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 the colourization of gray-scale images and color style transfer. We propose an architecture consisting of two parts: a style encoder based on a VGG19 and a UNet, tasked with colourization and enhanced by Adaptive Instance Normalization layers. Both the weights of the AdaIN layers and of the the style encoder, on which they rely upon, are trained implicitly. Moreover, we explore the influence of the chosen colorization loss (MSE vs LPIPS), the chosen colourspace (RGB vs LAB) and conduct an ablation study on different configurations of the network. Finally we show examples of colourization and color transfer from some paintings.

1. Introduction

Colorization of gray-scale 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 colourize gray-scale images in various artistic styles using a U-Net architecture enhanced with Adaptive Instance Normalization (AdaIN) layers [6]. 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 [1] from reference images to gray-scale inputs. AdaIN enables the decoupling of content and style in feature representations, allowing us to leverage the content information from gray-scale images while infusing them with the stylistic characteristics extracted from reference coloured images. This style-guided colorization approach opens up new possibilities for artistic expression, allowing users to apply various painting styles, from impressionism to surrealism, onto gray-scale inputs.

1.1. Motivation

This project has his roots in our fascination for the colourisation task: an underdetermined problem that is still under research and has big room for improvements. It is mathematically impossible to deduce from the single channel of gray-scale images the three 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. Concerning Adaptive Instance Normalization, we were interested to experiment with it even just for its regularization properties, that proved later to be a promising start for style transfer.

1.2. Related work

Regarding basic colourization with *UNet* [7], our reference was a paper titled *Image Colorization using U-Net with*

[1] We use style to refer to color choices

Skip Connections and Fusion Layer on Landscape Images [8].

We analyzed *Analysis of Different Losses for Deep Learning Image Colorization* [3] and concluded that a more complicated colorization loss does not lead to significantly better results.

A much more expansive research was done to learn and bend to our needs the concept of Adaptive Instance Normalization. Most of the related work on this topic deals with its non-parametric version, as introduced in *Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization* [5]. Other publications work instead with Generative Adversarial Networks, as they are more expressive and better suited for this task, but also much harder to implement and expensive to train. Our final work inherits most of its concepts from [5], but with the big difference that the parameters for AdaIN are learned rather then computed with a fixed formula. A backup for this approach, although in a different context, was found in the *Style GAN* paper [6].

Our final model only transfer colour styles, but we also found literature on how to transfer texture (see *Aesthetic-Aware Image Style Transfer* [4]). We believe this could be a future development of the project and that it would be a meaningful improvement. For example, when passing a painting in big brush strokes as style reference, also the output could resemble those thicker stokes.

2. Methods

2.1. The architecture

The baseline UNet

We invite the reader to reference our drawing of the architecture in fig. 1 while reading this paragraph.

The central component of our architecture is the UNet, which is nowadays still 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.

Train dataset size	2000
Validation dataset size	200
Batch size	16
Epochs trained	231
Colorspace	RGB
Optimizer	Adam w. Weight decay

Table 1: Details on UNet model for colourization

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 together.

The network expects input in the form of $[1, H, W]$ and produces outputs in the form of $[C, H, W]$, where the number of channels C can be 2 for the LAB colourspace and 3 for the RGB colourspace. We experimented training with both colour spaces and we made sure that everything was compatible with both. For results in the LAB colourspace see the Appendix A.

This architecture forms the backbone of our neural network, providing a robust and efficient structure for the image colorization task. This autoencoder 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 colourisation. Since this was but a halfway check, we didn't spend energies neither on training over big datasets nor on countering overfitting.

It's worth anticipating instead that in the AdaIN combined version of the model, we introduce dropout layers after every block of convolutions, in order to delay the growth of overfitting during training and to enhance the generalization properties of the network.

In fig.2 is a proof of the UNet capabilities to colourize images, in configuration detailed in table 1. As mentioned before, no care was taken to counter overfitting. Fig.3 shows the outputs of the UNet for one picture from the training and one from the validation datasets respectively.

Adaptive Instance Normalization and Style Encoder

In order to achieve style transfer (only colour) we augment the UNet with two additional components: a style encoder which encodes the color features of the original image [2] and AdaIN layers which align the style feature maps with the content image feature maps.

