Artistic colourization with Adaptive Instance Normalization

Cristi Andrei Prioteasa
Universität Heidelberg
github.com/PrioteasaAndrei
cristi.prioteasa@stud.uni-heidelberg.de
Matrikelnummer: 4740844
M.Sc. Data and Computer Science

Matteo Malvestiti
Universität Heidelberg
github.com/Matteo-Malve
matteo.malvestiti@stud.uni-heidelberg.de
Matrikelnummer: 4731243
M.Sc. Data and Computer Science

Jan Smoleń
Universität Heidelberg
github.com/smolenj
wm315@stud.uni-heidelberg.de
Matrikelnummer: 4734263
M.Sc. Data and Computer Science

Abstract

In this paper we explore a joint method for join colorization and color style transfer for grayscale images. We propose a UNet architecture enchanced with Adaptive Instance Normalization [6] and a style encoder based on a VGG19, which colorizes grayscale images and transfers the style (color style only) onto the content image. We explore the influence of the chosen colorization loss (MSE vs LPIPS), the chosen colorspace (RGB vs LAB) and conduct an ablation study on different configurations of the newtork.

1. Introduction

Colorization of grayscale images has been an area of significant interest in computer vision, with applications ranging from photo restoration to artistic expression. We propose an approach to colorize grayscale images in various artistic styles using a U-Net architecture enhanced with Adaptive Instance Normalization (AdaIN) layers. U-Net, known for its effectiveness in semantic segmentation tasks, provides an ideal framework for our colorization task due to its ability to capture spatial dependencies while preserving fine details. By incorporating AdaIN layers into the U-Net architecture, we introduce the capability to adaptively transfer artistic styles (here we use style to refer to color choices) from reference images to grayscale inputs. AdaIN enables the decoupling of content and style in feature representations, allowing us to leverage the content information from grayscale images while infusing them with the stylistic characteristics extracted from reference color images. This style-guided colorization approach opens up new possibilities for artistic expression, allowing users to apply various painting styles, from impressionism to surrealism, to grayscale inputs.

1.1. Motivation

This project has his roots in our fascination for the colorisation task: an underconstraied task that is still under research and has big room for improvements. It is mathematically impossible to deduce from the single channel of grayscale images the 3 RGB or LAB channels. The only certainty is the color intensity pixel per pixel, but it is necessary to ”guess” the colors. Deep Convolutional Neural Networks are a powerful tool to tackle this task, since they can learn to understand the semantics of the image itself. Moreover the AdaIN encoder, if used in training on the ground truth colored image itself, can be a powerful tool to encode the colour semantics. This addition alone, consisting in a regularisation and having yet no capacity to do style transfer, lead to a great improvement in colorisation quality.

1.2. Related work

Most of the related work deals with the non-parameter version of AdaIN, as introduced in paper *Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization* [5] or work with some sort of Generative Adversarial Networks as they are more expressive and better suited for this task. There are three main approaches in the literature of UNet based colorization: combining color and texture

in the transformation (called style), colorizing without any style information, such as in *Image Colorization using U-Net with Skip Connections and Fusion Layer on Landscape Images* [8] (which served as an initial starting point for our work) or considering color and texture separatly and allowing for a controllable blend between the two (see *Aesthetic-Aware Image Style Transfer* [4]). In our work we deal only with color transfer in different styles and no texture transfer. We analyze *Analysis of Different Losses for Deep Learning Image Colorization* [3] and conclude that the a more complicated colorization loss does not lead to significantly better results.

2. Methods

2.1. The baseline UNet

The central component of our architecture is the *UNet* [7], which remains a state-of-the-art solution for image segmentation, making it an ideal choice for our colorization task. The encoder comprises a series of convolutional layers with 3x3 filters, each followed by a ReLU activation and batch normalization. This structure is visually represented by the orange segments in the diagram adjacent to the convolution blocks. Following each set of convolutions, a max-pooling layer reduces the feature map dimensions by half.

The bottleneck section reduces the spatial resolution to 1/16 of the original, aligning with the computational constraints of working with 128x128 images. Thus, the bottleneck feature maps have a size of 8x8.

The decoder does not precisely mirror the encoder, but it maintains the same patterns of changes in channel counts and feature map dimensions. Upsampling is achieved using transpose convolution. Similarly, ReLU activations and batch normalization are applied throughout the decoder, indicated by the orange segments.

A distinguishing feature of UNet is the use of skip connections. In this configuration, there are five skip connections. At each skip connection, the feature maps from the encoder are concatenated with the corresponding upscaled feature maps from the decoder. These concatenated feature maps are then processed using 1x1 convolutions to fuse them.

The network expects input in the form of $[C, H, W]$, where the number of channels C can be 2 for the LAB colorspace and 3 for the RGB colorspace. We chose to train our model using the RGB color space because the LAB tranformation introduces a number of artefacts from numerical instability of the transformation which yields poorer results.

Together, this architecture forms the backbone of our neural network, providing a robust and efficient structure for the image colorization task.

This network alone (without the style encoder and AdaIN layers) was tested in a preliminary phase and resulted to be able to perform both the recreation of images and colorisation, although with notable overfitting problems.

