UNIVERSITÄT OSNABRÜCK

Image Colorization - Methods and Applications

Hanna Merkle, Paula Vondrlik, Christoph Werries

Abstract
For the course ”Implementing ANNs with TensorFlow” offered in the winter semester 2021/2022 our study group decided to deep dive into the topic of image colorization. We decided to examine various papers dedicated to this topic and provide an overview on the state-of-the-art in this field. Based on this investigation, we have selected a single paper, according to our self-developed value-benefit-analysis, with the aim of guiding students and tutors closer to this subject. The decision fell on the paper: ”Image-to-Image Translation with Conditional Adversarial Networks” by Isola et al.. We analyzed this paper in depth and reimplemented some of the proposed structures with the methods and frameworks taught in our course. Finally, we provide different optimization techniques that we applied and conclude our results in this term paper.

Keywords
Image Colorization, cGAN, U-Net, PatchGAN

1. Introduction

The history of colorized photographs doesn’t begin in the 20th century as one would expect, but in the 19th. Already in 1839, after popularizing the photographic process of the daguerreotype, the Swiss painter and printmaker Johann Baptiste Isenring began to color photographs by hand. [1] As colorization was done manually, it was a tedious and time-consuming task. Hand-colored pictures were expensive and not very popular at that time, even ill-reputed as unauthentic. Nevertheless, science was advancing, and new methods were developed. The physicist James Clerk Maxwell was the first scientist to present a method for the non-manual and durable creation of color images. In a lecture held on May 17, 1861, he presented the additive synthesis of the three primary colors blue, green and red, displayed in figure 1.

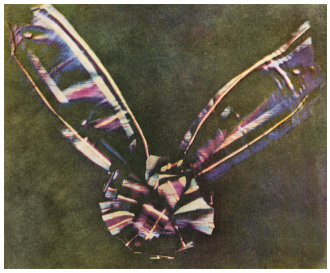

Figure 1. First ever public presented coloured picture generated through additive synthesis by James Clerk Maxwell [2]

Thus, he established the foundation of modern photography as it is known today. [3] Over time, machine and deep learning methods have been proposed and developed to automate this expensive task, making use of neural networks. Today, the process of automated image colorization has become essential in many different fields. Due to photographic limitations from the past and present, many images and videos exist in monochrome or unrealistic colorization. This includes old pictures and video recordings, electron microscopic images, infrared images, radar images, and many more. All these images can be made more accessible and vivid through image colorization, and also manipulated to reduce differences between images for further processing. Image colorization is therefore important not only in the creative field, but also in science and many of our daily life applications. Therefore, the participants of our study group, including Hanna Merkle, Paula Vondrlik and Christoph Werries, decided to base our term paper on this topic. As mentioned earlier, there were already several approaches developed for this problem. Therefore, we decided not to reinvent the wheel, but to investigate the state-of-the-art of different approaches, compare the reported results and try to implement a promising approach for this problem that also corresponds to the skills taught in the lecture “Implementing ANNs with TensorFlow”.

This led us to the following central research question of this project:

> Which state-of-the-art method allows us to implement our own image colorization algorithm given our acquired knowledge in the course IANNWTF2021/2022?

The title of the lecture already delimits the available solution space with two restrictions: Firstly each solution should resort on artificial neural networks (major restriction), sec-

ondly our own implementation should be based on the framework TensorFlow (minor restriction).

Furthermore, we decided that we will try to choose an approach that provides a good balance between performance, required computational power and network complexity, as we would like to highlight a solution that could serve as a possible homework assignment in the upcoming semesters.

The paper will introduce different approaches to image colorization considering the first constraint in section 2. Afterwards, the project team will present the decision process in section 3, that leads to the implementation of one selected approach that we re-implemented for this paper. Finally, we will show the results and present our conclusion about our findings.

2. Related Work

In the following chapter, we would like to gather different approaches that are used for the problem of image colorization. This paper will focus on approaches that are based on artificial neural networks, thus on so-called "learning based techniques".

We found several papers about image colorization like the paper 3 with the title "Image colorization: A survey and dataset", which is up-to-date and provides a comprehensive survey to the state-of-the-art up until 2020.

Further, we would like to mention the papers [4] with the title "Overview of image colorization and its applications" and [5] with the title "An Overview of Grayscale Image Colorization Methods". Both papers summarize and categorize the topic in different approches. Nevertheless, the paper [6] seems to provide the broadest overview on the subject.

Therefore, we rely on this rich source for our findings on the state of the art and address only those parts that are of greater importance to the content of this paper and the methods we examine in detail. Furthermore, we will not elaborate on single papers in this section in detail, but only give a rough overview of the basic concepts as well as advantages and disadvantages mentioned in various sources.

