Disclaimer: This presentation is a work of fiction written from the perspective of a 2020 researcher traveling back in time to mid 2013 to share some 2020 xNN based application ideas; references to credit the actual inventors of the various ideas is provided at the end

Generative Painter

Abhijit Balaji
axb190013@utdallas.edu
May 1, 2013

Disclaimer!

- The main paper used for this presentation:

Auto-painter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks

By:

Yifan Liu, Zengchang Qin, Zhenbo Luo, Hua Wang

Paper link: https://arxiv.org/abs/1705.01908

Disclaimer!

- Since this work builds on conditional GANs I am going to assume that the people in 2013 are familiar with GANs and conditional GANs so as to avoid the presentation being too lengthy
- I am going to focus this presentation on key parts such as the loss functions, architecture used etc.
- The model was not readily available and I tried my best to reproduce the results from the paper using the references from here

Disclaimer: This presentation is a work of fiction written from the perspective of a 2020 researcher traveling back in time to mid 2013 to share some 2020 xNN based application ideas; references to credit the actual inventors of the various ideas is provided at the end

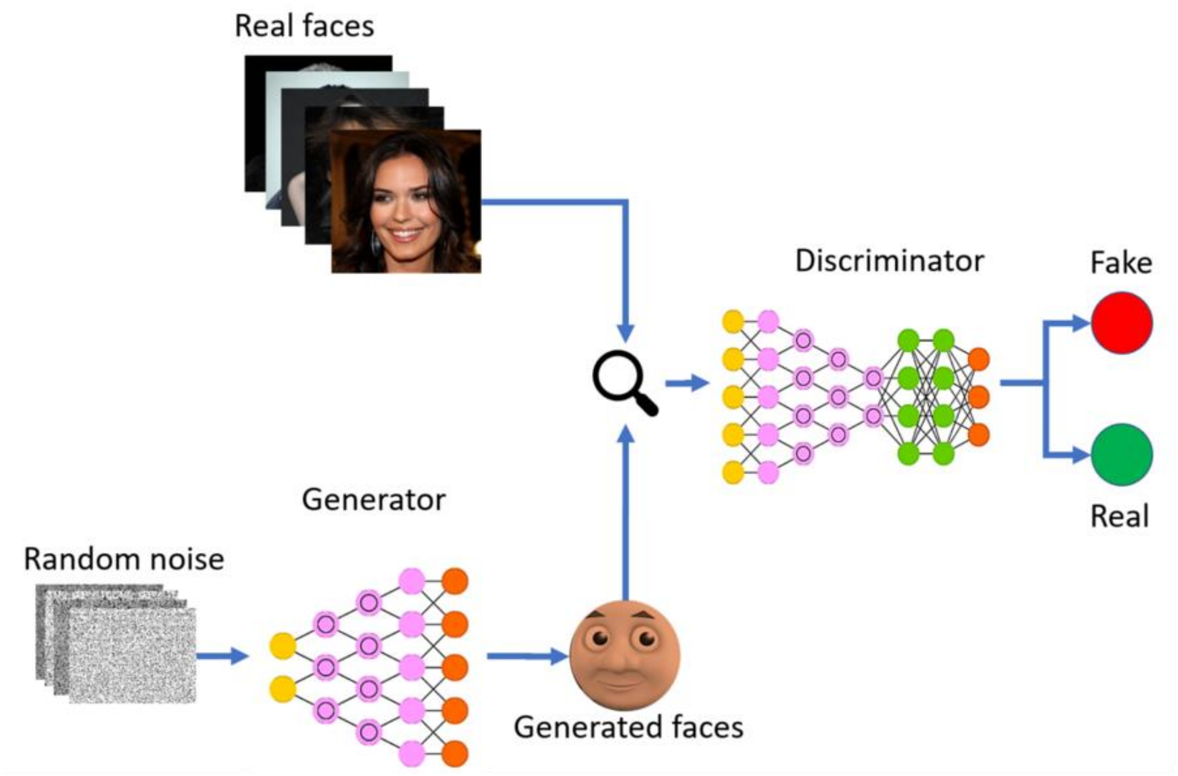
Motivation

- Coloring sketches automatically using neural networks opens a wide field of applications.
- This is especially useful to artistes and graphic designers as it helps them see the sketches from a new pair of eyes

