Generative Adversarial Networks

Hwann-Tzong Chen

National Tsing Hua University

20th March 2017

Quora: What are some recent and potentially upcoming breakthroughs in deep learning?

Yann LeCun, Director of AI Research at Facebook and Professor at NYU:

"The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This is an idea that was originally proposed by Ian Goodfellow when he was a student with Yoshua Bengio at the University of Montreal (he since moved to Google Brain and recently to OpenAI).

This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion."

(Ian Goodfellow moved back to Google Brain recently.)

Generative Adversarial Nets (GANs)

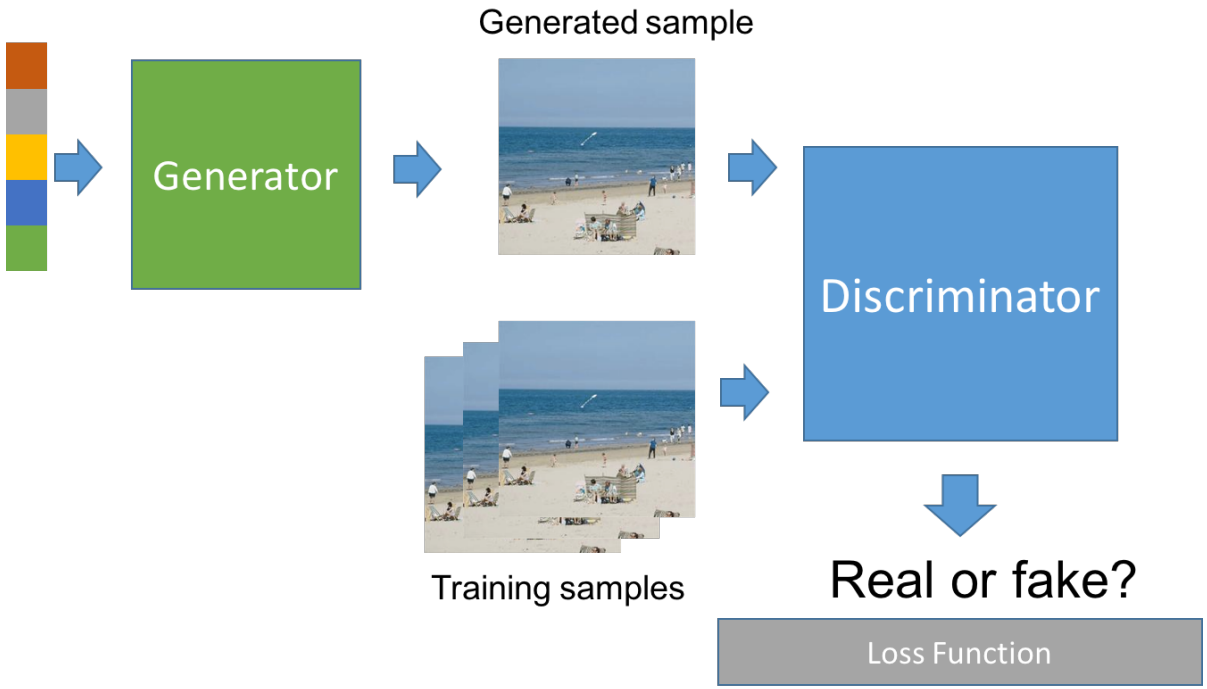


Figure 1: Generative Adversarial Nets.

Unsupervised Learning

Today's Topics

1. GAN
   Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio (NIPS 2014): "Generative Adversarial Networks".

2. DCGAN
   Alec Radford, Luke Metz, Soumith Chintala (ICLR 2015): "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks."

