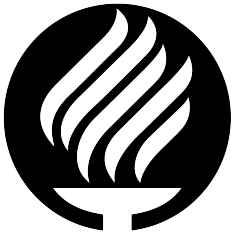
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Querétaro
Departamento de Computación y Mecatrónica



Tecnológico de Monterrey

Image Colorization Using Conditional Generative Adversarial Networks

by

Mariana Pérez García

Proyecto Integrador para el Desarrollo de Soluciones Empresariales

Ingeniería
en
Sistemas Computacionales

Asesor: Dr. José A. Cantoral-Ceballos

Santiago de Querétaro, Querétaro, México
03/06/2020

Abstract

Colorization refers to the process of adding colors to monochromatic frames which assigns a value to the RGB channels in it. Image colorization is a common problem in the art community but more recently it has been a recurrent subject that the Machine Learning community has been trying to tackle. The problem is that colorization of images does not have an exact objective solution. Previous works in the area that are able to colorize different types of greyscale images, meaning different subjects, such as everyday photographs, for instance landscapes, portraits, flowers, etc., tend to output desaturated results, and the ones that have an satisfactory performance tend to rely on significant human interaction. The goal of this paper is to give insight in the development and results of using a conditional generative adversarial network using ResNet50, a convolutional neural network that is 50 layers deep, as a high-level feature extractor. The proposed model was implemented and evaluated using a "Colorization Turing Test" [1].

Contents

Acrónimos

GAN	**G**enerative **A**dversarial **N**etwork
cGAN	**c**onditional **G**enerative **A**dversarial Network
CNN	**C**onvolutional **N**eural Network
LSTM	**L**ong **S**hort **T**erm **M**emory

Chapter 1

Introduction

Photography coloring or overpainting has been a common practice since the nineteenth century. When photography was introduced to society in 1839, it was undoubtedly the sensation of the technological scene, however, these photographs were color-blind in the sense that colors were not rendered, creating monochromatic images. In 1841, Henry Collen, a miniature painter started retouching photographs, and began experimenting not only with simple retouching for the sake of permanence, but also with coloring, thereby enhancing the illusion of reality. Practices that began with studio portraits were also extended to landscapes and eventually finished with photographs. In 1816 Lafon Camarsac used the production on lantern slides, by mixing the bitumen with turpentine and black resin, to create after exposure regions of various surface textures on hard substrates according to the difference in light exposure. The differentiation arose from the photo-polymerization properties of the bitumen. Oxide pigments could then be applied for later firing, resulting in a colored image, however, the final product could still be improved. This technique had a problem though, the slides did not take into account the color temperature of the light source, which, inevitably, affected the choice of the optimum coloring tint, in addition to effects caused by neighbour colors [2].

In the 1880's, Hans Jakob Schmid, invented the technique called Photocrom Process, which are ink-based images produced through the direct photographic transfer of an original negative onto litho and chromographic printing plates. The resulting printings look deceptively like color photographs. However, when viewed with a magnifying glass, a body of small dots were visible, thus compromising the authenticity of the image. Likewise the reproduce colors were not as

realistic as one would have expected. Nonetheless, the photo-mechanical process permitted the mass production of colorful printings. Each color in the final print required a separate asphalt-coated lithography stone, usually a minimum of six stones and often more than ten stones [3].

FIGURE 1.1: Example of Photochrom process, where the original greyscale image (left) can be seen alongside its colorization(right). Detroit Photographic Co.,Betsy Ross house,Philadelphia. Photochrom print, copyrighted 1900.

Today, image colorization is still performed by hand using graphic design software such as Photoshop or Adobe Illustrator along with a vast number of online resources that have led artists to reconstruct images with more accuracy than previous methods. However, it is a lengthy process where a single frame can take up to a month to be colorized, due to the extensive research needed to provide the most realistic and accurate colorization.

Image colorization has always been an interesting and relevant problem, not only for the art community but also, most recently, for the Computer Science community since it represents an important milestone for Machine Learning. This is supported due to the large variety of applications such as color restoration of old images, medical image colorization, animations, computer

graphics, and scientific visualization[4]. Merely in the medical field, colorization of images, such as X-rays or MRI images [5], can aid in the visual discrimination of biological structures and surgeons diagnoses [6]. In addition, the colorized images can also help other machine learning algorithms to build a clinical decision support system, such as the computer-aided detection and diagnosis [7]. This is because colorized images have a potential to portray richer information than conventional monochromatic images.

Traditional colorization created a need of trying to automate the process in order to do it in less time without losing the true colors of the picture [8]. However colorization of images is a complicated task because it requires prior knowledge about the image content and manual adjustments in order to achieve the best quality. Furthermore the possible colors for a single object means there is no unique solution [9].

Chapter 2

Related Work

Artificial intelligence methods have been used to generate plausible solutions to the colorization problem. However, designing and implementing an effective and reliable system that automates this process still remains a challenging task, with proposed models relying on two main approaches. Colorization techniques can be classified under two categories, semi automatic coloring and automatic coloring [10]. The first paradigm requires the user to assign colors to regions over a greyscale image whilst the algorithm extends that information to the whole image based on the assumption that neighboring pixels with similar intensities have the same color. This approach was popularized by Levin et al. 2004 [11].

More recent work has been carried out under the same paradigm, achieving less user interaction without compromising the quality of the results, as presented in Zhang et al. 2017, where a CNN was proposed to colorize images using in real time user interaction. This model was able to colorize images with sparse points, in contrast to previous models where more than fifty strokes were needed to get a realistic image colorization [11]. However, the network may produce undesired non-local effects depending on the reference points, e.g. points added on a foreground object may cause an undesired change in the background [12].

However, although models based on this paradigm deliver high quality results, they still require significant user interaction as regions must be explicitly defined by the user even when there is minimum uncertainty in the coloring problem. Furthermore, since the colorization relies on an user input, the resulting image could end up biased towards the user *a priori* knowledge.

To address these limitations, researches have explored more data driven methods, where the algorithm learns the individual pixel colors from a color image with similar content [9]. The automatic coloring approach may be achieved from two perspectives. The first one is to try to match a greyscale image to an exemplar color image from a database and 'steal' the colors from the original photograph, as popularized by Hertzmann et al. 2001 [13], or by learning parametric mappings from greyscale to color from large-scale image data as proposed by Iizuka et al. in 2006 [8], Larsson et al. in 2006, [14] and Zhang et al in 2006 [1] [12].