Train dataset size	2000
Validation dataset size	200
Batch size	16
Epochs trained	231
Colorspace	RGB
Optimizer	Adam w. Weight decay
Notes	Dropout used as regul.

Table 1: Details on UNet model for colourization

Below you can see:

1. Loss curves for only UNet tasked with colorization
2. Colorization performance on a training image
3. Colorization performance on a test image (never seen before)

The results were achieved using the network configuration detailed in table 1:

Adaptive Instance Normalization and Style Encoder

In order to achieve style transfer (only colour) we augment the base UNet with two additional components: a style encoder which encodes the color features of the original image(we will later discuss why only the color features and not also the texture features of the original image are encoded) and AdaIN layers which align the style feature maps with the content image feature maps.

We implement a version of Adaptive Instance Normalization as in the *Style GAN* paper [6], given by (see paper [6] for full description of the terms):

$$AdaIN(x_i, y) = y_{s,i} \cdot \frac{x_i \mu(x_i)}{\sigma(x_i)} + y_{b,i} \qquad (1)$$

This layers aims to align per channel the mean and variance of the feature space of the encoded style image with the feature maps of the original image at different depths in the network using learned affine transformations (MLPs in our network), in this way achieving style transfer (color transfer).

The AdaIN encoder runs only one per epoch, embedding the style image in a compressed latent space. After every block of convolutions, followed by ReLu activations and batch normalization, the feature maps get normalized with the mean and std of the adain interface. Since in the different stages of the UNet the feature maps have different channel sizes, a dense linear layer is put in between the AdaIN latent space and the feature maps.

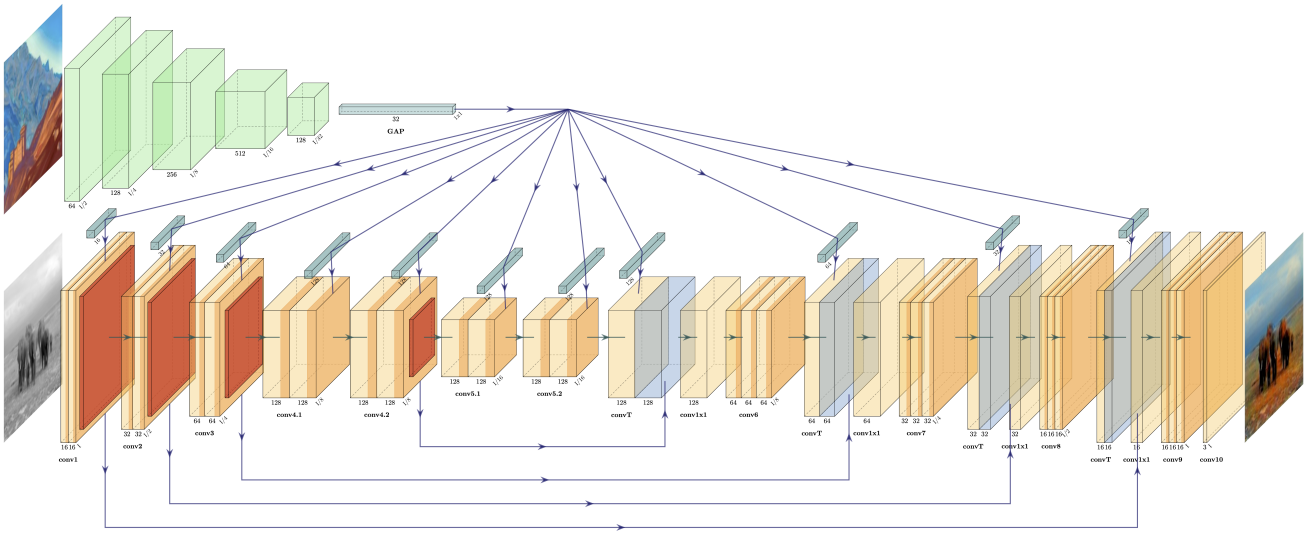


Figure 1: Map of our architecture

Figure 2: Loss landscape for the UNet trained solely for colourization

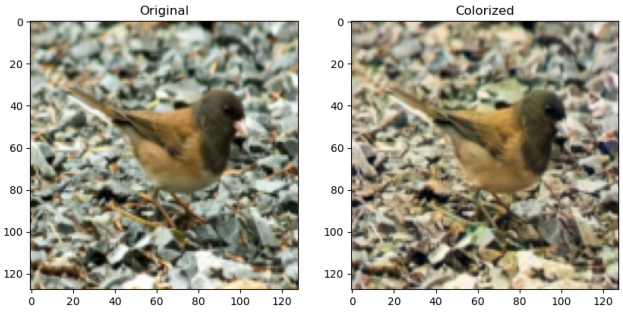


Figure 3: Example from the training dataset

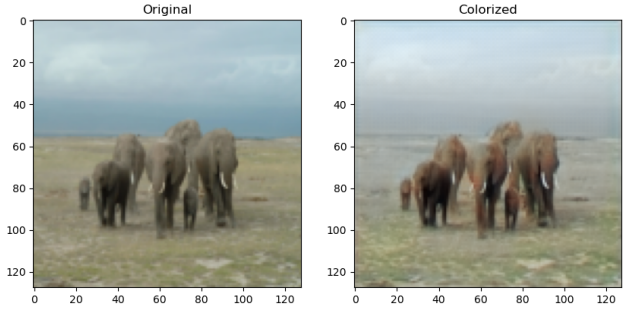


Figure 4: Example from the validation dataset

The style encoder is a simple encoder followed by a Global Average Pooling (GAP) layer, which encodes the colour features in a latent space dimension of 32 (we later discuss how the dimension of the latent space influences the colorization and style transfer).

