Ministry of Higher Education and Research
Higher School of Computer Science 08 May 1945 - Sidi Bel Abbes

Second Year Second Cycle - Artificial Intelligence and Data Science

---

Project Report

AUTOMATIC IMAGE COLORIZATION USING DEEP LEARNING

Students:

- Abdelnour FELLAH
- Abderrahmene BENOUNENE
- Adel Abdelkader MOKADEM
- Meriem MEKKI
- Yacine Lazreg BENYAMINA

Supervisor:

Dr. Nassima DIF

Release date: May 9, 2024

Table of contents

List of Figures

1 Introduction

2 Introduction

In recent years, the integration of deep learning techniques into computer vision has propelled significant advancements in image processing tasks, notably automatic image colorization. This once labor-intensive process, reliant on substantial human input and manual coding, has evolved into an end-to-end operation empowered by AI and deep learning. This project explores two distinct approaches to automatic image colorization: Conditional Generative Adversarial Networks (cGANs) based on the influential Pix2Pix paper, enhanced by a simple modification yielding notable improvements, and the DDColor method proposed in "DDColor: Towards Photo-Realistic Image Colorization via Dual Decoders". Effective solutions to the colorization challenge find broad applications across domains such as digital art, historical image restoration, and visual content enhancement. This report conducts a comparative analysis of these two methodologies, examining their techniques, implementation strategies, and performance in automatic color generation tasks. Leveraging diverse datasets and deep learning frameworks, we explore the strengths and limitations of ddcoolor (2) and pix2pix (1) in producing realistic colorizations from grayscale input images. The complete project is available on GitHub at the following link: https://github.com/Devnetly/Automatic-Image-Colorization.

3 Colorization problem

The colorization problem in image processing involves transforming grayscale or monochromatic images into their corresponding colored versions. This task is challenging due to the complexity of capturing realistic color information from grayscale inputs, as it requires understanding and replicating the intricate relationships between different image regions and objects. Automatic colorization techniques aim to overcome this challenge by leveraging deep learning models and image-to-image translation frameworks. These approaches learn from datasets of paired grayscale and color images, enabling them to infer color information based on contextual cues, semantic features, and global/local image structures. The ultimate goal of colorization algorithms is to produce visually appealing and natural-looking colorized images that preserve the essence and realism of the original scenes or subjects.

3.1 RGB vs L*a*b

In the context of automatic color generation using deep learning, the choice between RGB and Lab* color spaces carries significant implications for the complexity and effectiveness of colorization algorithms.

RGB color space, a familiar representation in digital imaging, encodes color information directly through three channels: Red, Green, and Blue. Each pixel in an image is represented by three values, typically ranging from 0 to 255 for each channel in an 8-bit image. While RGB is straightforward and intuitive, it poses challenges for colorization tasks due to the inherent complexities of predicting three color channels simultaneously.

Contrastingly, the Lab* color space separates color information into perceptually meaningful components: Lightness (L), and two chromatic channels, a* (green-red) and b* (blue-yellow). The Lightness channel (L) represents the brightness of the pixel, while the a* and b* channels encode color variations along specific axes. Importantly, Lab* is designed to mimic human perception of color, offering a more perceptually uniform color representation compared to RGB.

From a computational standpoint, the complexity of colorization tasks differs significantly between RGB and Lab* color spaces. In RGB colorization, predicting three color channels (R, G, B) for each pixel involves handling a larger search space with potentially millions of color combinations. For example, with 256 possible values per channel in an 8-bit image, RGB colorization entails predicting from $256\hat{3}$ (over 16 million) color combinations, leading to computational overhead and potential instability in the colorization process.

On the other hand, Lab* colorization simplifies the task by separating luminance (L) from chromaticity (a*, b*). Given a grayscale input (L channel), the model needs to predict only two chromatic channels, reducing the computational complexity and improving the stability and accuracy of colorization algorithms. This advantage becomes particularly pronounced when dealing with deep learning models, as the reduced search space in Lab* facilitates more efficient training and inference.

