# Project Report
# 30562 - Machine Learning and Artificial Intelligence
**Spring 2023**


Ardenghi Alessandro 3144656 | Bianco Camilla 3138578 | Saveri Leonardo 3139812 | Zoccante Andrea 3150188



## Abstract

*In this report, we will present our deep learning project, whose main focus is computer vision, more specifically image colorization. For the training and validation of our models, we have decided to use the MIRFlickr-25000 dataset and the Stanford Dogs dataset. Our aim is to create a model to predict the RGB colorization of grayscale images, and in order to do we will first map the images to the LAB color space, perform our predictions, and then map the results back to RBG. During this project we will explore different architectures (Convolutional Neural Networks, Attention Modules, etc.), presenting here the best model. We will explore the application to multi-domain and to class-specific datasets, explaining the results, the difficulties and the possible next steps.*


## 1. Introduction

Our aim through this project is to build a model to colorize black and white images, as this could be useful to colorize historical photos and videos (where each frame can be considered an image) as well as colorize images that were taken originally in B/W.
This has already been done by other researchers, as for ChromaGAN[1] (where is used an adversarial learning colorization coupled with semantic information, trained in a self-supervised manner.) or U-NET[2] (an architecture for semantic segmentation initially designed for biomedical image processing, also achieved good results in other computer vision tasks, such as image colorization.).

## 2. Approach

In order to build our model, we first explore the data and how to preprocess them. From there, we start using the MIRFlickr-25000 dataset to try and build a model to colorize a multi-domain dataset. We train and test on one image, on two images of different sizes, and then on the whole dataset.
Later, we decided to apply the best model we found with the multi-domain dataset to try and colorize images from a class-specific dataset, the Stanford Dogs dataset, starting from colorizing images of Golden Retrievers and then moving to a model trained over two different breeds.

## 3. Experiments and Results

### 3.1. Data and Preprocessing

To train and validate our model we decided to use the MIRFLICKR-25000[3] dataset, released in 2008. This dataset is made out of 25000 .jpg images, both colored and black and white, and contains also some annotations, which we didn't use for our project.
We chose this dataset because it contains a variety of very different images, varying in sizes, colors and subjects. Another reason why we chose such dataset was because it was available publicly.

To preprocess the data, we decided to first open the images using the RGB format, and then to convert them to LAB color space in order to work with fewer channels. The LAB color model is a color space that represents colors based on human perception. It consists of three channels: L (lightness), A (green-red), and B (blue-yellow). Unlike the RGB color space, which is based on additive color mixing, the LAB color model separates the lightness channel from the color channels, making it more suitable for certain image processing tasks. Converting the dataset images from RGB to LAB format allowed us to reduce the number of channels to predict from three (Red, Green, and Blue) to just two (A and B).

We then normalized the channels. The L channel's values range from 0 to 100, so we simply divided such channel

[1] https://arxiv.org/abs/1907.09837
[2] https://arxiv.org/abs/1505.04597
[3] https://press.liacs.nl/mirflickr/mirdownload.html

by 100 to get values for each pixel between 0 and 1. The A and B channel's values range from -128 to 127, so we divided the channel by 128 to get values between -1 and 1, perfect to generate prediction using a $\tanh$ function.

In this way, for each image, we were able to obtain an array X containing only 1 channel (the L channel, used as input) and the array y containing 2 channels (A and B, which we are trying to predict).

#### 3.1.1 Data Loader

Because our dataset was of considerable size, we decided to build a function that takes as input a directory and a batch size, and returns a data loader which allows us to train the model on the whole dataset without having to load it all at once.

### 3.2. MIRFlickr-25000

#### 3.2.1 One image

After having setup our preprocessing functions, we wanted to check that the preprocessing was working fine, and that we were able to predict values using a simple Convolutional Neural Network. To do this, we decided to train the NN with just one image, and then predict the values for the same image. We set the epochs to 1000, and in the end we were reaching a loss in the magnitude of $10^{-4}$. This was indeed pretty good and, plotting the image (Figure 1), we reached great results.


