Human Face Potrait Colorization Final Project 2025 Group 20

Oliver Sange
4753243
Msc. Physics

Elias Huber
4745649
Msc. Physics

Sam Rouppe van der Voort
4746512
Msc. Physics

Abstract

In this work, we develop and iteratively improve a computer vision model that colorizes grayscale portrait pictures of humans. For this, We use conditional Generative Adverserial Networks to translate from black and white to colorized image space. We then investigate how different generators perform within the conditional Generative Adversarial Network (cGAN) framework. We also study the effect of pretraining generator and discriminator, respectively, as well as the effect of adding an attention layer to the decoder of the U-net. The studied generators are a randomly initialized U-Net, an U-Net with a pre-trained Resnet18 as a backbone and a U-Net with the SqueezeNet 1.1 as a backbone. When combining all of the investigated techniques, we managed to improve scores on all three quantitative benchmarks significantly when compared to the baseline model, as well as visually improving the colorization to be more appealing.

1. Introduction

Colorization of grayscale images is a branch in computer vision that has sparked interest over the last decade for its wide-ranging applications [8]. In image restoration, colorization helps recover lost visual information in old, or black-and-white photographs, making them more visually appealing [16]. This is particularly valuable in preserving historical records, where colorized images can provide a more immersive view of the past. In entertainment and media, colorization can enhance classic films and archival footage, enabling modern audiences to experience historical moments with more realism [12].

In this work, we focus on colorizing portrait images of human faces, and try to optimize current techniques on this task. However, the methods we employ are versatile and can be applied to other domains as well. We hope that our work enables fast and accurate colorization of human portraits, as found in school or university yearbooks all around the world. Since these are often made to be budget-friendly and thus printed without colorization, our work would enable the users to colorize or recolorize their photos efficiently and accurately.

While traditional colorization methods rely on manual input or heuristic-based algorithms, they suffer from a lack of automation and adaptability. This leads to errors in color placement, unnatural results, and poor generalization across diverse images [16]. This led to the development of modern deep learning techniques that have recently revolutionized image colorization by automating the process with high accuracy and realism [16].

In particular, combining U-Nets and GANs is shown to be highly effective for image colorization, improving both the accuracy and quality of the generated images [7]. In this work, we develop, evaluate, and optimize our own model for the task of face colorization. For this we explore different network structures, training and pretraining methods, and also the addition of self-attention.

2. Related Work

Recent progress in image colorization has led to a variety of approaches that are able to colorize images accurately.

The most common approaches feature treating the image colorization task are divided into parametric and non-parametric methods.

Non-parametric methods begin by selecting one or more color reference images, either provided by a user or retrieved automatically, to guide the colorization process [15]. The method we focus on in our work, in contrast, relies on large datasets of color images to learn prediction models during a training phase. These models approach colorization either as a regression task, by predicting continuous color values, or as a classification problem, assigning discrete color labels from a quantized color space [8]. For the latter, Zhang et al. (2016) [16] reformulated the problem as a classification task. Their method predicts a probability distribution over a set of quantized color values for each pixel. To counteract the tendency of many approaches to produce desaturated outputs, they introduce class-rebalancing during training, which promotes the generation of more vivid colors. Their model demonstrates a significant improvement in fooling human observers compared to earlier techniques.

The work, our project mostly builds on, is the Pix2Pix framework for conditional image-to-image translation [7]. Here, Isola et al. use Generative Adverserial Networks to produce state of the art colorization results through conditional adversarial objective combined with an L1 reconstruction loss. The framework employs a U-Net architecture for the generator, which preserves spatial details through skip connections, and a PatchGAN discriminator that evaluates local image patches for realism. This dual-loss approach ensures both photorealistic outputs (via adversarial training) and pixel-level accuracy.

Mayali et. al. [9] then advanced this work by enhancing the GAN training by pretraining the generator for a general colorization objective on the CoCo dataset. We used their work and Git repository as a reference baseline model for our project.

3. Methods

The following sections describe the dataset used, the network architecture, and the training strategy.

