CS182/282A: Designing, Visualizing and Understanding Deep Neural Networks

John Canny

Spring 2020

Lecture 17: Generative Adversarial Networks

Slides adapted from Forrest Huang, Fei-fei Li, Justin Johnson, Serena Yeung, Ian Goodfellow et al.

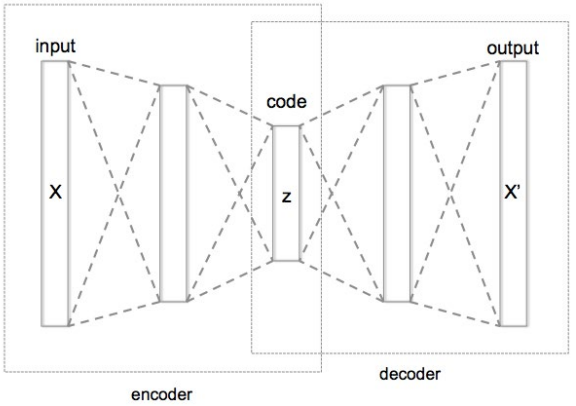
Last Time: Variational Auto-Encoders

Auto-encoders transform each input x into a "code" variable z, and then back to a synthetic input x' that should be as close as possible to x. The dimension of z is usually much lower than x, and forms an "information bottleneck" to x'.

Encoder: Very similar to, or could be exactly a neural classifier. Can be thought of as a data compressor.

Code: A representation of the input designed to support reconstruction. Could be the set of class labels, but also often encodes within-class variability.

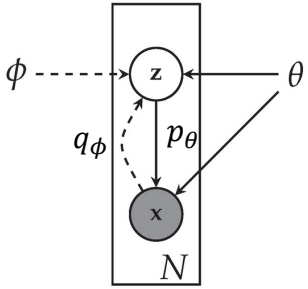
Decoder: Is a conditional synthetic input generator (a generative model).



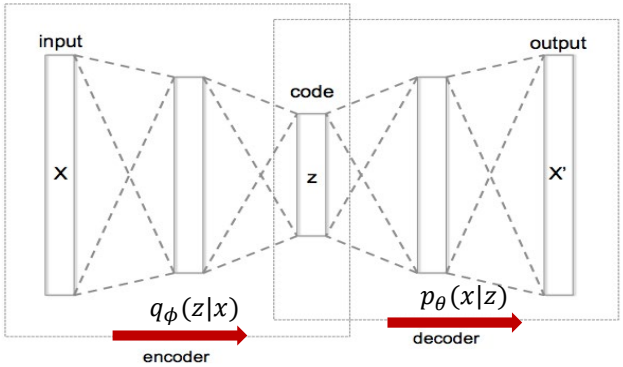
Last Time: Variational Auto-Encoders

The solid lines show the actual model $p_\theta(x|z)$, where values of x are computed by a θ-parametrized deep network from x.

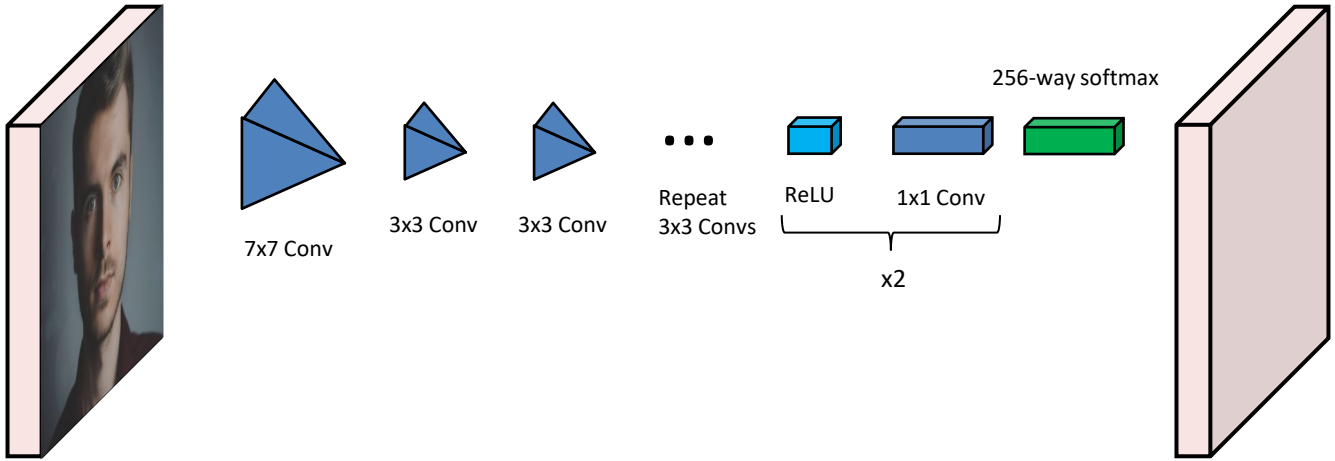
The dotted lines show the *variational approximation* $q_\phi(z|x)$ to the inverse model $p(z|x)$.



This can be viewed as a probabilistic auto-encoder:



Last Time: Pixel CNN/RNN

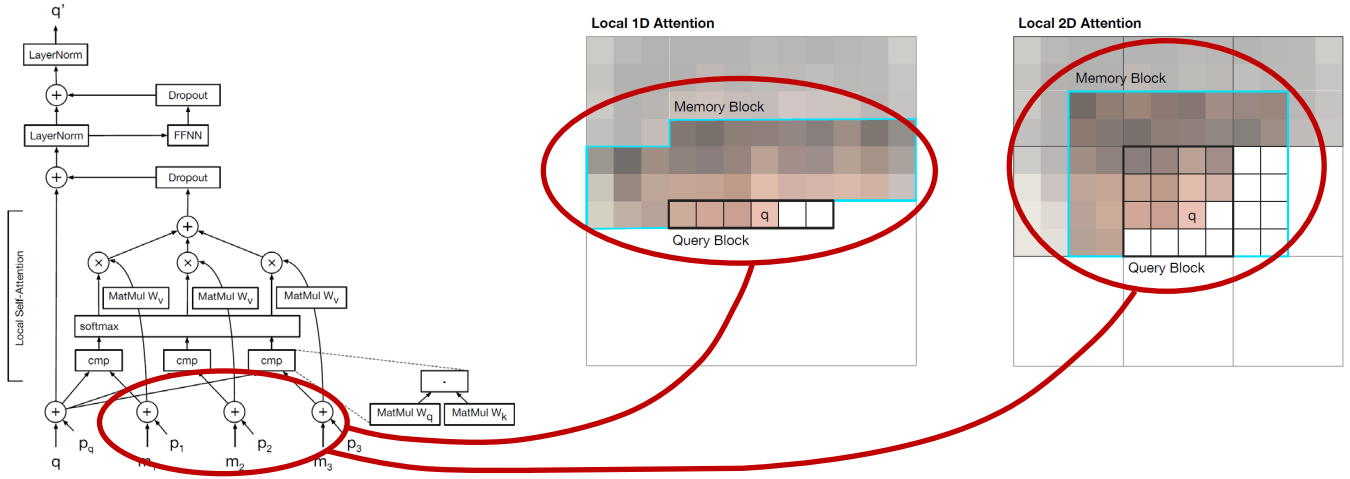


Last Time: Image Transformer

Like the Row-LSTM, it generates blocks of pixels at a time.

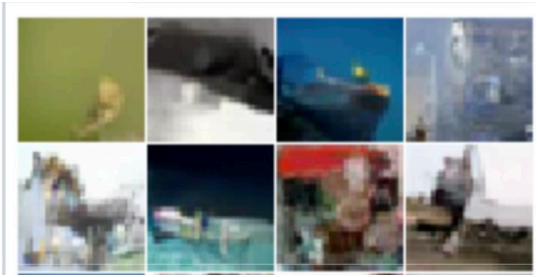
The query blocks show the block of pixels to be generated

The memory blocks show the context from the previous layer used to generate it.



So far…

- PixelRNN, VAEs tries to directly approximate P(x) with naïve measures
- They produces images that are not realistic even when these losses are very low.



- Suppose we could use a perceptual measure for generation: **have a human compare them**?

Weaknesses of VAEs

- Tries to maximize an ***approximation to*** the likelihood:

$$\sum_{i=1}^{N} \log p\,(x^i)$$

- Uses an ELBO (Evidence Lower Bound) equivalent to:

$$E_{z\sim q_\phi(z|x)}\left[\log p_\theta(x,z\,) - \log q_\phi(z|x)\right]$$
$$= KL\left(q_\phi(z|x)||p_\theta(x,z)\right)$$

  i.e. it compares densities ***in the latent z space***.

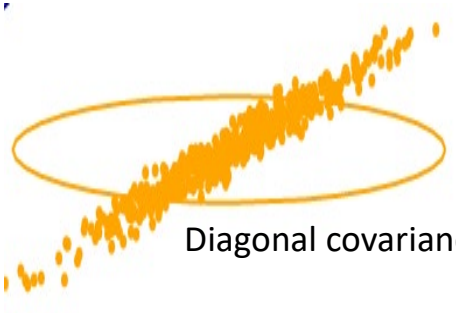
- Needs a **density estimate** for $q_\phi(z|x)$:
  most practical estimators assume diagonal covariance:

$$z = g(x,\epsilon) = g_0(x) + g_1(x) \circ \epsilon$$

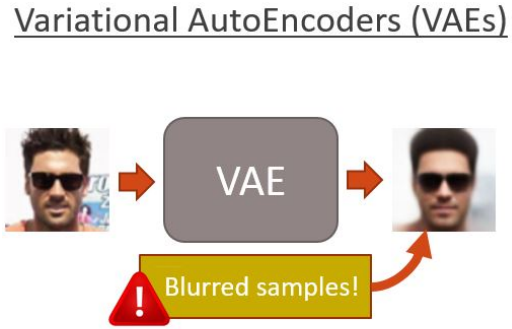
Weakness of VAEs

- Needs a density estimate for $q_\phi(z|x)$:
  most practical estimators assume diagonal covariance:
  $$z = g(x, \epsilon) = g_0(x) + g_1(x) \circ \epsilon$$
  but the output of a conv- or transposed conv-net is highly correlated. So the approximation is poor:

Actual z samples



Diagonal covariance $g(x, \epsilon)$



Ideally we would…

- Avoid approximations without error bounds (ELBOs), or such that errors can be made arbitrarily small.