3. Improved GAN
   Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen: "Improved Techniques for Training GANs."

Today's Topics

1. ALI
   Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, Aaron Courville (ICLR 2017): "Adversarial Learned Inference"

2. WGAN
   Martin Arjovsky, Leon Bottou (ICLR 2017): "Towards Principled Methods for Training Generative Adversarial Networks"

   Martin Arjovsky, Soumith Chintala, Léon Bottou (2017): "Wasserstein GAN"

Generative Adversarial Nets

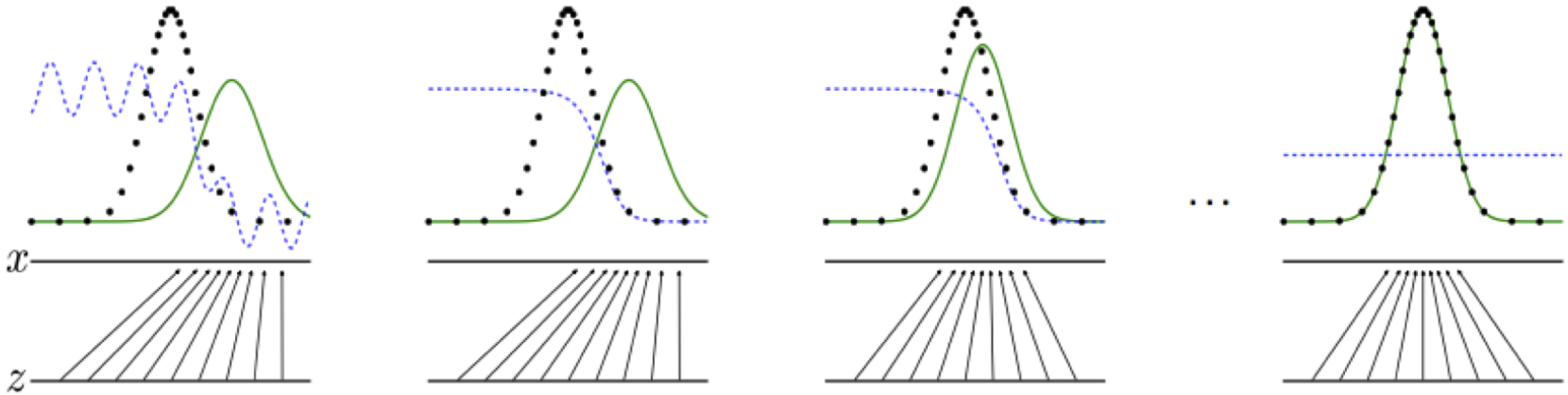
Ian J. Goodfellow, Jean Pouget-Abadie*, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair†, Aaron Courville, Yoshua Bengio‡
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

GAN Optimization

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



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.

Common Problems of Training GANs

1. Mode collapsing
2. Unstable, does not converge
3. The results are blurry
4. Lack of assessment

DCGAN

1. All convolutional
2. No spatial pooling
3. Eliminating fully connected
4. Batch normalization
5. ReLU for generator (except for the output using tanh)
6. Leaky ReLU for discriminator
7. Batch normalization not applied to generator output layer and discriminator input layer

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

Alec Radford & Luke Metz
indico Research
Boston, MA
{alec,luke}@indico.io

Soumith Chintala
Facebook AI Research
New York, NY
soumith@fb.com

ABSTRACT

In recent years, supervised learning with convolutional networks (CNNs) has seen huge adoption in computer vision applications. Comparatively, unsupervised learning with CNNs has received less attention. In this work we hope to help bridge the gap between the success of CNNs for supervised learning and unsupervised learning. We introduce a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and demonstrate that they are a strong candidate for unsupervised learning. Training on various image datasets, we show convincing evidence that our deep convolutional adversarial pair learns a hierarchy of representations from object parts to scenes in both the generator and discriminator. Additionally, we use the learned features for novel tasks - demonstrating their applicability as general image representations.