3.1. Dataset

The models were trained on 4000 images consisting of 2500 faces of the ”1 Million Fake Faces” data set from Kaggle [14] and 1500 pictures of the face recognition dataset [2], which helps to decrease the bias of the fake face dataset by providing faces in different sizes, angles and sceneries in not central positions. The ”1 Million Fake Faces” data set contains portrait pictures of fake faces centered in the middle. The ”Face Recognition” dataset contains pictures of real humans in different scenes, sizes and angles. We then use 500 images of the ”1 Million Fake Faces” dataset for evaluation. We train and evaluate on an image resolution of 128×128. We chose this resolution due to computational constraints.

3.2. Network

The network structure used in this work is based on the Image-to-Image Translation with Conditional Adversarial Networks paper [7]. At the core of the approach is a Conditional GAN (cGAN), which consists of two neural networks—the generator and the discriminator. The network structure is visualized in figure 1. The generator receives a grayscale input image L and predicts the corresponding color information in the form of two additional channels: a and b from the LAB color space. The discriminator, on the other hand, takes the generated two-channel output, concatenates it with the original grayscale image, and assigns a score to the original image, as well as the generated image. The ”conditional” aspect of this GAN is derived from the fact that both the generator and discriminator receive the grayscale input image as a condition. Unlike traditional GANs, which sample from a random latent vector, we do not provide an explicit noise vector to the generator. Instead, randomness is introduced through dropout layers during training, which encourages diversity in the generated outputs.

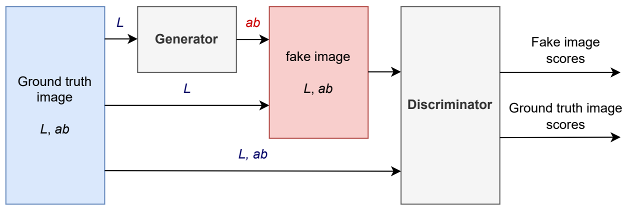


Figure 1. The network consists of a generator, which generates the colors ab based on the grayscale image L. The colors and the brightness are concatenated to the fake image. The discriminator assigns a score to both the fake and the ground truth image.

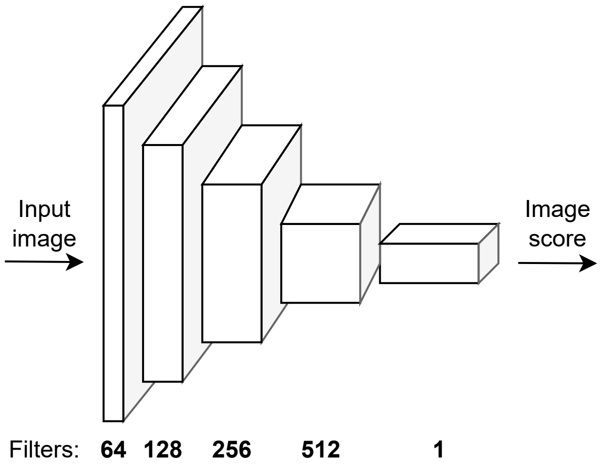


Figure 2. The discriminator has different blocks that consist of a convolution, a batch normalization and a LeakyReLU activation function. The number of filters increases over the first four blocks but is only 1 at the last block.

3.3. Discriminator structure

The discriminators structure is shown in figure 2. It is built in a straightforward manner by stacking several blocks that each include a convolution, a batch normalization, and a LeakyReLU activation. In the first block, the input image is processed by a convolutional layer without normalization. This initial layer extracts basic features from the image. As

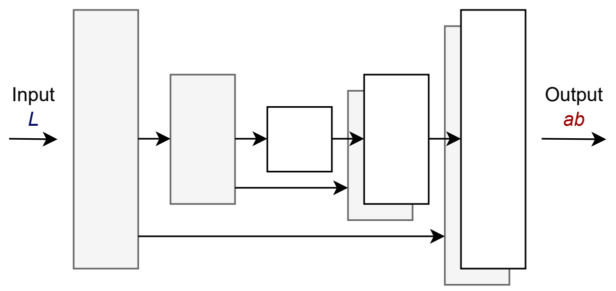


Figure 3. Structure of the U-Net based on source [7]. It features an encoder-decoder structure that downsamples the input to a bottleneck and then upsamples it back to the original resolution. Skip connections link corresponding layers.