- Compare the densities of real and synthetic images ***in the image space***.

- Use a generator $x = g(z)$ that is ***completely general***, i.e. doesn't have to provide density estimates for $p(x|z)$.

Jensen-Shannon Divergence

Let $x = g_\theta(z)$ be our generator, write $p_\theta(x|z)$ as the generated density, and $p_\theta(x) = E_z[p_\theta(x|z)]$ as the marginal.

Let $p_{re}(x)$ be the density of true images.

Define the ***Jensen-Shannon Divergence*** between $p_{re}(x)$ and $p_\theta(x)$ as:

$$JSD\big(p_{re}(x)||p_\theta(x)\big) = \frac{1}{2}KL\big(p_{re}(x)||m(x)\big) + \frac{1}{2}KL\big(p_\theta(x)||m(x)\big)$$

$$= \frac{1}{2}E_{x\sim p_{re}(x)}\left[\log\frac{p_{re}(x)}{m(x)}\right] + \frac{1}{2}E_{x\sim p_\theta(x)}\left[\log\frac{p_\theta(x)}{m(x)}\right]$$

where $m(x) = \frac{1}{2}\big(p_{re}(x) + p_\theta(x)\big)$

We would like to minimize this divergence, i.e. make the distributions of real and synthetic images as close as possible.

Jensen-Shannon Divergence

$$JSD\left(p_{re}(x)||p_{\theta}(x)\right) = \frac{1}{2}E_{x\sim p_{re}(x)}\left[\log\frac{p_{re}(x)}{m(x)}\right] + \frac{1}{2}E_{x\sim p_{\theta}(x)}\left[\log\frac{p_{\theta}(x)}{m(x)}\right]$$

We can approximate expected values by sampling

Jensen-Shannon Divergence

$$JSD\left(p_{re}(x)||p_{\theta}(x)\right) = \frac{1}{2}E_{x\sim p_{re}(x)}\left[\log\frac{p_{re}(x)}{m(x)}\right] + \frac{1}{2}E_{x\sim p_{\theta}(x)}\left[\log\frac{p_{\theta}(x)}{m(x)}\right]$$

But we cant estimate these densities

Discriminator Approximation

What happens though if we define an approximate density ratio (deep nets are good at estimating those)?:

$$d_\phi(x) \approx \frac{p_{re}(x)}{2m(x)} = \frac{p_{re}(x)}{p_{re}(x) + p_\theta(x)}$$

then:

Simplify by removing constants

$$JSD\left(p_{re}(x)||p_\theta(x)\right)$$
$$\approx \frac{1}{2} E_{x\sim p_{re}(x)}\left[\log d_\phi(x)\right] + \frac{1}{2} E_{x\sim p_\theta(x)}\left[\log\left(1 - d_\phi(x)\right)\right] + \log Z$$

then minimizing this quantity should approximately minimize the original JSD. But how do we train $d_\phi(x)$?

Training the Discriminator

The divergence to be minimized is:

$$\approx E_{x\sim p_{re}(x)}\left[\log d_\phi(x)\right] + E_{x\sim p_\theta(x)}\left[\log\left(1-d_\phi(x)\right)\right]$$

But how do we train $d_\phi(x)$?

Minimize its cross-entropy loss at labeling samples from $p_{re}(x)$ and $p_\theta(x)$, which is

$$-E_{x\sim p_{re}(x)}\left[\log d_\phi(x)\right] - E_{x\sim p_\theta(x)}\left[\log\left(1-d_\phi(x)\right)\right]$$

Wait, didn't we ***just*** see this formula ?! (without the minus signs)

We want to minimize the divergence of the distribution $p_\theta(x)$ from $p_{re}(x)$, but to do so we need to optimize $d_\phi(x)$ in the inner loop…

Classifiers as Density Estimators

If we minimize the cross-entropy loss:

$$-E_{x\sim p_{re}(x)}\left[\log d_\phi(x)\right] - E_{x\sim p_\theta(x)}\left[\log\left(1-d_\phi(x)\right)\right]$$

The gradient is

$$-\int_x \left(\frac{p_{re}(x)}{d_\phi(x)} - \frac{p_\theta(x)}{1-d_\phi(x)}\right)\nabla_\phi d_\phi(x)dx$$

Which vanishes when

$$d_\phi(x)\left(p_\theta(x)+p_{re}(x)\right) - p_{re} = 0$$

or

$$d_\phi(x) = \frac{p_{re}(x)}{p_{re}(x)+p_\theta(x)}$$

Adversarial Training

The adversarial optimization problem is:

$$\min_{\theta} \max_{\phi} \left(E_{x \sim p_{re}(x)} \left[\log d_{\phi}(x) \right] + E_{x \sim p_{\theta}(x)} \left[\log \left(1 - d_{\phi}(x) \right) \right] \right)$$

$$= \min_{\theta} \max_{\phi} \left(E_{x \sim p_{re}(x)} \left[\log d_{\phi}(x) \right] + E_{z \sim p(z)} \left[\log \left(1 - d_{\phi}\left(g_{\theta}(z) \right) \right) \right] \right)$$

$\max_{\phi}$: train the discriminator to classify real and synthetic points.

$\min_{\theta}$: train the generator to minimize the loss above.

But the loss above is the negative of the discriminators cross-entropy loss ! In other words, train generator to fool the discriminator !

Generative Adversarial Networks

- GAN uses a two-network architecture, with a generator that generates an image and **a neural network discriminator to compare real and synthetic images as an approximation to the human perceptual difference.**

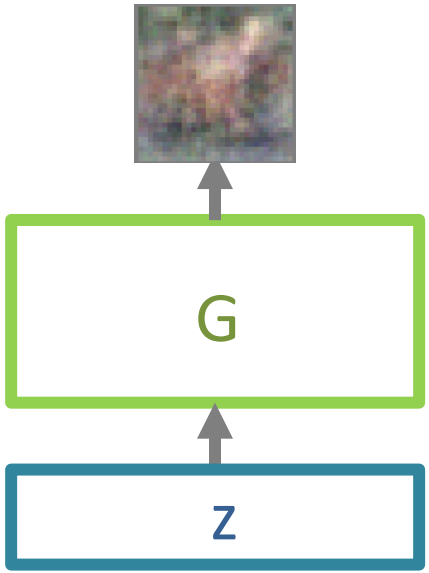
- GANs learn a transformation $p(x|z)$ from a known simple distribution $p(z)$ (e.g. Conditionally Independent Gaussian).

Adversarial Training

- **Any fixed discriminator model will probably be easy to fool**
- But since the discriminator have **learnable parameters**, it can learn to reject new synthetic images
- This is a **two-player minimax game**: generator tries to increase discriminator loss, discriminator tries to decrease it.
- Hence: it has a (local) Nash equilibrium which you can find with adversarial training.

Generation Network (G)

- Learn a transformation using a Neural network from a simple distribution to the target distribution



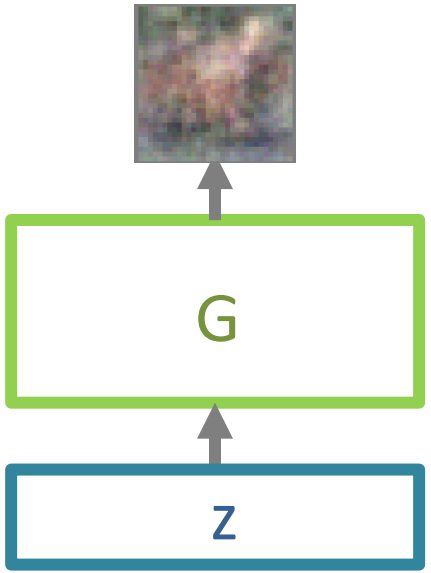
Output: Image

Transformation:
Neural-network,
Can many different architectures depending on the task

Input: Sample from $N(0, I)$

Generation Network (G)

- How exactly do we train this network?
  - We cannot use the same training scheme as VAE as we don't model the posterior likelihood $q(z|x)$.



Output: Image

Transformation:

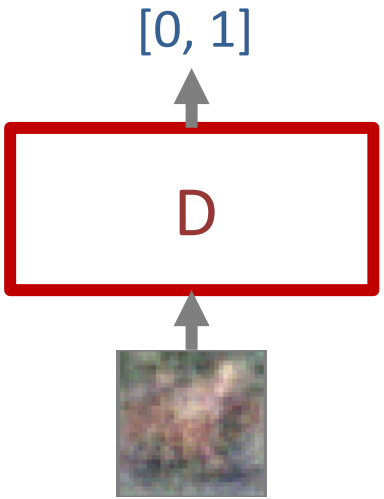
Neural-network,

Can many different architectures depending on the task

Input: Sample from $N(0, I)$

Discriminator Network (D)

- Recall..
  - All we want is to be able to learn a good enough transformation so that we can generate samples similar as the samples from the data.
  - Measuring the 'goodness' of the samples so far is hard, but we can use a neural network for it!



