Lecture 16

Generative Adversarial Networks (GAN)

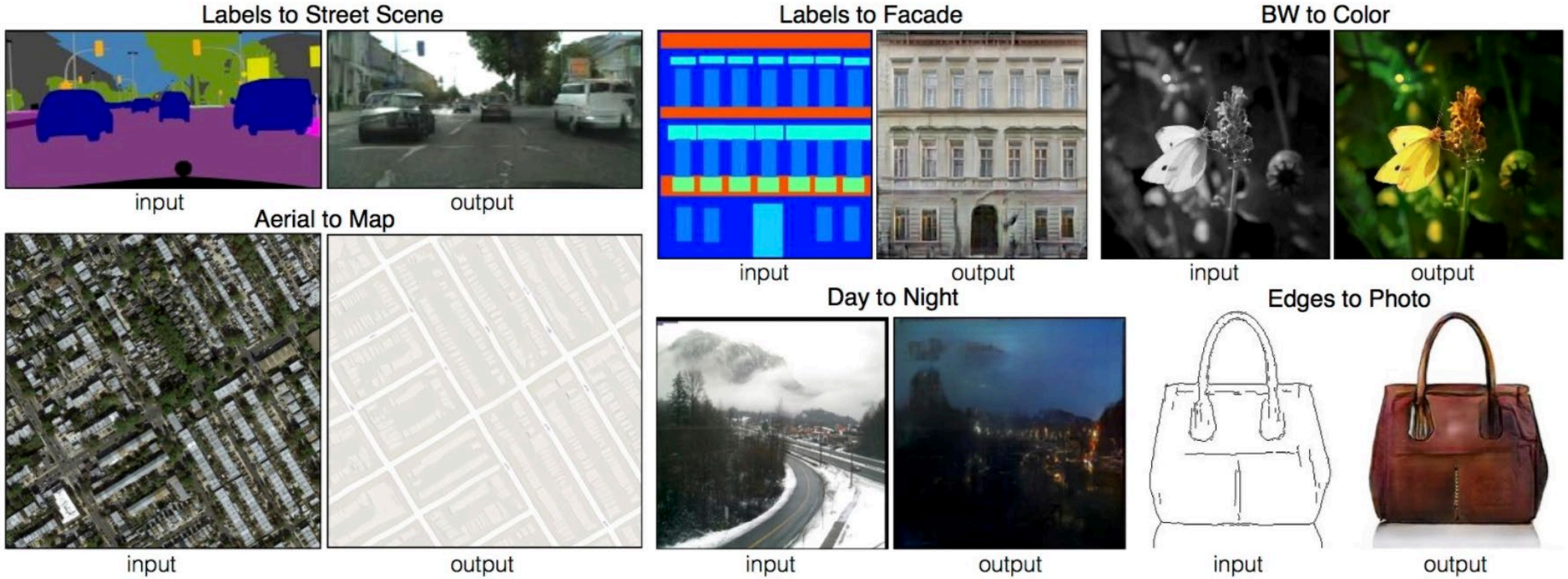
Generative Adversarial Networks (GAN)

- Original paper:
  - Generative Adversarial Nets
- Authors:
  - Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio (2014)
- Organization:
  - Université de Montréal
- URL:
  - https://arxiv.org/abs/1406.2661

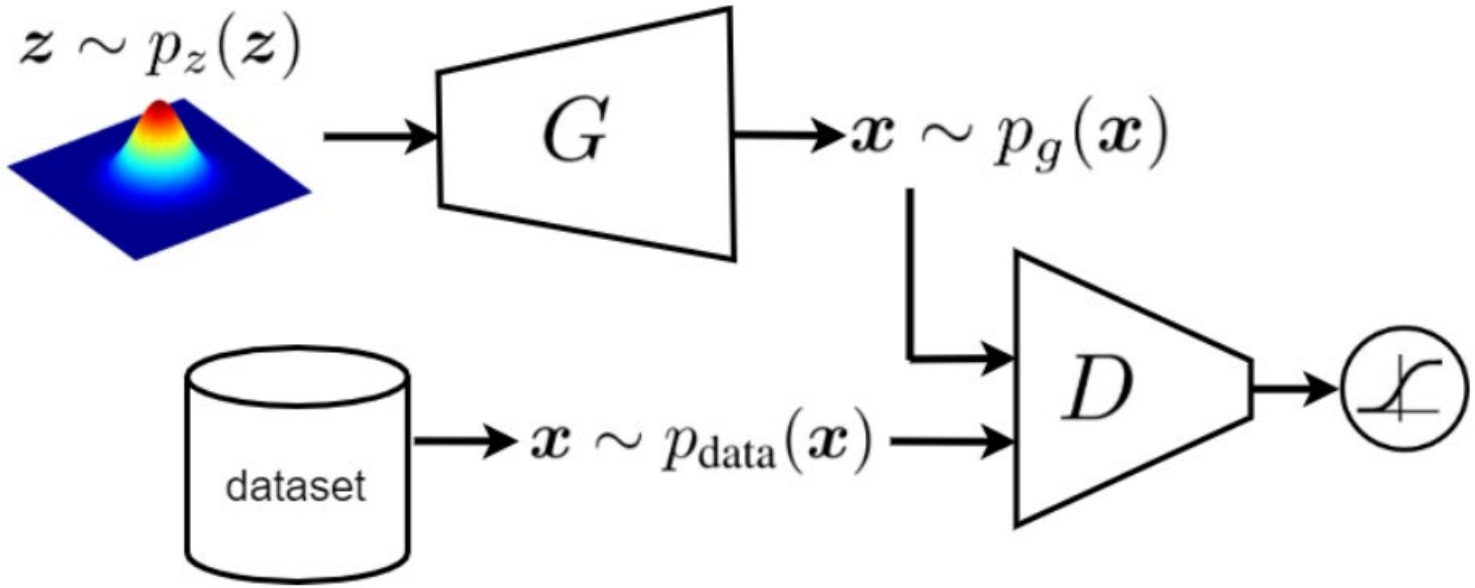
This person doesn't not exist

This person doesn't not exist

Different Applications



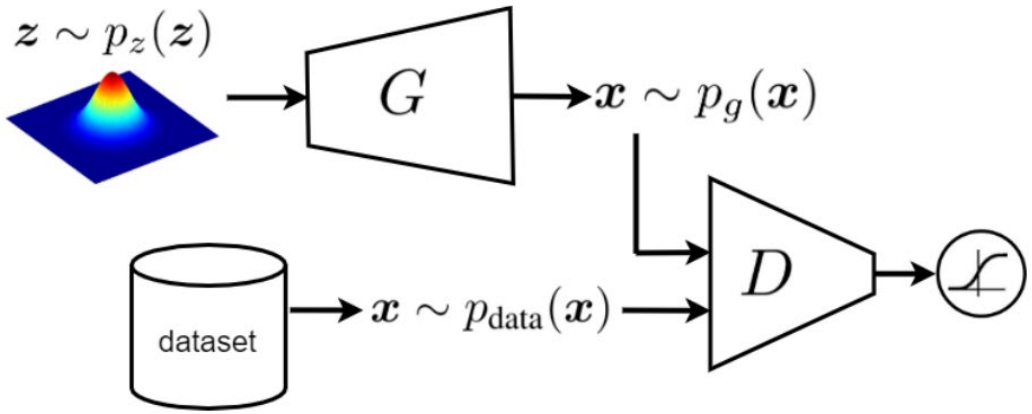
Adversarial Learning



Adversarial Learning

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

$$V(D, G) = \mathbb{E}_{x \sim P_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim P_z(z)}[log(1 - D(G(z)))]$$



Training
Generative Adversarial Networks

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

$$\max_D V(D, \mathbf{G}) \Rightarrow \min_G V(\mathbf{D}, G) \Rightarrow \max_D V(D, \mathbf{G}) \Rightarrow \min_G V(\mathbf{D}, G) \Rightarrow$$

$$V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$$

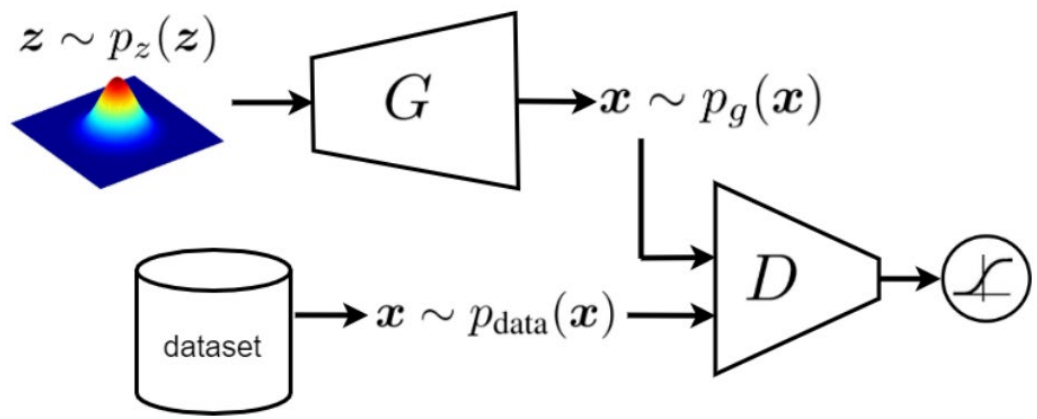
$$V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))]$$

$$= \int_x p_{data}(x) log(D(x)) dx + \int_z p_z(z) log(1 - D(G(z))) dz$$

$$V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$$

$$= \int_x p_{data}(x)log(D(x))dx + \int_z p_z(z)log(1-D(G(z)))dz$$

$$x = G(z) \Rightarrow z = G^{-1}(x) \Rightarrow dz = (G^{-1})'(x)dx$$



$$V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$$

$$= \int_x p_{data}(x)log(D(x))dx + \int_z p_z(z)log(1-D(G(z)))dz$$

$$x = G(z) \Rightarrow z = G^{-1}(x) \Rightarrow dz = (G^{-1})'(x)dx$$

$$\Rightarrow p_g(x) = p_z(G^{-1}(x))(G^{-1})'(x)$$

$$V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$$

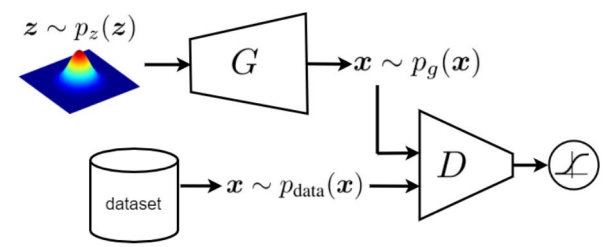
$$= \int_x p_{data}(x)log(D(x))dx + \int_z p_z(z)log(1-D(G(z)))dz$$

$$x = G(z) \Rightarrow z = G^{-1}(x) \Rightarrow dz = (G^{-1})'(x)dx$$

$$\Rightarrow p_g(x) = p_z(G^{-1}(x))(G^{-1})'(x)$$

If you have a random variable z with a known distribution $p_z(z)$, and you transform z using a function G, to get a new random variable $x = G(z)$, then the distribution of x will generally be given by:

$$p_g(x) = p_z(G^{-1}(x)) \cdot (G^{-1})'(x)$$



$$V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$$

$$= \int_x p_{data}(x) log(D(x)) dx + \int_z p_z(z) log(1-D(G(z))) dz$$

$$x = G(z) \Rightarrow z = G^{-1}(x) \Rightarrow dz = (G^{-1})'(x) dx$$

$$\Rightarrow p_g(x) = p_z(G^{-1}(x))(G^{-1})'(x)$$

$$= \int_x p_{data}(x) log(D(x)) dx + \int_x p_z(G^{-1}(x)) log(1-D(x))(G^{-1})'(x) dx$$

$$V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$$

$$= \int_x p_{data}(x) log(D(x)) dx + \int_z p_z(z) log(1-D(G(z))) dz$$

$$x = G(z) \Rightarrow z = G^{-1}(x) \Rightarrow dz = (G^{-1})'(x) dx$$

$$\Rightarrow p_g(x) = p_z(G^{-1}(x))(G^{-1})'(x)$$

$$= \int_x p_{data}(x) log(D(x)) dx + \int_x p_z(G^{-1}(x)) log(1-D(x))(G^{-1})'(x) dx$$

$$V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$$

$$= \int_x p_{data}(x)log(D(x))dx + \int_z p_z(z)log(1-D(G(z)))dz$$

$$x = G(z) \Rightarrow z = G^{-1}(x) \Rightarrow dz = (G^{-1})'(x)dx$$

$$\Rightarrow p_g(x) = p_z(G^{-1}(x))(G^{-1})'(x)$$

$$= \int_x p_{data}(x)log(D(x))dx + \int_x p_z(G^{-1}(x))log(1-D(x))(G^{-1})'(x)dx$$

$$= \int_x p_{data}(x)log(D(x))dx + \int_x p_g(x)log(1-D(x))dx$$

$$V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1-D(G(z)))]$$

$$= \int_x p_{data}(x)log(D(x))dx + \int_z p_z(z)log(1-D(G(z)))dz$$

$$x = G(z) \Rightarrow z = G^{-1}(x) \Rightarrow dz = (G^{-1})'(x)dx$$

$$\Rightarrow p_g(x) = p_z(G^{-1}(x))(G^{-1})'(x)$$

$$= \int_x p_{data}(x)log(D(x))dx + \int_x p_z(G^{-1}(x))log(1-D(x))(G^{-1})'(x)dx$$

$$= \int_x p_{data}(x)log(D(x))dx + \int_x p_g(x)log(1-D(x))dx$$

$$= \int_x p_{data}(x)log(D(x)) + p_g(x)log(1-D(x))dx$$

Understanding the objective function

$$\max_D V(D,G) = \max_D \int_x p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x)) dx$$

