FACULTEIT WETENSCHAPPEN

Automatische inkleuring van stripverhalen met deep learning

Simon Schellaert
Studentennummer: 01407082

Promotor: prof. dr. ir. Wilfried Philips
Begeleiders: Joris Roels, dr. ir. Jan Aelterman

Masterproef ingediend tot het behalen van de academische graad van
Master of Science in de informatica

Academiejaar 2018-2019

UNIVERSITEIT GENT

Automatische inkleuring van stripverhalen met deep learning

door

Simon Schellaert

Scriptie ingediend tot het behalen van de academische graad van
Master of Science in de Informatica

Academiejaar 2018–2019

Promotor: prof. dr. ir. Wilfried Philips
Begeleiders: Joris Roels, dr. ir. Jan Aelterman

Samenvatting

In dit werk stellen we een methode voor om volledig automatisch binaire lijntekeningen in te kleuren. Deze methode is gebaseerd op recente vooruitgangen in *deep learning* en *generative adversarial networks*. We stellen een gemodificeerde netwerkarchitectuur en een nieuwe loss-functie voor die leiden tot natuurlijkere resultaten. Onze methode wordt getraind en getest op prenten op Jommeke en is inzetbaar om zowel oude zwart-wit albums als nieuwe lijntekeningen in te kleuren. We evalueren onze resultaten met een perceptuele studie die aantoont dat onze methode de geloofwaardigheid van de inkleuringen drastisch verhoogt.

Trefwoorden

Inkleuring, Strips, GANs, Deep Learning, CNN

Automatic Colorization of Comics through Deep Learning

Simon Schellaert
Supervisors: prof. dr. ir. Wilfried Philips, Joris Roels, dr. ir. Jan Aelterman
Ghent University

ABSTRACT

Line art colorization is a challenging problem. Techniques used for photo colorization are typically not applicable since the input is devoid of any texture. Furthermore, color assignment is usually highly ambiguous due to the sparseness of the input. Recently, Generative Adversarial Nets (GANs) based methods have demonstrated great success at image synthesis tasks. We build upon this work and propose a novel network architecture and loss function that both significantly increase the realism of our results.

Specifically, we modify the standard U-Net architecture to increase its performance for the task at hand. This involves introducing an extra downsampling level, dilated convolutions and replacing batch normalization with instance normalization. That way, our method is able to efficiently process high-resolution imagery with a sufficiently large receptive field. Next, we introduce a multi-scale discriminator to efficiently assess the realism of these high-resolution images. Finally, we propose a novel loss based on total variation regularization that substantially increases the quality of the generated colorizations.

We evaluate our method and variants thereof on two datasets. The first dataset consists of line art generated from comic albums that were published in color. This allows us to compare the results of our method to a ground truth. The second dataset consists of old comic albums that were originally published in black-and-white. Images from this dataset are more challenging due to the use of an older drawing style and the presence of halftoning patterns. We show that our method can produce and realistic and aesthetically pleasing colorizations for both datasets.

CCS CONCEPTS

• **Computing methodologies** → **Image processing**; *Neural networks*;

KEYWORDS

Colorization; Comics; GANs; Convolutional Neural Network

1 INTRODUCTION

Nowadays, most comics are known for their extensive use of color. A few decades ago, however, color printing was still considered too expensive. So, there is an enormous collection of older comics that only appeared in black-and-white. Colorizing these older comics breathes new life into them and makes them more attractive for a wider audience. Unfortunately, doing this manually is a time-consuming process.

Another scenario where colorization takes a lot of time and effort is the creation of a new comic. Typically, a cartoonist starts by drawing the line art of the comic. These black-and-white images are then handed over to a colorist, who is responsible for adding color to them. This involves the colorist examining each image and manually applying an appropriate color to each object.

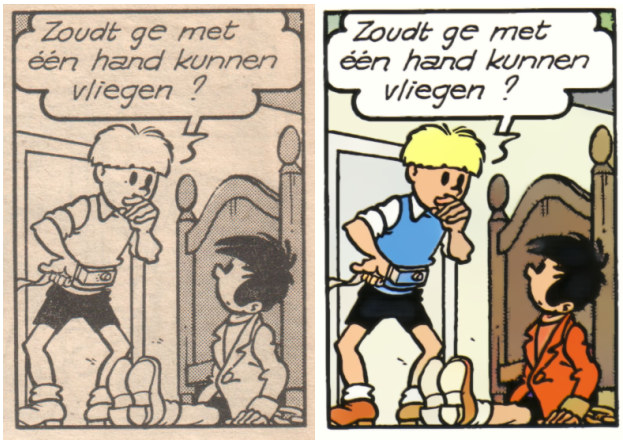


Figure 1: Our proposed method is able to colorize line art from B&W albums (left) without any assistance from the user.

In both cases, automating the colorization process would save a significant amount of time. Colorization, however, is a heavily ill-posed problem. There are frequently multiple conceivable colors that could be assigned to an object. This leads us to focus on generating plausible colorizations that are convincing and aesthetically pleasing for a human observer. Specifically, we tackle the problem of automatically generating a plausible colorization of a given binary comic book image.

2 RELATED WORK

The problem of fully automatically colorizing comic book images has not yet been widely studied. Prior work on comic colorization typically requires the user to provides color hints in the form of so-called *scribbles* [9, 17]. Since our goal is to develop a method that does not require any user intervention, this work is of limited relevance for us. Instead, we identify two different problems that share some important characteristics with our problem. This allows us to determine some approaches that are also relevant to the task at hand.

2.1 Photo Colorization

Photo colorization deals with the problem of hallucinating a plausible color version of a grayscale photograph. Just like in our problem, the input doesn't necessarily contain enough information to

recover the ground truth color. Therefore, rather than reproducing the ground truth, the goal is once again to generate plausible colorizations that can fool human observers.

Modern approaches for this problem successfully leverage large-scale data and convolutional neural networks (CNN) [10, 16, 28]. Recent research has indicated that these networks rely primarily on texture instead of shape [4]. This makes sense since the texture information in the grayscale channel is a big aid for object recognition. In our case, however, the input is a line drawing and is thus devoid of any texture. As a result, architectures that are effective for photo colorization don't necessarily perform well on our task.

Due to the multi-modal nature of the problem, optimizing the aforementioned networks using the traditional Euclidean distance yields unsatisfactory results. If an object can take on multiple distinct colors, the optimal solution according to the L_2-loss will be the mean of these colors. This results in grayish, desaturated colorizations. Consequently, Zhang et al. [28] propose to treat the problem as multinomial classification. While this indeed results in more vivid colorizations, the output also exhibits frequent splotches as the mode of the predicted distribution changes.

2.2 Image-to-image Translation

The goal of image-to-image translation is to translate an input image from one domain to another domain given input-output pairs as training data. This definition encloses a wide variety of tasks that have traditionally been tackled with separate, special-purpose machinery [2, 8, 26, 28]. Example applications include noise removal, super-resolution and image synthesis. Since L_1 loss typically leads to blurry images for many of these tasks [12, 14], the adversarial loss has become a popular choice [6] .

Recently, Isola et al. proposed the pix2pix framework [12], which employs image-conditional GANs for a variety of applications. These GANs learn a loss that penalizes unnatural results, while simultaneously training a generative model to minimize this loss. Because the learned loss automatically adapts to the data, it can be applied to a multitude of tasks that traditionally required very different kinds of loss functions.

Specifically, the architecture consists of two networks: a generator G and a discriminator D. For our task, the objective of the generator G is to translate a line drawing to realistic color image, while the discriminator D aims to distinguish between real and fake images. By pitting these two networks against each other, the generator gradually learns to produce colorizations that are indistinguishable from real ones.

3 APPROACH

Since deep learning is the approach that leads to the best results on related problems, we also opt to use a convolutional neural network. In what follows, we discuss the key features of our network architecture and the loss formulation.

3.1 Network Architecture

A straightforward idea might be to train a less powerful network to generate color hints. Afterward, these hints could be used to color connected components, which are groups of white pixels enclosed within a connected region of black pixels. More closely examining human colorizations, however, reveals that the use of one color per connected component is a crude approximation. In reality, comic images frequently contain multiple colors per connected component.

As such, we opt to utilize a fully convolutional network (FCN) [21] as an end-to-end solution. That way, the network can be applied to a line drawing of any size and maps it to a colorized version with the same resolution. The network thus needs to combine high-level semantics (to recognize objects) with low-level details (to color inside the lines). A popular architecture that provides this ability is the U-Net [19]. The use of skip connections in this architecture allows the upsampling path to exploit contextual information and location information from the downsampling path.

The standard U-Net architecture has a receptive field of 140 x 140 pixels [23]. Since objects in our input images are frequently larger than that size, an increase in contextual information is required. An efficient way to exponentially increase the receptive field is the use of dilated convolutions [27]. Specifically, we introduce an additional downsampling step and introduce dilated convolutions in the two lowest levels of the network. These changes allow us to increase the receptive field to 795 x 795 with only a modest increase in computational cost [3].

Another improvement we make is introducing normalization layers throughout the network. A recent recommendation is to use batch normalization to improve the speed, performance and stability of neural networks [11]. The effectiveness of batch normalization, however, is strongly dependent on the batch size. Since our input and output have a high resolution, the maximal batch size is limited by the available GPU memory. For this reason, we opt to use instance normalization [22]. This allows us to retain some of the benefits of batch normalization while being independent of the batch size [25].

3.2 Loss Formulation

As previously remarked, constructing an effective loss function is challenging due to the multi-modal nature of the problem. Traditional losses, such as the L_2-loss, lead to desaturated colorizations with annoying artifacts. This is caused by these losses only caring about the distance to the ground truth, and not about the realism or plausibility of the results. Since the goal here is to generate colorizations that are convincing for a human, we base our loss function on the adversarial one used in the pix2pix framework [12]. The discriminator can handle the multi-modality of the problem and penalizes deviations that expose our colorization as fakes.

Early experiments revealed that the use of a discriminator significantly boosts the realism of the colorizations. Nevertheless, the results still contain some mistakes that are insufficiently penalized by the discriminator. As such, we introduce another term in the loss function that penalizes these mistakes. This term is based on total variation regularization and is subsequently named *total variation loss*. The final loss function then consists of three terms: L_1-loss, discriminator loss and total variation loss. In what follows, we examine each of these terms in detail.

3.2.1 L_1-loss. While only using L_1-loss does not necessarily result in realistic colorizations, it is nevertheless a useful loss function. For a lot of objects, such as skin, sky or trees, the colorization is

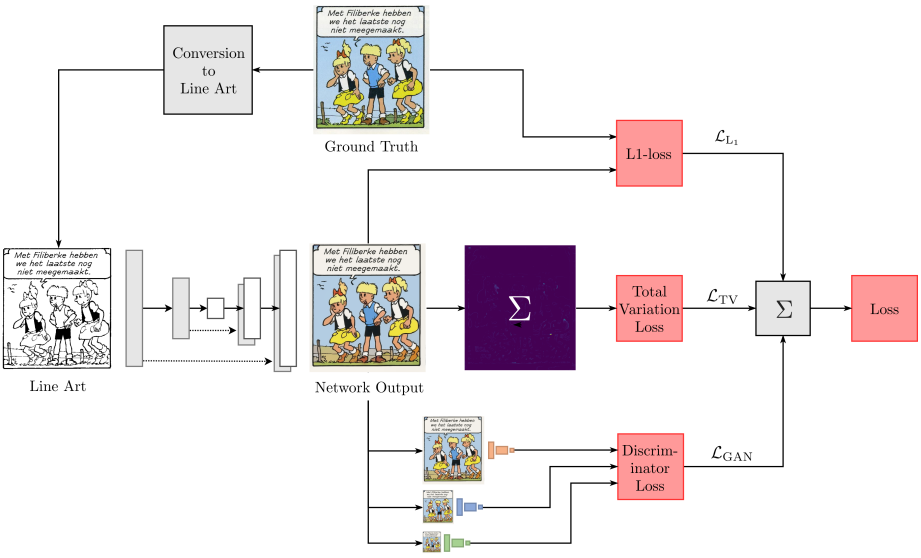


Figure 2: Schematic overview of training our method. The ground truth images are converted to line art and fed to the network as input. The network weights are then optimized based on a loss function consisting of three terms: L_1-loss, discriminator loss and total variation loss.

unambiguous and using this loss is a good approximation. Furthermore, adding an L_1 term makes sure that the output respects the input. This can be seen by noting that this loss penalizes the distance between ground truth outputs, which correctly match the input, and synthesized outputs, which may not. We calculate all distances in the CIELAB color space since distances in this space model perceptual distance.

3.2.2 Discriminator loss. Exclusively using L_1-loss results in reasonable colorizations for a lot of characters and objects. Nevertheless, the colorizations frequently contain imperfections that expose them as fake. The three most prominent mistakes are splotches, unwanted color transitions and desaturated colors.

While these mistakes are conspicuous to human observers, it is hard to construct a loss function that penalizes them. Based on this observation, we introduce a discriminator in our architecture. The task of this discriminator is then to make our colorizations indistinguishable from human colorizations.

Since our output has a high resolution, the receptive field of the discriminator has to be sufficiently large to evaluate it. A discriminator with such a large receptive field, however, is prone to overfitting and has a large memory footprint. To address the issue, we use a multi-scale discriminator, as proposed by Wang et al. [24]. We use 3 discriminator networks that have an identical architecture but operate at different image scales. Specifically, the output of the network is downsampled to scales $\frac{1}{2}$, $\frac{1}{4}$ and $\frac{1}{8}$ before being fed to the respective discriminator. This allows the discriminator at the finest scale to penalize splotches and other artifacts, while the coarsest discriminator enforces global consistency.

The architecture of each individual discriminator network is an unconditional *PatchGAN*, a FCN that classifies 70 x 70 patches as real or fake [12, 21]. The task of the colorization network is then no longer just to be near the ground truth, but also to fool the discriminators by producing natural colorizations.

3.2.3 Total variation loss. While introducing a discriminator reduces the number of artifacts, we still frequently observe unwanted color transitions. Since the discriminator doesn't seem to rectify these as much as other mistakes, we introduce a loss term that penalizes such transitions and thus improves the realism of our results. This term is based on total variation regularization, a process that is traditionally employed for noise removal. The key observation is that noise leads to a high total variation, i.e., the integral of the absolute gradient of the signal is high. Adapted to our problem, color transitions similarly lead to a high total variation of the output image.

Directly applying this idea to our case, however, leads to the gradient being dominated by the transitions between colored areas and black lines. To avoid this, we mask the matrix of gradient magnitudes with a dilated version of the input line art. Afterward, we apply the hyperbolic tangent function to the resulting matrix before summing it. This procedure leads to an estimate of the severity of color transitions in the output. By adding this term to the loss function, we discourage the network from introducing color changes within a surface without robbing it of the ability to do so if needed.

The final loss function is a weighted sum of the three terms defined above. Since each of these terms is made up of differentiable functions, we are able to apply backpropagation to optimize the weights of the network. The final architecture is depicted in figure 2.

4 DATASET

A crucial aspect of every machine learning pipeline is the dataset. By carefully choosing a dataset, we can get more insight into the performance of our method. In what follows, we elaborate on the choice and processing of our dataset.

4.1 Comic Series

Our method can be applied to any comic, as long as a sufficiently large set of colorized training samples is available. We opt to train and test our method on *Jommeke*, a popular Belgian comic series. A primary reason for choosing this series is that the first 91 of the over 290 comics were published in black-and-white. With the introduction of album #92, the series switched to publications in colors. This allows us to test our method on original black-and-white comics. In total, our dataset consists of 2818 scans of black-and-white pages and 9329 scans of color pages.

4.2 Conversion of Color Comics to Line Art

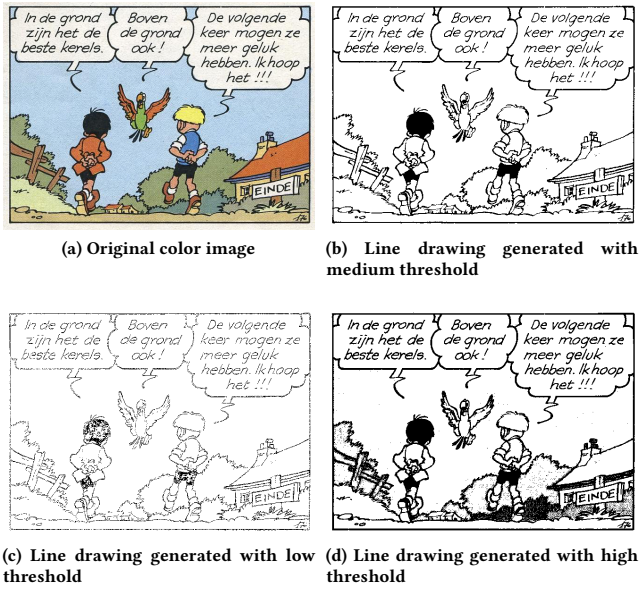
An important remark is that the dataset contains each album either in black-and-white or in color. We thus don't possess the pairs of black-and-white images with corresponding colorized versions that are required for training the network. We can, however, take advantage of the fact that converting color images to black-and-white images is typically much easier than the other direction.

We thus automatically generate a binary line drawing for each color image, to use as the input of our network. In this line drawing, every black pixel should correspond to a black line or surface in the original image. Since our datasets consist of scans, however, the pixels that we want to keep in the line art are not necessarily perfectly black in the scanned color image. Because the actual colors are perturbed by various independent sources of noise, we define a multivariate normal distribution over the CIELAB color space. This distribution expresses the probability that a pixel with a specific color belongs to a black line or surface. We then use this distribution and a chosen threshold to determine which pixels to color black in the line art.

The ideal threshold, however, varies from image to image and is hard to determine without human intervention. Rather than manually determining an appropriate threshold for each image, we randomly pick one from an interval containing the most common thresholds. Thus, each time we feed an image to the network during training, we uniformly sample this interval and generate a line drawing based on the chosen threshold. This results in the network seeing each ground truth color image with a variety of line drawings, varying in the used threshold, as input. This procedure can also be seen as a kind of data augmentation. Sometimes the threshold will be too high, resulting in extraneous black pixels in the input (see figure 3d). Other times, the threshold will be too low, resulting in the disappearance of some line segments (see figure 3c). This diversity makes the network robust to noisy input images.

4.3 Conversion of Old B&W Comics

In the previous section, we proposed a process for converting color images to line art. This allows us to generate the pairs of line art and color image that are needed to train the network. After training, we can use the network to colorize the line art drawn during the production process of a comic. Another major use case is the colorization of old black-and-white comics. As discussed in section 4.1, we also have 2818 scans of black-and-white pages available. These scans, however, are not binary images, as our network expects as input. Rather, they are color scans of yellowed black-and-white pages with specks of dust. This is problematic since an important assumption in supervised learning is that the train and test set are both sampled from the same underlying distribution [5]. We now discuss exactly how these scans differ from the images our network is trained on and how we tackle these differences.



(a) Original color image

(b) Line drawing generated with medium threshold

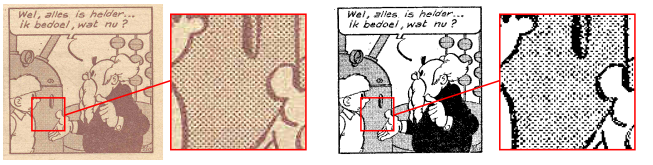
(c) Line drawing generated with low threshold

(d) Line drawing generated with high threshold

Figure 3: Influence of the used threshold on the generated line drawings

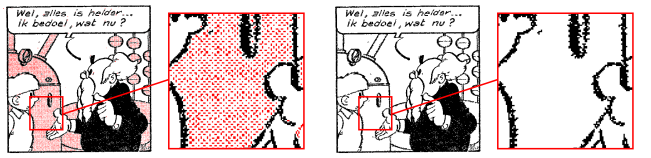
4.3.1 Halftoning removal. To convert these scans to binary images, that can be fed to our network, we start by applying the same procedure as above with a different threshold range. The result of applying this procedure to the image in figure 4a, is shown in figure 4b. We observe that the black lines and surfaces are correctly transferred to the binary image. A complication, however, is the presence of halftoning dots in the source image, and thus also in the resulting binary image. In particular, these black-and-white albums use one level of halftoning to add some depth to the scene. The usage of this pattern is purely an artistic choice and doesn't necessarily convey any information about the actual color of an object. Its presence, however, does interfere with the operation of our network as the network is not used to this kind of pattern.

The process of reconstructing a continuous-tone images from a halftone version, is called *inverse halftoning* or *descreening* [13, 15]. A straightforward way to do so is the application of a low-pass filter, such as a Gaussian filter. A problem with this approach is that it blurs the image and thus also throws away relevant details around the edges. For our purposes, however, the halftoning patterns don't



(a) Halftoning in a scan of a black-and-white comic

(b) Halftoning after conversion to a binary image



(c) Binary image with small connected components indicated in red. Notice how this are primarily halftoning dots, but also a piece of the hand.

(d) Binary image after removal of small connected components

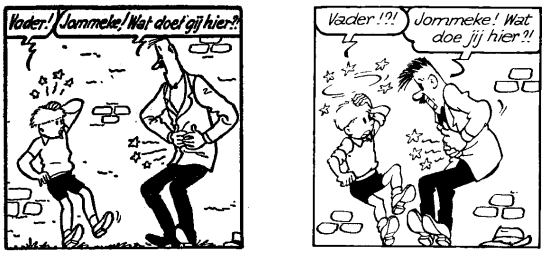
Figure 4: Example of the removal of halftoning patterns

contain useful information, so we can focus solely on removing them instead of converting them to a continuous tone.

To remove these patterns, we make use of *connected-component labeling*, which groups black pixels in the binary image in groups of adjacent pixels [7]. After grouping the black pixels by connected component, we mark the components that consist of less than 7 pixels, as shown in red on figure 4c. These small connected components are mainly halftoning dots. By removing them, we obtain an image without halftoning dots, as shown in figure 4d. This is, however, only an approximation so some artifacts remain around the edges and useful lines are sometimes discarded as well. Nevertheless, the process works well in general and can remove most halftoning dots from the binary image.

4.3.2 Drawing Style. A second important difference between the black-and-white albums and the color albums is the used drawing style. As mentioned in section 4.1, the black-and-white albums were all published before the color albums that our network was trained on. Throughout the years, however, the drawing style has evolved and is now very distinct from the style used in earlier albums. This difference is illustrated in figure 5. The line art on the left is generated from the first edition of the first album in the series, which appeared in black-and-white. The line art on the right is generated from a recently revised version of that first album. This difference is problematic since a lot of the features that distinguish characters in later albums are missing in earlier albums.

Resolving these discrepancies between older and more recent comics turns out to be challenging. The aforementioned album is the only one of which we have a version in both the old and new drawing style. These two versions, however, not only differ in drawing style but also frequently in content. This makes it hard to train a mapping from the old to the new drawing style. Luckily, the drawing style of these old albums converges rather quickly to the recent one. At the 15th album (out of 91 black-and-white albums), the drawing style is already quite close to the contemporary style



(a) Line art from black-and-white album #1

(b) Line art from a recently redrawn version of album #1

Figure 5: Example demonstrating the difference in drawing style between older and more recent albums.

for most characters. From that point on, we can rely on the network robustness to variations in the drawn lines.

To improve the performance on earlier albums, we could use *domain adaptation* [1] or manually colorize some older comics as extra training data. The first option, however, introduces a significant amount of extra complexity, while the second option is not in keeping with our method being fully automatic. We thus don't elaborate on these techniques and accept that the performance is degraded on the earliest albums.

5 RESULTS

Evaluating the quality of the generated colorizations is an open and difficult problem [20]. Traditional metrics, such as the mean absolute error (MAE) or peak signal-to-noise ratio (PSNR), measure the distance to the ground truth rather than the realism of the results. For this reason, we set up a human perceptual study that allows us to evaluate how plausible our colorizations look to human observers. In the following section, we discuss the design and results of this study.

5.1 Variants

As previously discussed, our final loss function consists of three terms. To evaluate the impact of each of these terms on the realism of the results, we train three variants of the networks. For each of these variants, the network architecture and training procedure are identical, except for omitting certain terms in the loss function. Specifically, we train the following variants:

- L_1-loss
- L_1-loss + discriminator loss
- L_1-loss + discriminator loss + total variation loss

The last of these variants is our full method. Besides training three variants, we also evaluate the performance on two datasets: *color comics* and *black-and-white comics*. The first dataset consists of line art that is generated from a set of color albums. This dataset thus has the same characteristics as the data our network trained on and allows comparison of the results to a ground truth. The second dataset consists of line art that is generated from a set of older black-and-white albums. This dataset is harder due to the aforementioned issues with the drawing style. Furthermore, there is no ground truth available for these albums.

5.2 Human Perceptual Study

The primary goal of our method is to produce colorizations that look natural to human observers. We thus follow the procedure outlined by Iizuka et al. and base our study around the question *"Does this image look natural to you?"* [10]. Specifically, we set up an experiment where 84 human observers are each shown 90 random color panels of which they have to determine if they look real or fake. These color images are randomly chosen from sets of colorized images for each of the three variants or from a set of ground truth images. As explained in the previous section, we also split based on whether the input was line art generated from a color album or from a black-and-white album. After viewing each panel, the user has to indicate whether the panel looks real or fake.

During these tests, we want to prevent participants from focusing too much on details they normally wouldn't pay attention to. To do this, we give participants instructions to treat the images like they would do when reading an ordinary comic and use to their gut feeling in case of doubt. Moreover, we also artificially limit the time that each image is shown. Previous experiments indicated that readers take roughly two and a half seconds per panel to read and inspect it. We thus show each panel for three seconds, after which the participant has unlimited time to indicate whether it looks real or not.

5.3 Analysis

The percentage of panels deemed real per network variant and per dataset is shown in table 1. We start by noting that participants labeled 92.6 % of the ground truth images as real. This demonstrates that participants are competent at the task but critical in case of doubt.

Next, we look at the results of colorizations based on line art from color albums (the column *color album*). Our full method can fool participants on 65.2 % of the panels. This is a substantial improvement over the 53.0 % when training the network using only L_1-loss. A remarkable result is that the performance of the variant L_1 + discriminator is lower than the variant trained without a discriminator. We're able to explain this phenomenon by more closely examining the resulting colorizations. While the discriminator leads to more vivid colors, it also leads to the introduction of checkerboard artifacts in the output [18]. The most effective way to fix these artifacts is to change the kernel size of the convolutions in the discriminator to be a multiple of the stride. Since these artifacts, however, are not present in our full method and we only modify the loss function for each variant, we don't do that here.

Finally, we inspect the results when applying our method to albums that originally appeared in black-and-white. The performance of our full method on this set is, with 45.9 %, significantly lower than when applied to line art generated from color albums. We didn't test the other variants of the network on this dataset since we expect the reductions in performance to be comparable to those determined above. Analyzing the least convincing images shows us that the major hurdles are indeed the aforementioned halftoning and drawing style. First, the halftoning removal is frequently imperfect, with the remaining dots leading to grayish and noisy colorizations. Manually removing these remaining dots typically results in a dramatic increase in performance. Secondly, the network is not always able to recognize characters when they are drawn in a more primitive style in the earlier albums.

Variant	Color album	B&W album
L_1	53.0 %	-
L_1 + discriminator	45.6 %	-
L_1 + discriminator + TV	**65.2 %**	45.9 %
Ground truth	92.6 %	N/A

Table 1: Percentage van de prenten dat als echt werd beoordeeld per variant en verzameling.

5.4 Examples

We show colorization results on the set color comics (figure 6) and B&W comics (figure 7). These images were chosen to demonstrate the ability of our method to colorize a wide variety of comic book panels. Note that all these results were generated automatically without any human intervention.

6 CONCLUSION

In this paper, we have presented an end-to-end trainable approach for automatic colorization of line art. We extended the pix2pix framework with a more capable U-Net and loss function that is custom-tailored for comic colorization. We then ran a perceptual study to evaluate the performance of our model on two datasets. This study indicates that our novel loss significantly improves the realism of our results. We've also shown that our method can be used to revive degraded line art of older black-and-white comics.

REFERENCES

[1] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A Theory of Learning from Different Domains. *Mach. Learn.* 79, 1-2 (May 2010), 151–175. https://doi.org/10.1007/s10994-009-5152-4

[2] Antoni Buades, Bartomeu Coll, and J.-M Morel. 2005. A non-local algorithm for image denoising. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2, 60– 65 vol. 2. https://doi.org/10.1109/CVPR.2005.38

[3] Vincent Dumoulin and Francesco Visin. 2016. A guide to convolution arithmetic for deep learning. *CoRR* abs/1603.07285 (2016). http://dblp.uni-trier.de/db/journals/corr/corr1603.html#DumoulinV16

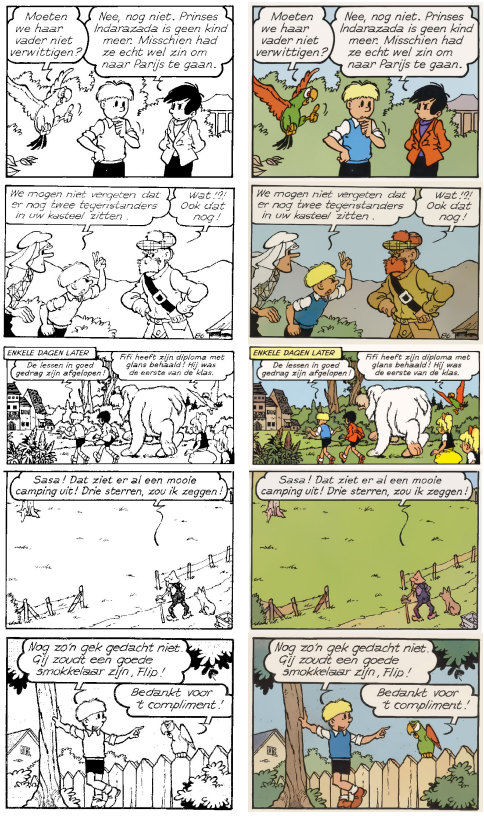
[4] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. 2018. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. *CoRR* abs/1811.12231 (2018). arXiv:1811.12231 http://arxiv.org/abs/1811.12231

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2672–2680. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

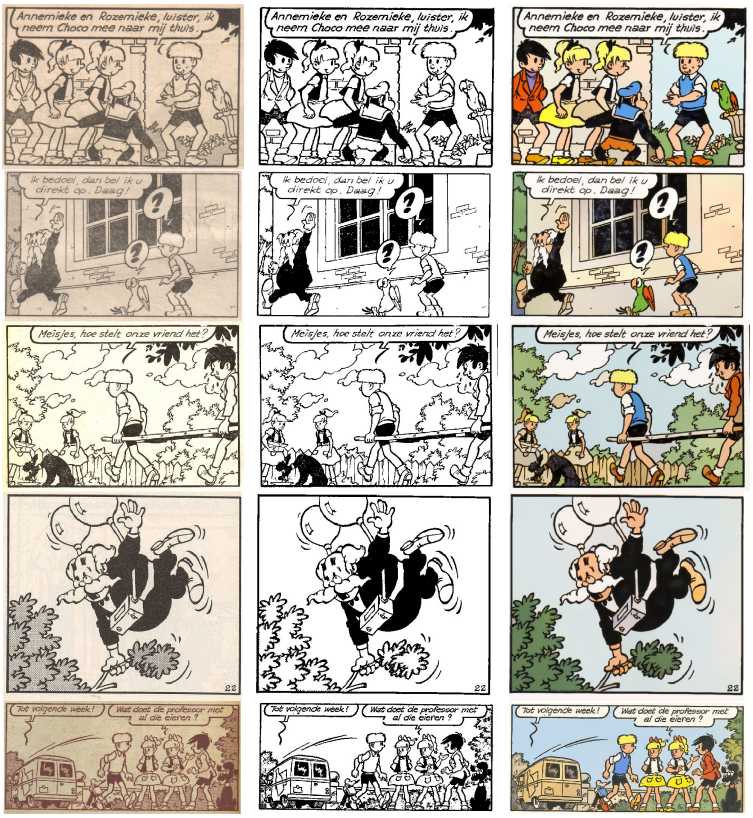
[7] Lifeng He, Xiwei Ren, Qihang Gao, Xiao Zhao, Bin Yao, and Yuyan Chao. 2017. The connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition* 70 (2017), 25 – 43. https://doi.org/10.1016/j.patcog.2017.04.018

[8] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. 2001. Image Analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. ACM, New York, NY, USA, 327–340. https://doi.org/10.1145/383259.383295



(a) Binary Input (b) Output

Figure 6: Results on 'Color Comics'



(a) Original Scan (b) Binary Input (c) Output

Figure 7: Result on 'B&W Comics'

[9] Yi-Chin Huang, Yi-Shin Tung, Jun-Cheng Chen, Sung-Wen Wang, and Ja-Ling Wu. 2005. An adaptive edge detection based colorization algorithm and its applications. 351–354. https://doi.org/10.1145/1101149.1101223

[10] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2016. Let There Be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Trans. Graph.* 35, 4, Article 110 (July 2016), 11 pages. https://doi.org/10.1145/2897824.2925974

[11] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR* abs/1502.03167 (2015). arXiv:1502.03167 http://arxiv.org/abs/1502.03167

[12] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2016. Image-to-Image Translation with Conditional Adversarial Networks. *CoRR* abs/1611.07004 (2016). arXiv:1611.07004 http://arxiv.org/abs/1611.07004

[13] Zhangying Jin and Enyin Fang. 2018. *Print Inverse Halftoning and Its Quality Assessment Techniques.* 211–220. https://doi.org/10.1007/978-981-10-7629-9_26

[14] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. 2016. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. *CoRR* abs/1603.08155 (2016). arXiv:1603.08155 http://arxiv.org/abs/1603.08155

[15] Ilya Kurilin, Ilia Safonov, Hokeun Lee, and Sang Kim. 2010. Descreening of scanned images. *Proceedings of SPIE - The International Society for Optical Engineering* 7528 (01 2010). https://doi.org/10.1117/12.838444

[16] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2016. Learning Representations for Automatic Colorization. *CoRR* abs/1603.06668 (2016). arXiv:1603.06668 http://arxiv.org/abs/1603.06668

[17] Simeng Li, Qiaofeng Liu, and Haoyu Yuan. 2018. Overview of Scribbled-Based Colorization. *Art and Design Review* 06 (01 2018), 169–184. https://doi.org/10.4236/adr.2018.64017

[18] Augustus Odena, Vincent Dumoulin, and Chris Olah. 2016. Deconvolution and Checkerboard Artifacts. *Distill* (2016). https://doi.org/10.23915/distill.00003

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR* abs/1505.04597 (2015). arXiv:1505.04597 http://arxiv.org/abs/1505.04597

[20] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved Techniques for Training GANs. *CoRR* abs/1606.03498 (2016). arXiv:1606.03498 http://arxiv.org/abs/1606.03498

[21] Evan Shelhamer, Jonathan Long, and Trevor Darrell. 2017. Fully Convolutional Networks for Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 4 (2017), 640–651. https://doi.org/10.1109/TPAMI.2016.2572683

[22] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. 2016. Instance Normalization: The Missing Ingredient for Fast Stylization. *CoRR* abs/1607.08022 (2016). arXiv:1607.08022 http://arxiv.org/abs/1607.08022

[23] Freerk G. Venhuizen, Bram van Ginneken, Bart Liefers, Freekje van Asten, Vivian Schreur, Sascha Fauser, Carel Hoyng, Thomas Theelen, and Clara I. Sánchez. 2018. Deep learning approach for the detection and quantification of intraretinal cystoid fluid in multivendor optical coherence tomography. *Biomed. Opt. Express* 9, 4 (Apr 2018), 1545–1569. https://doi.org/10.1364/BOE.9.001545

[24] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2017. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. *CoRR* abs/1711.11585 (2017). arXiv:1711.11585 http://arxiv.org/abs/1711.11585

[25] Yuxin Wu and Kaiming He. 2018. Group Normalization. *CoRR* abs/1803.08494 (2018). arXiv:1803.08494 http://arxiv.org/abs/1803.08494

[26] Saining Xie and Zhuowen Tu. 2015. Holistically-Nested Edge Detection. *CoRR* abs/1504.06375 (2015). arXiv:1504.06375 http://arxiv.org/abs/1504.06375

[27] Fisher Yu and Vladlen Koltun. 2015. Multi-Scale Context Aggregation by Dilated Convolutions. *CoRR* abs/1511.07122 (2015). arXiv:1511.07122 http://arxiv.org/abs/1511.07122

[28] Richard Zhang, Phillip Isola, and Alexei A. Efros. 2016. Colorful Image Colorization. *CoRR* abs/1603.08511 (2016). arXiv:1603.08511 http://arxiv.org/abs/1603.08511

Vulgariserende samenvatting

Tegenwoordig denken we bij strips aan levendige prenten die overvloedig gebruik maken van kleur om de lezer in het verhaal te trekken. Nog niet zo lang geleden was afdrukken in kleur echter nog erg duur en verschenen stripverhalen in zwart-wit. Deze oude stripverhalen inkleuren zou hen nieuw leven inblazen en aantrekkelijker maken voor een groter publiek. Dit handmatig doen is echter een erg arbeidsintensief proces. Een andere scenario waar inkleuren veel tijd en moeite kost is de productie van nieuwe stripalbums. Hierbij tekent een tekenaar eerst lijntekeningen voor elke prent. Die lijntekeningen worden dan doorgegeven aan een inkleurder die voor elke prent manueel de juiste kleuren aanbrengt.

In beide gevallen zou het automatiseren van dit inkleuringsproces veel tijd besparen. Het inkleuren van lijntekeningen is echter een uitdagend probleem. De lijntekening die als invoer dient bevat typisch te weinig informatie om elk object met zekerheid één bepaalde kleur toe te kennen. Het hoofddoel is dan ook het creëren van inkleuringen die geloofwaardig overkomen voor mensen. Specifiek ontwikkelen we in deze thesis een methode voor het automatisch genereren van een overeenkomstige, geloofwaardige kleurenversie op basis van een lijntekening. Om onze methode de kneepjes van het inkleuren aan te leren en te testen maken we gebruik van Jommeke-albums. De uiteindelijke methode kan echter net zo goed toegepast worden op andere stripreeksen.

Het manueel schrijven van een programma dat elk personage of object vanuit elk mogelijk standpunt herkent, zou gigantisch veel werk zijn en is dus niet realistisch. Gelukkig kunnen we ons laten inspireren door het menselijke brein, dat voortdurend bijleert op basis van voorbeelden. Specifiek maken we gebruik van een zogenaamd *neuraal netwerk*. Dit is een techniek die losjes gebaseerd is op het menselijk brein en zo kan patronen herkennen. Eens we de structuur van dit neuraal netwerk vastleggen, kan het vervolgens leren hoe het precies stripalbums inkleurt. Om dit te doen tonen we het netwerk duizenden voorbeelden van ingekleurde prenten uit Jommeke. Na een tijdje begint het netwerk dan zelf de personages en objecten te herkennen.

De bovenstaande techniek wordt reeds succesvol gebruikt voor bijvoorbeeld zelfrijdende auto's of gezichtsherkenning. Een moeilijkheid in ons probleem is dat we personages en objecten niet enkel willen herkennen, maar ook correct inkleuren. Het is echter niet makkelijk om inkleuringen volledig automatisch te beoordelen. Zo kan een appel bijvoorbeeld op meerdere manieren inge-

kleurd worden. Dat het netwerk die appel dan groen kleurt, terwijl hij eigenlijk een rode kleur heeft op de originele inkleuring, willen we dus niet afstraffen. Het is dus moeilijk automatisch te bepalen welke inkleuringen al dan niet steek houden. Nochtans heeft het neuraal netwerk die feedback hard nodig om beter te worden in het inkleuren.

Om het inkleuringsnetwerk de nodige feedback te kunnen geven, maken we gebruik van een tweede neuraal netwerk: een *discriminator*. De bedoeling van dit netwerk is om erg goed te worden in het beoordelen van inkleuringen. We kunnen het oordeel van dit netwerk dan gebruiken om ons eerste inkleuringsnetwerk te verbeteren. Een veelgebruikte analogie is om ons inkleuringsnetwerk te zien als een valsmunter die voortdurend tracht om valse valuta (automatische inkleuringen) te produceren en ongemerkt te gebruiken. De discriminator is dan analoog aan de politie die probeert om deze valse valuta (automatische inkleuringen) te onderscheiden van echte valuta (menselijke inkleuringen). De competitie in dit scenario lijdt ertoe dat beide partijen hun technieken verbeteren tot de valse valuta (automatische inkleuringen) nauwelijks nog te onderscheiden valt van echte valuta (menselijke inkleuringen).

Eens we het inkleuringsnetwerk getraind hebben via bovenstaande techniek, kunnen we het gebruiken om volledig automatisch nooit eerder geziene lijntekeningen in te kleuren. Zo kunnen we onze methode bijvoorbeeld toepassen op oude Jommeke-albums die oorspronkelijk in zwart-wit verschenen. Het resultaat hiervan wordt getoond in onderstaande afbeelding. De gekleurde rechterafbeelding is daarbij het resultaat van ons netwerk als we als invoer de linkerafbeelding gebruiken. Belangrijk op te merken is dat deze inkleuring volledig automatisch werd gegenereerd, zonder enige menselijke tussenkomst.