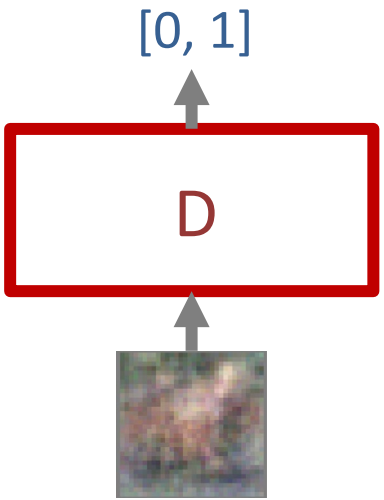
Output: "Realism" of the Image
(Probability of the image being real)

Discriminator:
Tries to distinguish between generated samples or samples from the dataset

Input: Generated/Real Image

Discriminator Network (D)

- We use ‘how distinguishable is the generated samples by the discriminator network’ as a proxy to the ‘goodness’ of examples
- All differentiable (or you can use REINFORCE for non-differentiable networks), so can be trained end-to-end with the generator



Output: “Realism” of the Image (Probability of the image being real)

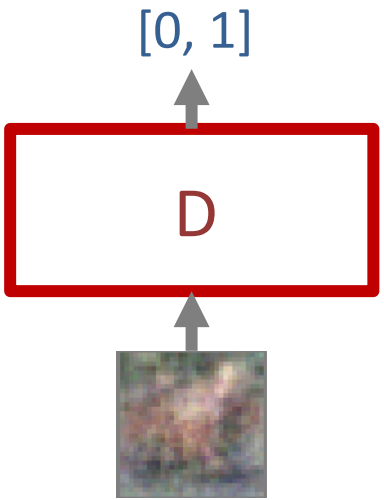
Discriminator:

Tries to distinguish between generated samples or samples from the dataset

Input: Generated/Real Image

Discriminator Network (D)

- We use ‘how distinguishable is the generated samples by the discriminator network’ as a proxy to the ‘goodness’ of examples
- All differentiable (or you can use REINFORCE for non-differentiable networks), so can be trained end-to-end with the generator



Output: “Realism” of the Image (Probability of the image being real)

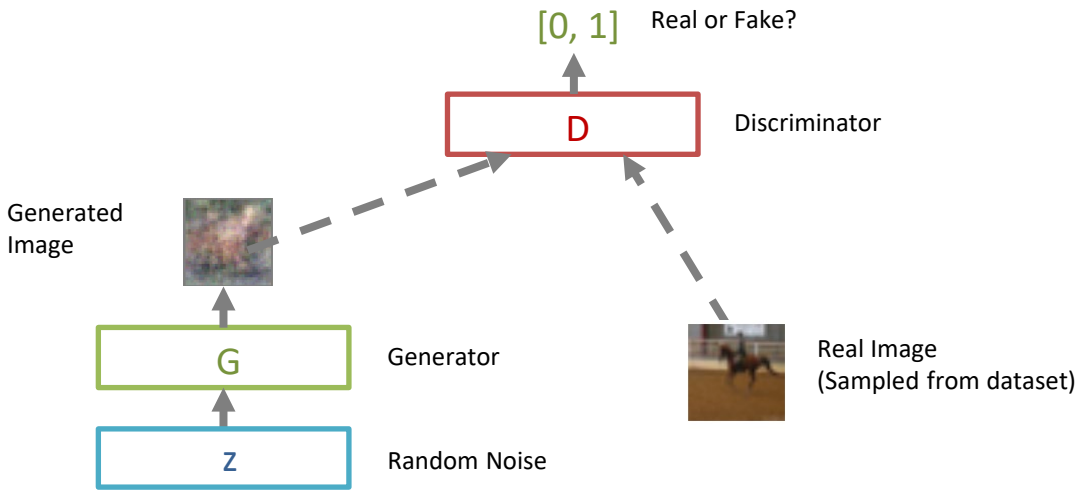
Discriminator:

Tries to distinguish between generated samples or samples from the dataset

Input: Generated/Real Image

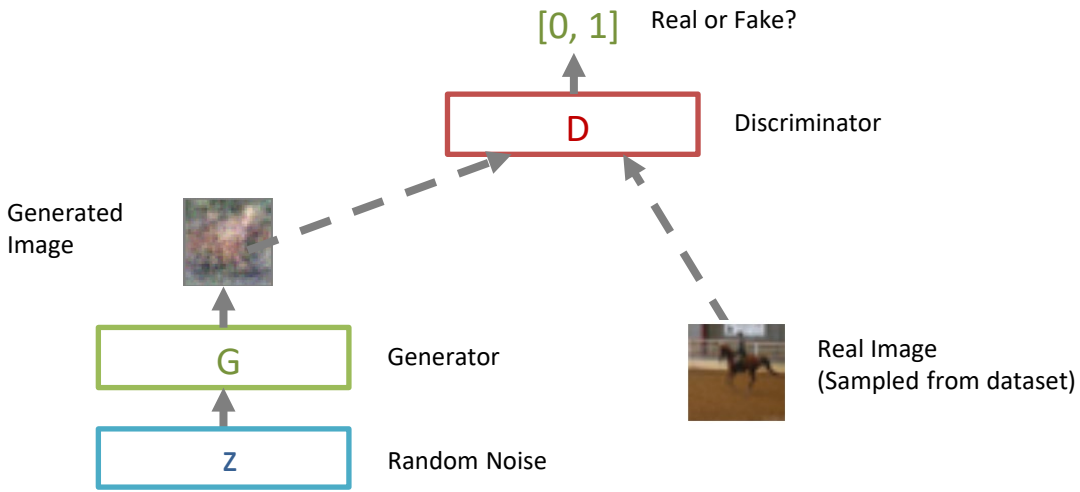
Generative Adversarial Networks

- Bring it all together:
  - **Discriminator Network (D):** Tries to distinguish between real images and fake images, hence preventing being fooled by G
  - **Generation Network (G):** Tries to fool D by generating realistic images.



Generative Adversarial Networks

- Bring it all together:
  - **Discriminator Network (D):** Tries to distinguish between real images and fake images, hence preventing being fooled by G
  - **Generation Network (G):** Tries to fool D by generating realistic images.



Generative Adversarial Networks

- Training Objective:
  - **Discriminator Network (D):** Tries to distinguish between real images and fake images, hence preventing being fooled by G
  - **Generation Network (G):** Tries to fool D by generating realistic images.

  Two-player Mini-max Game with Value function V(G, D):

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

Log Probability of x (real data) considered as 'real'

1 – log Probability of generated data considered as 'real'

  - This also minimizes Jensen-Shannon (JS) Divergence between x and $G(z)$, such that it minimizes the distance between the distributions of real data and generated data.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets.

Generative Adversarial Networks

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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets.

Generative Adversarial Networks

Two-player Mini-max Game with Value function V(G, D):

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

- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets.

Generative Adversarial Networks

Two-player Mini-max Game with Value function V(G, D):

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

Remember only to train the set of weights (G/D) when optimizing the loss, and freezing the rest of the weights.

- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets.

Practical Aside:

For G, rather than minimizing:

$$\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

Maximize:

$$\frac{1}{m}\sum_{i=1}^{m}\log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

Why?

Practical Aside:

For G, rather than minimizing:

$$\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

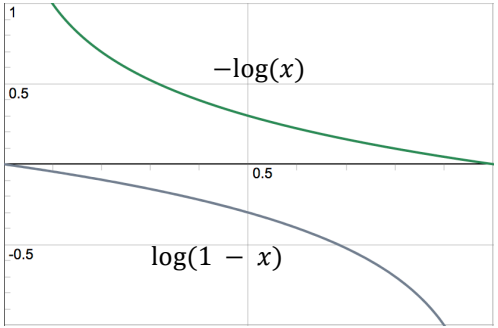
Maximize:

$$\frac{1}{m}\sum_{i=1}^{m}\log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

Why?

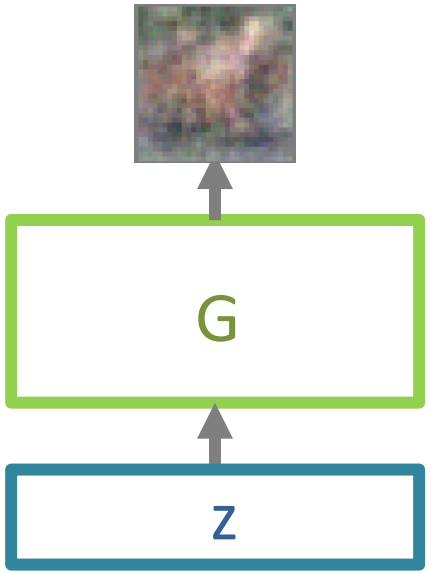
Large Gradients when generated samples are bad

Large Gradients when generated samples are already good



Generation Network (G)

- After training,
  - Simply take the Generator network to generate new samples.



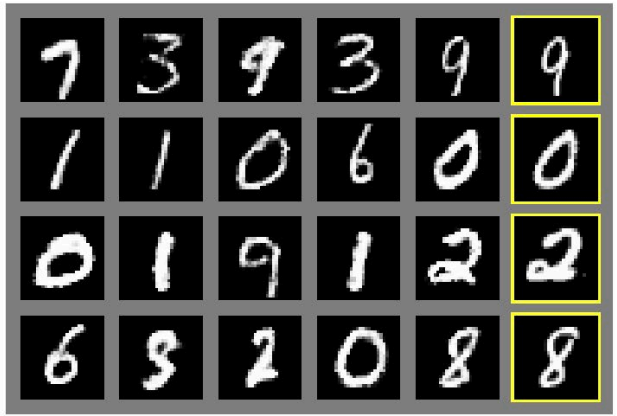
Output: Image

Transformation:
Neural-network,
Can use many different architectures depending on the task

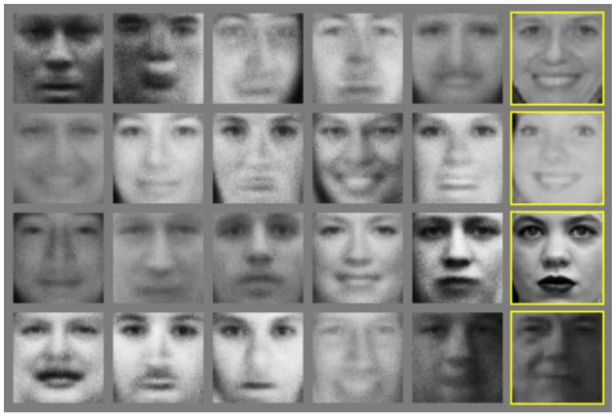
Input: Sample from $N(0, I)$

Generation Examples

- Original GAN paper only showed good samples for
  - MNIST
  - Toronto Face Database



Generated Images **NN (Nearest-neighbor) In Dataset**

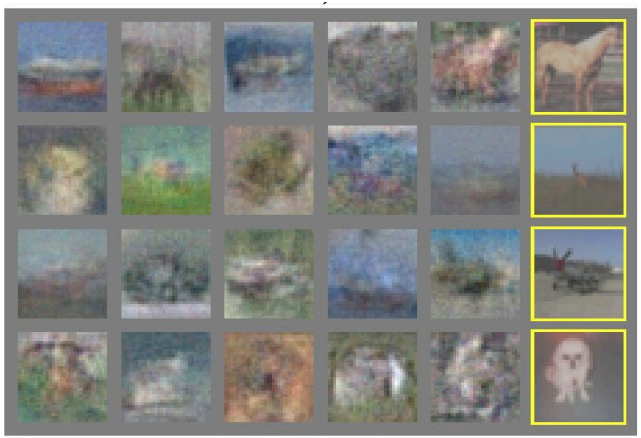


Generated Images **NN In Dataset**

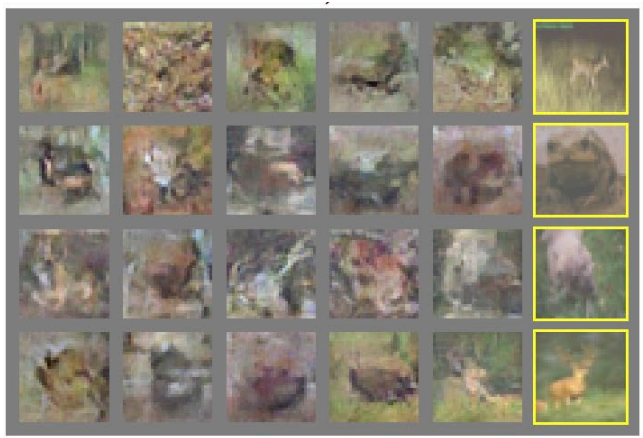
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets.

