Image Colorization Using CycleGAN

Zahil Shanis Yash Saraf Sai Varun

Generative Adversarial Networks

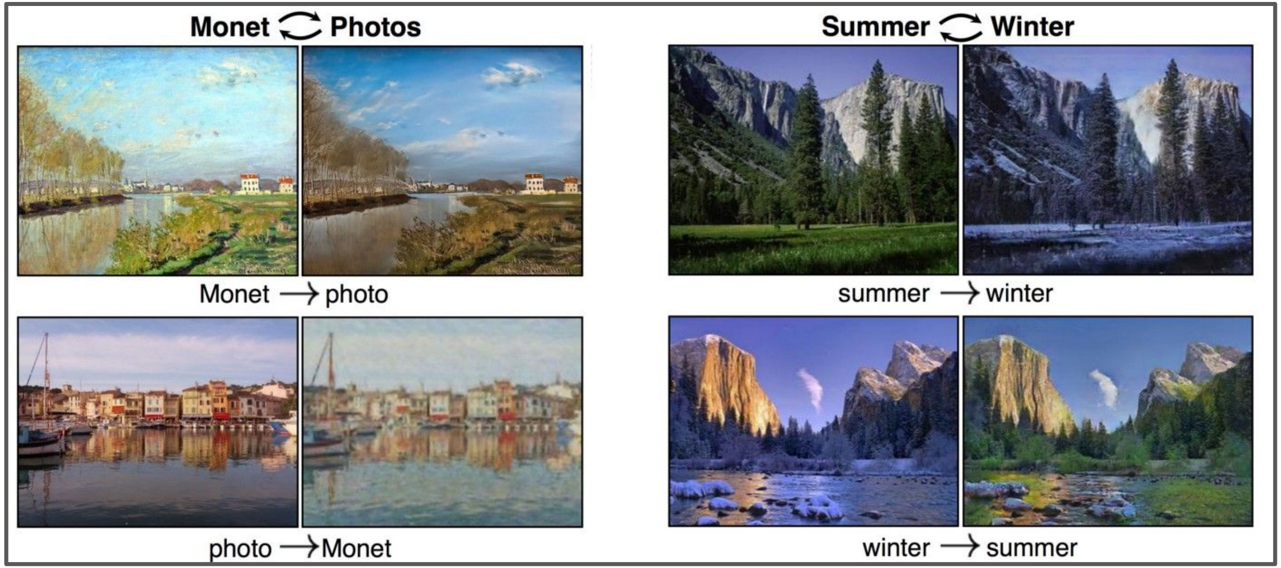
- **Generative models** with two competing differentiable functions, represented by neural networks.
- **Generator**: Generates data from random noise using feedback from discriminator.
- **Discriminator**: A classifier to identify real data from fake (synthesized) data.

We train the generator to create data towards what the discriminator thinks is real.

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

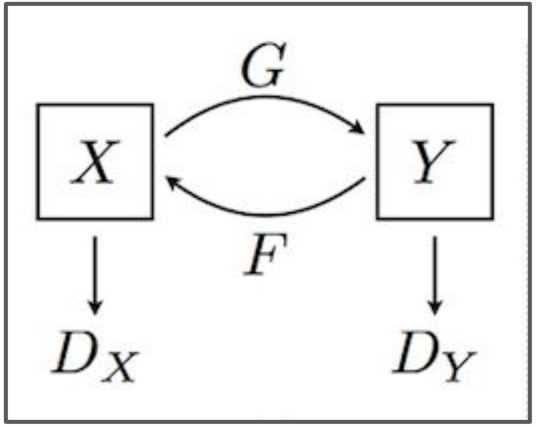
CycleGAN

- Proposed by Jun-Yan Zhu, Taesung Park, Phillip Isola and Alexei A. Efros
- Performs unpaired image to image translation.
- Unpaired translation - doesn’t require a training set of aligned image pairs.
- Cycle GAN can translate an image from a source domain X to a target domain Y in the absence of paired examples.