The authors of [6] classified seven different classes of colorization networks based on structure, input, domain, and type of networks. Thus, they proposed a taxonomy for image colorization. In the following subsections, we will shortly introduce those classes and sum up findings as well as appraisals. For further details and clarifications, we suggest reading the original paper.

2.1 Class 1: Plain Networks

The class labelled "plain networks" comprises any naive approach with simple networks that stack dense or convolutional layers with no or only simple skip connections between them. They have a simplistic architecture and are easy to implement. Still, they are limited in performance and suffer from typical problems like vanishing gradients if they are implemented too deep.

2.2 Class 2: User-Guided Networks

The class of "user guided networks" includes each algorithm that requires user input in the preprocessing or in the runtime by defining color dots, lines, or even areas. Thus, the effort required to perform this method is the greatest drawback. Additionally, colors may not look natural as they are user selected and thus subjective. However, in some applications this could also be seen as an advantage, since the user has some control over the output. One selected example from [7] is displayed in 2

Figure 2. Typical picture of chilling cats colored by a user guided algorithm using color lines or dots [7]

2.3 Class 3: Domain-Specific Networks

The class "Domain-Specific Networks" comprises algorithms that process images captured with special cameras or deal with certain themes. Examples of such networks are the colorization of infrared images [8] or sketch image colorization [9]. The main advantage is, that domain knowledge could be incorporated and thus the algorithm performs better than a more general algorithm. However, this also results in a lack of generalization and thus limited application possibilities.

2.4 Class 4: Text-based Networks

All algorithms that use additional input, such as text, to color images belong to the class "Text-based networks". The advantage lies again in the possibility of additional control by the user over the output through written text. However, this text needs to be precise, accurate and available as training data. This property is rather rare in image datasets and requires a lot of effort to create. An interesting up-to-date network is given in [10]. An example to this class is displayed in 3.

Figure 3. Example of colorizing a ship from [10]

2.5 Class 5: Diverse Colorization Networks

The class titled "Diverse Colorization Networks" comprises each algorithm that is based on GANs or Autoencoder approaches. Additionally, the aim of algorithms in this class is to generate diverse colored outputs instead of restoring the

original color. One of the great advantages is, that the algorithm can be trained unsupervised. This property also entails that the output is completely dependent on the input, as there does not need to be any ground truth to the input image. One of the disadvantages of these networks is the previously mentioned feature of being able to recover the original color of an image. Furthermore, it is difficult to quantify the results. However, these arguments only hold if unsupervised learning is applied. What seems to be a general issue are similar local texture regions, as these algorithms are failing to distinguish such image sections.

2.6 Class 6: Multi-Path Networks

The class labeled "Multi-Path Networks" contains any architecture, that learns features on different, multiple and/or parallel paths and layers within the network. These networks can be very specific to a domain, but still be used in a general context. The problem of vanishing gradients that usually occurs in straightforward and deep neuronal networks is tackled by introducing for example skip connections. However, the architecture of the networks usually becomes comparatively complex and computationally expensive.

2.7 Class 7: Exemplar-based Colorization Networks

The class titled "Exemplar-based Colorization Networks" comprises algorithms that colorize grayscale images by utilizing colors of diverse and discontiguous sample images. The availability of pictures is a major benefit, as nowadays pictures are available in freely accessible datasets on the internet. Some drawbacks of these networks are the strong dependency on the input, the lack of similar colors in the examples and difficulties regarding unusual or artistic colors.

2.8 Summary

With the comprehensive taxation on the diverse approaches to image colorization, the state of the art should be well represented in our paper. Each paper and algorithm should fit into one of the seven classes. However, this is only the state of the art as of March 2022. In the future, it may be necessary to add another class to the ones mentioned above. Still, in our research process in the beginning of 2022 we could not find a paper that does not fall into one of the seven classes. The issue we encountered was rather the difficulty to assign some paper to one single class only. Several approaches can easily meet the requirements of two or even three classes. Thus, it would have been reasonable to draw stricter boundaries in between.

3. Design evaluation

After we presented the state of the art to image colorization in the previous section, we will explain which approach we use for our own implementation. To do so, we will justify our decision by a value benefit analysis.