4 Conditional Generative Adversarial Networks (cGANs)

Our first approach revolves around leveraging the principles outlined in the "Image-to-Image Translation with Conditional Adversarial Networks" paper, commonly known as pix2pix. This paper presents a comprehensive framework for various image-to-image tasks within deep learning, including the challenging task of colorization.
Pix2pix introduces a dual-loss strategy comprising the L1 loss, framing the problem as a regression task, and an adversarial loss (GAN), which enhances the model's ability to generate realistic outputs in an unsupervised manner by evaluating their "realness."
In this report, we begin by replicating the methodology outlined in the original pix2pix paper. This involves implementing the dual-loss approach to train our colorization model. However, our exploration doesn't stop there. We introduce novel enhancements to the traditional pix2pix framework, including a redesigned generator model and strategic training tweaks aimed at reducing the required dataset size while achieving remarkable colorization results.

4.1 cGANS

GANs are generative models that learn a mapping from random noise vector z to output image y, in contrast, conditional GANs learn a mapping from observed image x and random noise vector z, to y.
The generator G is trained to produce outputs that cannot be distinguished from "real" images by an adversarially trained discriminator, D, which is trained to do as well as possible at detecting the generator's "fakes".
The cGAN will follow the following loss function:

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

Where G tries to minimize this objective against an ad-versarial D that tries to maximize it:

$$G^* = \arg\min_G \max_D L_{cGAN}(G, D)$$

The earlier loss function helps to produce good-looking colorful images that seem real, but to further help the models and introduce some supervision in our task, This loss function is comibned with L1 Loss (mean absolute error loss) of the predicted colors compared with the actual colors:

$$\mathcal{L}_{L1}(G) = E_{x,y,z}[\|y - G(x, z)\|]$$

If we use L1 loss alone, the model still learns to colorize the images but it will be "conservative" and most of the time uses colors like "gray" or "brown" because when it doubts which color is the best, it takes the average and uses these colors to reduce the L1 loss as much as possible. Also, the L1 Loss is preferred over L2 loss (mean squared error) because it reduces that effect of producing gray-ish images. So, the combined loss function will be:

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

Where λ is a coefficient to balance the contribution of the two losses to the final loss (of course the discriminator loss does not involve the L1 loss)

4.2 Architecture

4.2.1 Generator

The generator adopts the U-Net architecture. It comprises an encoder-decoder structure with skip connections, enabling the preservation of fine details during the colorization process. The encoder downsamples the input image to extract features, while the decoder upsamples these features to generate the colorized output. Skip connections facilitate direct information flow between corresponding encoder and decoder layers, enhancing spatial consistency. Techniques like batch normalization and dropout are employed for stabilization and regularization. The final output is a colorized image of the same dimensions as the input grayscale image. This architecture ensures high-quality colorization results while minimizing artifacts.

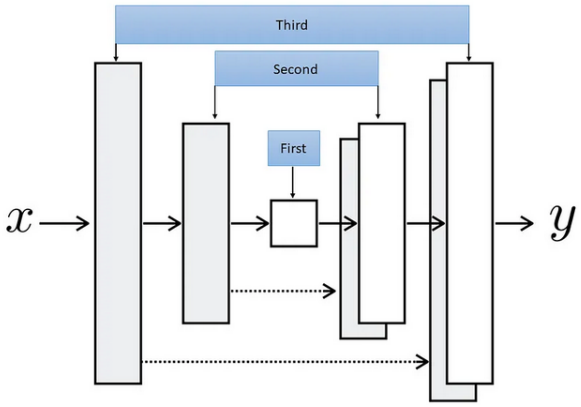

Figure 1: The U-Net architecture

Detailed U-Net architecture

4.2.2 Discriminator

The discriminator follows a PatchGAN architecture, which is designed to provide localized and high-resolution feedback to the generator. It consists of blocks comprising Convolutional (Conv), Batch Normalization (BatchNorm), and LeakyReLU layers. These blocks are stacked to analyze whether an input image is real or fake.

The PatchGAN architecture is structured to output a grid of values, each representing the ”realness” of a specific patch or region within the input image. Unlike a traditional discriminator that outputs a single probability for the entire image, the PatchGAN discriminator assesses multiple patches separately, allowing it to focus on local details and textures.