(a) Invoer van het netwerk

(b) Uitvoer van het netwerk

Dankwoord

Met trots en enige opluchting leg ik met het schrijven van dit voorwoord de laatste hand aan mijn masterproef. Vanaf het eerste moment dat ik las over het automatisch inkleuren van stripverhalen, was ik mateloos gefascineerd. Aan de ene kant volstaat één zin om de probleemstelling voor iedereen duidelijk te maken. Aan de andere kant vereist een oplossing een boeiende mix van state-of-the-art technieken. Het was dan ook een ongelofelijke ervaring om me de afgelopen maanden hier volledig in onder te dompelen. Een oplossing uitwerken bleek een erg leerrijk maar ook uitdagend proces te zijn. Gelukkig werd ik hierbij zeer goed bijgestaan door heel wat mensen. Ik grijp dan ook graag de gelegenheid aan om een woordje van dank tot hen te richten.

Eerst wil ik mijn promotor, prof. dr. ir. Wilfried Philips, bedanken voor het beschikbaar stellen van dit onderwerp. Hij gaf me de kans om me volledig te verdiepen in dit interessante onderwerp. Vervolgens bedank ik mijn begeleiders, Jan Aelterman en Joris Roels, voor de uitstekende begeleiding. Hun vindingrijke inzichten, kritische opmerkingen en passie voor het onderwerp waren een voortdurende bron van inspiratie. Mijn dankbaarheid gaat ook uit naar mijn vrienden en familie. De vele mooie momenten met hen waren de perfecte manier om even tot rust te komen. In het bijzonder wil ik mijn ouders bedanken voor hun onuitputtelijke steun in alles wat ik onderneem. Zij gaven mij de kans om dit allemaal te doen en daar ben ik hen eeuwig dankbaar voor. Tenslotte bedank ik iedereen die deelnam aan de perceptuele experimenten. Hun bereidheid om tien minuten van hun kostbare tijd op te offeren om mij te helpen werd erg geapprecieerd.

Toelating tot bruikleen

De auteur geeft de toelating deze masterproef voor consultatie beschikbaar te stellen en delen van de masterproef te kopiëren voor persoonlijk gebruik. Elk ander gebruik valt onder de bepalingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit deze masterproef.

Simon Schellaert, 10 augustus 2019

Inhoudsopgave

Dankwoord **i**

Toelating tot bruikleen **ii**

1 Inleiding **1**
1.1 Probleemstelling . . . 2
1.2 Doel . . . 3
1.3 Opbouw . . . 4

2 Literatuurstudie **5**
2.1 Gerelateerde problemen . . . 5
2.1.1 Inkleuren van zwart-witfoto's . . . 5
2.1.2 Semantische segmentatie . . . 7
2.1.3 Image-to-image translation . . . 8
2.2 Image Analogies . . . 8
2.3 Deep Learning . . . 9
2.3.1 Fully Convolutional Networks . . . 9
2.3.2 SegNet . . . 10
2.3.3 U-Net . . . 11
2.3.4 Dilated convolutions . . . 12
2.4 Loss-formulering . . . 13
2.4.1 CIELAB-kleurruimte . . . 13
2.4.2 L_p-loss . . . 14
2.4.3 Cross-entropy loss . . . 15
2.5 Generative adversarial networks . . . 17
2.5.1 Basisidee . . . 17
2.5.2 pix2pix . . . 19
2.5.3 pix2pixHD . . . 21

3 Dataset **26**
3.1 Keuze stripreeks . . . 26
3.2 Opbouw . . . 27

3.3 Train-test splitsing . . . 28
3.4 Data augmentatie . . . 29
3.5 Conversie naar lijntekeningen . . . 33

4 Methode **36**
4.1 Formele probleemstelling . . . 36
4.2 Netwerkarchitectuur . . . 37
4.2.1 Tekortkomingen van encoder-decoder . . . 37
4.2.2 U-Net met à trous convoluties . . . 38
4.3 Loss-formulering . . . 41
4.3.1 L_1-loss . . . 41
4.3.2 Discriminator loss . . . 43
4.3.3 Total variation loss . . . 46
4.4 Trainingsdetails . . . 50
4.4.1 Varianten . . . 51
4.4.2 Initialisatie . . . 51
4.4.3 Optimalisatie . . . 51
4.4.4 GAN . . . 52

5 Evaluatie en experimenten **54**
5.1 Testopstelling . . . 54
5.2 CIE76-kleurafwijking . . . 55
5.3 Perceptuele experimenten . . . 59
5.3.1 Opzet . . . 59
5.3.2 Resultaten . . . 60
5.4 Foutanalyse . . . 64
5.5 Resultaten op andere datasets . . . 67
5.5.1 Schetsen van fans . . . 67
5.5.2 Andere stripreeksen . . . 67

6 Toepassing op oude zwart-witalbums **71**
6.1 Dataset . . . 71
6.2 Verschillen met kleurenstrips . . . 73
6.2.1 Tekenstijl . . . 73
6.2.2 Halftoning . . . 75
6.3 Perceptuele experimenten . . . 83

7 Conclusie **87**
7.1 Toekomstig werk . . . 88

Bibliografie **91**

Appendices **99**

A Voorbeeldinkleuringen **99**
A.1 Gezien album: meest geloofwaardige inkleuringen 100
A.2 Gezien album: minst geloofwaardige inkleuringen 110
A.3 Ongezien album: meest geloofwaardige inkleuringen 115
A.4 Ongezien album: minst geloofwaardige inkleuringen 124
A.5 Zwart-witalbum: meest geloofwaardige inkleuringen 129
A.6 Zwart-witalbum: minst geloofwaardige inkleuringen 139

Hoofdstuk 1

Inleiding

Tegenwoordig denken we bij het begrip *afbeelding* steevast aan kleurrijke afbeeldingen. Nog niet zo lang geleden was de meerderheid van het beeldmateriaal echter zwart-wit. Zo was doorheen de 19e eeuw fotografie hoofdzakelijk monochroom. Pas in 1960 stond de techniek ver genoeg om kleurenfotografie beschikbaar te maken voor consumenten. Een ander voorbeeld zijn stripverhalen in binair zwart-wit. Tegenwoordig worden veel strips gekenmerkt door hun gebruik van felle kleuren om de lezer in het verhaal te trekken. In de eerste helft van de 20e eeuw was afdrukken in kleur echter nog erg duur, waardoor strips toen in zwart-wit verschenen. Er is dus een overvloed aan historisch beeldmateriaal dat enkel in zwart-wit beschikbaar is.

Nochtans spreken kleurbeelden typisch heel wat meer tot de verbeelding. De afgelopen jaren is er dan ook veel onderzoek verricht naar het automatisch omzetten van zwart-witafbeeldingen naar kleur. Daarbij wordt er voornamelijk gefocussed op monochrome foto's aangezien deze zo alomtegenwoordig zijn. Typisch worden daarbij methodes aangewend die sterk afhankelijk zijn van de textuurinformatie in monochrome beelden. Hierdoor zijn deze methodes niet inzetbaar voor stripverhalen waarbij de prenten weinig tot geen textuur bevatten. Voorgaand werk over het inkleuren van strips verwacht dan ook voor elk vlak een kleurhint van de gebruiker. Het aanbrengen van dergelijke kleurhints neemt echter veel tijd in beslag waardoor deze techniek slechts beperkt bruikbaar is. Het doel van deze thesis is dan ook een methode te ontwikkelen die in staat is volledig automatisch zwart-wit stripverhalen in te kleuren.

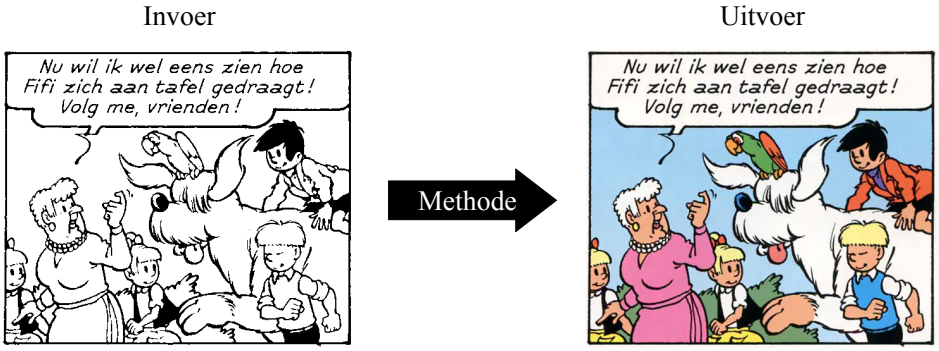
Dergelijke lijntekeningen inkleuren zonder menselijke tussenkomst was tot voor kort een zo goed als onmogelijke opgave. De recente opmars van deep learning leidde echter tot een gigantische vooruitgang in het interpreteren en genereren van afbeeldingen. Deze nieuwe technieken stellen ons in staat een methode te ontwikkelen die, in tegenstelling tot voorgaand werk, geen extra hints van de gebruiker vereist. We kunnen onze methode vervolgens gebruiken om oude zwart-witalbums nieuw leven in te blazen of lijntekeningen van onafgewerkte albums in te kleuren.

Naast deze praktische toepassingen is het probleem ook een erg interessante technische uitda-

ging. Om een afbeelding correct in te kleuren, dienen we namelijk heel wat deeltaken op te lossen, waaronder segmentatie en classificatie. Meer nog kunnen we ons probleem beschouwen als een geval van *image-to-image translation*. Onder deze noemer vallen ook andere relevante problemen, zoals ruisverwijdering, superresolutie en beeldsynthese [1]. Nieuwe technieken die ontwikkeld worden voor inkleuring zijn dan ook vaak nuttig voor andere computervisie taken. Een van de grootste uitdagingen hier is dat de invoer geen textuur bevat. Onderzoek heeft namelijk uitgewezen dat neurale netwerken voornamelijk vertrouwen op textuur [2]. Wel is in stripverhalen het aantal verschillende objecten en kleuren een stuk beperkter dan in de echte wereld. Door deze eigenschap hopen we alsnog elk voorwerp te kunnen onderscheiden en eenduidig een kleur toe te kennen. Zo leent het probleem zich ertoe om meer inzicht te verwerven in het gebruik van neurale netwerken wanneer enkel vorm en geen textuur aanwezig is.

1.1 Probleemstelling

Concreet formuleren we ons probleem als het **automatisch genereren van een overeenkomstige, geloofwaardige kleurenversie op basis van een strippprent in binair zwart-wit**. Dit wordt schematisch weergeven in figuur 1.1. In deze definitie verdienen drie begrippen speciale aandacht: *automatisch*, *geloofwaardig* en *binair zwart-wit*.



Figuur 1.1: Schematische weergave van het probleem dat we trachten op te lossen.

Het eerste begrip, *automatisch*, duidt aan dat we geen extra invoer verwachten van de gebruiker naast de in te kleuren lijntekening. Waar voorgaand werk typisch een kleurhint vereist voor elk vlak, kiezen wij er bewust voor om dit niet te vereisen. Zo kan onze methode, na een voorafgaande trainingsfase, zeer snel en goedkoop toegepast worden op grote hoeveelheden strippprenten.

Het tweede begrip, *geloofwaardig*, verwijst naar de multimodaliteit die inherent is aan het probleem. Ieder kind dat ooit twijfelde tussen een groen of rood potlood om een appel in te kleuren weet dat een lijntekening typisch te weinig informatie bevat om met zekerheid één kleur te kiezen.

Het probleem is dus multimodaal in wezen aangezien er meerdere geloofwaardige inkleuringen kunnen zijn voor eenzelfde tekening. Er is niet noodzakelijk één correcte oplossing; meer nog kunnen sommige objecten een scala aan geloofwaardige kleuren aannemen. Hierdoor is het uitdagend om twee inkleuringen objectief te vergelijken. Traditionele afstandsfuncties en metrieken straffen namelijk inkleuringen af die geloofwaardig zijn maar verschillen van de referentie-inkleuring. Aangezien de geloofwaardigheid van een inkleuring bepalen bijna even moeilijk is als het inkleuren zelf, vereist het beoordelen van resultaten vaak tussenkomst van mensen.

Het laatste begrip, *binair zwart-wit*, duidt aan dat we als invoer een binaire afbeelding gebruiken waar elke pixel ofwel zwart ofwel wit is. Dit is een belangrijk verschil met het inkleuren van monochrome foto's, waar elke pixel typisch één van 256 mogelijke grijswaarden aanneemt. Dit is een weloverwogen keuze om onze methode breed inzetbaar te maken. Door als invoer enkele een binaire lijntekening te vereisen, zonder bijhorend grijswaardenkanaal, kunnen we namelijk tekeningen inkleuren die nooit eerder in kleur verschenen. Denk hierbij aan oude zwart-witalbums maar ook aan de aanvankelijk ongekleurde lijntekeningen in het productieproces van een strip.

1.2 Doel

In deze thesis werken we een methode uit die automatisch binaire lijntekeningen inkleurt. Om onze methode te trainen en te evalueren, maken we gebruik van prenten uit *Jommeke*. De uiteindelijke methode kan echter eenvoudig toegepast worden op andere stripreeksen door opnieuw te trainen met een geschikte dataset. Onze dataset bestaat uit pagina's uit Jommeke, een populaire Belgische stripreeks waarvan sinds 1955 meer dan 290 albums verschenen. Sinds 1979, met de uitgave van het 92^{e} album, verschijnen de verhalen in kleur in plaats van binair zwart-wit. Dit laat ons toe niet enkel te trainen en te testen op kleuralbums die we vooraf omzetten naar zwart-wit, maar ook op originele zwart-witalbums. Zo geven we deze zwart-witalbums een nieuwe dimensie en leren we hoe onze methode omgaat met albums in een andere, oudere tekenstijl.

Het geloofwaardig inkleuren van een strippprent vereist onder meer betrouwbare objectherkenning. Geïnspireerd door de verbluffende resultaten van neurale netwerken op onder andere deze taak, kiezen we ervoor deep learning te gebruiken. Vanwege de multimodaliteit van het probleem presteren de afstandsmetrieken die traditioneel als loss-functie ingezet worden echter niet goed. Daarom maken we gebruik van *deep convolutional generative adversarial networks* (DCGANs) [3]. Bij dit type netwerken wordt de loss-functie automatisch aangeleerd in plaats van manueel gespecificeerd, waardoor multimodaliteit niet langer een probleem vormt. Specifiek baseren we ons op pix2pix, een recent framework dat DCGANs inzet voor image-to-image translation [4]. Met pix2pix in het achterhoofd, ontwerpen we een eigen architectuur specifiek voor onze taak, waardoor de geloofwaardigheid van de inkleuringen drastisch toeneemt.

1.3 Opbouw

We beginnen in hoofdstuk 2 met een overzicht van gerelateerde problemen en bestaande oplossingen die mogelijk nuttig zijn voor ons. Aangezien de zonet aangehaalde DCGANs een geavanceerde techniek zijn, is enige voorkennis over deep learning sterk aanbevolen. In hoofdstuk 3 gaan we vervolgens dieper in op de keuze en verwerking van de dataset. Hierna beschrijven we onze uiteindelijke methode uitvoerig in hoofdstuk 4.

Daarna bespreken we in hoofdstuk 5 onze resultaten en enkele experimenten die meer inzicht verschaffen. Daarbij maken we voornamelijk gebruik van data afkomstig uit dezelfde distributie als de trainingsdata, zijnde kleurprenten die we eerst omzetten naar lijntekeningen. Dit laat ons toe om het netwerk te trainen en de resultaten te vergelijken met de ground truth. In hoofdstuk 6 proberen we onze methode uit op oudere albums die in zwart-wit verschenen en waarvoor dus geen ground truth beschikbaar is. Vervolgens vatten we in hoofdstuk 7 ons werk samen en geven we een aanzet naar eventueel toekomstig werk. Appendix A toont tenslotte de meest en minst geloofwaardige voorbeeldinkleuringen van onze methode op verschillende datasets.

Hoofdstuk 2

Literatuurstudie

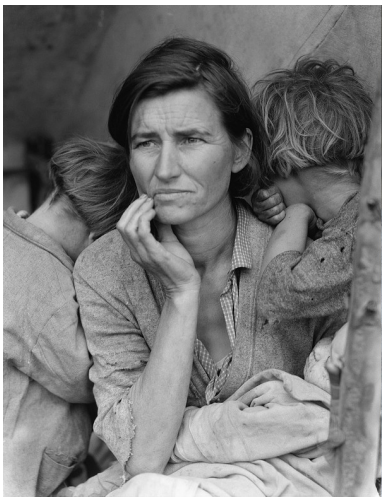
Specifiek over het volledig automatisch inkleuren van binaire stripafbeeldingen is er weinig tot geen literatuur beschikbaar. Voorgaand werk over het inkleuren van schetsen of lijnafbeeldingen focust zich hoofdzakelijk op het geval waar voor elk vlak een kleurhint beschikbaar is. Aangezien we er hier resoluut voor kiezen om dergelijke manuele kleurhints niet te vereisen, heeft dit werk slechts een beperkte relevantie voor ons. Daarom beginnen we met het identificeren van enkele andere, gerelateerde problemen waar reeds meer onderzoek naar verricht is. Vaak zijn er namelijk cruciale overeenkomsten met ons probleem, waardoor er een wisselwerking is tussen oplossingen voor deze problemen en het onze. Vervolgens evalueren we enkele bestaande oplossingen voor deze gerelateerde problemen en bepalen we in welke mate deze inzetbaar zijn voor ons probleem.

2.1 Gerelateerde problemen

In de literatuur vinden we drie concrete problemen die reeds uitgebreid bestudeerd zijn en opvallende gelijkenissen vertonen met het onze. Deze zijn het inkleuren van zwart-witfoto's, semantische segmentatie en image-to-image translation. We gaan nu dieper in op elk van deze problemen en het precieze verband met ons probleem.

2.1.1 Inkleuren van zwart-witfoto's

Pas midden de 20^{e} eeuw werd kleurenfotografie bekend bij het grote publiek, daarvoor waren foto's voornamelijk monochroom. Zoals geïllustreerd in figuur 2.1, spreken foto's in kleur nochtans een stuk meer tot de verbeelding. Een logische vraag is dan ook als het mogelijk is om zwart-witfoto's volledig automatisch om te zetten in kleur. Dit probleem is echter fundamenteel ondergespecificeerd. De invoer, zijnde een helderheidskanaal, bevat namelijk typisch te weinig informatie om de bijhorende kleurkanalen met zekerheid te reconstrueren. Gelukkig kunnen we vaak een doordachte gok wagen op basis van de informatie over textuur en vorm die beschikbaar is in de zwart-witfoto. Zo is gras typisch groen terwijl de hemel waarschijnlijk een blauwe kleur dient te krijgen. Hoewel deze aanpak soms erg indrukwekkend resultaten oplevert, zijn



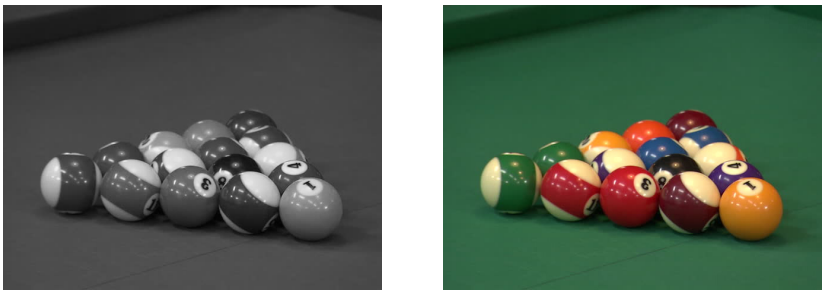
(a) Dorothea Lange, *Migrant Mother*, 1963

(b) Sanna Dullaway, *Handmatig ingekleurde versie van Migrant Mother*, 2012

Figuur 2.1: Voorbeeld van een historische zwart-witfoto die manueel ingekleurd werd.

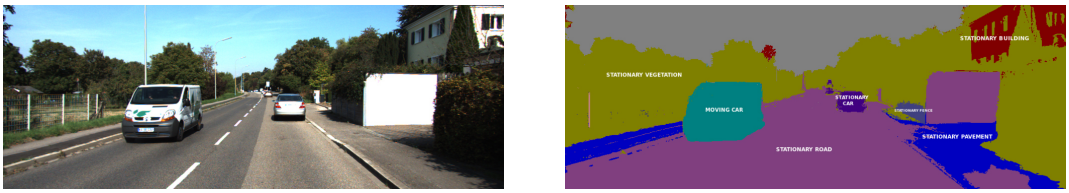
er eveneens tal van voorbeelden waar uitsluitend informatie over textuur en vorm niet volstaat. Denk bijvoorbeeld aan een foto van een aantal biljartballen die verschillend gekleurd zijn maar wel dezelfde textuur en vorm hebben (zie figuur 2.2). Zonder de oorspronkelijke kleurinformatie is het in dat geval niet mogelijk om met zekerheid te zeggen welke kleur elke bal dient te krijgen. Om aan deze ambiguïteit tegemoet te komen, wordt toegelaten om niet noodzakelijk de oorspronkelijke inkleuring te reconstrueren, zolang het een *geloofwaardige* inkleuring betreft.

Dit probleem vertoont op het eerste gezicht heel wat gelijkenissen met het onze. Zo willen we in beide gevallen een geloofwaardige inkleuring bekomen op basis van een invoer die te weinig informatie bevat om de originele inkleuring te reconstrueren. Toch zijn technieken voor het inkleuren van zwart-witfoto's niet noodzakelijk geschikt voor onze doeleinden. Zo bestaat een zwart-witfoto uit grijswaarden, terwijl de invoer in ons probleem binair is. Deze grijswaarden liggen echter aan de basis van heel wat bruikbare technieken aangezien ze informatie bevatten over de textuur (en dus mogelijke kleur) van oppervlakken. Naast het feit dat we geen gebruik kunnen maken van de informatie in dit helderheidskanaal, dienen we in ons geval dit kanaal ook zelf te reconstrueren. In plaats van twee kanalen te voorspellen op basis van één continu helderheidskanaal, dienen we drie kanalen te voorspellen op basis van één binair kanaal. Verder heeft de uitvoerafbeelding erg verschillende karakteristieken. Zo hebben we bij een kleurenfoto heel wat lokale variaties in de kleurtint van objecten. Bij een stripafbeelding willen we echter typisch grote oppervlakken één specifieke kleur toekennen. Hierdoor vallen kleine kleurafwijkingen veel sterker op in een strippprent dan in een foto en doen ze ook meer afbreuk aan het realisme.



(a) Zwart-witversie (b) Oorspronkelijke kleurversie

Figuur 2.2: Voorbeeld van een foto waarvan we de oorspronkelijke kleuren niet met zekerheid kunnen reconstrueren op basis van de zwart-witversie.



(a) Invoer: kleurenfoto (b) Uitvoer: semantisch label voor elke pixel

Figuur 2.3: Voorbeeld waarbij semantische segmentatie ingezet wordt om een verkeerssituatie te interpreteren. Bron: [7].

2.1.2 Semantische segmentatie

Een tweede verwant probleem is dat van semantische segmentatie [5]. Hierbij is de invoer typisch een kleurenfoto en trachten we aan elke pixel een label toe te kennen dat aanduidt welk type object zich op die positie bevindt. Deze informatie is erg nuttig om de inhoud van een afbeelding te begrijpen en kent dan ook heel wat toepassingen. Denk hierbij onder andere aan autonome voertuigen die alle objecten in hun omgeving correct dienen te classificeren (zie figuur 2.3). Maar ook biomedische toepassingen, waarbij bijvoorbeeld elk orgaan op een röntgenfoto automatisch herkend wordt, behoren tot de mogelijkheden [6].

Net als bij het inkleuren van zwart-witfoto's, dienen we ook hier laag-niveau informatie, zijnde textuur, met hoog-niveau informatie, zijnde vorm, te combineren om objecten te identificeren. We kunnen deze segmentatie dan ook zien als een soort van inkleuring waarbij de kleuren semantische labels zijn. Een belangrijk verschil tussen dit probleem en het inkleuren van zwart-wit foto's is echter dat we nu veronderstellen dat de invoer wel genoeg informatie bevat om elke pixel correct te classificeren. De notie van een 'geloofwaardige' uitvoer ontbreekt dus aangezien we louter geïnteresseerd zijn in de werkelijke klasse waartoe een pixel behoort.

2.1.3 Image-to-image translation

Traditioneel werden voor de twee bovenvermelde problemen telkens verschillende, gespecialiseerde technieken ontwikkeld en aangewend. We merkten echter reeds enkele opvallende gelijkenissen op. Meer nog, we kunnen deze beide problemen en ons probleem alle drie beschouwen als specifieke gevallen van het 'vertalen' van een invoerafbeelding naar een uitvoerafbeelding. Net zoals een concept uitgedrukt kan worden in het Engels of Nederlands, kan een scene voorgesteld worden als een kleurfoto, een zwart-witfoto, een schets, een afbeelding met semantische labels enzovoort. Als we de semantiek van de invoer en uitvoer buiten beschouwing laten, is de taak telkens om pixels te voorspellen op basis van pixels.

Op basis van recente doorbraken in deep learning en GANs, ontwikkelden Isola et al. een algemeen raamwerk voor zogenaamde *image-to-image translation* [4]. Elk van de voorgaande problemen reduceert zich dan tot het specificeren van het aantal invoer- en uitvoerkanalen en het trainen op een geschikte dataset. Verder blijft de gebruikte oplossing telkens gelijk en dienen we geen probleemspecifieke aanpassingen aan te brengen. Het grote voordeel van deze aanpak is zijn algemene inzetbaarheid. Concreet hoeven we bijvoorbeeld geen gespecialiseerde loss-functies op te stellen aangezien deze automatisch wordt geleerd. Zo wordt het ook mogelijk om problemen aan te pakken waarvoor het moeilijk is om manueel een geschikte loss-functie te specificeren. Denk hierbij bijvoorbeeld aan het omzetten van nachtfoto's naar foto's die overdag gemaakt lijken of het omzetten van semantische labels naar een realistische afbeelding. Deze aanpak is ook erg relevant voor ons probleem. Het is immers niet vanzelfsprekend handmatig een loss-functie op te stellen die een geloofwaardige inkleuring afdwingt. Hoewel ons probleem meer algemeen formuleren op het eerste gezicht erg aantrekkelijk lijkt, heeft deze aanpak toch enkele nadelen. Zo is het onderzoek omtrent image-to-image translation vooralsnog beperkt en zijn er nog heel wat open vragen [8]. Concreet is de gebruikte methode berucht omwille van een instabiele training en hoge computationele kost. Voordat we hier dieper op ingaan, bespreken we eerst enkele andere succesvolle oplossingen voor de zonet besproken verwante problemen. Op die manier kunnen we enkele technieken identificeren die ook nuttig zijn voor ons probleem.

2.2 Image Analogies

De eerste bruikbare methodes voor het omzetten van zwart-witfoto's naar kleur waren telkens *niet-parametrisch*. Dit betekent dat er geen expliciet model getraind wordt. In plaats daarvan dient de gebruiker één of meer kleurfoto's, die voldoende overeenkomsten vertonen met de te kleuren zwart-witfoto, te selecteren. Vervolgens wordt voor elke pixel in de invoer een corresponderende pixel geselecteerd in deze referentieafbeeldingen op basis van het Image Analogies framework [9]. Concreet wordt er gebruik gemaakt van de textuur (grijswaarden) van het omringende gebied om referentiepixels te selecteren met een gelijkaardige omringende textuur. Hierna wordt de kleur van deze corresponderende pixel in een referentieafbeelding overgedragen naar de originele pixel in de zwart-witfoto [10, 11, 12].

Hoewel met deze techniek mooie resultaten behaald werden op sommige afbeeldingen, zijn er toch een aantal belangrijke beperkingen. De belangrijkste beperking voor ons is het feit dat voor deze kleuroverdracht uitsluitend gebruikt wordt gemaakt van laag-niveau textuur informatie [13]. Het ontbreken van elke notie van hoog-niveau semantiek beperkt de performantie en inzetbaarheid drastisch. Zo dient er telkens een object met vergelijkbare textuur in de referentieafbeeldingen aanwezig te zijn om de uitvoerkleur te bepalen. De benodigde referentieafbeeldingen dienen dus zorgvuldig gekozen te worden voor elke invoerafbeelding. Aangezien onze invoerafbeeldingen binair zwart-wit zijn en nauwelijks textuur bevatten, is deze techniek niet toepasbaar op het gestelde probleem.

2.3 Deep Learning

De performantie van de voorgaande methode blijft beperkt door het feit dat er enkel gebruik wordt gemaakt van laag-niveau textuur. Een methode die toelaat om ook hoog-niveau semantiek in rekening te brengen is het gebruik van een diep neuraal netwerk. Dergelijke netwerken kunnen automatisch leren om laag-niveau features samen te voegen tot hoog-niveau features en zo een hiërarchie van features te leren. Hierdoor was deep learning in staat verbluffende resultaten te behalen op diverse taken, waaronder inkleuring [14, 15, 16, 17, 18]. Dergelijke hoog-niveau semantische features zijn van buitengewoon belang in ons geval aangezien deze, vanwege het ontbreken van textuur in de invoer, de enige bruikbare kenmerken zijn om de inkleuring te sturen. Gebaseerd op deze observatie en de drastische verbetering ten opzichte van eerdere technieken [18], zullen we ons vanaf nu hoofdzakelijk focussen op methodes gebaseerd op deep learning. Typisch wordt voor inkleuring en semantische segmentatie een zogenaamd *convolutional neural network* (CNN) gebruikt [19]. In wat volgt gaan we dieper in op drie recente CNN-architecturen voor segmentatie: Fully Convolutional Networks (FCN) [20], U-Net [21] en SegNet [22]. Vervolgens beschrijven we *dilated convolutions* [23], een uitbreiding op de gewone convolutie die de performantie voor ons probleem kan verbeteren.

2.3.1 Fully Convolutional Networks

Eén van de eerste toepassingen van deep learning was het classificeren van afbeeldingen [19]. Traditioneel doorloopt de invoerafbeelding hierbij een opeenvolging van drie types lagen: convolutionele lagen, niet-lineaire lagen en pooling lagen. Deze lagen worden tenslotte gevolgd door gevolgd door één laatste volledig-geconnecteerde laag. De uitvoer is dan een k-dimensionale vector, waarbij elk element de probabiliteit voorstelt dat de invoerafbeelding tot één van de k klassen behoort. Na deze doorbraak was de volgende uitdaging om meerdere objecten per afbeelding te herkennen en te lokaliseren door voor elke pixel een klasse te voorspellen. Dit is het probleem van semantische segmentatie dat eerder beschreven werd en nauw verwant is met ons probleem.

Een naïeve aanpak is om voor elke pixel een patch uit de invoerafbeelding te snijden met deze pixel

als centrum. Deze patch wordt vervolgens als invoer gegeven aan een traditioneel classificatienetwerk [24]. Een dergelijke sliding window aanpak heeft echter twee nadelen. Ten eerste is de computationele kost erg hoog aangezien het netwerk volledig opnieuw geëvalueerd wordt voor elke pixel, terwijl er nochtans heel wat overlap is tussen de patches. Ten tweede dient er een compromis gemaakt te worden tussen de localisatienauwkeurigheid en de beschikbare context. Grotere patches vereisen meer max-pooling lagen waardoor de localisatienauwkeurigheid afneemt, terwijl kleinere patches het netwerk minder in staat stellen rekening te houden met de omliggende context.

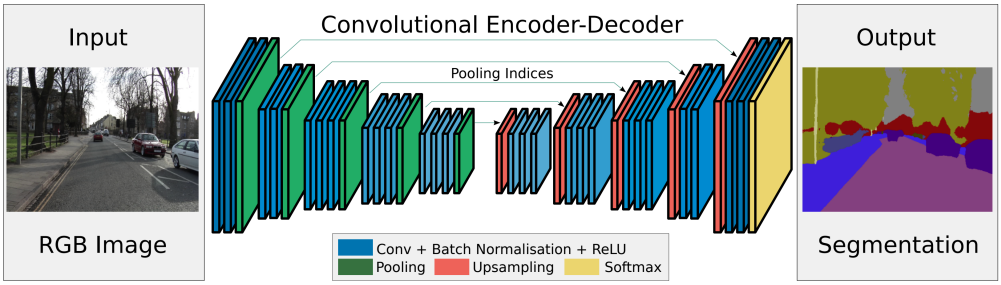
Een betere oplossing is het gebruik van een *fully convolutional network*, een netwerkstructuur waarbij enkel convolutionele lagen gebruikt worden [20]. Concreet vervangen we de laatste volledig geconnecteerde laag door een sequentie van *upconvolutions*, ook wel bekend als *transposed convolutions* of *fractionally-strided convolutions* [5]. In tegenstelling tot een gewone convolutie, gebruikt een upconvolutie zijn kernel niet om naburige waarden samen te vatten maar wel om waarden te verspreiden over een groter gebied. Door een opeenvolging van dergelijke resolutieverhogende lagen kunnen we de resolutie van de feature maps herstellen tot de invoerresolutie van het netwerk.

Door deze aanpassing is de uitvoer van het netwerk niet langer één probabiliteitsvector, maar een twee-dimensionale afbeelding met een probabiliteitsvector per pixel. Zo bekomen we voor elke pixel een predictie met beperkte computationele kost aangezien de activaties binnen het netwerk gedeeld worden tussen verschillende pixels. Een ander voordeel is dat het netwerk nu werkt op invoerafbeeldingen van eender welke resolutie, waarbij de uitvoer van het netwerk telkens dezelfde verhoudingen heeft als de invoer en één kanaal voor elke mogelijke klasse. Deze netwerkstructuur kan dus nauwkeurige lokale voorspellingen maken door middel van upsampling en eveneens rekening houden met de globale context door middel van downsampling. De architectuur kan bovendien eenvoudig aangepast worden om afbeeldingen in te kleuren in plaats van te segmenteren. Hiertoe wijzigen we de laatste laag aan zodat deze een kleurwaarde per pixel teruggeeft in plaats van een probabiliteitsvector.

2.3.2 SegNet

De hierboven besproken architectuur, waarbij eerst meermaals gedownsampled wordt en vervolgens opnieuw geupsampled, staat bekend als een *encoder-decoder* (zie figuur 2.4). Hoewel de resultaten van een dergelijke architectuur erg bemoedigend zijn, is de uitvoer vaak nog steeds grover dan gewenst. Dit is hoofdzakelijk te wijten aan het feit dat de opeenvolgende downsampling operaties de resolutie van de tussenliggende feature maps drastisch reduceren. Hierdoor gaat veel spatiale informatie, die nodig is om objecten scherp af te bakenen, verloren. Bij *SegNet* [22] wordt dit probleem opgevangen door bij het upsamplen gebruik te maken van de indices die de elementen leverden bij de corresponderende max-pooling operaties. Concreet worden de waarden bij het upsamplen geplaatst op de posities waar het maximum zich bevond bij het

downsamplen. De andere elementen worden op nul gezet. De tussenliggende uitvoer is zo een sparse feature map die vervolgens door één of meerdere convolutionele lagen wordt omgezet naar een dense feature map. In de praktijk verbetert deze aanpassing de granulariteit van de uitvoer drastisch met slechts een minimale extra geheugenkost.

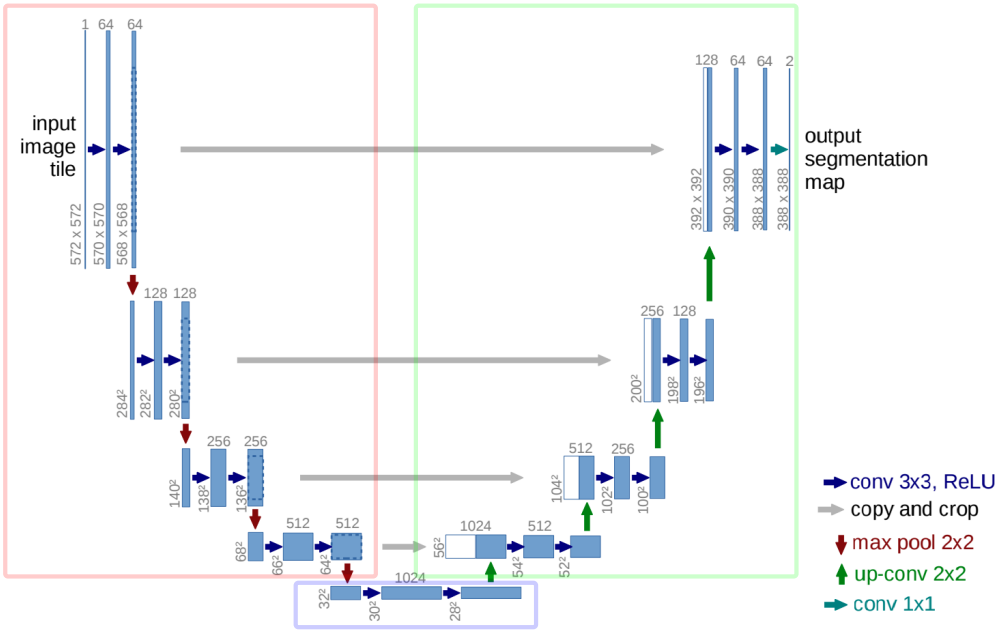


Figuur 2.4: De SegNet architectuur. In de decoder worden de pooling indices vanuit de encoder gebruikt om een sparse feature map to produceren. De hierop volgende convoluties zetten deze vervolgens om naar een dense feature map. Bron: [22]

2.3.3 U-Net

Naast het doorgeven van pooling indices, zoals hierboven geschetst, zijn er nog enkele andere mogelijkheden om te zorgen dat de spatiale informatie die verloren ging bij het downsamplen kan teruggewonnen worden bij het upsamplen. Een populair alternatief is het gebruik van een zogenaamd *U-Net* [21]. Het basisidee is een symmetrisch fully convolutional netwerk waar de feature maps die gegenereerd worden tijdens het downsamplen, geconcateneerd worden met de feature maps van de overeenkomstige upsampling stap. Deze verbindingen tussen het downsampling en upsampling pad worden *skip connections* genoemd. Ze zorgen ervoor dat spatiale informatie vlot doorheen het netwerk kan overgebracht worden. Op die manier kan het netwerk op een flexibele manier lokale informatie combineren met globale informatie tijdens het upsamplen.

Deze architectuur heeft zijn naam te danken aan de U-vorm, die bestaat uit drie delen, zoals weergeven op figuur 2.5. Het eerste deel bestaat uit het downsampling pad en heeft als voornaamste doel de resolutie te verlagen zodat het netwerk meer context te zien krijgt (links op de figuur in het rood). Dit wordt gevolgd door een bottleneck module (onderaan op de figuur in het blauw). Het laatste deel is is het upsampling pad (rechts op de figuur in het groen). Hierbij wordt telkens een feature map van lagere resolutie, die globale context biedt, samengevoegd met de uitvoer van de corresponderende downsampling stap, waardoor precieze lokalisatie mogelijk wordt.



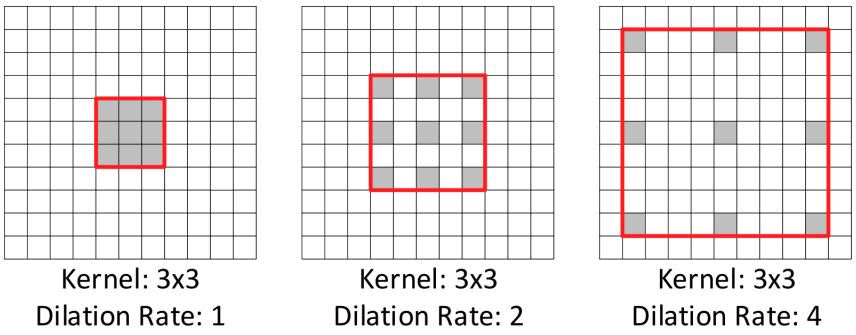
Figuur 2.5: Architectuur van een U-Net. Bron: [21]

2.3.4 Dilated convolutions

In zowel de encoder-decoder als het U-Net starten we met een opeenvolging van downsampling lagen. De bedoeling hiervan is om de hoeveelheid context te verhogen die beschikbaar is dieper in het netwerk. Deze regio van de invoer die invloed kan hebben op de uitvoer op een bepaalde locatie, wordt het *receptive field* genoemd. Specifiek voor ons probleem is het belangrijk dat dit receptive field zo groot mogelijk is. Aangezien er geen textuur beschikbaar is in de invoer, kunnen we namelijk enkel vertrouwen op de vorm van lijnen die typisch relatief ver van elkaar liggen. Door te downsamplen kunnen we het receptive field vergroten, maar gaat ook spatiale informatie verloren aangezien de resolutie afneemt. Deze informatie dient dan weer teruggewonnen te worden door de skip connections (U-Net) of unpooling met de maximumindices (SegNet).

Idealiter willen we het receptive field drastisch vergroten en tegelijkertijd de oorspronkelijke resolutie bewaren. En dat is nu net de premisse van *dilated convolutions*, ook wel *convolutions à trous* genoemd [23]. De basisobservatie is het feit dat de kernelgrootte van een convolutie verhogen, het receptive field vergroot én de resolutie behoudt, maar wel erg duur is vanwege het kwadratisch toenemen van het aantal bewerkingen en parameters. Het idee is dan ook om lege plaatsen te introduceren tussen de elementen van de kernel, zoals op figuur 2.6. Deze lege plaatsen of gaten leiden tot de benaming *à trous*.

