U-Net Vivid Picture Colorization Using HSV Color Space

Darío Caballero, Leo Joly
The Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
dcp@connect.ust.hk, lvajoly@connect.ust.hk

Abstract

As fields of computer vision become more and more precise and able to obtain complex and good results with pre-trained deep learning models, the use cases and the importance of interpreting images and videos grow more and more in our society. In this paper, we are going to propose a real application of image colorization using a U-net model. Based on previous papers, we believe than the results can be better using a specific loss function and modified model architecture. Using the famous ImageNet dataset, we can convert any colored picture into HSV color space and keep the V channel to get the grayscale version. Our model then needs to learn how to predict the other 2 color channels under supervised learning.

1. Introduction

Image colorization is challenging because it requires the model to understand complex context situation such as cloudy days or night light color. Moreover, a lot of variations are possible for one same object or animal and it is up to the model to interpret which variation to use based on the environment. see Figure 1.
The process of re-colorization can find it's utility in many field such as low light enhancement and could give a way to store or capture images only in gray shape, which is way less space consuming and can give a second life to old black and white pictures.

Our solution implement a better learning loop including the calculation and monitoring of the vividness of the picture. We are now able to establish a loss from this new parameter and make our model learn how to interpret this field and hopefully get pictures with more quality than previous U-net colorization algorithm.

We decided to split our project into different steps. Firstly because of our low access to good calculation power and to try different angle to solve the vividness problem. To have a better understanding of it we decided to start by implementing a very famous model to process image and perform segmentation called U-Net. Our 4 level U-Net model needed to be totally constructed from scratch and trained from the beginning on how to decode and understand a HSV color space picture because of it is substantially different from the standard CIELAB color space used for colorization.

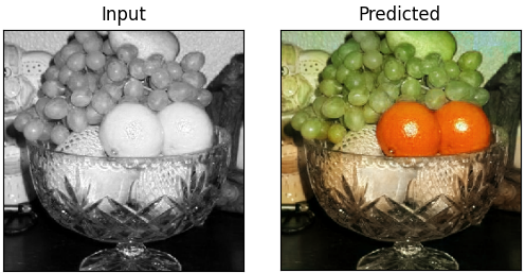

Figure 1. Example of colorization

Our paper is gonna demonstrate the different results obtained by theses experiments and how we decided to evolve the model and it's environment to obtain better results. 7
Our model and whole code is available on our Github : https://github.com/Dario-CP/comp4471_final_project

2. Related work

Most papers on colorization models use the CIELAB color space by feeding the L (lightness) channel and predicting the A (red-green) and B (blue-yellow) channels like this paper [1]. This is an advantage over RGB color space, since it allows them to just need to predict two channels, being one of the three channels in the final image the same they use as an input.

This drastically reduces the margin of error of the model, not only because you predict one less channel, but because one of the final channels has no error at all, since it is the input one. Seeing this advantages, we knew that we had to discard the RGB color space as following to theses example articles [3, 5]

3. Data

We used COCO dataset [2] for a preliminary experiment. This dataset contains 123,287 images and was a good starting dataset to test our U-Net model and modified loss function that we are experimenting with.

After the first results we saw the importance of having a bigger dataset for image colorization with the U-net model and we decided to learn our model on a bigger one. To get a more robust and complete dataset, we are using ImageNet (2010) dataset, which contain more than 1.3 millions pictures for the training set and 300 thousand for the test set [4].

The idea is to convert every pictures into HSV color space. We are then able to input the model with only the V channel and rebuild a new fully generated color image with the 2 others channels predicted by the model.

It is important to note that others papers use the CIELAB color space, which differs from ours, because of its more natural and accurate representation of colors. We chose HSV color space for the accessibility of the saturation channel which directly impacts the vividness of the picture. We will describe more about the method and what results we expect in the next part.

4. Methods

In general, colorization models get less vivid colors, so we propose two slightly modified loss functions that address this problem. In the first one, we introduce the use of a parameter K calculated to indicate the saturation of the current picture. In the second one, we simply arbitrarily weight the saturation channel.
We expect this addition to allow the model to obtain a more vividly colored picture, while still being able to retrieve the original colors. We designed a U-net model Figure 2 that takes an input grey scale image 256x256 and outputs a colored image with the same scale and with the 3 channels in HSV color space. The U-Net model was chosen due to its great segmentation capabilities, useful when colorizing different objects of a picture.