Understanding the objective function

$$\max_D V(D,G) = \max_D \int_x p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x)) dx$$

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

Understanding the objective function

$$\max_D V(D,G) = \max_D \int_x p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x)) dx$$

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

$$\Rightarrow \frac{p_{data}(x)}{D(x)} - \frac{p_g(x)}{1 - D(x)} = 0$$

Understanding the objective function

$$\max_D V(D,G) = \max_D \int_x p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x)) dx$$

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

$$\Rightarrow \frac{p_{data}(x)}{D(x)} - \frac{p_g(x)}{1 - D(x)} = 0$$

$$\Rightarrow D(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

Understanding the objective function

Suppose the discriminator is optimal $D_G^*(x)$,
the optimal generator makes: $p_{data}(x) = p_g(x)$

$$\Rightarrow D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

Understanding the objective function

$$C(G) = \max_D V(G, D)$$

Understanding the objective function

$$C(G) = \max_D V(G, D)$$

$$= \max_D \int_x p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x)) dx$$

Understanding the objective function

$$C(G) = \max_D V(G, D)$$

$$= \max_D \int_x p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x)) dx$$

$$= \int_x p_{data}(x) log(D^*_G(x)) + p_g(x) log(1 - D^*_G(x)) dx$$

Understanding the objective function

$$C(G) = \max_D V(G, D)$$

$$= \max_D \int_x p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x)) dx$$

$$= \int_x p_{data}(x) log(D_G^*(x)) + p_g(x) log(1 - D_G^*(x)) dx$$

$$= \int_x p_{data}(x) log(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}) + p_g(x) log(\frac{p_g(x)}{p_{data}(x) + p_g(x)}) dx$$

Understanding the objective function

$$C(G) = \max_D V(G, D)$$

$$= \max_D \int_x p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x)) dx$$

$$= \int_x p_{data}(x) log(D_G^*(x)) + p_g(x) log(1 - D_G^*(x)) dx$$

$$= \int_x p_{data}(x) log(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}) + p_g(x) log(\frac{p_g(x)}{p_{data}(x) + p_g(x)}) dx$$

$$= \int_x p_{data}(x) log(\frac{p_{data}(x)}{\frac{p_{data}(x)+p_g(x)}{2}}) + p_g(x) log(\frac{p_g(x)}{\frac{p_{data}(x)+p_g(x)}{2}}) dx - log(4)$$

Understanding the objective function

$$C(G) = \max_D V(G, D)$$

$$= \max_D \int_x p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x)) dx$$

$$= \int_x p_{data}(x) log(D_G^*(x)) + p_g(x) log(1 - D_G^*(x)) dx$$

$$= \int_x p_{data}(x) log(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}) + p_g(x) log(\frac{p_g(x)}{p_{data}(x) + p_g(x)}) dx$$

$$= \int_x p_{data}(x) log(\frac{p_{data}(x)}{\frac{p_{data}(x)+p_g(x)}{2}}) + p_g(x) log(\frac{p_g(x)}{\frac{p_{data}(x)+p_g(x)}{2}}) dx - log(4)$$

$$= KL[p_{data}(x) || \frac{p_{data}(x) + p_g(x)}{2}] + KL[p_g(x) || \frac{p_{data}(x) + p_g(x)}{2}] - log(4)$$

Understanding the objective function

$$C(G) = \underbrace{KL[p_{data}(x)||\frac{p_{data}(x)+p_g(x)}{2}]}_{\geq 0} + \underbrace{KL[p_g(x)||\frac{p_{data}(x)+p_g(x)}{2}]}_{\geq 0} - log(4)$$

Understanding the objective function

$$C(G) = \underbrace{KL[p_{data}(x)||\frac{p_{data}(x)+p_g(x)}{2}]}_{\geq 0} + \underbrace{KL[p_g(x)||\frac{p_{data}(x)+p_g(x)}{2}]}_{\geq 0} - log(4)$$

$$\min_G C(G) = 0 + 0 - log(4) = -log(4)$$

Understanding the objective function

$$C(G) = \underbrace{KL[p_{data}(x)||\frac{p_{data}(x)+p_g(x)}{2}]}_{\geq 0} + \underbrace{KL[p_g(x)||\frac{p_{data}(x)+p_g(x)}{2}]}_{\geq 0} - log(4)$$

$$\min_G C(G) = 0 + 0 - log(4) = -log(4)$$

$$KL[p_{data}(x)||\frac{p_{data}(x) + p_g(x)}{2}] = 0$$

Understanding the objective function

$$C(G) = \underbrace{KL[p_{data}(x)||\frac{p_{data}(x)+p_g(x)}{2}]}_{\geq 0} + \underbrace{KL[p_g(x)||\frac{p_{data}(x)+p_g(x)}{2}]}_{\geq 0} - log(4)$$

$$\min_G C(G) = 0 + 0 - log(4) = -log(4)$$

$$KL[p_{data}(x)||\frac{p_{data}(x) + p_g(x)}{2}] = 0$$

when $p_{data}(x) = \frac{p_{data}(x)+p_g(x)}{2}$

Understanding the objective function

$$C(G) = \underbrace{KL[p_{data}(x)||\frac{p_{data}(x)+p_g(x)}{2}]}_{\geq 0} + \underbrace{KL[p_g(x)||\frac{p_{data}(x)+p_g(x)}{2}]}_{\geq 0} - log(4)$$

$$\min_G C(G) = 0 + 0 - log(4) = -log(4)$$

$$KL[p_{data}(x)||\frac{p_{data}(x) + p_g(x)}{2}] = 0$$

$$\text{when} \quad p_{data}(x) = \frac{p_{data}(x)+p_g(x)}{2}$$

$$\Rightarrow p_{data}(x) = p_g(x)$$

KL (Kullback-Leibler) divergence

- Jensen-Shannon Divergency (symmetric KL):

$$JSD(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M),$$

$$M = \frac{1}{2}(P + Q)$$

Summary:

- Generator G, Discriminator D

$$V = \mathbb{E}_{x \sim P_{data}}[logD(x)] + \mathbb{E}_{x \sim P_G}[log(1 - D(x))]$$

Summary:

- Generator G, Discriminator D
- Looking for G^* such that

$$G^* = arg \min_G \max_D V(G, D)$$

$$V = \mathbb{E}_{x \sim P_{data}}[logD(x)] + \mathbb{E}_{x \sim P_G}[log(1 - D(x))]$$

Summary:

- Generator G, Discriminator D
- Looking for G^* such that

$$G^* = arg \min_G \max_D V(G, D)$$

$$V = \mathbb{E}_{x \sim P_{data}}[logD(x)] + \mathbb{E}_{x \sim P_G}[log(1 - D(x))]$$

- Given $G, \max_D V(G, D)$

$$= -2log(2) + 2JSD(P_{data}(x)||P_G(x))$$

Summary:

$$V = \mathbb{E}_{x \sim P_{data}}[logD(x)] + \mathbb{E}_{x \sim P_G}[log(1 - D(x))]$$

- Generator G, Discriminator D
- Looking for G^* such that

$$G^* = arg \min_G \max_D V(G, D)$$

- Given $G, \max_D V(G, D)$

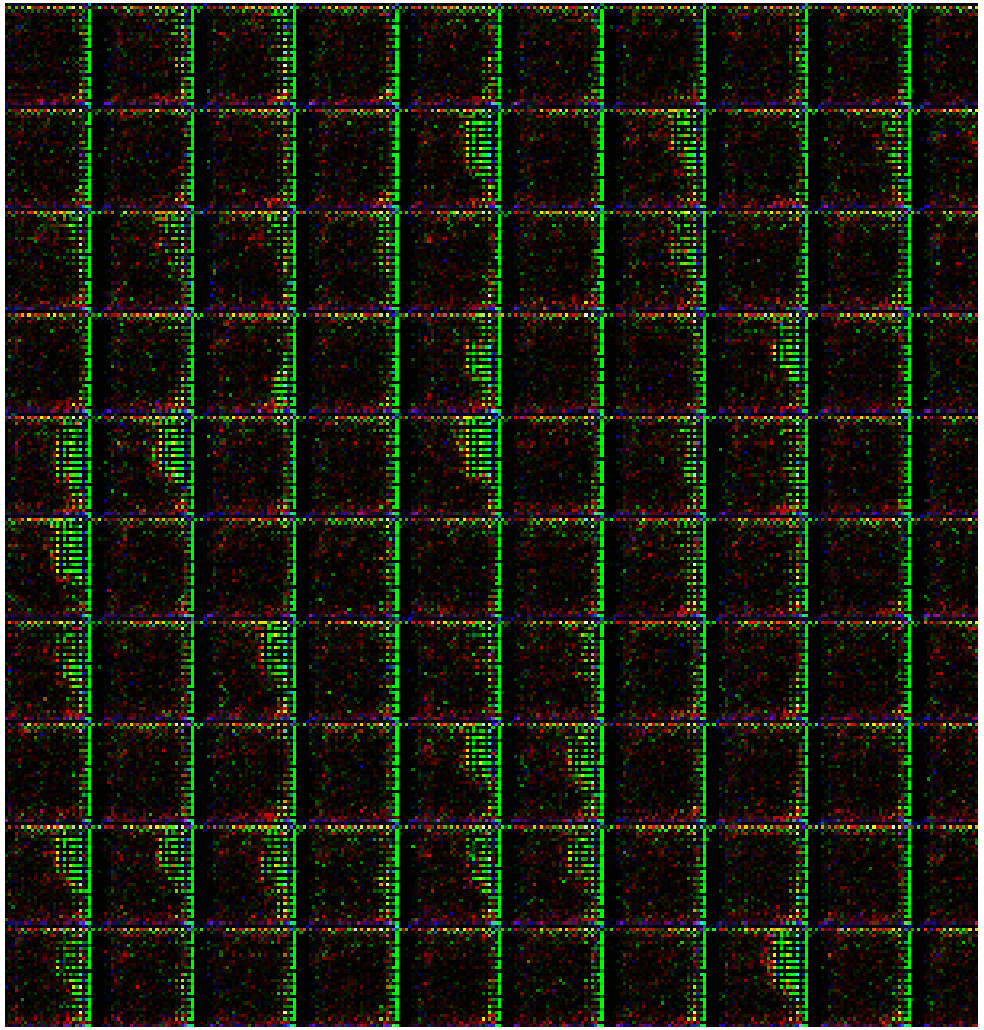
$$= -2log(2) + 2JSD(P_{data}(x)||P_G(x))$$

