Project Report: CS 7643 - Coloring is all you need

Faisal A. Adlukhi
Georgia Institute of Technology
North Ave NW, Atlanta, GA 30332, USA
faldukhi3@gatech.edu

Josip Franic
Georgia Institute of Technology
North Ave NW, Atlanta, GA 30332, USA
jfranic3@gatech.edu

Aleksandr Shirkhanyan
Georgia Institute of Technology
North Ave NW, Atlanta, GA 30332, USA
ashirkhanyan3@gatech.edu

Abstract

As a result of tremendous advancements in the field of machine learning over the last few years, CNN-based algorithms for image colorization have gradually squeezed out traditional 'hard-coding' techniques. To contribute to this evolving research domain, this study explores possible improvements to the existing methodology. Specifically, we propose and evaluate four modifications to the compound CNN model introduced by the group of researchers from the KTH Royal Institute of Technology. The experiments showed that Feature Extractor based on Inception-v3 and ConvNeXt architectures deliver results comparable to those from the baseline (which utilizes Inception-ResNet-v2). What is more, we found that the model does not lose on performance even when the contribution of Feature Extractor is reduced from 1000 to 256 dimensions. Nevertheless, more research is needed to understand whether and in what way does the Feature Extractor help in image colorization.

1. Introduction

The early history of photography and filming can be best labeled as grayscale era, given that the images at that time were composed solely of the different shades of gray [6, 8]. Luckily, this has changed soon after the World War II, with the advent of machines able to capture scenes in RGB format. Nevertheless, black-and-white photos have remained with us until today, not only as precious memorabilia of our ancestors and important historical moments but also as constituent parts of certain systems (e.g., surveillance cameras).

Along with the eternal desire of historians and restorers to revive original colors, recent advancements in machine learning brought to light some other benefits of the 'colorization task'. For instance, it was found that object-detection systems work much better on RGB inputs [3, 10, 5]. Likewise, interpretative automated systems can provide more information on the input if the colors are clearly discernable [7]. Strong aspiration to understand and contribute to this exciting field was, in fact, the main reason why we decided to tackle the colorization task in this project [1].

Before describing what we did in this respect, it is, however, important to first delineate the current state of affairs in this research field. Traditionally, grayscale images were colorized either following the so-called 'scribble-based' approach [6, 8] or the 'example-based' approach [2, 4]. The former not only requires considerable input from the user but is also time-consuming, which makes it quite inefficient [3]. The example-based methods, on the other hand, stipulate the existence of reference images that are used for transferring the color information to the target images.

The rescue has come with the florescence of deep learning. The notable pioneering attempts to apply neural networks to this task were done by Cheng *et al.* [3] and Zhang *et al.* [6]. These studies relied on something that can be best seen as a self-supervision, given that grayscale images were generated directly from the original color images. This was done by transforming colored images from RGB color space to CIE Lab [10] or YUV color space [3]. These spaces have one luminance channel and two-color channels. The former is used as the input, while the latter becomes the target of the model.

[1] In the project proposal we expressed the intention to deal with image colorization and image super-resolution in a single pipeline. However, this proved to be highly demanding and resource consuming task. After a number of failures to construct a model which would deliver satisfactory results, the decision was made to focus on one task only. Since we found it more challenging and more interesting, image colorization was the preferred option.

However, despite representing a substantial step forward, these early end-to-end methods required large amounts of training data and took a long training time to achieve satisfactory results. This limitation was partially addressed by Dahl [5] who applied transfer learning in order to speed up the learning process. While his model did have much lower training cost and generally achieved outstanding results, in certain cases the colorized photos were sepia-toned and muted in color [7]. Hwang and Zhou, who blamed Dahl's regression-based approach for this, proposed the solution in the form of classification-based learning. This was done by discretizing the continuous data via a binning function [7].

To further simplify the learning and increase accuracy, recent attempts to solve this task took the best of the existing approaches. A noteworthy example is the study by Baldassarre *et al.* [1], who combined transfer learning and 'learning from scratch' into a single end-to-end pipeline. More precisely, they combined the outcomes of encoder with those from pre-trained lnception-Resnet-v2 model, which resulted in more realistic coloring. Since this architecture proved to be suitable for many image colorization applications, we decided to start from this model. Above all, we were guided by the desire to address key limitation of its, namely the limited ability to correctly sort out objects that can take multiple colors (e.g., apples, t-shirts, and walls).

The ability to do such conversion would have an impact on a plethora of applications. First, an individual will be able to restore (or bring back to life) old memories that were stored or taken in a black and white format. Another application for this is to bring old photography with strong impacts back to life, which is a dream for many historians.

