Introduction to Generative Adversarial Networks (GAN)

IA716 - Perception pour les systemes autonomes

Gianni Franchi

24/04/2023

Approximate Bayesian Computation

Let us consider that we have a set $\{X_i\}$ of data that follow a distribution $P(X)$. Our goal is to generate new data from this distribution but.
Let us assume we have access to a model of distribution $P(X/\theta)$ then we can generate new data X. This distribution is called the likelihood.
However, for particular problems, we may find that **we can not express the likelihood** in closed-form, or it is prohibitively costly to compute it.

Approximate Bayesian Computation

A solution is to approximate the likelihood.
We aim at using a function $\delta(\cdot)$ to obtain a practically good enough approximation to the true likelihood:

$$\lim_{\epsilon \to 0} \delta(X, \hat{X}, \epsilon) = P(X/\theta)$$

We introduce a tolerance parameter ϵ because the chance of generating a synthetic data-set $\hat{X}$ being equal to the observed data X is virtually null for most problems

GAN principle: Why Generative learning

We've only seen discriminative models in the past

- Given an image X, predict a label Y;
- Estimate $P(Y|X)$.

Discriminative models have several key limitations

- Can't model $P(X)$, i.e. the probability of seeing a certain image;
- Thus, can't sample from $P(X)$, i.e. can't generate new images;
- Fixed loss.

Generative models (in general) cope with all of above

- Can model $P(X)$
- Can generate new images.
- Learned loss link with perception (perceptual loss)

GAN principle

In generative adversarial networks (GANs), the task of learning a generative model is expressed as a two-player zero-sum game between two networks.

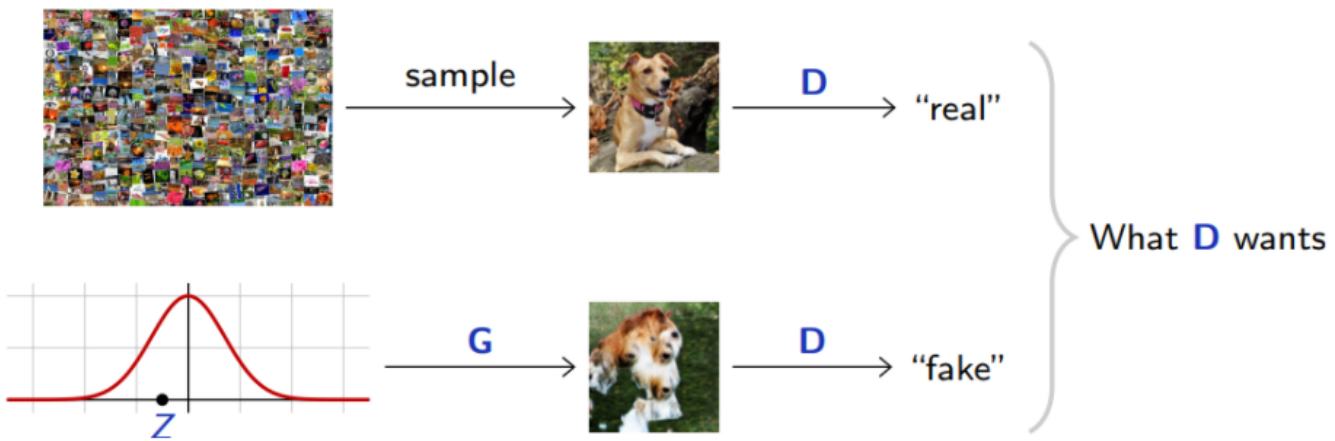


Figure: Principle of the GAN **[Goodfellow2014]**

With a random noise.

GAN principle[1]

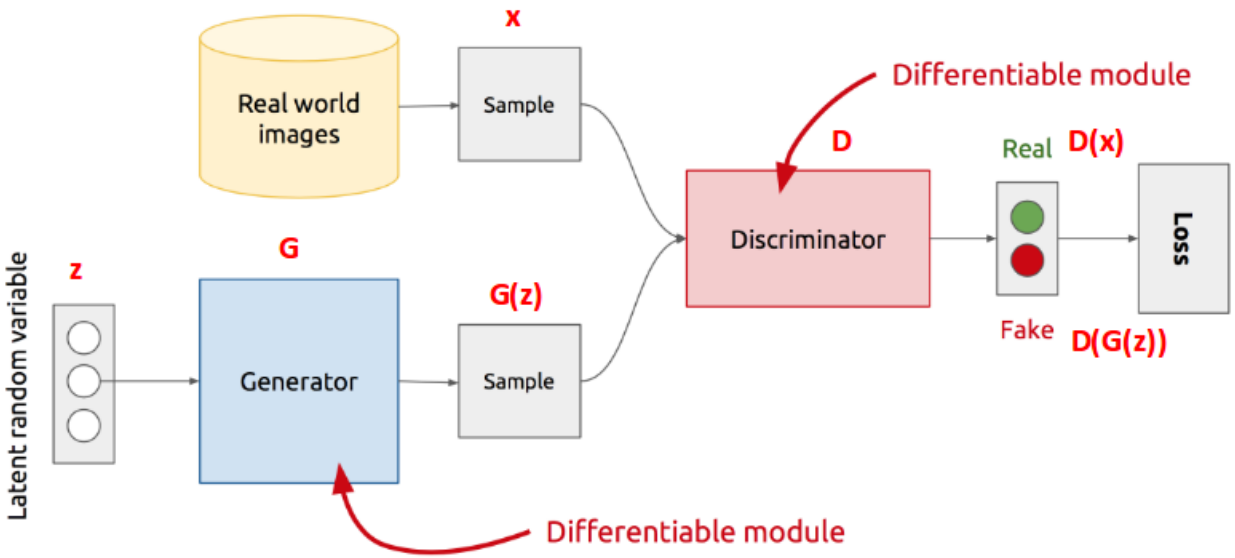


Figure: Principle of the GAN **[Goodfellow2014]**

We need two DNNs.

[1] Gilles Louppe

GAN principle : training [Goodfellow2014]

The first network is called a generator $g(\cdot;\theta) : \mathcal{Z} \to \mathcal{X}$, mapping a latent space equipped with a prior distribution $p(z)$ to the data space, thereby inducing a distribution

$$x \sim q(x;\theta) \Leftrightarrow z \sim p(z), x = g(z;\theta)$$

The second network $d(\cdot;\phi) : \mathcal{X} \to [0,1]$ is a classifier called discriminator trained to distinguish between true samples $x \sim p(x)$ and generated samples $x \sim q(x;\theta)$.

GAN principle : training [Goodfellow2014]

For a fixed generator g, the discriminator d can be trained by generating a two-class training set
$\mathrm{d} = \{(\mathrm{x}_1, y = 1), ..., (\mathrm{x}_N, y = 1), (g(\mathrm{z}_1; \theta), y = 0), ..., (g(\mathrm{z}_N; \theta), y = 0)\}$,
and minimizing the cross-entropy loss

$$\begin{aligned}\mathcal{L}(\phi) &= -\frac{1}{2N}\sum_{i=1}^{N}\left[\log d(\mathrm{x}_i;\phi) + \log\left(1 - d(g(\mathrm{z}_i;\theta);\phi)\right)\right] \\ &\approx -\mathbb{E}_{\mathrm{x}\sim p(\mathrm{x})}\left[\log d(\mathrm{x};\phi)\right] - \mathbb{E}_{\mathrm{z}\sim p(\mathrm{z})}\left[\log(1 - d(g(\mathrm{z};\theta);\phi))\right].\end{aligned}$$

However, the situation is slightly more complicated since we also want to train g to fool the discriminator, which is equivalent to maximize d's loss.

GAN principle : training [Goodfellow2014]

Let us consider the value function

$$V(\phi, \theta) = \mathbb{E}_{\mathsf{x} \sim p(\mathsf{x})} \left[\log d(\mathsf{x}; \phi)\right] + \mathbb{E}_{\mathsf{z} \sim p(\mathsf{z})} \left[\log(1 - d(g(\mathsf{z}; \theta); \phi))\right].$$

For a fixed g, $V(\phi, \theta)$ is high if d is good at recognizing true from generated samples.
d is the best classifier given g, and if V is high, then this implies that the generator is bad at reproducing the data distribution.
Conversely, g will be a good generative model if V is low when d is a perfect opponent.

