Colorizing Grayscale Images with Convolutional Neural Networks (CNN)

Shri Akhil Chellapilla (CS, Blacksburg), Kian Lua (STS, Blacksburg)

Introduction

- Recolor grayscale images.
- Compare and reproduce the results using 2 colorization methods (Zhang et. al. (2016) vs ibid. (2017)).
- Both use CNN to reproduce colorizations, but with slight different methods to colorize images.

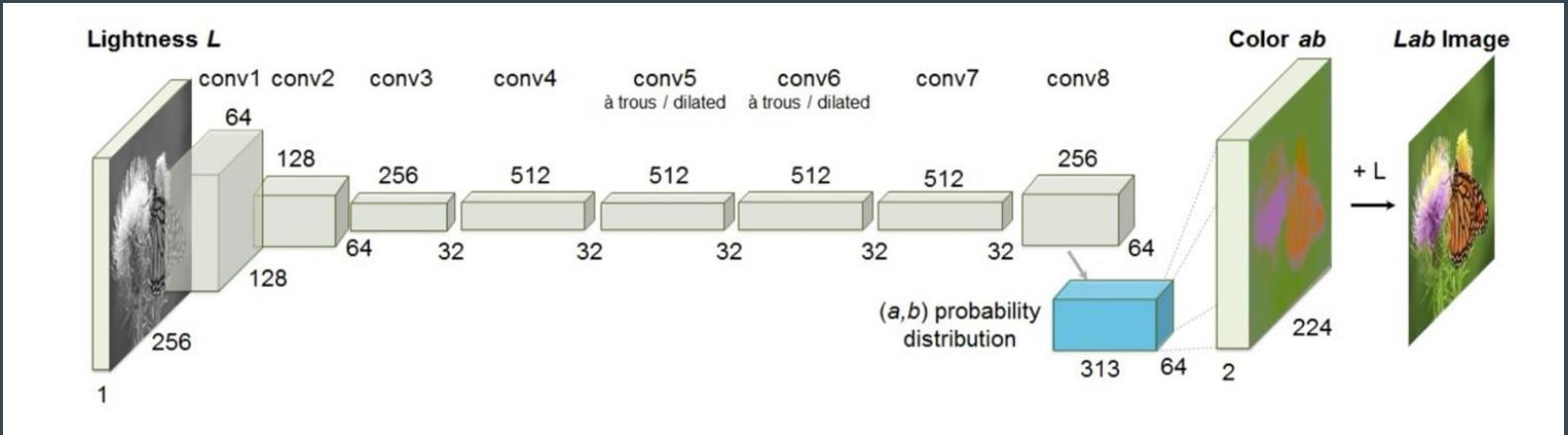
Method 1: Fully automated color selection

- Fully automated approach to color grayscale images.
- Color prediction is multimodal - there could be several ways to color an object.

- Goal: AI system to “imagine” or produce a plausible color version for a grayscale photograph; compelling enough to fool the human observer - unable to tell the difference between the colored image and original image.

Method 1: CNN architecture

- Train a CNN to map from grayscale input to a distribution over quantized color value outputs using the architecture shown below.
- CNN is trained over a 1.3 million color images from ImageNet w/o semantic label information.



Architecture details in article (Zhang et. al. 2016)

Method 1: Objective function

- The CNN works together with the objective function to **produce results close to real color photos.**
- Need an objective function that handles the multimodal uncertainty of the colorization problem and captures a wide variety of colors.
- Briefly, given an input lightness channel $X \in \mathbb{R}^{H \times W \times 1}$, the objective is to learn a mapping $\hat{Z} = G(X)$ to a probability distribution over all possible colors, $\hat{Z} \in [0,1]^{H \times W \times Q}$,
  - where Q is the number of quantized ab (the color space) values; H and W are image dimensions.

Method 1: Objective function

- After finding the probability dist. over possible colors, $\hat{Z}$, we map it to color values $\hat{Y}$ with function $\hat{Y} = H(\hat{Z})$, where H is defined as the eqn below:

$$\mathcal{H}(\mathbf{Z}_{h,w}) = \mathbb{E}\big[f_T(\mathbf{Z}_{h,w})\big], \qquad f_T(\mathbf{z}) = \frac{\exp(\log(\mathbf{z})/T)}{\sum_q \exp(\log(\mathbf{z}_q)/T)}$$

- The mapping H operates on each pixel independently, with a single parameter, and can be implemented as part of the feed-forward pass of the CNN.