Figure 1. Original, B/W and Colorized versions using the same image as training

Even if this result is in fact good, we are training the network only on one image and predicting the same values. The result was unsurprising, but our goal was to check the correctness of our preprocessing functions, and this was indeed a success.

#### 3.2.2 Two images (Different Sizes)

After having assessed the correct functioning of the preprocessing functions, we tried to train the model using two different images, that had very different colors and shapes. Running the same simple Convolutional Neural Network, we were able to reach good results, and to predict both images almost correctly.


Figure 2. Original, B/W and Colorized versions using the same 2 images as training

We are also happy to see that our post processing let us keep the original sizes of the images, and that even using 2 images, the model was able to predict the right colors (Figure 2).

We decided then to test it using a never seen in training image. Although, as expected, the model performs well on the two training images, its performance is not particularly nice when faced with unseen images (Figure 3.

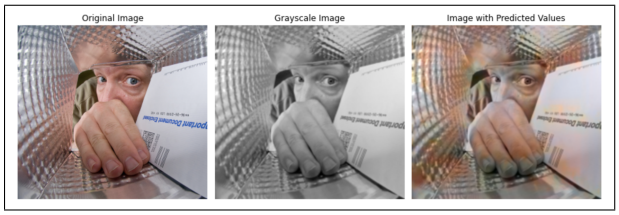


Figure 3. Original, B/W and Colorized version of a test image

#### 3.2.3 Full Dasatet

Let's now move to the full dataset and use the data loader we built. We mainly worked with Convolutional Neural Networks, testing various parameters and architectures. We will look at some of the results and only go in depth for the last and more efficient Neural Network.

#### 3.2.4 Model

In general, by training with about 20k images, it was difficult to reach a loss that was smaller than 0.0180, although 0.0180 is order of magnitudes bigger than the error we obtained when training with only 2 images.

We started with the simplest model possible and then iteratively updated it based on our intuitions and comparisons with previous attempts, which we found in the lit-

erature review we conducted. Initially, our first model (*Basic_Model*) consisted of just three Convolutional Layers (Figure 4). As we progressed, we utilized a similar model to the one we used to analyze single and dual images. This updated model (*Batch_Model*, Figure 8) included Convolutional Layers, Batch Normalization Layers and Upsampling Layers, which were put in place to handle the transition from one input channel to two predicted channels.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 200, 200, 64)      640
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 200, 200, 32)      18464
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 200, 200, 2)       578
=================================================================
Total params: 19,682
Trainable params: 19,682
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. The First Model's Architecture

However, we noticed that the results were not improving significantly, so we decided to make our model deeper and refine the architecture further. Taking inspiration from the UNet architecture (or more precisely, from its variation CUNet[4]), we designed a new model (*Skip_Model*) with two main components: a downsampling path and an upsampling path. These paths were connected through skip connections, which proved to be vital for combining low-level details with high-level contextual information.

In our last model (*Attention_Model*) we introduced a Multi-Head Attention layer. This layer induced self-attention on the feature maps and allowed us to assign varying levels of importance to different spatial locations. By incorporating this attention mechanism, we aimed to improve the overall performance of our model.

By looking at the losses of the different models we tried, we came to the conclusion that, even if the improvement was very small, the *Batch_Model* was the best (Figure 5)

In any model we implemented, the last layer had as activation function a $\tanh$. This is because we wanted the predicted values to be in the range $[-1, 1]$, and a ReLU function would have given us only positive values.

### 3.2.5 Loss function

We decided to use, given the type of data and the type of task, a simple Mean Squared Error as loss function.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

[4]https://arxiv.org/pdf/2205.12867.pdf


Figure 5. The losses of the different models

This was mainly due to the fact that is was able to correctly show the differences between values of the predictions, which are all between -1 and 1.

### 3.2.6 Optimizer

We also decided to use, as an optimizer, Adam. The choice was mainly made because of its adaptive learning rate mechanism. We experimented extensively with different optimizers, such as for instance "SGD with Nesterov's Momentum", as well as different adaptive learning rate schemes, but we did not obtain any significant improvement on the predictions.