The data used in this project was taken from Places-Extra69 [11] with more than 100K images and 69 scene categories. The reason for choosing this data set in particular was to find a reasonable balance between size of the data and the limited computational resources.

2. Approach

In its core, the network by Baldassarre *et al.* [1] consists of four different segments (see Figure 1). The first is the Encoder, which is just a standard deep convolutional neural network (CNN) that compresses the relevant information from the original grayscale image. In parallel with the Encoder, every image from the training set is also processed by the Feature Extractor. As mentioned, the authors used a pretrained Inception-ResNet-v2 network for this purpose, and hence no training was required for this segment. The outputs of the Encoder and Feature Extractor are then concatenated elementwise. The output of this 'Fusion Layer' (which is also not trainable) is then passed through a single convolutional layer to prepare it for the Decoder. Decoder is a separate CNN that up-samples the received tensor and

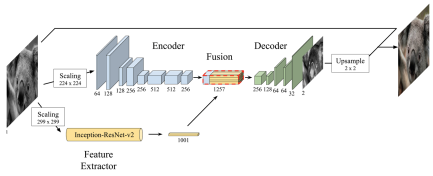

Figure 1. Architecture of the model by Baldassarre *et al.* [1]

produces a 2-channel image with the dimensions identical to those from the input. At this point, it is also important to mention that the input is just a L-channel of an image resulting from the translation of the original RGB image to the CIE Lab color space.

A detailed description of the architecture (henceforth 'baseline') is given in the original paper, so we do not present it here. It suffices to say that our architecture and overall approach closely matched the baseline, except in a few import details.

By far the most important of these details relates to the dataset used. In fact, data storage and handling represented the biggest obstacle for the execution of this project. Since we did not have enough memory and computational power to handle ImageNet dataset, which was used to train the baseline model, we were forced to find a convenient alternative. After examining a vast universe of available datasets, the decision was made to proceed with Places-Extra69 [11]. Available as part of the Places dataset provided by the Massachusetts Institute of Technology, this dataset contains 98,721 training images and 6,600 test images of 69 scene categories. The diversity of objects presented in individual images within this dataset was the main reason why we preferred it over those that contain substantially lower number of categories (e.g., COCO). On the other hand, the original Places dataset was too large for our machines and external memory (as it comprises more than 10 million images), which forced us to rely on this 'smaller version'.

Since the original dataset does not have validation and the split between training and testing is skewed (only 6% of the data was testing), we combined the images to a larger dataset and did the splitting ourselves. Specifically, the dataset was randomly divided in three parts following 80:10:10 split ratio while ensuring we have a representative sample from each category. As a result, 84,313 images were assigned to the training set, 10,510 to the validation set and 10,500 to the test set.

The second important difference relates to the Feature Extractor. Since PyTorch, which was used for execution of this project, does not have Inception-ResNet-v2 network readily available, we decided to replace it with Inception-v3. The main reason for this choice can be found in the compatibility of the inputs and outputs between these two

architectures.

However, to test the effect of this decision on the model performance, we also wanted to explore other options. The initial idea was to use the Vision Transformer for this purpose, given that these models are considered as a breakthrough in computer vision applications. However, when we tried to get the high-level features of images during the training, the (premium) GPU went out of memory. One solution was to use a smaller model of ConvNet that bore resemblance to the Vision Transformer. This brought us to the ConvNeXt, an architecture developed several months ago which competes well with the Transformers [9]. This model modernizes the standard Resnet and makes it similar to the hierarchical Vision Transformer in terms of its design [9]. Several configurations of this model were available in PyTorch. We used the smallest one called ConvNeXt-T. Its complexity is comparable to that of Inception-v3 as it has 29mln parameters (the former has 24mln parameters).

Since these two architectures gave comparable results, we became curious to understand the exact role of the Feature Extractor. For this reason, we first tried to completely remove this segment. However, since the results were far from satisfying, this option was eventually discarded.

Instead, we tried to see what happens if the number of dimensions the Feature Extractor contributes to the Fusion Layer is reduced. Since the Feature Extractor is responsible for 1,000 dimensions, compared to only 256 contributed by the Encoder, it is logical to assume that precisely this imbalance could be responsible for the limited ability of the baseline to handle objects that can come in more than one color. For this reason, in our third model we took a random sample of 256 dimension from the Future Extractor, so as to equalize the contribution of the two segments. Inception-v3 was used for this purpose.

