Image2Image Translation

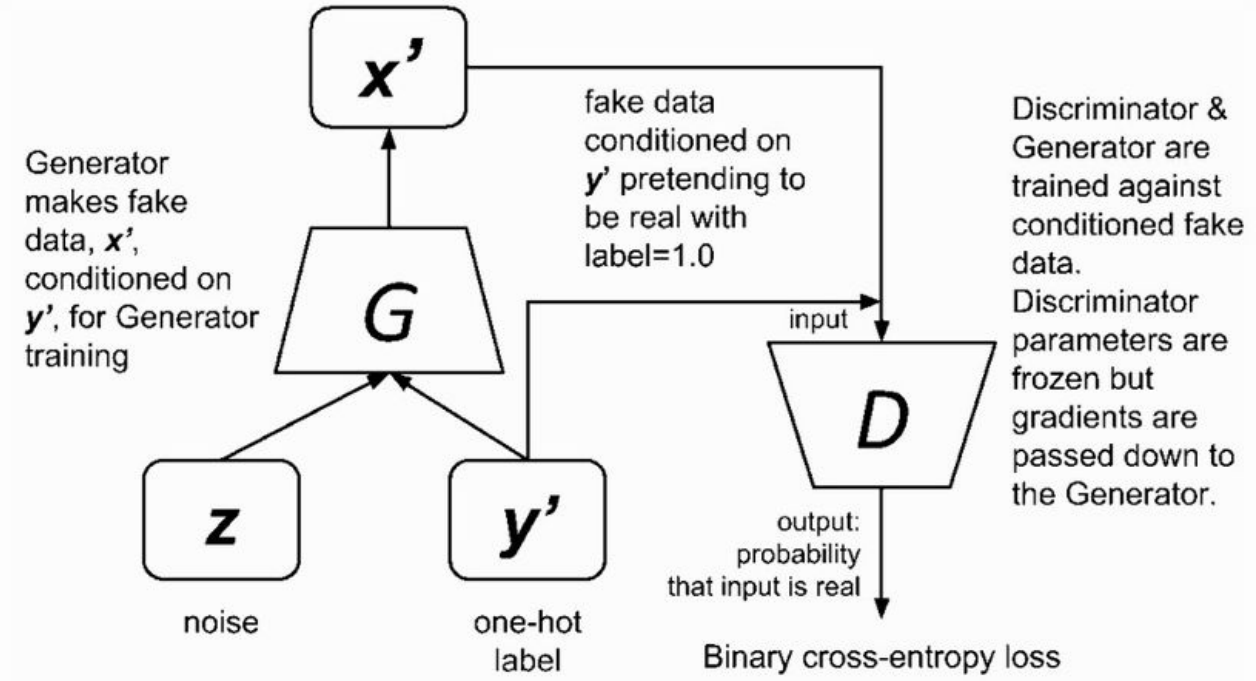
Alireza Javaheri
Arsham Golamzadeh Khoee

Conditional GAN (CGAN)

