Machine Learning and Big Data - Project
Grayscale image colorization

DIETRICH ALBAN, MUSIBAU SOLOMON, SAHLI YASSINE

Teaching staff :
Prof. Deligiannis
Prof. Munteanu

Project supervisors :
Ir. Remco Royen
Ir. Yuqing Yang

ELEC-Y591

Academic year
2020 - 2021

Contents

1 Introduction

Coloring a picture from a black and white image can be a daunting task. Today, thanks to AI based on grayscale image colorization, it can be done automatically. It can be very helpful in various fields such as in medicine to better interpret and analyze medical images. In this report, such a method using Convolutional Neural Network (CNN) will be presented. The aim here is to give to an image colors plausible to the human eye since some elements can have arbitrary colors like a ball, a shoe, etc.

During this project, two different approaches have been investigated. The first one is based on regression and the other on classification methods. Their results will be compared and discussed.

2 State of the Art

The image colorization can be achieved according to 3 different methods [7]:

- A non-parametric method where the user gives hints to the algorithm in the form of scribbles, i.e. by giving certain color information in different places of the image.
- A more automated approach where the user has to give one or several reference images to make color transfers (with texture matching or statistical data) onto the input grayscale image.
- A fully automated method based on CNN where the only input is the grayscale image to colorize.

We will focus on the last method which is fully automatic. Since 2012, the success of CNN has exploded with the apparition of the AlexNet [1]. It has found a lot of application in the field of imaging, video recognition, natural language processing, etc. The CNN also leads to the creation of the fully automated image colorization, as previously presented. With a big training set of images, it should find the best color transfers. This idea has been used and developed by several researchers in the past years [7].
A CNN model composed of 22 layers with a multinomial cross entropy loss using class rebalancing has been proposed by Zhang et al. [3] (based on the ImageNet dataset). This method predicts a particular color class (color histogram) for each pixel of the grayscale image.
A color histogram was also proposed by Larsson et al. with a CNN of 16 layers linked to a *hypercolumn* layer to guess the pixel color (more details in [2]).
Iizuka et al. [4] presented a network with two different channels : first one to determine the global features and the other for the specific features. Thus, we have first a classification of the global features which are then coupled to the local features where they are trained for colorization (by using a L_2 loss function). The use of these two channels allows the model to better understand the image.

Recently, generative adversarial networks (GAN) have been used for image colorization [8]. This model is composed of two networks, a generator and a discriminator, which are trained separately. The generator creates the colored version of the grayscale image and the discrminator judges if the result is good enough. If it is not the case, the weights are corrected in order to obtain a better match. Cao et al. [5] is a work example using GAN which obtains very good results based on the SUN dataset.

In this project, the automatic image colorization will be achieved by using regression [6] and classification [3] methods.

3 Algorithm Description

3.1 Data managing

The data used are images. The most famous color model is RGB (Red-Green-Blue). Our purpose being to retrieve a colored image from a grayscale one, we will prefer to work in the Lab color space where L is the lightness channel (coded from 0 up to 100, with 0 for black and 100 for white), a is related to the green-red colors and b to the blue-yellow colors (a and b both varying between -128 and 127). The Lab space allows to have a channel where the luminance is decorrelated to the maximum of the chrominance and was created to correspond as much as possible to the human perception of an image [7].
The datasets used for the model can come from different platforms (MIT Places365[1], Berkeley Segmentation dataset[2], etc). The choice of the data are important because it has an impact on the result of our model (for example if only pictures of places are used, the model will probably have difficulties to color a dog image[3]).
These data are downloaded and separated into training, validation and testing sets with a specific batch size. After resizing and cropping the RGB images (to obtain square pictures), they are transformed in the Lab color space such that for every images, we have its grayscale image (corresponding to the lightness channel L) and its 'colored' image (containing the color information of the image, corresponding to the channels a and b). The aim of the model is to find from the grayscale picture its corresponding 'colored' image. In other words, we want to determine the parameters a and b from the lightness L. The final datasets have thus for each picture the corresponding L and ab images.

Our datasets being prepared, we can use them according to a regression or classification model.

3.2 Regression

A first approach proposed is based on regression where the ab image (from a certain grayscale image) will be continuously determined.

