
Real Time Video Colorization Using Color Priors

Sneha Venkataramana
Software Engineer
KLA-Tencor
snehavr@stanford.edu

Abstract

I worked on developing a system to perform guided colorization of grayscale videos using color priors. For this work, I built on two existing methodologies - Real-Time User-Guided Image Colorization with Learned Deep Priors (1) (4), and, Neural Color Transfer between Images (2).

I employed two models in this work, one to colorize the grayscale videos and the other to transfer the color between frames in the videos. I experimented by tweaking the architectures in the prior implementations to identify what works best in our case, which essentially combines the two of them. From the perspective of the flow the system, the user provides sparse local color hints for the first frame in the video - these color hints provided by the user acts the color style source for the next part of the model. I implement color style transfer to incorporate the colors into the rest of the video using the previous frame as the source for the colorization.

1 Introduction

In this project I extended Zhang et. al.'s (1) (4) model to colourize black and white videos and then used the color transfer work in Gatys et. al (2). This project allows the added flexibility of allowing the user to guide the colorization of a video (or a gif) by adding color priors. If no color priors are provided, the colors learned by the CNN model (through training on the imagenet dataset) will be used to color the videos. In addition to colorization, I also provide the user the ability to style the video (e.g. adding a starry night effect on top of their choice of colorization) to their wish.

2 Related work

2.1 Manual Colorization with references

Levin et al.(9) propose an effective approach that gets the user input in the form of scribbles on the grayscale images. Least square optimization is used to propagate this color to the rest of the image. Yatziv et al.(10) uses weighted combination of user scribbles to color the image. Many have used a reference image to colorize a black and white image. Welsh at al.(11) make use of the pixel intensity and the neighborhood statistics to find a similar pixel in the reference image and then transfer the color of the matched pixel to the target pixel.

2.2 Automatic Colorization Without references

The work done in Cheng et al.(5) uses the assumption of a perfect patch matching technique in an extensive database to color the images. The paper proposed by Zhang et al.(4) solves the colorization

issue using a feed-forward pass in a CNN at test time and is trained over a million images. The algorithm is evaluated using a "Colorization Turing test" which is basically using humans to distinguish the ground truth and generated color images.

Zhang et al.(1) uses color priors to have the added flexibility of colorizing grayscale images in a way that the user wishes. If no priors are provided, then the model colorizes the image based on its trained data. According to the model developed in Zhang et al.(1), a reference image can also be provided to give a reference for coloring the image.

2.3 Color Transfer

There are two types of color transfer done in the field today. In non-parametric color transfer, color is transferred onto the input image using Image Analogies framework (12). Under parametric color transfer, learn prediction functions from large data sets of color images at training time. This approach is the explored in Larsson et al.(13) and Iizuka et al.(14) using CNN.

3 Dataset and Features

I trained the model using 2 datasets - CIFAR-10 (7) and ImageNet (6). Initially I used the CIFAR-10 as the images are smaller and faster to train; this was simply used as a proof-of-concept for the initial milestone; and I moved on to training the model imagenet afterwards.

3.1 CIFAR-10 Dataset

The images from the CIFAR-10 are colour images of the dimension 32x32. The data was split into train, test and validation sets. The model was trained on 50,000 images. The test and validation contained 10,000 images each.

3.2 ImageNet Dataset

These images are larger and have an average resolution of 469x387 but were re-sized to 256x256 in a preprocessing step. There were 50,000 images in the training image set and 10,000 images in the test and validation tests.

3.3 Features

- My input image is re-sized to 256x256 to fit memory constraints
- The other input is a user-given hint, which is converted to a 256x256 RGB image
- The network represents the 3-channel hint in 2-channels
- Ground truth is a color image which was de-colored for training

4 Methods

My final approach for this system was to use the two tower model in the colorization work, and build a pipeline that feeds the output of the colorization model into the color transfer model (built on VGG). This process happens in a loop so as to colorize (and add a style if required) the entire video. The first half of the model is as shown below.

