Deep Generative Models

Generative Adversarial Networks

Hamid Beigy

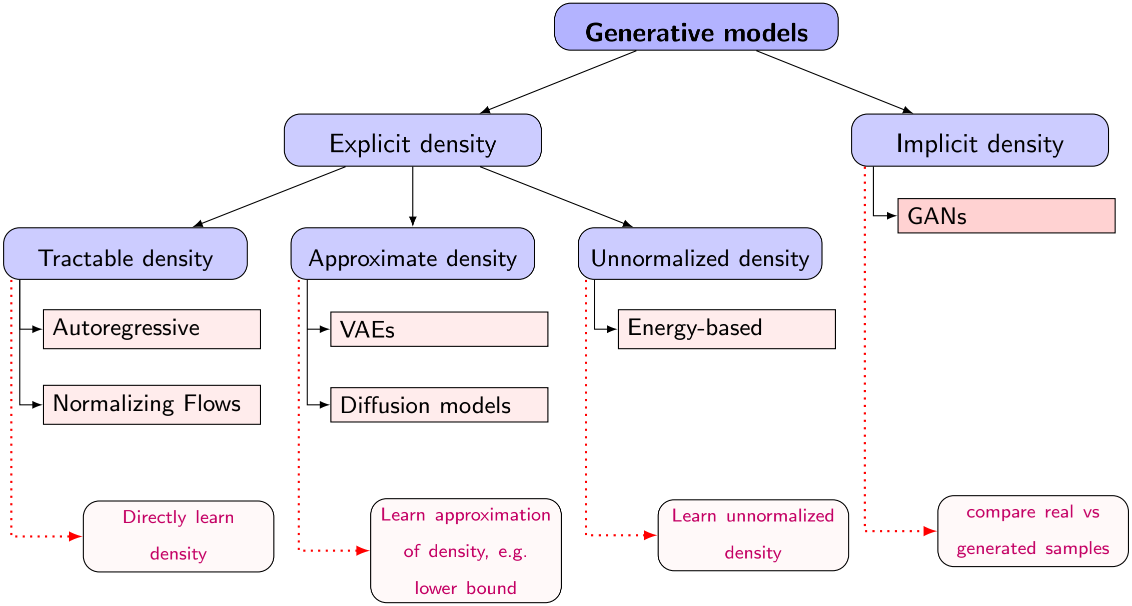
Sharif University of Technology

April 12, 2025

Table of contents

Introduction

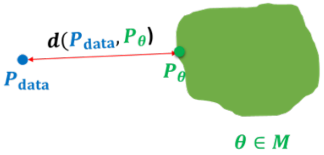
Generative models categories



Generative models

1. Assume that the observed variable $\mathbf{x}$ is a random sample from an underlying process, whose true distribution $p_{\mathrm{d}}(\mathbf{x})$ is unknown.

$\mathbf{x}_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



Model family

2. We attempt to approximate this process with a chosen model, $p_\theta(\mathbf{x})$, with parameters θ such that $\mathbf{x} \sim p_\theta(\mathbf{x})$.
3. Learning is the process of searching for the parameter θ such that $p_\theta(\mathbf{x})$ well approximates $p_{\mathrm{d}}(\mathbf{x})$ for any observed $\mathbf{x}$, i.e.

$$p_\theta(\mathbf{x}) \approx p_{\mathrm{d}}(\mathbf{x})$$

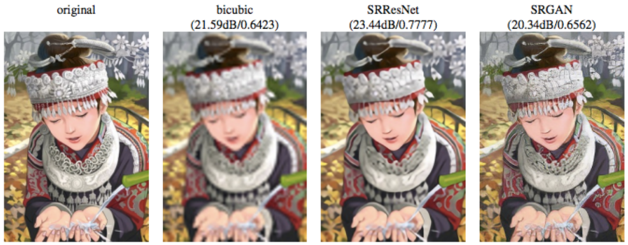
4. We wish $p_\theta(\mathbf{x})$ to be sufficiently flexible to be able to adapt to the data for obtaining sufficiently accurate model and to be able to incorporate prior knowledge.

Credit: Aditya Grover

Generative Aversarial networks

Generative Adversarial Networks

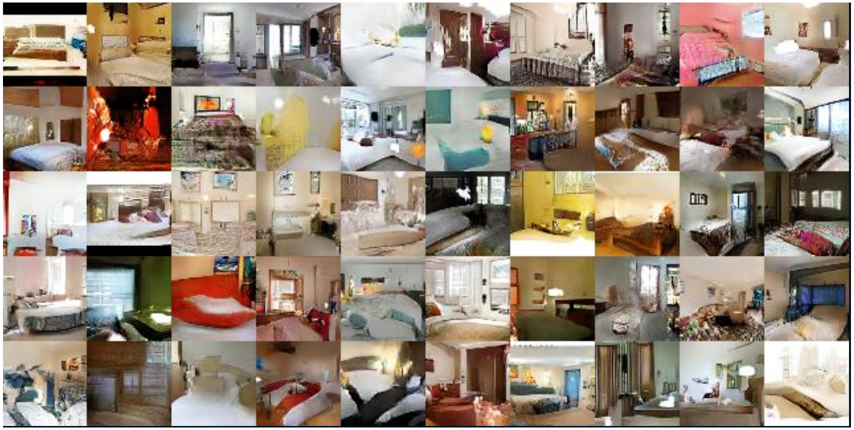
1. Generative adversarial networks (GANs) are a new way to implicitly build generative models $p(\mathbf{x})$ (Goodfellow et al. 2014).
2. Generative adversarial networks
   - Generative: Learns a generative model.
   - Adversarial: Trained in an adversarial setting
   - Networks: Use Deep Neural Networks
3. Which one is computer generated?



4. How do we generate a fake image?
5. Can we generate a fake image from a random number?

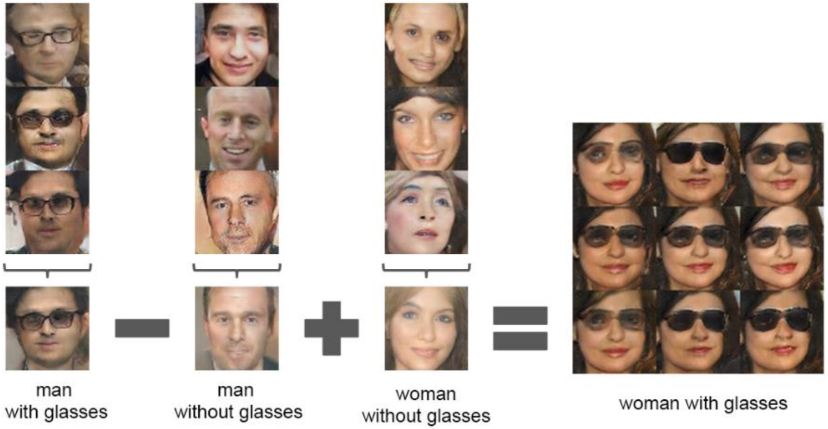
GAN results

1. Results obtained from GAN (Radford, Metz, and Chintala 2016).



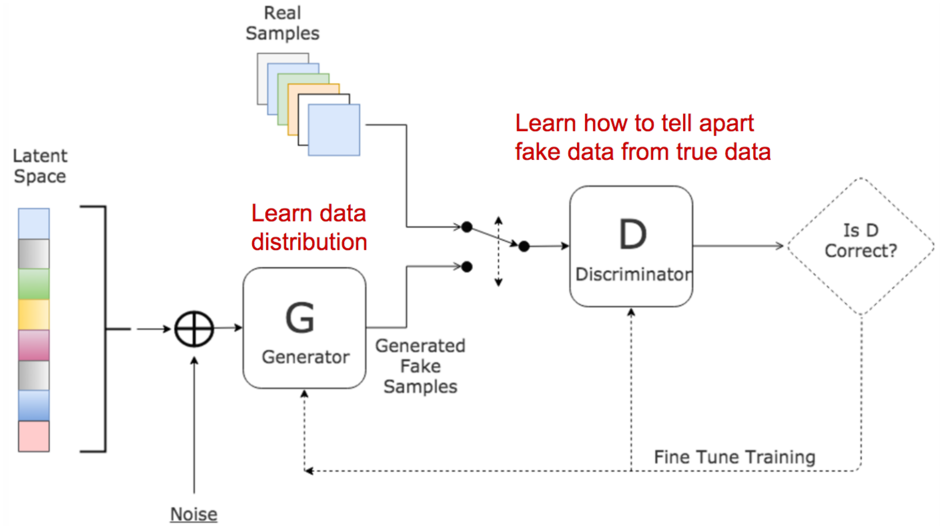
GAN results

1. Results obtained from GAN (Radford, Metz, and Chintala 2016).



GAN Architecture

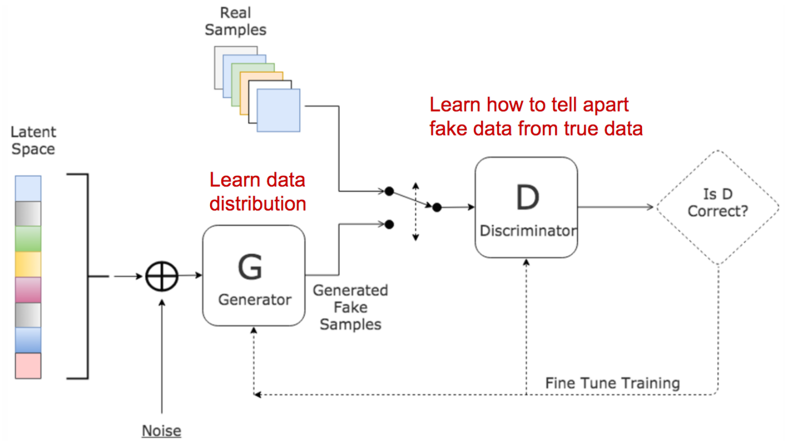
1. GAN has the following architecture



2. **z** (input to generator) is some random noise (Gaussian/Uniform).
3. **z** can be thought as the latent representation of the image.

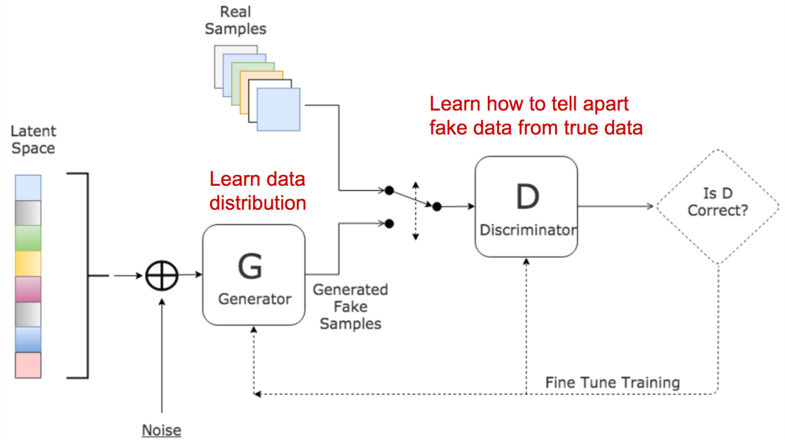
Generator

1. The generator tries to learn $p(\mathbf{x} \mid \mathbf{z})$.
2. Inputs are directly sampled from $q(\mathbf{z})$.
3. Problem: No true data $\mathbf{x}$ is provided when training the generator
4. Instead of a traditional loss function, gradient is provided by a discriminator (another network).



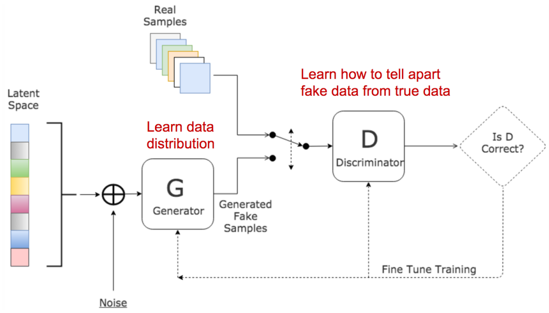
Discriminator

1. The discriminator attempts to tell the difference between real and fake images.
2. It tries to learn $p(\mathbf{y} \mid \mathbf{x})$: $\mathbf{y}$ is the label and $\mathbf{x}$ is the real/generated data.
3. Trained using standard cross entropy loss to assign the correct label.
4. Generator weights are frozen while training discriminator; inputs are generated data and real data, labels are 0 and 1
5. From generator's point-of-view, discriminator is a black-box loss function



GAN Architecture

1. Let
   - $p_g(\mathbf{x})$ be the probability of generating a fake sample.
   - $p_\mathrm{d}(\mathbf{x})$ be the probability of generating a real sample.
   - $D(\mathbf{x})$ (output of discriminator), be the probability that $\mathbf{x}$ is a real sample.
2. The discriminator is expected to output a probability $D(G(\mathbf{z}))$:
   - For a fake sample $G(\mathbf{z})$: close to zero by **maximizing** $\mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}(\mathbf{z})}[\log(1-D(G(\mathbf{z})))]$.
   - For a real sample $\mathbf{x}$: close to one by **maximizing** $\mathbb{E}_{\mathbf{x}\sim p_\mathrm{d}(\mathbf{x})}[\log D(\mathbf{x})]$.



3. The generator is trained to increase the chances of D producing a high probability for a fake example, thus **minimize** $\mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}(\mathbf{z})}[\log(1-D(G(\mathbf{z})))]$.

Min-Max Game

1. When combining both aspects together, D and G are playing a minimax game in which we should optimize

$$\begin{aligned}\min_G \max_D V(D,G) &= \min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\text{d}}(\mathbf{x})}[\log D(x)] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1-D(G(\mathbf{z})))] \\ &= \min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\text{d}}(\mathbf{x})}[\log D(x)] + \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}(g)}[\log(1-D(\mathbf{x}))]\end{aligned}$$

2. Thus, $\mathbb{E}_{\mathbf{x}\sim p_{\text{d}}(\mathbf{x})}[\log D(\mathbf{x})]$ has no impact on G during gradient descent updates.
3. Loss function is

$$\begin{aligned}V(G,D) &= \int_{\mathbf{x}} p_{\text{d}}(\mathbf{x})\log(D(\mathbf{x}))d\mathbf{x} + \int_{\mathbf{z}} p_{\mathbf{z}}(\mathbf{z})\log(1-D(G(\mathbf{z})))d\mathbf{z} \\ &= \int_{x}(p_{\text{d}}(\mathbf{x})\log(D(\mathbf{x})) + p_g(\mathbf{x})\log(1-D(\mathbf{x})))d\mathbf{x}\end{aligned}$$

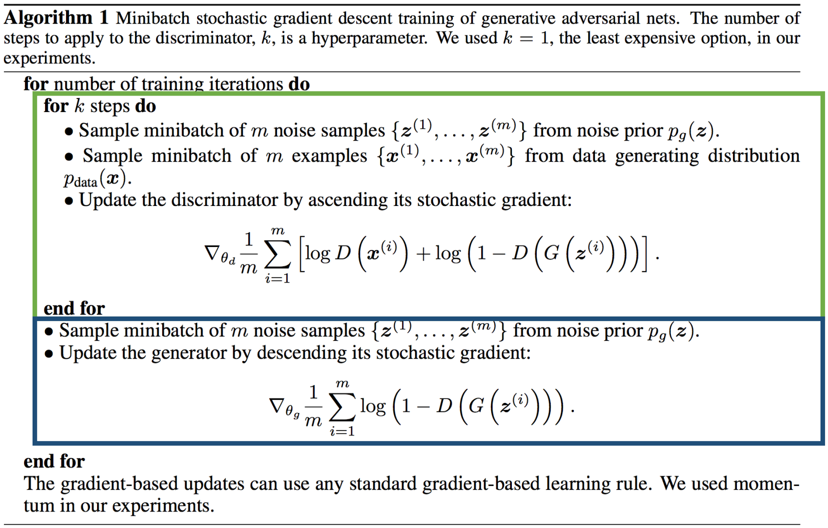
4. The full two-player game can be summarily described by the below.

$$\min_G \max_D V(D,G)$$

Training GAN

1. It is important to understand that both the generator and discriminator are trying to learn moving targets.
2. Both networks are trained simultaneously.
3. The discriminator needs to update based on how well the generator is doing.
4. The generator is constantly updating to improve performance on the discriminator.
5. These two need to be balanced correctly to achieve stable learning instead of chaos.