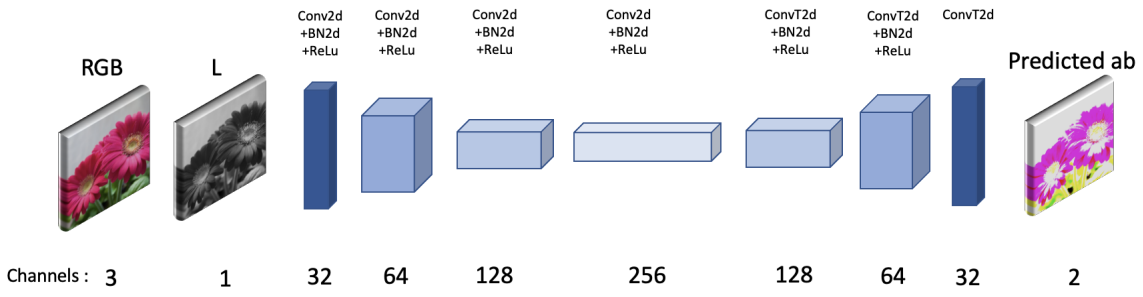

Figure 1: Global CNN architecture of the regression method.

The model used is a CNN composed of different convolutional layers of different dimensions, BatchNorm layers and ReLu activation functions (see figure 1). The optimizer used is the Adam optimizer with a learning rate $\alpha = 10^{-3}$. The criterion measures the mean squared error (corresponding to a L_2 loss function) between the reference ab image and the predicted ab

[1] see http://places2.csail.mit.edu/download.html.
[2] see https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/.
[3] We will see it in more details in the results section.

image by the CNN :

$$L_2(\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2} \sum_{h,w} \|\mathbf{Y}_{h,w} - \hat{\mathbf{Y}}_{h,w}\|_2^2 \tag{1}$$

where $\hat{\mathbf{Y}}_{h,w} \in \mathbb{R}^{H \times W \times 2}$ is the predicted ab image, and $\mathbf{Y}_{h,w} \in \mathbb{R}^{H \times W \times 2}$ is the ground truth ab image. H and W are respectively the height and the width of the images in pixels.

The training is done for a certain number of iterations (determined by the number of epochs). In each iteration, all the images from every batches of the training set will be browsed. In parallel, the validation step is done for the same number of iterations to see if the final results are satisfactory. Indeed we have to check if the trained model is applicable to a validation set and check for possible overfits.

The learning carried out with sufficiently low losses, we can apply the determined model on the test images and discuss the results.

3.3 Classification

The second approach is based on per-pixel classification where the chrominance space ab is discretised in 313 classes. Most datasets are poorly balanced in the chrominance space as low values are more often present, leading to desaturated images. We will see that the classification can tackle this problem by applying a weight rebalancing in the loss function to compensate for low occurrences.

The datasets being arranged, we express the ab images in the discretised space thanks to an one-hot encoding function. Each colored image can be represented in the ab space (with 2 dimensions, where a and b are the two axis) in which each pixel can be described by a coordinate position. By defining 313 classes corresponding to 313 pixels positions in the ab space, we can work with a classification approach. The encoding function browses all the pixels of the input ab image and assigns it a class between 1 and 313 (the closest to the pixel coordinate point, using nearest neighbors method).

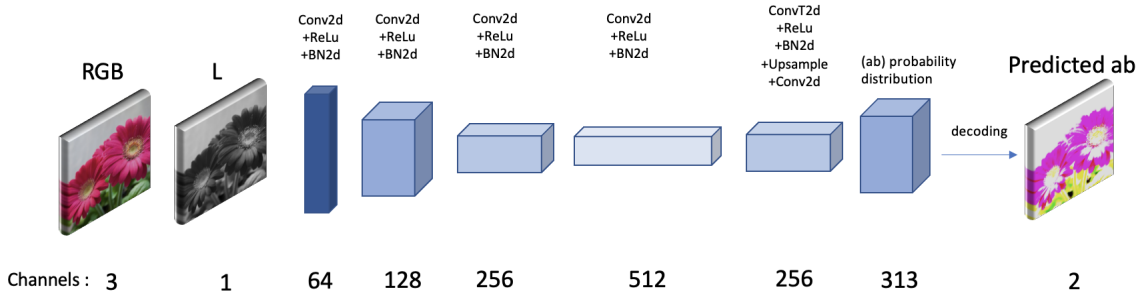

Figure 2: Global CNN architecture of the classification method.