DCGAN Structure

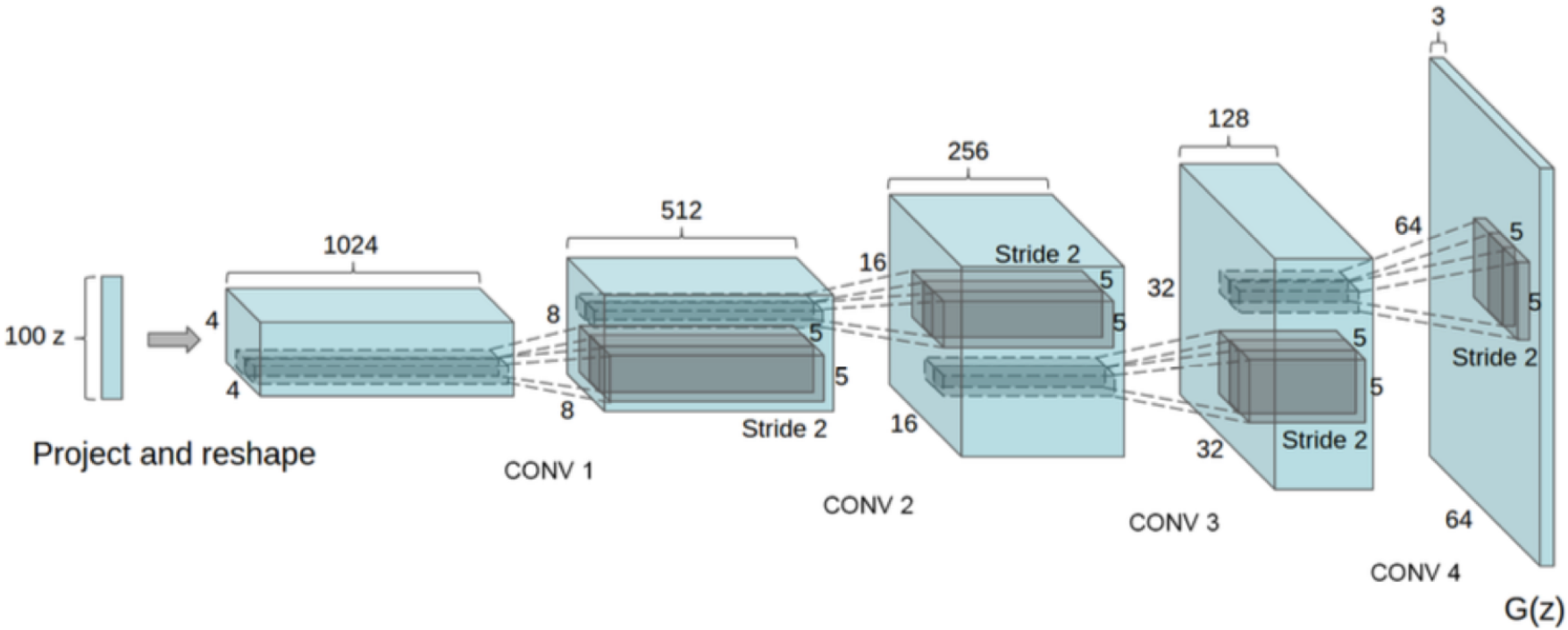


Figure 2: Fractionally-stride convolution (not deconvolution).

DCGAN Results

Figure 3: Just memorizing the training data?

DCGAN Results—Morphing

Figure 4: Walking in the latent space.

Visualizing the Discriminator Features

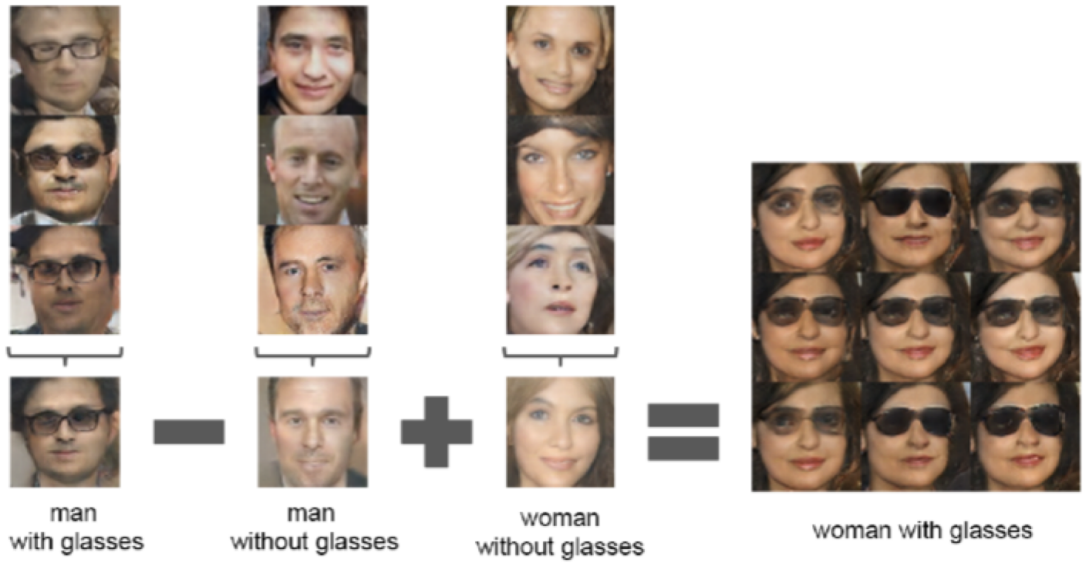
Figure 5: With guided backpropagation.

