Assignment 3
Landscape colorization

194.077 Applied Deep Learning – WS 2023/24

Grégoire de LAMBERTYE (12202211)

December. 2023

Table of contents

I. Introduction

This project aims to use convolutional neural network to colorize landscape images.

Image colorization is a popular topic in the field of image processing, and despite the substantial amount of effort invested in research, there always seems to be room for improvement.

The paper *Colorful Image Colorization* from Zhang et al [1] showed a great improvement in image colorization using a clever idea of color categorization and re-balancing classes. It inspired a lot Chen et al for the paper *Image Colorization Algorithm Based on Deep Learning* [2] where they improved the neuron network architecture and used different activation functions. In practise they focused on faces colorization what inspired this project with the idea of using deep learning to train a colorization CNN specialized in landscapes.

II. Method

1. Model

The usual metric while colorizing image is to use the Euclidean distance between a predicted color and its ground truth but this usually generated greyish or sepia colors. To avoid this effect an idea is to classify the colors and deal with the task like a classification problem.

Color theory

The RGB (red, green and blue) color space is usually used to encode a pixel color but for this task we will use the Lab* CIE 1976 (CIELAB) color space. The CIELAB space is composed of 3 channels: L* for lightness and a* and b* for the color components. The a* channel encodes the green-red component and the b* channel encodes the blue-yellow component. When the conversion is possible, the transformation from CIELAB to RGB is done without loss. Since our network takes as input a grayscaled image, corresponding to the L channel we will stay in this color space and predict the a* and b* channels.

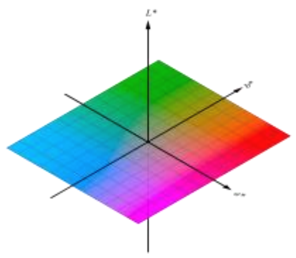


Figure 1: CIELAB color space

Color classes

As explain in the introduction, our goal is to turn the colorization task into a classification task. The classes we try to predict are the following: a bin has size 0,066 x 0,066 when a* and b* are in the range [0, 1]. The shapes comes from the empirical observation of the colors in the train set once mapped to the closest bin in a 15x15 grid, they all fall into one of these 104 classes.

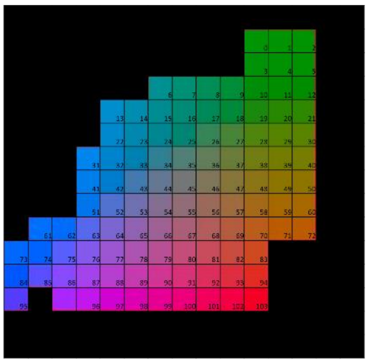


Figure 2: Color class for L=50

At first the colors were mapped into a 10x10 grid but only a few of the 100 classes were really used and the actual palette of color was very limited. The initial plan was also to run to application locally and my personal setup couldn't handle the 315 colors used in the papers referenced in the introduction. This trade off was selected as it didn't imply much more space or weight to train and the result look acceptable as see in the figure 3. The truth seen by the model during training is the discretized images.