The basic structure of automatic colorization is almost similar to the color transfer approaches. Firstly, the features from a reference image and the greyscale image are extracted and then those features are matches using either a luminance or texture matching algorithm. The the color is transferred from reference image to greyscale image. Finally, the colored version of the greyscale image is retrieved [10].

Hertzmann et al. 2001 presented the idea of "stealing" colors from a source photograph and transfer them onto a greyscale image to produce colorization, in a process called image analogies, which is divided in two stages. The design phase, in which a pair of images are filtered to be used as training data, and the application phase, when the resulting filter is applied on the target image. This is the base of style transfer where images can get filters according to a source image, however this approach was not used as a method to colorize images until Welsh et. al. in 2002 [13], where image analogies were used to propose a model capable of colorizing an input image by transferring the color from a related reference image by matching luminance and texture information between the two images. The color is transferred from the source to the target if the pixels match in both images [4]. Because a single luminance value could represent entirely different parts of an image, statistics were gathered in the neighboring pixels to guide the matching process. The results with this approach were good, however the quality of the result depended heavily on the choice of reference. [15]. To remove this spatial consistency problem, "swatches" were used, and users were asked to identify and associate these swatches in both source and destination images to indicate how colors should be transfered [15] [10].

In 2018 He et. al. expanded more in the topic of color transfering by proposing a model that selects, propagates, and predicts colors from large-scale data by using reference images unrelated

to the input greyscale image. This approach solves the problem of the user assigning unrelated colors to the greyscale image, however the model requires to enforce luminance similarity in the local regions [16].

Convolutional Neural Networks (CNNs) have been proven experimentally to nearly halve the error rate for object recognition, making them the most popular tool not just in image colorization but in the whole computer vision community [17]. Using the base of image analogies, CNNs can be trained on colored images and their greyscale counterpart to predict color on new datasets. In 2015 Cheng et. al. proposed a deep neural network that treated colorization as a regression problem. The model used DAISY features and semantic features as inputs in order to provide a better colorization in contrast with the state of the art at that time [18]. In order to solve the discrepancy of plausible colors for each pixel, a prediction of the distribution of all the possible colors was developed. This encourages models to exploit large scale datasets during training. However, the resulting images tended to look desaturated, mainly because of the high variability in the colorization space and because the loss functions in the last layers encourage conservative predictions, such as brownish/sepia. [1].

In 2016 Larsson et al. developed a model that was able to colorized images by exploiting low level and semantic representations. This was achieved by making the deep network predict per-pixel color histograms of the image to handle uncertainty and ambiguities inherent in colorization. The results achieved were impressive, however the model had some common problems such as texture confusion, color bleeding and objects not being able to be recognized [14]

Also in 2016, Iizuka et. al.[8] proposed a encoder-decoder arquitecure with CNNs and using global-level and mid-level features to encode the images and colorize them, and in 2017 Baldssarre et.al. [19] improved this architecture by applying transfer learning as a high level feature extraction using Inception-ResNet-v2 to enhance the coloring process improving results considerably, however the resulting images tended to look desaturated in contrast to the ground truth.

Currently there is a novel approach that has gained traction, the generative adversarial networks, also called GANs. This model was proposed by Goodfellow, et al.[20] . According to

Goodfellow et. al. a GAN is a framework for estimating generative models via an adversarial process, in which two models are trained simultaneously, a generative model $\mathbb{G}$ that captures the data distribution, and a discriminative model $\mathbb{D}$ that estimates the probability that a sample came from the training data rather than $\mathbb{G}$. The training procedure for $\mathbb{G}$ is to maximize the probability of $\mathbb{D}$ making a mistake, this is commonly called min-max game. This interaction can be expressed as the equation 2.1 and equation 2.2.

$$\min_{\theta_G} J^{(G)}(\theta_D, \theta_G) = \min_{\theta_G} \mathbb{E}_z[\log(1 - D(G(z)))]. \tag{2.1}$$

$$\max_{\theta_D} J^{(D)}(\theta_D, \theta_G) = \max_{\theta_D}(\mathbb{E}_x[\log(D(x))] + \mathbb{E}_z[\log(1 - D(G(z)))]). \tag{2.2}$$

The first equation explains the loss of the generator while minimizing the probability that the discriminator makes a correct prediction in the given input, meanwhile the second equation is the maximization of the probability of the discriminator in assigning the correct label. Both equations can be synthesize in a single equation showed in the equation 2.3

$$\min_G \max_D V(G, D) = \mathbb{E}_x[\log D(x)] + \mathbb{E}_z[log(1 - D(G(z)))]. \tag{2.3}$$

A Generative Adversarial Network can improve results as demonstrated in case studies, however it can be challenging to train because as explained by Goodfellow et. al. the discriminator must be synchronized with the generator, otherwise the generator will have a near-zero gradient during back-propagation, slowing down convergence rate [21].This is especially critical at early stages in training because the generator is still learning, therefore it is easier for the discriminator to reject generated samples with high confidence because they are clearly different from the training data.

It is important to notice that whilst in a traditional GAN, the input of the generator is random noise, in the presented problem the input is a greyscale image. This was addressed by using a variant of a GAN called conditional generative adversarial network (cGAN) [21][22]. This model allows the generative model $\mathbb{G}$ to a vector $\hat{Y}$ as input data, in order to train and test the discriminator $\mathbb{D}$.

In 2018. Isola et. al. investigated Conditional Adversarial Networks as a general purpose solution to image-to-image translation problems, this due to the network's ability of learning from the mapping of an input image and the loss during training, which creates a versatile model. Their work demonstrated that it is possible to apply generative adversarial networks to different problems such as synthesizing photographs from label maps, reconstructing objects from edge maps, and colorizing images, among other tasks. The results show that the conditional GAN is able to produce compelling colorizations but can fail by producing grayscale or desaturated results [22]. Later that year Nazeri et. al. improved the architecture of the cGAN by applying the concept of encoder-decoder in the generator where the layers in the encoder are concatenated to layers of the decoder. The architecture was also improved by researching different training strategies such as alternative cost functions, one sided label smoothing, applying batch normalization to prevent the collapse of a parameter setting where it always emits the same output, momentum reduction and using LeakyReLU as an activation function instead of ReLU. This proposal was able to produce better looking images than a conventional U-Net, however the network was trained on the CIFAR-10 dataset and when tested with the Places365 dataset the network produced mis-colorizations where images contained high levels of texture details [21].