Training GAN



Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

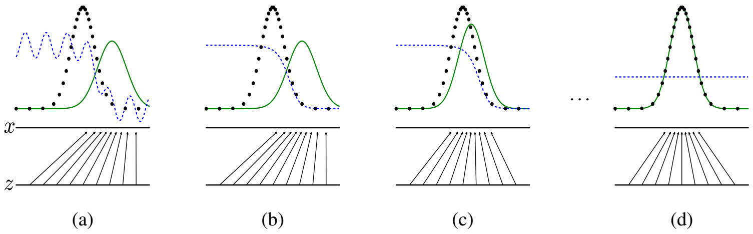
```
for number of training iterations do
    for $k$ steps do
        • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
        • Sample minibatch of $m$ examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
        • Update the discriminator by ascending its stochastic gradient:
            $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right].$
    end for
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Update the generator by descending its stochastic gradient:
        $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right).$
end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.
```

Discriminator updates

Generator updates

Training GAN

1. How GAN is trained?



2. discriminative distribution $D(\mathbf{x})$, real data $p_{\mathrm{d}}(\mathbf{x})$, generative distribution $p_g(\mathbf{x})$.

   (a) An adversarial pair near convergence: $p_g(\mathbf{x})$ is similar to $p_{\mathrm{d}}(\mathbf{x})$ and D is a partially accurate classifier.

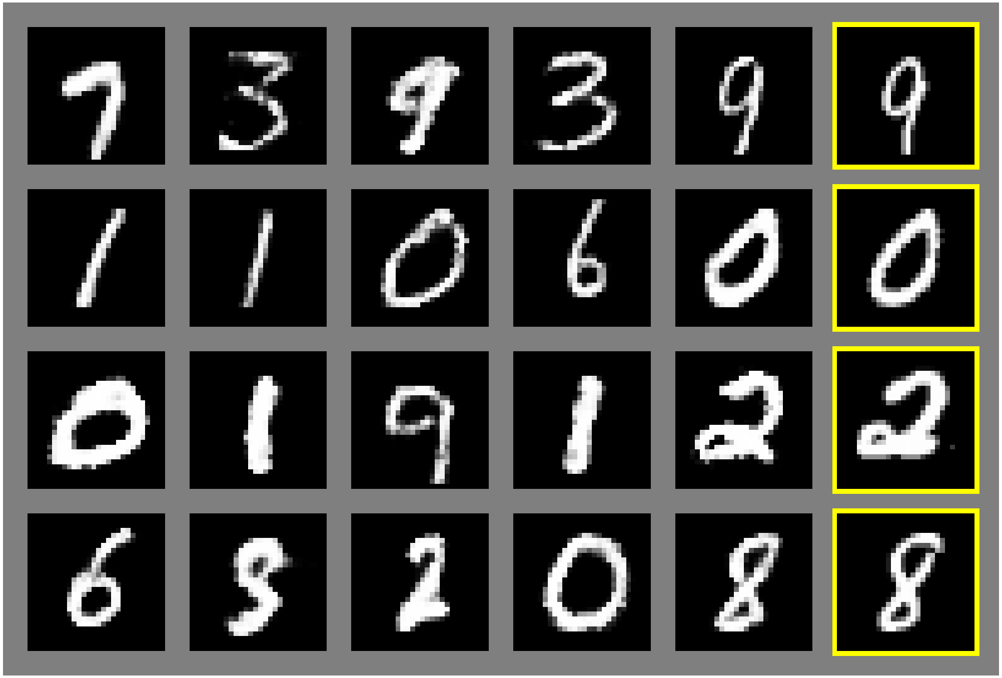
   (b) In inner loop of algorithm, D is trained to discriminate samples from data, converging to $D^*(\mathbf{x})$.

   (c) After an update to G, gradient of D has guided $G(\mathbf{z})$ to flow to regions that are more likely to be classified as data.

   (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g(\mathbf{x}) = p_{\mathrm{d}}(\mathbf{x})$.

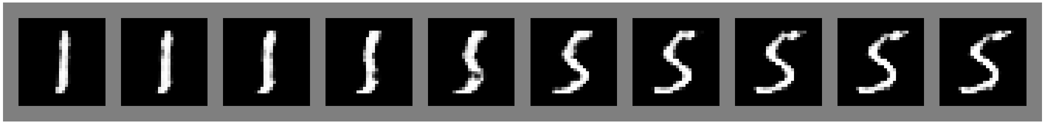
GAN Results

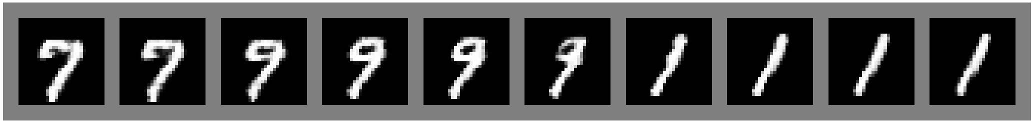
Rightmost column shows the nearest training example of the neighboring sample.



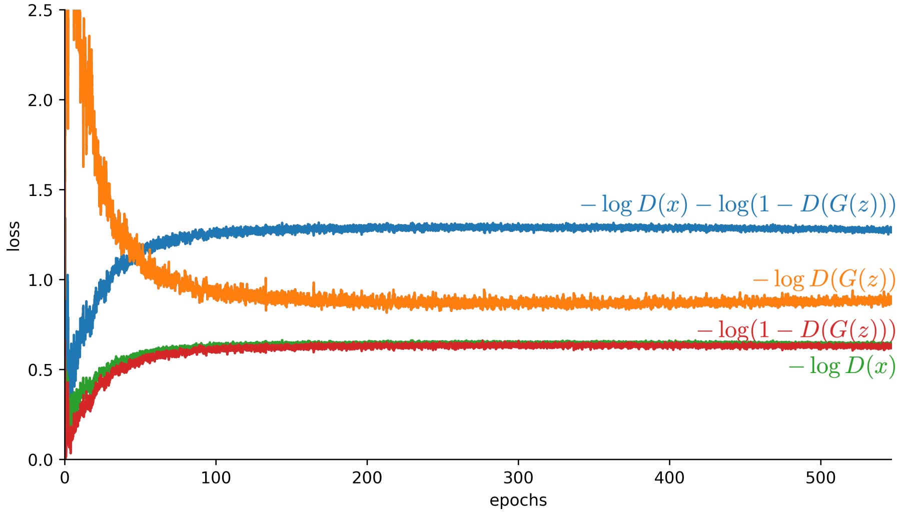
GAN Results

Digits obtained by linearly interpolating between coordinates in $\mathbf{z}$ space of the full model.





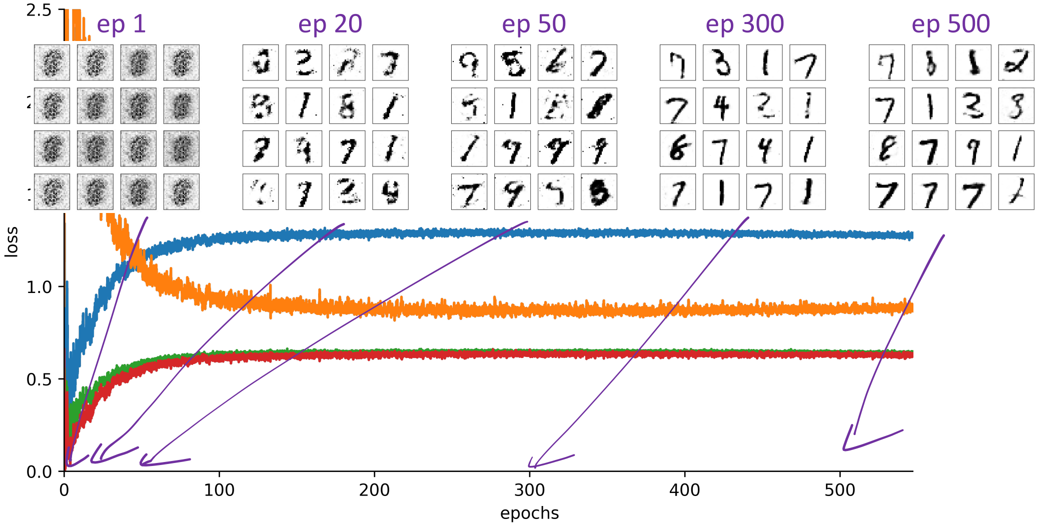
GAN Results (MNIST)



- D loss
- G loss
- $\text{loss}_{\text{fake}}$
- $\text{loss}_{\text{real}}$

Credit: Kaiming Her

GAN Results (MNIST)



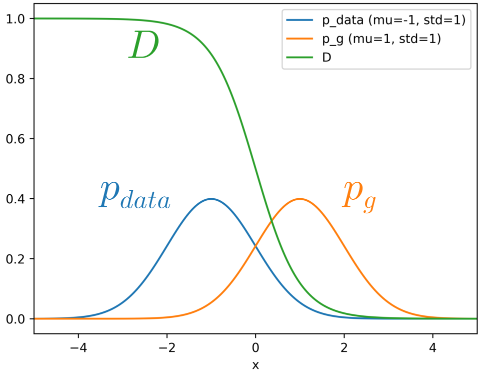
Credit: Kaiming Her

Optimality of GAN

Theorem (Optimality of GAN)

For G fixed, the optimal discriminator D is

$$D^*(\mathbf{x}) = \frac{p_{\mathrm{d}}(\mathbf{x})}{p_{\mathrm{d}}(\mathbf{x}) + p_g(\mathbf{x})}$$



Optimality of GAN

Proof.

The training criterion for the discriminator D, given any generator G, is to maximize the quantity $V(G, D)$.

$$\begin{aligned} V(G,D) &= \int_{\mathbf{x}} p_{\mathrm{d}}(\mathbf{x}) \log D(\mathbf{x}) d\mathbf{x} + \int_{\mathbf{z}} p_{\mathrm{z}}(\mathbf{z}) \log(1 - D(G(\mathbf{z}))) d\mathbf{z} \\ &\leq \int_{\mathbf{x}} \max_{y} \; p_{\mathrm{d}}(\mathbf{x}) \log y + p_g(\mathbf{x}) \log(1-y) d\mathbf{x} \end{aligned}$$

Let $f(y) = a \log y + b \log(1-y)$. To find the critical point,

$$f'(y) = 0 \Rightarrow \frac{a}{y} - \frac{b}{1-y} = 0 \Rightarrow y = \frac{a}{a+b}$$

When $a + b \neq 0$, then we test the second derivative:

$$f''\left(\frac{a}{a+b}\right) = -\frac{a}{(\frac{a}{a+b})^2} - \frac{b}{(1-\frac{a}{a+b})^2} < 0$$

When $a, b \in (0,1)$, point $\frac{a}{a+b}$ is **maximum**. □

Convergence of training algorithm of GAN

Theorem (Convergence of training algorithm of GAN)

If G and D have enough capacity, and at each step of training Algorithm, the discriminator is allowed to reach its optimum given G, and $p_g(\mathbf{x})$ is updated so as to improve the criterion $V(D,G)$, then, $p_g(\mathbf{x})$ converges to $p_{\mathrm{d}}(\mathbf{x})$

What is the global optimal?

1. When both G and D are at their optimal values, we have $p_g(\mathbf{x}) = p_{\mathrm{d}}(\mathbf{x})$ and $D^*(\mathbf{x}) = \frac{1}{2}$, and the loss function becomes:

$$\begin{aligned} V(G, D^*) &= \int_{\mathbf{x}} p_{\mathrm{d}}(\mathbf{x}) \log(D^*(\mathbf{x})) + p_g(\mathbf{x}) \log(1 - D^*(\mathbf{x})) d\mathbf{x} \\ &= \log \frac{1}{2} \int_{\mathbf{x}} p_{\mathrm{d}}(\mathbf{x}) d\mathbf{x} + \log \frac{1}{2} \int_{\mathbf{x}} p_g(\mathbf{x}) d\mathbf{x} \\ &= -2 \log 2 \end{aligned}$$

What does the loss function represent?

1. KL divergence measures how one probability distribution $p(\mathbf{x})$ diverges from a second probability distribution $q(\mathbf{x})$

$$D_{KL}(\, p(\mathbf{x}) \,||\, q(\mathbf{x})) = \int_x p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x}$$

2. KL divergence is asymmetric.
3. In cases where $p(\mathbf{x})$ is close to zero, but $q(\mathbf{x})$ is significantly non-zero, the $q(\mathbf{x})$'s effect is disregarded.
4. Jensen–Shannon Divergence is a measure of similarity between two probability distributions, bounded by $[0, 1]$.

$$D_{JS}(\, p(\mathbf{x}) \,||\, q(\mathbf{x})) = \frac{1}{2} D_{KL}\left(p(\mathbf{x}) \,||\, \frac{p(\mathbf{x}) + q(\mathbf{x})}{2} \right) + \frac{1}{2} D_{KL}\left(q(\mathbf{x}) \,||\, \frac{p(\mathbf{x}) + q(\mathbf{x})}{2} \right)$$

5. JS divergence is symmetric and more smooth.

What does the loss function represent?

1. JS divergence between $p_{\mathrm{d}}(\mathbf{x})$ and $p_g(\mathbf{x})$ can be computed as:

$$\begin{aligned}
\mathrm{D}_{JS}(p_{\mathrm{d}}(\mathbf{x}) \,||\, p_g(\mathbf{x})) =& \frac{1}{2}\mathrm{D}_{KL}\left(p_{\mathrm{d}}(\mathbf{x}) \,||\, \frac{p_{\mathrm{d}}(\mathbf{x}) + p_g(\mathbf{x})}{2}\right) + \frac{1}{2}\mathrm{D}_{KL}\left(p_g(\mathbf{x}) \,||\, \frac{p_{\mathrm{d}}(\mathbf{x}) + p_g(\mathbf{x})}{2}\right) \\
=& \frac{1}{2}\left(\log 2 + \int_{\mathbf{x}} p_{\mathrm{d}}(\mathbf{x}) \log \frac{p_{\mathrm{d}}(\mathbf{x})}{p_{\mathrm{d}}(\mathbf{x}) + p_g(\mathbf{x})} d\mathbf{x}\right) \\
+& \frac{1}{2}\left(\log 2 + \int_{\mathbf{x}} p_g(\mathbf{x}) \log \frac{p_g(\mathbf{x})}{p_{\mathrm{d}}(\mathbf{x}) + p_g(\mathbf{x})} d\mathbf{x}\right) \\
=& \frac{1}{2}(\log 4 + V(G, D^*))
\end{aligned}$$

2. Thus

$$V(G, D^*) = 2\,\mathrm{D}_{JS}(p_{\mathrm{d}}(\mathbf{x}) \,||\, p_g(\mathbf{x})) - 2\log 2$$

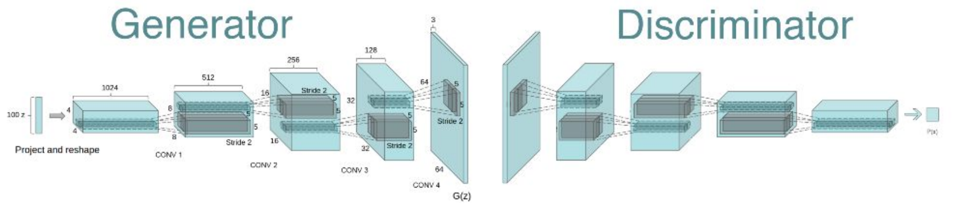
3. The best G^* that replicates the real data distribution leads to the minimum $V(G^*, D^*) = -2\log 2$, which is aligned to the optimal solution.