Generation Examples

- Original GAN paper produced perceptually poor image samples for CIFAR:



Generated Images **NN In Dataset**



Generated Images **NN In Dataset**

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets.

GAN Training Tips

- In practice, to train a good GAN requires a very fine balance between the discriminator/generator.

- Because if the discriminator lags behind in training to the generator, it fails to provide meaningful gradients to the generator, similarly when classification loss is very small, the discriminator gradients will become very small.

- Gradient descent on the cost functions **not guaranteed to converge to actual Nash Equilibrium.** https://arxiv.org/abs/1806.07268

- Many practical 'tricks' are often required to get it to work, these tricks are derived empirically:
  - https://github.com/soumith/ganhacks

The GAN Zoo

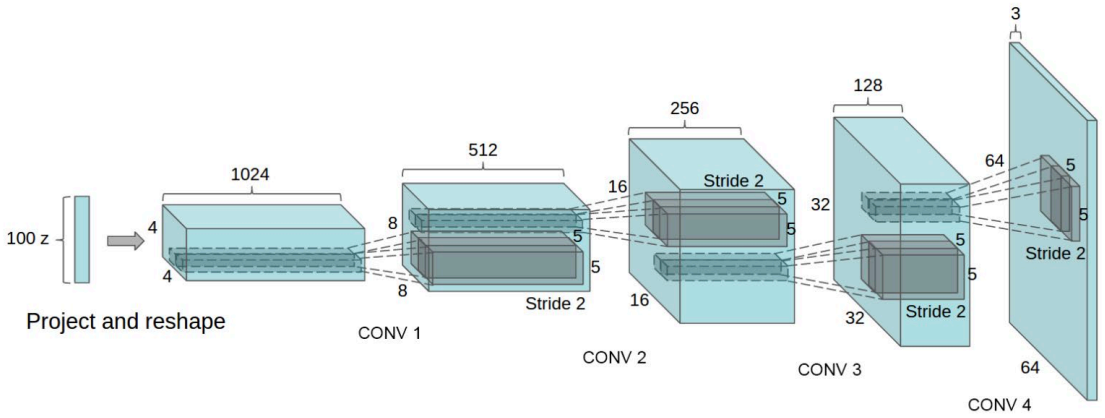
- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

We will go over some of the GAN architectures in the remainder of the lecture:

CGAN, ACGAN, GAN-CLS

DCGAN

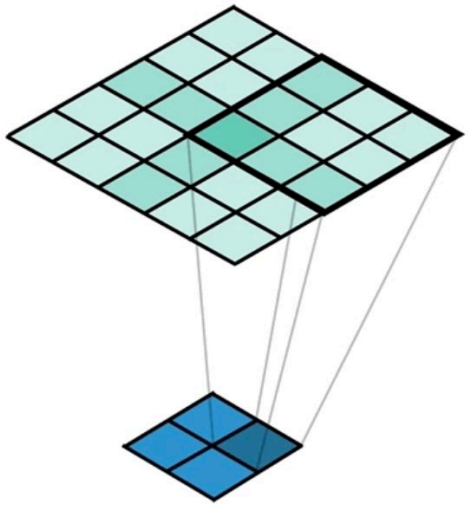
- **GAN design does not specify what the generator is**, only that the generator will be steered toward the best images it can produce by the discriminator.
- DCGAN has large Improvements over Image-Generation Tasks
  - Generation Network: Adopted from all convolutional net:



Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks.

DCGAN

- DCGAN has large Improvements over Image-Generation Tasks
  - Generation Network: Adopted from all convolutional net:
  - Fractional-transposed Convolution (Strided Transposed Convolution for Up-sampling)
  - Equivalent to Stride 1/(output stride) On the Input: Fractional



Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks.

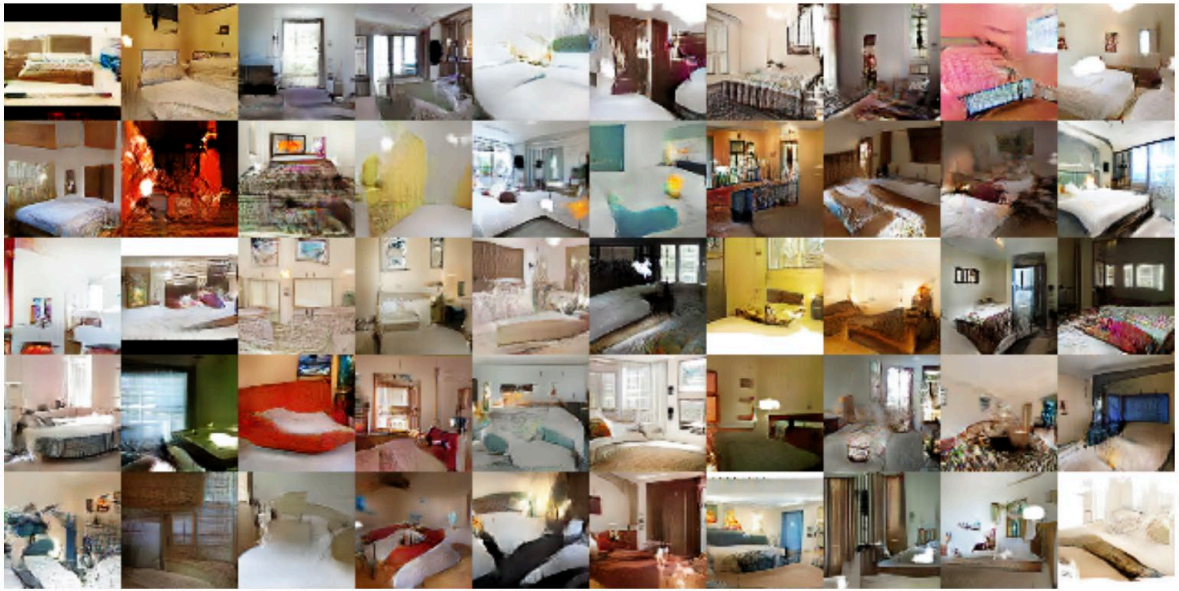
DCGAN

- Contribution: Architecture Guidelines:

- Replace any pooling layers with strided convolutions
- Use Batch Norm in both discriminator and generator
- Remove FC layers for deeper architectures
- **Use ReLU in generator**, except tanh for output
- **Use LeakyReLU in discriminator** for all layers

DCGAN

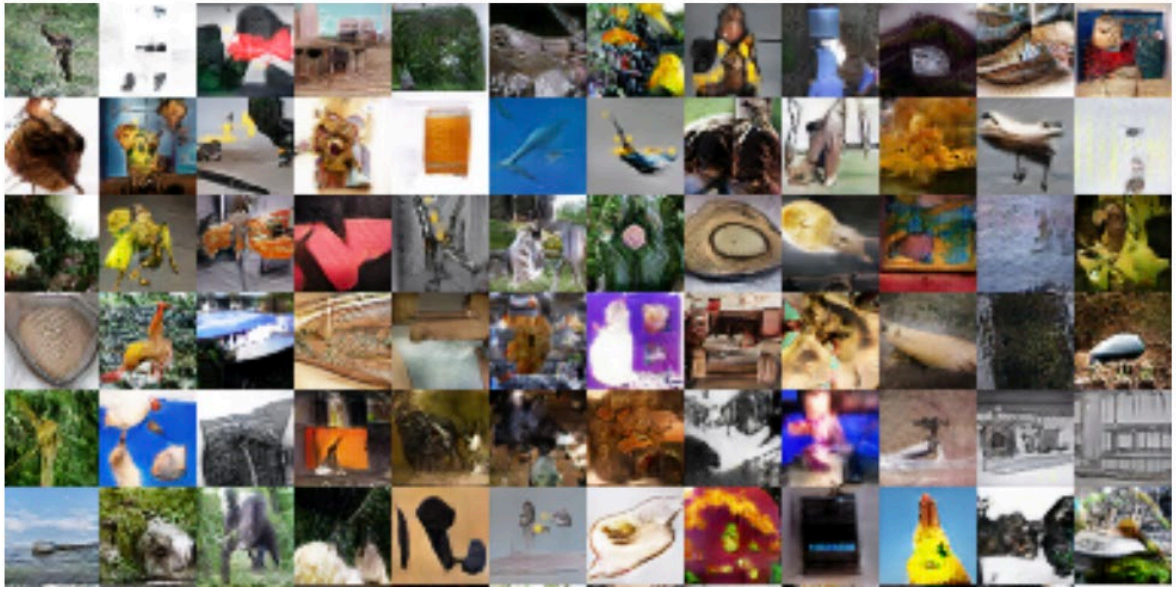
- Results:
  - LSUN Bedroom (3M Bedroom Images Only) shows very promising results



Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks.

