Analysis of Colourization of Monochrome Images using CNNs

Intro Deep Learning Systems

Spring 2021

Varun Menon
vk2148@nyu.edu, New York University

Arka Talukdar
at4786@nyu.edu, New York University

1 Introduction

In this project, we present a study on colorizing monochrome images using Convolutional Neural Networks. We analyze the performances of various network architectures on the colorization of in-domain data and out-of-domain images. We also compare the performance of similar architectures trained with and without adversarial training paradigm.

Colorization is the process of adding plausible color information to monochrome images. Given the fact that not even humans can accurately pinpoint to which colors have to be filled in a grayscale image, this problem becomes subjective and hard for machines to solve. Colorization is intrinsically an incomplete problem as it requires mapping a Grayscale input to an appropriate 3 Channel output (depending on what color system is preferred), where the output does not have to be the same as the target but needs to be perceptually natural. The usual supervision does not take into consideration the perceptual naturalness of an image and attempts to reconstruct the image true to the ground truth. Without additional supervision (like GAN critic) the model is incentivized to average to grayish image [7]. As noted in [7], L1 Loss or MSE Loss solely will still be able to teach the model about the color distribution but the color palette will be mostly conservative and it will pick the average color which more often than not happens to be sepia. It is also noted that L1 Loss is preferred over MSE Loss as L1 Loss reduces the chance that the images are all painted with the sepia tint. In the prior art, it is shown that models are first trained with L1 Loss and then fine-tuned with MSE Loss (MSE being more stable and helps to settle on the learned color distribution).

Our primary goal is to build a Resnet based U-Net[1] and benchmark its the performance with similar architectures of the same ability. We go further and show how to avoid using involved training mechanisms like Adversarial supervision to get good results.

2 Prior Art

2.1 Conventional Methods

Conventionally, Grayscale images were colored by hand. It was time-consuming and required high-skill labor. Image Processing advancements like color bleeding and color continuity helped increase the rate of work but the pipeline still required human supervision.

2.2 Deep Learning for Auto colorization

Deep Colorization, Cheng et al. 2016[4] did seminal work on Automatic Image colorization using deep learning. This was one of the first works to use deep learning to attempt fully automatic colorization. This method uses cues from a clustered database of images to get associated chrominance maps to colorize the image.
Similarly, Zhang et al. 2017[5] attempt automatic colorization of monochrome images with cues that are sparse pixel values to induce different color palettes.
We benchmark our model's performance vs the results reported in NTIRE 2019 Challenge on Image Colorization CVPR workshop [3].

3 Experimental Result

3.1 Datasets

We used two different datasets for our experiments:
Div2k Datasets: This is a dataset of high-resolution diverse images. 800 images in the training set and 100 in the validation set. [10] This dataset is widely used in prior work to compare model performance in image colorization tasks.
Landscape Pictures: This is a domain-specific dataset created by crawling Flickr. Consists of 4319 images.[11]

3.2 Evaluation Metrics

It is a difficult task to quantitatively evaluate colorized images. In image colorization, image quality is a prime criterion. The most traditional estimator is MSE**(Mean-Square Error)**. MSE measures the average squared difference between the estimated values (predicted values) and the actual value (ground truth). **PSNR (Peak Signal to Noise Ratio)** is the second traditional estimator. It's used to calculate the ratio between the maximum possible signal power and the power of the distorting noise which affects the quality of its representation. This ratio between two images is computed in decibel form. But PSNR is a variation of the MSE and still concentrates on the pixel-by-pixel comparison. **SSIM (Structural Similarity Method)** is correlated with the quality and perception of the human visual system (HVS color model). Instead of using traditional error summation methods, the SSIM models image distortion as a combination of three factors that are loss of correlation, luminance distortion, and contrast distortion.[12]
Even though we use a number of metrics that take into account the structural similarity and noise of the generated image with respect to the ground truth, the

visual perception still remains the best way to differentiate the images. This has been used as a standard to rank model performance in prior work.[3]