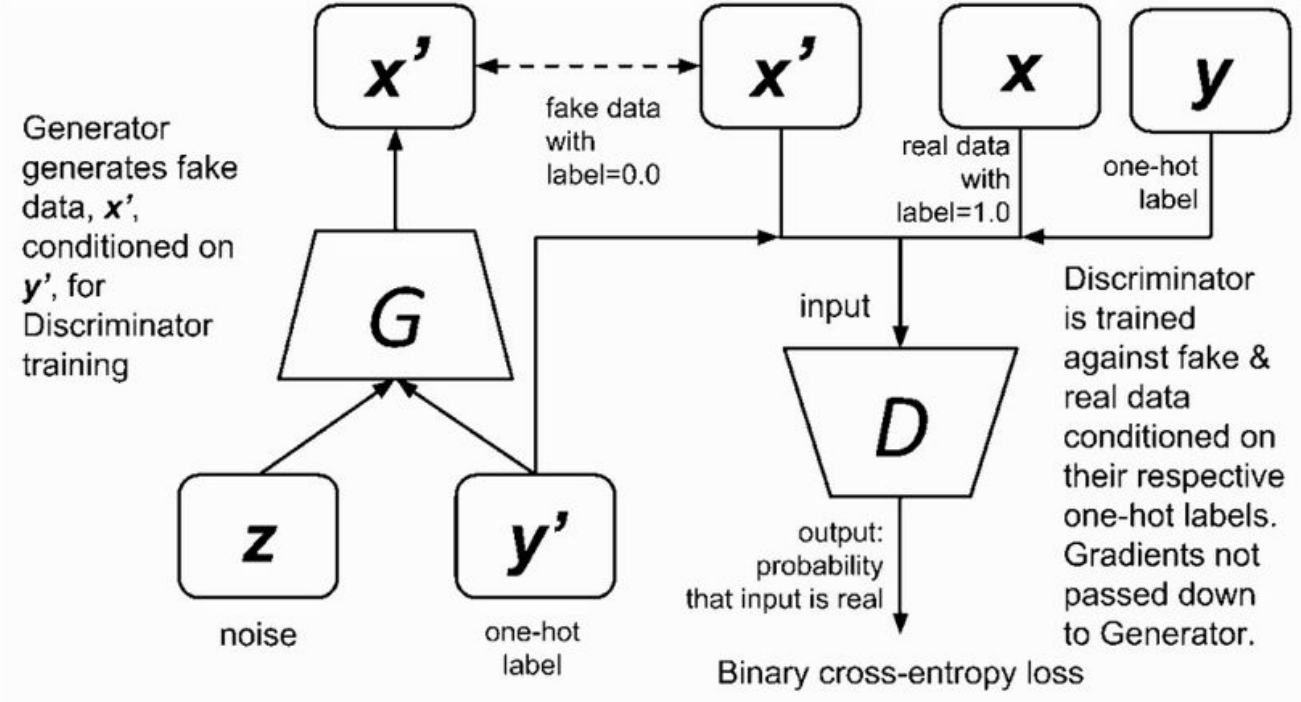
Conditional GAN (CGAN)

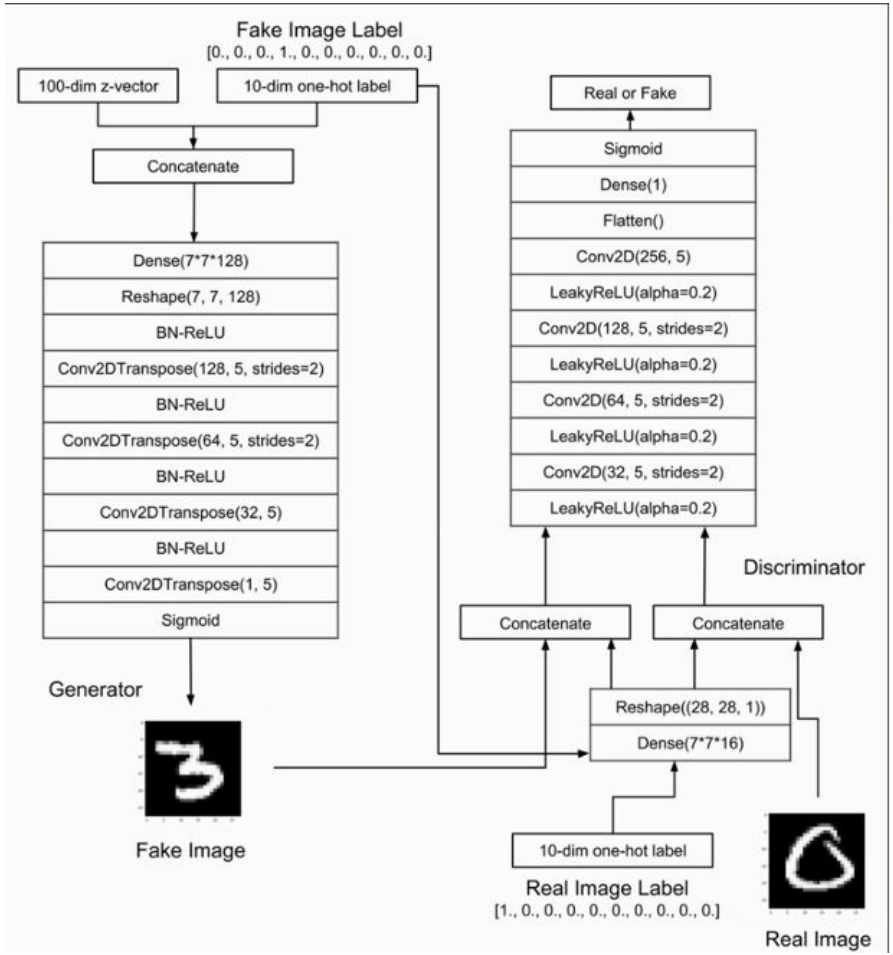
- The Generator and Discriminator both receive some additional conditioning input information. This could be the class of current image or some other property
- Fore example, if we train a DCGANs to generate new MNIST images, There is no control over which specific digits will be produced by the Generator.
- There is no mechanism for how to request a particular digit from the Generator.