[2]We will later discuss why only the color features and not also the texture features of the original image are encoded

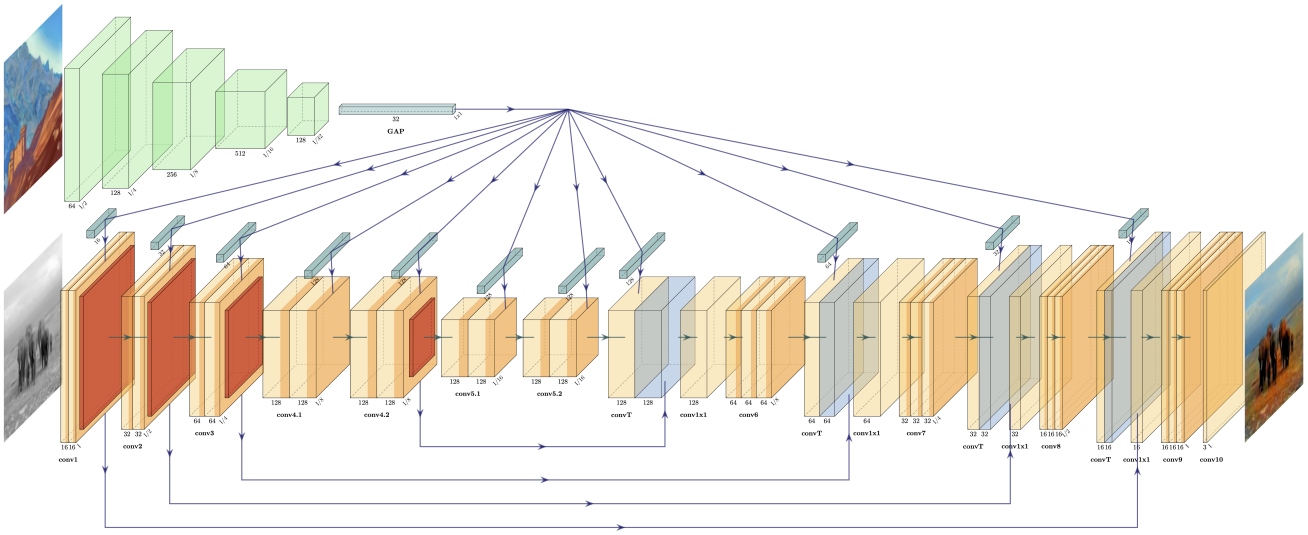


Figure 1: Map of our architecture

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

We implement a version of Adaptive Instance Normalization as in the *Style GAN* paper [6], given by [3]:

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

This layers aim to align in every channel the mean and variance of the feature space of the encoded style image with those of feature maps of the original image. This is repeated at different depths in the network using learned affine transformations (MLPs in our network). When you pass to the AdaIN encoder the colour image itself, this acts as regularizer and the network learns to pull information on

[3] see paper [6] for full description of the terms

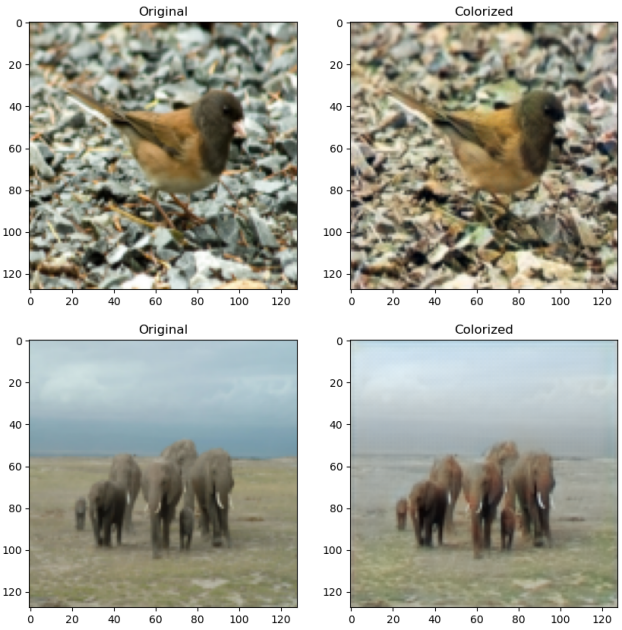


Figure 3: Example from the training dataset (top) and validation dataset (bottom)

the colour distribution from it. When you pass a different style image at inference phase, the network is used to rely on this information, resulting in a style (colour) transfer on the recreated image.

The AdaIN encoder runs only once 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 standard deviation 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.

The style encoder is a simple encoder followed by a Global Average Pooling (GAP) layer, which encodes the colour features in a latent space, which in our final model has dimension 32. We experimented with different dimensions and we discuss later about its effects on the colourization and style transfer capabilities of the model.

2.2. Training & Dataset

Since we didn't have access to departimental or external computing resources, we executed all the code and training on a personal laptop with NVIDIA GeForce RTX 3050TI, 4096MiB VRAM and on a M1 MacBook Air (16Gb RAM). This limited our ability in terms of resolution of the input images. Training with patches of size 256×256 forced us to use very small batch sizes, took a long time even for few epochs, but using patches of size 128×128 allowed us to run more experiments and train for loger periods of time and on bigger datasets, thus obtaining models that yielded good results and with good generalization properties.