2.2. Training & Dataset

Since we didn't have available computing resources, we used a personal laptop with NVIDIA GeForce RTX 3050TI, 4096MiB VRAM and a M1 MacBook Air (16Gb RAM) for training. This limit our ability in terms of resolution of the input images. Training with patches of size 256×256 yielded poor results even after prolonged training, but using patches of size 128×128 yields decent results which generalize.

We experiment with two types of losses: MSE against the original image and *Learned Perceptual Similarity* [9], as a way to encourage the network to learn semantic colorization information about the image instead of faithfully repro-

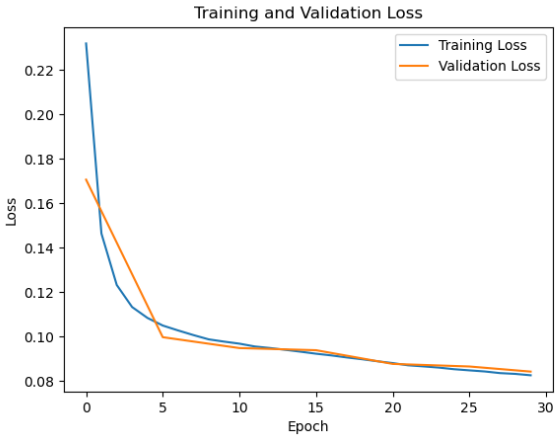


Figure 5: Loss curves of the final model. All details about the model in section 3.

ducing the colors of the original image. We use a weighted sum of both losses for our training.

Implicit training

During training we pass as style image the colour original image itself. Then, we train on colorizing the gray-scale image, computing the loss of the recreated image against the original colour image. The loss is a combination of MSE pixel by pixel and of *Learned Perceptual Similarity* [9]. We use learned perceptual similarity (LPIPS) as it gives the network more freedom in learning semantically equivalent colors for different regions, instead of learning to reproduce the exact color from the training set. During backpropagation the various AdaIN interfaces and the encoder get implicitly trained.

In the final version of the model, there is no training against different style images. This approach was tried and will be discussed futher ahead. It needs the definition of a new style loss and good balancing with the colorization loss. Anyways, this should not be necessary: so long as scarcity is induced in the baseline UNet, it should resort in relying on the encoded data of the style image, provided by AdaIN interface. Thus, when passing a new style image at inference time, this implicit training should be enough to transfer the style.

Dataset and training analysis

We use the *imagenet-1k* dataset [2] for our training. We experiment with different training set size and for our final model we use 50k training images and 500 validation images. As style images to test at inference phase, we sample from the *wikiart* dataset [1].

Figure 7 shows the loss landscape of the final model for 32 epochs, after which a divergence between train and validation's losses was observed. We delay all the details on the training of the final model to section 3.5.

3. Experiments and ablation studies

In this section we present a reorganized history of the thought process that led us to the final prototype. This will only contain the most crucial experiments, those from which we learned something that brought us a step closer to out goal. Many more experiments were done, like those on the LAB color-space, and they can be found in the [Appendix]().

3.1. Achieving colorization without style transfer

Our first struggle was definitely to train a Neural Network that was able to recreate first, and then to colourise images. We started from an encoder-decoder architecture that had a pretrained VGG19 encoder, discarding the classification head. The idea was to save a lot on computational cost. Unfortunately it wasn't enough: if we froze the encoder and trained the decoder alone, we were underfitted. If we unfroze the encoder, VGG19 was simply not worth it. Indeed, it was a big network, taking a lot of computational resources, but at the same time it didn't offer skip connections or the flexibility to insert AdaIN layers, not while wanting to load the pretrained weights at least.

Our first results in recreation and colourisation came when we started following the approach described in *Image Colorization using U-Net with Skip Connections and Fusion Layer on Landscape Images* [8]. It was by manually recreating layer by layer their UNet that we overcame our troubles.

3.2. Introducing style transfer using AdaIN layers

The next step, namely the introduction of AdaIN layers for regularisation, was discussed in a meeting with our tutor and had no direct backup from scientific papers. The Adaptive Instance Normalization is done like in [Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization](https://arxiv.org/abs/1703.06868), but with a big difference: the parameters of mean and standard deviation are learned every time from simple linear layers that act as interfaces between the latent space of the AdaIN encoder and the feature maps where normalization takes place. If you refer to the big architecture map in the beginning of this documentation, we are talking about the blue pins. This idea of learning the AdaIN parameters comes from NVIDIA's 2019's *Style GAN* paper [6].