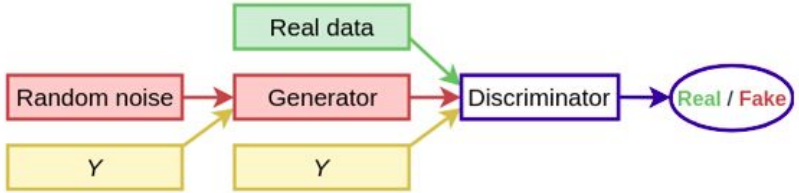
Generator's Training Flow



Discriminator's Training Flow







$$\mathcal{L}^{(D)}\left(\theta^{(G)},\theta^{(D)}\right) = -\mathbb{E}_{x\sim p_{data}} \log \mathcal{D}\left(x \mid y\right) - \mathbb{E}_{z} \log\left(1-\mathcal{D}\left(\mathcal{G}\left(z \mid y'\right)\right)\right)$$

CGAN problems

- Conditional GANs have enabled a variety of applications, but the results are often limited to loss resolution and still far from realistic

Image2Image translate

- learn the mapping between an input image and an output image.
- style transfer, object transfiguration, colorization and super-resolution.

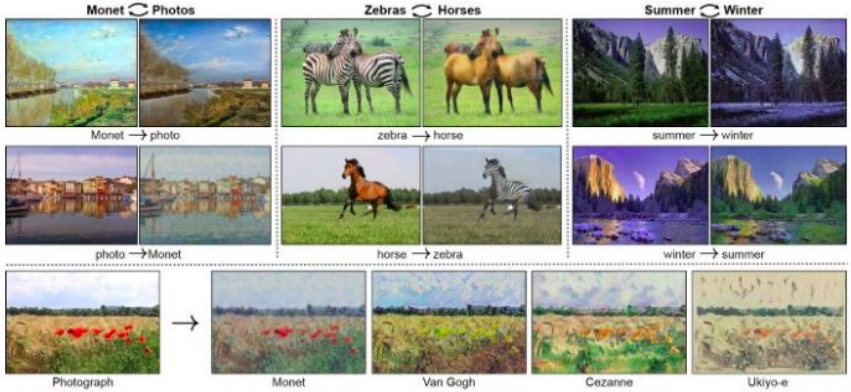


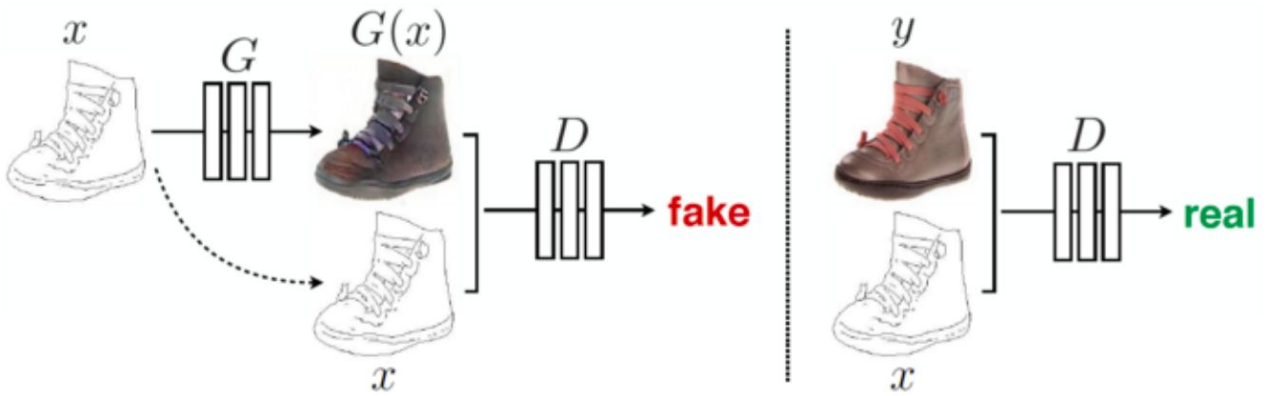
Image2Image translate networks

- Pix2pix GAN
- CycleGAN
- DiscoGAN