Chapter 3

Proposed Model Implementation

The approach in this paper to achieve colorization uses the premise that given a lightness input channel $X \in \mathbb{R}^{H \times W \times 1}$, there is a mapping $\hat{Y} = \mathbb{F}(X)$ to the two associated color channels $Y \in \mathbb{R}^{H \times W \times 2}$, where H and W are image dimensions. [1] In order to produce the previous assumption the CIELab color space is used. This color space is defined by three channels, the L* which defines lightness, a* denotes the red/green values, and b* which defines the yellow/blue values [23]. The CIELab color space was used because of its equal perceptual differences for equal changes in the coordinates L*, a* and b* and because the color differences are defined as Euclidean distances as opposed to the RGB color space whose values are nonlinearly distorted values for each channel [24].

In order to arrive to the proposed architecture of the cGAN using transfer learning, first, a 'normal' conditional GAN was developed in order to make a comparison between results and training times between the network using transfer learning and the one that learns from zero. This model consisted of a common Sigmoid classifier model and a U-Net, which is an architecture proposed in 2015 by Ronneberger et al. and is characterized by having the whole model in two parts, a contracting path that captures context, and a symmetric expanding path that enables precise localization, as a generator. [25]. Figure 3.1 [22] [21] and 3.2 show the initial architecture of both the generator and the discriminator without transfer learning.

The model's input is a batch of 32 images sized 256x256. Each image first is converted from the RGB to the CIELab color space. The resulting Lab images are then separated into two