Generative Aversarial networks

Deep Convolutional GAN

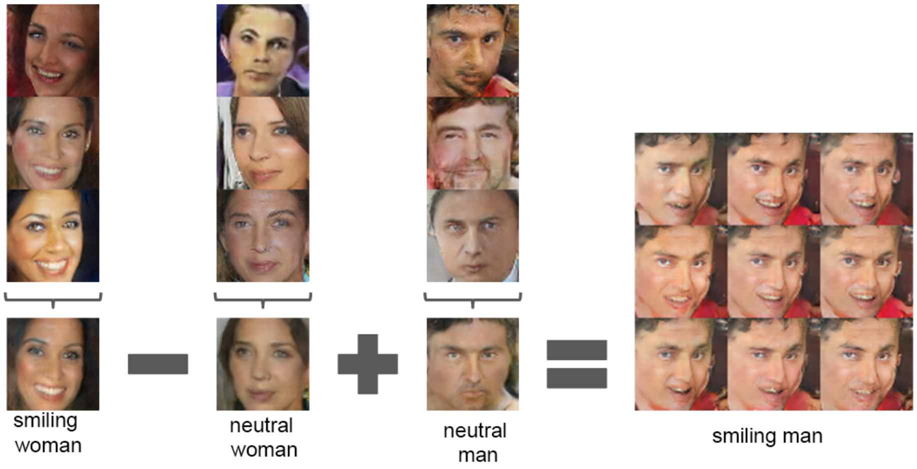
Deep Convolutional Generative Adversarial Networks

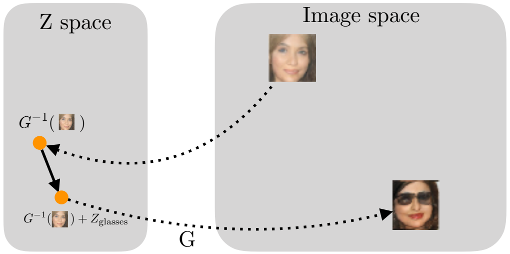
1. DCGAN maps from random noise to an image matrix.
2. It uses convolutional Layers in the generator network to produce better results (Radford, Metz, and Chintala 2016).
3. Combine CNN and GAN for unsupervised learning.
4. Learns a hierarchy of feature representations
5. The Generator uses fractional-strided convolutions followed by batch normalisation and ReLU activation for all layers except for the last that uses tanh activation.
6. The Discriminator uses strided convolutions followed by batch normalisation and LeakyReLU activation for all layers except for a single sigmoid output.



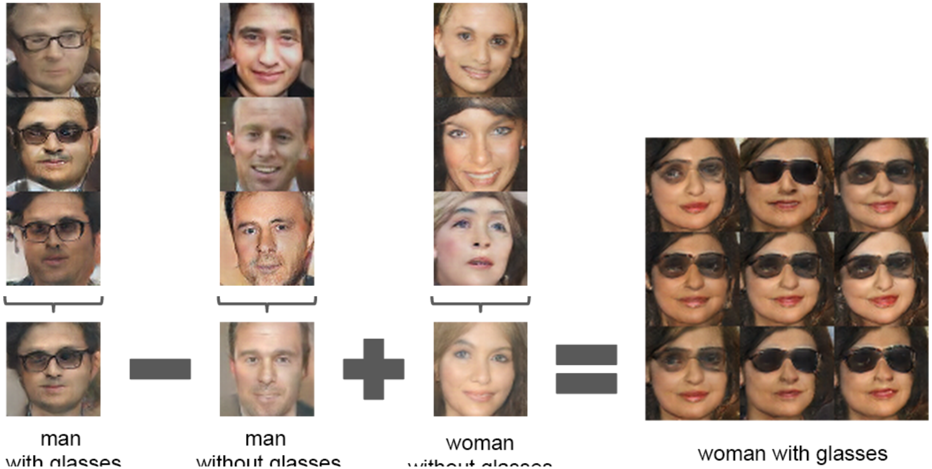
DCGAN results

DCGAN results

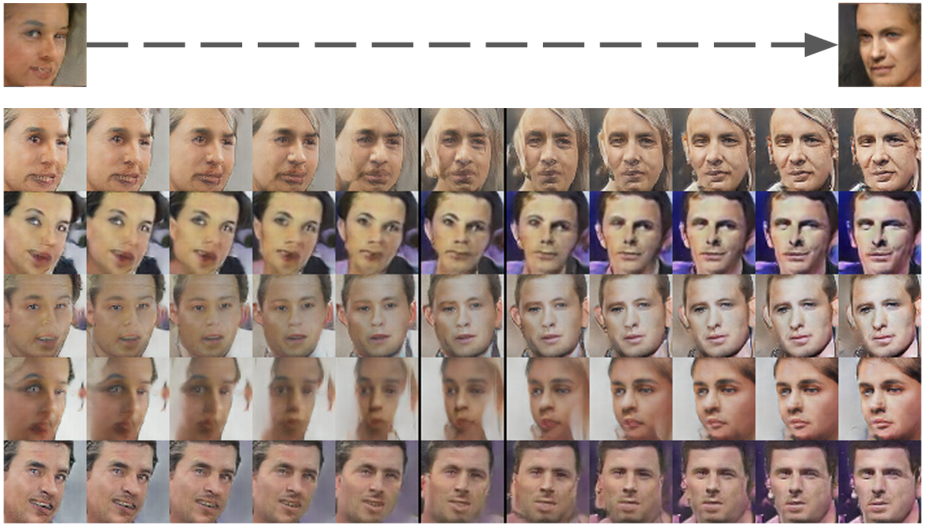




DCGAN results



DCGAN results



Problems in GANs

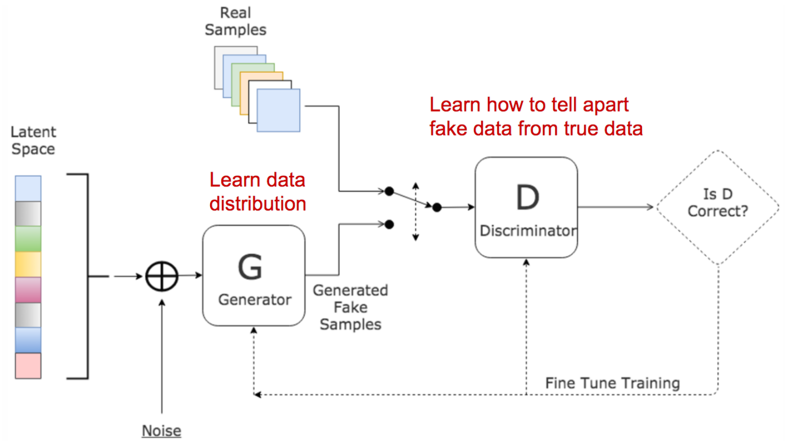
1. No explicit representation of $p_g(\mathbf{x})$.
2. Easy to get trapped in local optima that memorize training data.
3. Hard to invert generative model to get back latent $\mathbf{z}$ from generated $\mathbf{x}$.
4. Hard to achieve Nash equilibrium (Salimans et al. 2016).
5. Low dimensional supports: When the intrinsic dimension is low, then training GAN will be unstable (Arjovsky and Leon Bottou 2017).
6. Vanishing gradient: When the discriminator is perfect, loss function is zero and there is not any training.
7. Mode collapse: During the training, the generator may collapse to a setting where it always produces same outputs.

Improved GAN Training

1. Feature Matching: This suggests to optimize the discriminator to inspect whether the generator's output matches expected statistics of the real samples (Salimans et al. 2016). New objective function
$$\|\mathbb{E}_{x\sim p_r} f(x) - \mathbb{E}_{z\sim p_z(z)} f(G(z))\|_2^2$$
where $f(x)$ can be any computation of statistics of features, such as mean or median.
2. Mini-batch Discrimination: Instead of processing each point independently, the discriminator is able to digest the relationship between training data points in one batch.
3. Historical Averaging: This adds the following term to the loss function
$$\left\|\theta - \frac{1}{t}\sum_{i=1}^{t}\theta[i]\right\|^2$$
that penalizes the training speed when parameters are changing too dramatically in time.
4. One-sided Label Smoothing: When feeding the discriminator, instead of providing 1 and 0 labels, use soften values such as 0.9 and 0.1

Improved GAN Training

1. Virtual Batch Normalization: Each data sample is normalized based on a fixed batch (reference batch) of data rather than within its mini-batch. The reference batch is chosen once at the beginning and stays the same through training.
2. Adding Noises:



3. Use Better Metric of Distribution Similarity: The JS divergence fails to provide a meaningful value when two distributions are disjoint.

Generative Aversarial networks

Non-saturating GAN

Non-saturating GAN

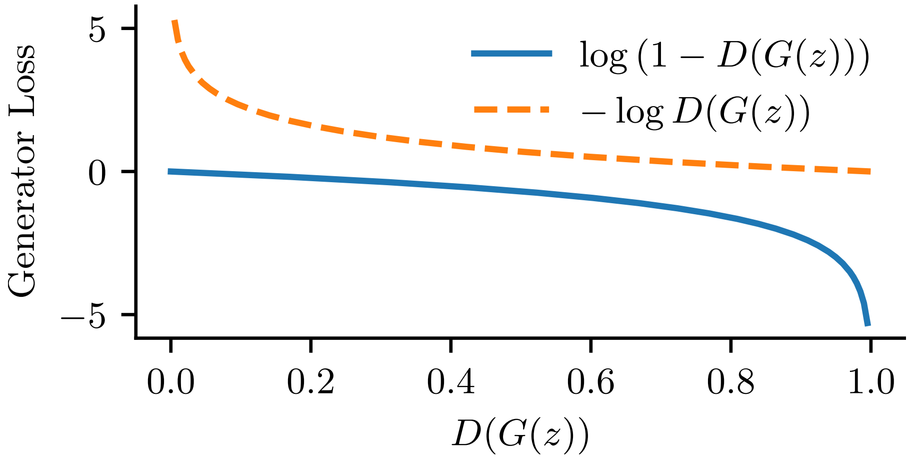
1. G is poor in early learning and samples are clearly different from the training data.
2. Therefore, D can reject the generated samples with high confidence.
3. In this situation, $\log(1 - D(G(z)))$ saturates. This means $D(G(z))$ is close to zero.
4. As a result, the back-propagated gradient $\nabla_{\theta_G} \log(1 - D(G(z)))$ is also small.
5. Fortunately, the following simple mathematical trick solves the problem:

$$\begin{aligned}
\min J(G) &= \min \frac{1}{m}\sum_{i=1}^{m} \log(1 - D(G(\mathbf{z}_i))) \\
&= \max \frac{1}{m}\sum_{i=1}^{m} \log D(G(z_i)) \\
&= \min -\frac{1}{m}\sum_{i=1}^{m} \log D(G(z_i))
\end{aligned}$$

6. This trick ensures a higher gradient signal early in the training.

Non-saturating GAN

1. This new objective function results in the same fixed point of the dynamics of D and G but provides much larger gradients early in learning.



2. The non-saturating game is heuristic, not being motivated by theory.

Non-saturating GAN

1. The non-saturating game has other problems such as unstable numerical gradient for training G.
2. With optimal D^*, we have

$$\begin{aligned}\mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[-\log D^*(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[\log(1-D^*(\mathbf{x}))] &= \mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}\left[\frac{\log(1-D^*(\mathbf{x}))}{\log D^*(\mathbf{x})}\right] \\ &= \mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}\left[\frac{p_g(\mathbf{x})}{p_{\mathrm{d}}(\mathbf{x})}\right] \\ &= \mathrm{D}_{KL}(\, p_g(\mathbf{x}) \,||\, p_{\mathrm{d}}(\mathbf{x}))\end{aligned}$$

3. Therefore, we have

$$\mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[-\log D^*(\mathbf{x})] = \mathrm{D}_{KL}(\, p_g(\mathbf{x}) \,||\, p_{\mathrm{d}}(\mathbf{x})) - \mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[\log(1-D^*(\mathbf{x}))]$$

4. We also had

$$\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{d}}(\mathbf{x})}[\log D^*(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[\log(1-D^*(\mathbf{x}))] = 2\,\mathrm{D}_{JS}(\, p_g(\mathbf{x}) \,||\, p_{\mathrm{d}}(\mathbf{x})) - 2\log 2$$

5. Therefore, we obtain

$$\mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[\log(1-D^*(\mathbf{x}))] = 2\,\mathrm{D}_{JS}(\, p_g(\mathbf{x}) \,||\, p_{\mathrm{d}}(\mathbf{x})) - 2\log 2 - \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{d}}(\mathbf{x})}[\log D^*(\mathbf{x})]$$

Non-saturating GAN

1. By combining the preceding equations, we obtain

$$\begin{aligned}\mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[-\log D^*(\mathbf{x})] = \mathrm{D}_{KL}(p_g(\mathbf{x}) \mid\mid p_{\mathrm{d}}(\mathbf{x})) - 2\,\mathrm{D}_{JS}(p_g(\mathbf{x}) \mid\mid p_{\mathrm{d}}(\mathbf{x})) \\ + \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{d}}(\mathbf{x})}[\log D^*(\mathbf{x})] + 2\log 2\end{aligned}$$

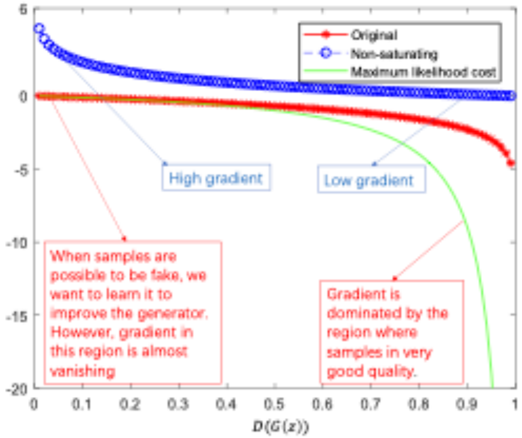
2. This alternative loss is contradictory because of:
   - The first term aims to make the divergence between the generated distribution and the real distribution as small as possible.
   - The second term aims to make the divergence between these two distributions as large as possible due to the negative sign.
3. This will bring unstable numerical gradient for training G.

Maximum likelihood game

1. There are many methods to approximate the objective function of GANs.
2. Under the assumption that the discriminator is optimal, minimizing

$$\mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{x})}\left[-\exp\left(\sigma^{-1}(\log D^*(G(\mathbf{z})))\right)\right] = \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{x})}\left[-\frac{D^*(G(\mathbf{z}))}{1-D^*(G(\mathbf{z}))}\right]$$

where σ is the logistic sigmoid function.