- What is the optimal G? It is G that makes JSD smallest $= 0$:

$$P_G(x) = P_{data}(x)$$

VAE

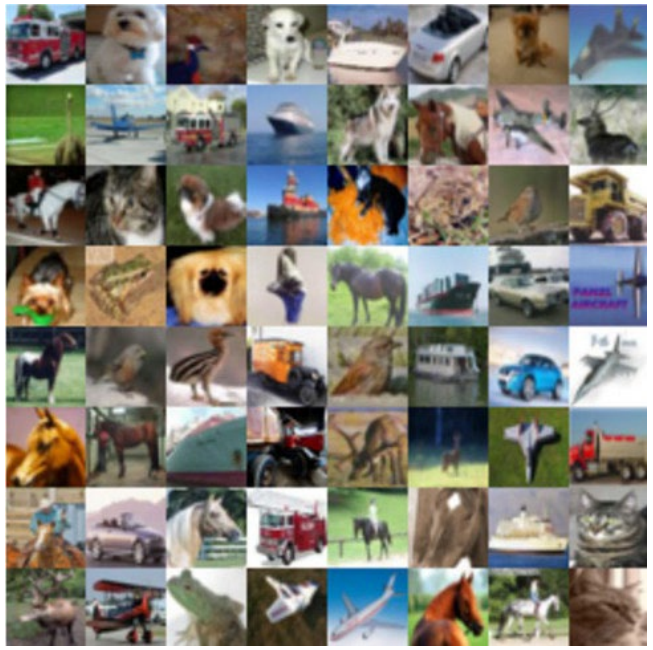
GAN



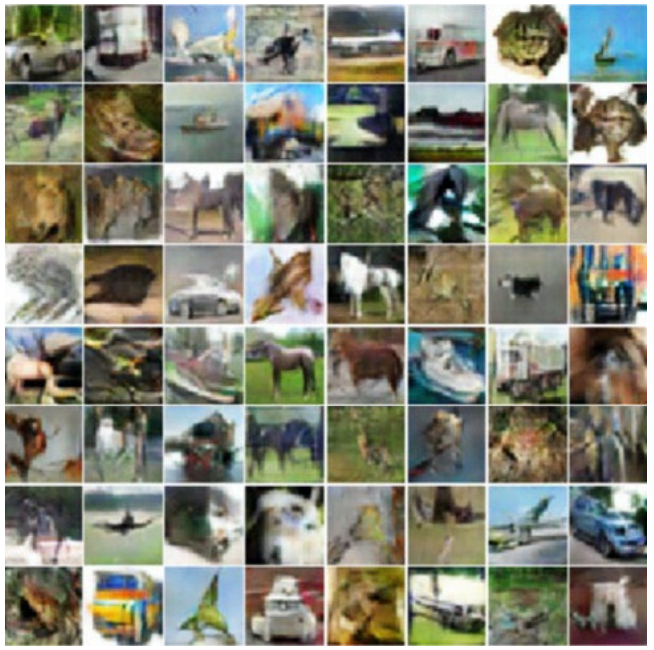
VAE

GAN

Real images (CIFAR-10)



Generated images



Source Code

- Original paper (theano):
  - https://github.com/goodfeli/adversarial
- Tensorflow implementation:
  - https://github.com/ckmarkoh/GAN-tensorflow

Conditional GAN, Mode Collapse, AAE and Applications of GAN

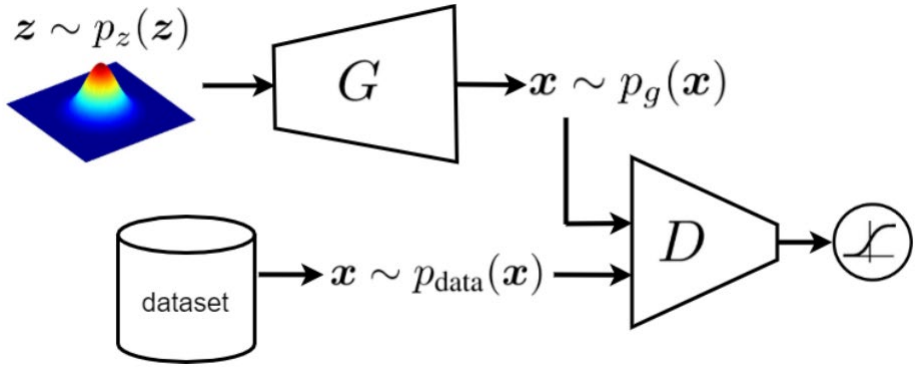
- Tutorial

  Generative Adversarial Networks and Adversarial Autoencoders: Tutorial and Survey

Generative adversarial network (GAN)

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

$$V(D,G) = \mathbb{E}_{x\sim P_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim P_z(z)}[\log(1-D(G(z)))]$$



Mode Collapse Problem in GAN

- The mode collapse problem (Metz et al., 2017), also known as the Helvetica scenario (Goodfellow, 2016), is a common problem in GAN models.

Mode Collapse Problem in GAN

- The mode collapse problem (Metz et al., 2017), also known as the Helvetica scenario (Goodfellow, 2016), is a common problem in GAN models.

- The generator learns to map several different z values to the same generated data point x.

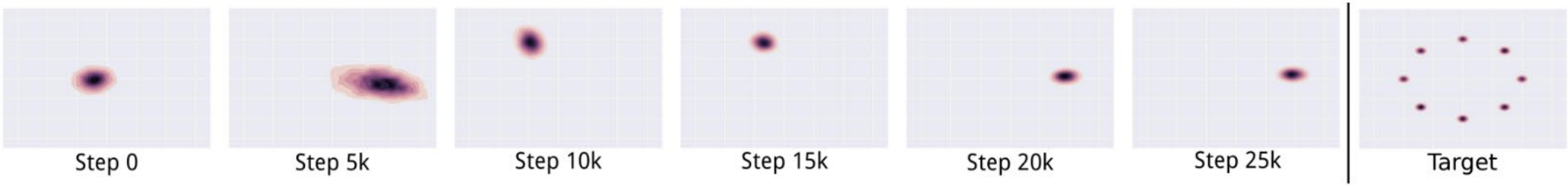
Mode Collapse Problem in GAN

- The mode collapse problem (Metz et al., 2017), also known as the Helvetica scenario (Goodfellow, 2016), is a common problem in GAN models.

- The generator learns to map several different z values to the same generated data point x.

- Mode collapse usually happens in GAN when the distribution of training data, $p_{data}(x)$, has multiple modes.

Mode Collapse example

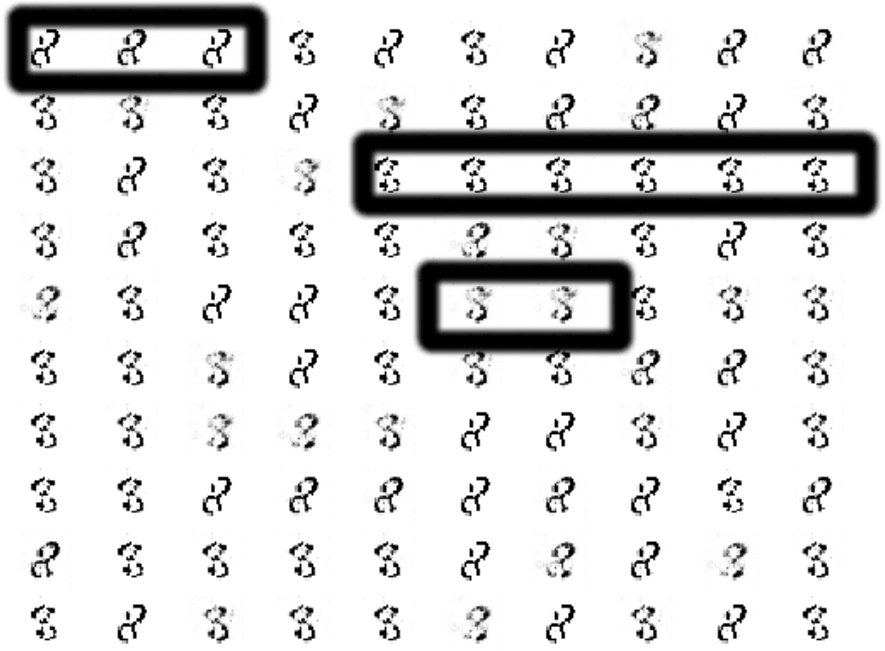


An example of mode collapse in GAN.

Credit : Metz et al., 2017

Mode Collapse example

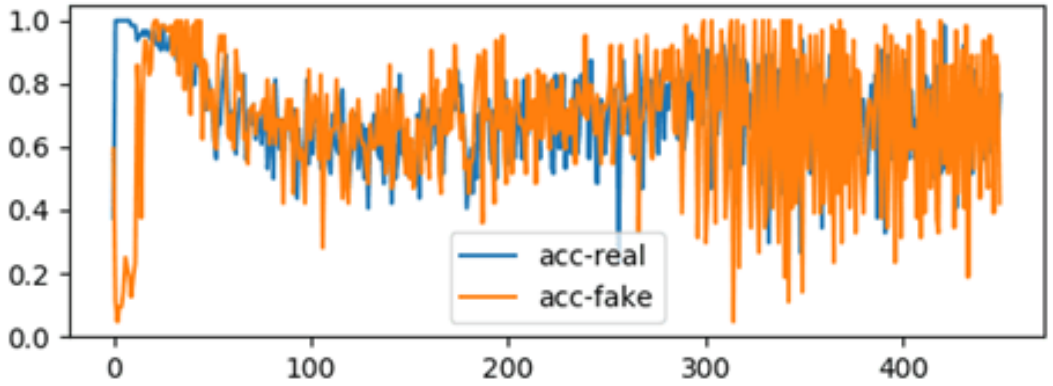
Stable GAN



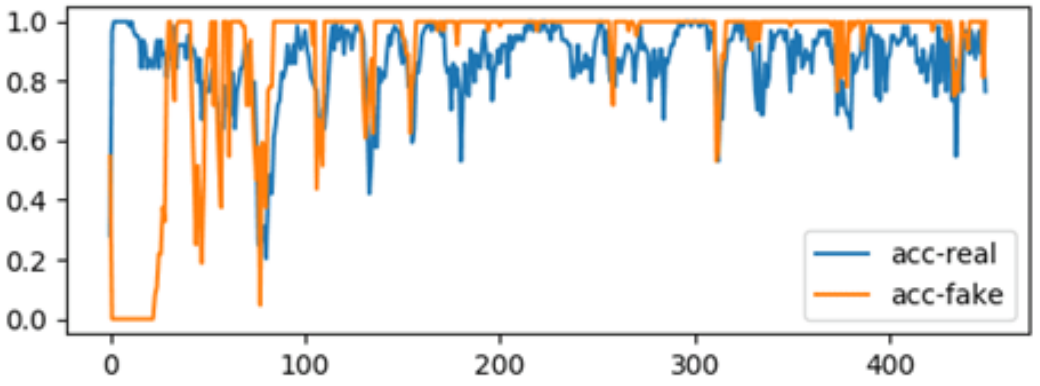
Mode Collapse.