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 reproducing the colors of the original.

After some balancing, we came up with a weighted sum of both losses for our training.

Implicit training

With *implicit training* we mean that no specific loss function is defined on the AdaIN branch of the network, but rather only on the baseline UNet. Training is always done by passing as style reference the ground truths of the images themselves. Thanks to the chain rule of derivatives and the mechanism of auto differentiation, the gradient gets backpropagated also through the AdaIN encoder and thus it learns to produce a meaningful latent representation of the colourspace, that the UNet can make a use for. The choice of using learned perceptual similarity (LPIPS) rather than pure MSE helps in this regards, 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.

We stress the point that there is no training against different style images. This approach was actually tried and will be discussed further ahead in sec. 3.2. The pillar of the success of the implicit training's technique is the choice to intentionally provoke scarcity in the UNet, so that it relies on the AdaIN layers, and thus on the colour representation of the style image as much as possible.

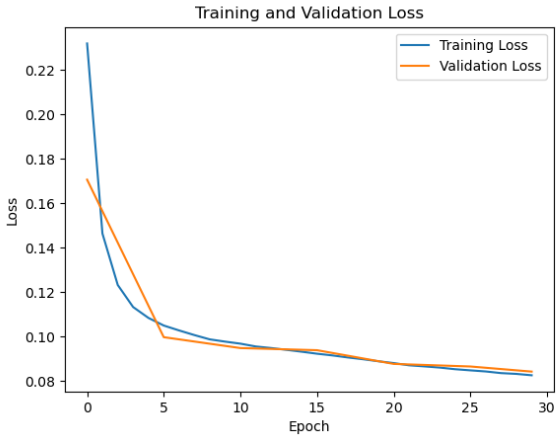


Figure 4: Loss curves of the final model. All details about the model in section 3.3.

Dataset and training analysis

We use the *imagenet-1k* dataset [2] for our training. We experiment with different training set sizes and for up to 50.000 images for the last bit of training on the final model, coupled with 500 more for validation purposes. Figure 4 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 the details on the training of the final model to section 3.3.

As mentioned in sec. 2.2 we don't need to feed any external style image to the network at training phase. On the other hand, at testing phase, we sample from the *wikiart* dataset [1] to check the colour transfer capabilities of the network. Indeed, loss values are not the only criteria of evaluation, human perception is the definitive judge. This came to play in the choice of the best checkpoint for the final model: we had also further trained the model on 100.000 images, and the colourization loss had decreased slightly, but the model had become more stiff and the colour transfer was less vivid. This lead us to discard this checkpoint in favour of the aforementioned one, trained on 50.000 images.

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 our goal. Many more experiments were done, like those on the LAB color-space, and they can be found in the Appendix A.

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.

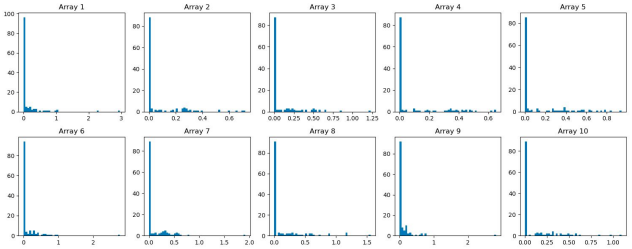


Figure 5: Histogram of the values in the latent space of the style encoder for random style images. It can be noticed how the values are all distributed very close to zero.

3.1. First steps and troubles

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 pre-trained 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, at least not while wanting to load the pre-trained weights.

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.

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 [5], 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. AdaIn layers are marked in blue in the big architecture drawing in fig. 1. The idea of learning AdaIN parameters takes inspiration from NVIDIA's 2019's *Style GAN* paper [6].

Investigation on the latent space of the style encoder

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

In these first results, 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, indicating that the network was suppressing its influence in favor of straightforward reconstruction. This is evident if we look at the values of the latent space of the style encoder after some epochs: in fig. 5 we feed some random style images to the style encoder (checkpoint at 32 epochs) and we plot the histogram of the values in the latent space. It can be noticed how the values are all distributed very close to zero. As a consequence, at inference phase, when different style images were used, there was no noticeable effect on the colorization of the content image, proving that the style information was not contributing to the reconstructed image.