3.1 The criteria and weights

In section 1 we defined that we would like to find a tradeoff between performance, required computational power to train a network in acceptable time intervals and network complexity. Thus, these three factors will be our main criteria. Additionally, a network architecture capable to solve different tasks would be a plus as well. We believe that performance for this implementation is less important, as we want to provide a demonstrator useful for formative learning. Due to this, we assigned the lowest weight of 20% to it. An important aspect for us is to create a solution that is comprehensible without being an expert in the domain. We would like to choose an implementation that is understandable for a student who starts working with TensorFlow in this course. In Addition, the algorithm should be trainable with reasonable computational power to which a student has access to. Therefore, we assigned 35% as a weight to (low) complexity and 25% for trainability. Last but not least, a flexible approach would be a plus, as it would enhance reusability. Hence, this factor gets an additional 20% .

3.2 The concepts

We decided to compare three up-to-date concepts from three different classes defined in [6] and evaluate them against the defined criteria. Our selection of papers is based on a extensive research, public awareness and our personal opinion of the validity concerning results and findings reported in the three following original papers.

3.2.1 Class 5: Learning Diverse Image Colorization [11]

The paper by the title "Learning Diverse Image Colorization" uses a variational autoencoder network to learn a low dimensional embedding of color fields. The goal of this approach is, to model the diversity intrinsic to the problem of colorization. With this in mind, it is possible to produce multiple colorization that display long-scale spatial co-ordination. Furthermore, blurry outputs are avoided through a specific loss term for the VAE. Additionally, uneven distributions of pixel colors are taken into account. A conditional model for the multi-modal distribution between grey-level image and the color field embeddings is implemented. Finally, those features are resulting in samples of multiple viable colorization for a single grey-scale image this algorithm is capable to produce. The authors also claim, that their method obtains better results than prior approaches that utilizes standard conditional variational autoencoder and conditional generative adversarial network.

3.2.2 Class 6: Let there be color [12]

The paper by the title "Let there be color" uses an algorithm that is based on convolutional neural networks to automatically colorize grayscale images. The classic network is improved, featuring a fusion layer that merges local information with global priors. Local information is filtered by utilizing small image patches, global priors are computed using the entire image. All mappings are then trained in an end-to-end

fashion. A great improvement to the conventional approach is the independence of the resolution from the input images.

3.2.3 Class 7: Image-to-Image Translation with Conditional Adversarial Networks [13]

The paper by the title "Image-to-Image Translation with Conditional Adversarial Networks" uses conditional adversarial networks (cGAN) to provide an algorithm that serves as general-purpose solution to image-to-image translation. Instead of only learning the mapping from input image to output image, this network also includes a functionality to learn the loss function of the mapping. A great advantage of this approach is, that the adopter doesn't need to hand craft a loss function for a specific functionality. Thus, this approach is very flexible in its applicability, as the results merely depends absolutely on the input to output mapping.

3.3 The evaluation

As we want to implement and deep dive into a single paper's content, we prepared the value benefit analysis displayed in table 1. Each concept is abbreviated by the corresponding class.

For the performance, we see the best result in the paper of "Class 6: Let there be color". Anyway, it also has the greatest complexity in its architecture and seems to need a lot of time for the training procedure. The complexity is strongly reduced in the paper "Class 5: Learning Diverse Image Colorization", as this approach is based on a comparably simple variational auto encoder architecture. Though, the flexibility and performance is reduced due to the dependence on the embedding in the VAE.

Value benefit analysis

Criteria	Weight	Class 5	Class 6	Class 7
Performance	20%	◐	●	◔
Complexity	35%	●	◴	◔
Trainability	25%	◔	◴	◔
Flexibility	20%	◐	◔	●
Value benefit	100%	74%	50%	80%

Table 1. Value benefit analysis of different concepts to image colorization approaches

Last but not least, the "Class 7: Image-to-Image Translation with Conditional Adversarial Networks" seems comparably easy to train than the approach with VAE, especially if an U-Net architecture is used in the colorization. The printed results also educe a better performance of class 7 compared to class 5. Finally, the biggest advantage should be the flexibility of the approach displayed in the paper [13], as the same network could be used for different tasks with the same architecture and objective, and simply train on different data.

In the end we decided to implement a conditional adversarial network as presented in paper [13], as it could provide a trainable and comprehensible network design, that is also flexible and could be used on different problems by only switching the training data. Nevertheless, the VAE in approach provided in [12] could also be worth a try for a future investigation, as it's not performing bad against our value benefit analysis either. Only the approach in class 6 falls off a bit, but this may be varying from a different point of view.

4. Implementation

As mentioned above, we decided to implement a conditional generative adversarial network (cGAN) based on the one proposed by Isola et al. [13]. In the following, we will briefly discuss the difference between the traditional generative adversarial network (GAN) and the cGAN we used for our implementation and explain our choice for the color space. Thereafter, we elaborate on our cGAN architecture and the hyperparameters. Finally, we describe the different techniques we applied to improve the performance.