Credit: Jason Brownlee

Mode Collapse example

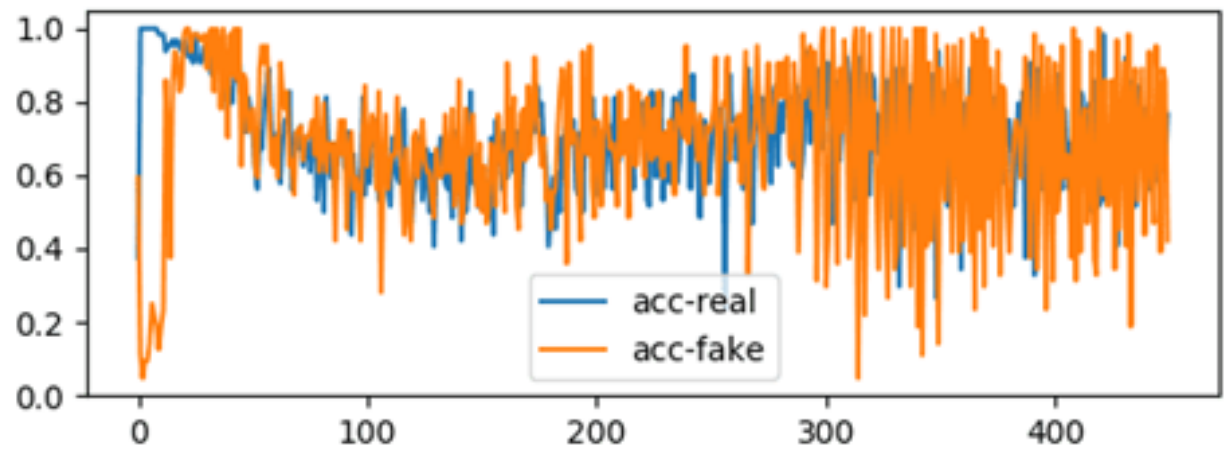


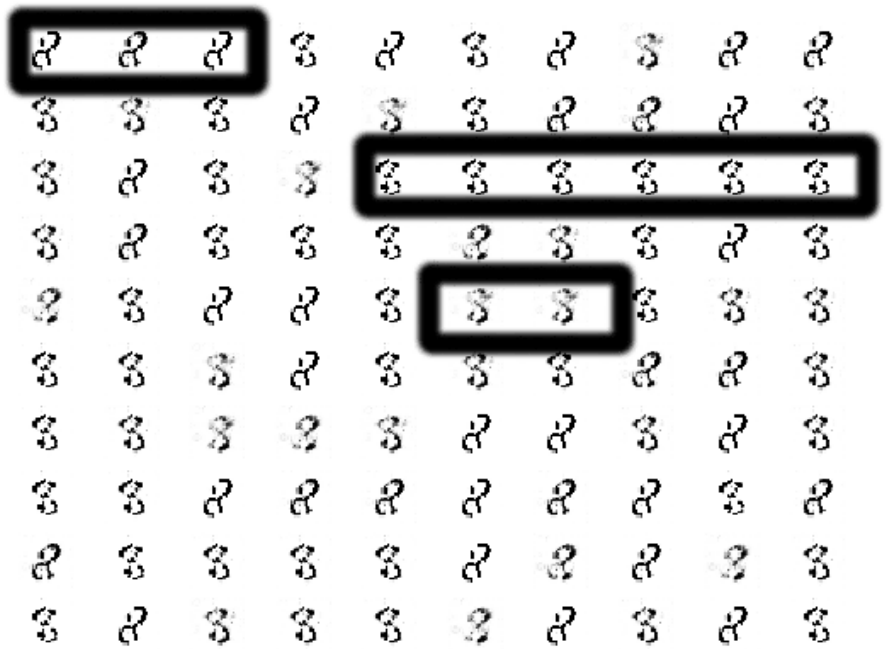
Accuracy for a Stable GAN

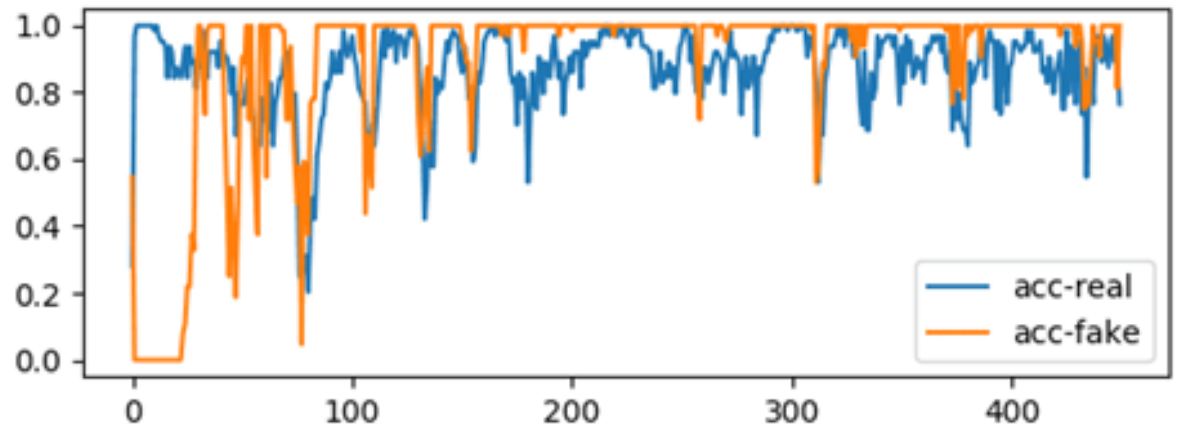


Accuracy for a GAN with Mode Collapse

Credit: Jason Brownlee


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
acc-real
acc-fake




1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
acc-real
acc-fake

Minibatch discrimination

- One way to resolve the mode collapse problem is minibatch discrimination (Salimans et al., 2016)

Minibatch discrimination

- One way to resolve the mode collapse problem is minibatch discrimination (Salimans et al., 2016)

- When mode collapses, all images created look similar.

Minibatch discrimination

- One way to resolve the mode collapse problem is minibatch discrimination (Salimans et al., 2016)

- When mode collapses, all images created look similar.

- Feed real images and generated images into the discriminator separately in different batches and compute the similarity of the image x with images in the same batch.

Minibatch discrimination

- Append the similarity $\boldsymbol{o}(\boldsymbol{x})$ in one of the dense layers in the discriminator to classify whether this image is real or generated.

Minibatch discrimination

- append the similarity $\boldsymbol{o}(\boldsymbol{x})$ in one of the dense layers in the discriminator to classify whether this image is real or generated.

- If the mode starts to collapse, the similarity of generated images increases.

Minibatch discrimination

- append the similarity $\boldsymbol{o}(\boldsymbol{x})$ in one of the dense layers in the discriminator to classify whether this image is real or generated.

- If the mode starts to collapse, the similarity of generated images increases.

- The discriminator can use this score to detect generated images and penalize the generator if mode is collapsing.

Alternative Strategies to Prevent Mode Collapse

- **Unrolled GANs**: They allow the generator to update itself using a copy of the discriminator from several steps ahead, preventing the generator from overfitting to the current discriminator.
- **Experience Replay**: This involves keeping a memory bank of past generated images and occasionally showing them to the discriminator, which helps maintain diversity in the generator's output.
- **Modified Training Objectives**: Alternative loss functions, like Wasserstein loss or least squares loss, provide more stable gradients and can help prevent mode collapse.

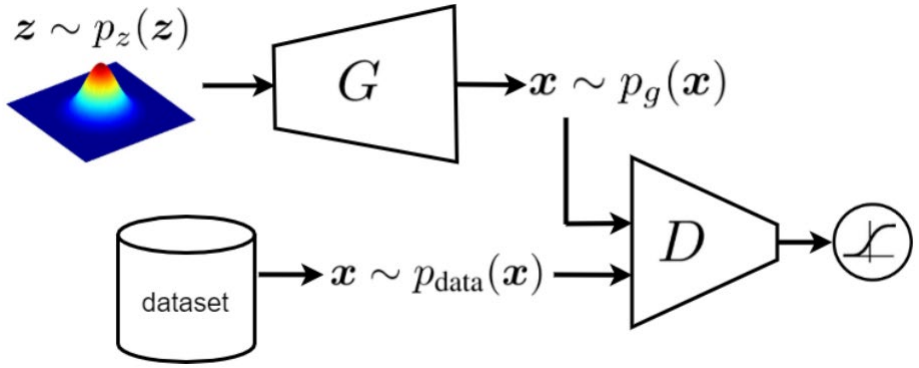
Alternative Strategies to Prevent Mode Collapse

- **Regularization**: Adding noise to inputs or labels, or using dropout in the discriminator, can prevent it from making overly confident decisions, which in turn pressures the generator to be more diverse.
- **Architecture Tweaks**: Adjusting the network architecture, such as adding more layers or changing activation functions, can help the generator explore a wider range of outputs.
- **Feature Matching**: The generator is trained to match the statistical features of the real data in some intermediate layer of the discriminator, promoting diversity in the generated data.

Alternative Strategies to Prevent Mode Collapse

- **Penalizing the Discriminator**: Introducing penalties for the discriminator when it gets too confident can prevent it from overpowering the generator, leading to a more varied generation.
- **Two Time-Scale Update Rule (TTUR)**: Using different learning rates for the generator and the discriminator can help balance their training and prevent the generator from collapsing to a few modes.

Generative adversarial network (GAN)



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

$$(D,G) = \mathbb{E}_{x\sim P_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim P_z(z)}[\log(1 - D(G(z)))]$$

Conditional GAN

- Assume that the dataset with which GAN is trained has c number of classes.

Conditional GAN

- Assume that the dataset with which GAN is trained has c number of classes.
- The original GAN generates points from any class, and we do not have control to generate a point from a specific class.

Conditional GAN

- Assume that the dataset with which GAN is trained has c number of classes.
- The original GAN generates points from any class, and we do not have control to generate a point from a specific class.

- Conditional GAN (Mirza & Osindero), also called the conditional adversarial network, gives the user the opportunity to choose the class of generation of points.

GAN

$$\min_{G} \max_{D} V(D,G) := \mathbb{E}_{x\sim P_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim P_z(z)}[\log(1 - D(G(z)))]$$

Conditional GAN

For the dataset $\{x_i \in \mathbb{R}^d\}_{i=1}^n$, let the one-hot encoded class labels be $\{y_i \in \mathbb{R}^c\}_{i=1}^n$.
In conditional GAN, we can use the following loss function instead:

$$\min_G \max_D V_C(D,G) \coloneqq \mathbb{E}_{x\sim p_{data}(x)}[\log(D(x|y))] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z|y)))]$$

$$\min_G \max_D V(D,G) := \mathbb{E}_{x\sim P_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim P_z(z)}[\log(1-D(G(z)))]$$

the discriminator and generator are both conditioned on the labels

Conditional GAN

For the dataset $\{x_i \in \mathbb{R}^d\}_{i=1}^n$, let the one-hot encoded class labels be $\{y_i \in \mathbb{R}^c\}_{i=1}^n$.
In conditional GAN, we can use the following loss function instead:

$$\min_G \max_D V_C(D,G) \coloneqq \mathbb{E}_{x\sim p_{data}(x)}\big[\log\big(D(x|y)\big)\big] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z|y)))]$$

the discriminator and generator are both conditioned on the labels

Implementation of Conditional GAN

- Concatenate the one-hot encoded label y to the point x for the input to the discriminator.

Implementation of Conditional GAN

- Concatenate the one-hot encoded label y to the point x for the input to the discriminator.

- Concatenate the one-hot encoded label y to the noise z for the input to the generator.

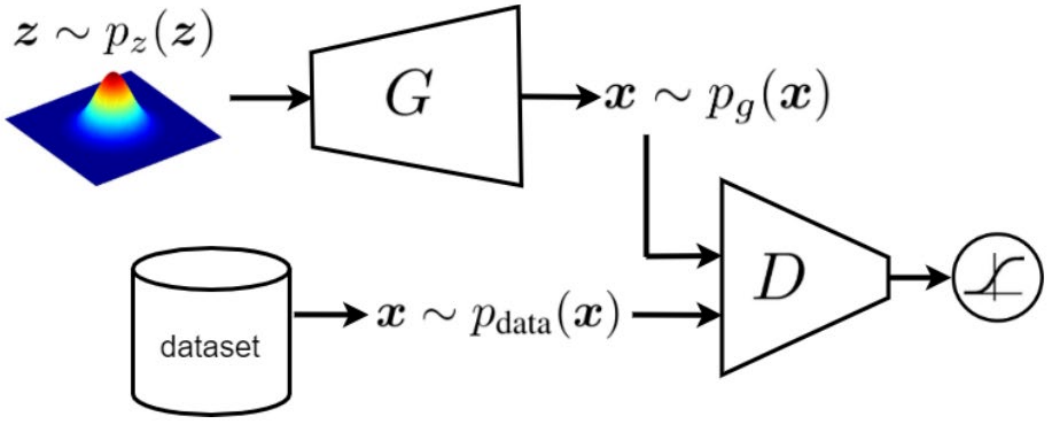
Implementation of Conditional GAN

- Concatenate the one-hot encoded label y to the point x for the input to the discriminator.

