Final Project - Vision & Cognitive Systems

Image Colorization

Alberto Sinigaglia
alberto.sinigaglia
ID: 2044712

Beatrice Sofia Bertipaglia
beatricesofia.bertipaglia
ID: 2054766

Chiara Colato
chiara.colato
ID: 2062670

February 15, 2023

Abstract

In this work we compare multiple architectures and techniques for the colorization problem in computer vision, which consists of transforming images from their greyscale version back to their colorized version.
This task is notorious tough due to the fact that the model has to learn the association color-shape, and theoretically, given a greyscale image, there are multiple possible colorization, and we will show how GAN based model outperforms regression based thanks to the fact that they are not impacted by this latter problem.
In order to have a fair comparison, we exploited all the group member computational resources to do a full grid search over the possible hyperparameters for each model.

1. Introduction

In this project we deal with the *Image Colorization* task with which black and white images are colored. In particular we start from a *single channel image* and aim to be able to predict the correspondent colored version of the same, composed by the three color channels. Making automatic this procedure is a challenging topic in computer vision and the main issue is represented by the *multi-modal nature* of the problem: in the same image, indeed, we can have different colors at the same time.
Thanks to the new architectures like the ones proposed by the deep learning field, here we try to make the models learn the colorization process in a end-to-end fashion. In this project we start presenting as baseline an AutoEncoder model in order to deal with the colorization of greyscale images of flowers coming from a Kaggle dataset [8].

2. Dataset

In this project we use the *Flowers dataset* [8], which is composed by 18540 colored images of flowers and plants with dimensions 500×500 px. The entire dataset is firstly divided into *Training set* and *Test set*, with ratios 80% and 20%; then from the first one we extract 25% of the images in order to constitute the *Validation set*. We use them for training the models, choosing the best structures for each of them and finally evaluating them. In this way we obtain the following subdivision:

Set	Size (# Instances)
Training Set	14832
Validation Set	3708
Test Set	3708

2.1. Preprocessing

Due to the fact that the original dimensions comport high computational costs, we proceed with a resizing of the images, bringing them to size 128×128, that allows us to maintain a good quality of the images but at the same time we have the possibility to process a lower dimension data, so we can reach an accessible time of computation.

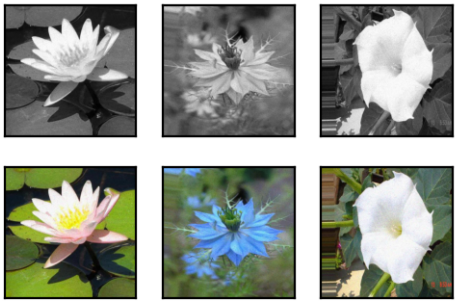

Figure 1. Example of pairs input-output.

Then we proceeded averaging the RGB channels in order to get the greyscale version of them.

3. Methods and Related Works

We try and compare multiple architectures, losses and techniques. Most of them are set-up as a *Pixel-Wise regression problem*, which tries to predict the value for the Red, Green and Blue channels for each pixel [12]. In addition to this, we also compare the regression-based approach with a Conditional GAN approach [11].
All the presented models had their hyperparameters optimized using grid-search to find the best configuration for the current problem.

3.1. Convolutional Auto-Encoders

In order to deal with the colorization task we start using a Convolutional Auto Encoder (CAE). We consider them as the best choice as baseline, in particular because this family of neural networks is able to detect the salient features of an image, and then regenerate the image starting from them.
Autoencoders are originally designed for unsupervised machine learning to learn an efficient representation of an image or other type of data. Autoencoders are composed of two parts, the encoder and the decoder. The encoder, through a series of CNN and down-sampling, learns a reduced dimensional representation of the input data. The decoder then, through the use of CNN and up-sampling, attempts to regenerate the data from the these representations.
Traditionally, autoencoders are used to filter noise, compress data or for dimensional reduction aims. More recently, however, there is an increased use of the decoder to generate novel data, and in this case the autoencoders can be considered as *Generative Models*. Indeed, if a well-trained decoder can generate accurate data from a set of features/representations, maybe it could be able to generate data from representations it has never seen before. That's the case we are dealing with: observing images in gray-scale style, the model should learn the main important features from the data and then, from what it learned, it should be able to generate the correspondent input data in three channels.