Stel dat $F : \mathbb{Z}^2 \to \mathbb{R}$ een discrete functie is die een feature map voorstelt. Stel vervolgens $\Omega_r = [-r, r]^2 \cap \mathbb{Z}^2$ en stel $K : \Omega_r \to \mathbb{R}$ een discrete filter met grootte $(2r + 1)^2$. Dan kunnen we



Figuur 2.6: Voorbeeld van hoe er lege plaatsen lijken te zitten tussen de filterparameters bij een dilated convolution. Bron: [25]

de l-dilated convolutie, die een punt $\mathbf{p} \in \mathbb{Z}^2$ afbeeldt op het resulterende element, definiëren als volgt:

$$(F *_l K)(\mathbf{p}) = \sum_{\mathbf{s}+l\mathbf{t}=\mathbf{p}} F(\mathbf{s})K(\mathbf{t}) \tag{2.1}$$

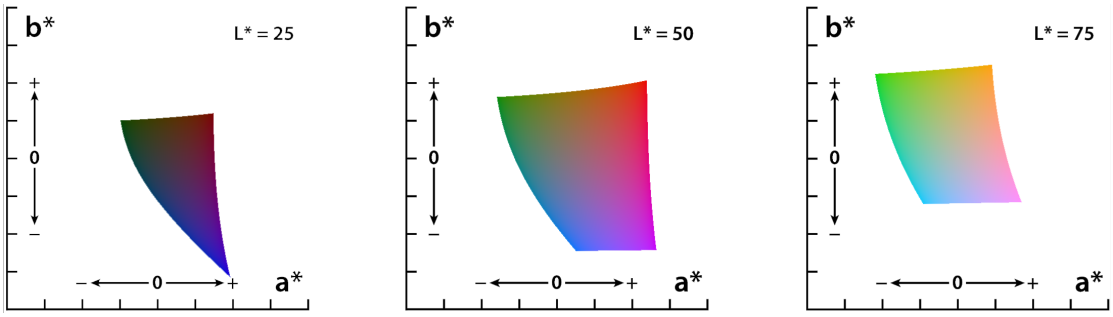
De bekende discrete convolutie is dan simpelweg de 1-dilated convolutie. Bovenstaande definitie is belangrijk om in te zien dat er nooit een 'gedilateerd filter' wordt geconstrueerd. De convolutie-operator wordt namelijk zelf aangepast zodat de filterparameters op een andere wijze worden gebruikt. Hierdoor kunnen we meer context aggregeren zonder toename van de computationele kost. Door meerdere dilated convoluties na elkaar te plaatsen, kunnen we het receptive field exponentieel vergroten zonder in te boeten aan resolutie of performantie. Aangezien deze convolutions de resolutie behouden, kunnen we ze eenvoudig inbouwen tussen de traditionele lagen in de encoder-decoder en het U-Net [26]. Op die manier bekomen we een nog groter receptive field in die architecturen.

2.4 Loss-formulering

De zojuist geziene architecturen en technieken laten ons toe om krachtige neurale netwerken op te stellen die bruikbaar zijn voor zowel het inkleuren van zwart-witfoto's als voor semantische segmentatie. Naast de architectuur van het netwerk is een andere cruciale factor het definiëren van een geschikte loss-functie om het netwerk mee te trainen. In de komende sectie zullen we zien dat dit hier een stuk gecompliceerder is dan het op het eerste gezicht lijkt.

2.4.1 CIELAB-kleurruimte

Voordat we een loss-functie kunnen formuleren, dienen we even stil te staan bij hoe we kleur modelleren in ons probleem. Er zijn namelijk verschillende kleurruimtes die elk hun eigen voor- en nadelen hebben. Een weloverwogen keuze van de kleurruimte is dan ook belangrijk om optimale resultaten te bekomen. De meest populaire kleurruimte is ongetwijfeld RGB, waarbij er



Figuur 2.7: De CIELAB kleurruimte, waarbij voor verschillende waarden van L de kleuren worden afgebeeld die binnen het kleurengamma van de sRGB kleurruimte vallen. Bron: [27]

drie kanalen zijn die elk de hoeveelheid licht voorstellen die nodig is om een welbepaalde kleur te reproduceren. In eerder werk omtrent inkleuring, wordt echter voornamelijk de CIELAB kleurruimte (ook bekend als CIE L*a*b* of simpelweg Lab) gebruikt. In dit systeem wordt kleur voorgesteld als een punt in een driedimensionale ruimte met cordinaten $L*$, $a*$ en $b*$ (zie figuur 2.7). Hierbij staat $L*$ voor de helderheid van zwart tot wit, varieert $a*$ van groen tot rood en $b*$ van blauw tot geel.

Werken met CIELAB voor inkleuring heeft twee grote voordelen. Een eerste voordeel is dat de aanwezigheid van een helderheidskanaal ($L*$) erg praktisch is. Bij zwart-witfotografie, kunnen we namelijk de oorspronkelijke grijswaardenafbeelding gebruiken als het $L*$ kanaal. Het model hoeft daarna enkel nog de $a*$ en $b*$ kanalen te voorspellen. Deze eigenschap is minder van belang voor ons probleem aangezien de invoer een binaire afbeelding is in plaats van een grijswaardenafbeelding. We hebben het $L*$ kanaal dus niet ter beschikking in de invoer en dienen alsnog drie kanalen te voorspellen. Een tweede voordeel, dat wel nuttig is voor onze doeleinden, is dat deze kleurruimte specifiek ontworpen is voor zogenaamde *perceptuele uniformiteit*. Dit betekent dat dezelfde hoeveelheid numerieke verandering ongeveer overeenkomt met dezelfde hoeveelheid visueel waargenomen verandering. Dit is een erg wenselijke eigenschap aangezien ze ons toelaat metrieken op te stellen die direct verband houden met de afwijking waargenomen door mensen. Daarom kiezen we ervoor gebruik te maken van de CIELAB kleurruimte bij het opstellen van onze loss-functies.

2.4.2 L_p-loss

Na het kiezen van een kleurruimte, kunnen we het inkleuren van een afbeelding formeel definiëren. Deze definitie is vervolgens nuttig om de loss-functies beschrijven. Concreet definiëren we het probleem van het inkleuren van een zwart-witfoto als volgt: Gegeven een helderheidskanaal $\mathbf{x} \in \mathbb{R}^{H \cdot W}$, is het doel om een functie $\hat{\mathbf{y}} = G(\mathbf{x})$ te leren met als uitvoer de drie-kanaals afbeelding $\hat{\mathbf{y}} \in \mathbb{R}^{H \cdot W \cdot 3}$, waarbij H en W respectievelijk de hoogte en breedte van de afbeelding voorstellen.

Bij het inkleuren van een zwart-witfoto kan het helderheidskanaal rechtstreeks overgedragen worden van de invoer naar de uitvoer. In ons geval is de invoer echter een binaire afbeelding $\mathbf{x} \in \{0,1\}^{H \cdot W}$ waardoor we drie in plaats van twee kanalen dienen voorspellen. De principes achter de besproken loss-functies blijven echter dezelfde en zijn onafhankelijk van het precieze aantal kanalen.

Een voor de hand liggende optie is om gebruik te maken van de afstand tussen de voorspelde waardes $\hat{\mathbf{y}}$ en de verwachte waardes $\mathbf{y}$. Gegeven een reëel getal $p \geq 1$, kunnen we een loss-functie L_p definiëren als volgt:

$$\mathrm{L}_p(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{3HW} \sum_i |\mathbf{y}_i - \hat{\mathbf{y}}_i|^p \tag{2.2}$$

Door een geschikte waarde voor p te kiezen, bekomen we enkele populaire loss-functies. Zo leidt $p = 1$ tot de L_1-loss, ook bekend als de mean absolute error (MAE). Voor $p = 2$, bekomen we de L_2-loss, ook bekend als de mean squared error (MSE). Deze traditionele loss-functies zijn echter niet robuust ten opzichte van de ambiguïteit en multimodaliteit die inherent is aan het probleem [28]. Beschouw bijvoorbeeld een object dat telkens één specifieke kleur uit een beperkte verzameling van kleuren heeft, zoals een biljartbal (zie sectie 2.1.1). In dat geval is het optimum volgens de L_2-loss de kleur die het gemiddelde van deze verzameling is, wat leidt tot grauwe en gedesatureerde kleuren.

Dit fenomeen wordt geïllustreerd in figuur 2.8. De trainingsdata bestaat daar uit de afbeeldingen a, b en c. Elk van deze afbeeldingen heeft exact hetzelfde grijswaardenkanaal, zijnde afbeelding d. De optimale inkleuring volgens de L_2-norm is in dit geval afbeelding e, waarbij de kleuren het gemiddelde zijn van afbeeldingen a, b en c. Deze inkleuring is echter niet realistisch aangezien biljartballen felle kleuren hebben en niet de grauwe kleur die we hier zien. Een realistischere inkleuring zou zijn om resoluut één van de drie kleuren te kiezen in plaats van de uitmiddeling die inherent is aan de L_2-loss.

2.4.3 Cross-entropy loss

Een mogelijke oplossing voor het tegengaan van de gedesatureerde kleuren die samengaan met de L_2-loss is het formuleren van het probleem als multinomiale classificatie [28]. Merk op dat we zo het probleem kunnen zien als een soort semantische segmentatie. Concreet wordt de Lab-kleurenruimte gekwantiseerd door deze onder te verdelen in een vast aantal regio's. Voor een gegeven invoer $\mathbf{x}$, leren we nu een functie $\hat{\mathbf{q}} = G(\mathbf{x})$ naar een probabiliteitsverdeling over mogelijke kleuren $\hat{\mathbf{q}} \in [0,1]^{H \cdot W \cdot K}$, waarbij er K gekwantiseerde waarden zijn. Om de voorspelde waarden $\hat{\mathbf{q}}$ te vergelijken met de referentie-waarden, definiëren we een kwantisatiefunctie $\mathbf{q} = Q(\mathbf{y})$ die de verwachte kleuren $\mathbf{y}$ omzet naar een vector $\mathbf{q}$. Vervolgens kunnen we de traditionele cross-entropy loss gebruiken:

(a) Kleurenafbeelding 1 in trainingsdata

(b) Kleurenafbeelding 2 in trainingsdata

(c) Kleurenafbeelding 3 in trainingsdata

(d) Grijswaardenversie van afbeelding (a), (b) en (c)

(e) L_2-optimum voor inkleuring van afbeelding (d)

Figuur 2.8: Voorbeeld waarbij het L_2-optimum (het gemiddelde van alle kleuren in de trainingsdata) niet leidt tot een geloofwaardige inkleuring.

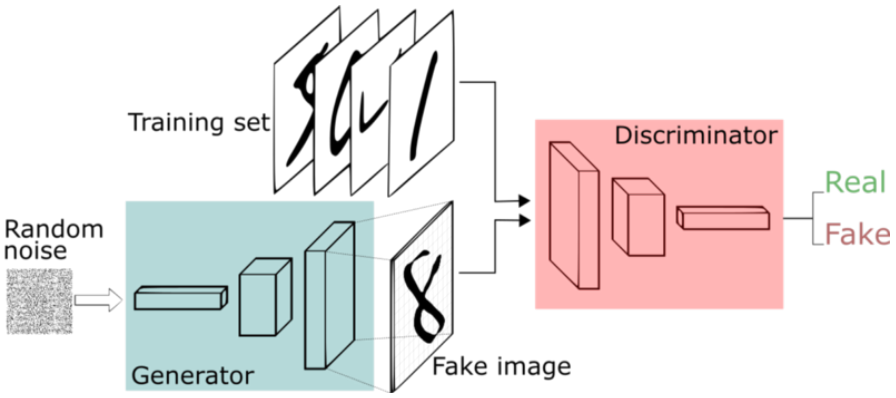
$$L_{ce}(\hat{\mathbf{q}}, \mathbf{q}) = -\sum_i \mathbf{q}_i \log(\hat{\mathbf{q}}_i) \tag{2.3}$$

Hoewel deze loss-functie typisch resulteert in veel levendigere kleuren, duiken er ook enkele nieuwe uitdagingen op. Een eerste probleem is dat bepaalde kwantisatiewaarden typisch veel meer voorkomen dan andere en hierdoor de loss-functie gaan overheersen. Dit probleem werd opgelost door Zhang et al. [28] door de bijdrage van elke kwantisatieregio tot de loss-functie te normaliseren. Een meer fundamenteel probleem is het feit dat de uitvoer van het netwerk nu beperkt is tot K discrete waarden. Doordat de loss voor elke pixel apart berekend wordt en er geen notie van continuïteit is, treden er in de praktijk vaak bruuske en storende kleurovergangen op middenin een object. Meer nog heeft het netwerk niet eens de mogelijkheid om vloeiende kleurovergangen, zoals die af en toe voorkomen in strips, als uitvoer te genereren. Tenslotte heeft het discretiseren van de uitvoer ook een negatief effect op de trainingstijd. Door elk kleur onder te brengen in één klasse heeft het netwerk immers geen notie van afstand meer, wat resulteert in tragere convergentie.

2.5 Generative adversarial networks

In de voorgaande sectie is gebleken dat het, onder andere vanwege het multimodale karakter van het probleem, lastig is om een effectieve loss-functie op te stellen. De meest betrouwbare metriek is menselijke evaluatie, maar het is uiteraard niet mogelijk het netwerk te trainen op basis hiervan. Een erg wenselijk scenario zou zijn dat we een soortgelijk abstract doel kunnen opleggen, zoals "gegenereerde inkleuringen mogen niet te onderscheiden zijn van handmatig ingekleurde afbeeldingen", waarna er automatisch een geschikte loss-functie wordt opgesteld. Gelukkig is dit precies wat generative adversarial networks (GANs) beogen [29, 30, 3, 31].

2.5.1 Basisidee



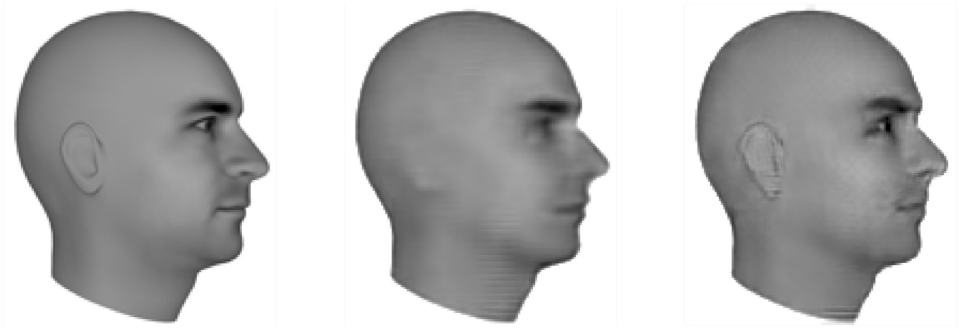
Figuur 2.9: Schematische voorstelling van het Generative Adversarial Networks framework. Bron: [32]

Het basisidee in het adversarial nets framework is om het generatieve model (d.w.z. het inkleurende netwerk) het te laten opnemen tegen een tegenstander: een discriminatief model dat leert om samples (d.w.z. inkleuringen) gegenereerd door het model te onderscheiden van echte samples [29]. Een veelgebruikte analogie is om het generatief model te zien als een valsmunter die voortdurend tracht om valse valuta te produceren en ongemerkt te gebruiken. Het discriminator model is dan analoog aan de politie die probeert om deze valse valuta te onderscheiden van echte. De competitie in dit scenario lijdt ertoe dat beide partijen hun technieken verbeteren tot de valse valuta nauwelijks nog te onderscheiden valt van echte valuta.

In bovenstaande informele redenering komt het er in feite op neer dat de generator G een minimax spel speelt tegen de discriminator D met de volgende waardefunctie:

$$\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_{\mathbf{y}}[\log D(\mathbf{y})] + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))] \tag{2.4}$$

Herbij stelt $\mathbf{y}$ een sample uit de echte distributie voor, terwijl $\mathbf{z}$ een vector is die bestaat uit willekeurige ruis om de generator te verhinderen telkens hetzelfde sample te genereren. De generator



(a) Ground truth

(b) Predictie van model getraind met L_2-loss. De uitvoer is een gemiddelde van alle mogelijke volgende frames. Hierdoor worden de ogen wazig en verdwijnen de oren.

(c) Predictie van model getraind met L_2-loss en GAN-loss. Het netwerk leert dat er meerdere mogelijke volgende frames zijn en kiest één om een scherpe en realistische afbeelding te bekomen.

Figuur 2.10: Voorbeeld van de invloed van een GAN-loss bij problemen met een multi-modale uitvoer. Bron: [33]

G probeert dan deze waardefunctie te minimaliseren tegenover een tegenstander D die ze probeert te maximaliseren. De optimale generator is dan $G^* = \text{argmin}_G \max_D \mathcal{L}_{\text{GAN}}(G, D)$. Deze generator produceert samples $G^*(\mathbf{z})$ die exact dezelfde verdeling volgen als de verdeling waar een echt sample $\mathbf{y}$ uit afkomstig is. In de praktijk implementeren we dit spel door telkens afwisselend D en G te optimaliseren. Onder bepaalde theoretische voorwaarden kunnen we aantonen dat de verdeling van de samples voortgebracht door de generator convergeert naar de echte verdeling. Dit spel is schematisch voorgesteld in figuur 2.9.

Het grote voordeel van deze aanpak is dat het niet langer nodig is om manueel een loss-functie te specificeren. Door het bovenstaande spel te spelen, gaat de generator na enige tijd een grote diversiteit aan samples genereren die niet te onderscheiden zijn van echte samples. GANs kunnen eveneens overweg kunnen met een multimodale uitvoer, waarbij eenzelfde invoer kan leiden tot verschillende correcte uitvoeren. Zoals eerder aangehaald, is dit iets waar traditionele trainingsmethodes, zoals de L_2-loss minimaliseren, moeite mee hebben. Een goed voorbeeld hiervan is het voorspellen van het volgende frame in een video [33]. In dit geval bestaat de trainingsdata uit een roterend 3D-model van een hoofd. Afbeelding 2.10a is de ground truth en dus wat het model idealiter zou voorspellen. Afbeelding 2.10b toont de uitvoer van het model na training met de L_2-loss. Deze afbeelding is het gemiddelde van alle mogelijke opvolgerframes, wat leidt tot een wazig en onrealistisch resultaat. Afbeelding 2.10c is de uitvoer van het model na training met de L_2-loss gecombineerd met de GAN-loss. Deze keer is het netwerk in staat rekening te houden met de multimodaliteit van het probleem, waardoor de uitvoer nu wel scherp en realistisch is.

2.5.2 pix2pix

De voorgaande sectie gaf reeds een voorproefje van de kracht van GANs. De toepassingen zijn dan ook praktisch eindeloos. Zo zijn GANs reeds succesvol ingezet voor een grote diversiteit aan taken: gaande van face-aging, tot super-resolutie of zelfs tekst-naar-afbeelding generatie [34, 35, 36]. Een indrukwekkende en voor ons relevante toepassing is afkomstig van Isola et al. [4]. Daarbij worden conditionele GANs ingezet als algemene oplossing voor image-to-image translation (zie sectie 2.1.3). Zo wordt het mogelijk om met exact dezelfde architectuur en trainingsprocedure luchtfoto's om te zetten naar kaarten, dag om te zetten naar nacht, foto's in te kleuren enzovoort (zie figuur 2.12). Aangezien onze uiteindelijke oplossing geïnspireerd is door de pix2pix architectuur, wordt er nu dieper ingegaan op de belangrijkste ontwerpbeslissingen ervan.

Objectief

In sectie 2.5.1 zagen we reeds de waardefunctie van een onconditioneel GAN (d.w.z. een GAN met enkele random noise als invoer). Deze waardefunctie is makkelijk aan te passen om de uitvoer te conditioneren op de invoer, wat exact is wat we willen bij image-to-image translation. Op die manier bekomen we het objectief van een conditioneel GAN (cGAN), dat leert om een geobserveerde afbeelding $\mathbf{x}$ en ruisvector $\mathbf{z}$ af te beelden op een uitvoer $G(\mathbf{x}, \mathbf{z})$. In ons geval is $\mathbf{x}$ de originele lijntekening en is de uitvoer van de generator $G(\mathbf{x}, \mathbf{z})$ de corresponderende inkleuring. Zowel de generator G als de discriminator D hebben nu de originele invoer $\mathbf{x}$ als extra argument:

$$\mathcal{L}_{\text{cGAN}}(G, D) = \mathbb{E}_{\mathbf{x},\mathbf{y}}[\log D(\mathbf{x}, \mathbf{y})] + \mathbb{E}_{\mathbf{x},\mathbf{z}}[\log(1 - D(\mathbf{x}, G(\mathbf{x}, \mathbf{z})))] \tag{2.5}$$

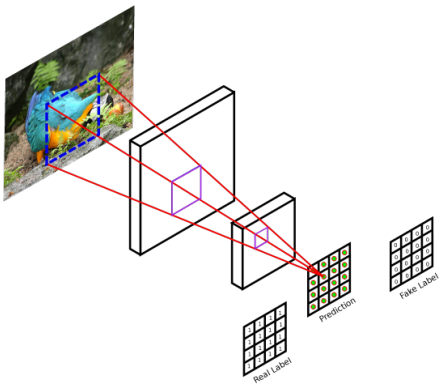
In de praktijk blijkt de generator echter te leren om de vector van willekeurige ruis $\mathbf{z}$ te negeren en enkel samples te genereren op basis van $\mathbf{x}$ [37]. Daarom wordt de ruisvector $\mathbf{z}$ typisch achterwege gelaten en wordt ruis aangeleverd in de vorm van dropout in zowel de training als de test fase.

Verder blijkt het nuttig om deze GAN-loss te combineren met een meer traditionele loss-functie, zoals de L_2-loss [38]. De taak van de discriminator blijft hierdoor onveranderd, maar de uitvoer van de generator dient nu niet enkel de discriminator te misleiden maar ook dicht bij de ground truth te liggen. Deze aanpassing leidt tot een stabielere training. Experimenteel is gebleken dat de L_1-loss typisch tot visueel aantrekkelijkere resultaten leidt dan de L_2-loss:

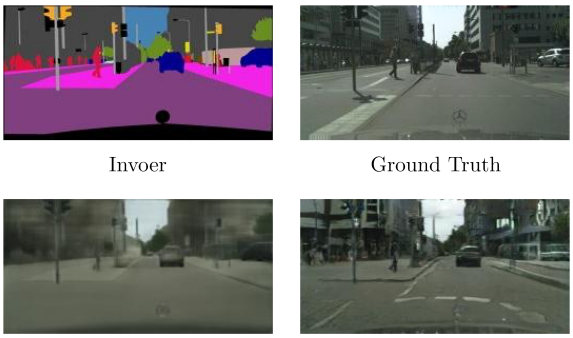
$$\mathcal{L}_{\text{L1}}(G) = \mathbb{E}_{\mathbf{x},\mathbf{y}}[\text{L}_1(\mathbf{y}, G(\mathbf{x}))] \tag{2.6}$$

Het uiteindelijke objectief, waarbij λ de afweging bepaalt tussen de twee termen, ziet er dan als volgt uit:

$$G^* = \arg\min_G \max_D \mathcal{L}_{\text{cGAN}}(G, D) + \lambda \mathcal{L}_{\text{L1}}(G) \tag{2.7}$$



(a) PatchGAN discriminator. Elke waarde van de uitvoermatrix stelt de probabiliteit voor die de discriminator toekent aan het echt zijn van corresponderende patch. Bron: [44]



(b) Invloed van de PatchGAN discriminator (cGAN) op de resulterende afbeeldingen. Bron: [4]

Figuur 2.11: Invloed en toepassing van de PatchGAN discriminator uit pix2pix

Netwerkarchitectuur

Bij image-to-image translation problemen heeft de uitvoer typisch dezelfde hoge resolutie als de invoer. Traditioneel werd hiervoor gebruik gemaakt van een encoder-decoder architectuur (zie sectie 2.3.2) [38, 39, 40, 41, 42]. Hierbij dient laag-niveau informatie echter door alle lagen, inclusief de bottleneck, te passeren. Om dit te vermijden, wordt bij pix2pix gebruik gemaakt van een U-Net voor de generator (zie sectie 2.3.3). Zo kan deze laag-niveau informatie rechtstreeks doorheen het netwerk stromen door middel van skip connections. Verder bestaan zowel de generator als discriminator uit modules van de vorm convolutie-BatchNorm-ReLU.

Zoals eerder besproken, leidt het gebruik van enkel L_1- of L_2-loss tot wazige resultaten bij het genereren van afbeeldingen [43]. Desondanks slagen deze loss-functies er wel goed in om de lage frequenties accuraat te vervatten. Er kan dus vertrouwd worden op de L_1-loss om laag-frequente correctheid af te dwingen, terwijl de GAN discriminator zich beperkt tot het afdwingen van hoog-frequente correctheid. Dit is de de motivatie van het gebruik van een discriminatorarchitectuur, genaamd *PatchGAN*, die enkel structuur beoordeelt op het niveau van patches.

De discriminator probeert dus elke $N \times N$ patch te classificeren als echt of vals. Vervolgens kan de discriminator convolutioneel toegepast worden op de afbeelding, waarbij de uiteindelijke uitvoer het gemiddelde is van alle classificaties. In de praktijk kunnen we zo met N een stuk kleiner dan de volledige afmetingen van de afbeelding goede resultaten bekomen. Dit is erg nuttig aangezien een kleinere PatchGAN minder parameters heeft waardoor hij minder overfit en sneller is. De invloed en toepassing van deze PatchGAN wordt gedemonstreerd in figuur 2.11.

Optimalisatie

Om deze netwerken te trainen, wordt de standaardaanpak van stapsgewijs afwisselend D en G te optimaliseren gebruikt (zie figuur 2.13) [29].

De eerste stap, voorgesteld in figuur 2.13a, is het optimaliseren van de generator. Eerst genereert de generator op basis van de invoer (de ongekleurde kat) een uitvoer (de blauwe kat). Vervolgens wordt deze generatoruitvoer vergeleken met de ground truth ($\mathcal{L}_{\mathrm{L}_1}$) en als invoer gebruikt voor de discriminator ($\mathcal{L}_{\mathrm{cGAN}}$). Een gewogen som van de afwijking t.o.v. de ground truth en de beoordeling door de discriminator vormt dan de loss-functie waarop backpropagation kan toegepast worden. De optimizer werkt zo de gewichten van de generator bij zodat de uitvoer meer kans maakt om door de discriminator als echt beoordeeld te worden en dichter bij de ground truth ligt.

De tweede stap, voorgesteld in figuur 2.13b, is het optimaliseren van de discriminator. Het doel van de discriminator is om echte (ground truth) van valse (gegenereerd door de generator) afbeeldingen te onderscheiden. Hiertoe voeren we het discriminator netwerk twee keer uit: eenmaal met de ground truth als invoer en eenmaal met de uitvoer van de generator als invoer. De loss-functie wordt geminimaliseerd als de ground truth als echt bestempeld wordt en de uitvoer van de generator als vals. Nu kan de optimizer de gewichten van de discriminator bijwerken zodat de discriminator net iets beter wordt in het onderscheiden van valse en echte samples.

Merk op dat we in elke stap slechts één netwerk optimaliseren. Zo worden in de eerste stap de gewichten van de discriminator bevroren, terwijl we in de tweede stap de gewichten van de generator bevriezen.

2.5.3 pix2pixHD

Sinds de aankondiging van pix2pix heeft deze methode al zijn success bewezen voor een grote diversiteit aan image-to-image translation taken. Zo is de architectuur ingezet voor een overvloed aan creatieve projecten, gaande van *Do-As-I-Do* pose overdracht [47] tot Sketch → Pokémon [48] of *edges2cats* [49]. Hoewel deze resultaten bemoediging zijn, zijn ze nog steeds ver verwijderd van realisme, zeker bij hogere resoluties. Op basis van deze observaties opperden Chen en Koltun [50] recent dat adversarial training fundamenteel onstabiel is bij het genereren van afbeeldingen van hoge resolutie. Om hieraan tegemoet te komen, stelden Wang et al. [46] enkele aanpassingen voor aan de originele pix2pix architectuur. De resulterende architectuur wordt pix2pixHD genoemd en leidt tot realistischere resultaten bij hogere resoluties. Concreet stellen ze drie aanpassingen voor aan het originele framework: een grof-naar-fijn generator, een meerschalige discriminator een robuustere objectief.

Grof-naar-fijn generator

Om de training bij hogere resoluties te stabiliseren, wordt de generator opgesplitst in twee deelnetwerken: G_1 en G_2. Hierbij is is G_1 het globaal generator netwerk en G_2 het lokale verbetering

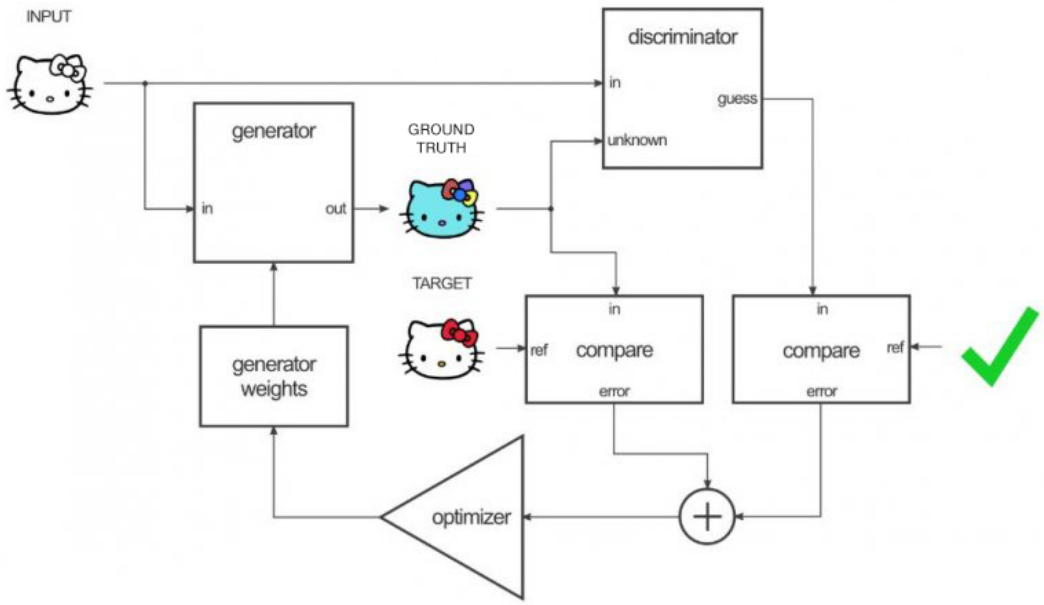
(a) Luchtfoto's naar kaarten

(b) Dag naar nacht

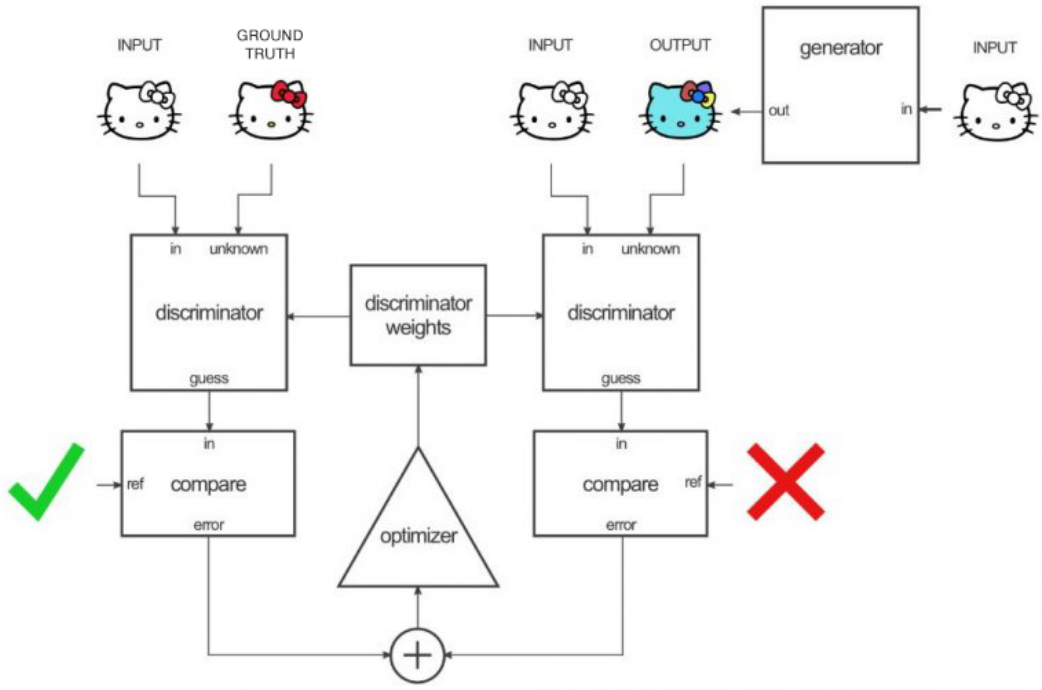
(c) Zwart-wit naar kleur

(d) Semantische labels naar voorgevel

Figuur 2.12: Voorbeeld van verschillende image-to-image translation problemen via pix2pix. Telkens wordt exact dezelfde architectuur en trainingsprocedure gebruikt, enkel de trainingsdata verschilt. Bron: [4]

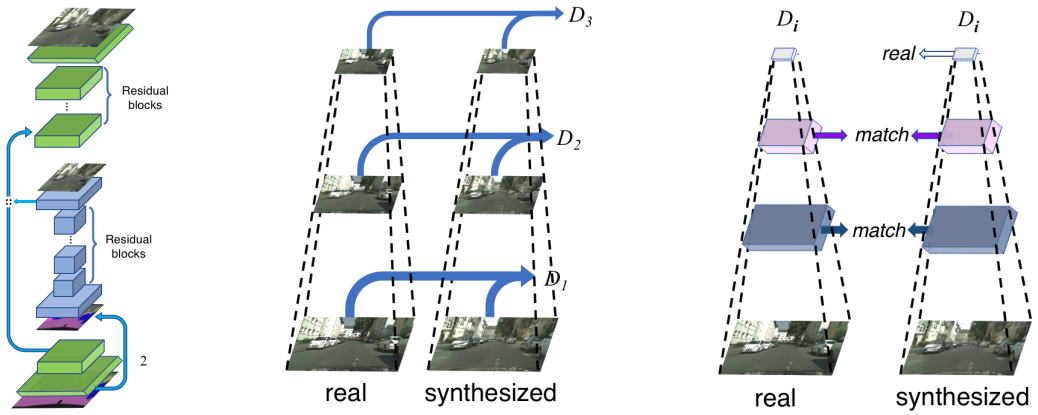


(a) Optimalisatie van de generator



(b) Optimalisatie van de discriminator

Figuur 2.13: De twee optimalisatiestappen in de pix2pix architectuur. Bron: [45]



(a) Grof-naar-fijn generator

(b) Meerschalige discriminator

(c) Robuuster objectief

Figuur 2.14: De drie aanpassingen voorgesteld in pix2pixHD om training bij hogere resoluties te stabiliseren. Bron: [46]

netwerk. De uiteindelijke generator is dan de tuple $G = \{G_1, G_2\}$. Specifiek werkt de globale generator aan een resolutie van 1024×512, waarna de lokale verbeteraar de resolutie $4\times$ verhoogt. Hierbij wordt eerste de globale generator getraind, en vervolgens de lokale verbeteraar. Uiteindelijk worden beide netwerken gemeenschappelijk verder getraind als finetuning. Een dergelijke opstelling laat toe op een efficiënte manier zowel hoog-niveau als laag-niveau kenmerken te modelleren.

Meerschalige discriminator

Afbeeldingen van hoge resolutie zijn erg uitdagend voor de discriminator. Om echte van valse hoge-resolutie afbeeldingen te onderscheiden, heeft de discriminator namelijk een groot receptive field nodig. Dit betekent dat het netwerk dieper moet zijn of grotere kernels moet bevatten. Beide opties verhogen de netwerkcapaciteit drastisch en maken zo het netwerk gevoeliger voor overfitting. Ook leiden beide opties tot een verhoging van het benodigde geheugen, wat reeds een schaarse bron is bij het genereren van afbeeldingen van hoge resolutie.

In plaats daarvan wordt er gekozen om een meerschalige discriminator te gebruiken. Dit betekent dat er gebruik gemaakt wordt van drie discriminatornetwerken (D_1, D_2 en D_3) die een identieke structuur hebben maar op verschillende schalen werken. Concreet herschalen we zowel de echte als valse afbeelding met een factor $\frac{1}{2}$ en $\frac{1}{4}$ om een afbeeldingspiramide van drie groottes te krijgen. Elk discriminatornetwerk werkt dan op één van deze drie schalen. Op die manier kan het discriminatornetwerk dat werkt op de grofste schaal globale consistentie afdwingen, terwijl het discriminatornetwerk op de fijnste schaal rekening kan houden met details en textuur.

Robuuster objectief

De laatste aanpassing is het introduceren van een nieuwe loss term, de zgn. *feature matching loss*, gebaseerd op de activaties in de discriminator. Het doel hiervan is om de training te stabiliseren door te vereisen dat de generator natuurlijke statistieken produceert binnenin de discriminator. Specifiek worden hiertoe features van meerdere lagen van de discriminator geëxtraheerd en wordt getracht om deze tussenliggende voorstellingen te laten overeenkomen tussen de echte en valse afbeelding. Als we de feature extractor van laag i van discriminatornetwerk D_k voorstellen als $D_k^{(i)}$, dan ziet de feature matching loss $\mathcal{L}_{\text{FM}}(G, D_k)$ er als volgt uit:

$$\mathcal{L}_{\text{FM}}(G, D_k) = \mathbb{E}_{\mathbf{x},\mathbf{y}} \sum_{i=1}^{T} \frac{1}{N_i} \|D_k^{(i)}(\mathbf{x}, \mathbf{y}) - D_k^{(i)}(\mathbf{x}, G(\mathbf{x}))\|_1 \tag{2.8}$$

Hierbij stelt T het totaal aantal lagen voor en is N_i het aantal elementen in elke laag. Deze feature matching loss is gerelateerd aan aan de perceptual loss [51, 15, 40], die de performantie drastisch verhoogt voor taken zoals superresolutie [36] of stijloverdracht [40]. De intuïtie hierachter is dat de generator afbeeldingen wenst te genereren die leiden tot activaties in de discriminator die vergelijkbaar zijn met diegene veroorzaakt door echte afbeeldingen.

Hoofdstuk 3

Dataset

Eén van de meest cruciale aspecten in elke machine learning pipeline is de dataset. De keuze van de dataset heeft immers een rechtstreekse invloed op de performantie. Verder laat een zorgvuldig gekozen dataset ons toe interessante experimenten uit te voeren en gefundeerde conclusies te trekken. Eerst gaan we dieper in op de keuze van de stripreeks en de opbouw van de dataset. Vervolgens hebben we het over de train-test splitsing en data augmentatie. Tenslotte bespreken we de omzetting van kleurprenten naar de binaire lijntekeningen die we als netwerkinvoer gebruiken.

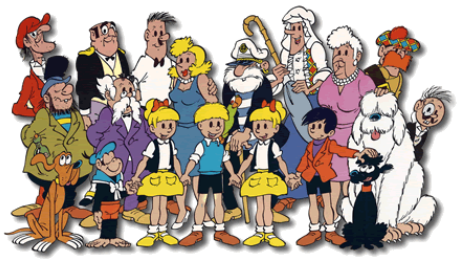
3.1 Keuze stripreeks

We kiezen er voor om gebruik te maken van scans van de Belgische stripreeks Jommeke. Deze stripreeks heeft namelijk enkele eigenschappen die ze uitermate aantrekkelijk maakt om onze methode op te trainen en te testen: een diverse cast, een grote populariteit en de beschikbaarheid van zwart-witalbums.

Ten eerste is de diverse cast handig om onze methode te evalueren. Zo zijn er een viertal hoofdpersonages die in nagenoeg elk album voorkomen (zie figuur 3.1a), gevolgd door een grote diversiteit aan nevenpersonages die in een beperkt aantal albums voorkomen (zie figuur 3.1b). Deze personages dragen vaak dezelfde kledij, waardoor het makkelijk af te leiden is als onze methode een personage herkent en correct inkleurt. Dit laat ons toe om conclusies te trekken over bijvoorbeeld de performantie van onze methode voor zowel veel als weinig voorkomende personages.

Ten tweede kent de reeks een enorme populariteit, met meer dan 50 miljoen verkochte exemplaren sinds 1959. Hierdoor is het relatief makkelijk om een uitgebreide dataset van gescande albums te bekomen. Verder zijn vanwege deze populariteit ook tekeningen gemaakt door fans te vinden op het internet. Die laten ons toe om onze methode te testen op prenten die dezelfde personages afbeelden maar niet afkomstig zijn van de originele tekenaar.

(a) De hoofdpersonages. Deze komen in nagenoeg ieder album voor.



(b) Een deel van de nevenpersonages. Deze komen elk in een beperkt aantal albums voor.