  Current methods:
  - Most of the current approaches uses GANs and variations of it.
  - GANs are hard to train
  - We can use conditional GANs to condition our GAN outputs

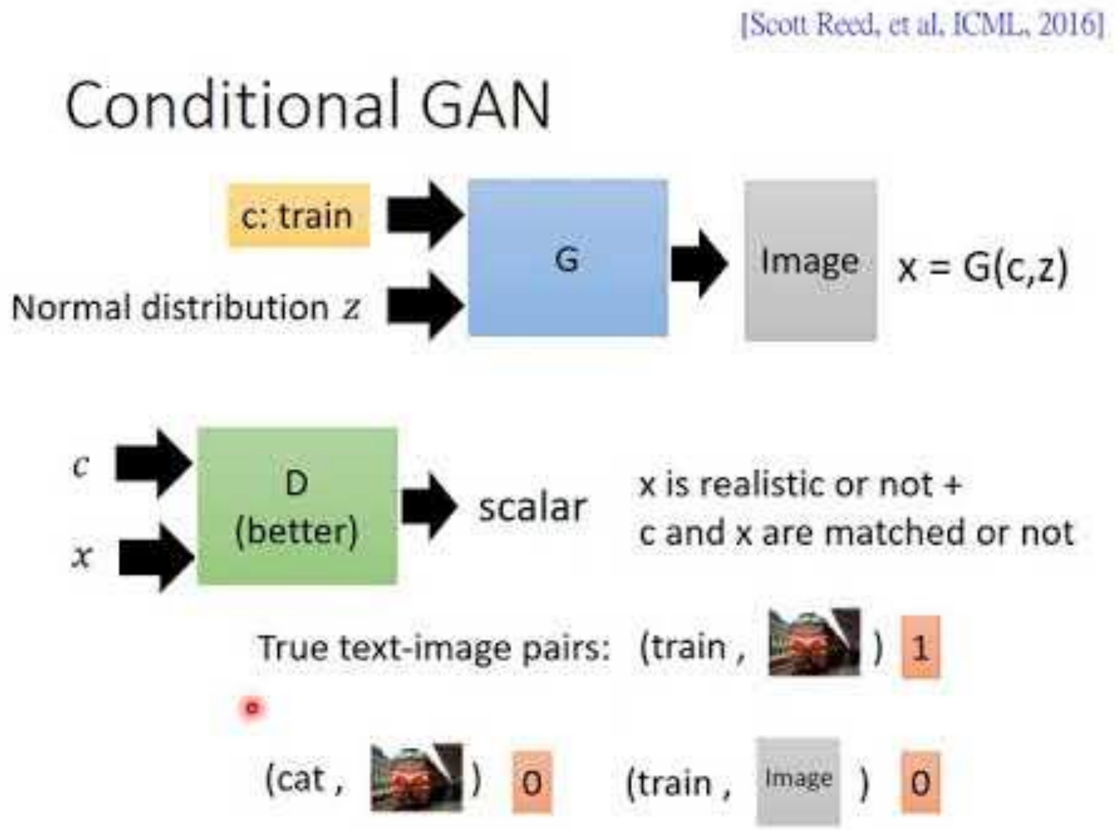
Key Insight

- Generative Modeling:
  - This can be boiled down to a classification problem when we introduce the concept of a discriminator
  - We can have two models. One generator and one discriminator (which discriminates whether the input is from the real dataset or from the generator).
  - Thus we can train the two models end to end and jointly like a classification problem using the cross entropy loss.



Proposed Approach

- Given a black & White sketch, our model can generate a painted colorful image.
- We use Conditional GANs (CGANs) to model our generative painter
- We use 4 Loss functions to train the CGAN:
  - Normal GAN loss
  - Pixel Level Loss
  - Feature Level loss
  - Total Variation loss



1. GAN Loss

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

- Objective function is a min max function (Game theory)
- Think of this as a police thief game
- The discriminator tries to maximize the above using gradient ascent
- The generator tries to minimize the above using gradient descent
- Our objective is to reach Nash Equilibrium

Ok now we have a loss function which looks just fine! so why the other 3 losses?