We wanted more control of the saturation of the images generated. In that point the CIELAB color space really suffers, since there is no easy way of controlling the saturation while maintaining the hue and lightness. That is the reason why we are using HSV (see [6]). It divides the images in 3 channels: H (hue), S (saturation) and V (value) see Figure 3. The V channel is really similar to the L channel in CIELAB, with the difference being its scale. Thus, we can mimic the technique of feeding the V channel and predict the other two. However, in this case we have the advantage of having an easy access to the saturation of the pixels.

4.1. Technical Approach

We used a U-net model with 4 levels of encoders/decoders and one bottom layer. The U-net model uses concatenation to keep the results from every encoding layers to be able to better reconstruct the image. Finally, the model condenses the output in 2 color channels and concatenates the grey scale channel to form the colored picture as output. see Figure 2.

Our complete model has $31,030,658$ trainable parameters, natively working on 256x256 images. This amount of parameters require a large amount of data to not overfit and a quite powerful GPU to learn faster, which we did not have access to. The training was performed on a x64 laptop with an NVIDIA GeForce RTX 2080 with Max-Q Design, 8 GB GDDR6 VRAM and 2944 CUDA cores as GPU, an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 2592 Mhz, 6 cores, 12 threads as CPU, and 32 GB of RAM. The amount of VRAM was problematic, since when training on batches of 32 images, it would sometimes run out. That made us have to work on batches of 16 images.

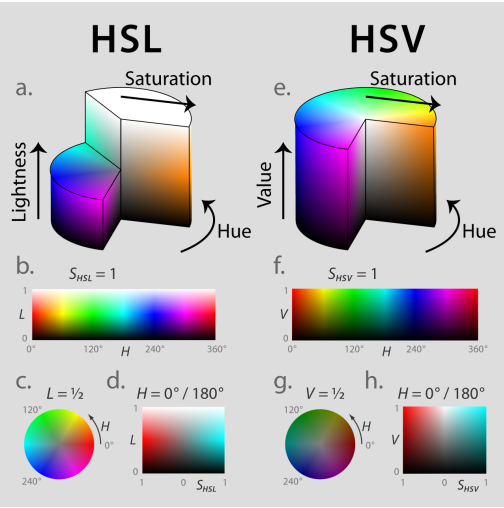

Figure 3. HSL and HSV color spaces

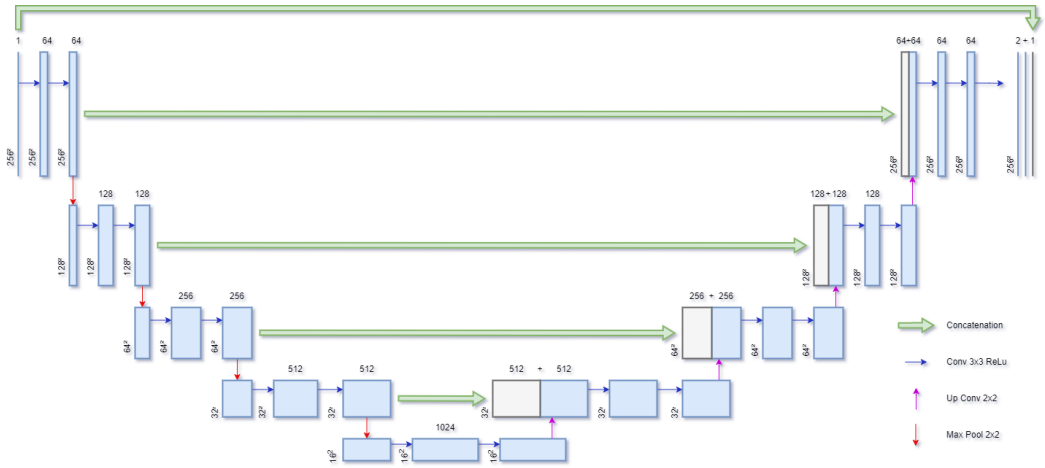

Figure 2. U-net model layers diagram

This is the first proposed loss function:

$$\mathcal{L} = \left[\frac{1}{H \times W \times C} \sum^{H} \sum^{W} \sum^{C} |y - \hat{y}| \right] + \lambda K \quad (1)$$

$$K = \frac{1}{H \times W} \sum_{i}^{H} \sum_{j}^{W} (1 - S_{i,j}) \quad (2)$$

with $S_{i,j}$ being the saturation channel value of a specific pixel of the image.

The value of the parameter K is larger whenever the image is overall less saturated. We expect than minimizing this loss makes the images more vivid. It should be noted that the parameter K is multiplied by a hyper-parameter λ, used to control its effect.

This is the second proposed loss function:

$$\mathcal{L} = \frac{1}{3} \left[MAE(H) + \lambda \times MAE(S) + MAE(V) \right] \quad (3)$$

with H, S, V referring to the Hue, Saturation and Value channels, respectively.

This one works by giving more importance to the saturation channel when backpropagating.

5. Experiments

We started by training the model on the much smaller COCO dataset. We found that the model was rapidly over fitting over just few epochs and that we needed a bigger dataset. No matter the number of epochs that we trained, the validation loss would not decrease That is why we changed to ImageNet However, the selection of loss function was performed based on the experiments on the COCO dataset.

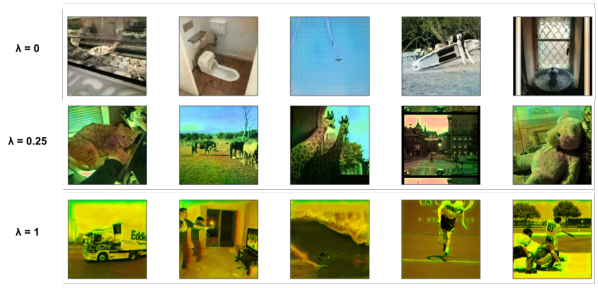

Figure 4. Results obtained with different lambda using the first loss

Using the first loss, we can see that our approach of increasing the learning of the saturation channel seems to work to a certain level and that the model is very sensible to this parameters, since we can observe an huge modification of the results with just $\lambda = 0.25$ (see figure 4). However, the effect of this loss does not reflect our goal of achieving realistically a high saturation while maintaining the correct hue. The overall effect seems as if the images were painted with highlighter pens, which is far from optimal. This is because the first loss does not decrease the saturation penalty when it reaches a reasonable level, and incentives the model to just increase it to the maximum. In addition, even with a reasonable lambda, the K term tends to overweight the Hue loss, so the model prioritises increasing the saturation to the maximum over obtaining a good hue.

This flaws were addressed with the second loss. With it, the model learns faster the saturation of the images, but cannot exceed it infinitely. Since we multiply the MAE of the saturation channel by lambda, the effect of the hyper-parameter will vanish once the saturation error decreases, approaching the real values. Since the saturation cannot increase indefinitely, the hue is not overlooked and more precise colors are obtained. For our final model, we used a lambda of 1.15 to train the first four epochs, and a lambda of 1.0 to train the last two, adding for a total of 6 epochs. We decreased the lambda when we arrived at a point in which we thought that the saturation was already good enough, but the hue was slightly behind.