the model deepens, subsequent blocks gradually increase the number of filters and use a stride of 2 to downsample the spatial dimensions, except for the last block in the loop which uses a stride of 1. In the final block, the model again uses a convolution but neither a batch normalization nor an activation function. The output is a score that describes how real the input image looks. This discriminator is referred to as a ”Patch” Discriminator. In contrast to a traditional discriminator, it puts out a grid of numbers. Each number in the output corresponds to a patch of the input image. By evaluating the authenticity of each patch independently, the model focuses on local details rather than making a blanket decision for the entire image. The discriminator learns to assign high scores to the ground truth image and low scores to the generated image.

3.4. Generator structure

In this work we implement different generator structures to enhance performance. We use an U-Net with randomly initialized weights, a U-Net with pretrained ResNet18 as base, and a U-Net with SqueezeNet1.1. as base, which are explained in the following sections 3.4.3 3.4.1 and 3.4.2 respectively. Our goal is to improve feature extraction and overall model performance, while maintaining low complexity and short training times.

3.4.1 U-Net

The U-Net is based on the generator proposed in source [7]. Its structure is displayed in figure 3. In the U-Nets architecture, input and output share a similar structure. For our image to image coloration task, this allows the model to retain the features and focus on the coloration task. It is an encoder-decoder network that gradually downsamples the input image until a bottleneck is reached and then upsamples it back to the original resolution. However, instead of forcing all information to pass solely through the bottleneck, the U-Net incorporates skip connections that directly pass low-level information from the encoder layers to the corresponding decoder layers [11]. This preserves crucial information feature information at different scales that helps produce more accurate outputs

3.4.2 U-Net with pretrained Resnet18

In our first modification to the baseline model U-Net, we implemented a U-Net generator that uses the pretrained ResNet18 as the backbone. ResNet-18 is a deep convolutional neural network that is part of the ResNet family, which was introduced by He et al. in 2015 [4]. It consists of 18 layers, including convolutional layers, batch normalization, ReLU activations, and residual connections. The architecture follows a pattern of initial convolution and max pooling layers, followed by four groups of residual blocks with increasing filter sizes, and ends with global average pooling and a fully connected layer for classification. We chose ResNet-18 because it shows strong performance in image classification and feature extraction.

3.4.3 U-Net with SqueezeNet1.1

For this, we implemented an U-Net with the SqueezeNet1.1 as a base. SqueezeNet1.1 is a lightweight deep neural network designed for efficient image classification [6]. It reduces the number of parameters while maintaining accuracy by using Fire modules, which replace standard convolutional layers with squeeze-convolution and expand-convolution layers that down- and upsample the input during inference. SqueezeNet1.1 further optimizes the architecture by reducing the kernel sizes in the initial convolutional layers and making the network faster with fewer parameters, making it well-suited for our setup.

3.4.4 Attention mechanism in the U-Net

We also added an Attention mechanism to the third last block of the U-Net decoder to improve the model performance. The Attention mechanism enables the model to capture the relationship between different regions more accurately. This is done by computing keys and queries for every input and matching them accordingly to produce an attention value that describes the relationship of inputs with each other.

3.5. Training

3.5.1 Loss

We use the same loss function as proposed by [7]. The cGAN loss,

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] \\ + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))], \quad (1)$$

which penalizes wrong discriminator decisions is combined with the loss for the pure generator. For this we use an L1 loss

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z} = [\|y - G(x, z)\|_1], \quad (2)$$

which penalizes inaccurate coloring when comparing to the original coloring of the image. As outlined by [7] the L_1 is chosen over L_2 because it encourages less blurring. This then results in the final loss,

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L_1}(G). \quad (3)$$

where the GAN loss and the L_1 loss can be balanced with the hyperparameter λ, controlling the relative contribution of each. This is done because adversarial loss pushes the generator to produce images that appear realistic to the discriminator, but it does not directly ensure that the predicted colors match the ground truth. The L_1 loss addresses this, which penalizes deviations between the predicted and actual color channels. However, relying solely on L_1 loss often results in desaturated or overly conservative colorizations, as the model tends to choose average colors, such as brown or gray.

