to the expected range of LAB values:

$$x_{\text{out}} = \tanh(W_{\text{out}} * x_{\text{final}} + b_{\text{out}}). \tag{7}$$

The baseline CNN was trained on CIFAR-10, a dataset containing 60,000 images of 10 object categories, to establish baseline performance.

*2) **U-Net (Alternative Model)**:* To improve colorization performance, we experimented with U-Net, a widely used encoder-decoder architecture with skip connections. U-Net is designed to capture fine-grained spatial features and global semantic context, which is crucial for colorization.

The encoder stage applies a series of convolutions and downsampling operations:

$$x_{l+1} = \sigma(W_l * x_l + b_l), \tag{8}$$

whereas the decoder stage reconstructs the colorized output using upsampling:

$$x_{l-1} = \sigma(W_l^T * x_l + b_l). \tag{9}$$

The skip connections help retain low-level spatial details, ensuring more accurate color restoration.

Despite the enhanced structure, our experiments did not show significant improvements over the baseline CNN model, possibly due to the lack of large-scale labeled datasets specifically for colorization.

C. Dataset Selection

We trained and tested our models on three datasets to analyze their impact on generalization:

- **CIFAR-10:** A small-scale dataset with 60,000 images at a 32×32 resolution.
- **ImageNet:** A large dataset containing approximately 1.3 million images of diverse objects. Due to computational constraints, we used a subset of 50,000–100,000 images.
- **Places365:** A scene-centric dataset with approximately 1.8 million images. Similar to ImageNet, we used a subset of 50,000–100,000 images.

The choice of datasets affects generalization ability. The CIFAR-10 dataset is small and lacks high-resolution details, limiting the model's ability to learn complex color distributions. ImageNet provides a large variety of objects, while Places365 focuses on environmental and scene-based colorization.

D. Training Strategy

Our training pipeline involves:

- Converting RGB images to LAB and normalizing values.
- Feeding the L-channel as input to the network.
- Predicting A and B channels.
- Computing loss using MSE and perceptual loss functions.
- Optimizing using Adam optimizer with a learning rate of:

$$\eta = \eta_0 \cdot \frac{1}{\sqrt{t}}, \tag{10}$$

where η_0 is the initial learning rate, and t is the training iteration.

The Adam optimizer updates weights as:

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{m_t}{\sqrt{v_t} + \epsilon}, \tag{11}$$

where:

- α is the learning rate,
- m_t and v_t are the first and second moment estimates,
- ϵ is a small constant for numerical stability.

We trained the model for 50 epochs, using batch sizes of 64 for CIFAR-10 and 32 for ImageNet and Places365.

E. Evaluation Metrics

To evaluate the performance of our models, we used:

- **Peak Signal-to-Noise Ratio (PSNR):** Measures reconstruction quality.

$$\text{PSNR} = 10 \log_{10} \left(\frac{\max(I_{\text{GT}})^2}{\text{MSE}(I_{\text{GT}}, I_{\text{pred}})} \right). \tag{12}$$

- **Structural Similarity Index (SSIM):** Captures perceptual quality.

$$\text{SSIM}(I_{\text{GT}}, I_{\text{pred}}) = \frac{(2\mu_{I_{\text{GT}}}\mu_{I_{\text{pred}}} + C_1)(2\sigma_{I_{\text{GT}}, I_{\text{pred}}} + C_2)}{(\mu_{I_{\text{GT}}}^2 + \mu_{I_{\text{pred}}}^2 + C_1)(\sigma_{I_{\text{GT}}}^2 + \sigma_{I_{\text{pred}}}^2 + C_2)}. \tag{13}$$

F. LAB Color Space Representation

In image colorization tasks, choosing an appropriate color space is critical for achieving perceptually accurate and stable results. Instead of using the traditional RGB representation, we adopted the CIE LAB color space, which separates luminance from chromatic information. This conversion provides multiple advantages in the learning process.