4.1 GAN vs. cGAN

In general, a traditional GAN consists of a Generator and a Discriminator. The goal of the Generator is to generate a realistic output that closely resembles the real data. The discriminator decides whether the data is real or generated. The generator and the discriminator are trained simultaneously. During training, the generator tries to improve producing results that fool the discriminator, while the discriminator aims to optimize correct data classification [14]. In a traditional GAN, a random noise vector z is used as an input to the Generator [15]. However, in our image colorization task, we want to create realistic colorful images from given grayscale images. Therefore, implementing a cGAN is a more suitable approach [14]. In a cGAN, additionally to the noise vector z, a label y is given as an input to the Generator [15]. In our case, this label represents the grayscale image. This additional feature vector conditions the generator. The rough structure of a cGAN is depicted in figure 4.

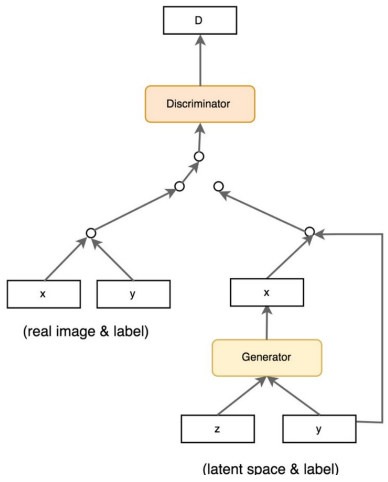

Figure 4. Architecture of a cGAN [16]

4.2 Colorspace RGB vs. CIELab

Before we look at the cGAN architecture in detail, we must decide which color space to use for the task.

An image is represented as a three-dimensional array. The first and second axis include its height and width, and the last axis the color information.

In RGB images, the third axis contains three color channels. The channels R, G and B represent the red, green, and blue color value for each pixel of an image. All three values add together to a united color and therefore to a realistic, colorful image. To generate the original colorful image from the grayscale image, the generator must predict three channels. The RGB channels are depicted in 5.

Another method is to transform the image into the CIE Lab color space. Like RGB, the Lab color space also consists of three channels, but with a different composition. The first channel is the L-channel, which is relevant for the lightness of the image. It only contains the contrast between darker and lighter areas. The L-channel includes values from 0 to 100 [17]. The a- and b-channel usually contain values ranging from -128 to +127 [17]. The color spectrum between green and red is part of the a-channel, and the spectrum between blue and yellow is in the b-channel [18]. To generate a colorful image, the generator receives the L-channel as an input and has to predict the a- and b-channel. The Lab channels are displayed in 5.

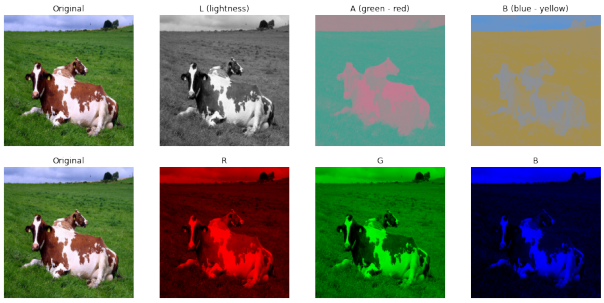

Figure 5. The LAB (first row) and RGB (second row) channels.

In many of the reviewed papers, the Lab color space is used instead of RGB. This choice becomes evident when looking at the numerous possibilities for different color values. For each color channel, we assume a choice of 256 values [19]. Therefore, in an RGB image we obtain a total number of 256*256*256 $\sim$ 16,800,000 possible colors for each pixel [18]. In contrast, using Lab the generator only needs to predict two channels and therefore there are just 256*256 $\sim$ 65,500 possible colors for each pixel [18]. This fact makes the use of the Lab color space the more reasonable choice, as the color prediction is less complex and less computationally expensive.

4.3 Neural network architecture

For the general structure of our model, we use a cGAN consisting of a Generator with a U-Net structure and a Discriminator with a 70x70 PatchGAN structure as proposed by Isola et al.[13]. In the paper they compare the performance of a U-Net structure with the normal encoder-decoder structure (Chapter 4.3) and also different Discriminator architectures (Chapter 4.4). They discovered the best performance using a U-Net structure for the Generator and a 70x70 Patch Discriminator. Due to this, we decided on a similar implementation. In the following, the architecture of the two models is described in detail.