- PAN(Perceptual Adversarial Network)
- StartGan

Pix2Pix GAN

Pix2Pix

- Basically a Conditional GAN (cGAN) that learn the mapping from an input image to output an image.

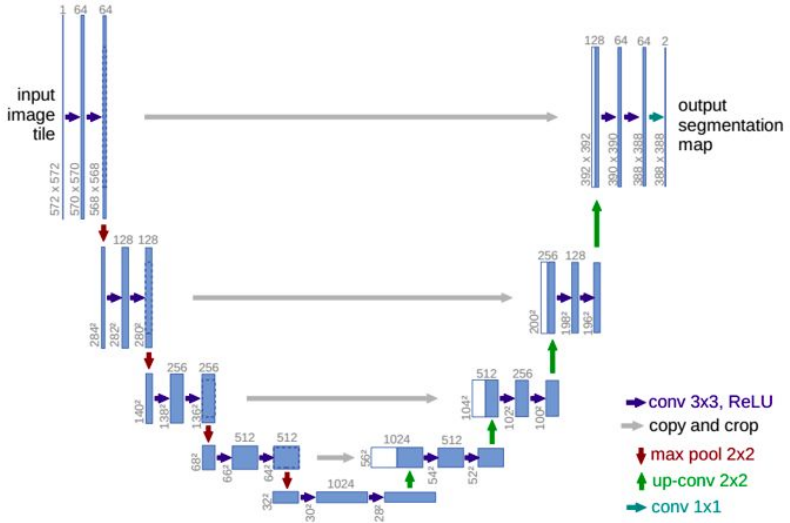


The Generator's Network

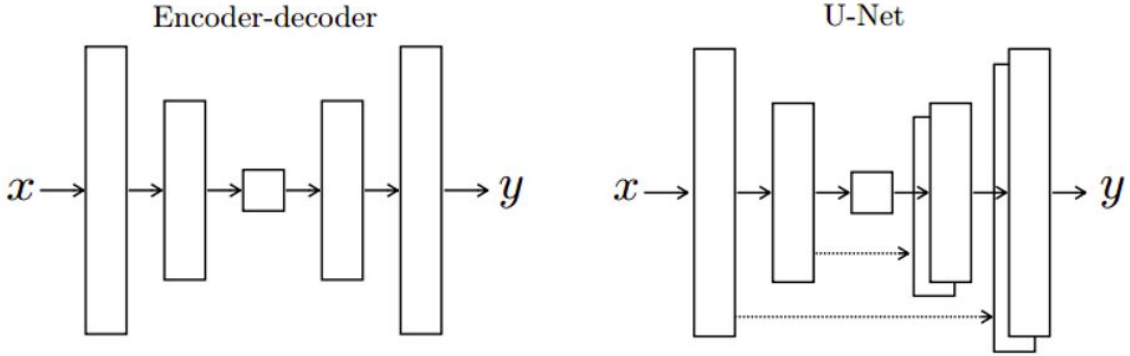
- **Generator** network uses a **U-Net**-based architecture.
- **U-Net's** architecture is similar to an **Auto-Encoder** network except for one difference
- Both **U-Net** and **Auto-Encoder** network has two networks The **Encoder** and the **Decoder.**

U-Net's Architecture Diagram

- U-Net's network has skip connections between **Encoder** layers and **Decoder** layers
- First layer of Encoder is **directly passed** to the last layer of the Decoder
- Second layer of **Encoder** is pass to the second last layer of **Decoder** and so on ...



