BIG DATA CONTET ANALYSIS

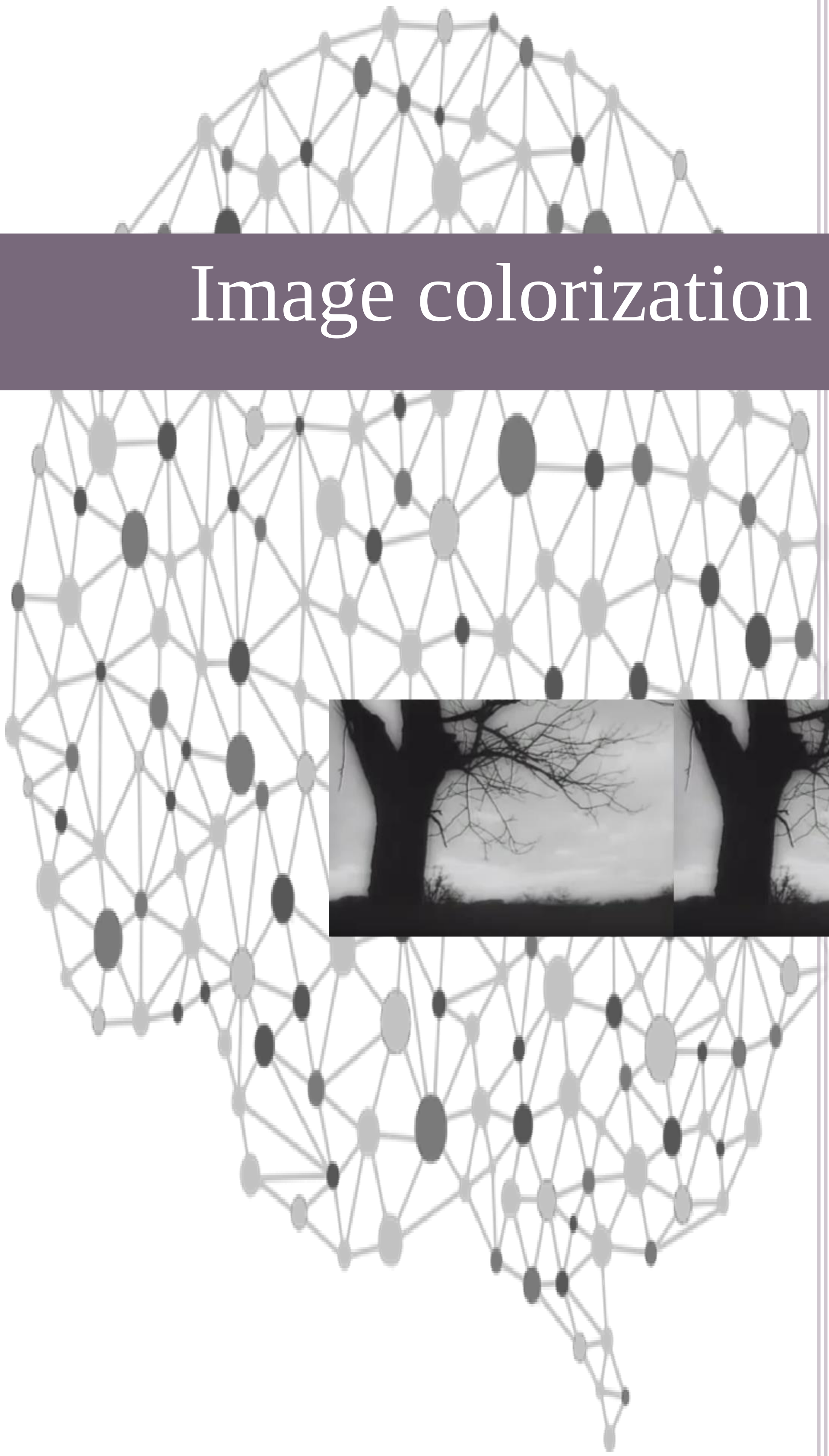


Image colorization for **Netflix**

ALEXIOU DIMITRIS-FOTOPOULOS SPUROS-KAILANI ANASTASIA-KOURTESI IOANNA

9/8/2019

Table of Contents

Figures

1.Introduction

In this assignment the main goal will be dealing with a model of Artificial Neural Network. Initially, we have to explain what a neural network is. This type of systems learn" to perform tasks by considering examples, generally without programmed with task-specific rules. They have no previous clues about the thing we are going to search, for example let’s say we are interested to find the cats on an image. The machine has no clue how a cat looks, instead they automatically generate identifying characteristics from the examples that they process.

A neural network is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The original goal of these networks approach was to solve problems in the same way that a human brain would. However, over time, attention moved to performing specific tasks, leading to deviations from biology. Neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.