Next, a CNN model for these 313 classes is applied, it is composed of convolutional layers of different dimensions and of several ReLu activation functions (see figure 2). In the training part, the model will be trained with the help of an Adam optimizer (with $\alpha = 10^{-3}$) and a cross entropy loss function where the losses increase when the class probability found for a certain pixel is far away from the right corresponding class:

$$L_{cl}(\hat{\mathbf{Z}}, \mathbf{Z}) = -\sum_{h,w} \boldsymbol{w} \sum_{q} \mathbf{Z}_{h,w,q} \log\left(\hat{\mathbf{Z}}_{h,w,q}\right) \tag{2}$$

where $\hat{\mathbf{Z}}, \mathbf{Z} \in [0,1]^{H \times W \times Q}$ are the predicted and ground truth encoded probability distribution over possible ab colors. Note that w are the weigths (without class rebalancing, $w = 1$).

Again the training is done for several iterations where the classes found for the pixels of the image according to the designated CNN try to converge to the ones from the ground truth image. In parallel, a validation loop for the same number of iterations is carried out in order to see if the model learns global features correctly.

With low losses, we can consider that the training is finished and move on to the test part. In order to display the results, we use a decoding function to go back from the predicted space $(313 \times H \times W)$ to their corresponding discretized ab value $(2 \times H \times W)$.

According to Zhang et al. [3], it was noticed that a weight rebalancing method can be used to avoid desaturated images. To do so, we can just modify the loss function by weighting it for each of the 313 color classes. Here below is the expression of the rebalancing weights proposed by Zhang et al.:

$$\boldsymbol{w} \propto \left((1-\lambda)\tilde{\boldsymbol{p}} + \frac{\lambda}{Q} \right)^{-1} \tag{3}$$

$$w_q \propto \left((1-\lambda)\tilde{p}_q + \frac{\lambda}{Q} \right)^{-1} \tag{4}$$

$$E[\boldsymbol{w}] = \sum_q \tilde{p}_q w_q = 1 \tag{5}$$

where a distribution based on the probabilities of occurrences is mixed with a uniform distribution weighted by the parameter $\lambda \in [0,1]$. In these formula's, $\tilde{p}_q$ is the occurrence probability[4] of the class q which is smoothed according to a Gaussian Kernel with standard deviation σ, Q corresponds to the number of classes ($Q = 313$), and $E[\boldsymbol{w}]$ is the expected value of the weights. Dedicated functions have been implemented to compute the weights, based on the whole training data set. As in the work of Zhang et al., values worked well in our case for $\lambda = 0.5$ and $\sigma = 5$ [3].

Here below on figure 3, we compute the probability distribution of classes without smoothing, then with smoothing, and finally the weights distribution.

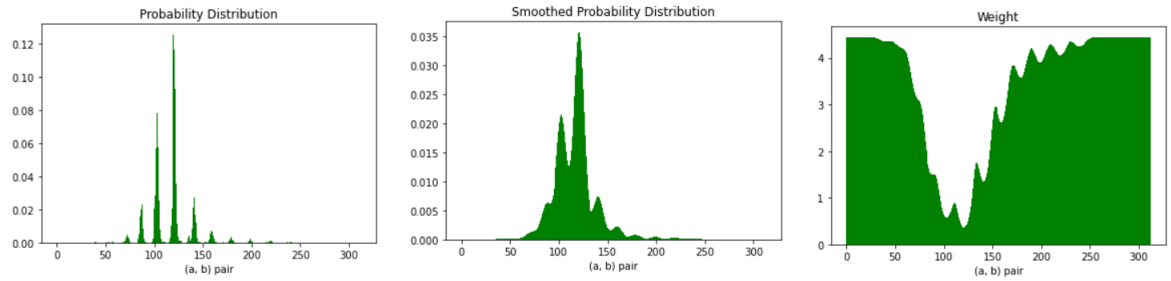

Figure 3: Probability distribution before smoothing (left), after smoothing (middle) and weights distribution (right), computed for the Berkeley training dataset.

[4] In our training dataset.

4 Experimental Results

In this part, two different approaches will be compared: the regression and the classification methods. Let us first begin with the regression results.

