Colourization of Greyscale Images using Deep Learning

J Sai Sri Hari Vamshi
indian Institute of Technology,
Hyderabad, India
ai21btech11014@iith.ac.in

Aayush S. Prabhu
Indian Institute of Technology,
Hyderabad, India
ai21btech11002@iith.ac.in

Rudransh Mishra
Indian Institute of Technology,
Hyderabad, India
ai21btech11025@iith.ac.in

Chandrahas Reddy D
Indian Institute of Technology,
Hyderabad, India
ai21btech11010@iith.ac.in

Vaishnavi W
Indian Institute of Technology,
Hyderabad, India
ai20btech11025@iith.ac.in

April 26, 2024

Abstract

Taking old black-and-white images and converting them into colour images is an ongoing effort in deep learning. It helps us better visualise records and bring to life their stories. Historically, Manual-colourization has been a skill-intensive and slow process; however, in recent years, colourisation has also been achieved using machine learning models with great potential in this field. This study aims to create a robust deep-learning model to colourise greyscale images. We aim to create a new algorithm for colourisation on our own, including both the model and the dataset.

1. Introduction

Colourisation, the process of adding colour to monochromatic images, is an emerging field in machine learning research. The ability to breathe colour into historical photographs and old videos and enhance medical imaging has immense practical and aesthetic value. Traditional colourisation methods relied heavily on manual intervention, where expert artists painstakingly painted over greyscale images to add colour. Approaches like these were labour-intensive, time-consuming, and often prone to artistic biases. Also, the quality of colourisation would greatly depend on the skill of the artist. [17]

Initial efforts to partially automate this process involved scribble-based colourisation, which was a classical algorithm for image colourisation. [16] Here, the model would identify and separate different pixel patches, which would then have to be coloured by the user. While certainly an improvement, this still required a great deal of human intervention.

An early attempt to fully automate the process required the user to input a reference image alongside the greyscale image for the model to take the colour palette from. [17] However, this was still far from the goal of fully automating the process where a user could colourise several images without painstakingly preprocessing the inputs first.

In recent years, significant progress has been made in developing deep learning-based approaches for colourisation. These approaches leverage large datasets and innovative network architectures to achieve impressive results. [2] By learning from vast amounts of colour images, these models can infer plausible colour distributions for greyscale inputs, effectively automating the colourisation process.

In this study, we attempt to create such a model to take as in[put a greyscale image and output a colourised image using deep learning techniques. We aim to create a model that is robust and precise in order to colourise a wide variety of images in a natural manner.

2. Literature Review

We analysed 17 papers, which were chosen with the following considerations - Varied time of publication (new and old), Clear description of the methodology, Varied methodology, and Achieving good, natural results. These were then

analysed to achieve the conclusions described below.

Earlier techniques involved in image colourisation were centred around scribble-based colourisation, a method that did not require any machine learning algorithm. Here, the program would simply separate the image into blobs, which would then have to be individually coloured by a human expert. While this was sped up by software like Adobe Photoshop, many images would still take a user as much as one month. [5]

A better approach was achieved by taking as input a sample image, which would then serve as the reference image based on which the machine learning model would apply colour schemes on the input image. [17]

The first-ever paper to achieve fully automatic was an advancement of the image transfer approach - The model was trained to recognise patches from the image and compare them against a dataset to automatically get the the correct colour pattern for the specific patch. This was a major leap in image colourisation and led to several follow-up models. [5]

An improvement to this model was done by using Convolutional Neural Networks (CNNs) to learn the global and local image characteristics by training the model with labelled image data, using which it could identify the colour pattern for the image much more accurately. [8]

This was followed by several models that would be focused on using CNNs for feature extraction, followed by colourisation. [11] [19] [7] [4] [15] These would try to impart their own modifications to reach more natural colourisation for broader image types. These included modified testing parameters, such as using the Turing test with human examiners [19]. A common one would be to augment the CNN model with pre-trained image classifier segments, such as InceptionResnet V2 [4], or VGG16 [7] [18] that would help in feature extraction.

Some models first use separate models for extracting different information about the subject and the image in total and then recombine the two to get a higher level of accuracy [15] [16]. Models also utilised different output formats in order to try and achieve colours more accurate to the original image - These included formats like CIELUV [7], LAB [14] and Per Pixel Histograms [11] that use the greyscale image itself as a channel.