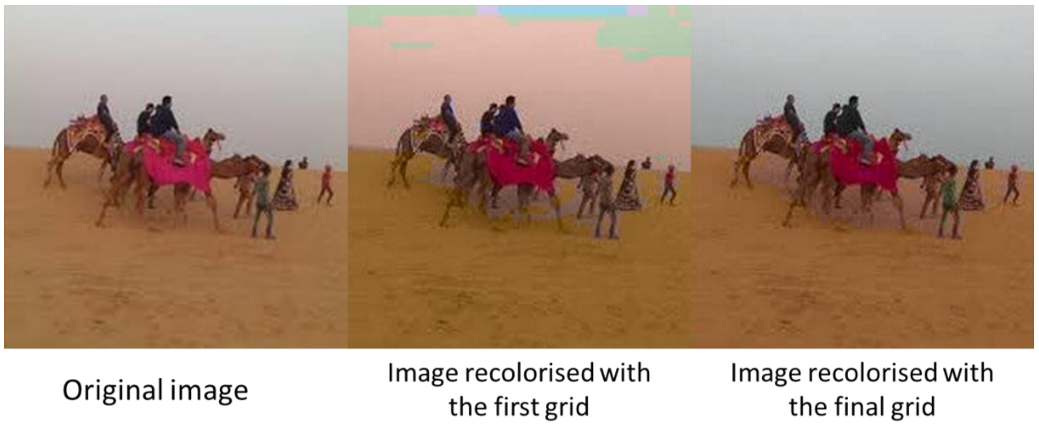

Original image
Image recolorised with the first grid
Image recolorised with the final grid

Figure 3: Image recolorization

CNN architecture

The final architecture is a U-net with 4 layers. The input is a grayscaled image and the output is a 256x256x104 tensor encoding the probability of each pixel to be from the class. The particularity of a U-net work is to pass data from a layer not only to the next one but also to some distant layer. This architecture enables the model to stress early detected pattern i.e. more detailed patterns.

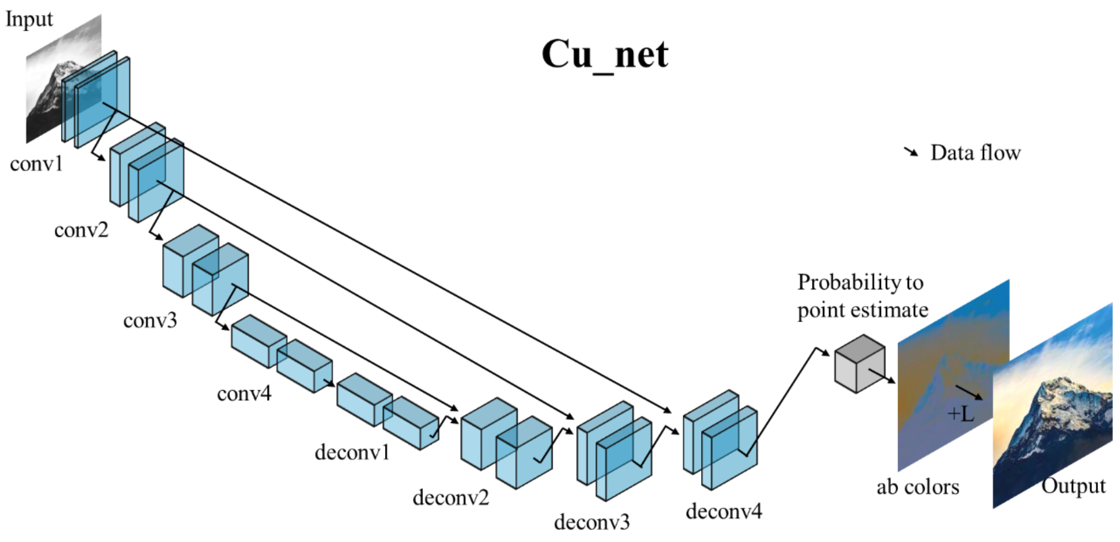


Figure 4: CNN model

One blue bloc in the scheme is composed of 2 convolutional layers. The first with a kernel size = 3, stride=1 and padding=1. And the second with a kernel size = 4. This enables the cropping operation usually used in a Cu architecture. The table one summarises the input and output size of the data passed from a layer to another when the input is a tensor of size 256x256. After each layer a sigmoid activation function and a batch normalization is performed. The choice of sigmoid functions was motivated by Wang, N et al paper were they show its dominance over other common activation function for a similar architecture.

Layer	Dimensions (in)	Dimensions (out)
conv1	1x256x256	64x128x128
conv2	64x128x128	128x64x64
conv3	128x64x64	256x32x32
conv4	256x32x32	512x16x16
deconv1	512x16x16	256x32x32
deconv2	512x32x32	128x64x64
deconv3	256x64x64	64x128x128
deconv4	128x128x128	104x256x256

Loss function

Our task is a classification problem, and we will therefore use a weighted cross entropy loss function. Weight are discussed below.