GAN principle : training [Goodfellow2014]

Therefore, the ultimate goal is optmized this minimax loss:

$$\theta^* = \arg\min_{\theta} \max_{\phi} V(\phi, \theta).$$

GAN principle : training [Goodfellow2014]

Here I change $d(x;\phi)$ by $D(x)$ and also $g(x;\theta)$ by $G(x)$

$$V(\phi,\theta) = \mathbb{E}_{\mathrm{x}\sim p(\mathrm{x})}\left[\log d(\mathrm{x};\phi)\right] + \mathbb{E}_{\mathrm{z}\sim p(\mathrm{z})}\left[\log(1 - d(g(\mathrm{z};\theta);\phi))\right]$$

$$V(G,D) = \mathbb{E}_{\mathrm{x}\sim p(\mathrm{x})}\left[\log D(\mathrm{x})\right] + \mathbb{E}_{\mathrm{z}\sim p(\mathrm{z})}\left[\log(1 - D(G(\mathrm{z})))\right]$$

Remember : If we have

- I a real interval;
- $\varphi : [a,b] \to I$ a derivable function and whose derivative has an integral;
- I a real interval;
- $f : I \to \mathrm{R}$ a continuous function.

then :

$$\int_a^b f(\varphi(t))\varphi'(t)\,\mathrm{d}t = \int_{\varphi(a)}^{\varphi(b)} f(x)\,\mathrm{d}x.$$

GAN principle : training [Goodfellow2014]

So

$$\mathbb{E}_{z\sim p(z)}\left[\log(1-D(G(z)))\right] = \mathbb{E}_{x\sim p_g(x)}\left[\log(1-D(x))\right]$$

So we have

$$V(G,D) = \int_x (p(x)\log(D(x)) + p_g(x)\log(1-D(x)))dx$$

Let us fix the generator and look for the best discriminator D^*

$$\frac{\partial V(G,D)}{\partial D} = \frac{\partial}{\partial D}\int_x (p(x)\log(D(x)) + p_g(x)\log(1-D(x)))dx$$

GAN principle : training [Goodfellow2014]

We want to find

$$\frac{\partial}{\partial D} p \log(D) + p_g \log(1 - D) = 0$$

$$\frac{p}{D} - \frac{p_g}{(1 - D)} = 0$$

$$\frac{D}{(1 - D)} = \frac{p_g}{p}$$

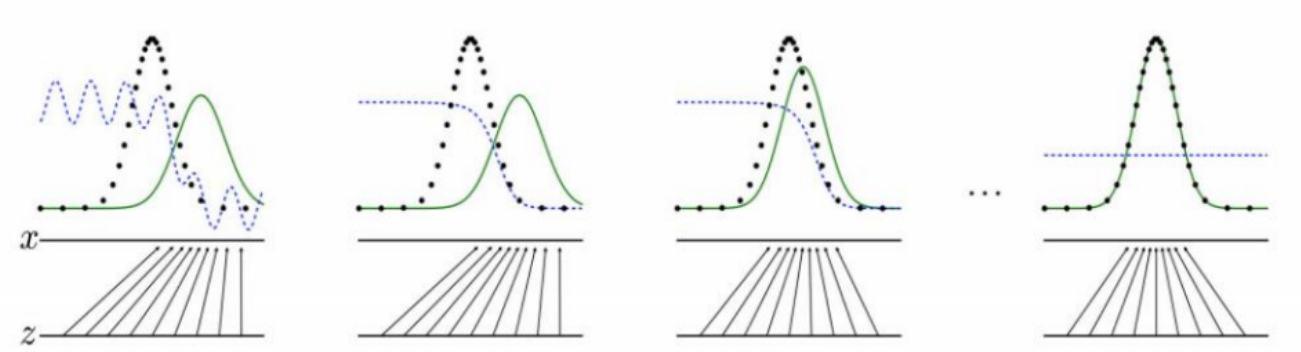
$$D = \frac{p_g}{p + p_g}$$

When the generator is perfectly driven $p = p_g$ then $D = 1/2$

GAN principle : training [Goodfellow2014]

Optimal solution to the adversarial game:

- Generator distribution = data distribution
- D = 1/2



GAN results

Results

a) b) c) d)

GAN issues:

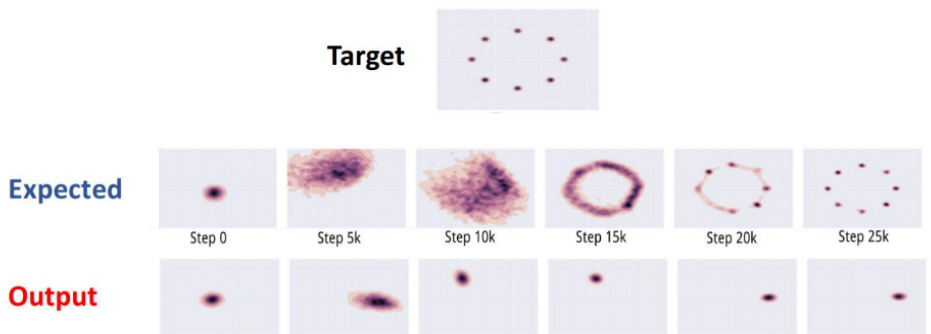
Disadvantages of Generative Adversarial Networks (GANs):

- **Training Instability:** GANs can be difficult to train, with the risk of instability, **mode collapse**, or **failure to converge**.
- **Computational Cost:** GANs can require a lot of computational resources and can be slow to train, especially for high-resolution images or large datasets.
- **Overfitting:** GANs can overfit the training data, producing synthetic data that is too similar to the training data and lacking diversity.
- **Bias and Fairness:** GANs can reflect the biases and unfairness present in the training data, leading to discriminatory or biased synthetic data.
- **Interpretability and Accountability:** GANs can be opaque and difficult to interpret or explain, making it challenging to ensure accountability, transparency, or fairness in their applications.

GAN issues: Mode collapse [Srivastava2017]

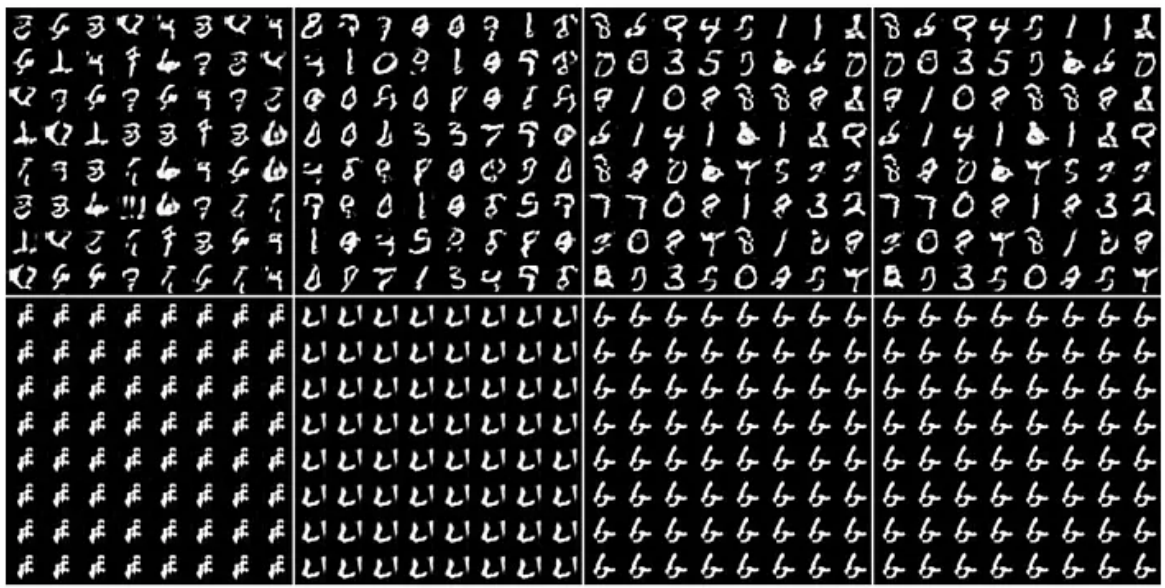
Training a standard GAN often results in pathological behaviors:

- Oscillations without convergence: contrary to standard loss minimization, alternating stochastic gradient descent has no guarantee of convergence.
- Vanishing gradients: when the classifier **d** is too good, the value function saturates and we end up with no gradient to update the generator.
- Mode collapse: the generator **g** models very well a small sub-population, concentrating on a few modes of the data distribution.
- Performance is also difficult to assess in practice.



GAN issues: Mode collapse **[jonathan-hui]**

Real-life data distributions are multimodal. For example, in MNIST, there are 10 major modes from digit '0' to digit '9'. The samples below are generated by two different GANs. The top row produces all 10 modes while the second row creates a single mode only (the digit '6'). This problem is called mode collapse when only a few modes of data are generated.



GAN issues: Mode collapse **[jonathan-hui]**

The objective of the GAN generator is to create images that can fool the discriminator D the most.

GAN issues: Mode collapse **[jonathan-hui]**

But let's consider one extreme case where G is trained extensively without updates to D. The generated images will converge to find the optimal image x^* that fool D the most, the most realistic image from the discriminator perspective. In this extreme, x^* will be independent of z.

GAN issues: Mode collapse **[jonathan-hui]**

When we restart the training in the discriminator, the most effective way to detect generated images is to detect this single mode. Since the generator desensitizes the impact of z already, the gradient from the discriminator will likely push the single point around for the next most vulnerable mode. This is not hard to find. The generator produces such an imbalance of modes in training that it deteriorates its capability to detect others. Now, both networks are overfitted to exploit short-term opponent weakness.

