Colorizer: A CNN AE for Colorization of Grayscale Images

Davide Vitagliano - 904120@stud.unive.it
Image Video Understanding
Academic year 2024/2025

1. Introduction

Like to the github repository for the code: https://github.com/Vitolus/IVU_Colorizer
Colorization of greyscale images involves adding plausible colors to images that are originally in shades of grey, making them appear more realistic and visually appealing. This task is particularly relevant in various fields such as photography, film restoration, and digital art, where the goal is to enhance the visual quality of images. The main challenges in colorization include determining the appropriate colors for different objects and scenes, maintaining consistency across the image, and ensuring that the added colors do not distort the original content. Moreover, due to the subjective nature of color perception, the process often requires a balance between artistic interpretation and adherence to realistic color schemes.

2. Literature Review

Earlier works on image colorization relied on scribble-based methods: these methods required users to provide color scribbles on the grayscale image, which were then used to guide the colorization process. However, this approach was limited by the need for user input and could be time-consuming. Another method was color transfer [2]: this technique involved transferring colors from a reference image to the grayscale image, but it often resulted in unrealistic colors and lacked the ability to generate new colors.

Recent advancements in deep learning have led to the development of more sophisticated methods for image colorization, such as Deep Convolutional Generative Adversarial Networks (DCGAN) [6]: these method utilize a generator network to produce colorized images and a discriminator network to evaluate the quality of the generated images. Another deep neural network approach involves Autoencoders and Variational Autoencoders (VAE) [7] [9]: these methods learn to encode the grayscale image into a latent space and then decode it back into a colorized image. These deep learning-based methods have shown promising results in generating realistic and visually appealing colorized images, but they still face challenges such as maintaining color consistency and handling complex scenes.

3. Model description

The implemented model is a Convolutional Variational Autoencoder based on the vanilla AE model presented in [1].

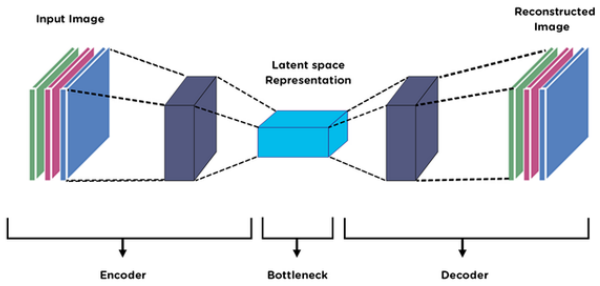

Figure 1. General architecture of the VAE

Specifically, the encoder applies four stages 1: an initial block, mapping the input grayscale image to 64 channels at half resolution; followed by blocks that expand to 128 and 256 channels at quarter and one eighth resolution, respectively. A final block processes the feature maps at one eighth resolution with 512 intermediate channels, outputting a 256-channel representation. Global average pooling reduces the spatial dimension to a 1×1 tensor, which is linearly projected into parameters of the latent distribution: mean and log variance. Sampling from this distribution is performed via the reparameterization trick. Moreover, inside the deeper blocks of the encoder and after the pooling layer for the latent representation computation, is inserted a dropout layer to avoid learning biases of the training data.

The decoder mirrors the encoder with transposed convolutional blocks, progressively increasing spatial resolution while decreasing channel dimensionality. The final decoder block outputs a 2-channel image at the original input resolution, representing the predicted ab color channels in the Lab color space. A Tanh activation is applied to constrain the output values to the valid range of [-1, 1].