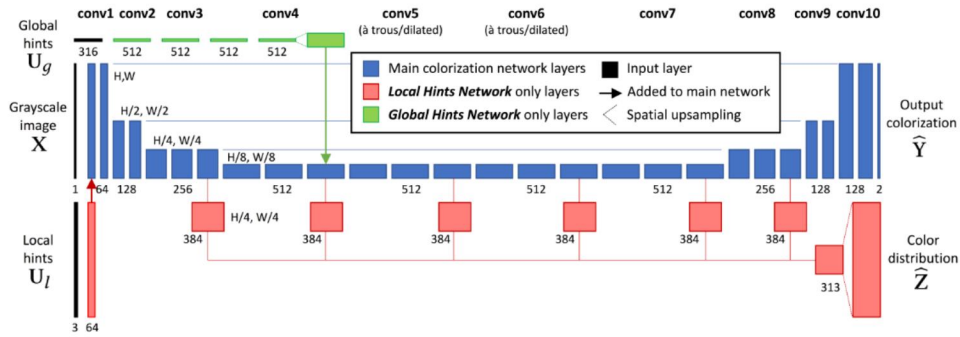


Figure 1: Deep Colorization Architecture using CNN

This is the mostly built on the model based on Real-Time User-Guided Image Colorization with Learned Deep Priors. I modified the hyper-parameters (learning rate over 3 magnitudes) and the training models (adding and removing convolution layers). I used squeeze convolution filters in the colorization model to reduce the number of parameters. The models used are outlined below. I trained this model for 15 epochs for both Imagenet and CIFAR-10, and adding convolution layers did not significantly reduce the loss on the generator at the end of the 15 epochs.

The model mainly consists of 3 parts - the main colorization network that trains and learns colors from the training dataset, the local hints network which is the input sparse vector from the user containing color values for certain areas in the image and the global hints network.

The main colorization network uses a U-net architecture. The network consists of 10 convolution layers and the activation used in each layer is ReLU activation function. Conv2 and Conv3 blocks are connected to Conv8 and Conv9. This symmetric shortcut connection help to use the low level information for later layers. Each convolution uses a 3x3 filter. The last Conv layer (which is a 1x1 filter) is used to map the Conv10 to the output color. In the last layer, tanh is added for activation.

The local hints network are shown in red color in fig:1. The sparse vector containing the user given colors are concatenated with the input grayscale image. There is no back propagation of the gradients from side task to the main branch (global network).

The global hints are shown in green in the network architecture figure. This is given every 4 convolution layers into the main colorization network.

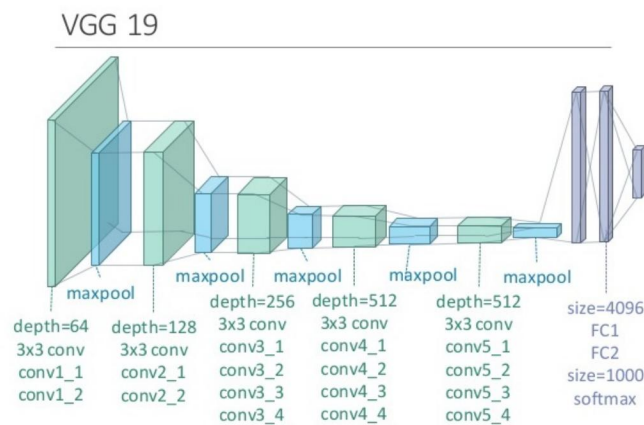


Figure 2: Color Transfer Architecture using VGG

The second part of the model is to use a color transfer model using VGG network2. I followed the model based off . I modified the loss function on the color transfer network by adding a heavier loss on the content model that helped generalizing the color scheme.

Final loss function:

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = 12 \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2 + |F_{ij}^l - P_{ij}^l|$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = 12 \sum_{l=0}^L w_l E_l$$




Input		Output		
Content Image	Style Image	Max Pool Adam Optimizer Old Loss	Max Pool Adam Optimizer New Loss	Average Pool Lbfgs Optimizer New Loss
				

Figure 3: Results of modifications in the color transfer model

The above mentioned models are used in sequence using the pipeline shown in the figure below. The first frame of the video is input to the colorization model along with the user defined color hints. The color hint image is converted from a 3-channel (RGB) image to a 2-channel image to reduce the input parameters. Next the trained U-net model colorizes the image base on the hints provided and outputs a 256X256 colorized image.

This image is then taken as a reference image in the color transfer model and applied iteratively on rest of the frames. If we increase the weight on the style transfer, this model can be used to generate a video using the style of the reference image.

