Sketch Colorization

Yuki Inoue
Department of Electrical Engineering
Stanford University

Problem

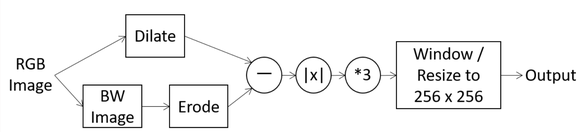
- **Many sketches never get colored** due to the monotonous and the time-consuming nature of the coloring process.
- **This is unfortunate**, since colored drawings are much more interesting than just the lines.
- Given an image of line drawing, **can we train a network to output a 3-channel image** which correspond to the colors of the input, automating the coloring process?

Photo Colorization?

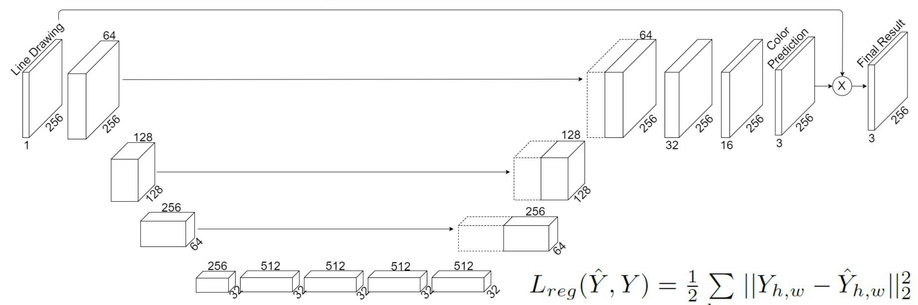
- A notable and significant earlier work in this field is the **photo colorization problem**, where the inputs to the network are the **black and white images**.
- Coloring sketches is **different** from the photo colorization in **3 major ways**:
  1. Sketches only provide the outlines of object, so the input **lacks the brightness info**. In other words, **it is a 3-channel prediction task**, not 2.
  2. Colors of **illustrations are more arbitrary** than real images, so the prediction is much harder- ex. hair color can virtually be any color.
  3. Colorized sketches are **much more sparse** than real images - ex. background is often **monotonic**.

Dataset Generation

- Unfortunately, there are **no publicly available datasets** that only contain illustrations.
- The dataset was created in somewhat of a reverse order: the images are collected first, then converted to line drawings using the pipeline below.
- **0.6 million images** were collected from online repositories, and they were processed to create a 300GB dataset.



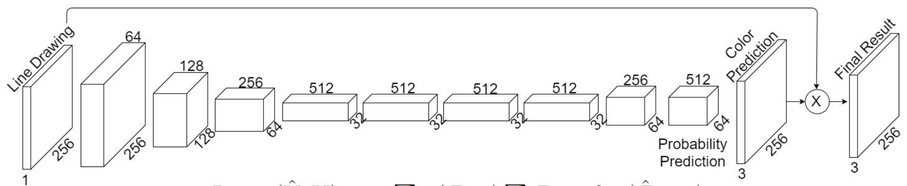
Regression Model



$$L_{reg}(\hat{Y}, Y) = \frac{1}{2}\sum_{h,w} ||Y_{h,w} - \hat{Y}_{h,w}||_2^2$$

- A natural way to formulate the problem is to treat it as a **regression problem.**
- The loss is calculated by taking the **pixel-wise L2 loss** between the predicted and the ground truth.
- The network has a typical autoencoder-type structure - the input image is first **condensed** into images of 32x32, and **"reconstructed"** to a 3-channel image.
- The network also uses **pass-through layers**, as introduced in ***U-net*** (Ronneberger et. al.), to speed up the learning process and for a better performance.
- The final result is created by combining the line drawing input and the color prediction.