4.3.1 Generator

The Generator receives the L channel with shape [256, 256, 1] as an input and generates the ab channel with shape [256, 256, 2] as an output. As mentioned before, the Generator is based on a U-Net structure 6. It has an encoder–decoder structure. In the encoder, the input is first downsampled until a bottleneck layer is reached. Then, the data is upsampled again until the desired output size is attained. The result is the generated ab channel. Important in a U-Net are the skip connections that concatenate the outputs of the downsampling process with the outputs of the upsampling process. The idea behind this is, that due to these skip connections, low-level information is preserved which would have been lost at the bottleneck layer.[13]

The encoder and decoder are each made up of convolutional blocks. From Isola et al. we adopted the following structures mentioned in the appendix of the paper [13]. Note that we changed the notation of the decoder structure, as according to our understanding it was formulated a little unclear. The structure itself remains the same.

Encoder: C64-C128-C256-C512-C512-C512-C512-C512
Decoder: CD512-CD512-CD512-C512-C256-C128-C64

In the notation, ”C” represents a block with the configuration *Convolutional layer-Batch Normalization-LeakyReLU* and ”CD” with *Convolutional layer-Batch Normalization-Dropout-ReLU*. The number following the letter describes the number of filters.

For the downsampling process in the encoder a strided convolution is applied. The decoder makes use of strided transposed convolutions for the upsampling. As proposed in the paper, batch normalization is not used in the first layer of the encoder as well as in the bottleneck layer. For the output layer, a convolution with filter size two and a hyperbolic tangent activation function is applied to obtain the ”correct number of output channels” [13].

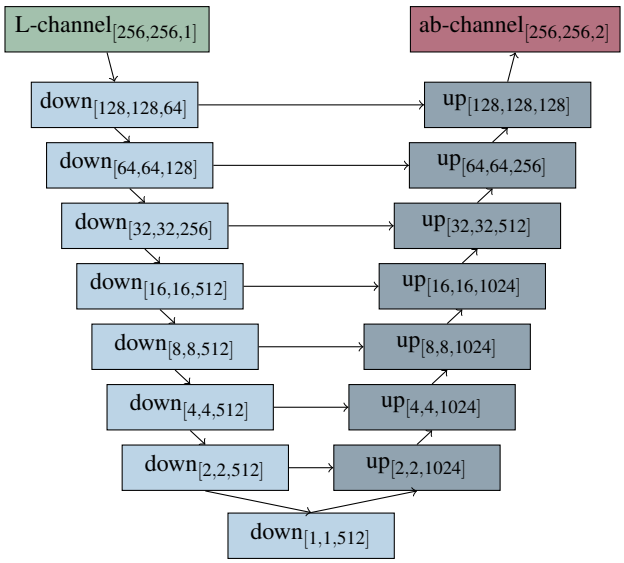

Figure 6. Our Generator architecture with the corresponding output shapes based on the U-Net proposed by Isola et al. [13]

4.3.2 Discriminator

The Discriminator receives the L channel and ab channel as an input. The three channels are concatenated, and the discriminator classifies the images as real or fake. For our Discriminator we designed a markovian architecture (PatchGAN). A traditional convolutional Discriminator classifies an image with a "single scalar vector" [20], meaning one probability for the whole image. We tested this approach in our network, but did not receive satisfactory results as shown in figure 7.

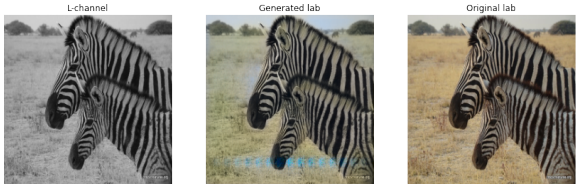

Figure 7. Discriminator with only one output probability results in desaturated colors and artifacts

Therefore, we were encouraged to implement a PatchGAN to reach better results. In contrast to the traditional approach, a PatchGAN decides whether patches of an image are real or generated. To realize this, the Discriminator splits the image in NxN local patches and outputs a matrix of classifications [13]. By applying this architecture, the discriminator is able to classify on a more local level and therefore offers a more detailed feedback. This method is based on Markov random fields, assuming that the NxN patches are independent [13]. We decided to base our implementation on the PatchGAN structure proposed by Isola et al.[13]. This proved to yield the best results and fewest artifacts. We decided on a model implementation with a receptive field size of 70x70.