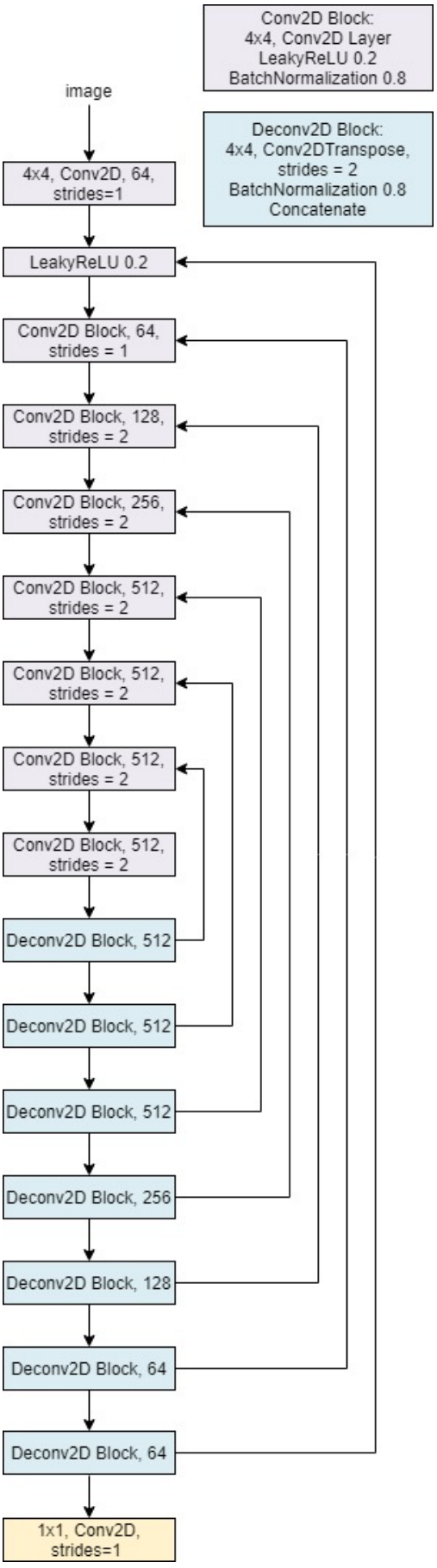


FIGURE 3.1: Architecture of the U-Net generator without transfer learning

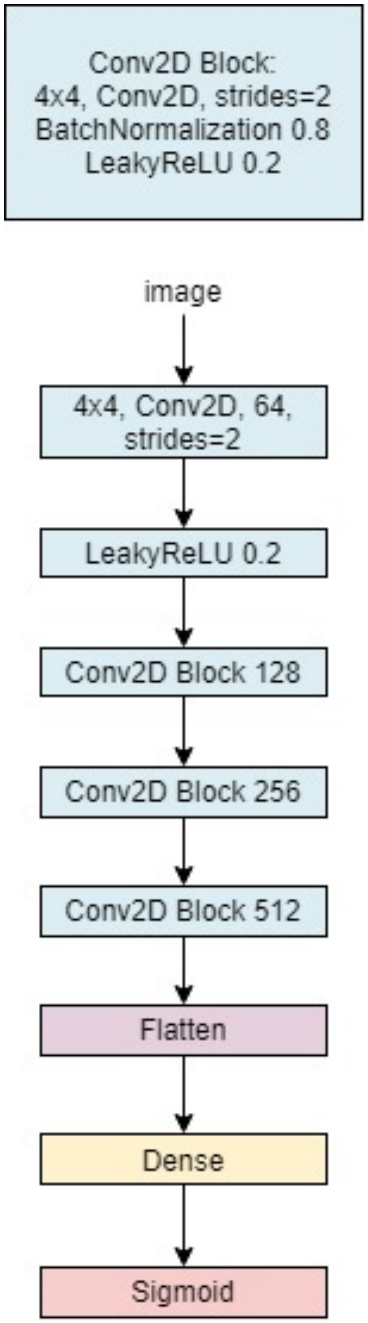


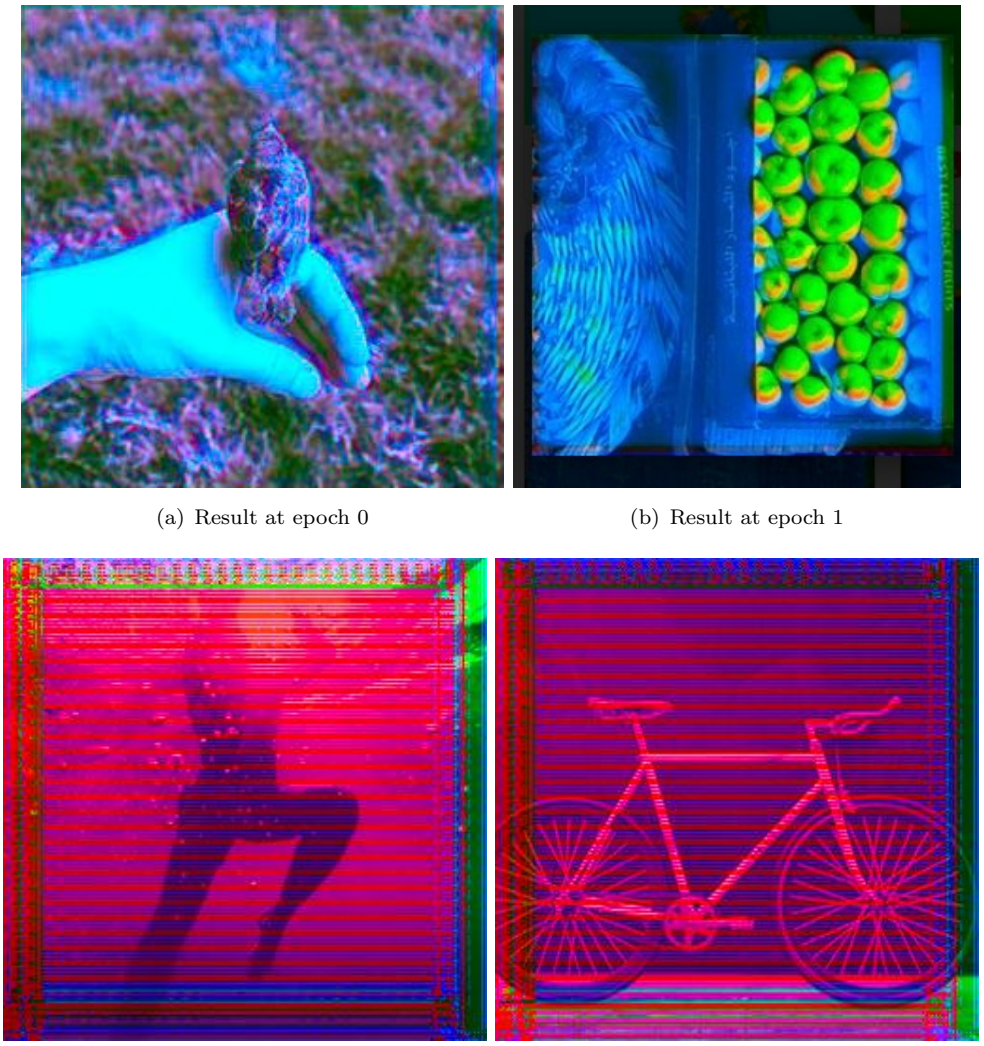
FIGURE 3.2: Architecture of the sigmoid discriminator

vectors, vector $\hat{X}$ which consists only of the L* layer and vector $\hat{Y}$ which stores the remaining a* and b* layers. The generator is first given one half of a batch of images from the $\hat{X}$ vector, resulting in a new vector $\hat{Y}'$ which contains the predicted a* and b* channels. Vectors $\hat{Y}$ and $\hat{Y}'$ are then given corresponding labels consisting of real numbers from 0 to 0.99 where the range from 0 to 0.49 is used to represent a real a* and b* image, whilst the range from 5 to 0.99 identifies a fake or prediction. These two vectors are then fed to the discriminator which estimates the probability of an image in vectors $\hat{Y}$ and $\hat{Y}'$ of being part of the original dataset. During this process the loss of the discriminator is used to train the generator to improve its performance in order to get better predictions. Using labels with noise can create some spikes in the discriminator loss, however it helps preventing both networks from halting each other by denying the discriminator to reject generated samples with high confidence.

However with this architecture the desired results were not achieved, figure 3.3 shows some resulting images of this network. The results at epoch 0 and epoch 1 can show that the generator was learning and identifying objects and had a high level feature extraction, however results at epoch 40 and 44 illustrate that the generator was diverging and displaying vanishing gradient symptoms as (regardless of the epoch or image sample) the result was always the same. At first glance, it seemed the generator is the problem, however in retrospective the problem with this architecture lies the discriminator because it is not learning properly, therefore deviating the learning process of the generator.

Therefore, the discriminator was changed from a common convolutional sigmoid architecture to a convolutional "PatchGan" classifier architecture, which penalizes structure at the scale of the image patches as opposed to penalization as a whole [22]. Figure 3.4 illustrates the new architecture of the discriminator. Changing the discriminator's architecture allowed the generator to learn without diverging. Figure 3.5 shows some results during training. The resulting images however do not look realistic enough due to the generator's inability to predict the color in certain areas, as it is possible to see the red and blue patches on the surface of the building.

Networks that use Convolutional layers are usually trained using models such as Inception, ResNet or VGG [19]. However the ResNet model was used instead of the others because of its low classification error, which is approximately a 3.57 percent error as opposed to a 7.3 for



(a) Result at epoch 0

(b) Result at epoch 1

(c) Result at epoch 40

(d) Result at epoch 44

FIGURE 3.3: Results from the model without transfer learning and with the sigmoid classifier architecture as the discriminator

VGG and a 6.7 for Inception [26]. ResNet is an architecture proposed in 2015 by He K. et al. This network reformulates the layers as learning residual functions with reference to the layer inputs instead of learning unreferenced functions [27]. This approach introduces a term called "identity shortcut" which means the skip of one or more layers. This method deals with a common but notorious issue in deep neural networks, the vanishing gradient problem. This problem happens when the network starts converging, the accuracy gets saturated and degrades rapidly. ResNet50 is a smaller version of the the ResNet152 model, in which only the first 50 layers are used. The ResNet model consist of 5 stages each with a convolution and identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers. The last block of the model consist in an Average Pooling layer, a Flatten layer and a SoftMax

