Image Colorization Using Deep Neural Networks

Alireza Dehghan
alireza.dehghan@studenti.unipd.it

Abstract

Grayscale image colorization is an interesting problem in computer graphics, the goal is to add color data to a monochromatic input image to produce a colorized picture. In this project, the aim is to generate color data for grayscale images using deep neural network trained on COCO dataset. The produced model's efficacy is examined by comparing different hyper parameters and fine tuning them, the results for each experiment are discussed and exhibited along their original representation. The ending discussion considers the pros and cons of the model and ideas on improving it.

1. Introduction

The colorization of grayscale images is a daily-usable and formidable challenge that has garnered substantial scholarly and professional interest within the realm of computer vision and image processing. The task of infusing color and hues into monochromatic pictures has practical implications in medical applications and artistic significance. Deep neural networks, a class of artificial intelligence models, have emerged as potent instruments for tackling this problem. Deep neural networks have revolutionized the field of computer vision by exhibiting exceptional capabilities in understanding and analyzing complex visual data. These networks, composed of multiple layers of interconnected artificial neurons, can learn hierarchical representations of features and patterns directly from data. Their ability to automatically extract high-level features makes them well-suited for tasks such as image colorization. On this project, we have explored the usage of these neural networks to colorize grayscale images with acceptable results which is discussed through the report, our model understands how to color most general things found in images, for example grass or sky, but falls short on specific details, also it gives a brown hue to all the images which is explained thoroughly in section 5 along with the experiments.

2. Related Works

Our project is mostly inspired by the Deep Koalarization paper[1] where the authors have created a model consisting of deep convolutional neural networks and inception models. To achieve this they use a pre-trained, Microsoft ResNet model to account for the inception and train the CNN, which has designed as such to be an encoder-decoder model. In contrast our model does not feature a pre-trained part to exhibit to limit the discussion to neural networks behaviour when training on input and when generating output. Another interesting approach to colorization is to use user generated information to let the model generate informed color data[2], this approach as shown by the authors yield favorable results but the main drawback is that it is not fully automatic and needs human interaction on each input. A quite effective method of colorization is to use reference pictures, this is done in a section of [4] where the authors try to color CT scans of patients lungs by using steak and meat color as references, resulting in favorable output, but this only works on the condition that a good reference image is provided and the subjects to be colorized are not diverse, thus, making it a good choice for CT scan colorization, but not necessarily everyday image colorization.

3. Dataset and Preprocessing

Datasets are one of the most crucial part of training a deep learning model. If the dataset is not diverse enough, the model fails to generalize, if it is small in terms of samples, the model fails to grasp the correct important features needed to create a desirable output. The dataset used for this experiment is Microsoft's COCO (Common Objects in Context) which focuses mostly on segmenting individual object instances[3], though this remarkable feature is not used in this project. Regarding the problem of colorization, any colored picture can be used for the training process, but the reason behind choosing COCO was because it is a diverse dataset with over 100 thousand of images and about 19 gigabytes in size, and the annotation data even though not used here, paves pathways for more future experiments, e.g. informed colorization. To convert the raw images into trainable data for the model, first they were resized into smaller (256 by 256) resolutions to reduce computational complexity and then, the image was converted from RGB color space to CIELAB, that way, the grayscale image (brightness) can be separated from colors without the color layers carrying additional data which happens in RGB space, thus creating our input features (the grayscale image) and desired output (color data). using CIELAB color space turns the RGB values into Lightness (0 ¡ L ¡ 100 where 0 is black and 100 is white) and a, b values which respectively from their lowest to highest values represent green-magenta and blue-yellow colors (a, b are unbounded but in software implementations they are often bounded between -128 ¡ a, b ¡ 127). Finally, all of the images are altered randomly by rotation, zoom and flipping to account for generalization over future input data.

4. Method

The provided images as mentioned in the dataset section, are divided into color and lightness parts out of which, the lightness will be the input and the color will be the desired target given the lightness. The approach undertaken for this project is to use deep neural networks with multiple convolutional layers to train a model adequate enough to generate color data for an arbitrary input. Another common approach is to use additional data about the colors of the image or the objects in pictures, for example an approximate color data can be given to the model as an input alongside the grayscale image by the user. The choice of using the lightness of the image as the standalone feature comes from the fact that we wanted our model to need as little human interaction as possible to generate result, this improves the reliability on discussions on how the model is generating the color data for the input since the human error has been almost neutralized, being only present in the input data. This also allows the model to train on any image file that is presented to it, since using almost primitive software, any RGB color spaced image can be converted into Lab image and used as input, giving it more flexibility. After the preprocessing, the grayscale input image goes through our model which is an encoder-decoder which uses ReLU as the activation function for all the convolutional layers except for the last one, which uses tanh. other activation functions were experimented with but they added almost no considerable difference, but using tanh as the last layer's activation resulted in more visually appealing outputs, also leaky ReLU was prone to over-fitting and coloring everything green. To create the desired image given a grayscale input, first the image is ran through the model and the resulted output is concatenated with the input, giving it two extra channels two exhibit and hence, making it colorful.

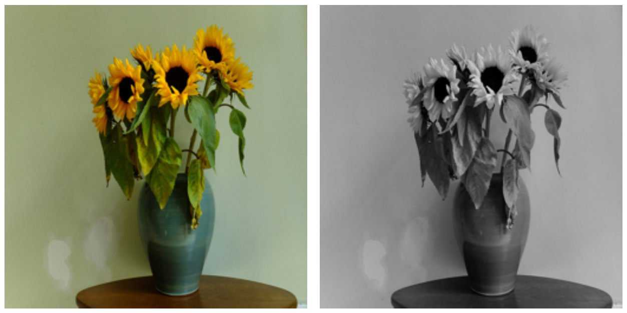
Figure 1. Image in L*a*b color space (left) and Lightness only image (right)

5. Experiments

One of the early versions of the model used 512x512 pictures as input, but that soon proved to be too much computational complexity to be handled in a reasonable time, so it was reduced to 256x256. Another problem that hindered the performance was the fact that there was not enough memory available to keep all the images in ram, so they were all forced to be read from a drive, further slowing down the process, limiting us time wise from trying more training data with more epochs. as seen on Table 1, one of our two best models suffer from not being able to load data directly from ram. The models were all trained with a batch size of 50 and a step size of 2000 per epoch, adam optimizer was used with Mean Squared Error as the loss function. along with adam, rmsprop was used too, but did not exhibit significant performance improvement. Overall multiple models were trained using different activation functions and epochs, most of them showing quite similar characteristics as far as numbers are considered as seen on Table 2 which given the outputs, brings to the attention the fact that accurate colorization is hardly measurable. Going through the results, the varying quality of the colorized image given the input is obvious. For example on Figure 2, a fairly simple image has been tested as an input, with

Device	TrainSize	storage	epochs	Training Time
M1 Pro	10000	RAM	10	6.95 hours
RTX 3070	40000	SSD	1	4.9 hours

Table 1. Training time on different hardware

Model	TrainSize	Activation	epochs	loss
1*	10000	ReLU/tanh	10	0.0129
2	10000	ReLU/tanh	1	0.0131
3	40000	leaky ReLU/tanh	2	0.9765
4*	40000	Relu/tanh	1	0.0123

Table 2. Models trained

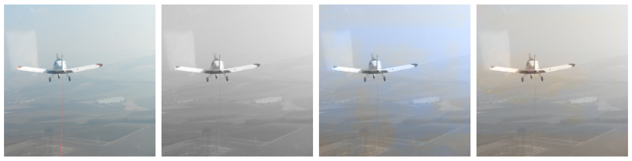
Figure 2. Left to right: Original, Grayscale, Model 1, Model 2

little to color, our two best models output adequate results, with model 1 being more vibrant than the model 2, but its coloring is less uniformed, this trend continues on Figure 3 where the image of the cat, is again colored close to the ground truth on both models, here the generalization of the model 2 shows itself, thus not mistaking the ceiling with the sky. Taking a look at Figure 4, we see another good

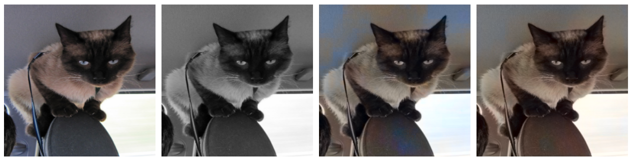
Figure 3. Left to right: Original, Grayscale, Model 1, Model 2

example of colorization, the scene does not look complicated and both models clearly have recognized the vegetation from their training and reasonably colorized it, but a yellow hue can be seen on the output of model 2. Of course

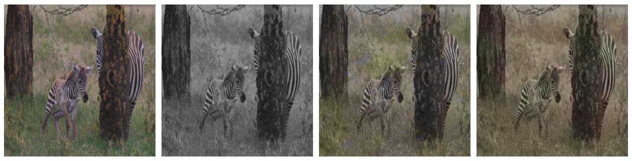
Figure 4. Left to right: Original, Grayscale, Model 1, Model 2

the previous outputs were cherry picked and when it comes to lesser seen objects or complex scenes, the models cannot function like they do as seen of Figure 5, where colorful, small fruits prove to be quite a burden for the model to colorize correctly, with both models applying a yellow tint to the images Also Figure 6 is a good exhibition of the fact that

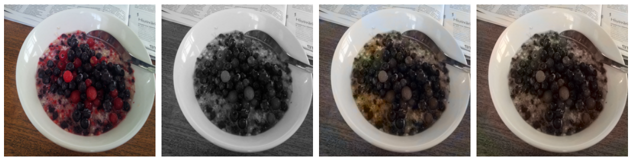
Figure 5. Left to right: Original, Grayscale, Model 1, Model 2

the neural networks, unaware of the information about the ground truth, will work purely based on their training and statistically measuring their deviation from the ground truth might not be always desirable. again in this picture, Model 2 has produced an image with yellow tint, while model 1 has more vibrant colors. The yellow tint seen on almost every output of the model 2 is the result of generalization and demonstrates an intrinsic behaviour of the neural networks where they try to minimize loss when training by finding a middle ground, averaging every picture's colors which results in a tint on almost all pictures. The vibrancy of the first model though is susceptibly the result of better learning process, focusing on fewer objects (because of the lower number of training input) and higher accuracy to detect and color them courtesy of higher epochs. Figure 7 perfectly

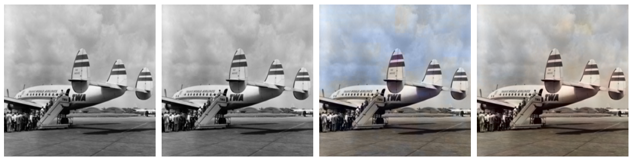
Figure 6. Left to right: Original, Grayscale, Model 1, Model 2

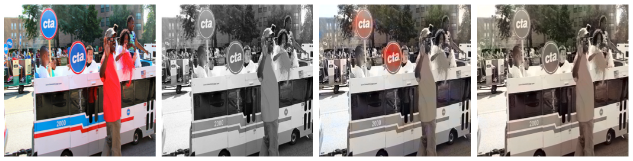
Figure 7. Left to right: Original, Grayscale, Model 1, Model 2

signifies the shortcomings of both models where they fail to colorize people effectively, given the clutter, occlusion and the overall complexity of the image. Still the issue of yellow tint and lack of contrast stands for the second model but one curious phenomenon that has happened is the sign that was most probably recognized by the first model to be a stop sign, colored almost perfectly, even though the ground truth is far different than a stop sign.

6. Conclusion and future works

In conclusion, we produced a fairly adequate image colorization model based on deep neural networks that shows moderate success in colorizing images with no additional information provided and is able to generate fair results conditioned on the fact that the image has few objects of interests and those objects are quite common. Given our result, it is almost certain that having additional information about objects regions or segment types, could give the model some informed direction to what color it should use and where exactly on the image needs to be colorized, which could be the future direction that colorization could go, also interactive coloring can be another good solution, but the training requires human time and effort which is a drawback.

References

[1] Lucas Rodes-Guirao Federico Baldassarre, Diego Gonzalez-Morin. Deep-koalarization: Image colorization using cnns and inception-resnet-v2. *ArXiv:1712.03400*, Dec. 2017.

[2] Michael Maire Gustav Larsson and Gregory Shakhnarovich. Real-time user-guided image colorization with learned deep priors. *ArXiv:1603.06668*, 2017.

[3] Serge Belongie-Lubomir Bourdev Ross Girshick James Hays Pietro Perona Deva Ramanan C. Lawrence Zitnick Piotr Dollár Tsung-Yi Lin, Michael Maire. Microsoft coco: Common objects in context, 2015.

[4] Wei Qi Yan Yuewei Wang. Colorizing grayscale ct images of human lungs using deep learning methods. *Multimedia Tools and Applications*, page 37819, 2022.