Class rebalancing

In landscapes some colors are much more present than others. Blue for sky and sea, green for trees and gras ... If we want to predict colourful images it could be interesting to weight our loss function to consider this color distribution. The following heat map shows the color distribution (among our classes) of around 30k images of the 3rd dataset. Colors in the middle of the ab map correspond to the brown / least vibrant colors. As we can see in the figure 5, these colors are way more frequent than other.

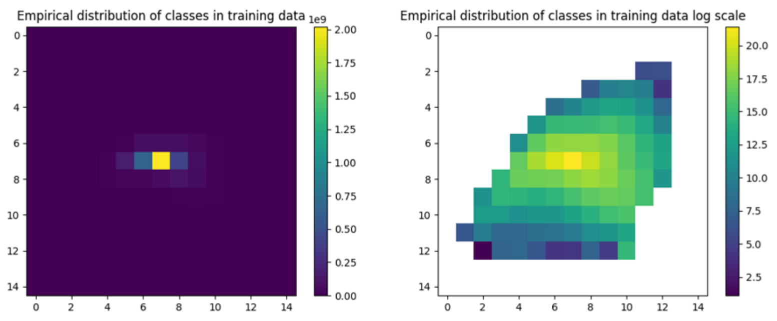
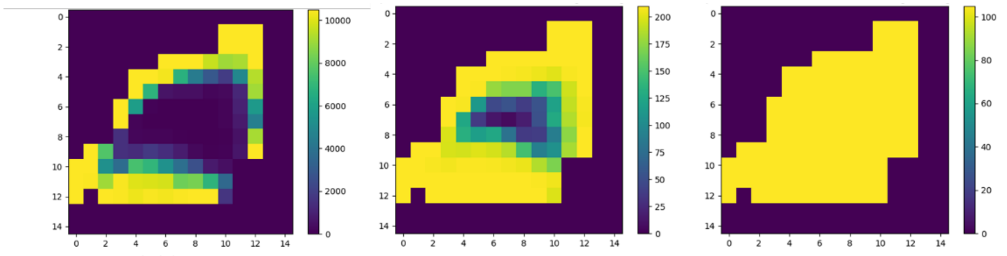


Figure 5: Empirical color distribution

To transform the empirical distribution to a weight for the loss function we can use the following formula (took from Zang et al paper):

$$w_q = ((1-\lambda)p + \frac{\lambda}{Q})^{-1}$$

Where p is the empirical distribution and q is a color and Q the number of classes. The parameter lambda controls the influence of the empirical distribution on the final weight.



0,01 0,5 1

Figure 6: Lambda influence on the final weight

Probability to point estimate.

The output of the network is a 256x256x104 tensor encoding the probability of each pixel to be from the class. We can imagine different mapping to get to our final colorization. Taking the most probable class for each pixel is the most obvious but it can lead to a colorization with a lot of noise and less coherence. On the opposite, taking the average of the probability for each class will lead to a more coherent colorization but with less contrast and a result similar to what a model that use euclidean loss would output. To find a good balance between coherence and contrast we can use a temperature parameter. The temperature parameter is a scalar that will modify the max probability influence. The higher the temperature the more the max probability will be important.

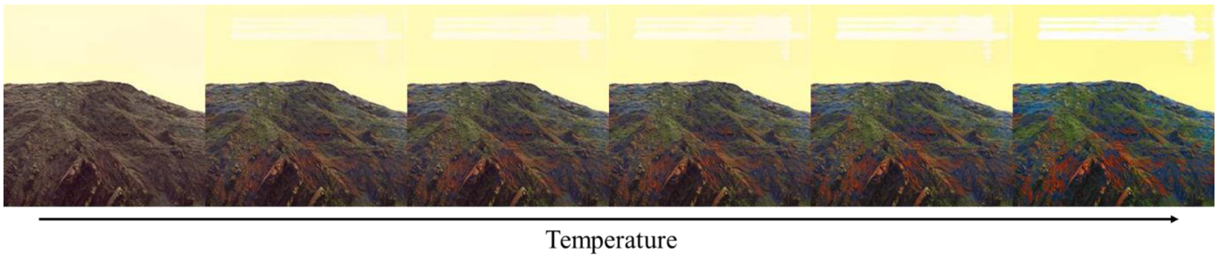


Figure 7: Temperature influence

With the figure 7, we see that with a low temperature we obtain a very coherent image but less interesting. With a high temperature we obtain a more contrasted image but with less coherence as we can clearly see in the sky.

