IMAGE COLORIZATION WITH DEEP LEARNING

Diego Cerretti **Beatrice Citterio** **Mattia Martino** **Sandro Mikautadze**

BAI - Academic Year 2023-2024

ABSTRACT

In this work, we assess the performance of various deep learning architectures to colorize grayscale images, using the MS COCO dataset. We train three main models: a convolutional neural network (CNN), a U-Net, and a generative adversarial network (GAN). For the CNN and U-Net, we use three loss functions to understand their impact on the colorization properties. We evaluate the models' performances using mean squared error (MSE), peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM), and Fréchet inception distance (FID) score. The results indicate that CNNs struggle to capture the color structure of images, whereas U-Nets achieve significantly better colorization across all evaluation metrics. GANs, although challenging to train, demonstrate comparable performance to U-Nets and show potential for improvement with additional tuning. The code is available at `https://github.com/sandromikautadze/image-colorization`.

Figure 1: Examples of images colorized by our models

1 Introduction

Image colorization is an image-to-image translation task and it consists of adding color to black-and-white images. This process dates back to more than a century ago ([27]) when it was typically performed manually by skilled artists, and today it has been proven useful in several fields, including photograph restoration ([12]), animations ([7], [28]), etc. Recently, deep learning has enabled automatic image colorization to become much more feasible and efficient, allowing for a new wave of research in the field.

In this project, we explored the performance of a classical convolutional neural network (CNN) [18], with convolutional and linear layers, a U-Net ([23]), an encoder-decoder architecture with skip connections, and finally a generative adversarial network (GAN) ([11]), consisting of a generator and a discriminator network (further details in Section 2 and 3). We aimed to determine whether CNNs, U-Nets, and GANs could effectively handle image colorization tasks, how these models compared when using different loss functions, and what the strengths and limitations of each model were. We trained our models using the Microsoft Common Objects in Context (MS COCO) *2014train* dataset [19], which includes over 82k images, with various objects and scenes.

This report will be structured as follows: in Section 2 we provide a short overview of the relevant literature that has informed and inspired our research. Then, in Section 3 we will explain the methodology behind our project, providing the architectures and training setups. In Section 4 we will present the results obtained and finally in Section 5 we will discuss what our findings, the limitations of this analysis and propose further improvements.

2 Related Work

The problem of image colorization has been extensively studied in various research projects using different models. In this section, we will briefly overview the literature and mention the ideas that inspired our work.

In 2015, Cheng et al. [6] proposed the first deep-learning-based colorization algorithm, which consisted of a modified multi-layer perceptron. Shortly after, improvements were made using CNNs ([17], [32], [8]), as their ability to learn complex features from images had already been proven successful by that time. Other variants using U-Nets were also shown to be effective in image colorization tasks ([9], [29], [30]). A U-Net, originally designed for biomedical image segmentation [23], is an instance of a fully convolutional network, i.e., a CNN without dense linear layers at the end. Its architecture consists of a downsampling path (encoder) that reduces the image size through a series of convolutional layers to a bottleneck, from which an upsampling path (decoder) increases the image resolution back to the original size, forming a U-shaped architecture. A key aspect of U-Nets is the skip connections between the encoder and decoder paths at each resolution level, which helps the decoder reuse and combine the low-level feature maps from the encoder.

All these works also explored various loss functions. Identifying the proper objective is crucial in tasks where the desired output is not easily quantifiable, such as image colorization. Commonly used losses include mean-squared error (MSE) or $\mathcal{L}_1$ loss ([8], [2], [32]), cross-entropy ([9], [2]), and other variations, which all tend to produce slightly different results, as we will later see with our models.

The introduction of GANs [10] partially addressed this problem since the GAN loss aims to match fake data to real data by predicting the underlying distribution. A GAN consists of a generator and a discriminator: the former learns to produce images that look like they were drawn from a given dataset to fool the discriminator, while the latter learns to distinguish between real images and fake images produced by the generator. Overall, GANs and their variations (DCGANs [22], WGANs [3], cGANs [20], etc.) have been widely successful in image colorization ([14], [21], [16], [5], [31]), providing state-of-the-art results in the field.