In recent years, Generative Adversarial Networks (GANs), cycleGANs in particular, alongside UNet-like models have gained prominence. Here, the Unet-like model is used to generate an image, which is then evaluated for a natural image colourisation. [14] [1] [2] [13]. They have given some of the highest accuracy and are still under research for further enhancements.

Addressing issues like blurriness and mode-mixing observed in GANs and VAEs, [3] presents conditional Invertible Neural Networks (cINNs) as a novel approach for conditional image generation and colorization.

Using Mixture Density Networks (MDN), muti-modal conditioning models have successfully generated multiple realistic and diverse color variations for a single grayscale image. [6]

Colorization Transformer [10] is the first application of transformers to image colorization. This novel approach with axial self-attention blocks sets a new standard, highlighting the versatility of transformer-based architectures in image processing tasks.

3. Preliminary Investigations

We review three different codes in this Section, all used to colourise images in different ways. We have done a traditional scribble-based colouriser and two which use deep learning models to colourise the black and white images.

3.1. Scribble-based Colourisation

Scribble based colourisation was a popular method prior to the advancements in Deep Learning. It involves scribbling lines colour onto a target greyscale images, from which the algorithm extrapolates those onto other areas of the image.

The code was taken from a python implementation of a 2004 paper, A novel method for colorization (Levin, Lischinski Weiss, 2004). The algorithm uses YIQ colour scheme, where Y denotes brightness and I and Q denote the colour parameters. This ensures that the overall brightness is the same as the input greyscale image. [12]

For each greyscale pixel in the "scribbled" image, the variance of the intensities of it's window area is calculated using the formula,

$$\sigma^2 = \frac{1}{N}\sum_{i=0}^{N}(Y(p_i) - \frac{1}{N}\sum_{i=0}^{N}Y(p_i))^2$$

and the similarity of pixels with each other is decided by,

$$\omega \propto e^{(Y(p_0)-Y(p_i))^2/2\sigma^2}$$

This is then used to solve for the final colour parameters I and Q.

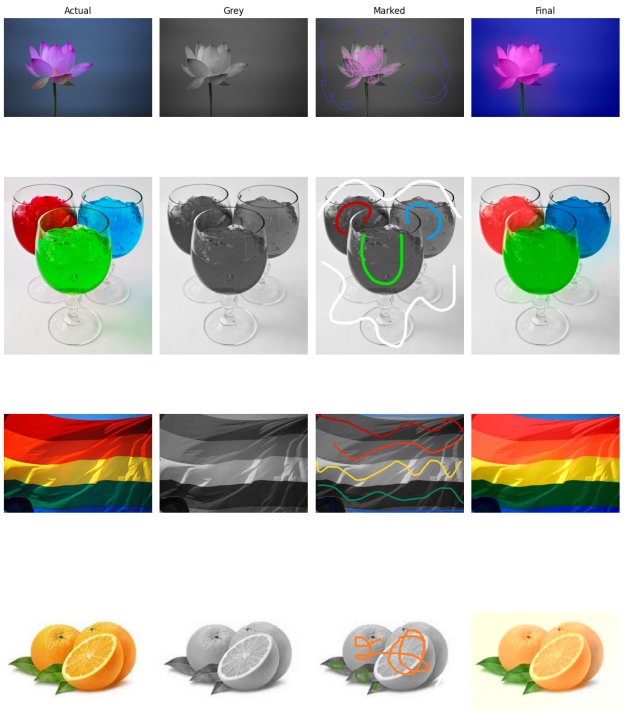

Figure 1. Scribble Based Colourisation Method

As we can see, this works well for simple images, but requires a human to identify all the colours correctly and scribble for each image. We see the solutions to this issue below.

3.2. Auto-Colorization

The Automatic Colorization model is a model developed by AlexSeongJu-Sr that automatically colours a given grey-scale image using a Deep Convolutional Network with an encoder-decoder structure. Normally, a model would have problems such as poor segmentation and dull results colours. The segmentation issues are handled by using a skip connection between the encoder and decoder inspired by the structure of a U-Net model. The colour issues are handled using L1 loss rather than L2 loss as the objective function of the discriminator, and the GAN is fine-tuned.