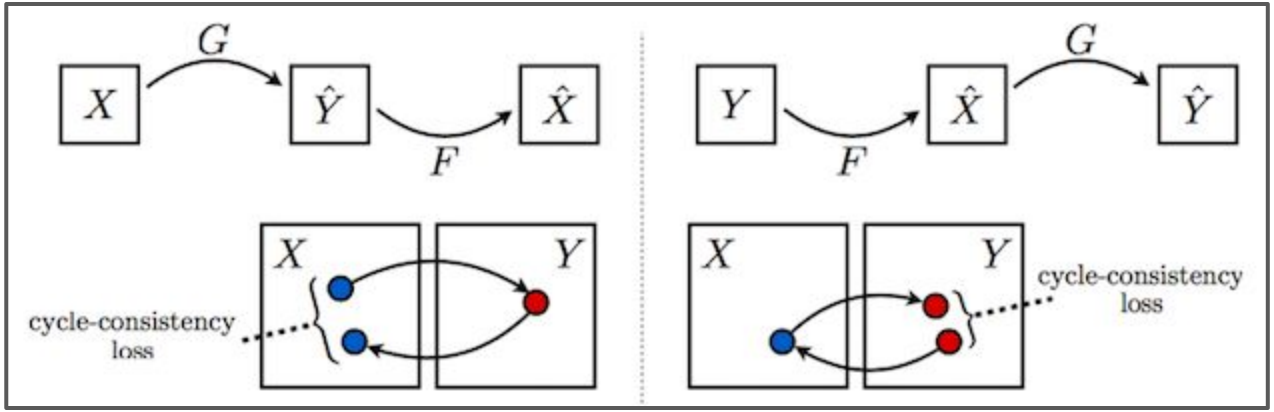
Cycle GAN Architecture

- Architecture consists of two mappings: G : X -> Y and F : Y -> X.
- A generator G is used to translate real image from domain X to domain Y.
- A generator F is used to translate real image from domain Y to domain X.
- Discriminators (D_x and D_y) are used to discriminate real and fake images at respective domains.



Cycle GAN Cost Function

- In addition to the Generator and Discriminator losses, CycleGAN uses one more type of loss called Cycle Consistency Loss.



- This enforces that the input and generated output are recognizably the same.
- Final Objective Function is given by:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F),$$

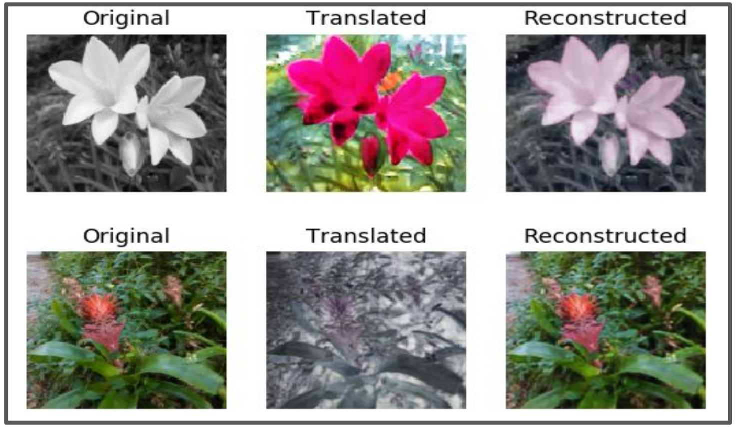
Image Colorization with Cycle GAN

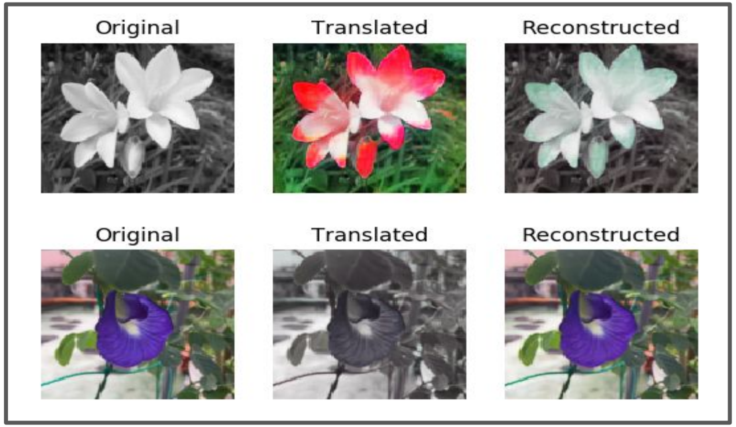
- Colorize gray scale images using Cycle GAN architecture.
- Training on unpaired flowers dataset - domain X as gray scale images and domain Y as color images.

