GENERATIVE ADVERSARIAL NETWORKS

What are GANs?

$\underbrace{\text{Generative}}_{}$ Adversarial Networks

Generative Models
We try to learn the underlying the distribution
from which our dataset comes from.

What are GANs?

$$\text{Generative } \underbrace{\text{Adversarial}} \text{ Networks}$$

Adversarial Training
GANS are made up of two competing networks (adversaries) that are trying beat each other.

What are GANs?

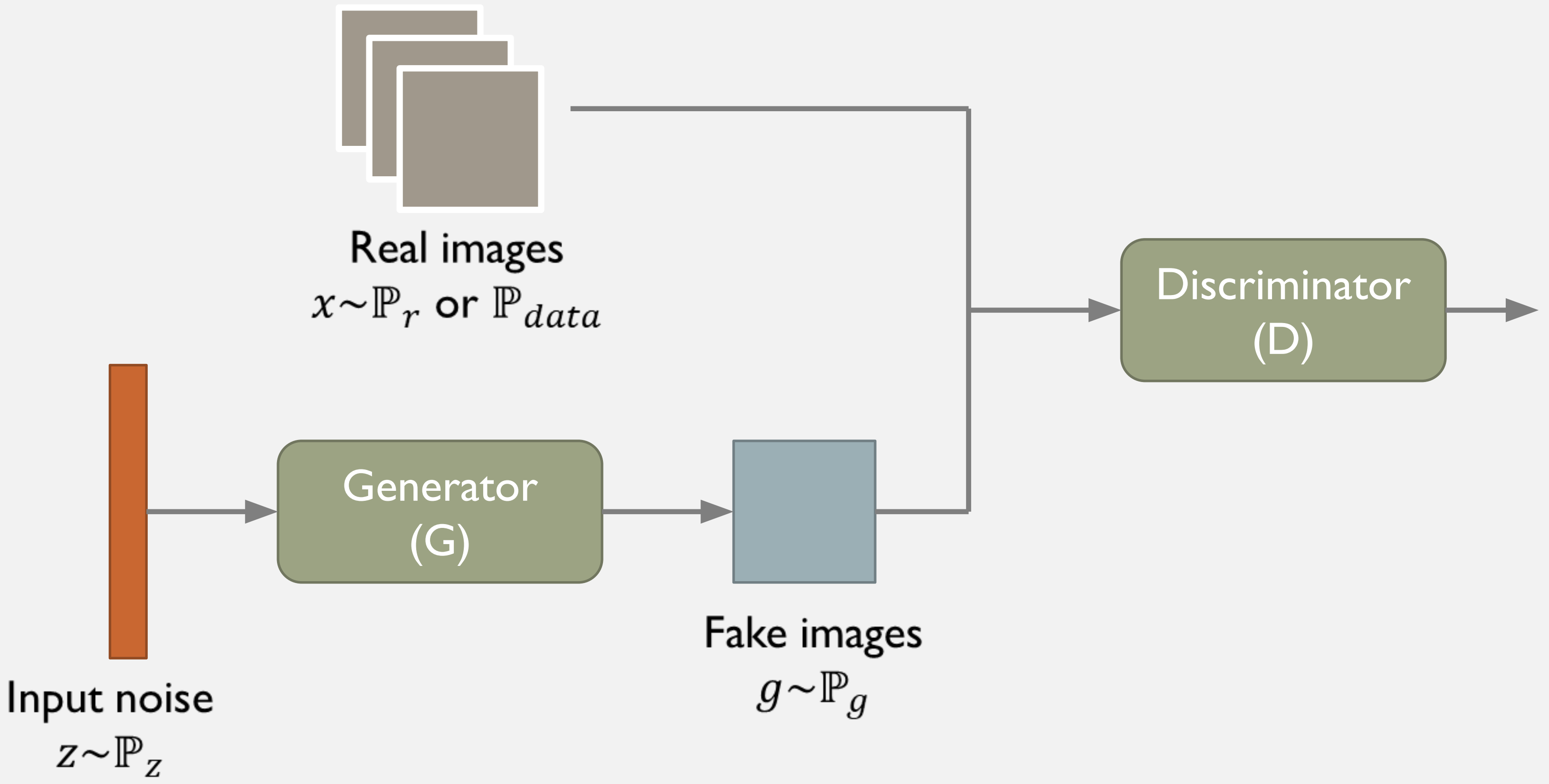
Generative Adversarial $\underbrace{\text{Networks}}_{\text{Neural Networks}}$

GENERATIVE VS DISCRIMINATIVE

Goal of ML algorithms: "predict the probability of a sample belonging to class y, based on a set of features x". In other words: $p(y \mid x)$.

