Automated Image Colorization using Generative Adversarial Networks

Jaisal Sharma
Institute of Engineering Pulchowk Campus, Tribhuvan University
Lalitpur, Nepal
076bei015.jaisal@pcampus.edu.np

Rahul Shrestha
Institute of Engineering Pulchowk Campus, Tribhuvan University
Lalitpur, Nepal
076bei028.rahul@pcampus.edu.np

Dibakar Raj Pant*
Institute of Engineering Pulchowk Campus, Tribhuvan University
Lalitpur, Nepal
drpant@ioe.edu.np

Jukka-Pekka Skon
South-Eastern Finland University of Applied Sciences
Mikkeli, Finland
Jukka-Pekka.Skon@xamk.fi

Jukka Heikkonen
University of Turku
Turku, Finland
jukhei@utu.fi

Rajeev Kanth
Savonia University of Applied Sciences
Kuopio, Finland
rajeev.kanth@savonia.fi

Abstract

Automated colorization of grayscale images is one of the fundamental challenges in the Computer Vision(CV) domain. Self-supervised learning methods are an effective approach to learn general visual features automatically, without having to manually annotate image datasets. Generative Adversarial Networks (GAN) are applied for the colorization method as they are capable of learning visual characteristics from any image without the need for annotated data. They possess a loss function to train the mapping and have ability to learn the mapping from input to output image. The effectiveness of different color spaces (LAB, YUV, and HSI) for image colorization using GAN is investigated. The evaluation of colorization quality is carried out by calculating the pixel accuracy using Peak Signal-to-Noise Ratio (PSNR), and assessing structural integrity using Structural Similarity Indexing Method (SSIM). Using such methods, a comparative approach is demonstrated to study colorization effectiveness of the old black and white images. The SSIM and PSNR values obtained for different color space models are LAB (0.947 and 32.69 dB), YUV (0.842 and 27.65 dB) and HSI (0.851 and 28.72 dB) respectively. Among the three models, LAB color space performed the best with the SSIM and PSNR value of 0.947 and 32.69 dB.

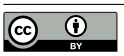
CCS Concepts

• **Computing methodologies** → Artificial intelligence; Computer vision; Computer vision representations; Image representations; Computer graphics; Image manipulation; Image processing.

Keywords

Generative Adversarial Network, Color Space, Loss Function, Self-Supervised Learning, Peak Signal-to-Noise Ratio, Structural Similarity Indexing Method

*Corresponding Author



ICACS 2025, Bangkok, Thailand

ACM ISBN 979-8-4007-1961-5/2025/12
https://doi.org/10.1145/3789418.3789424

ACM Reference Format:
Jaisal Sharma, Rahul Shrestha, Dibakar Raj Pant, Jukka-Pekka Skon, Jukka Heikkonen, and Rajeev Kanth. 2025. Automated Image Colorization using Generative Adversarial Networks. In *The 9th International Conference on Algorithms, Computing and Systems (ICACS 2025), December 12–14, 2025, Bangkok, Thailand*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3789418.3789424

1 INTRODUCTION

Image colorization is a technique in which every pixel of grayscale images is mapped with RGB color values [1]. The purpose of colorization is not to precisely rebuild the color but to trick the observer into thinking that the colorized image is real. The image colorization is suitable for restoring ancient grayscale pictures, movie restoration and many other applications in recreating an appealing visual appearance. The colorized version of grayscale images significantly changes the viewer's viewpoint of the image. However, no unique color can be associated with objects such as shoes, clothes or other household items. Therefore, the colorization outcome cannot be verified with any specific color solution [2].