3 Methods

3.1 Setup

The general task in the deep-learning framework is to learn a mapping $\mathbf{f}$, from a grayscale image $\mathbf{X}$, that returns a colorized image $\hat{\mathbf{Y}} = \mathbf{f}(\mathbf{X})$, which is as similar as possible to the true color image $\mathbf{Y}$. Images are commonly represented in the RGB color space, and when colorizing, we would aim at predicting those three channels from a single-channel black-and-white image. Although colorizing from a single-channel to RGB is feasible, luminance-chrominance color spaces provide a much better setup: they separate the luminance from the chromatic information in separate channels, allowing us to predict fewer than three channels without losing information. A famous luminance-chrominance color space choice is CIELAB (also known as *Lab*), where the *L* channel represents the luminance component, capturing the intensity of light in the pixels, and the *a*, *b* channels encode the green-red and blue-yellow colors, respectively. By representing images in the *Lab* space, the objective reduces to using the *L* channel as our grayscale input to predict the *a* and *b* channels, which fully contain the color of the image.

Therefore, given an input grayscale image $\mathbf{X} \in \mathbb{R}^{1\times h\times w}$, represented as the *L* channel in the *Lab* color space, our goal was to learn a function $\mathbf{f} : \mathbb{R}^{1\times h\times w} \rightarrow \mathbb{R}^{2\times h\times w}$ such that $\hat{\mathbf{Y}} = \mathbf{f}(\mathbf{X})$, where $\hat{\mathbf{Y}}$ is the predicted output matching the true *ab* color channels $\mathbf{Y} \in \mathbb{R}^{2\times h\times w}$ and $h \times w$ are the height and width of the image, respectively.

3.2 Dataset

As said in Section 1, we used the MS COCO *2014train* dataset. We chose this dataset to guarantee a complete assessment of our models' performances under different contexts, scenarios, and colors. The final models were trained on a subset of 66k images (80% of the dataset), and tested on the remaining 16k images. Each image was downsized to 256×256 and the pixel values were normalized to $[0, 1]$ in the *Lab* color space.

3.3 Architectures

To determine the optimal model architectures and hyperparameters, we initially trained various configurations on a random subset of 5000 images from the dataset. This allowed us to evaluate the performance of different architectures and settings before training on the full dataset. Given our time and computational resources we did not attempt cross-validation. In this section, we present our final chosen architectures that have been trained on the whole training set of 66k images, and from which we drew our results.

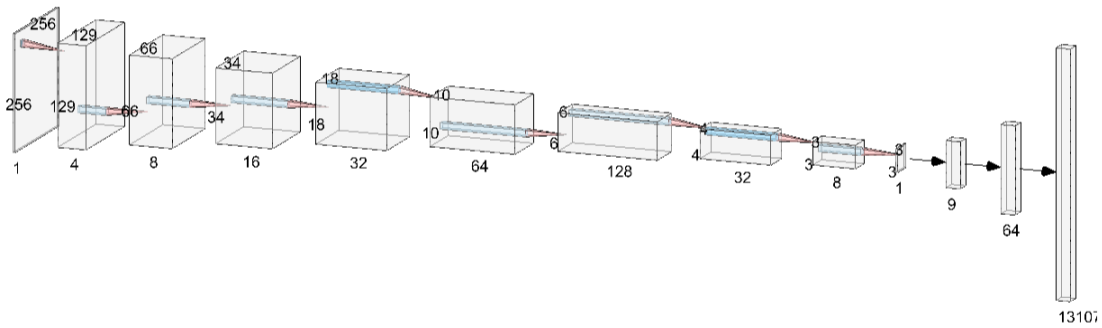

Figure 2: An illustration of the architecture of our CNN.

CNN Taking inspiration from Agrawal and Sawhney [2], we designed an architecture of 9 convolutional layers, with kernel size 3×3, stride 2, and padding 2. Each layer was followed by batch normalization and ReLU activation. The layers progressively increased the number of filters, starting from 4 to 128 filters, before reducing again towards the final layers. The output of the CNN was flattened and passed through two fully connected layers to produce the final color predictions, with sigmoid output to map values in the desired $[0, 1]$ range (see Figure 2).