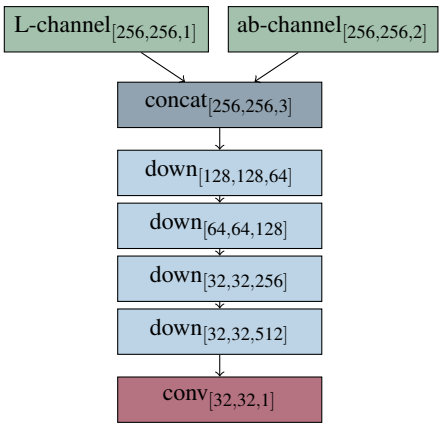

Figure 8. Our Discriminator architecture with the corresponding output shapes based on the 70x70 PatchGAN proposed by Isola et al. [13]

The precise structure is the following [13]:

C64-C128-C256-C512

"C" again represents the block *Convolutional layer-Batch Normalization-LeakyReLU*. Following the last layer, a convolution with filter size 1 and Sigmoid activation function is applied to obtain the classification values. Note that no batch normalization is used in the first layer.

4.4 Objective function

To train our discriminator and generator, we used two different losses which are optimized by the Adam optimizer.

For the **adversial loss** we use a binary cross entropy. The Generator G and Discriminator D are both trained by trying to optimize formula (1) [15]. In the formula, x represents the input L channel, y the output ab color channels and z some noise vector in the form of dropout.

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

As proposed by Mirza et al. [15] we did not train the Generator to minimize $\log(1-D(G(x,z)))$ but to maximize $\log(D(G(x,z))$. Due to this, the gradients in early learning are supposed to be stronger [15].

When using the described objective function for our network, the results are very poor, as can be seen in the figure 9 (λ=0). Isola et al. also encountered this problem and proposed adding an additional **L1 loss term**. As the PatchGAN is only modeling high frequency structures, the L1 loss is added to the Generator loss to guide the model to also include low frequency details [13]. This results in the following final objective function [13]:

$$G^* = min_G max_D \, L_{cGAN}(G,D) + \lambda * L_{L1}(G) \quad (2)$$

This combination has the advantage, that the Generator does not only try to trick the Discriminator, but also tries to generate images similar to the originals using the mean absolute error (L1 Loss). The coefficient λ regulates the influence of the L1 loss.

4.5 Dataset

For our network we use the COCO (Common Objects in Context) dataset [21]. Even though this dataset is designed for computer vision tasks like object detection and segmentation, we initially regarded its big amount of different categories and scenes as fitting for the colorization task. Due to hardware and time limitations we were only able to train and test the network with 2240 images. In the beginning, we randomly selected the 2000 images from the 2017 COCO test dataset. After training the network for 50 epochs, we still obtained only very poor results. All images consisted only of sepia colors and were lacking actual color. We realized, that for the comparatively small size of the dataset there are just too many diverse images given. Because of this, our network was not able to correctly identify the colors and uses the average instead, resulting in a consistent sepia shade. Therefore, we decided to select the images manually using the 2017 COCO validation and testing set. To receive a more homogeneous dataset, we selected only landscape and natur images. Still, without restrictions to the dataset size, we assume that our network would have also performed well with more different categories.

5. Results

In the following section we will present our results with different optimisations on our hyperparameters.

5.1 Hyperparameters

There are several important hyperparameters which can be tuned in our implementation: image size, the batch size, learning rate, kernel sizes and different Dropout rates. However, due to time and resource constraints, as well as the difficulty of analyzing the quality of a model, we used a mixture of trial and error and the proposals of related papers to find good values for the majority of hyperparameters. Still selected parameters, we compared results for different values.

5.2 Optimizations

Training a cGAN brings several difficulties. We discovered problems such as vanishing gradients, artifacts, instability that affect the general performance. To improve the model we applied several techniques which are commonly used in model construction like using Adam Stochastic Gradient Descent, Gaussian weight initialization and rescaling the images to a range of [-1, 1]. In the following, we will describe some other techniques and tests we conducted to improve our model.

5.2.1 L1 Loss

The general influence of the L1 loss was significant as depicted in figure 9. The resulting images proved to be more realistic, and the colors of larger patches became more homogeneous than without the L1 loss. We experimented with two different λ values of 10 and 100. In our model there was no significant difference present after a training process with 30 epochs as depicted in 9. Therefore we decided on a λ of 100 for our standard implementation.

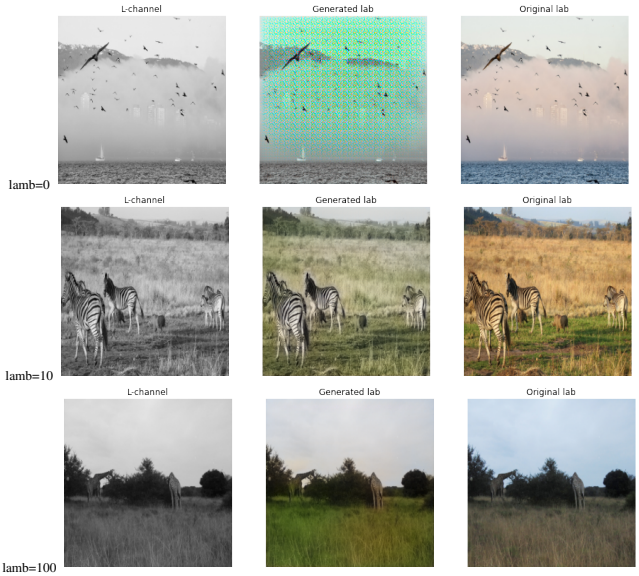

Figure 9. Images generated with a different impact of the L1 loss for the generator loss (lamb = λ)