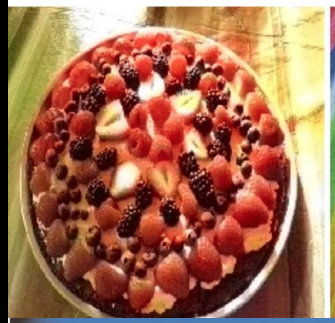
Method 1

Grayscale

Colorized

Original

Colorized



Original

Original

Colorized

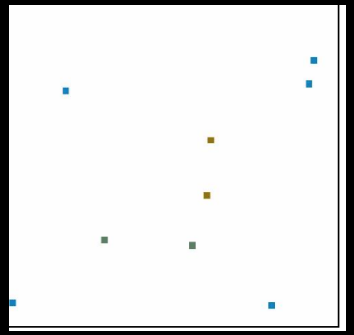
Method 2: Automated color selection + human intervention

- Like previously, CNN is trained with more than 1 million ground truth natural color images.
- Instead of automatically producing a fixed color image from a grayscale image, we now have the option for real-time human guidance to change/update image color.
- The user chooses to give color “hints” in the image by clicking on different points.
- The AI then automatically colors the rest of the object based on user’s color hints.

Method 2: Examples

Grayscale

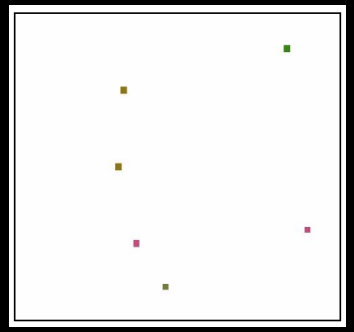
Fully Automated



Clicked points

User guided

Original Image



Method 2: Global Hints Network

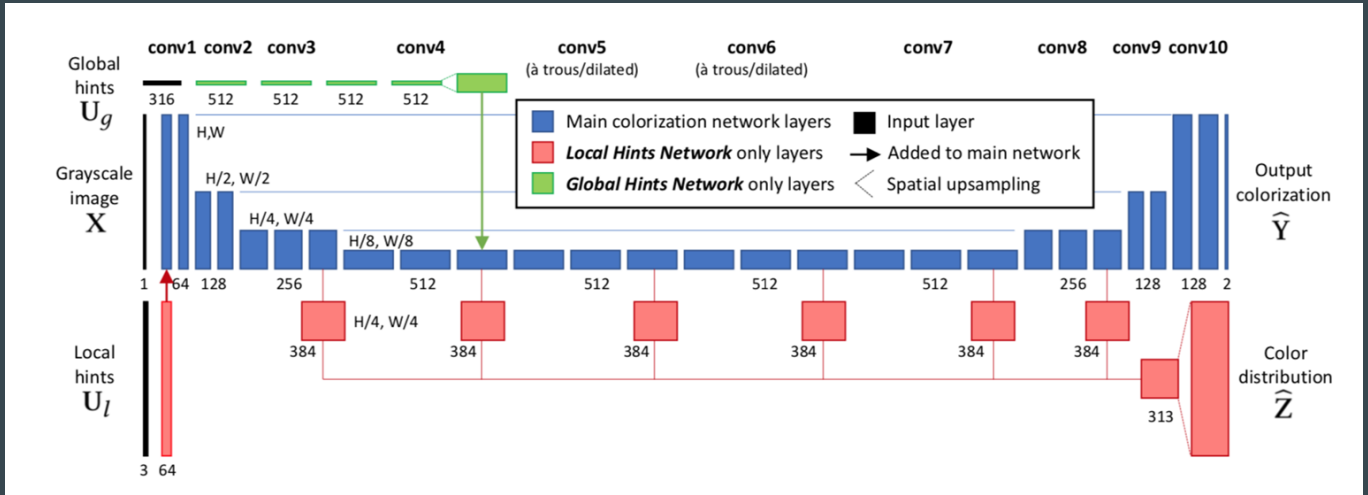
- Fixes a set of possible color distribution and saturation which comes from a global histogram of colors extracted from another reference image.

Method 2: Network Architecture to colorize an image

Blue: Main colorization network layers.

Red: Local hints network–sparse user points integrated with the input grayscale image.

Green: Global hints network–provides a global distribution of colors available for an image obtained from other reference images.



The CNN system recognizes the high-level semantic information (objects and areas in the image). It propagates the “local” color edits by fusing these low-level cues with high-level semantic information.