There are certain problems when colourising black and white images, including identifying the colour of objects like a car, where the same grey tone can be either red, green or blue of similar intensity. Therefore, obtaining complete results through Supervised Learning, which simply trains data, is difficult. We use GANs for this reason. Although GAN has the disadvantage of being unstable and taking a long time to learn, the results shown by GAN are more realistic.

The possibility of over-fitting is high because the data used for learning is very small. To compensate for this, data augmentation was applied. The augmentation that was applied is 256x256 Randomized Crop and RandomHoriontalFlip.

The generator consists of an encoder and a decoder. The encoder is responsible for extracting features from the image and has the form of Resnet50 with the last FC Layer removed. The decoder is responsible for changing the feature map obtained from the encoder back into an ab map of 2 x 256 x 256. The accuracy and loss plots from the training are shown below. (Encoder Loss 2, Generator Loss 3)

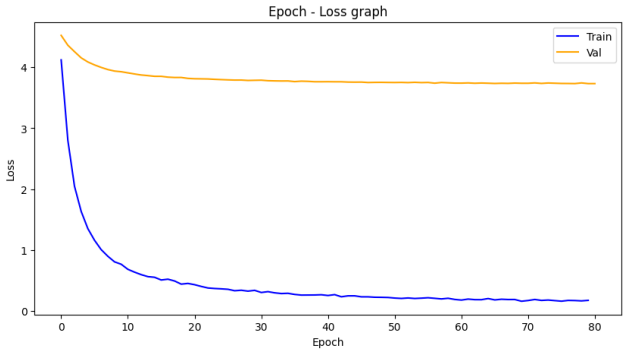

Figure 2. The Encoder Epoch - Loss Graph

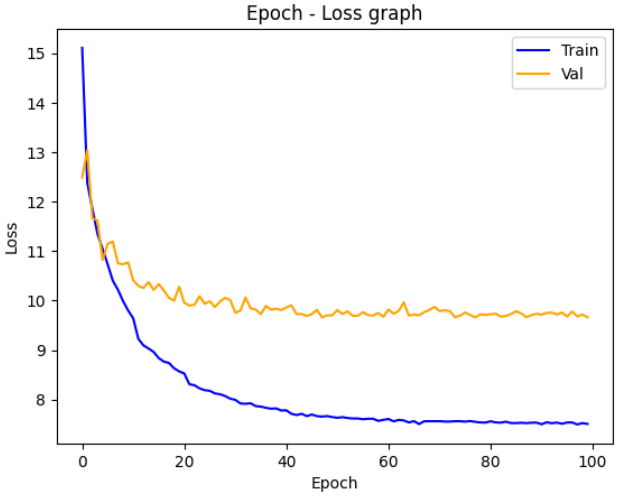

Figure 3. The Generator Overall Epoch - Loss graph

The GAN consists of the generator from above and a Discriminator that distinguishes real and fake images from the generator's output and ground truth. The Discriminator was created with reference to the DCGAN paper for smooth learning and is a continuation of the Conv layer – BN layer

– LeakyReLU. In the last layer, Sigmoid is passed to obtain the probability value.

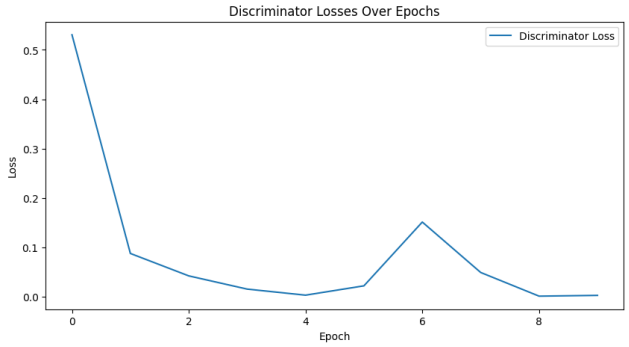

Figure 4. Discriminator loss During GAN Training

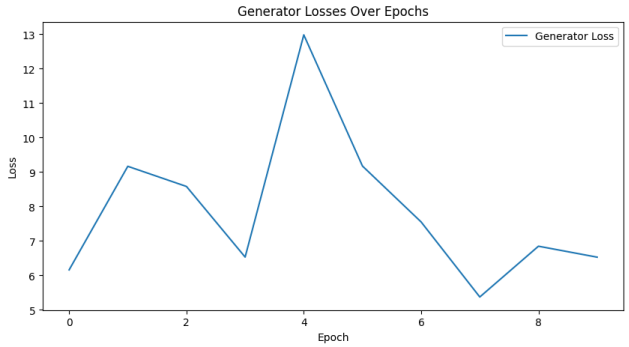

Figure 5. Generator loss During GAN Training

The Objective function used by the Discriminator is:

$$E[log(D(x))] + E[log(1 - D(G(z))]$$

The discriminator learns to maximise the above Objective Function, and the generator learns to minimise it. To solve this problem of selecting the correct colour while maintaining the brightness of the result, when learning the generator in the GAN model, the existing loss was multiplied by a certain ratio (ρ) to L1 and then added. As a result, the final loss we used in the GAN model is as follows.

$$E[log\ log(D(x))] + E[log\ log(D(G(z)))] + \rho L_1(G(z), y)$$

Here, 'ρ' is a Hyper Parameter between 0 and 1. If it is close to 1, you will get results similar to those of existing generators without using GAN, and if it is close to 0, you will get results similar to those of using only GAN. After several experiments, we chose 0.1 to get a result closer to pure GAN usage. The variation of the Generator and Discriminator output losses over the epochs can be studied from the below plot:[6]

The Flower102 Dataset for categorising is used to train and test the model after preparing the data appropriately.

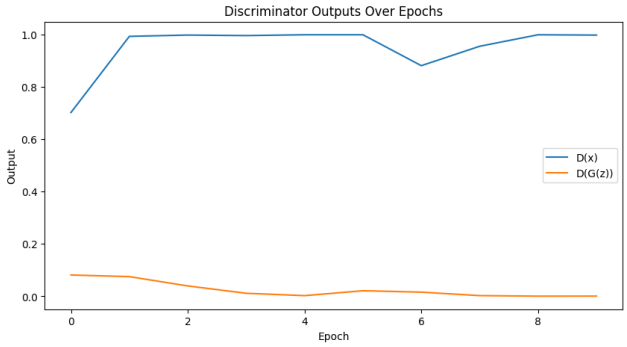

Figure 6. Discriminator and Generator Function Losses

The results from the Colourizing model are given below. (Ground Truth 7, Grey-scale 8, Colourised 9):

Figure 7. Ground Truth

3.3. Instance-aware Image colorization

Previous methods (including the one above) use deep neural networks to map input grayscale images to possible color outputs directly. Despite their impressive performance, these methods often fail on images that contain multiple objects. This is due to the fact that these models perform learning and colourization on the entire image, thereby ignoring object-level information.

In this paper, the authors (Jheng-Wei Su, Hung-Kuo Chu, and Jia-Bin Huang) proposed a model architecture made up of 3 parts:

(i) an off-the-shelf pre-trained object detection model to get

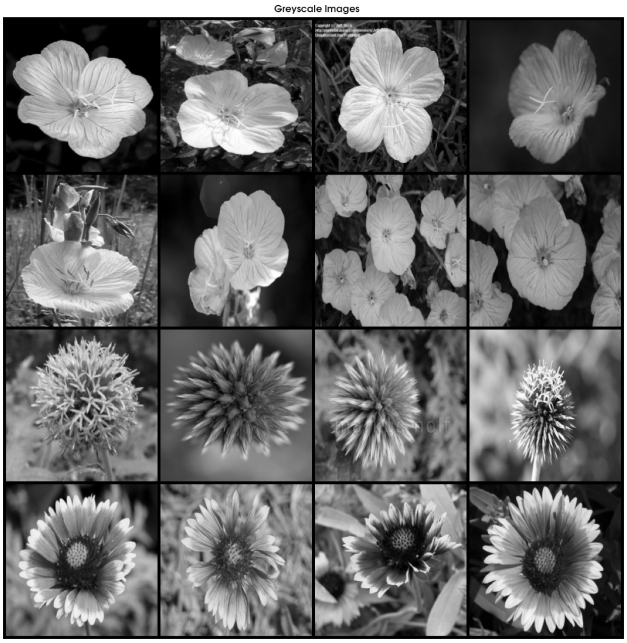

Figure 8. Grey-scaled Input Images

Figure 9. Colourised Results

cropped images of objects;

(ii) a instance colourization network to extract object-level features and a similar network to extract full-image features;

(iii) a fusion module to selectively blend the object-level and image-level features to get final colours.