Train dataset size	50000
Validation dataset size	500
Batch size	16
Epochs trained	18
Colorspace	RGB
Latent space dimension	128
Epochs	32

Table 2: Details on UNet+AdaIN model for colourization and style transfer.

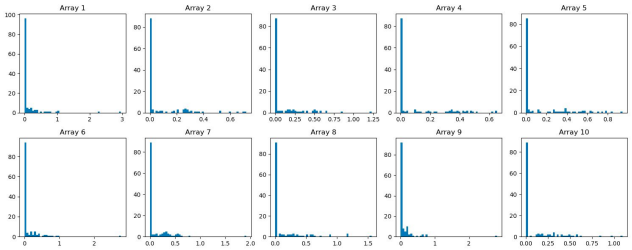


Figure 6: Histogram of the values in the latent space of the style encoder for random style images. It can be noticed how the values are all distributed very close to zero.

3.3. Investigation on the latent space of the style encoder

For the following section, the experiments were conducted using the configuration outlined in table 2:

AdaIN (Adaptive Instance Normalization) was effectively disregarded by the baseline UNet architecture. After a few iterations, the latent space in the AdaIN encoder was driven entirely to zero (see next image), indicating that the network was suppressing its influence in favor of straightforward reconstruction. As a result, during the inference phase, when different style images were input, there was no noticeable effect on the output, demonstrating that the style information was not contributing to the reconstructed image.

3.4. Attempted solutions

Solution 1: training with different pairs of (context + style) images

To achieve this, we build a new dataloader that sampled pairs of context and style images and we fed them both to the network. The forward pass is pretty intuitive. The backward pass needed some more careful thinking though. Our first idea was to introduce a new loss, thus distinguishing between colorization and style loss. The colorization loss was exactly the same as before, a linear combination of MSE and lpips between recreated image and original colour image.

The style loss was obtained in the following way: 1) We stored the latent space of the AdaIN encoder, namely the embedding of the style image 2) After the forward pass, without computing the gradient, we passed the recreated image through the AdaIN encoder, thus obtaining the embedding of the recreated image 3) We computed MSE between the two latent spaces The final loss was a weighted linear combination of the two losses.

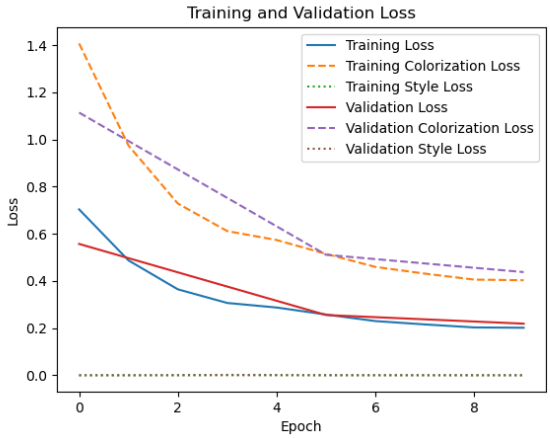


Figure 7: Failed attempt with combined loss approach.

We hoped to manually force the network to learn style transfer, but we were unfortunately stopped by a very basic inconvenience: the balancing of the two losses. Indeed, the colorization loss was of the order of magnitude of 10^{-2} or 10^{-3}, while the style loss, which we remember was computed on the embedding obtained through the same encoder, was much lower, with values oscillating between 10^{-5} and 10^{-8}. We realised that without additional learning techniques, we would have never found an optimal way to balance these two losses. Using a sigmoid function for the style encoder ensured that the two losses were now in the same range, but that did not make any difference in the training process. The combined loss was defined as

$$combined\_loss = \alpha \cdot color\_loss + (1 - \alpha) \cdot style\_loss$$

Solution 2: using two different optimizers

The dual-opt model was exactly what it seems from the name: at every iteration we executed two forward passes and two backward with two different optimizers and loss functions: first the colorization loss and then the style loss. More importance was given to the colorization by passing the optimizer a learning rate 10 times higher than the one of the style optimizer.

We soon experienced bad results:

- The loss of the style optimizer, although small, wasn't converging at all. Not in the first 15 epochs at least.

Train dataset size	50.000
Validation dataset size	500
Batch size	16
Colorspace	RGB
AdaIN encoder latent space dimension	32
Dropout rate after every convolution	0.2
Epochs	32

Table 3: Details on UNet+AdaIN model for colourization and style transfer.

- The loss of the colorization optimizer was better, but polluted by the constant meaningless changes of the style optimizer.

Soon we would realise, by investigating the values of the latent space of the AdaIN encoder, that it was cast to zero by the backpropagation of the colorization optimizer, cutting out all its power to represent the style input. The style optimizer had little of sense to work on, leading just to worse and worse results.

The dual-opt approach was not viable in the brutal state we described and tested. One way to refine it would have been to associate with each optimizer only certain parameters, especially allowing the colorization optimizer only to backpropage on the baseline UNet.

