M2177.0043 Introduction to Deep Learning

Lecture 14: Deep Generative Models[1]

Hyun Oh Song[1]

[1]Dept. of Computer Science and Engineering, Seoul National University

May 7, 2020

[1]Many slides and figures adapted Justin Johnson

Last time

- VAE
- Reparameterization trick

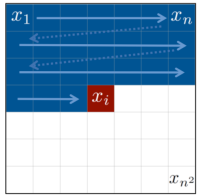
Outline

PixelRNN[2]

Autoregressive Image Modeling

- Autoregressive models train a network that models the conditional distribution of every individual pixel given previous pixels (raster scan order dependencies).

$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, ..., x_{i-1}).$$



⇒ **sequentially predict pixels** rather than predicting the whole image at once (like as GAN, VAE)

- For color image, 3 channels are generated successive conditioning, blue given red and green, green given red, and red given only the pixels above and to the left of all channels.

[2] from Yohei Sugawara

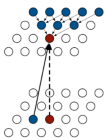
PixelRNN[3]

Previous work: Pixel Recurrent Neural Networks.

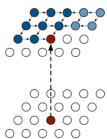
- "Pixel Recurrent Neural Networks" got best paper award at ICML2016.
- They proposed two types of models, **PixelRNN** and **PixelCNN** (two types of LSTM layers are proposed for PixelRNN.)

PixelRNN

Row LSTM

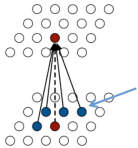


Diagonal BiLSTM

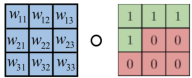


- LSTM based models are natural choice for dealing with the autoregressive dependencies.

PixelCNN



masked convolution



- CNN based model uses masked convolution, to ensure the model is causal.

	PixelRNN	PixelCNN
Pros.	• effectively handles long-range dependencies ⇒ **good performance**	Convolutions are easier to parallelize ⇒ **much faster** to train
Cons.	• Each state needs to be computed sequentially. ⇒ **computationally expensive**	Bounded receptive field ⇒ **inferior performance** **Blind spot problem** (due to the masked convolution) needs to be eliminated.

3 from Yohei Sugawara

PixelRNN

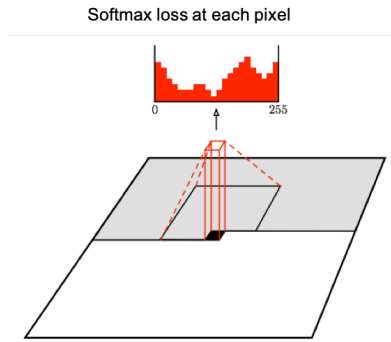
- Refer to the original paper `https://arxiv.org/abs/1601.06759` for the implementation details and parallelizations

PixelCNN

- Dependency on previous pixels now models using a CNN with **masked convolutions** over context region
- Training: maximize likelihood of training images

$$p(x) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

- Training faster than PixelRNN by parallelizing convolutions over each context regions
- Generation still slow because it need to happen sequentially



Outline

Generative adversarial networks

- VAEs define intractable density function with latent variable z.

$$p_\theta(x) = \int p_\theta(z) p_\theta(x \mid z) dz$$

  However, we cannot optimize this directly, so we derive and optimize the lower bound on likelihood (ELBO) instead

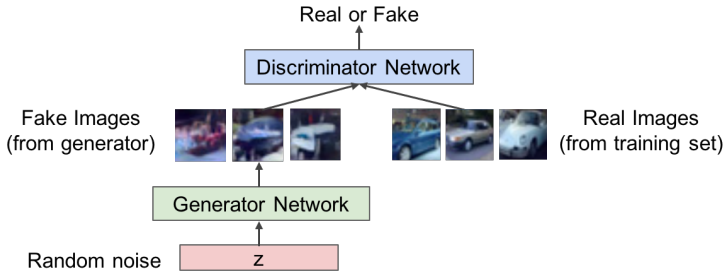
- GANs: don't work with any explicit density function. Instead, take game-theoretic approach: learn to generate from training distribution through 2-player game.

Generative adversarial networks

- Problem: want to sample from complex, high-dimensional training distribution. No direct way to do this!
- Solution: Sample from a simple distribution. *e.g.* random noise. Learn transformation (generator network) to training distribution.

Training GANs: Two-player game

- Generator network: try to fool the discriminator by generating real-looking images
- Discriminator network: try to distinguish between real and fake images



Fake and real images copyright Emily Denton et al. 2015. Reproduced with permission.