4.1 Regression

Here below are the results for a training with:

dataset	image size (H,W)	batch size	epochs	learning rate (lr)
Berkeley[5]	112x112	10	500	10^{-3}

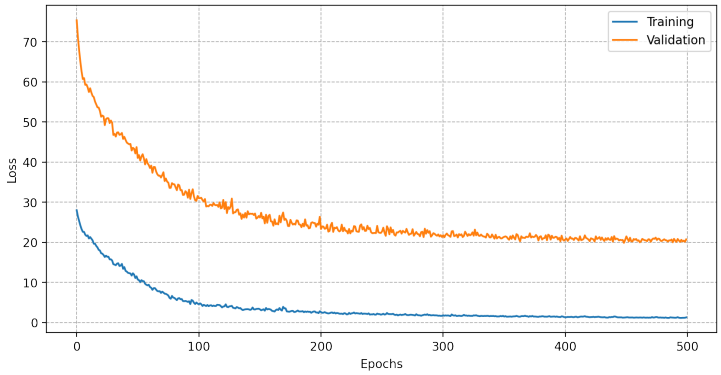

Figure 4: Training and validation losses according to the number of epochs.

We can notice that the validation and training losses both decrease and converge.

First, we will check that the regression model work, i.e. that it learn from the data. To do so, we will check the output based on grayscale images from the training set.

Figure 5: Results obtained based on grayscale images form the training set.

[5]Train size = 200, validating size = 200, test size = 100. See https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/.

We can notice that the model is working, it manages to reproduce the correct colors from the training images. Below are the results obtained for the test dataset.

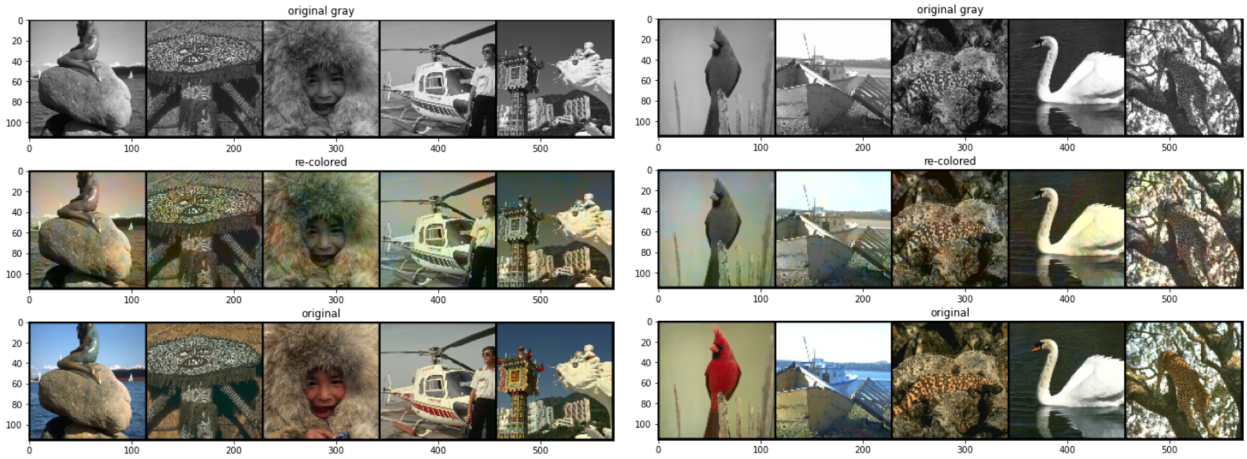

Figure 6: Testing of the model.

In general, as mentioned by Zhang et al. [3], we can notice that the images obtained on the test dataset by the CNN are desaturated. A way to avoid this is by using the weight rebalancing in the classification model.

4.2 Classification

Here below are the results for a training with:

dataset	image size	batch size	epochs	weight rebalancing	lr
Berkeley	56x56[6]	5	200	no	10^{-3}