- **Discriminative** algorithms try to solve this by correlating features to labels.
  E.g. logistic regression, SVM
- **Generative** algorithms try to learn the distribution of each class.
  "How did we get x?" $\rightarrow p(x \mid y)$
  E.g. Naïve Bayes

GENERATIVE ADVERSARIAL NETWORKS



GENERATOR VS DISCRIMINATOR

- Generator’s goal: produce “fake” data that will trick the discriminator
- Discriminator’s goal: distinguish between real and fake data

- The two compete in a zero-sum game, which will hopefully improve the performance of both models.

- Very hard to train. Especially during early stages.

IN MORE DETAILS…

Original conception (Goodfellow et al.):

- D and G play a two-player minimax game with value function $V(D, G)$:

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

p_{data}: real data distribution

$D(x)$: probability that x came from the data rather than p_g

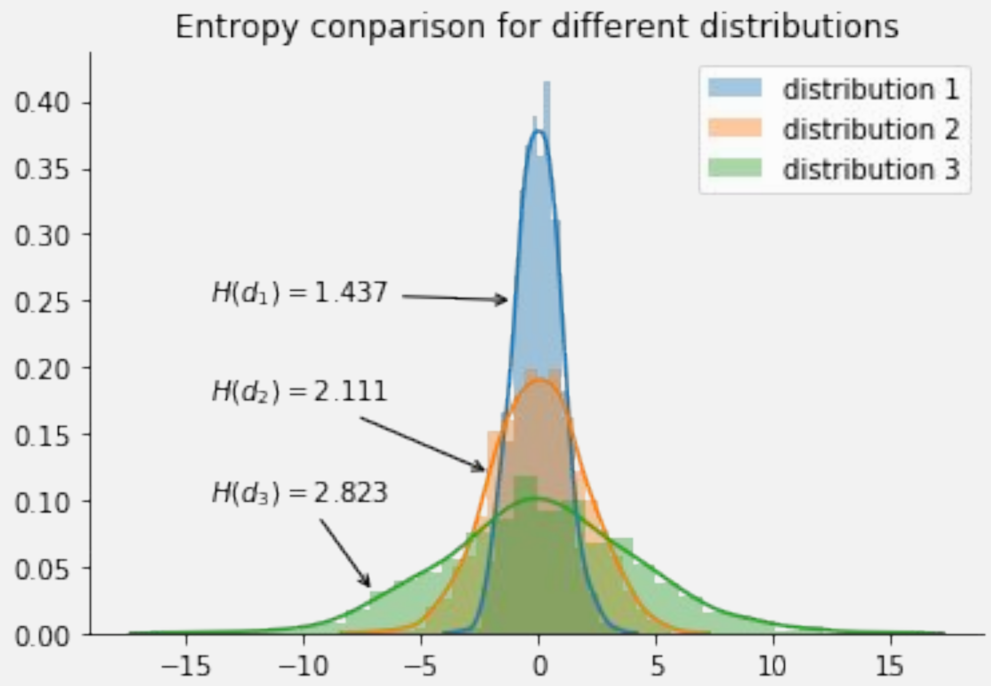
p_z: input noise prior

- Discriminator's loss is essentially a **cross-entropy** loss.

SHANNON ENTROPY

- $$H(P) = \sum_x P(x) \cdot \log\left(\frac{1}{P(x)}\right)$$
  $$H(P) = -\mathbb{E}_{x \sim p}[\log P(x)]$$

- Bits needed to encode "message" x with probability $\mathrm{P}(x)$.
- Entropy H: sum over all messages.
- Measure of "**impurity**" in data.
- **Low entropy → data more uniform.**