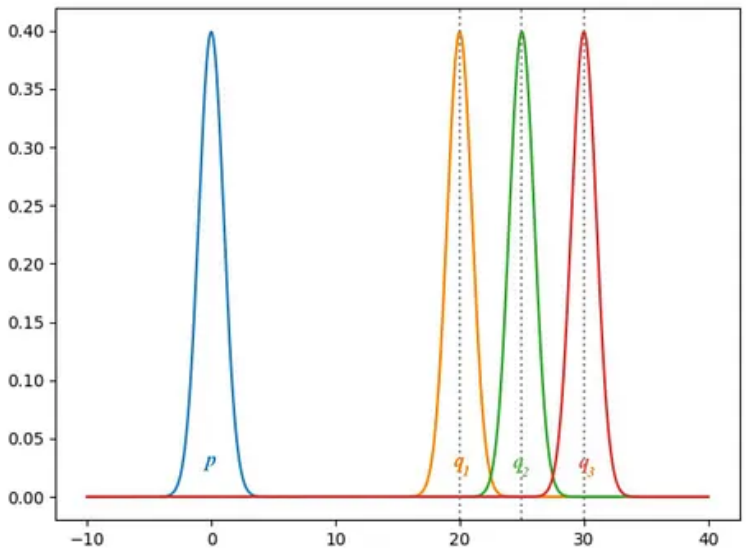
GAN issues: Vanishing Gradient **[jonathan-hui]**

Recall that when the discriminator is optimal, the objective function for the generator is:

$$V(G, D^*) = 2D_{JS}\left[p||p_g\right] + cst$$

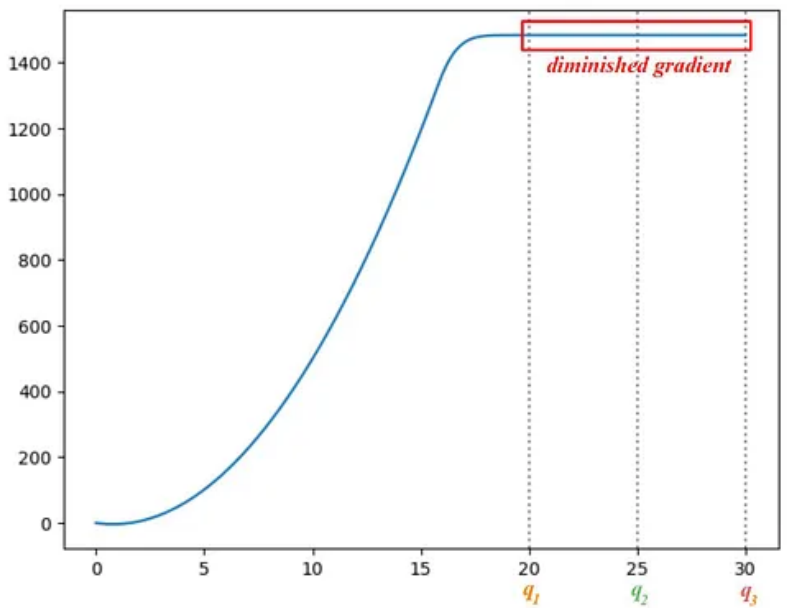
GAN issues: Vanishing Gradient **[jonathan-hui]**

Let's consider an example in which p and p_g are Gaussian distributed and the mean of p is zero. Let's consider p_g with different means to study the gradient of $D_{JS}[p||p_g]$. We denote these distribution q_1, q_2, q_3



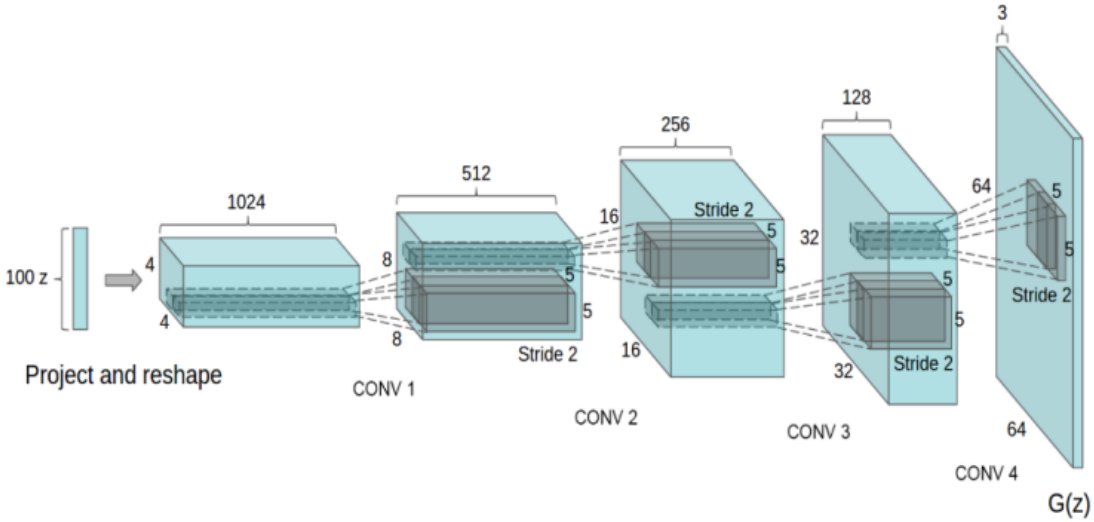
GAN issues: Vanishing Gradient **[jonathan-hui]**

As shown below, the gradient for the JS-divergence vanishes from q_1 to q_3. The GAN generator will learn extremely slow to nothing when the cost is saturated in those regions.



DCGAN [Radford2015]

Replace FC hidden layers with Convolutions/deconvolutional layers.



DCGAN [Radford2015]

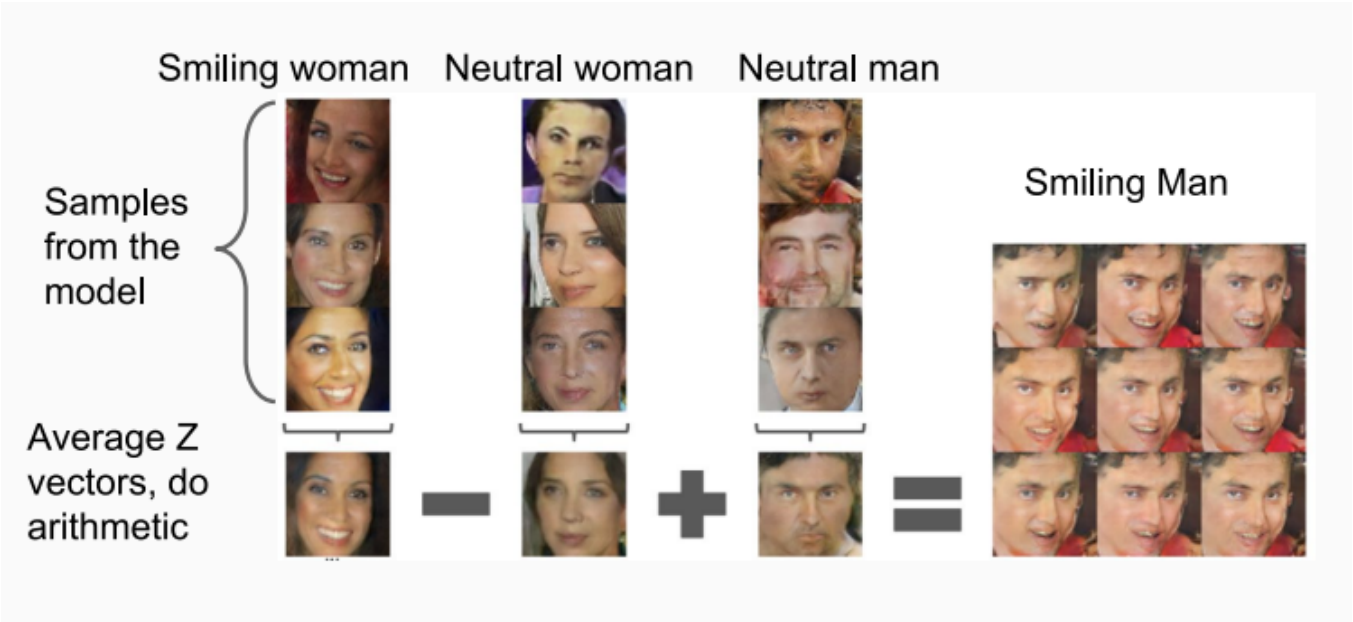
Generations of realistic bedrooms pictures, from randomly generated latent variables

DCGAN [Radford2015]

Interpolation in between points in latent space.

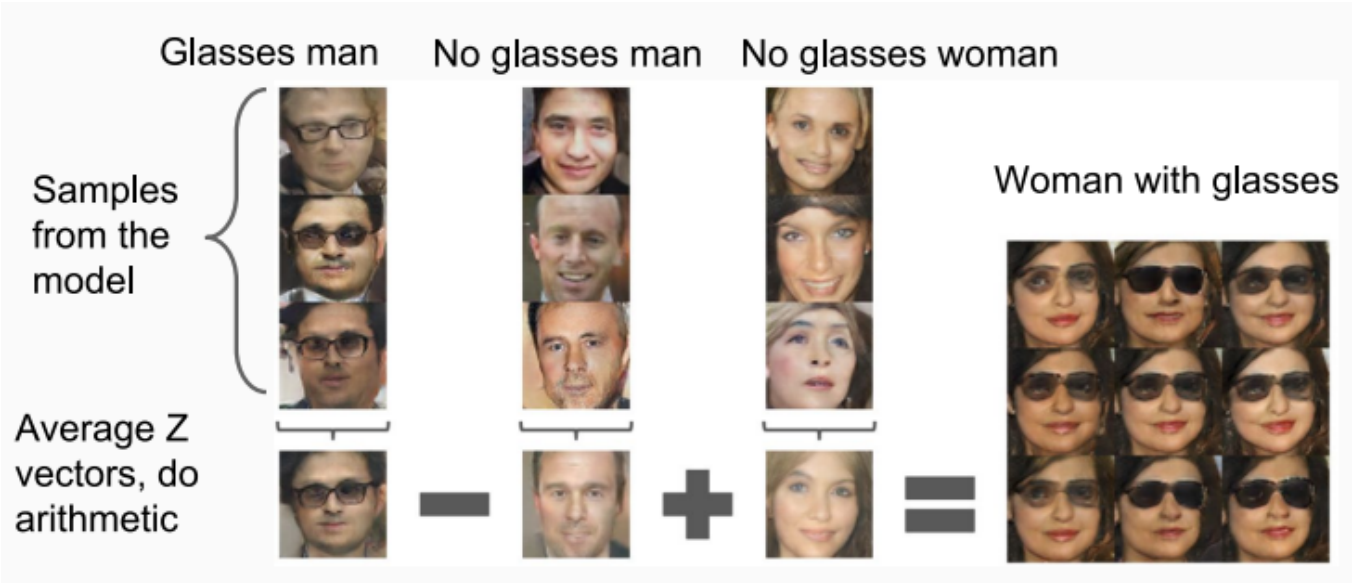
DCGAN [Radford2015]