2. Datasets

During this project, 3 datasets were considered:

1. 'Landscape Pictures from Rougetet Arnaud' dataset, available on Kaggle here. This comprehensive dataset comprises 7 folders, totaling 4300 landscape pictures. These folders encompass various landscapes such as general scenery, mountains, deserts, seas, beaches, islands, and specific scenes from Japan.
2. 'Landscape Recognition | Image Dataset | 12k Images', available on Kaggle here. This dataset comprises 12,000 landscape pictures, divided into 5 categories: Coast, Desert, Forests, Glaciers and Mountains.
3. 'LHQ dataset', available on Yandex Disk here. This dataset comprises 90'000 images of (mostly) landscapes. I used the preprocessed set LHQ1024 resized to 256x256 with Lanczos interpolation.

The main difference between the datasets is the number of images they offer. For the 2 first dataset the images have been cropped to a square and reduce to 256x256 pixels. The last dataset was already well formatted for the task.

3. Realisation

During this project I build step by step my final model and trained lots of model, I kept 3 of them, all based on the architecture present above. They all used pytorch CrossEntropyLoss as loss function and Adam as optimizer. Due to time constraints the models couldn't be trained as long as I would have liked.

Model_1: The first model is trained on the 2nd dataset (10k images) without penalty (lambda = 1). Initial learning rate was set to 0.015 (no scheduler). The 15th epoch was selected as the best model. To generate the colorization I used a temperature of 2.

Model_2: The second model is trained on the 3rd dataset (30k/60k images) with a penalty (lambda = 0.5). Initial learning rate was set to 0.015 and set to 0.005 after the 5th epoch. The training was stopped after 8 epochs. To generate the colorization, I used a temperature of 0.25.

Model_3: The last model is similar to the 2nd, but the penalty was lowered (lambda = 0.8). Initial learning rate was set to 0.0015 and stayed the same over the 3 iterations. To generate the colorization, I used a temperature of 2.

During training many difficulties were encountered. Due to my private setup (NVIDIA GEFORCE GTX 950M _ 2 GB VRAM) I opt for a training in the cloud using Google Colab (GPU T4 15GB RAM) but the execution there seemed unstable.

After some experiment it turned out the number of images was crucial, more images gave from the first iteration way better results. This explains the 3 dataset I used, the last one contains 90k images this would have taken too much time on my own setup. I used Google Colab to handle it and I placed my data on google drive. Unfortunately google drive doesn't accept files bigger than 3GB therefore I had to split my training data in 2 sets. But the execution on colab seems to be unstable and from an epoch to the other it could apparently use only one of the sets. The training also broke very often which forces me to add regular save points during the epoch and cost me a lot of time.

The major mistake I did was also not to track my experiments properly. I added new brick step by step but also changed the datasets making it hard to compare one setup to another properly. There is also a lot of hyper parameters tuning I could have carried better (lambda, temperature …). I relied on the referenced paper for the majority of the architecture choices (activation function, structure, kernel, padding …) but those parameters would have deserved experimentation as well.

III. Results

1. Evaluation

As discussed in Zhang et al paper [1] there is no perfect metric for our task this is mostly due to the different colors an object or anything can take. We would prefer a model that colorize with coherent color over a model that would partially colorize with the truth color.

Truth Image

Coherent Colorization

Uncoherent Colorization

To evaluate our prediction, we will therefore use the 2 same metrics as Wang, N. et al [2] and a user evaluation.