CROSS-ENTROPY

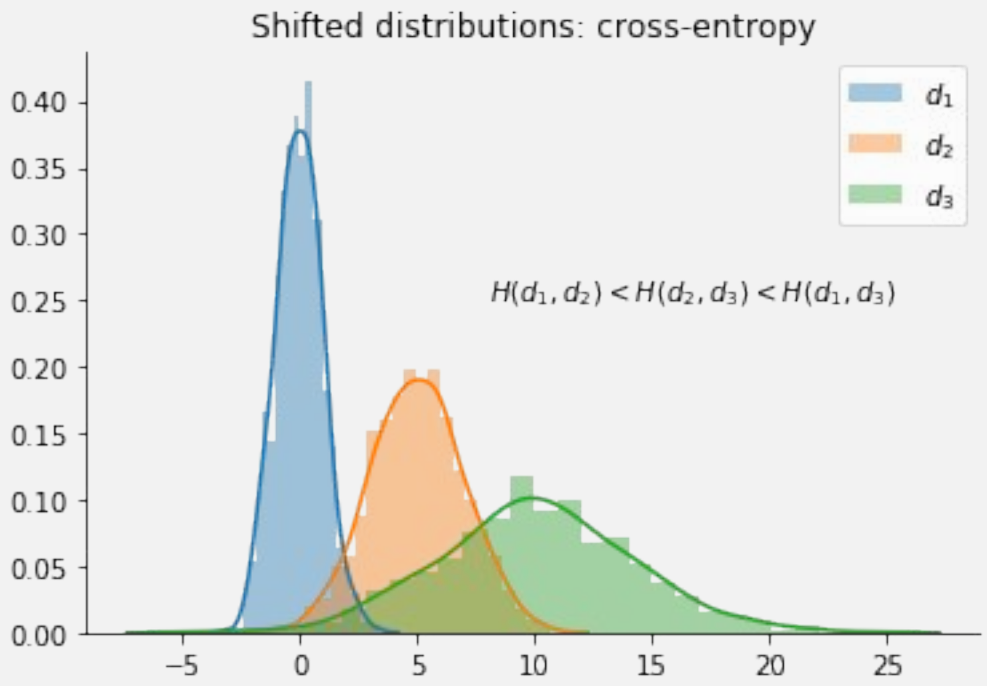
- $$H(y, \hat{y}) = \sum_i y_i \cdot \log\left(\frac{1}{\hat{y}_i}\right)$$
$$H(y, \hat{y}) = -\mathbb{E}_y[log\ \hat{y}]$$

- Bits needed to encode "message" i from distribution y with "symbols" from distribution $\hat{y}$.

- Measure "**relatedness**" between y and $\hat{y}$.