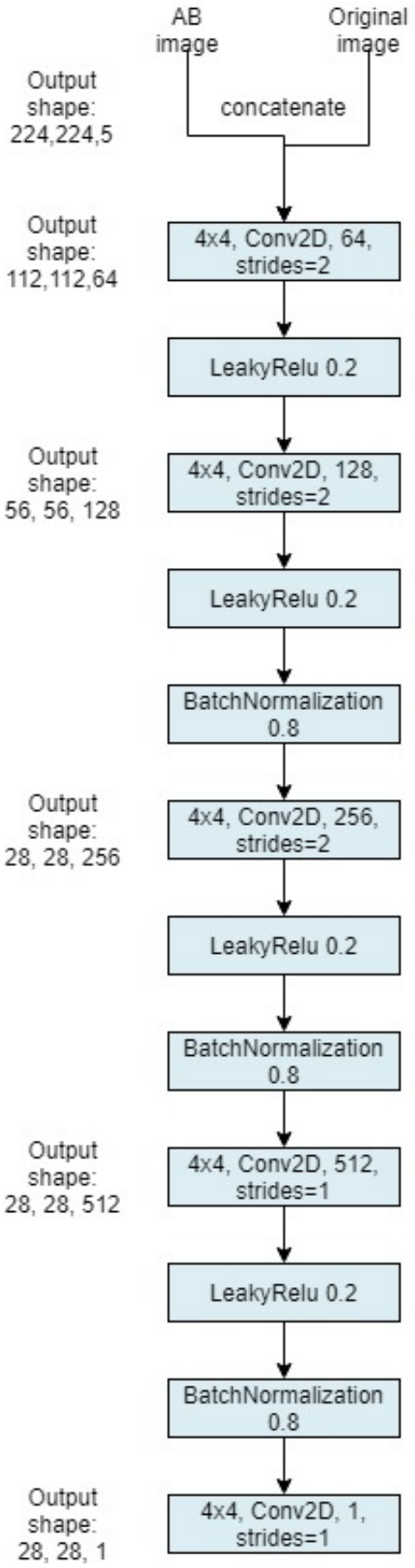


FIGURE 3.4: Architecture of the PatchGan discriminator.

FIGURE 3.5: Results from the model without transfer learning at epoch 47 after changing the discriminator's architecture

layer [27].

The proposed model in this paper is a conditional Generative Adversarial Network using an Encoder-Decoder [28] architecture for the generator. The generator uses Restnet 50 as a high feature extractor. This architecture is based on the works done in Image-to Image Translation with conditional adversarial networks by Isola et. al 2017 [22], Image Colorization with Generative Adversarial Networks by Nazeri et. al. 2018 [21], and Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2 by Baldassarre et.al. 2017 [19].

The Encoder-Decoder architecture was proposed by Sutskever et al. in 2014. This model consist of a multilayer Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from a vector [28]. As opposed to a U-Net architecture the Encoder-Decoder is not connectected through a series of nested dense convolutional blocks. ResNet 50 is used as the encoder, meanwhile the decoder consist of Convolutional 2D and Upsampling layers in order to estimate the output from the encoder. The proposed architecture is illustrated in Figure 3.6 [27]

In order to train or use the network, ResNet50 was used as a pre-trained model to extract the