3.2. 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 was pretty intuitive. The backward pass instead needed more careful thinking. Our first idea was to introduce a new loss, 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 the MSE between the two latent spaces,

The final loss was a weighted linear combination of the two losses, specifically:

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

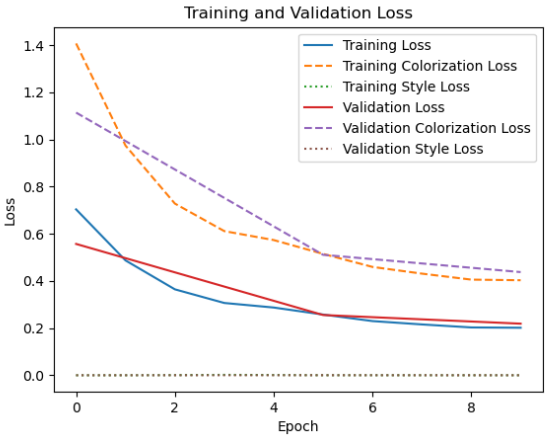


Figure 6: Failed attempt with combined loss approach.

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} (see fig. 6). 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.

Solution 2: using two different optimizers

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

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

We investigated once again the values of the latent space of the AdaIN encoder, and we realised that they were cast to zero by the backpropagation of the colorization optimizer, cutting out all its power to represent the style input.

The dual-opt approach was not viable in the brutal state we described and tested and was not promising enough to justify more investment of resources. The takeaway from this experiments was to learn how the training proceeded,

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 (the final model) for colourization and style transfer.

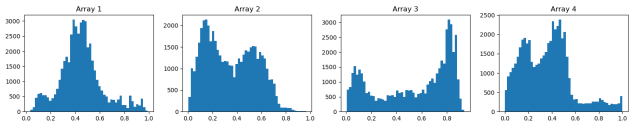


Figure 7: Histogram of the values in the latent space of the style encoder for random style images. This distributions cannot be directly interpreted, but their high variance guarantees that the encoder manages to produce a meaningful representation of the colour style.

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, to let it learn the true colour semantics of the style image, without forcing the network externally with different couples of content and style-images.

3.3. 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 zero. We rewrote the UNet to be much lighter, by reducing the number of channels to $1/4$ of the original size throughout the whole network. Naturally, we adapted skip connections, fusion and AdaIN layers 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. [4]

After introducing dropout layers and training for 32 epochs on 50k training images we obtain clear colorization and style transfer. Results for a test image against multiple style inputs are shown in fig. 8, more examples can be found in the appendix B.

The latent space of the style encoder is no longer null and, as fig. 7 clearly shows, we can see different distribu-