Approaches

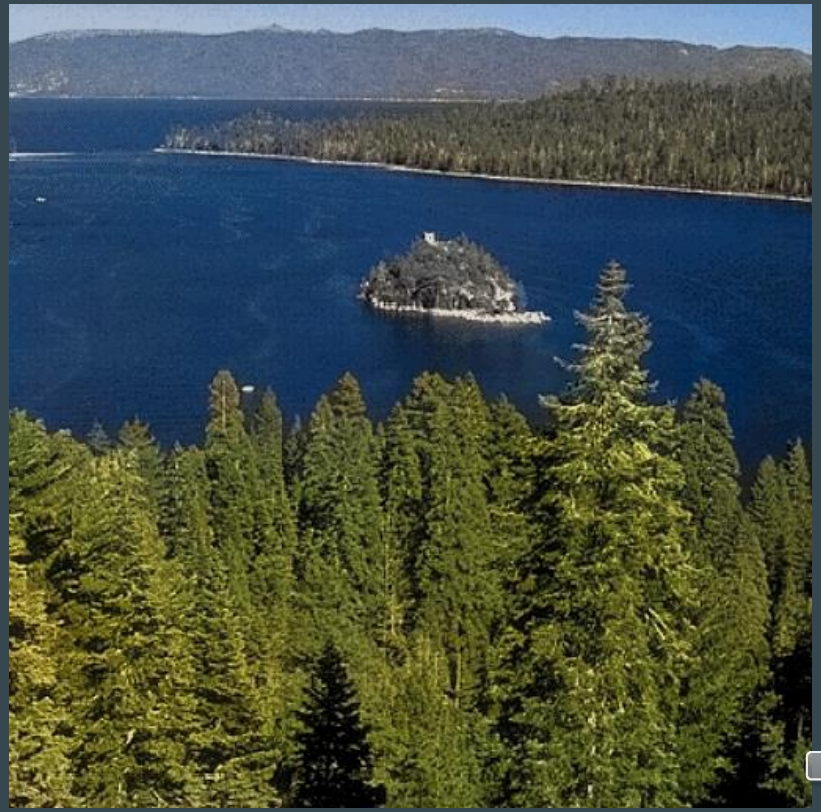
- Pre-trained colorizer models for Caffe and PyTorch were provided along with training code on the authors' Github repositories for each paper.

- The ECCV '16 repository was updated to include a comparison between the automatic results.

- Generated outputs for various types of images taken from online datasets and photographed by us.

Results - Automatic Colorization

ECCV ‘16



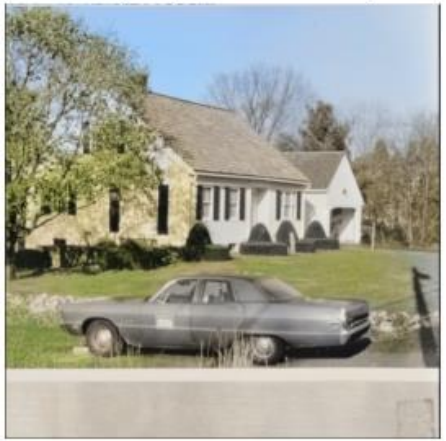
SIGGRAPH ‘17

Original

Input

Output (ECCV 16)

Output (SIGGRAPH 17)



ECCV ‘16

SIGGRAPH ‘17

Original

Input

Output (ECCV 16)

Output (SIGGRAPH 17)

Original

Input

Output (ECCV 16)

Output (SIGGRAPH 17)

Original

Input

Output (ECCV 16)

Output (SIGGRAPH 17)

Results - Interactive Deep Colorization

- A lot tougher to reproduce the results of this paper because it was built for Linux and MacOS with poor Windows support.
- Could not run the full GUI demo, thus limiting to barebones examples which were only 256 x 256 px.

Original Image

Automatic Colorization

Results - Interactive Deep Colorization

Lessons Learned

- Major changes were related to discontinued / outdated / unsupported libraries:
  - skimage -> scikit-image
  - PIL -> Pillow
  - Qt4 not well supported on Windows; Qt5 not properly supported in code
- Docker would have been perfect for taking such a snapshot of dependencies.
- Availability of pre-trained models was very helpful in reproducing results.
- Easy to get a “good enough” result, hard to get a perfect result.

Future work

- Colorization sees a lot of improvements due to its undetermined nature.
- Newer models like Palette make use of GANs and also tackle uncropping, inpainting and decompression.
- Technology based on interactive deep colorization was incorporated in Adobe Photoshop Elements 2020.

Questions?