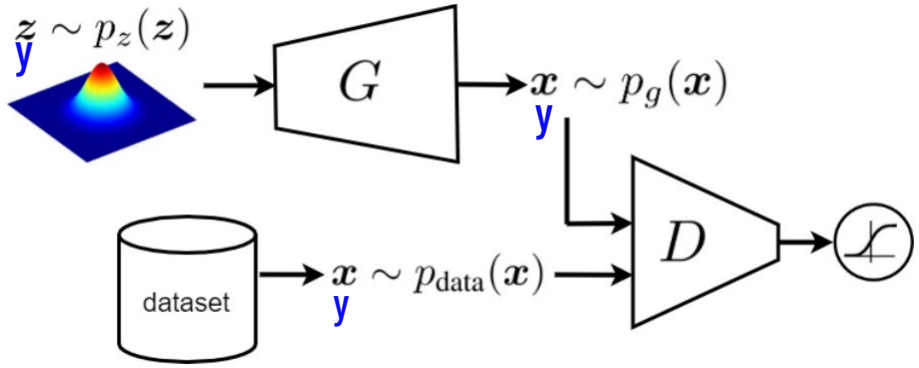
- Concatenate the one-hot encoded label y to the noise z for the input to the generator.

- For these, the input layers of discriminator and generator are enlarged to accept the concatenated inputs

GAN

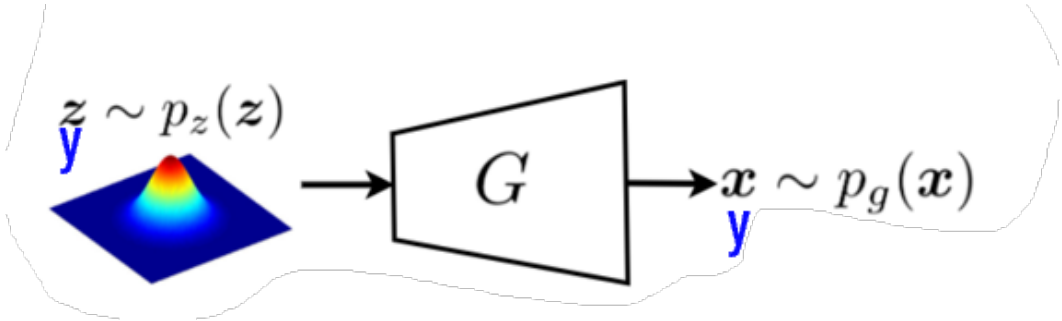


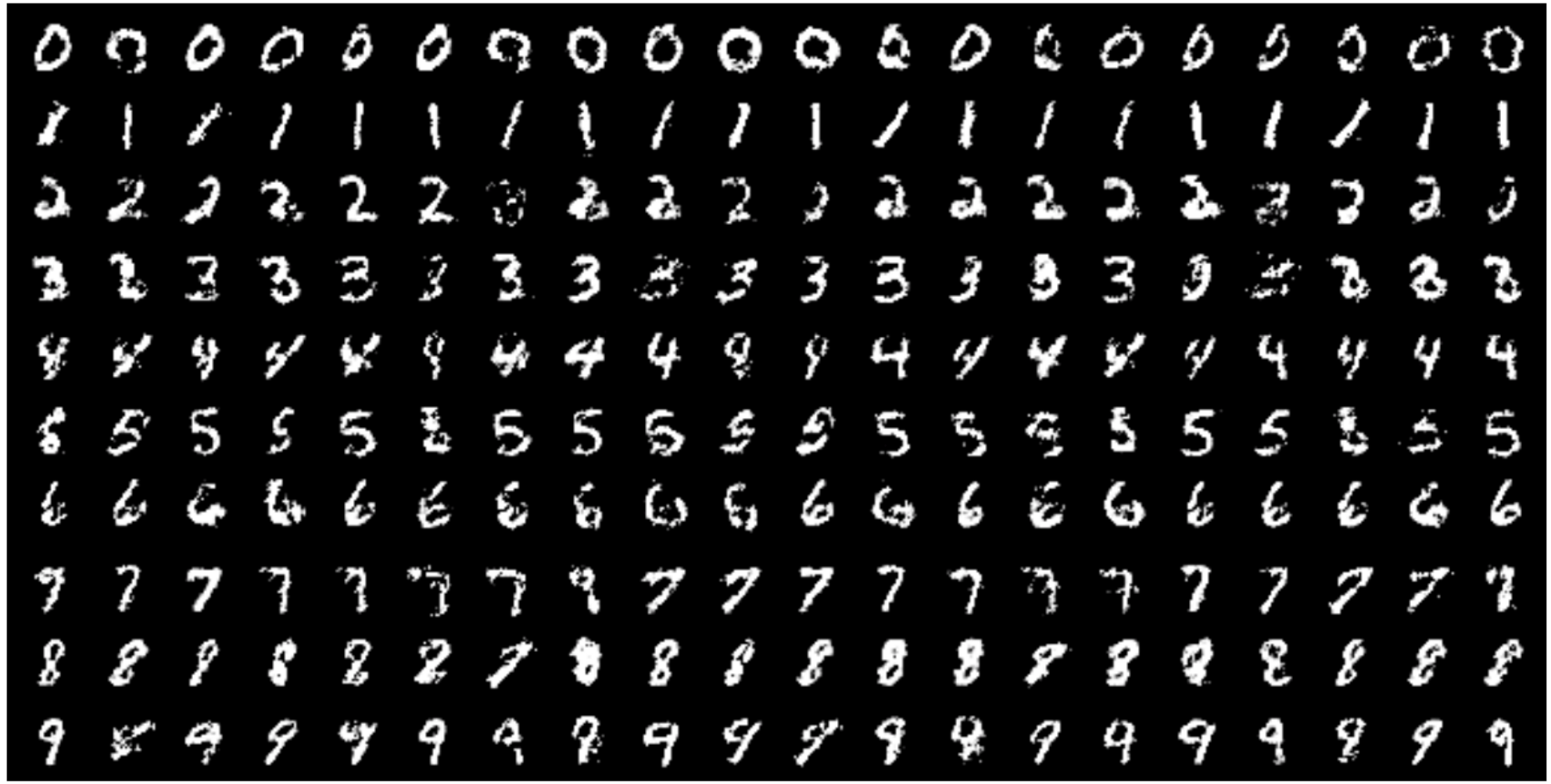
Conditional GAN



Inference from Conditional GAN

- In the inference phase, the user choose the desired class label and the generator generates a new point from that class.





Generated MNIST digits, each row conditioned on one label

Adversarial Autoencoder (AAE)

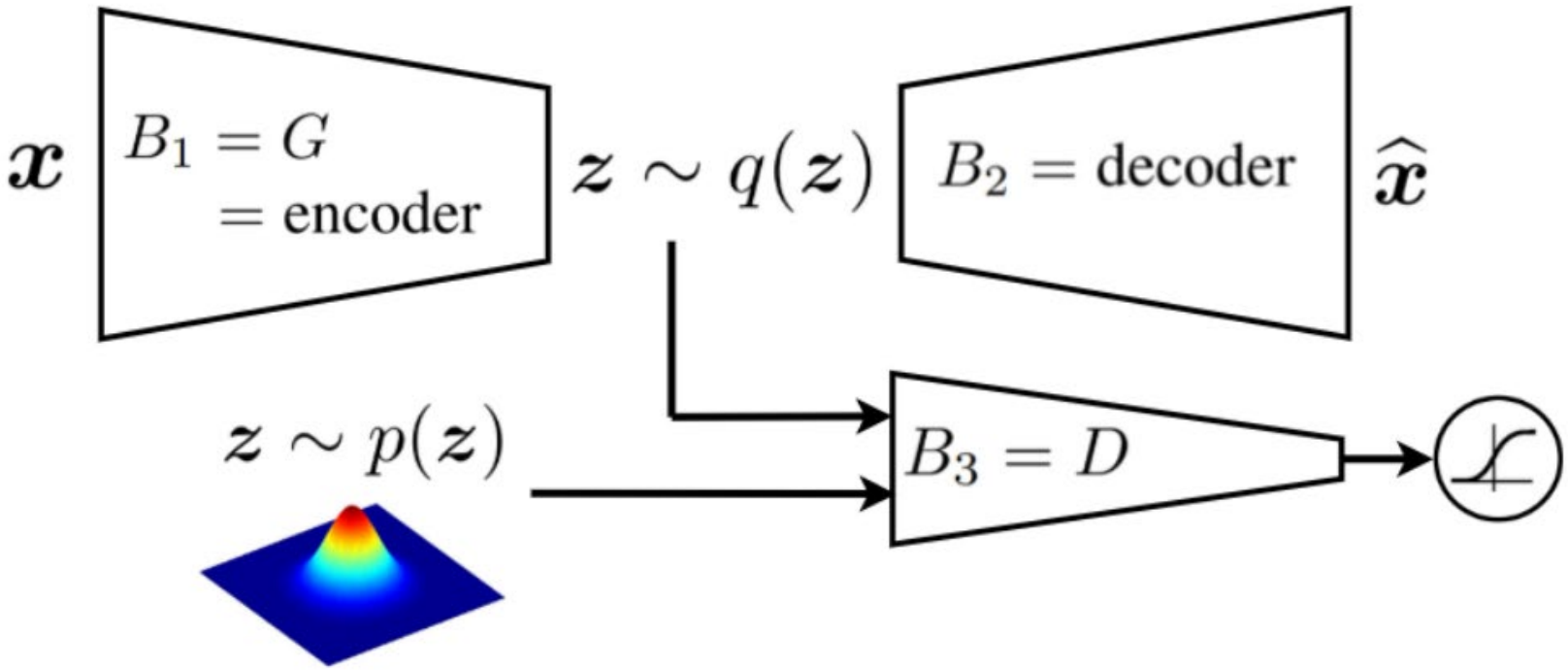
- Adversarial Autoencoder (AAE) was proposed in (Makhzani et al., 2015).

Adversarial Autoencoder (AAE)

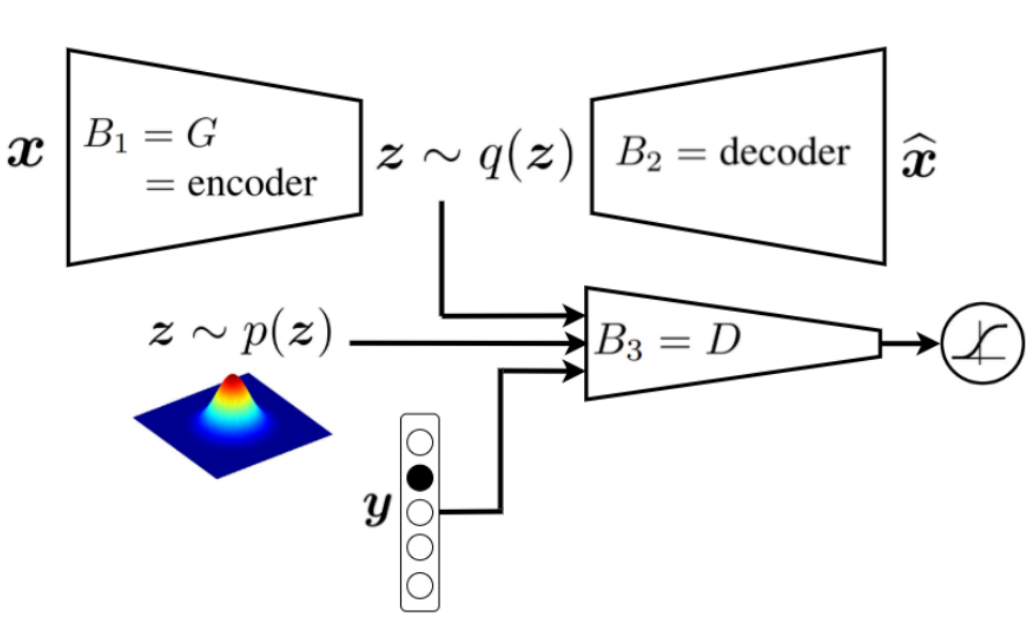
- Adversarial Autoencoder (AAE) was proposed in (Makhzani et al., 2015).