### 3.2.7 Results

After having tried different models, with different optimizers, learning rates and batch sizes, we were able to obtain a model which was able to correctly spot the shapes of the subjects in the images , although it colored all the images in almost completely uniform sets of colors, namely colors which tend to the shades of brown-pink. Almost as if the colors of all the images were blended together. Our interpretation of this result was that the optimisation process got stuck in a local minimum, which in terms of coloring is represented by the fact that the image is colorized uniformily. Most likely, this brownish color minimizes the loss across all the images because its numerical value is very close to the mean value of the pixel coloration.(Figure 6).

The fact that the shapes and the shades were correctly predicted made us think that the model wasn't too bad (with a loss higher than 0.02/0.03 also the shapes weren't good enough), but in order to predict the colors we either needed to train for much longer, with more epochs, or needed higher quality data. Unfortunately, the training with such a huge dataset was too slow with the resources we had.


Figure 6. Image not used in training, its grayscale version and the predicted version

#### 3.2.8 Conclusion

Because of the problems that we observed, trying many different type of models, we came to the conclusion that probably the problems we were having were given by the fact that the images in the dataset were too different between them (too many subjects, too different between them) without any way to fully find patterns between images and colors.
This would be a very difficult task even for a human. It is not easy to predict colors just by the lightness of the pixel. Because we weren't able to leave the model running for that many epochs to see if we were able to minimize the loss enough, we decided to change our direction, and try with a single-class dataset. We decided to apply the final model we obtained before on a dataset of dogs' images. In this way the model might not only look at the pixel's lightness, but might also find patterns from the subjects.

### 3.3. Stanford Dogs

For this, we decided to use the Stanford Dogs Dataset[5], the dataset is made of 20580 images, divided in 120 breeds. Because we wanted to check how the model would work with similar images, we decided to first isolate only one breed, the Golden Retriever and then to apply it to two different breeds, the Golden Retriever and the Husky.

#### 3.3.1 Data Augmentation

Because each class is made by about 150-200 images, we decided to use some Data Augmentation techniques to generate some more examples that the Neural Network could use to train. First we simply looked at 5 rotations (Figure 7), then we explored also other tactics.

Figure 7. Rotations performed for Data Augmentation

[5]http://vision.stanford.edu/aditya86/ImageNetDogs/

#### 3.3.2 Model

The model we decided to use for the dogs dataset was the one shown in Figure 8.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| batch_normalization_21 (Batc | (None, 200, 200, 1) | 4 |
| conv2d_51 (Conv2D) | (None, 100, 100, 8) | 80 |
| conv2d_52 (Conv2D) | (None, 100, 100, 8) | 584 |
| batch_normalization_22 (Batc | (None, 100, 100, 8) | 32 |
| conv2d_53 (Conv2D) | (None, 100, 100, 16) | 1168 |
| conv2d_54 (Conv2D) | (None, 50, 50, 16) | 2320 |
| batch_normalization_23 (Batc | (None, 50, 50, 16) | 64 |
| conv2d_55 (Conv2D) | (None, 50, 50, 32) | 4640 |
| conv2d_56 (Conv2D) | (None, 25, 25, 32) | 9248 |
| up_sampling2d_15 (UpSampling | (None, 50, 50, 32) | 0 |
| conv2d_57 (Conv2D) | (None, 50, 50, 32) | 9248 |
| conv2d_58 (Conv2D) | (None, 50, 50, 32) | 9248 |
| up_sampling2d_16 (UpSampling | (None, 100, 100, 32) | 0 |
| conv2d_59 (Conv2D) | (None, 100, 100, 16) | 4624 |
| conv2d_60 (Conv2D) | (None, 100, 100, 16) | 2320 |
| up_sampling2d_17 (UpSampling | (None, 200, 200, 16) | 0 |
| conv2d_61 (Conv2D) | (None, 200, 200, 2) | 290 |

Total params: 43,870
Trainable params: 43,820
Non-trainable params: 50

Figure 8. Model used for the dog's dataset

#### 3.3.3 One Breed

As we said, we decided to start by looking at images of Golden Retrievers. There were 150 images of this breed, with different backgrounds and situations. We used 120 as training, and 30 as validation set. We used directly the best model we found for the MIRFlickr dataset, and started from there.

