Advanced Topics Of Deep Learning - Final Project

Ofek Naim (ID. 314782996), Student name ID. 205417678)

Submitted as final project report for Advanced Topics Of Deep Learning course, Colman, 2025

1 Introduction

Image colorization is a challenging problem in computer vision that involves transforming grayscale images into their realistic color counterparts. This project applies deep learning techniques to automate the colorization process, using convolutional neural networks (CNN) and GAN.

2 Method A

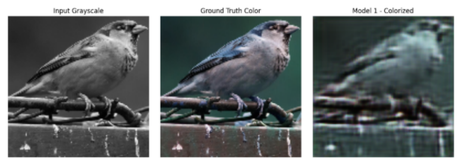
For this project, we implemented a CNN-based autoencoder architecture for image colorization. The model follows an encoder-decoder structure, which extracts features from a grayscale image and reconstructs a colorized version. We developed and compared two architectures:

2.1 Architecture A

For this architecture, we used convolutional autoencoder to extract features from grayscale images and reconstruct their colorized version with dropout to reduce overfitting and early stopping to prevent unnecessary training cycles.

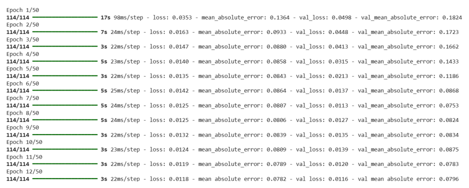
2.1.1 Hyper Parameters

We used 60 percent of the data for training, 20 percent for test and 20 percent for validation Kernel Size: 3x3 Dropout : 0.1 Stride : 2 Activation Function : Leakyrelu Optimizer : Adam Loss Function : Mean Squared Error (MSE) Patience : 5 Epochs (If validation loss does not improve for five consecutive epochs, the model stops training and restores the best-performing weights) Batch Size : 16



2.1.2 Performance

The performance of our model did not meet expectations, as early stopping was triggered after just 12 epochs, indicating that the model was not improving beyond this point. Despite using dropout and early stopping to prevent overfitting, the chosen hyperparameters did not allow the model to learn meaningful colorization features effectively.



```
Epoch 1/50
114/114 ━━━━━━━━━━━━━━━━━━━━ 17s 98ms/step - loss: 0.0353 - mean_absolute_error: 0.1364 - val_loss: 0.0498 - val_mean_absolute_error: 0.1824
Epoch 2/50
114/114 ━━━━━━━━━━━━━━━━━━━━ 7s 24ms/step - loss: 0.0163 - mean_absolute_error: 0.0933 - val_loss: 0.0448 - val_mean_absolute_error: 0.1723
Epoch 3/50
114/114 ━━━━━━━━━━━━━━━━━━━━ 3s 22ms/step - loss: 0.0147 - mean_absolute_error: 0.0880 - val_loss: 0.0413 - val_mean_absolute_error: 0.1662
Epoch 4/50
114/114 ━━━━━━━━━━━━━━━━━━━━ 5s 23ms/step - loss: 0.0140 - mean_absolute_error: 0.0858 - val_loss: 0.0315 - val_mean_absolute_error: 0.1433
Epoch 5/50
114/114 ━━━━━━━━━━━━━━━━━━━━ 3s 22ms/step - loss: 0.0135 - mean_absolute_error: 0.0843 - val_loss: 0.0213 - val_mean_absolute_error: 0.1186
Epoch 6/50
114/114 ━━━━━━━━━━━━━━━━━━━━ 5s 25ms/step - loss: 0.0142 - mean_absolute_error: 0.0864 - val_loss: 0.0137 - val_mean_absolute_error: 0.0868
Epoch 7/50
114/114 ━━━━━━━━━━━━━━━━━━━━ 5s 24ms/step - loss: 0.0125 - mean_absolute_error: 0.0807 - val_loss: 0.0113 - val_mean_absolute_error: 0.0753
Epoch 8/50
114/114 ━━━━━━━━━━━━━━━━━━━━ 5s 24ms/step - loss: 0.0125 - mean_absolute_error: 0.0806 - val_loss: 0.0127 - val_mean_absolute_error: 0.0824
Epoch 9/50
114/114 ━━━━━━━━━━━━━━━━━━━━ 3s 22ms/step - loss: 0.0132 - mean_absolute_error: 0.0839 - val_loss: 0.0135 - val_mean_absolute_error: 0.0834
Epoch 10/50
114/114 ━━━━━━━━━━━━━━━━━━━━ 3s 23ms/step - loss: 0.0124 - mean_absolute_error: 0.0809 - val_loss: 0.0139 - val_mean_absolute_error: 0.0875
Epoch 11/50
114/114 ━━━━━━━━━━━━━━━━━━━━ 3s 22ms/step - loss: 0.0119 - mean_absolute_error: 0.0789 - val_loss: 0.0120 - val_mean_absolute_error: 0.0783
Epoch 12/50
114/114 ━━━━━━━━━━━━━━━━━━━━ 3s 22ms/step - loss: 0.0118 - mean absolute error: 0.0782 - val loss: 0.0116 - val mean absolute error: 0.0796
```

The result after 12 epochs:

We can see that the model is beginning to add color to the images, but the results are not at the level of accuracy and realism that we aim to achieve.

2.2 Architecture B

In Architecture B, we removed early stopping and dropout to allow the model to train for more epochs without interruptions. This approach gives the network more time to learn complex patterns and improve colorization quality, rather than stopping training prematurely.