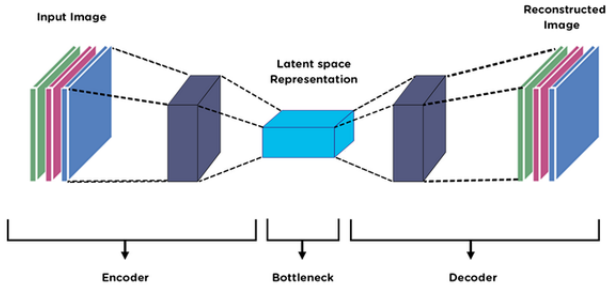

Figure 2. Example of Convolutional Autoencoder structure.

The entire structure is composed by 15 blocks, after each 3 block there is a downsample/upsample layer, and each block is composed by:

- Linear Convolutional Layer;
- Batch Normalization over the channel dimension [14];
- Activation Layer element-wise.

We will employ this structure for all the architecture, as BatchNorm [14] has been shown to be very helpful for training convergence, by reducing the internal covariate shift of the model weights.

3.2. Standard Deep Convolutional Neural Network

The second model that we try is a Deep Neural Network (DNN) with structure similar to the previous one presented with the CAE, in which we remove the *downsampling* and the *upsampling* passages.
We build our Deep Convolutional Neural Network stacking a series of convolutional layers, through which takes the black-and-white image is processed in order to obtain its correspondent colorized version.
This structure is considered the standard approach for solving image classification problems, as reported in [5], and their success is due to their ability to learn patterns, shapes but also colors and to associate them to particular objects. In our study we try to use these abilities for coloring images considering in particular the general presence of correlation between patterns and shapes with colors.

3.3. UNet

After trying with the basic models, in the following step we try to implement different architectures that introduce some innovative aspects.
The first one is the UNet architecture [10], a *Fully Convolutional Network* which acts almost as a CAE, but introducing particular connections between layers called *Residual Connections*: those additional connection allows it to decrease the distance between the input and the output, thus not forcing the network to learn only shared information across the dataset' images as for the CAE, but still allowing to learn deeper features thanks to the "U" shape. Additionally, considering input/output, as reported in the original paper [10], there is a lot of information shared between input and output image, which thanks to the first residual connection is transfer without any problem, and without requiring the network to learn it.
Thus, compared to the CAE, after every down/up-sampling we find a residual connection, and the rest of the network uses the same block as the CAE reported before, using the Conv-BN-Activation structure.

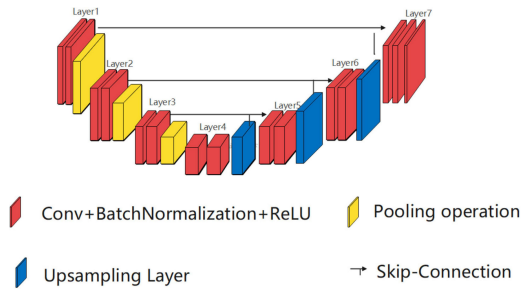

Figure 3. Example of UNet structure

3.4. ResNet

In this paragraph we try to stress the concept of residual connection, creating a deeper network and to observe if we are able to obtain more accurate results.
ResNet [9] constitutes a solution for the construction of deeper models: they introduce a deep residual learning framework in where they explicitly let layers fit a residual mapping.
Practically speaking, the network can be though as a shallow network at each residual connection, where the input of the shallow network is the input and some features extracted by the previous layers. This concept/behaviour of giving as input some latent feature learned by other models, has been exploited already in other architectures [13], however ResNet aims to fully automate this reasoning.
In particular, we exploit the ResNet v2 version of the residual block, which aims to reduce the network and directly connect input and output, achieved by re-arranging the layers order and dividing the convolution in three steps: the first one is depth-wise 1x1 convolution, which aims to reduce the number of channel, the second step the usual spatial convolution, and finally a depth-wise 1x1 convolution to project the tensor back to the original input size.
In our experiment, we use 5, 10, and 20 residual blocks, and we can clearly see how the deeper network outperforms the shallow ones.

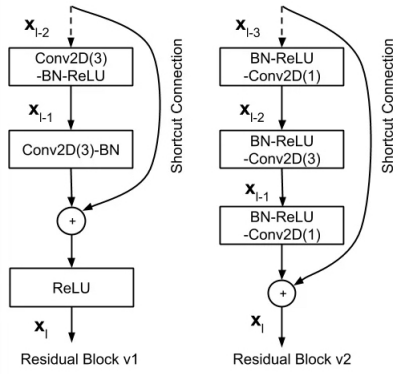

Figure 4. Representation of a single block of the ResNet proposed.