This gave a much better performance on images that had objects of multiple categories.

Figure 10. Ground Truth

Figure 11. Grey-scaled Input Images

Training this model presented a computational challenge, due to which we had to take steps to reduce the time and memory requirements to a reasonable level. This led to a reduced level of accuracy, however, it allowed us to train the model and get an initial estimate of its advantages and disadvantages. The changes made by us were :-

1. The model was trained with 5,000 images instead of 118,000
2. The size of the images was reduced from 256x256 to 128x128 pixels
3. The batch size was increased from 16 to 96
4. The learning rate was doubled
5. The number of training epochs were reduced from 150 to 10.

The results of the image colourisation are given below in the images for Ground Truth 10, Grey-scaled Images 11

Figure 12. Colourised Results

and Colourised Results 12

We can see that it correctly applies shades of red, yellow and green to the objects, though it does not work well with blue objects and colours applied are dull. The full image model is very under-trained causing parts of the image that were not detected as objects to remain almost uncoloured. The fusion model is also not trained enough causing colors to bleed from an object to others around it.

Due to the reduction in epochs and the size of the dataset as well as the reduction in image resolution, the results were not as good as those achieved by the authors.

4. Proposed Method

Generative Adversarial Networks (GANs) have revolutionized image generation tasks, including image colorization. Unlike traditional methods that rely on explicit rules, GAN-based approaches learn colorization implicitly from data, capturing complex relationships between grayscale and color images. Their ability to preserve details, generate natural color distributions, and produce diverse outputs makes them ideal for image colorization tasks.

In this paper, we propose a GAN-based method for image colorization, leveraging adversarial training to generate realistic colorizations.

4.1. Methodology

Our code was inspired by Colorizing black / white images with U-Net and conditional GAN. This, in turn, was inspired by a paper [9] published in 2017. The model used a simple, hand-coded Unet Generator with a hand coded discriminator.

After analysing the code, we found several inefficiencies in the code:-

1. It did not have a proper encoder - Due to the basic, shallow CNN layout of its encoder, it would suffer from loss of image information while encoding, which would lead to the UNet outputs lacking context from the original image. The result of this was visible in the uncharacteristic and unnatural colour pallet in the coloured image outputs produced by the model.
2. The model was hard-coded, which meant for most users it was difficult to compare the architecture to any known reference architecture.

```
Unet(
  (model): DynamicUnet(
    (layers): ModuleList(
      (0): Sequential(
        (0): Sequential(
          (0): Conv2d(1, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (1): ReLU(inplace=True)
          (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (3): ReLU(inplace=True)
          (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
          (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (6): ReLU(inplace=True)
          (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (8): ReLU(inplace=True)
          (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
          (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (11): ReLU(inplace=True)
          (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (13): ReLU(inplace=True)
          (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (15): ReLU(inplace=True)
          (16): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (17): ReLU(inplace=True)
          (18): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
          (19): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (20): ReLU(inplace=True)
          (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (22): ReLU(inplace=True)
          (23): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (24): ReLU(inplace=True)
          (25): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (26): ReLU(inplace=True)
          (27): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
          (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (29): ReLU(inplace=True)
          (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (31): ReLU(inplace=True)
          (32): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (33): ReLU(inplace=True)
          (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (35): ReLU(inplace=True)
          (36): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
        )
      )
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): Sequential(
        (0): ConvLayer(
          (0): Conv2d(512, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (1): ReLU()
        )
        (1): ConvLayer(
          (0): Conv2d(1024, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (1): ReLU()
        )
      )
      (4): PixelShuffle_ICNR(
        (0): ConvLayer(
          (0): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1))
          (1): ReLU()
        )
        (1): PixelShuffle(upscale_factor=2)
      )
      (5): ResizeToOrig()
      (6): MergeLayer()
      (7): ResBlock(
        (convpath): Sequential(
          (0): ConvLayer(
            (0): Conv2d(513, 513, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
            (1): ReLU()
          )
          (1): ConvLayer(
            (0): Conv2d(513, 513, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          )
        )
        (idpath): Sequential()
        (act): ReLU(inplace=True)
      )
      (8): ConvLayer(
        (0): Conv2d(513, 2, kernel_size=(1, 1), stride=(1, 1))
      )
      (9): fastai.layers.ToTensorBase(tensor_cls=<class 'fastai.torch_core.TensorBase'>)
    )
  )
)
```