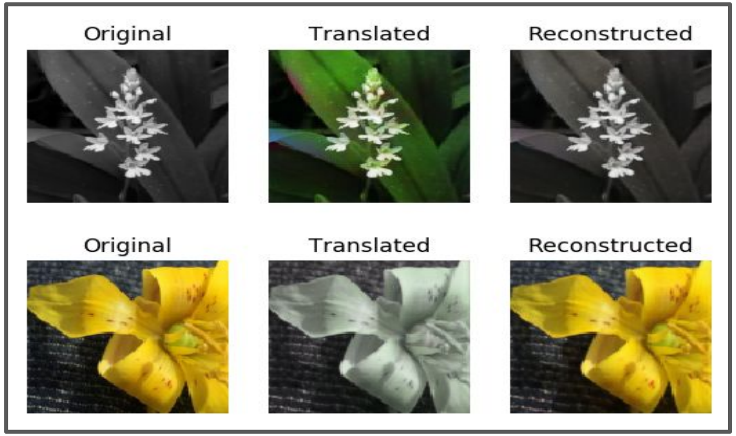
Network Architecture

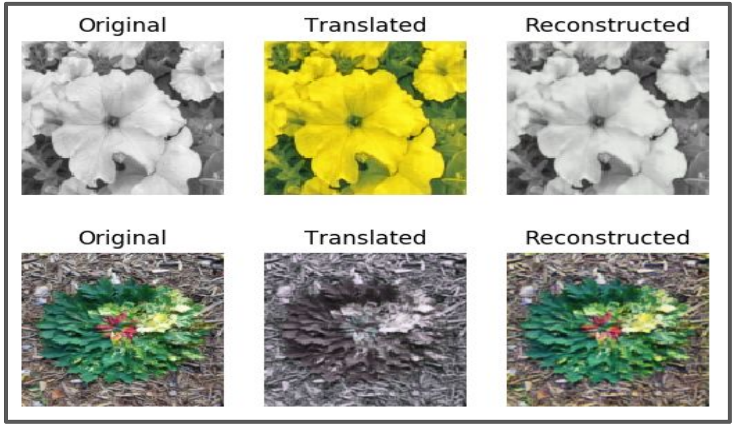
- Generator: A UNet like architecture with an encoder, transformer and decoder.
- Discriminator: PatchGANS which look at a “patch” of the input image, and output the probability of the patch being “real”.
- Trained with a batch size of 1 with Adam as the optimizer.