- It is not as easy as it sounds to train a GAN using the above loss function
- GAN training collapses easily!
- We need to stabilize GANs
- That's the main purpose of the other 3 loss functions!

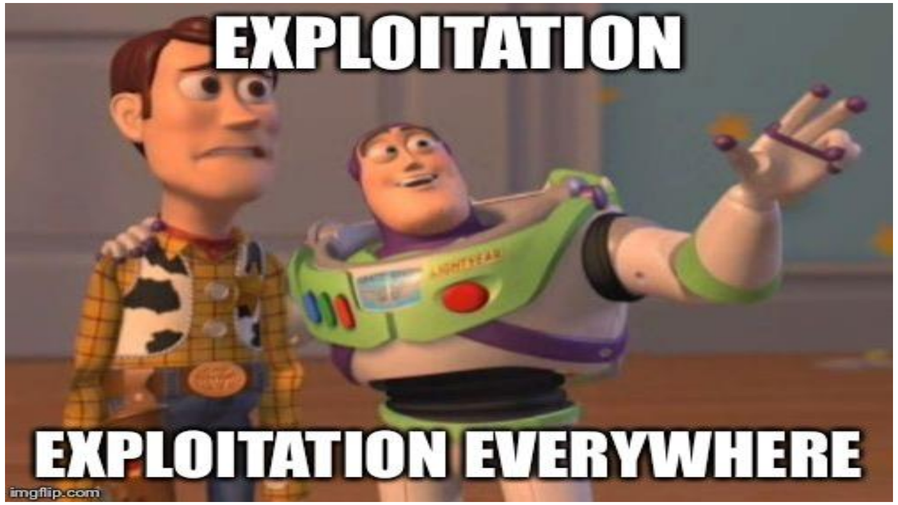
2. Pixel level Loss

$$L_p = \mathbb{E}_{x,y \sim p_{data(x,y)}, z \sim p_{data}(z)} [\|y - G(x,z)\|_1]$$

- This is just the L1 distance between each pixel of target color image and the generated color image.
- Why?: Intuitively this should help the model learn the correct color shades.
- What is missing in this?: Even though this will help the model learn the correct color shades, it doesn't correlate which features should get which color shades.
- The feature and color correspondence is not properly represented by the Pixel level Loss.

Learning the relation between image features and colors

- We know that CNNs extract features from the image
- We can exploit this fact to model the relation between image features and colors.



3. Feature Level Loss

$$L_f = \mathbb{E}_{x,y\sim p_{data(x,y)}, z\sim p_{data}(z)} \left[\|\phi_j(y) - \phi_j(G(x,z))\|_2 \right]$$

- This is similar to Pixel level loss. But we use L2 distance instead of L1
- Instead of calculating it on the images we pass the images through VGG-16 and calculate it on the 4th layer output VGG-16. This should have some information about the features in the image.
- Why 4th layer?: Well it's through experimentation we found that this works best for our case :P

4. Total Variation Loss

$$L_{tv} = \sqrt{(y_{i+1,j} - y_{i,j})^2 + (y_{i,j+1} - y_{i,j})^2}$$

- We use this loss to prevent color mutation.
- This constraints the pixel changes in the generated results and encourages smoothness.
- Think of this like a form of regularization