U-Net The encoder pathway of our U-Net model consisted of 4 convolutional layers with batch normalization, followed by ReLU. The feature maps were downsampled with kernel size 4×4, stride 2, padding 1, starting from 32 filters and doubling at each layer up to 256 filters. At the bottleneck, there was an additional convolutional layer with 256 filters, a dilation rate 2, and padding 3, followed by batch normalization and ReLU. The decoder pathway mirrored the encoder with transposed convolutional layers for upsampling and then concatenated with corresponding encoder outputs through skip connections. This pathway reduced the number of filters progressively from 256 back to 2. The final output layer had a kernel size of 3×3, stride 1, and padding 1 with a sigmoid output function for the same reason as the CNN.

GAN The generator took the *L* channel and output the predicted *ab* channels, while the discriminator received the 3-channel color image and predicted the probability of it being real or artificially colorized. Taking inspiration from Isola et al. [14], we used an encoder-decoder generator (fully convolutional) and a PatchGAN discriminator. The encoder comprised 3 convolutional layers, with kernel size 4×4, stride 2, and padding 1, so they progressively downsampled the input image, with feature maps increasing from 128 to 512. Each layer, but the last, was followed by Leaky ReLU (slope 0.2). The decoder mirrored this structure with transposed convolutions to upsample the encoded image back to the original size, reducing the feature maps from 512 to 2. The final output layer used a sigmoid activation, for the usual reason. In all layers of the generator, apart from the first and last, we applied batch normalization before the activation. The discriminator we built was a PatchGAN, meaning that it produced a tensor of probabilities representing the likelihood of patches of the input image being real. It consisted of 4 convolutional layers with kernel size 4×4, stride 2, and padding 1, Leaky ReLU activation (slope 0.2), and feature maps of sizes $64, 128, 256$, and 2. Batch normalization was applied in the second and third layers. The final output layer with sigmoid had shape 16×16, meaning that each pixel of the output corresponded to the probability of a 46×46 patch of the original 256×256 image being real.

3.4 Training

We trained the CNN and the U-Net on batches of size 64, using Adam [15] with fixed learning rate $\alpha = 10^{-4}$ and decay rates $\beta_1 = 0.9, \beta_2 = 0.999$. Recalling that the output channels c where two (i.e. *a* and *b*) and that the images had shape $h = w = 256$, for both models we used three loss functions:

$$\text{MSE}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{2hw} \sum_{i=1}^{h} \sum_{j=1}^{w} \sum_{c=1}^{2} \|\mathbf{Y}_{cij} - \hat{\mathbf{Y}}_{cij}\|_2^2 \tag{1}$$

$$\mathcal{L}_1(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{2hw} \sum_{i=1}^{h} \sum_{j=1}^{w} \sum_{c=1}^{2} |\mathbf{Y}_{ijc} - \hat{\mathbf{Y}}_{ijc}| \tag{2}$$

$$\text{Smooth-}\mathcal{L}_1(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{2hw} \sum_{i=1}^{h} \sum_{j=1}^{w} \sum_{c=1}^{2} \begin{cases} \frac{1}{2\beta} \|\mathbf{Y}_{cij} - \hat{\mathbf{Y}}_{cij}\|_2^2 & \text{if } |\mathbf{Y}_{ijc} - \hat{\mathbf{Y}}_{ijc}| < \beta \\ |\mathbf{Y}_{ijc} - \hat{\mathbf{Y}}_{ijc}| - \frac{\beta}{2}, & \text{otherwise} \end{cases} \tag{3}$$