But actually, the takeaway from this experiments was learning how the training proceeded, how the weights were cast to zero by the colorization loss and also how we were underestimating the capabilities of the AdaIN encoder. The whole point of leaning the AdaIN parameters was intended for implicit training, for letting it learn the true colour semantics of the style image, without forcing the network externally with different couples of content and style-images.

3.5. The Final Model

The final configuration follows the parameters outlined in table 3

The solution lied in introducing scarcity in the baseline UNet, so that it would look into the style representation, instead of trying to recreate itself perfectly and cast all the AdaIN parameters to 0. We rewrote the UNet to be much lighter, by reducing the number of channels to $1/4$ throughout the whole network and all skip connections and AdaIN pins respectively. Another change was to reduce the latent space of the AdaIN encoder from size 128 to size 32, making it more meaningful and compatible with the channel dimensions of the new UNet. [1]

After introducing dropout layers and training for 32 epochs on 50k training images we obtain clear colorization and style transfer. For some results refer to fig. 9.

[1] We also tested sizes 16, 64 and 128. Size 32 proved to be the one for which we had better results.

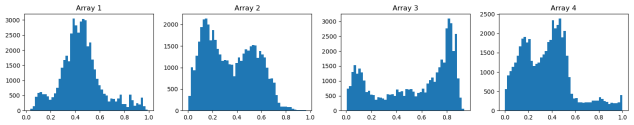


Figure 8: Histogram of the values in the latent space of the style encoder for random style images. This distributions cannot be directly interpreted, but their high variance guarantees that the encoder manages to produce a meaningful representation of the colour style.

The latent space of the style encoder is no longer null and, as fig. 8 clearly shows, we can see different distributions for every image passed through the AdaIN encoder. This experiment ensures that the encoder manages to produce a meaningful representation of the colour style. The fact that it's not all set to zero also proves that the training process works as expected, with the baseline UNet actively relying on the style image.

We encourage the reader to read the Appendix for more result samples.

4. Conclusion

In conclusion, we have presented a novel UNet architecture enhanced with Adaptive Instance Normalization (AdaIN) layers for gray-scale image colorization in diverse artistic styles. We achieve good colorization results and clear style transfer, while keeping the network lightweight (3.5M parameters) and can fit on consumer grade GPUs. We conducted experiments on the influence of the latent space dimension and investigated the expressiveness of the network in term of depth and number of parameters. We investigated different normalization techniques, influence of the used colorspace (RGB vs LAB) and texture vs color transfer.

5. Appendix

5.1. Further experiments

In this section, we outline some of the experiments that didn't make it into the final cut and didn't present meaningful results toward our goal. However, they were an interesting study and provide a more comprehensive picture of the efforts put into this project.

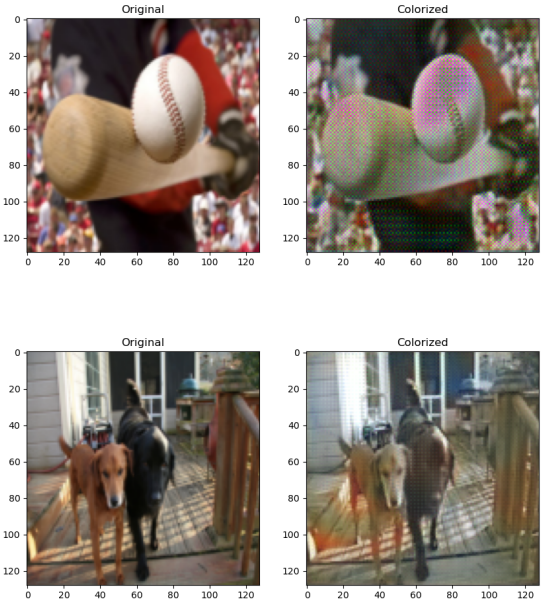
5.1.1 Using only LPIPS (Learned Perceptual Similarity) as the loss function

Using only LPIPS as our loss function yields some interesting artifacts in a grid-like pattern that we cannot explain.

LPIPS colorization artifacts (without style transfer)

Train dataset size	50,000
Validation dataset size	500
Batch size	16
Colorspace	LAB
AdaIN encoder latent space dimension	32
Dropout rate after every convolution	0.2
Epochs	32