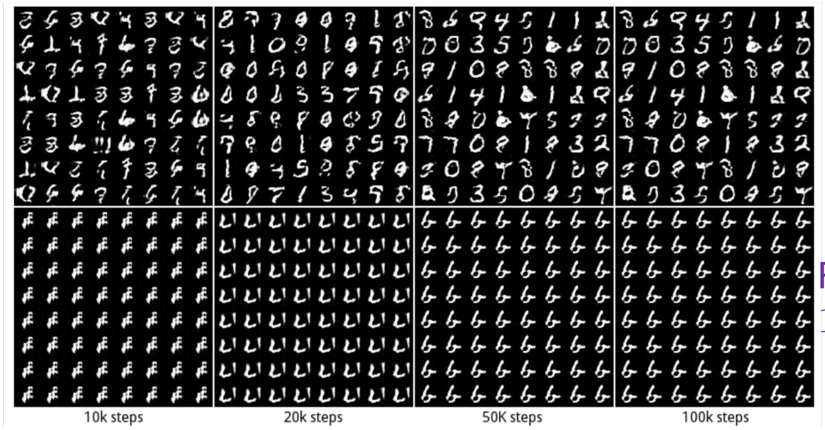
Generative Aversarial networks

Mode Collapse[a]

[a] Some slides are taken from Sargur N. Srihari lectures

Mode Collapse

1. Real-life data is multi-modal (e.g.10 in MNIST).
2. Mode collapse occurs when GAN generates only few modes.

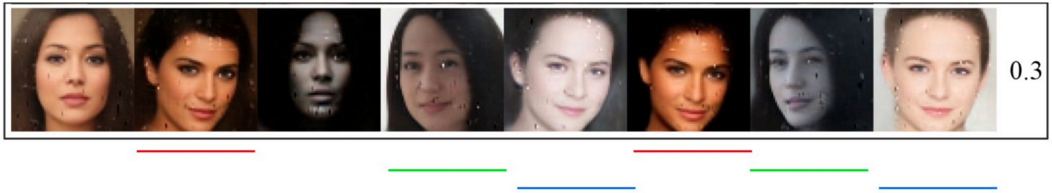


Row 1 has all 10 modes

Row 2 has only 1 mode for 6

Mode Collapse

1. Mode collapse is a hard problem to solve in GAN.
2. A complete collapse is not common but a partial collapse happens often.
3. The following images with the same underlined color look similar and the mode starts collapsing.



Reason for Mode Collapse

1. The goal of generator is to create images to overcome discriminator.
2. The generator use the following gradient to update its parameters.

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(\mathbf{z}_i)))$$

3. Consider the case when generator is trained without updates to discriminator.
   - Generated images converge to $\mathbf{x}^*$ that overcome discriminator.
   - These images are most realistic from the discriminator perspective.
   - In this extreme, $\mathbf{x}^*$ will be independent of $\mathbf{z}$.

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} D(\mathbf{x})$$

   - Mode collapses to a single point in which the gradient associated with $\mathbf{z}$ approaches zero, i.e.

$$\frac{\partial J}{\partial \mathbf{z}} \approx 0$$

Reason for Mode Collapse

1. When we restart the training in discriminator, the most effective way to detect generated images is to detect this single mode.
2. Since generator desensitizes the impact of G already, the gradient from discriminator will likely push the single point around for the next most vulnerable mode.
3. This is not hard to find, because the generator produces such an imbalance of modes in training that it deteriorates its capability to detect others.
4. Now, both networks are overfitted to exploit the short- term opponent weakness.
5. This turns into to a cat-and-mouse game and the model will not converge.

Nash Equilibrium

1. GAN is a zero-sum non-cooperative game.
    - If one wins the other loses.
    - Zero-sum game is also called minimax.
    - Our opponent wants to maximize its actions and our actions are to minimize them.
2. In game theory, GAN converges when discriminator and generator reach a **Nash equilibrium**.
3. This is the **optimal** point for the minimax equation.

$$\min_G \max_D V(D,G) = \min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{d}}(\mathbf{x})}[\log D(x)] + \mathbb{E}_{\mathbf{z}\sim p_{\mathrm{x}}(g)}[\log(1 - D(\mathbf{x}))]$$

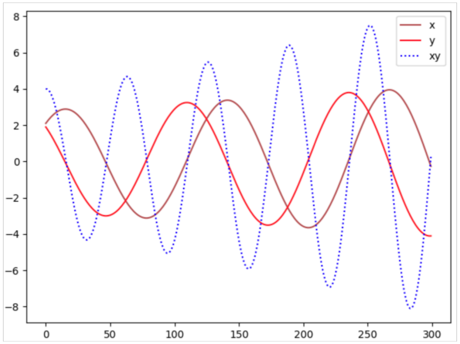
4. Nash equilibrium is when one player will not change action irrespective of opponent action.

Nash Equilibrium

1. Consider two player A and B which control the values x and y, respectively. A to maximize xy while B wants to minimize it.

$$\min_B \max_A V(D, G) = xy$$

2. Nash equilibrium is $x = y = 0$. (**Show it.**)
3. We update x and y based on gradients of V.



4. These plots show x, y, xy against training iterations.
5. It is apparent that solution does not converge

Variants of GAN

Variants of GAN

There are many papers related to GANs (Gui et al. 2023a). Some examples are:

1. InfoGAN (Chen et al. 2016)
2. Information Bottleneck GAN (Jeon et al. 2021)
3. Conditional GANs (Mirza and Osindero 2014)
4. BiGAN (Donahue, Krähenbühl, and Darrell 2017)
5. CycleGAN (Zhu, Park, et al. 2017)
6. f-GAN (Nowozin, Cseke, and Tomioka 2016)
7. Wasserstein GAN (Arjovsky, Chintala, and Léon Bottou 2017).

Variants of GAN

InfoGAN

InfoGAN

1. Rather than utilizing a single unstructured noise vector $\mathbf{z}$, the input noise vector was decomposed into two parts (Chen et al. 2016):
   - $\mathbf{z}$ which is called the incompressible noise, and
   - $\mathbf{c}$ which is called the latent code and will target the significant structured semantic features of the real data distribution.
2. InfoGAN aims to solve

$$\min_G \max_D V_I(D, G) = V(D, G) - \lambda MI(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$$

   where
   - $V(D, G)$ is the objective function of original GAN,
   - $G(\mathbf{z}, \mathbf{c})$ is the generated sample,
   - $MI(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$ is the mutual information, and
   - λ is the tunable regularization parameter.
3. Maximizing $MI(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$ means maximizing the mutual information between $\mathbf{c}$ and$G(\mathbf{z}, \mathbf{c})$ to make $\mathbf{c}$ contain as much important and meaningful features of the real samples as possible.

InfoGAN

1. We need the posterior $p(\mathbf{c} \mid \mathbf{x})$ to optimize $MI(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$ directly.
2. Hence, it is very difficult to optimize $MI(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$ directly in practice.
3. We want $p(\mathbf{c} \mid \mathbf{x})$ to have a **small entropy**.

   Meaning:

   The information in latent code c **should not be lost in the generation process**.
4. Instead, we can have a lower bound of $MI(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$.
5. It is obtained by defining an auxiliary distribution $q(\mathbf{c} \mid \mathbf{x})$ to approximate $p(\mathbf{c} \mid \mathbf{x})$.
6. The new objective function of InfoGAN is

$$\min_{G} \max_{D} V_I(D, G) = V(D, G) - \lambda L_I(\mathbf{c}, q)$$

InfoGAN

1. The new objective function of InfoGAN is

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda L_I(\mathbf{c}, q)$$

2. $L_I(\mathbf{c}, q)$ is the lower bound of $MI(\mathbf{c}, G(\mathbf{z},\mathbf{c}))$ and is

$$\begin{aligned} MI(\mathbf{c}, G(\mathbf{z},\mathbf{c})) &= H(\mathbf{c}) - H(\mathbf{c} \mid G(\mathbf{z},\mathbf{c})) \\ &= \mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})}\left[\mathbb{E}_{\mathbf{c}'\sim p(\mathbf{c}\mid\mathbf{x})}[\log p(\mathbf{c}' \mid \mathbf{x})]\right] + H(\mathbf{c}) \\ &= \mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})}\left[\underbrace{\mathrm{D}_{KL}(p(.\mid\mathbf{x}) \,||\, q(.\mid\mathbf{x}))}_{\geq 0} + \mathbb{E}_{\mathbf{c}'\sim p(\mathbf{c}\mid\mathbf{x})}[\log q(\mathbf{c}' \mid \mathbf{x})]\right] + H(\mathbf{c}) \\ &\geq \mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})}\left[\mathbb{E}_{\mathbf{c}'\sim p(\mathbf{c}\mid\mathbf{x})}[\log q(\mathbf{c}' \mid \mathbf{x})]\right] + H(\mathbf{c}) \end{aligned}$$

3. This technique of lower bounding mutual information is known as Variational Information Maximization.

InfoGAN

Lemma

For random variables X, Y *and function* $f(\mathbf{x}, \mathbf{y})$ *under suitable regularity conditions, we have*

$$\mathbb{E}_{\mathbf{x}\sim \mathrm{X}, \mathbf{y}\sim \mathrm{Y}}[f(\mathbf{x},\mathbf{y})] = \mathbb{E}_{\mathbf{x}\sim \mathrm{X}, \mathbf{y}\sim \mathrm{Y} \mid \mathbf{x}, \mathbf{x}'\sim \mathrm{X} \mid y}[f(\mathbf{x}',\mathbf{y})]$$

Proof.

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}\sim \mathrm{X}, \mathbf{y}\sim \mathrm{Y} \mid \mathbf{x}}[f(\mathbf{x},\mathbf{y})] &= \int_{\mathbf{x}} p(\mathbf{x}) \int_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}) f(\mathbf{x},\mathbf{y}) d\mathbf{x} d\mathbf{y} \\
&= \int_{\mathbf{x}} \int_{\mathbf{y}} p(\mathbf{x},\mathbf{y}) f(\mathbf{x},\mathbf{y}) d\mathbf{x} d\mathbf{y} \\
&= \int_{\mathbf{x}} \int_{\mathbf{y}} p(\mathbf{x},\mathbf{y}) f(\mathbf{x},\mathbf{y}) \int_{\mathbf{x}'} p(\mathbf{x}' \mid \mathbf{y}) d\mathbf{x}' d\mathbf{x} d\mathbf{y} \\
&= \int_{\mathbf{x}} p(\mathbf{x}) \int_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}) \int_{\mathbf{x}'} p(\mathbf{x}' \mid \mathbf{y}) f(\mathbf{x}',\mathbf{y}) d\mathbf{x}' d\mathbf{x} d\mathbf{y} \\
&= \mathbb{E}_{\mathbf{x}\sim \mathrm{X}, \mathbf{y}\sim \mathrm{Y} \mid \mathbf{x}, \mathbf{x}'\sim \mathrm{X} \mid y}[f(\mathbf{x}',\mathbf{y})]
\end{aligned}$$

□

InfoGAN

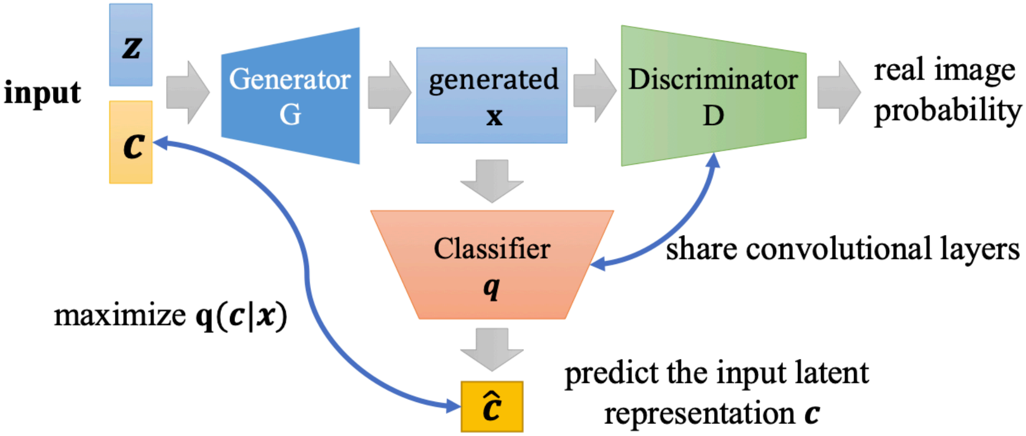
1. By using the above lemma, we can define $L_I(G, q)$ of $MI(\mathbf{c}, G(\mathbf{z}, \mathbf{c}))$ as

$$\begin{aligned} L_I(G, q) &= \mathbb{E}_{\mathbf{c}\sim p(\mathbf{c}), \mathbf{x}\sim G(\mathbf{z},\mathbf{c})}[\log q(\mathbf{c} \mid \mathbf{x})] + H(\mathbf{c}) \\ &= \mathbb{E}_{\mathbf{x}\sim G(\mathbf{z},\mathbf{c})}\left[\mathbb{E}_{\mathbf{c}'\sim p(\mathbf{c} \mid \mathbf{x})}[\log q(\mathbf{c} \mid \mathbf{x})]\right] + H(\mathbf{c}) \\ &\leq MI(\mathbf{c}, G(\mathbf{z}, \mathbf{c})) \end{aligned}$$

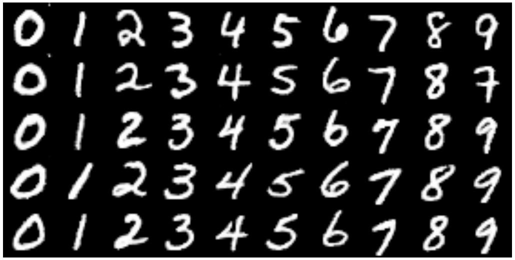
2. The entropy of latent codes $H(\mathbf{c})$ can be optimized, since it has a simple analytical form for common distributions.
3. In InfoGAN, latent code is fixed and $H(\mathbf{c})$ is treated as a constant.
4. $L_I(G, q)$ is easy to approximate with Monte Carlo simulation.
5. $L_I(G, q)$ can be added to GAN's objectives with no change to GAN's training procedure.
6. InfoGAN is defined as the following minimax game (λ is a hyperparameter) :

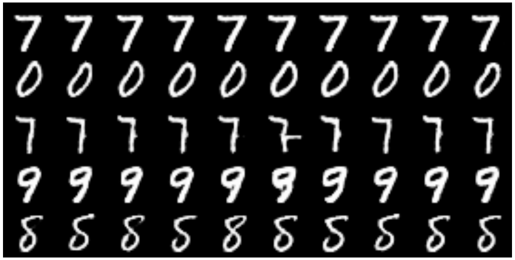
$$\min_{G,q} \max_{D} V_I(D, G, q) = V(D, G) - \lambda L_I(\mathbf{c}, q)$$

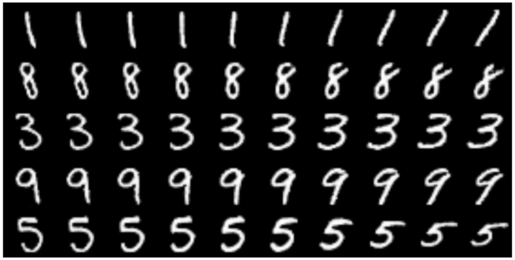
InfoGAN

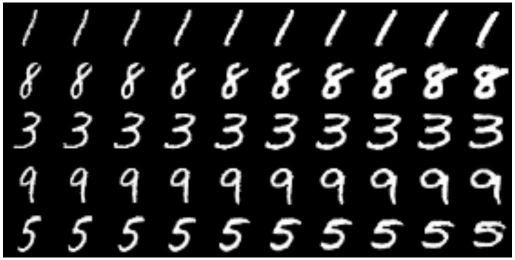

input
z
c
Generator G
generated $\mathbf{x}$
Discriminator D
real image probability
Classifier q
share convolutional layers
maximize $\mathbf{q}(c|x)$
$\hat{c}$
predict the input latent representation c

InfoGAN


(a) Varying c_1 on InfoGAN (Digit type)


(b) Varying c_1 on regular GAN (No clear meaning)


(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)


(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

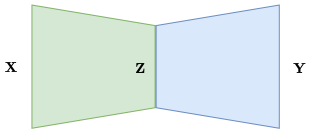
The performance of InfoGAN for disentanglement is constantly reported to be lower than VAE-based models.