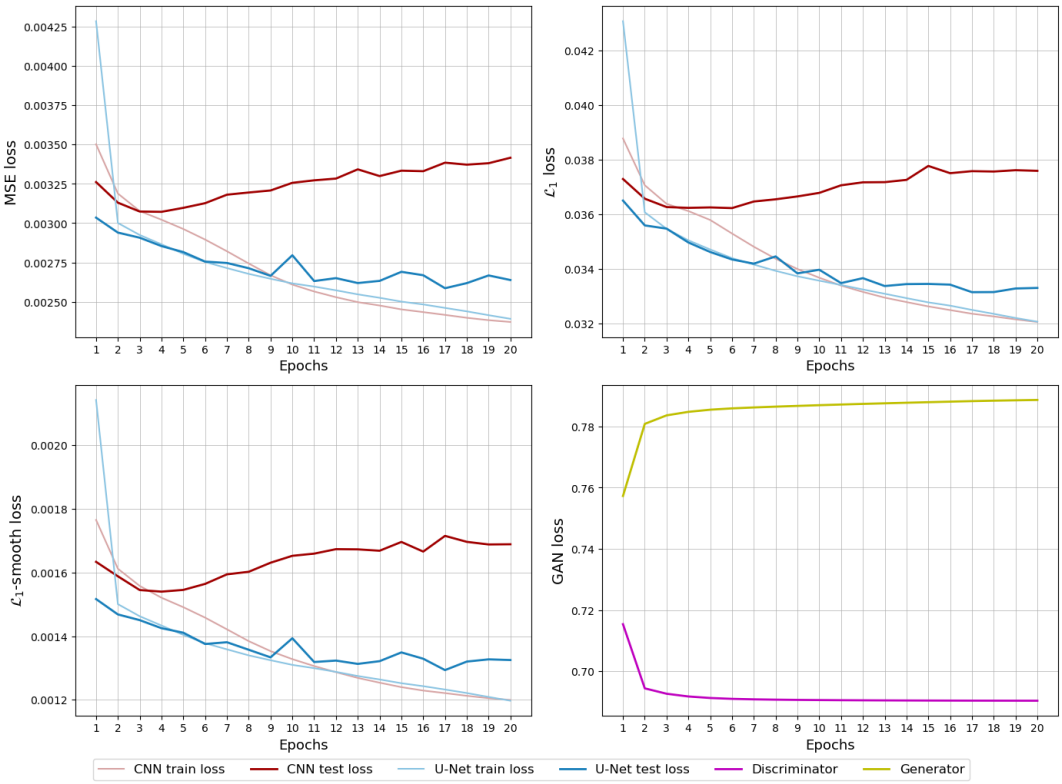

Figure 3: Values of different losses on our models

We chose $\beta = 1$ for the smooth $\mathcal{L}_1$ loss.

For the GAN, the generator was trained using the Adam optimizer with parameters $\alpha = 0.0002, \beta_1 = 0.5, \beta_2 = 0.999$. The discriminator was trained using Nesterov SGD with fixed learning rate $\alpha = 0.0002$ and momentum 0.9. The batch size was set to 32 for both architectures. Now, let D be the discriminator, G the generator, $\mathbf{X}$ the input L channel, $\mathbf{Y}$ the true *ab* channels and $\hat{\mathbf{Y}} = G(\mathbf{X})$ the generated *ab* channels. We chose to optimize the usual GAN loss

$$\min_G \max_D V(D, G) = \mathbb{E}_{(\mathbf{X},\mathbf{Y})\sim p_{\text{data}}(\mathbf{X},\mathbf{Y})}[\log D(\mathbf{X}, \mathbf{Y})] + \mathbb{E}_{\mathbf{X}\sim p_{\mathbf{X}}(\mathbf{X})}[\log(1 - D(\mathbf{X}, G(\mathbf{X})))] \tag{4}$$

The discriminator tries to maximize V, distinguishing between real pairs $(\mathbf{X}, \mathbf{Y})$ and fake pairs $(\mathbf{X}, G(\mathbf{X}))$. Notice that here a pair represents the entire 3-channel image. Conversely, the generator tries to minimize V, attempting to generate colors $G(\mathbf{X})$ that can fool the discriminator and be classified as real.

We adopted various known techniques [24] to avoid vanishing gradients, mode collapse, and other training failures, which we extensively observed when trying architectures on 5k images. To be more precise, we added $\mathcal{L}_1$ regularization with $\lambda = 0.5$ on G [1], one-sided label smoothing to true labels (random values in $[0.9, 0.95]$), and Gaussian noise of $\sigma = 0.01$ to the inputs of D. We observed that these regularizations reduced the overconfidence of the discriminator, improving the training dynamics and the generated images.

All the models were trained for 20 epochs.