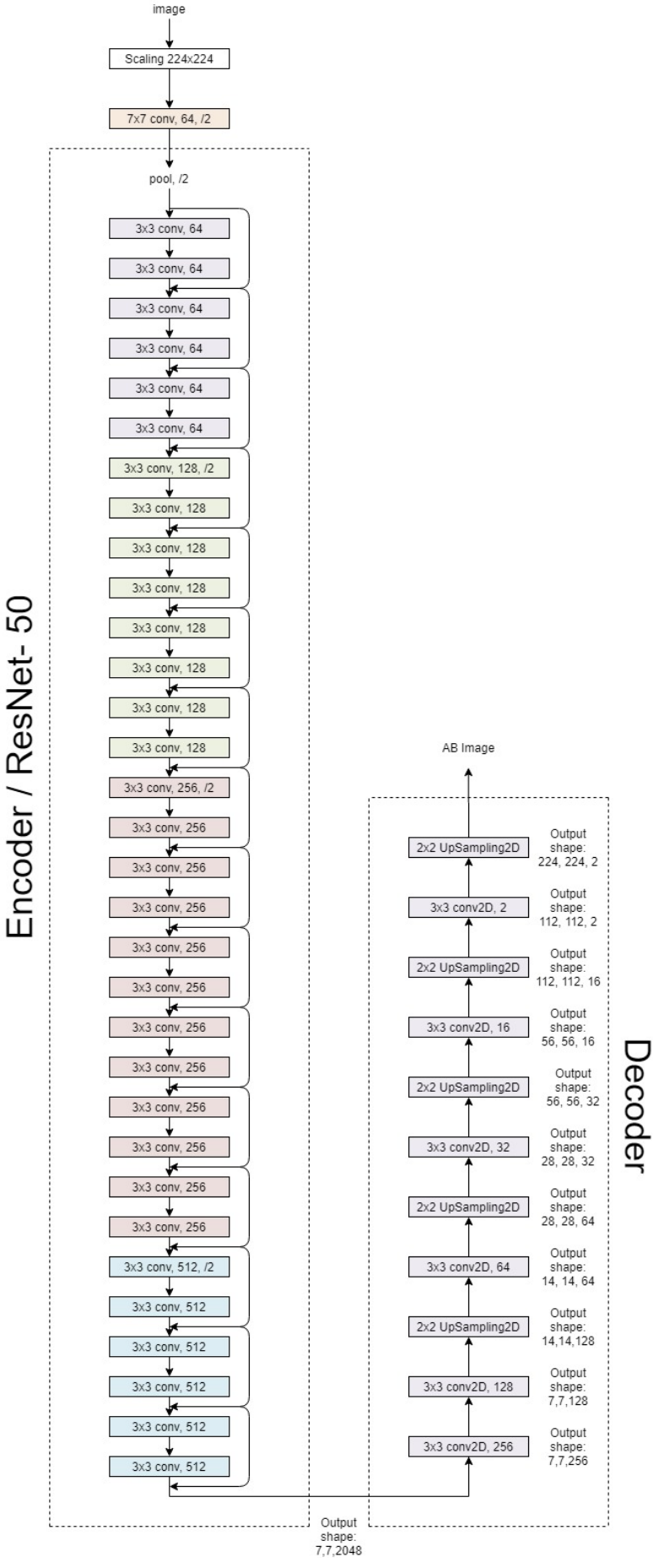
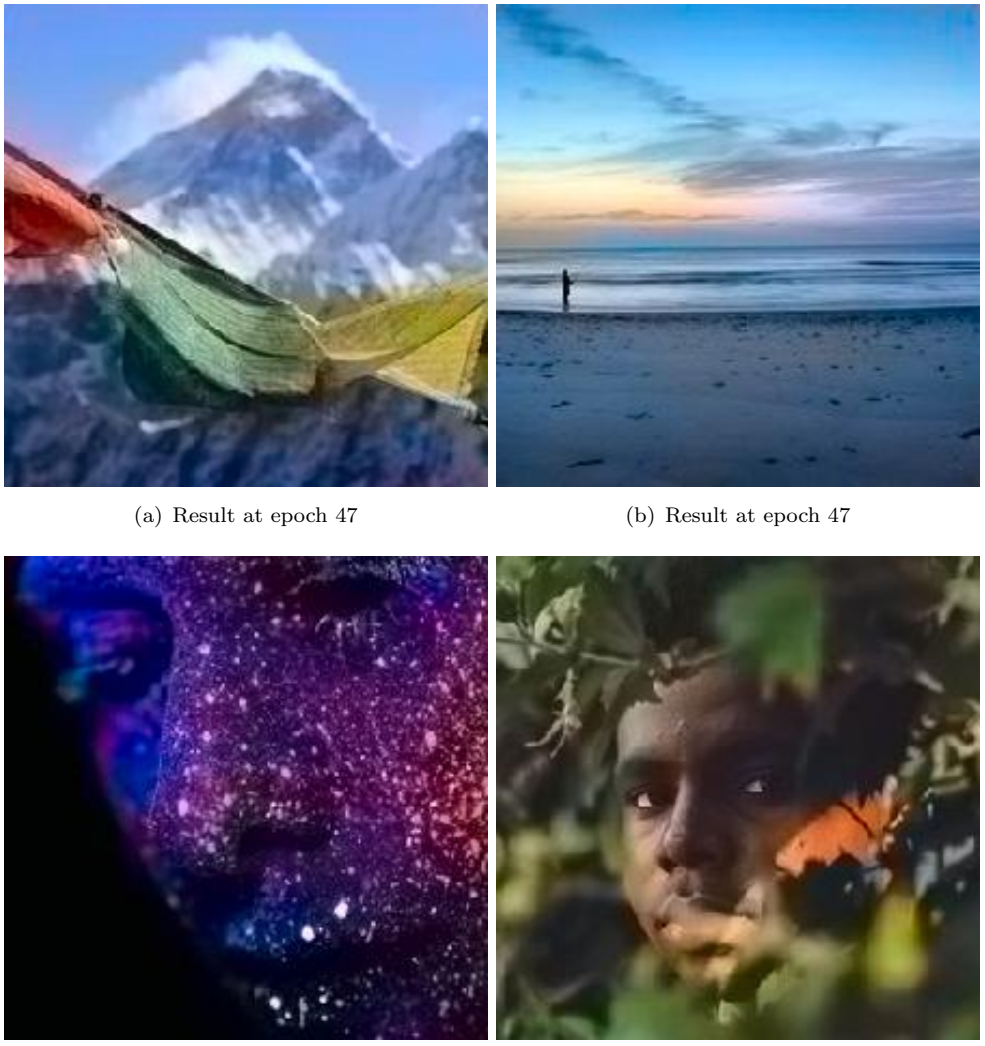


FIGURE 3.6: Architecture of the generator with transfer learning.

image embeddings. The process in training the model is very similar to the previous method explained, however the input images are rescaled to a size of 224x224, to meet the specifications of the pre-trained ResNet network. The RGB images are then transformed to the CIELab color space and are again separated into two vectors $\hat{X}$ which contains the lightness channel and $\hat{Y}$ which consist of the a* and b* channels. Before feeding the generator the lightness images, the $\hat{X}$ vector is stack with itself to obtain a three channel image in order to satisfy the ResNet dimension requirements. The generator then predicts the vector $\hat{Y}'$ which then is concatenated with the $\hat{Y}$ and is then given to the "PatchGan" discriminator as input. The discriminator then tries to classify if each N X N patch in an image is real or fake [22]. Finally the generator is then trained with the loss of the discriminator in order to improve its performance. Figure 3.7 show some results gathered from training. It is possible to observe that the resulting images look very realistic and colorful

The model was trained using the unsplash dataset by Wallner E. [29] which consist in 10,000 real life images size 256x256 from the Unsplash website. All the training tasks were carried out using Google Colab on a single Tesla P4 GPU.



(a) Result at epoch 47

(b) Result at epoch 47

(c) Result at epoch 180

(d) Result at epoch 180

FIGURE 3.7: Results from the model with transfer learning

Chapter 4

Results

In order to validate the results, a colorization Turing Test proposed by Zhang et. al. [1] was used. This test consists in showing some participants real and generated colorizations for the same image, and ask them to identify the real colorization. The Turing Test used resulting images from the training dataset. In this test the model was able to fool participants on a 60% whereas ground truth colorizations would achieve 50% on this metric [1]. Figure 4.1 shows the difference between some original images and their prediction counterpart which were used in the Turing Test. It is possible to see that the predictions of color in the images are the same as the original.

Once trained, the network was fed some testing images that the model had never seen. Figure 4.2 shows some examples from testing. The resulting colorizations from testing look a little desaturated in contrast with the original image, and there are small uncolored areas, however the results still look very realistic, especially when the elements to colorize are natural landscapes. Using these colorizations another colorization Turing Test was performed. Figure 4.3 shows some predictions alongside with the original images that were used in the Turing Test. In this case the model was able to fool up to a 45.9% of the participants. As opposed to the training images the color predictions are not the same as the original colors, however, the predicted colors are within range of the possible colorizations for a certain object, as it is visible in the image with the dog in the Subfigure b in Figure 4.3 where the grass colorization could look like grasslands.