Deep learning methods use data to train neural network algorithms to do a variety of machine learning tasks, such as classification of different classes of objects. In this project we will be running an algorithm for Convolutional neural network. Convolutional neural networks are deep learning algorithms that are particularly powerful for analysis of images. This type of network is using the data in order to learn. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better. The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

We are supposed to be employees of Netflix. Netflix, Inc. is an American media-services provider and production company headquartered in Los Gatos, California, founded in 1997 by Reed Hastings and Marc Randolph in Scotts Valley, California. The company's primary business is its subscription-based streaming service which offers online streaming of a library of films and television programs, including those produced in-house.[9] As of April 2019, Netflix had over 148 million paid subscriptions worldwide, including 60 million in the United States, and over 154 million subscriptions total including free trials.[8] It is available almost worldwide except in mainland China (due to local restrictions) as well as Syria, North Korea, Iran, and Crimea (due to US sanctions). The company also has offices in the Netherlands, Brazil, India, Japan, and South Korea.[10] Netflix is a member of the Motion Picture Association of America (MPAA).

So, Netflix is thinking to add a new feature to reproduction of movies and videos. They want the user to have the ability to convert a grayscale movie to a colorized one. A video, as it is logical, is a production of a lot of images. So, Netflix created this project and build a team in order to make it happen. Netflix ended up to add this feature to the options of the user after several searches and questions they have made to the users.

2. Mission & Dataset

The working policy of the Netflix is project oriented. That means that the managers are thinking of a project and they assign it to the appropriate team. After a lot of researches regarding the preferences of the users for the old movies. They perceived that a lot users have a predilection of old movies from 80s or 90s. But at this time, most of the movies maybe all of them were black and white. After putting the movie Netflix observed that after 10 to 15 minutes they did not want to watch this movie anymore, because as they said they were bothered from the grayscale images on their screen The problem here that young people, nowadays, are not used to watch something without a color.

They created a project, in which they wanted to make the black and white videos with color. To do so, the created a Research & Development department with four people. Each one of them has a different role. The goal is to find a good algorithm in order to make an image from black and white to color. We will start with images and later the Netflix will expand this algorithm to the videos.

Till now, we have described widely how this network works and what we are suppose to do. The goal of this particular project is to take a number of photographs and try to colorize them, as we have described above. A very important part of this procedure is the dataset. We have to have a good amount of pictures in order to make the algorithm every time even better for this task. The point here is to have a grayscale image as the input and we have to put color on it. This is a very challenging problem in terms of color. And that's because a grayscale image could have a variety of colors, that we need to track and put on.

At this point we introduce an overview of previous work in the field of automatic colorization, with particular focus on works that have inspired, influenced and helped shape this thesis. In general, working with colour channels in images is a rather well studied and surveyed area. However, compared to other similar problems, the idea of generating some or all of the color information given other data is one that has seen relatively limited amount of widespread application and, conversely, research. However, it is quickly becoming one of the more popular image-to-image tasks in the computer vision field, with several works published in the recent years using various approaches. Countless algorithms that are not considered colorization methods, but rather mere color enhancements which aim to improve existing poor color information or modify the color palette of an image are worth mentioning in this section, as they serve as a sort of a precursor to full colorization techniques.

The goal of these methods is to take images with color data as their inputs and transform them into images with better visual properties. Often, they serve to remedy certain camera defects, such as overexposure or underexposure contrast adjustment through histogram equalization. The simple transformations performed by these non-parametric methods are frequently used to describe behaviors of other, more complex algorithms, even in the domain of CNNs. It is for example possible to say that one of the transformations performed by a CNN resembles histogram equalization.

When considering full image colorization, there are generally three major types of approaches that have been used. Firstly, a non-parametric approach, in which the user provides hints to an algorithm as to what the final colorization should look like. These hints come in the form of scribbles - small patches of color in specific areas of the image. More automated methods developed still rely on additional user input, but instead of providing direct color data, the user is expected to provide one or more reference images from which to perform color transfer onto the target image using statistical data or texture matching.

Recently, with the advent CNNs, the approach has shifted more towards a fully automated solution, where the only input provided by the user is the target grayscale image. However, this enormous advantage can also turn into a disadvantage - in case the CNN result turns out to be unsatisfactory, there are few to no options to easily remedy it trivially, unless the model has been specifically designed with this requirement in mind.

We aim to infer a full-colored image, which has 3 values per pixel (lightness, saturation, and hue), from a grayscale image, which has only 1 value per pixel (lightness only). For simplicity, we will only work with images of size 256 x 256, so our inputs are of size 256 x 256 x 1 (the lightness channel) and our outputs are of size 256 x 256 x 2 (the other two channels). Rather than work with images in the RGB format, as people usually do, we will work with them in the LAB colorspace (Lightness, A, and B) . This colorspace contains exactly the same information as RGB, but it will make it easier for us to separate out the lightness channel from the other two (which we call A and B). We'll make a helper function to do this conversion later on. We'll try to predict the color values of the input image directly. We cannot proceed any further without having a dataset. We have already mentioned the dimensions of the pictures we are going to use. The dataset will be consisted of approximately 300 thousand grayscale pictures.