3.3 Loss Function

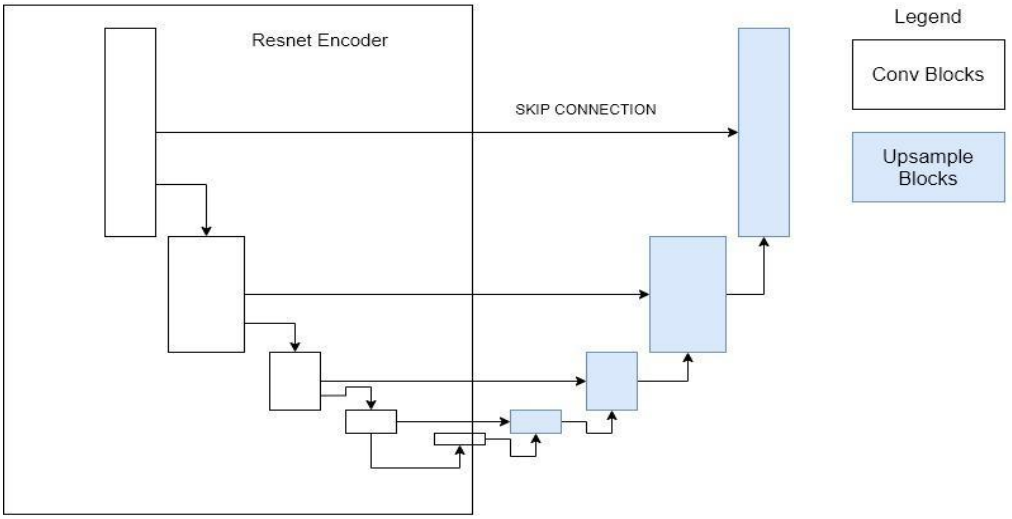
We used a multitask loss consisting of L1 Loss + SSIM for training and MSE Loss + SSIM for fine-tuning. We weight the SSIM with ƛ. We empirically find the optimal value of ƛ to be 1e-3. We introduce this multitask loss to incentivize the network to learn to preserve high structural similarity while testing color palettes that may not be true to the ground truth.

3.4 LAB Color Space

Our dataset is available in 3-channel RGB. However, we convert it into a LAB color scheme for the purpose of the colorization task. In the RGB color scheme, the model would have had to produce a three-channel output for each of the greyscale inputs. However, in the LAB color scheme, the L-channel corresponds to grayscale and the model is provided with the grayscale image. The model needs to predict just the A and B channels. This reduces the solution space for the model from 3 channels in RGB to 2 channels in LAB. Reduction in solution space significantly reduces the task complexity and thus the LAB is used in the most prior art of image colorization problems as the preferred color scheme.

3.5 Model Architecture

We build a Resnet34 based U-Net architecture in PyTorch. We use the Upsample+Convolution layer in the upscaling blocks because the Transpose convolution layers present themselves with checkerboard artifacts. The Retnet34 backbone is not truncated as done in prior work and we show that going deeper did in fact help reduce loss at a higher rate as shown in Section 4. Figure 1 describes the network architecture.

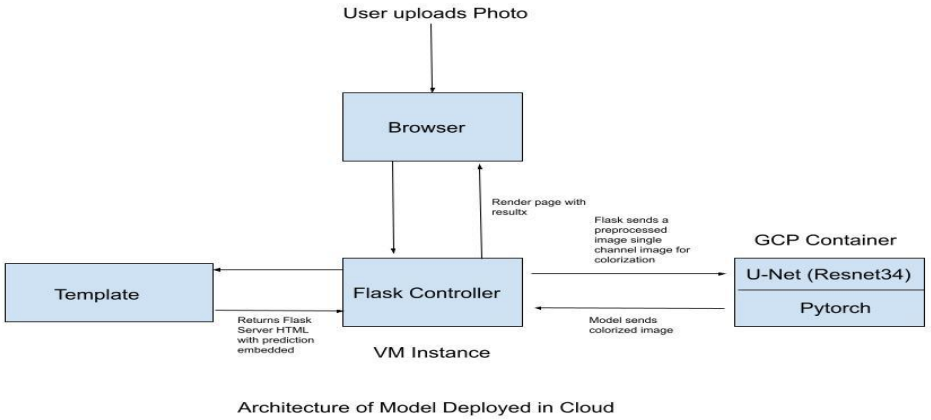


Fig[1]: Representation of the Resnet-UNet architecture

Skip connections we implemented were by using the concatenation function. There are works that use element-wise summation as well. Compute cost is less in element-wise summation, but flexibility for representation learning is more in concatenation as the network gets to optimize over a larger representation space.

Cloud Deployment

We deployed our model to the cloud using the microservice architecture. As described in Fig[2] we create a python flask-based middleware that interacts with the user using a web browser-based template. The colorization model is contained in an isolated GCP container accessible via REST API. For users who ping the server via the browser, the results are returned embedded in an HTML template.



Fig[2]: Representation of the Resnet-UNet architecture

4 Results

Our U-Net-based model performs exceedingly well when trained on Landscape Dataset and presented with in-domain test samples,however, it performs poorly when presented with images out of the domain.