In this paper, a comprehensive overview and the effectiveness is provided for different color spaces (the LAB [3], the YUV [4] and the HSI [5]) with the help of GANs to achieve convincing colorization results. The aim is to identify the most efficient color space for generating convincing colorization through a comparative analysis using objective evaluations such as the PSNR (Peak Signal-to-Noise Ratio) [6] and the SSIM (Structural Similarity Indexing Method) [7]. Other metrics like Mean Square Error (MSE) is redundant, as PSNR is derived from it and Feature Similarity Indexing Method (FSIM) is less critical because the focus is on grayscale to color conversion and SSIM can capture that structural preservation effectively. Many research works have been conducted using CNN [8] and Variational Auto Encoders (VAE) [9] in this direction, but a lot of manual work is required to design a loss function in such models. However, Generative Adversarial Networks (GANs) [10] train both the generator and discriminator models concurrently, where the generator learns to produce more realistic colorized images, while the discriminator learns to distinguish between real and fake colorization accurately [11]. This study has examined the feasibility of leveraging GANs for

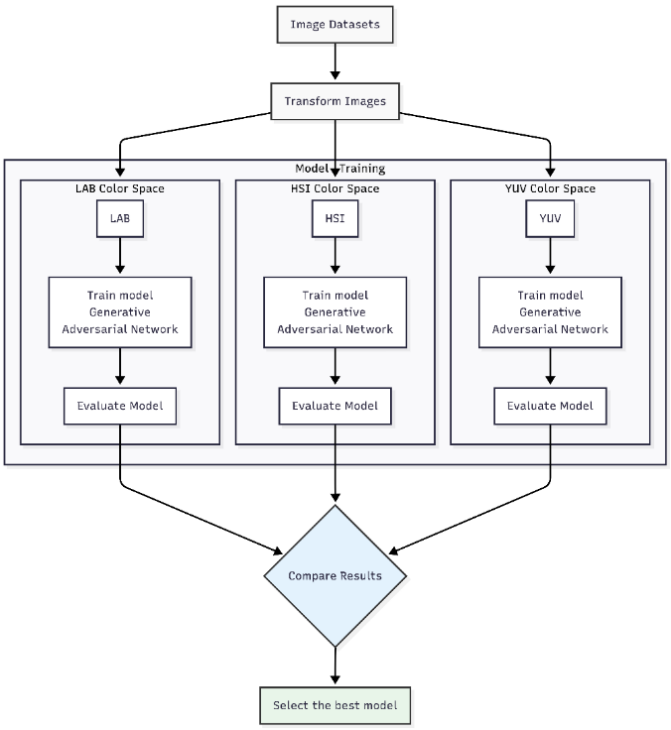


Figure 1: System Block Diagram for Image Colorization

image colorization, comparing their performance in the different color spaces (LAB, YUV, and HSI).

2 PREVIOUS WORKS

The concept of Conditional GANs (cGANs) was introduced in the work "Colorful Image Colorization" [12]. They proposed a framework in which the generator accepts a grayscale picture and a color hint as inputs to produce a colored image. The discriminator was modified to assess the realism of the colored images in conjunction with the grayscale inputs. This approach demonstrated promising results in producing plausible and vibrant colorizations. Building upon the success of cGANs, a novel approach [13] was introduced in which their model employed a deep neural network to predict color distributions in both global and local contexts, enabling the generation of coherent colorizations. By integrating image classification as an auxiliary task, they further improved the quality of colorization outputs.

3 METHODOLOGY

In this section, the framework employed to implement image colorization is described. The chosen approach involves utilizing a U-Net architecture [14] as the generator and a PatchGAN discriminator. A high-level overview of the work is shown in Figure 1 below:

3.1 Datasets

The Microsoft COCO (Common Objects in Context) dataset [15] is a widely used dataset for object detection, segmentation, and other computer vision tasks. The dataset is open source and consists of different images with variation in environment and subject. It is used for training purposes only to extract its features for colorization.

3.2 Image Transformation

Here, several transformations are applied for the image dataset. The PyTorch library torchvision's transforms. Resize function [16] is used to resize the images to a specific size (SIZE x SIZE) using bicubic interpolation where the size is 256 for both height and width. Bicubic interpolation is a method of interpolating values in a grid based on surrounding data points [17]. It ensures smoother rescaling and is com monly used for image resizing. The Bicubic interpolation is defined as below:

$$W(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & for \ \ |x| \le 0, \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & for \ \ 1 \le |x| \le 2, \\ 0 & otherwise, \end{cases} \tag{1}$$

3.3 Color Space Transformation

After obtaining the transformed datasets, the RGB values are converted into different color spaces(LAB, YUV, and HSI). Some of the image information might get lost when the color is transformed.