Euclidean distance: The euclidean distance is the average distance between the predicted color and the real color (For a 256x256 pxl image).

$$Euclidean\ distance\ = \frac{1}{256^2}\sum_{i=0}^{255}\sum_{j=0}^{255}((Y_r - \widehat{Y_r})^2 + (Y_v - \widehat{Y_v})^2 + (Y_b - \widehat{Y_b})^2)^{-1}$$

PSNR: The PSNR is the peak signal-to-noise ratio. It is a measure used to compare the quality of an image with the original image.

$$PSNR\ =\ 10\ log(\frac{255^2}{MSE})$$

User evaluation: I selected 10 images from the LHQ dataset and asked people to compare the output of Zhang et al and Model_1. I selected these images aiming to show that model_1 could be better than Zhang model on some very specific cases. The dominance of Zhang model over my model is obvious.

92 people answered a survey where they had to choose the most realistic image between the prediction of Zhang et al e16 model and Model_1.

2. Results

Model	Euclidean distance		PSNR	
	mean	std	mean	std
Zhang et al	799.74	363.77	-9.16	3.58
model_1	970.18	428.14	-10.97	3.15
model_2	1258.47	377.63	-13.51	2.45
model_3	1032.374	375.67	-11.64	2.88

The user survey can be found in the annexe, it shows that the images obtained with the first model could sometimes beat Zang et al model, but it globally performs worse.

IV. Demo application

To serve this model I developed a small application where user can drag & drop images to colorize. It uses the 3 models presented earlier and the colorized and grayscaled images are then displayed and saved in an image folder. The application was generated using Pyinstaller and might not be optimized. The exe file is 2 GB and need lot of space on a device to run (more than 5GB).

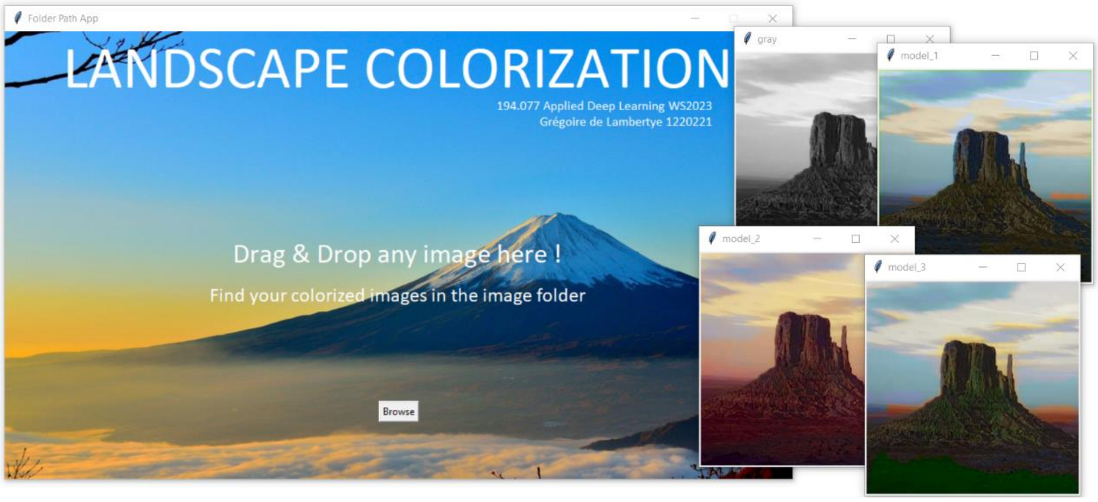

Folder Path App
LANDSCAPE COLORIZATION
194.077 Applied Deep Learning WS2023
Grégoire de Lambertye 1220221
Drag & Drop any image here !
Find your colorized images in the image folder
Browse
gray
model_1
model_2
model_3

Figure 8: Demo application

V. Work-breakdown structure

Initially this project was supposed to be handled in 35hours, I was really interested in it and decided to invest more time to obtain a result I could be proud of. It turned out that my result didn't match my expectations, but I learned a lot both in terms of good practise and technical knowledge. I also struggled a lot because of a lack of experience and hardware issues.