Forgetting to Draw Objects

Figure 6: Dropping out "window" filters.

Visual Arithmetic

Get several **z**'s and compute their mean;
Feed the mean into the generator will get a mean image;
Do the arithmetic on the mean vectors of different groups of **z**'s and get a new **z**;
Add some noise to the new **z** and feed them into the generator.



Improved Techniques for Training GANs

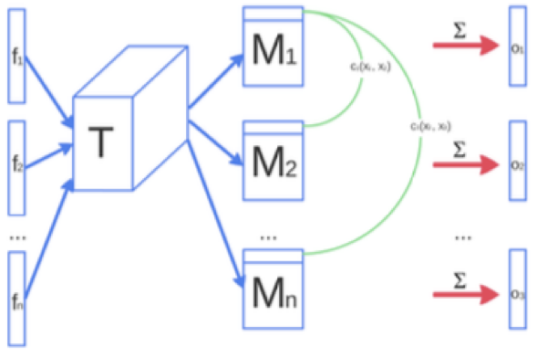
Tim Salimans tim@openai.com
Ian Goodfellow ian@openai.com
Wojciech Zaremba woj@openai.com
Vicki Cheung vicki@openai.com
Alec Radford alec.radford@gmail.com
Xi Chen peter@openai.com

Abstract

We present a variety of new architectural features and training procedures that we apply to the generative adversarial networks (GANs) framework. We focus on two applications of GANs: semi-supervised learning, and the generation of images that humans find visually realistic. Unlike most work on generative models, our primary goal is not to train a model that assigns high likelihood to test data, nor do we require the model to be able to learn well without using any labels. Using our new techniques, we achieve state-of-the-art results in semi-supervised classification on MNIST, CIFAR-10 and SVHN. The generated images are of high quality as confirmed by a visual Turing test: our model generates MNIST samples that humans cannot distinguish from real data, and CIFAR-10 samples that yield a human error rate of 21.3%. We also present ImageNet samples with unprecedented resolution and show that our methods enable the model to learn recognizable features of ImageNet classes.

Some Tricks

1. Feature matching: to address the instability issue.
   A new objective for the generator that prevents it from overtraining on the current discriminator. The new objective requires the generator to generate data that matches the expected value of the features on an intermediate layer of the discriminator.
2. Minibatch discrimination: to prevent from collapsing to a single point by detecting closeness in a minibatch.



Other Tricks

1. Historical averaging: include a term $\|\theta - \frac{1}{t}\sum_i^t \theta[i]\|^2$ on parameters in the cost function.
2. One-sided label smoothing: The original label smoothing trick changes the target values 1 and 0 to $\alpha = 0.9$ and $\beta = 0.1$. The optimal discriminator becomes

$$D(\mathbf{x}) = \frac{\alpha p_{\text{data}}(\mathbf{x}) + \beta p_{\text{model}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})}.$$

   One-sided label smoothing is to set $\beta = 0$.
3. Virtual batch normalization

   Batch normalization causes the output of a neural network for an input example $\mathbf{x}$ to be highly dependent on several other inputs $\mathbf{x}'$ in the same minibatch. Virtual batch normalization avoids this problem by normalizing each example $\mathbf{x}$ using the statistics collected on a reference batch of examples that are chosen once and fixed at the start of training. It is used only in the generator network.