Fig[2]: Output of U-Net trained on Landscape dataset colourization of in-domain (unseen) scenery image and out of domain human.

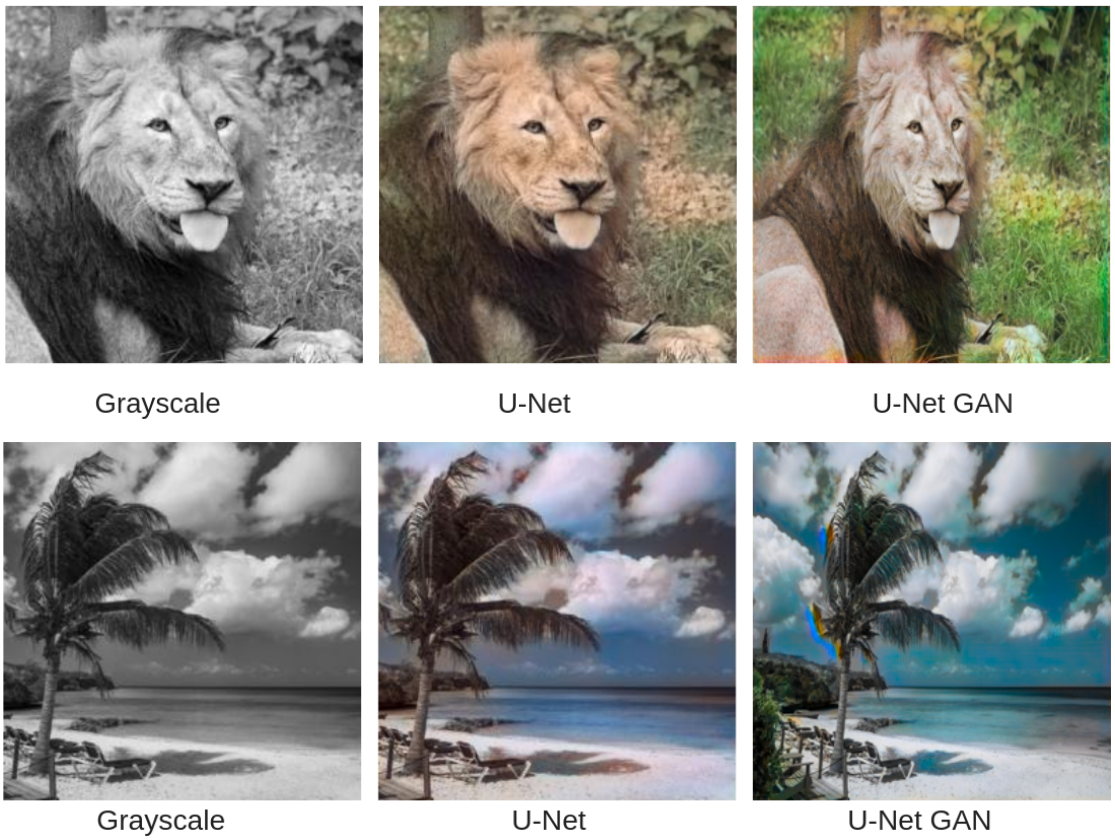
We compare our results with prior work. The Div2k dataset was introduced in NTIRE 2019 Challenge on Image Colorization, CVPR. Our

Team	PSNR	SSIM	Perceptual rank
PCVIITM	22.1232	0.9406	1
Athi	20.871	0.9229	2
VIDAR	22.1949	0.9419	4
Our Model	**21.0723**	**0.8802**	**-**
TeamIndia	17.9643	0.8472	3
pksvisionmm	21.2248	0.9279	5