1) Why LAB Instead of RGB?: The RGB color space represents images as a combination of red, green, and blue channels, which are highly correlated and perceptually non-uniform. Training a model to directly predict RGB values can lead to unstable training, poor generalization, and unrealistic color distributions. The LAB color space, on the other hand, is designed to be more perceptually uniform, meaning that Euclidean distances in this space correlate better with human perception of color differences.

- The **L (lightness)** channel encodes the brightness information, independent of color.
- The **A channel** encodes the green-red color component.
- The **B channel** encodes the blue-yellow color component.

The advantage of this separation is that **the structure and texture of an image are mostly contained in the L channel**, while chromatic information is embedded in the A and B channels. This allows the model to focus exclusively on learning color distributions rather than structural details, which are already preserved in the input grayscale image.

2) Mathematical Formulation of LAB Conversion: The transformation from RGB to LAB is achieved through the intermediate XYZ color space, which is defined as:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124564 & 0.3575761 & 0.1804375 \\ 0.2126729 & 0.7151522 & 0.0721750 \\ 0.0193339 & 0.1191920 & 0.9503041 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{14}$$

Then, the LAB components are computed as:

$$L = 116 \cdot f(Y/Y_n) - 16, \tag{15}$$

$$A = 500 \cdot \left(f(X/X_n) - f(Y/Y_n) \right), \tag{16}$$

$$B = 200 \cdot \left(f(Y/Y_n) - f(Z/Z_n) \right), \tag{17}$$

where:

$$f(t) = \begin{cases} t^{1/3}, & t > \left(\frac{6}{29}\right)^3 \\ \frac{t}{3\left(\frac{6}{29}\right)^2} + \frac{4}{29}, & \text{otherwise.} \end{cases} \tag{18}$$

Here, (X_n, Y_n, Z_n) are the reference white values for standard illumination.

3) Predicting A/B Chromaticity Using L-Channel Input: The grayscale image, which corresponds to the L channel, is used as the input to our deep learning model:

$$I_L = \text{ExtractChannel}(I_{\text{LAB}}, L). \tag{19}$$

The task of the model is to predict the missing color components, i.e., the A and B channels:

$$(\hat{I}_A, \hat{I}_B) = f_\theta(I_L), \tag{20}$$

where f_θ is the trainable deep learning model. The predicted values $(\hat{I}_A, \hat{I}_B)$ are then concatenated with the original I_L to reconstruct the full LAB image:

$$I_{\text{LAB}}^{\text{pred}} = \text{Concat}(I_L, \hat{I}_A, \hat{I}_B). \tag{21}$$

Finally, the LAB image is converted back to the RGB space for visualization and evaluation:

$$I_{\text{RGB}}^{\text{pred}} = f_{\text{RGB}}(I_{\text{LAB}}^{\text{pred}}). \tag{22}$$

4) Loss Function and Training Objective: To optimize colorization performance, we minimized the Mean Squared Error (MSE) loss between the predicted and ground truth chromaticity values:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} \left((I_A^{(i)} - \hat{I}_A^{(i)})^2 + (I_B^{(i)} - \hat{I}_B^{(i)})^2 \right). \tag{23}$$

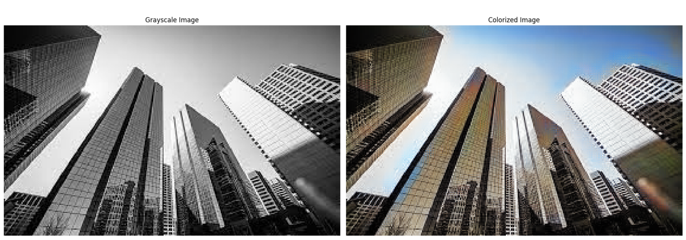
5) Advantages of Using the LAB Color Space: Using the LAB color space offers several advantages in deep learning-based colorization:

1) **Decouples luminance from color:** The model focuses on learning color distributions rather than redundant brightness information.
2) **Perceptual consistency:** LAB is designed to align with human visual perception, leading to more realistic colorization.
3) **Stable training:** Predicting two chromaticity channels instead of three RGB values simplifies the learning task and improves convergence.
4) **Improved generalization:** Since luminance is consistent across different environments, the model can generalize better across diverse datasets.