Inception Score

GANs lack an objective function/evaluation metric for comparing performance of different models.

1. Human annotator?
2. Inception score: Apply the Inception model to every generated image to get the conditional label distribution $p(y|\mathbf{x})$. Images that contain meaningful objects should have a conditional label distribution $p(y|\mathbf{x})$ with low entropy. Moreover, we expect the model to generate varied images, so the marginal $\int p(y|\mathbf{x} = G(\mathbf{z}))\, dz$ should have high entropy. Combine these two requirements to get a metric like

$$\exp(\mathbb{E}_{\mathbf{x}} \mathrm{KL}(p(y|\mathbf{x}) \| p(y))) .$$

Adversarial Learned Inference (ALI)

Adversarially Learned Inference

Vincent Dumoulin[*]
vincent.dumoulin@umontreal.ca

Ishmael Belghazi[*]
ishmael.belghazi@gmail.com

Ben Poole [†]
poole@cs.stanford.edu

Alex Lamb[*]
alex6200@gmail.com

Martin Arjovsky[*]
martinarjovsky@gmail.com

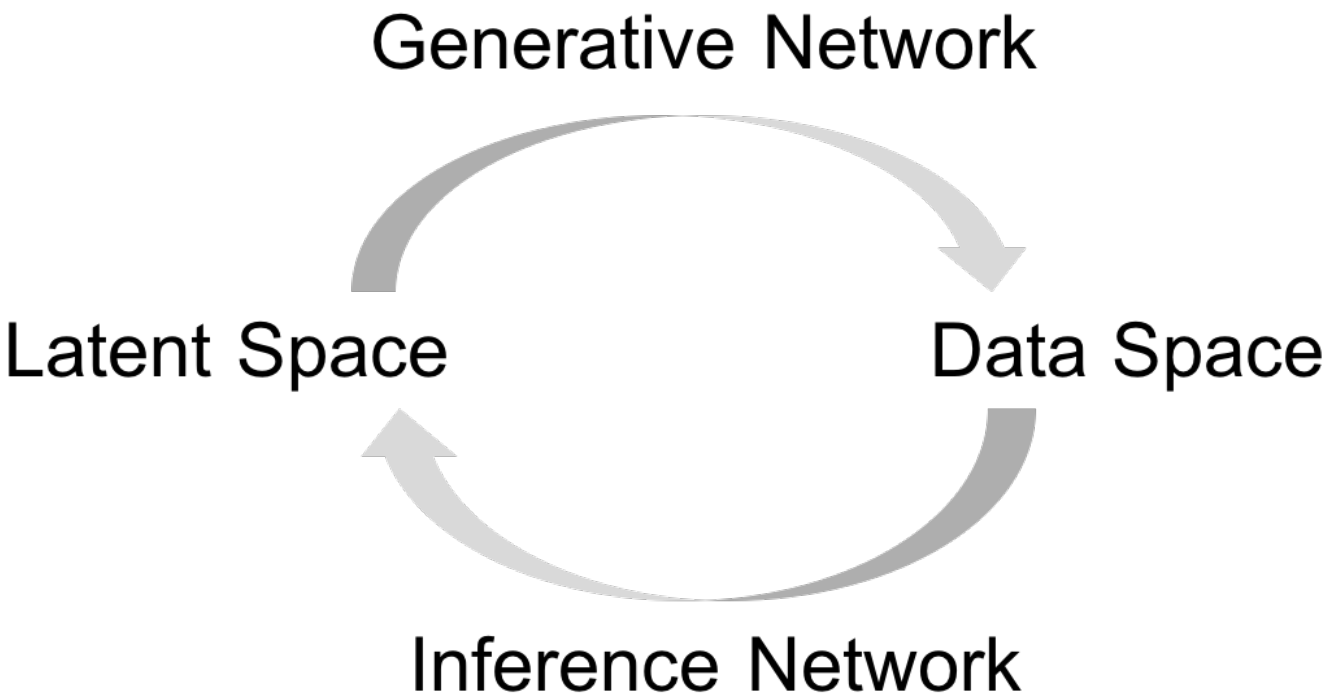
Olivier Mastropietro[*]
oli.mastro@gmail.com