3.4 LAB Color Space

In this step, the parameter required for conversion to the LAB color space is calculated using the following equation:

$$L^* = 116 f\left(\frac{Y}{Y_n}\right) - 16 \tag{2}$$

$$a^* = 500\left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right) \tag{3}$$

$$b^* = 200\left(f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right) \tag{4}$$

Where:

- L^*: Lightness component (0-100), where 0 is black and 100 is white.
- a^*: Green-red opponent channel, with positive values representing red and negative values representing green.
- b^* : Blue-yellow opponent channel, with positive values representing yellow and negative values representing blue.
- X_n, Y_n, Z_n: The CIEXYZ tristimulus values for the illuminant.
- $f(x)$: Non-linear function approximating the human visual system's response to light intensity.

3.5 YUV Color Space

Likewise, the parameters for YUV color space are also determined using the following equation:

$$Y = 0.299R + 0.587G + 0.114B \tag{5}$$

$$U = 0.492(B - Y) \tag{6}$$

$$V = 0.877(R - Y) \tag{7}$$

Where:

- R, G, *and* B are the red, green, and blue color channel values respectively.
- Y represents the luma (luminance) component.

- U represents the chroma component indicating the blue-yellow color difference.
- V represents the chroma component indicating the red-green color difference.

3.6 HSI Color Space

For this conversion, the values of hue (H), saturation (S) and intensity/brightness (I) are determined using the following equations:

$$H(x) = \begin{cases} 360^\circ - \theta \ \ if \ B > G \\ \theta \ \ if \ B \leq G \end{cases} \tag{8}$$

where (θ) is the angle in degrees given by:

$$\theta = \cos^{-1}\left(\frac{0.5 \times (2R - G - B)}{\sqrt{(R-G)^2 + (R-B)(G-B)}}\right) \tag{9}$$

The saturation component (S) is calculated by:

$$S = 1 - \frac{3}{R+G+B} \times \min(R, G, B) \tag{10}$$

Similarly, the intensity component (I) is given by:

$$I = \frac{R+G+B}{3} \tag{11}$$

3.7 Generator

The U-net generator, a powerful architecture suitable for semantic segmentation tasks is used. A distinct class label is assigned to each pixel in the input using semantic segmentation. This helps in object identification and precisely delineate their boundaries. This architecture is comprised up of an encoder and decoder. The encoder network in the U-Net generator is responsible for capturing the spatial features of the input image at multiple scales. The components found in the encoder are as follows:

- Convolutional layers: Their role is to perform convolutional operations to extract features from input at various scales. The number of filters and the size of the filters can vary depending on the specific implementation.
- Activation functions: Rectified Linear Unit (ReLU) or variants like Leaky ReLU are often used for adding non-linearity to help model capture complex patterns.
- Pooling or stride-convolution: Max-pooling or stride-convolution operations are commonly applied to downsample the feature maps, capturing increasingly abstract representations as the network goes deeper.

Similarly, the decoder network in the U-Net generator performs the upsampling and reconstruction of the feature maps obtained from the encoder. Following are the steps which happens in the decoder:

- Up-sampling layers: Transposed convolutions or nearest-neighbor interpolation operations are used to upscale the feature maps.
- Skip connections: These connections concatenate feature maps from the corresponding scales in the encoder to the decoder layers.
- Convolutional layers and Activation Functions: The decoder also process the concatenated feature maps and capture the necessary details for generating the final output and introducing non-linearity.

The generator workflow can be observed in Figure 2.

The U-Net model consists of multiple layers. Input image is passed through a series of down-convolutional layers, which downscales it and simultaneously increase the channel count. A 1x1 image is formed at the transition between encoder and decoder model. The down-convolutional layers are followed by up-convolutional layers, which gradually upsample the feature maps, restore the spatial dimensions of the image and produces a 256x256 image having same size as the input. Each pixel in the output image will be assigned a label or class that represents the segmented region or object to which it belongs. The model's training process aims to optimize the parameters to accurately segment the input image based on the given classes or labels and both local and global image features are effectively captured.