- **High cross-entropy $\rightarrow$ $(y, \hat{y})$ unrelated.**

CROSS-ENTROPY INSIGHTS

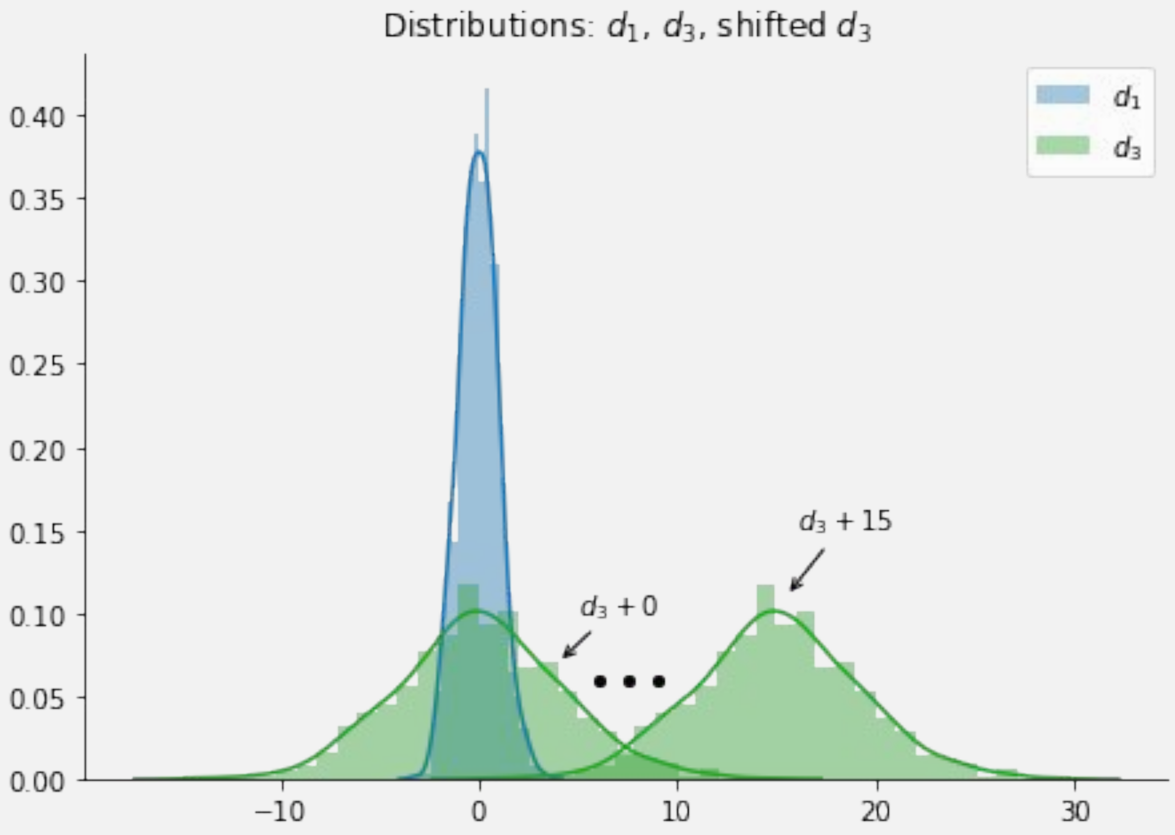


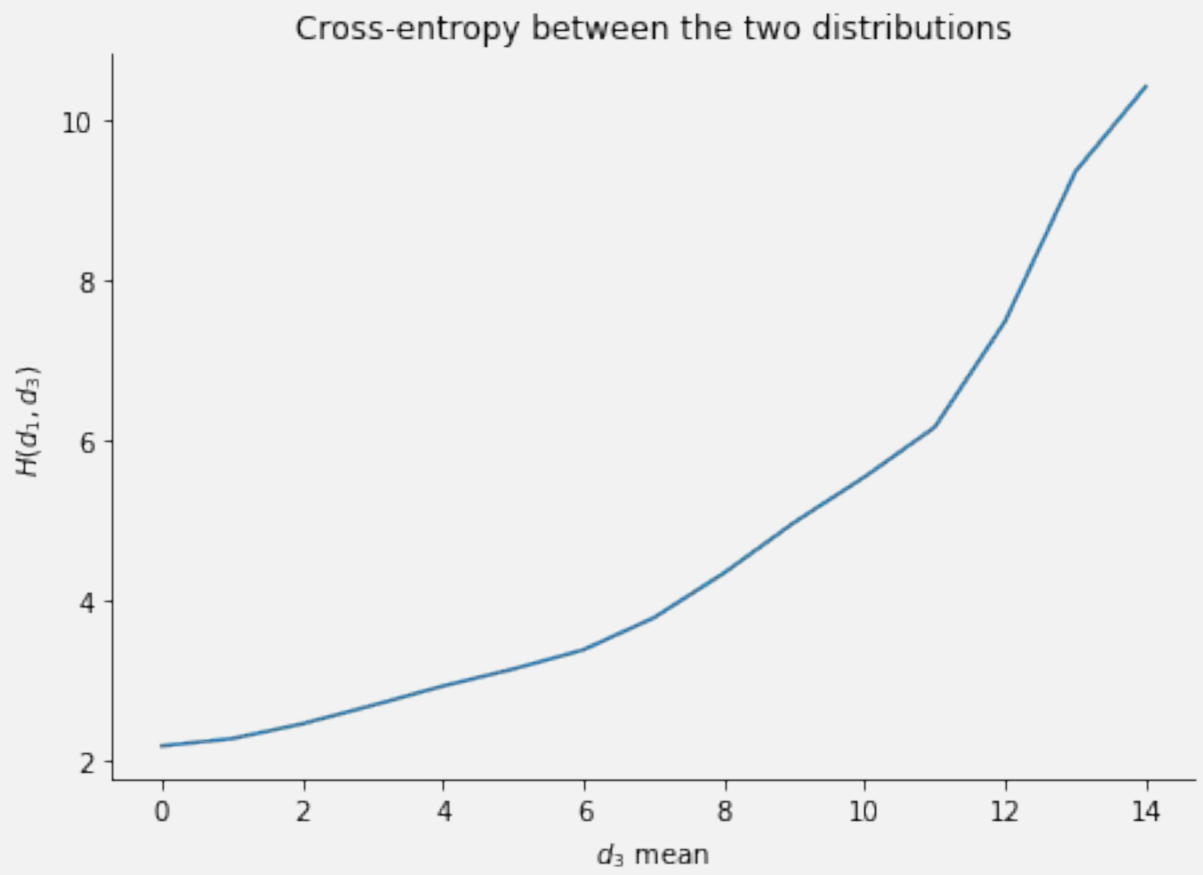
	1.437	1.748	2.184
	2.865	2.087	2.311
	5.583	3.495	2.577

$$H(d_1, d_1) = H(d_1) = 1.437$$
$$H(d_2, d_2) = H(d_2) = 2.087$$
$$H(d_3, d_3) = H(d_3) = 2.577$$

$$H(d_1, d_2) \neq H(d_2, d_1)$$
$$H(d_1, d_3) \neq H(d_3, d_1)$$
$$\dots$$

CROSS-ENTROPY INSIGHTS





KULLBACK-LEIBLER (KL) DIVERGENCE

- Measures **difference** between two probability distributions $(y, \hat{y})$.
- Essentially the **difference from entropy to cross-entropy**.

$$KL(y \,||\, \hat{y}) = H(y, \hat{y}) \; - H(y)$$
$$KL(y \,||\, \hat{y}) = \sum_i y_i \cdot \log\left(\frac{y_i}{\widehat{y}_i}\right)$$

- How many more bits are required to encode messages from distribution y if we use symbols from $\hat{y}$.
- **Non symmetric**: $KL(y \,||\, \hat{y}) \; \neq \; KL(\hat{y} \,||\, y)$. Not a distance metric.
- Always takes **positive** values: $H(y, \hat{y}) \geq H(y)$.
- The **higher** the value, the more two distributions **differ**.