We were happy to see that the loss was much better when training over one single class of images, reaching values of roughly 0.0095. Even to the human eye, the results were much better than before. (Figure 9)

As Figure 9 shows, the predictions are already much better than the ones that were made by the model trained on the MIRFlickr dataset. They are not exactly like the original, but we can see that the model predicts the color, for the dogs, almost correctly (or at least very coherently with what they would look like) and colors also the backgrounds in a plausible way.
We suspect that this is because the model not only recog-

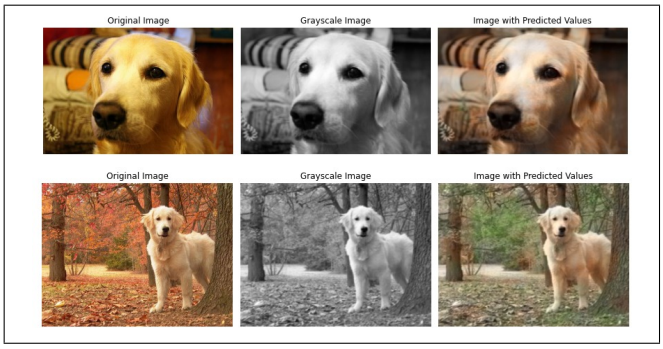


Figure 9. Original, B/W and Colorized image using data augmentation. Prediction over a validation image, not used in training.

nizes the luminescence of the pixel, but also the context of the image. In fact, it colors the grass and the leaves in green, which are not as in the original image, but as we would expect them to be in reality (Figure 10).

We can observe that the dogs, indeed, exhibit colors that are all plausible. But we can also observe that other subjects, such as the human in the second row, are not colored correctly. This is, probably, because the convolutional network was able to comprehend what a dog is, and what color it is, with just 120 images and their augmentation, while it was not able to understand how to identify a human, and how to color it correctly.
This increases our hopes that, with a more structured dataset, a general image colorization (or at least over a set of classes) is possible.

#### 3.3.4 Two Breeds

Given our results from above, we decided to try to implement the same model using two dog breeds, more images and more data created by the augmentation. We applied the same procedure as we did for all the other datasets, and trained the Batch Norm Model on this mixed dataset. Our objective was to test whether the model would not only be able to detect the presence of a dog in the picture, but also to correctly discern the different breeds and color them accordingly.

#### 3.3.5 Results

Using the Stanford Dogs dataset, we were able to understand that a model, trained over a specific class of images, can predict pretty well, or at least in a plausible way, the colors of the subject. The results on the two breeds dataset were not as good as the ones we obtained when we trained on one single class, but they were still acceptable. The model was clearly able to detect the presence of a dog, but in the two breeds case, it was coloring Huskies and Golden Retrievers in the same way, which made us believe that it