3.5.2 Generator Pretraining

For pretraining the generator, we generate colorized images by feeding it grayscale images, and then evaluate them on the pure generator loss outlined in equation 2.

3.5.3 Discriminator Pretraining

We pretrain the patch discriminator by generating fake images using the pretrained generator, and using binary cross entropy for classification objective.

4. Experiments

4.1. Invsetigated Models

See table 1.

4.2. Training

We use a batch size of 16 and train for 20 epochs. The training process utilized the Adam optimizer with separate learning rates for the generator and discriminator. The learning rate for both the generator and the discriminator was set to 2×10^{-4} by default. The optimizer parameters also included $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Training was performed on a Google Colab instance with an NVIDIA T4 GPU. During each epoch, the training loop iterated through the dataset, optimizing both the generator and discriminator alternatively.

Acronym	Full Model Name
PUG	Pretrained U-Net Generator
GU	GAN-trained, pretrained U-Net
PRG	Pretrained U-ResNet Generator
GPRG	GAN-trained pretrained U-ResNet
GPRGD	GPRG with pretrained Discriminator
PSG	Pretrained U-SqueezeNet Generator
GPSG	GAN-trained, pretrained U-SqueezeNet
GPSGD	GPSG with pretrained Discriminator
PSAG	PSG with Attention
GPSAG	GPSG with Attention
GPSAGD	GPSGD with Attention

Table 1. Acronyms and corresponding full model names.

4.3. Evaluation

Evaluating colorization results is inherently subjective, as different colorizations can be perceived as plausible even if they vary significantly. To provide objective measurements, we used Peak Signal-to-Noise Ratio (PSNR) to assess the fidelity of the generated colors compared to the ground truth. Additionally, we employed histogram similarity to compare the overall color distributions and the Fréchet Inception Distance (FID) to quantify the realism of the generated images in relation to real images. These metrics offer a quantitative basis for evaluation and will be explained in detail in the following sections.

4.3.1 Peak Signal-to-Noise Ratio

Peak Signal-to-Noise Ratio (PSNR) is a widely used metric in image processing that quantifies the quality of a reconstructed image compared to a reference image [3]. It is defined as:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{(L_{\max})^2}{\text{MSE}} \right)$$

where $L_{\max}$ is the maximum possible pixel intensity and MSE is the Mean Squared Error between the two images. A higher PSNR value indicates that the reconstructed image is closer to the original.

4.3.2 Histogram similarity

Histogram similarity is a method used to assess the resemblance between two images based on their color distributions [13]. It involves computing color histograms, which represent the frequency of pixel intensities across different color channels. Images are converted to HSV color space, and histograms are calculated separately for hue, saturation, and value. These histograms are then normalized and concatenated into a single feature vector. Cosine similarity is applied to compare the histograms of the two images. A higher similarity score indicates that the images share a more similar color distribution.

4.3.3 Fréchet inception distance

Fréchet Inception Distance (FID) is a metric to evaluate the similarity between two sets of images, typically real and generated ones [5]. It measures the difference in feature distributions by modeling them as multivariate Gaussian distributions and computing the Fréchet distance between them. The features are extracted from an intermediate layer of a pre-trained Inception network, capturing high-level perceptual information. Mathematically, FID is given by:

$$\text{FID} = ||\mu_r - \mu_g||^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2})$$

where μ_r, Σ_r and μ_g, Σ_g are the means and covariances of the real and generated image feature distributions, respectively. In application, standardized neural networks are used to calculate these features from a picture distribution. Lower FID scores indicate higher similarity. We use the python library clean-fid to apply this measure.

5. Results

All numerical results for the models we trained are outlined in table 2.

5.1. Baseline model

The first result for colorization with a standard U-Net with randomly initialized weights can be seen in figure 4. It is obvious that even though the model manages to generate colors that are close to each other in color space, for a human these pictures do not look realistic nor visually appealing at all, due to the not consistent and not realistic coloring. The solution to this issue, proposed by [7], is to use Generative Adverserial training, introducing a Discriminator that evaluates generated pictures.