Variants of GAN

Information Bottleneck Generative Adversarial Networks

Information Bottleneck Principle

1. Let input variable $\mathbf{x}$ and target variable $\mathbf{x}$ distributed according to some joint data distribution $p(\mathbf{x}, \mathbf{y})$.
2. The goal of information bottleneck (IB) is to obtain a compressive representation $\mathbf{z}$ from the input $\mathbf{x}$, while maintaining the predictive information about the target $\mathbf{y}$ as much as possible (Tishby, Pereira, and Bialek 1999; Tishby and Zaslavsky 2015).



3. The objective for the IB is $\max_{q_\phi(\mathbf{z} \mid \mathbf{x})} \{MI(\mathbf{z}, \mathbf{y}) - \beta MI(\mathbf{z}, \mathbf{x})\}$.
4. IB aims at obtaining the optimal representation encoder $q_\phi(\mathbf{z} \mid \mathbf{x})$ that simultaneously balances the trade-off between maximization and minimization of both MI.
5. The learned representation $\mathbf{z}$ can act as a minimal sufficient statistic of $\mathbf{x}$ for predicting $\mathbf{y}$.
6. The IB principle provides an intuitive meaning for good representation from perspective of information theory.

Information Bottleneck GAN (IB-GAN)

1. InfoGAN's objective lacks a MI minimization term compare to the IB objective.
2. Hence, MI minimization term adopted to InfoGAN's objective to get the IB-GAN objective (Jeon et al. 2021):

$$\begin{aligned} min_G\, max_D \mathcal{L}_{IB-GAN}(D,G) &= V(D,G) - \left[MI^L(\mathbf{z}, G(\mathbf{z})) - \beta MI^U(\mathbf{z}, G(\mathbf{z}))\right] \\ s.t. \quad & MI^L(\mathbf{z}, G(\mathbf{z})) \leq MI_g(\mathbf{z}, G(\mathbf{z})) \leq MI^U(\mathbf{z}, G(\mathbf{z})) \end{aligned}$$

where MI^L / MI^U are lower / upper bound of generative MI ($MI_g(\mathbf{z}, \mathbf{x})$):

$$MI_g(\mathbf{z}, \mathbf{x}) = \mathbb{E}_{p_\theta(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})}\left[\log \frac{p_\theta(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})}{p_\theta(\mathbf{x})\, p(\mathbf{z})}\right]$$

3. For the optimization, first a tractable lower-bound of the generative MI is defined as

$$\begin{aligned} MI_g(\mathbf{z}, G(\mathbf{z})) &= \mathbb{E}_{p_\theta(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})}\left[\log \frac{p_\theta(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})}{p_\theta(\mathbf{x})\, p(\mathbf{z})}\right] \geq MI^L(\mathbf{z}, G(\mathbf{z})) \\ &= \mathbb{E}_{p_\theta(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})}\left[\log \frac{q_\phi(\mathbf{z} \mid \mathbf{x})}{p(\mathbf{z})}\right] \\ &= \mathbb{E}_{p_\theta(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})}[\log\, q_\phi(\mathbf{z} \mid \mathbf{x})] + H(\mathbf{z}) \end{aligned}$$

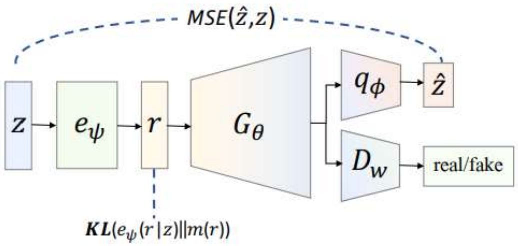
4. A variational re-constructor $q_\phi(\mathbf{z} \mid \mathbf{x})$ approximates $p_\theta(\mathbf{z} \mid \mathbf{x}) = \frac{p_\theta(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})}{p_\theta(\mathbf{x})}$.

Information Bottleneck GAN (IB-GAN)

1. The same approach was used to derive the upper bound (Jeon et al. 2021).
2. Then, a variational approximation of objective function can be obtained as

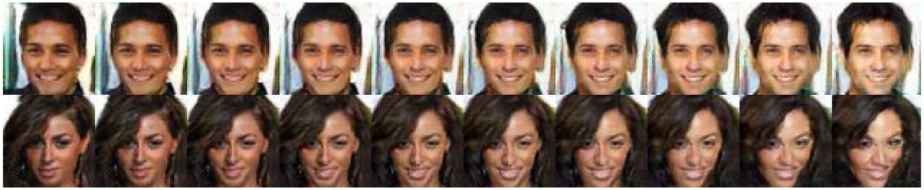
$$
\begin{aligned}
min_{G,\, q_\phi(\mathbf{x}),\, e_\psi(\mathbf{x})} max_D \widehat{\mathcal{L}}_{IB-GAN}(D, G,\, q_\phi(\mathbf{x}),\, e_\phi(\mathbf{x})) = {} & V(D, G) \\
& - \mathbb{E}_{p(\mathbf{z})}\left[\mathbb{E}_{p_\theta(\mathbf{x}\,|\,\mathbf{z})\, e_\psi(\mathbf{r})}[\log\, q_\phi(\mathbf{z} \mid \mathbf{x})]\right] \\
& + \beta\, \mathrm{D}_{KL}(\, e_\psi(\mathbf{r} \mid \mathbf{z}) \,||\, m(\mathbf{r}))
\end{aligned}
$$

where $m(\mathbf{r}) = \mathcal{N}(0, \mathbf{I})$ is a prior and $e_\psi(\mathbf{r} \mid \mathbf{z}) = \mathcal{N}(\mu_\psi(\mathbf{z}), \sigma_\psi(\mathbf{z}))$ is a encoder as in VAE.

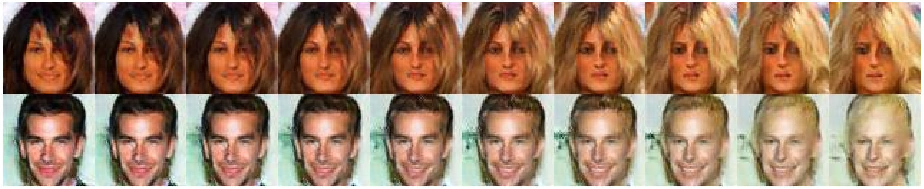


Information Bottleneck GAN Results

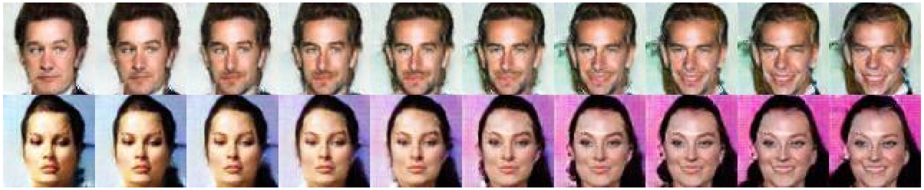
Azimuth



Hair Color

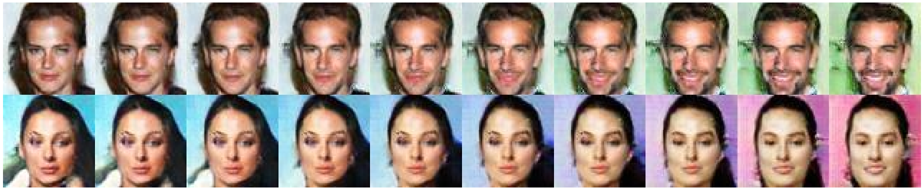


Smile

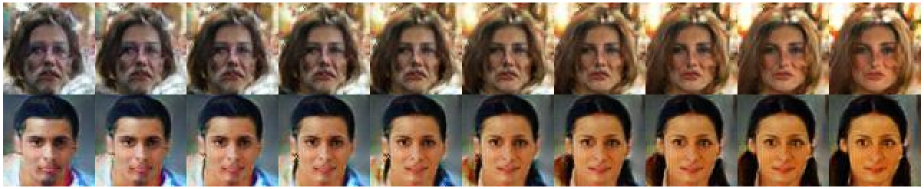


Information Bottleneck GAN Results

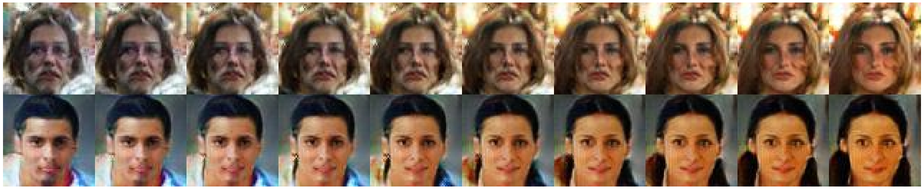
Background



Skintone



Gender



Variants of GAN

f-GAN

f-Divergence

1. KL divergence is a way to measure the discrepancy between two probability distributions.

$$D_{KL}(\, p(\mathbf{x}) \,||\, q(\mathbf{x})) = \int_{\mathbf{x}} p(\mathbf{x}) \log\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right) d\mathbf{x}$$

2. Jensen-Shannon distance is another way to measure the discrepancy between two probability distributions.

$$D_{JS}(\, p(\mathbf{x}) \,||\, q(\mathbf{x})) = \frac{1}{2} D_{KL}\left(p(\mathbf{x}) \,||\, \frac{p(\mathbf{x}) + q(\mathbf{x})}{2}\right) + \frac{1}{2} D_{KL}\left(q(\mathbf{x}) \,||\, \frac{p(\mathbf{x}) + q(\mathbf{x})}{2}\right)$$

3. Both $D_{KL}(\, p(\mathbf{x}) \,||\, q(\mathbf{x}))$ and $D_{JS}(\, p(\mathbf{x}) \,||\, q(\mathbf{x}))$ are special cases of the more general **f-divergence** (Ali and Silvey 2018).

Definition (f-divergence)

Given a convex function f[a] and two densities $p(\mathbf{x})$ and $q(\mathbf{x})$, then f-divergence defined as

$$D_f(\, p(\mathbf{x}) \,||\, q(\mathbf{x})) = \int_{-\infty}^{\infty} q(\mathbf{x}) f\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right) d\mathbf{x}$$

[a]There are more constraints on function f, such as f must be a convex, lower-semicontinuous function with $f(1) = 0$.

f-Divergence

1. Some f-divergences are:

KL-divergence	$f(t) = t \log t$
Hellinger distance	$f(t) = \left(\sqrt{t} - 1\right)^2$
Total variation distance	$f(t) = \frac{1}{2}\lvert t - 1\rvert$
Pearson χ^2-divergence,	$f(t) = (t-1)^2$

2. **Homework**
   - Show that for any two probability distributions $p(\mathbf{x})$ and $q(\mathbf{x})$, we have $D_{JS}(p(\mathbf{x}) \,||\, q(\mathbf{x})) \geq 0$.
   - Show that for any two probability distributions $p(\mathbf{x})$ and $q(\mathbf{x})$, we have $D_{JS}(p(\mathbf{x}) \,||\, q(\mathbf{x}))$ is symmetric.
   - Show that for any two probability distributions $p(\mathbf{x})$ and $q(\mathbf{x})$, we have $D_{JS}(p(\mathbf{x}) \,||\, q(\mathbf{x}))$ is not a metric.
   - Show that for any two probability distributions $p(\mathbf{x})$ and $q(\mathbf{x})$ and any convex function f, we have $D_f(p(\mathbf{x}) \,||\, q(\mathbf{x})) \geq 0$.
3. For more about mathematical background of GAN, please read (Wang 2020).

f-GAN

1. How to minimize $D_{KL}(\, p_d(\mathbf{x}) \,||\, p_g(\mathbf{x}))$?

$$\begin{aligned} D_{KL}(\, p_d(\mathbf{x}) \,||\, p_g(\mathbf{x})) &= \int_{\mathbf{x}} p_d(\mathbf{x}) \log \frac{p_d(\mathbf{x})}{p_g(\mathbf{x})} d\mathbf{x} \\ &= \int_{\mathbf{x}} p_d(\mathbf{x}) \log\, p_d(\mathbf{x}) d\mathbf{x} - \int_{\mathbf{x}} p_d(\mathbf{x}) \log\, p_g(\mathbf{x}) d\mathbf{x} \\ &= C - \int_{\mathbf{x}} p_d(\mathbf{x}) \log\, p_g(\mathbf{x}) d\mathbf{x} \\ &= C - \mathbb{E}_{\mathbf{x} \sim\, p_d(\mathbf{x})}[\log\, p_g(\mathbf{x})] \end{aligned}$$

2. Can we use the same trick to minimize $D_{JS}(\, p_d(\mathbf{x}) \,||\, p_g(\mathbf{x}))$?

$$D_{JS}(\, p(\mathbf{x}) \,||\, q(\mathbf{x})) = \frac{1}{2} D_{KL}\left(p(\mathbf{x}) \,||\, \frac{p(\mathbf{x}) + q(\mathbf{x})}{2} \right) + \frac{1}{2} D_{KL}\left(q(\mathbf{x}) \,||\, \frac{p(\mathbf{x}) + q(\mathbf{x})}{2} \right)$$

3. We cannot do the same trick, because requires knowledge of mixture, which is unknown!

f-GAN

1. Every convex, lower-semicontinuous function f has a convex conjugate function f^*, known as **Fenchel conjugate**.
2. This function is defined as $f^*(t) = \sup_{u \in dom(f)}\{ut - f(u)\}$
3. The function f^* is again convex and lower-semicontinuous and the pair (f, f^*) is **dual** to another in the sense that $f^{**} = f$.
4. Specifically, we obtain a lower bound to any f-divergence via its Fenchel conjugate:

$$\mathrm{D}_f(\,p_{\mathrm{d}}(\mathbf{x}) \,||\, p_g(\mathbf{x})) \geq \sup_{T \in \mathcal{T}} \left(\mathbb{E}_{\mathbf{x} \sim p_{\mathrm{d}}(\mathbf{x})}[T(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim p_g(\mathbf{x})}[f^*(T(\mathbf{x}))] \right)$$

   $\mathcal{T}$ is an arbitrary class of functions $T : \mathcal{X} \mapsto \mathbb{R}$.