3.5. DenseNet

Finally we stress the concept of residual connection implementing DenseNet [3]. If using ResNet we let each residual block to have information directly from the input and from the previous layers adding them up, in DenseNet we concatenate them, allowing each layer to have full freedom over how to combine those information.
This however poses a problem when going deeper, as the concatenation of the outputs from all the previous layers will create a huge input tensor when stacking many layers. However, it's been shown that the dense residual connections are much powerful then the residual as for ResNet, thus there is no need to go as deep. Furthermore, we exploit the same trick used for ResNet v2 to decrease the number of parameters of the network, by firstly performing a depth-wise 1x1 convolution, and then the spatial one (there is no need for the last 1x1 convolution as the output will be concatenated and not added, thus the channel dimension can be different from the input one).
In our experiments, we use a DenseNet structure with 7 dense blocks on $128 \times 128 \times 64$ input tensor.

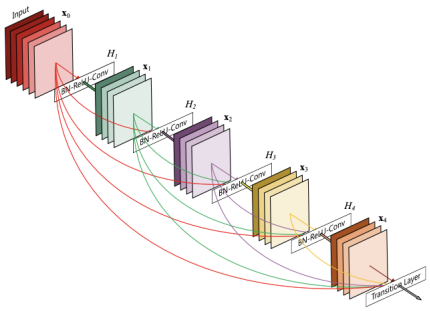

Figure 5. An example of DenseNet with 5 blocks where each block is connected to every successive ones.

3.6. Pix2Pix

The last trial we propose consists into the implementation of an architecture that acts as a Generative Adversarial Network (GAN). We can distinguish two main parts in our model that are called respectively *Generator* and *Discriminator*. The generator, that we call here G, is trained with the aim to produce outputs that cannot be distinguished, from the "real" images belonging to the dataset, by the discriminator D. The D part is adversarially trained for detecting the generated images from the original ones, as well as possible. The training procedure, as reported in [11], can be summarized in the Figure 6.

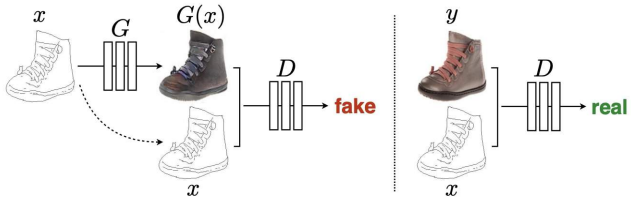

Figure 6. Representation of the training procedure, explained both for the Generator and the Discriminator parts.

In particular, Pix2Pix [11] is based on 2 nets: a UNet and a PatchGAN. The first one consists into just the encoder, which maps from greyscale images to RGB version, while the second consists into the discriminator, which instead of looking at the whole picture, considers only patches of them.
The objective of the network can be described as:

$$G^* = \arg\min_G \max_D \beta \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \quad (1)$$

where in particular we can distinguish the objective function of the *conditional GAN* 2 and the one that corresponds to the *Reconstruction Loss*4, with regression meaning.

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) =& \mathbb{E}_{x,y}[\log D(x, y)] + \\ & \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))]\end{aligned} \quad (2)$$

$$\begin{aligned}\mathcal{L}_{GAN}(G, D) =& \mathbb{E}_{y}[\log D(y)] + \\ & \mathbb{E}_{x,z}[\log(1 - D(G(x, z))]\end{aligned} \quad (3)$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[||y - G(x, z)||_1] \quad (4)$$

The λ and β parameters are used to tune the gradient from the 2 losses ($\lambda = 0$ is fully GAN based, $\beta = 0$ brings us back the regression based method), in order to find a good trade-off between reconstruction and credibility of the output image. Furthermore, the discriminator can be conditioned or not with respect to the input as in the 6. The conditioning does not seem to improve in a significant way the results, in contrast to what reported in the original paper [11]. However in concordance with the original input, the PatchGAN receptive fields actually have shown to impact the resulting outputs.
We also tried to consider an additional loss using 2 discriminators, one conditioned to the input and one not, however the training was too unstable to have any results any better than just using regression.
Additionally, as used in the Cycle-Consistent Adversarial Networks (CycleCAN) [7] paper, the discriminator loss that we used is the $L2$ loss and not the binary cross-entropy as usually, because has a much more stable training, as the gradient produced is much well behaved when the discriminator output is far from the aimed one.