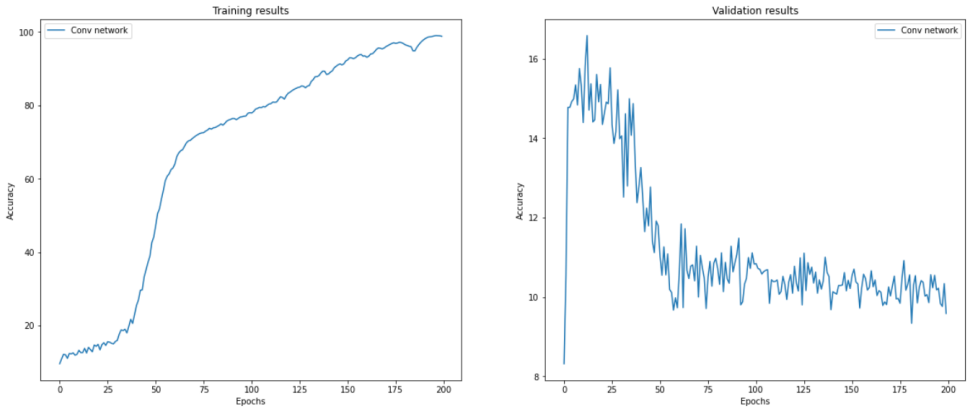

Figure 7: Training and validation accuracy according to the number of epochs.

In the above figures, the accuracy is defined as the number of pixels matching their ground truth ab class over the total number of pixel. We observe that the validation accuracy is much lower that the training one. This corresponds to an overfitted model, i.e. that it has learned too much from a restrictive training data set. That is a problem of our model (solutions will be proposed in the conclusion for further perspectives) but we will see that the results are still

[6]This resolution is chosen to reduce the calculation time.

interesting.

First, we will check that our model works, i.e. that it learns from the data. To do so, we will check the output based on grayscale images from the training set.

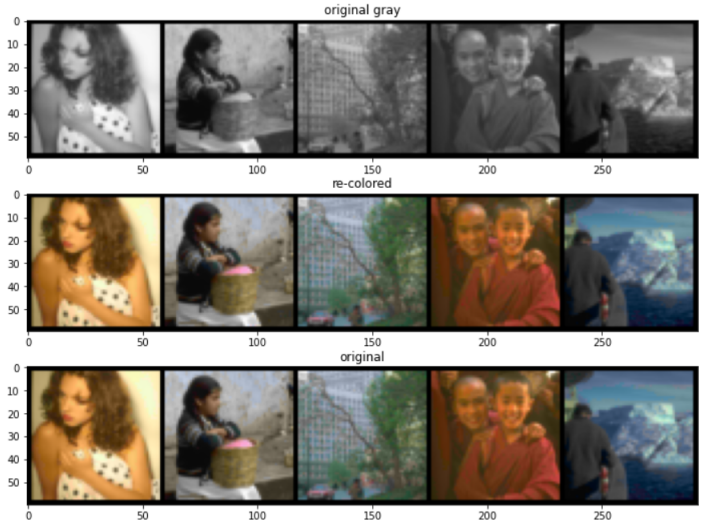

Figure 8: Results obtained based on grayscale images form the training set.

As for the regression approach, we can conclude that the classification model is learning. Below are the results obtained for the test images.

Figure 9: Test of the model.

We observe that the results are not very good but we have the colorization of the grayscale images. Once again, the colored images look desaturated. These results are probably due to overfitting as we could already understand thanks to figure 7. Learning to color an image is very difficult because there is a lot of features to take into account. This big amount of features can lead to overfitting if for example we do not have a large enough data set.

We now look at the results with the weight rebalancing :

dataset	image size	batch size	epochs	weight rebalancing	λ	σ	lr
Berkeley	56x56	10	200	yes	0.5	5	10^{-3}

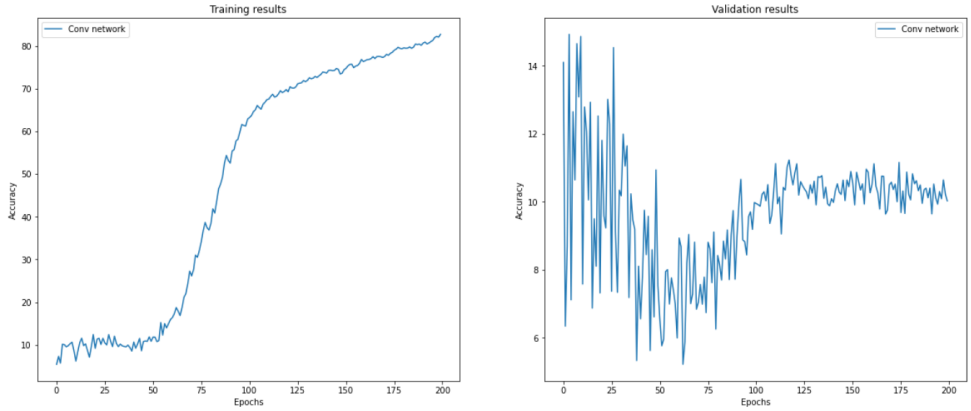

Figure 10: Training and validation accuracy according to the number of epochs.