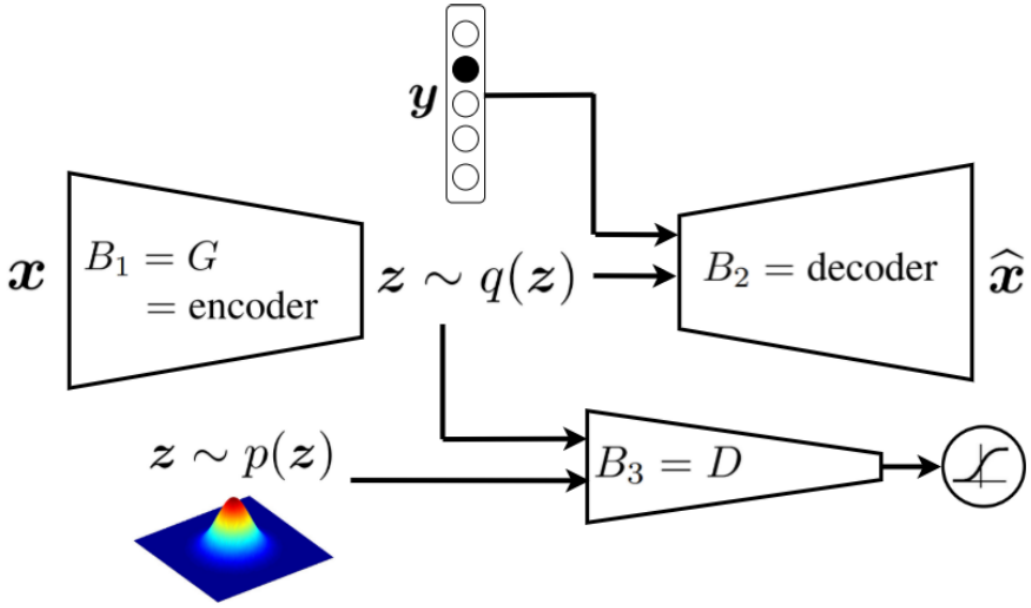
- In contrast to variational autoencoder (Kingma & Welling, 2014) which uses KL divergence and evidence lower bound, AAE uses adversarial learning for imposing a specific distribution on the latent variable in its coding layer.

Unsupervised AAE



Supervised AAE





Clustering with AAE

- Assume we have c number of clusters.

Clustering with AAE

- Assume we have c number of clusters.

- All points are unlabeled.

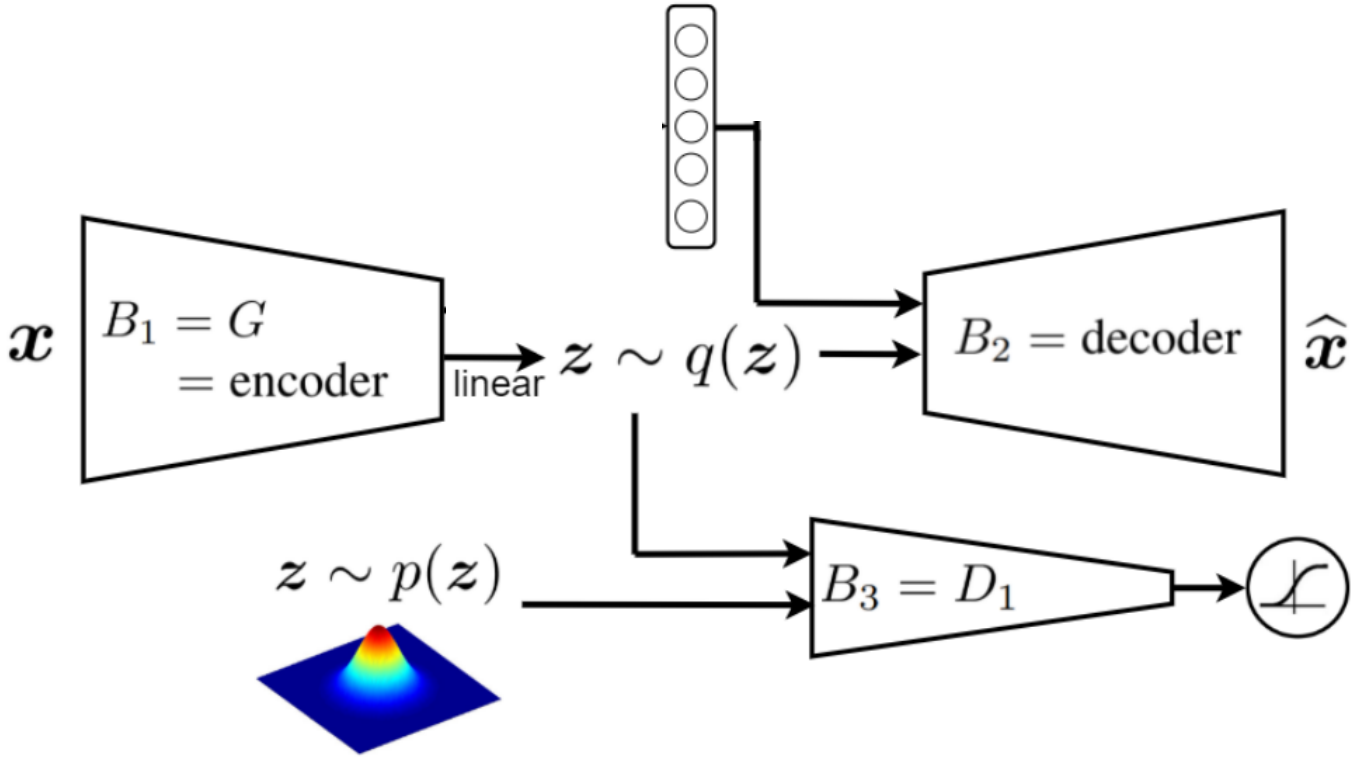
Clustering with AAE

- Assume we have c number of clusters.
- All points are unlabeled.
- The cluster indices are sampled randomly by the categorical distribution.

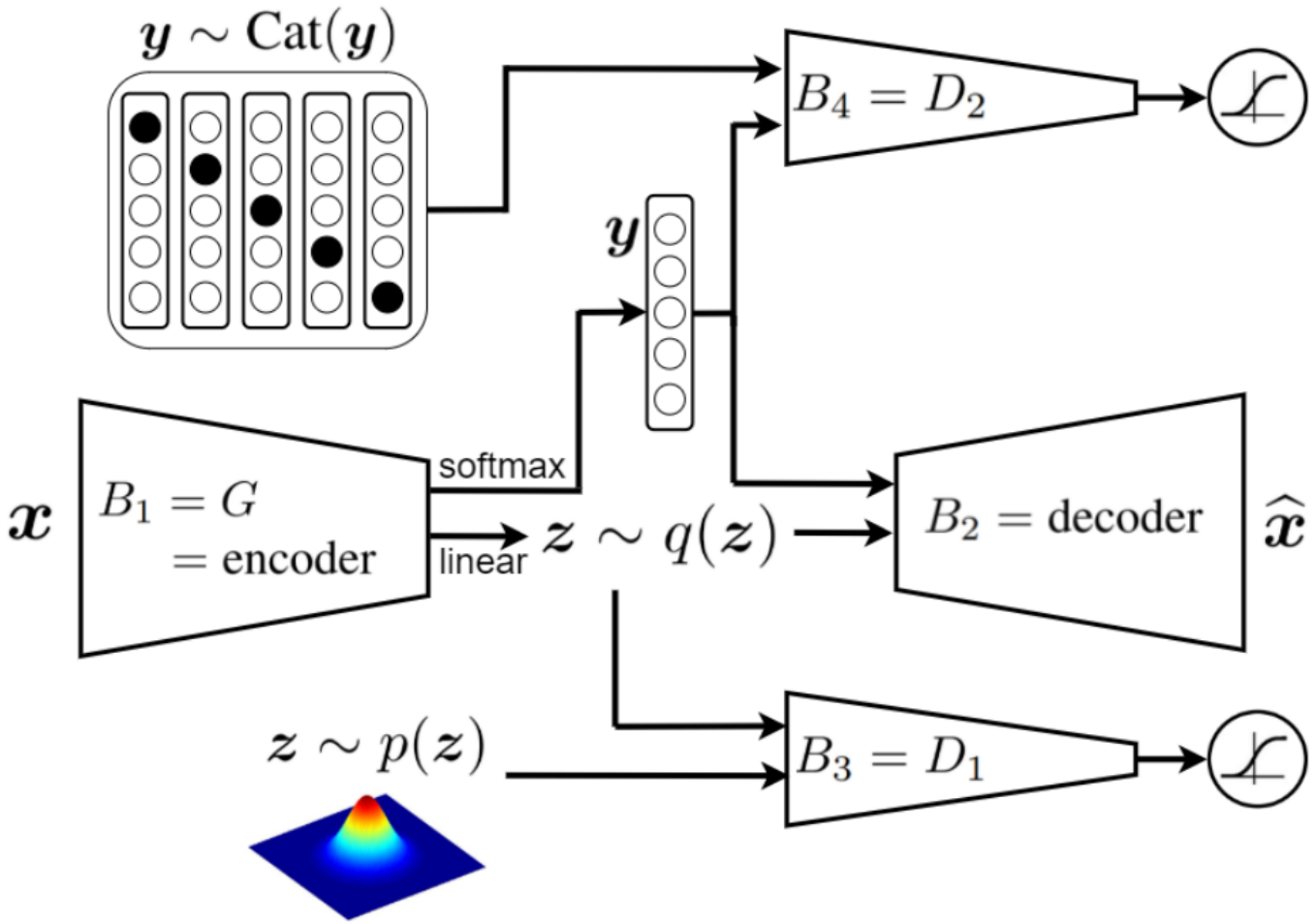
Clustering with AAE

- Assume we have c number of clusters.
- All points are unlabeled.
- The cluster indices are sampled randomly by the categorical distribution.
- The cluster labels and the latent code are both trained.

Supervised AAE



Clustering with AAE



Semi-Supervised AAE

- Consider a partially labelled dataset.

Semi-Supervised AAE

- Consider a partially labelled dataset.

- The labelled part of data has c number of classes.

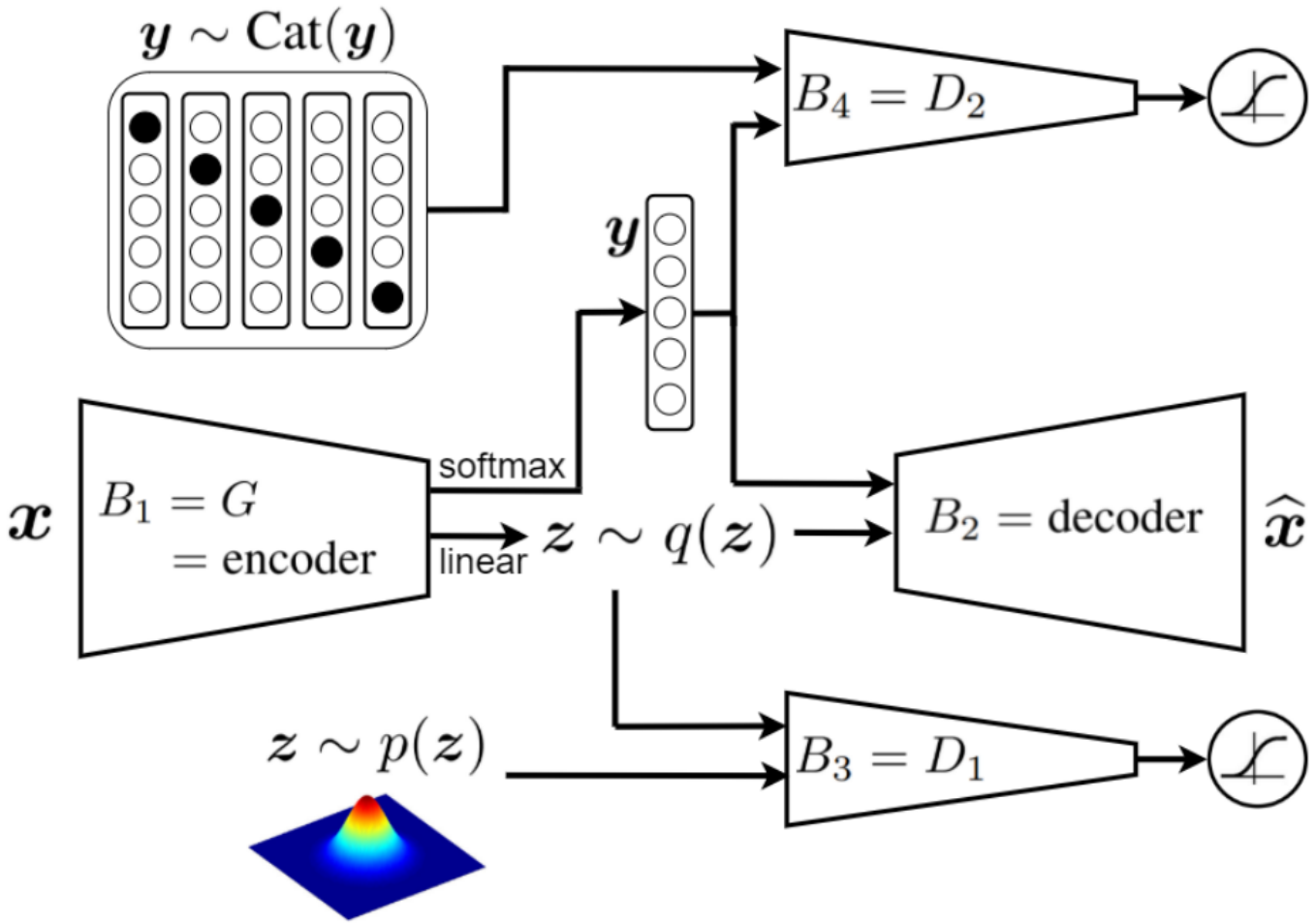
Semi-Supervised AAE

- Consider a partially labelled dataset.
- The labelled part of data has c number of classes.
- AAE can be used for semi-supervised learning with the partially labelled dataset.