5. Therefore we can choose any f-divergence that we desire.
6. Let parameterize T by ϕ and G by θ and obtain the following f-GAN objective:

$$min_\theta \, max_\phi F(\theta, \phi) = \mathbb{E}_{\mathbf{x} \sim p_{\mathrm{d}}(\mathbf{x})}[T_\phi(\mathbf{x})] - \mathbb{E}_{\mathbf{x} \sim p_g(\mathbf{x})}[f^*(T_\phi(\mathbf{x}))]$$

7. Intuitively, we can think about this objective as the generator trying to minimize the divergence estimate, while the discriminator tries to tighten the lower bound.

f-GAN

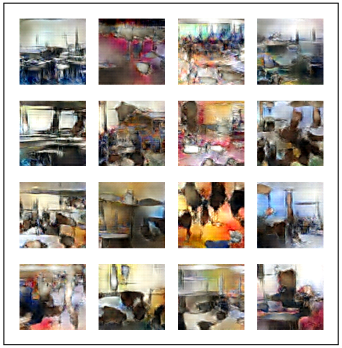
1. Similar to a GAN, we use two neural networks:
   - G is the generative model and
   - T is the variational function.
2. The process is Similar to GAN
   - G takes a random vector and outputs a sample and
   - T takes a sample and returns a scalar.
3. The objective function is

$$min_\theta max_\phi F(\theta, \phi) = \mathbb{E}_{\mathbf{x}\sim p_{\text{d}}(\mathbf{x})}[T_\phi(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim p_g(\mathbf{x})}[f^*(T_\phi(\mathbf{x}))]$$

4. The expected values are approximated through minibatches.
   - The first term is sampled from the training set without replacement.
   - The second term is taken from current $G(\mathbf{z})$.

f-GAN Results

(a) GAN


(b) KL

(c) Squared Hellinger

Variants of GAN

Wasserstein GAN

Wasserstein GAN (WGAN)

1. Wasserstein Distance is a measure of the distance between two probability distributions.
2. When dealing with the continuous probability domain, the distance becomes

$$D_{WD}(\, p_{\mathrm{d}}(\mathbf{x}) \,||\, p_g(\mathbf{x})) = \inf_{\gamma \sim \Pi(\, p_{\mathrm{d}}(\mathbf{x}),\, p_g(\mathbf{x}))} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma}[\|\mathbf{x}-\mathbf{y}\|]$$

   where $\Pi(\, p_{\mathrm{d}}(\mathbf{x}),\, p_g(\mathbf{x}))$ is the set of all possible joint probability distributions of $p_{\mathrm{d}}(\mathbf{x})$ and $p_g(\mathbf{x})$.
3. It is intractable to exhaust all the possible joint distributions in $\Pi(\, p_{\mathrm{d}}(\mathbf{x}),\, p_g(\mathbf{x}))$ to compute $\inf_{\gamma \sim \Pi(\, p_{\mathrm{d}}(\mathbf{x}),\, p_g(\mathbf{x}))} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma}[\|\mathbf{x}-\mathbf{y}\|]$, the following metric is used.

$$D_{WD}(\, p_{\mathrm{d}}(\mathbf{x}) \,||\, p_g(\mathbf{x})) = \frac{1}{K} \sup_{\|f\|_L \le K} \mathbb{E}_{\mathbf{x}\sim\, p_{\mathrm{d}}(\mathbf{x})}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim\, p_g(\mathbf{x})}[f(\mathbf{x})]$$

   where $\|f\|_L \le K$ means that f is K-Lipschitz.
4. Why Wasserstein is better than JS or KL divergence?

   Even when two distributions are located in lower dimensional manifolds without overlaps, Wasserstein distance can still provide a meaningful and smooth representation of the distance in-between.

Wasserstein GAN (WGAN)

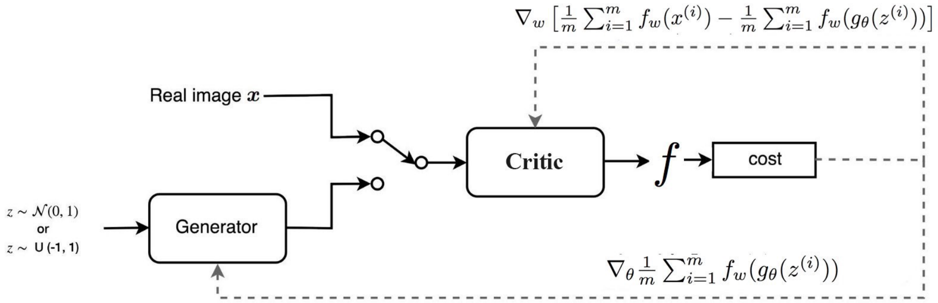
1. In WGAN, discriminator network instead of producing the probability of generating real data, the network produces a scaler score (Arjovsky and Leon Bottou 2017).
2. This score can be interpreted as how real the input images are.
3. In reinforcement learning, we call it the value function which measures how good a input is.
4. We rename the discriminator to critic to reflect its new role.
5. The loss function for WGAN is

$$\begin{aligned} V(p_{\text{d}}(\mathbf{x}), p_g(\mathbf{x})) &= \text{D}_{WD}(p_{\text{d}}(\mathbf{x}) \,||\, p_g(\mathbf{x})) \\ &= \max_{w \in W} \mathbb{E}_{\mathbf{x} \sim p_{\text{d}}(\mathbf{x})}[f_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[f_w(g_\theta(z))] \end{aligned}$$

   f comes from a family of K-Lipschitz continuous functions $\{f_w\}_{w \in W}$ parameterized by w.
6. The discriminator model is used for learning w to find a good f_w and the loss function is configured as measuring the Wasserstein distance between $p_{\text{d}}(\mathbf{x})$ and $p_{\text{d}}(\mathbf{x})$.

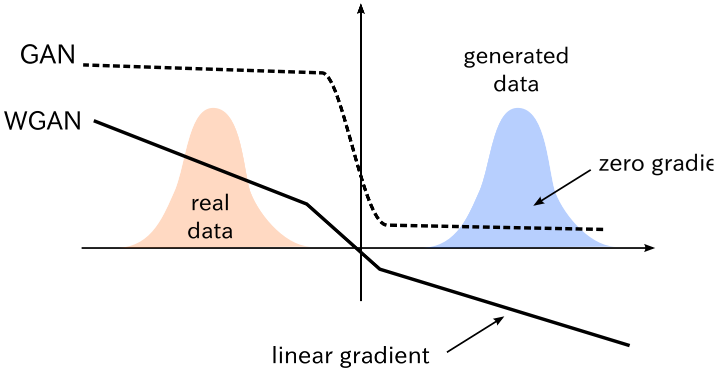
Wasserstein GAN (WGAN)

1. WGAN architecture is (Arjovsky, Chintala, and Léon Bottou 2017).

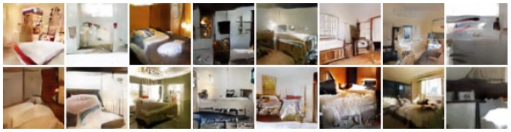

$\nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
Real image x
Critic
f
cost
$z \sim \mathcal{N}(0,1)$
or
$z \sim$ U (-1, 1)
Generator
$\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$

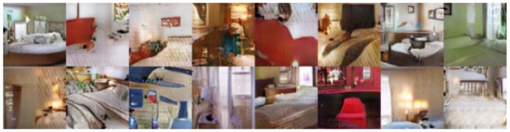
Wasserstein GAN (WGAN)

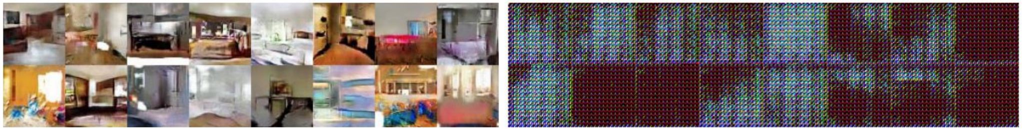
1. The Wasserstein formulation is empirically shown to avoid mode collapse.
2. The reason is that, in this formulation one can maximize the discriminator to optimality, before updating the generator.
3. In contrast to the JS formulation, the optimal discriminator does not introduce vanishing gradients.
4. The following toy example shows this concept.

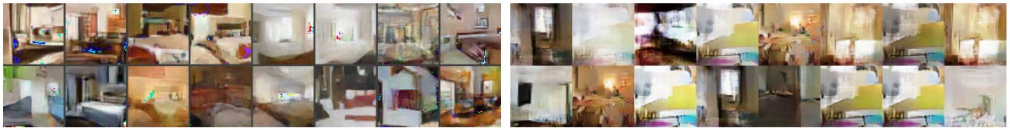


Wasserstein GAN Results


WGAN with DCGAN generator


GAN with DCGAN generator


Without batch normalization &
constant number of filters at each layer


Using a MLP as the generator

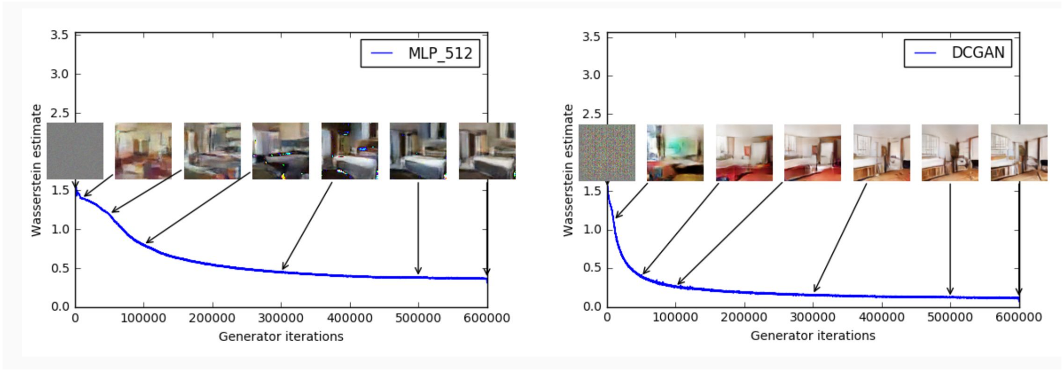
All critics and discriminators follow the same discriminator design in DCGAN

Wasserstein GAN Results

Training curves and samples at different stages of training.

Left: The generator is an MLP with 4 hidden layers and 512 units at each layer.

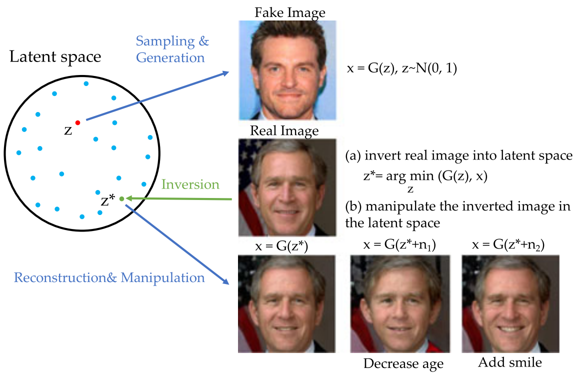
Right: The generator is a standard DCGAN



GAN Inversion

Introduction

1. Recent studies have shown that GANs effectively encode rich semantic information in intermediate features and latent spaces from the supervision of image generation.
2. These methods can synthesize images with a diverse range of attributes, such as faces with different ages and expressions, and scenes with different lighting conditions.
3. GAN inversion aims to invert a given image back into the latent space of a pretrained GAN model.
4. The image can then be faithfully reconstructed from the inverted code by the generator.



Applications of GAN Inversion

GAN inversion does not require task-specific dense-labeled datasets and can be applied to many tasks such as :

1. image manipulation,
2. image interpolation,
3. image restoration,
4. style transfer,
5. novel-view synthesis,
6. adversarial defense
7. 3D reconstruction
8. image understanding
9. multimodal learning
10. medical imaging

Image Manipulation

1. Given an image $\mathbf{x}$, we can vary its latent code $\mathbf{z}$.
2. Then we can obtain $\mathbf{z}'$ of the target image $\mathbf{x}'$ by linearly transforming the latent representation from a trained GAN model G.
3. This can be formulated in the framework of GAN inversion as the operation of adding a scaled difference vector:

$$\mathbf{x}' = G(\mathbf{z} + \alpha\mathbf{n})$$

   where
   - $\mathbf{n}$ is the normal direction corresponding to a particular semantic in the latent space
   - α is the step for manipulation.
4. If a latent code is moved in a certain direction, then the semantics contained in the output image should vary accordingly.

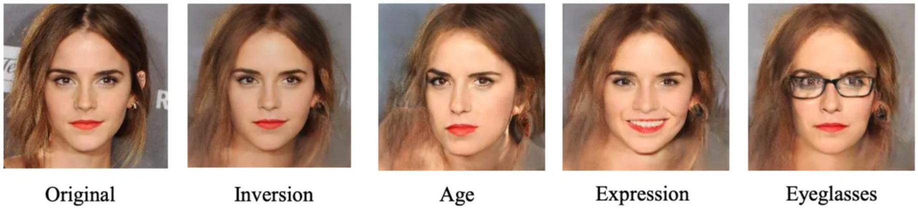
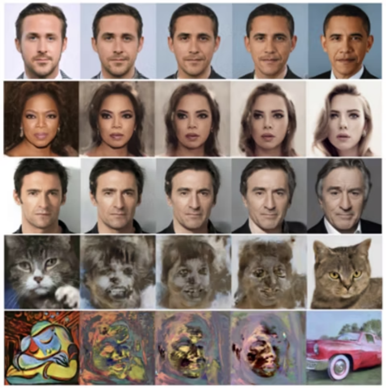


Image interpolation

1. Image interpolation refers to the process of morphing between two images by interpolating between their corresponding latent vectors in the latent space.
2. The basic strategy is to perform linear interpolation in the latent space to achieve this, and can be formulated as follows:

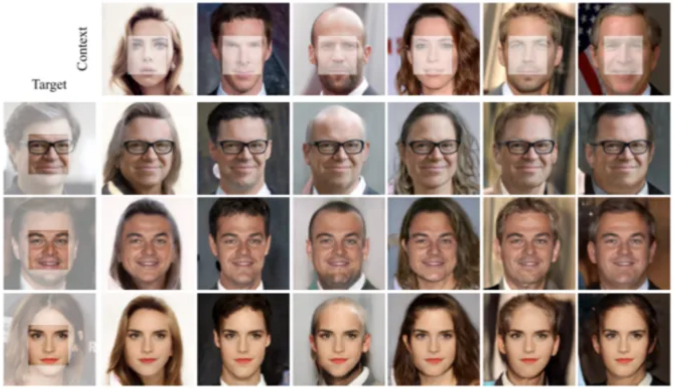
$$\mathbf{z} = \lambda \mathbf{z}_1 + (1 - \lambda)\mathbf{z}_2$$

where $\lambda \in (0, 1)$.



Semantic diffusion

1. This task aims to diffuse a specific part of a target image into a context image, expecting the output to retain the characteristics of the target image while conforming to the context image.



GAN Inversion

Definition (GAN Inversion)

1. The generator of an unconditional GAN learns the mapping $G : \mathcal{Z} \mapsto \mathcal{X}$.
2. When $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{Z}$ are close in $\mathcal{Z}$ space, the corresponding images $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$ are visually similar.
3. GAN inversion maps data $\mathbf{x}$ back to latent representation $\mathbf{z}^*$ or, equivalently, finds an image $\mathbf{x}^*$ that can be entirely synthesized by the well-trained generator G and remain close to the real image $\mathbf{x}$.
4. Formally, denoting the signal to be inverted as $\mathbf{x} \in \mathbb{R}^n$, the well-trained generator as $G : \mathbb{R}^{n_0} \mapsto \mathbb{R}^n$, and the latent vector as $\mathbf{z} \in \mathbb{R}^{n_0}$, we study the following inversion problem:

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \ell(G(\mathbf{z}), \mathbf{x}),$$

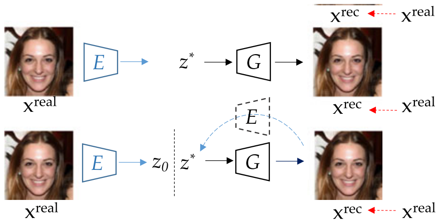
where ℓ is a distance metric in the image or feature space.