The same is true for the fourth model, whose purpose was to explore the effect of loss function on the performance. All three previously described models were trained with the MSE loss that compared values of the channels a and b pertaining to the ground-truth image with those from the 2-channel output of the model. Hwang and Zhou [7], on the other hand, managed to increase performance on this task (although using different architecture) by shifting to a modified version of cross-entropy loss. To see whether this 'pixel classification' approach would also work in our case, the last model was trained with the loss introduced in the accompanying paper [7].

Turning to implementation details, we used Adam as an optimizer in all four cases. Considering the training time and computational complexity, the downward pattern of the training and validation losses, as well as their distance from each other (see next section), we set the initial learning rate equal to 0.001, and decreased it with a rate of 0.5 every 2 epochs. All models were trained for 20 epochs with the batch size of 64. Finally, all other hyperparameters were identical to those used by Baldassarre *et al.* [1].

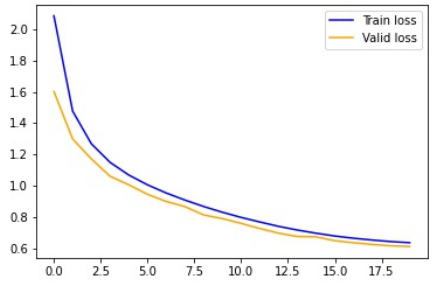

Figure 2. Loss function for training and validation set by epoch, Model 1 (Inception-v3) with the x-axis showing the number of epochs and the y-axis showing the MSE loss

3. Experiments and Results

We ran four different experiments with Inceptions-v3 as the Feature Extractor, the ConvNeXt-T as the Feature Extractor, Inceptions-v3 as the Feature Fxtractor but with less representation in the Decoder, and lastly Inception-v3 with cross-entropy as the loss.

For all experiments, the loss was decreasing substantially, although it did not seem to converge for all (see Figures 2-5). Specifically, Model 4 with its cross-entropy loss was lagging behind other models in terms of performance and difference between training and validation.

Regarding other models, Model 1 takes more iterations than Model 2,3 to converge, although it converges to a low loss later. Model 2 with its transformer-inspired architecture (ConvNeX-T) converged very quickly and achieve an excellent performance that is similar to Model 1. For Model 3, reducing the Feature Extractor in the Fusion layer helped a lot in making the converging faster for Inception-v3 although the final performance was slightly worse than the original.

Based on the learning curves, it seems that the hyperparameters were set correctly because both the validation and training are going down. Since the validation loss is still lower than the training loss, we could have increased the number of epochs to ensure full convergence. However, this was not a possibility given the limited computational resources.

Based on Figure 6, there are few observations one can make. First, most of the experiments failed with the red color. It was hard to capture it with these unseen objects. Second, when using the full Inception-v3, the Feature Extractor embeds high-level features about the grass and its color, which makes it more green, while the ground truth in fact had a dry version of the grass, which looks yellow-

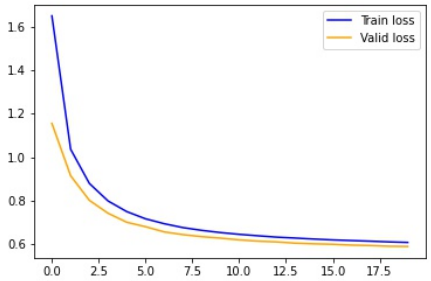

Figure 3. Loss function for training and validation set by epoch, Model 2 (ConvNeXt-T) with the x-axis showing the number of epochs and the y-axis showing the MSE loss

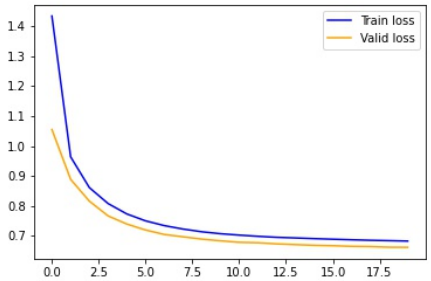

Figure 4. Loss function for training and validation set by epoch, Model 3 (reduced contribution of the Feature Extractor) with the x-axis showing the number of epochs and the y-axis showing the MSE loss

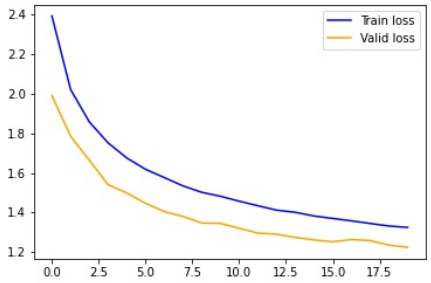

Figure 5. Loss function for training and validation set by epoch, Model 4 (cross-entropy loss) with the x-axis showing the number of epochs and the y-axis showing the cross-entropy loss

ish. This disagreement is expected and should not be concerning as long as the model was able to generate a picture with a realistic colors. Also, it seems that both the vanilla inception-v3 and ConvNeXt-T has learned more blue than they should, which lead some of the pictures to look more bluish, especially the ones with either sky or water.

