CS 6955
Final Project Paper
Fall 2019

Deep Learning
Final Project Paper

Team members:

- Al Obaidi, Ali
- Liechty, Matthew
- Nelson, Soren

Data history:

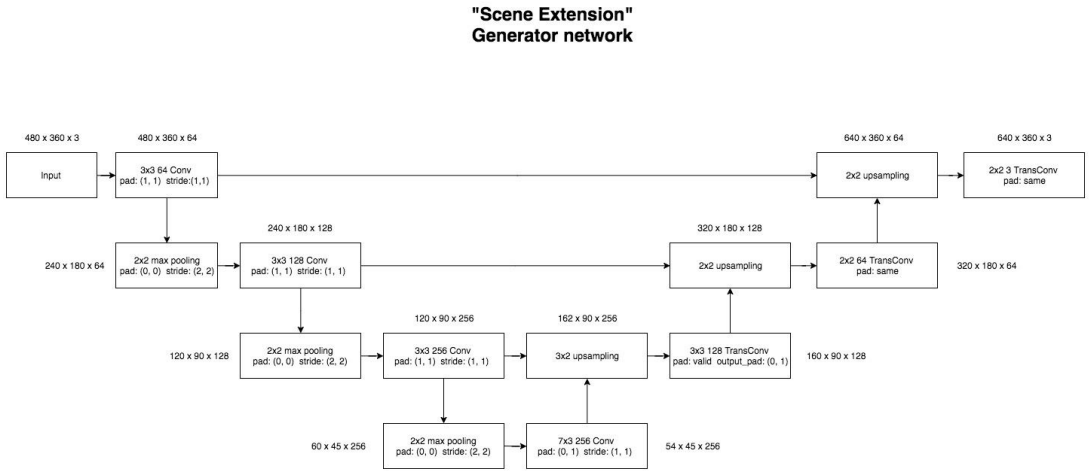
We started out with 640x360x3 images cropped from images in the dataset for the Kaggle challenge "Painter by the Numbers" that was derived from WikiArt images (1). We've selected a small subset from the whole set based on images that have a 4 to 7 ratio when dividing the width over the height of the image, to remove portrait images and images that were too wide for our purpose, since we wanted to scale down our dataset to 640x360x3 without affecting the quality of the image content. However, over time we realized that these images were hard for the network to work with and that reducing these images to 320x180x1 (grayscale), which produced much better results.

Architecture history:

Generator

Our initial problem in creating a generator network for producing a 16:9 image given a 4:3 was how to increase the pixel space to be wider.

The original design we came with up for the generator network used 3x2 upsampling to stretch the feature map that was created after a few convolutional and max pooling layers. This design looked something like the following image, which was designed using U-Net type architecture introduced in (2) and made popular in the Pix2Pix paper (3).