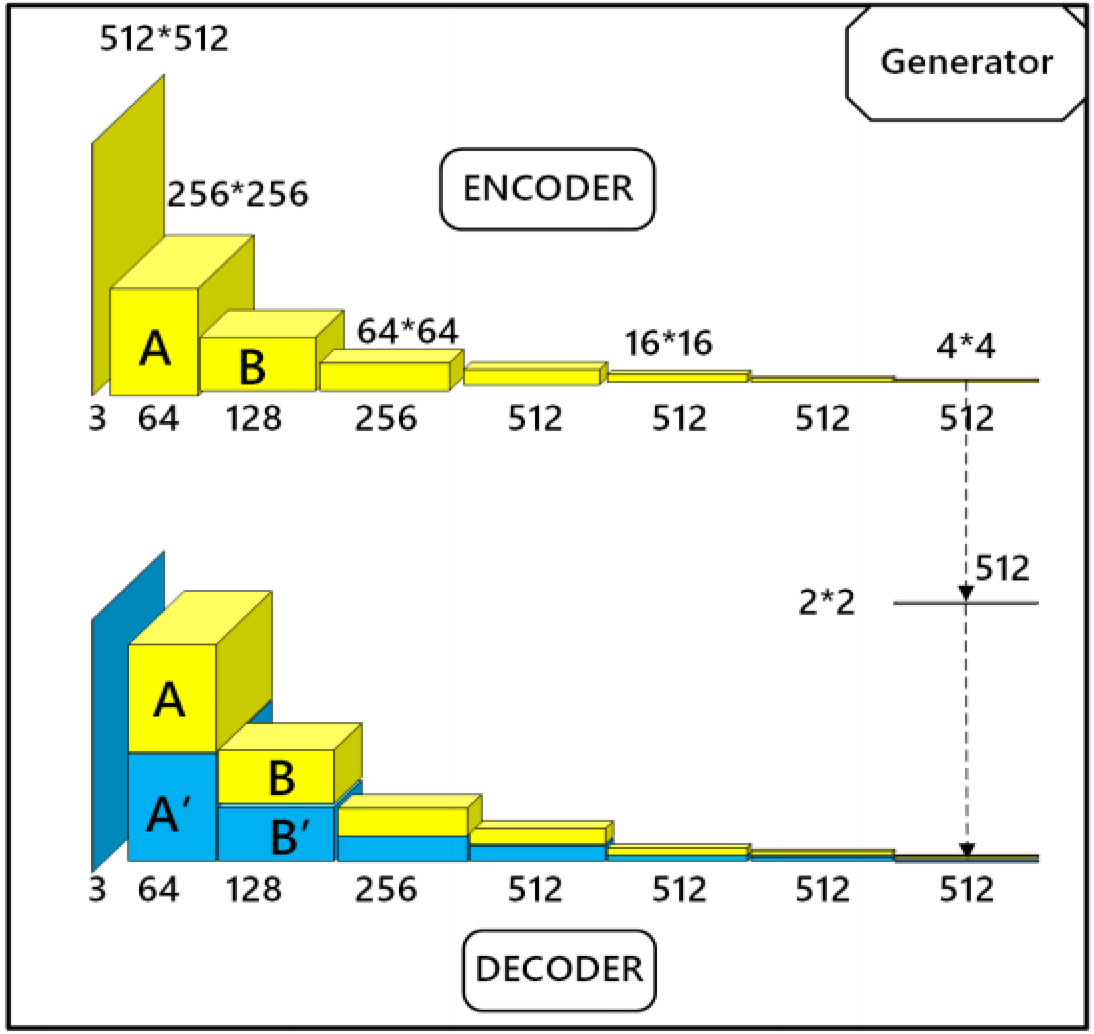
Combining the 4 losses

$$L = w_p L_P + w_f L_f + w_G L_G + w_{tv} L_{tv}$$

- We train the CGAN using a weighted combination of the 4 losses mentioned
- The weights are adjusted according to how important each loss is.
- We use the following weights:
  - Pixel Level Loss weight: 100
  - Feature Level Loss weight: 0.01
  - Total Variation Loss weight: 0.0001
  - Normal Gan Loss weight: 1.0