Semi-Supervised AAE

- We can use the same structure for Semi-Supervised.
- But rather than the clusters, we assume we have c number of classes.
- We have a partially labelled part of the dataset.

Semi-Supervised AAE



Semi-Supervised AAE

- If the point x has a label use it. Labels are one-hot vectors.

Semi-Supervised AAE

- If the point x has a label use it. Labels are one-hot vectors.

- If the point x does not have any label, randomly sample a label $y \in \mathbb{R}^c$ from a categorical distribution, i.e., $y \sim Cat(y)$

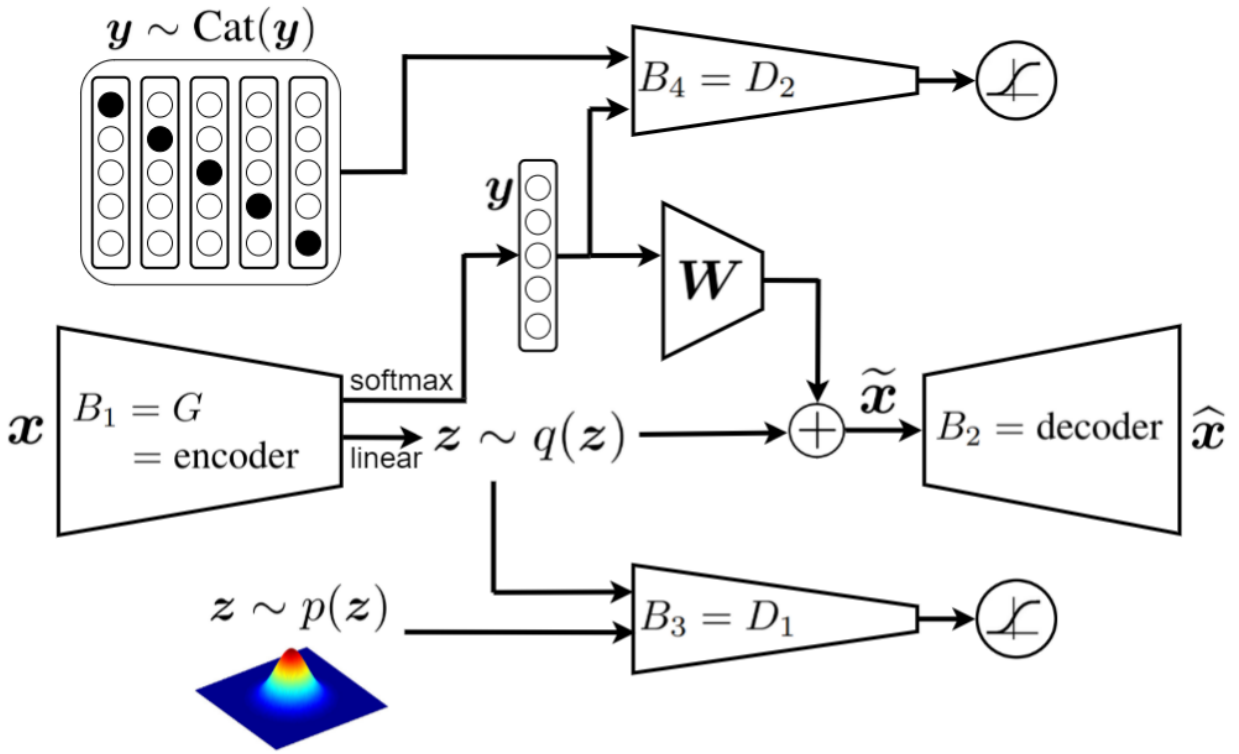
Semi-Supervised AAE

- If the point x has a label use it. Labels are one-hot vectors.
- If the point x does not have any label, randomly sample a label $y \in \mathbb{R}^c$ from a categorical distribution, i.e., $y \sim Cat(y)$
- This categorical distribution gives a one-hot encoded vector where the prior probability of every class is estimated by the proportion of the class's population to the total number of labelled points.

Dimensionality Reduction with AAE

The low-dimensional representation of $\tilde{\boldsymbol{x}} \in \mathbb{R}^p$ is obtained as:

$$\tilde{\boldsymbol{x}} = \boldsymbol{W}^T \boldsymbol{y} + \boldsymbol{z},$$



Applications

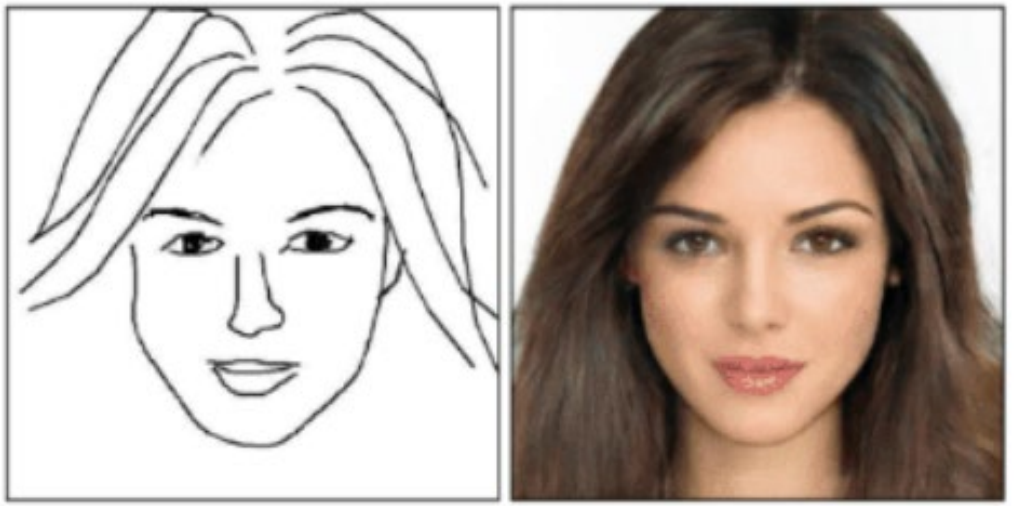
Application: Image-to-Image Translation by GAN

- Interactive GAN (iGAN) (Zhu et al., 2016)
- PatchGAN (Isola et al., 2017)
- CycleGAN (Cycle-Consistent Generative Adversarial Networks) (Zhu et al., 2017)
- Simulated GAN (SimGAN) (Shrivastava et al., 2017)
- DeepFaceDrawing (Chen et al., 2020)

PatchGAN: (a) coloring a sketch, (b) changing daylight to night darkness in image, (c) changing an aerial image to a map, (d) coloring a black-and-white image,

Transforming zebra to horse and vice versa

CycleGAN
Credit: (Zhu et al., 2017)



Generating a facial image from a facial sketch.

DeepFaceDrawing
Credit: (Chen et al., 2020)

Application: Text-to-Image Generation

- Image is generated from some descriptive caption.

Application: Text-to-Image Generation

- Image is generated from some descriptive caption.
- Reed et al., 2016a;b
- Zhang et al., 2017
- Reed et al., 2017
- Nguyen et al., 2017a
- Zhang et al., 2018

This bird is red and brown in color, with a stubby beak.

This small bird has a white breast, light grey head, and black wings and tail.

StackGAN
Credit: Zhang et al., 2017

Application: Mixing Image Characteristics

- FineGAN (Singh et al., 2019)

Application: Mixing Image Characteristics

- FineGAN (Singh et al., 2019)
  - An unsupervised GAN model which disentangles the features of the generated image to background, shape, and color/texture.

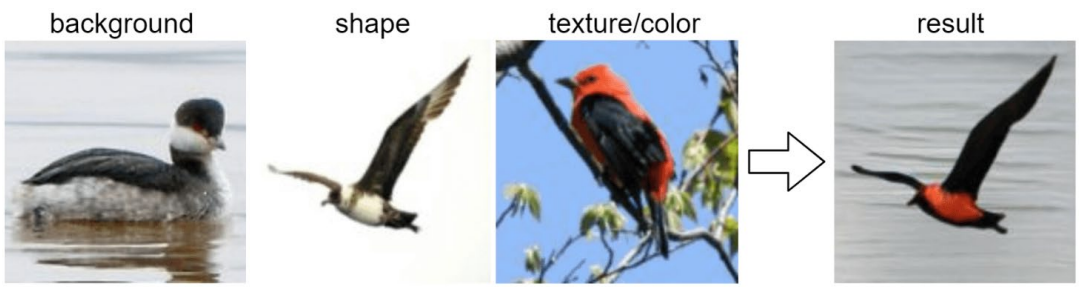
Application: Mixing Image Characteristics

- FineGAN (Singh et al., 2019)
  - An unsupervised GAN model which disentangles the features of the generated image to background, shape, and color/texture.
  - FineGAN generates an image hierarchically. It starts with generating the background.

- MixNMatch (Li et al., 2020)

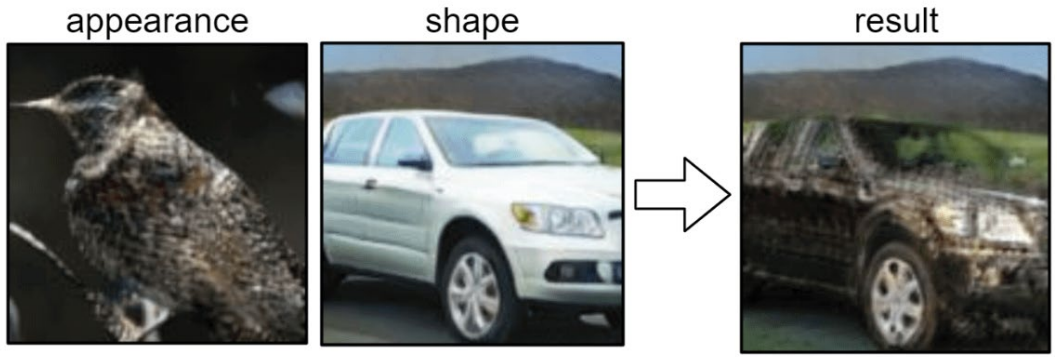
- MixNMatch (Li et al., 2020)
  - built upon FineGAN

- MixNMatch (Li et al., 2020)
  - built upon FineGAN
  - It gives the user the opportunity to choose the background, shape, and colour/texture from three pictures and it generates an image with the chosen characteristics.



generating an image by borrowing its characteristics from three images using MixNMatch

Credit: Li et al., 2020



generating an image by borrowing its characteristics from different domains using improved MixNMatch.

Credit: Ojha et al., 2021b

Other Applications

- In some structures of GAN, learned latent space is meaningful and we can do vector arithmetic in the latent space.

Other Applications

- In some structures of GAN, learned latent space is meaningful and we can do vector arithmetic in the latent space.