Aaron Courville[*], CIFAR Fellow
aaron.courville@gmail.com

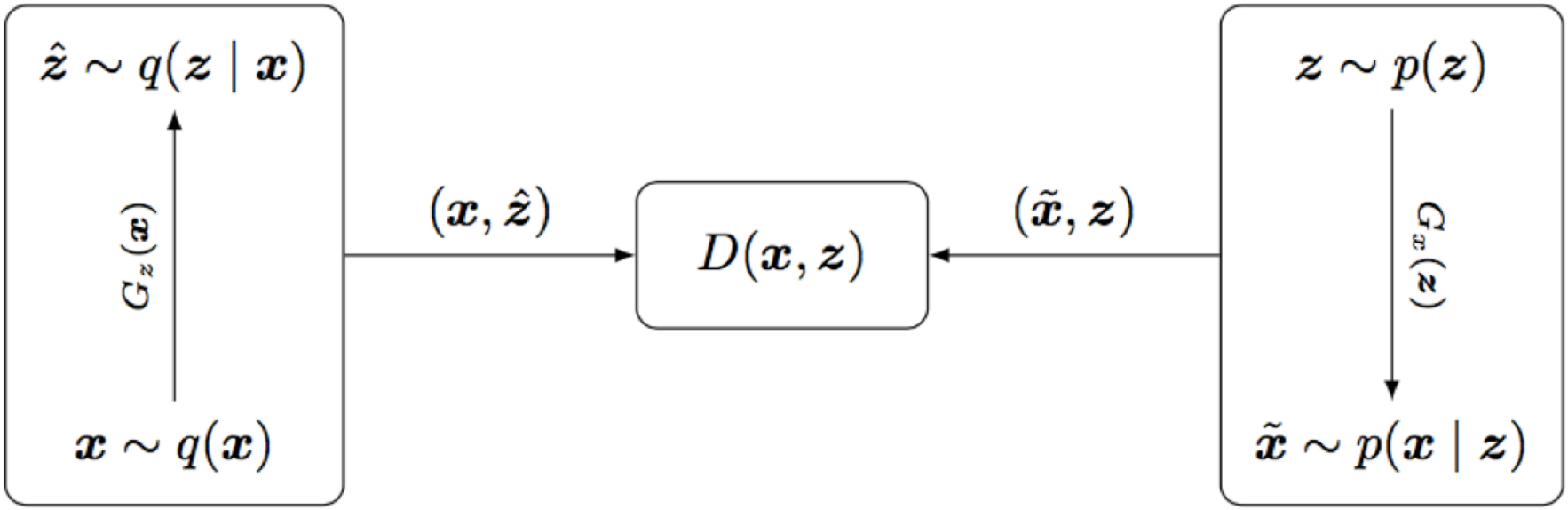
Abstract

We introduce the adversarially learned inference (ALI) model, which jointly learns a generation network and an inference network using an adversarial process. The generation network maps samples from stochastic latent variables to the data space while the inference network maps training examples in data space to the space of latent variables. An adversarial game is cast between these two networks and a discriminative network that is trained to distinguish between joint latent/data-space samples from the generative network and joint samples from the inference network. We illustrate the ability of the model to learn mutually coherent inference and generation networks through the inspections of model samples and reconstructions and confirm the usefulness of the learned representations by obtaining a performance competitive with other recent approaches on the semi-supervised SVHN task.