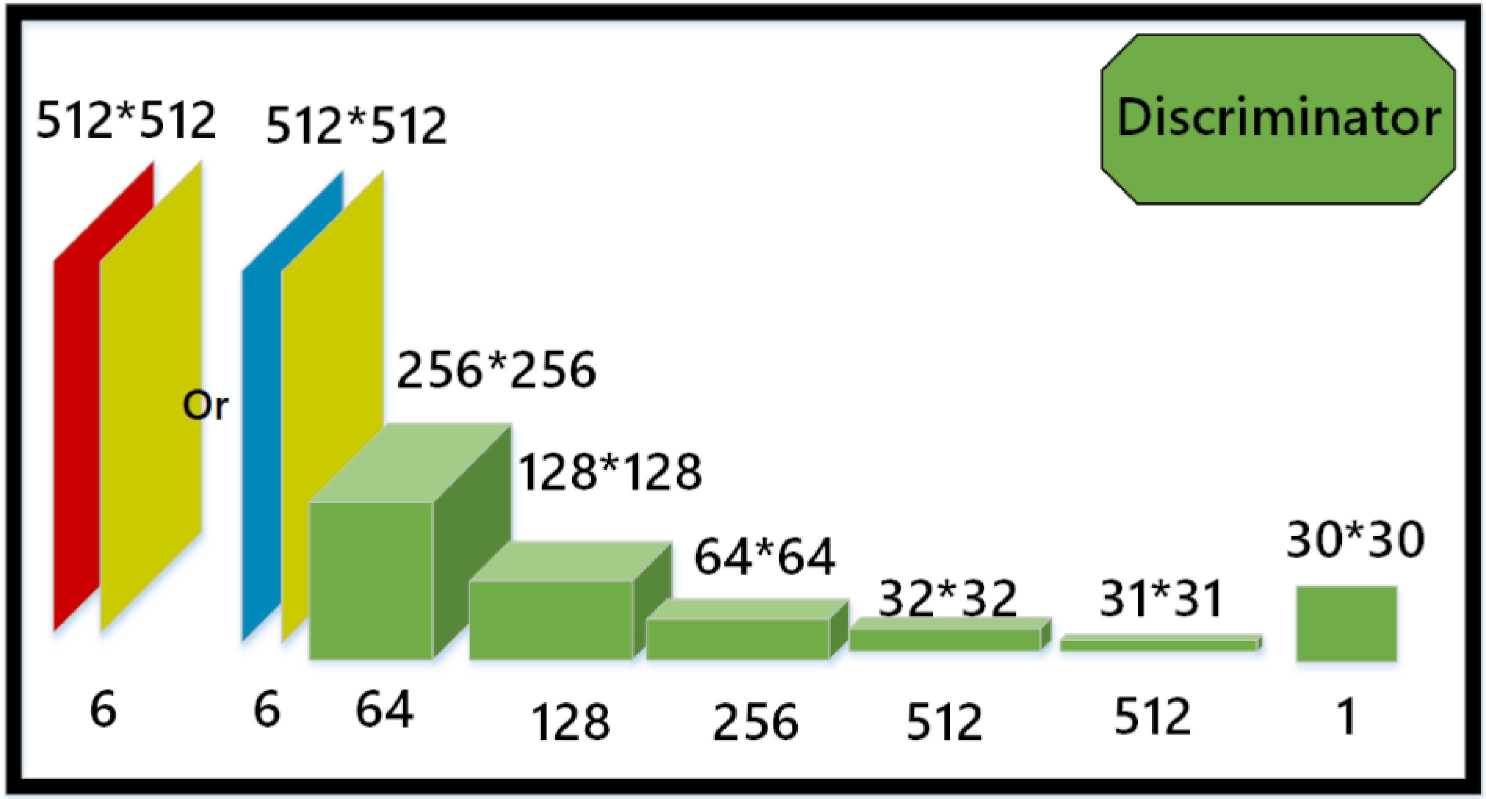
Generator Architecture Inspiration: U-Net

- It consists of an Encoder which reduces the 512 x 512 x 3 image to a latent dimension of 2 x 2x 512 using comvolution operations
- The decoder takes this latent dimension as input and upsamples it to 512 x 512 x 3 using Transposed convolution operation
- To improve the performance of the decoder, we concatenate the corresponding layers of the encoder with the decoder
- The generator is conditioned on the input sketch (obviously :P)



Discriminator Architecture

- The discriminator takes a 512 x 512 x 3 image (either from the training set or the generator output)
- This maps this input to a latent dimension of 30x30x1
- Each element of that gives the probability of being real for a pair of corresponding patches from the input sketch and the colored image.