Image Colorization Results









Network Modifications

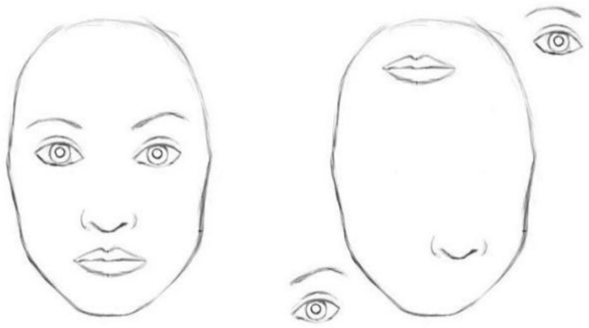
1) Cycle GAN with Stochastic Generators

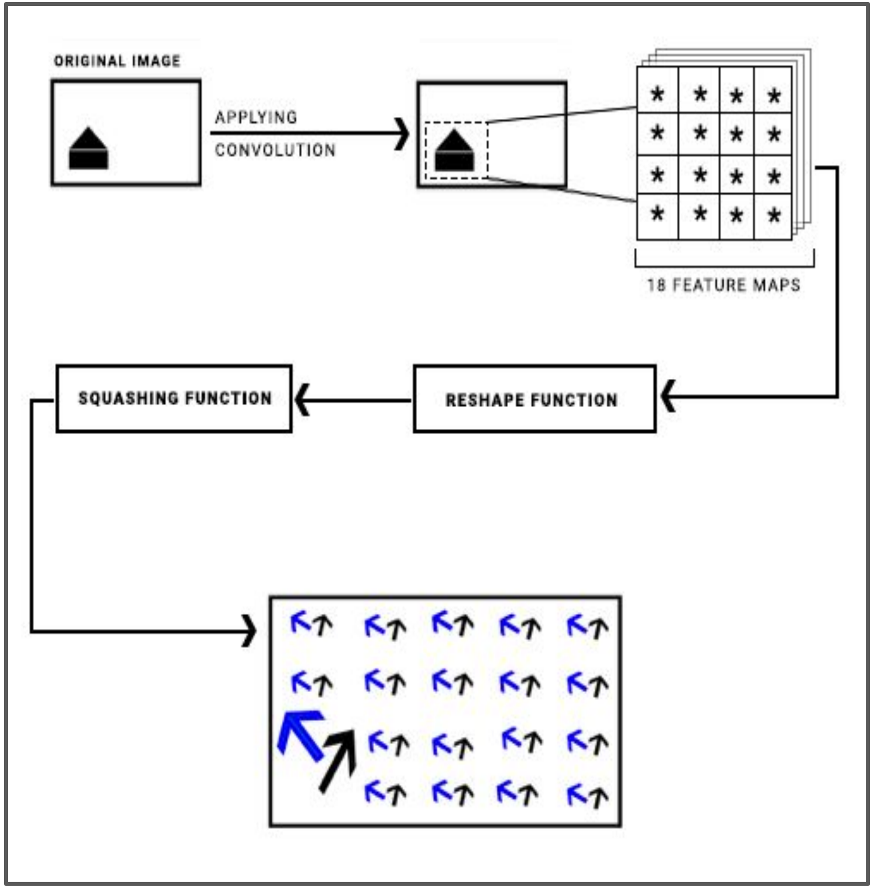
- Inter domain mapping from unpaired data need not always be one-to-one or deterministic.
- Stochastic Cycle GAN - Generates multiple color images for a single grayscale image.
- Can be achieved by modifying the generator G_{AB} to take a vector of noise and a sample from the source domain, and generates a non-deterministic sample in the target domain.
- With different noise $z \sim p(z)$, model can generate different domain B mappings.
- Inspired from **Conditional Instance Normalization** for Style Transfer paper by Huang et al.
- We are working on implementing this.

Network Modifications

2) Cycle GAN with Capsule Nets

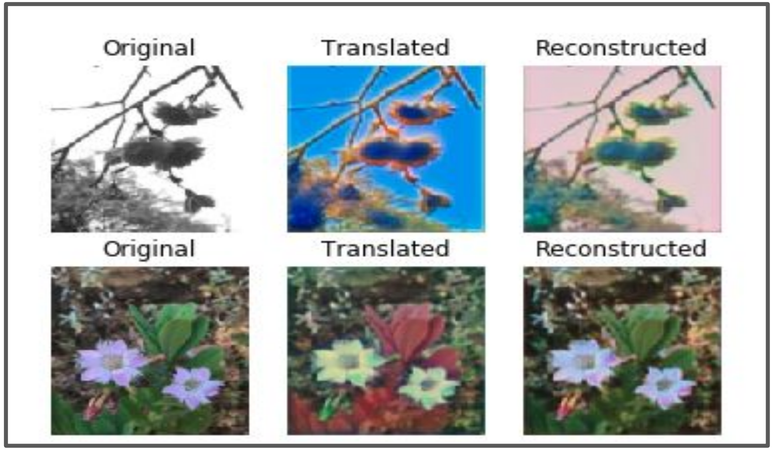
- In CNN, Pooling layers are used to increase the field of view and predict higher order features by combining values.
- Use of Capsule Nets helps preserve hierarchical pose relationships between object parts.