Figuur 3.1: Een greep uit de personages in Jommeke-albums. Bron: [52]

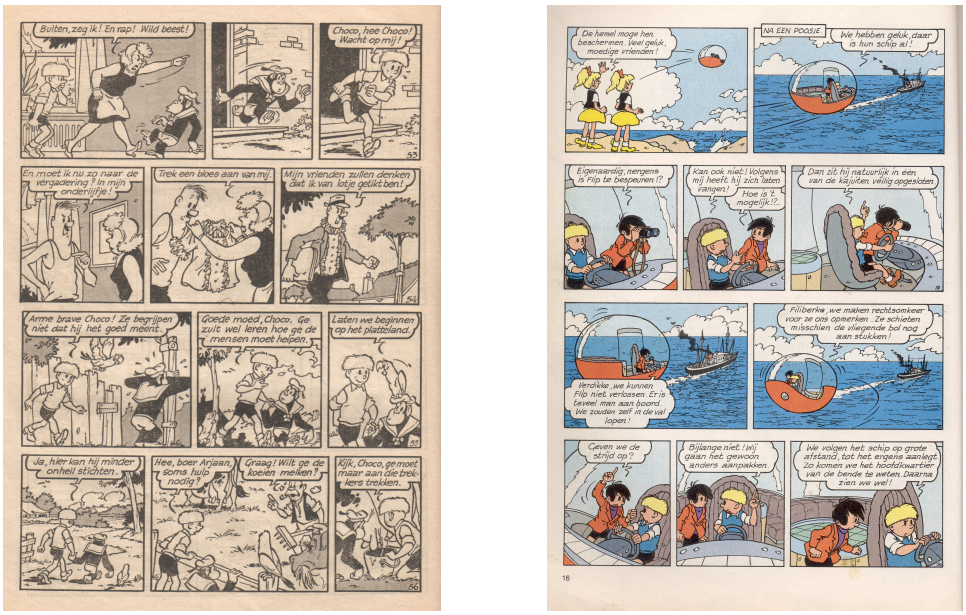
Tenslotte werden de eerste 91 van de 290 albums oorspronkelijk uitgegeven in zwart-wit. Pas sinds het uitbrengen van het 92e album in 1979 verschijnen de verhalen in kleur. Hierdoor kunnen we onze methode niet enkel testen op kleuralbums die eerst door ons omgezet zijn naar lijntekeningen, maar ook op originele zwart-witalbums. Zo geven we deze zwart-witalbums een nieuwe dimensie en leren we hoe onze methode omgaat met albums in een andere, oudere tekenstijl.

3.2 Opbouw

De dataset die we gebruiken bestaat uit scans van 252 albums, waarvan 61 in zwart-wit en 191 in kleur. Elk album bestaat uit ongeveer 48 pagina's, waardoor we 2818 pagina's in zwart-wit en 9329 pagina's in kleur ter beschikking hebben. Voorbeelden van een willekeurige pagina in zwart-wit en in kleur worden weergegeven in figuur 3.2a en 3.2b respectievelijk. Belangrijk op te merken is dat elk album ofwel in kleur ofwel in zwart-wit beschikbaar is in onze dataset. We hebben met andere woorden geen paren van eenzelfde album in kleur en in zwart-wit.

Aangezien het hier scans betreft en geen bronafbeeldingen, is er een grote variatie in de kwaliteit van de pagina's. Zo varieert de scan-resolutie tussen de 180 DPI (1500 x 2120) en 260 DPI (2200 x 3112). We kiezen er hier doelbewust voor om dit niet te normaliseren naar eenzelfde resolutie. Op die manier wordt het netwerk robuust ten opzichte van veranderingen in schaal. Hier wordt verder op ingegaan in sectie 3.4.

Verder bestaat elk album uit een frontcover (figuur 3.3a), een titelpagina (figuur 3.3b), ongeveer 45 gewone pagina's en een backcover (figuur 3.3c). Ook hier kiezen we er bewust voor om deze afwijkende pagina's niet te verwijderen uit de dataset. Deze pagina's hebben namelijk een andere layout dan de gewone pagina's, maar zijn nog steeds nuttig aangezien ze eveneens de stripfiguren afbeelden.



(a) Voorbeeldscan van een zwart-witalbum (b) Voorbeeldscan van een kleuralbum

Figuur 3.2: Twee voorbeeldscans uit de gebruikte dataset

Tenslotte kiezen we er voor om geen voorverwerking, zoals het uitsnijden van prentjes, toe te passen. Dit manueel doen is immers tijdrovend en zou zo afbreuk doen aan de toepasbaarheid van onze methode. Automatisch uitsnijden is dan weer foutgevoelig en loopt fout in speciale gevallen, zoals personages die uit het kader stappen [53]. We nemen dan ook volledige pagina's als invoer voor onze methode. Op die manier heeft het netwerk alle relevante informatie ter beschikking en de maximale vrijheid om de prent in te kleuren.

3.3 Train-test splitsing

Na het verzamelen van de dataset, splitsen we de dataset in een training set en een test set. Hierna werd de test set aan de kant geschoven en pas opnieuw aangeraakt om de performantie van onze uiteindelijke methode te beoordelen. Om de architectuur, loss-functie en hyperparameters te kiezen maken we gebruik van een deel van de training set die het netwerk niet te zien kreeg: de validation set. Op die manier geeft de test set een onbevooroordeelde indicatie van de performantie op nooit eerder geziene data. Aangezien er voor de zwart-witalbums geen ground truth is en we hier dus niet op kunnen trainen, worden deze allemaal ingedeeld bij de test set. Voor de kleuralbums zijn er echter verschillende mogelijkheden om de splitsing te maken.

Een eerste optie zou zijn om uit elk album willekeurig enkele pagina's te selecteren en die te gebruiken als test set. Dit is echter niet volledig representatief aangezien het mogelijk is dat

(a) Frontcover (b) Titelpagina (c) Backcover

Figuur 3.3: De drie atypische pagina's die in elk album aanwezig zijn

prentjes op de gekozen pagina sterke gelijkenissen vertonen met prentjes op de pagina ervoor of erna. Bij het inspecteren van de test set bleek dat dit fenomeen in de praktijk zeer beperkt blijkt.

Een alternatieve optie is om enkel volledige albums te selecteren als test data. Aangezien zo geen enkele prent uit het album gezien werd, is deze splitsing het meest representatief voor de performantie bij het inkleuren van ongeziene albums. De extra moeilijkheid daarbij is namelijk dat er vaak objecten zijn die in slechts één album opduiken. Wanneer we volledige albums indelen als train of test, dient het netwerk dus soms objecten in te kleuren die het nooit eerder heeft gezien.

We verwachten dat deze twee opties leiden tot een verschillende performantie. Aangezien het interessant is om de precieze invloed hiervan te bepalen, kiezen we ervoor om onze test set op te delen in drie categorieën: *zwart-wit*, *gezien album* en *ongezien album*. De categorie *zwart-wit* bevat alle zwart-witalbums, waarvoor geen ground truth beschikbaar is. Vervolgens bestaat de categorie *gezien album* uit pagina's waarvan de resterende pagina's van datzelfde album beschikbaar zijn in de training set. Tenslotte bevat *ongezien album* volledige albums waarvan geen enkele pagina beschikbaar is in de training set. Belangrijk op te merken is dat er nooit overlap is tussen pagina's in de training set en de test set. Het onderscheid *gezien album* en *ongezien album* duidt enkel aan of de resterende pagina's in het album al dan niet in de training set vervat zijn. Deze onderverdeling is samengevat in tabel 3.1.

3.4 Data augmentatie

Zoals in hoofdstuk 4 besproken wordt, is onze oplossing gebaseerd op deep learning. De performantie van neurale netwerken neemt typisch toe naarmate de grootte en diversiteit van de dataset toeneemt. Onze dataset is echter niet zo groot naar huidige deep learning standaarden. We maken dan ook sterk gebruik van data augmentatie. Daarbij gebruiken we domein-specifieke

Categorie	**# verschillende albums**	**# pagina's**
Zwart-wit	61	2818
Gezien album	120	120
Ongezien album	8	387

Tabel 3.1: Opdeling van de test set in categorieën

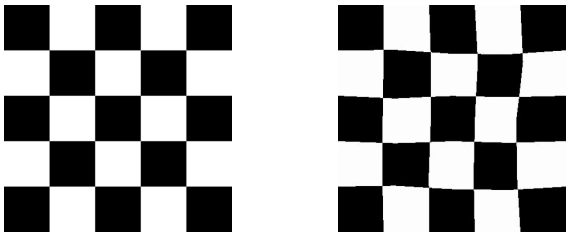
kennis om nieuwe, gediversifieerde trainingsdata te genereren op basis van de originele data. Concreet gebruiken we vier transformaties die elk leiden tot een nieuwe stripafbeelding die een aannemelijke variatie is op de oorspronkelijke afbeelding. Zo kunnen we willekeurig één of meerdere van deze transformaties toepassen met willekeurige parameters om een grote hoeveelheid nieuwe trainingsdata te bekomen. Hiervoor maken we gebruik van Augmentor, een image augmentation bibliotheek voor Python [54]. Voor elke transformatie is een voorbeeld weergegeven in figuur 3.5. We gaan nu dieper in op de verschillende transformaties.

Horizontale spiegeling

Een eerste transformatie die we gebruiken is een spiegeling langs de horizontale as. Deze transformatie is nuttig aangezien het resultaat, afgezien van mogelijke tekst, zo goed als altijd een nieuwe stripafbeelding is die steek houdt. Op die manier verdubbelt de grootte van onze trainingset. Merk op dat we enkel horizontale en geen verticale spiegelingen toepassen. Dit is een bewuste keuze aangezien een verticale spiegeling typisch niet resulteert in een nieuwe strippprent die semantisch steek houdt. Daardoor zijn verticale spiegelingen niet nuttig om de performantie van het netwerk te verbeteren. De horizontaal gespiegelde tekst vormt echter geen probleem aangezien deze letters ongewijzigd overgenomen worden in de uitvoer, zonder rekening te houden met hun betekenis.

Elastische distortie

Een andere nuttige transformatie is een zogenaamde *elastische distortie* [55]. Deze transformatie resulteert in een nieuwe afbeelding met dezelfde afmetingen maar met een lichtjes vervormde inhoud. Oorspronkelijk zijn elastische distorties ontworpen om de MNIST dataset [56] van handgeschreven cijfers te augmenteren. De achterliggende redenering was dat deze elastische distorties corresponderen met lichte, ongecontroleerde oscillaties van de hand tijdens het neerpennen van een letter. Aangezien strips eveneens met de hand getekend worden, en er dus heel wat variabiliteit is in de precieze vormen van personages en voorwerpen, is deze transformatie ook hier nuttig. Op die manier wordt het netwerk beter in het herkennen van hetzelfde voorwerp dat telkens op lichtjes verschillende wijze is getekend. Een voorbeeld van een willekeurige elastische distortie is weergeven in figuur 3.4. In figuur 3.5b is het verschil minder duidelijk maar in de vervormde afbeelding is onder andere de papegaai lichtjes vervormd.



(a) Originele afbeelding

(b) Afbeelding na elastische distortie

Figuur 3.4: Voorbeeld van een willekeurige elastische distortie toegepast op een schaakbordpatroon

Rotatie

Een volgende transformatie die steek houdt is het lichtjes roteren van de afbeelding. De bedoeling hiervan is om het netwerk invariant te maken ten opzichte van kleine rotaties. Zo kunnen personages ingekleurd worden onafhankelijk van de precieze hoek waaronder ze getekend zijn. Een aandachtspunt hierbij is dat een rotatie niet enkel de personages, maar ook de lijnen die het prentje afbakenen, wijzigt. In stripverhalen zijn de lijnen die het kader vormen echter typisch perfect horizontaal of verticaal. Hier werken we echter niet met originele bronbestanden maar met imperfecte scans. De dataset bevat dan ook veel pagina's die niet volledig recht op de scanner lagen en daardoor lichtjes geroteerd zijn. We kiezen er voor om ons netwerk hier robuust tegen te maken door elke afbeelding willekeurig te roteren over een hoek van $-3°$ tot $3°$. Door te kiezen voor rotaties over een beperkte hoek houden de resulterende afbeeldingen nog steeds semantisch steek. Zo zou een rotatie over een hoek van pakweg $90°$ resulteren in onrealistische prenten met rechtopstaande tekstballonnen. Dergelijke prenten die semantisch geen steek houden komen niet voor bij rotaties over een kleine hoek.

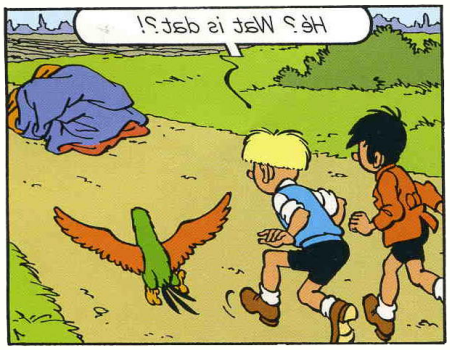
Herschaling

De laatste transformatie die we toepassen is het herschalen van de invoer. Dit houdt in dat we de oorspronkelijke afbeelding ofwel vergroten ofwel verkleinen met een willekeurige factor. De bedoeling hiervan is om het netwerk schaal-invariant te maken, waarbij personages ingekleurd kunnen worden onafhankelijk van hoe groot of klein ze getekend zijn. Dit is een logische augmentatie aangezien stripfiguren doorheen een album in een veelvoud aan groottes getekend worden. Bovendien werd bij het scannen reeds een veelvoud aan resoluties gebruikt. Door zelf te herschalen, lijkt het dus alsof elke oorspronkelijke afbeelding beschikbaar is in een veelvoud aan scan-resoluties.

(a) Originele afbeelding

(b) Elastische distortie

(c) Rotatie



(d) Horizontale spiegeling

(e) Herschaling

Figuur 3.5: Voorbeeld van elke mogelijke transformatie die we gebruiken voor data augmentatie

3.5 Conversie naar lijntekeningen

Het doel van onze methode is om ongekleurde lijntekeningen om te zetten naar gekleurde prenten. Voor het trainen van zo'n netwerk hebben we, naast de gegeven gekleurde prenten als ground truth, ook ongekleurde lijntekeningen nodig die als invoer kunnen dienen. Zoals eerder besproken, komt elk album slechts éénmaal voor in de dataset: ofwel in kleur ofwel in zwart-wit. We hebben dus geen paren van invoer-uitvoer gegeven waarop we kunnen trainen. We kunnen echter dankbaar gebruik maken van het feit dat de omzetting van een kleurafbeelding naar een zwart-witafbeelding een stuk eenvoudiger is dan de omgekeerde richting. Zo kunnen we voor elke kleurafbeelding in onze dataset een corresponderende binaire lijntekening genereren, die vervolgens gebruikt kan worden als invoer voor het netwerk. In deze zwart-witafbeelding zijn idealiter enkel de zwarte vlakken en lijnen van de oorspronkelijke afbeelding zwart en zijn alle andere pixels wit.

Uiteraard heeft niet elke pixel die we wensen over te houden in de binaire versie, een perfecte zwarte kleur in de originele kleurenafbeelding. Zelfs binnen één enkele kleurenafbeelding is er een grote variatie in de kleur van de pixels die we bestempelen als zwarte lijn of vlak. Daarom definiëren we een waarschijnlijkheidsdichtheid over de Lab-kleurenruimte die de kans uitdrukt dat een pixel met een bepaalde kleur behoort tot een zwarte lijn of vlak. Vervolgens kunnen we deze probabiliteiten gebruiken om enkel de pixels over te houden die boven een bepaalde drempelwaarde liggen.

Concreet is de gebruikte dichtheidsfunctie het product van de dichtheidsfuncties van drie onafhankelijke univariate normaalverdelingen (één voor elk dimensie van de Lab-kleurenruimte). Zo bekomen we een multivariate normaalverdeling $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ met een diagonale covariantiematrix $\boldsymbol{\Sigma}$. De verwachtingswaarde $\boldsymbol{\mu}$ is de drie-dimensionale nulvector. De motivering hierachter is de observatie dat elke scheidingslijn in de prent onder ideale omstandigheden perfect zwart zou zijn ($L^* = 0$, $a^* = 0$ en $b^* = 0$). In de praktijk zijn er echter meerdere onafhankelijke bronnen van ruis die deze kleur verstoren, vandaar de keuze voor een normaalverdeling. Zo wordt ruis geïntroduceerd bij het scannen, zijn verschillende albums ingescand met verschillende scannerconfiguraties en is de kleurweergave op de oorspronkelijke afdruk ook niet perfect.

De covariantiematrix $\boldsymbol{\Sigma}$ en verwachting $\boldsymbol{\mu}$ van de beschouwde normaalverdeling zien er als volgt uit:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_l^2 & 0 & 0 \\ 0 & \sigma_a^2 & 0 \\ 0 & 0 & \sigma_b^2 \end{bmatrix} \qquad \boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.1}$$

Vervolgens kunnen we op basis van deze waarschijnlijkheidsdichtheid een binaire lijntekening bekomen. Concreet genereren we voor elke Lab-kleurenafbeelding $\mathbf{y} \in \mathbb{Z}^{H \cdot W \cdot 3}$ een corresponderende binaire afbeelding $\mathbf{x} \in \{0, 1\}^{H \cdot W}$. Als $F(\cdot \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ de cumulatieve distributiefunctie van de

verdeling $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ voorstelt, dan hebben we:

$$\mathbf{x}_i = \begin{cases} 1 & \text{voor } F(\mathbf{y}_i \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \leq 0.9 \\ 0 & \text{anders} \end{cases} \tag{3.2}$$

We houden met andere woorden enkel de pixels over die binnen een 90 % betrouwbaarheidsinterval vallen voor de afwijking van zwarte pixels. Zo kunnen we automatisch voor elke gekleurde prent in de dataset een lijntekening genereren die we als invoer gebruiken voor het netwerk.

Empirische testen hebben uitgewezen dat de waarden $\sigma_a^2 = 25$ en $\sigma_b^2 = 25$ een goede keuze zijn voor het merendeel van de kleurafbeeldingen in de dataset. Deze waarden leiden er toe dat de pixels die behoren tot een zwarte lijn of zwart vlak een erg hoge probabiliteit hebben in vergelijking met de overige pixels. Een goede waarde voor σ_l^2 is minder duidelijk en verschilt van afbeelding tot afbeelding. Wel ligt de optimale waarde van σ_l^2 voor de meeste kleurafbeeldingen ergens in het interval $[25 \,..\, 35]$.

Een mogelijke aanpak is om voor elke kleurenafbeelding manueel een waarde voor σ_l^2 te bepalen die optimaal is voor het extraheren van lijnen uit die specifieke afbeelding. Dit neemt echter veel tijd in beslag en is dus geen goede optie voor grotere datasets. Wij kiezen er dan ook voor om tijdens het trainen de waarde van σ_l^2 telkens uniform te samplen uit het interval $[25 \,..\, 35]$. Aangezien het netwerk elke afbeelding een groot aantal keren te zien krijgt tijdens het trainen, zal het netwerk elke afbeelding ook meermaals te zien krijgen met een waarde voor σ_l^2 die dicht bij het optimum ligt. Verder geldt dit ook als een vorm van data augmentatie. Soms zal de waarde σ_l^2 namelijk te hoog zijn, waardoor bepaalde pixels foutief als zwart gemarkeerd worden (zie figuur 3.6d). In andere gevallen zal de waarde σ_l^2 net te laag zijn, waardoor er stukjes lijn wegvallen in de invoerafbeelding (zie figuur 3.6c). Hierdoor wordt het netwerk robuust tegen invoerafbeeldingen die te veel of net te weinig lijnen bevatten. Verder zorgt deze variatie er voor dat de precieze dikte of tekenstijl van lijnen minder invloed heeft.

(a) Originele kleurenafbeelding

(b) Binaire versie gegenereerd met $\sigma_l^2 = 25$

(c) Binaire versie gegenereerd met $\sigma_l^2 = 15$

(d) Binaire versie gegenereerd met $\sigma_l^2 = 45$

Figuur 3.6: Conversie naar een lijntekening met verschillende waarden voor σ_l^2

Hoofdstuk 4

Methode

In dit hoofdstuk ontwikkelen we een methode om binaire lijntekeningen volledig automatisch in te kleuren. We beginnen eerst met het probleem formeel te definiëren. Deze formele definitie legt de basis voor de wiskundige notatie die verder in het hoofdstuk wordt gebruikt. Hierna worden de verschillende componenten van onze oplossing stapsgewijs beschreven en gemotiveerd. Concreet ontwerpen we eerst een netwerkarchitectuur die afgestemd is op het bestudeerde probleem. Vervolgens stellen we een loss-functie op om zo'n geloofwaardig mogelijke resultaten te bekomen. De laatste stap is dan het trainen van het vooropgestelde netwerk.

4.1 Formele probleemstelling

In sectie 1.1 formuleerden we het probleem als het *automatisch genereren van een overeenkomstige, geloofwaardige kleurenversie op basis van een strippprent in binair zwart-wit.* De verschillende aspecten van deze formulering werden daar reeds besproken. Een belangrijk aspect is dat wij als invoer binaire afbeeldingen gebruiken en geen monochrome afbeeldingen. Dit is een bewuste keuze die ons toelaat onze methode in te zetten voor het inkleuren van prenten die nooit eerder ingekleurd werden en dus geen grijswaardenkanaal hebben. Hierdoor is de uiteindelijke oplossing breed inzetbaar. Zo gebruiken we in hoofdstuk 6 onze methode om oudere zwart-witalbums volledig automatisch in te kleuren. Verder demonstreren we in sectie 5.5.1 dat onze methode zonder aanpassingen in staat is schetsen van fans in te kleuren.

Het bovenstaande probleem kunnen we ook wiskundig uitdrukken, wat later van pas komt om onder andere een loss-functie op te stellen. Concreet willen we een functie $\hat{\mathbf{y}} = G(\mathbf{x})$ construeren die een binaire lijntekening $\mathbf{x} \in \{0,1\}^{H \cdot W}$ afbeeldt naar een geloofwaardige inkleuring $\hat{\mathbf{y}} \in \mathbb{R}^{H \cdot W \cdot 3}$. Hierbij stellen H en W respectievelijk de hoogte en breedte voor en is de uitvoer een afbeelding in de CIELAB-kleurruimte. We gaan nu dieper in op hoe we deze functie G precies opstellen.

4.2 Netwerkarchitectuur

In de literatuurstudie zagen we dat deep learning de aanpak is die, met voorsprong, het best presteert op verwante problemen. Geïnspireerd door dit succes, kiezen we ervoor om een convolutioneel neuraal netwerk te gebruiken. De eerste belangrijke ontwerpbeslissing is dan de te gebruiken netwerkarchitectuur. De architectuur bepaalt immers de capaciteit van het netwerk en zo de uiteindelijke performantie. We gaan nu dieper in op hoe we terechtkwamen bij de uiteindelijke netwerkarchitectuur en waarom deze bij uitstek geschikt is voor ons probleem.

4.2.1 Tekortkomingen van encoder-decoder

In een kleurprent uit Jommeke lijkt het op het eerste gezicht zo dat er telkens slechts één kleur per *connected component* wordt gebruikt. Met een connected component bedoelen we hier een groep pixels die binnen een gesloten en geconnecteerde regio van zwarte pixels liggen. Een voor de hand liggend idee is dan ook om het netwerk een kleurhint (een zgn. *scribble* [57, 58]) te laten genereren voor elk connected component. Vervolgens kunnen we traditionele technieken gebruiken om de connected components in te kleuren op basis van deze kleurhints. Aangezien het netwerk op die manier niet zelf verantwoordelijk is om 'binnen de lijntjes' te kleuren, kan de architectuur relatief eenvoudig zijn, bijvoorbeeld een encoder-decoder [22]. Verder kan de resolutie van de uitvoer van het netwerk een stuk lager zijn dan die van de uiteindelijke uitvoer. Het netwerk dient immers slechts een kleur per connected component te voorspellen en niet per pixel. We kleuren dan als naverwerking de hoge-resolutie invoerafbeelding in, hierbij begeleid door de lage-resolutie kleurhints.

Wanneer we bovenstaand idee echter implementeren, komen al snel enkele problemen aan het licht. Een eerste probleem is dat prenten vaak connected components bevatten die slechts enkele pixels groot zijn (zie figuur 4.1a). Om ook deze components correct in te kleuren, dient de resolutie van de kleurhints dicht aan te leunen bij de resolutie van de uiteindelijke afbeelding. Hierdoor dient het netwerk dat kleurhints aanbiedt in de praktijk toch relatief complex te zijn.

Een tweede, groter probleem is dat de prenten soms meerdere kleuren blijken te bevatten per connected component. Vaak is de opening tussen twee connected components relatief klein (zie de rechterafbeelding in figuur 4.1b). Dit scenario zou nog opgevangen kunnen worden door bij het berekenen van connected components geen rekening te houden met erg smalle verbindingstukjes. In sommige gevallen is de verbinding echter erg groot waardoor ook deze techniek niet werkt (zie de linkerafbeelding in figuur 4.1b).

Uit bovenstaande voorbeelden blijkt dat één kleur per connected component toekennen een te eenvoudige benadering is van de werkelijkheid. Elke aanpak die gebruikt maakt van inkleuring op basis van connected components is dus gedoemd om te falen in het genereren van een realistisch uitvoer in sommige gevallen. We richten onze aandacht dan ook op andere methodes.

(a) Kleine connected components (b) Kleurovergang binnen connected component

Figuur 4.1: Probleemgevallen bij het inkleuren op basis van lage resolutie kleurhints. Beide ground truth afbeeldingen illustreren een scenario waar deze methode niet mee overweg kan.

4.2.2 U-Net met à trous convoluties

In de vorige sectie werd duidelijk dat het netwerk meer vrijheid dient te krijgen om realistische resultaten te bekomen. Concreet willen we het netwerk de mogelijkheid geven om elke individuele pixel naar wens in te kleuren. Een logische keuze is dus om gebruik te maken van een fully convolutional network als end-to-end oplossing. Dit netwerk kunnen we dan toepassen op afbeeldingen van eender welke grootte, waarna we als uitvoer een ingekleurde prent bekomen met dezelfde resolutie als de invoer.

Verder is het cruciaal dat het netwerk in staat is hoog-niveau context te combineren met laag-niveau details. Deze noodzaak wordt geïllustreerd in figuur 4.2. Punt A maakt daar deel uit van de hemel bovenaan de afbeelding en dient dus een hemelsblauwe kleur te krijgen. Vervolgens dient het receptive field van het netwerk groot genoeg te zijn om te kunnen bepalen dat punt A en punt B tot dezelfde achtergrond behoren. Enkel zo kan immers de informatie over deze blauwe inkleuring doorgegeven worden naar punt B, dat 600 pixels verwijderd ligt van punt A. Op dat punt dient tenslotte deze hoog-niveau context gecombineerd te worden met de laag-niveau informatie over de precieze vorm van de rij bomen er net onder. Enkel door deze combinatie is het mogelijk om een geloofwaardige en consistente inkleuring te bekomen.

Een ideale architectuur om informatie over deze verschillende niveaus heen te verenigen, is het U-Net dat in sectie 2.3.3 reeds ter sprake kwam. Standaard heeft deze architectuur een receptive field van 140 x 140 pixels [59]. Zoals hierboven aangetoond, is dit te weinig om informatie van de ene kant van de afbeelding naar de andere te kunnen overdragen. Het is namelijk niet uitzonderlijk dat één stripprentje een breedte van ~ 1000 pixels heeft. We brengen dan ook twee belangrijke wijzigingen aan de U-Net architectuur uit pix2pix aan [4]. De eerste aanpassing is het toevoegen een extra downsampling stap, waardoor het receptive field verdubbelt. De tweede aanpassing is het introduceren van een zogenaamde *context module* [23]. Concreet is deze context module een opeenvolging van 3 x 3 convoluties met verschillende dilatatie factoren. Een derge-

Figuur 4.2: Originele kleurenafbeelding met belangrijke informatie op meerdere niveaus.

lijke context module laat toe om het receptive field substantieel te vergroten met een beperkte computationele meerkost [26].

De finale architectuur is weergegeven in figuur 4.3. Hierin herkennen we een gemodificeerde U-Net architectuur met toevoeging van gedilateerde convoluties. In de praktijk bleken we de beste prestaties te bekomen door gebruik te maken van meerdere context modules op verschillende schalen in plaats van één context module met verschillende dilatatie-factoren. De drie context modules zijn op de figuur te herkennen als de drie opeenvolgingen van blauwe pijlen. In totaal gebruiken we zeven types transformaties in het netwerk, die we nu elk bespreken.

Gewone convoluties
Dit zijn standaard 3 x 3 convoluties. Het aantal invoer- en uitvoerkanalen hangt af van de plaats in het netwerk en is af te leiden op de figuur aan het getal boven de omringende feature maps. Typisch hebben we op niveaus met een lagere spatiale resolutie een hoger aantal kanalen. De stride factor is bij deze convoluties 1 waardoor hun invoer en uitvoer altijd dezelfde resolutie hebben.

Strided convoluties
Dit zijn 3 x 3 convoluties waarbij de stridefactor 2 is. Het aantal invoer- en uitvoerkanalen is opnieuw plaatsafhankelijk. Door de stride factor zijn de afmetingen (breedte en hoogte) van hun uitvoer de helft van deze van hun invoer. Deze strided convoluties worden daarom gebruikt om de resolutie te verminderen bij het overgaan naar een lager niveau in het netwerk.

Gedilateerde convoluties
Dit zijn 3 x 3 convoluties met dilatatiefactor 2 (zie sectie 2.3.4). Deze convoluties vormen de

basis van de hierboven besproken context modules. Door hun dilatatiefactor van 2 verdubbelt elke dergelijke convolutie het receptive field. De stride is 1 waardoor hun invoer en uitvoer altijd dezelfde resolutie hebben. We gebruiken deze convoluties enkel in de lagere niveaus van het netwerk aangezien we verwachten daar reeds een meer semantische representatie te hebben. Het idee hierachter is dat de gedilateerde convoluties dan helpen om globale context in rekening te brengen [26].

ReLU

Om niet-lineair gedrag te introduceren, maken we gebruik van de rectified linear unit (ReLU) activatie functie [60]. Deze populaire activatiefunctie heeft al vaak zijn nut bewezen bij zowel gewone als adversarial training [61, 3]. Merk op dat we de activatiefunctie voor de normalisatie plaatsen en niet erna aangezien dit het best presteert in de praktijk [62].

Instance normalisation

In recent onderzoek wordt *batch normalization* aanbevolen om de snelheid, performantie en stabiliteit van neurale netwerken te verbeteren [63]. Concreet wordt het zo mogelijk om hogere learning rates te gebruiken zonder de verdwijnende of exploderende gradients die hier traditioneel mee gepaard gaan. Een probleem met batch normalization is echter dat de performantie sterk afhangt van de gebruikte batchgrootte. Bij kleinere batchgroottes (< 32) worden de batchstatistieken instabiel en neemt de fout van het getrainde model drastisch toe [64]. Aangezien we hier grote invoerafbeeldingen nodig hebben om de nodige context te kunnen bieden aan het netwerk, is de maximale batch size beperkt. Alternatieven die beter overweg kunnen met een kleine batchgrootte zijn *instance normalization* en *group normalization* [65, 66]. Uit experimenten met onze eigen architectuur is gebleken dat instance normalization hier het best presteert. We gebruiken dan ook instance normalization op de plaatsen waar normaalgezien batch normalization wordt ingezet.

Upsampling

Naast resolutie-verminderende transformaties (hier strided convoluties), hebben we ook transformaties nodig die de resolutie verhogen. Bij het traditionele U-Net wordt daarvoor gebruik gemaakt van *transposed convoluties*, ook wel *fractionally-strided convoluties* of *deconvolutions* genoemd [21, 67]. Hoewel deze operatie erg nuttig is, vereist ze veel extra rekentijd en geheugen door de gewichten in de gebruikte kernels. Verder dienen we de strides ook zorgvuldig te kiezen zodat we geen dambordpatroon introduceren in de uitvoer [68]. Aangezien geheugen hier schaars is vanwege de hoge resolutie, maken we geen gebruik van transposed convolutions. In plaats daarvan gebruiken we nearest-neighbor upsampling gevolgd door één of meerdere gewone convoluties. We kiezen voor elke upsampling operatie de schaalfactor 2. Zo herstellen we telkens de resolutie die we hadden voor het downsamplen met een stride-2 convolutie (d.w.z. schaalfactor $\frac{1}{2}$).

Copy and concatenate

De laatste transformatie is diegene die zo kenmerkend is voor een U-Net. Dit is het invoegen van de data afkomstig uit het downsampling pad in het upsampling pad via skip connections. Concreet kopiëren en concateneren we bij het upsamplen telkens de laatste feature maps van het corresponderende niveau op het downsampling pad. In onze architectuur is het aantal feature maps dat wordt ingevoegd via een skip connection telkens gelijk aan het aantal feature maps na de upsampling operatie van het onderliggende niveau. De volgende stap is dan een convolutie die dit aantal kanalen vermindert om zo het geheugenverbruik en de rekentijd te beperken.

De hierboven besproken transformaties komen in onze architectuur vaak voor in vaste opeenvolgingen, zoals bijvoorbeeld convolutie-ReLU of convolutie-ReLu-InstanceNorm. Deze basisblokken worden getoond in de legende linksonder op figuur 4.3. De uiteindelijke architectuur is dan een fully convolutional network is, waardoor we als invoer afbeeldingen van eender welke grootte kunnen gebruiken. Het receptive field van onze architectuur is 795 x 795 pixels [67]. Dit verder vergroten zou het netwerk krachtiger maken, maar bleek moeilijk vanwege de beperkte hoeveelheid beschikbaar GPU-geheugen.

4.3 Loss-formulering

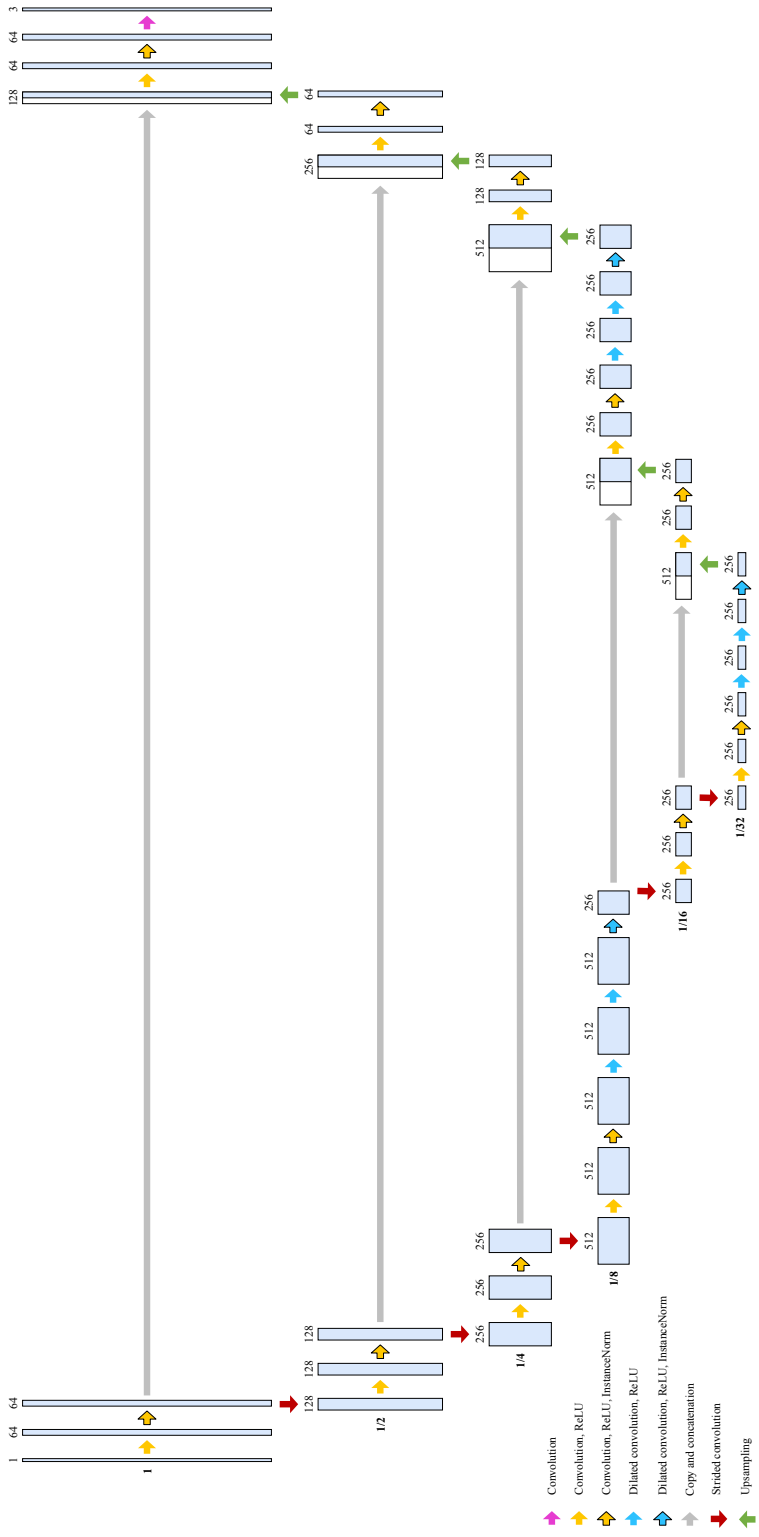
We merkten in sectie 2.4 reeds op dat het opstellen van een geschikte loss-functie een hachelijke aangelegenheid is. Het is namelijk erg moeilijk om de geloofwaardigheid van de uitvoer automatisch te beoordelen. Vaak is het ook niet evident om vooraf de tekortkomingen van een bepaalde loss-functie in te schatten. Neurale netwerken zijn namelijk berucht omwille van hun bekwaamheid in het genereren van resultaten die wel de loss-functie minimaliseren, maar niet diegene zijn waarop we hoopten toen we de loss-functie opstelden [69].

Met deze kennis indachtig kozen we ervoor om de loss-functie incrementeel op te bouwen. De fouten na training met een bepaalde loss-functie gaven telkens de gebreken van die functie aan. Vervolgens konden we onze loss-formulering bijstellen om hieraan tegemoet te komen. In de volgende secties gaan we dieper in op de termen in de loss-functie en hun impact op de performantie van het netwerk.

4.3.1 L_1-loss

Een eerste voor de hand liggende optie om de uitvoer van het netwerk te vergelijken met de verwachte uitvoer, is het gebruik van de L_1-norm of L_2-norm. Wij kiezen om de L_1-norm te gebruiken aangezien deze over het algemeen leidt tot scherpere resultaten dan de L_2-norm [4, 70]. Alle afstanden worden berekend in de Lab-kleurenruimte, aangezien deze beoogt perceptueel uniform te zijn voor menselijke kleurenvisie (zie sectie 2.4.1).

Concreet drukt de loss-term $\mathcal{L}_{L_1}(\hat{\mathbf{y}}, \mathbf{y})$ de L_1-afstand of mean absolute error (MAE) uit tussen de



Figuur 4.3: Gebruikte netwerkarchitectuur. Elke blauwe rechthoek is een feature map met het erboven vermelde aantal kanalen. De resolutie ten opzichte van de originele invoer staat links in het vet. Witte rechthoeken duiden gekopieerde feature maps aan. Pijlen duiden de verschillende operaties aan.

gereconstrueerde afbeelding $\hat{\mathbf{y}} \in \mathbb{R}^{H \cdot W \cdot 3}$ en de oorspronkelijke afbeelding $\mathbf{y} \in \mathbb{R}^{H \cdot W \cdot 3}$ als volgt:

$$\mathcal{L}_{\mathrm{L}_1}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{H \cdot W \cdot 3} \sum_i |\mathbf{y}_i - \hat{\mathbf{y}}_i| \tag{4.1}$$

Zoals eerder aangehaald in sectie 2.4.2, is het voornaamste probleem met deze functie dat ze niet robuust is ten opzichte van de ambiguïteit en multimodaliteit die inherent is aan het probleem. Wanneer het netwerk niet zeker weet welk kleur een object dient te krijgen, kiest het niet noodzakelijk voor één van de mogelijke kleuren. In plaats daarvan, worden vaak een tussenliggend kleur gekozen, wat resulteert in een grauw en gedesatureerd resultaat. Verder leidt de L_1-loss typisch tot wazige resultaten bij het genereren van afbeeldingen [43].

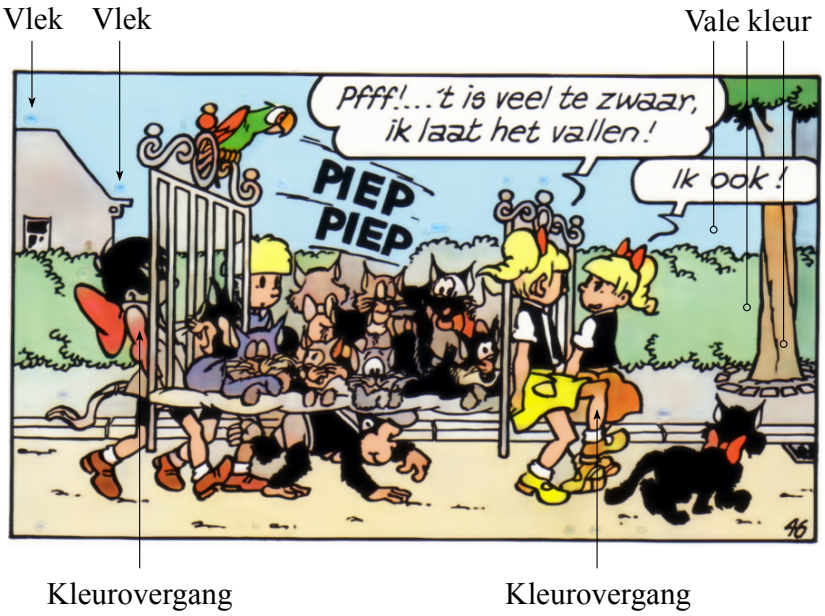
Hoewel louter deze loss-functie gebruiken niet leidt tot erg realistische inkleuringen, is ze toch nuttig in ons geval. Voor veel objecten, zoals de kledij van hoofdpersonages, is de inkleuring niet ambigu en is het optimum volgens de L_1-loss een goede benadering. Door deze loss te gebruiken, leert het netwerk reeds een goede inkleuring voor een aanzienlijk deel van de voorwerpen.