We used Conv2DTranspose instead of UpSampling2D

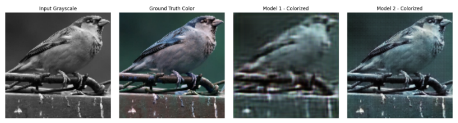
2.2.1 Hyper Parameters

We used the same hyperparameters as in Architecture A, ensuring that the only changes were the removal of early stopping and dropout. This allowed us to evaluate the impact of these modifications on the model's ability to learn and generate more accurate colorized images.

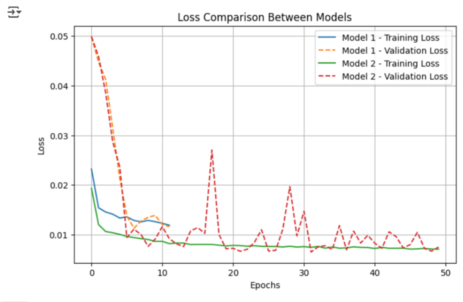
2.2.2 Performance

Provide information about your second experiment (changing hyperparameters). Make sure this part is clear to understand, provide as much details as possible. Provide results with tables and figures.

This is the result after 50 epochs compare to Architecture A:

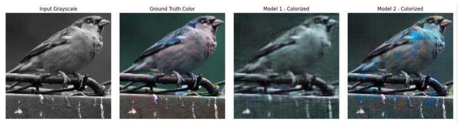


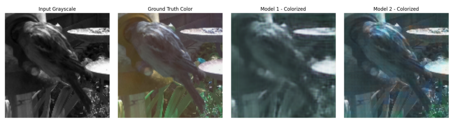
This is our Loss Comparsion between models:



as we can see, model A stop after 12 epochs, while model B keep running and getting better.

After 300 epochs, we can start seeing a better results.





3 Code

Training Notebook: https://colab.research.google.com/drive/1pBoKfUHvnakcTsBc1S4ohoaP7qw-H0Bi?usp=sharing

Inference Notebook: https://colab.research.google.com/drive/1cXRaU1VWygZHL1JDDJqj9uQJjXSGpubI

4 Method B

In this project, we employ a Pix2Pix-based Generative Adversarial Network (GAN) with a U-Net architecture to perform image colorization. The goal is to take grayscale images as input and generate realistic colorized versions while preserving details and structure.

4.1 Architecture

The **Generator** is responsible for transforming a grayscale input (128×128×1) into a colorized output (128×128×3). We used a U-Net architecture, which is an encoder-decoder network with skip connections. The encoder extracts spatial features, while the decoder reconstructs a full-color image. The skip connections help preserve fine details.

The **Discriminator** determines whether an image is real or fake.

4.1.1 Hyperparameters

Generator: We used 7 convolutional layers (`Conv2D`) with stride = 2 (reduces spatial size). LeakyReLU activation Batch Normalization

Bottleneck Layer: The deepest part of the network, capturing the compressed feature representation.

Decoder (Upsampling): Uses Conv2DTranspose to restore the original size.

tanh Activation function

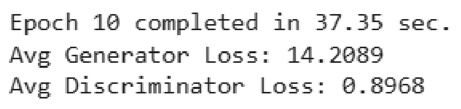
Discriminator: Conv2d layers LeakyRELU activation Batch Normalization Sigmoid for the output layer

Binary Cross-Entropy + L1 Loss

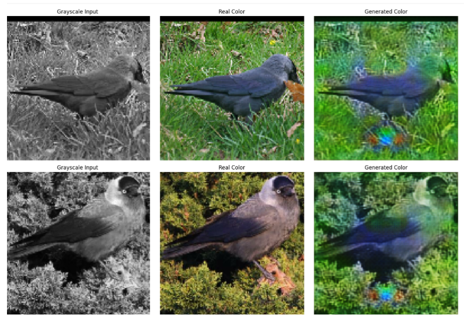
4.1.2 Performance

After 10 epochs, we still haven't achieved the desired results.

Looking at the average Generator loss and average Discriminator loss, we can see that the Generator loss remains high, indicating that the model still needs further improvement.

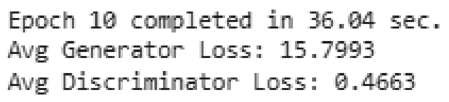


```
Epoch 10 completed in 37.35 sec.
Avg Generator Loss: 14.2089
Avg Discriminator Loss: 0.8968
```



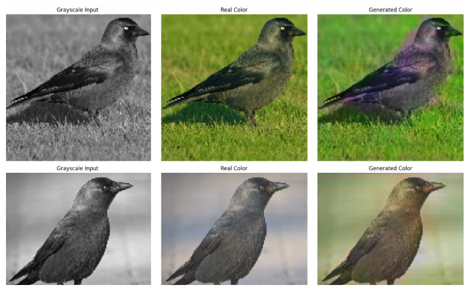
Generated images looking not bad, but we still looking for better results

After more 10 epochs result getting better



```
Epoch 10 completed in 36.04 sec.
Avg Generator Loss: 15.7993
Avg Discriminator Loss: 0.4663
```

we achieved low Avg Discriminator Loss means that our Discriminator is improved and can tell if an image is generated or not but we still got high Avg Generator Loss



With every epochs we done, our model getting better and better

5 Code

Training Notebook: https://colab.research.google.com/drive/13hRTMtEL$_GMrRXVA9oMuqfzlyMUuL7LX$ $sharing$

Inference Notebook: https://colab.research.google.com/drive/1TbQcC234Qqc-cmDCsrplPDzhO-qXzDCS?usp=sharing