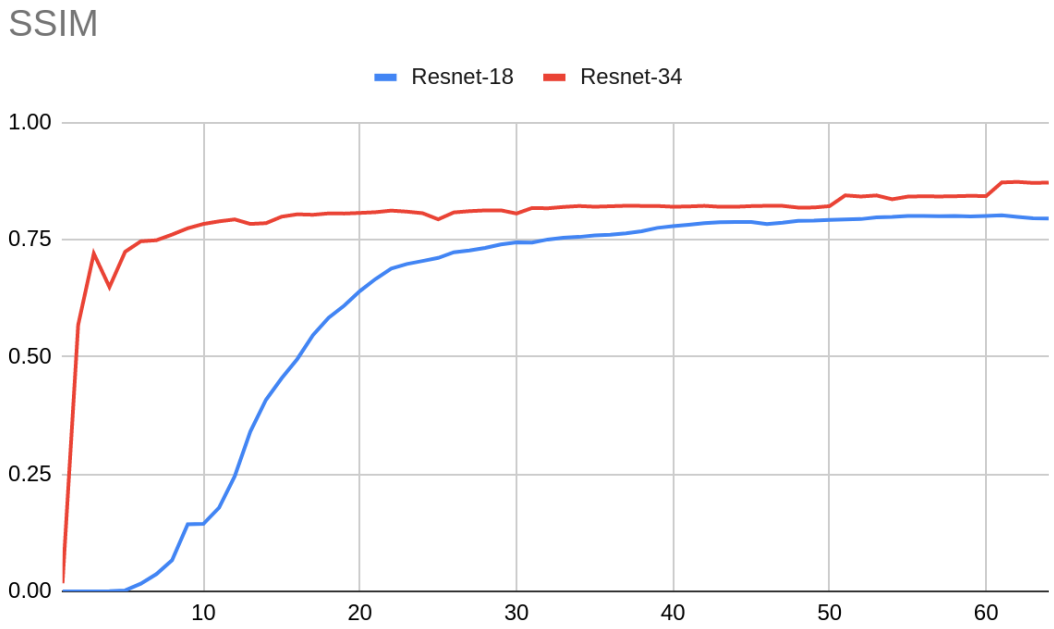
ITU-GO	21.0773	0.8526	6

Table 1: Our Model compared with NTRITE Challenge 2019.

When we train our dataset on Div2k dataset, due to the diverse nature of images the model struggles to learn the various coloring patterns. Visual inspection of validation images after a few epochs shows us that the model, unable to learn to color correctly, resorts to using the most common shade (Sepia for div2k) for a large number of images. In these same images when we train GANs we get a much more vibrant set of colors. However, introducing GANs increases the artifacts in the image.



Fig[2]: Output of U-Net and GAN (using same U-Net as encoder). U-Net is less confident in using diverse colors while GAN introduces artefacts (deep blue patch on left of tree image and legs and head of lion).



Fig[3]: SSIM comparison for varying Resnet encoder depth.

5 Conclusion

In our experiments, we find that comprehensive and diverse datasets can help models achieve good performance without adversarial training. This fact is corroborated by the visual perception of images colored by our U-Net-based model when compared to the performance of the same model on the Div2k dataset.

We also note that while GANs allow us to have a more colorful palette it is much more prone to having artifacts as seen in perceptual evaluation.

ACKNOWLEDGMENTS

We would like to thank Prof Dr Parijat Dubey for his mentorship and guidance throughout the project. We benefited from the discussions we had with him to gain clarity on how to proceed.

REFERENCES

[1] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and computer-assisted intervention*. Springer, Cham, 2015. [1505.04597] U-Net: Convolutional Networks for Biomedical Image Segmentation

[2] Kodali, Naveen, et al. "On convergence and stability of gans." *arXiv preprint arXiv:1705.07215* (2017).[1705.07215] On Convergence and Stability of GANs

[3] Gu, Shuhang, Radu Timofte, and Richard Zhang. "Ntire 2019 challenge on image colorization: Report." *Proceedings of the IEEE/CVF Conference on Computer*

Vision and Pattern Recognition Workshops. 2019. https://ieeexplore.ieee.org/document/9025578

[4] Cheng, Zezhou, Qingxiong Yang, and Bin Sheng. "Deep colorization." *Proceedings of the IEEE International Conference on Computer Vision*. 2015. [1605.00075] Deep Colorization

[5] Zhang, Richard, et al. "Real-time user-guided image colorization with learned deep priors." *arXiv preprint arXiv:1705.02999* (2017).[1705.02999] Real-Time User-Guided Image Colorization with Learned Deep Priors

[6] U-Net deep learning colourisation of greyscale images

[7] Colorizing black & white images with U-Net and conditional GAN — A Tutorial

[8] Image Colorization with Convolutional Neural Networks

[9] Deploying PyTorch in Python via a REST API with Flask — PyTorch Tutorials 1.8.1+cu102 documentation

[10]Timofte, Radu, et al. "Ntire 2017 challenge on single image super-resolution: Methods and results." *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017.

[11]Landscape Pictures https://www.kaggle.com/arnaud58/landscape-pictures

[12]Hore, Alain, and Djemel Ziou. "Image quality metrics: PSNR vs. SSIM." *2010 20th international conference on pattern recognition*. IEEE, 2010.