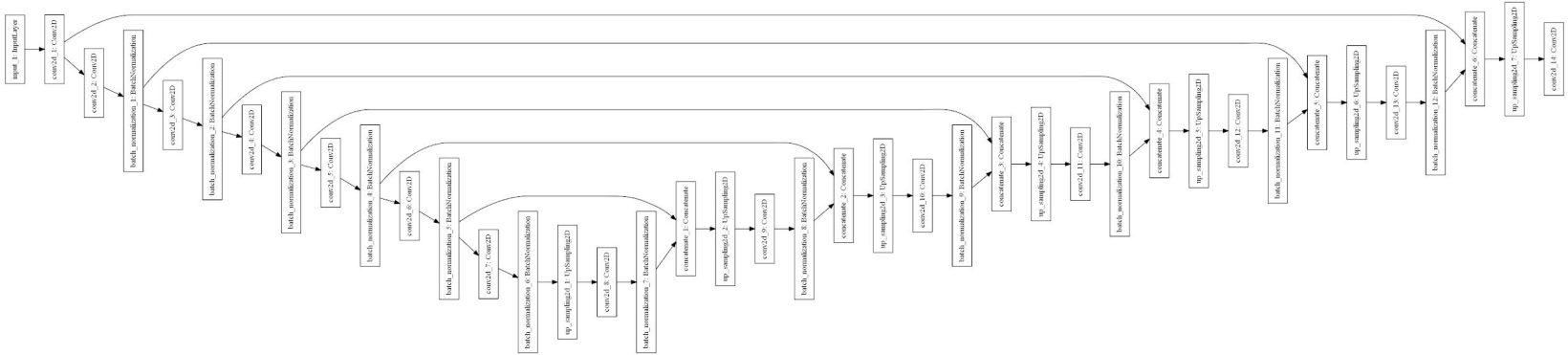
U-Net's Architecture Diagram



- To give the generator a means to circumvent the bottleneck for information like this , we add skip connections, following the general shape of a **U-Net**

The Generator's Architecture

- The Generator network is made up of these two networks
- The Encoder network is a downsampler
- The Decoder network is an upsampler



The Generator's Encoder Architecture

- The **Encoder** network of the **Generator** network has seven convolutional blocks.
- Each convolutional block has a convolutional layer, followed a **LeakyRelu** activation function.
- Each convolutional block also has a batch normalization layer except the first convolutional layer.

The Generator's Decoder Architecture

- The **Decoder** network of the **Generator** network has seven upsampling convolutional blocks.
- Each upsampling convolutional block has an upsampling layer, followed by a convolutional layer, a batch normalization layer and a **ReLU** activation function.

Pix2Pix Network's Training

$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x,z))],$$

$$\mathcal{L}_{GAN}(G,D) = \mathbb{E}_{y}[\log D(y)] + \mathbb{E}_{x,z}[\log(1 - D(G(x,z))].$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1].$$

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

Previous approaches have found it beneficial to mix the GAN objective with a more traditional loss, such as L2 distance. The discriminator's job remains unchanged, but the generator is tasked to not only fool the discriminator but also to be near the ground truth output in an L2 sense. We also explore this option, using L1 distance rather than L2 as L1 encourages less blurring:

$$L1LossFunction = \sum_{i=1}^{n} |y_{true} - y_{predicted}|$$

$$L2LossFunction = \sum_{i=1}^{n} (y_{true} - y_{predicted})^2$$

It is well known that the L2 loss – and L1, see Figure 4 – produces blurry results on image generation problems Although these losses fail to encourage high-frequency crispness, in many cases they nonetheless accurately capture the low frequencies. For problems where this is the case, we do not need an entirely new framework to enforce correctness at the low-frequencies. L1 will already do