5.2.2 Batch Normalization

In general, one problem of GANs is instability[22]. This is because a better performance of one component of the GAN causes the failing of the other one. In our case, a generator loss equal to zero means that every generated image would be labeled as true. However, this means that the Discriminator is failing. To achieve nash equilibrium and good results, we try to reach a more stable construction of the whole model by applying Batch Normalization in most layers, which stabilize learning by normalizing the input [22].

5.2.3 Leaky ReLu

When having a good or bad Discriminator with a sigmoid activation function, we get very small updates, resulting in vanishing gradients [23]. Since small gradients lead to an ineffective training and can lead to an inaccuracy of the network, we try to avoid them by using a leaky ReLu which gives small negative outputs instead of zeros. Using ReLu in general was found to increase the learning speed of the model [23].

5.2.4 Dropout

We also experimented with the dropout rate, to check whether the algorithm is highly depending on specific parts of the network. We expected, that with a lower dropout rate the generator will deliver worse results in the first thirty epochs. Especially pictures displaying huge monotonous structures, like a horizon or a forest, could influence the whole algorithm. The results are displayed in figure 10.

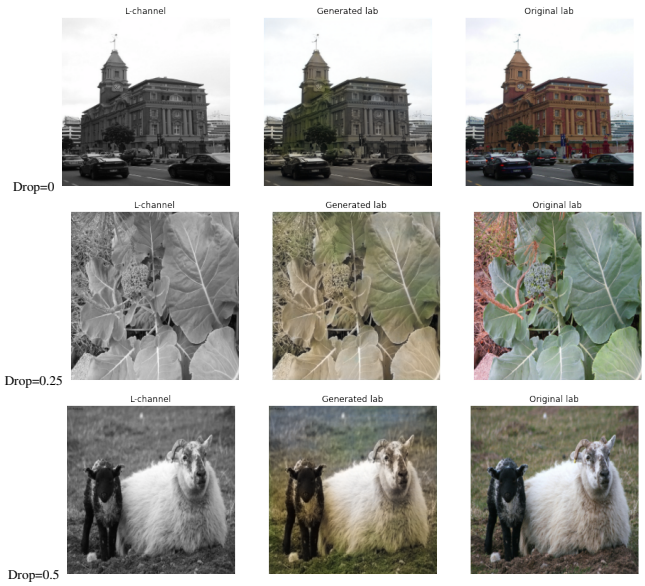

Figure 10. Results of different dropout rates after 30 epochs

As expected, the dropout rate leads to improvements. Especially the example with zero dropout shows, that the algorithm is advancing slower towards a colorization that comes up with inhomogeneous color areas and artifacts that randomly appear.

5.2.5 Strided Convolutions

In the Discriminator, the conventional approach is to use pooling for the downsampling process. However, we used strided Convolutions instead which allow the networks to learn its own way to sample down the input and seems to improve the general performance of image generation models [22].

5.2.6 Gaussian noise

Additionally we applied Gaussian noise to the input of the discriminator as this reduces overfitting and leads to a more stable training [24]. This layer is only activated during training as it is a regularization layer. In one approach we also applied the Gaussian noise to the Generator input but the performance of the Generator dropped and the generated images included artifacts. This is depicted in figure 11.

Figure 11. Adding Gaussian noise to Generator results in artifacts.

5.3 Final outcome

Now that we have been able to try out many different setups, we have settled on the following hyperparameters for the final outcome:

- Batch size: 4
- Learning rate: 0.0002
- Adam Beta 1: 0,5
- Dropout rate generator: 0.5
- Lambda: 100

Some of the generated outputs are displayed in figure 12.

Figure 12. Images generated with our standardized parameters

6. Conclusion and Outlook

Apparently the cGAN architecture including a U-Net structure for the Generator and a markovian discriminator structure proved to be a well suited approach for Image Colorization. The U-Net supports the flow of low-level information in the generation process and the patch-based classification provides a well working technique to determine local continuity in the image textures.

We adopted and adapted an implementation which is comprehensible and functioning with restricted resources and therefore achieved our goal. The results presented in this

paper are already promising after only 30 epochs considering our limited resources.