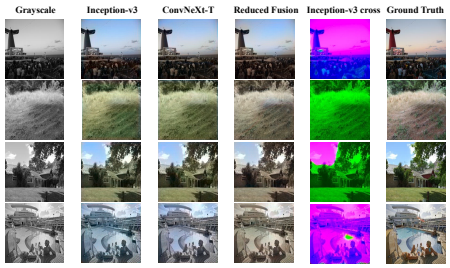

Figure 6. All models are shown with the final results starting from the left with the grayscale that was fed to the model as an input and the ground truth on the far right which was the target where the model was trying to learn the coloring. The figures shown here belong to the test set, and the model for all of them gives acceptable results except for the Inception-v3 with the cross-entropy loss.

The images obtained using the cross-entropy loss are not surprising for us, because it is not intuitive to use this type of a loss function for a*b channel distributions which are continuous. Rather, we could follow the procedure used in [7] to discretize the continuous distributions into bins.

4. Other Issues

Beside previously mentioned obstacles, we also encountered a number of other problems during the process. Many of them occurred during the data manipulation part. Specifically, it took us a fair amount of time to make sure that we: i) correctly convert images from RGB to CIE Lab; ii) scale the luminance and a*b* channels properly considering the different ranges that each channel has; iii) feed the right image components into the feature extractor and the fusion layer; iv) correctly scale back the predicted image components and covert them to RGB.

Another issue was related to the training of the model. On several occasion the system threw an error due to lost internet connection, so we had to restart the process. This was especially costly because the training of the models took at least 2.5 hours with a premium GPU from Google Collab. Moreover, we did not have access to premium GPUs all the time.

Lastly, the group members worked on this project together throughout the semester from inception to writing the report. The distribution of the work in this project can be found on Table 1

5. Conclusion

Using deep learning to tackle the colorization has been a great advantage to the toolkit of historians, government officials, or even random people. The model can infer the environment based on the information stored within the picture,

sometimes with the help of other pre-trained model that can extract other high-level features on-the-fly. Although the results were promising, some challenges and questions remain unanswered. For example, how would transformer-based feature extractor help this task to have more realistic colors? How to ensure the data that was used for training is not biased?

References

[1] Federico Baldassarre, Diego González Morín, and Lucas Rodés-Guirao. Deep koalarization: Image colorization using cnns and inception-resnet-v2. *arXiv preprint arXiv:1712.03400*, 2017. 2, 3

[2] Guillaume Charpiat, Matthias Hofmann, and Bernhard Schölkopf. Automatic image colorization via multimodal predictions. In *European conference on computer vision*, pages 126–139. Springer, 2008. 1

[3] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *Proceedings of the IEEE international conference on computer vision*, pages 415–423, 2015. 1

[4] Alex Yong-Sang Chia, Shaojie Zhuo, Raj Kumar Gupta, Yu-Wing Tai, Siu-Yeung Cho, Ping Tan, and Stephen Lin. Semantic colorization with internet images. *ACM Transactions on Graphics (TOG)*, 30(6):1–8, 2011. 1

[5] Ryan Dahl. Automatic colorization, 2016. http://https://tinyclouds.org/colorize. 1, 2

[6] Yi-Chin Huang, Yi-Shin Tung, Jun-Cheng Chen, Sung-Wen Wang, and Ja-Ling Wu. An adaptive edge detection based colorization algorithm and its applications. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 351–354, 2005. 1

[7] Jeff Hwang and You Zhou. Image colorization with deep convolutional neural networks, 2016. http://cs231n.stanford.edu/reports/2016/pdfs/219_Report.pdf. 1, 2, 3, 4

[8] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*, pages 689–694. 2004. 1

[9] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. 3

[10] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016. 1

[11] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017. 2

6. Work Division

Student Name	Contributed Aspects	Details
Aleksandr Shirkhanyan	Data Acquisition	Contacted the places team to get the private link to the data
Faisal A. Aldukhi	Data preprocessing	Prepared the data in a way that can be fed easily to the pipeline
All members	Designing the experiments	Discussed and proposed the interesting parts to modify, add, or experiment with
Josip Franic	Analyzing the outcome	Analyzed the resulting patterns and provide reasons why it occurred
All members	Writing the report	Combining all the efforts into a concise report

Table 1. The contribution of each group member is outlined above.