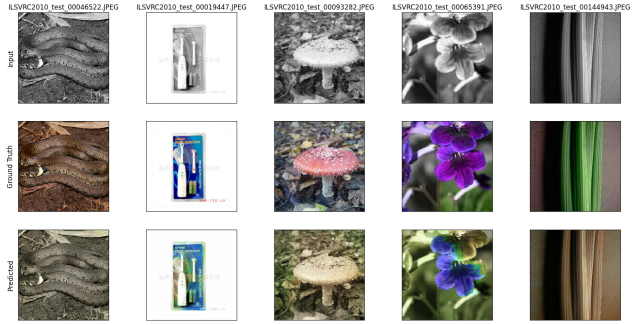

Figure 5. Example of results at 2 epochs of training

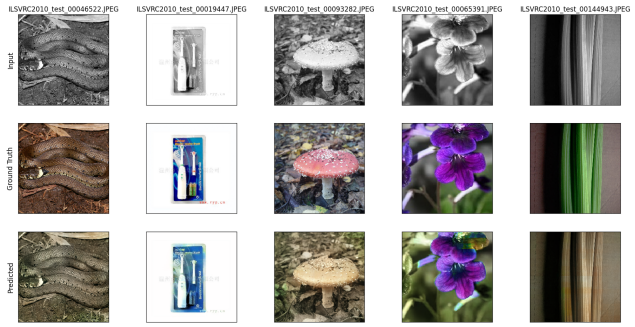

Figure 6. Example of results at 6 epochs of training

In Figures 5 and 6 we can observe the difference between the performance at the second and at the sixth epoch, respectively. The improvement is significant, especially in the second and forth pictures. The blue toothbrush package has a much more uniform color, with a drastic decrease in artifacts. For the purple flower, its color got a lot more saturated and the hue was corrected from blue to purple. Nonetheless, some artifacts can still be spotted on the top part of the flower after the six epochs. The changes in the rest of images are less perceptible, but the snake and mushroom pictures are more saturated. In the case of the last image, we got a worse result after training more. The model was not able to colorize it, and more artifacts were added. In Figures **??** and 11, the evolution from four to six epoch can be observed. The main change being the artifacts reduction in the metronome image.

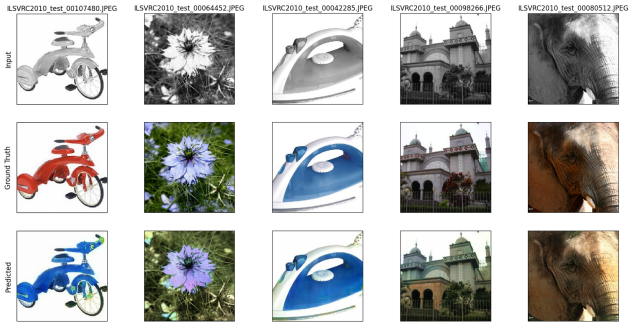

Figure 7. Example 1 of results on ImageNet dataset

In the results we can see some very good interpretations such as the pepper 13 which is mostly one color and quite easy to learn, but also the human faces which have a very realistic color. We can see some surprisingly well colored results on flowers mainly with beautiful tones and even more saturation than in the ground truth.

We noticed that our model has a preference towards blue and green colors, as we can see on the bike picture 7. We may explain this trend by the overall predominance of theses colors in the training dataset. This could also be caused by the cyclic nature of the Hue channel, divided exactly at pure red. This means that the red color is at Hue level 0, but reddish colors are both at values close to 0 and values close to 1 (see Figure 3). This division in the hue gradient could result in an increased difficulty to represent red colors, since really similar hues of red would have extremely opposite values in the range $[0, 1]$. This could maybe be fixed by introducing a module operation in the last layer of the network for the Hue channel, just before calculating the loss.

While some pictures are so well colorized that could pass as the original image, there are also bad examples. We can see 3 main types of bad results :

- Mix of colors on objects that should only have one (see the shirt of the man in the first picture of Figure 14)
- Loss of depth effect because of the grey-scale channel loosing the information (see the carved pumpkin in Figure 13).
- Artifacts in the Hue channel. (See Figure 9)
- Artifacts in Saturation channel (see olives picture in Figure 15).

For the first aspect, we suppose that the model still needs more training. Due to the size of the database and the large number of parameters in our network, each epoch takes 12 hours to train, which only allowed us to train the final model for 6 epochs. In Figure 8 we can observe that the validation loss was always decreasing with the training loss, even being lower than it, except on the last epoch. This suggests that the model could still improve with more training. Please note that the loss graph lacks information for the first epoch, since the data was lost. For the second problem, this is due to the HSV color space Value channel specification. It would be interesting to learn our model on a different color space like HSL or LAB. Finally, we try to focus our efforts on the artifacts phenomena seen in Figure 9.