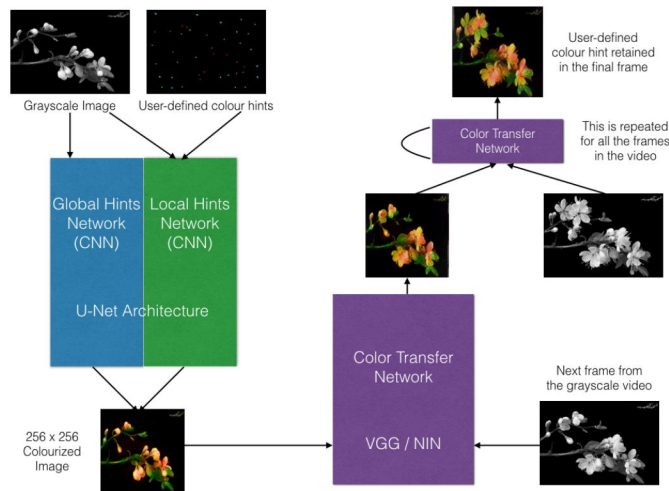


Figure 4: Pipeline

5 Experiments & Results

There were totally 4 approaches I tried for colourizing the grayscale videos.

- Using only the colorization model
- (1) + (2) with minor param tweaks
- (1) + updated loss function on (2)
- (1) with reduced params + (2) with updated loss function

The first approach had the drawback that the color hints for the first frame may not be applicable to the 100th frame.

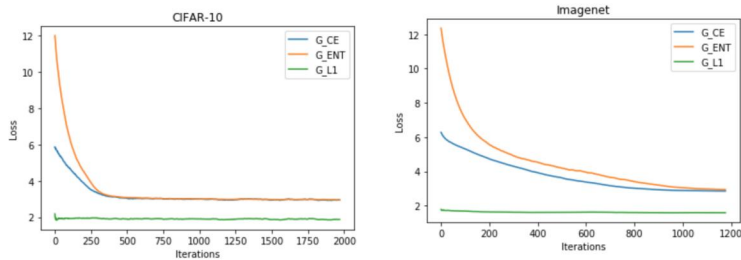
The second approach was quite improved, but could have been better on relaxing the weights to reference image.

The third method worked better on generalizing the color scheme between frames. I felt that this suffered from the amount of time it took to colorize the whole video. This was my motivation to try the next variant.

The final variant was a test to see if reducing the parameters in the network had a strong effect on the outcome - the motivation here is that the color transfer is rather slow and could be perhaps sped up - this had reasonable results as the difference in the quality of the generated images was low (<10% PSNR loss). This was done by borrowing ideas from SqueezeNet (8) by reducing the number of params. Ideally I would have liked to do this on both the models, but only experimented with doing this with the colorization model.

Overall, the third method that combined both models was both qualitatively and from a metric perspective, the best candidate.

5.1 Learning Curves



5.2 Results

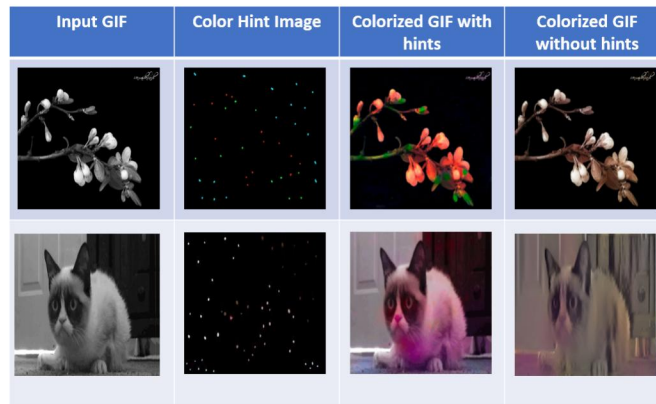


Figure 5: Test Results of the Model

The third column in fig:5 demonstrates the results from this work, where the input gif file and a color hint file is used to generate the colorized frame, which is then used by the color transfer model to colorize the rest of the frames in the gif. The final column is an generated using the same network, but when there were no user provided hints on colorizing the gif.

5.3 Evaluation

I will calculate the Peak Signal to Noise Ratio (PSNR) value by comparing the frames from ground truth videos to the frames from our predicted output – a high PSNR would indicate that our model is relatively good.

The results were evaluated objectively using PSNR (Peak Signal to Noise Ratio) and qualitatively as well. All the variations below were tested with 15-20 points of color hints.