Convolutional GAN - Arithmetic



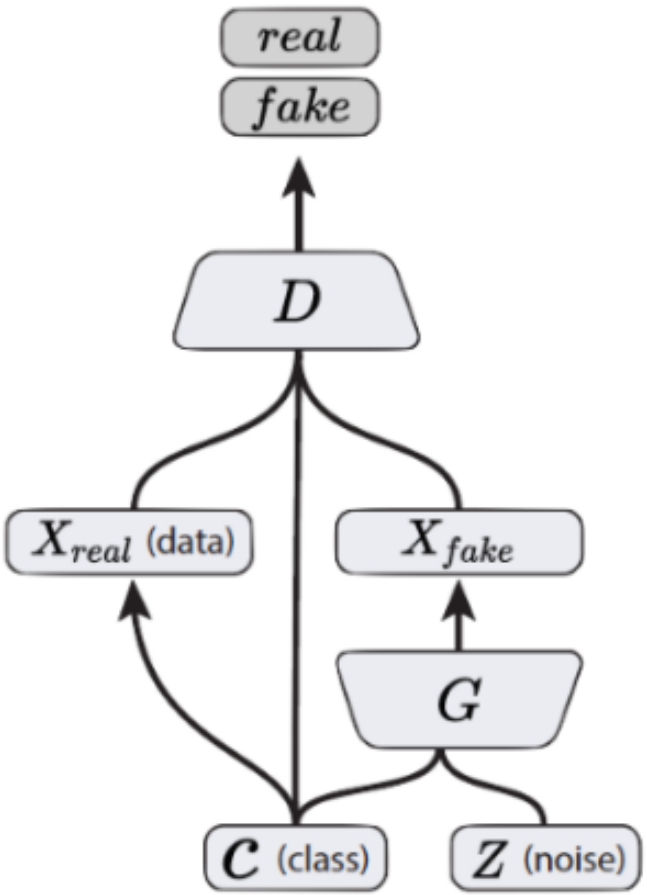
DCGAN [Radford2015]

Convolutional GAN - Arithmetic



Conditional GAN [Mirza2014]

Conditional generative adversarial network, or cGAN, is a type of GAN that involves the conditional generation of images by a generator model. Hence you can control the kind of output you want



Optimal transport [alexhwilliams]

A fundamental problem in statistics and machine learning is to come up with useful measures of 'distance' between pairs of probability distributions. One can compute the Kullback-Lieibler (KL) divergence from Q to P is defined by :

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

While the KL divergence is incredibly useful and fundamental in information theory, it also has its shortcomings.
For instance, one of the first things we learn about the KL divergence is that it is not symmetric A bigger problem is that the divergence may be infinite if the support of P and Q are not equal.

Optimal transport [alexhwilliams]

One of the nice aspects of optimal transport theory is that it can be grounded in physical intuition through the following thought experiment. Suppose we are given the task of filling several holes in the ground. The image below shows an overhead 2D view of this scenario - the three **red regions** correspond to dirt piles, and the **eight blue regions** correspond to holes.

Optimal transport [alexhwilliams]

Our goal is to come up with the most efficient transportation plan to which moves the dirt to fill all the holes. We assume **the total volume of the holes** is equal to **the total volume of the dirt piles**.

Optimal transport [alexhwilliams]

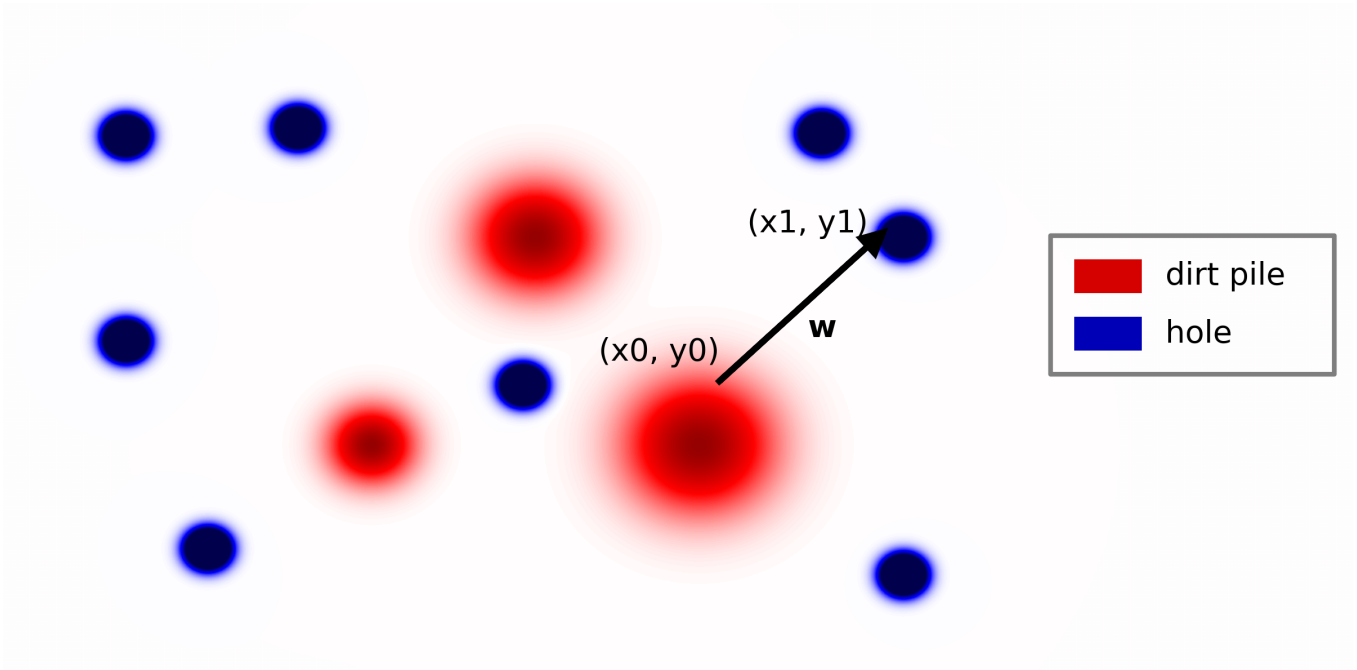
The "most efficient" plan is the one that minimizes the total transportation cost. To quantify this, let's say the transportation cost C of moving 1 unit of dirt from $(x_0, y_0) \to (x_1, y_1)$ is given by the squared Euclidean distance:

$$C(x_0, y_0, x_1, y_1) = (x_0 - x_1)^2 + (y_0 - y_1)^2$$

Now we'll define the transportation plan T, which tells us how many units of dirt to move from $(x_0, y_0) \to (x_1, y_1)$ which is given by

$$T(x_0, y_0, x_1, y_1) = \omega$$

Optimal transport [alexhwilliams]



Optimal transport [alexhwilliams]

The transportation plan, T, specifies an arrow like this from every possible starting position to every possible destination. Further, in addition to being non negative, the plan must satisfy the following two conditions:

$$\int \int T(x_0, y_0, x, y) dx dy = p(x_0, y_0) \ \forall x_0, y_0$$

$$\int \int T(x, y, x_1, y_1) dx dy = q(x_1, y_1) \ \forall x_1, y_1$$

Where $p(\cdot, \cdot)$ and $q(\cdot, \cdot)$ are density functions encoding the units of dirt and hole depth at each 2D location. Intuitively, the first constraint says that the amount of piled dirt at is "used up" or transported somewhere. The second constraint says that the hole at is "filled up" with the required amount of dirt (no more and no less).

Optimal transport [alexhwilliams]

Suppose we are given a function T that satisfies all of these conditions (i.e. we are given a feasible transport plan). Then the overall transport cost is given by:

$$\text{total cost} = \int\int\int\int C(x_0, y_0, x_1, y_1)\, T(x_0, y_0, x_1, y_1)\, dx_0\, dy_0\, dx_1\, dy_1$$

Xe multiply the amount of dirt transported, given by T, by the per unit transport cost, given by C. Integrating over all possible origins and destinations gives us the total cost.

Optimal transport [alexhwilliams]

Given two probability distributions μ_0 and μ_1, and a positive cost function $c : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^+$ The Wasserstein distances is given by

$$OP(\mu_0, \mu_1) = \inf_{\gamma \in \Pi(\mu_0, \mu_1)} \int c(x, y) \, \mathrm{d}\gamma(x, y)$$

where $\Pi(\mu_0, \mu_1)$ is the set of probability distributions γ with marginal distributions μ_0 and μ_1.