5.2. GAN-training the model

The results for our GAN-trained Baseline model are shown in figure 5. The generated pictures tend to look way more visually consistent and realistic when compared to the generator only training. This becomes clear when looking at figure 6. Still, the model fails to generate visually complex and colorful images.

Method	Hist	PSNR	FID
PUG	0.858	27.73	41.64
GU	0.868	26.36	38.28
PRG	0.847	27.84	35.95
GPRG	0.875	27.66	30.00
GPRGD	0.872	26.87	32.81
PSG	0.858	27.76	33.03
GPSG	0.884	26.55	30.86
GPSGD	**0.892**	27.59	28.27
PSAG	0.862	27.92	32.99
GPSAG	0.877	27.54	28.69
GPSAGD	0.884	**28.04**	**27.06**

Table 2. Performance comparison of different models, evaluated on 500 images of the fake faces dataset. The best model of each metric is shown in thick black coloring, the second to best in red and the third to best in pink coloring.

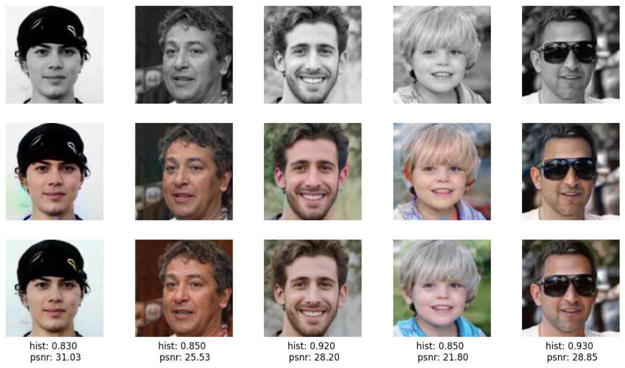


Figure 4. Face colorization examples of our just pretrained baseline model (PUG). The upper row shows the grayscale image, the second row the generated image and the lost row the original one. Corresponding metrics are printed below each column.

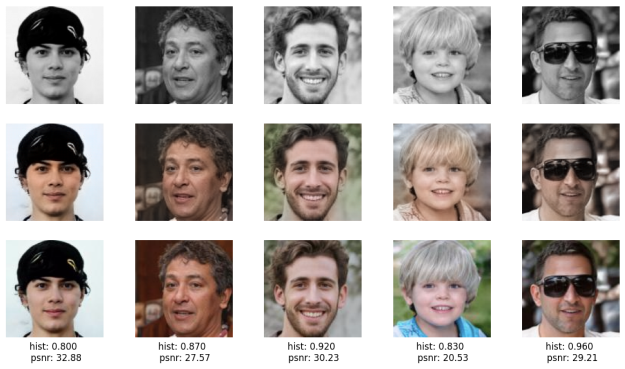
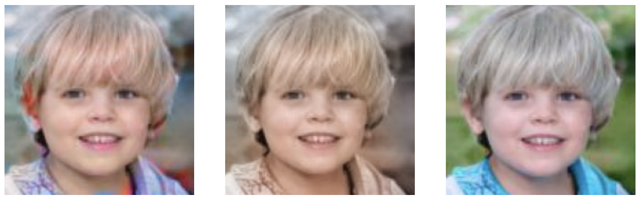


Figure 5. Face colorization examples of our GAN-trained baseline model (GU). The upper row shows the grayscale image, the second row the generated image and the lost row the original one. Corresponding metrics are printed below each column.



(a) Image colorized by PUG model (without GAN training) (b) Image colorized by GU model (with GAN training) (c) Original image

Figure 6. Example of common colorization issue of just pretrained models. Even though the psnr and histogram similarity are better for the left picture, it looks more unrealistic. The scores can be found in figure 4 and 5.

In general when looking at table 2 we do see that the models which have GAN training (those with names starting with G) have better scores, especialy histogram similarity and FID than purely generative trained models (those with name starting with P).