Here below are the test results:

Figure 11: Test of the model.

Thanks to the weight rebalancing, we get better results with brighter colors. The results are good but they can be better by using a simplier dataset. Indeed, the large number of features that the model has to take into account to find a good result is limiting. To minimize this number of features, we built a simple dataset[7] of white images over which several regular polygons with specific colors are randomly placed.

Here are the parameters used:

dataset	image size	batch size	epochs	weight rebalancing	λ	σ	lr
Form dataset[8]	56x56	10	50	yes	0.5	5	10^{-3}

[7]Thanks to a Python geometric form generator code.
[8]Train size = 200, validating size = 200, test size = 100.

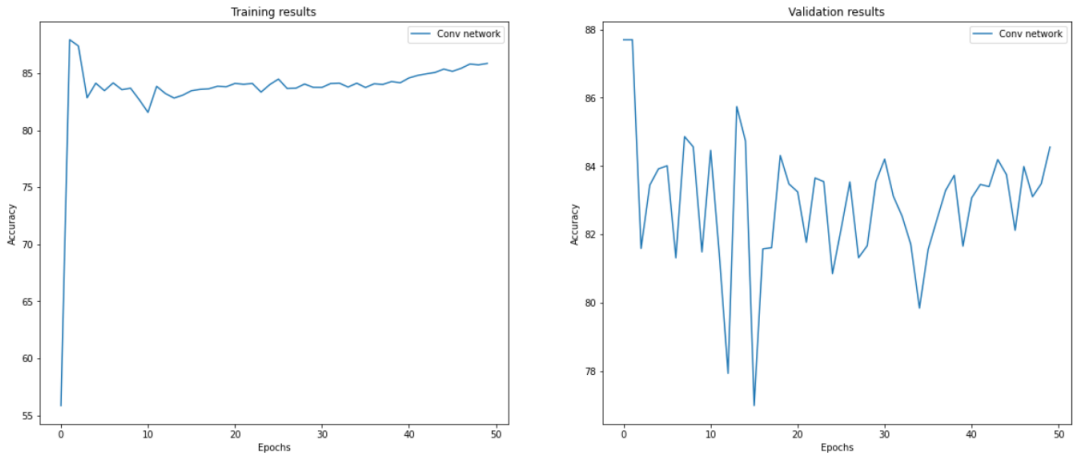

Figure 12: Training and validation accuracy according to the number of epochs.

Thanks to the simpler dataset, both the validation and the training accuracies are high. It allows to avoid overfitting.

The results obtained from the tests are presented below:

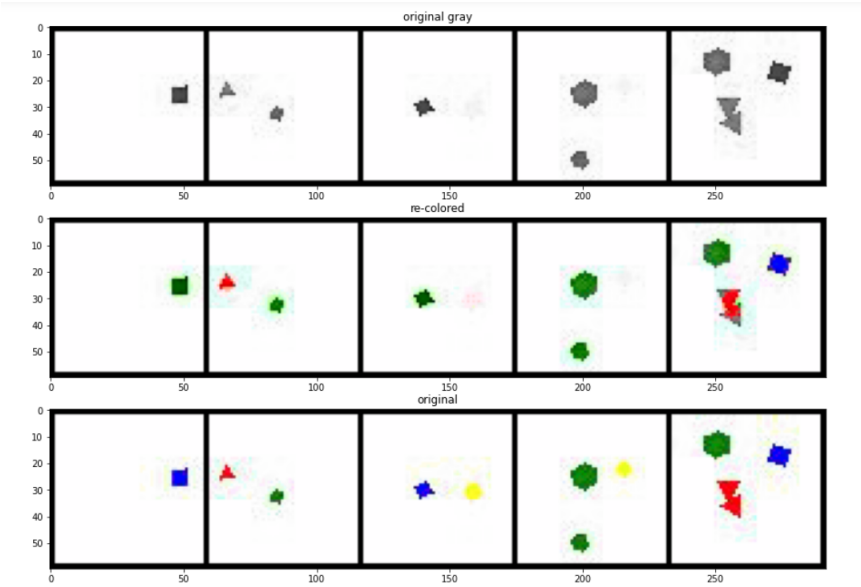

Figure 13: Test of the model.