The use of Convolutional layers helps extract features from the image, while Batch Normalization aids in stabilizing training by normalizing the inputs to each layer. LeakyReLU activation functions introduce non-linearity, enhancing the discriminator's ability to capture complex patterns.

Detailed PatchGAN architecture

4.3 Dataset

The model was trained on a subset of the COCO 2017 dataset which contains 10000 images (8000 were used for training and 2000 for validation), additionally some further pre-processing was involved, since the COCO dataset contains images that are already black and white they were removed, because they are considered as outliers and could elevate the model's error.

4.4 Training & Results

The model was trained on a GTX 1650 with 4GB vRAM for 100 epochs, with a learning rate of 0.0002, Adam optimizer, a batch size of 16 and Data Augmentation. Each epoch took exactly 7 minutes and 28 seconds (some had a second more or less), so in total the training time took 12hours.

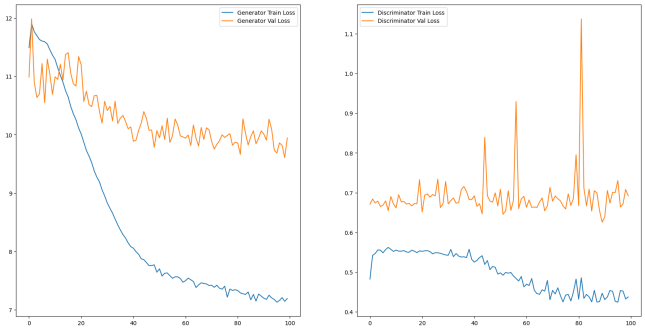

Figure 2: Discriminator And Generator Losses

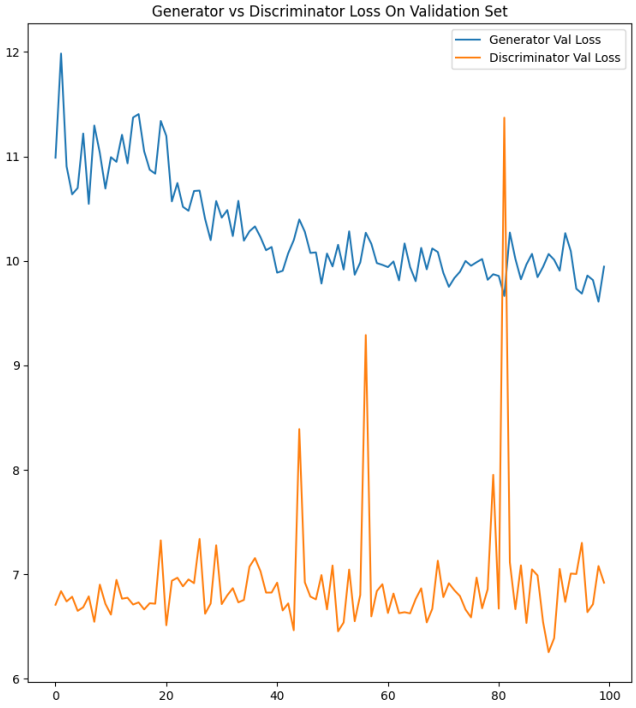

Figure 3: Comparison between Discriminator And Generator Validation Losses

Figure3 shows that the Generator loss is decreasing while the Discriminator loss is not decreasing, which is the objective of the cGAN (the discriminator loss was multiplied by 10 for better visualisation).