Adversarial Learned Inference (ALI)



Encoder, Decoder



The encoder joint distribution: $q(\mathbf{x}, \mathbf{z}) = q(\mathbf{x})q(\mathbf{z}|\mathbf{x})$
The decoder joint distribution: $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$

Game Value Function of ALI

$$\min_G \max_D V(D,G) = \mathbb{E}_{q(\boldsymbol{x})}[\log(D(\boldsymbol{x}, G_z(\boldsymbol{x})))] + \mathbb{E}_{p(\boldsymbol{z})}[\log(1 - D(G_x(\boldsymbol{z}), \boldsymbol{z}))]$$

$$= \iint q(\boldsymbol{x})q(\boldsymbol{z} \mid \boldsymbol{x}) \log(D(\boldsymbol{x}, \boldsymbol{z}))d\boldsymbol{x}d\boldsymbol{z}$$

$$+ \iint p(\boldsymbol{z})p(\boldsymbol{x} \mid \boldsymbol{z}) \log(1 - D(\boldsymbol{x}, \boldsymbol{z}))d\boldsymbol{x}d\boldsymbol{z}.$$

Sampling from $q(\boldsymbol{z} \mid \boldsymbol{x})$ amounts to drawing an $\boldsymbol{x}$ from the data distribution and propagating the sample through the encoder network $(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z) = G_z(\boldsymbol{x})$, then generating $\hat{\boldsymbol{z}} \sim \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z)$. Sampling from $p(\boldsymbol{x} \mid \boldsymbol{z})$ amounts to sampling from $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and propagating the sample through the decoder network to recover $\tilde{\boldsymbol{x}}$.

ALI Training

Algorithm 1 The ALI training procedure.

$\theta_g, \theta_d \leftarrow$ initialize network parameters

repeat

$\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(M)} \sim q(\boldsymbol{x})$ ▷ Draw M samples from the dataset and the prior

$\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(M)} \sim p(\boldsymbol{z})$

$\hat{\boldsymbol{z}}^{(i)} \sim q(\boldsymbol{z} \mid \boldsymbol{x} = \boldsymbol{x}^{(i)}), \quad i = 1, \ldots, M$ ▷ Sample from the conditionals

$\tilde{\boldsymbol{x}}^{(j)} \sim p(\boldsymbol{x} \mid \boldsymbol{z} = \boldsymbol{z}^{(j)}), \quad j = 1, \ldots, M$

$\rho_q^{(i)} \leftarrow D(\boldsymbol{x}^{(i)}, \hat{\boldsymbol{z}}^{(i)}), \quad i = 1, \ldots, M$ ▷ Compute discriminator predictions

$\rho_p^{(j)} \leftarrow D(\tilde{\boldsymbol{x}}^{(j)}, \boldsymbol{z}^{(j)}), \quad j = 1, \ldots, M$

$\mathcal{L}_d \leftarrow -\frac{1}{M}\sum_{i=1}^{M} \log(\rho_q^{(i)}) - \frac{1}{M}\sum_{j=1}^{M} log(1 - \rho_p^{(j)})$ ▷ Compute discriminator loss

$\mathcal{L}_g \leftarrow -\frac{1}{M}\sum_{i=1}^{M} \log(1 - \rho_q^{(i)}) - \frac{1}{M}\sum_{j=1}^{M} \log(\rho_p^{(j)})$ ▷ Compute generator loss

$\theta_d \leftarrow \theta_d - \nabla_{\theta_d}\mathcal{L}_d$ ▷ Gradient update on discriminator network

$\theta_g \leftarrow \theta_g - \nabla_{\theta_g}\mathcal{L}_g$ ▷ Gradient update on generator networks