We notice that the results are better due to the simplicity of the pictures (which leads to fewer features). We also notice that sometimes, the model confuses hexagons and squares. This may be due to a too low resolution. Using the same parameters as previously but with epochs = 10 and a higher resolution of 112x112[9], we see in the figure 14 that the model confuses less often the polygons and obtains in the case of this test nearly perfect results.

[9]A lighter CNN model was used (four layers were removed) due to the lack of memory in our machines. However, given the simplicity of the images, the decrease in the number of parameters in the model should not be felt too much.

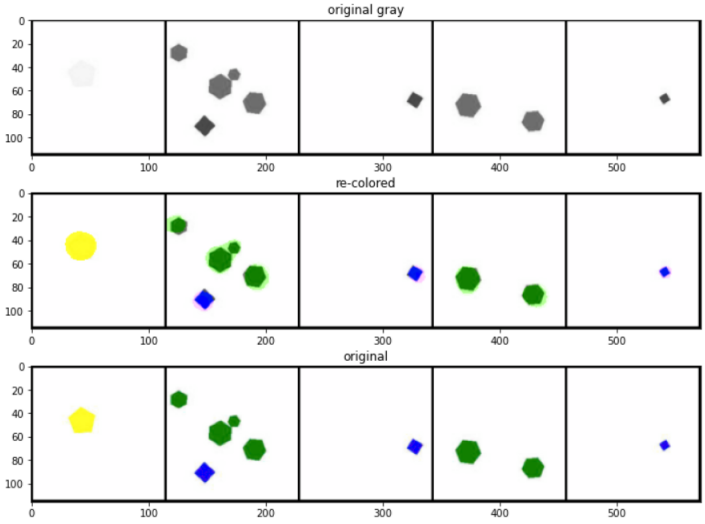

Figure 14: Test of the model with higher resolution.

5 Conclusion

Coloring an image automatically is a huge challenge. Thanks to advances in CNN, several researchers were able to bring new models with very good results obtained by doing very intensive trainings (see for example [3]). These models can be based on regression or classification. We have noticed that for the regression approach, the results were good but the colors are desaturated. A solution to avoid this is by using classification with weights rebalancing. With this classification, good results were obtained for our simple dataset. Indeed, an image and its colors contain a lot of information that we have to manage. The complexity of these data leads to a very large number of features to learn. This big amount can lead to overfitting if we do not have enough data, as we have seen in the previous section. With fewer features (with the simple form dataset) and simpler data, we managed to obtain good results. To avoid overfitting with the more complex images, we could use larger datasets[10]. We could also use more pooling layers to reduce the number of parameters. A method called *Dropout* where neurons are dropped to avoid a model too focused on the training set could also be used.

In conclusion, two approaches, regression and classification, have been proposed. These work well (with a preference for weight rebalancing classification to avoid desaturated images) but care must be taken to avoid overfitting by using for example one of the solutions proposed above.

6 Contributions

- Dietrich Alban : Regression + Classification with class rebalancing
- Musibau Solomon : Regression + Classification without class rebalancing
- Sahli Yassine : Classification without + with class rebalancing
- Everyone : Report + Presentation

[10]We were unable to do so because of the time that it would take. In [3], the data set used contains around one million images. This confirms the need to have large datasets, especially in the case of image colorization which contains a lot of characteristics to learn.

References

[1] A. Krizhevsky et al. 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*, volume 25. Curran Associates, Inc., 2012.

[2] G. Larsson et al. Learning representations for automatic colorization. *CoRR*, abs/1603.06668, 2016.

[3] R. Zhang et al. Colorful image colorization. *CoRR*, abs/1603.08511, 2016.

[4] S. Iizuka et al. Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics*, 35(4), July 2016.

[5] Y. Cao et al. Unsupervised diverse colorization via generative adversarial networks. *CoRR*, abs/1702.06674, 2017.

[6] Z. Cheng et al. Deep colorization. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 415–423, 2015.

[7] D. Futschik. *Colorization of black-and-white images using deep neural networks*. PhD thesis, Czech Technical University in Prague, 2018.

[8] K. Nazeri and E. Ng. Image colorization with generative adversarial networks. *CoRR*, abs/1803.05400, 2018.