4.4.1 Testing on images from test set

Figure 4: 9 random greyscale images from the coco test set

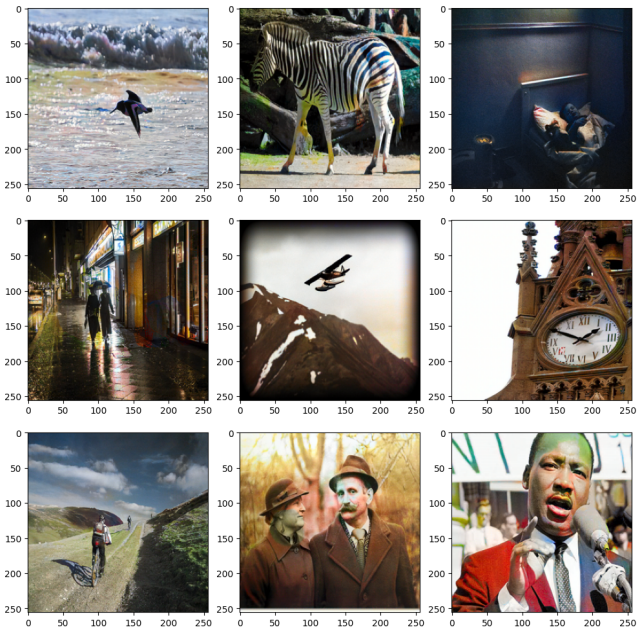

Figure 5: Colorization result of the cGAN

Even though the model seems to be really good at colorizing natural items (sky, grass...etc), it still is conservative with rare objects and people's faces i.e: it just uses a neutral color (like brown) when it does not know what color to use, it also displays some color spillovers and circle-shaped mass of color, and that is due to the the small dataset being used.

4.5 Improving the model

To overcome the problem mentioned earlier the generator was pre-trained separately in a supervised and deterministic manner to avoid the problem of "the blind leading the blind" in the GAN game where neither generator nor discriminator know anything about the task at the beginning of training. The method involves two stages of pre-training for the generator. First, a pre-trained ResNet18 model (on ImageNet dataset) is used as the backbone of the U-Net which encompasses the

down-sampling path, to teach the generator basic features. Then, the entire generator (inclusive of the backbone) is trained specifically for colorization task using the L1 loss function.
By following this approach, the generator is given a head start in learning useful features before diving into the colorization task. This can lead to better and more stable results compared to starting from scratch.

4.6 New training results

The last two layers (GlobalAveragePooling and a Linear layer for the ImageNet classification task) were removed from the pre-trained ResNet18, then it was used as a backbone for the U-Net and trained for 20 epochs. The enitre cGAN is then trained for 40 epochs.

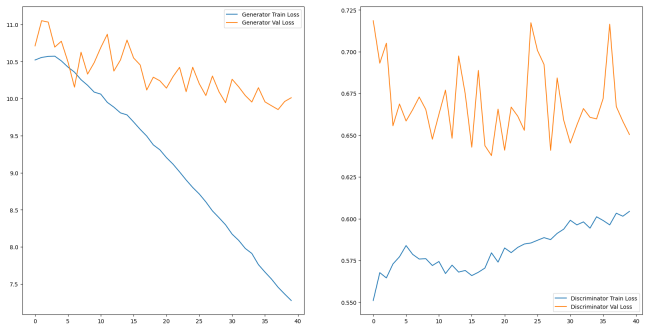

Figure 6: New model's Discriminator And Generator Losses

The improvement is clear just based on the loss graph (Figure 6). The Generator loss is decreasing very fast, while the Discriminator loss is increasing which is the most ideal for a GAN architecture.

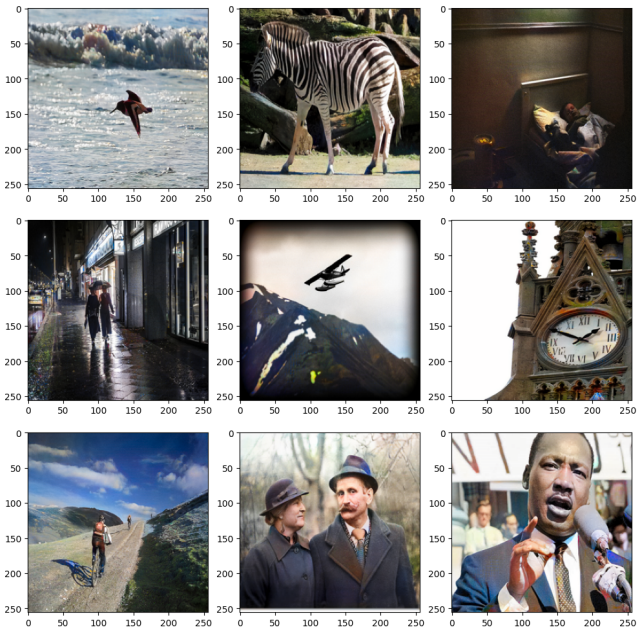

Figure 7: New colorization result of the cGAN

As depicted in Figure 7, the improvement is significant, the new model is not as conservative as it used to be and it can color rare objects and people's faces, it can even correctly predict skin color (as seen in bottom right image).
Comparing the results of the same images used earlier on the non-pretrained generators there is massive improvements, however it still suffers from color bleeding in some parts and it still does not correctly color some parts. To address this issue a new architecture was used, which is the DDColor discussed in the next section.