Training GANs: Two-player game

- Generator network: try to fool the discriminator by generating real-looking images
- Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Training GANs: Two-player game

- Generator network: try to fool the discriminator by generating real-looking images
- Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Discriminator output for real data x

Discriminator output for generated fake data G(z)

Training GANs: Two-player game

- Generator network: try to fool the discriminator by generating real-looking images
- Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

- Discriminator (θ_d) wants to **maximize objective** such that D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake)
- Generator (θ_g) wants to **minimize objective** such that D(G(z)) is close to 1 (discriminator is fooled into thinking generated G(z) is real)

Training GANs: Two-player game

Train jointly in **minimax game**

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

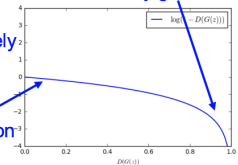
2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

Gradient signal dominated by region where sample is already good

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!



x-axis: $D(G(z))$, y-axis (blue): $\log(1 - D(G(z)))$

Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

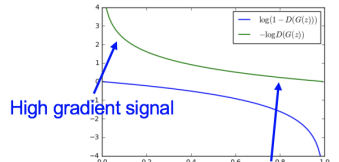
1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Instead: Gradient ascent** on generator, different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.
Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.



x-axis: $D(G(z))$, y-axis (blue): $\log(1 - D(G(z)))$, (green): $-\log(D(G(z)))$

Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

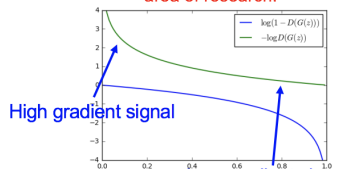
$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Instead: Gradient ascent** on generator, different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.
Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.

Aside: Jointly training two networks is challenging, can be unstable. Choosing objectives with better loss landscapes helps training, is an active area of research.



x-axis: $D(G(z))$, y-axis (blue): $\log(1 - D(G(z)))$, (green): $-\log(D(G(z)))$

Training GANs: Two-player game

Putting it together: GAN training algorithm

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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

  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 ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

Training GANs: Two-player game

Putting it together: GAN training algorithm

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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

  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 ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

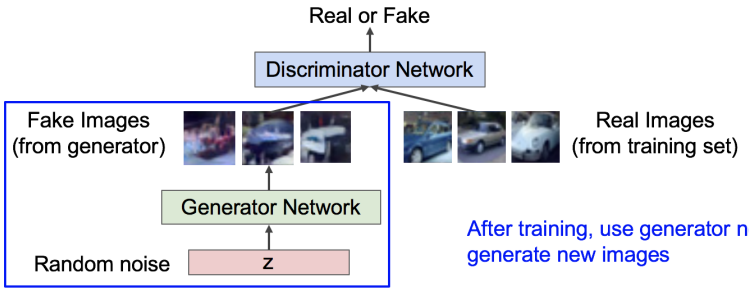
end for

Some find k=1 more stable, others use k > 1, no best rule.

Recent work (e.g. Wasserstein GAN) alleviates this problem, better stability!

Training GANs: Two-player game

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



After training, use generator network to generate new images

Fake and real images copyright Emily Denton et al. 2015. Reproduced with permission.

Jensen-Shannon Divergence

Definition 1 (Jensen-Shannon Divergence)

Jensen-Shannon Divergence (JSD) is defined by

$$JSD(P \parallel Q) = \frac{1}{2}KL(P \parallel M) + \frac{1}{2}KL(Q \parallel M), \text{ where } M = \frac{1}{2}(P + Q)$$

- Note, the JSD is bounded below by zero and above by 1 (base 2) or $\ln(2)$ (base e)
- Side note, KL is not bounded above and can reach ∞

Optimal discriminator D for any given generator G

Proposition 2

For G fixed, the optimal discriminator D is

$$D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{1}$$

Proof.

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

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

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \to a\log(y) + b\log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $Supp(p_{data}) \cup Supp(p_g)$. ■

- Note that the training objective for D can be interpreted as maximizing the log-likelihood for estimating the conditional probability $P(Y = y \mid x)$, where Y indicates whether x comes from p_{data} (with $y = 1$) or from p_g (with $y = 0$). The minimax game in Equation (1) can now be reformulated as following

-

$$
\begin{aligned}
C(G) &= \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}} \log D_G^*(x) + \mathbb{E}_{z \sim p_z} \log(1 - D_G^*(G(z))) \\
&= \mathbb{E}_{x \sim p_{data}} \log D_G^*(x) + \mathbb{E}_{x \sim p_g} \log(1 - D_G^*(x)) \\
&= \mathbb{E}_{x \sim p_{data}} \log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} + \mathbb{E}_{x \sim p_g} \log \frac{p_g(x)}{p_{data}(x) + p_g(x)} \\
&= -\log(4) + KL\left(p_{data} \parallel \frac{p_{data} + p_g}{2}\right) + KL\left(p_g \parallel \frac{p_{data} + p_g}{2}\right) \\
&= -\log(4) + 2JSD(p_{data} \parallel p_g) \qquad (2)
\end{aligned}
$$

Global optimality of $p_g = p_{data}$

Theorem 3

The global minimum of the training criterion $C(G)$ in Equation (2) is achieved iff $p_g = p_{data}$. At that point, $C(G)$ achieves the value of $-\log(4)$.