5.3. Enhancing performance with Transfer Learning

Our first approach to improve the implemented models to enhance visual perception in coloration, especially with little available compute, we deployed pretrained models as a backbone for our U-Net.

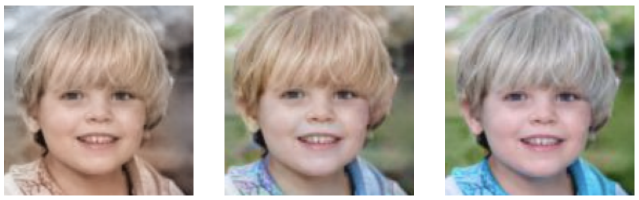
The pretrained models are trained for image classification on ImageNet. The models we are using are, as mentioned, ResNet18 and SqueezeNet1.1.
For both cases, we use them as the encoder in the U-Net architecture. The implementation is done With the DynamicUnet class from fastai. Here, the decoder copies the structure of the encoder. This way the pretrained weights are encoding the image features to the bottleneck layer. We then use our data to train the decoder for our purposes. In practice we also allow the encoder weights to train. We estimate that the pretrained weights are retaining useful information when encoding the images to the bottleneck layer such that we get better results with the same amount of training.

5.3.1 ResNet

Our first impulse to improve performance was to use a pretrained network that improves well on image recognition tasks. The most obvious pick was the ResNet18 Using the pretrained weights, pretraining the generator on our data and GAN training the models increased the Histogram similarity and psnr scores. The FID score also improved significantly, across the board, see GPRG and GPRGD in table 2.
From the table we also see that the performance of the pure generator model is worse than then the models that also are GAN trained.

5.3.2 SqueezeNet

Even though the benchmarks improved across the board when using the ResNet model as a backbone, we became suspicious that this model might be too big for our small dataset size and small resolution. As a result, we tried the SqueezeNet1.1 as a backbone instead, which is an smaller, more efficient model, trained on the same tasks and dataset as the ResNet18. Our results did not disappoint, we managed to further top our benchmarks (see GSPG and GPSGD table 2) and also achieved better visual results.
Figure 7 shows the improvement from using a pretrained SqueezenNet as a backbone model instead of using just randomly initialized weights. As we observed and there plot showed, the model outputs more diverse colors. This is also suggested by the computed FID score. This stems from the already pretrained understanding of objects the model has due to the SuqeezeNet.



(a) Image colorized by PUG model (with randomly initialized weights) (b) Image colorized by GPSGD model (with Squeezenet pretrained weights) (c) Original image

Figure 7. Example of common colorization issue of using models that are not pretrained. Even though the psnr and histogram similarity are fine, the model lacks variability in colors and is biased towards vrown and grey colors.

After training and evaluating the ResNet18, we noticed that the model was too big for our limited data and compute. We figured that it would be beneficial to use a smaller model that was also pretrained on image classification to get better results. Since we deal with low resolution images, we expected this to drastically improve our model.

5.3.3 Pretraining the Discriminator

To further enhance the performance of our model, we also tried to improve the performance of the model by pretraining not only the generator but also the discriminator. We propose that this would enhance performance since Generative Adverserial training is negatively impacted if either generator or discriminator are on a non-compatible training level. This idea was also implemented in the deOLDify model [1] where we took inspiration for this idea. We observe a good spike in performance across all three benchmarks when applying this technique on the squeezenet models when using this technique (see GPSGD and GPSAGD table 2). For the model with Resnet backbone the measures did not increase, this might be due to an unlucky instance where we have to little data for the size of the network. We write unlucky since with the GPRSG we did see an increase in performance.

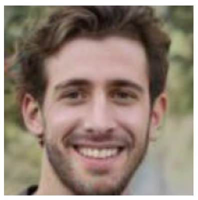
5.4. Adding Attention

One major flaw we then observed for the last models was that the models were performing poorly on regions that have multiple objects of different colors. The previous models failed to reconstruct the sharp edges of the original pictures and instead generated a smeared out, softer color. An example of a generated image like this, is shown in figure 8. To tackle this problem, we decided to implement an Attention mechanism in the decoder layer of the U-net to help the model learn the relationship between different regions. Since Neural Networks with attention are often used for image segmentation tasks, we figured this modification would naturally help with this problem. This idea was also recently studied in [10], where we took guidance.