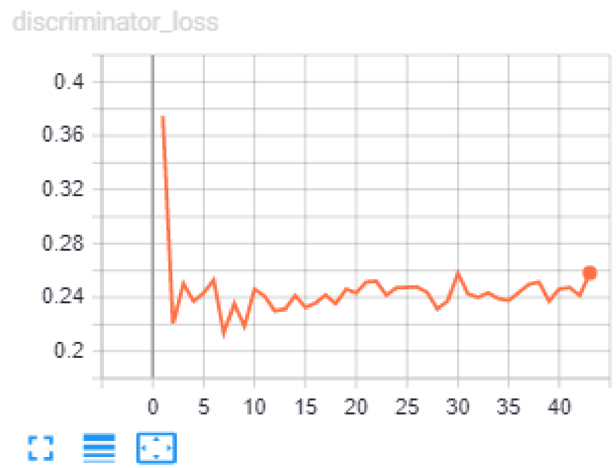


Dataset details

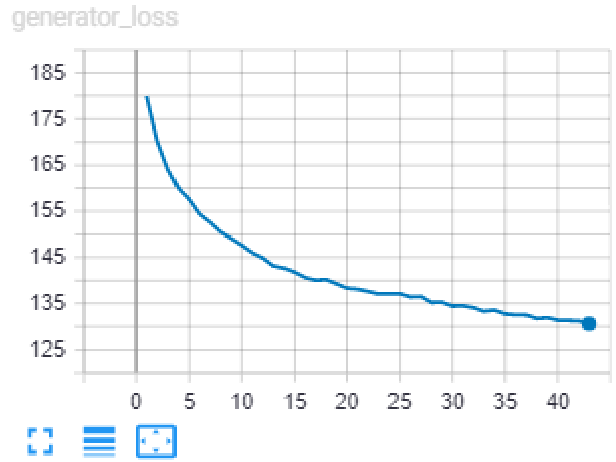
- Anime Sketch Colorization Pair from Kaggle
- This Dataset consists of Anime sketches and their corresponding image after colorization
- This dataset is perfect for our case
- This dataset has 14,224 pairs of training sketches and colored images
- 3545 pairs of test sketches and colored images

Training progress

discriminator_loss



generator_loss



Took approximately 12 hrs to train!

Code link

https://github.com/Abhijit-2592/cnn_project3

Disclaimer: This presentation is a work of fiction written from the perspective of a 2020 researcher traveling back in time to mid 2013 to share some 2020 xNN based application ideas; references to credit the actual inventors of the various ideas is provided at the end