Table 4: Experiment Parameters



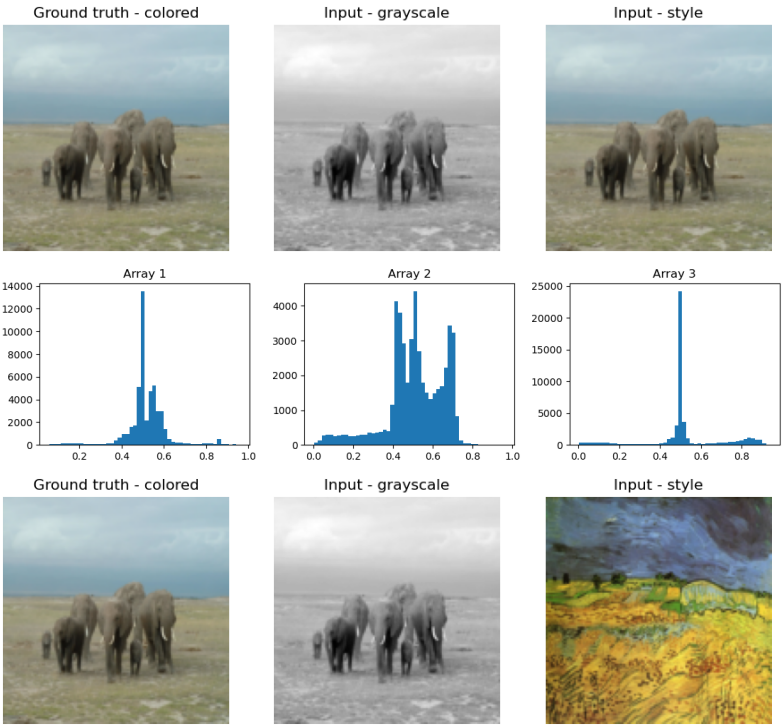
5.1.2 The LAB colorspace

We spent considerable effort trying to make our model compatible with both LAB and RGB colorspaces. The former should have been better suited for our task in theory. Here are some reasons why:

- The input grayscale image has 1 channel, similar to RGB.
- The output of the baseline UNet only affects the a, b channels, while L must not be learned but just concatenated.
- The AdaIN encoder should just take a, b channels as input, resulting in a more informative latent space.

However, in practice, we didn't record any improvement over the RGB counterpart. The colorization was more stable but also gray-ish, and we lost the ability to perform color transfer. Below is the sibling training of the RGB-Latent32, which we presented at the end of Section **??**.

In this particular case, we analyzed the histograms of the values of the latent space of the AdaIN encoder for different artistic images and noticed values clustering around 0.5. This suggests that the poor performance may be due to a lack of scarcity. The network is otherwise the same as its RGB counterpart but is more efficient in the LAB framework. Although we aimed to improve upon this with further experiments, constraints on time and resources led us to focus on perfecting the RGB network, which initially gave positive feedback on color transfer.



References

[1] Dataset on huggingface: huggan/wikiart. https://huggingface.co/datasets/huggan/wikiart. 4

[2] Dataset on huggingface: imagenet-1k. https://huggingface.co/datasets/imagenet-1k. 4

[3] Coloma Ballester, Aurélie Bugeau, Hernan Carrillo, Michaël Clément, Rémi Giraud, Lara Raad, and Patricia Vitoria. Analysis of different losses for deep learning image colorization, 2022. 2

[4] Zhiyuan Hu, Jia Jia, Bei Liu, Yaohua Bu, and Jianlong Fu. Aesthetic-aware image style transfer. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 3320–3329, New York, NY, USA, 2020. Association for Computing Machinery. 2

[5] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017. 1

[6] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019. 1, 2, 4

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 2

[8] Muhammad Hisyam Zayd, Novanto Yudistira, and Randy Cahya Wihandika. Image colorization using u-net with skip connections and fusion layer on landscape images, 2022. 2, 4