We added the attention mechanism to all SqueezeNet-based U-nets. We then observed an increase in peak-to-signal ratio across the board for all SqueezeNet models. While the histogram similarity dropped just slightly, the Frechet interception distance decreased by a solid amount, creating our best performing model. (see GPSAG and GPSAGD table 2) The attention mechanism did what we expected, the images now show more color consistency alongside objects and also now have sharply colorized edges (see figure 8). An example of five face colorizations, generated with our best model, can be found in figure 9.

Finally, table 2 shows the performance of all implemented models across the three applied benchmarks. We also provide the evolution of the deployed metrics during GAN training in figures 12 and 13. An exemplary training and validation loss curve of GAN training is also provided in figure 10 and 11.

(a) Image colorized by GPSGD model (without Attention)


(b) Image colorized by GPSAGD model (with Attention)

(c) Original image

Figure 8. Example of common colorization issue of models that lack the attention mechanism. The generated images often lack color continuity within objects and also tend to smear edges. This is effect becomes apparent when observing the left ear of the shown person.

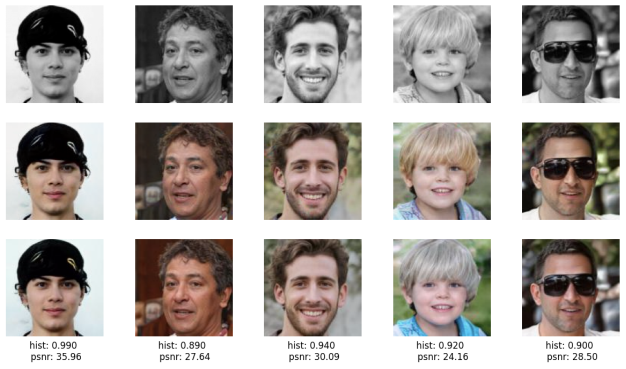


Figure 9. Face colorization examples of our best model (GPSAGD). The upper row shows the grayscale image, the second row the generated image and the lost row the original one. Corresponding metrics are printed below each column.

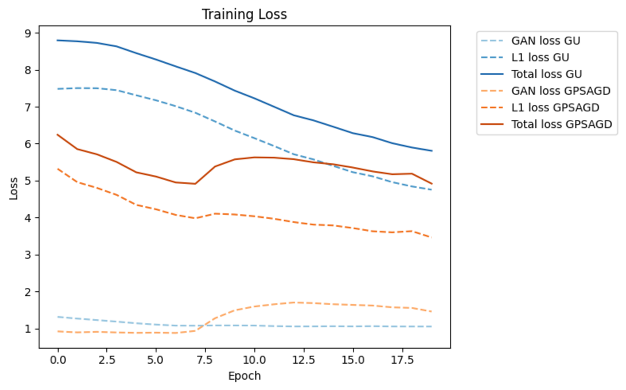


Figure 10. This is a place holder for a plot of the training loss

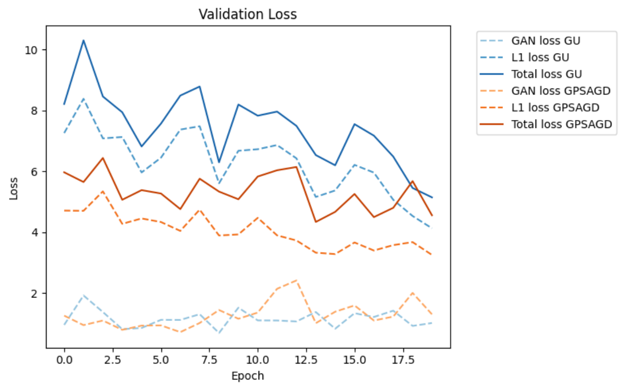


Figure 11. This is a place holder for a plot of the validation loss

5.5. Discussion of metrics

We want to note that the metrics we chose are, even though they are used in state of the art colorization papers, not the only reliable metric one has to evaluate our work with. We stress that colorization is an highly subjective and artistic task. So even though we managed to improve all of our quantitative benchmarks, it is also important to note