Proof.

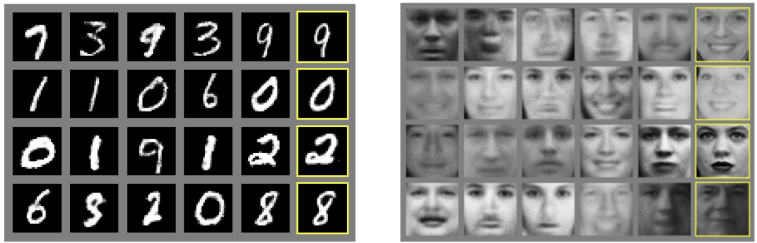
For $p_g = p_{data}$, it is obvious $D^*_G(x) = 1/2$ from Equation (1). Hence, by inspecting Equation (2) at $D^*_G(x) = 1/2$, we find $C(G) = \log \frac{1}{2} + \log \frac{1}{2} = -\log(4)$. To see that this is the best possible value of $C(G)$, reached only for $p_g = p_{data}$, observe that

$$\mathbb{E}_{x \sim p_{data}} - \log 2 + \mathbb{E}_{x \sim p_g} - \log 2 = -\log 4$$

and since Jensen-Shannon divergence between two distributions is always non-negative, and zero iff they are equal, we conclude from Equation (2) that $C^* = -\log(4)$ is the global minimum of $C(G)$ and that the only solution is $p_g = p_{data}$, *i.e.* the generative model perfectly replicating the data distribution. ■

Generative Adversarial Nets

Generated samples



Nearest neighbor from training set

Generative Adversarial Nets

Generated samples (CIFAR-10)

Nearest neighbor from training set

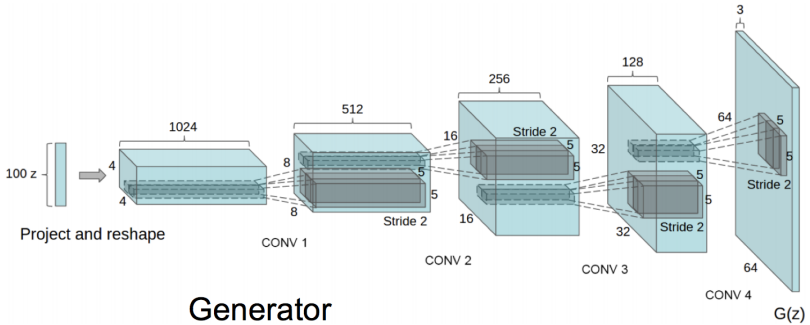
DCGAN

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

DCGAN



Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

DCGAN

Samples from the model look amazing!