DCGAN

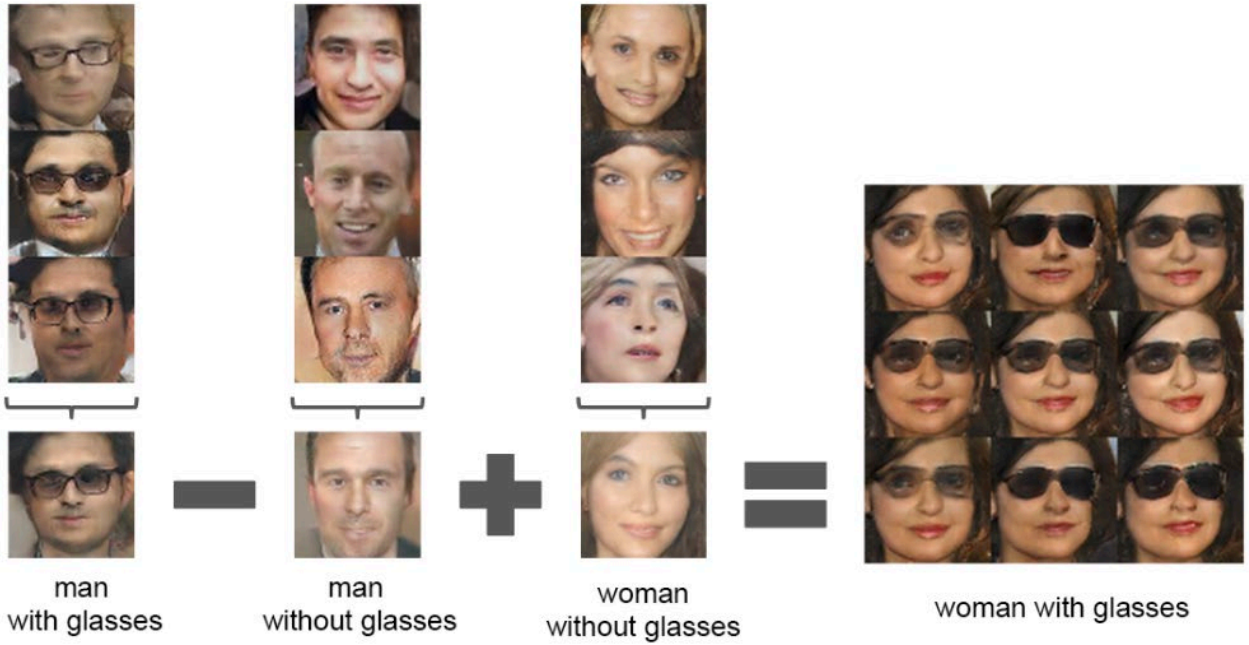
- Results:
  - ImageNet-1k (1000 class Imagenet, 32x32 Center Crop)
  - Generated samples far from impressive, which will be improved by later work



Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks.

DCGAN

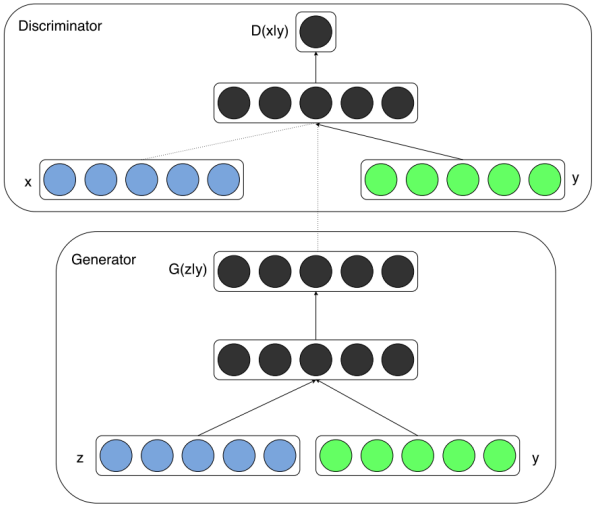
- Vector arithmetic in z:
  - Retain the z used by the G network to generate these images, do arithmetic and use the new z to generate new images, i.e. z is used as the latent vector



Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks.

Conditional GAN

- On top of Gaussian Noise z, we can also add a label $\boldsymbol{y}$ to specify classes in the dataset that you want to sample from.



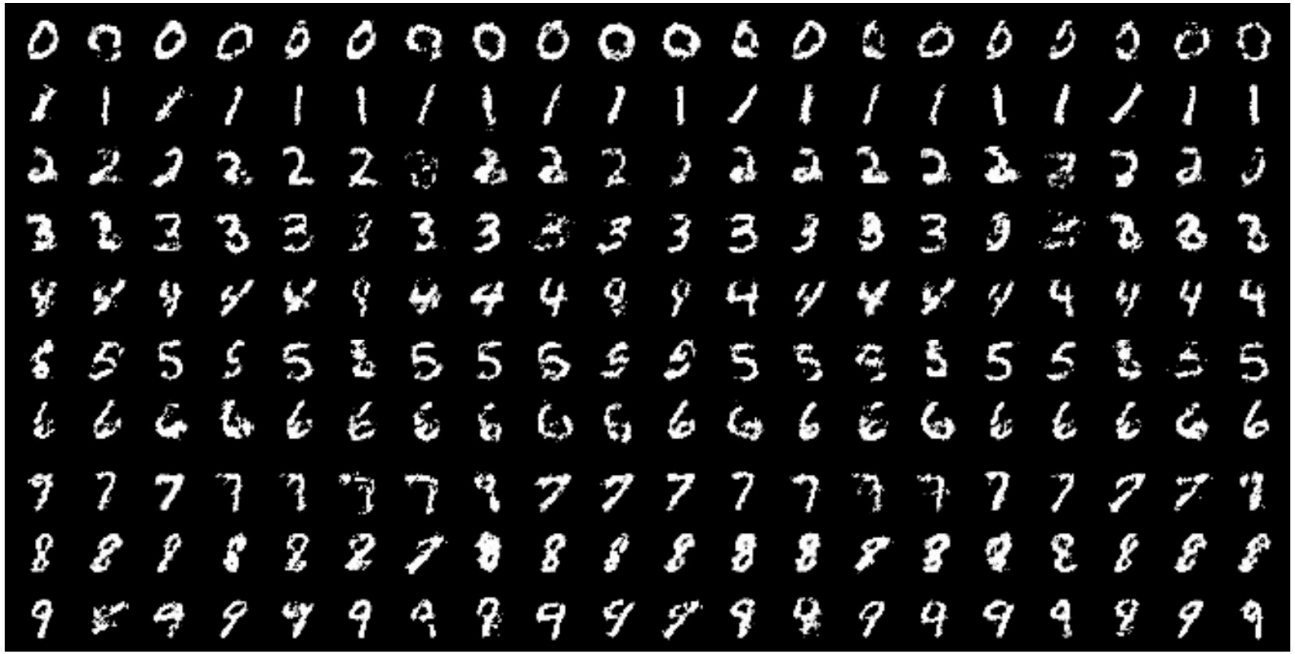
New Objective:

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

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets.

Conditional GAN

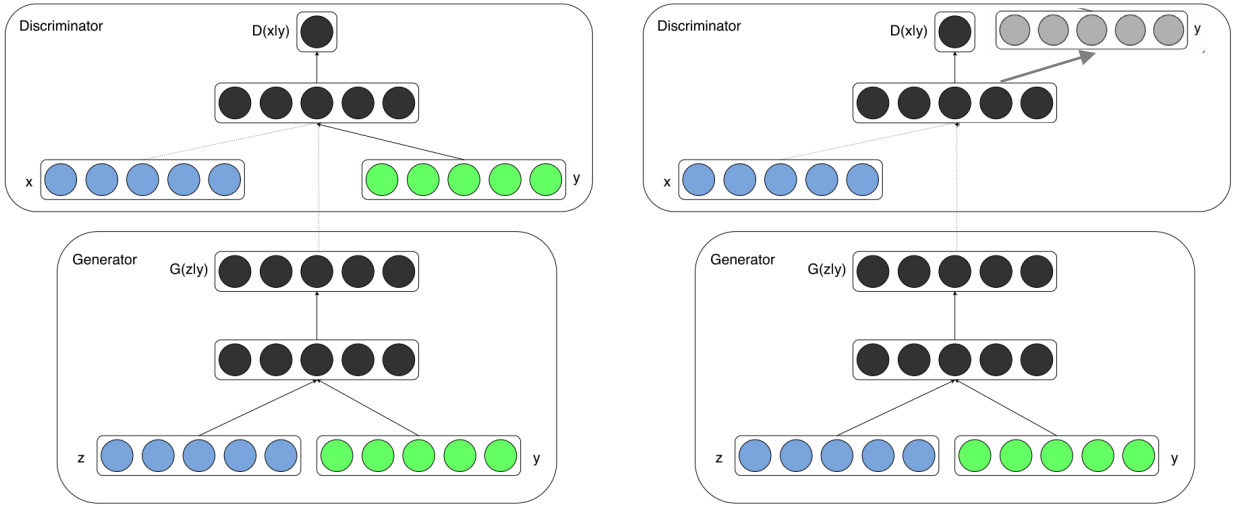
- Generated samples on MNIST with class label = numbers



Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets.

ACGAN

- Compared to CGAN, instead of adding labels as input to the discriminator, asks the discriminator to output a class label.



CGAN ACGAN

Odena, A., Olah, C., & Shlens, J. (2017, August). Conditional image synthesis with auxiliary classifier gans.