Resulting good images

Input

Output

Resulting good images

Input

Output

Resulting good images

Input

Output

Resulting bad image

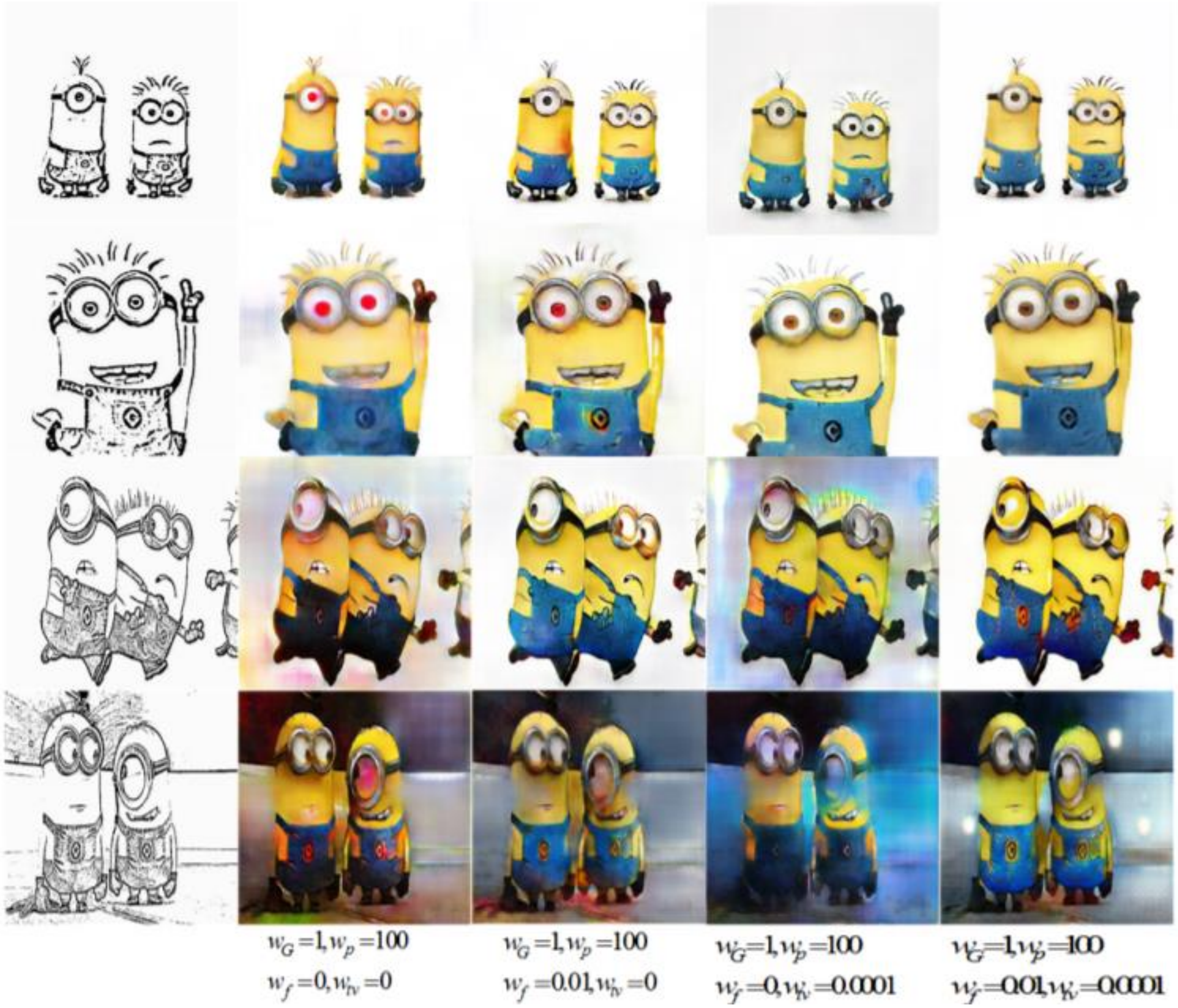
Input

Output

Disclaimer: This presentation is a work of fiction written from the perspective of a 2020 researcher traveling back in time to mid 2013 to share some 2020 xNN based application ideas; references to credit the actual inventors of the various ideas is provided at the end

Results

Results of Ablation experiments on Different values of weights for the loss function



Next Steps

- The generative painter even though performs fairly on some sketches it fails in others.
- The color looks kind of washed away in most generated images and sometimes look unnatural.
- In future we will try to improve this work by increasing the stability of the generated color image so that the images look as natural as possible

References

- Note: As stated above, this presentation is a work of fiction; the following are the actual inventors of the ideas described in this presentation

- Liu, Yifan, Zengchang Qin, Zhenbo Luo and Hua Wang. “Auto-painter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks.” *ArXiv* abs/1705.01908 (2017): n. pag.
- Ronneberger, Olaf, Philipp Fischer and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” *MICCAI* (2015).
- Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville and Yoshua Bengio. “Generative Adversarial Networks.” *ArXiv* abs/1406.2661 (2014): n. pag.
- Radford, Alec, Luke Metz and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.” *CoRR* abs/1511.06434 (2016): n. pag.
- Mirza, Mehdi and Simon Osindero. “Conditional Generative Adversarial Nets.” *ArXiv* abs/1411.1784 (2014): n. pag.
- Code reference: https://github.com/sanjay235/Sketch2Color-anime-translation
- Other references: https://machinelearningmastery.com/

Thank You!