Optimal transport [Arjovsky2017]

When using $c(x, y) = \|x - y\|^p$ one defines Wasserstein distances.
The p-Wasserstein distance W^p between μ_0 and $\mu_1$is defined as

$$W^p(\mu_0, \mu_1) = \inf_{\gamma \in \Pi(\mu_0, \mu_1)} \int \|x - y\|^p \, \mathrm{d}\gamma(x, y)$$

Which is similar to solve the dual following dual problem for $p = 1$

$$W^p(\mu_0, \mu_1) = \sup_{\phi \in \mathrm{Lip}_1} [E_{x \sim \mu_0}(\phi(x)) - E_{x \sim \mu_0}(\phi(x))]$$

with

$$\mathrm{Lip}_1 = \{f : \mathbb{R}^D \to \mathbb{R} \text{ such that } \forall (x, y) \|f(x) - f(y)\| \leq \|x - y\|\}$$

Wassestein GAN **[Arjovsky2017]**

GAN (Vanilla):

$$\min_{D_\theta} \max_{G_\phi} \mathbb{E}_{\mathrm{x}\sim p(\mathrm{x})} [\log D(\mathrm{x})] + \mathbb{E}_{\mathrm{z}\sim p(\mathrm{z})} [\log(1 - D(G(\mathrm{z})))]$$

Wassestein GAN :

$$\min_{D_\theta} \max_{G_\phi} \mathbb{E}_{\mathrm{x}\sim p(\mathrm{x})} [D(\mathrm{x})] - \mathbb{E}_{\mathrm{z}\sim p(\mathrm{z})} [(D(G(\mathrm{z})))]$$

We just got rid of the log and D is not a probability... but we now have a constrained optimization $D \in \mathrm{Lip}_1$ The original WGAN paper uses weight clipping to restrict the Lipschitz constant (heuristic)

Cycle GAN [geeksforgeeks]

We use two Conditional GAN:

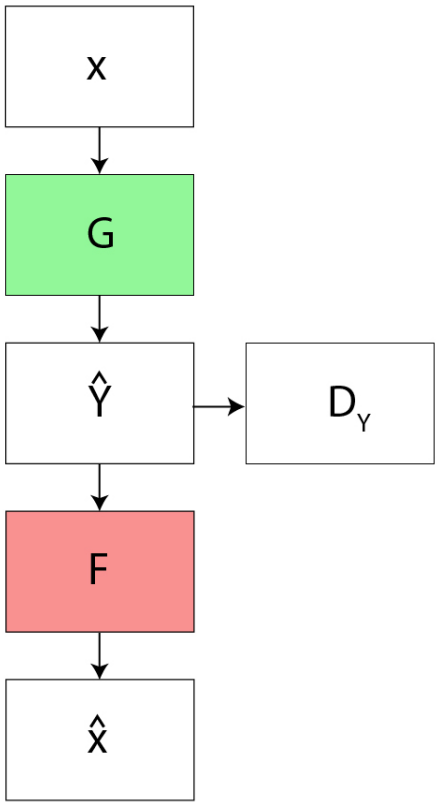
$$G_{Y \to X} : Y \to X$$

$$G_{X \to Y} : X \to Y$$

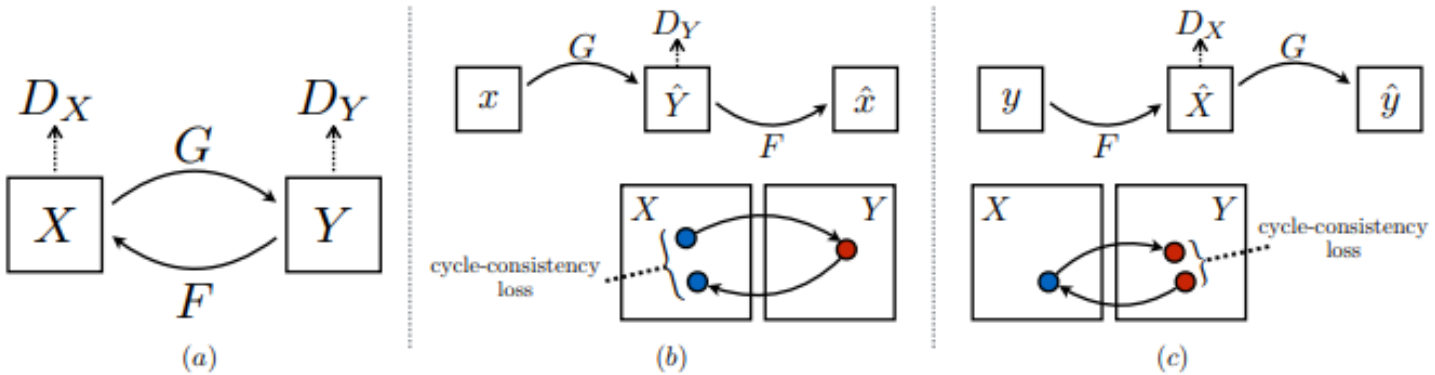
Cycle GAN [geeksforgeeks]

More specifically the CycleGAN architecture is different from other GANs in a way that it contains 2 mapping function (G or $G_{Y \to X}$ and F or $G_{X \to Y}$) that acts as generators and their corresponding Discriminators (D_x and D_y): The generator mapping functions are as follows:

Cycle GAN [geeksforgeeks]



Cycle GAN [geeksforgeeks]



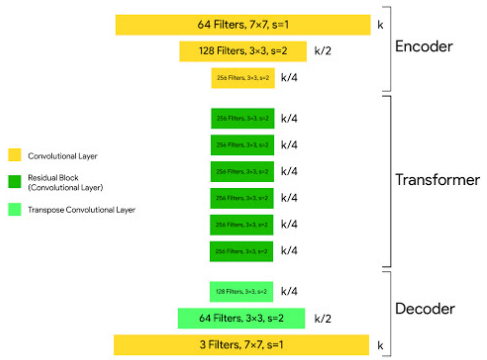
Cycle GAN [geeksforgeeks]

Each CycleGAN generator has three sections:

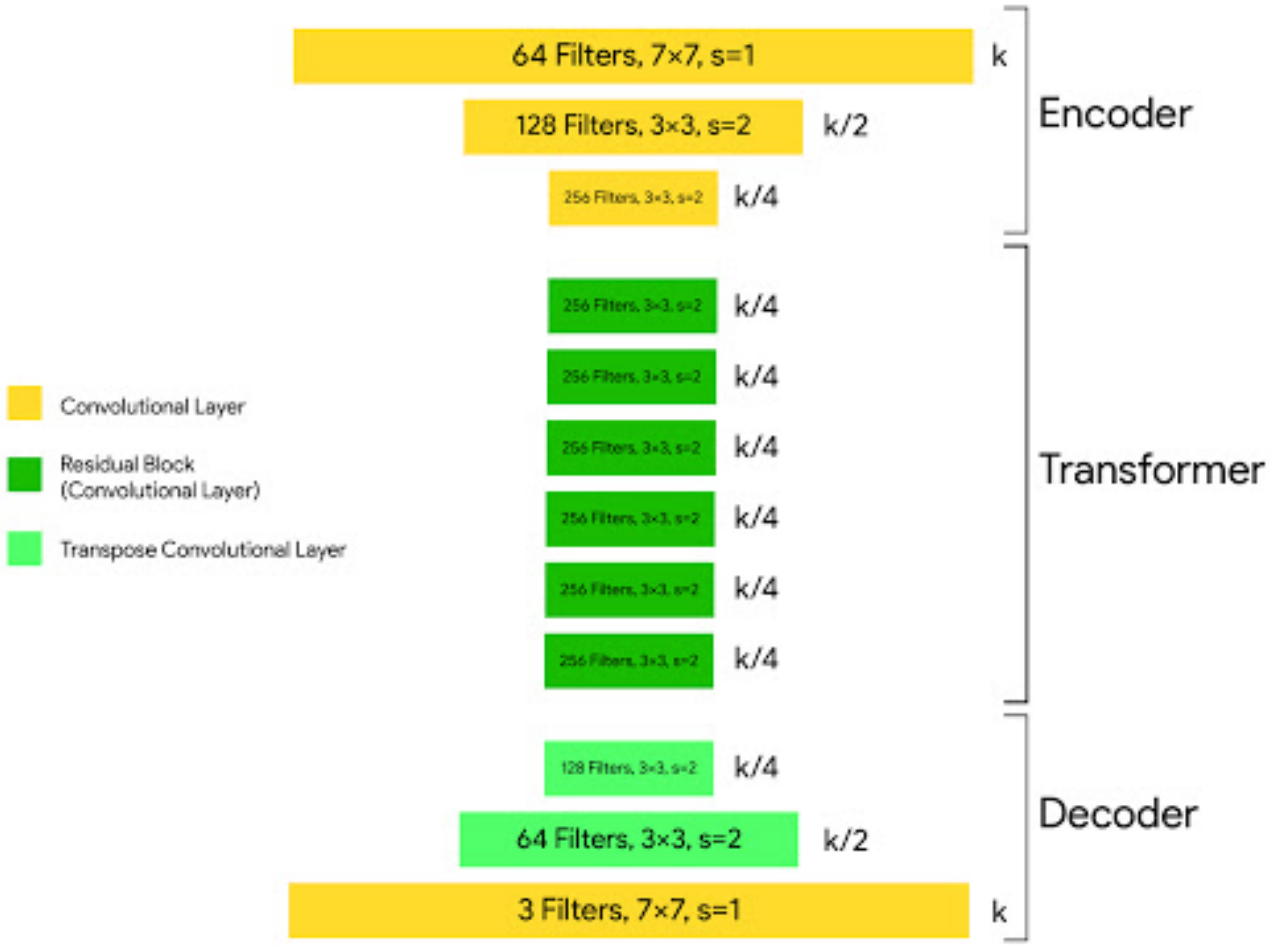
- Encoder
- Transformer
- Decoder

Cycle GAN [geeksforgeeks]

The input image is passed into the **encoder**. The encoder extracts features from the input image by using Convolutions and compressed the representation of image but increase the number of channels.
Then the output of encoder after activation function is applied is passed into the **transformer**. The transformer contains 6 or 9 residual blocks based on the size of input.
The output of transformer is then passed into the **decoder** which uses 2 -deconvolution block of fraction strides to increase the size of representation to original size.