4 Results

4.1 Loss

Figure 3 shows the training and testing loss curves for our models. CNN and U-Net show similar decreasing loss trends in training, while in testing losses tend to decrease initially and then stabilize. For the CNN, we see that with both losses the gap between test and train losses increases, indicating early signs of possible overfitting. Instead, the difference in testing and training losses for the U-Net is yet negligible. U-Net generally outperforms CNN with lower loss values, indicating better performance. As for the GAN, the discriminator loss decreases while the generator loss increases,

which is typical in the early phases of GAN training. However, after a few epochs, the discriminator loss stabilizes, while the generator loss slowly increases. The cause of this needs further investigation, but a possible explanation could be due to vanishing gradients of the discriminator, that fail to send signals to the generator and improve.

4.2 Metrics

Loss values alone do not provide a complete picture of the performance. Indeed, the main goal of a colorization result is to convince the viewer, to make him believe in the authenticity of the colorized image, and not necessarily to reconstruct the color accurately. Since group subjective quality assessments were out of our scope, we opted for objective quantitative metrics. We used the following chrominance quality metrics: MSE, Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM)[26], and Fréchet Inception Distance (FID) score [13].

MSE is defined as in Equation 1. Lower MSE means better results.

PSNR assesses the ratio between the maximal possible value of the signal and the power of the noise (error):

$$\text{PSNR}(\mathbf{Y}, \hat{\mathbf{Y}}) = 20 \cdot \log_{10} \left(\frac{\text{MAX}_{\mathbf{Y}}}{\sqrt{\text{MSE}(\mathbf{Y}, \hat{\mathbf{Y}})}} \right) \tag{5}$$

where $\text{MAX}_{\mathbf{Y}}$ is the maximum pixel intensity, i.e. 1 in our case. Higher values indicate a better colorization quality; results are expressed in decibels.

SSIM finds the similarity between two images by observing the changes in properties like luminance, contrast, and structure:

$$\text{SSIM}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{(2\mu_{\mathbf{Y}}\mu_{\hat{\mathbf{Y}}} + C_1)(2\sigma_{\mathbf{Y}\hat{\mathbf{Y}}} + C_2)}{(\mu_{\mathbf{Y}}^2 + \mu_{\hat{\mathbf{Y}}}^2 + C_1)(\sigma_{\mathbf{Y}}^2 + \sigma_{\hat{\mathbf{Y}}}^2 + C_2)} \tag{6}$$

where the terms μ, σ are mean and variance of the images. SSIM ranges from -1 to 1, taking higher values when there is strong similarity.

FID uses a pre-trained Inception network [25] to compare the distribution of features that are extracted from the colorized images to those from the real images. Lower FID scores suggest that the colorized images are more similar to the real images.

A visual summary of the final metrics is in Figure 4. The results suggest that the U-Net outperforms the CNN architecture in all losses, as evidenced by lower MSE values, higher PSNR and SSIM values, and better FID scores. Furthermore, using the $\mathcal{L}_1$ smooth loss function seems to improve reconstruction quality for both models compared to the $\mathcal{L}_1$ or MSE loss functions alone.

For the GAN, as seen in the previous section, the generator and discriminator losses started to saturate after a few epochs. So we decided to compare the results also at epoch 10, shown in Table 1. We do not see a general difference across MSE, PSNR, and SSIM, but we notice a sharp increase in FID at epoch 20. Doing visual inspection we indeed see that the model trained for 20 epochs started creating artifacts over the sepia patches produced at the model of epoch 10. With great surprise, compared to U-Nets, we don't see improvements in these metrics: MSE and PSNR stay between the ones obtained by CNNs and U-Nets, while SSIM and FID are even worse than both CNNs and U-Nets.

Model	**Avg. MSE** $\pm\sigma$	**Avg. PSNR** $\pm\sigma$	**Avg. SSIM** $\pm\sigma$	**FID**
Generator epoch 10	0.0031 ± 0.0005	50.2292 ± 1.4314	0.7556 ± 0.0862	10.5602
Generator epoch 20	0.0031 ± 0.0005	50.3388 ± 1.4666	0.7414 ± 0.0909	14.4715

Table 1: GAN performance metrics at different epochs.