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

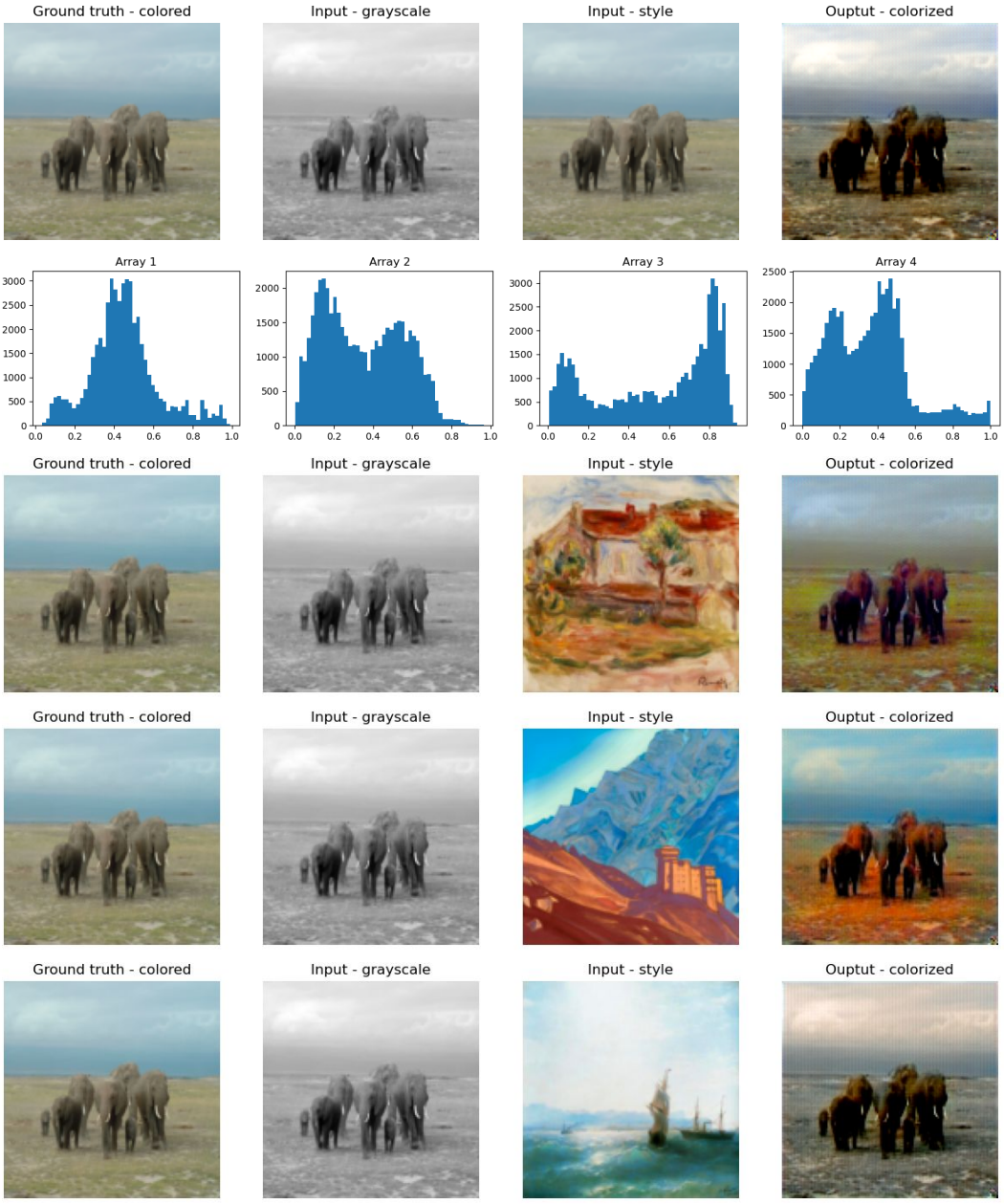


Figure 8: Here is an example for a random test image against multiple random paintings. The first row shows the colourization obtained by passing the ground truth image to the network, like in training phase. That is not a realistic scenario, since in practice the coloured image is not available. On the other hand, it provides a good baseline for comparison. 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.

tions 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.

4. Conclusion

In conclusion, we have presented a novel UNet architecture enhanced with Adaptive Instance Normalization

(AdaIN) 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), such that it 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. Finally, we investigated different normalization techniques and the influence of the used colourspace (RGB vs LAB). The results are supported by multifaceted analysis: study of the loss profiles, of distribution of values in the latent space of the style encoder and qualitative assessment of the model's colour transfer capabilities.

There is big room for further study and improvement. We highlight the necessity of upscaling the images and of training on larger datasets, two steps from which we were limited by our lack of computational resources. Another interesting study would be centered on investigating the ability of the model to transfer texture as well as colour from the style images.

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. 2, 5

[6] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019. 1, 2, 3, 5

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 1

[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, 5

[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. 4

A. Appendix

A.1. Further experiments

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

A.1.1 Using only LPIPS (Learned Perceptual Similarity) as the loss function

Utilizing LPIPS as the sole loss function in our experiments resulted in the emergence of unexpected artifacts, specifically manifesting in a distinctive grid-like pattern. The underlying cause of these patterns remains unclear, suggesting that further investigation into the behavior of LPIPS under these conditions is warranted.

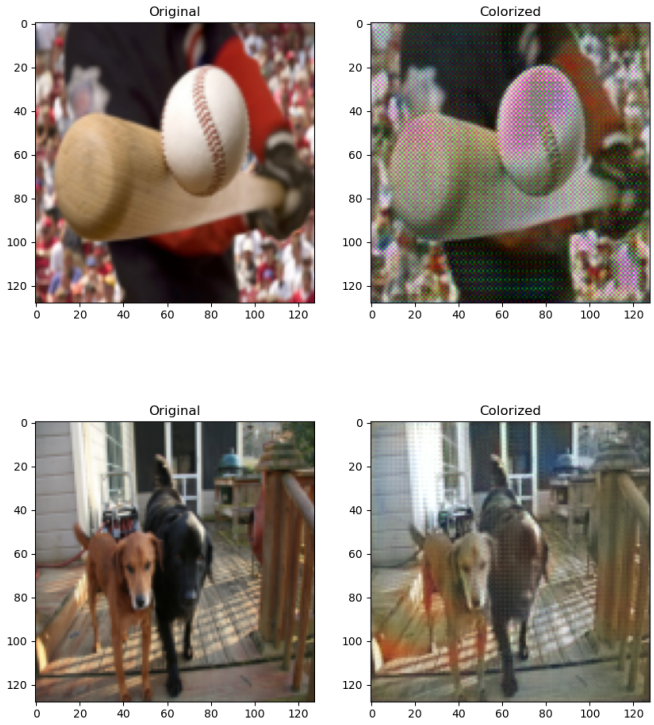


Figure 9: Reconstructed image presents grid-like artefacts when using the LPIPS loss.