4. Metrics

We considered many possible metrics, but none of them respected the actual image fidelity actually produced by the networks.
This is also faced in the [11] where, for the colorization task, they ends up using human feedback and not a single math-based metric, as none of the existing one truly targets the aim of the task.
Following are some metrics that we considered and tried, but we won't report any result of, as they are far from what our group believes are the true results, due to the problems that are explained in the following sections.
For those reasons, we relied strictly to our personal opinion to evaluate the results, in order to approximate the metrics used in [11].

4.1. Pixel-wise metrics

One possible metric is using the $L1/L2$ distance between target and produced output. However, the regression based methods would naturally outperform the GAN based as they are optimized for that. However, colorization is not about pixel-wise distance, as different humans can easily color an image in different ways based on their experience; for example, in the case of flowers, people from different part of the words might be used to see flowers in different color, thus, if asked to re-color an image, they might be biased to give it different colors.
Furthermore, this is based on the *Maximum Likelihood principle*, which aims to minimize the error, not to give good fidelity. For this reason [11] introduces a GAN, which allows the network to shift to a distribution mode instead of the expected value, thus creating more faithful images colorization.

4.2. PSNR

Peak-Signal-to-Noise-Ratio is another common metric used in those kind of tasks. However, it also relies on a non linear transformation of the *Mean Squared Error* of the reconstruction, which brings the same issues as for the pixel-wise based metrics: it's not capturing any correlation in the image, and neither is considering that it can be recolored in multiple different ways.

4.3. SSIM

Same as for the previous ones, it misses the whole real target of re-colorization, as it uses the targets as ground truth, and it only captures aggregate statistics of the produced images instead of it's fidelity.

4.4. Density based

Thanks to what todays neural networks can achieve, we can also try to estimate the underlying density of the dataset,

for example using a VAE [2].
However, this method does not seem to give any significant difference with respect the different models tested, as it tends to give a lot of importance to the image details (since they also are based on the reconstruction loss), and not to the colorization part, thus the results are neither relevant nor significant.

5. Training

All the models are trained for 50 epochs, and the best one, with respect to validation loss, is used for evaluation. Since there is no validation loss in case of the Pix2Pix model, as it also depends on the discriminator, the last one is used.
Furthermore, all the models are trained using Adam optimizer, which is been shown to be very fast if used with batch-normalization layer and residual connections, since those decreases the conditioning number of the loss, and the optimizer exploits this fully with the estimate of the diagonal Hessian.
For all the models it's been used 0.001 as step-size with the rest of parameters as suggested in [1], and for the Pix2Pix model instead the step-size for the 2 networks is 0.0001 with $\beta_1 = 0.5$ as also used in [11], which will use $\lambda = 0.1$.

6. Cheap training on bigger images

In addition to what has been tried so far, we also tested a difference technique, which exploits the network structure to keep the resolution of the original images, while also maintain the training performance of the models trained in the smaller images.
If recent works [6] exploits super-resolutions networks to upscale the results, in our case all of this is achieved by the elasticity of the convolution and batch normalization operators. In fact, unlike other layers as the Dense one, none of them relies on the input size, if not for the channels number. However, changing height and width of the input images won't impact on the number of parameters of the network. This means that a model trained on 128x128 images have the same number of parameters of a model trained on 256x256 images, however the forward and backward pass of the network will take more time. To overcome this, we randomly crop the input images to 128x128 patches, then we train the network on those patches, and evaluate on the 256x256 images. During training, the network will learn to color flowers/elements individually, which will come from the same distribution of the original dataset, thus, apart from the zero padding used for the convolution, the network will perform the same calculations in the smaller images/patches, and in the original ones.
Additionally, this also acts inevitably as regularization, which is very helpful since we noticed that UNet at the end of the training starts over-fitting.

7. Results and Failures

All the models shares the same types of failures:

1. *Approximate region colorization*. The models are not very accurate in coloring regions of the images, by which we mean that they tend to color areas of the images with some colors that do not respect the actual boundaries of that are, as if it's leaking.