until convergence

Latent Space Interpolation

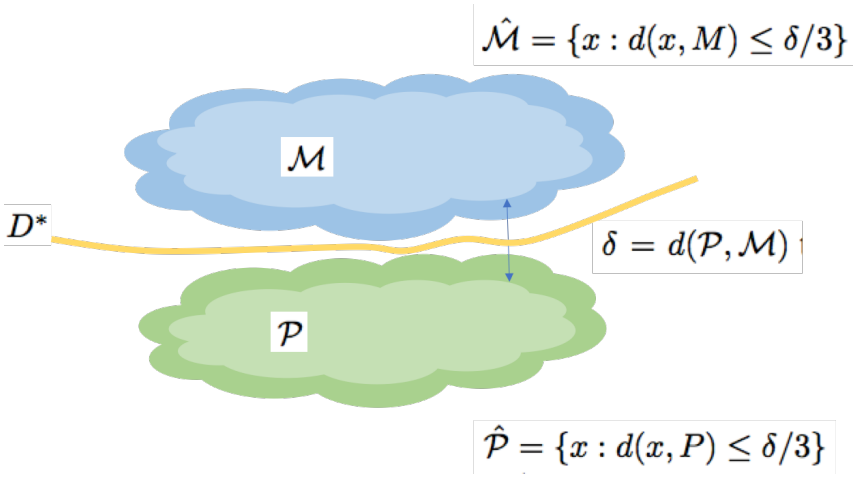
TOWARDS PRINCIPLED METHODS FOR TRAINING GENERATIVE ADVERSARIAL NETWORKS

Martin Arjovsky
Courant Institute of Mathematical Sciences
martinarjovsky@gmail.com

Léon Bottou
Facebook AI Research
leonb@fb.com

Why are GANs hard to train?
Mode collapse
Unstable training behavior
Loss function lacking information about convergence
Theory?

Perfect Discrimination



The discriminator is trained to maximize

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

Solution?

Softer metrics: i) add noise; ii) Wasserstein metric.

Wasserstein GAN

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

Summary

Prof. Yoshua Bengio's Talk at ECCV 2016

The next big challenge: unsupervised learning

1. Recent progress mostly in supervised DL
2. Real technical challenges for unsupervised DL
3. Potential benefits
   1. Exploit tons of unlabeled
   2. Answer new questions about the variables observed
   3. Regularizer – transfer learning – domain adaptation
   4. Easier optimization (local training signal)
   5. Structured outputs (compositional, joint distributions of many things)
   6. Necessary for RL without given model or domain simulator

Prof. Jitendra Malik's Talk at ACCV 2016

1. Prediction is a sign of understanding
2. Planning actions
3. What has been responsible for recent AI success?
    1. Big computing
    2. Big data
    3. Big annotation
    4. Big simulation: Game scenarios can be simulated, but it's not so easy in other settings
4. Being interested in computer vision tasks where feedback is key to the solution. This is a very natural way to capture "context".

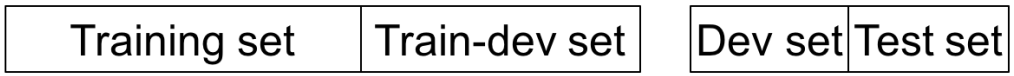
References

1. Soumith Chintala @ FAIR, "A path to unsupervised learning through adversarial networks"
2. Ruslan Salakhutdinov @ Apple & CMU, "Learning deep generative models"

Nuts and Bolts of Applying Deep Learning (Andrew Ng)

@ Deep Learning School, September 2016
https://www.youtube.com/watch?v=F1ka6a13S9I

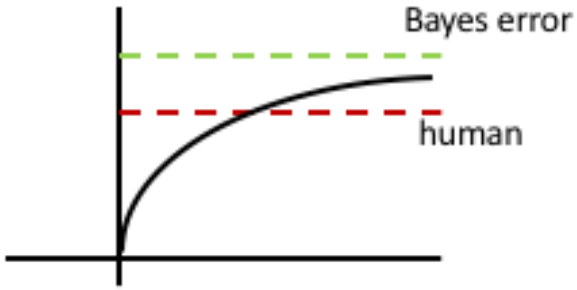
Bias and variance



Make sure Dev+Test are from the same distribution

Human level error	1%	↕ Bias
Training set error	10%	↕ Variance
Train-dev set error	10%	↕ Train-test mismatch
Dev set error	10%	↕ Overfitting of dev set
Test set error	10.2%	

Nuts and Bolts of Applying Deep Learning (Andrew Ng)



AI passes human level:
Using subset of data

Human level error as a proxy to Bayes error

Training set error high? Y → Bigger model / Train longer / New model architecture

Train-dev set error high? Y → More data / Regularization / New model architecture

Dev set error high? Y → More data from the similar distribution / Data synthesis / New model architecture

Test set error Y → Get more dev set data

Thank You