3.8 Discriminator

In a PatchGAN discriminator, the input image is split into non-overlapping patches, and each patch is processed independently to determine its authenticity. The output of the discriminator is a grid of prediction values, where each prediction corresponds to a local patch. This grid of predictions provides spatial information about the generated and real images, enabling the generator to focus on producing realistic and coherent local structures. Inside the discriminator, the following tasks are done:

3.8.1 Convolution. The convolutional operation applies a small filter over the input data in a sliding window manner. The whole input data undergoes this process, producing an output feature map that captures specific patterns and features.

$$G_{out}(x,y) = \omega_{out} \cdot F(x,y) \left(\sum_{\delta x=-k_i}^{k_i} \sum_{\delta y=-k_j}^{k_j} w_{out}(\delta x, \delta y) . F(x+\delta x, y+\delta y) \right) + \tag{12}$$

Where:

- $G_{out}(x,y)$: Output value at position(x,y) after convolution.
- ω_{out}: Weight parameter for the convolution operation.
- $F(x,\ y)$: Input feature map value at position $(x,\ y)$.
- $\delta x,\ \ \delta y$: Displacement values within the convolution kernel.
- $w_{out}(\delta x, \delta y)$: Weight parameter for the specific displacement $(\delta x, \delta y)$ within the kernel.
- $\omega_{out_{bias}}$: Bias parameter for the convolution operation.

3.8.2 Batch Normalization. Batch normalization works by normalizing the activations of a layer and helps stabilize training by maintaining consistent activation distributions for faster convergence and improved model generalization. The batch normalization operation can be described as follows:

$$\mu_c = \frac{1}{N \cdot H \cdot W} \sum_{i=1}^{N} \sum_{j=1}^{H} \sum_{k=1}^{W} x_{i,c,j,k} \tag{13}$$

$$\sigma_c^2 = \frac{1}{N \cdot H \cdot W} \sum_{i=1}^{N} \sum_{j=1}^{H} \sum_{k=1}^{W} \left(x_{i,c,j,k} - \mu_c\right)^2 \tag{14}$$

$$\hat{x}_{i,c,\ j,k} = \frac{x_{i,c,j,k} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} \tag{15}$$

$$y_{i,c,j,k} = \gamma_c \hat{x}_{i,c,\ j,k} + \beta_c \tag{16}$$

Where:

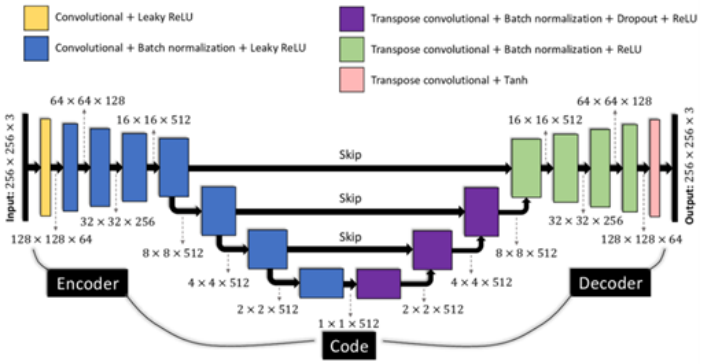


Figure 2: Functional block diagram of generator.

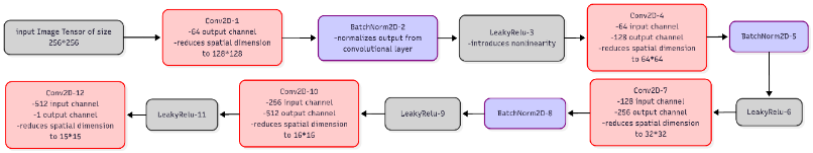


Figure 3: Functional block diagram of the discriminator.