5 DDColor

5.1 Definition

DDColor is a work from the ICCV 2023 Paper "DDColor: Towards Photo-Realistic Image Colorization via Dual Decoders". It is an end-to-end method with dual decoders for image colorization which achieves superior performance to existing state-of-the-art works. It alleviates color bleeding and improves the colorization of complex contexts and small objects significantly.

5.2 Method

Given a grayscale input image, the DDColor model predicts the two missing color channels ŷAB, where the L, AB channels represent the luminance and chrominance in CIELAB color space, respectively.

It first extracts features using a backbone network, which are then input to a pixel decoder to restore the spatial structure of the image. Concurrently, a color decoder performs color queries on visual features of different scales, learning semantic aware color representations. Then a fusion module combines the outputs of both decoders to produce a color channel output ŷAB. Finally, It concatenates ŷAB and the xL along the channel dimension to obtain the final colorization result ŷ.

5.3 Architecture

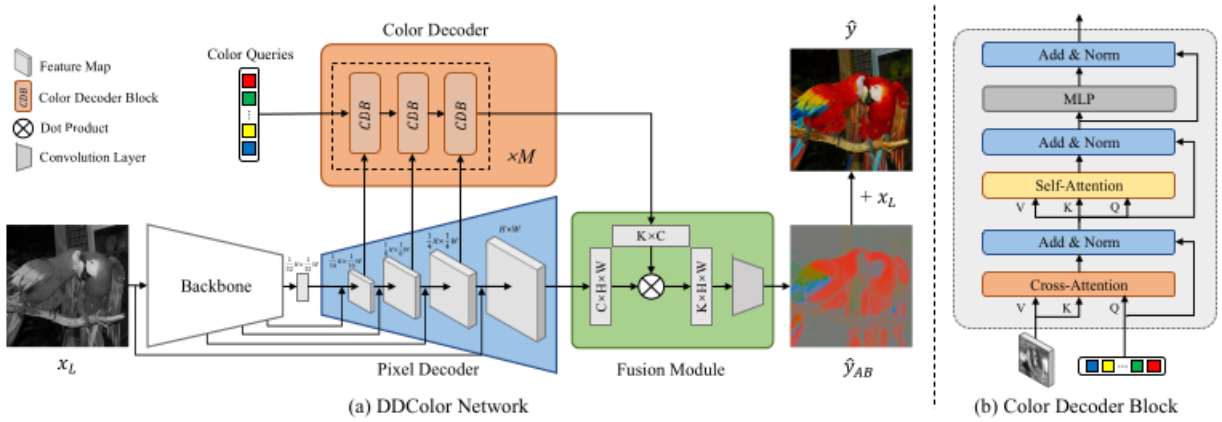

Figure 8: DDColor network architecture

5.3.1 Encoder

A backbone network is utilized as the encoder to extract high-level semantic information from grayscale images. The backbone network is designed to extract image semantic embedding, which is crucial for colorization. ConvNeXt is chosen as the backbone network.

5.3.2 Pixel Decoder

The pixel decoder is composed of four stages that gradually expand the image resolution. Each stage includes an upsampling layer and a shortcut layer.

5.3.3 Color Decoder

The color decoder is composed of a stack of blocks, with each block receiving visual features and color queries as input. The color decoder block (CDB) is designed based on a modified transformer decoder.

5.3.4 Fusion Module

The fusion module is a lightweight module that combines the outputs of the pixel decoder and the color decoder to generate a color result.