ACGAN

- Compared to CGAN, instead of adding labels as input to the discriminator, asks the discriminator to output a class label.

$$L_S = E[\log P(S = real \mid X_{real})] + E[\log P(S = fake \mid X_{fake})] \quad (2)$$

$$L_C = E[\log P(C = c \mid X_{real})] + E[\log P(C = c \mid X_{fake})] \quad (3)$$

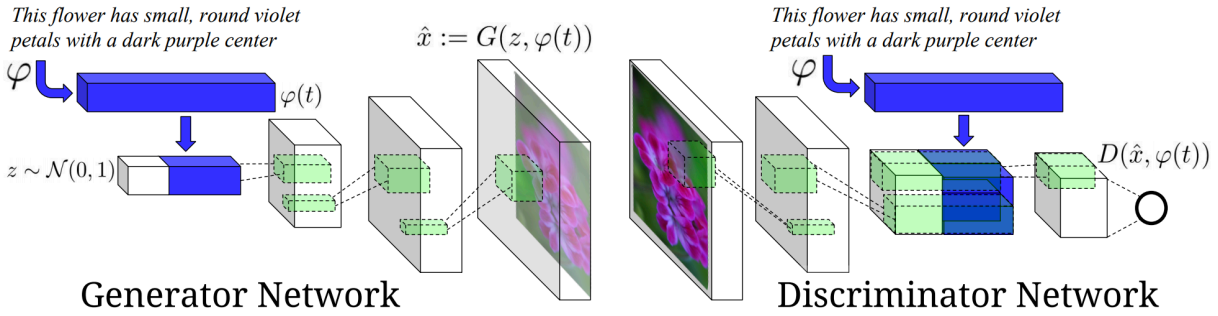
Discriminator: Maximize $L_c + L_S$

Generator: Maximize $L_c - L_S$

Odena, A., Olah, C., & Shlens, J. (2017, August). Conditional image synthesis with auxiliary classifier gans.

Text-to-Image Synthesis

- Text-to-Image Synthesis is a special case of CGAN (GAN-CLS)
  - y (label) is the sentence vector, either pre-trained or taken from a RNN encoder

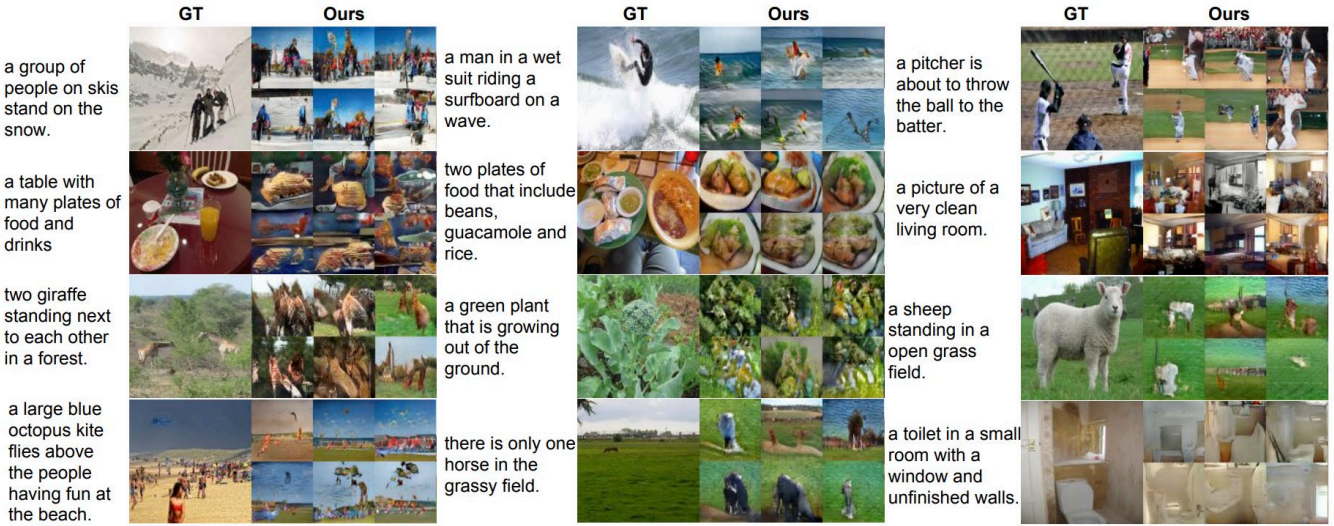


- Discriminator Loss function considers matching:

$$s_r \leftarrow D(x, h) \text{ {real image, right text}}$$
$$s_w \leftarrow D(x, \hat{h}) \text{ {real image, wrong text}}$$
$$s_f \leftarrow D(\hat{x}, h) \text{ {fake image, right text}}$$
$$\mathcal{L}_D \leftarrow \log(s_r) + (\log(1 - s_w) + \log(1 - s_f))/2$$

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. (2016). Generative adversarial text to image synthesis.

Text-to-Image Synthesis



- Supports interpolation on y (sentence vector)

‘Blue bird with black beak’ →
‘Red bird with black beak’

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. (2016). Generative adversarial text to image synthesis.

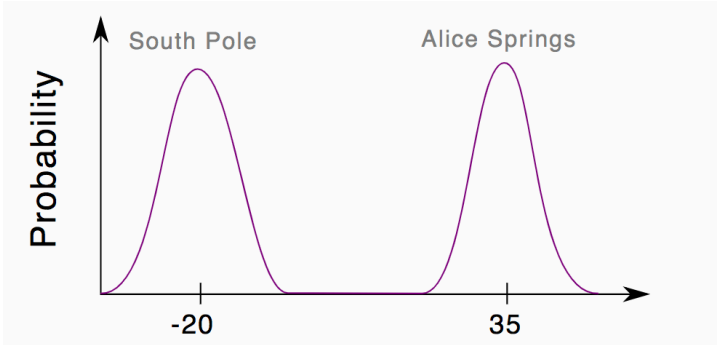
Common Problems with GANs (and Solutions)

- Mode Collapse
  - Unrolled GAN
- Difficulty in Training
  - WGAN – Wasserstein Distance
- Difficulty in understanding global structure
  - StackGAN/BigGAN

- Difficulty in Evaluation (as with many other generative models, but particularly for GANs)
  - Inception Score
  - Frechet Inception Distance
- Unpaired Conditional Generation
  - CycleGAN/XGAN

Mode Collapse

- Mode Collapse is a common problem in GAN, it can be illustrated by:

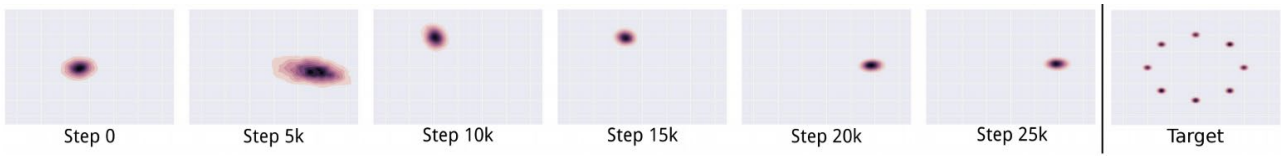


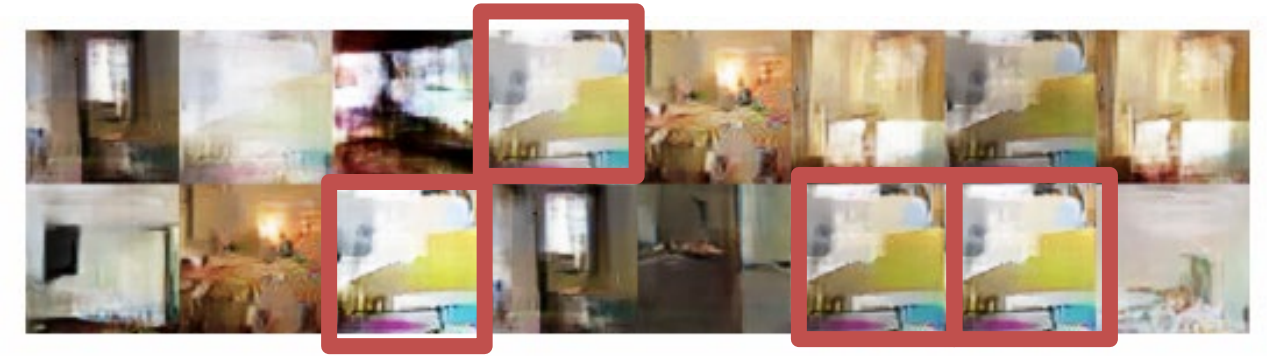
1) G Learns to generate all examples in south pole
2) D Learns all Alice springs temps. are real, but guess south pole as 0.5
3) G now switches to Alice springs temps., hence fooling D
4) D now knows south pole temps. are all real, switches the decision around.
5) Repeat

http://aiden.nibali.org/blog/2017-01-18-mode-collapse-gans/

Mode Collapse

- In practice, this happens frequently:





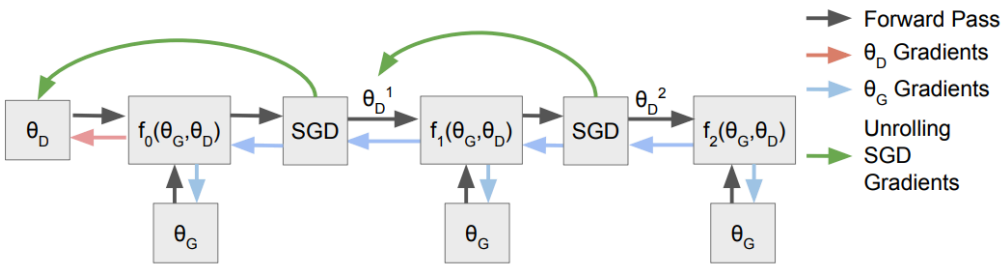
https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b

Mode Collapse

Solutions:

- Experience replay: substituting an old discriminator/generator every couple of iterations

- Anticipate the gradients that the discriminator is taking, and allow generator to ‘look-ahead’ to prevent the cycle. (Unrolled GAN)



Metz, L., Poole, B., Pfau, D., & Sohl-Dickstein, J. (2016). Unrolled generative adversarial networks.