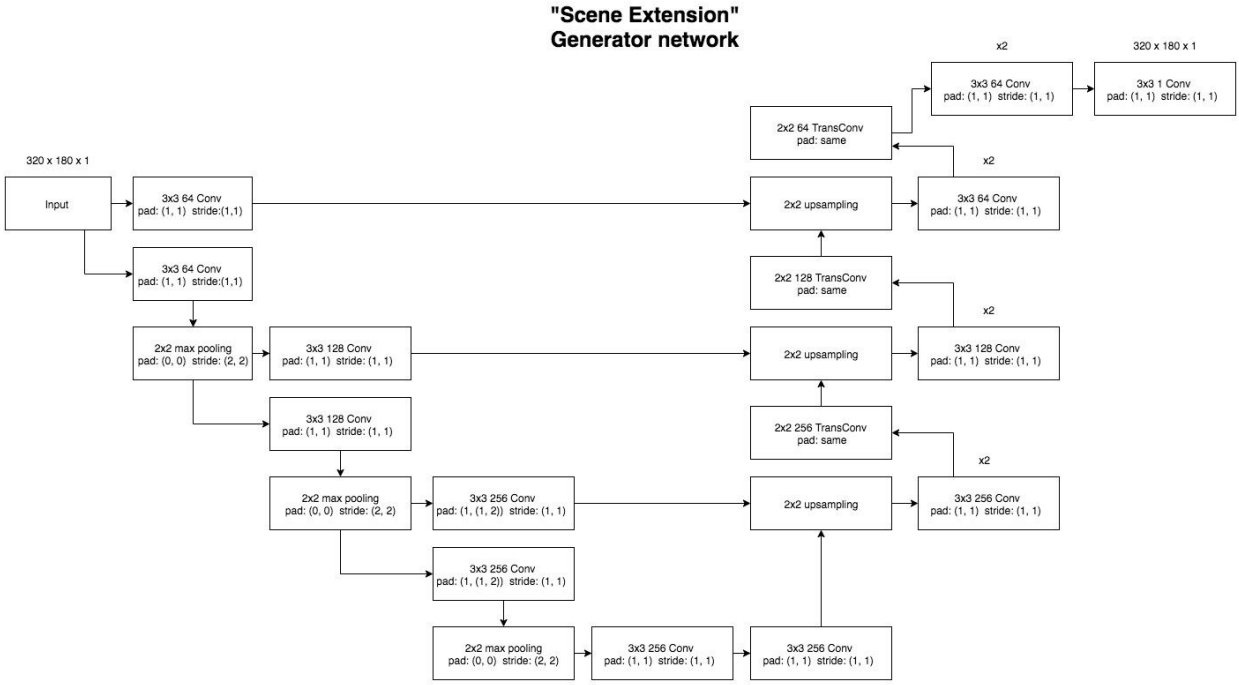


However, this generator network tended to just use a stretched version of the original image to fit in the sides. It lacked continuity with its edges didn't produce a plausible image.

Next, we had the idea of using random static to create the extra width. Before images went into training, we would append two side bars of random static of size 80x360x3 to each side of the image. This started to work much better with both cGAN and L1 loss (as suggested in the Pix2Pix paper [3]) and the network began to learn to convert some of the static to blurry continuations of the original image.

However, the sheer size of the images prevented getting higher quality representations. Thus, as described in the Data History section above, we experimented and reduced the image sizes to 320x180x1. We started to see much better results.

Our final generator used the following architecture. Again, this was inspired by the U-Net architecture (2), with some changes. Each convolutional output uses ReLu activation.



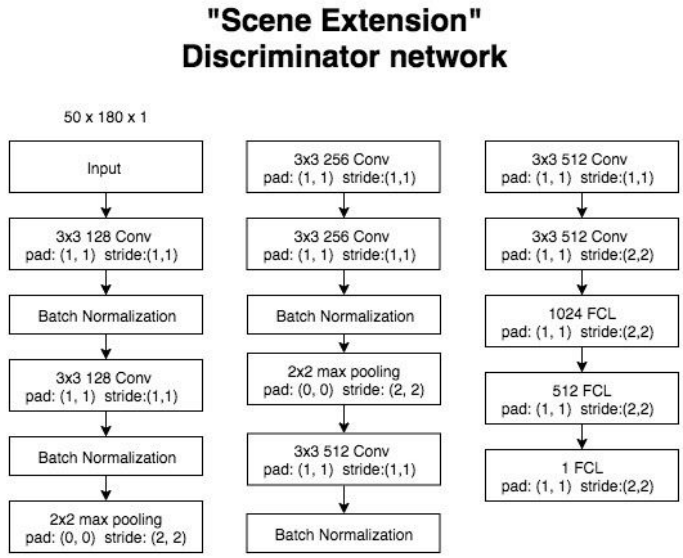
Discriminator

Originally, we used a PatchGAN discriminator loss on the whole generated image as described in the Pix2Pix paper (3). However, over time we evolved to using a normal discriminator GAN loss which seemed to produce better results.

When we switched from using upsampling to increase the pixel space to using static, we started using discriminator GAN loss on the static edges exclusively since the L1 loss in the generator was adequate for replicating the 4:3 piece of the image. To do this, we created two different instances of a discriminator network with the same architecture: the first to evaluate the output of the left bar of the image and the second to evaluate the right bar of the image. We also let the bars “bleed” into the original image 10 pixels so as to encourage continuity with the 4:3 portion.

Our final discriminators (one for the left generated bar, one for the right) used the following architecture. The architecture is based off a similar architecture patterns as VGG-16 (4) with some batch normalization layers. Each convolutional and fully connected output is activated using leaky ReLu (besides the last one for some reason).