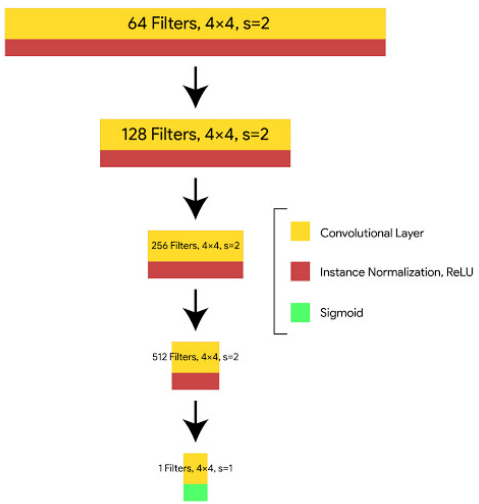


Cycle GAN [geeksforgeeks]



Cycle GAN [geeksforgeeks]

In discriminator the authors use **PatchGAN discriminator**. The difference between a PatchGAN and regular GAN discriminator is that rather the regular GAN maps from a 256×256 image to a single scalar output, which signifies 'real' or 'fake', whereas the PatchGAN maps from $256 \times 256$to an $N \times N$ (here 70×70) array of outputs X, where each X_{ij} signifies whether the patch ij in the image is real or fake.



Cycle GAN losses

Cycle GAN losses has several losses : First, the standard loss function for cGAN training is defined as follows:

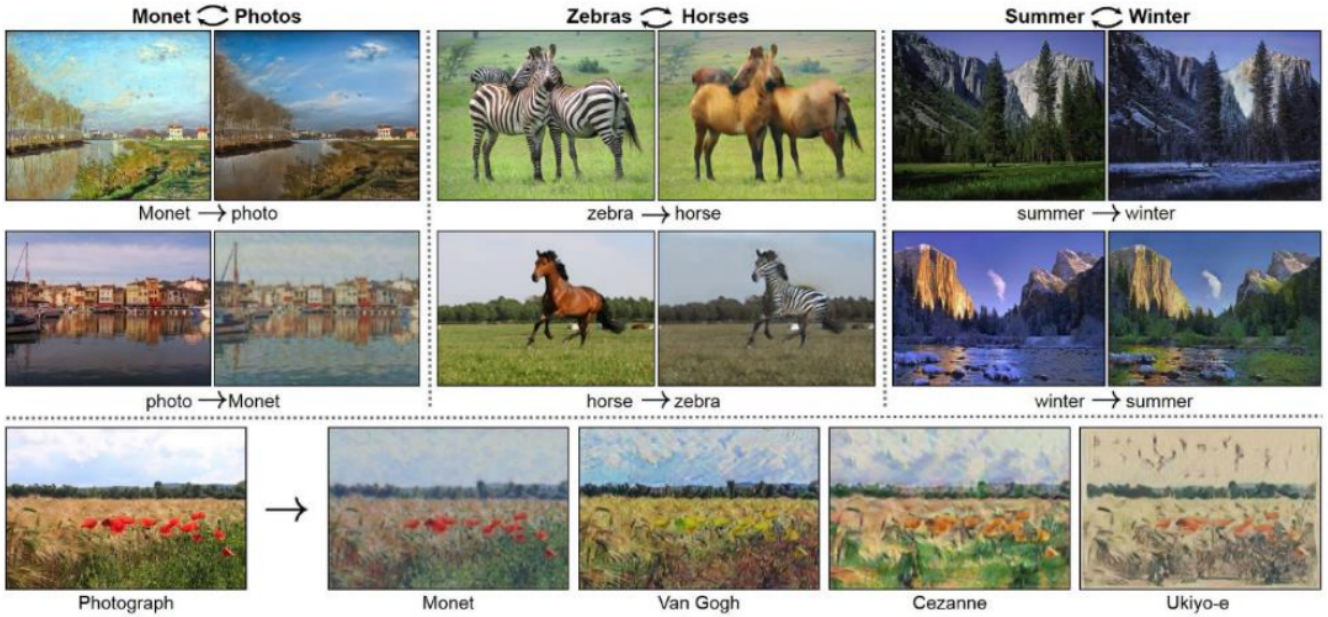
$$\mathcal{L}_{cGAN}(\theta_G, \theta_D) = \mathbb{E}_x[\log D_x(x \mid y)] + \mathbb{E}_z[\log(1 - D_x(G_{Y \to X}(z \mid y)))], \tag{1}$$

Cycle GAN losses

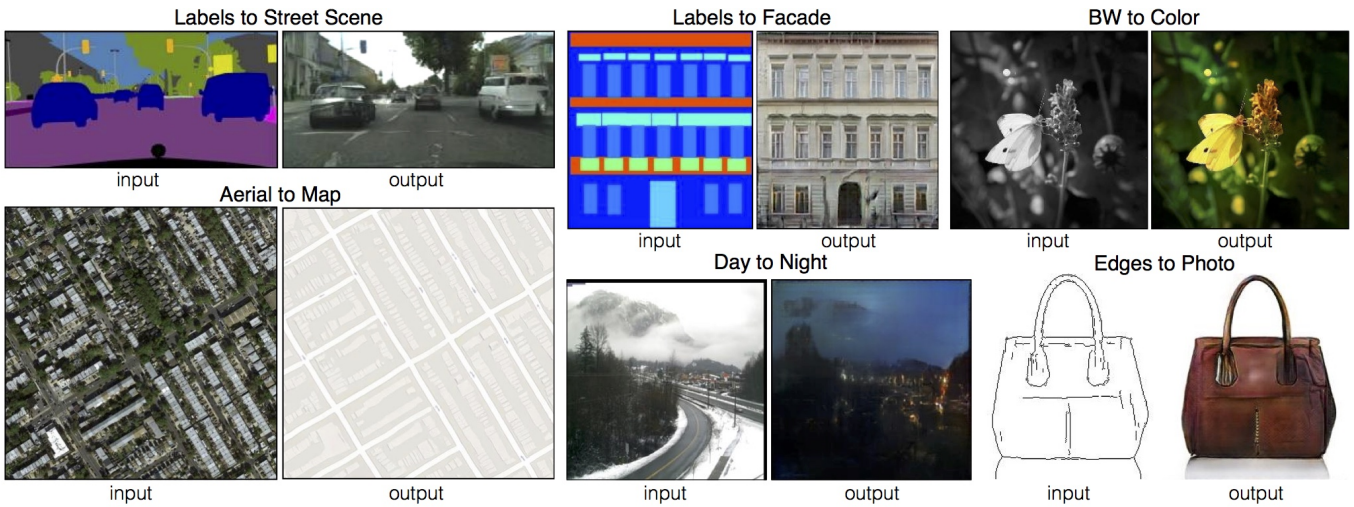
Cycle GAN losses has several losses : Secondly, in addition to the adversarial loss defined, it also have a cycle loss defined by:

$$\mathcal{L}_{L2}(\theta_G) = \mathbb{E}_{(x,y)}[\|y - G_{X\to Y}((G_{Y\to X}(y)) \|_2] + \mathbb{E}_{(x,y)}[\|x - G_{Y\to X}((G_{X\to Y}(x)) \|_2] \quad (2)$$

Cycada [Hoffman2018]

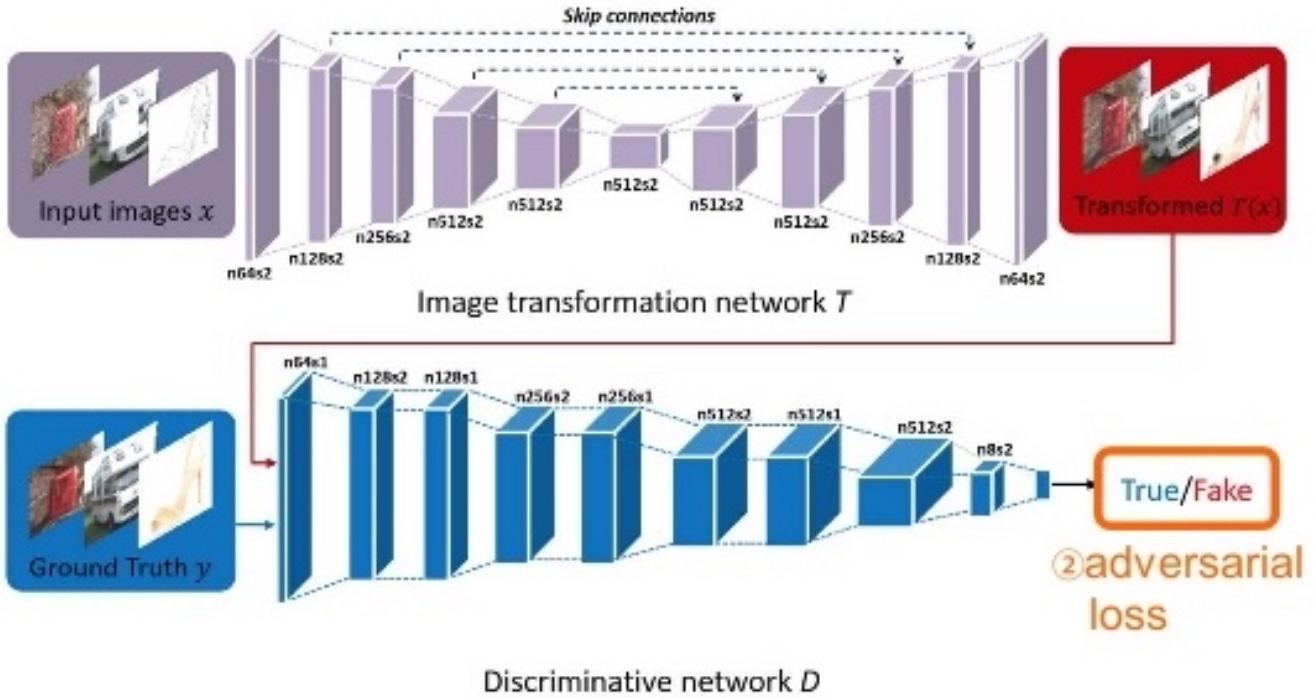


Pix2Pix [Phillip2017]



Pix2Pix [Phillip2017]

it is composed of a Conditional Generator and one discriminator. For Pix2Pix the discriminator is PatchGan and the Generator is a Unet.