5.4 Loss Functions

the following four losses are adopted for training:

5.4.1 Pixel loss

The pixel loss is the distance between the colorized image $\hat{y}$ and the ground truth image y, which provides pixel-level supervision and encourages the generator to produce outputs that are similar to the real image.

5.4.2 Perceptual loss

To ensure that the generated image $\hat{y}$ is semantically reasonable, a perceptual loss is used to minimize the semantic difference between it and the real image y. This is accomplished using a pre-trained VGG16 to extract features from both images.

5.4.3 Adversarial loss

A PatchGAN discriminator is added to tell apart predicted results and real images, pushing the generator to generate indistinguishable images.

5.4.4 Colorfulness loss

A new colorfulness loss was introduced, inspired by the colorfulness score. This loss encourages the model to generate more colorful and visually pleasing images.

5.4.5 Objective formula

Each of the losses mentioned above has a certain weight, here is the full objective formula:

$$\mathrm{L}_\theta = \lambda_{pix}\mathcal{L}_{pix} + \lambda_{per}\mathcal{L}_{per} + \lambda_{adv}\mathcal{L}_{adv} + \lambda_{col}\mathcal{L}_{col}$$

5.4.6 Training

The original paper utilized the AdamW optimizer with specific parameter configurations: 1 = 0.9, 2 = 0.99, and weight decay = 0.01. The initial learning rate was set to 1e4. For the loss terms, the weights were assigned as follows: pix = 0.1, per = 5.0, adv = 1.0, and col = 0.5. ConvNeXt-L served as the backbone network. The entire network underwent end-to-end self-supervised training for 400,000 iterations, employing a batch size of 16. Learning rate decay occurred by a factor of 0.5 at 80,000 iterations, with subsequent decays every 40,000 iterations thereafter.

During our training endeavors, we faced challenges related to hardware limitations differing from those in the original paper. While the original setup benefited from 4 Tesla V100 GPUs, exceeding our current resources, we aimed to adapt by leveraging Kaggle's GPU offering, particularly the Tesla P100. However, the memory requirements of the project surpassed our available resources. In response, we explored various alternative strategies:

- Reduction of input and output image dimensions from 3x256x256 to 3x32x32.
- Implementation of Convnext-T instead of Convnext-L.
- Decrease in the number of heads within both multi-head self-attention and multi-head cross-attention, from 8 to 2.
- Reduction of embedding size from 100 to 50.
- Reduction of batch size from 16 to 1.
- Adoption of a single multiscale color decoder group in place of three.
- Adoption of a single singlescale color decoder group in place of three.

Despite our concerted efforts to adapt the model for training on Kaggle's infrastructure, which presented the most potent hardware available to us, the outcomes fell short of our expectations. Notably, the downscaled image size of 3x32x32 exacerbated performance issues. We attempted to mitigate this limitation by employing a pretrained upsampler, yet the results remained unsatisfactory.

6 Conclusion

In conclusion, this project has been a comprehensive exploration of automatic image colorization techniques, delving into various methodologies and technologies within the realm of deep learning. Alongside our primary investigation into Pix2Pix and DDColor approaches, we encountered practical challenges, particularly regarding resource limitations when implementing DDColor. However, this obstacle provided invaluable learning opportunities, allowing us to delve into advanced concepts such as transformers, query-based transformers, attention mechanisms, and various GAN types including cGAN and PatchGAN.

Despite initial setbacks, our adaptation and enhancement of the cGAN methodology showcased our ability to innovate and improve existing techniques. Through rigorous experimentation and analysis, we have gained deeper insights into the intricacies of automatic image colorization, laying the groundwork for further advancements in this field.

Overall, this project underscores the importance of continuous exploration and adaptation in the ever-evolving landscape of deep learning and computer vision. Our journey not only contributes to the body of knowledge in automatic image colorization but also serves as a testament to our commitment to innovation and mastery in the field of AI and machine learning.

References

[1] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros, *Image-to-Image Translation with Conditional Adversarial Networks*, CVPR, 2017.

[2] Xiaoyang Kang, Tao Yang, Wenqi Ouyang, Peiran Ren, Lingzhi Li, Xuansong Xie, *DDColor: Towards Photo-Realistic Image Colorization via Dual Decoders*, Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 328-338.