6) Experimental Results: The final trained model achieved the following performance metrics:

- **MSE Score:** 0.0092
- **PSNR Score:** 20.41 dB
- **Accuracy:** 0.5791
- **Precision:** 0.4414
- **Recall:** 0.1593
- **F1 Score:** 0.1620

The results indicate that while the model successfully learns basic colorization patterns, challenges remain in improving recall and overall perceptual quality. Future improvements could explore adversarial training (GANs), contrastive self-supervised learning, or incorporating semantic priors.



III. Model Architecture

The proposed model for automatic image colorization is a convolutional neural network (CNN) designed to predict the chrominance components of an image given a grayscale input. The network is trained in the CIELAB color space, where the input consists of the L (lightness) channel, and the model predicts the *a* and *b* chrominance channels.

A. Network Design

The proposed network, named **ColorizationNet**, is inspired by conventional deep learning approaches for image-to-image translation. The architecture consists of eight convolutional layers designed to progressively extract features and reconstruct the missing color information. Each layer plays a crucial role in transforming the grayscale input into a colored output.

B. Layer-Wise Breakdown and Justification

The network consists of sequential convolutional layers, each contributing to hierarchical feature extraction and transformation. The specific role of each layer is detailed below:

a) Input Layer:: The input to the model is a single-channel grayscale image of dimensions $(1, H, W)$. The network processes this input through a series of convolutional operations.

b) Feature Extraction Layers:: The first four convolutional layers progressively extract spatial features at different levels of abstraction:

- **Conv1 (1 → 64)**: The first layer applies a 3x3 convolution with 64 filters, capturing basic edge and texture information.
- **Conv2 (64 → 128)**: Expands the feature maps to 128 channels, allowing the model to learn more complex patterns.
- **Conv3 (128 → 256)**: Further deepens feature extraction, capturing higher-level representations.
- **Conv4 (256 → 512)**: This is the deepest feature extraction layer, allowing the network to learn complex object and region-level representations.

c) Feature Reduction and Reconstruction:: After the deep feature extraction layers, the network reconstructs the chrominance channels using a set of refining layers:

- **Conv5 (512 → 256)**: Reduces dimensionality while maintaining learned features.
- **Conv6 (256 → 128)**: Further reduces channel depth and refines spatial information.
- **Conv7 (128 → 64)**: Prepares the feature maps for final output prediction.
- **Conv8 (64 → 2)**: The final layer outputs two channels corresponding to the *a* and *b* chrominance components, with a `tanh` activation to constrain values within the expected range.