The obtained latent code for a given image should have two properties:

- Reconstructing the input image faithfully and photo-realistically (how to solve the above optimization problem).
- Facilitating downstream tasks (use of latent space).

GAN Inversion Methods

1. A **learning-based inversion method** aims to learn an encoder network to map an image into the latent space such that the reconstructed image based on the latent code looks as similar to the original one as possible.
2. An **optimization-based inversion approach** directly solves the objective function through back-propagation to find a latent code that minimizes pixel-wise reconstruction loss.
3. A **hybrid approach** first uses an encoder to generate initial latent code and then refines it with an optimization algorithm.



Generally,

- Learning-based GAN inversion methods cannot faithfully reconstruct the image content.
- Optimization-based techniques have achieved superior image reconstruction quality, their inevitable drawback is the significantly higher computational cost.

Learning-Based GAN Inversion

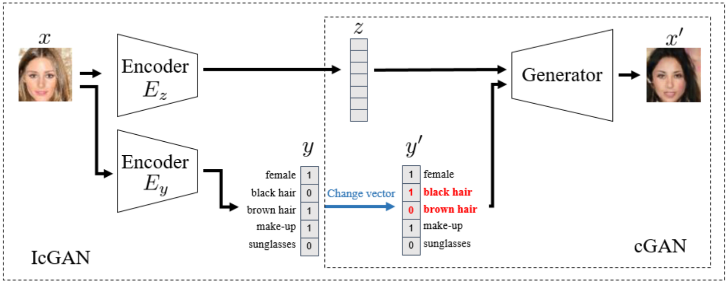
1. Learning-based GAN inversion typically involves training an encoding neural network $E_{\theta_E}(\mathbf{x})$ to map an image, $\mathbf{x}$, into the latent code $\mathbf{z}$ by

$$\theta_E^* = \arg\min_{\theta_E} \sum_n \mathcal{L}(G(E_{\theta_E}(\mathbf{x})), \mathbf{x})$$

2. A good encoder for GAN inversion should have the following properties:
   - have accurate reconstruction
   - lightweight
   - data-efficiency
   - supporting high-resolution images
   - generalizability to arbitrary images

Invertible cGAN

1. Invertible cGAN enables to re-generate real images with deterministic modifications (Perarnau et al. 2016).



2. The encoder E is composed of two sub-encoders:
   - E_z, which encodes an image to latent representation $\mathbf{z}$,
   - E_y , which encodes an image to conditional information $\mathbf{y}$.
3. To train E_z, create dataset $(\mathbf{x}', (\mathbf{z}, \mathbf{y}'))$ and minimize

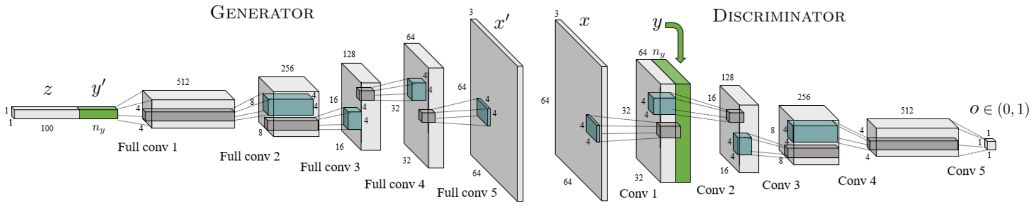
$$\mathcal{L}_{E_z} = \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z}), \mathbf{y}'\sim p_y(\mathbf{y})}\left[\|\mathbf{z} - E_z(G(\mathbf{z}, \mathbf{y}'))\|^2\right]$$

.

4. To train E_y, create dataset $(\mathbf{x}, \mathbf{y})$ and minimize $\mathcal{L}_{E_y} = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim p_d(\mathbf{x},\mathbf{y})}\left[\|\mathbf{y} - E_y(\mathbf{x})\|^2\right]$.

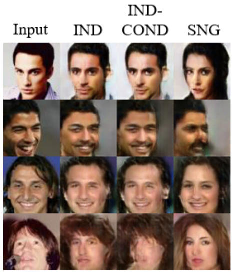
Invertible cGAN

1. For encoders, several strategies can be used:
   - **SNG:** One single encoder with shared layers and two outputs.
   - **IND:** Two independent encoders and are trained separately.
   - **IND-COND:** Two encoders, where E_z is conditioned on the output of encoder E_y.
2. Architecture of the generator and discriminator of the used cGAN model.

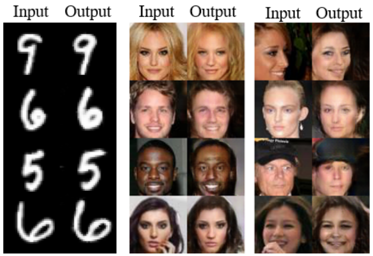


Invertible cGAN Results

1. Some IcGAN results:
   - (a) Comparison of different encoder configurations, where IND yields the most faithful reconstructions.
   - (b) Reconstructed samples from MNIST and CelebA using IND configuration.



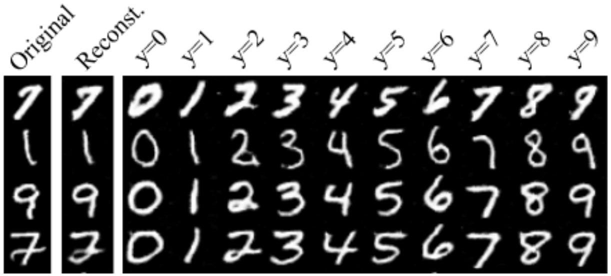
(a)

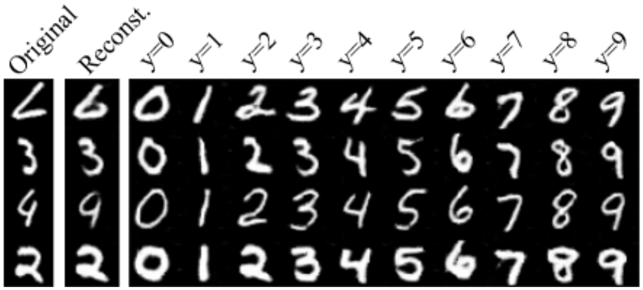


(b)

Invertible cGAN Results (MNIST dataset)

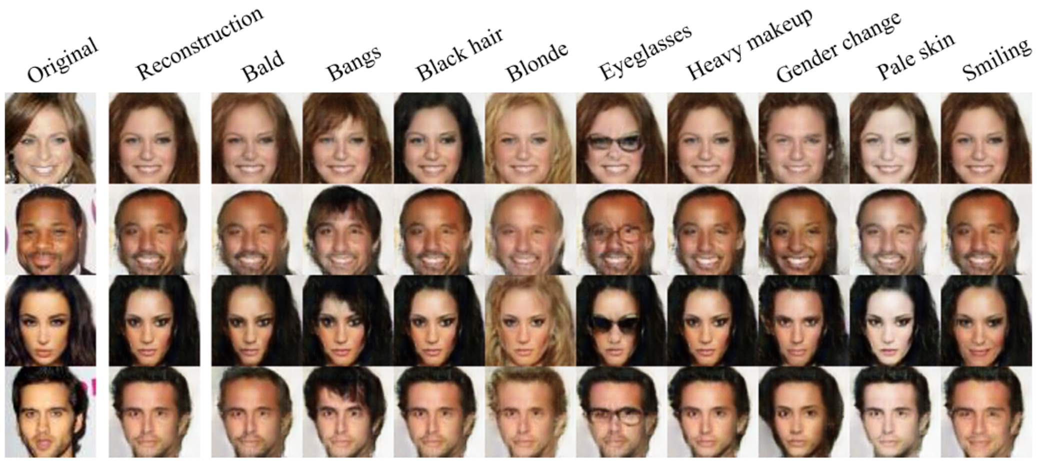
A real image is encoded into a latent representation **z** and conditional information **y**, and then decoded into a new image. **z** is fixed for every row and **y** is modified for each column to obtain variations.





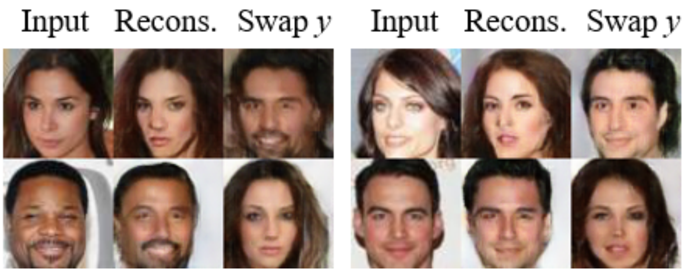
Invertible cGAN Results (CelebA dataset)

A real image is encoded into a latent representation **z** and conditional information **y**, and then decoded into a new image. **z** is fixed for every row and **y** is modified for each column to obtain variations.

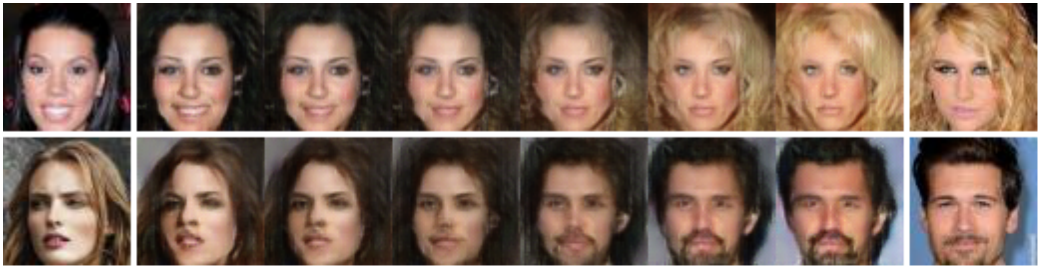


Invertible cGAN Results

1. **Swap attributes**



2. **Interpolate between faces**



Optimization-Based GAN Inversion

1. Existing optimization-based GAN inversion methods typically reconstruct a target image by optimizing the latent vector

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \ell(\mathbf{x}, G_\theta(\mathbf{z}))$$

   where $\mathbf{x}$ is the target image and G_θ is a GAN generator parameterized by θ.
2. It is critical to
   - choose the optimizer since a good optimizer helps alleviate the local minima problem and
   - the initialization of latent code, because the objective function is not convex.
3. The optimization-based methods typically require an expensive iterative process in terms of both memory and runtime, as they have to be applied to each latent code independently.

Image2StyleGAN

1. These models need to measure the similarity between an input image and the embedded image during optimization.
2. Image2StyleGAN employed a loss function that is a weighted combination of the VGG-16 perceptual loss (Abdal, Qin, and Wonka 2019).

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \ell_{percept}(\mathbf{x}, G_\theta(\mathbf{z})) + \frac{\lambda_{mse}}{N}\|\mathbf{x} - G_\theta(\mathbf{z})\|^2$$

$$\ell_{percept}(\mathbf{x}, G_\theta(\mathbf{z})) = \sum_{j=1}^{4} \frac{\lambda_j}{N_j}\|F_j(\mathbf{x}_1) - F_j(\mathbf{x}_2)\|^2$$

where $\mathbf{x} \in \mathbb{R}^{n\times n\times 3}$ is the target image and $N = n \times n \times 3$ is the number of scalers in the image, and F_j is the feature output of VGG-16 layers (conv(1,1), conv (1, 2), conv(3, 2) and conv(4, 2)).

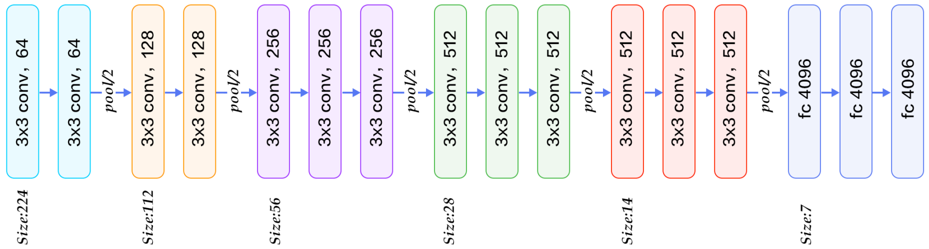


Image2StyleGAN Results

1. First column: original image (1024×1024).
2. Second column: embedded image with the perceptual loss applied to resized images of 256×256 resolution.
3. Third column: embedded image with the perceptual loss applied to the images at the original 1024×1024 resolution.

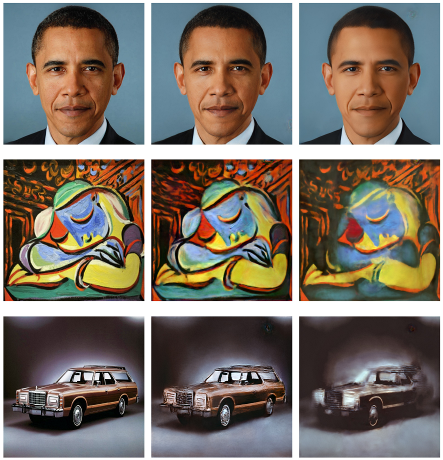


Image2StyleGAN Results

1. The results using different initialization of network weights.

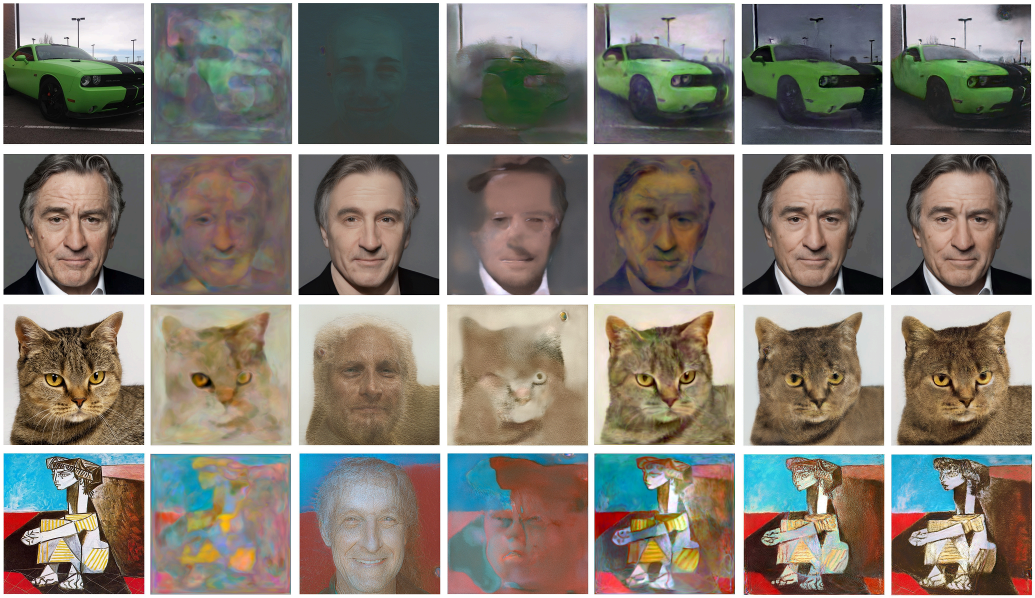
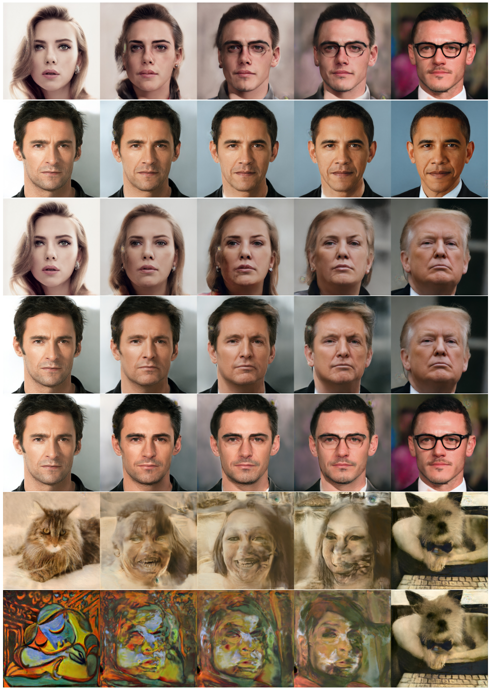