[9] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 3, 4

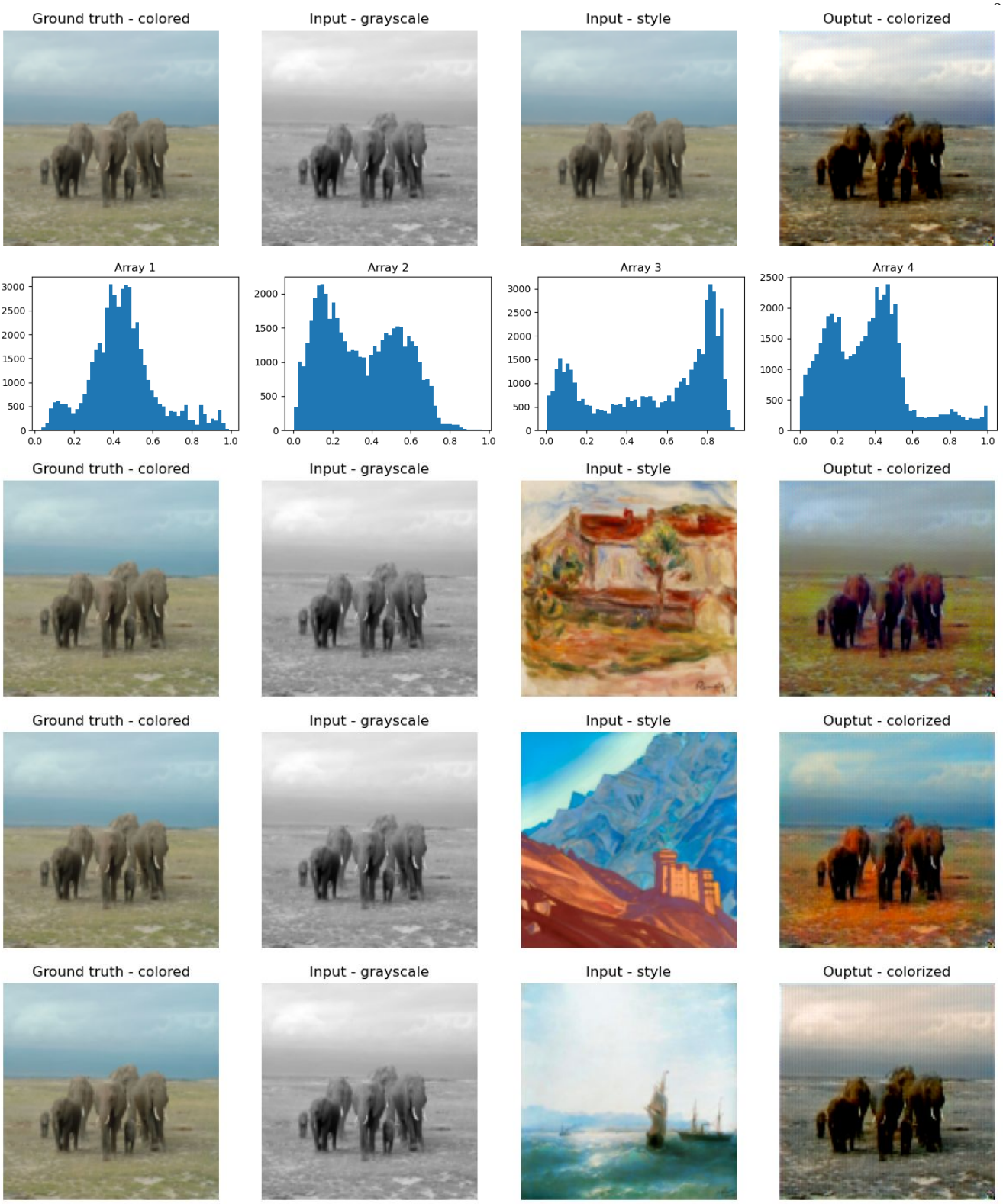


Figure 9: Some results. It's possible to see that: 1) the model is able to colourize a gray-scale image, despite the quality not being perfect; 2) Our style-transfer feature works correctly. Different paintings passed to the AdaIN encoder influence significantly and coherently the output image.