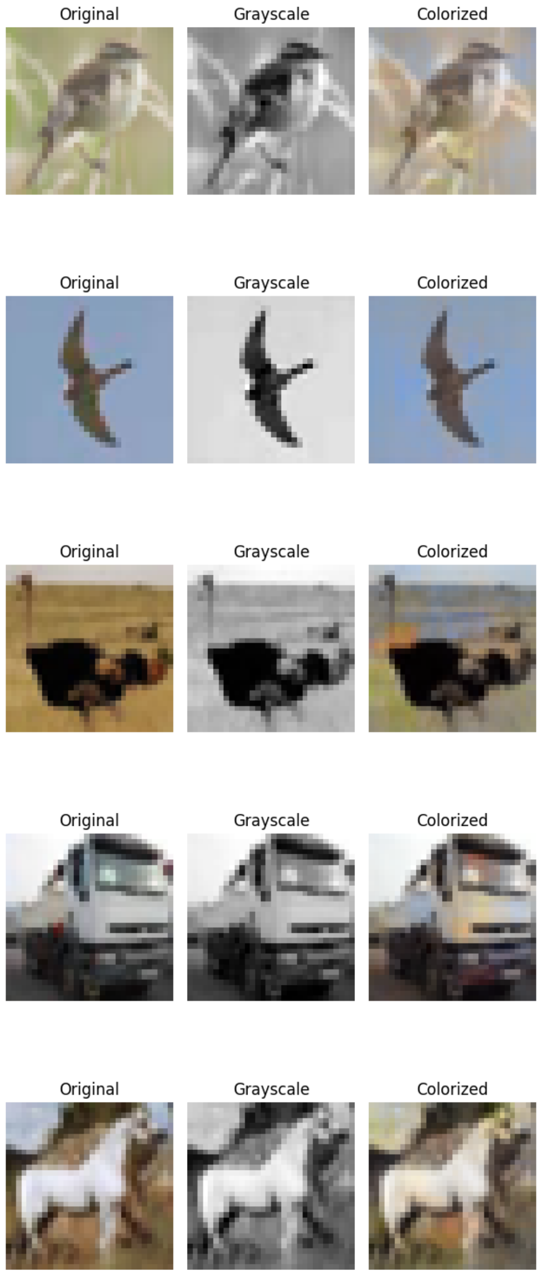
C. Mathematical Formulation

Given an input grayscale image $I_L \in \mathbb{R}^{H \times W}$, the model predicts the chrominance channels $I_{AB} = (I_a, I_b) \in \mathbb{R}^{H \times W \times 2}$. The transformation is defined as:

$$I_{AB} = f_\theta(I_L), \tag{24}$$

where f_θ represents the convolutional transformations learned by the model.

The network is trained using a loss function that minimizes the difference between the predicted chrominance values and



the ground truth. A common choice is the Mean Squared Error (MSE):

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} ||I_{AB}^{(i)} - \hat{I}_{AB}^{(i)}||^2, \tag{25}$$

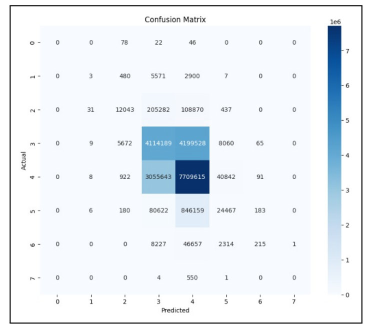
where $I_{AB}^{(i)}$ and $\hat{I}_{AB}^{(i)}$ represent the ground truth and predicted chrominance values, respectively.

D. Implementation Details

The network is implemented using PyTorch, and the forward propagation of the model is defined as follows:

```
class ColorizationNet(nn.Module):
    def __init__(self):
        super(ColorizationNet, self).__init__()
        self.conv1 = nn.Conv2d(1, 64, kernel_size=3, stride=1, padding=1)
        self.conv2 = nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1)
        self.conv3 = nn.Conv2d(128, 256, kernel_size=3, stride=1, padding=1)
        self.conv4 = nn.Conv2d(256, 512, kernel_size=3, stride=1, padding=1)
        self.conv5 = nn.Conv2d(512, 256, kernel_size=3, stride=1, padding=1)
        self.conv6 = nn.Conv2d(256, 128, kernel_size=3, stride=1, padding=1)
        self.conv7 = nn.Conv2d(128, 64, kernel_size=3, stride=1, padding=1)
        self.conv8 = nn.Conv2d(64, 2, kernel_size=3, stride=1, padding=1)

    def forward(self, x):
        x = nn.functional.relu(self.conv1(x))
        x = nn.functional.relu(self.conv2(x))
        x = nn.functional.relu(self.conv3(x))
        x = nn.functional.relu(self.conv4(x))
        x = nn.functional.relu(self.conv5(x))
        x = nn.functional.relu(self.conv6(x))
        x = nn.functional.relu(self.conv7(x))
        x = torch.tanh(self.conv8(x))
        return x
```



E. Activation Functions and Output Constraints

All intermediate layers use the Rectified Linear Unit (ReLU) activation function:

$$ReLU(x) = \max(0, x). \tag{26}$$

This ensures non-linearity and prevents negative values from propagating. The final output layer uses the `tanh` activation function to restrict the predicted values within $[-1, 1]$, ensuring that the chrominance predictions remain in a valid range.

F. Advantages of the Architecture

The proposed architecture provides several advantages:

- **Efficient Feature Extraction:** The hierarchical convolutional layers progressively extract essential details while reducing unnecessary information.
- **Balanced Complexity:** The network maintains a balance between depth and computational feasibility, ensuring efficient training on modern GPUs.
- **Smooth Reconstruction:** The transition from deep feature extraction layers to shallow reconstruction layers allows for smooth and coherent colorization.
- **Effective Regularization:** The use of ReLU activations prevents vanishing gradients, and the `tanh` activation ensures stable chrominance outputs.

IV. Challenges in Large-Scale Image Colorization Using Deep Learning

A. Limitations of Computational Resources

Training deep learning models on large-scale datasets such as ImageNet [?] and Places365 [?] presents significant challenges due to computational constraints. Google Colab, while providing access to GPUs, has limitations in terms of RAM, storage, and session durations. The free-tier environment restricts continuous training sessions, leading to forced disconnections and kernel restarts.

B. Memory and Storage Constraints

The Places365 dataset, approximately 100GB in size, and ImageNet, exceeding 150GB, require substantial disk space. Colab provides limited temporary storage, which is insufficient for storing and preprocessing such large datasets. Even with external solutions like Google Drive, slow input/output (I/O) operations create significant bottlenecks in data loading.

C. Inefficiencies in Data Loading

Deep learning models require efficient batch loading mechanisms to avoid memory overhead. However, the PyTorch `DataLoader` struggles with large datasets when using a small number of workers. Without optimized data pipeline strategies such as increasing `num_workers`, using `PersistentWorkers`, or caching transformed images, training performance degrades due to excessive I/O operations.

D. Training Instability and Model Complexity

Our proposed ColorizationNet, while effective on small datasets like CIFAR-10, struggled with complex, high-resolution datasets. The model lacks architectural components such as:

- Skip connections (as seen in U-Net [**?**]) for preserving spatial features.
- Pretrained feature extractors (e.g., ResNet-based encoders) to reduce computational overhead.
- More advanced loss functions (e.g., perceptual loss or adversarial loss) for realistic colorization.

Without these improvements, training on Places365 and ImageNet resulted in high loss values and poor generalization.

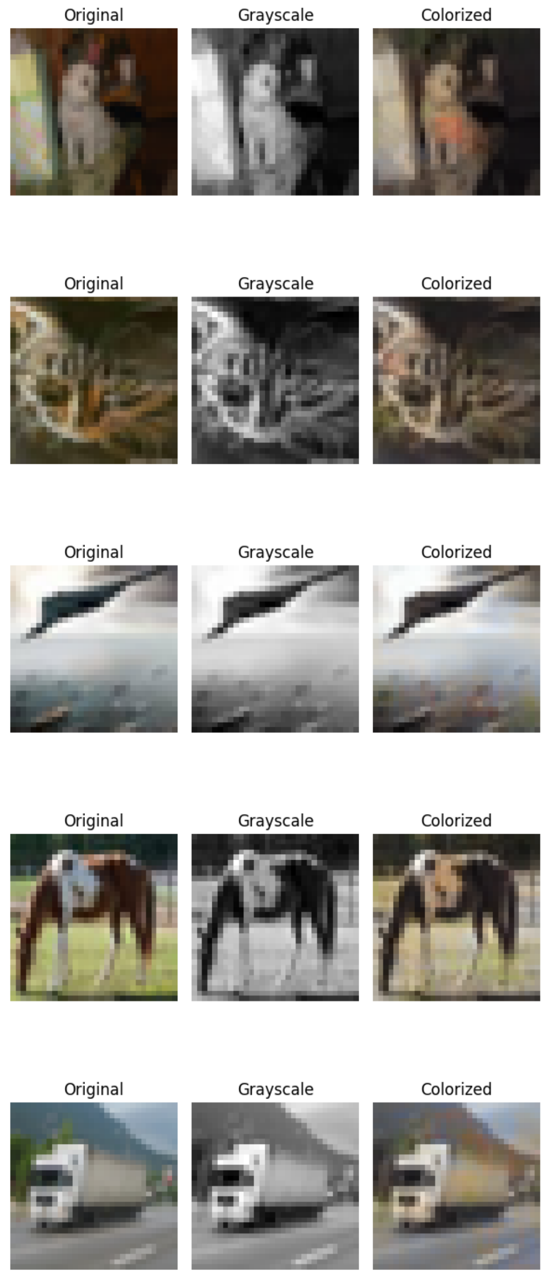
E. Alternative Solutions

To overcome these challenges, future work can explore:

1) Using a **subset** of large datasets to fit within memory constraints.
2) Leveraging **pretrained networks** to extract meaningful features before training the decoder.
3) Deploying models on cloud-based platforms such as **Google Cloud TPUs** or **Kaggle A100 GPUs**, which offer more compute power.