Image2StyleGAN Results

1. Additional morphing results between two embedded images (left-most and right-most).

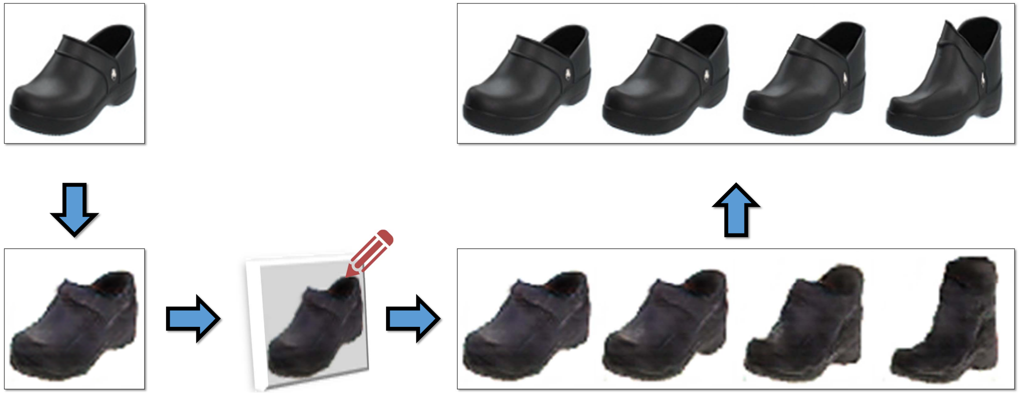


Hybrid GAN Inversion

1. The hybrid methods exploit the advantages of both approaches discussed above.
2. Zhu et al. proposed a framework that
   - First, predicts $\mathbf{z}$ of a given real photo $\mathbf{x}$ by training a separate encoder $E_{\theta_E}(\mathbf{x})$ (Zhu, Krähenbühl, et al. 2016).
   - Then they used the obtained $\mathbf{x}$ as the initialization for optimization.
3. The learned predictive model serves as a fast bottom-up initialization for the nonconvex optimization problem.

Hybrid GAN Inversion

1. Zhu et al. used the following framework (Zhu, Krähenbühl, et al. 2016).
   - At first project an original photo onto a low-dimensional latent vector representation.
   - Then regenerating it using GAN.
   - Next, modify the color and shape of the generated image using various brush tools.
   - Finally, apply the same amount of geometric and color changes to the original photo to achieve the final result.

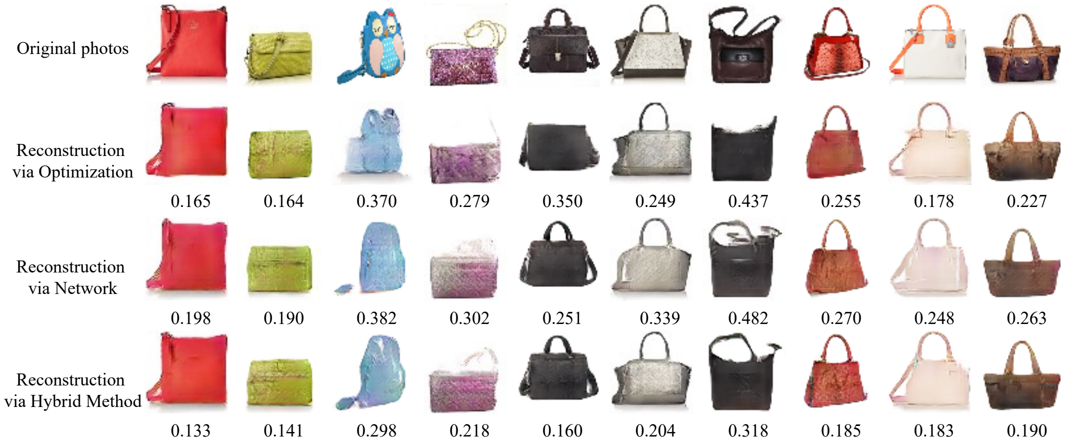


Hybrid GAN Inversion Results

- **Left column:** Randomly generated examples from a GAN, trained on the shirts dataset.
- **Mid column:** Random jittering: each row shows a random sample from a GAN (the first one at the left), and its variants produced by adding Gaussian noise to **z** in the latent space.
- **Right column:** Interpolation: each row shows two randomly generated images (first and last), and their smooth interpolations in the latent space.

Hybrid GAN Inversion Results

1. Top row: original photos (from handbag dataset);
2. 2nd row: reconstruction using optimization-based method;
3. 3rd row: reconstruction via learned deep encoder;
4. bottom row: reconstruction using the hybrid method.



Original photos										
Reconstruction via Optimization	0.165	0.164	0.370	0.279	0.350	0.249	0.437	0.255	0.178	0.227
Reconstruction via Network	0.198	0.190	0.382	0.302	0.251	0.339	0.482	0.270	0.248	0.263
Reconstruction via Hybrid Method	0.133	0.141	0.298	0.218	0.160	0.204	0.318	0.185	0.183	0.190

The number below each image is the reconstruction loss.

Properties of GAN Inversion Methods

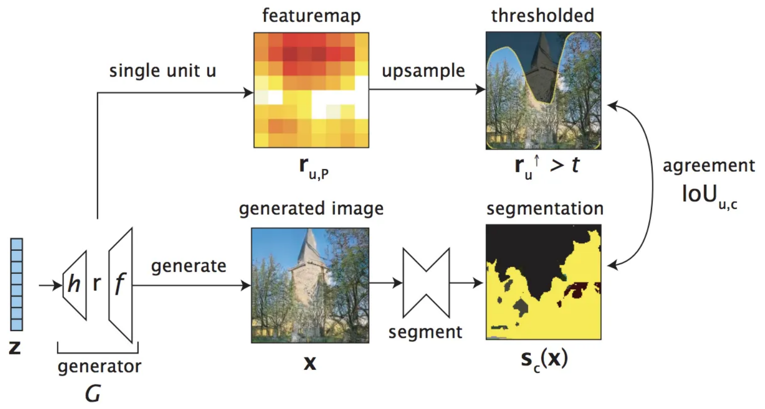
Properties of GAN Inversion methods (Xia et al. 2023).

Method	Publication	Type	S.-A.	L.-W.	S.-R	Space	GAN Model	Dataset	Keywords
Zhu *et al.* [20]	ECCV'16	H.			64	$\mathcal{Z}$	[38]	[39], [40], [41]	inversion for GANs
Creswell *et al.* [42], [43]	NeurIPS'16	O.			128	$\mathcal{Z}$	[38], [44]	[39], [45]	first using the term *inversion*
Perarnau *et al.* [46]	NeurIPS'16	L.			64	$\mathcal{Z}$	[38]	[45], [47]	inversion for conditional GAN
GANPaint [15]	TOG'19	H.	✓	✓	256	$\mathcal{Z}$	[11]	[41]	learn an image-specific generator
GANSeeing [23]	ICCV'19	H.	✓	✓	256	$\mathcal{Z}, \mathcal{W}$	[11], [13], [44]	[41]	visualization of mode collapse
Image2StyleGAN [21]	ICCV'19	O.	✓		1024	$\mathcal{W}$	[13]	[13]	first inversion for StyleGAN
Image2StyleGAN++ [22]	CVPR'20	O.	✓	✓	1024	$\mathcal{W}^+$	[11], [13]	[11], [13]	
mGANPrior [48]	CVPR'20	O.	✓	✓	256	$\mathcal{Z}$	[11], [13]	[11], [13], [41]	multi-code GAN prior
Editing in Style [49]	CVPR'20	O.	✓		1024	$\mathcal{W}$	[11], [13], [14]	[13], [41]	
YLG [50]	CVPR'20	O.			128	$\mathcal{Z}$	[51]	[52]	attention
Huh *et al.* [24]	ECCV'20	O.	✓		1024	$\mathcal{Z}$	[12], [14]	[13], [41], [52]	class-conditional
IDInvert [19]	ECCV'20	H.	✓	✓	256	$\mathcal{W}^+$	[13]	[13], [41]	in-domain
SG-Distillation [53]	ECCV'20	O.	✓	✓	1024	$\mathcal{W}^+$	[14]	[13]	
MimicGAN [54]	IJCV'20	O.			64	$\mathcal{Z}$	[45]	[38]	for corrupted images
Chai *et al.* [55]	ICLR'21	L.	✓	✓	1024	$\mathcal{Z}, \mathcal{W}^+$	[11], [14]	[11], [13], [41]	data augmentation
pSp [28]	CVPR'21	L.	✓	✓	1024	$\mathcal{W}^+$	[14]	[11]	map2style module
StyleSpace [56]	CVPR'21	O.	✓	✓	1024	$\mathcal{S}$	[14]	[13], [41]	$\mathcal{S}$-space
GH-Feat [57]	CVPR'21	L.	✓	✓	256	$\mathcal{S}$	[13]	[13], [41], [47]	generative hierarchical feature
GANEnsembling [58]	CVPR'21	H.	✓	✓	1024	$\mathcal{W}^+$	[14]	[13], [41]	
e4e [59]	TOG'21	L.	✓	✓	1024	$\mathcal{W}^+$	[14]	[11], [13], [41]	encoder for editing
Xu *et al.* [60]	ICCV'21	O.	✓	✓	1024	$\mathcal{W}^+$	[13]	[13], [61]	for consecutive images
ReStyle [30]	ICCV'21	L.	✓	✓	1024	$\mathcal{W}^+$	[14]	[11], [13], [41], [62]	iterative refinement
BDInvert [63]	ICCV'21	O.	✓	✓	1024	$\mathcal{F}/\mathcal{W}^+$	[11], [14]	[11], [13], [41]	out-of-range, $\mathcal{F}/\mathcal{W}^+$-space
Zhu *et al.* [64]	arxiv'21	O.	✓	✓	1024	$\mathcal{P}$	[13], [14]	[13]	$\mathcal{P}$ and $\mathcal{P}^+$space
Wei *et al.* [29]	arxiv'21	L.	✓	✓	1024	$\mathcal{W}^+$	[14]	[11], [13]	efficient encoder architecture
PTI [65]	arxiv'21	H.	✓		1024	$\mathcal{W}$	[14]	[11], [13]	tune G around a pivot latent code
HyperStyle [66]	CVPR'22	H.	✓		1024	$\mathcal{W}$	[14]	[11], [13], [67]	learn to optimize the generator
HFGI [68]	CVPR'22	L.	✓	✓	1024	$\mathcal{W}^+$	[13], [14]	[11], [13], [67]	
HyperInverter [69]	CVPR'22	L.	✓		1024	$\mathcal{W}$	[14]	[11], [13], [41]	two-phase inversion

GAN dissection

GAN dissection

1. **GAN dissection** is understanding
   - what the GAN learns internally?
   - how the architectural choices could affect the learning?
2. Feature map of a filter gives a good approximation of the segmentation of the object class.
3. The idea is to see whether a particular class c is present in the image (Bau et al. 2019).
4. The feature maps r extracted from layers of GAN generators encode some information about the existence of a class c at particular locations in an image.



References

Reading

1. Peper A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications (Gui et al. 2023b).
2. Peper GAN Inversion: A Survey (Xia et al. 2023).
3. Chapter 26 of Probabilistic Machine Learning: Advanced Topics (Murphy 2023).
4. Chapter 8 of Deep Generative Modeling (Tomczak 2024).

References i

Abdal, Rameen, Yipeng Qin, and Peter Wonka (2019). “Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space?” In: *IEEE/CVF International Conference on Computer Vision*, pp. 4431–4440.

Ali, S. M. and S. D. Silvey (Dec. 2018). “A General Class of Coefficients of Divergence of One Distribution from Another”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 28.1, pp. 131–142.

Arjovsky, Martin and Leon Bottou (2017). “Towards Principled Methods for Training Generative Adversarial Networks”. In: *International Conference on Learning Representations.*

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). “Wasserstein GAN”. In: *ArXiv.*

Bau, David et al. (2019). “GAN Dissection: Visualizing and Understanding Generative Adversarial Networks”. In: *International Conference on Learning Representations.*

Chen, Xi et al. (2016). “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems*, pp. 2172–2180.

Donahue, Jeff, Philipp Krähenbühl, and Trevor Darrell (2017). “Adversarial Feature Learning”. In: *International Conference on Learning Representations.*

Goodfellow, Ian J. et al. (2014). “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems*, pp. 2672–2680.

References ii

- Gui, Jie et al. (2023a). “A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications”. In: *IEEE Trans. Knowl. Data Eng.* 35.4, pp. 3313–3332.
- – (2023b). “A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications”. In: *IEEE Transactions on Knowledge and Data Engineering* 35.4, pp. 3313–3332.
- Jeon, Insu et al. (2021). “IB-GAN: Disentangled Representation Learning with Information Bottleneck Generative Adversarial Networks”. In: *Thirty-Fifth AAAI Conference on Artificial Intelligence*, pp. 7926–7934.
- Mirza, Mehdi and Simon Osindero (2014). “Conditional Generative Adversarial Nets”. In: *CoRR* abs/1411.1784. URL: `http://arxiv.org/abs/1411.1784`.
- Murphy, Kevin P. (2023). *Probabilistic Machine Learning: Advanced Topics*. The MIT Press.
- Nowozin, Sebastian, Botond Cseke, and Ryota Tomioka (2016). “f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization”. In: *Advances in Neural Information Processing Systems*, pp. 271–279.
- Perarnau, Guim et al. (2016). “Invertible Conditional GANs for image editing”. In: *NIPS Workshop on Adversarial Training*.
- Radford, Alec, Luke Metz, and Soumith Chintala (2016). “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *International Conference on Learning Representations*.

References iii

- Salimans, Tim et al. (2016). "Improved Techniques for Training GANs". In: *Advances in Neural Information Processing Systems*, pp. 2226–2234.
- Tishby, Naftali, Fernando C. N. Pereira, and William Bialek (1999). "The information bottleneck method". In: *The 37th annual Allerton Conference on Communication, Control, and Computing*. Vol. physics/0004057.
- Tishby, Naftali and Noga Zaslavsky (2015). "Deep learning and the information bottleneck principle". In: *IEEE Information Theory Workshop*, pp. 1–5.
- Tomczak, Jakub M. (2024). *Deep Generative Modeling*. Springer.
- Wang, Yang (2020). "A Mathematical Introduction to Generative Adversarial Nets (GAN)". In: *CoRR* abs/2009.00169.
- Xia, Weihao et al. (2023). "GAN Inversion: A Survey". In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 45.3, pp. 3121–3138.
- Zhu, Jun-Yan, Philipp Krähenbühl, et al. (2016). "Generative Visual Manipulation on the Natural Image Manifold". In: *European Conference on Computer Vision*. Vol. 9909. Lecture Notes in Computer Science. Springer, pp. 597–613.
- Zhu, Jun-Yan, Taesung Park, et al. (2017). "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *IEEE International Conference on Computer Vision*, pp. 2242–2251.

Questions?