A.1.2 The LAB colorspace

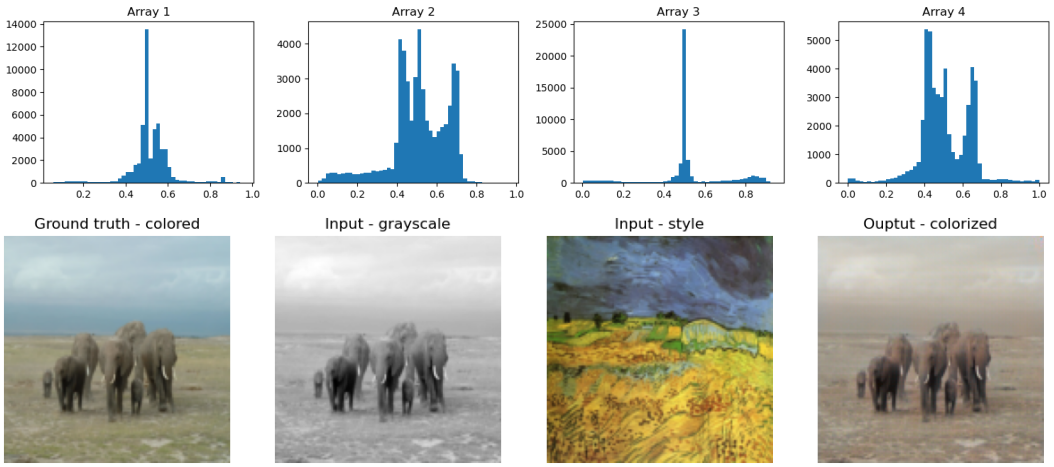
We spent considerable effort trying to make our model compatible with both LAB and RGB colorspaces. The former should have been better suited for our task in theory. Here are some reasons why:

- 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 LAB training sibling to the RGB-Latent32, which we presented at the end of sec. 3.3.

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



Also in this particular case, we analyzed the histograms of the values of the latent space of the style encoder for different artistic images and noticed that values clustered around 0.5. This suggests that the poor performance may be once again due to a lack of scarcity.

The network is otherwise the same as its RGB counterpart, but it is now more efficient in the LAB framework.

With more time at our disposal we would have liked to try more experiments in this framework, trying to induce more scarcity and obtain results analogous to those for RGB. However, we chose to concentrate and spend more energies in experimenting and polishing the RGB model, for which we had had a breakthrough earlier.