(a) Original (b) Generated

(c) Original (d) Generated

(e) Original (f) Generated

FIGURE 4.1: Training images used for the colorization Turing Test

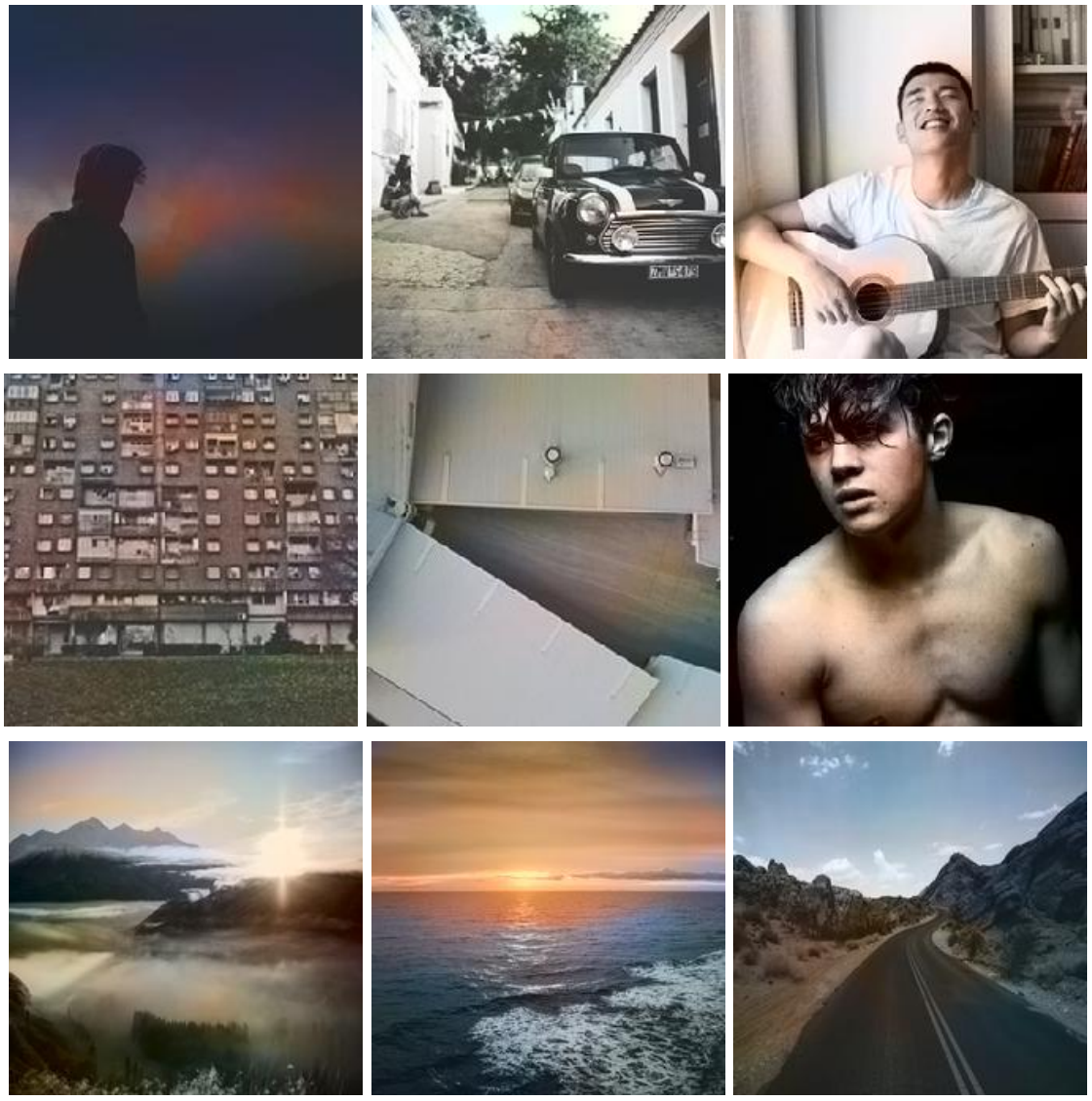
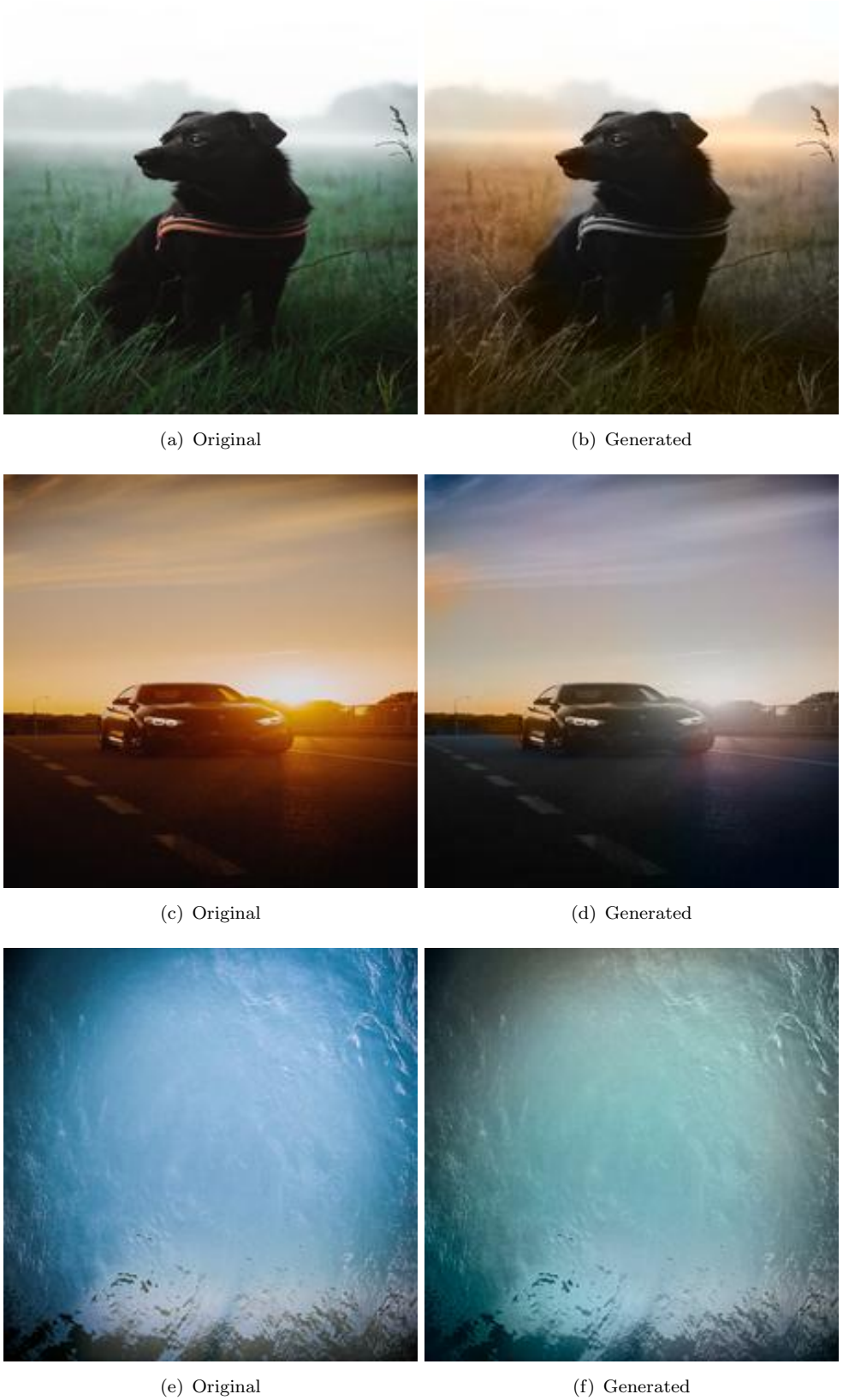


FIGURE 4.2: Results from testing

Nonetheless, there are also predictions that doesn't achieve the same realism as illustrated on Figure 4.4 where it is possible to observe that the generator was not able to properly colorize the images, and some common problems from automatic colorization can be seen such as desaturation, color bleeding and inability to recognize objects, this problems occur mainly on images with a high level content. This could be, partly, due to overfitting given the limited training dataset size.



(a) Original (b) Generated

(c) Original (d) Generated

(e) Original (f) Generated

FIGURE 4.3: Testing images used for the colorization Turing Test

FIGURE 4.4: Examples of some failures during testing where it is possible to observe that color bleeding, objects not being recognized and desaturated results

Chapter 5

Conclusions and Future Work

The neural network with transfer learning achieved image colorization more effectively than the model trained from scratch. It was possible to see the results of both networks at epoch 47, where the one without transfer learning was not able to predict the color in certain areas as opposed to the results using a pre-trained ResNet50, where the images ended up looking pretty realistic. The colorization Turing Test from the test dataset proved that the model was able to fool the human observer up to a 45.9%, whereas ground truth colorizations would achieve a 50% on this metric [1] meaning that the proposed model yields better results than previous architectures. However, due to the small size of the dataset, the network was not able to colorize some images properly causing problems that are common in automatic colorization, such as incorrect object recognition, desaturation, and color bleeding. Future work would be to train the network using a larger and more diverse dataset in order to prevent overfitting. More future work could also revolve around using a smaller pre-trained network such as ResNet-18 to see how much the model's approximation influences overfitting with the test data to try to make the models predictions more general, alongside with using much stronger regularization methods.

Bibliography

[1] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization.

[2] Heinz K. Henisch and Bridget Ann Henisch. *The painted photograph, 1839-1914 : origins, techniques, aspirations*. University Park, Pa. : Pennsylvania State University Press.

[3] Steven Joseph. Photochrom prints - the photochrom process. Library Catalog: www.loc.gov.

[4] Adrian Pipirigeanu, Vladimir Bochko, and Jussi Parkkinen. A computationally efficient technique for image colorization. In Alain Trémeau, Raimondo Schettini, and Shoji Tominaga, editors, *Computational Color Imaging*, volume 5646, pages 120–129. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.

[5] Przemyslaw Lagodzinski and Bogdan Smolka. Colorization of medical images. page 4.

[6] Machine learning in medical imaging: 8th international workshop, MLMI 2017, held in conjunction with MICCAI 2017, quebec city, QC, canada, september 10, 2017, proceedings.

[7] Muhammad Usman Ghani Khan, Yoshihiko Gotoh, and Nudrat Nida. Medical image colorization for better visualization and segmentation. In María Valdés Hernández and Víctor González-Castro, editors, *Medical Image Understanding and Analysis*, volume 723, pages 571–580. Springer International Publishing. Series Title: Communications in Computer and Information Science.

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

[9] Tung Nguyen, Kazuki Mori, and Ruck Thawonmas. Image colorization using a deep convolutional neural network. page 2.

[10] Upasana Bisht and Tushar Patnaik. Overview of automatic image colorization schemes. 03(10):5.

[11] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. page 6.

[12] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S. Lin, Tianhe Yu, and Alexei A. Efros. Real-time user-guided image colorization with learned deep priors.

[13] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 327–340. Association for Computing Machinery.

[14] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, volume 9908, pages 577–593. Springer International Publishing. Series Title: Lecture Notes in Computer Science.

[15] Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller. Transferring color to greyscale images. page 4.

[16] Mingming He, Dongdong Chen, Jing Liao, Pedro V. Sander, and Lu Yuan. Deep exemplar-based colorization.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

[18] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 415–423. IEEE Computer Society.

[19] Federico Baldassarre, Diego González Morín, and Lucas Rodés-Guirao. Deep koalarization: Image colorization using CNNs and inception-ResNet-v2. version: 1.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

[21] Kamyar Nazeri, Eric Ng, and Mehran Ebrahimi. Image colorization with generative adversarial networks. 10945:85–94.

[22] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976. IEEE.

[23] Adobe. The lab color mode in photoshop.

[24] Gernot Hoffmann. CIELab color space.

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

[26] Priya Dwivedi. Understanding and coding a ResNet in keras. Library Catalog: towardsdatascience.com.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition.

[28] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks.

[29] Emil Wallner. emilwallner/datasets/colornet: 10k images from unsplash, 256x256, mostly portraits. | FloydHub.