JENSEN-SHANNON (JS) DIVERGENCE

- A derivative of KL divergence, that **can be used as a distance metric**.

$$\mathrm{JS}(\mathrm{y} \,||\, \hat{\mathrm{y}}) = \frac{1}{2}\mathrm{KL}(\mathrm{y} \,||\, \mathrm{m}) + \frac{1}{2}\mathrm{KL}(\hat{\mathrm{y}} \,||\, \mathrm{m})$$

$$\text{where } \mathrm{m} = \frac{1}{2}(\mathrm{y} + \hat{\mathrm{y}})$$

- **Symmetric**.
- **Positive**.
- The **higher** the value the **larger** the distance between y, ŷ.

GAN LOSSES

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

- Discriminator loss (sum two losses):

  - Loss on "real" images:
  $$-\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)]$$
  - Loss on "fake" images:
  $$-\mathbb{E}_{z \sim p_z(z)}[1 - \log D(G(z))]$$

- Generator loss (**only relevant for "fake" images**):

$$\mathbb{E}_{z \sim p_z(z)}[1 - \log D(G(z))]$$

TRAINING

- Initially:
  - Generator does not know how to produce realistic images.
  - Discriminator does not know how to separate the two.

- As training progresses:
  - G: starts to learn the distribution of real images.
  - D: becomes better at distinguishing real from fake.

- After successful training:
  - G: has learned to produce realistic images.
  - D: can’t distinguish from the two .

TRAINING DETAILS

- When you train the discriminator, **hold the generator values constant**; and when you train the generator, **hold the discriminator constant**. Each should train against a static adversary.

- Each side of the GAN can overpower the other.

  - If the discriminator is too good, it will return values so close to 0 or 1 that the generator will struggle to read the gradient.

  - If the generator is too good, it will persistently exploit weaknesses in the discriminator that lead to false negatives.

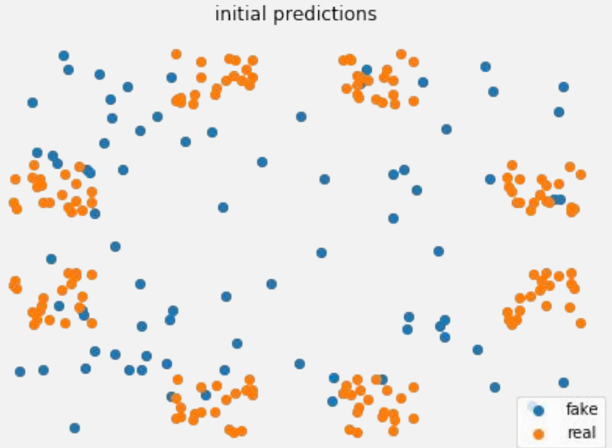
GAN TRAINING ISSUES

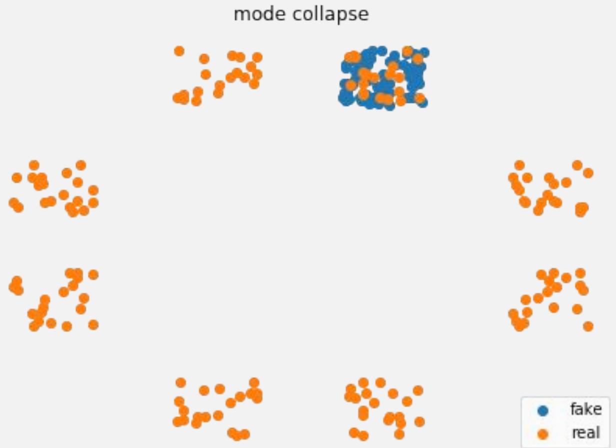
- $$min_G max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[1 - \log D(G(z))]$$

- Hard to achieve Nash Equilibrium (each player plays independently; cannot ensure convergence).
- **Vanishing Gradient**:
  As D becomes better (makes more confident predictions), G loss approaches 0 → no gradient.
  Perfect discriminator exists (proven).
- If D behaves badly, G cannot gather accurate feedback.
- JS divergence fails to provide a meaningful metric when the two distributions are disjointed.
- Highly sensitive to hyperparameter selection.
- **Mode collapse**:
  Multimodal distribution "modes" collapse into a single "mode".

MODE COLLAPSE

- Consider the extreme case that G is trained without updating D. The generated images will converge towards the optimal image x^*, which maximizes D's uncertainty: $x^* = argmax_x D(x)$.
- The mode collapses to a single point, the gradient associated with z becomes 0.