Figure 10: Exotic Bird

Figure 11: Small dog in grass

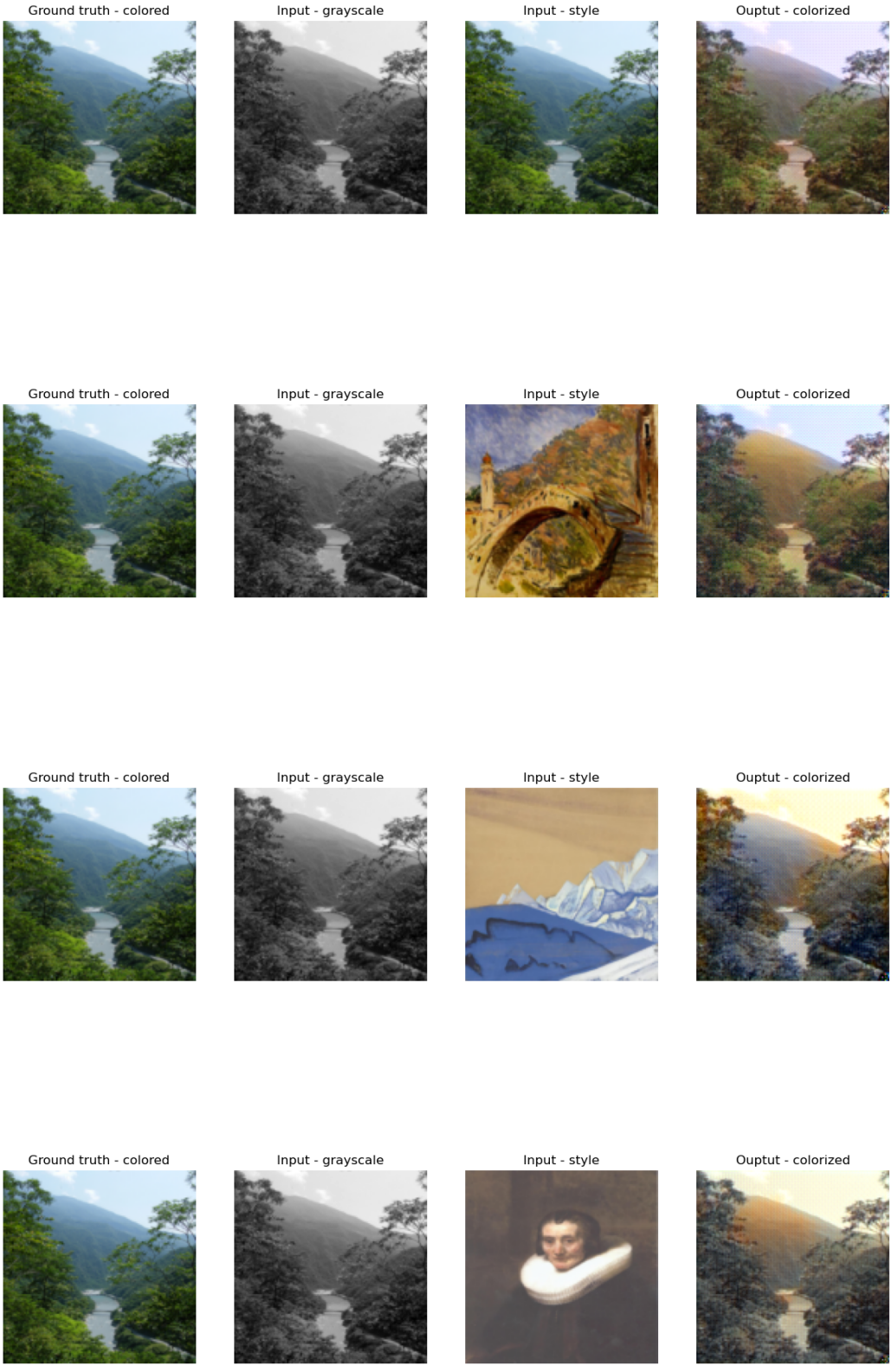


Figure 12: Landscape of forest

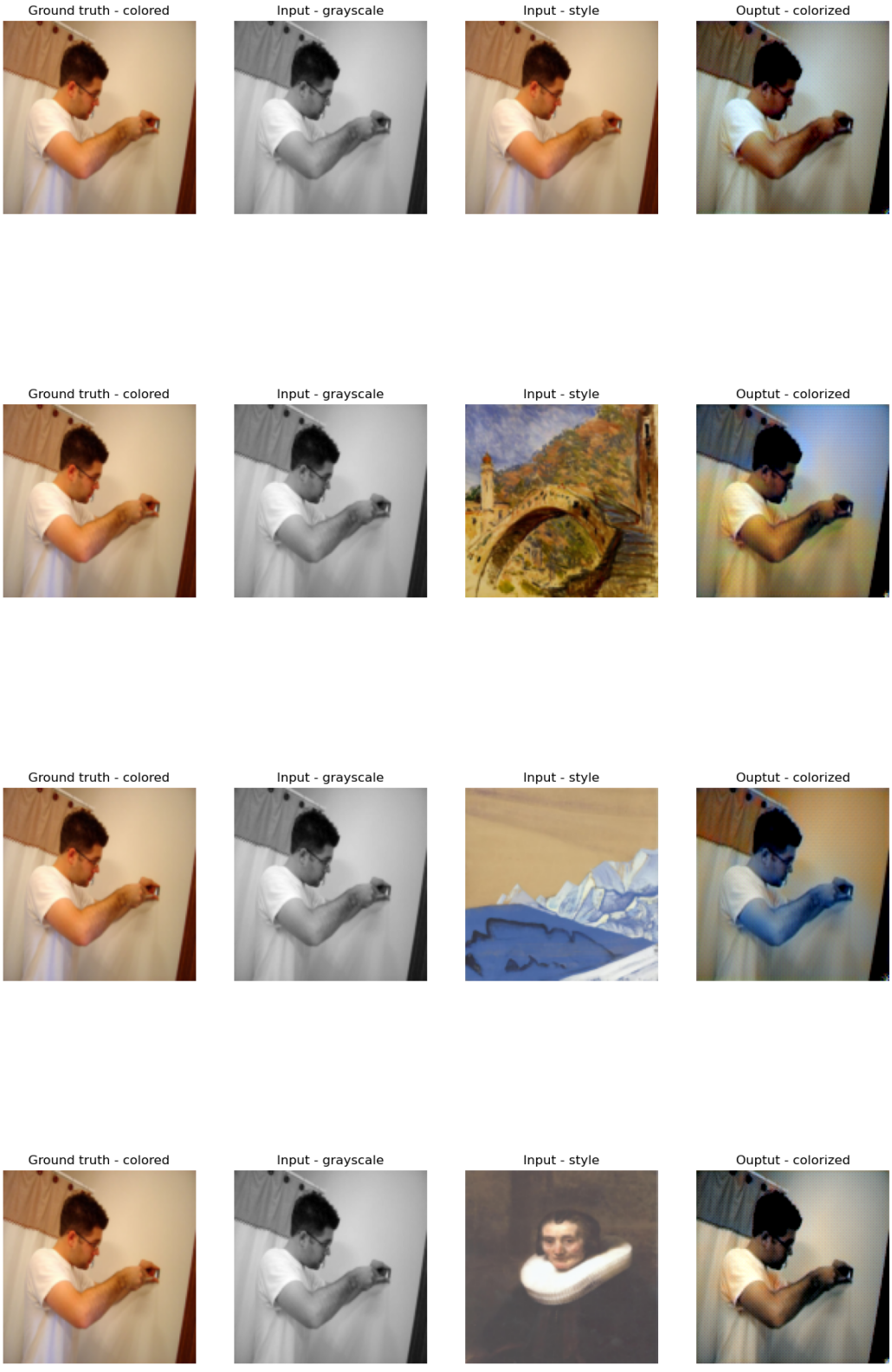


Figure 13: Person