4.3.2 Discriminator loss

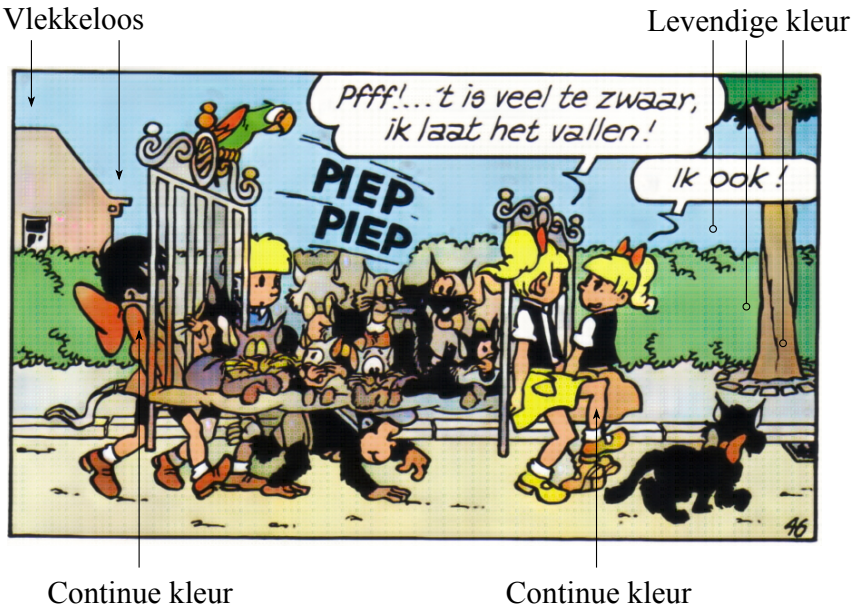
Het resultaat van een inkleuring na training met de L_1-loss wordt getoond in figuur 4.4a. We observeren dat het netwerk leert om de meerderheid van de personages en objecten correct in te kleuren. Toch bevat de uitvoer nog heel wat imperfecties, aangeduid op de figuur, die verraden dat het hier niet om een originele inkleuring gaat. Concreet zien we drie storende fouten: *vlekken*, *ongewenste kleurovergangen* en *vale kleuren*.

Een eerste fout zijn kleine, donkere vlekken in continue gebieden. Dergelijke vlekken komen op de voorbeeldinkleuring voor in de lucht, het voetpad en de straat. Aangezien dergelijke vlekken typisch optreden in laag-frequente gebieden zijn ze erg storend. Verwant aan deze fout zijn ongewenste kleurovergangen. Bij die fout gaat het niet over een klein, donker gebied maar wel over een plotse, grotere overgang naar een volledig verschillende kleur. In beide gevallen hebben we echter een kleurverandering die onnodig en storend is. Dergelijke fouten kunnen we terugleiden naar het feit dat de L_1-loss louter de afstand tot de ground truth minimaliseert. Het netwerk ziet dan ook geen graten in het introduceren van ongeloofwaardige kleurovergangen in een poging deze afstand te minimaliseren.

De andere fout, vale en grauwe kleuren, werd reeds voorspeld in de vorige sectie. Hier zien we deze fout terug bij de inkleuring van de lucht, de struiken en de boomstam. Hoewel de tint in elk van de gevallen correct is, is de toegekende kleur telkens gedesatureerd. Zoals hierboven besproken, is dit te wijten aan de L_1-loss die niet overweg kan met de multimodaliteit van het probleem en zo gaat voor gedesatureerde kleuren.

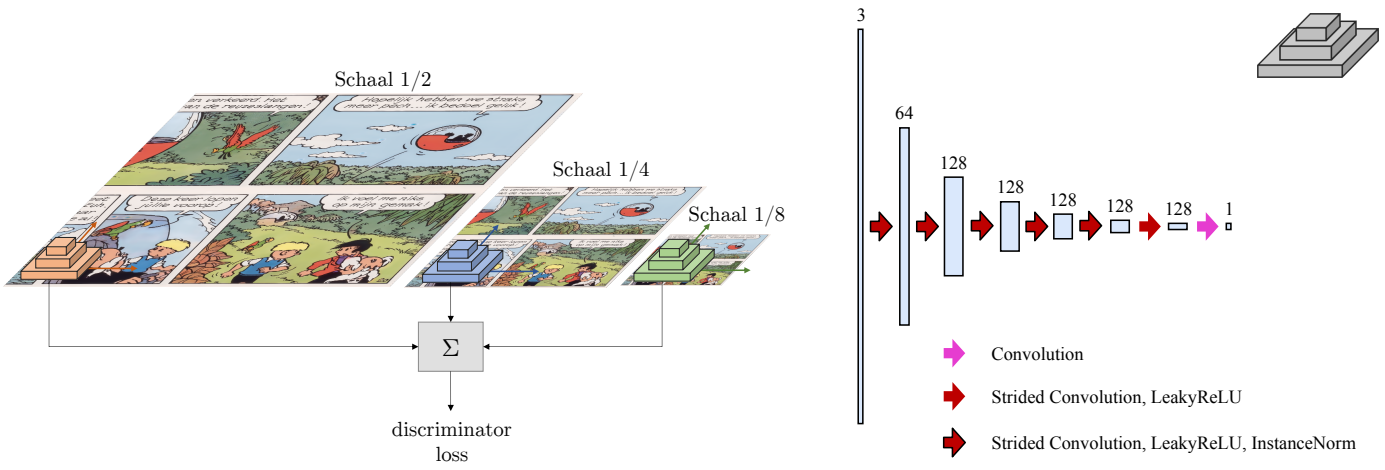


(a) Inkleuring door netwerk getraind met L_1-loss



(b) Inkleuring door netwerk getraind met L_1-loss en discriminator loss

Figuur 4.4: Invloed van discriminator loss op de inkleuring. Deze afbeelding werd gekozen aangezien ze alle fouten bevat die we ook terugzien in andere afbeeldingen.



(a) Toepassen en aggregeren van de drie discriminatornetwerken.

(b) Netwerkarchitectuur van één discriminator. Deze architectuur is hetzelfde voor elke schaalgrootte.

Figuur 4.5: Toepassing en architectuur van de discriminator

.

De bovenstaande fouten zijn telkens opvallend voor mensen en doen sterk afbreuk aan de geloofwaardigheid van onze inkleuringen. Een loss-functie opstellen die deze fouten correct afstraft is echter niet evident. Idealiter willen we inkleuringen bekomen die niet te onderscheiden zijn van handmatig ingekleurde prenten. Zoals we in sectie 2.5 zagen, is dit net de premisse van generative adversarial networks (GANs). We breiden onze loss formulering dus uit met een discriminator loss term. Deze term zorgt er vervolgens voor dat elke afwijking die verraadt dat onze inkleuring niet echt is, afgestraft wordt.

We zagen in sectie 4.2.2 dat het receptive field van de generator (U-Net) voldoende groot dient te zijn om realistische inkleuringen te bekomen. Analoog dient ook het receptive field van de discriminator groot genoeg te zijn om inkleuringen accuraat te kunnen beoordelen. Aangezien een discriminator met een dergelijk receptive field veel geheugen inneemt en gevoelig is voor overfitting, gebruiken we in plaats daarvan een meerschalige discriminator. Net zoals in pix2pixHD [46], hebben we drie eenvoudigere discriminatornetwerken die een identieke structuur hebben maar op verschillende schalen werken. Concreet herschalen we de uiteindelijke inkleuring met een factor $\frac{1}{2}$, $\frac{1}{4}$ en $\frac{1}{8}$. Op die manier kan elke discriminator efficiënt fouten op zijn schaal herkennen. Zo kan de discriminator op schaal $\frac{1}{2}$ lokale fouten, zoals vlekken, afstraffen, terwijl de discriminator op schaal $\frac{1}{8}$ globale fouten, zoals een inconsistent kleurgebruik, afstraft.

De architectuur van elk individueel discriminatornetwerk is hetzelfde als in de originele pix2pix implementatie [4]. Zo is elk van de drie discriminatornetwerken een *PatchGAN* die een 70 x 70 patch classificeert als echt of vals. Dit discriminatornetwerk wordt convolutioneel toegepast over

de inkleuring waarna de uitvoer het gemiddelde is van alle activaties. De som van de uitvoer van elk van de drie discriminatornetwerken vormt vervolgens de uiteindelijke discriminator loss. Belangrijk op te merken is dat de discriminator enkel de gekleurde uitvoer van de generator te zien krijgt en niet de lijntekening die als invoer gebruikt werd. De discriminator is met andere woorden niet geconditioneerd op de invoer. Dit is een bewuste keuze aangezien de L_1-loss reeds verantwoordelijk is voor de correspondentie tussen invoer en uitvoer. Daardoor is het niet mogelijk dat de generator een prent genereert die ongerelateerd is aan de invoer puur om de discriminator te misleiden. De toepassing en architectuur van de discriminator worden voorgesteld in figuur 4.5.

Vervolgens trainen we het netwerk opnieuw met de L_1-loss en discriminator loss. De inkleuring van bovenstaande prent door dit nieuwe netwerk is weergegeven in figuur 4.4b. We zien dat een groot deel van de eerdere fouten nu weggewerkt zijn. De uitvoer bevat geen vlekken meer en de kleuren zijn een stuk levendiger.

4.3.3 Total variation loss

Ondanks dat de discriminator loss het aantal ongewenste kleurovergang sterk reduceert, komen deze toch nog regelmatig voor. Dit is problematisch aangezien dergelijke overgangen sterk afbreuk doen aan de geloofwaardigheid van de uitvoer. Niettegenstaande talloze experimenten met de architectuur en invloed van de discriminator, blijven deze fouten echter meer aanwezig dan we zouden hopen. Een voorbeeld van zo'n overgang wordt geïllustreerd in figuur 4.6c. Merk op hoe de inkleuring door een netwerk getraind op basis van de L_1-loss en discriminator loss (figuur 4.6c) veel kleurovergangen bevat die niet aanwezig zijn in de ground truth (figuur 4.6b). Zo zijn er storende kleurovergangen op de huid en kledij van de afgebeelde personages. Aangezien de discriminator deze fout niet voldoende afstraft, voegen we een derde term toe aan de loss-functie die dit type fout uitdrukkelijk afstraft. We hopen op die manier het aantal kleurovergangen te beperken en zo een realistischer resultaat te bekomen.

Een simpele benadering is om kleurovergangen binnenin één connected component niet toe te laten. Dit is echter niet correct aangezien we in sectie 2.3.2 zagen dat dergelijke kleurovergangen soms wel noodzakelijk en gewenst zijn. We proberen er dan ook voor te zorgen dat kleurovergangen hard genoeg worden afgestraft zodat het netwerk ze niet nodeloos introduceert, maar niet zo hard dat het netwerk ze achterwege laat als ze effectief nodig zijn. Op die manier heeft het netwerk nog steeds de vrijheid om een kleurovergang te introduceren, maar wordt dit over het algemeen ontmoedigd.

Om dergelijke kleurovergangen af te straffen, introduceren we een nieuwe loss term gebaseerd op *total variation regularization* [71, 72], een proces dat traditioneel gebruikt wordt voor ruisverwijdering. Het achterliggende idee daar is dat ruis en andere artefacten leiden tot een hoge total variation, d.w.z. de som van de absolute gradiënt van de afbeelding is groot. Toegepast

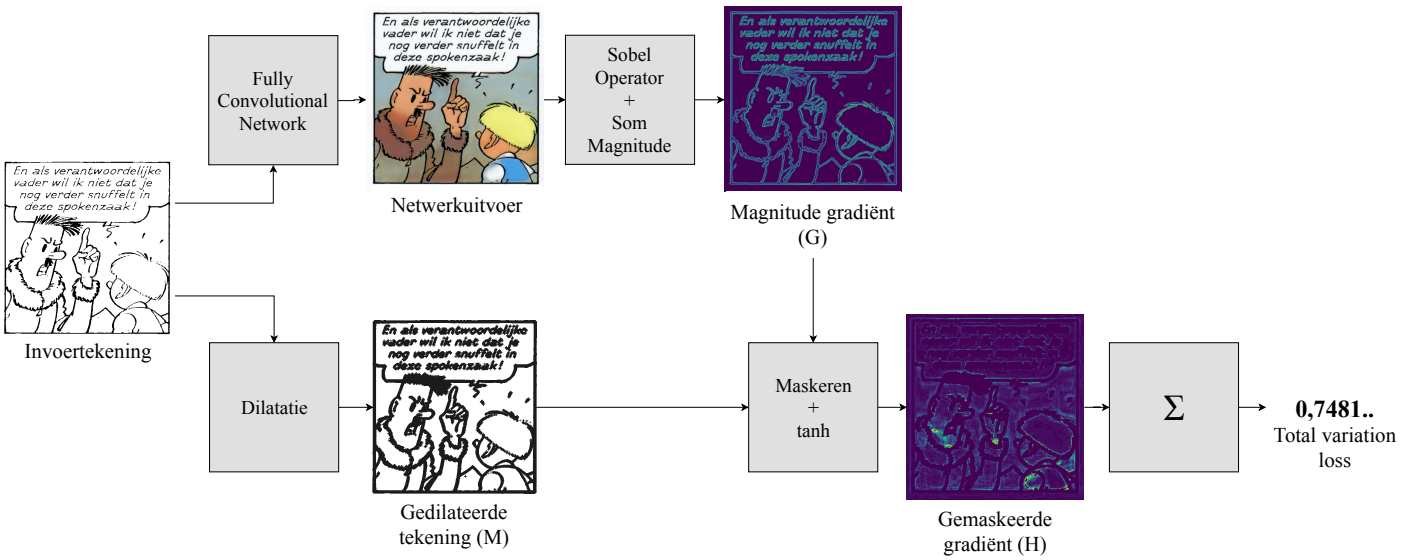
(a) Invoerafbeelding

(b) Ground truth

(c) Inkleuring met netwerk getraind op basis van de L_1-loss en discriminator loss

(d) Inkleuring met netwerk getraind op basis van de L_1-loss, discriminator loss en total variation loss

Figuur 4.6: De invloed van de total variation loss



Figuur 4.7: Schematische weergave van de berekening van de total variation loss

op ons geval, komt het vermijden van kleurovergangen eveneens neer op het minimaliseren van de total variation van de uitvoerafbeelding [73]. Een kleine magnitude van de gradiënt betekent namelijk dat de afbeelding niet veel verandert op dat punt en dat er dus geen kleurovergang is op die plaats. Aangezien de total variation de som neemt over al deze magnitudes, bekomen we zo een maat voor het aantal aanwezige kleurovergangen.

Omdat we in deze sectie vaak gebruik maken van de spatiale structuur van afbeeldingen, stellen we deze hier voor als matrices in plaats van vectoren. De inkleuring heeft dan de vorm $\hat{\mathbf{Y}} \in \mathbb{R}^{H \times W \times 3}$, in plaats van het eerder gebruikte $\hat{\mathbf{y}} \in \mathbb{R}^{H \cdot W \cdot 3}$. De magnitude van de discrete gradiënt op elk punt berekenen komt neer op het toepassen van de Sobel operator [74]. Traditioneel is deze operator enkel gedefinieerd voor grijswaardenafbeeldingen. We beginnen dus met de Sobel operator afzonderlijk toe te passen op elk van de drie kanalen van de inkleuring $\hat{\mathbf{Y}} \in \mathbb{R}^{H \times W \times 3}$ om zo de drie twee-dimensionale matrices $\mathbf{G_L}, \mathbf{G_A}, \mathbf{G_B} \in \mathbb{R}^{H \times W}$ te bekomen. Deze drie matrices drukken de magnitude van de gradiënt op elk punt uit voor het corresponderende kanaal en zijn als volgt gedefinieerd:

$$\mathbf{G_L} = \sqrt{\mathbf{G_{L,x}}^2 + \mathbf{G_{L,y}}^2} \qquad \mathbf{G_A} = \sqrt{\mathbf{G_{A,x}}^2 + \mathbf{G_{A,y}}^2} \qquad \mathbf{G_B} = \sqrt{\mathbf{G_{B,x}}^2 + \mathbf{G_{B,y}}^2} \tag{4.2}$$

Hierbij stellen de matrices $\mathbf{G_{\bullet,x}}$ en $\mathbf{G_{\bullet,y}}$ de gradiënt in respectievelijk de horizontale en verticale richting van het kanaal $\bullet$ voor. Deze worden berekend als de convolutie van dat kanaal met de horizontale of de verticale Sobel kernel:

$$\mathbf{G_{\bullet,x}} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \hat{\mathbf{Y}}_\bullet \qquad \mathbf{G_{\bullet,y}} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \hat{\mathbf{Y}}_\bullet \tag{4.3}$$

De samenvattende gradiëntmatrix $\mathbf{G}$ definiëren we dan als de som van de magnitude van de gradiënt over ieder kanaal, oftewel:

$$\mathbf{G} = \mathbf{G_L} + \mathbf{G_A} + \mathbf{G_B} \tag{4.4}$$

Als we deze procedure toepassen op de ingekleurde afbeelding in figuur 4.6c, ziet de matrix $\mathbf{G}$ eruit zoals weergegeven in figuur 4.7. We zien dat de gradiënt volledig overheerst wordt door de overgangen tussen kleurvlak en lijn. De magnitude van de gradiënt bij fouten binnenin een kleurvlak is namelijk typisch klein ten opzichte van de magnitude van de gradiënt rondom zwarte lijnen. We willen deze overgangen tussen kleurvlak en zwarte lijn echter niet afstraffen aangezien deze wel gewenst zijn. Daarom maskeren we de gradiënt met een gedilateerde versie van de lijnafbeelding (figuur 4.6a) die als invoer werd gebruikt. Op die manier dragen de lijnen die reeds aanwezig waren in de invoer niet bij tot de loss term wanneer deze worden overgenomen in de uitvoer. Het masker $\mathbf{M}$ is een twee-dimensionale matrix met dezelfde afmetingen als de invoerafbeelding en is als volgt gedefinïeerd:

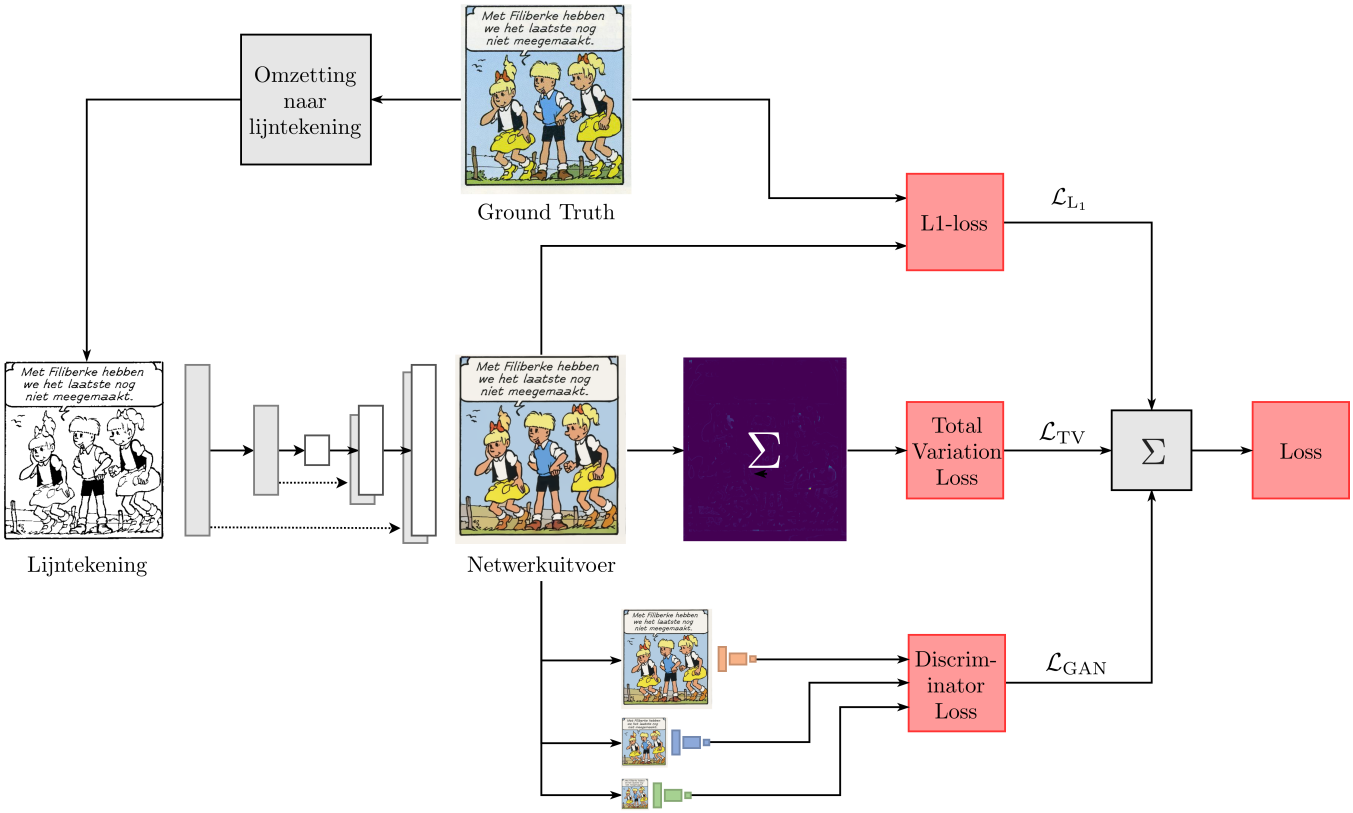
$$\mathbf{M}_{i,j} = \begin{cases} 0 & \text{indien de binaire lijntekening minstens één zwarte pixel} \\ & \text{bevat in een omgeving van 3 pixels rond de pixel } (i,j) \\ 1 & \text{anders} \end{cases}$$

Intuïtief komt de matrix $\mathbf{M}$ tot stand door de oorspronkelijke zwarte lijnen breder te maken. Zo bekomen we een masker dat enkel een 1 bevat voor pixels die binnenin een kleurvlak liggen en zich niet in de directe omgeving van een lijn bevinden. Tenslotte passen we op elk element van de gemaskeerde gradiëntmatrix de hyperbolische tangensfunctie toe. Dit is een sigmoïde-vormige functie die we gebruiken om de impact van uitschieters, d.w.z. uitzonderlijk grote magnitudes, beperkt te houden. De uiteindelijke matrix $\mathbf{H}$ is dan gedefinïeerd als volgt:

$$\mathbf{H}_{i,j} = \mathbf{M}_{i,j} \cdot \tanh \mathbf{G}_{i,j} \tag{4.5}$$

In afbeelding van matrix $\mathbf{H}$ in figuur 4.7 is het duidelijk dat de resterende grote magnitudes overeenstemmen met fouten in de inkleuring 4.6c. Het beperken van ongewenste kleurovergangen in de uitvoer $\hat{\mathbf{Y}}$, hebben we dus herleid tot het minimaliseren van de elementen van de gemaskeerde gradiënt-matrix $\mathbf{H}$. Op dit punt kunnen we dus makkelijk een nieuwe loss-term $\mathcal{L}_{\text{TV}}$ definiëren om kleurovergangen af te straffen. Deze nieuwe term is simpelweg het gemiddelde van alle elementen in $\mathbf{H}$:

$$\mathcal{L}_{\text{TV}} = \frac{1}{H \cdot W} \sum_{h,w} \mathbf{H}_{h,w} \tag{4.6}$$



Figuur 4.8: Architectuur van onze oplossing

De uiteindelijke loss-functie bestaat dan uit drie termen: de L_1-loss, de discriminator loss en de zojuist geïntroduceerde total variation loss. Merk op dat elk van deze termen opgebouwd is uit afleidbare functies waardoor we in staat zijn backpropagation toe te passen om de gewichten van het netwerk te optimaliseren. Deze uiteindelijke architectuur is schematisch weergegeven in figuur 4.8.

4.4 Trainingsdetails

De hierboven besproken architectuur werd geïmplementeerd in PyTorch, een Python library voor onder andere deep learning [75]. Andere opties die overwogen werden zijn TensorFlow [76], Caffe [77] en Keras [78]. Uiteindelijk kozen we voor PyTorch aangezien dit framework gericht is op onderzoek waardoor we snel kunnen itereren. Elk van de hierboven vermelde frameworks had echter gebruikt kunnen worden aangezien de onderliggende principes telkens dezelfde zijn.

Het overgrote deel van de code is door ons geschreven in Python. Voor de implementatie van de

GAN-component, werd inspiratie gehaald bij de referentie-implementatie van pix2pix [4]. Het uiteindelijke netwerk werd vervolgens end-to-end getraind met de hierboven besproken architectuur en loss-functie. In wat volgt wordt dieper ingegaan op de details van dit trainingsproces.

4.4.1 Varianten

We bespraken eerder dat de uiteindelijke loss-functie bestaat uit drie afzonderlijke termen. Om meer inzicht te krijgen in de invloed van elk van deze termen trainen we drie varianten van het netwerk. Hierbij blijft de netwerkarchitectuur telkens dezelfde en verandert enkel de opbouw van de loss-functie. Concreet trainen we de volgende drie varianten:

- L_1-loss
- L_1-loss + discriminator loss
- L_1-loss + discriminator loss + total variation loss

Hierbij is de laatste variant het meest geavanceerd. De andere varianten zijn variaties waarbij telkens slechts enkele van de drie termen van de uiteindelijke loss-functie gebruikt worden.

Elk van deze varianten werd vanaf nul getraind voor 150 000 stappen op een GeForce GTX 1080 met 8 GB geheugen. Dit komt overeen met ongeveer één week trainingstijd per variant. Naast de hoge trainingstijd, zorgde ook de beschikbare hoeveelheid geheugen voor een uitdaging. We willen het netwerk namelijk zoveel mogelijk context geven, maar worden daarbij beperkt door de hoeveelheid GPU-geheugen. We kiezen er dan ook voor om de batch grootte in te stellen op 1 en het netwerk crops van 900 x 900 aan te leveren. Dit is immers hoogste resolutie die nog net in het beschikbare GPU-geheugen past.

4.4.2 Initialisatie

Voor de initialisatie van de gebruikte lagen werd geëxperimenteerd met verschillende initialisatiefuncties. Specifiek werden LeCunn initialisatie [79], Xavier initialisatie [80] en Kaiming Uniform initialisatie [14] getest. De keuze tussen deze drie methoden leek echter geen invloed te hebben op de convergentie of de uiteindelijke performantie. We kiezen er dan ook voor om de standaardoptie in PyTorch, zijnde LeCunn initialisatie, te gebruiken.

4.4.3 Optimalisatie

Net als voor de initialisatie, zijn er voor het updaten van de gewichten ook heel wat mogelijkheden. Traditioneel was stochastisch gradient descent (SGD) de standaard voor het trainen van CNNs. Later werden AdaGrad [81] en RMSProp [82] geïntroduceerd, waarna deze erg populair worden. Na de introductie van Adam [83] in 2015, is dit echter de methode bij uitstek voor het trainen van CNNs. Tegenwoordig wordt in zo goed als elk werk rond deep learning gebruik gemaakt van de Adam optimizer vanwege de goede performantie. Wij kiezen er daarom ook voor om gebruik te maken van deze methode om de gewichten te updaten.

4.4.4 GAN

Cruciaal voor onze uiteindelijke methode is het gebruik van een generative adversarial network. Om deze GAN te trainen maken we gebruik van de standaardprocedure die voorgesteld werd door Goodfellow et al. [29]. Hierbij wisselen we af tussen een gradient descent stap op de discriminator D, en een stap op de generator G. Ondanks de eenvoud van deze procedure, divergeert de training in de praktijk vaak. Doorheen de jaren zijn er dan ook heel wat trucjes ontdekt die deze training net iets stabieler maken [84, 3, 31]. Voor veel van deze trucjes is er helaas nog geen sluitende verklaring waarom ze precies een gunstig effect hebben. Desondanks blijken ze erg nuttig in de praktijk. Hieronder gaan we in op enkele methodes die we gebruiken om de GAN training te stabiliseren.

Normalisatie van de invoer

De invoer is standaard een afbeelding in de CIELAB kleurruimte. Dit betekent dat het L^* kanaal varieert van 0 tot 100, terwijl het a^* en b^* kanaal beide variëren tussen -128 en 127. In plaats van deze invoer direct aan het netwerk te geven, normaliseren we deze eerst zodat elk kanaal tussen -1 en 1 varieert.

Gemodificeerde loss-functie

Zoals in sectie 2.5.1 besproken, is een GAN in feite een minimax spel. Hierbij probeert de discriminator D de volgende waarde-functie te maximaliseren, terwijl de generator G deze wil minimaliseren:

$$\mathcal{L}_{\text{cGAN}}(G, D) = \mathbb{E}_{\mathbf{x},\mathbf{y}}[\log D(\mathbf{x}, \mathbf{y})] + \mathbb{E}_{\mathbf{x},\mathbf{z}}[\log(1 - D(\mathbf{x}, G(\mathbf{x}, \mathbf{z})))] \tag{4.7}$$

Hoewel deze formulering wiskundig correct is, leidt ze in de praktijk tot vanishing gradients. We passen de procedure dan ook lichtjes aan zodat we bij het trainen van G, in plaats van $\log(1 - D(\mathbf{x}, G(\mathbf{x}, \mathbf{z})))$ te minimaliseren, we $\log D(\mathbf{x}, G(\mathbf{x}, \mathbf{z}))$ maximaliseren.

Batch normalisatie

Een andere zinvolle aanpassing is het gebruik van batch normalisatie [63], aangezien dit leidt tot een grotere diversiteit in de gegenereerde samples. In ons geval is het gebruik van batch normalisatie echter geen optie. De batch grootte is namelijk gelijk aan 1 vanwege beperkingen in het beschikbare GPU-geheugen, waardoor batch statistieken niet nuttig zijn. Zoals in sectie 4.2.2 reeds aangehaald, maken we in plaats daarvan gebruik van instance normalisation [65]. Deze techniek biedt een deel van de voordelen van batch normalisatie en werkt onafhankelijk van de andere afbeeldingen in de batch. Hierdoor kan instance normalization zonder problemen gebruikt worden voor batches van slechts één afbeelding.

Vermijden van afwezige gradiënten

De stabiliteit van de GAN training lijdt onder het verdwijnen van gradiënten. Om dit te vermijden, maken we telkens gebruik van LeakyReLU [85] in de discriminator. Op die manier hebben we nog steeds een kleine, niet-nul gradiënt wanneer de unit niet actief is. Zoals voorgesteld door Radford et al., blijven we in de generator normale ReLU activaties gebruiken [3].

Bijhouden van een buffer

Een andere probleem bij adversarial training is dat de discriminator zich enkel gaat focussen op de meest recent gegenereerde afbeeldingen [86]. Hierdoor kan de training divergeren of herintroduceert de generator fouten die de discriminator reeds vergeten is. Elke vorige afbeelding die gegenereerd is door de generator tijdens de training is echter 'vals'. De discriminator zou dus in staat moeten zijn elk van deze afbeeldingen als vals te classificeren. Gebaseerd op deze observatie kiezen we ervoor om een buffer van 50 valse afbeeldingen bij te houden. We kiezen dan telkens een willekeurige afbeelding uit deze buffer om aan de discriminator te tonen. Zo vermijden dat de discriminator vergeet wat hij eerder leerde.

Opvolgen van training

Zelfs met het toepassen van bovenstaande trucjes, blijft het trainen van een GAN een erg gevoelig en moeilijk proces. Tijdens de training verzamelden we daarom voortdurend informatie over de loss van de generator, de loss van de discriminator, de gradiënten en de gegenereerde afbeeldingen. Op die manier waren we in staat het trainingsproces nauwgezet op te volgen en in te grijpen bij problemen. Een veelvoorkomend probleem is bijvoorbeeld dat de loss van de discriminator naar 0 gaat. Hoewel de discriminator dan elke afbeelding perfect als vals of echt kan beoordelen, hebben we in dat geval amper gradiënten om de generator te verbeteren. In dat geval moet de discriminator bijvoorbeeld minder krachtig worden gemaakt. Na talloze experimenten waren we uiteindelijk in staat een opstelling te bekomen die stabiel trainde.

Hoofdstuk 5

Evaluatie en experimenten

In het vorige hoofdstuk ontwikkelden we een methode voor automatische inkleuring. Daarbij trainden we naast onze volledige methode nog twee andere varianten van het netwerk. In dit hoofdstuk analyseren we de prestaties van deze varianten en voeren we experimenten uit om meer inzicht te krijgen in hun gedrag. We beginnen met het vergelijken van de varianten op basis van de L_1-loss. Vervolgens voeren we testen uit met menselijke waarnemers om kwalitatief te kunnen vergelijken. Tenslotte bespreken we enkele veelvoorkomende fouten en proberen we onze methode uit op invoerafbeeldingen uit andere datasets.

5.1 Testopstelling

In sectie 3.3 bespraken we de opsplitsing van de dataset in een train en test set. Om de performantie van onze methode te evalueren, maken we uitsluitend gebruik van de test set. We brengen hier even in herinnering dat deze test set bestaat uit drie deelverzamelingen:

- **Gezien album**: 120 pagina's die werden weggelaten uit albums in de train set maar waarvan de overige pagina's van het album zich wel in de train set bevinden.
- **Ongezien album**: 387 pagina's uit 8 verschillende albums waarvan geen enkele pagina van het album zich in de train set bevindt.
- **Zwart-wit album**: 2818 pagina's uit albums die oorspronkelijk in zwart-wit verschenen en waarvoor dus geen ground truth beschikbaar is.

Aangezien we onze resultaten willen vergelijken met de ground truth, maken we hier enkel gebruik van de verzamelingen *gezien album* en *ongezien album*. Het toepassen van onze methode op albums die oorspronkelijk in zwart-wit verschenen is het onderwerp van het volgend hoofdstuk. Tenslotte merken we nogmaals op dat elke afbeelding waarop we testen een ongeziene afbeelding is voor het netwerk. Het verschil tussen gezien en ongezien album zit in de aanwezigheid van de overige pagina's in het album in de training set. De pagina in de test set is sowieso zelf nooit

aanwezig in de train set.

Het doel van deze splitsing in gezien en ongezien album is om meer inzicht te krijgen in de uiteindelijke resultaten. Veel voorwerpen duiken namelijk slechts op in één album. Wanneer we evalueren op de verzameling ongezien album, heeft het netwerk dus nooit de kans gehad om de inkleuring van deze voorwerpen te leren. We verwachten daarom dat de resultaten voor ongezien album typisch iets minder goed zullen zijn. Dus splitsen we de test set om dit verschil precies te kunnen kwantificeren.

Naast een testverzameling, hebben we uiteraard ook een netwerk nodig waarvan we de performantie willen evalueren. In sectie 4.4.1 zagen we dat we drie varianten ter beschikking hebben. De netwerkarchitectuur en trainingsprocedure is, op het weglaten van bepaalde loss-termen na, telkens volledig identiek. De namen van deze drie varianten duiden de gebruikte loss-termen aan:

- L_1-loss
- L_1-loss + discriminator loss
- L_1-loss + discriminator loss + total variation loss

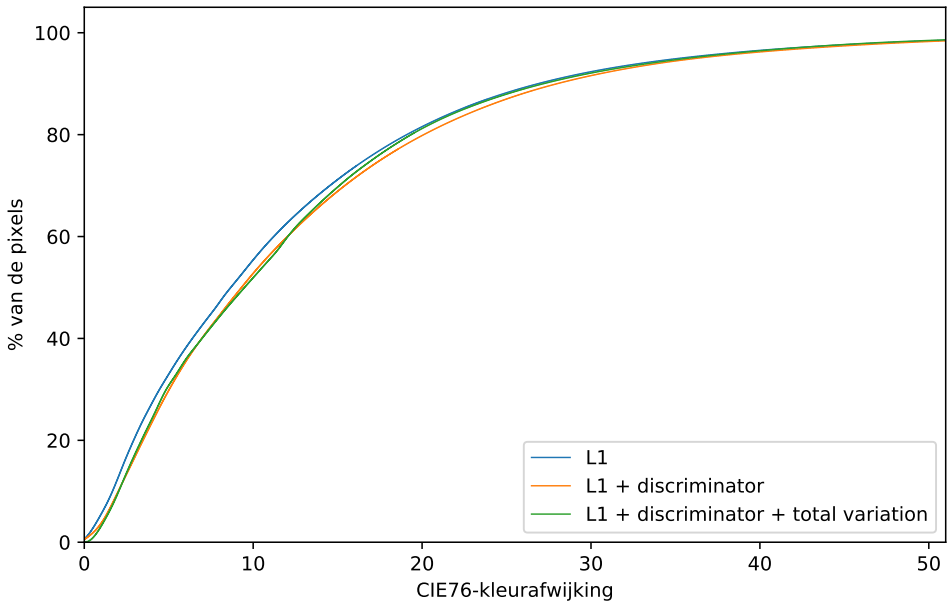
De laatste variant, waarbij elke loss-term aanwezig is, is onze volledige methode. We verwachten dat deze variant uiteindelijk het best zal presteren. Het trainen van de andere varianten laat ons toe om de invloed van elke loss-term afzonderlijk te bepalen.

5.2 CIE76-kleurafwijking

De eerste metriek die we gebruiken is de CIE76-kleurafwijking tussen de uitvoer van ons netwerk en de ground truth voor elke pixel. Deze afstandsmetriek, ook wel ΔE^*_{ab} genoemd, is specifiek ontworpen voor perceptueel uniformiteit [87]. Dit betekent dat dezelfde hoeveelheid numerieke verandering ongeveer overeenkomt met dezelfde hoeveelheid visueel waargenomen verandering. Later werd de formule uitgebreid om nauwkeuriger te zijn in enkele probleemgebieden [88, 89]. De originele, eenvoudige formule is echter meer dan accuraat genoeg voor onze doeleinden. Concreet wordt de afwijking tussen twee kleuren in de CIELAB kleurruimte, (L^*_1, a^*_1, b^*_1) en (L^*_2, a^*_2, b^*_2) als volgt uitgedrukt:

$$\Delta E^*_{ab} = \sqrt{(L^*_2 - L^*_1)^2 + (a^*_2 - a^*_1)^2 + (b^*_2 - b^*_1)^2} \tag{5.1}$$

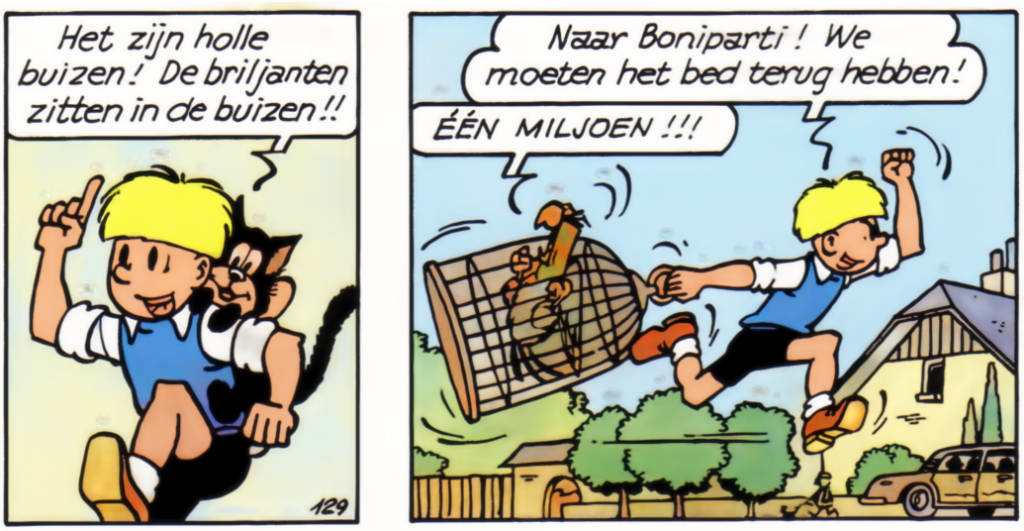
We berekenen deze kleurafwijking voor alle pixels van alle afbeeldingen in de dataset. In tabel 5.1 is de gemiddelde afwijking over alle pixels voorgesteld voor elk van de getrainde varianten. Verder wordt in figuur 5.1 een cumulatieve distributiefunctie getoond van de afwijking per pixel voor elke variant. Voor beide test sets zien we dat de relatieve ordering hetzelfde is en dat het netwerk getraind via enkel de L_1-loss de laagste afwijking behaalt. Dit is exact wat we verwachten aangezien deze variant zich er louter op toespitst om de kleurafstand tussen de ground truth en de uitvoer te minimaliseren.



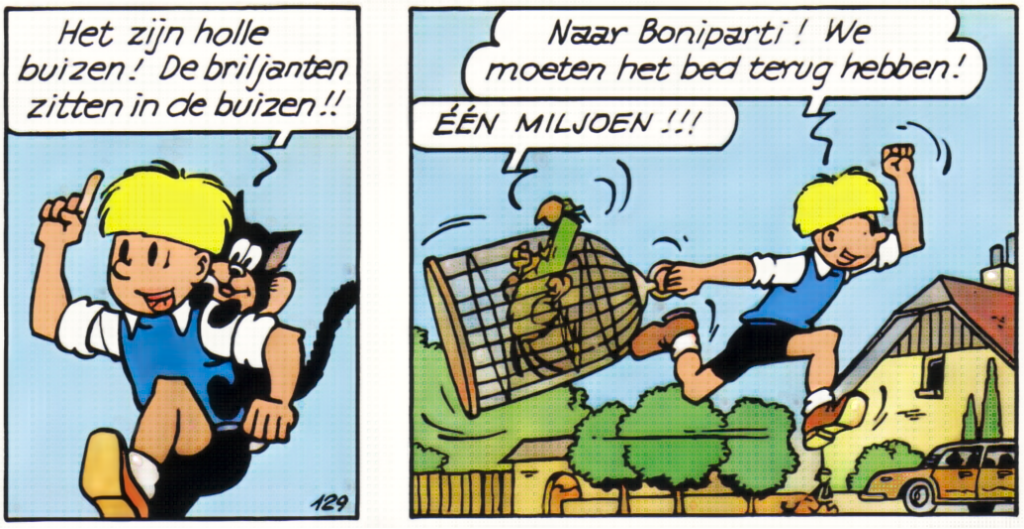
Figuur 5.1: Cumulatieve distributiefunctie van de gemiddelde CIE76 kleurafwijking per pixel voor de categorieën gezien album en ongezien album

De twee andere varianten daarentegen, proberen naast een kleine kleurafstand ook te gaan voor een geloofwaardig resultaat. Dit verschil tussen het netwerk met en zonder discriminator loss wordt geïllustreerd in figuur 5.2. In het grootste deel van de gevallen zijn beide netwerken het eens over de inkleuring, bijvoorbeeld voor de bomen en kledij. De inkleuring daarvan is voor beide varianten min of meer gelijk aangezien deze voorwerpen vaak voorkomen en beide netwerken dus hetzelfde idee hebben over de te gebruiken kleur.