4.3 Overall Performance

The CNN returns sepia-toned images, so mostly brownish colors. This matches known literature, especially with our loss choices. Indeed, CNNs are not complex enough to learn the true colors well, so the loss tends to produce colors that can most often minimize the difference between the true and fake images. It turns out that brownish colors are the ones that do the job. Overall, the results are far from good.

U-Nets instead, even with the same losses, obtain significantly better results. They capture a much higher chromatic range and identify more complex colors like green and blue with more accuracy. Even though there are imperfections, we can say that there is a significant improvement with respect to CNNs.

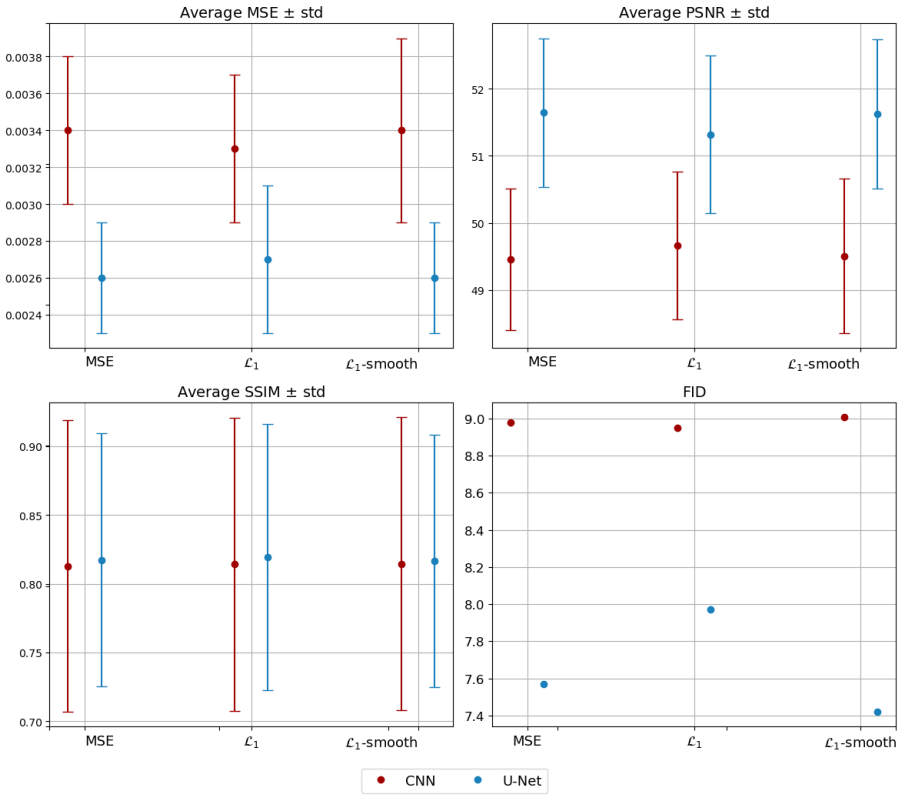

Figure 4: CNN and U-Net performance metrics on the test set.

GANs instead show a qualitative behavior similar to that of U-Nets, doing visual inspections. We observe that the model trained for 10 epochs displays better metrics and colors, as the 20-epochs model oversaturates red areas. In GANs we also observe the presence of artifacts in the corners, which we do not have an explanation for.

For visual examples supporting our observations, please refer to the images in the Supporting Images section at the end of this document.

5 Discussion

Overall, we have trained three different architectures, CNN, U-Net and GAN, with different losses, and then we have used several metrics to assess the performance of our models. As mentioned in Section 4.3, U-Nets perform slightly better than CNNs, while the GAN was not as satisfactory as we expected, which makes sense given that it has been hard to train and given the constraints we had on time and computational power. The main issue we encountered with CNNs and U-Nets was that the results had brownish colors and the models struggled to capture most of the other shades, even though some were easier to identify, such as green and blue. These things may be caused by many factors, but one could be the presence of an unbalanced dataset, where certain colors are more present than others. Improvements could be made by better inspecting the dataset first.

Architecture-wise the CNN we built shrinks the size of the image too much, making it very hard to capture general colors of objects in an image. Improvements could be reached by using different convolutions. As for the U-Net and GAN, taking inspiration from other papers could be a useful exercise to test the performance. It could also be interesting to improve the other GAN networks we tried on the 5k subset of images to see what results they yield on the entire dataset.