Here is the approximate time I spend on each part of this project:

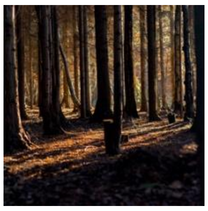
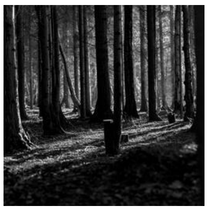
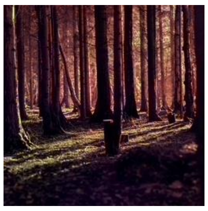
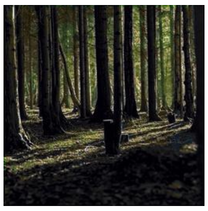
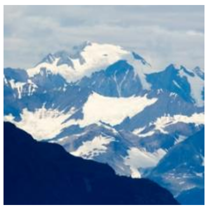
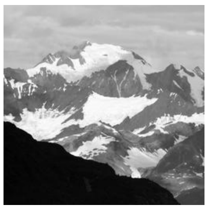
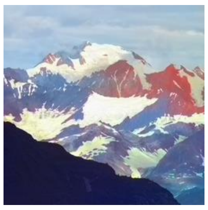
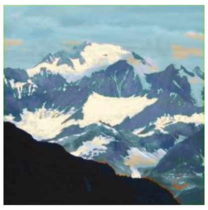
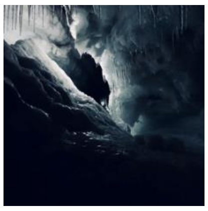
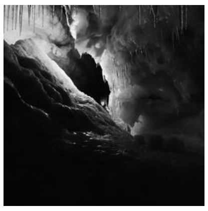
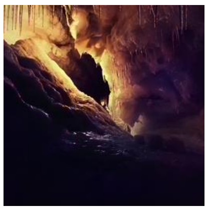
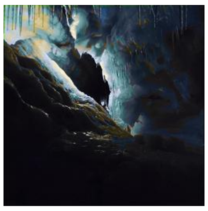
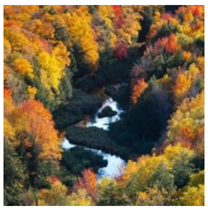
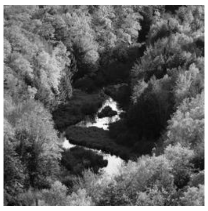
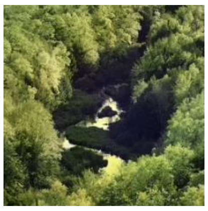
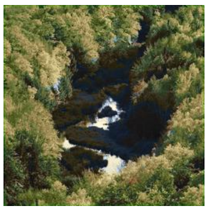
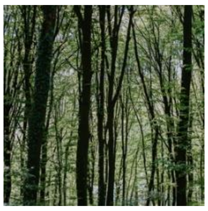
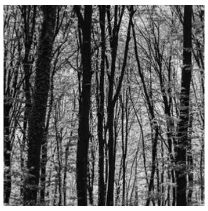
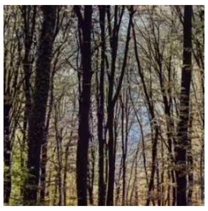
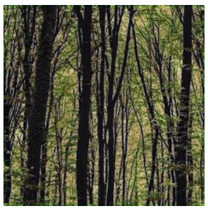
- Gathering information, paper lecture (4h)
- Gathering and preparing data (6h)
- Development of a first version of the CNN including training and environment installation (cuda) (20h)
- Workaround to improve the model (colors classes, penalty, temperature...) (80h)
- Evaluation of the final CNN including a user comparison to Zhang's CNN output (8h)
- Redaction of a report and video (5h)