Layer	Output Shape
Input	$[B, 1, SIZE, SIZE]$
Conv2d(1, 64, 3, 2, 1)	$[B, 64, SIZE/2, SIZE/2]$
LeakyReLU	$[B, 64, SIZE/2, SIZE/2]$
Conv2d(64, 128, 3, 1, 1)	$[B, 128, SIZE/2, SIZE/2]$
LeakyReLU	$[B, 128, SIZE/2, SIZE/2]$
Dropout2d(0.2)	$[B, 128, SIZE/2, SIZE/2]$
Conv2d(128, 128, 3, 2, 1)	$[B, 128, SIZE/4, SIZE/4]$
LeakyReLU	$[B, 128, SIZE/4, SIZE/4]$
Conv2d(128, 256, 3, 1, 1)	$[B, 256, SIZE/4, SIZE/4]$
LeakyReLU	$[B, 256, SIZE/4, SIZE/4]$
Dropout2d(0.2)	$[B, 256, SIZE/4, SIZE/4]$
Conv2d(256, 256, 3, 2, 1)	$[B, 256, SIZE/8, SIZE/8]$
LeakyReLU	$[B, 256, SIZE/8, SIZE/8]$
Conv2d(256, 512, 3, 1, 1)	$[B, 512, SIZE/8, SIZE/8]$
LeakyReLU	$[B, 512, SIZE/8, SIZE/8]$
Conv2d(512, 512, 3, 1, 1)	$[B, 512, SIZE/8, SIZE/8]$
LeakyReLU	$[B, 512, SIZE/8, SIZE/8]$
Dropout2d(0.2)	$[B, 512, SIZE/8, SIZE/8]$
Conv2d(512, 256, 3, 1, 1)	$[B, 256, SIZE/8, SIZE/8]$
LeakyReLU	$[B, 256, SIZE/8, SIZE/8]$
AdaptiveAvgPool2d(1)	$[B, 256, 1, 1]$
Dropout(0.2)	$[B, 256, 1, 1]$
Conv2d(256, $2 \cdot latent\_dim$, 1)	$[B, 2 \cdot latent\_dim]$
Linear($latent\_dim$, $256 \cdot (SIZE/8)^2$)	$[B, 256, SIZE/8, SIZE/8]$
ConvT(256, 128, 3, 2, 1, 1)	$[B, 128, SIZE/4, SIZE/4]$
LeakyReLU	$[B, 128, SIZE/4, SIZE/4]$
ConvT(128, 64, 3, 2, 1, 1)	$[B, 64, SIZE/2, SIZE/2]$
LeakyReLU	$[B, 64, SIZE/2, SIZE/2]$
ConvT(64, 32, 3, 1, 1)	$[B, 32, SIZE/2, SIZE/2]$
LeakyReLU	$[B, 32, SIZE/2, SIZE/2]$
ConvT(32, 16, 3, 1, 1)	$[B, 16, SIZE/2, SIZE/2]$
LeakyReLU	$[B, 16, SIZE/2, SIZE/2]$
ConvT(16, 2, 3, 2, 1, 1)	$[B, 2, SIZE, SIZE]$
Tanh	$[B, 2, SIZE, SIZE]$

Table 1. Architecture of proposed VAE

As mentioned, the model is trained using images in the CIELAB color space 2. The L channel (luminance) is used as input, which act as the grayscale image; while the ab channels (chrominance) are the target output for training. Standard Lab space has L channel in range [0, 100] and the ab channels in range [-127, 128]. The input to the network is rescaled to range [0, 1], while the network outputs are rescaled to [-1, 1] via the Tanh activation. Because of this fact, the predicted ab channels are rescaled to range [-128, 127] before being used in loss computation.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \leftarrow \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$X \leftarrow X/X_n, \text{where} X_n = 0.950456$$

$$Z \leftarrow Z/Z_n, \text{where} Z_n = 1.088754$$

$$L \leftarrow \begin{cases} 116 * Y^{1/3} - 16 & \text{for } Y > 0.008856 \\ 903.3 * Y & \text{for } Y \leq 0.008856 \end{cases}$$

$$a \leftarrow 500(f(X) - f(Y)) + delta$$

$$b \leftarrow 200(f(Y) - f(Z)) + delta$$

$$f(t) = \begin{cases} t^{1/3} & \text{for } t > 0.008856 \\ 7.787t + 16/116 & \text{for } t \leq 0.008856 \end{cases}$$

$$delta = \begin{cases} 128 & \text{for 8-bit images} \\ 0 & \text{for floating-point images} \end{cases}$$

Figure 2. RGB to XYZ to LAB conversion process

4. Experimental settings

4.1. Datasets

The dataset used for training and testing the model is the Landscape Color Image dataset[1] present on Kaggle, which is suitable for building an autoencoder network for colorization tasks. The dataset contains more than 7128 grayscale images of landscapes, each with a corresponding color image. Considering that the images are converted to CIELAB color space, the grayscale images are not used and the input for the neural network is extracted from the L channel of the colored image and the ab channels are used as the ground truth output. The images are resized to 224x224 pixels to ensure uniformity with the VGG19 network used for perceptual loss calculation. The dataset is split into training, validation and test sets with a ratio of 80% for training, and the remaining 20% is split again at a ratio of 80% for validation and 20% for testing. Follows two examples of images from the trainset in figures 3 and 4.

[1] https://www.kaggle.com/theblackmamba31/landscape-image-colorization

Figure 3. Example of image

Figure 4. Example of image

4.2. Competitors

Zhang et al. (2016) [10] proposed one of the most influential approaches, formulating colorization as a classification problem over quantized color bins and achieving strong perceptual realism. Exemplar-based methods, such as the model by He et al. (2018) [4], demonstrated high-quality results by transferring chrominance information from reference images, though they require carefully chosen exemplars to perform well. Generative adversarial networks have also been widely adopted; Nazeri et al. (2018) [6] showed that adversarial training leads to more vibrant and realistic colorizations, albeit with increased training instability. More recently, Xin et al. (2022) [9] highlighted the importance of skip connections within autoencoder architectures, demonstrating that multi-scale feature fusion significantly improves reconstruction fidelity.

The proposed model is compared against the baseline model by Xin et al. (2022) [9], concentrating on the PSNR and RMSE metrics to evaluate the reconstruction quality of the colorized images.

4.3. Evaluation criteria

The colorization is treated as a regression task, and the model is trained to minimize a weighted composition of loss functions:

- **Pixel-wise Loss:** Charbonnier loss between the predicted and ground truth ab channels. This loss is a smooth approximation of the L1 loss, defined as:

$$L_{char}(x, y) = \sqrt{(x - y)^2 + \epsilon^2}$$

  where ϵ is a small constant to ensure numerical stability. Thus, it acts as MSE loss when the error is small, and L1 loss when the error is large, making it less sensitive to outliers.

- **Cosine Similarity Loss:** Measures the cosine similarity between the predicted and ground truth ab channels to encourage angular alignment in the color space, it is defined as:

$$L_{cos}(x, y) = 1 - \frac{x \cdot y}{\|x\|\|y\|}$$

  where the denominator is clamped to a small value to avoid division by zero.

- **Perceptual Loss:** Computed using a pre-trained VGG19 network [8], comparing high-level feature representations of the predicted and ground truth images, via L1 loss, which is defined as:

$$L_{perc}(x, y) = \|\phi(x) - \phi(y)\|_1$$

  where ϕ represents the feature extraction function of the VGG19 network at selected layers.

- **Kullback-Leibler Divergence Loss:** Used to regularize the latent space in the variational autoencoder framework, ensuring that the learned latent distribution approximates a prior distribution (typically a standard normal distribution). It is defined as:

$$L_{KL}(q(z|x)||p(z)) = -\frac{1}{2}\sum_{i=1}^{d}(1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2)$$

  where $q(z|x)$ is the learned latent distribution, $p(z)$ is the prior distribution, and μ and σ are the mean and standard deviation of the latent variables.

The layer of the VGG19 network used for perceptual loss computation is the ReLU activation corresponding to relu3_4 and relu4_4. After hyperparameter tuning, the final features map used is relu3_4, as it provided a good balance between in capturing useful features, while maintaining spatial details.

One problem of this reconstruction loss is that the cosine similarity loss and the Charbonnier loss could conflict with each other: the first one pushes direction of vectors towards allignment, ignoring magnitude; while the second one pushes pixelwise differences toward zero. This could lead to opposing gradients in initial epochs, when the output of the network is still far from the target. An attempt to mitigate this effect is to use a lower coefficient for the cosine similarity loss.

Other metrics used to evaluate the performance of the model are Mean Root Squared Error (MRSE) and the Peak Signal-to-Noise Ratio (PSNR): the former measures the average magnitude of the errors between predicted and ground

truth values;

$$L_{MRSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

while the latter measures the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation

$$L_{PSNR} = 10\log_{10}\left(\frac{255^2}{MSE + \epsilon}\right)$$

where ϵ is a small constant to avoid division by zero and MSE is the mean squared error. Higher PSNR values indicate better quality of the reconstructed image.

5. The results

5.1. Performance achieved on the dataset

The best configuration of hyperparameters found during cross validation tuning is used to train the VAE on the entire training set, and the performance is evaluated on a validation set not used in cross validation. A final test is performed on the test set. Even though the dataset is small, the model has the potential to reach greater results if it is trained for more epochs. The model hardly reaches the PSNR threshold of 25, plateauing around 24.5 after 30 epochs. This could be due to the small size of the dataset, which limits the ability of the model to generalize. To address this issue, a scheduler is implemented to reduce the learning rate by a factor of 0.05 after a certain number of epochs without improvement in the PSNR metric, to allow the model to converge to a better minimum. The training and validation loss, RMSE and PSNR are shown in the following figures 5, 6 and 7. The final value of the metrics are reported in 2.