Het verschil tussen trainen met of zonder discriminator loss wordt vooral duidelijk wanneer de correct kleur niet eenduidig bepaald is, zoals bij het dak van het huis of de achtergrond van de linkerprent. De L_1-loss variant kiest dan gedesatureerde kleuren in de hoop de afstand tot de werkelijke kleur te minimaliseren (zie figuur 5.2). De variant met discriminator loss weet echter dat dergelijke grauwe, ongeloofwaardige kleuren afgestraft worden door de discriminator en kiest dan ook om te gokken op levendige kleuren. Dit wordt aangemoedigd door de discriminator loss, maar afgestraft door de L_1-loss wanneer deze gok ver van de ground truth blijkt te liggen. Hierdoor is de L_1-loss hoger bij het trainen met discriminator loss, terwijl de inkleuringen vaak wel realistischer zijn. Als de achtergrond van de linkerprent bijvoorbeeld groen is in de ground truth, dan heeft inkleuring 5.2b de laagste gemiddelde kleurafwijking terwijl deze toch minder geloofwaardig overkomt. Algemeen kunnen we stellen dat de inkleuring na het trainen met discriminator loss een stuk levendiger en geloofwaardiger is dan zonder.



(a) Uitvoer van netwerk getraind met L_1-loss



(b) Uitvoer van netwerk getraind met L_1 en discriminator loss

Figuur 5.2: Verschil tussen inkleuring met en zonder discriminator loss. Merk op hoe de variant met discriminator loss leidt tot levendigere kleuren.

Variant	**CIE76-kleurafwijking (per pixel)**
L1	12,59
L1 + discriminator	13,39
L1 + discriminator + total variation	13,18
Ground truth	0,00

(a) Resultaten voor de dataset *gezien album*

Variant	**CIE76-kleurafwijking (per pixel)**
L1	14,04
L1 + discriminator	14,37
L1 + discriminator + total variation	14,23
Ground truth	0,00

(b) Resultaten voor de dataset *ongezien album*

Tabel 5.1: De gemiddelde resultaten voor de twee dataset en drie varianten

5.3 Perceptuele experimenten

In de vorige sectie zagen we dat we voorzichtig dienen te zijn met ruwe performantiemetrieken. Deze kunnen namelijk een vertekend beeld geven van het resultaat van onze methode. Zoals reeds uitvoerig besproken, is het doel namelijk niet noodzakelijk om dicht bij de ground truth aan te leunen maar wel om een *geloofwaardige* inkleuring te bekomen. Om in kaart te brengen hoe overtuigend onze inkleuringen zijn, zetten we een experiment op waarbij menselijke waarnemers deze beoordelen.

5.3.1 Opzet

De ultieme test is hoe geloofwaardig onze inkleuringen overkomen voor een menselijk waarnemer. We willen met andere woorden kwantificeren in hoeverre mensen de automatische inkleuringen kunnen onderscheiden van originele kleurprenten. Om dit te doen, voerden we een *echt of vals* experiment uit waarbij deelnemers gekleurde strippprenten dienen te beoordelen als ofwel echt ofwel vals. Hierbij volgen we hetzelfde protocol als Iizuka et al. [90]. Om het experiment uit te voeren en de resultaten te verzamelen, bouwden we een webapplicatie. We beschrijven nu het precieze verloop van een sessie.

Het experiment start met een korte introductie over het onderwerp van deze thesis. Hierna krijgen de deelnemer instructies te zijn over wat er precies verwacht wordt (zie figuur 5.3). De deelnemer wordt zo geïnformeerd dat hij een reeks strippprenten zal te zien krijgen waarvoor hij telkens dient aan te geven als deze natuurlijk lijken of niet. Hierbij worden de deelnemers aangemoedigd om te vertrouwen op hun buikgevoel en geen tijd te verliezen met het lezen van tekstballonnetjes of overdreven op details te letten. We willen dus dat deelnemers zich gedragen alsof ze deze prent (met uitzondering van de tekst) zouden zien staan in een willekeurig Jommeke-album.

Nadat de deelnemer bevestigt dat deze instructies duidelijk zijn, worden de prenten gepresenteerd. We kiezen ervoor om individuele prenten te tonen in plaats van volledige pagina's omdat we op die manier meer informatie halen uit het oordeel van de deelnemer. De kans is namelijk groot dat er zich in minstens één van de prenten van een volledige pagina een ongeloofwaardige artefact bevindt. Wanneer we volledige pagina's tonen, zou dit ertoe leiden dat de volledige pagina als vals bestempeld zou worden door één detail. Hierdoor zouden we geen informatie krijgen over welke prenten op de pagina de gebruiker al dan niet geloofwaardig vindt.

Bij het afnemen van de testen willen we verhinderen dat deelnemers zich te sterk focussen op details waar ze niet op zouden letten tijdens het lezen van een stripalbum. Daarom limiteren we de tijd die beschikbaar is om elke prent te bekijken. Om een geschikte tijdsduur te bepalen lieten we vooraf 8 personen elk 10 willekeurige pagina's uit een Jommeke-album lezen. Deze personen deden er gemiddeld 2 minuten en 15 seconden over om 10 pagina's te lezen met gemiddeld 9 prenten per pagina. Uit dit experiment leiden we af dat 2,83 seconden per prent een gemiddeld leestempo is. Op basis hiervan beslisten we elke prent te tonen gedurende 3 seconden (zie figuur

		Verzameling		
		Gezien album	Ongezien album	Zwart-witalbum
Variant	L_1	9	9	-
	L_1 + discriminator	9	9	-
	L_1 + discriminator + total variation	9	9	30
	Ground truth	3	3	N.B.

Tabel 5.2: Het aantal prenten dat per variant en verzameling wordt getoond aan de deelnemer. Merk op dat er geen ground truth beschikbaar is voor prenten uit originele zwart-witalbums.

5.4a). Hierna krijgen de deelnemers onbeperkte tijd om aan te duiden als de prent al dan niet geloofwaardig lijkt (zie figuur 5.4b). Vervolgens verschijnt de volgende prent en begint de klok opnieuw te lopen.

In totaal krijgt elke deelnemer 90 willekeurige prenten te zien, waardoor het experiment ongeveer acht minuten duurt. Deze prenten worden willekeurig gekozen volgens het schema voorgesteld in tabel 5.2. De achterliggende dataset is een verzameling van 300 prenten die vooraf willekeurig gekozen werden en elk in vier varianten beschikbaar zijn (drie netwerkvarianten + ground truth). Voor de twee verzamelingen gezien album en ongezien album tonen we telkens 9 afbeeldingen voor elk van de drie getrainde varianten. Verder tonen we ook 3 ground truth afbeeldingen per verzameling om te verifiëren dat de deelnemer de taak correct uitvoert. De overige 30 prenten bestaan uit onze volledige methode (L_1 + discriminator + total variation) toegepast op albums die in zwart-wit verschenen en waarvan dus geen ground truth beschikbaar is.

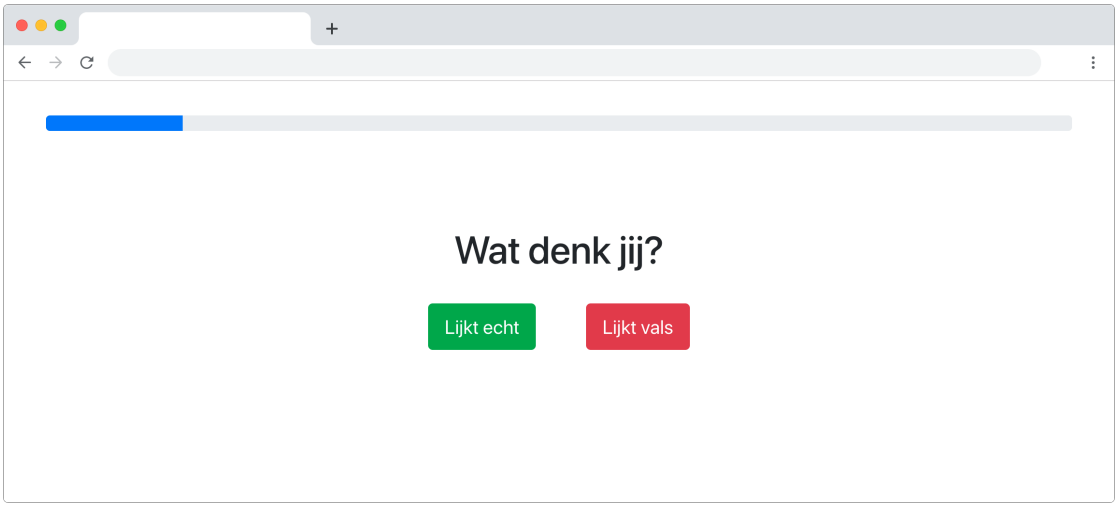
5.3.2 Resultaten

In totaal namen 84 personen deel aan de experimenten, waardoor we 7560 beoordelingen hebben van prenten als echt of vals. Het aantal prenten dat als echt werd bestempeld per verzameling en variant is weergegeven in tabel 5.3. Dezelfde data wordt ook getoond als een staafdiagram in figuur 5.7 om de vergelijking tussen de varianten te vergemakkelijken. Het eerste dat we opmerken is dat deelnemers de ground truth afbeeldingen in 92,6 % van de gevallen als echt beoordeelden. Dit toont aan dat deelnemers over het algemeen competent zijn maar wel kritisch in het geval van twijfel.

Vervolgens kijken we naar de performantie van het netwerk getraind met enkel L_1-loss. We zien dat dit netwerk een geloofwaardigheid van 55,8 % haalt op de dataset met geziene albums en een geloofwaardigheid van 50,2 % op ongeziene albums. Deze afname in performantie is precies wat we voorspelden in sectie 5.1 aangezien deze tweede verzameling vaak ongeziene voorwerpen bevat. Wanneer we het netwerk L_1 + discriminator beschouwen, verwachten we dat het

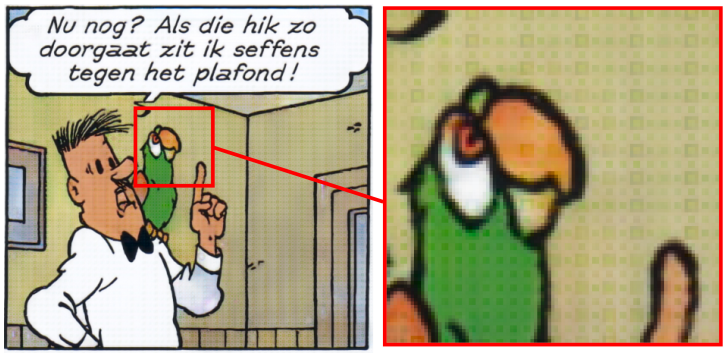
Figuur 5.3: De instructies die gegeven werden bij aanvang van het perceptuele experiment

(a) Eerst wordt een afbeelding getoond gedurende 3 seconden.



(b) Vervolgens beslist de gebruiker als de afbeelding al dan niet echt lijkt.

Figuur 5.4: De gebruikersinterface die werd gebruikt voor de perceptuele experimenten

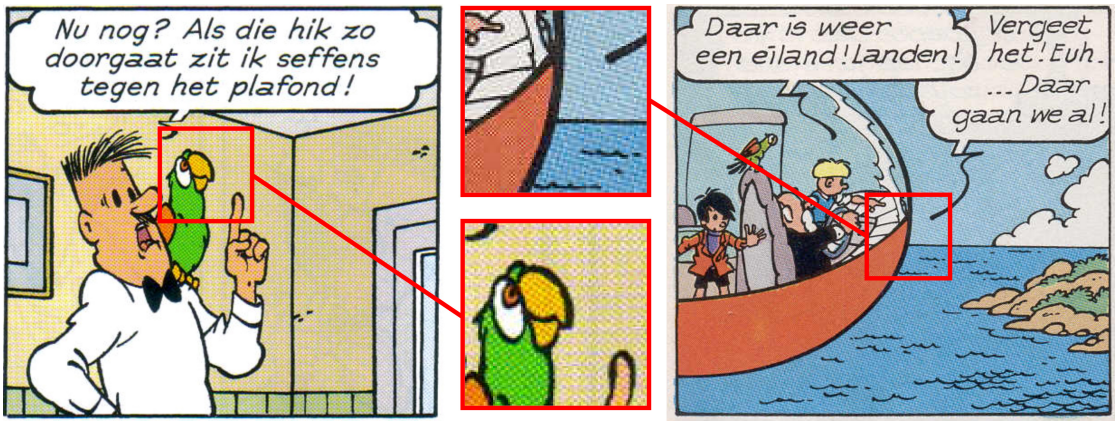


Figuur 5.5: Inkleuring door het netwerk L_1 + discriminator. Merk op hoe de inkleuring storende patronen bevat.

verschil daar kleiner is. Het netwerk dat getraind is met discriminator loss zou namelijk tot realistischere inkleuringen moeten leiden in het geval van twijfel. Dit is eveneens precies wat we observeren aangezien het verschil bij die netwerkvariant slecht 3,1 % in plaats van 5,6 % bedraagt.

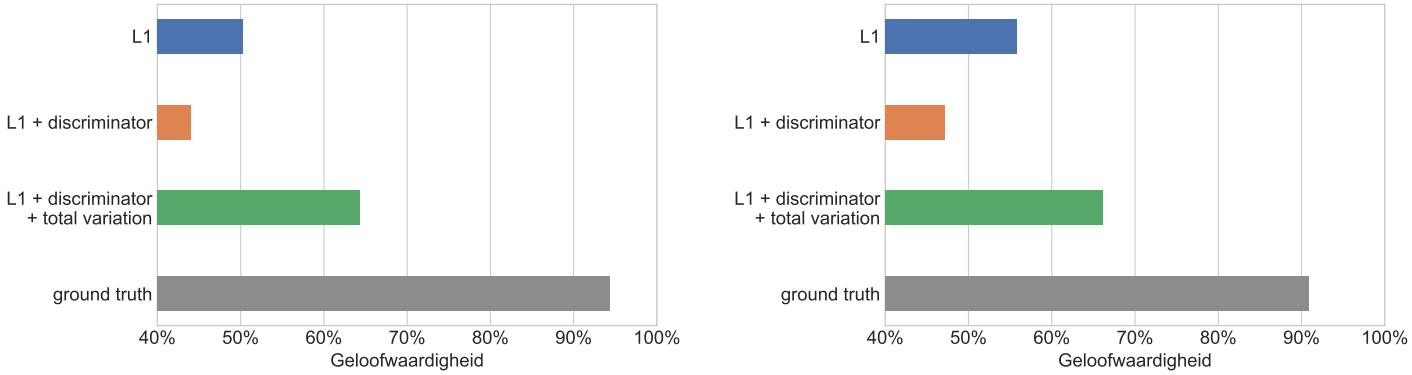
Opmerkelijk is dat de geloofwaardigheid van de netwerkvariant L_1 + discriminator een stuk lager is dan de variant L_1. Intuïtief verwachten we namelijk dat de discriminator loss zou leiden tot geloofwaardigere resultaten. Wanneer we echter de inkleuringen gegenereerd door deze netwerkvariant bekijken, wordt duidelijk waarom deze als minder geloofwaardig ervaren worden. De inkleuringen van dit netwerk bevatten namelijk een storend patroon dat zich steeds herhaalt (zie figuur 5.5). We vermoeden dat dit patroon het resultaat is van een combinatie van twee factoren: de aanwezigheid van vergelijkbare patronen in de invoer en de strided convolutions in de discriminator. Wanneer we de ground truth afbeeldingen bekijken, zien we dat deze ook vaak storende patronen bevatten (zie figuur 5.6). Deze patronen zijn te wijten het gebruik van dithering bij het afdrukken. De generator zal dus, onder invloed van de discriminator, proberen dit patroon te reproduceren in de inkleuringen. Door het gebruik van strided convolutions in de discriminator, vloeit de gradiënt niet gelijkmatig terug naar elke pixel [68]. Wanneer we ofwel de afbeeldingen met halftoning weglaten uit de train test, ofwel de discriminator loss uitzetten, verdwijnen deze patronen volledig. Een betere oplossing zou zijn om de pix2pix discriminator aan te passen door de stride van de convoluties te wijzingen. Aangezien deze artefacten niet aanwezig zijn in onze uiteindelijke methode en we voor elke variant enkel de loss-functie veranderen, doen we dit hier echter niet.

Het laatste en belangrijkste resultaat is tenslotte de performantie van onze volledige methode. De geloofwaardigheid is daar 66,1 % op geziene albums en 64,3 % op ongeziene albums. Dit is een verbetering van respectievelijk 10,3 % en 14,1 % ten opzichte van de variant met enkel L_1-loss. Verder zien we ook een grote verbetering tegenover de variant L_1 + discriminator. Dit is te wijten aan het feit dat de total variation term ervoor zorgt dat artefacten en andere ongewenste overgangen (zoals hierboven besproken) verdwijnen. Tenslotte merken we op dat ook hier een



Figuur 5.6: Aanwezigheid van dithering en JPEG artefacten patronen in de ground truth afbeeldingen

verschil is in performantie tussen de geziene en ongeziene albums. Het verschil is hier echter een stuk kleiner dan bij de andere netwerkvarianten. We concluderen dat onze volledige methode in staat is nieuwe albums in te kleuren waarbij deze inkleuringen in 64,3 % van de gevallen volledig geloofwaardig overkomen voor proefpersonen.



(a) Geloofwaardigheid ongezien album

(b) Geloofwaardigheid gezien album

Figuur 5.7: Percentage van de prenten dat als echt werd beoordeeld per variant en verzameling

5.4 Foutanalyse

De loss-functie van onze volledige methode bestaat uit drie termen die elk een belangrijke bijdrage leveren aan de geloofwaardigheid van de resulterende inkleuringen. In de vorige sectie bleek dat onze inkleuringen hierdoor in een groot deel van de gevallen niet te onderscheiden zijn van

Variant	Verzameling: Gezien album	Verzameling: Ongezien album
L_1	55,8 %	50,2 %
L_1 + discriminator	47,1 %	44,0 %
L_1 + discriminator + total variation	**66,1 %**	**64,3 %**
Ground truth	90,9 %	94,2 %

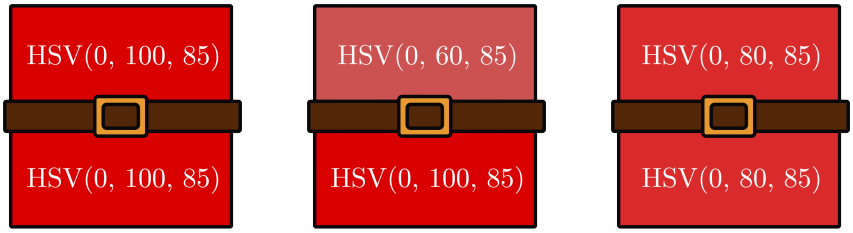
Tabel 5.3: Percentage van de prenten dat als echt werd beoordeeld per variant en verzameling.

echte inkleuringen. Op basis van deze experimenten hebben we ook data over welke inkleuringen het meest en minst geloofwaardig werden bevonden. Het is interessant om even te kijken naar de minder geloofwaardige afbeeldingen. Zo kunnen we analyseren welke fouten precies afbreuk doen aan het realisme en waarom ons netwerk deze fouten maakt.

Bij het bekijken van de mislukte inkleuringen, zien we dat de meest veelvoorkomende fout is dat verschillende connected components die semantisch tot éénzelfde vlak behoren toch een verschillende kleur krijgen. Dit fenomeen wordt geïllustreerd in de inkleuring in figuur 5.9c. Merk op hoe de blauwe achtergrond daar uit zes connected components bestaat, die echter niet allemaal blauw gekleurd worden. Zo hebben de twee connected components onderaan rechts en onderaan midden op de figuur een bruine inkleuring. Dergelijke afwijkingen worden snel opgemerkt door mensen en doen zo afbreuk aan het realisme.

Het feit dat het netwerk dergelijke fouten maakt, valt te verklaren door de gebruikte lossformulering verder te bestuderen. De hierboven geziene situatie is namelijk analoog aan die in figuur 5.8. Merk op dat de inkleuring A en B beide exact dezelfde L_1-loss en gradient loss hebben ten opzichte van de gewenste inkleuring. Concreet is de L_1-loss identiek aangezien de pixels in de twee vlakken in beide gevallen dezelfde absolute afwijking hebben ten opzichte van de gewenste inkleuring. Verder zijn er binnen één vlak geen kleurovergangen, waardoor de gradiënt loss telkens 0 is. Voor een menselijke observator is inkleuring A echter realistischer aangezien de twee vlakken tot hetzelfde object behoren en dus dezelfde kleur horen te krijgen (ook al is deze kleur niet perfect).

Dergelijke fouten, die duidelijk zijn voor een menselijke waarnemers, maar die moeilijker zijn om wiskundig te quantificeren, zijn één van de redenen om GANs te gebruiken. We zouden dus kunnen hopen dat de discriminator in staat is om dergelijke inconsistenties af te straffen. In de praktijk blijkt dit echter een delicate evenwichtsoefening. Van zodra we de discriminator complex genoeg maken om te herkennen dat beide vlakken tot hetzelfde object behoren, en dus dezelfde kleur dienen te krijgen, gaat deze het spel volledig overheersen. Hierdoor vloeit er geen bruikbaar signaal meer naar de generator waardoor deze niet meer verbetert [91].



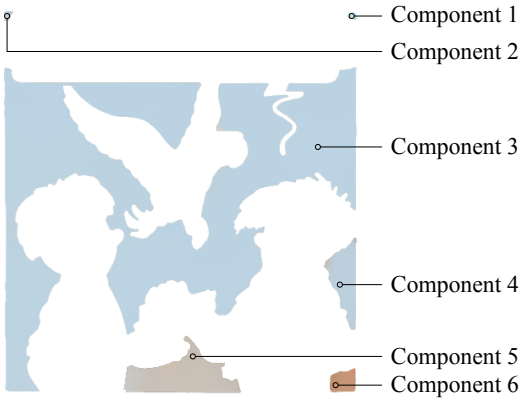
(a) Inkleuring A (b) Inkleuring B (c) Ware inkleuring

Figuur 5.8

(a) Binaire invoerafbeelding

(b) Ground truth

(c) Inkleuring via L_1 + discriminator + total variation



(d) De connected components die de achtergrond vormen

Figuur 5.9: Soms bestaat één semantisch object (de lucht in dit geval) uit meerdere connected components. Die components krijgen soms elk een verschillende kleur toegewezen.

5.5 Resultaten op andere datasets

Tot nu toe hebben we onze methode telkens getraind en getest op prenten uit gekleurde Jommeke albums die we omzetten naar binair zwart-wit. Het is echter interessant om te zien hoe onze methode presteert op andere types afbeeldingen. Zo kunnen we inschatten in hoeverre het netwerk in staat is te generaliseren naar andere datasets. De belangrijkste dergelijke toepassing is het inkleuren van albums die oorspronkelijk in zwart-wit verschenen. Die toepassing vormt het onderwerp van het volgende hoofdstuk. Voordat we dit doen, proberen we onze methode uit op enkele afbeeldingen die een stuk verder van de train set liggen. Zo zetten we onze methode eerst in om schetsen te kleuren en daarna om prenten uit twee andere stripreeksen te kleuren.

5.5.1 Schetsen van fans

Jommeke is met meer dan 50 miljoen verkochte albums zeer populair. Er is dan ook geen gebrek aan tekeningen gemaakt door fans, zogenaamde *fanart*. Typisch hanteren de artiesten achter dergelijke fanart een andere tekenstijl dan de oorspronkelijke striptekenaar. Het is dan ook interessant om onze methode uit te proberen hierop om te kijken als ze robuust is tegenover een gewijzigde tekenstijl.

Concreet kleuren we twee portretschetsen in van personages in Jommeke (zie figuur 5.10a en 5.10d). Deze schetsen correct inkleuren is niet vanzelfsprekend aangezien de karakteristieken ervan sterk verschillen van die van strippprenten. Zo komen dergelijke portretten, die enkel een hoofd afbeelden, zonder bijhorend lichaam, niet voor in de trainingsdata en zijn de personages getekend met scherpere gezichtskenmerken dan gewoonlijk. Verder gaat door het binariseren van de schetsen veel nuance van de artiest verloren.

Na wat experimenten bleek het cruciaal te zijn om de schetsen te omringen door een zwarte rechthoek (zoals in een echte strippprent) om goede resultaten te behalen. Hierna slaagt onze methode er relatief goed in deze schetsen in te kleuren (zie figuur 5.10c en 5.10f). Zo worden zowel de huid als snor herkend en correct ingekleurd. We besluiten dat onze methode een zekere robuustheid heeft tegenover een gewijzigde tekenstijl en ontbrekende informatie.

5.5.2 Andere stripreeksen

Een tweede interessant experiment is om ons getraind netwerk uit te proberen op andere stripreeksen. Concreet kozen we twee prenten uit de stripreeks *F.C. De Kampioenen* en twee prenten uit *Kiekeboe*. Deze prenten zetten we om naar binair zwart-wit en laten we vervolgens inkleuren door onze volledige methode (L_1 + discriminator + total variation) zonder het netwerk opnieuw te trainen. Het resultaat hiervan is voorgesteld in figuur 5.11 en figuur 5.12.

Op deze figuren zien we dat het netwerk getraind op Jommeke-albums deels veralgemeent naar andere stripreeksen. Zo worden tekstballonnen, bomen en lucht telkens bijna perfect ingekleurd.

(a) Oorspronkelijke schets (b) Binaire invoerafbeelding (c) Inkleuring

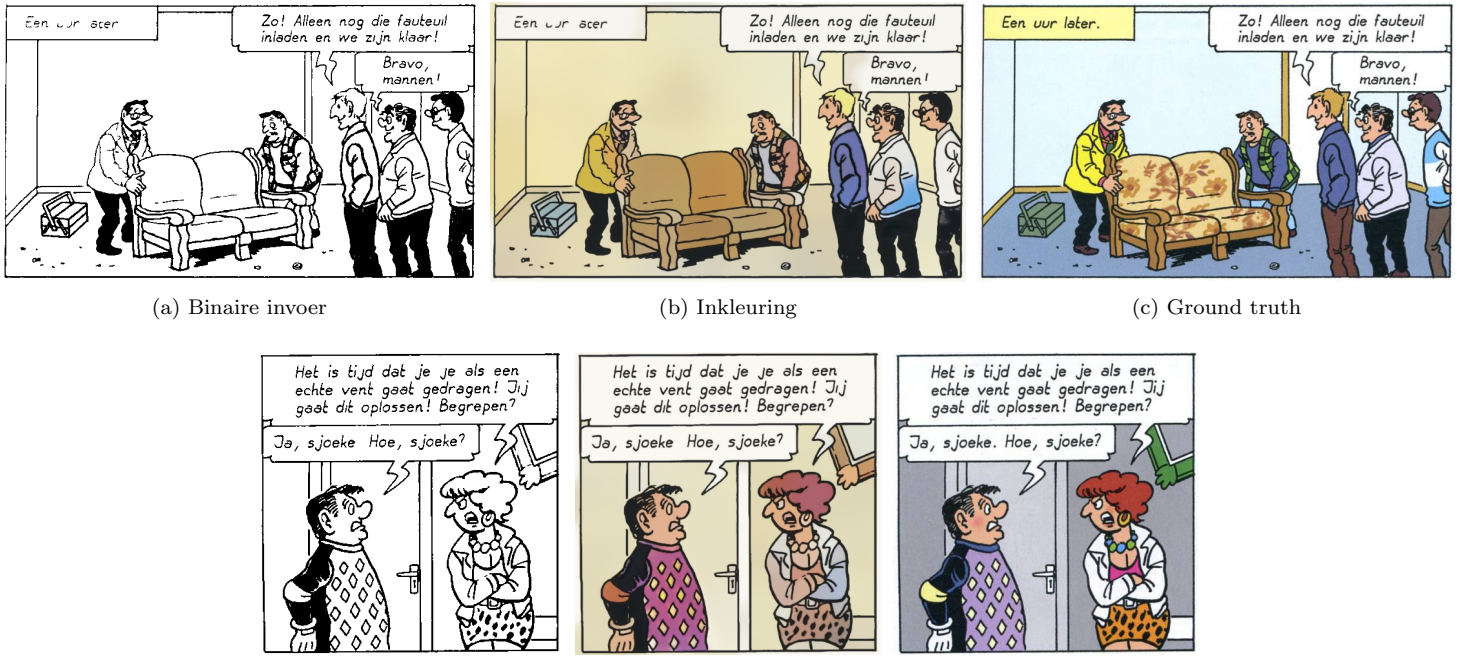
(d) Oorspronkelijke schets (e) Binaire invoerafbeelding (f) Inkleuring

Figuur 5.10: Inkleuring van twee schetsen van personages door onze volledige methode

Dit is niet zo verwonderlijk aangezien deze sterke gelijkenissen vertonen met de overeenkomstige voorwerpen in Jommeke-albums. Verder lijkt het netwerk ook in staat huid en haar relatief goed in te kleuren. Dit is minder vanzelfsprekend aangezien de vormen van personages heel wat anders getekend worden dan in Jommeke-albums. We vermoeden dat het netwerk al heel wat nevenpersonages heeft gezien in de train set en zo in staat is een grote diversiteit aan gezichten en handen te herkennen. De grootste moeilijkheid lijkt het netwerk te hebben met het inkleuren van kledij. Ook dit houdt steek aangezien het netwerk deze personages nooit eerder heeft gezien en dus ook geen idee heeft welke kleur hun kledij dient te krijgen. Hierdoor gokt het netwerk voornamelijk op witte, blauwe of rode kledij. Deze inkleuring is echter niet geloofwaardig voor mensen die vertrouwd zijn met de normale kledij van de personages.

De belangrijkste kanttekening bij bovenstaande resultaten is uiteraard dat we het netwerk niet opnieuw trainen. Mochten we het netwerk trainen op *F.C. De Kampioenen* of *Kiekeboe*, zouden de resultaten op deze reeksen ongetwijfeld nog zeer sterk verbeteren. De reden dat we dit hier niet doen is dat het verzamelen van een geschikte dataset en het trainen van het netwerk beide heel wat tijd in beslag nemen. Aan onze methode zelf dient er echter niets veranderd te worden.

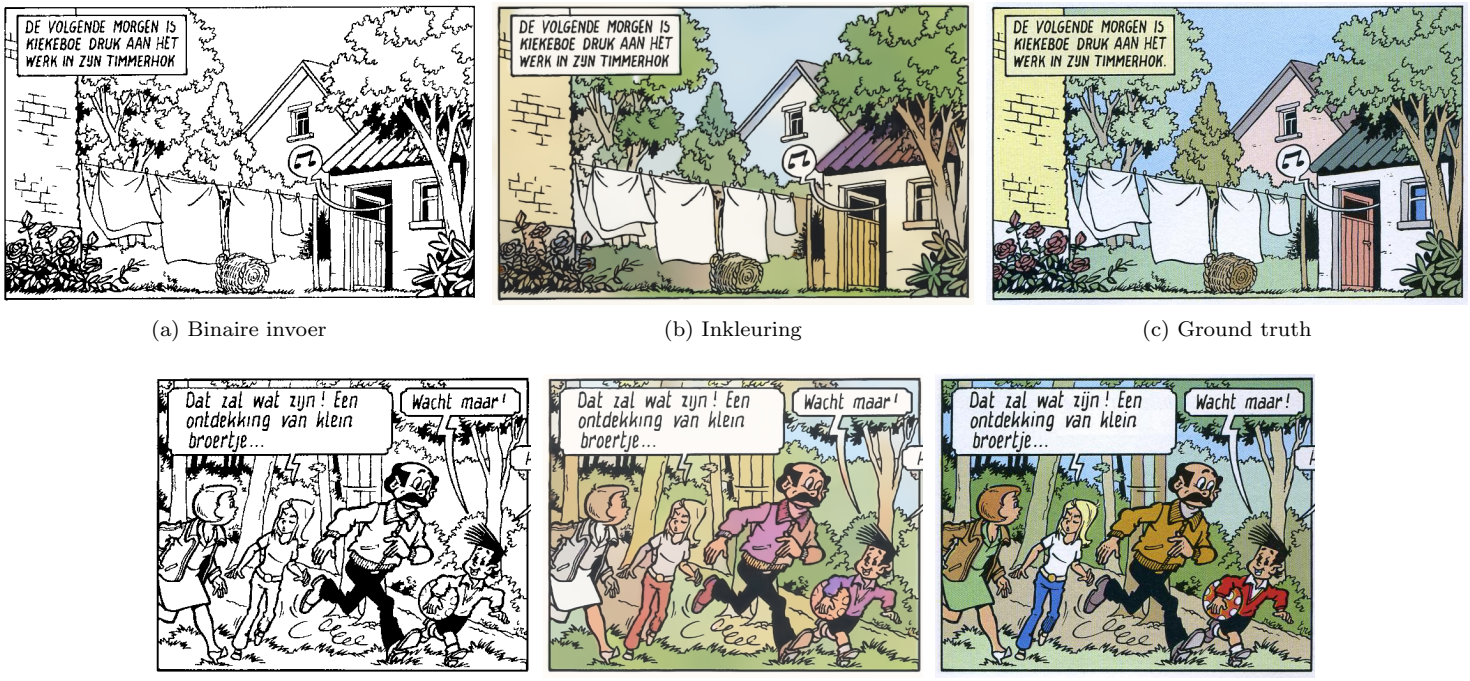
Een andere mogelijkheid is om het netwerk te trainen op een verzameling die bestaat uit meerdere stripreeksen. Aangezien het netwerk dan een diversere train set heeft, verwachten we dat de performantie op albums uit ongeziene stripreeksen drastisch toeneemt. De keerzijde is echter dat de performantie op één van stripreeksen waarmee we trainen waarschijnlijk minder goed is dan wanneer we enkel trainen op een (voldoende grote) dataset van die ene reeks.



(a) Binaire invoer (b) Inkleuring (c) Ground truth

(d) Binaire invoer (e) Inkleuring (f) Ground truth

Figuur 5.11: Voorbeeld van het toepassen van onze volledige methode op *F.C. De Kampioenen*. Bovenstaande prenten werden gekozen omwille van hun representativiteit voor de performantie op ongeziene stripreeksen.



(a) Binaire invoer (b) Inkleuring (c) Ground truth

(d) Binaire invoer (e) Inkleuring (f) Ground truth

Figuur 5.12: Voorbeeld van het toepassen van onze volledige methode op *Kiekeboe*. Bovenstaande prenten werden gekozen omwille van hun representativiteit voor de performantie op ongeziene stripreeksen.

Hoofdstuk 6

Toepassing op oude zwart-witalbums

In het vorige hoofdstuk trainden en testten we onze methode op kleurenprenten die we vooraf omzetten naar zwart-wit. Op die manier konden we de getrainde varianten vergelijken met elkaar en de ground truth. Nu zetten we onze methode in voor het inkleuren van oudere albums die enkel in zwart-wit verschenen en waarvoor dus geen ground truth bestaat. Zo blazen we deze albums nieuw leven in en leren we hoe onze methode omgaat met een oudere tekenstijl. We beginnen met de beschikbare dataset toe te lichten. Daarna gaan we dieper in op de verschillen tussen deze oude zwart-witalbums en de nieuwe kleuralbums waarop we trainden en hoe we hiermee omgaan. Tenslotte bespreken we de resultaten van de perceptuele experimenten met deze oudere albums.

6.1 Dataset

De stripreeks Jommeke heeft in de loop der tijd heel wat wijzigingen ondergaan in tekenstijl, omslagontwerp en drukwijze. De belangrijkste gebeurtenis voor ons doel is het verschijnen van het eerste album in kleur in 1979. De 91 albums die uitkwamen voor het 92^{e} album werden telkens in binair zwart-wit gedrukt. De enige vorm van kleurschakeringen in die albums is het gebruik van één halftoningpatroon om aan te duiden dat sommige gebieden een andere tint hebben [92].

In totaal hebben we scans van een zestigtal dergelijke zwart-witalbums tot onze beschikking (zie sectie 3.2). Elke pagina is ingescand met een resolutie van 200 DPI, een factor waar wij jammergenoeg geen controle over hebben. Deze lage resolutie zal namelijk een uitdaging vormen om het halftoning patroon volledig te verwijderen. Twee typische scans uit deze dataset zijn weergegeven in figuur 6.1. Zoals te zien is op deze figuur zijn de pagina's typisch vergeeld en bevatten ze vaak spikkeltjes stof of andere imperfecties.

(a) Scan uit album 10 - De straalvogel

(b) Scan uit album 18 - Met Fifi op reis

Figuur 6.1: Twee voorbeeldscans uit zwart-wit albums

6.2 Verschillen met kleurenstrips

Een belangrijke aanname bij supervised learning is dat de training en test set beide gesampled zijn uit dezelfde distributie. Er is dus één *data-genererende distributie* die verantwoordelijk is voor de generatie van elk train en test sample [93]. Uiteraard gebruiken we niet dezelfde samples in de train en test set, maar de onderliggende voortbrengende distributie is wel dezelfde.

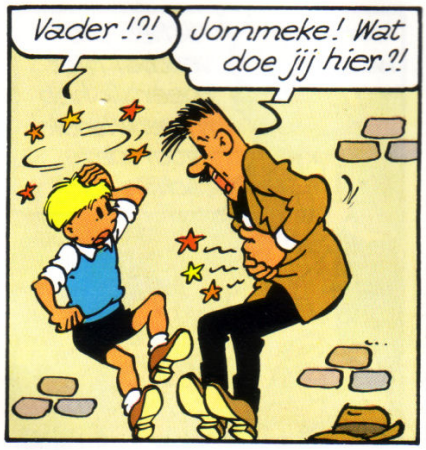
In het vorige hoofdstuk was deze aanname telkens voldaan aangezien we bij het trainen en testen telkens de automatisch gegenereerde lijntekeningen als invoer gebruikten. Nu gebruiken we lijntekeningen uit zwart-witalbums als test set. Het netwerk wordt echter nog steeds getraind met lijntekeningen uit gekleurde albums aangezien we enkel daar de ground truth kennen. Wanneer we deze twee types lijntekeningen vergelijken, blijken er vaak opmerkelijke verschillen te zijn waardoor de bovenstaande aanname niet langer vervuld is. In wat volgt gaan we dieper in op twee dergelijke verschilpunten, waarom deze precies een probleem vormen en hoe we dit kunnen verhelpen.

6.2.1 Tekenstijl

Een eerste belangrijke factor is de tekenstijl, die doorheen de jaren sterk geëvolueerd is. Zoals in sectie 6.1 reeds aangehaald, zijn de zwart-witalbums die we willen inkleuren, telkens oudere albums. De albums waarop het netwerk getraind is, zijn echter kleurenalbums uitgegeven vanaf 1979, waardoor deze een moderne tekenstijl hebben. Dit verschil in tekenstijl wordt geïllustreerd in figuur 6.2. Linksboven op afbeelding 6.2a, is een scan te zien van de oorspronkelijke druk van album *1 - De jacht op een voetbal* uit 1959. De corresponderende gegenereerde lijntekening is te zien op afbeelding 6.2c. In 1987 werd een volledig hertekende en ingekleurde versie van dit album uitgegeven. Het overeenkomstige prentje uit deze nieuwe uitgave is te zien in figuur 6.2b en de gegenereerde lijntekening op afbeelding 6.2d. Het is duidelijk dat er heel wat verschil is in tekenstijl tussen het originele album en de recentere heruitgave. Zo zijn bijvoorbeeld de opgerolde mouwen en de specifieke vorm van het kapsel duidelijke kenmerken om Jommeke te herkennen in moderne albums. Deze kenmerken ontbreken volledig of zijn veel minder prominent aanwezig in oudere albums. Spijtig genoeg is dit het enige album waarvan we zowel een versie in oude tekenstijl als in nieuwe tekenstijl hebben. Verder is vaak niet enkel de tekenstijl maar ook de inhoud van het prentje significant veranderd, waardoor het niet evident is een mapping van oude naar nieuwe tekenstijl te leren.