- D with then push to exploit the next best mode. G will soon follow and **the two will be stuck in an overfit** state where both will try to exploit the other's short term weaknesses.
- D is more likely to overfit than G.

SOME WORKAROUNDS TO GAN'S TRAINING ISSUES…

- **Minibatch discrimination:**
  - Separate real/fake into different batches.
  - Compute similarity within batch.
  - Feed this to the discriminator.
  - Mode drops -> similarity increases.
  - D can spot fake images from this parameter and penalize G.
- **One-sided label smoothing:**
  - Penalize D when prediction for real is high. E.g. $D(x) > 0.9$.
  - Avoid overconfidence.
- **Add noise to stabilize the model.**
- **Alternative generator cost function:**

  Replace $\mathbb{E}_{z\sim p_z(z)}[1 - \log D(G(z))]$ with $-\mathbb{E}_{z\sim p_z(z)}[\log D(G(z))]$
  - More resistant to vanishing gradients

WASSERSTEIN GAN (1/4)

- Most of GAN's training issues emanate from the **cost function**.
- Replace JS with "Wasserstein distance".
- **Distance metric between distributions**.
- How much "earth" we need to move to reach from one distribution to the other.
- Continuous everywhere and differentiable almost everywhere (not true for KL, JS).

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\, ||x - y\,||\,]$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ is the set of all joint distributions (x, y) whose marginals are $\mathbb{P}_r, \mathbb{P}_g$.

WASSERSTEIN GAN (2/4)

- The previous cost function depends on the optimal "transport plan" γ, which is tricky to compute.
- After some assumptions and by using the Kantorovich-Rubinstein duality, we can simplify the calculation to:

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = sup_{||f||_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

where f is a 1-Lipschitz function.

- In practice we build a deep neural network to **approximate** this function ☺. This new network is very similar to the Discriminator, which we will call the **"critic"** to reflect its new role.

WASSERSTEIN GAN (3/4)

- New gradients:
  - **Critic**:

$$\nabla_w \frac{1}{m} \sum_{i=1}^{m} \left[f\left(x^{(i)}\right) - f\left(G\left(z^{(i)}\right)\right) \right]$$

  - **Generator**:

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \left[-f\left(G\left(z^{(i)}\right)\right) \right]$$

- Unfortunately f needs to be a 1-Lipschitz function. To enforce this constraint the authors simply **clipped** f if it exceeded an arbitrary range.

WASSERSTEIN GAN (4/4)

Pros:

- Much more **meaningful loss** (correlation between loss value and image quality).
- Increased training **stability**:
  - Less likely to collapse.
  - **Generator can still learn when critic performs well**.
  - Allows training to **optimality**.

Cons (all have to do with weight clipping):

- Clipping parameter is very sensitive.
- Decreases model capacity.
- Slows down training.

GRADIENT PENALTY (WGAN-GP)

- Done instead of weight clipping.
- A 1-Lipshitz function has a gradient norm of 1 almost everywhere under $\mathbb{P}_r, \mathbb{P}_\theta$.
- The model is **penalized** if the gradient norm strays away from 1.

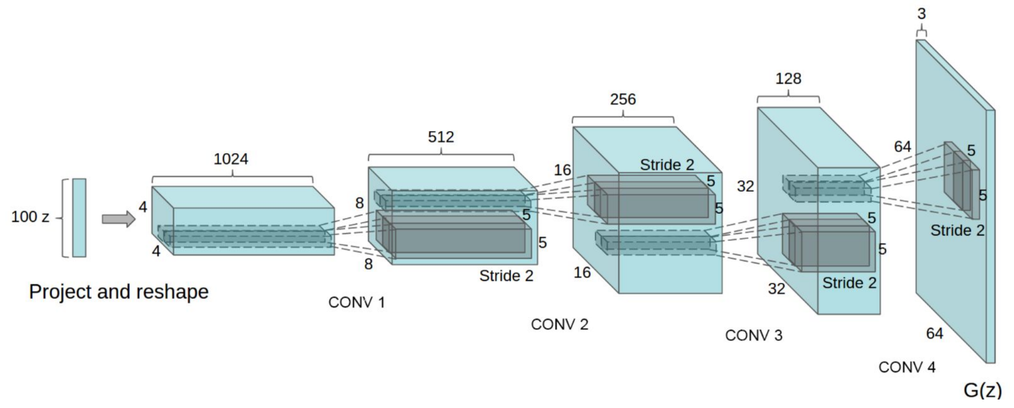
$$L = \underbrace{\mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x\sim\mathbb{P}_r}[D(x)]}_{\text{critic loss}} + \lambda \underbrace{\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}} \mathrm{D}(\hat{x})||_2 - 1)^2]}_{\text{gradient penalty}}$$

where $\hat{x}$ is sampled from $\tilde{x}$ and x.