- $x_{i,c,j,k}$is the activation at batch i, channel c, height j, and width k.
- μ_c is the mean of activations for channel c.
- $\sigma_c^2$is the variance of activations for channel c.
- $\hat{x}_{i,c,\ j,k}$ is the normalized activation.
- γ_c is the scale parameter for channel c.
- $\beta_c$is the shift parameter for channel c.
- $\in$ is a small constant added to the denominator for numerical stability.

The discriminator workflow can be observed in Figure 3. The input image tensor is initially processed by a convolutional layer (Conv2d-1) with 64 output channels, producing 128x128 feature maps. Batch normalization (BatchNorm2d-2) is applied to normalize the output, followed by a leaky ReLU activation function (LeakyReLU-3) to introduce non-linearity. The output from the first layer is then passed to the next set of layers. The similar function is performed until it reaches the final layer where the output is passed through the last convolutional layer (Conv2d-12). Finally, Conv2d-12 produces a single-channel output. It takes the output of LeakyReLU-11 and generates a 15x15 output image. This image contains the prediction of discriminator for the given input image. Higher values indicate a higher probability of the input patch being realistic.

3.9 Loss Functions

The two types of loss functions are used: GAN loss and L1 loss. The "gan_mode" parameter specifies the type of GAN loss and it is set to 'vanilla' which is a binary cross-entropy loss that calculates the difference between the prediction of discriminator for real and generated samples. Mathematically, the GAN loss is shown by following equation:

$$\mathcal{L}_{GAN}\left(\mathrm{G},\mathrm{D}\right) = \mathrm{E}_{\mathrm{x,y}}\left[\log \mathrm{D}\left(\mathrm{x},\mathrm{y}\right)\right] + \mathrm{E}_{\mathrm{x,z}}\left[\log\left(1-\mathrm{D}\left(\mathrm{x},\mathrm{G}\left(\mathrm{x},\mathrm{z}\right)\right)\right)\right] \tag{17}$$

Where:

- x is the grayscale image
- z is the input noise for the generator
- y is the 2-channel output
- $G\ and\ D$ are the generator and discriminator model respectively
- $D(x)$and $D(G(z))$: The discriminator's output for the image x and the generated image G(z) respectively

The loss function is combined with L1 Loss (also known as mean absolute error) which calculates the element-wise absolute difference between the predicted and target values and then takes the mean over all elements. It can be understood mathematically as:

$$\mathcal{L}_1\left(G\right) = E_{x,y,z}\left[||y - G\left(x,z\right)||_1\right] \tag{18}$$

Where:

- x is the grayscale image
- z is the input noise for the generator
- y is the 2-channel output
- G is the generator model

By minimizing these losses, the generator learns to generate colorized images that possess realistic color distributions and overall visual quality.

4 RESULTS AND ANALYSIS

The model was trained using the training images from the coco-dataset that were preprocessed and split into training and validation sets of 8000 and 2000 images respectively. The model was trained for 100-200 epochs. During training, the model minimized a loss

Figure 4: Output of LAB color space model.

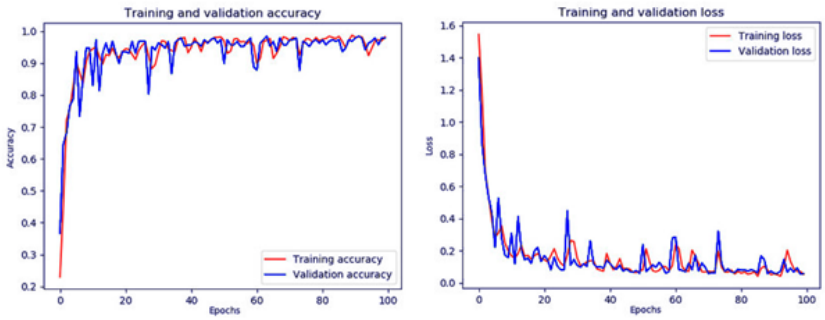


Figure 5: Accuracy and loss of LAB color space model over training epochs.

function that penalized both the overall difference between the generated and real-colored images and potential inaccuracies in specific color spaces. For the analysis, the results and performances are classified into three parts as of the color spaces.