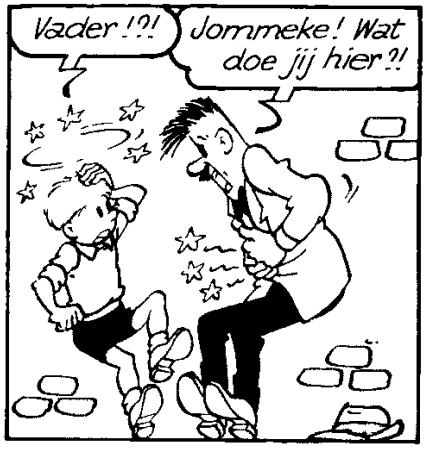
Dit verschil in tekenstijl vormt een probleem voor de performantie, aangezien het netwerk afhankelijk is van dergelijke details om karakters en objecten te herkennen. Deze afhankelijkheid wordt geïllustreerd in figuur 6.3. Hier zijn we de uitvoer van het netwerk op een afbeelding in oude tekenstijl en een hertekende afbeelding in nieuwe tekenstijl. Bij de oudere tekenstijl (links) heeft het netwerk moeite met het herkennen van bijvoorbeeld de kleding van Jommeke, waardoor deze niet de correcte blauwe inkleuring krijgt. Bij de nieuwere tekenstijl (rechts) vormt dit geen probleem, aangezien het netwerk op deze stijl getraind is, waardoor de resulterende inkleuring

(a) Originele zwart-witafbeelding uit het eerste album



(b) Originele kleurenafbeelding uit een hertekende nieuwe uitgave van het eerste album

(c) Binaire versie van afbeelding a



(d) Binaire versie van afbeelding b

Figuur 6.2: Verschil in tekenstijl tussen oudere en nieuwe albums

een stuk realistischer oogt.

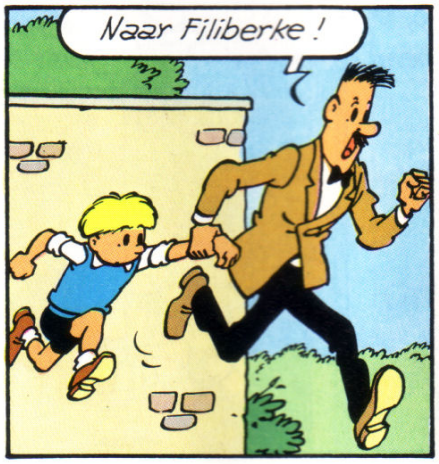
De performantie verbeteren voor deze allereerste albums blijkt een uitdagend probleem te zijn. Hun lijntekeningen verschillend immers beduidend van de trainingsdata en we hebben geen ground truth over de kleur van personages en objecten. Een mogelijke oplossing zou zijn om *transfer learning* te gebruiken [94, 95]. Concreet kleuren we daarbij handmatig enkele oudere albums in, zodat we deze kunnen benutten als trainingsdata. Vervolgens kunnen we het reeds getrainde netwerk als startpunt gebruiken en dit fine-tunen op deze oudere albums. Het voordeel van deze aanpak is dat een groot deel van de kennis van het reeds getrainde netwerk relevant blijft, waardoor we slechts een beperkte hoeveelheid oudere albums manueel dienen te kleuren als trainingsdata.

De dataset en behaalde resultaten verder inspecteren leert ons dat het hierboven aangekaarte probleem hoofdzakelijk beperkt blijft tot deze allereerste albums. Bij het 15^{e} album (uit de 91 zwart-witalbums) is de tekenstijl reeds geconvergeerd naar een punt waarop het netwerk bijna altijd in staat is om de hoofdpersonages correct in te kleuren. Op dat punt zien we dan ook dat veel van de typische kenmerken uit moderne albums (opgerolde mouwen van Jommeke, vollere gezichten) reeds aanwezig zijn. Er is nog steeds wat verschil in tekenstijl, maar het netwerk is robuust genoeg om hiermee om te gaan zolang deze typische kenmerken min of meer aanwezig zijn. Het feit dat het netwerk met deze subtielere verschillen kan omgaan is te wijten aan de stochasticiteit bij het omzetten van de trainingsdata naar lijntekeningen (zie sectie 3.5). Het variëren van de drempelwaarde zorgt er namelijk voor dat de precieze dikte van lijnen niet uitmaakt en dat het netwerk overweg kan met het wegvallen van bepaalde lijnen. Gelet op de beperkte impact van het probleem, besluiten we dat een beperkte verbetering op een klein deel van de albums de extra complexiteit van het introduceren van transfer learning niet rechtvaardigt. Een andere factor is het feit dat het manueel inkleuren van zwart-witalbums als extra trainingsdata niet strookt met het volledig geautomatiseerd zijn van onze methode. We gaan dan ook niet dieper in op dergelijke technieken en aanvaarden dat de inkleuring van de allereerste albums typisch minder realistisch is dan deze van latere albums.

6.2.2 Halftoning

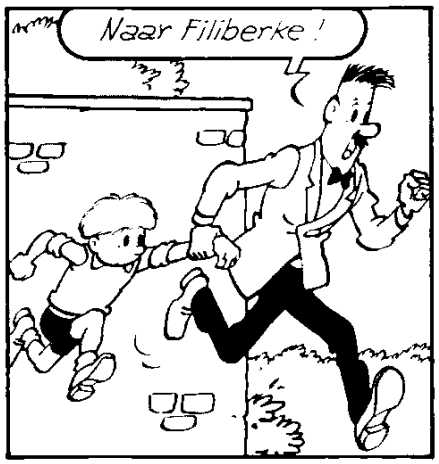
Een tweede belangrijk verschil tussen de zwart-witalbums en de automatische gegenereerde lijntekeningen is de aanwezigheid van zogenaamde *halftoning dots* [92, 96]. Een voorbeeld van een dergelijk patroon wordt weergegeven in figuur 6.4a. Belangrijk hierbij op te merken is dat er telkens slechts één specifiek halftoning patroon wordt gebruikt en dat het al dan niet aanwezig zijn ervan een puur artistieke keuze is. Zo is het gebruik van halftoning niet consistent en zijn de gebieden met halftoning dots puur gekozen om zo bepaalde objecten meer of net minder te benadrukken. Aangezien er slechts één type patroon puur artistiek wordt gebruikt, zonder variatie in grootte of spatiëring van de dots, valt er uit het patroon geen informatie af te leiden over de werkelijke kleur die een gebied dient te krijgen.

(a) Scan van tekening in oudere tekenstijl



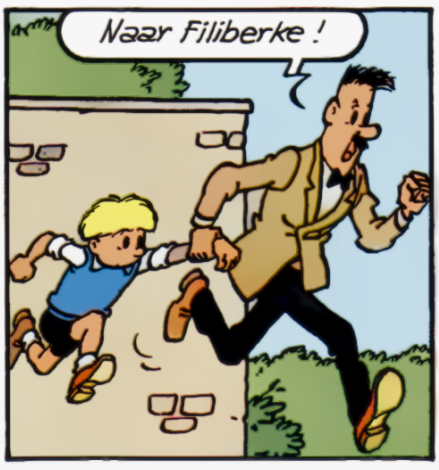
(b) Scan van tekening in nieuwere tekenstijl

(c) Binaire versie van figuur 6.3a



(d) Binarie versie van figuur 6.3b

(e) Gegenereerde inkleuring van figuur 6.3c



(f) Gegenereerde inkleuring van figuur 6.3d

Figuur 6.3: Invloed van tekenstijl op de inkleuring

Belangrijk op te merken is dat deze halftoning dots dezelfde kleur hebben als de lijnen die objecten afbakenen. Dit is namelijk net de premisse van halftoning: grijstinten simuleren door zwarte puntjes te plaatsen. Dit betekent ook dat, indien de resolutie van de scan hoog genoeg is, de halftoning puntjes nog aanwezig zijn na de conversie van deze scan naar een binaire lijntekening. Voor de scan uit figuur 6.4a, wordt het resultaat van deze conversie afgebeeld in figuur 6.4b. Zoals te zien is op deze laatste figuur, wordt elke halftoning puntje als lijn gemarkeerd in de binaire versie.

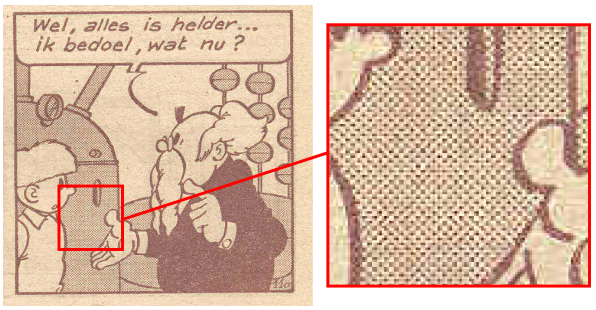
Invloed op inkleuring

We vermoeden dat de aanwezigheid van halftoning dots in de invoer een negatieve invloed zal hebben op de geloofwaardigheid van de resulterende inkleuring. Tijdens het trainen verwachten we namelijk dat elke zwarte pixel in de automatisch gegenereerde lijntekening, die als invoer van het netwerk dient, een zwarte lijn voorstelt in de corresponderende kleurenafbeelding. In dat geval leert het netwerk om elke zwarte pixel in de invoer eveneens zwart te kleuren in de uitvoer en om de witte pixels een kleur toe te kennen op basis van de omringende vormen. De fout die we op basis van deze analyse verwachten is dat het netwerk elk halftoning puntje een volledig zwarte kleur geeft in de uitvoer.

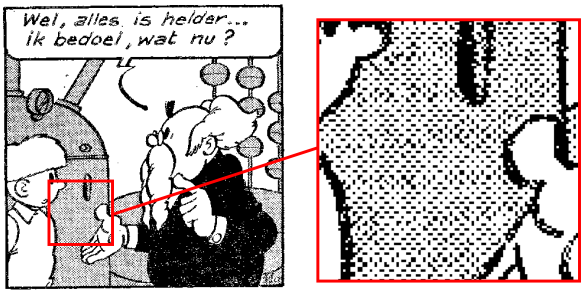
Wanneer we echter de uitvoer van het netwerk inspecteren, merken we dat deze niet overeenstemt met bovenstaande voorspelling. We zien namelijk dat het volledige vlak waarin de halftoning puntjes zich bevinden een stuk donkerder dan gewenst gekleurd wordt. De halftoning puntjes zelf worden dus niet overgenomen als volledig zwarte pixels in de uitvoer. Dit fenomeen wordt geïllustreerd bij de inkleuring van de wagen en de bomenrij in figuur 6.5b.

Een verklaring voor dit op het eerste gezicht merkwaardige fenomeen vinden we door de trainingsdata grondiger te bestuderen. Zoals in sectie 3.5 besproken, wordt er voor elke kleurenafbeelding automatisch een corresponderende lijntekening gegenereerd. Het netwerk wordt vervolgens getraind om de originele kleurenafbeelding te reconstrueren op basis van deze binaire lijntekening. De voorafgaande conversie naar een binaire lijntekening is echter niet perfect en introduceert soms ruis in de binaire versie. Dit probleem wordt geïllustreerd in figuur 6.6. De kleur van de bomen rechts op de originele kleurenafbeelding 6.6a varieert lichtjes omwille van een imperfecte afdruk of scan. Aangezien de bomen een erg donkere kleur hebben, leiden deze variaties ertoe dat sommige pixels net onder de drempelwaarde vallen die gebruikt wordt om zwarte pixels te herkennen. Deze ongewenste zwarte pixels in de binaire versie zijn zichtbaar op de automatische gegeneerde binaire afbeelding 6.6b. In het algemeen zal dergelijke ruis telkens geïntroduceerd worden in gebieden waarvan de kleurwaarde dicht aanleunt bij de gebruikte drempelwaarde (d.w.z. donkere gebieden).

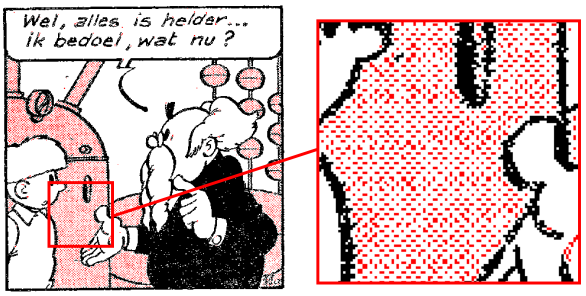
Het netwerk leert dan ook dat dergelijke ruis erop wijst dat het omringende gebied een donkere kleur dient te krijgen. Wanneer we vervolgens het netwerk een binaire afbeelding met halfto-



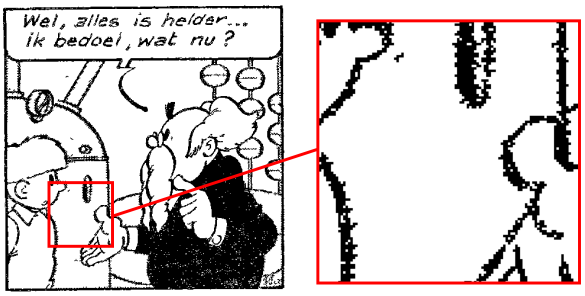
(a) Halftoning in een scan van een origineel zwart-witalbum



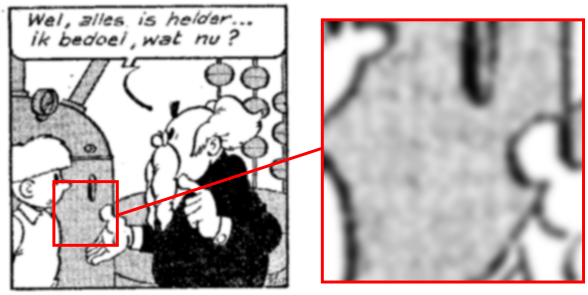
(b) Halftoning na de conversie naar een binaire afbeelding



(c) De kleine connected components zijn hier aangeduid. Merk op dat dit halftoning dots zijn maar ook een stukje van de hand.



(d) Resultaat na het verwijderen van kleine connected components



(e) Resultaat na het toepassen van een low-pass filter op de binaire afbeelding

Figuur 6.4: Voorbeeld van de aanwezigheid en het verwijderen van halftoning

(a) Binaire versie voor het verwijderen van halftoning

(b) Inkleuring op basis van figuur a

(c) Binaire versie na het verwijderen van halftoning

(d) Inkleuring op basis van figuur c

Figuur 6.5: Impact van halftoning in de invoer op het realisme van de uitvoer

ning laten inkleuren, interpreteert het netwerk het 'ruispatroon' van deze halftoning puntjes op dezelfde manier. Hierdoor wordt het gebied dat halftoning puntjes bevat erg donker gekleurd zonder daarbij noodzakelijk de halftoning puntjes zelf over te nemen, zoals te zien is op de bomen in inkleuring 6.5b.

Verwijderen van halftoning

Het feit dat halftoning ervoor zorgt dat het omringende gebied donkerder ingekleurd wordt, is niet gewenst aangezien dit leidt tot ongeloofwaardige inkleuringen. Halftoning is hier immers puur een artistieke keuze en duidt er niet noodzakelijk op dat vlakken een donkerdere kleur dienen te krijgen. We kunnen dus nauwelijks bruikbare informatie halen uit deze halftoning patronen wat betreft de inkleuring. Daarom kiezen we ervoor om deze patronen te verwijderen als we zwart-witalbums wensen in te kleuren tijdens de testfase. Op die manier heeft het al dan niet aanwezig zijn van halftoning geen invloed op de inkleuring.

Het reconstrueren van een afbeelding met continue kleurschakeringen op basis van een afbeelding met halftoning, wordt *inverse halftoning* of *descreening* genoemd [97, 98]. Een simpele en veelgebruikte techniek is het toepassen van een low-pass filter, zoals een Gaussiaans filter, op de afbeelding [99]. Hierdoor wordt hoog-frequente informatie verworpen en verdwijnt het halftoning patroon. Dit is te vergelijken met de manier waarop onze ogen een halftoning patroon van op voldoende afstand beschouwen als een egaal kleurvlak. Het nadeel van deze aanpak is de nood om zorgvuldig een bandbreedte te kiezen voor het filter. Bij een te lage bandbreedte worden namelijk ook andere randen onscherp, terwijl een te hoge bandbreedte het patroon niet volledig verwijdert. In de praktijk bleek deze afweging het erg moeilijk te maken om deze techniek hier succesvol in te zetten. Het netwerk vertrouwt namelijk volledig op lijnen voor de inkleuring en elke afname in scherpte van deze lijnen leidt onmiddellijk tot een sterk gereduceerde performan-

(a) Originele kleurenafbeelding die donkere gebieden bevat (de twee bomen in dit geval)

(b) Resultaat van de conversie naar een binaire afbeelding

Figuur 6.6: Introductie van ruis bij de automatische omzetting van een kleurenafbeelding naar een lijntekening

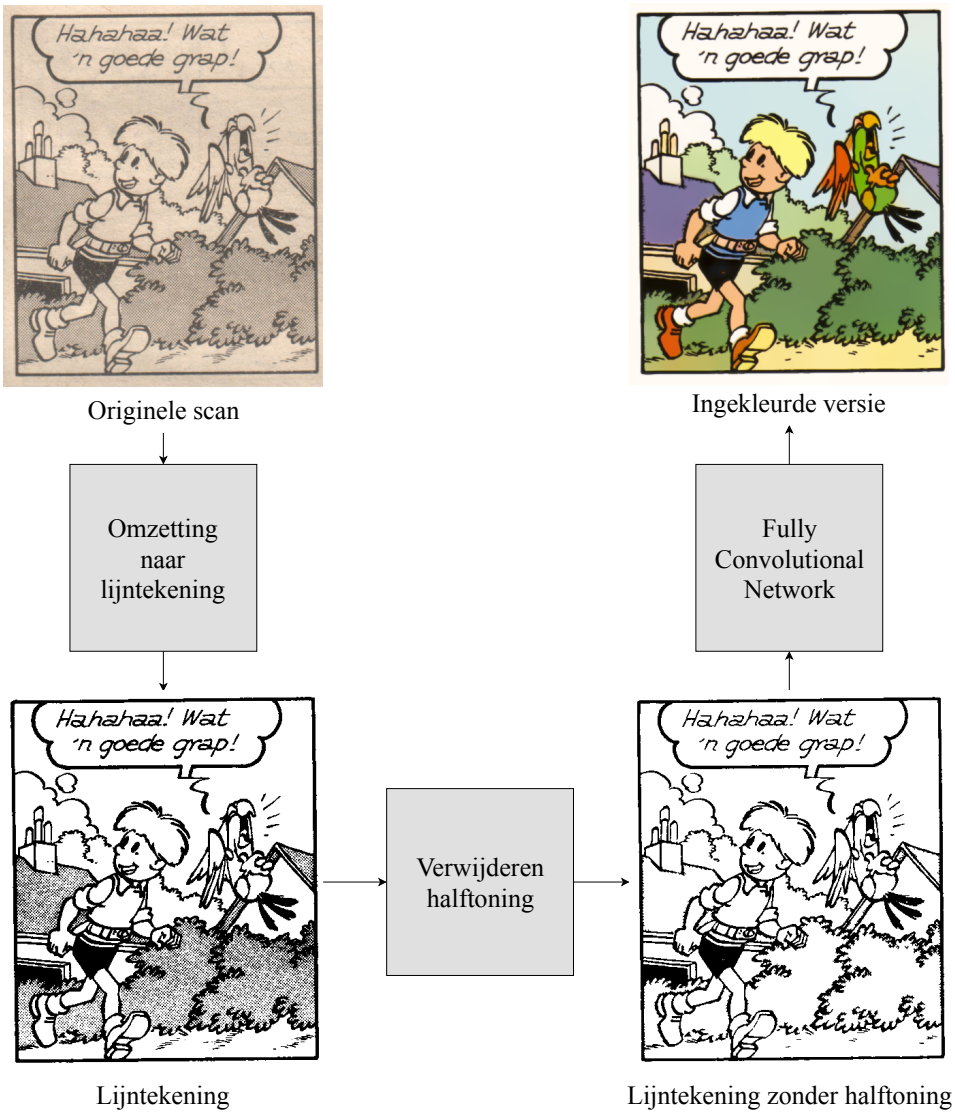
tie. Een voorbeeld van het verlies in scherpte in deze lijnen, vóór het opnieuw binariseren van het resultaat, wordt getoond in figuur 6.4e.

In ons geval kunnen we echter dankbaar gebruik maken van het feit dat we de resulterende continue kleuren niet nodig hebben. Aangezien de invoer van het netwerk binair zwart-wit dient te zijn, zou de volgende stap toch het verwijderen van deze grijze vlakken zijn. In plaats van de halftoning patronen om te zetten naar continue kleuren met behulp van een low-pass filter, focussen we ons louter op het verwijderen van de patronen. Om dit te doen, maken we gebruik van *connected-component labeling* [100]. Hierbij wordt elke zwarte pixel toegewezen aan één connected component. Twee zwarte pixels behoren tot dezelfde connected components als er een pad van zwarte pixels bestaat van de ene pixel naar de andere pixel. In ons geval mogen deze paden enkel uit rechte verbindingstukjes bestaan en geen diagonale.

Na het onderverdelen van de zwarte pixels in connected components, bepalen we welke components uit ten hoogste 6 pixels bestaan. Deze zijn in het rood aangeduid in figuur 6.4c. We zien dat deze kleine connected components voornamelijk halftoning puntjes zijn. Jammergenoeg sluipen er soms ook nuttige lijnstukjes, zoals een stukje van de hand op de figuur, in. De volgende stap is het verwijderen van deze kleine connected components. Dit resulteert in figuur 6.5c, waarin bijna alle halftoning puntjes verdwenen zijn. Desondanks blijven er aan de randen nog enkele artefacten over waar de halftoning puntjes een lijn kruisten. Ook deze artefacten verwijderen, zonder daarbij gewenste details in lijnstukken te verwijderen, bleek erg uitdagend. Aangezien de invloed van deze imperfecties op de uiteindelijke inkleuring relatief klein is, laten we ze dan ook staan. We hebben nu een binaire lijntekening, zonder halftoning puntjes, die we als netwerkinvoer kunnen gebruiken.

Op dit moment hebben we een methode om scans uit oude zwart-witalbums volledig automatisch in te kleuren. De procedure om een prent in te kleuren is weergegeven in figuur 6.7. Dit is dezelfde procedure als diegene uit het vorige hoofdstuk, op het toevoegen van de stap *verwijderen halftoning* na. We kunnen onze methode nu toepassen en evalueren op deze zwart-witalbums.

Belangrijk is dat er in dit geval geen ground truth beschikbaar is. Om de performantie van onze methode te meten, maken we daarom louter gebruik van menselijke evaluatie. Concreet is de opzet van het perceptueel experiment hetzelfde als in het vorige hoofdstuk (zie sectie 5.3.1). Van de 90 prenten die de deelnemers dienden te beoordelen als echt of vals, waren er 30 prenten afkomstig uit de dataset *zwart-witalbum* (zie tabel 5.2). Aangezien we in het vorige hoofdstuk reeds observeerden dat onze volledige methode (L_1 + discriminator + total variation) het best presteert, testen we enkel deze variant. Enkele voorbeeldinkleuringen hiervan worden getoond in sectie A.5 en A.6.



Figuur 6.7: Schematische weergave van het inkleuren van een scan uit een zwart-witalbum

6.3 Perceptuele experimenten

Opnieuw beoordeelden 84 deelnemers elk 30 prenten, waardoor we 2520 beoordelingen hebben. Het aantal prenten dat als echt bestempeld werd is 45,9 %. We zijn dus in ongeveer de helft van de gevallen in staat een prent in te kleuren waarbij het resultaat nauwelijks te onderscheiden is van een echte inkleuring. Enkele inkleuringen die vaak als geloofwaardig werden beoordeeld, worden getoond in figuur 6.8.

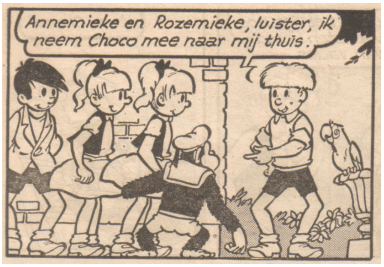
Hoewel we op sommige prenten erg overtuigende resultaten behalen, is de hier behaalde 45,9 % beduidend minder dan de 64,3 % op de dataset *ongezien album*. Het is daarom interessant even dieper in te gaan op prenten waarop onze methode minder goed presteert. Enkele prenten die het vaakst als vals werden gemarkeerd, zijn getoond in figuur 6.8. In deze prenten wordt het duidelijk dat het voornaamste probleem het onvoldoende verwijderen van halftoning is. Dit leidt ertoe dat het netwerk het vlak met de nog aanwezige halftoning een erg donkere en ongeloofwaardige kleur toekent. Dit probleem treedt vooral op bij prenten waar het contrast en de kwaliteit van de originele scan laag is, zoals in figuur 6.8d. In dat geval lopen de halftoning puntjes deels in elkaar over waardoor ze niet allemaal verwijderd worden.

Wanneer we in die prenten manueel de overblijvende halftoning puntjes weghalen, zien we dat de resulterende inkleuring heel wat realistischer wordt. Dit verschil wordt gedemonstreerd in figuur 6.10. Het volledig automatisch bepalen welke pixels al dan niet halftoning voorstellen blijft echter een uitdaging. Elke automatische methode die we probeerden had moeite met een deel van de afbeeldingen. Het manueel bepalen van een geschikte drempelwaarde is dan ook de meest betrouwbare methode. Aangezien dit echter heel wat tijd in beslag neemt en niet strookt met het *volledig automatisch* zijn van onze methode, doen we dit hier expliciet niet. Een andere optie is om de in te kleuren zwart-witalbums in te scannen aan een hogere resolutie. Een hogere resolutie maakt het immers makkelijker om de halftoning puntjes automatisch te onderscheiden en te verwijderen. Dit is dan ook de meest aangewezen keuze wanneer een grotere dataset dient ingekleurd te worden.

(a) Originele scan

(b) Lijntekening na verwijderen van halftoning

(c) Ingekleurde versie



(d) Originele scan

(e) Lijntekening na verwijderen van halftoning

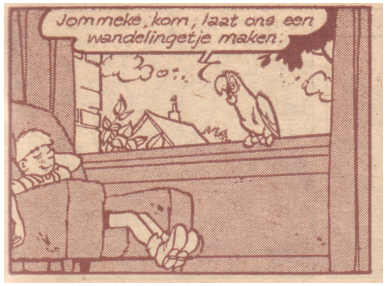
(f) Ingekleurde versie

(g) Originele scan

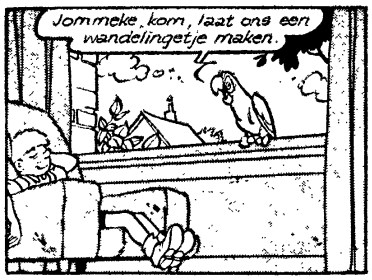
(h) Lijntekening na verwijderen van halftoning

(i) Ingekleurde versie

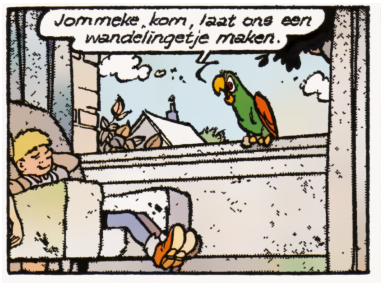
Figuur 6.8: Voorbeeld van enkele geloofwaardige inkleuringen van zwart-witalbums



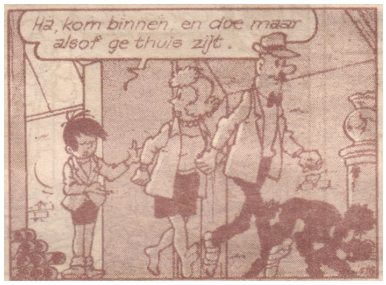
(a) Originele scan



(b) Lijntekening na verwijderen van halftoning



(c) Ingekleurde versie

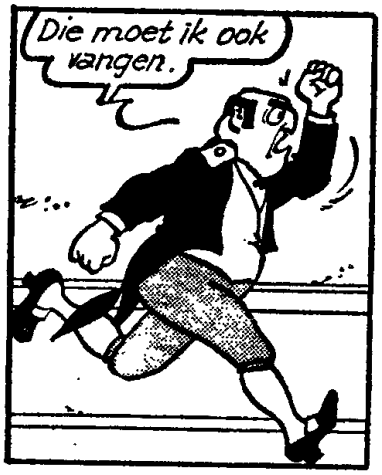


(d) Originele scan

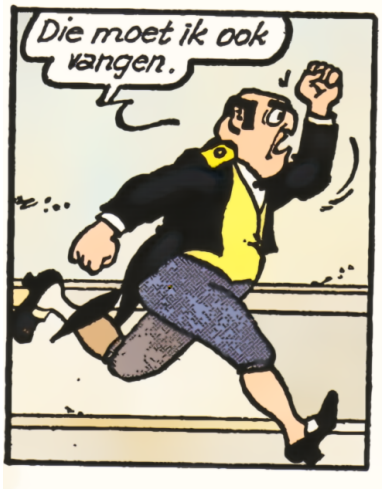
(e) Lijntekening na verwijderen van halftoning

(f) Ingekleurde versie

(g) Originele scan



(h) Lijntekening na verwijderen van halftoning



(i) Ingekleurde versie

Figuur 6.9: Voorbeeld van enkele ongeloofwaardige inkleuringen van zwart-witalbums

(a) Originele scan

(b) Lijntekening na automatisch verwijderen van halftoning. Veel halftoning puntjes blijven aanwezig door het lage contrast van de scan.

(c) Inkleuring van afbeelding 6.10b

(d) Lijntekening na manueel verwijderen van halftoning.

(e) Inkleuring van afbeelding 6.10d

Figuur 6.10: Voorbeeld van invloed van ongewenste halftoning puntjes in de invoer op de inkleuring

Hoofdstuk 7

Conclusie

Het doel van deze masterproef was een methode te ontwikkelen om volledig automatisch binaire strippprenten in te kleuren. We begonnen met het identificeren van enkele verwante problemen, zoals het inkleuren van foto's en semantische segmentatie. Voor deze problemen bleken de best presterende oplossingen telkens gebaseerd te zijn op deep learning. Het grote voordeel van deep learning tegenover eerdere methodes is de mogelijkheid om naast laag-niveau textuur ook hoog-niveau semantiek te leren en te benutten. Dit is van buitengewoon belang voor ons probleem aangezien er enkel vorm en geen textuur aanwezig is in de invoer. Daarom kozen ook wij ervoor om gebruik te maken van een convolutional neural network (CNN).

Een eerste idee was om een eenvoudiger netwerk enkel kleurhints te laten voorspellen. Vervolgens kan als naverwerking elk connected component (een vlak omgeven door zwarte lijnen) ingekleurd worden op basis van deze hints. Bij nadere inspectie bleek deze techniek ontoereikend aangezien een connected component niet altijd overeenstemt met één kleur. Daarom kozen we ervoor om het netwerk zelf verantwoordelijk te maken voor het inkleuren van de hoge-resolutie uitvoer. Dit inkleuren vereist een combinatie van laag-niveau details met hoog-niveau context. Deze combinatie kunnen we mogelijk maken door een U-Net architectuur te gebruiken. Aangezien de invoer een hoge resolutie heeft en bruikbare informatie verspreid is over de volledige prent, is het receptive field van een standaard U-Net echter te klein. Daarom breidden we de architectuur uit met een extra downsampling stap en gedilateerde convoluties, die het receptive field drastisch vergroten met een beperkte computationele meerkost.

Vervolgens was een nieuwe uitdaging om een effectieve loss-functie op te stellen om deze architectuur te trainen. Vaak zijn er namelijk meerdere geloofwaardige inkleuringen mogelijk, waardoor het automatisch beoordelen van de uitvoer moeilijk is. We begonnen met het netwerk te trainen op basis van de L_1-afstand tussen de uitvoer en ground truth. Het resulterende netwerk was zo reeds in staat geloofwaardige inkleuringen te genereren voor een deel van de prenten. Aangezien deze loss niet overweg kan met de multi-modaliteit van het probleem, zien we echter grauwe en slordige resultaten bij moeilijkere prenten. Om dergelijke imperfecties te vermijden, introduceren

we een discriminator in de architectuur. Deze discriminator is geïnspireerd door pix2pix en heeft als doel om de uitvoer van het netwerk niet te onderscheiden te maken van echte inkleuringen. Zo bekomen we een generative adversarial network (GAN), waarbij de discriminator en het inkleuringsnetwerk elkaar steeds verbeteren. Hierdoor verminderde het aantal afwijkingen in de uitvoer en werden de resulterende kleuren levendiger. Een fout die echter nog steeds vaak voorkwam was de aanwezigheid van ongewenste kleurovergangen. Om ook deze fout weg te werken, introduceerden we een loss term gebaseerd op total variation regularization. Deze term bleek erg effectief om de geloofwaardigheid van de inkleuringen te verhogen. De uiteindelijke loss-functie bestaat dan uit drie termen: de L_1-loss, de discriminator loss en de total variation loss.

Om de ontwikkelde methode te testen en te trainen, maakten we gebruik van pagina's uit Jommeke. De methode kan eenvoudig ingezet worden voor andere stripreeksen door opnieuw te trainen met een geschikte dataset van kleurprenten. Eerst evalueerden we de performantie op nooit eerder geziene pagina's uit kleurenalbums. Daartoe zetten we deze pagina's om naar binaire lijntekeningen, die we vervolgens inkleurden met het getrainde netwerk. Vervolgens voerden we een perceptueel experiment uit waarbij deelnemers inkleuringen dienden te beoordelen als echt of vals. Hieruit bleek dat de inkleuringen van onze uiteindelijke methode in $\sim 65\%$ van de gevallen waarnemers kunnen misleiden. Wanneer we dezelfde architectuur trainen met enkel L_1-loss, bekomen we slechts $\sim 53\%$. Dit duidt aan dat de discriminator loss en total variation loss een significante bijdrage leven aan de geloofwaardigheid van de inkleuringen.

Tenslotte zetten we onze methode in om oude zwart-witalbums, die nooit eerder in kleur verschenen, in te kleuren. Tot hiertoe maakten we steeds gebruik van kleurenalbums die we vooraf omzetten naar binaire lijntekeningen. Aangezien er voor deze albums een ground truth beschikbaar is, konden we zo het netwerk trainen en de inkleuringen vergelijken met het origineel. Voor de zwart-witalbums die we nu inkleuren is er geen kleurenversie beschikbaar aangezien deze werden uitgegeven toen de reeks nog in zwart-wit verscheen. Verder wordt het inkleuren van deze albums bemoeilijkt door de oudere tekenstijl die erin gehanteerd wordt. Zo ontbreken veel van de typische kenmerken waaraan personages te herkennen zijn in nieuwere albums. Een andere uitdaging is de aanwezigheid van een halftoning patroon in sommige vlakken. Door de lage resolutie van de beschikbare scans is het moeilijk deze halftoning automatisch weg te halen zonder daarbij andere details te verliezen. Deze factoren leiden ertoe dat de behaalde geloofwaardigheid hier een stuk lager ligt met $\sim 46\%$. Hoewel we op sommige prenten erg overtuigende resultaten behalen, wordt het netwerk vaak gehinderd door de oudere tekenstijl en overblijvende halftoning puntjes. Het manueel kiezen van een drempelwaarde om halftoning weg te halen maakt de inkleuringen geloofwaardiger maar doet afbreuk aan het volledig automatisch zijn van onze methode.

7.1 Toekomstig werk

Bij het opstellen van deze masterproef werd uitvoerig geëxperimenteerd met netwerkarchitecturen, loss-functies en hyperparameters. Onze methode vormt dan ook een degelijke basis om op

verder te bouwen. Vanwege de beperkte tijdspanne en beperkte computationele middelen, was het jammergenoeg niet mogelijk om ieder beloftevol idee te implementeren en te testen. In wat volgt bespreken we enkele van deze mogelijke uitbreidingen.

Een eerste mogelijke richting voor toekomstig werk is het verbeteren van de resultaten op oudere zwart-witalbums. Zoals eerder aangehaald, leiden de overblijvende halftoning puntjes en oudere tekenstijl tot een daling in performantie. Een nauwkeurigere methode voor het weghalen van deze halftoning, zonder daarbij andere details te verliezen, zou de performantie sterk verbeteren. Alternatief kunnen we deze halftoning ook introduceren in de trainingsdata om het netwerk er robuust tegen te maken. Het probleem met de tekenstijl is in wezen een discrepantie tussen de brondistributie (lijntekeningen gegenereerd uit kleuralbums) en de doeldistributie (zwart-witalbums). Omgaan met zo'n discrepantie vormt het onderwerp van het onderzoeksveld *domain adaptation* [101]. Specifiek is een Y-Net [102] een recente architectuur uit dat veld die interessant lijkt voor onze toepassing. Zo kunnen we het verschil tussen de twee distributies opvangen en de performantie verbeteren.

Een tweede mogelijkheid is de methode uit te breiden om consistentere inkleuringen te bekomen. Momenteel gaan we ervan uit dat alle context beschikbaar is in een regio ter grootte van het receptive field in de invoerafbeelding. Als we echter een volledig album inkleuren, willen we ook consistentie kunnen afdwingen over meerdere pagina's. Zo dient een voorwerp dat een bepaalde kleur krijgt op één prent daarna telkens opnieuw dezelfde kleur te krijgen. Netwerken die in staat zijn om te gaan met dergelijke temporele structuren worden *recurrente neurale netwerken* genoemd [103]. Zo'n architectuur zou toelaten om inkleuringen te onthouden en voorwerpen consistent te kleuren doorheen het album [104]. Een andere mogelijkheid is om de gebruiker toe te laten de inkleuring bij te sturen door het aanbieden van optionele kleurhints [105]. Op die manier gebeurt de inkleuring nog steeds automatisch, maar kan de gebruiker eenvoudig correcties aanbrengen indien nodig.

Een derde mogelijkheid is het uitbreiden van de evaluatie. Aangezien het automatisch beoordelen van de geloofwaardigheid van inkleuringen erg moeilijk is, maakten wij gebruik van menselijke observators. Deze perceptuele experimenten kunnen uitgebreid worden om nog meer inzicht te verkrijgen in de verschillen tussen methodes. Zo kan bijvoorbeeld gevarieerd worden hoe lang of hoe groot elke prent getoond. Een andere mogelijkheid is om een framework op te zetten waaraan we resultaten kunnen aftoetsen. Zo zou de invloed van kleurovergangen of grauwe kleuren op de geloofwaardigheid kunnen ingeschat worden. Door vervolgens het aantal dergelijke fouten te kwantificeren, krijgen we een inschatting van de geloofwaardigheid.

Een laatste interessant uitbreiding zou zijn om onze methode te trainen op een dataset die bestaat uit meerdere stripreeksen. Hier maakten we namelijk bij zowel het testen als trainen gebruik van albums uit Jommeke. Dit laat het netwerk toe zich sterk toe te spitsen op het inkleuren van die reeks, maar het getrainde netwerk presteert niet noodzakelijk goed op andere

stripreeksen. Wanneer de train set bestaat uit meerdere stripreeksen, verwachten we dat de performantie op albums uit nieuwe, ongeziene stripreeksen drastisch toeneemt. Op die manier zou het bijvoorbeeld mogelijk zijn prenten uit een stripreeks in te kleuren waarvan geen enkele kleurprent beschikbaar is. Hoewel dit geen aanpassingen vereist aan onze methode zelf, kan het verzamelen van een geschikte dataset en trainen wat tijd in beslag nemen.

Bibliografie

[1] S. Kaji and S. Kida, "Overview of image-to-image translation by use of deep neural networks: denoising, super-resolution, modality conversion, and reconstruction in medical imaging," *Radiological Physics and Technology*, Jun 2019.

[2] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *CoRR*, vol. abs/1811.12231, 2018.

[3] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2015.

[4] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.

[5] A. Karpathy, "Convolutional neural networks for visual recognition, course notes." `http://cs231n.github.io/`. Accessed: 2019-03-06.

[6] A. A. Novikov, D. Major, D. Lenis, J. Hladuvka, M. Wimmer, and K. Bühler, "Fully convolutional architectures for multi-class segmentation in chest radiographs," *CoRR*, vol. abs/1701.08816, 2017.

[7] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," *CoRR*, vol. abs/1511.02680, 2015.

[8] A. Odena, "Open questions about generative adversarial networks," *Distill*, 2019. https://distill.pub/2019/gan-open-problems.

[9] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, (New York, NY, USA), pp. 327–340, ACM, 2001.

[10] G. Charpiat, M. Hofmann, and B. Schölkopf, "Automatic image colorization via multimodal predictions," in *Proceedings of the 10th European Conference on Computer Vision: Part III*, ECCV '08, (Berlin, Heidelberg), pp. 126–139, Springer-Verlag, 2008.

[11] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," *ACM Trans. Graph.*, vol. 21, pp. 277–280, July 2002.

[12] R. K. Gupta, A. Y.-S. Chia, D. Rajan, E. S. Ng, and H. Zhiyong, "Image colorization using similar images," in *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, (New York, NY, USA), pp. 369–378, ACM, 2012.

[13] R. Irony, D. Cohen-Or, and D. Lischinski, "Colorization by example," in *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*, EGSR '05, (Aire-la-Ville, Switzerland, Switzerland), pp. 201–210, Eurographics Association, 2005.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, vol. abs/1502.01852, 2015.

[15] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2414–2423, June 2016.

[16] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman, "Synthesizing obama: Learning lip sync from audio," *ACM Trans. Graph.*, vol. 36, pp. 95:1–95:13, July 2017.

[17] R. Dahl, M. Norouzi, and J. Shlens, "Pixel recursive super resolution," *CoRR*, vol. abs/1702.00783, 2017.

[18] Z. Cheng, Q. Yang, and B. Sheng, "Deep colorization," *CoRR*, vol. abs/1605.00075, 2016.

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[20] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, 2017.

[21] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[22] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 2481–2495, Dec 2017.

[23] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015.

[24] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 1915–1929, Aug 2013.

[25] Z. Wang and S. Ji, "Smoothed dilated convolutions for improved dense prediction," *CoRR*, vol. abs/1808.08931, 2018.

[26] S. Vesal, N. Ravikumar, and A. K. Maier, "Dilated convolutions in neural networks for left atrial segmentation in 3d gadolinium enhanced-mri," *CoRR*, vol. abs/1808.01673, 2018.