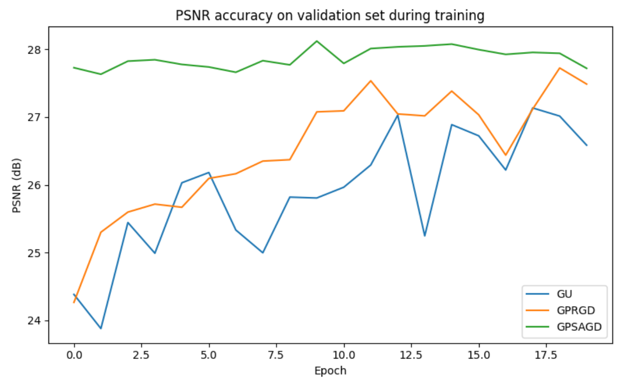


Figure 12. This plot shows the peak to signal noise ration of the GU, GPRGD, GPSAGD models during GAN training, when evaluated on the validation set. It is possible to see that our final model (GPSAGD) completely outperforms the previous models during GAN training, even though the model doesn't improve by a lot after pretraining.

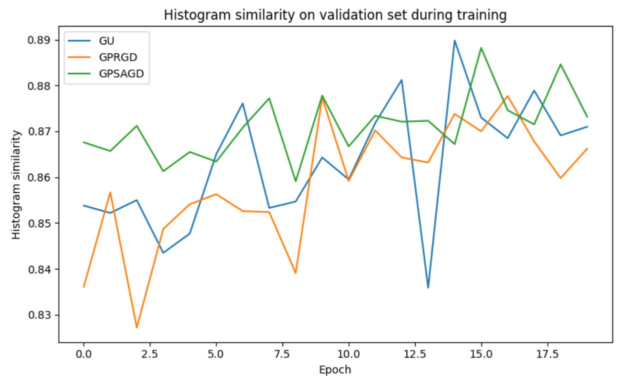


Figure 13. This plot shows the peak to signal noise ration of the GU, GPRGD, GPSAGD models during training, when evaluated on the validation set.

that we checked our results visually and tried to give strong examples of our improvement. A user study could quantify this further in future works.

6. Conclusion

In our work, we managed to implement and test a basic U-Net for face portrait image colorization. While the first results were rather basic, we managed to improve the model by using transfer learning, using GAN training and pretraining techniques as well as introducing an attention mechanism to our model. We did not only manage to improve our model on all three applied benchmarks, but also managed to yield far better visual results than the baseline model.

Further work could study these results with human user feedback and also diversify the applications of the colonization model by expanding the training dataset.

References

[1] Jason Antic. Deoldify: A deep learning based project for colorizing and restoring old images. https://github.com/jantic/DeOldify, 2018. Accessed: 2025-03-28. 6

[2] Fares Elmenshawii. Face detection dataset, 2023. Accessed: 2025-04-01. 2

[3] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Pearson International, 4th edition, 2017. 4

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 3

[5] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017. 5

[6] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016. 3

[7] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016. 1, 2, 3, 4, 5

[8] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23, 06 2004. 1

[9] Berkay M. Image colorization: Deep learning based image colorization. https://github.com/mberkay0/image-colorization, 2023. Accessed: 2025-03-21. 2

[10] Oliverio Nathanael. Color and attention for u: Modified multi attention u-net for a better image colorization. *Journal of Informatics and Visualization (JOIV)*, 8(3):1828, 2024. 7

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 3

[12] WIRED Staff. Ai magic makes century-old films look new, 2024. Accessed: 2025-03-26. 1

[13] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991. 5

[14] Bojan Tunguz. 1 million fake faces, 2019. Accessed: 2025-03-29. 2

[15] Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller. Transferring color to greyscale images. *ACM Trans. Graph.*, 21(3):277–280, July 2002. 1

[16] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. *CoRR*, abs/1603.08511, 2016. 1