Several alternative improvements could be made to expand this analysis. For example, we could try to use a different color space, such as YUV, where the Y component represents the luminance and U and V represent the chrominance. It could be interesting to even use RGB itself, and then compare the performance with the same architectures we employed here. Furthermore, instead of considering pixel-wise losses only, which can sometimes lead to overly smooth

or unrealistic results, one could use perceptual losses [4], similar to those of GANs. This could allow the model to focus on capturing the perceptual similarity and overall visual quality rather than just pixel-level accuracy.

With more computational resources, one could expand on the hyperparameter search, finding better learning rate parameters and constructing potentially better architectures. Alternatively, it could be interesting to use pre-trained models and fine-tune them on our specific task.

References

[1] A. Abu-Srhan, M. A. Abushariah, and O. S. Al-Kadi. The effect of loss function on conditional generative adversarial networks. *Journal of King Saud University - Computer and Information Sciences*, 34(9):6977–6988, 2022.

[2] M. Agrawal and K. Sawhney. Exploring convolutional neural networks for automatic image colorization.

[3] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan, 2017.

[4] C. Ballester, A. Bugeau, H. Carrillo, M. Clément, R. Giraud, L. Raad, and P. Vitoria. Analysis of different losses for deep learning image colorization, 2022.

[5] Y. Cao, Z. Zhou, W. Zhang, and Y. Yu. Unsupervised diverse colorization via generative adversarial networks, 2017.

[6] Z. Cheng, Q. Yang, and B. Sheng. Deep colorization, 2016.

[7] Y. Ci, X. Ma, Z. Wang, H. Li, and Z. Luo. User-guided deep anime line art colorization with conditional adversarial networks. In *Proceedings of the 26th ACM international conference on Multimedia*, MM '18. ACM, Oct. 2018.

[8] R. Dahl. Automatic colorization. *Automatic colorization*, 2016.

[9] D. Goel, S. Jain, D. Kumar Vishwakarma, and A. Bansal. Automatic image colorization using u-net. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 01–07, 2021.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[12] M. He, D. Chen, J. Liao, P. V. Sander, and L. Yuan. Deep exemplar-based colorization, 2018.

[13] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[14] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks, 2018.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[16] X. Kuang, X. Sui, C. Liu, Y. Liu, Q. Chen, and G. Gu. Thermal infrared colorization via conditional generative adversarial network, 2018.

[17] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization, 2017.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context, 2015.

[20] M. Mirza and S. Osindero. Conditional generative adversarial nets, 2014.

[21] K. Nazeri and E. Ng. Image colorization with generative adversarial networks. *CoRR*, abs/1803.05400, 2018.

[22] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[23] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[24] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans, 2016.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions, 2014.

[26] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[27] A. Weitz. The basics of hand-coloring black-and-white prints, 2020. https://www.bhphotovideo.com/explora/photography/tips-and-solutions/the-basics-of-hand-coloring-black-and-white-prints%E2%80%82%E2%80%82%E2%80%82.

[28] S. Yoo, H. Bahng, S. Chung, J. Lee, J. Chang, and J. Choo. Coloring with limited data: Few-shot colorization via memory-augmented networks, 2019.

[29] M. H. Zayd, N. Yudistira, and R. C. Wihandika. Image colorization using u-net with skip connections and fusion layer on landscape images, 2022.

[30] L. Zhang, Y. Ji, and X. Lin. Style transfer for anime sketches with enhanced residual u-net and auxiliary classifier gan, 2017.

[31] L. Zhang, C. Li, T.-T. Wong, Y. Ji, and C. Liu. Two-stage sketch colorization. *ACM Transactions on Graphics (SIGGRAPH Asia 2018 issue)*, 37(6):261:1–261:14, November 2018.

[32] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization, 2016.