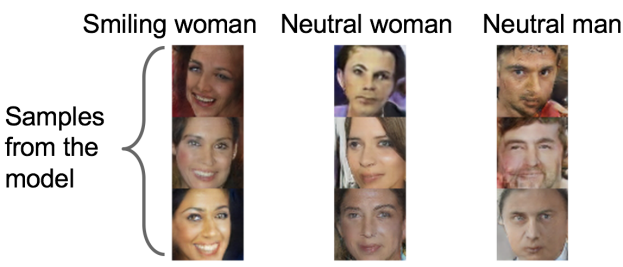
Radford et al, ICLR 2016

DCGAN

Interpolating between random points in latent space

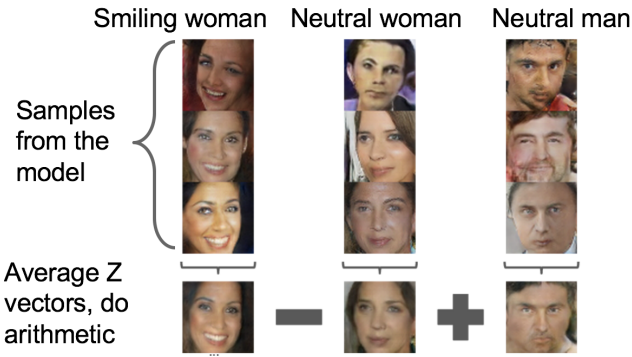
Radford et al, ICLR 2016

GAN vector math



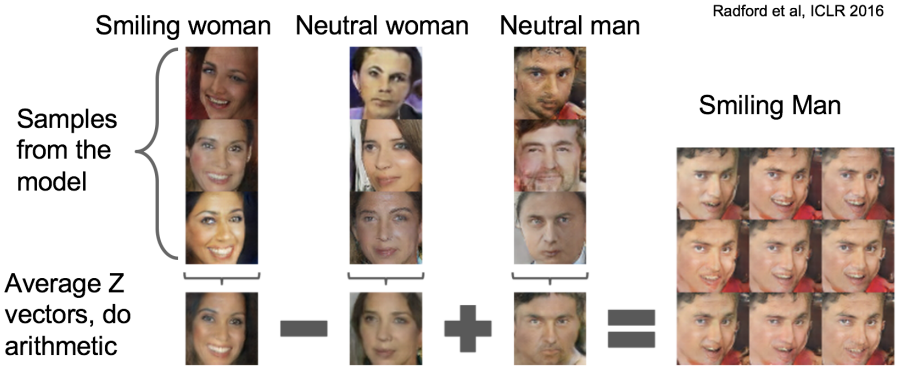
Radford et al, ICLR 2016

GAN vector math

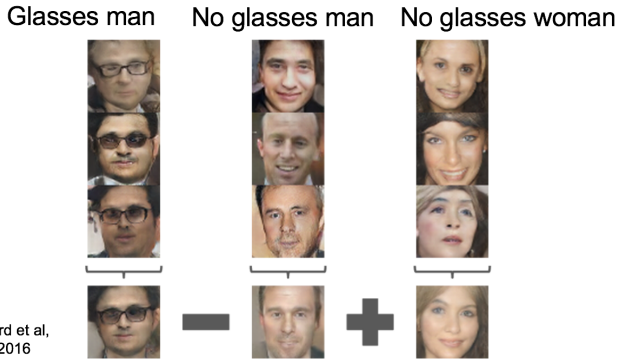


Radford et al, ICLR 2016

GAN vector math



GAN vector math



Radford et al, ICLR 2016

GAN vector math

“The GAN Zoo”

See also: https://github.com/soumith/ganhacks for tips and tricks for trainings GANs

- 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

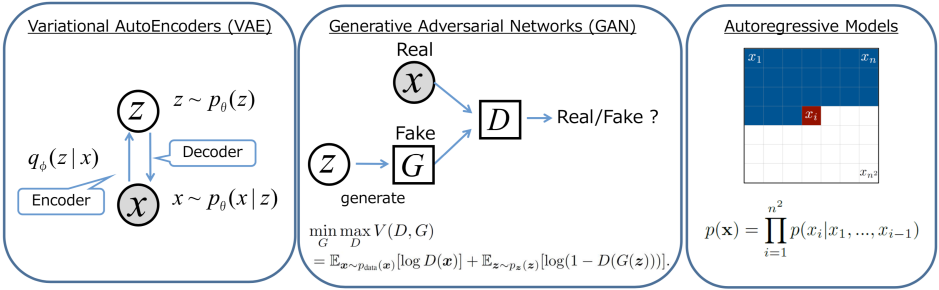
https://github.com/hindupuravinash/the-gan-zoo

Outline

Comparisons[4]

Image Generation Models

-Three image generation approaches are dominating the field:



	VAE	GAN	Autoregressive Models
Pros.	- Efficient inference with approximate latent variables.	- generate sharp image. - no need for any Markov chain or approx networks during sampling.	- very simple and stable training process - currently gives the best log likelihood. - tractable likelihood
Cons.	- generated samples tend to be blurry.	- difficult to optimize due to unstable training dynamics.	- relatively inefficient during sampling

(cf. https://openai.com/blog/generative-models/)

[4] from Yohei Sugawara

Comparisons[5]

- **Variational Autoencoders** allow us to perform both learning and efficient Bayesian inference in sophisticated probabilistic graphical models with latent variables. However, their generated samples tend to be slightly blurry.

[5] https://openai.com/blog/generative-models/

Comparisons[5]

- **Variational Autoencoders** allow us to perform both learning and efficient Bayesian inference in sophisticated probabilistic graphical models with latent variables. However, their generated samples tend to be slightly blurry.

- **GANs** currently generate the sharpest images but they are more difficult to optimize due to unstable training dynamics.

[5] https://openai.com/blog/generative-models/

Comparisons[5]

- **Variational Autoencoders** allow us to perform both learning and efficient Bayesian inference in sophisticated probabilistic graphical models with latent variables. However, their generated samples tend to be slightly blurry.
- **GANs** currently generate the sharpest images but they are more difficult to optimize due to unstable training dynamics.
- **PixelRNNs** have a very simple and stable training process (softmax loss) and currently give the best log likelihoods (that is, plausibility of the generated data). However, they are relatively inefficient during sampling and don't easily provide simple low-dimensional codes for images.
- All of these models are active areas of research and we are eager to see how they develop in the future!

[5] https://openai.com/blog/generative-models/