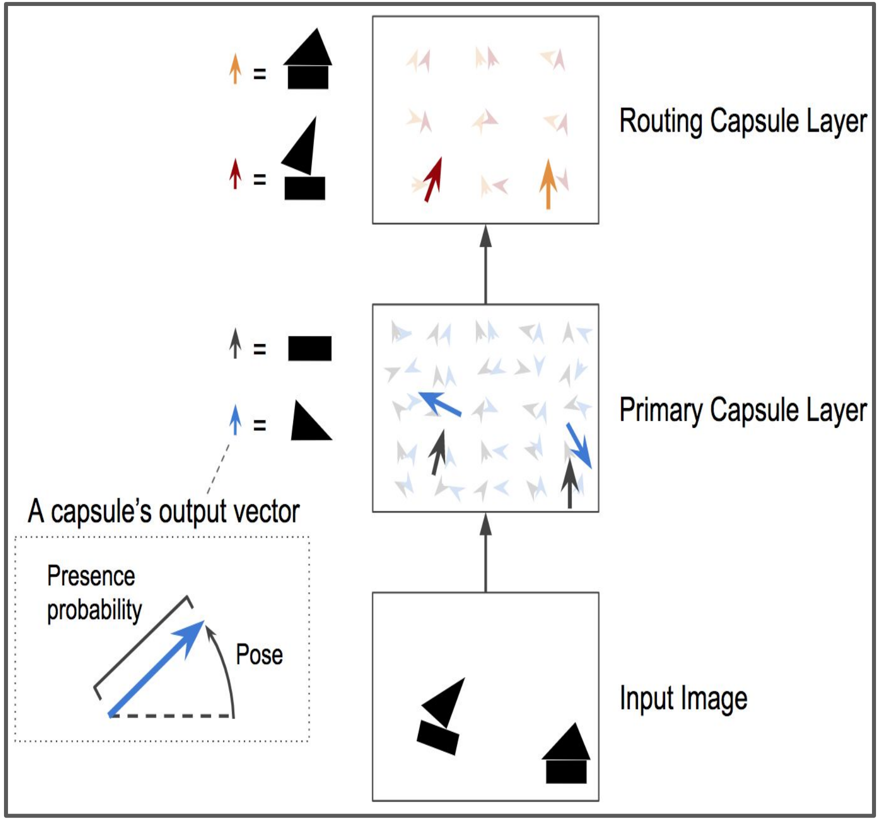




Network Modifications

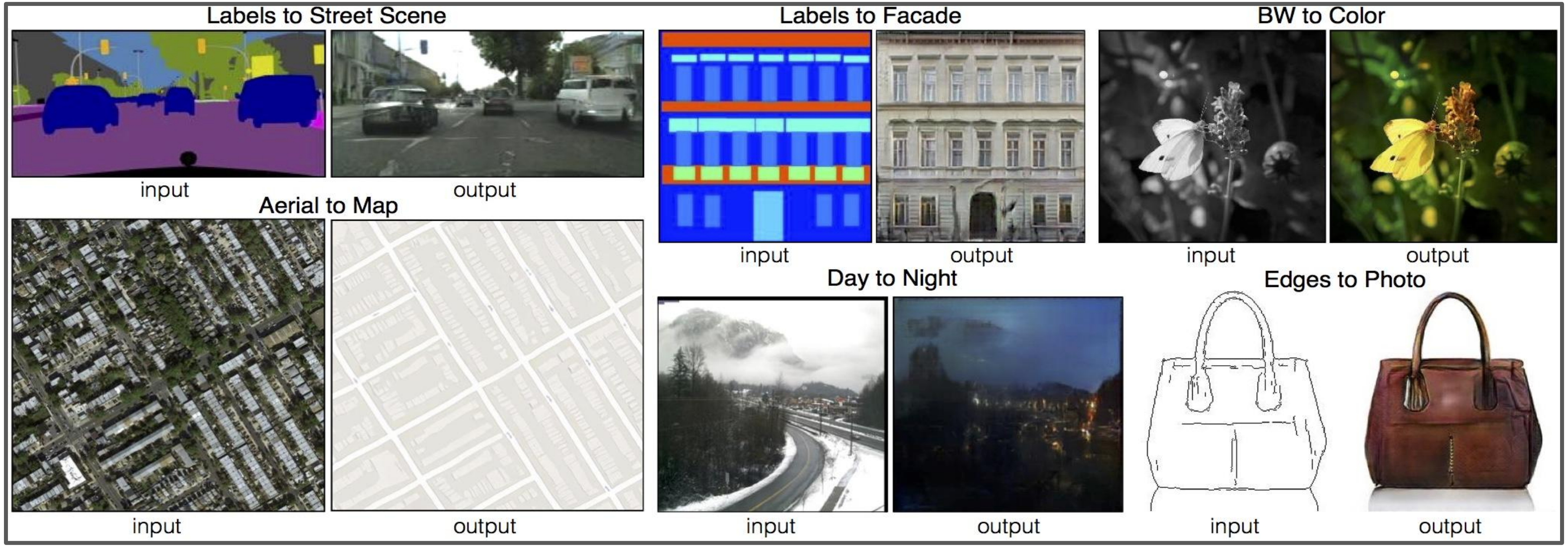
- Capsule Networks and GANs - Using a Capsule Network as a discriminator to better train the model to understand spatial differences.
- Papers CapsGAN, and CapsuleGAN, takes forward the idea by replacing the DCGAN discriminator with CapsuleGANs.