Pix2Pix [Phillip2017]

Pix2pix has the following two losses :

$$\mathcal{L}_{cGAN}(\theta_G, \theta_D) = \mathbb{E}_x[\log D_x(x \mid y)] + \mathbb{E}_z[\log(1 - D_x(G_{Y \to X}(z \mid y)))], \tag{3}$$

In addition to the adversarial loss defined, an L1 loss, is added to the cost function of cGANs to reduce blur:

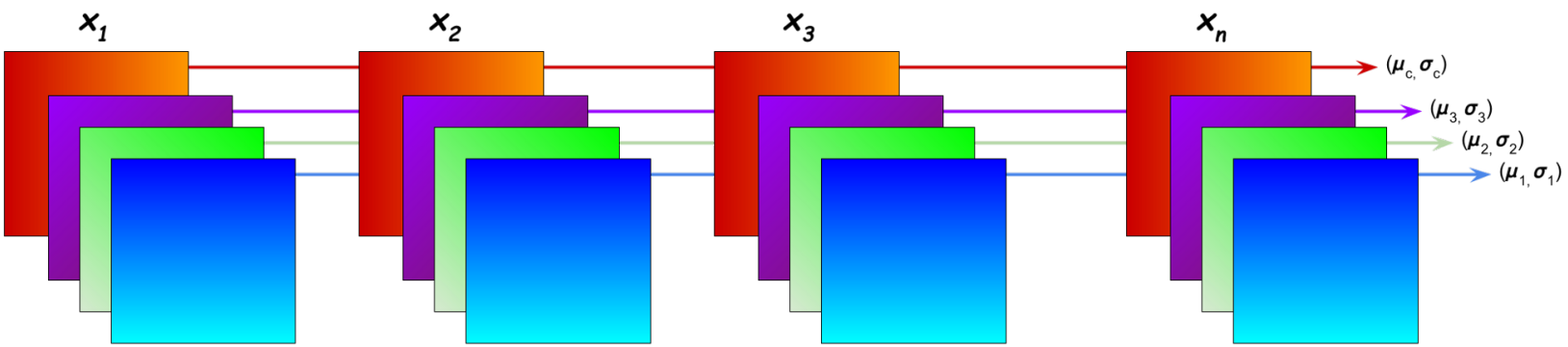
$$\mathcal{L}_{L1}(\theta_G) = \mathbb{E}_{(x,y,z)}[\|x - G(z \mid y)\|_1]. \tag{4}$$

Remember: regularization with Batch normalization

For every channel c we estimate

$$\mu_c = \frac{1}{NHW}\sum_{i=1}^{N}\sum_{j=1}^{H}\sum_{K=1}^{W} x_{icjk} \text{ and } \sigma_c = \frac{1}{NHW}\sum_{i=1}^{N}\sum_{j=1}^{H}\sum_{K=1}^{W}(x_{icjk} - \mu_c)^2 \quad (5)$$

$$\hat{x} = \frac{x - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} \quad (6)$$

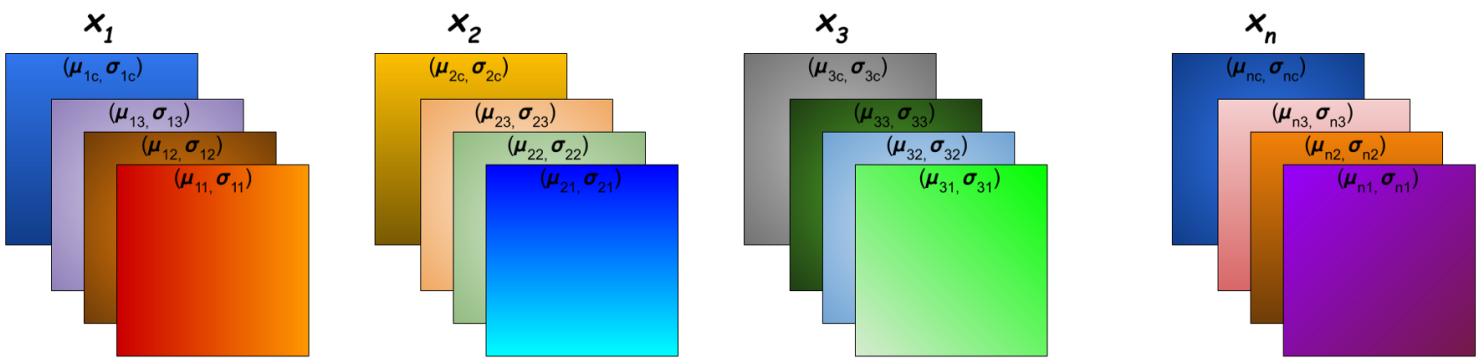


Remember: regularization Instance Normalization

For every channel c we estimate

$$\mu_{nc} = \frac{1}{HW}\sum_{j=1}^{H}\sum_{K=1}^{W} x_{ncjk} \text{ and } \sigma_{nc} = \frac{1}{HW}\sum_{j=1}^{H}\sum_{K=1}^{W}(x_{ncjk} - \mu_{nc})^2 \tag{7}$$

$$\hat{x} = \frac{x - \mu_{nc}}{\sqrt{\sigma_{nc}^2 + \epsilon}} \tag{8}$$



Remember: Batch normalization or Instance Normalization

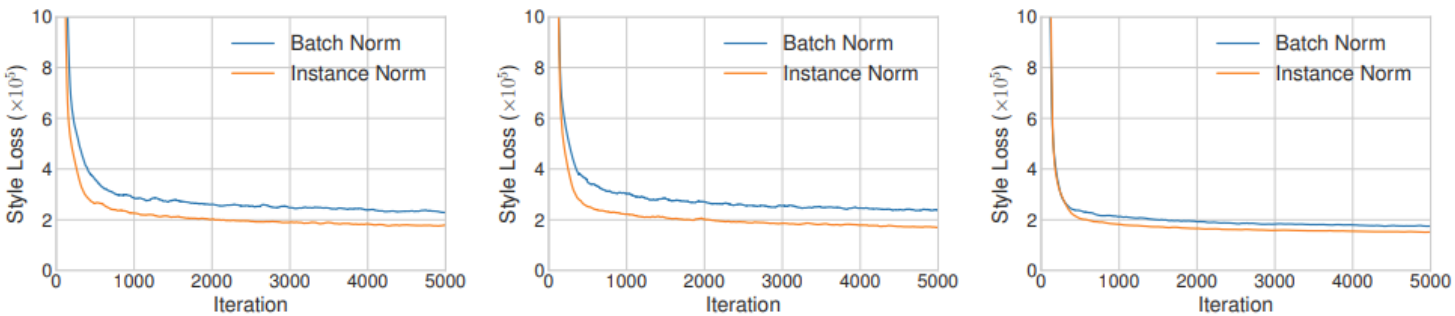
After we have assess $\hat{x}$ we to de-normalise the data

$$BN(x) = \gamma \frac{x - \mu(x)}{\sqrt{\sigma^2(x) + \epsilon}} + \beta \tag{9}$$

$$IN(x) = \gamma \frac{x - \mu(x)}{\sqrt{\sigma^2(x) + \epsilon}} + \beta \tag{10}$$

where γ and β are affine parameters learned during the training

Batch normalization vs Instance Normalization



(a) Trained with original images.

(b) Trained with contrast normalized images.

(c) Trained with style normalized images.

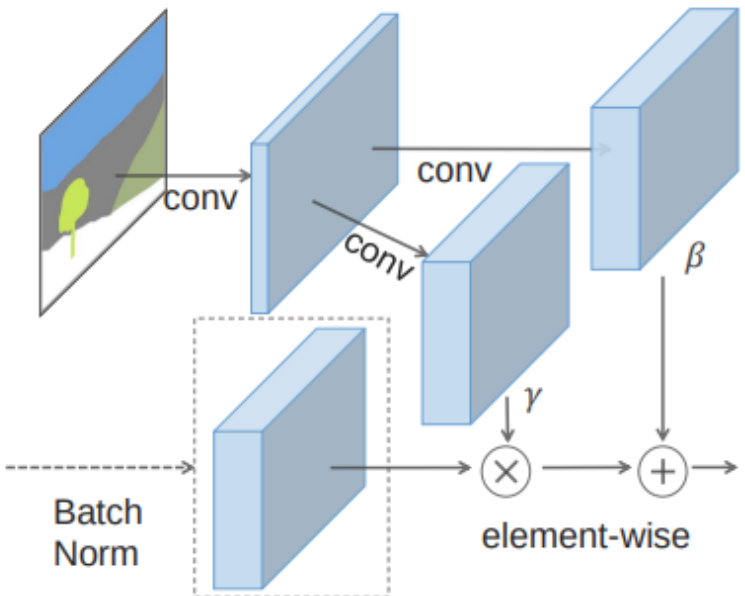
adaptive instance normalization (AdaIN) **[Huang2017]**

AdaIN receives a content input x and a style input y, and simply aligns the channelwise mean and variance of x to match those of y.

$$AdaIN(x, y) = \gamma(y)\frac{x - \mu(x)}{\sqrt{\sigma^2(x) + \epsilon}} + \beta(y) \tag{11}$$

in which we simply scale the normalized content input with $\gamma(y)$, and shift it with $\beta(y)$. Similar to IN, these statistics are computed across spatial locations.