Difficulty in Training

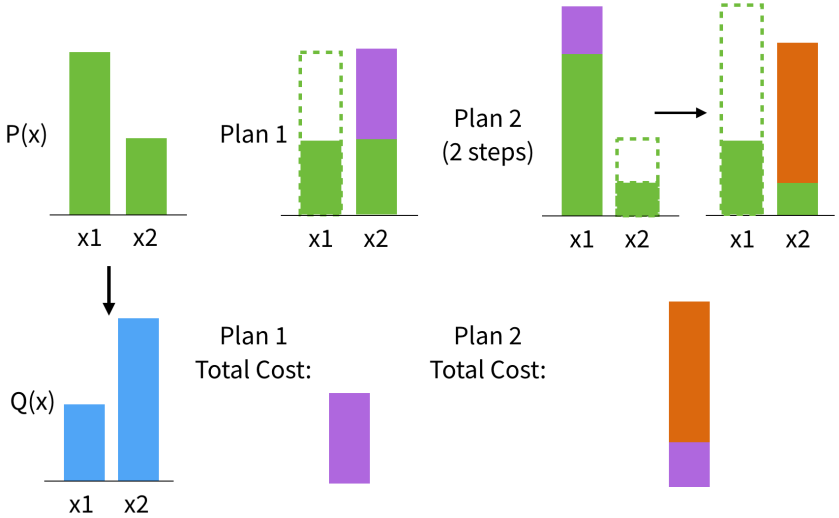
- The alternating training scheme using gradient descent might be difficult for convergence.
- WGANs: Instead of asking discriminator to make a decision, ask it to approximate a metric for computing the minimum distance to match the distributions p_model to p_data, in this case, using the Wasserstein distance, hence **W**-GAN.
- Intuition of Wasserstein Distance (Earth-Mover/EM distance):
  - Assume both distributions as piling up dirt over a region D
  - Earth Mover's distance is the minimum cost of turning one pile into another.

Intuition of Wasserstein Distance

Goal: Move mass from P(x) to match Q(x)

Cost: Mass moved in total × Distance to move (1 in our case)

P(x), Q(x): Discrete Distributions



However: There can be many plans (e.g. Plan 2) with higher costs, especially in continuous distributions

Wasserstein Distance: **Minimum** cost to match the distributions (**Plan 1**)

Wasserstein GAN

- For an image generation task, Wasserstein distance measures the minimum cost for matching the distributions of $g_\theta(z)$ and x
- The discriminator now attempts find a good 'critic' function f_w (Intuition: realism score of the generated samples) that quantifies distance between the distributions of generated images and real images
- Critic f_w Objective (Maximize):

$$\left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$$

- Generator g_θ attempts to maximize the realism score of the synthetic examples:

$$\frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$$

Wasserstein GAN

- To build a W-GAN, simply:

1) Remove the Sigmoid activation at the D (now 'critic' f_w) output layer, so it now outputs a linear score

2) Clip the weights to a fixed range after each gradient update step, to enforce the Lipschitz constraint (see paper for details)

3) Use new learning objectives:

Ascend gradients for w, parameters of the critic:

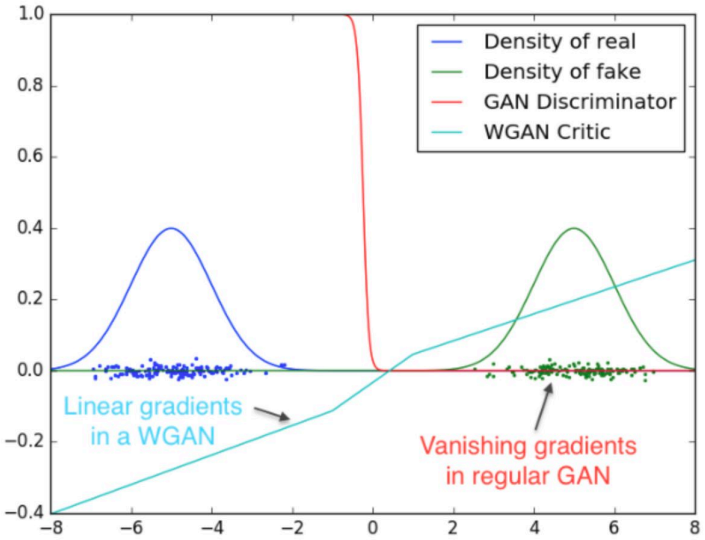
$$\nabla_w \left[\frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)})) \right]$$

Ascend gradients for θ , parameters of the generator:

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$$

Advantages of Wasserstein GAN

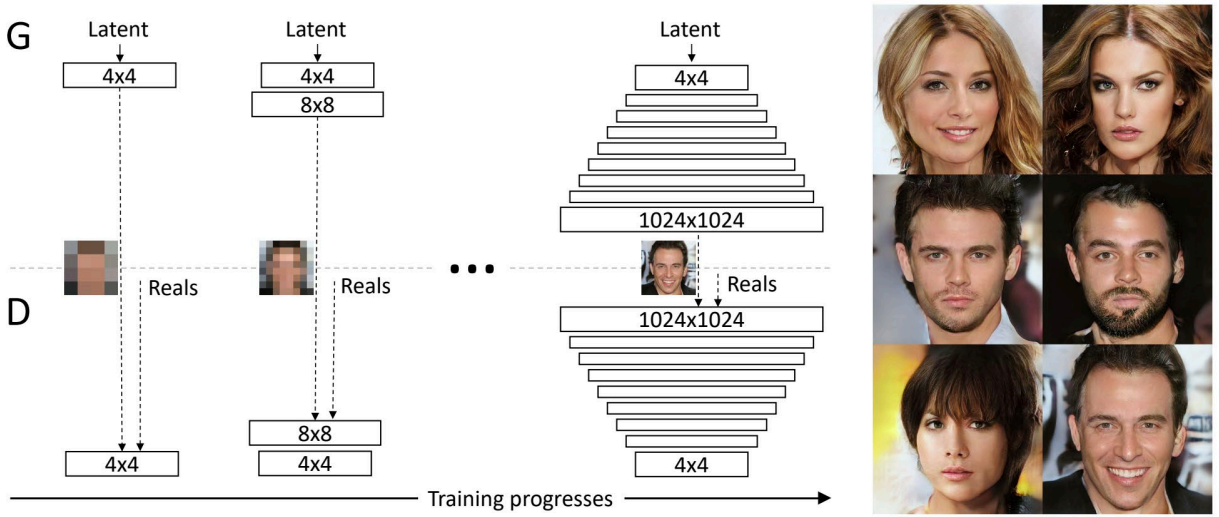
- Can keep learning until convergence: **gradients everywhere**
- More stable training/consistent convergence
- Train Critic till optimality: Avoids **Mode Collapse**
- A lot of theory behind it!



Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein gan.

Difficulty in Understanding Global Structure

- Progressive GANs
  - Incrementally add layers to GANs, each in charge of a certain resolution of output generations, and progressively learn the up-sampling of the images.
  - All previous layers remain trainable.



Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. ≈

Progressive GANs

- Progressive GANs Results

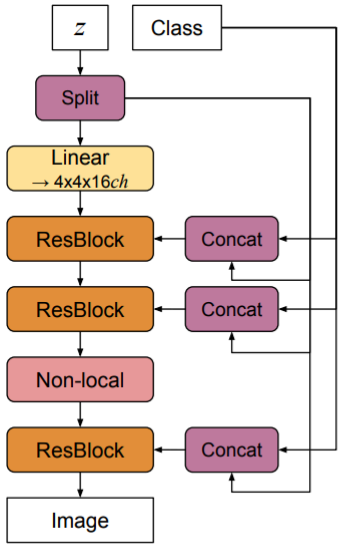
Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation.

BigGAN

- State-of-the-art: trained with large amount of computing power and better scaling/sampling methods with ResNet-style Up/Down Blocks
  - (Paper mentioned Progressive GAN not needed)

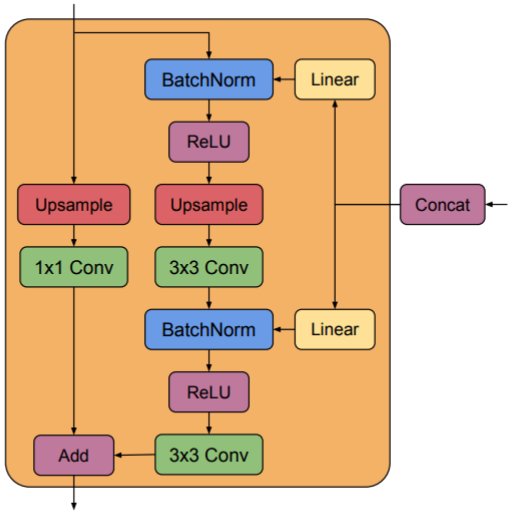
Brock, A., Donahue, J., & Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis.

BigGAN



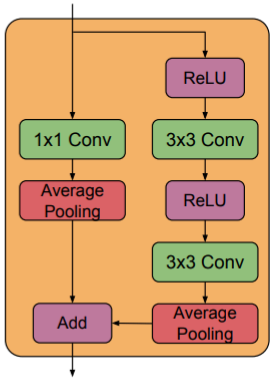
(a)

Generator Archirtecture



(b)

Generator ResBlock up



(c)

Discriminator ResBlock down

Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2018). Self-attention generative adversarial networks.

Common Problems with GANs (and Solutions)

- Mode Collapse
  - AdaGAN, Unrolled GAN
- Difficulty in Training
  - WGAN – Wasserstein Distance
- Difficulty in understanding global structure
  - StackGAN/BigGAN

- Difficulty in Evaluation (as with many other generative models, but particularly for GANs)
  - Inception Score
  - Frechet Inception Distance
- Unpaired Conditional Generation
  - CycleGAN/XGAN

Evaluating GANs

- Inception Score:

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_g} D_{KL}(p(y|\mathbf{x}) \parallel p(y)) \right),$$

  - Use the Inception v3 network to compute both $\mathrm{P}(y|x)$ and $\mathrm{P}(y)$,
  - $\boldsymbol{P}(\boldsymbol{y}|\boldsymbol{x})$ is the classification result of a single sample x
  - $P(y)$ is estimated by summing all classifications of $\boldsymbol{x}$
  - GANs with high Inception scores would have low entropy for $\mathrm{P}(y|x)$, and high entropy for $P(y)$
  - $P(y)$ stands for **diversity**, $\mathrm{P}(y|x)$ stands **for clearness**, meaning Inception network should be confident that there should be an object in the object.
  - A High KL divergence means the network can generate high-quality individual examples, and diverse overall examples.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training gans.

Evaluating GANs

- Inception Score:

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x} \sim p_g} \, D_{KL}(\, p(y|\mathbf{x}) \parallel p(y) \,) \right),$$

  - Works mostly for natural images
  - Large dataset of images + dense labels for a well-designed network to pre-train on.
  - Correlates well to human judgment: Inception Model

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium.