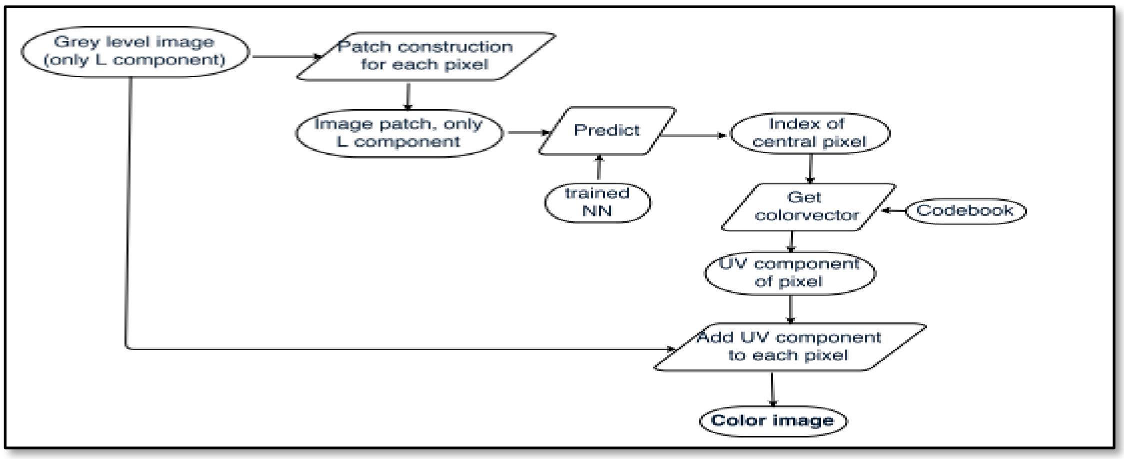

Grey level image (only L component)
Patch construction for each pixel
Image patch, only L component
Predict
trained NN
Index of central pixel
Get colorvector
Codebook
UV component of pixel
Add UV component to each pixel
Color image

Figure 1: Color Prediction Workflow

3. Methodology

At first it is important for the explanation of the process to clear that we are going to have a test and a train dataset. The image set used for the model is divided into the training image set and the testing image set. The train image set is given to the machine in raw input vector representation. On this input set, deep neural network (convolution operations) is applied. On the test image set, image patch segmentation and effective patch extraction is done. Effective image patch extraction gives us the set on which deep neural network is applied. Deep neural network is applied in the form of layers: input layer, hidden layer, output layer, soft max classification. From the CNN applied train image set, network parameter is passed to the CNN applied test image set. Using the network parameter, both the sets are compared and majority voting is done and we finally get the classification results.

The model we are going to use is a convolutional neural network model. We will use Python as the main programming language of this assignment. Of course it is necessary to use a appropriate library in order to implement this algorithm. For our model we will use PyTorch. We'll build and train our model with PyTorch. We'll also use torchvision, a helpful set of tools for working with images and videos in PyTorch, and scikit-learn for converting between RGB and LAB colorspces.

Deep learning education and tools are becoming more and more democratic each day. There are only a few major deep learning frameworks; and among them, PyTorch is emerging as a winner. PyTorch is a machine learning library based on the Torch library, used for applications such as deep learning and natural language processing. It is primarily developed by Facebook's artificial intelligence research group. It is free and open-source software released under the Modified BSD license.

Python continues to be a very popular language even among academics. PyTorch creators wanted to create a great deep learning experience for Python which gave birth to a cousin Lua-based library known as Torch. Hence, PyTorch wants to become a Python-based deep learning and machine learning library which is open source. At the same time, it can give every Python user to build great machine learning applications for research prototyping and also production deployment.

PyTorch builds deep learning applications on top of dynamic graphs which can be played with on runtime. Other popular deep learning frameworks work on static graphs where computational graphs have to be built beforehand. The user does not have the ability to see what the GPU or CPU processing the graph is doing. Whereas in PyTorch, each and every level of computation can be accessed and peaked at.

PyTorch is similar to NumPy in the way that it manages computations, but has a strong GPU support. NumPy is designed for computations, and not machine learning. It is usually used for machine learning together with a machine learning package. Similarly to NumPy, it also has a C (the programming language) backend, so they are both much faster than native Python libraries. NumPy could be GPU accelerated (with some extra code), but it doesn't have this strong GPU support that PyTorch or TensorFlow do. Finally, PyTorch was specifically tailored for GPU functionality in Python.