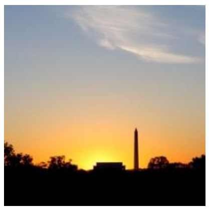
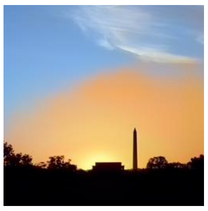
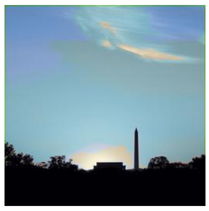
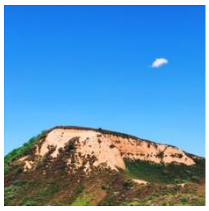
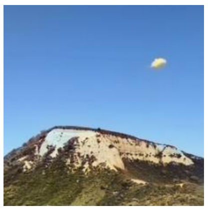
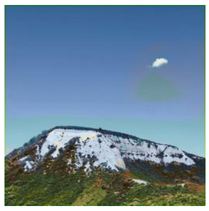
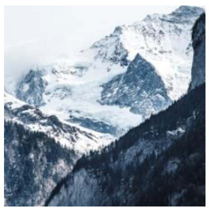
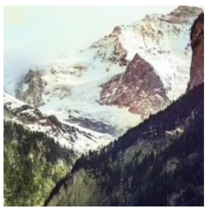
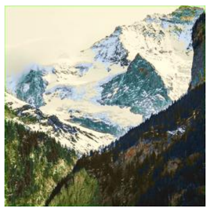
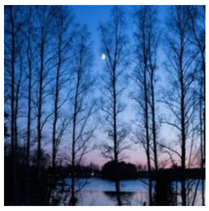
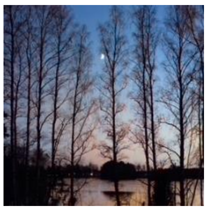
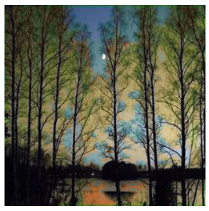
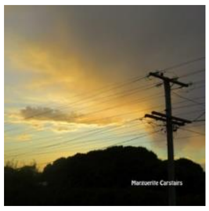
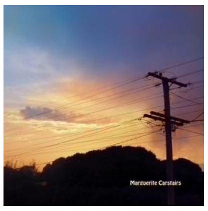
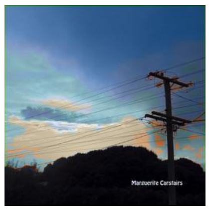
VI. Annexe

1. Survey results

I selected 10 images from the LHQ dataset and asked people to compare the output of Zhang et al and Model_1. I selected these images aiming to show that model_1 could be better than Zhang model on some very specific cases. The dominance of Zhang model over my model is obvious.

92 people answered a survey where they had to choose the most realistic image between the prediction of Zhang et al e16 model and Model_1. The results are the following:

Ground truth	**Grayscale**	**Zhang**	**Model_1**	**Score (% in favor of Model_1)**
				95%
				84%
				65%
				51%
				44%

Ground truth	Grayscale	Zhang	Model_1	Score (% in favor of Model_1)
				42%
				26%
				14%
				1%
				1%

2. References

[1] Zhang, R., Isola, P., Efros, A.A. (2016). Colorful Image Colorization. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science(), vol 9907. Springer, Cham. https://doi.org/10.1007/978-3-319-46487-9_40

[2] Wang, N.; Chen, G.-D.; Tian, Y. Image Colorization Algorithm Based on Deep Learning. Symmetry 2022, 14, 2295. https://doi.org/10.3390/sym14112295

Dataset 1: Rougetet Arnaud, Landscape Pictures (2020), https://www.kaggle.com/datasets/arnaud58/landscape-pictures

Dataset 2: DEEPNETS, Landscape Recognition | Image Dataset | 12k Images', available on Kaggle https://www.kaggle.com/datasets/utkarshsaxenadn/landscape-recognition-image-dataset-12k-images

Dataset 3: Skorokhodov, Ivan and Sotnikov, Grigorii and Elhoseiny, Mohamed, LHQ dataset, https://disk.yandex.ru/d/HPEEntpLv8homg, original paper: Skorokhodov, Ivan and Sotnikov, Grigorii and Elhoseiny, Mohamed. Aligning Latent and Image Spaces to Connect the Unconnectable (2021) https://arxiv.org/abs/2104.06954