2. *Grayish and greenish prior*. As most the images are centered and mostly green, the model tends to color the external part of the image with green even if some flowers fall in that region. When it's not the case, instead, if the color of the flower is very strong, the model tends not to color it, as keeping it grey is probably the maximum likelihood under that level of uncertainty.
   This last observation is not valid for the Pix2Pix model, as it specifically tries to fix this problem.

7.1. Convolutional Auto Encoder

From the very first trial, we can see that the model produces very blurred images. This is due to the particular structure of the auto-encoder itself and in particular to the presence of the bottleneck in the middle. All the information, indeed, flow through it with an unavoidable loss of information that is also translated into a loss of reconstruction power.

7.2. Standard Deep Convolutional Neural Network

Changing topology, we immediately observe the emerging of a different problem: here the images are better reconstructed but the resulting outputs are *grayish* versions of the original ones given in input to the model. In particular, looking at the images we can see that the model is able to recognize the bright areas from the darker ones, but the coloring avoiding the presence of more lively colors.

7.3. UNet

Thanks to the residual connection, this model is finally able to perfectly reconstruct the image with respect to its details. However it is still affected by the two problems reported at the beginning of the section.

7.4. ResNet

The same holds for this network which, even though we are using the v2 that has a "direct" connection from the input to the output layer, and even though it's much deeper than the UNet, seems that it's not able to fix that problems.

7.5. DenseNet

Surprisingly this models performs worst than the UNet, even though its structure can be simplified to become a UNet without bottleneck, removing most of the residual connections. This is probably due to a optimization problem, as the dense residual connections tend to increase the *Internal Covariate Shift* [14] in the weights, which is mitigated by the Batch-Norm, but probably not as much as needed.

7.6. Pix2Pix

This model finally is able to overcome the grayish problem, exploiting the second GAN loss, which allows the model to guess brighter and more saturated colors, if that helps fooling the discriminator, thus shifting from maximizing maximum likelihood to minimizing the KL-divergence, as reported in the original paper [4].

However, the model is very sensitive with respect to the hyper-parameters, and the tuning of them is very difficult, as small changes brigs very big differences, which is also a behaviour observed in most of the GAN-based methods.

8. Conclusion

We observed that the model architecture is very relevant only comparing small models, where instead giving the network enough expressivity, reduces the dependence to the architecture. However, this does not hold for the grayish colorization problem, as that mainly depends to the objective that we are optimizing, and thus, there, is where the problem should be fixed. That is exactly what Pix2Pix does, and it's actually able to fix.
As future work would be interesting to compare this techniques to new ones like diffusion models, which are becoming very famous for their ability to map from one domain to another in the CV community.

Figure 7. Images from the test-set colorized with the various models

References

[1] Jimmy Lei Ba Diederik P. Kingma. Adam: A method for stochastic optimization, 2017.

[2] Max Welling Diederik P. Kingma. Auto-encoding variational bayes, 2022.

[3] Laurens van der Maaten Kilian Q. Weinberger Gao Huang, Zhuang Liu. Densely connected convolutional networks, 2018.

[4] Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Yoshua Bengio Ian J. Goodfellow, Jean Pouget-Abadie. Generative adversarial nets, 2014.

[5] You Zhou Jeff Hwang. Image colorization with deep convolutional neural networks.

[6] Jing Yu Koh Thang Luong Gunjan Baid Zirui Wang Vijay Vasudevan Alexander Ku Yinfei Yang Burcu Karagol Ayan Ben Hutchinson Wei Han Zarana Parekh Xin Li Han Zhang Jason Baldridge Yonghui Wu Jiahui Yu, Yuanzhong Xu. Scaling autoregressive models for content-rich text-to-image generation, 2022.

[7] Phillip Isola Alexei A. Efros Jun-Yan Zhu, Taesung Park. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.

[8] Kaggle. Flower classification dataset, 2021.

[9] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition, 2015.

[10] Philipp Fischer Olaf Ronneberger and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[11] Tinghui Zhou Alexei A. Efros Phillip Isola, Jun-Yan Zhu. Image-to-image translation with conditional adversarial networks, 2018.

[12] Alexei A. Efros Richard Zhang, Phillip Isola. Colorful image colorization, 2016.

[13] Hiroshi Ishikawa Satoshi Iizuka, Edgar Simo-Serra. Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification, 2016.

[14] Christian Szegedy Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.