However, we are aware that a training performance using a small and restricted dataset like ours is prone to overfitting. Additionally, there still remains a certain amount of instability in our model. Not all patches are recognized, colored correctly and looking at the results more closely, it is obvious that our model is only able to resemble the colors of larger patches. Small details are usually ignored. Regarding the results of Isola et al., we assume that using a more diverse and also larger dataset would help address this problem. Further, extending the number of training epochs would possibly lead to a more consitent success. This however, would contradict our aim to create a computationally inexpensive implementation.

A major challenge of GANs in general is finding the Nash equilibrium. We also had difficulties with this and were not satisfied with the resulting losses until the end. We think that there still lies considerable potential in the correct adjustment of the hyperparameters. However, due to the long training times and restrictions on the Google Colaboratory platform, we were only able to pursue this problem to a certain extent.

An idea for the future would be to improve the hyperparameter tuning and extend this implementation to other domains such as black-and-white movies.

References

[1] Ifolor GmbH. Geschichte der Fotografie - Teil 4: Die Entdeckung des Farbfilms 1936. Ifolor GmbH, 2019. https://www.ifolor.de/inspirationen/geschichte-fotografie-teil4, Online; accessed 14-March-2022].

[2] James Clerk Maxwell. Tartan Ribbon. The Illustrated History of Colour Photography, Jack H. Coote, 1861. https://upload.wikimedia.org/wikipedia/commons/7/7f/Tartan_Ribbon.jpg, Online; accessed 15-March-2022].

[3] M. Riat. Graphische Techniken - Eine Einführung in die verschiedenen Techniken und ihre Geschichte. Burriana, 2006. M. Riat.

[4] Hao Wang and Xuedong Liu. Overview of image colorization and its applications. In *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, volume 5, pages 1561–1565. IEEE, 2021.

[5] Ivana Žeger and Sonja Grgić. An overview of grayscale image colorization methods. In *2020 International Symposium ELMAR*, pages 109–112. IEEE, 2020.

[6] Saeed Anwar, Muhammad Tahir, Chongyi Li, Ajmal Mian, Fahad Shahbaz Khan, and Abdul Wahab Muzaffar. Image colorization: A survey and dataset. *arXiv preprint arXiv:2008.10774*, 2020.

[7] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*, pages 689–694. 2004.

[8] Matthias Limmer and Hendrik PA Lensch. Infrared colorization using deep convolutional neural networks. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 61–68. IEEE, 2016.

[9] Junsoo Lee, Eungyeup Kim, Yunsung Lee, Dongjun Kim, Jaehyuk Chang, and Jaegul Choo. Reference-based sketch image colorization using augmented-self reference and dense semantic correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5801–5810, 2020.

[10] Shuchen Weng, Hao Wu, Zheng Chang, Jiajun Tang, Si Li, and Boxin Shi. L-code: Language-based colorization using color-object decoupled conditions. 2022.

[11] Aditya Deshpande, Jiajun Lu, Mao-Chuang Yeh, Min Jin Chong, and David Forsyth. Learning diverse image colorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6837–6845, 2017.

[12] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (ToG)*, 35(4):1–11, 2016.

[13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[14] Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi. Image colorization using generative adversarial networks. In *International conference on articulated motion and deformable objects*, pages 85–94. Springer, 2018.

[15] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[16] Jonathan Hui. cGAN architecture. Towards Data Science, 2018. https://miro.medium.com/max/1400/1*CVnxcMtCLendyrnLvOlKzA.png, Online; accessed 29-March-2022].

[17] Hannah Weller. CIELab Analyses. https://cran.r-project.org/web/packages/colordistance/vignettes/lab-analyses.html, 2021. Online; accessed March 29 2022.

[18] Moein Shariatnia. Colorizing black white images with U-Net and conditional GAN — A Tutorial. http://towardsdatascience.com/colorizing-black-white-images, 2020. Online; accessed March 29 2022.

[19] Adrian Rosebrock. OpenCV Color Spaces (cv2.cvtColor). https://pyimagesearch.com/2021/04/

28/opencv-color-spaces-cv2-cvtcolor/, 2021. Online; accessed March 29 2022.

[20] TheAILearner. TAG ARCHIVES: PATCHGAN. https://theailearner.com/tag/patchgan/, 2020. Online; accessed March 29 2022.

[21] . CoCO2017. https://cocodataset.org/#home, 2017. Online; accessed March 29 2022.

[22] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[23] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, page 3, 2017.

[24] Simon Jenni and Paolo Favaro. On stabilizing generative adversarial training with noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12145–12153, 2019.