[27] Wikipedia, "CIELAB color space — Wikipedia, the free encyclopedia." `http://en.wikipedia.org/w/index.php?title=CIELAB%20color%20space`, 2019. [Online; accessed 06-July-2019].

[28] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," *CoRR*, vol. abs/1603.08511, 2016.

[29] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, Curran Associates, Inc., 2014.

[30] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," *CoRR*, vol. abs/1506.05751, 2015.

[31] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *CoRR*, vol. abs/1606.03498, 2016.

[32] T. Silva, "An intuitive introduction to generative adversarial networks (gans)."

[33] W. Lotter, G. Kreiman, and D. D. Cox, "Unsupervised learning of visual structure using predictive generative networks," *CoRR*, vol. abs/1511.06380, 2015.

[34] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. N. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," *CoRR*, vol. abs/1612.03242, 2016.

[35] G. Antipov, M. Baccouche, and J. Dugelay, "Face aging with conditional generative adversarial networks," *CoRR*, vol. abs/1702.01983, 2017.

[36] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," *CoRR*, vol. abs/1609.04802, 2016.

[37] M. Mathieu, C. Couprie, and Y. Lecun, "Deep multi-scale video prediction beyond mean square error," 11 2016.

[38] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," *CoRR*, vol. abs/1604.07379, 2016.

[39] X. Wang and A. Gupta, "Generative image modeling using style and structure adversarial networks," *CoRR*, vol. abs/1603.05631, 2016.

[40] J. Johnson, A. Alahi, and F. Li, "Perceptual losses for real-time style transfer and super-resolution," *CoRR*, vol. abs/1603.08155, 2016.

[41] Y. Zhou and T. L. Berg, "Learning temporal transformations from time-lapse videos," *CoRR*, vol. abs/1608.07724, 2016.

[42] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. Kweon, "Pixel-level domain transfer," *CoRR*, vol. abs/1603.07442, 2016.

[43] A. B. L. Larsen, S. K. Sønderby, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *CoRR*, vol. abs/1512.09300, 2015.

[44] U. Demir and G. B. Ünal, "Patch-based image inpainting with generative adversarial networks," *CoRR*, vol. abs/1803.07422, 2018.

[45] M. ul Hassan, "Pix2pix: Image-to-image translation neural network." `https://neurohive.io/en/popular-networks/pix2pix-image-to-image-translation`. Accessed: 2018-12-24.

[46] T. Wang, M. Liu, J. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," *CoRR*, vol. abs/1711.11585, 2017.

[47] B. Dorsey. `https://twitter.com/brannondorsey/status/806283494041223168`. Accessed: 2018-12-16.

[48] B. Gondouin. `https://twitter.com/bgondouin/status/818571935529377792`. Accessed: 2018-12-16.

[49] C. Hesse. `https://affinelayer.com/pixsrv/`. Accessed: 2018-12-16.

[50] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," *CoRR*, vol. abs/1707.09405, 2017.

[51] A. Dosovitskiy and T. Brox, "Generating images with perceptual similarity metrics based on deep networks," *CoRR*, vol. abs/1602.02644, 2016.

[52] E. Solie, "Jommeke digitaal." `https://stripheld.hoembeka.be/jommeke/`. Accessed: 2019-02-11.

[53] A. K. NGO HO, J.-C. Burie, and J.-M. Ogier, "Panel and speech balloon extraction from comic books," 03 2012.

[54] M. D. Bloice, P. M. Roth, and A. Holzinger, "Biomedical image augmentation using Augmentor," *Bioinformatics*, 04 2019.

[55] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pp. 958–963, Aug 2003.

[56] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[57] Q. Luan, F. Wen, D. Cohen-Or, L. Liang, Y.-Q. Xu, and H.-Y. Shum, "Natural image colorization," in *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, (Aire-la-Ville, Switzerland, Switzerland), pp. 309–320, Eurographics Association, 2007.

[58] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Transactions on Graphics*, vol. 23, 06 2004.

[59] F. G. Venhuizen, B. van Ginneken, B. Liefers, F. van Asten, V. Schreur, S. Fauser, C. Hoyng, T. Theelen, and C. I. Sánchez, "Deep learning approach for the detection and quantification of intraretinal cystoid fluid in multivendor optical coherence tomography," *Biomed. Opt. Express*, vol. 9, pp. 1545–1569, Apr 2018.

[60] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (G. Gordon, D. Dunson, and M. Dudk, eds.), vol. 15 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 315–323, PMLR, 11–13 Apr 2011.

[61] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[62] D. Mishkin, N. Sergievskiy, and J. Matas, "Systematic evaluation of convolution neural network advances on the imagenet," *Computer Vision and Image Understanding*, 2017.

[63] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[64] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[65] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *CoRR*, vol. abs/1607.08022, 2016.

[66] Y. Wu and K. He, "Group normalization," *CoRR*, vol. abs/1803.08494, 2018.

[67] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning.," *CoRR*, vol. abs/1603.07285, 2016.

[68] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, 2016.

[69] D. Hadfield-Menell, "Berkeley artificial intelligence research - cooperatively learning human values." `https://bair.berkeley.edu/blog/2017/08/17/cooperatively-learning-human-values/`. Accessed: 2018-04-08.

[70] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for neural networks for image processing," *CoRR*, vol. abs/1511.08861, 2015.

[71] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," in *Proceedings of the Eleventh Annual International Conference of the Center for Nonlinear Studies on Experimental Mathematics : Computational Issues in Nonlinear Science: Computational Issues in Nonlinear Science*, (New York, NY, USA), pp. 259–268, Elsevier North-Holland, Inc., 1992.

[72] P. Rodrguez and B. Wohlberg, "A generalized vector-valued total variation algorithm," in *2009 16th IEEE International Conference on Image Processing (ICIP)*, pp. 1309–1312, Nov 2009.

[73] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259 – 268, 1992.

[74] I. Sobel and G. Feldman, "A 33 isotropic gradient operator for image processing," *Pattern Classification and Scene Analysis*, pp. 271–272, 01 1973.

[75] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[76] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[77] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, (New York, NY, USA), pp. 675–678, ACM, 2014.

[78] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[79] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, (London, UK, UK), pp. 9–50, Springer-Verlag, 1998.

[80] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 249–256, PMLR, 13–15 May 2010.

[81] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, July 2011.

[82] T. Tieleman and G. Hinton, "Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning," *Tech. Rep., Technical report*, p. 31, 2012.

[83] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[84] S. Chintala, E. Denton, M. Arjovsky, and M. Mathieu, "How to train a gan." NIPS 2016, 2016.

[85] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *CoRR*, vol. abs/1505.00853, 2015.

[86] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," *CoRR*, vol. abs/1612.07828, 2016.

[87] K. McLAREN, "Xiiithe development of the cie 1976 (l* a* b*) uniform colour space and colour-difference formula," *Journal of the Society of Dyers and Colourists*, vol. 92, no. 9, pp. 338–341, 1976.

[88] G. Sharma, W. Wu, and E. N. Dalal, "The ciede2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations," *Color Research & Application*, vol. 30, no. 1, pp. 21–30, 2005.

[89] M. Habekost, "Which color differencing equation should be used?," *International Circular of Graphic Education and Research*, vol. No. 6, 2013, pp. 20–33, 2013.

[90] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification," *ACM Trans. Graph.*, vol. 35, pp. 110:1–110:11, July 2016.

[91] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *CoRR*, vol. abs/1701.04862, 2017.

[92] A. Campbell, *The Designer's Lexicon: The Illustrated Dictionary of Design, Printing, and Computer Terms*. Chronicle Books, 2000.

[93] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[94] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. on Knowl. and Data Eng.*, vol. 22, pp. 1345–1359, Oct. 2010.

[95] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," *CoRR*, vol. abs/1808.01974, 2018.

[96] J. Ming Guo and S. S, "Digital halftone database (dhd): A comprehensive analysis on halftone types," pp. 1091–1099, 11 2018.

[97] and, "Inverse halftoning for monochrome pictures," in *Proceedings of 1st International Conference on Image Processing*, vol. 2, pp. 1022–1026 vol.2, Nov 1994.

[98] I. Kurilin, I. Safonov, H. Lee, and S. Kim, "Descreening of scanned images," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 7528, 01 2010.

[99] Z. Jin and E. Fang, *Print Inverse Halftoning and Its Quality Assessment Techniques*, pp. 211–220. 01 2018.

[100] L. He, X. Ren, Q. Gao, X. Zhao, B. Yao, and Y. Chao, "The connected-component labeling problem: A review of state-of-the-art algorithms," *Pattern Recognition*, vol. 70, pp. 25 – 43, 2017.

[101] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Mach. Learn.*, vol. 79, pp. 151–175, May 2010.

[102] J. Roels, J. Hennies, Y. Saeys, W. Philips, and A. Kreshuk, "Domain adaptive segmentation in volume electron microscopy imaging," *CoRR*, vol. abs/1810.09734, 2018.

[103] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *CoRR*, vol. abs/1808.03314, 2018.

[104] J. Chen, L. Yang, Y. Zhang, M. S. Alber, and D. Z. Chen, "Combining fully convolutional and recurrent neural networks for 3d biomedical image segmentation," *CoRR*, vol. abs/1609.01006, 2016.

[105] R. Zhang, J. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros, "Real-time user-guided image colorization with learned deep priors," *CoRR*, vol. abs/1705.02999, 2017.

Bijlage A

Voorbeeldinkleuringen

Deze bijlage bevat een reeks voorbeeldinkleuringen van de verschillende varianten van onze methode. We selecteerden deze afbeeldingen op basis van de resultaten van de menselijke evaluatie. Concreet tonen we telkens de prenten waarvan de inkleuring door onze volledige methode (L_1 + discriminator + total variation) als erg geloofwaardig of net erg ongeloofwaardig beoordeeld werd.

A.1 Gezien album: meest geloofwaardige inkleuringen

Inkleuring 1

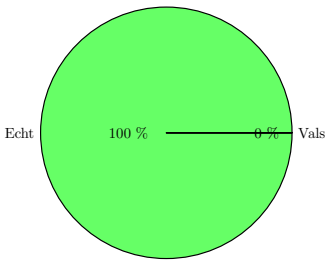
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 2

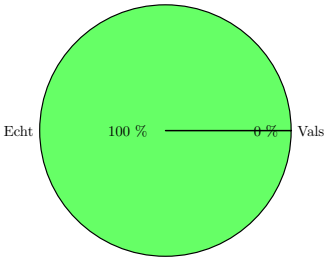
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 3

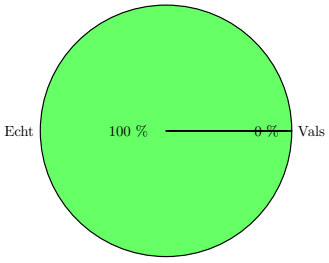
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 4

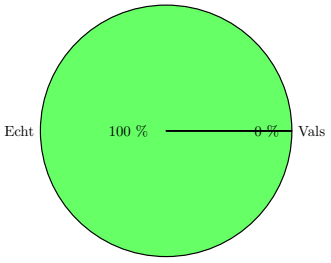
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 5

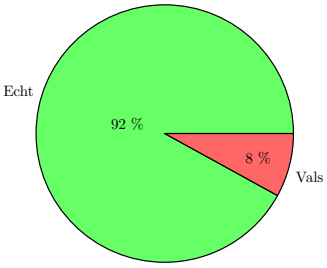
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 6

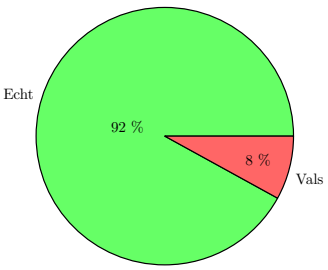
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 7

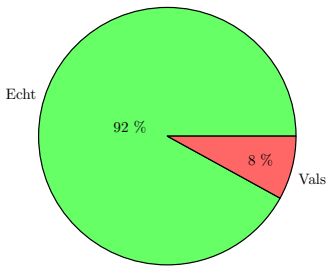
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 8

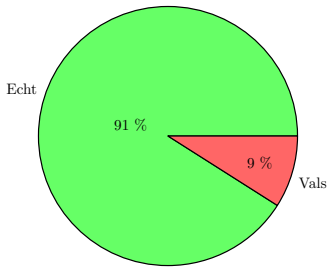
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 9

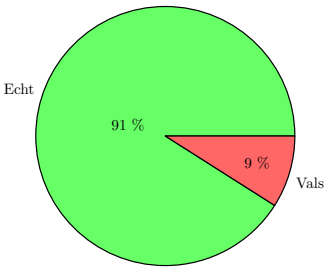
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

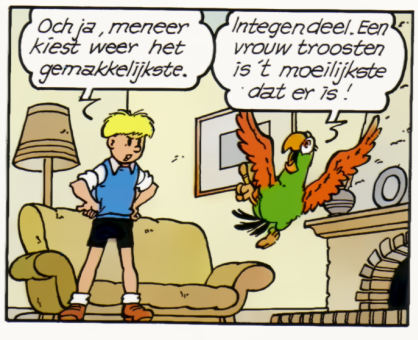
(e) Ground truth



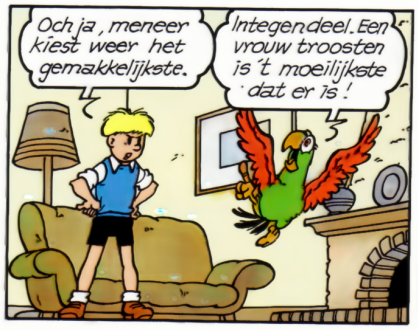
(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 10

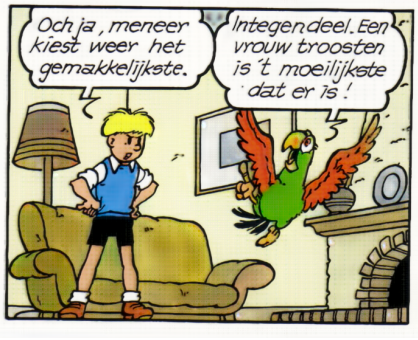
(a) Binaire invoerafbeelding



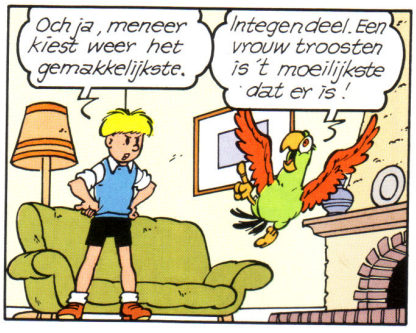
(b) Inkleuring via L_1 + discriminator + total variation



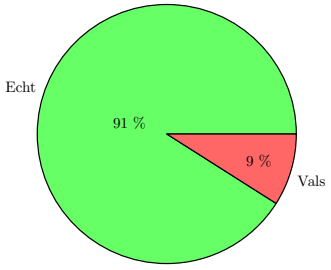
(c) Inkleuring via L_1



(d) Inkleuring via L_1 + discriminator



(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

A.2 Gezien album: minst geloofwaardige inkleuringen

Inkleuring 1

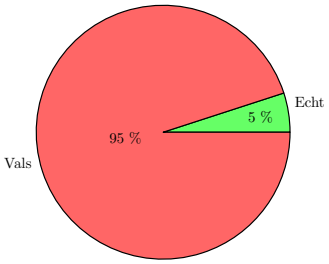
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 2

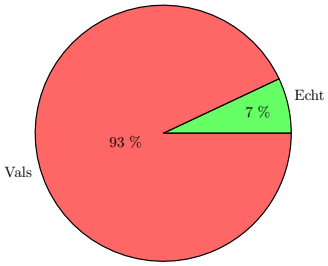
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 3

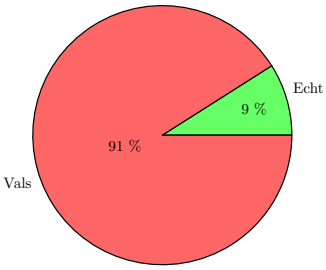
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

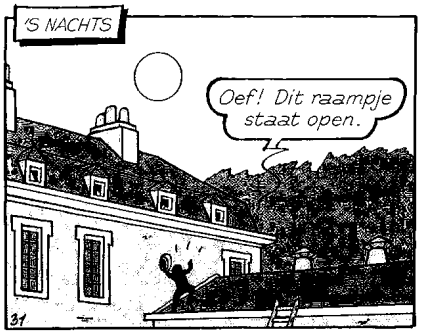
(d) Inkleuring via L_1 + discriminator

(e) Ground truth

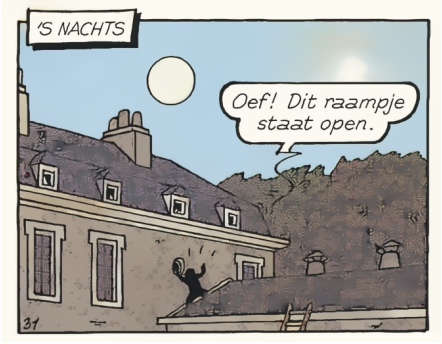


(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 4



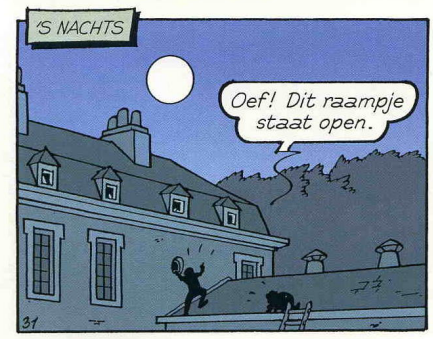
(a) Binaire invoerafbeelding



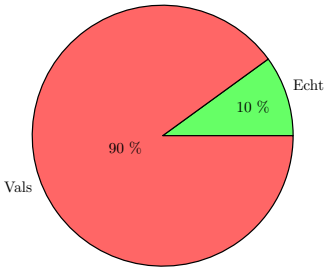
(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator



(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 5

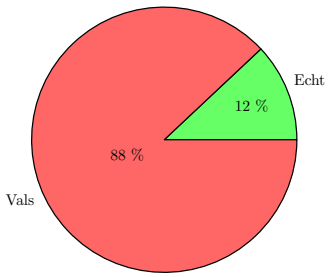
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

A.3 Ongezien album: meest geloofwaardige inkleuringen

Inkleuring 1

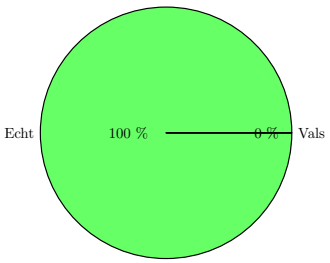
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

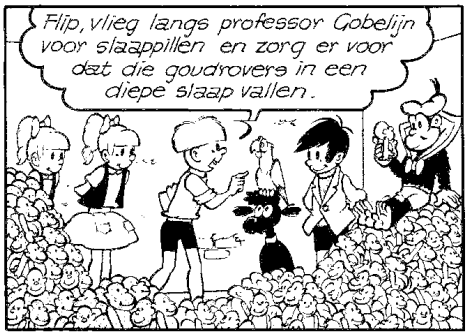
(d) Inkleuring via L_1 + discriminator

(e) Ground truth

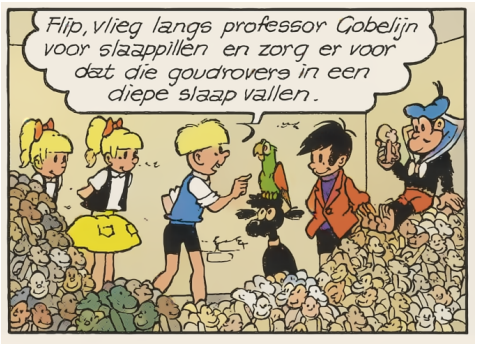


(f) Resultaten perceptuele studie voor inkleuring b

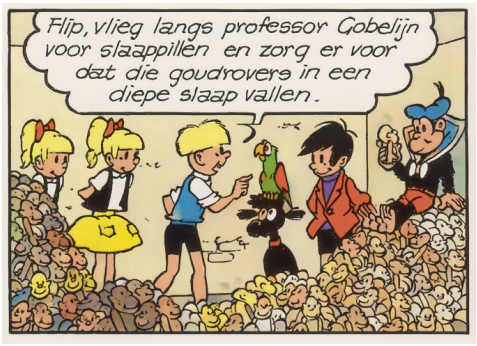
Inkleuring 2



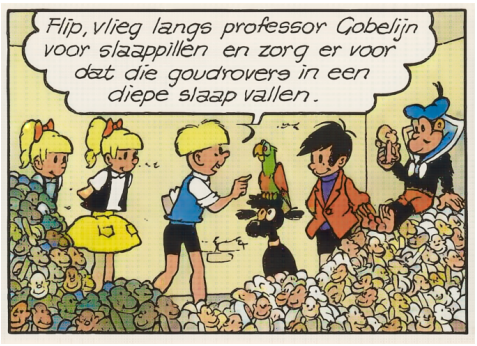
(a) Binaire invoerafbeelding



(b) Inkleuring via L_1 + discriminator + total variation

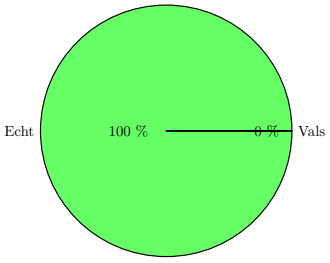


(c) Inkleuring via L_1



(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 3

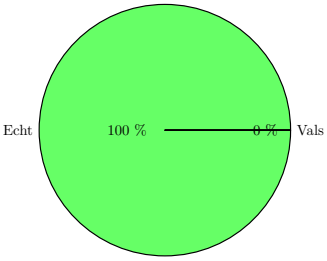
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 4

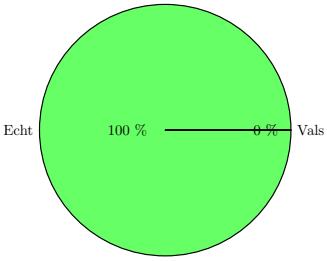
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 5

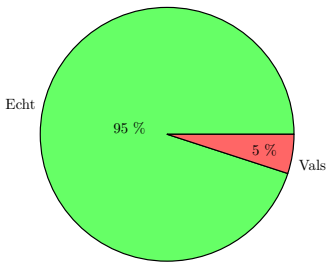
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 6

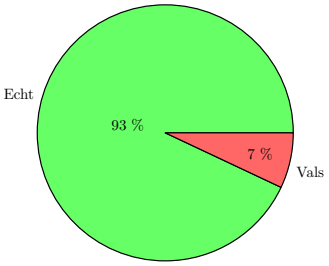
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 7

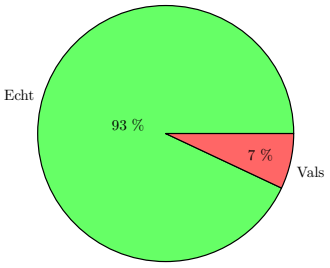
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 8

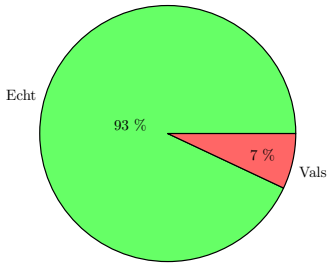
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

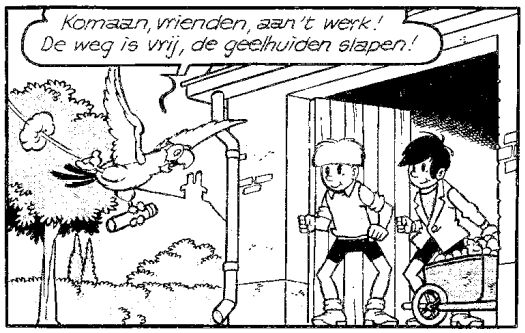
(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 9



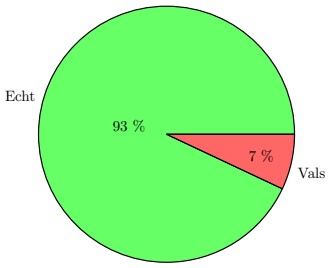
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

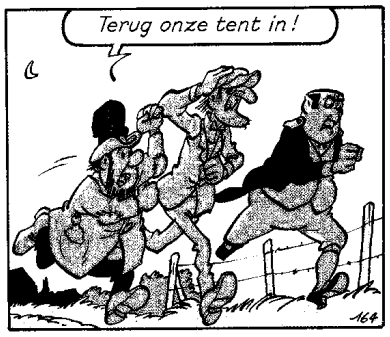
(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

A.4 Ongezien album: minst geloofwaardige inkleuringen

Inkleuring 1



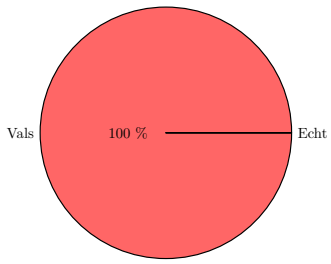
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 2

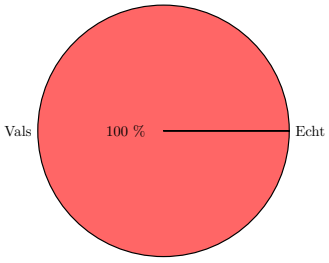
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 3

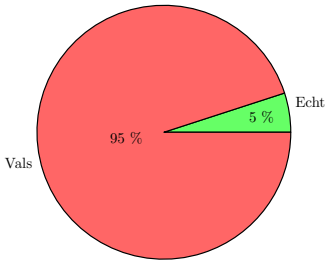
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

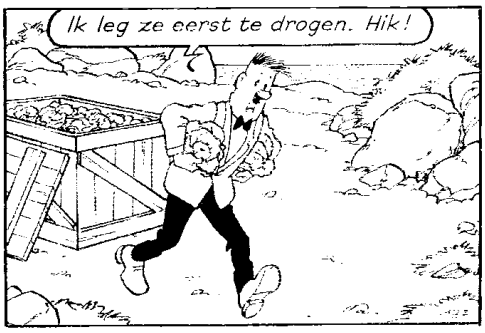
(d) Inkleuring via L_1 + discriminator

(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

Inkleuring 4



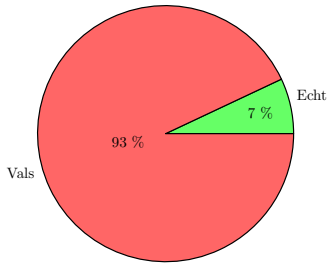
(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Inkleuring via L_1

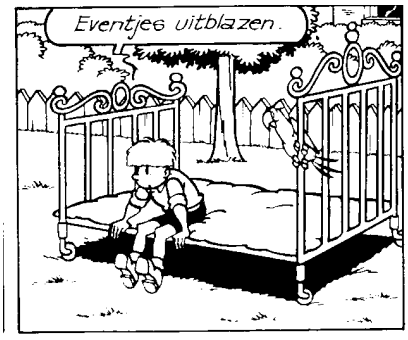
(d) Inkleuring via L_1 + discriminator

(e) Ground truth

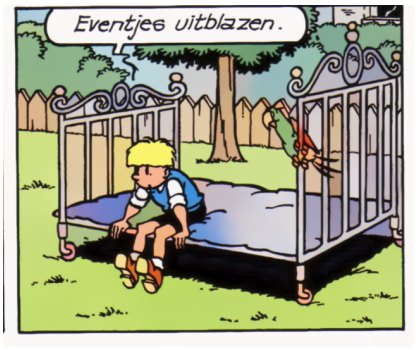


(f) Resultaten perceptuele studie voor inkleuring b

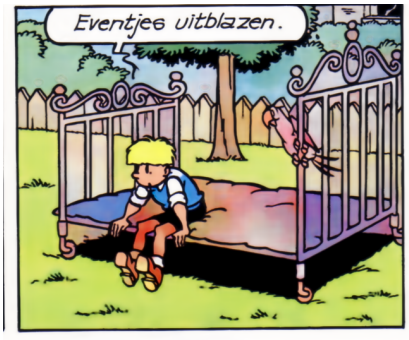
Inkleuring 5



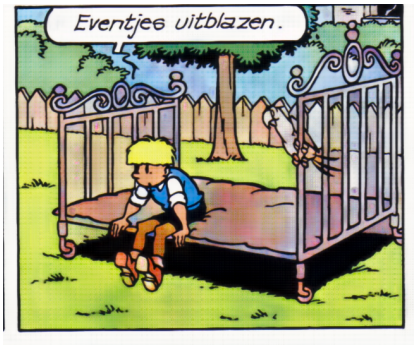
(a) Binaire invoerafbeelding



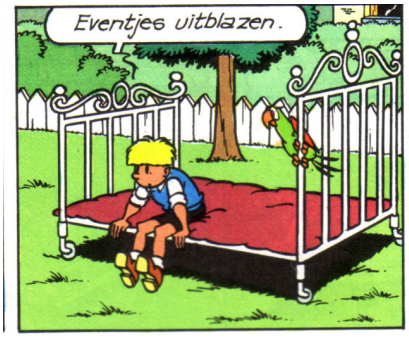
(b) Inkleuring via L_1 + discriminator + total variation



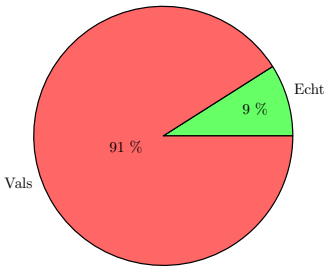
(c) Inkleuring via L_1



(d) Inkleuring via L_1 + discriminator



(e) Ground truth



(f) Resultaten perceptuele studie voor inkleuring b

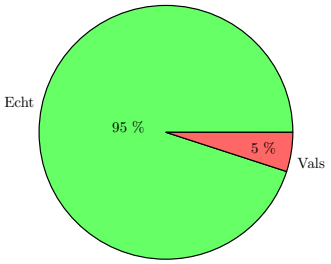
A.5 Zwart-witalbum: meest geloofwaardige inkleuringen

Inkleuring 1

(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Originele afbeelding



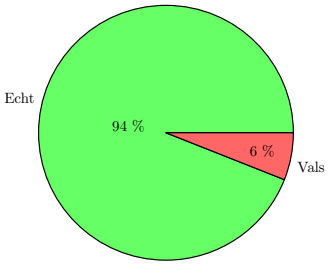
(d) Resultaten perceptuele studie voor inkleuring b

Inkleuring 2

(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Originele afbeelding



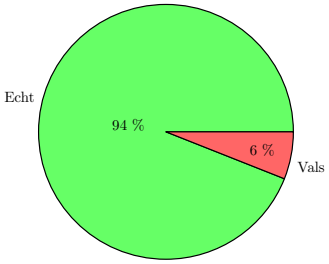
(d) Resultaten perceptuele studie voor inkleuring b

Inkleuring 3

(a) Binaire invoerafbeelding

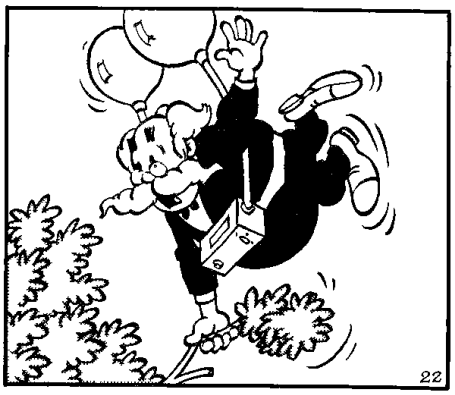
(b) Inkleuring via L_1 + discriminator + total variation

(c) Originele afbeelding

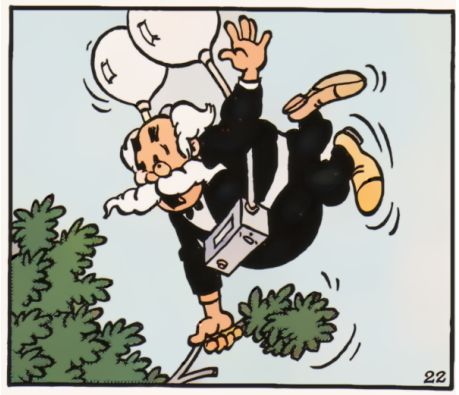


(d) Resultaten perceptuele studie voor inkleuring b

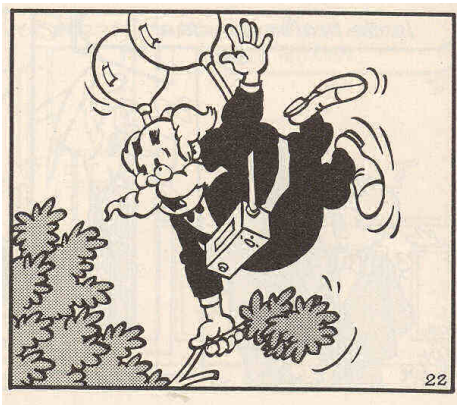
Inkleuring 4



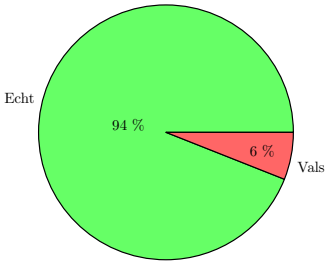
(a) Binaire invoerafbeelding



(b) Inkleuring via L_1 + discriminator + total variation



(c) Originele afbeelding



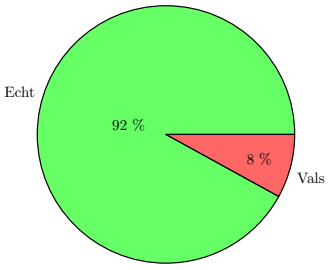
(d) Resultaten perceptuele studie voor inkleuring b

Inkleuring 5

(a) Binaire invoerafbeelding

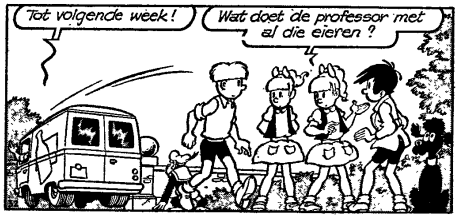
(b) Inkleuring via L_1 + discriminator + total variation

(c) Originele afbeelding



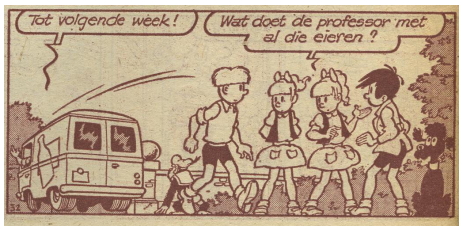
(d) Resultaten perceptuele studie voor inkleuring b

Inkleuring 6

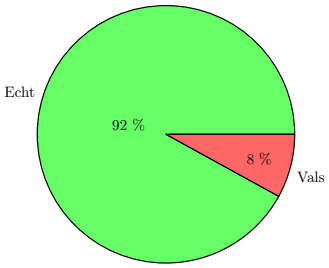


(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation



(c) Originele afbeelding



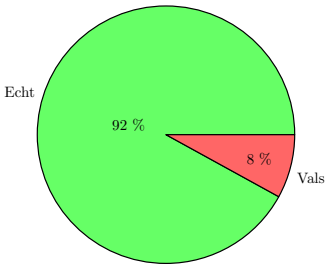
(d) Resultaten perceptuele studie voor inkleuring b

Inkleuring 7

(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Originele afbeelding



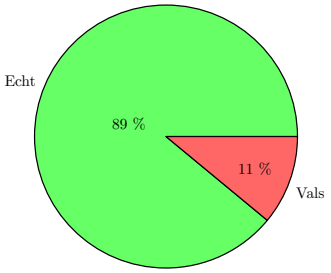
(d) Resultaten perceptuele studie voor inkleuring b

Inkleuring 8

(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

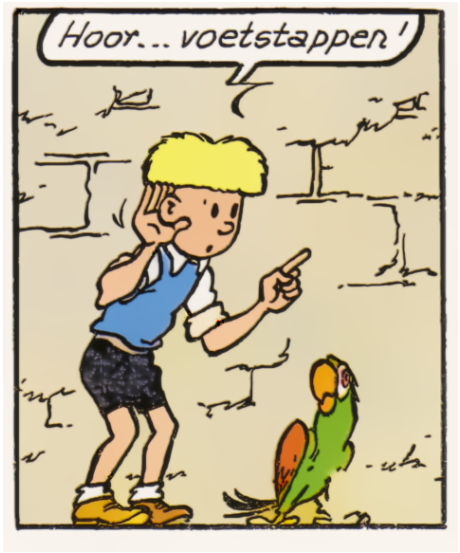
(c) Originele afbeelding



(d) Resultaten perceptuele studie voor inkleuring b

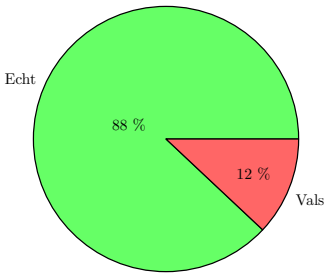
Inkleuring 9

(a) Binaire invoerafbeelding



(b) Inkleuring via L_1 + discriminator + total variation

(c) Originele afbeelding



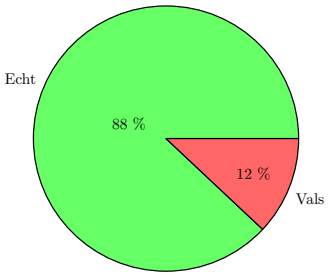
(d) Resultaten perceptuele studie voor inkleuring b

Inkleuring 10

(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Originele afbeelding

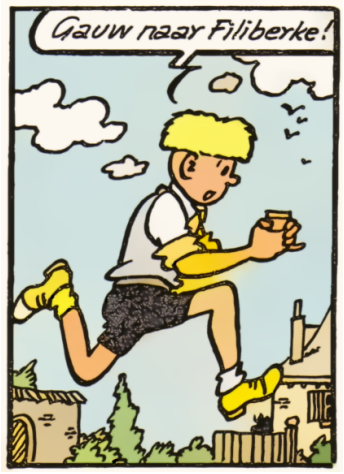


(d) Resultaten perceptuele studie voor inkleuring b

A.6 Zwart-witalbum: minst geloofwaardige inkleuringen

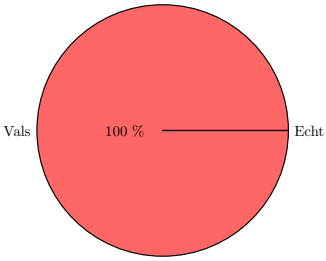
Inkleuring 1

(a) Binaire invoerafbeelding



(b) Inkleuring via L_1 + discriminator + total variation

(c) Originele afbeelding



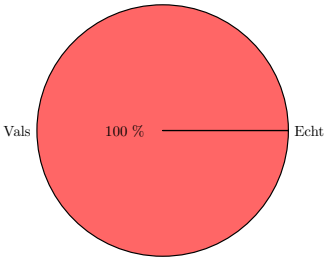
(d) Resultaten perceptuele studie voor inkleuring b

Inkleuring 2

(a) Binaire invoerafbeelding

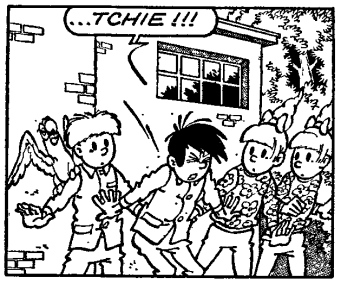
(b) Inkleuring via L_1 + discriminator + total variation

(c) Originele afbeelding

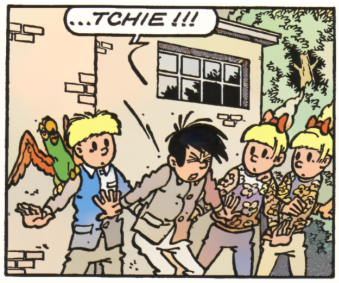


(d) Resultaten perceptuele studie voor inkleuring b

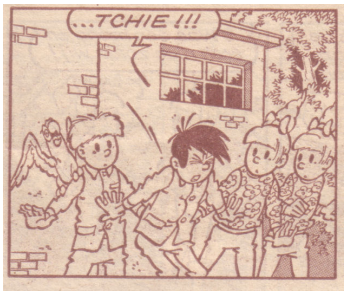
Inkleuring 3



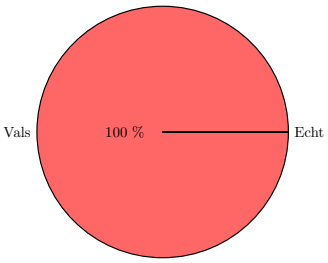
(a) Binaire invoerafbeelding



(b) Inkleuring via L_1 + discriminator + total variation



(c) Originele afbeelding



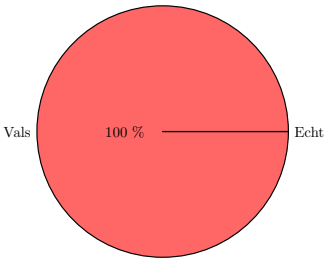
(d) Resultaten perceptuele studie voor inkleuring b

Inkleuring 4

(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Originele afbeelding



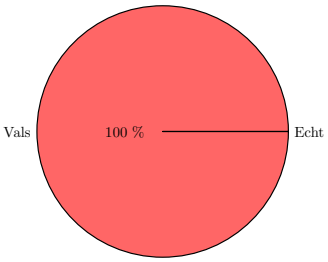
(d) Resultaten perceptuele studie voor inkleuring b

Inkleuring 5

(a) Binaire invoerafbeelding

(b) Inkleuring via L_1 + discriminator + total variation

(c) Originele afbeelding



(d) Resultaten perceptuele studie voor inkleuring b