Figure 13. The Neural Network Architecture Used By the Model

Our approach focused on rewriting the Generator, by using a VGG19 based encoder. As VGG19 has been a long proven classification model, it was safe to assume that the

information retention in the model is high.

From this model, the final classification layer was removed and a simple decoder was added to complete the UNet model. Once this was done, we modified to discriminator slightly to improve its performance, and we set the model for training.

4.2. Dataset

COCO, or the Common Objects in Context dataset, is a widely used benchmark in computer vision, offering diverse object categories for various tasks, including image colorization. While training our model, we trained our model on 10,000 images from the COCO dataset, with an additional 2,000 images used for validation.

For the purposes of training, the code converts the images into grayscale, and together with the parent image, feeds it to the GAN model. This way, any large image dataset is compatible with in this code, and it does not need any special datasets.

4.3. Model Architecture

The Neural Network Architecture used by the model can be found in Figure 13.

5. Results

Figure 14. Baseline UNet Colourized Images; (top) Greyscaled; (center) Colourized; (bottom) Ground Truth

Unlike the baseline, this model favours bright colours while still having a more natural colour palette. However, it must be noted that the model can be trained further, with more images and for longer duration to allow for even better results. The current model, while performing very well, is still under-fitted compared to its full potential, as VGG19 can be trained to extract more accurate data than what is currently expressed in the outputs.

Figure 15. Enhanced UNet Colourized Images; (left) Ground Truth; (center) Greyscaled; (right) Colourized

References

[1] Abdullah Althbaity, Mohamed M. Dessouky, and Ishtiaq Rasool Khan. Colorization of grayscale images using deep learning. In *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 131–138, 2022. 2

[2] Dr. Pawan Kumar Angelica D Pyngrope. Colorization of grayscale images in deep learning. In *International Journal of Engineering Applied Sciences and Technology, 2022*, pages 203–212, 2022. 1, 2

[3] Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Guided image generation with

conditional invertible neural networks, 2019. 2

[4] Federico Baldassarre, Diego González Morín, and Lucas Rodés-Guirao. Deep koalarization: Image colorization using cnns and inception-resnet-v2, 2017. 2

[5] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 415–423, 2015. 2

[6] Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, Min Jin Chong, and David Forsyth. Learning diverse image colorization, 2017. 2

[7] Jeff Hwang and You Zhou. Image colorization with deep convolutional neural networks. In *Stanford University Deep Learning for Computer Vision Reports, 2016*, number 219, 2016. 2

[8] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Trans. Graph.*, 35(4), 2016. 2

[9] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks, 2017. 6

[10] Manoj Kumar, Dirk Weissenborn, and Nal Kalchbrenner. Colorization transformer, 2021. 2

[11] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization, 2017. 2

[12] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23, 2004. 2

[13] Chen Liang, Yunchen Sheng, and Yichen Mo. Grayscale image colorization with gan and cyclegan in different image domain, 2024. 2

[14] Vijaya Raghavan, S Rupa Sree, Kaladevi R, S Hariharan, and Bhanuprasad A. Black and white image colorization using deep learning. In *2022 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, pages 27–30, 2022. 2

[15] A Venugopal Rao, Santosh Kumar Vishwakarma, and Shakti Kundu. Artificial intelligent approach for colorful image colorization using a dcnn. In *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 54–58, 2022. 2

[16] Jheng-Wei Su, Hung-Kuo Chu, and Jia-Bin Huang. Instance-aware image colorization. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7965–7974, 2020. 1, 2

[17] Hao Wang and Xuedong Liu. Overview of image colorization and its applications. In *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 1561–1565, 2021. 1, 2

[18] Min Xu and YouDong Ding. Fully automatic image colorization based on semantic segmentation technology. *PLOS ONE*, 16(11):e0259953, 2021. 2

[19] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization, 2016. 2