CS 6955
Final Project Paper
Fall 2019

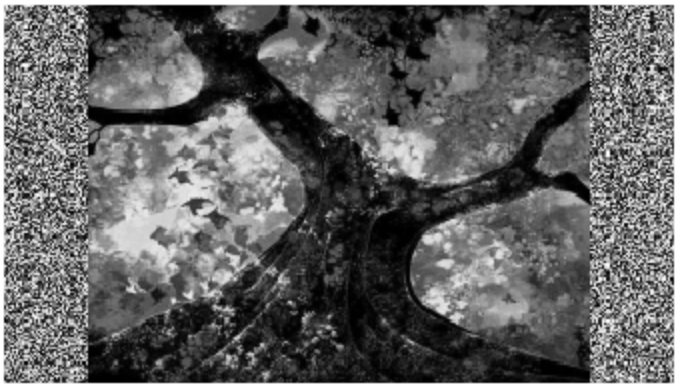


Results:

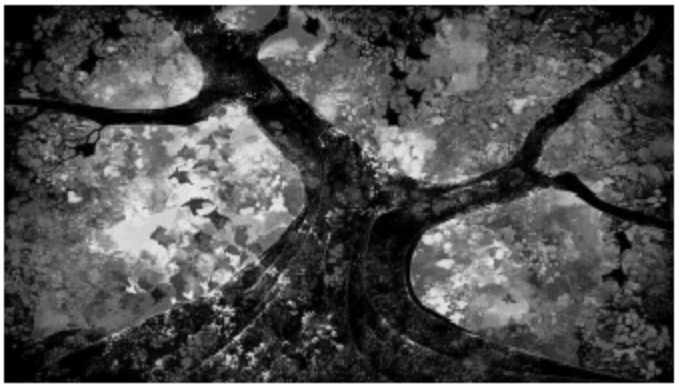
Our results are shown below. We trained on 500 images for 2,000 epochs with both models. In the full-sized images, you can see the static placed on the sides of the cropped image as discussed above. Clearly, the L1 loss model overfit the training data. It may be that the L1 predictions would have worked better with more training examples, but we wanted to show a direct comparison with the examples produced by the GAN with only 500 examples. Since the GAN training took so long, we didn't use more than 500. The L1 and GAN loss model did extremely well in most images, although it did struggle on images with borders. Our results were as good, if not better than the similar work done on classic films (5).