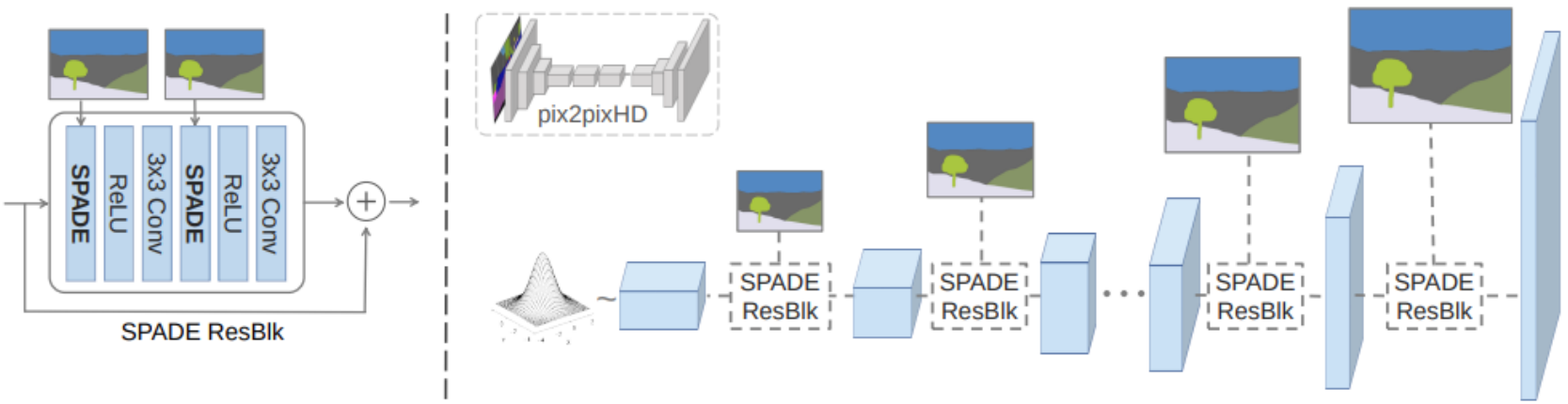
SPatially-Adaptive (DE)normalization (SPADE) [Park2019]



$$Spade(x, \text{label}) = \gamma(\text{label})\frac{x - \mu(x)}{\sqrt{\sigma^2(x) + \epsilon}} + \beta(\text{label}) \tag{12}$$

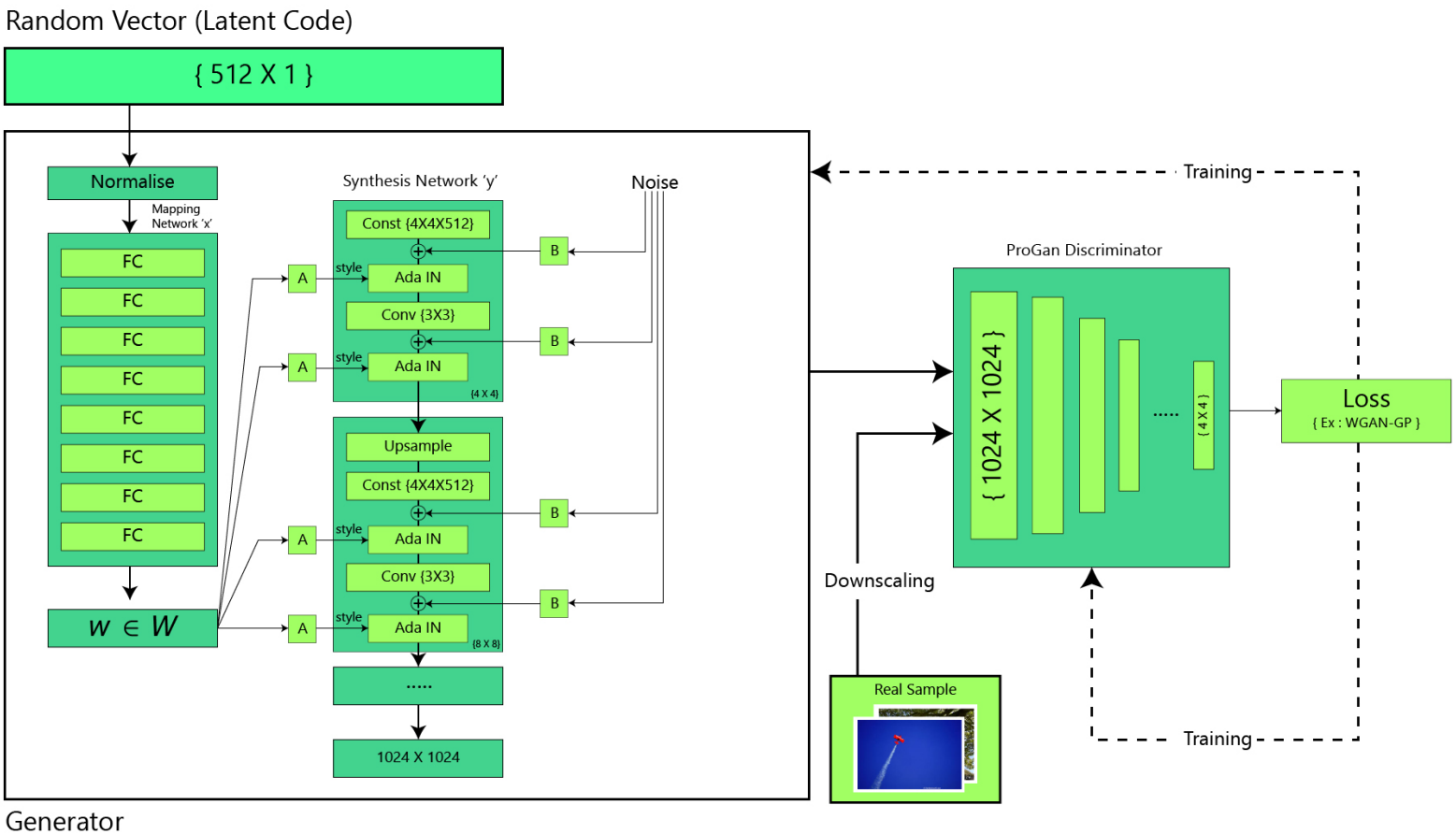
SPatially-Adaptive (DE)normalization (SPADE) [Park2019]

In the SPADE generator, each normalization layer uses the segmentation mask to modulate the layer activations.



The SPADE generator contains a series of the SPADE residual blocks with upsampling layers ant it is just a decoder.

Style GAN [geeksforgeeks]



Style GAN Novelties [geeksforgeeks]

Baseline Progressive Growing GANs: Style GAN uses baseline progressive GAN architecture which means the size of generated image increases gradually from a very low resolution (4×4) to high resolution (1024×1024). This is done by adding a new block to both the models to support the larger resolution after fitting the model on smaller resolution to make it more stable.

Style GAN Novelties [geeksforgeeks]

Mapping Network and Style Network: The goal of the mapping network is to generate the input latent vector into the intermediate vector whose different element control different visual features. Instead of directly providing latent vector to input layer the mapping is used. In this paper, the latent vector (z) of size 512 is mapped to another vector of 512 (w). The mapping function is implemented using 8-layer MLP (8-fully connected layers). The output of mapping network (w) then passed through a learned affine transformation (A) before passing into the synthesis network which AdaIN (Adaptive Instance Normalization) module. This model converts the encoded mapping into the generated image.

Style GAN Novelties [geeksforgeeks]

Removing traditional (Latent) input: Most previous style transfer model uses the random input to create the initial latent code of the generator i.e. the input of the 4×4 level. However the style-GAN authors concluded that the image generation features are controlled by w and AdaIN. Therefore they replace the initial input with the constant matrix of $4 \times 4 \times 512$. This also contributed to increase in the performance of the network.

Style GAN Novelties **[geeksforgeeks]**

Addition of Noisy: Input A Gaussian noise (represented by B) is added to each of these activation maps before the AdaIN operations. A different sample of noise is generated for each block and is interpreted on the basis of scaling factors of that layer.

Style GAN Novelties [Karras2019]

Generative Aversarial Networks: Style GAN (Karras et al., 2019)

Image size:
1024× 1024 px
(source: Karras et al.)

Bibliography

[Srivastava2017] Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U.,& Sutton, C. (2017). Veegan: Reducing mode collapse in gans using implicit variational learning. Advances in neural information processing systems, 30.

[Goodfellow2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. Advances in neural information processing systems, 27.

[Radford2015] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

Bibliography

- **[Hoffman2018]** Hoffman, J., Tzeng, E., Park, T., Zhu, J. Y., Isola, P., Saenko, K., ... & Darrell, T. (2018, July). Cycada: Cycle-consistent adversarial domain adaptation. In International conference on machine learning (pp. 1989-1998). PMLR.
- **[Mirza2014]** Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.
- **[jonathan-hui]** https://jonathan-hui.medium.com/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b
- **[geeksforgeeks]** https://www.geeksforgeeks.org/
- **[alexhwilliams]** http://alexhwilliams.info/itsneuronalblog/2020/10/09/optimal-transport/
- **[Phillip2017]** Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

Bibliography

[Huang2017] Huang, Xun, and Serge Belongie. "Arbitrary style transfer in real-time with adaptive instance normalization." Proceedings of the IEEE international conference on computer vision. 2017.

[Park2019] Park, Taesung, et al. "Semantic image synthesis with spatially-adaptive normalization." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.

[Karras2019] Karras, Tero, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.

[Arjovsky2017] Arjovsky, Martin, Soumith Chintala, and LÃ©on Bottou. "Wasserstein generative adversarial networks." International conference on machine learning. PMLR, 2017.