B. More results

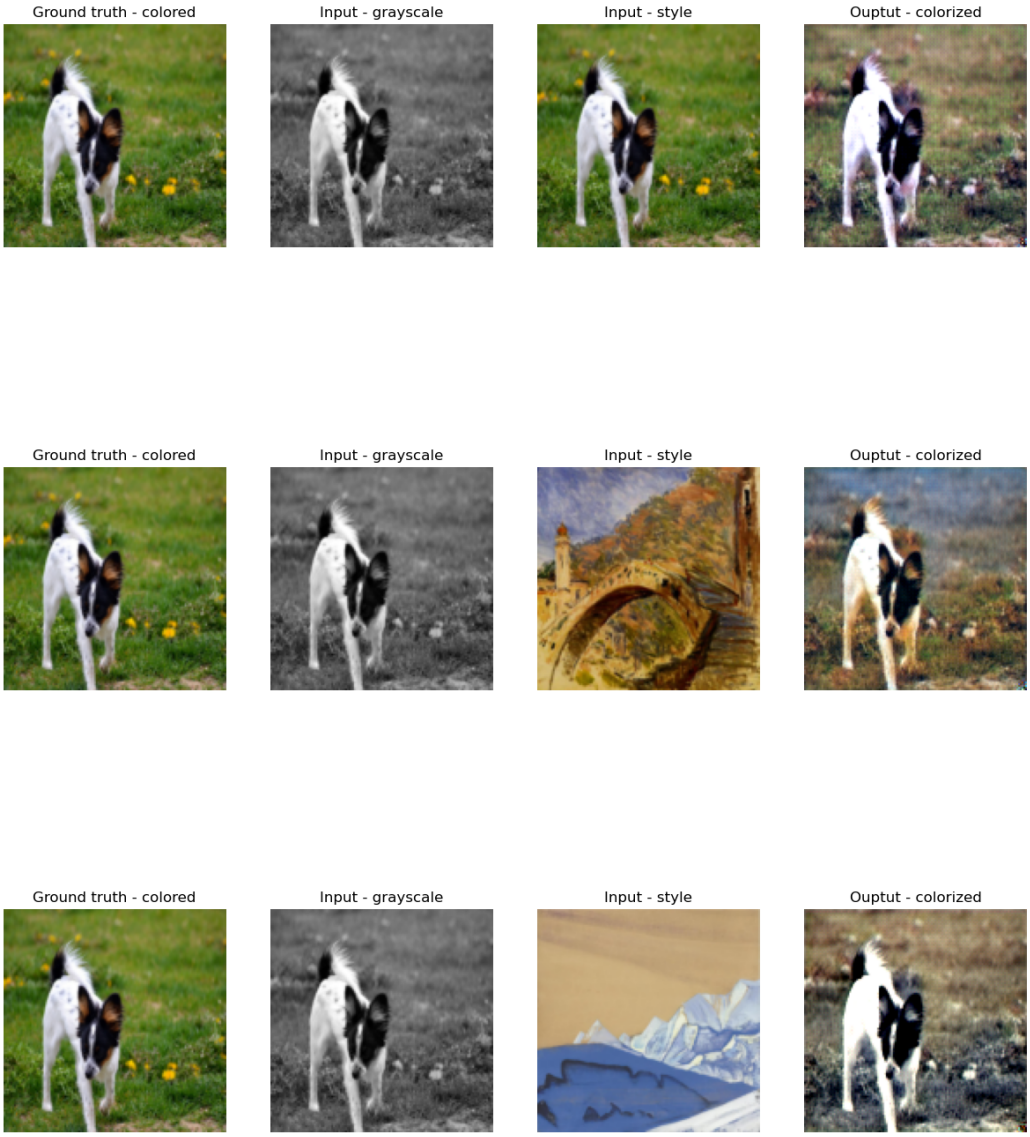


Figure 10: Small dog in grass

Figure 11: Exotic Bird