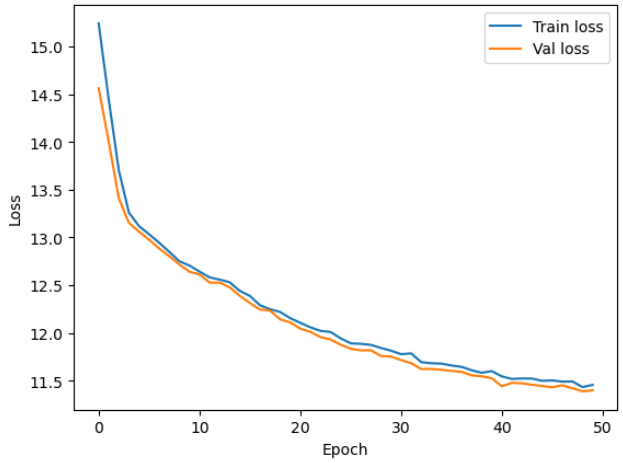

Figure 5. Training and validation loss

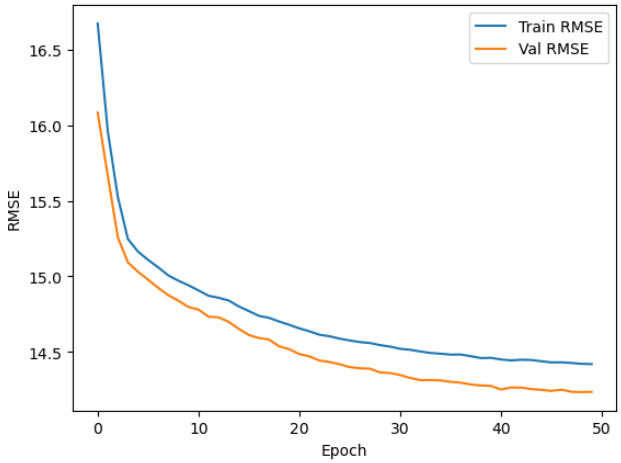

Figure 6. Training and validation RMSE

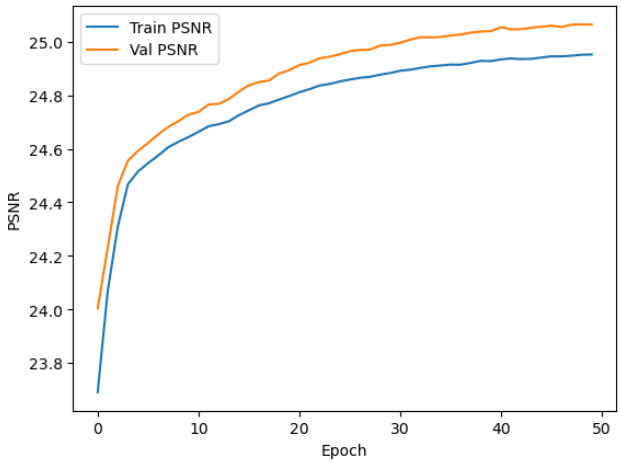

Figure 7. Training and validation PSNR

5.2. Additional analysis: Sensitivity to hyperparameters

Before training the VAE with the entire dataset, hyperparameters tuning is performed on the trainset, which is used in K Fold cross validation. The hyperparameters considered are the learning rate, the batch size, the latent space dimension, the coefficients that multiply each loss component, and which layers of VGG19 are used to extract feature maps for the perceptual loss. The best configuration is selected based on the average performance of the validation loss of the k folds across n trials. The best configuration is then used to train the VAE on the entire training set, and the performance is evaluated on a validation set not used in cross validation.

It is important to notice that the best coefficients for the perceptual loss is far lower than anticipated, and as expected lower contribution of the cosine similarity loss don't destabilize the training process. Another note is that using a

learning rate higher of the one found by hyperparameter tuning, leads to faster convergence and slightly higher performance, with lower initial loss. However, this also leads to oscillations in the training and validation loss after a certain number of epochs, which is not ideal.