4.1 Colorization using LAB color space

The effectiveness of using the LAB color space for image colorization is explored. Figure 4 shows the output of our training dataset of the LAB color space model program.

The leftmost images are the original images, the one after that is the grayscale version and the rightmost images are the generated colored version.

4.1.1 Performance. This section evaluates the performance of the LAB color space model developed in this work using different metrics. But only the accuracy and error calculation graph is displayed here. Figure 5 shows the accuracy and loss over epochs on both the training and validation sets.

The training accuracy starts higher than the validation accuracy and remains above it throughout the training process. After around 30 epochs, the training and validation curves converge quickly because the lab separates the luminance and chrominance effectively, that allows the model to generalize well. This demonstrates that the model is effectively colorizing the images.

4.2 Colorization using HSI Color Space

Similarly, the HSI color space is used to train the datasets and utilized in the model for image colorization.

The leftmost images are original, middle ones are grayscale and rightmost are colorized versions as shown in the output (Figure 6). On visual comparison, the output seems to be slightly faded even though this model was trained for higher epochs than the prior one.

4.2.1 Performance. A similar performance test was conducted for this color space. Figure 7 shows the accuracy and loss over epochs on both the training and validation sets.

The training accuracy remains above the validation accuracy with significant difference. After around 50 epoch, the model's performance improves significantly because the adversarial training becomes stable and the generator starts to learn meaningful luminance-chrominance relation. However, the difference between training and validation never closed out to a converging point even after training for higher epochs.

4.3 Colorization using YUV color space

Finally, the YUV color space is utilized for image colorization. Figure 8 shows the output of our training dataset of the YUV color space model program.

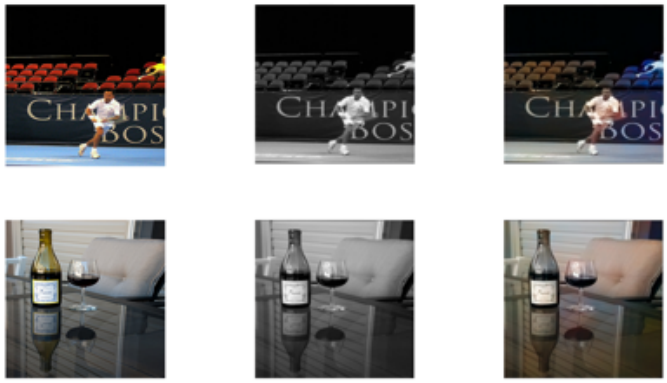


Figure 6: Output of HSI color space model.

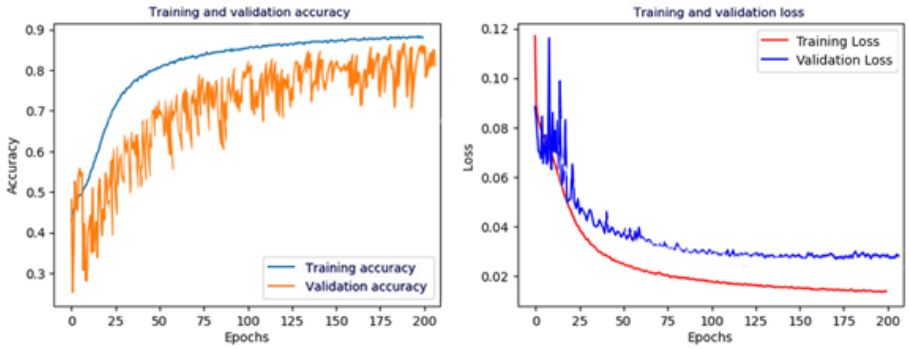


Figure 7: Accuracy and loss of HSI color space model over training epochs.

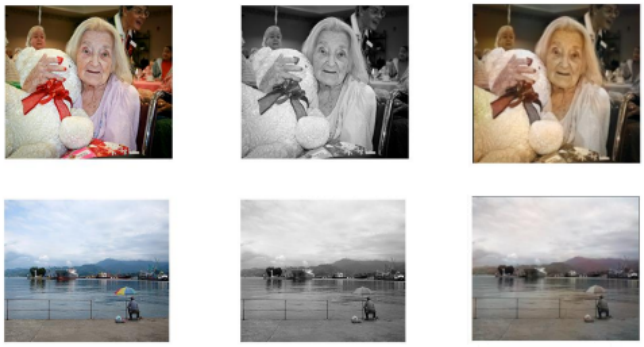


Figure 8: Output of YUV color space model.

Images are in order of original, grayscale and colorized version from left to right.

4.3.1 Performance. The performance of the YUV color space program developed in our work is also measured using different metrics. Figure 9 shows the accuracy and loss over epochs on both the training and validation sets.

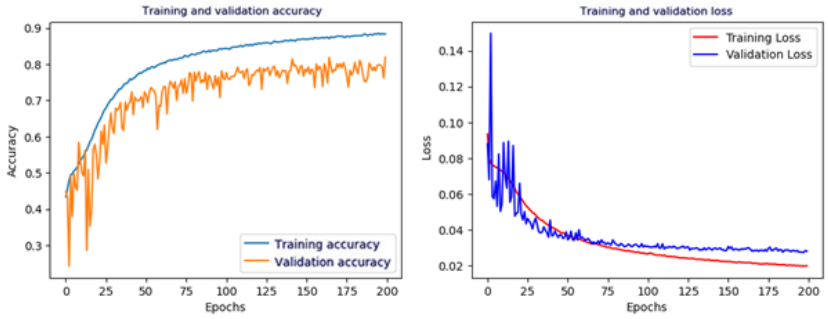


Figure 9: Accuracy and loss of YUV color space model over training epochs.

Table 1: Evaluation of Color Models Using PSNR and SSIM

Color Models	PSNR(dB)	SSIM
LAB	32.69	0.947
YUV	27.65	0.842
HSI	28.72	0.851

Similar results for the YUV model are observed compared to the HSI model. After around 50 epochs, the performance improves notably because the generator captures and learns the color pattern effectively, which is a result of successful adversarial training. The validation accuracy remains stagnant around 0.8 from roughly 100-200 epochs of training, indicating the overfitting case. Similar trend can be seen in minimizing the loss by the model.
In the training-validation plots for YUV and HSI, a significant gap can be observed. However, in the LAB model plots, a minimal gap is observed. Despite the small overfitting in YUV and HSI, the ranking of the color spaces is not affected. LAB model still performs best which may be supported by its perceptual uniformity that separates the luminance and chrominance more effectively. This helps the model generalize slightly better than YUV or HSI.

5 EVALUATION

A comparative analysis of all three color models: LAB, YUV, and HSI, is presented in Table 1. using two metrics, PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index).

Higher PSNR value suggests better preservation of processed image while SSIM value closer to 1 indicate stronger resemblance. LAB achieves the highest PSNR (32.69 dB), suggesting potential superiority in retaining overall image quality. YUV follows with a PSNR of 27.65 dB, while HSI falls in the middle at 28.72 db. LAB again dominates with the highest SSIM (0.947). YUV demonstrates a lower SSIM of 0.842, while HSI score is 0.851, both indicating some structural differences compared to LAB. Only Microsoft COCO dataset was used to train and evaluate all three color-space models for ensuring a consistent comparison framework. While this provides a fair benchmark, future work can extend this comparison to a cross-dataset validation that would present a robust generalization of our findings.

6 CONCLUSION

This study explores the colorization and restoration of historical images using deep learning technique, specifically focusing on the effectiveness of different color spaces (LAB, YUV, and HSI). While subjective evaluation remains valuable for artistic preferences, the limitations of time, cost, and potential bias necessitate quantitative metrics. Based on the results, the LAB color space model achieved the best performance, exhibiting the highest PSNR (32.69 dB) and SSIM (0.947), indicating good overall image quality and structural preservation. However, the choice of optimal color space depends on the specific application and desired balance between factors like fidelity and structural preservation. Image colorization is best evaluated using perceptual quality beyond numerical metrics. The output results of LAB align more closely with human vision as it produces more natural-looking results. YUV and HSI have produced slightly faded outputs. Therefore, human subjective testing and perceptual evaluation could be included in future work that will further strengthen the comparison study of different models. Future work could focus on the incorporation of perceptual quality metrics more closely aligned with human perception. Additionally, investigating the advanced GAN based architectures or hybrid regularizations for improved colorization is an interesting potential avenue to explore. Overall, this work demonstrates the study and feasibility of leveraging deep learning in various color spaces for effective image colorization, establishing a direction for further advancements in this evolving field.

Acknowledgments

We would like to thank the University of Turku for the financial support for conference participation.

References

[1] Shuang Huang, Xiaofeng Jin, Qian Jiang, and Lin Liu. 2022. Deep learning for image colorization: current and future prospects. Engineering Applications of Artificial Intelligence 114 (2022), Article 105006. https://doi.org/10.1016/j.engappai.2022.105006
[2] Ivana Žeger, Sonja Grgic, Josip Vuković, and Gordan Šišul. 2021. Grayscale image colorization methods: Overview and evaluation. IEEE Access 9 (2021), 113326–113346. https://doi.org/10.1109/ACCESS.2021.3104515
[3] Wojciech Mokrzycki and Maciej Tatol. 2009. Perceptual difference in Lab* color space as the base for object colour identification. https://doi.org/10.13140/2.1.1160.2241
[4] Michal Podpora, Grzegorz Korba´s, and Aleksandra Kawala-Sterniuk. 2014. YUV vs RGB – Choosing a Color Space for Human-Machine Interaction. Annals of Computer Science and Information Systems 3 (2014).

https://doi.org/10.15439/2014F206

[5] Kazuya Yoshinari, Kota Murahira, Yoshikatsu Hoshi, and Akira Taguchi. 2013. Color image enhancement in improved HSI color space. In 2013 International Symposium on Intelligent Signal Processing and Communication Systems, 429–434. https://doi.org/10.1109/ISPACS.2013.6704588

[6] Prateek Gupta, Priyanka Srivastava, Satyam Bhardwaj, and Vikrant Bhateja. 2011. A modified PSNR metric based on HVS for quality assessment of color images. In 2011 International Conference on Communication and Industrial Application, 1–4. https://doi.org/10.1109/ICCIndA.2011.6146669

[7] Mohammed Hassan and Chakravarthy Bhagvati. 2012. Structural similarity measure for color images. International Journal of Computer Applications 43, 14 (April 2012), 7–12. https://doi.org/10.5120/6169-8590

[8] Leila Kiani, Mohammad Saeed, and Hamid Nezamabadi-Pour. 2020. Image colorization using a deep transfer learning. In Proceedings of the 2020 8th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), Mashhad, Iran, September 2–4, 2020. https://doi.org/10.1109/CFIS49607.2020.9238737

[9] Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, Min Jin Chong, and David Forsyth. 2017. Learning diverse image colorization. arXiv preprint arXiv:1612.01958. https://arxiv.org/abs/1612.01958

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. Communications of the ACM 63, 11 (2020), 139–144.

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 5967–5976. https://doi.org/10.1109/CVPR.2017.632

[12] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2018. Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004. https://arxiv.org/abs/1611.07004

[13] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2016. Let there be color! Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. ACM Transactions on Graphics 35, 4 (July 2016), Article 110, 11 pages. https://doi.org/10.1145/2897824.2925974

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional networks for biomedical image segmentation. arXiv preprint arXiv:1505.04597. https://arxiv.org/abs/1505.04597

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. 2015. Microsoft COCO: Common objects in context. arXiv preprint arXiv:1405.0312. https://arxiv.org/abs/1405.0312

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703. https://arxiv.org/abs/1912.01703

[17] Vaishali Patel and Kinjal Mistree. 2013. A review on different image interpolation techniques for image enhancement. International Journal of Emerging Technology and Advanced Engineering 3, 12 (2013), 129.