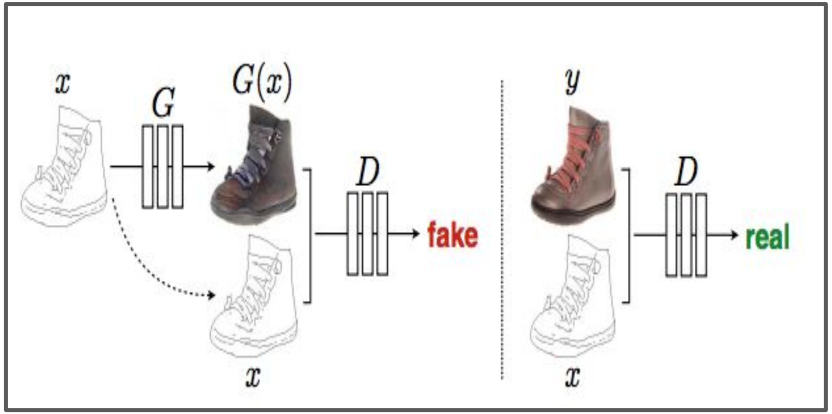
Conditional GAN (pix2pix)

- Performs paired image to image translation.
- In an unconditioned generative model, there is no control on modes of the data being generated.
- In the CGAN, the generator learns to generate a fake sample with a specific condition or characteristics rather than a generic sample from unknown noise distribution.



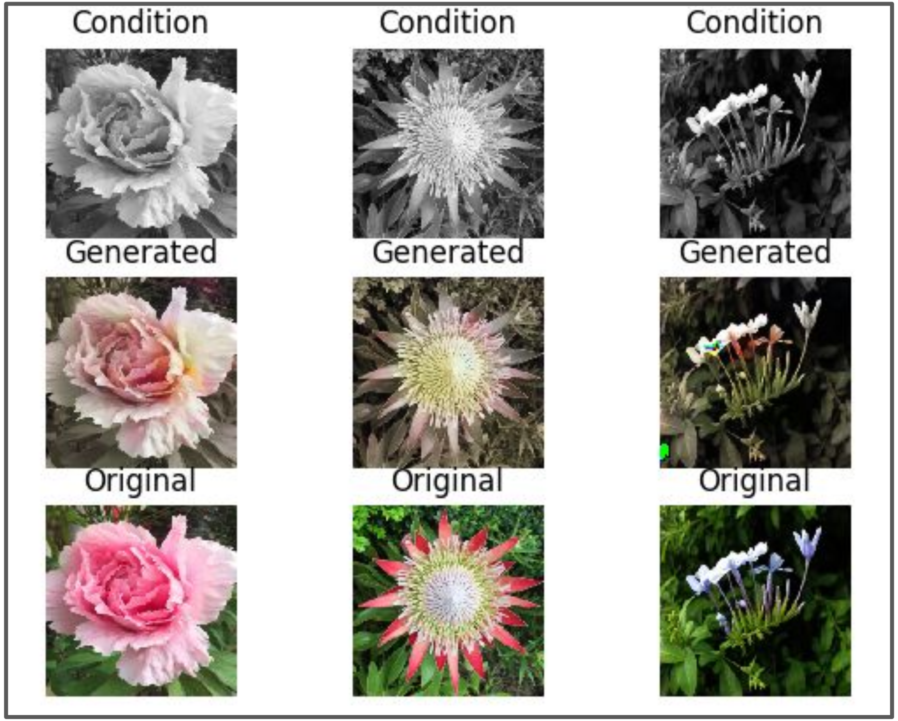
Conditional GAN (pix2pix)

Training a conditional GAN



Combined Loss Function

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$



References

- Cycle GAN paper by Zhu et al - https://arxiv.org/pdf/1703.10593.pdf
- Blog Cycle GAN - https://medium.com/@jonathan_hui/gan-cyclegan-6a50e7600d7
- Cycle GAN implementation - https://github.com/eriklindernoren/Keras-GAN
- Keras documentation - https://keras.io/
- CapsuleGAN implementation - https://github.com/gusgad/capsule-GAN/blob/master/capsule_gan.ipynb
- CapsGAN - https://arxiv.org/abs/1806.03968
- CapsuleGAN - https://arxiv.org/abs/1802.06167
- Capsule Networks - https://arxiv.org/abs/1710.09829
- Blog Capsule Networks - https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition-b4b559d1159b
- Conditional Instance Normalization - https://arxiv.org/pdf/1703.06868.pdf
- Pix2pix implemntation - https://github.com/eriklindernoren/Keras-GAN/tree/master/pix2pix
- Pix2pix - https://arxiv.org/abs/1611.07004