Classification Model

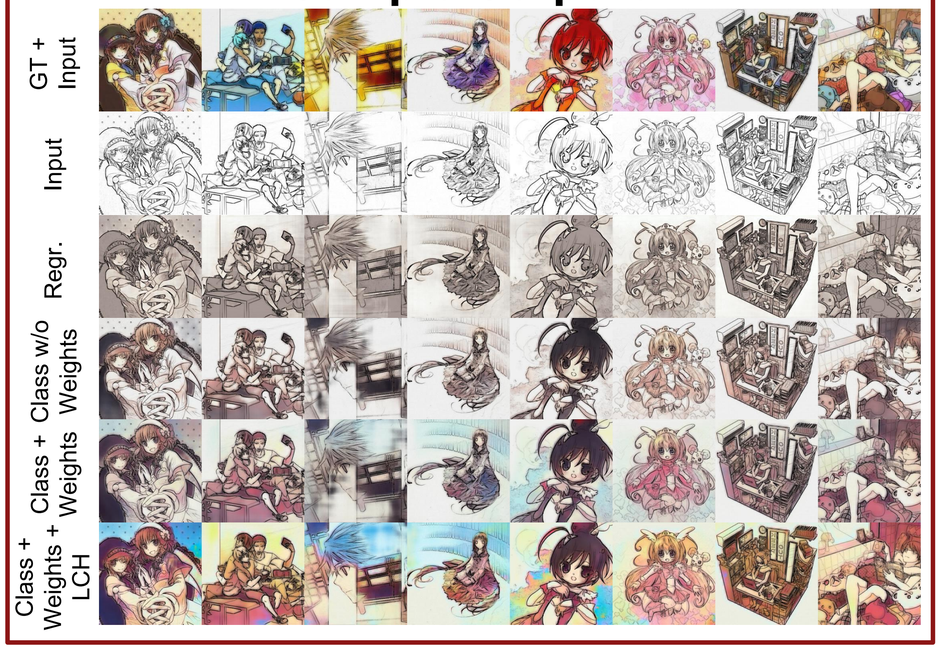


$$L_{cl,wei}(\hat{Y}, Y) = -\sum_{h,w} v(Z_{h,w}) \sum_q Z_{h,w,q} \log(\hat{Z}_{h,w,q})$$

$$w \propto \left((1-\lambda)\tilde{p} + \frac{\lambda}{Q}\right)^{-1}, \quad \mathbf{E}[w] = \sum_q \tilde{p}_q w_q = 1$$

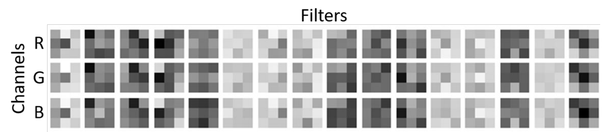
- Another way to view the problem is to set it up as a **classification problem.**
- The image channels are **discretized into 512 classes** (8 equal-sized bins for each channel), and the model is trained to predict the class label for the pixels.
- The network structure is very similar to that of the regression model, but the image is **only reconstructed to ¼ of the original**, mainly due to **memory constraints.**
- To better capture the pre-discretized images, **soft-encoding scheme** was used- the **class label** of a pixel **is a probability distribution** of the classes, not one-hot.
- **3 classification models** are trained
  - With and without **class rebalancing term** ($v()$), which weights the loss contributions from each classes, to account for **class rarity**.
  - Training with images that are converted to the **CIE LCH** color space (**L**ightness, **C**hroma (i.e. saturation), **H**ue)

Sample Outputs



Conclusion

- As in photo colorization problem, the model outputs turn out much better if the problem is treated as a classification problem instead as a regression problem.
- Because illustration colors are somewhat arbitrary, **the models often opted to predict desaturated, sepia colors**
  - Regression model is only able to identify which pixels are part of the same object, but plays "safe" by coloring them sepia with different brightness.
  - In fact, the filters for the last layer is basically identical for the RGB channels!



- **Illustrations have many monotonic colors**, like a white background, that **heavily affect the loss function**, so class rebalancing term improved the output.
- Model outputs significantly **improved when the prediction was made on CIE LCH color space,** presumably because **lightness and chroma channels are more consistent** than the RGB values for the illustrations.
- Since even the best model seems to **output desaturated pixels**, we are working on training the network with **non 0-1 loss matrix**, to penalize the model more for predicting wrong chroma values.
- In the future, we would like to **tweak the network structure** as well.