TensorFlow, on the other hand, was written mainly in C++ and CUDA (NVIDIA's language for programming GPUs), and was not specifically created for Python. It provides functionalities in C, C++, Java, Go (Google's language), and there is community support for Haskell and Rust. So, with TF, you are not restricted by Python. Even if the syntax differs a bit across languages, the concepts are the same.

Now, PyTorch has deep neural networks functionalities and that is why it is often compared with TensorFlow, sklearn, etc. Moreover, TensorFlow has a peculiar. So, for TensorFlow, you need to make that extra effort. Knowing NumPy, it is easier to switch to PyTorch than TensorFlow, that is why it is gaining popularity so fast.

As TensorFlow was used by Google for so long, it is very easy to deploy algorithms using it. So you can think about it as more product oriented. Logically, you want to be able to deploy the algorithms that you are creating . PyTorch, on the other hand, is more recent, so it does not have the full range of capabilities of other packages.

Let's sum up the advantaged of the PyTorch in comparison with other libraries:

- Dynamic Approach To Graph Computation
- Faster Deep Learning Training Than TensorFlow
- Increased Developer Productivity
- Easier To Learn And Simpler To Code
- Simplicity and transparency
- Easy To Debug
- Data Parallelism

Now, let's start explaining the model and how this is going to work. At this point, let's see how a black and with image works. A black and white image can be represented in grids of pixels. Each pixel has a value that corresponds to its brightness. The values span from 0–255, from black to white. On the other hand, color images consist of three layers: a red layer, a green layer, and a blue layer. This might be counter-intuitive to you. Imagine splitting a green leaf on a white background into the three channels. Intuitively, you might think that the plant is only present in the green layer. Just like black and white images, each layer in a color image has a value from 0–255. The value 0 means that it has no color in this layer. If the value is 0 for all color channels, then the image pixel is black.

As you may know, a neural network creates a relationship between an input value and output value. To be more precise with our colorization task, the network needs to find the traits that link grayscale images with colored ones. In sum, we are searching for the features that link a grid of grayscale values to the three color grids.

Convolutional Neural Networks(CNN) are very similar to ordinary Neural Networks. But instead of connecting all neuron from one layer to a single neuron in the next layer, only a patch of neuron from one layer is connected to a single neuron in the next layer. Its neurons is inspired by the organization of the animal visual cortex. In the below figures we take a look for ordinary Neural Networks and Convolutional Neural Networks(CNN).

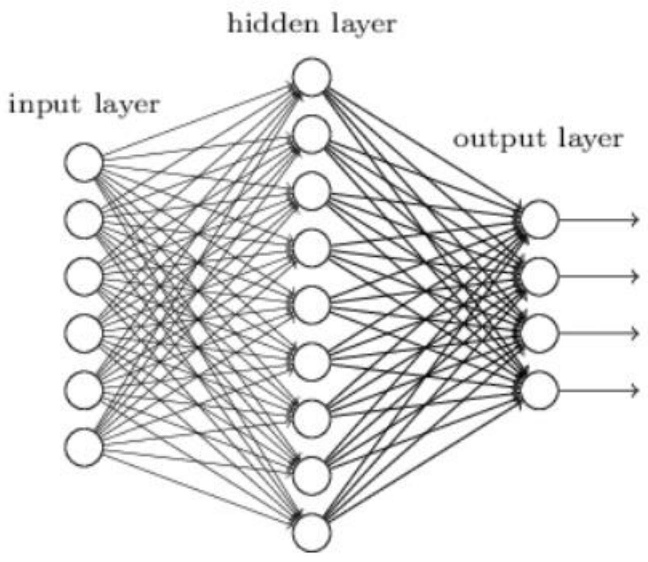


Figure 2: Ordinary neural network

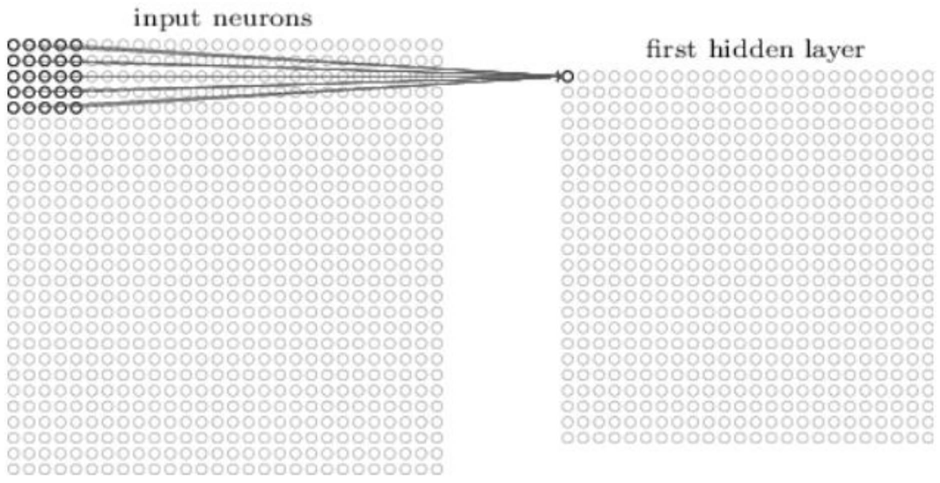


Figure 3: CNN

We can construct a deep neural network with stacked Convolution layers which is called as Deep Convolutional Neural Network. It’s been proof that deeper neural network give much more accurate results compared to shallower networks.