Legend: (1): Colorization model (2): Color transfer model

Method	Avg. PSNR
Only colorization model	11.46 ± 0.35
(1) + (2) with minor param tweaks	12.57 ± 0.13
(1) + updated loss function on (2)	13.04 ± 0.18
(1) with reduced params + (2) with updated loss function	12.55 ± 0.24

6 Conclusion / Future Work

I observe that colorization of a video or a gif using a single hint can produce convincing results as long as the video is small (I tested videos up to 10 seconds). Another observation that I had is that my initial approach of using a single hint and colorization network alone was not enough, and combining colorization and a color transfer model was able to produce the best quality videos. Finally, the current system could be made to run faster, for which I validated by adding fire modules, an idea for smaller number of parameters in the SqueezeNet paper. Reducing the number of parameters in the complete network, while applying this to fairly small videos seems like the best step forward when using this work.

I would like to explore the work of He, et.al. on Deep Exemplar based Colorization where this network is augmented with a similarity-sub net which showed very realistic colorizations in their paper. This would be done by modifying the colorization network to have an additional parallel path.

Another interesting problem I would have liked to work on, is to train this network end-end instead of two separate networks. This could be done by formulating a joint loss function for the learning step after connecting the two networks, where I try to minimize (i) the color difference between two adjacent frames, (ii) the distance from the best predicted pixel color and the given user color.

7 Contributions

My codebase is [here](#). The following code changes were made from the original work that I borrowed from:

- Expanding and contracting the colorization model using convolution filters (models/networks_v2.py, models/networks_v3.py)
- Reducing the params on the colorization model (models/networks_squeeze.py)
- Pipeline (modified in train.py and neural_style.py)
 - Looping over a set of photos, but a single color hint to generate a gif (train.py)
 - Feeding the output of the colorization model and then calling the color transfer model (full.sh)
- Tweaked loss function on the color transfer to the one discussed earlier (models/neural_style.py)

References

- [1] Real-Time User-Guided Image Colorization with Learned Deep Priors (2017) - Zhang, Richard and Zhu, Jun-Yan and Isola, Phillip and Geng, Xinyang and Lin, Angela S and Yu, Tianhe and Efros, Alexei A <https://richzhang.github.io/ideepcolor>
- [2] A Neural Algorithm of Artistic Style (2015) - Leon A. Gatys, Alexander S. Ecker, Matthias Bethge <https://arxiv.org/abs/1508.06576>
- [3] Deep photo style transfer (2017) - Luan, Fujun and Paris, Sylvain and Shechtman, Eli and Bala, Kavita <https://arxiv.org/abs/1703.07511>

- [4] Colorful Image Colorization (2016) - Zhang, Richard and Isola, Phillip and Efros, Alexei A <http://richzhang.github.io/colorization/>
- [5] Deep colorization (2015) - Cheng, Zezhou and Yang, Qingxiong and Sheng, Bin *Book : Proceedings of the IEEE International Conference on Computer Vision Pages : 415 – 423*
- [6] ImageNet: A large-scale hierarchical image database (2009) - Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. <http://image-net.org/>
- [7] Learning Multiple Layers of Features from Tiny Images (2009) - Alex Krizhevsky <https://www.kaggle.com/c/cifar-10/data>
- [8] SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size (2016) - Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer <https://arxiv.org/abs/1602.07360>
- [9] Colorization using optimization (2004) - Anat Levin, Dani Lischinski, and Yair Weiss. In ACM Transactions on Graphics (TOG), Vol. 23. ACM, 689–694
- [10] Fast image and video colorization using chrominance blending (2006) - L. Yatziv , G. Sapiro, IEEE Transactions on Image Processing, v.15 n.5, p.1120-1129, May 2006 [doi>10.1109/TIP.2005.864231]
- [11] Transferring color to greyscale images (2002) - Tomihisa Welsh, Michael Ashikhmin, and Klaus Mueller. ACM Transactions on Graphics (TOG) 21, 3 (2002), 277–280.
- [12] Image analogies (2001) - Hertzmann, Aaron and Jacobs, Charles E. and Oliver, Nuria and Curless, Brian and Salesin, David H. <https://dl.acm.org/citation.cfm?id=383295>
- [13] Learning Representations for Automatic Colorization (2017) - Gustav Larsson, Michael Maire and Gregory Shakhnarovich <https://arxiv.org/pdf/1603.06668.pdf>
- [14] Let There Be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification (2016) - Iizuka, Satoshi and Simo-Serra, Edgar and Ishikawa, Hiroshi