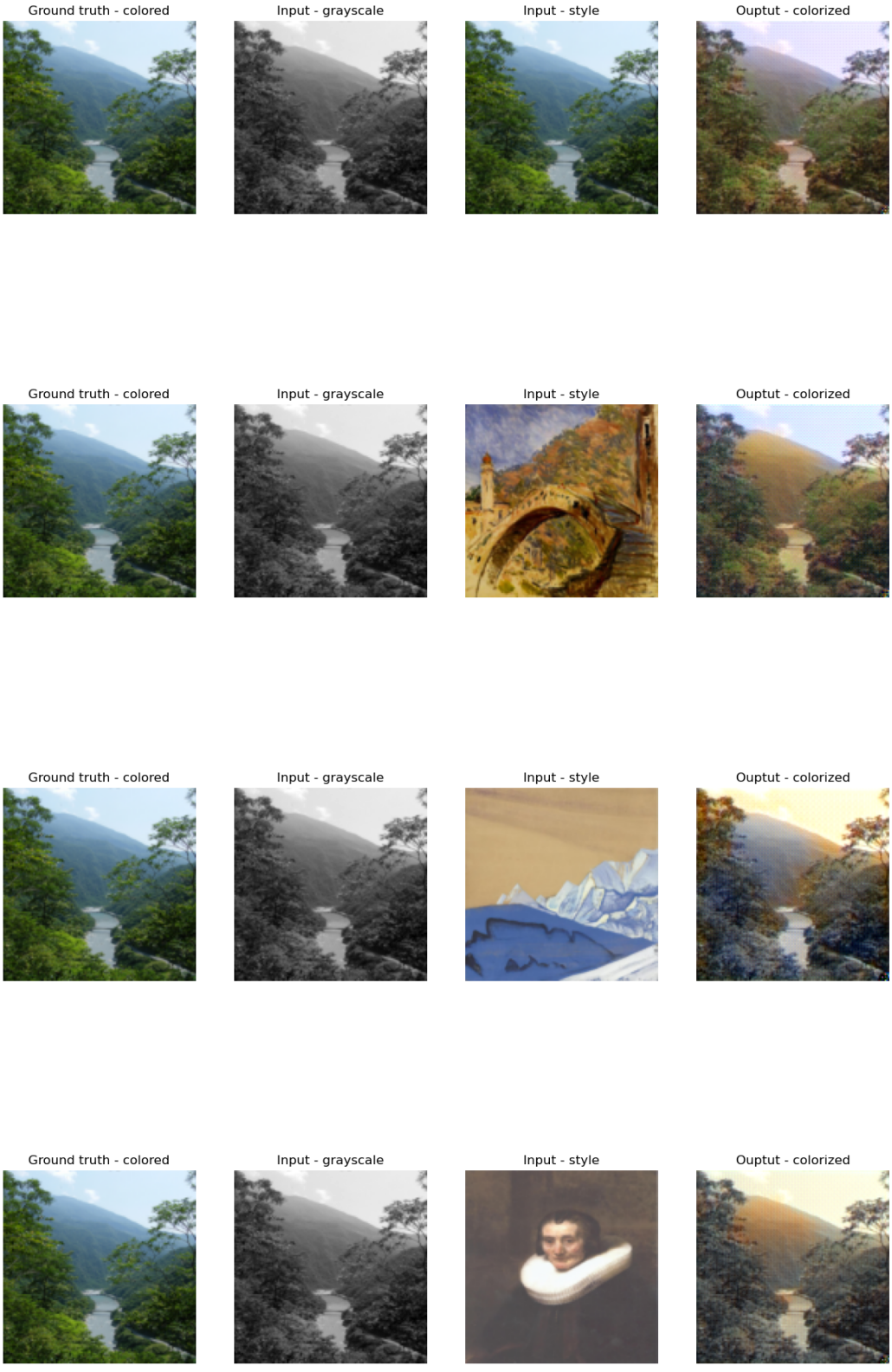


Figure 12: Landscape of forest

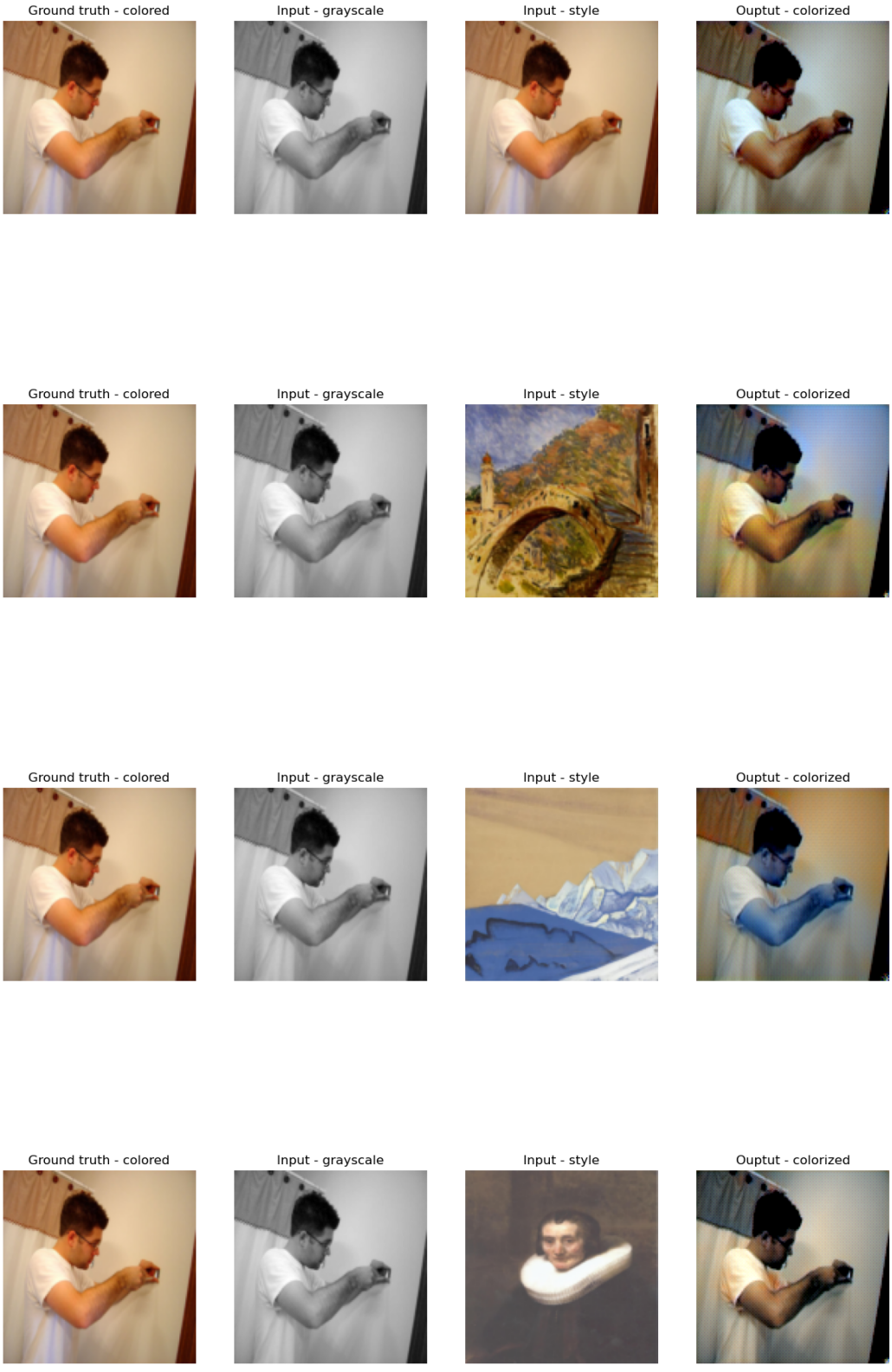


Figure 13: Person