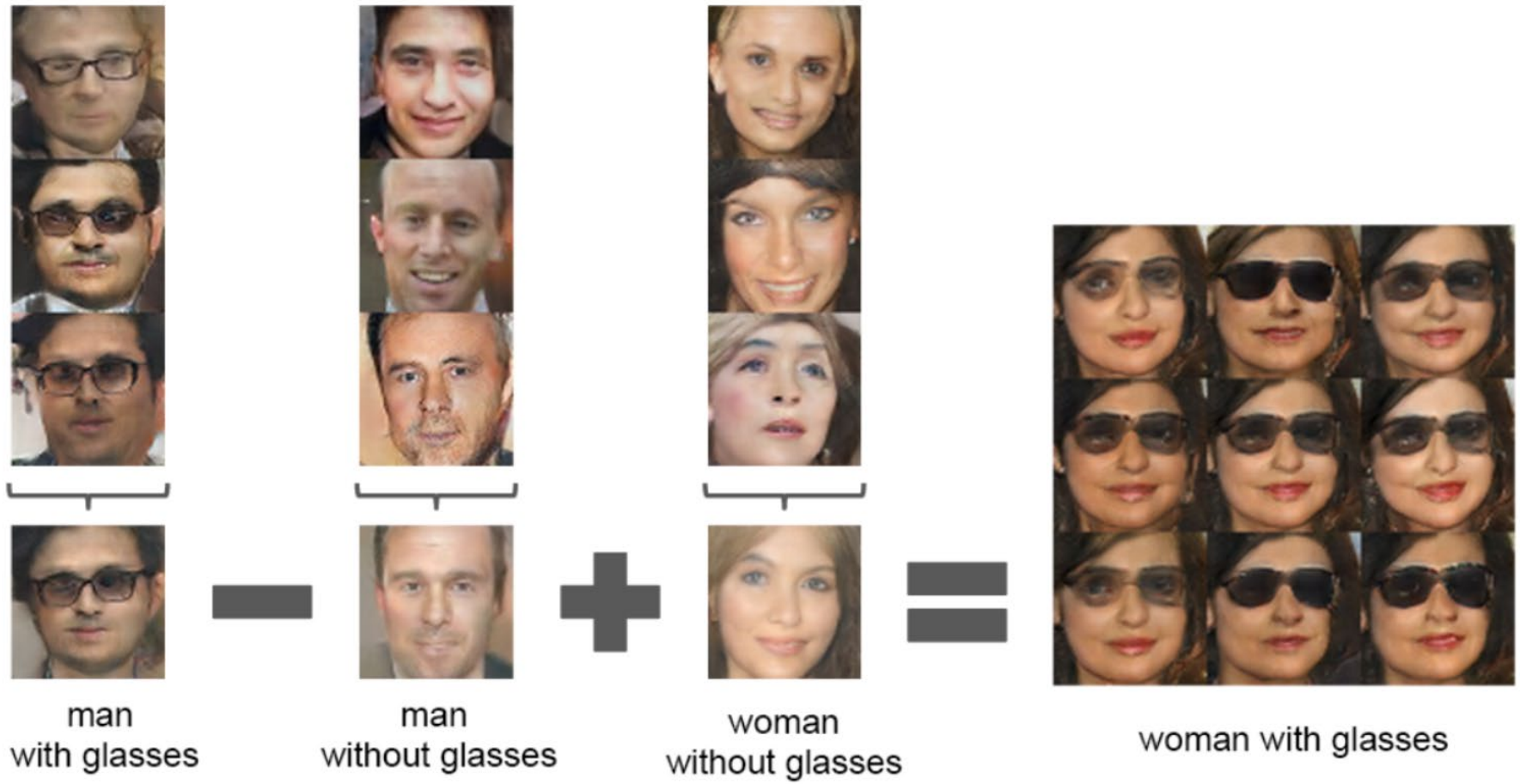
- For example in Deep Convolutional GAN (DCGAN), proposed in (Radford et al., 2016)

Other Applications

- In some structures of GAN, learned latent space is meaningful and we can do vector arithmetic in the latent space.

- For example in Deep Convolutional GAN (DCGAN), proposed in (Radford et al., 2016)

- DCGAN uses an all-convolutional network for both generator and discriminator.



Vector arithmetic in the latent space on images by DCGAN.

Credit of image : Radford et al., 2016

Other Applications

- Inpainting

Other Applications

- Inpainting
  - (Pathak et al., 2016). GAN learns to inpaint the lost part based on the available pixels in the image.

Other Applications

- Inpainting
  - (Pathak et al., 2016). GAN learns to inpaint the lost part based on the available pixels in the image.

- Medical application

Other Applications

- Inpainting
  - (Pathak et al., 2016). GAN learns to inpaint the lost part based on the available pixels in the image.

- Medical application
  - Generating histopathology images which can give insight into cancer diagnosis from pathology whole slide images (Levine et al., 2020).

Other Applications

- NLP (Li et al., 2018; Wang et al., 2019)

Other Applications

- NLP (Li et al., 2018; Wang et al., 2019)
- Speech processing (Pascual et al., 2017; Sriram et al., 2018)

Other Applications

- NLP (Li et al., 2018; Wang et al., 2019)
- Speech processing (Pascual et al., 2017; Sriram et al., 2018)
- Network embedding (Dai et al., 2018)

Other Applications

- NLP (Li et al., 2018; Wang et al., 2019)
- Speech processing (Pascual et al., 2017; Sriram et al., 2018)
- Network embedding (Dai et al., 2018)
- Logic (Nagisetty et al., 2021)

Other Applications

- NLP (Li et al., 2018; Wang et al., 2019)
- Speech processing (Pascual et al., 2017; Sriram et al., 2018)
- Network embedding (Dai et al., 2018)
- Logic (Nagisetty et al., 2021)
- Sketch retrieval (Creswell & Bharath, 2016).

References

Chen, Shu-Yu, Su, Wanchao, Gao, Lin, Xia, Shihong, and Fu, Hongbo. DeepFaceDrawing: Deep generation of face images from sketches. *ACM Transactions on Graphics (TOG)*, 39(4):72–1, 2020.

Creswell, Antonia and Bharath, Anil Anthony. Adversarial training for sketch retrieval. In European Conference on Computer Vision, pp. 798–809. Springer, 2016.

Dai, Quanyu, Li, Qiang, Tang, Jian, and Wang, Dan. Adversarial network embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Goodfellow, Ian. NIPS 2016 tutorial: Generative adversarial networks. In *Advances in neural information processing systems, Tutorial rack*, 2016.

Goodfellow, Ian, Warde-Farley, David, Mirza, Mehdi, Courville, Aaron, and Bengio, Yoshua. Maxout networks. In *International conference on machine learning*, pp. 1319–1327, 2013.

References

Isola, Phillip, Zhu, Jun-Yan, Zhou, Tinghui, and Efros, Alexei A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017

Kingma, Diederik P and Welling, Max. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014

Levine, Adrian B, Peng, Jason, Farnell, David, Nursey, Mitchell, Wang, Yiping, Naso, Julia R, Ren, Hezhen, Farahani, Hossein, Chen, Colin, Chiu, Derek, et al. Synthesis of diagnostic quality cancer pathology images by generative adversarial networks. *The Journal of pathology*, 252(2):178–188, 2020.

Li, Changliang, Su, Yixin, and Liu, Wenju. Text-to-text generative adversarial networks. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7. IEEE, 2018

Li, Yuheng, Singh, Krishna Kumar, Ojha, Utkarsh, and Lee, Yong Jae. MixNMatch: Multifactor disentanglement and encoding for conditional image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8039–8048, 2020.

References

Makhzani, Alireza, Shlens, Jonathon, Jaitly, Navdeep, Goodfellow, Ian, and Frey, Brendan. Adversarial autoencoders. *arXiv preprint arXiv*:1511.05644, 2015.

Metz, Luke, Poole, Ben, Pfau, David, and Sohl-Dickstein, Jascha. Unrolled generative adversarial networks. In *International Conference on Learning Representations*, 2017.

Mirza, Mehdi and Osindero, Simon. Conditional generative adversarial nets. *arXiv preprint arXiv*:1411.1784, 2014.

Nagisetty, Vineel, Graves, Laura, Scott, Joseph, and Ganesh, Vijay. xAI-GAN: Enhancing generative adversarial networks via explainable AI systems. 2021.

Nguyen, Anh, Clune, Jeff, Bengio, Yoshua, Dosovitskiy, Alexey, and Yosinski, Jason. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4467– 4477, 2017a

References

Ojha, Utkarsh, Singh, Krishna Kumar, and Lee, Yong Jae. Generating furry cars: Disentangling object shape & appearance across multiple domains. In *International Conference on Learning Representations*, 2021b

Pascual, Santiago, Bonafonte, Antonio, and Serra, Joan. SEGAN: Speech enhancement generative adversarial network. In *Conference of the International Speech Communication Association (INTERSPEECH)*, 2017.

Pathak, Deepak, Krahenbuhl, Philipp, Donahue, Jeff, Darrell, Trevor, and Efros, Alexei A. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2536–2544, 2016.

Radford, Alec, Metz, Luke, and Chintala, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016

Reed, Scott, Akata, Zeynep, Mohan, Santosh, Tenka, Samuel, Schiele, Bernt, and Lee, Honglak. Learning what and where to draw. *Advances in neural information processing systems*, 29:217–225, 2016a

References

Reed, Scott, Akata, Zeynep, Yan, Xinchen, Logeswaran, Lajanugen, Schiele, Bernt, and Lee, Honglak. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, pp. 1060–1069, 2016b

Reed, Scott, van den Oord, Aaron, Kalchbrenner, Nal, ¨ Bapst, Victor, Botvinick, Matt, and De Freitas, Nando. Generating interpretable images with controllable structure. In *International Conference on Learning Representations*, Workshop track, 2017.

Salimans, Tim, Goodfellow, Ian, Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved techniques for training GANs. *Advances in neural information processing systems*, 29:2234–2242, 2016.

Shrivastava, Ashish, Pfister, Tomas, Tuzel, Oncel, Susskind, Joshua, Wang, Wenda, and Webb, Russell. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2107–2116, 2017

Singh, Krishna Kumar, Ojha, Utkarsh, and Lee, Yong Jae. FineGAN: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6490–6499, 2019.

References

Sriram, Anuroop, Jun, Heewoo, Gaur, Yashesh, and Satheesh, Sanjeev. Robust speech recognition using generative adversarial networks. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5639–5643. IEEE, 2018.

Wang, William Yang, Singh, Sameer, and Li, Jiwei. Deep adversarial learning for nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pp. 1–5, 2019.

Zhang, Han, Xu, Tao, Li, Hongsheng, Zhang, Shaoting, Wang, Xiaogang, Huang, Xiaolei, and Metaxas, Dimitris N. StackGAN++: Realistic image synthesis with stacked generative adversarial networks. *IEEE transactions on pattern analysis and machine intelligence*, 41 (8):1947–1962, 2018.

Zhang, Han, Xu, Tao, Li, Hongsheng, Zhang, Shaoting, Wang, Xiaogang, Huang, Xiaolei, and Metaxas, Dimitris N. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 5907–5915, 2017

Zhu, Jun-Yan, Krahenb ¨ uhl, Philipp, Shechtman, Eli, and ¨ Efros, Alexei A. Generative visual manipulation on the natural image manifold. In *European conference on computer vision*, pp. 597–613. Springer, 2016.

Zhu, Jun-Yan, Park, Taesung, Isola, Phillip, and Efros, Alexei A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

Important papers on GAN

- Original paper

  Generative Adversarial Networks

- Tutorial

  Generative Adversarial Networks and Adversarial Autoencoders: Tutorial and Survey

- DCGANs

  Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

- Improved Techniques for Training GANs

  Improved Techniques for Training GANs

- Conditional GANs

  Conditional Generative Adversarial Nets

Important papers on GAN

- Progressively Growing GANs

  Progressive Growing of GANs for Improved Quality, Stability, and Variation

- BigGAN

  Large Scale GAN Training for High Fidelity Natural Image Synthesis

- StyleGAN

  A Style-Based Generator Architecture for Generative Adversarial Networks

- CycleGAN

  Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

- Pix2Pix

  Image-to-Image Translation with Conditional Adversarial Networks

- StackGAN

  StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks