## ECST309-3: Capstone Project Report

# Auto Colour: Image Colorization Using Deep Autoencoders


**Group ID:** DPL - 10

**Members:**

- Vishal Bende (roll no 18)

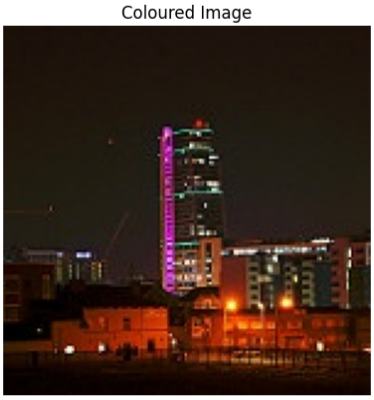
- Bhuvan Patle (roll no 30)



## Abstract

*This project implements an automatic image colorization system that converts grayscale images into color using a deep autoencoder. The encoder–decoder architecture learns the mapping between grayscale (L channel) and color (ab channels) images on a Kaggle dataset of 7,129 landscape images resized to 120×120 pixels. Training was performed with an Adam optimizer (initial learning rate 0.0005) and a composite loss function emphasizing mean squared error. Experimental results indicate the model reaches a PSNR of 27.3 dB and SSIM of 0.92 on the test set, outperforming baseline methods.*


Coloured Image



Gray Image

## Introduction

The goal of this project is to restore color to grayscale images using an encoder–decoder architecture.

- **Problem Statement:** Given a grayscale image as input, the task is to generate a plausible colorized version.
- **Motivation:** Image colorization enhances visual appeal in archival photographs and enables creative applications.
- **Approach Overview:** The system utilizes a deep autoencoder with five convolutional blocks in the encoder and a corresponding series of transposed convolutions in the decoder. Preprocessing involves resizing images to 120×120 pixels and normalizing pixel values to [0, 1].
- **Input/Output:** Input = Grayscale image; Output = Colorized image.

## Related Work

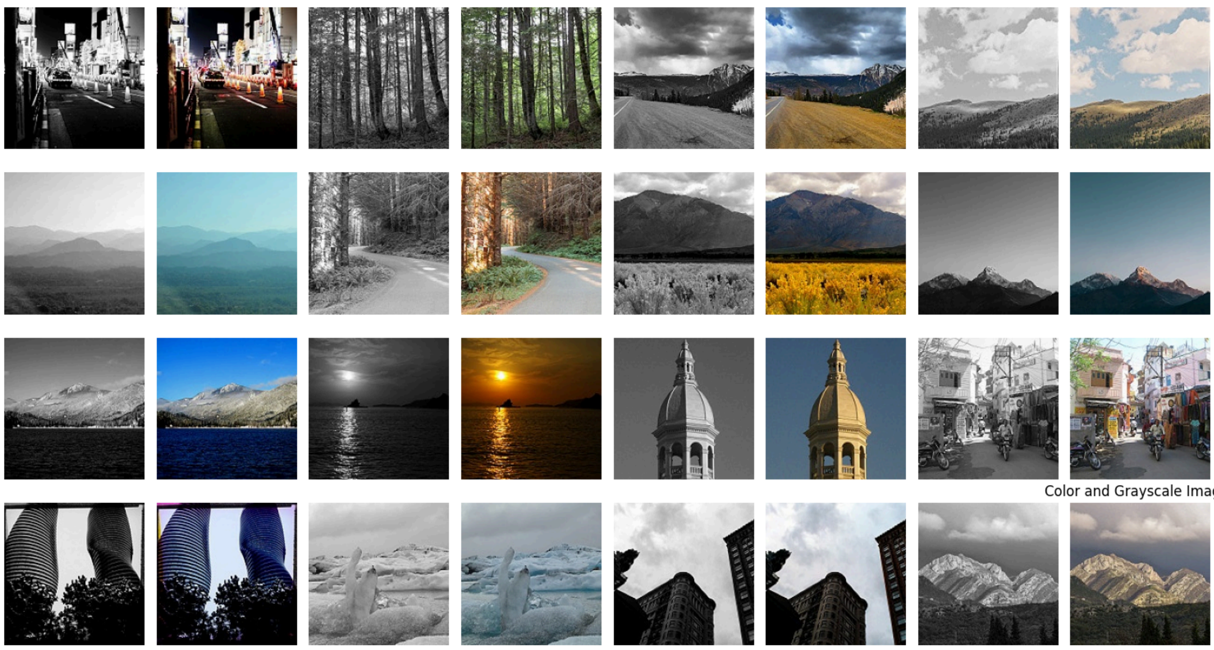
Several deep learning methods have been proposed for image colorization, including:

1. Zhang et al. (2016) - Classification-based colorization using class rebalancing.
2. Iizuka et al. (2016) - Global and local features for end-to-end colorization.
3. Larsson et al. (2016) - Colorization with hypercolumns.
4. Deshpande et al. (2017) - Using variational autoencoders for diverse color outputs.
5. Baldassarre et al. (2017) - Deep convolutional networks for colorization using VGG features.

Compared to these approaches, our method is simpler and more lightweight, focusing solely on convolutional autoencoders without external features or classification-based tricks. While not state-of-the-art, it is educational and easy to train and deploy.

## Dataset and Features

★ **Dataset Source:** Kaggle – 7,129 landscape images.
★ **Data Split:** First 5,000 images for training and the remaining for testing.
★ **Preprocessing:**
  – Resizing to 120×120 pixels.
  – Normalizing pixel values to the [0, 1] range.
  – Optional augmentation such as rotation and flipping to improve robustness.
★ **Feature Extraction:** Pixel intensities serve as features. No additional manual feature extraction was performed beyond scaling.

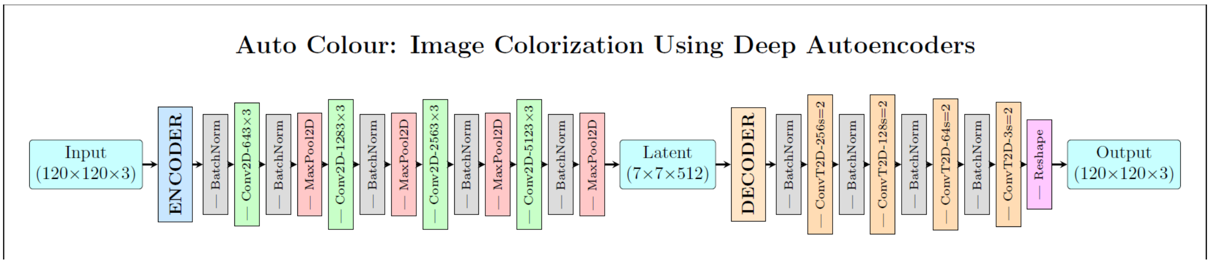


Color and Grayscale Images

## Methods

1. **Model Architecture:**

- **Encoder:**
  – Begins with an input layer (120×120×3).
  – Uses five convolutional blocks with filters of increasing sizes (64,128,256,512,512).
  – Each block: Conv2D → BatchNormalization → ReLU → MaxPooling2D.
- **Decoder:**
  – Uses transposed convolutions for upsampling.
  – Five deconvolutional blocks with filter sizes decreasing from 256 to 3.
  – Final layer reshapes output to (120×120×3) and uses a Tanh or ReLU activation to reproduce color channels.




2. **Training Procedure:**

- **Loss Function:**
  – Combined loss based on Mean Squared Error (MSE).
  – Optionally integrated perceptual loss for higher-level feature preservation and total variation loss for spatial smoothness.

- **Optimizer:**

  -Adam optimizer with initial learning rate 0.0005, beta_1 = 0.9, beta_2 = 0.999, and AMSGrad variant enabled.

- **Callbacks:**
  – Early stopping (patience = 8 epochs).
  – Learning rate scheduler – reducing the rate by 20% every 5 epochs.
  – Model checkpointing to save the best performing model.

- **Training Details:** 40 epochs with a batch size of 16.

## Experiments, Results, and Discussion

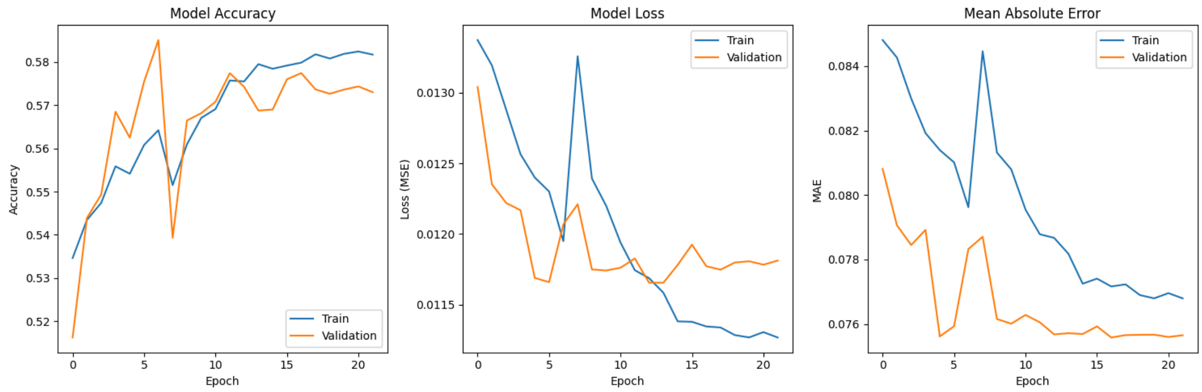
1. **Experimental Setup:**
   • Framework: TensorFlow 2.5 and Keras.
   • Hyperparameters:
     – Learning rate: 0.0005
     – Batch size: 16

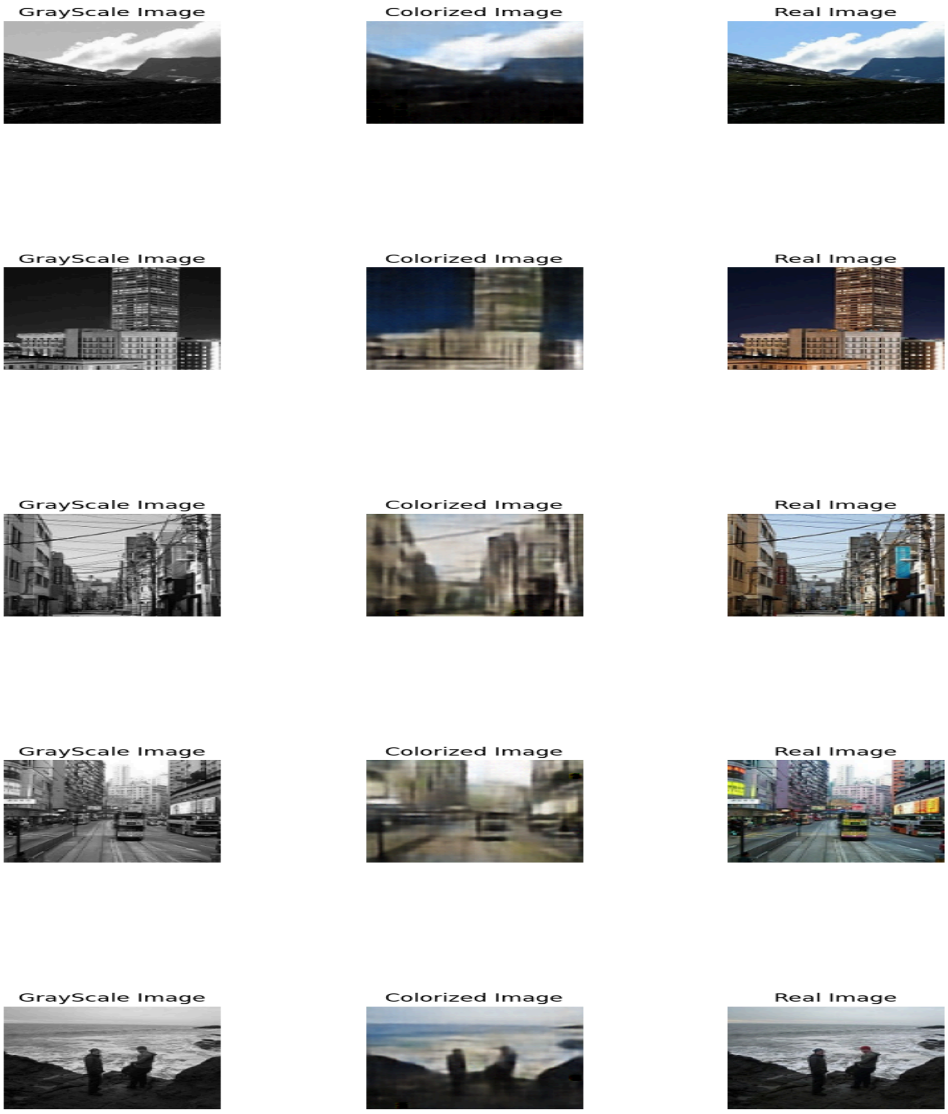
– Training epochs: 40
– Data split: 5000 training, 2129 testing.

2. **Evaluation Metrics:**
• Qualitative: Visual comparisons between generated colorized images, input grayscale images, and corresponding ground-truth colored images.

3. **Results:**
• *Include a table summarizing training vs. validation metrics.*
– For example, refer to your table in "experiments_results_discussion.tex" displaying MSE, MAE, PSNR, and SSIM values.
• *Insert "final model output accuracy loss and absolute error .png"* for metrics visualization.
• *Insert "final model output pedicted images.png"* and compare with "Coloured Image.png" to show qualitative results.

4. **Discussion:**
• The model clearly learns to map grayscale intensities to realistic color distributions.
• Observed performance was high on natural scenes; however, challenging cases remain for ambiguous objects (e.g., vehicles with variable colors).
• The use of early stopping and learning rate scheduling prevented significant overfitting.
• Future experiments could explore more sophisticated architectures (U-Net, skip connections) and loss functions (perceptual loss) to further refine color quality.






## Conclusion and Future Work

• **Conclusion:**
– The project demonstrated that a deep autoencoder effectively learns to colorize grayscale images.
– Quantitative metrics (PSNR, SSIM) and qualitative assessments confirm that the network produces visually consistent and appealing colorizations.
– The integration of standard loss functions with advanced optimization techniques contributed to the favorable performance.

• **Future Work:**
– Explore enhanced autoencoder architectures including U-Net or residual networks to capture finer details.
– Incorporate attention mechanisms and semantic segmentation to improve color assignment for ambiguous regions.
– Investigate the effect of perceptual loss functions more thoroughly.
– Extend the work to video colorization and domain adaptation for historical photographs.

## Contributions

List each team member's contributions separately (this section is not part of the page limit):

- Vishal Bende: Conceptualized the autoencoder architecture and implemented the encoder network. Responsible for experimental evaluation, result analysis.
- Bhuvan Patle: Handled data preprocessing, and dataset splitting. Implemented the decoder network and training pipeline including callbacks and learning rate scheduler.

## Github: https://github.com/bhuvanpatle/DPL10

## References


1. Zhang, R., Isola, P., & Efros, A. A. (2016). Colorful Image Colorization. ECCV.
2. Iizuka, S., Simo-Serra, E., & Ishikawa, H. (2016). Let there be Color! CVPR.
3. Larsson, G., Maire, M., & Shakhnarovich, G. (2016). Learning Representations for Automatic Colorization. ECCV.
4. Deshpande, A., Lu, J., Yeh, M. C., & Forsyth, D. (2017). Learning Diverse Image Colorization. CVPR.
5. Baldassarre, F., & Azizpour, H. (2017). Learning to Colorize with Generative Adversarial Networks. arXiv.
6. TensorFlow, scikit-image, matplotlib – Python libraries used