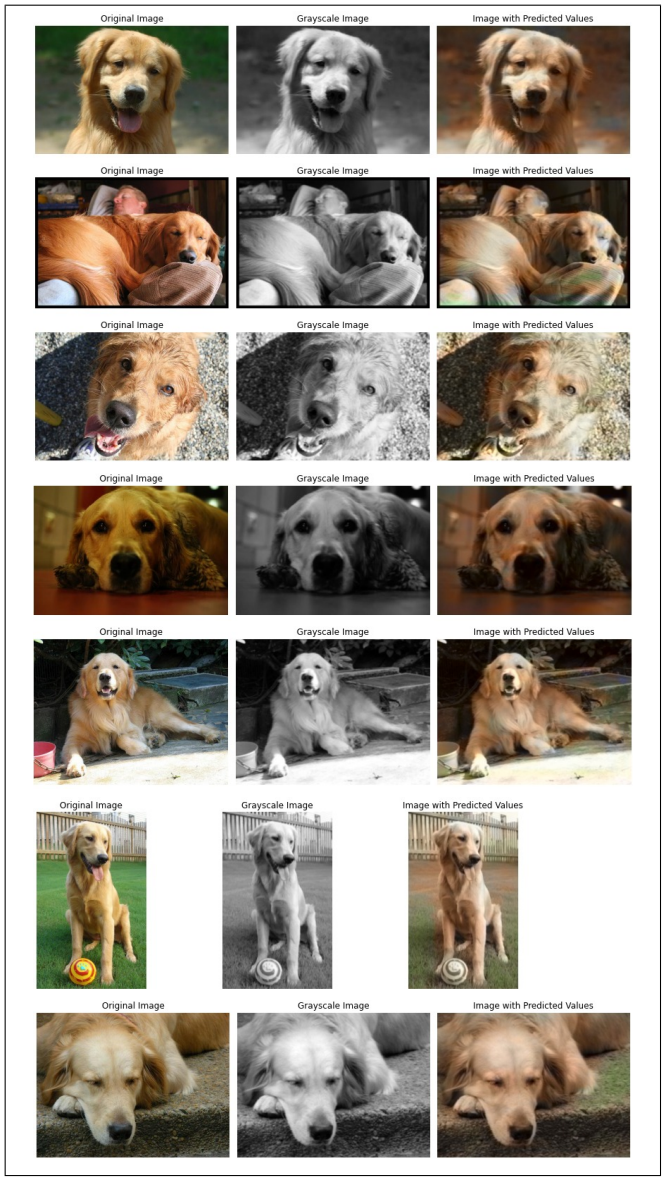


Figure 10. Original, B/W, and Colorized images, using data augmentation. Prediction over images, not used in training

was only capable of detecting the presence of the dog, but not its breed (Figure 11). Another interesting observation is that the model seems to have understood when there is grass around, and it was almost always coloring it in the correct way.

## 4. Conclusion

In conclusion, we are happy to report that image colorization seems possible when the images are enough for each class of data. Building a model to predict the colors of a particular subject (i.e. golden retrievers) is quite easy, even with not so many images, using some techniques of


Figure 11. Prediction on a Husky picture from the 2 breeds model

data augmentation and a not so complex neural network.
When using a multi-domain dataset (the MIRFlickr-25000), we never succeeded in obtaining a MSE smaller than 0.0180. This led the images to be colorized with what seemed to be a blend of all the colors. But the convolutional neural networks we tried were always quite good at recognizing shapes and shades.
When using a dataset that was tailored to different classes (thus containing various samples of similar subjects) we were able to obtain better results, with our model being able to colorize the main subjects and most of the backgrounds with plausible colors.

### 4.1. Overfitting

A little comment on overfitting. While we did try to apply an EarlyStopping in different ways during the training of our models, we came to the conclusion that models with no early stopping were best at recognizing and predicting colors. More specifically, the images created by models with no early stopping had less noise (halos, better defined shapes and more various colors) than the ones in which we implemented it. This is probably because a little bit of overfitting, when colorizing images of a specific class, is not too bad.

## 5. Difficulties

During our project, we found many difficulties that we had to overcome. First, we understood that the first dataset had images too different from each other, and given that the model not only tried to predict from the lightness, but also from the subjects, having too many different subjects shown to it, the model was not able to gain enough information to correctly color the images, not even backgrounds like trees. Another main difficulty that we faced, was from the computational point of view. Often the GPU was running out of memory, and we were never able to try more than one complex model at once, making it difficult to compare many different architectures. Often, this problem occurred even when training a single model, especially when they were quite deep (i.e. the architecture with the attention module). Also, it was often taking very long for models to train, making it again difficult to try many different possibilities.

## 6. Future Efforts

Given our results in colorization, after understanding that it works better with many images for a small/determined amount of classes, we thought it could be interesting to build an architecture that first classifies an image (for example, the breed of the dog) and then colorizes it based on the above result.

Furthermore, at the beginning of the project, we had plans to experiment with the use of GANs as well. However, this proved to be more challenging than expected, especially considering the fact that we hadn't had the opportunity to study their theoretical aspects or practically try them in any context yet. Nevertheless, during our literature review, we noticed that their utilization yields excellent results, and we are confident that in the future, we could implement them to enhance our work.