More specifically, Convolutional Neural Networks, or CNNs, were designed to map image data to an output variable.They have proven so effective that they are the go-to method for any type of prediction problem involving image data as an input. The benefit of using CNNs is their ability to develop an internal representation of a two-dimensional image. This allows the model to learn position and scale in variant structures in the data, which is important when working with images.

Image recognition is the process which is used to recognize objects in images like places, writing and actions. Computers use very advanced vision technologies along with a camera and artificial intelligence software for recognizing images. Image recognition has a huge number of visual machine-based applications, such as self-driving cars, accident avoidance systems, guiding autonomous robots, performing image content search, etc. While human brains recognize objects easily, computers have difficulty in recognizing images. Deep machine learning is required by image recognition software. Performance of the software is finest at convolutional neural net processors because otherwise, the task requires very large amount of power due to its compute-intensive nature.

Image colorization is the process of taking an input grayscale (black and white) image and then producing an output colorized image that represents the semantic colors and tones of the input. Previous methods for image colorization are Relied on significant human interaction and annotation Produced desaturated colorization. The new approach we are going to use today is based on deep learning. We will use a Convolutional Neural Network capable of painting black and white images with effects that can even fool people.

More generally, CNNs work well with data that has a spatial relationship. The CNN input is traditionally two-dimensional, a field or matrix, but can also be changed to be one-dimensional, allowing it to develop an internal representation of a one-dimensional sequence. This allows the CNN to be used more generally on other types of data that has a spatial relationship. For example, there is an order relationship between words in a document of text. There is an ordered relationship in the time steps of a time series. Although not specifically developed for non-image data, CNNs achieve state-of-the-art results on problems such as document classification used in sentiment analysis and related problems.

We first apply a number of convolutional layers to extract features from our image, and then we apply deconvolutional layers to upscale (increase the spacial resolution) of our features. The layers not only determine color, but also brightness. Specifically, the beginning of our model will be ResNet-18, an image classification network with 18 layers and residual connections. We will modify the first layer of the network so that it accepts grayscale input rather than colored input, and we will cut it off after the 6th set of layers. So we will create a loss function. This loss function is slightly problematic for colorization due to the multi-modality of the problem. For example, if a gray dress could be red or blue, and our model picks the wrong color, it will be harshly penalized. As a result, our model will usually choose desaturated colors that are less likely to be "very wrong" than bright, vibrant colors.

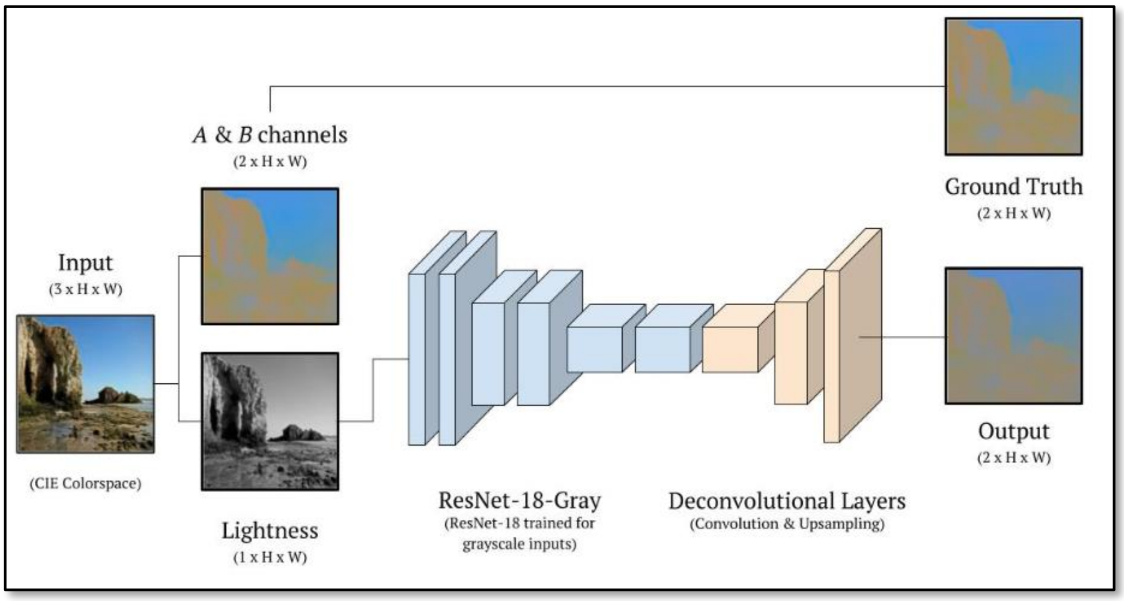


Figure 4: How the model works

So, as we said before we have a training first. We will start by explaining how the model works at the training test. Since we are doing regression, we'll use a mean squared error loss function: we minimize the squared distance between the color value we try to predict, and the true (ground-truth) color value. Mathematically, it is represented as:

$$MSE = \sum_{i=1}^{n} \frac{\left(w^T x(i) - y(i)\right)^2}{n}$$

It is the average of the squared error i.e. it is the sum of the square of the difference between our target variable and the predicted values, over all the data points. This function effectively disciplines larger errors severely because of its formulation. This loss function is slightly problematic for colorization due to the multi-modality of the problem. For example, if a gray dress could be red or blue, and our model picks the wrong color, it will be harshly penalized. As a result, our model will usually choose desaturated colors that are less likely to be "very wrong" than bright, vibrant colors. There has been significant research on this issue, but we will stick to this loss function for this assignment. The next step is to optimize the loss function using an optimizer. For this purpose, we will use the Adam optimizer.

Adam is an adaptive learning rate optimization algorithm that's been designed specifically for training deep neural networks. Adam is an adaptive learning rate method, which means, it computes individual learning rates for different parameters. Its name is derived from adaptive moment estimation, and the reason it's called that is because Adam uses estimations of first and second moments of gradient to adapt the learning rate for each weight of the neural network. Here, it is important to explain what a moment is. N-th moment of a random variable is defined as the expected value of that variable to the power of n. More formally:

$$m_n = E[X^n]$$

m — moment, X -random variable.

The first moment is mean, and the second moment is uncentered variance (meaning we don't subtract the mean during variance calculation). Now, we will list some of the properties of this Adam optimizer.

- Actual step size taken by the Adam in each iteration is approximately bounded the step size hyper-parameter.
- Step size of Adam update rule is invariant to the magnitude of the gradient, which helps a lot when going through areas with tiny gradients.

The code for this optimizer is the following:

```
optimizer = torch.optim.Adam(model.parameters(), lr=1e-2, weight_decay=0.0)
```

4. Results

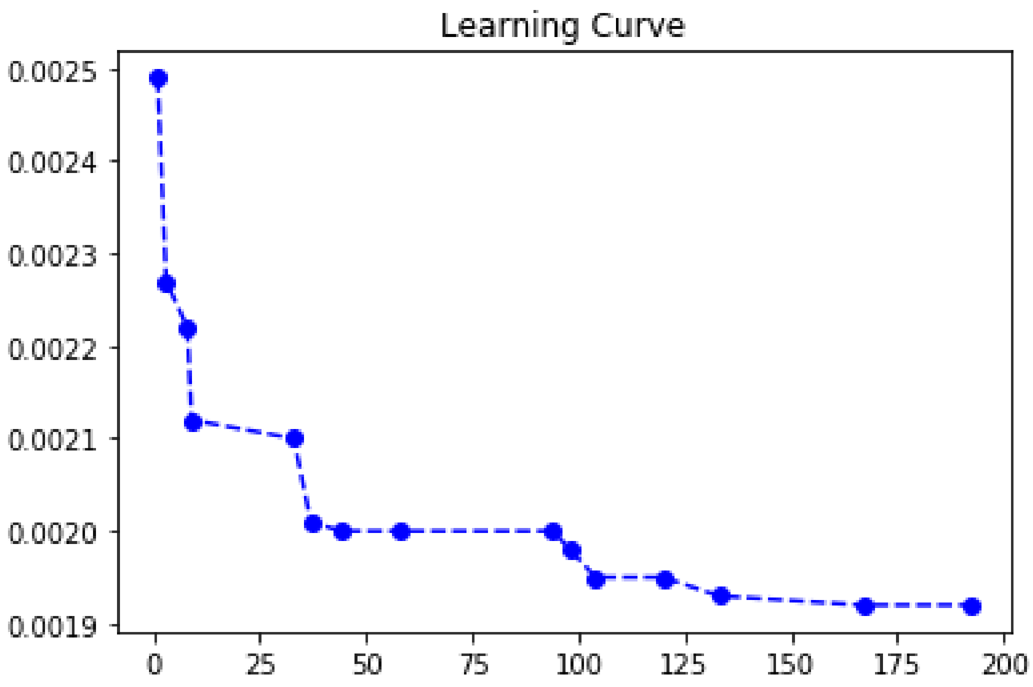
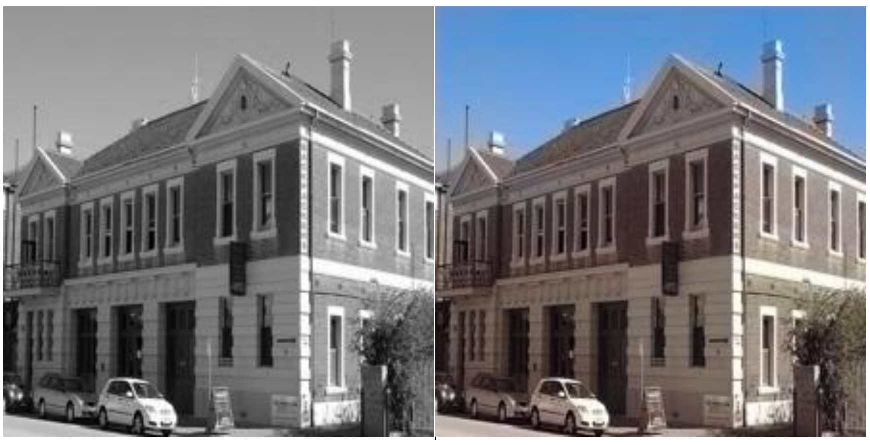
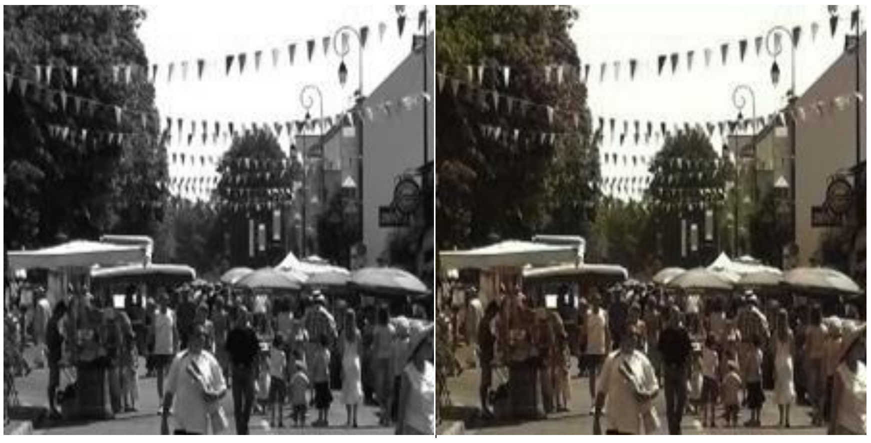


Figure 5: mean square error loss function - epocs

We choose the model of epoch 104 since it had the best results and we didn't want to over train our model. One example of the output of our model is in above.



We observed that in complicated pictures with many different colors in small areas of the picture our model usually fails. As it is shown in figure below although our model has correctly colorize the trees, in the little flags and the people the colorization have been failed.



After a lot of trials the final model has been train for 4 days. The problem that we faced is that colab which was the platform that we used to train our model had limitations in time of using the google cloud. More specifically every 12 hours we have to mount to the last epoch and rerun from this epoch our model.

One major problem was also the choice of the appropriate dataset. It seems that the appropriate dataset is one of the most important decisions in order to have good results. For example the first dataset that we tried was from "MIT places" and contained pictures of various themes such as places, people, dogs etc. We observed that although we had good results for the places we have bad results for the dogs (figure 6). So we decide to add in our dataset a part of dataset of dogs' pictures that we found in "standford vision" and we saw that the results we improved a lot (figure 7).

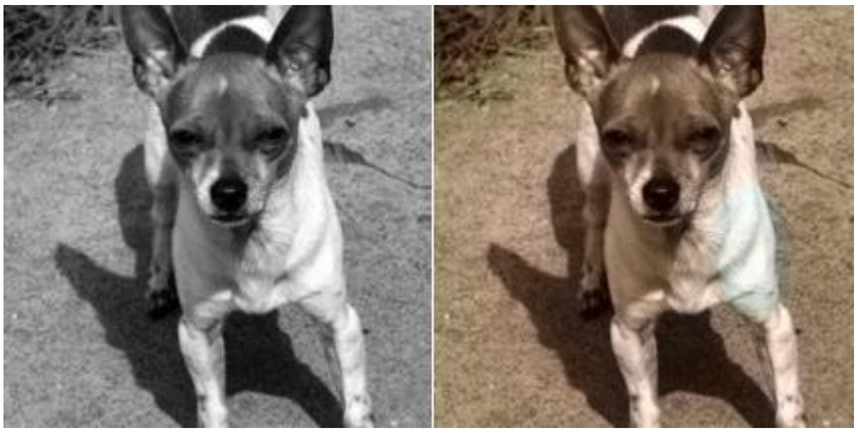


Figure 6: Before and after colorization

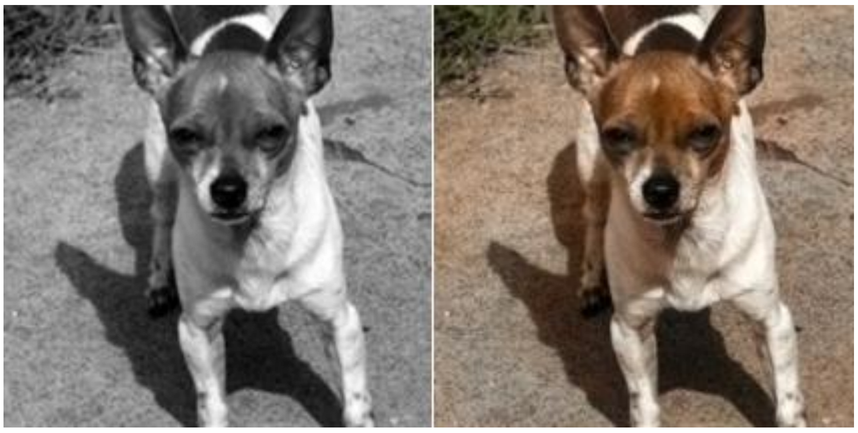


Figure 7: Before and after Colorization with the new mixed dataset

Finally another problem was that people liked more the warm colors even though the colorization was not so accurate. More specifically in a survey that we have made in a sample of 70 people we found that 82% of them chose as more accurate the pictures with warm colors although they were not. For example the left picture in figure 8 was judged to be better colorized than the right although the dog’s left side on it was green. This was a business problem which made us consider if we want to be more accurate in our colorization or if we want to have model that have been trained to use warmer colors.

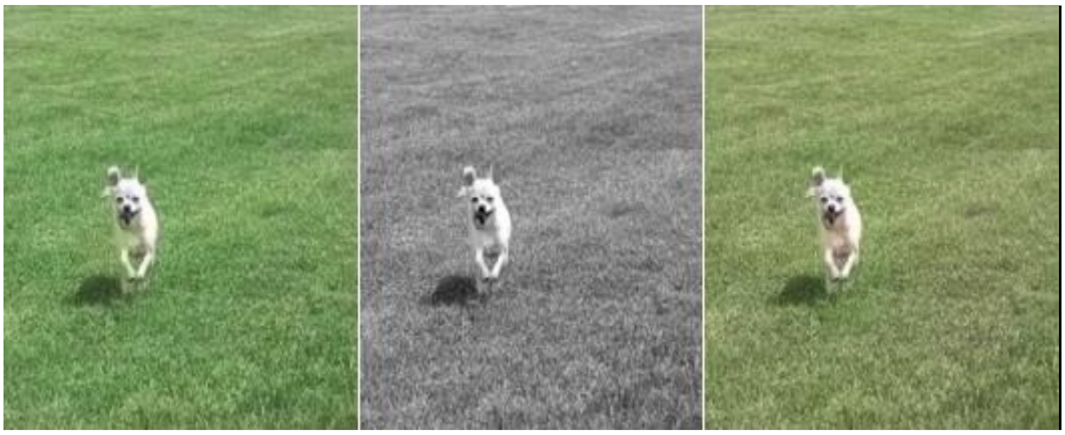


Figure 8: warm color problem.

5. Members

The members of this team are four. This team has a Project Manager. He is responsible to brings the projects to the team. The General Manager goes to the Project Manager and gives him a specific project to analyze and bring results. The other three people are two data scientists and one business analyst. The business analyst is the one who is writing the report and take the results and analyze them in a more business way. The remaining two are data scientists, who are responsible to create the algorithm, make it run properly and bring the results to the business analyst. Finally, the Project manager is the one who controls all the procedures above and judge them if they are correct and what the Manager asked them or not. More specifically, Anastasia Kailani is the Project Manager, Ioanna Kourtesi is the Business Analyst and Spyros Fotopoulos and Dimitris Alexiou are the two Data Scientists.

6. Bibliography

Published Sources

[1] Sudha Narang, Srishti Bansal, Deepali, “Still Image Colorization using CNN”

[2] Mat´ıas Richart, Jorge Visca, Javier Baliosian, “Image Colorization with Neural Networks”

[3] W. Markle and B. Hunt, “Coloring a black and white signal using motion detection,”

[4] R. Zhang, P. Isola, and A. A. Efros, “Colorful image colorization,”

[5] Alex Aranburu, Arun Kumar Giri , “Image Colorization using Convolutional Neural Networks”

Electronic Sources

[6] https://www.wikipedia.org/

[7] https://pytorch.org/

7. Contact Person

Kailani Anastasia, (+30)-6949269487