- **Avoid** Batch Normalization (creates correlation between samples within batch, impacts the effectiveness of GP).
- Makes more **stable** training and will **converge** better and faster.
- Allows for the use of more **complex** models as G, D (e.g. ResNet architecture).

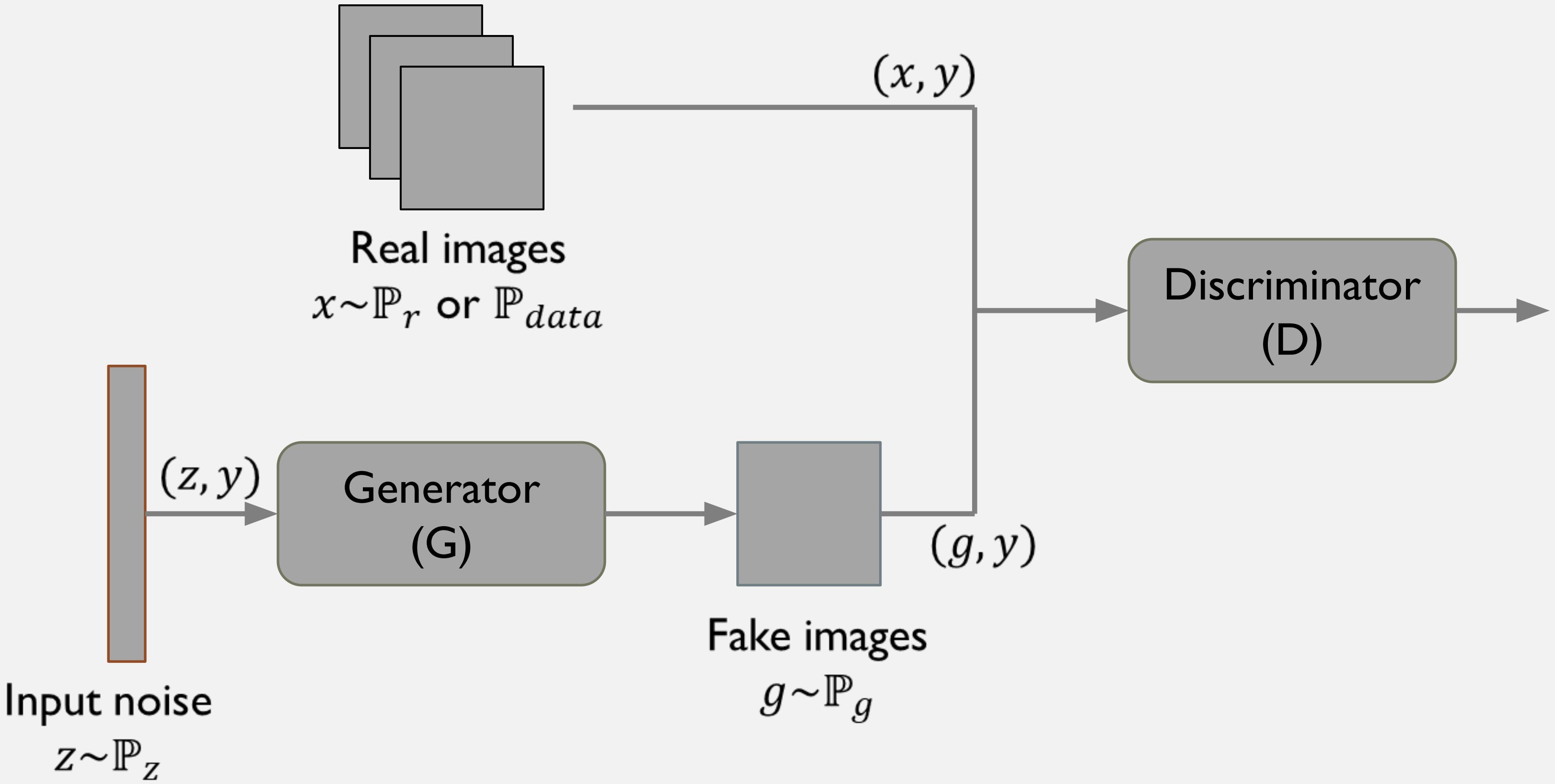
DEEP CONVOLUTIONAL GAN (DCGAN)

- Original GANs were MLP models.
- Can build GANs with **convolutional layers**.
- Guidelines:
  - Replace pooling layers (which destroy spatial information) with convolutional stride.
  - Use transposed convolutions for up-sampling.
  - Eliminate FC layers.
  - Use Batch Normalization.



CONDITIONAL GAN (CGAN)

- Both generator and discriminator also accept the image labels as their input.