6 Supporting Images

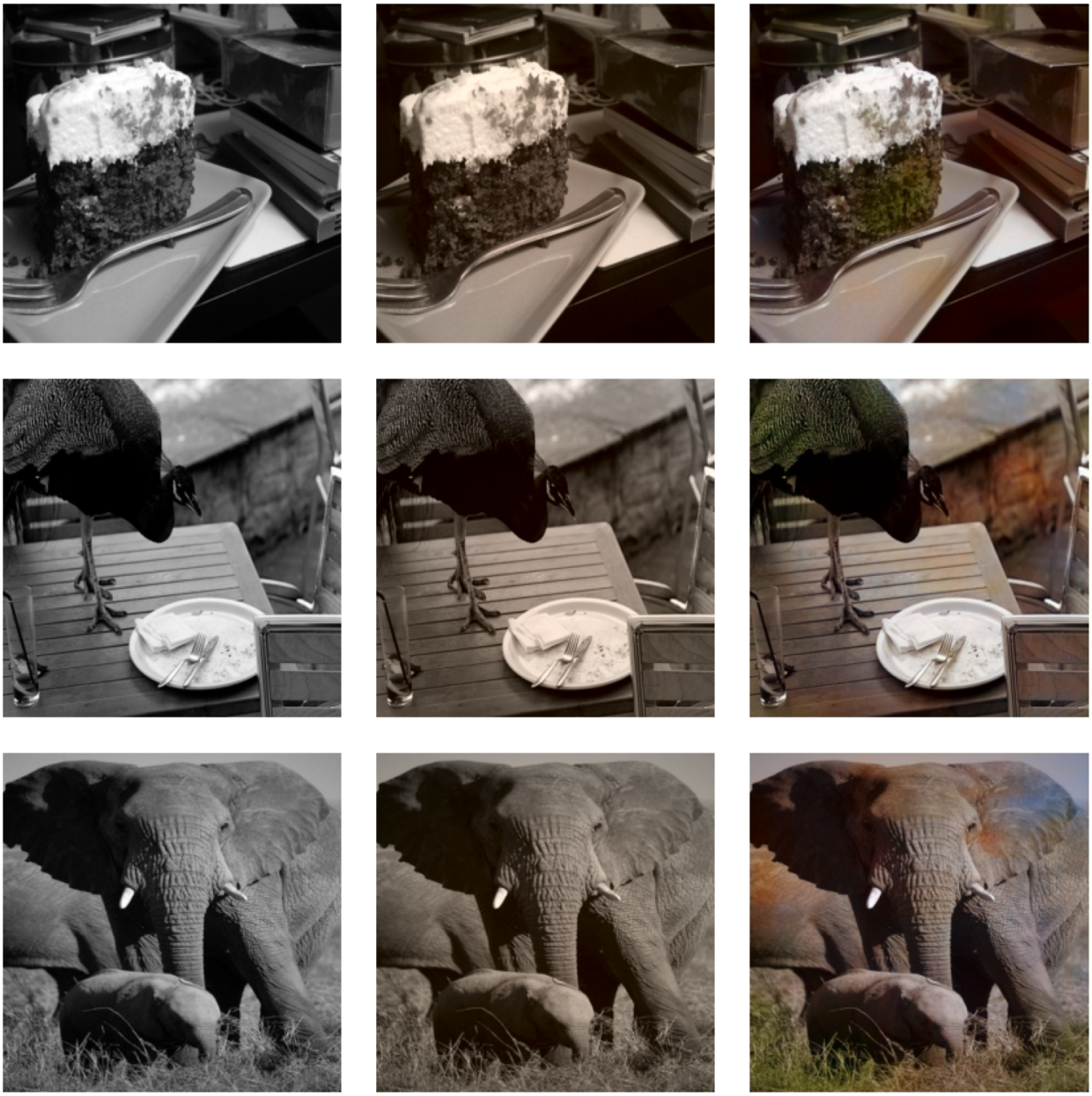

Figure 5: CNN (middle) generates images with sepia colors, while U-Net (right) has a wider color spectrum. Still, we observe a preference for brownish colors.

Figure 6: U-Net can capture green and blue colors, ideally representing the grass and the sky.

Figure 7: GAN results. The first row is the model at 10 epochs; the second at 20 epochs. In both cases, the model effectively predicts brownish tones but struggles with accurately representing other colors in the spectrum. The 10-epoch model produces sepia patches (probably due to the $\mathcal{L}_1$ regularization on the generator) and the 20-epoch model tends to over-saturate those regions. This also seems to occur in blue regions.