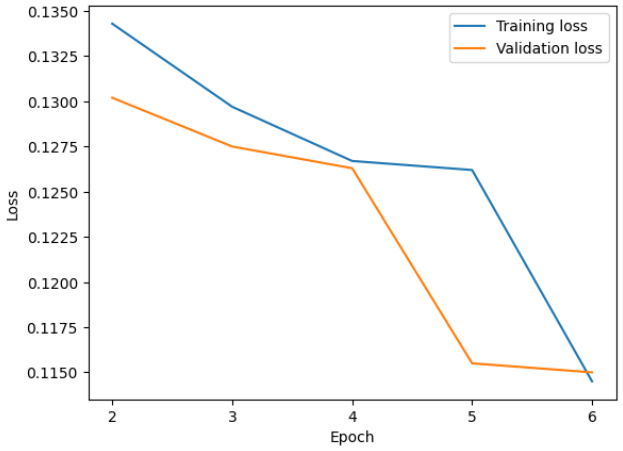

Figure 8. Training and validation loss over epochs 2 to 6

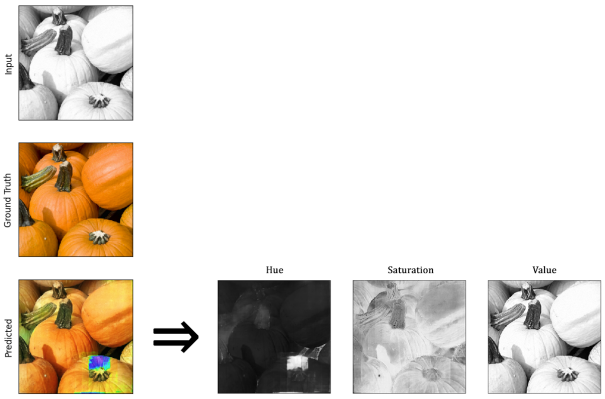

Figure 9. Channel separation for Hue artifact visualization

This artifacts could come form one of the first skip connections of the U-Net. However, we did not have the computational power to test it on time. In order to solve the artifacts problem, we tried to freeze the model and add an encoder-decoder module at the end that would act in a similar way as a Stacked Denoising Autoencoder does, taking the artifacts as noise added to the original images and trying to restore the denoised ones. However, even though the trainable parameters were reduced to $42,598$, a 99.86% reduction with respect to the U-Net, the huge size of the ImageNet dataset acted as a bottleneck, meaning that the training time for each epoch was of 5 hours. After training for 5 epochs, the results were not great, but we do not discard that this approach would give good results if further trained.

6. Conclusion

Our main goal of producing correctly saturated images was achieved in some cases, while in others we did not fulfill it. We think that the easiest way to improve our model would be to train it on the HSL color space, since the Lightness channel better represents the perception of light and is more similar to real black and white pictures. In contrast, the Value channel looks like an over-exposed black and white image, which results in a significant loss in color and depth detail. As previously stated, we would also like to research on the architecture of the U-Net itself, to try to pinpoint the origin of the artifacts on certain images.

We also expect that treating the Hue channel as a cyclic data increases the color understanding of the model and thus, reduces the randomness in some images and boosts the red presence. The final encoder-decoder post-processing layers are also a research path that we would like to explore. However, if we had to choose two main ideas, those would be the change to HSL color space and the cyclic properties of the Hue channel, since they could greatly improve the performance of the model without needing to change the architecture. That would also allow us to transfer the current learning of our model to that modified one, since HSL and HSV color spaces are really similar, sharing two of the channels. Training from that point would probably lead to a faster convergence of the loss function compared to a random wight initialization.

This project introduced us into the different color spaces and the benefits and flaws of each one of them. It was an exciting journey into a interesting and novel field for us, that being the machine learning based image colorization techniques and models. It also made us work with a large dataset of over 150 GB of images and face the problems that arise from that fact. An example of that would be the existence of a corrupted file in the dataset that halted our training. We had to program exception handling methods in our data generator to avoid that.