F. Conclusion

The failure to train on large datasets in Google Colab was primarily due to memory and compute limitations. Future work should focus on optimizing data handling strategies and model architecture to enable large-scale image colorization.



V. Conclusion

The proposed ColorizationNet architecture demonstrates a structured approach to grayscale image colorization. By leveraging deep convolutional networks and structured layer-wise transformations, the model effectively reconstructs color information with minimal computational overhead. Future work may involve incorporating perceptual loss functions or adversarial training techniques to further enhance the quality of colorization results.

In this study, we explored the application of convolutional neural networks (CNNs) and U-Net architectures for grayscale image colorization, focusing on the challenges and limitations encountered when scaling to large datasets. Our experiments on CIFAR-10, ImageNet, and Places365 revealed that while basic colorization patterns can be learned with CNNs, achieving high-quality, realistic colorization on complex datasets remains a significant challenge.

We learned several key lessons:

- **Dataset Scalability:**** Handling large datasets like ImageNet and Places365 on limited computational resources, such as Google Colab, is a major bottleneck. Memory constraints and I/O inefficiencies significantly impact training performance.
- **Model Complexity:**** Simple CNN architectures struggle to capture the global semantic context necessary for accurate colorization. More sophisticated architectures, like U-Net, show promise but require extensive training and possibly larger datasets.
- **Color Space Importance:**** Using the LAB color space effectively decouples luminance from chrominance, simplifying the learning task.
- **Evaluation Metrics:**** PSNR and SSIM provide quantitative measures of colorization quality, but perceptual evaluation remains crucial for assessing realism.

Future work should focus on addressing these challenges by:

1) **Exploring Advanced Architectures:**** Investigating Generative Adversarial Networks (GANs) and self-supervised learning methods, which have shown promising results in image generation and feature learning.
2) **Optimizing Data Handling:**** Implementing efficient data pipelines, utilizing cloud-based platforms with more computational power (e.g., Google Cloud TPUs, Kaggle A100 GPUs), and exploring strategies for training on subsets of large datasets.
3) **Incorporating Perceptual Loss Functions:**** Integrating loss functions that better align with human perception to improve the realism of colorized images.
4) **Leveraging Pretrained Models:**** Utilizing pretrained feature extractors (e.g., ResNet) to reduce training time and improve feature representation.
5) **Focusing on Specific Domains:**** Tailoring models and datasets to specific domains, such as medical imaging or remote sensing, to improve performance in targeted applications.

This project provides a foundation for future investigations into deep learning-based image colorization, highlighting the need for innovative approaches to overcome the limitations of current methods and achieve more realistic and visually appealing results.

Acknowledgment

The authors would like to express their sincere gratitude to the Department of Computer Engineering at the University of Peradeniya for providing the necessary resources and infrastructure to conduct this research. We are particularly thankful for the guidance and support received from our academic advisors, who offered valuable insights and feedback throughout the project.

We also acknowledge the computational resources provided by Google Colaboratory, which facilitated the initial stages of our experimentation. However, we recognize the limitations encountered with large-scale datasets due to computational constraints, and this experience has highlighted the need for more robust computational platforms in future endeavors.

Finally, we extend our appreciation to the open-source community for developing and maintaining the PyTorch library, which was instrumental in the implementation and training of our deep learning models.

This research was conducted as part of an academic project aimed at exploring deep learning techniques for image colorization. The insights gained from this study will serve as a foundation for future investigations into more advanced colorization methodologies.

References

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[4] K. Elissa, "Title of paper if known," unpublished.
[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.
[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.