CS 6955
Final Project Paper
Fall 2019

Cropped Image With Static

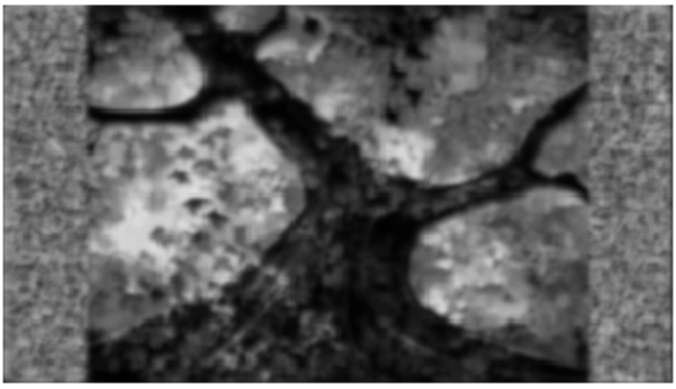


Input Image

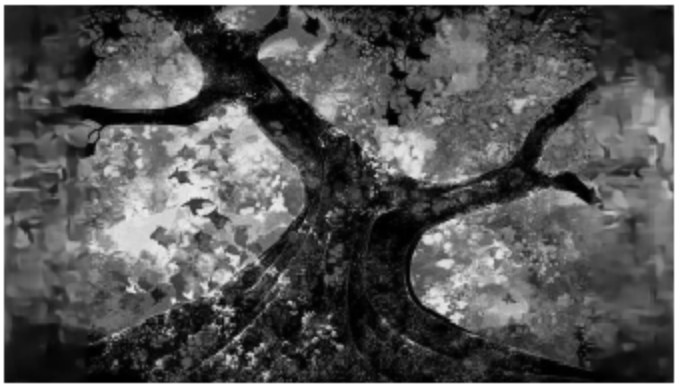


CS 6955
Final Project Paper
Fall 2019

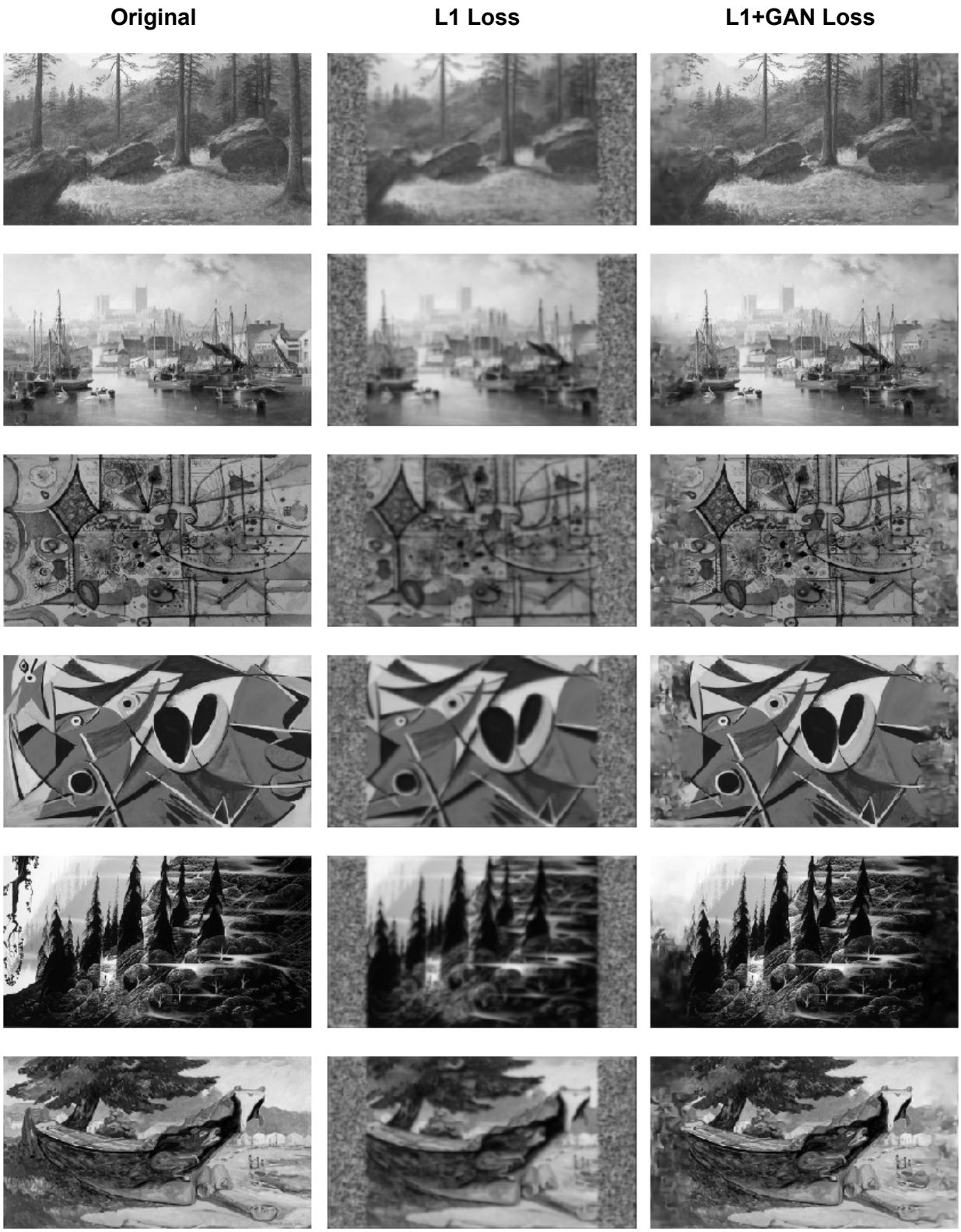
L1 Loss



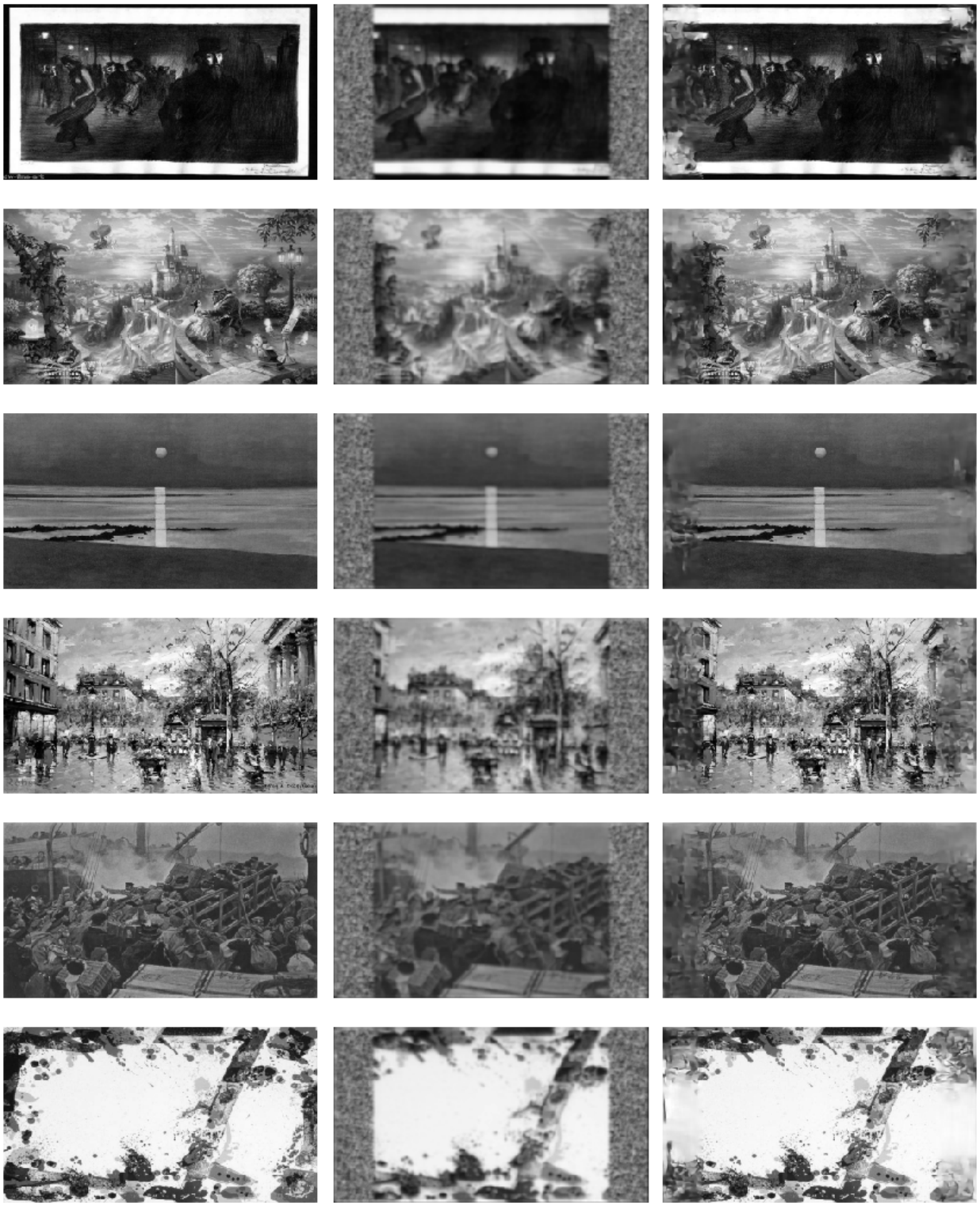
L1+GAN Loss



CS 6955
Final Project Paper
Fall 2019



CS 6955
Final Project Paper
Fall 2019



CS 6955
Final Project Paper
Fall 2019

CS 6955
Final Project Paper
Fall 2019

References:

1. “Painter by the Numbers”. Kaggle. https://www.kaggle.com/c/painter-by-numbers/data
2. U-Net: Convolutional Networks for Biomedical Image Segmentation. https://arxiv.org/pdf/1505.04597.pdf
3. Image-to-Image Translation with Conditional Adversarial Networks. https://arxiv.org/pdf/1611.07004.pdf
4. Very Deep Convolutional Networks for Large-Scale Image Recognition. https://arxiv.org/pdf/1409.1556.pdf
5. Remastering Classic Films in TensorFlow wit Pix2Pix https://medium.com/hackernoon/remastering-classic-films-in-tensorflow-with-pix2pix-f4d551fa0503?