LEAST SQUARES GAN (LSGAN)

- Defines a new cost function which help get a smoother gradient everywhere.

$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x}) - b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})) - a)^2\right]$$

$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})) - c)^2\right],$$

ENERGY BASED GAN (EBGAN) AND BOUNDARY EQUILIBRIUM GAN (BEGAN)

- Both replace D with an **AutoEncoder**.

- **EBGAN**:
  - Measure reconstruction error (MSE)
  - Motivates GAN to have broader goals and avoid greedy optimization.
  - Ensures GAN generates images with features found in natural images.

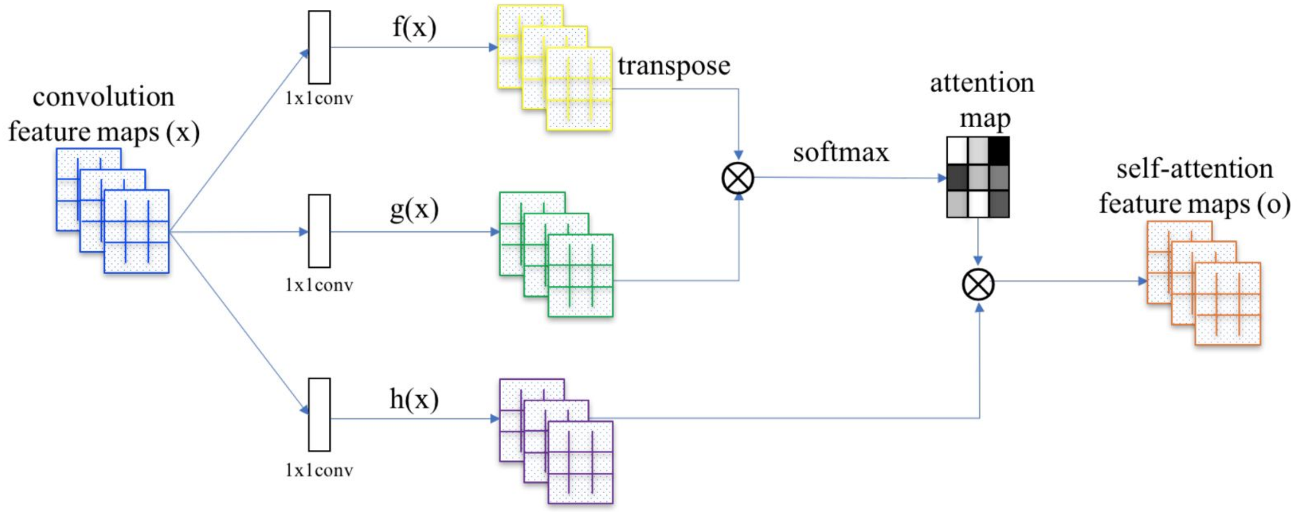
- **BEGAN**:
  - Wasserstein distance
  - Loss function has two goals (hyperparameter γ to balance them):
    - Good critic (helps diversity).
    - Good reconstructor (helps image quality).

SPECTRAL NORMALIZATION

- Popular discriminator regularizer.
- Can enforce Lipschitz constraint.

SELF-ATTENTIONAL GAN (SEGAN)

- **State-of-the-art!**
- Refines each spatial location with an extra term computed by the self-attentional mechanism.
- Can be used on both G and D.
- Spectral Normalization is used to stabilize the GAN.



EVALUATION METRICS (1/2)

Inception Score:

- Entropy can be viewed as randomness.
- When training a classifier we want $p(y \mid x)$ to be **highly predictable**.
- Use an **Inception** network to classify the images and predict $p(y \mid x)$. This reflects on the quality of the images.
- If the generated images are diverse the data distribution of y should be **uniform**:

$$p(y) = \int_z p(y \mid x = G(z))dz$$

- Inception score is computed as:

High $p(y \mid x)$ requires high quality images

$$IS(G) = \exp\left(\mathbb{E}_{x \sim p_g}[KL(p(y \mid x) \,||\, p(y))]\right)$$

Large $p(y)$ requires high diversity between classes

EVALUATION METRICS (2/2)

Fréchet Inception Distance (FID)

- Use **Inception** Network to **extract features from an intermediate layer**.
- Model these with a multivariate Gaussian distribution with mean μ and covariance Σ.
- FID between real images x and generated g images is computed as:

$$FID(x,g) = ||\mu_x - \mu_g||_2^2 + Tr\left(\Sigma_{\mathrm{x}} + \Sigma_{\mathrm{g}} - 2(\Sigma_{\mathrm{x}}\Sigma_g)^{\frac{1}{2}}\right)$$

- **Lower** FID → better image quality and diversity.
- Sensitive to mode collapse (increases with missing modes).
- More robust to noise than IS.
- Better measure for image diversity.