7. Annexes

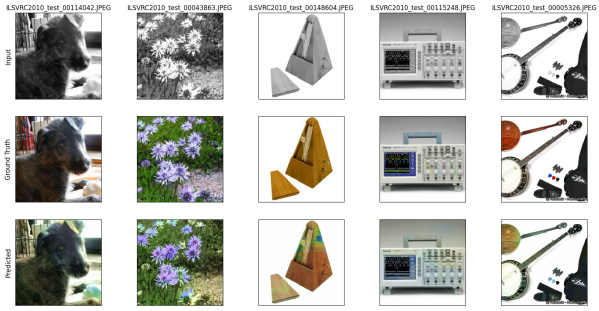

Figure 10. Example of results at 4 epochs of training

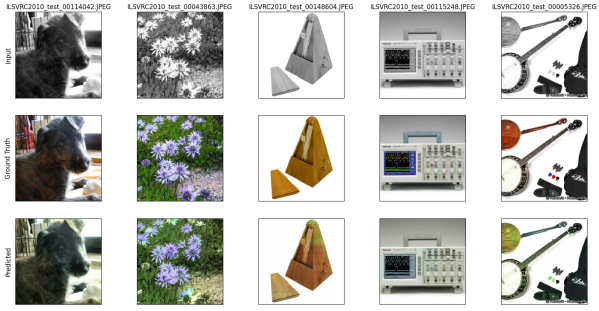

Figure 11. Example of results at 6 epochs of training

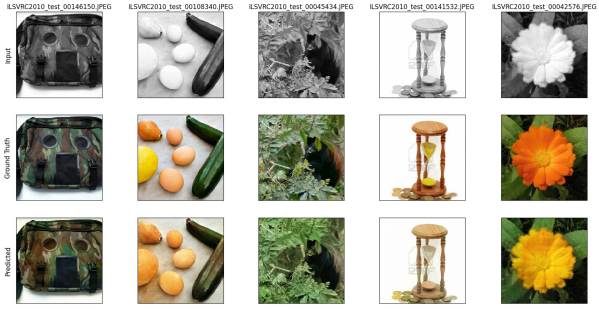

Figure 12. Example 2 of results on ImageNet dataset

Figure 13. Example 3 of results on ImageNet dataset

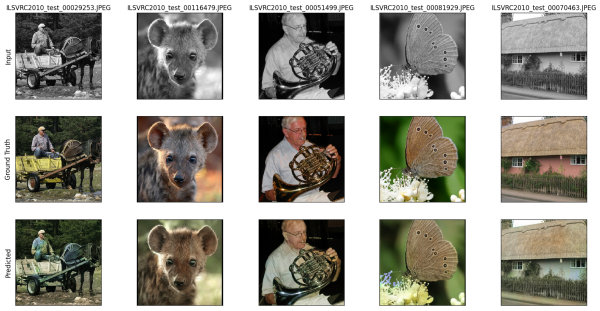

Figure 14. Example 4 of results on ImageNet dataset

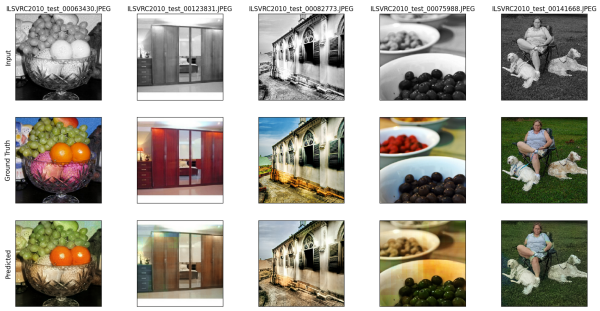

Figure 15. Example 5 of results on ImageNet dataset

References

[1] Divyansh Goel, Sakshi Jain, Dinesh Kumar Vishwakarma, and Aryan Bansal. Automatic image colorization using u-net. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 01–07, 2021. 1

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. 2

[3] Christopher Thomas BSc Hons. MIAP. U-Net deep learning colourisation of greyscale images, 2019. 2

[4] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 2

[5] Moein Shariatnia. Colorizing black and white images with U-Net and conditional GAN, 2020. 2

[6] Wikipedia. HSV and HSL comparisons, 2022. 2