Evaluating GANs

- Frechet Inception Distance:

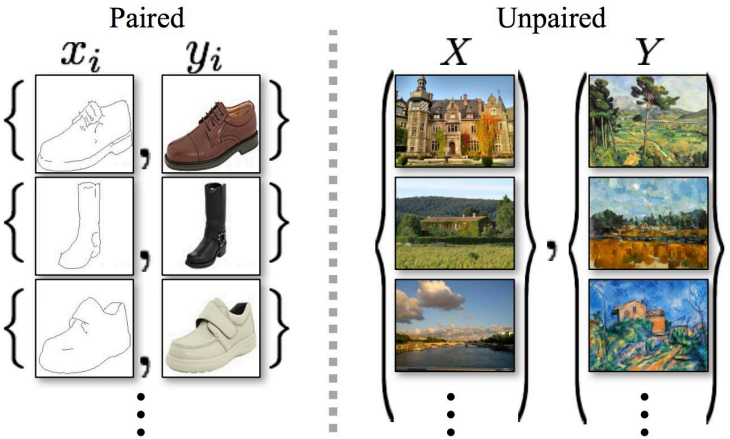
$$d^2((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_w, \boldsymbol{C}_w)) = \|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2 + \operatorname{Tr}\left(\boldsymbol{C} + \boldsymbol{C}_w - 2\left(\boldsymbol{C}\boldsymbol{C}_w\right)^{1/2}\right).$$

  - Compares the 2048-FC-layer activations, instead of the output probability of the Inception Networks
  - We model a Gaussian around the activations, (m, C) is the Gaussian of activation values of sampled real images, and (m_w, C_w) is the Gaussian of the activation values of sampled generated images
  - Intuition: Compute the distance between the distributions of activations between fake + real images.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training gans.

Unpaired Conditional Image Generation

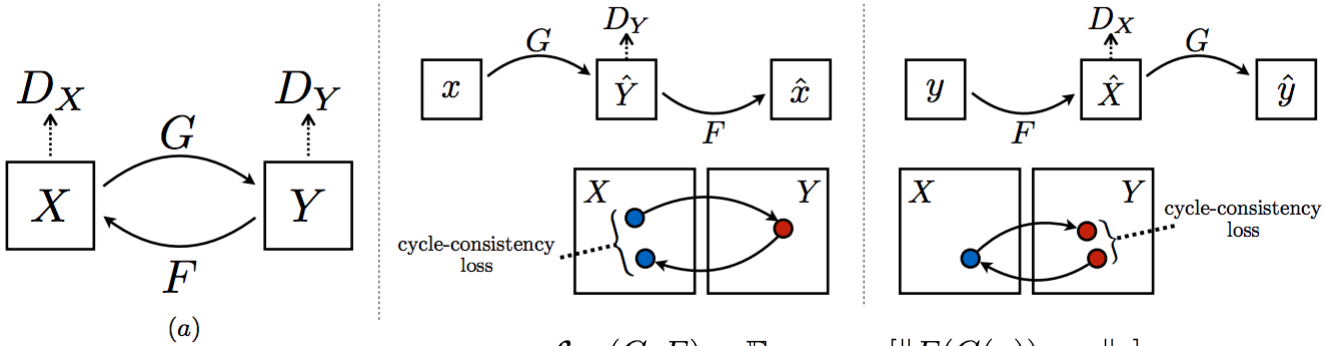
- So far, we have looked at paired image data for translating between image <-> image / text <-> image
- What if we only have **two sets of data that we know are in separate domains, but are not paired against each other** (e.g. artwork and photos)



Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks.

CycleGAN

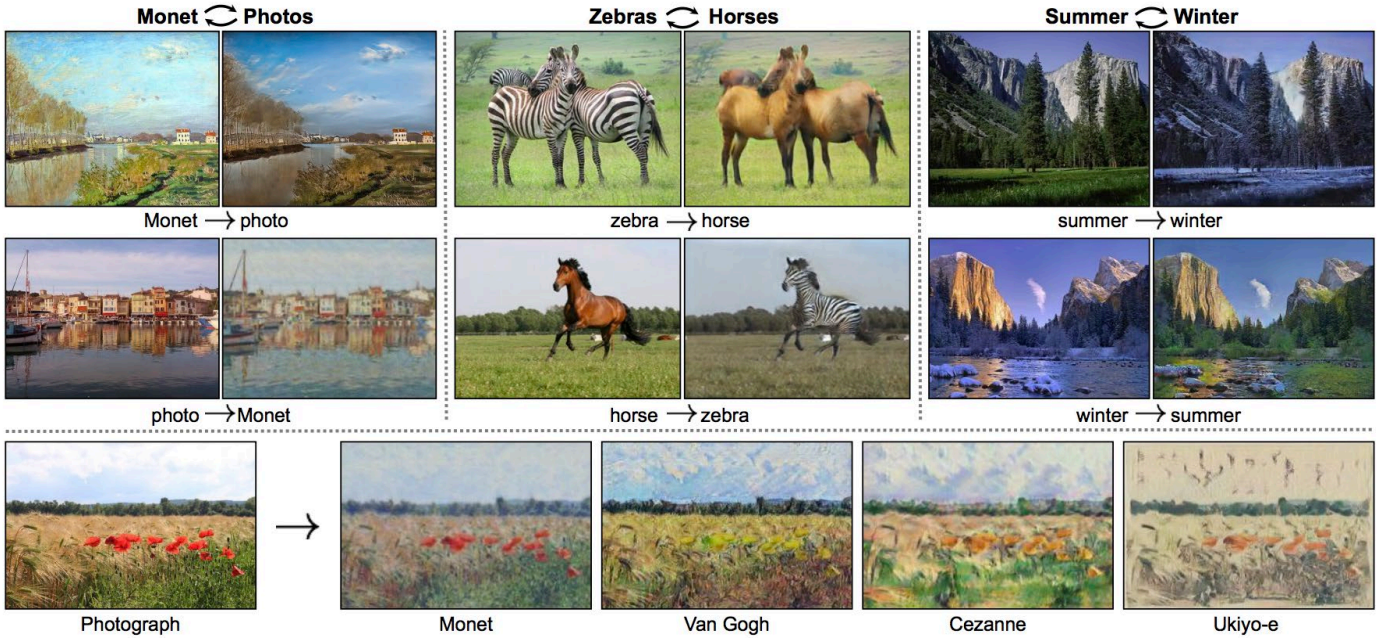
- 2 generators + 2 discriminators, 1 pair for each domain
- Instead of having to use pairing, only relies on real images on the other domain for realism
- Use cycle-consistency loss for pairing instead (L1 distance)



$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks.

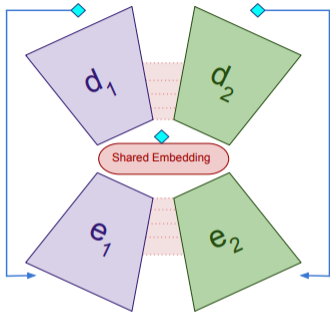
CycleGAN



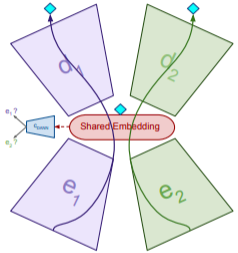
Zhu, J. Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks.

X-GAN

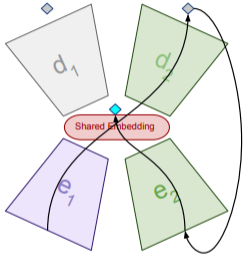
- Explicit dual auto-encoder representation, with a shared embedding for many-to-many mapping



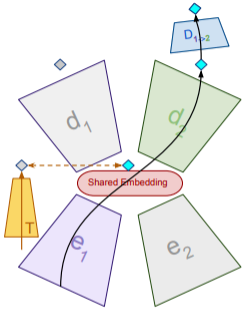
(A) High-level view of the XGAN dual auto-encoder architecture



(B1) Domain-adversarial auto-encoder



(B2) Semantic consistency feedback loop

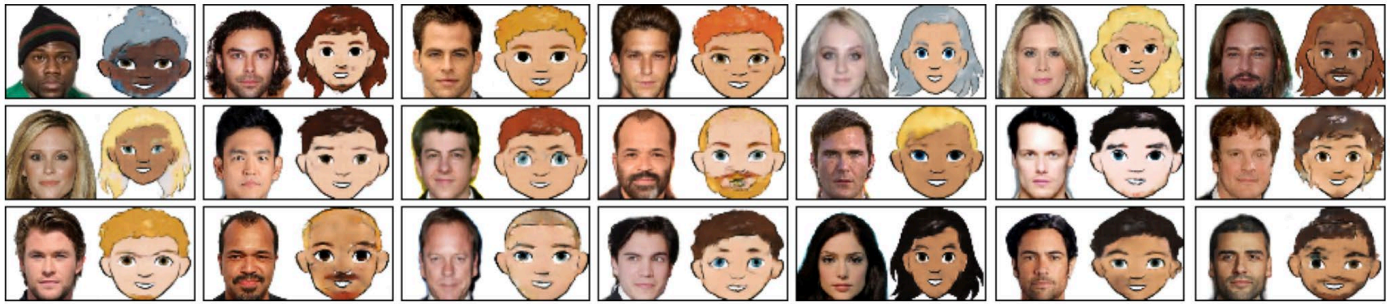


(B3) GAN and Teacher loss modules

Royer, A., Bousmalis, K., Gouws, S., Bertsch, F., Mosseri, I., Cole, F., & Murphy, K. XGAN: Unsupervised Image-to-Image Translation for Many-to-Many mappings.

X-GAN

- Explicit dual auto-encoder representation, with a shared embedding for many-to-many mapping



Royer, A., Bousmalis, K., Gouws, S., Bertsch, F., Mosseri, I., Cole, F., & Murphy, K. XGAN: Unsupervised Image-to-Image Translation for Many-to-Many mappings.

GANs Summary

- Model generates samples **but does not estimate density** p(x) at image x.
- **2-player Mini-max game** between the discriminator and generator
- Original GAN minimizes JS-divergence
- Multiple **architectural improvements** for Generator/Discriminator itself: DCGAN/Progressive GAN allowed GANs to generate very impressive high-resolution
- Multiple **training scheme improvements:** WGAN for stable training
- Training can be unstable, many empirical ‘tricks’ might be required