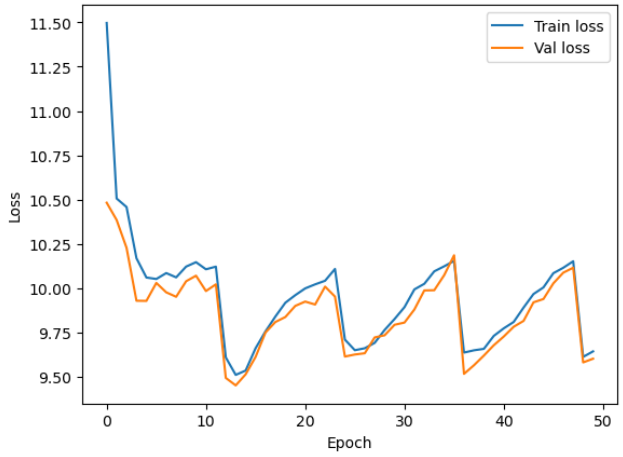

Figure 8. Training and validation Loss with higher learning rate

6. Discussion

6.1. Findings

A Variational Autoencoder is able to learn the task by starting from a grayscale images, specifically the latent space learned by the encoder suffices to provide enough information to the decoder to reconstruct the colorized image in a meaningful way. The use of cosine similarity loss can be potentially detrimental, destabilizing the training process even if its coefficient is small. Comparing to the baseline model 2, the proposed one achieves worse results in terms of PSNR and RMSE, probably due to the fact that the model is not able to learn a good latent representation of the input image. Or due to the lack of chrominance information in the input image, because of the nature of the input used, again, only the luminance channel of the CIELAB color space. The use of perceptual loss helps in improving the quality of the generated images, but not enough with the current architecture compared with other results in the literature.

There is a clear improvement step in the fidelity of the generated images around a PSNR value of 25: below this value, the images are desaturated or monochromatic, while above this value the images are more colorful. RMSE is a less reliable metric, because it is affected by the data; furthermore, at parity of value, the table 2 shows that is not much related to the visual quality of the images given, instead, by PSNR.

Model	RMSE	PSNR
Zhang's [10]	0.13	17.96
Proposed LAB	15.81	24.14
Xin's RGB [9]	0.13	22.46
Xin's YUV [9]	0.05	27.05

Table 2. Performance comparison

Figure 9. Example of predicted image

Figure 10. Example of predicted image

Figure 11. Example of predicted image

6.2. Points of failures

The network fails to capture shades of red, which means that the 'a' channel of the CIELAB color space is not reconstructed properly, tending to produce a greenish output or yellowish tint. This could be due to the fact that the dataset used is not balanced in terms of colors, and red is underrepresented. Another point of failure is that the network tends to produce a desaturated image, which is due to the intrinsic nature of regression based loss functions for

image generation tasks, which tend to average the possible outputs. In addition, the dataset is also unbalanced with images that have blue sky on the upper portion of the image, this leads to a lack of diversity in the training data and may contribute to the model's inability to generalize well to unseen images.

Figure 12. Example of predicted image at PSNR 22

The model also tested training with skip connections, like in a U-Net architecture, but the results were not satisfactory, probably due to the fact that the model was not able to learn to use them properly. Moreover, another test was done using residual connections within each block, using projection convolutional layers when the input and output dimensions did not match. This was done to help the model learn identity mappings, but it was immediately diverging during training instead of learning the mapping properly.

6.3. Limitations

The network would require a large amount of data to be trained properly, and a large number of epochs; one option to improve the performance could be to use data augmentation techniques to artificially increase the size and diversity of the training dataset. The training time is also a bit long per se, without considering the potential increase due to the above considerations. is also a bit long per se, without considering the potential increase due to the above considerations.

References

[1] Automatic image colorization using machine learning. 1

[2] Faming Fang, Tingting Wang, Tieyong Zeng, and Guixu Zhang. A superpixel-based variational model for image colorization. *IEEE Transactions on Visualization and Computer Graphics*, 26(10):2931–2943, 2020. 1

[3] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015.

[4] Mingming He, Dongdong Chen, Jing Liao, Pedro V. Sander, and Lu Yuan. Deep exemplar-based colorization. *CoRR*, abs/1807.06587, 2018. 3

[5] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016.

[6] Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi. Image colorization using generative adversarial networks. In *Articulated Motion and Deformable Objects*, pages 85–94, Cham, 2018. Springer International Publishing. 1, 3

[7] Rachaell Nihalaani. Image colorization using autoencoders. *International Journal for Research in Applied Science and Engineering Technology*, 9:1679–1683, 2021. 1

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 3

[9] Jin Xin, Yide Di, Qian Jiang, Xing Chu, Qing Duan, Shaowen Yao, and Wei Zhou. Image colorization using deep convolutional auto-encoder with multi-skip connections. *Soft Computing*, 27, 2022. 1, 3, 5

[10] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization, 2016. 3, 5