CSCI 599: Deep Learning and its Applications

Lecture 7

Spring 2019
Joseph J. Lim

Disclaimer

- This course is taught for the 2nd time @ USC. This course is 599, and thus an **experimental** course.
- The syllabus, course policy, and grading details **may change** over the semester (**check website!**)
- If you prefer a well-structured course, this is **NOT** a course for you, and I encourage you to take the course next year. We really mean this.
- But, it will be **fun** and **challenging**!

Important Dates

Subject to change!

- ~~Entrance exam: 1/15~~
- **Assignment 1: 2/20 (tomorrow)**
- **Midterm: 2/26 (next week)**
- Project meeting with Instructor #1: 3/6 — 3/8
- Assignment 2: 3/26
- Project meeting with Instructor #2: 4/1 — 4/3
- Project meeting with TA: 2 times (arranged later)
- Final presentation: 4/23 5-9:00pm **4 hours**

Course Project

Subject to change!

- Computational resource (**be conser**
  $150 Google Cloud credit per student
  $125 Amazon AWS credit per student
- Tentative Schedule for Project
  - ~~Week 5 (2/5): Course Project Team~~
  - **Week 9 (3/5): Course Project Proposal (in 2weeks)**
  - Week 13 (4/2): Mid-report
  - Week 16 (4/23): Project Presentation (5-9pm) + Report

Assignment 1

- DUE February 20th, 2019
- Collaboration is OK! Please list all collaborators' names.
- No code sharing :)

Proposal

- 1 page write-up (refer to our Piaza post)
- Google slide

Proposal (write-up)

- 1 page write-up (refer to our Piaza post)
  - Goal
  - Motivation
  - Problem formulation (e.g., input/output)
  - Milestones
  - Expected approach + results

Proposal (google slide)

- 1 Google slide (1-3 slides)
  - 70 seconds spotlight (during Lecture 7)
  - 60 teams => 70mins
  - Brief motivation
  - Problem formulation (input & output)
  - Expect approach

Today's agenda

- Sponsor Talk: Neudesic
- Sample midterm
- Part 1: Generative Adversarial Networks
- Part 2: Variational Autoencoders
- Part 3: PixelRNN and PixelCNN

Today's agenda

- Sponsor Talk: Neudesic
- Sample midterm
- Part 1: Generative Adversarial Networks
- Part 2: Variational Autoencoders
- Part 3: PixelRNN and PixelCNN

Today's agenda

- Sponsor Talk: Neudesic
- Sample midterm
- Part 1: Generative Adversarial Networks
- Part 2: Variational Autoencoders
- Part 3: PixelRNN and PixelCNN

Section 1

Q

1. Overfitting corresponds to:
   a. High variance of the estimator
   b. High bias of the estimator
   c. Neither a. nor b.
   d. Could be either a. or b. depending on a situation
   e. Both a. and b.

A

1. Overfitting corresponds to:
   a. High variance of the estimator
   b. High bias of the estimator
   c. Neither a. nor b.
   d. Could be either a. or b. depending on a situation
   e. Both a. and b.

A	B	C	D	E
48%	20%	9%	18%	5%

Q

11. In LeNet-5 (3 conv layers + 2 fully connected layers and poolings), by using **max pooling layers** one can achieve: i) reduce the number of parameters in a neural network ii) reduce spatial extent of activation map in a neural network iii) reduce computation of the forward pass.
    a. i
    b. ii
    c. ii and iii
    d. I and iii
    e. i, ii and iii

A

11. In LeNet-5 (3 conv layers + 2 fully connected layers and poolings), by using **max pooling layers** one can achieve: i) reduce the number of parameters in a neural network ii) reduce spatial extent of activation map in a neural network iii) reduce computation of the forward pass.
    a. i
    b. ii
    c. ii and iii
    d. I and iii
    e. i, ii and iii

A	B	C	D	**E**
4%	1%	2%	33%	**60%**

Q

13. When you compute batch normalization (in its original paper) for a tensor of an image activation map which dimensions do you have to average over:
    a. Batch dimension and spatial dimension
    b. Batch dimension and filter dimension
    c. Batch dimension only
    d. Spatial dimensions only
    e. All dimensions except for the batch dimension

A

13. When you compute batch normalization (in its original paper) for a tensor of an image activation map which dimensions do you have to average over:
    a. Batch dimension and spatial dimension
    b. Batch dimension and filter dimension
    c. Batch dimension only
    d. Spatial dimensions only
    e. All dimensions except for the batch dimension

A	B	C	D	E
24%	5%	51%	12%	8%

Q

14. Which of the statements is **Not True** when comparing ReLU to the sigmoid function:
    a. ReLU sparsify the activation maps (by making negative activations to zero) and can additionally speed up computations
    b. ReLU is faster to compute in general
    c. Dead activations are more likely to appear when using ReLU
    d. The problem of exploding gradients is more likely to appear with sigmoid
    e. None of the above statements

A

14. Which of the statements is **Not True** when comparing ReLU to the sigmoid function:
    a. ReLU sparsify the activation maps (by making negative activations to zero) and can additionally speed up computations
    b. ReLU is faster to compute in general
    c. Dead activations are more likely to appear when using ReLU
    d. The problem of exploding gradients is more likely to appear with sigmoid
    e. None of the above statements

A	B	C	**D**	E
0%	0%	22%	**57%**	21%

Q

19. Artificial neural networks (ANN) has existed for a very long time (back to 1960-s). Only after the famous success in ILSVRC12 image recognition competition, ANN (which became known as Deep Learning) starts to dominate multiple domains in artificial intelligence.

Which of the following are **True** about factors causing such late success?

i) The availability of huge amount of labeled data: ImageNet dataset became available just a few years before this success
ii) The availability of computational resources, such as large memory, powerful GPUs
iii) The availability of efficient optimization techniques: back-propagation was invented within ten years before the competition
iv) The late invention of the Convolutional Neural Networks: CNN is invented in 2000-s

a. i, ii
b. ii, iv
c. iii, iv
d. ii, iii, iv
e. all of the above

A

19. Artificial neural networks (ANN) has existed for a very long time (back to 1960-s). Only after the famous success in ILSVRC12 image recognition competition, ANN (which became known as Deep Learning) starts to dominate multiple domains in artificial intelligence.

Which of the following are **True** about factors causing such late success?

i) The availability of huge amount of labeled data: ImageNet dataset became available just a few years before this success
ii) The availability of computational resources, such as large memory, powerful GPUs
iii) The availability of efficient optimization techniques: back-propagation was invented within ten years before the competition
iv) The late invention of the Convolutional Neural Networks: CNN is invented in 2000-s

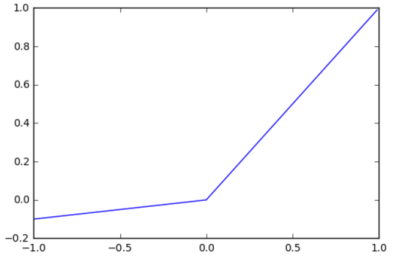
a. i, ii
b. ii, iv
c. iii, iv
d. ii, iii, iv
e. all of the above

A	B	C	D	E
60%	5%	0%	4%	31%

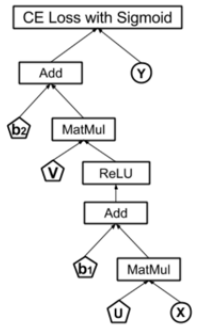
Section 2

Activation function

Leaky ReLU



Back-Propagation



$$H = ReLU(U^T \cdot X + b_1)$$
$$\hat{Y} = Sigmoid(V^T \cdot H + b_2)$$
$$J(\hat{Y}, Y) = Loss_{CE}(\hat{Y}, Y) = Y \cdot log(\hat{Y}) + (1 - Y) \cdot log(1 - \hat{Y})$$

Today's agenda

- Sponsor Talk: Neudesic
- Sample midterm
- Part 1: Generative Adversarial Networks
- Part 2: Variational Autoencoders
- Part 3: PixelRNN and PixelCNN

Motivation

- **Want:** Sample from complex, high-dimensional distribution. Too difficult to directly do this!

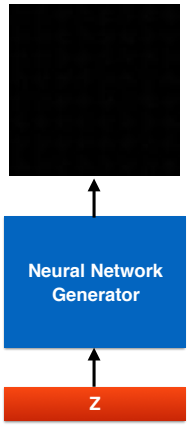
Inspired by Stanford CS231n

Motivation

- **Want:** Sample from complex, high-dimensional distribution. Too difficult to directly do this!
- **Workaround:** Learn the transformation from a simpler distribution, e.g. random noise, to the desired distribution

Inspired by Stanford CS231n

NN as the transformation



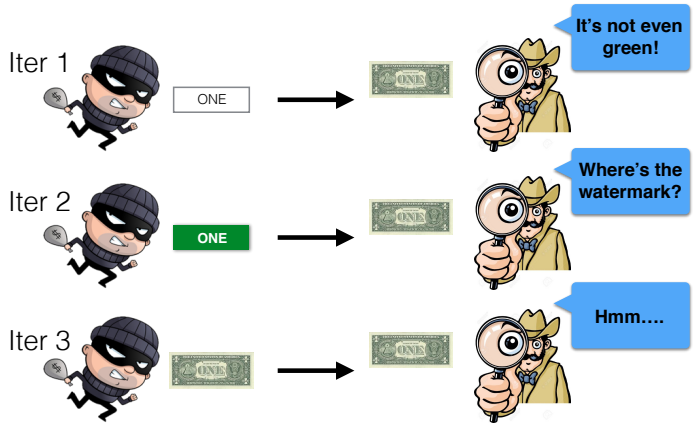
Output: Sample from the distribution (Sampled across training epochs)

Neural Network To represent the transformation

Input: Random Noise

Inspired by Stanford CS231n

GAN, the two player game



pic credit: http://cliparting.com/free-detective-clipart-39056/
pic credit: https://fateclick.com/dream/dreaming-about-thieves

GAN, the two player game

- **Generator:** Generate real-looking images to fool the discriminator

Inspired by Stanford CS231n

GAN, the two player game

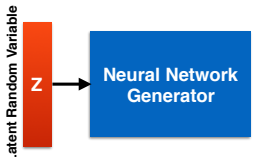
- **Generator:** Generate real-looking images to fool the discriminator
- **Discriminator:** Distinguish between real (GT) images and generated images

Inspired by Stanford CS231n

GAN, the two player game

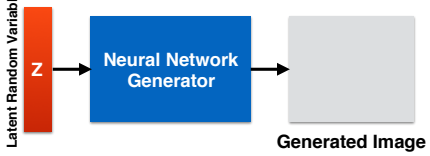
Inspired by https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

GAN, the two player game



Inspired by https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

GAN, the two player game

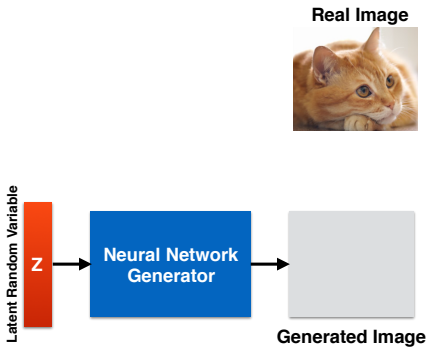


Inspired by https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

GAN, the two player game

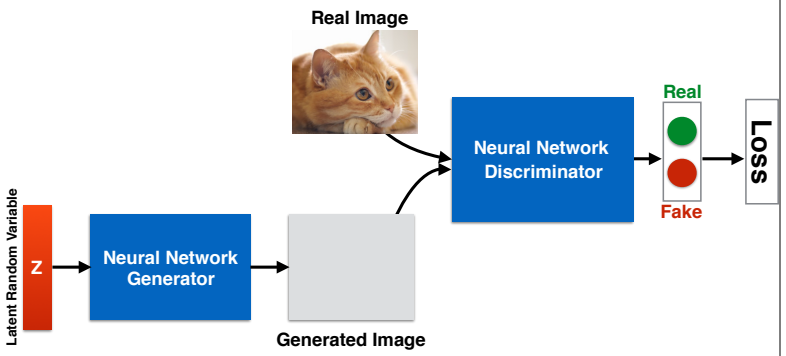
Inspired by https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

GAN, the two player game



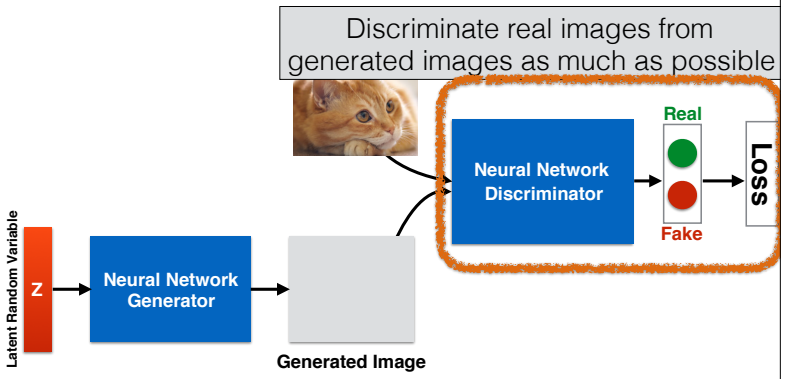
Inspired by https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

GAN, the two player game



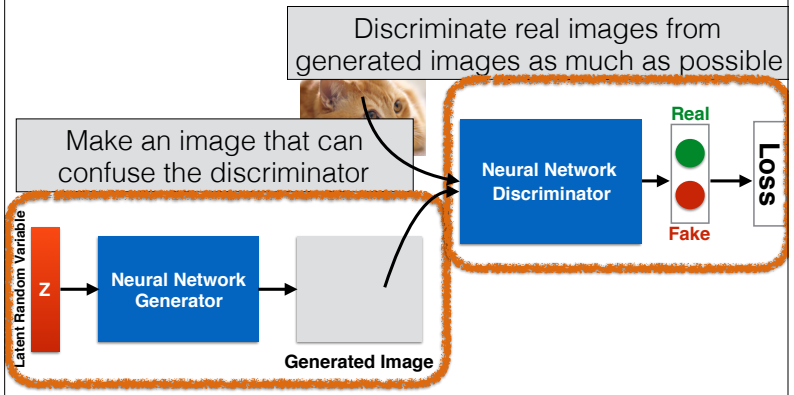
Inspired by https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

GAN, the two player game



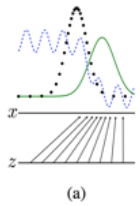
Inspired by https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

GAN, the two player game



Inspired by https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

GAN, the two player game



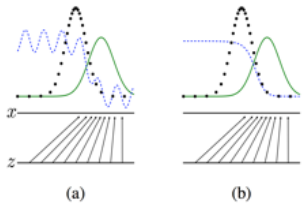
(a)

Here, $\boldsymbol{z}$ is uniformly sampled

Discriminative distribution
Generative distribution
Data distribution

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

GAN, the two player game



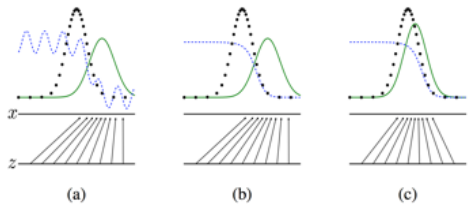
(a) (b)

Here, $\boldsymbol{z}$ is uniformly sampled

Discriminative distribution
Generative distribution
Data distribution

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

GAN, the two player game



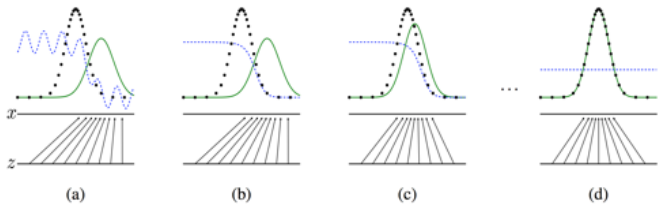
(a) (b) (c)

Here, $\boldsymbol{z}$ is uniformly sampled

Discriminative distribution
Generative distribution
Data distribution

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

GAN, the two player game



(a) (b) (c) (d)

Here, $\boldsymbol{z}$ is uniformly sampled

Discriminative distribution
Generative distribution
Data distribution

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

GAN, the two player game

- The ***minimax*** game:

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

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

GAN, the two player game

- The ***minimax*** game:

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

output of discriminator for real data x

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

GAN, the two player game

- The ***minimax*** game:

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

output of discriminator for real data x

output of discriminator for generated data G(z)

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

GAN, the two player game

- The ***minimax*** game:

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

output of discriminator for real data x

output of discriminator for generated data G(z)

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

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

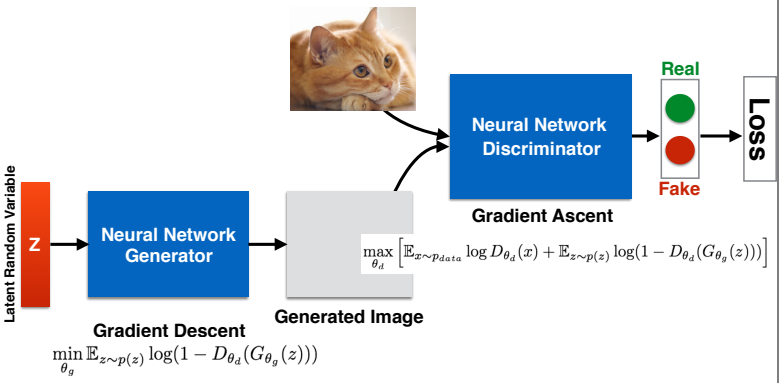
GAN, the two player game

1: Real **0:** Fake

- **Generator: Minimize** the objective such that D(G(z)) is close to 1
- **Discriminator: Maximize** the objective such that D(x) is close to 1 and D(G(z)) is close to 0

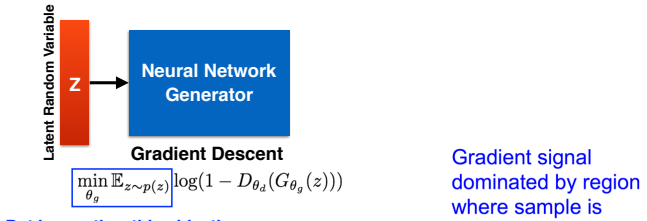
Inspired by Stanford CS231n

GAN, the two player game



Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

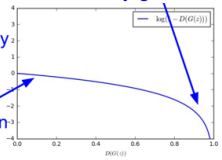
GAN, the two player game



But in practice, this objective is hard to optimize!

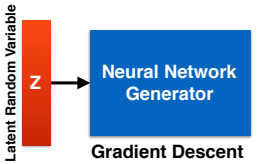
Gradient signal dominated by region where sample is already good

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



Inspired by Stanford CS231n

GAN, the two player game

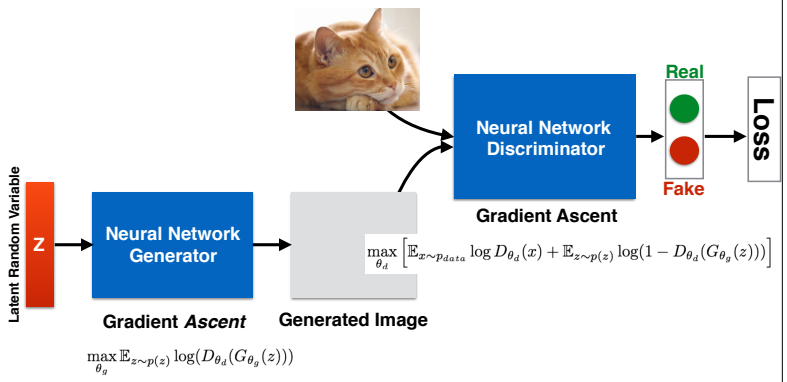


$\max_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$

Instead: We do Gradient *Ascent* on generator on the revised objective

Inspired by Stanford CS231n

GAN, the two player game



Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Training GANs: Algorithm

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.

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Training GANs: Algorithm

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:

Design Choice

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

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.

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Other Distance Functions

Wasserstein-1 or Earth Mover Distance:

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

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

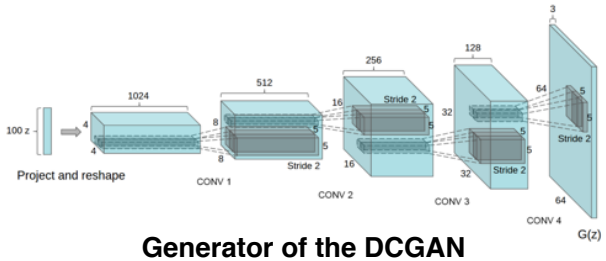
Least Squares Objective:

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

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

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein gan." *arXiv preprint arXiv:1701.07875* (2017).
Mao, Xudong, et al. "Least squares generative adversarial networks." *arXiv preprint ArXiv:1611.04076* (2016).

Example: DC-GAN



Generator of the DCGAN

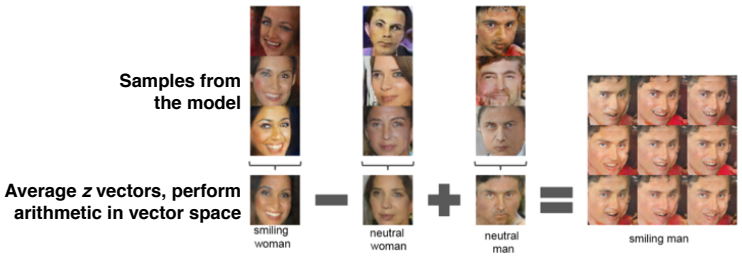
Radford et al, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

Example: DC-GAN

Generated results on LSUN Dataset

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

Example: DC-GAN

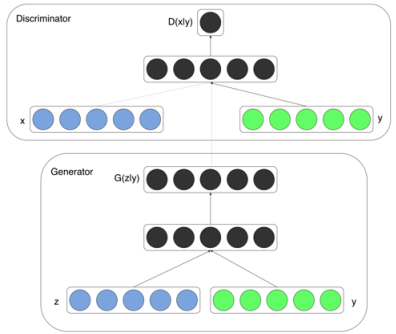


Interpretable Vector Math

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

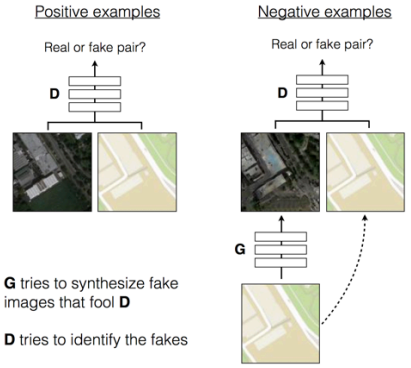
Example: Conditional GAN

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



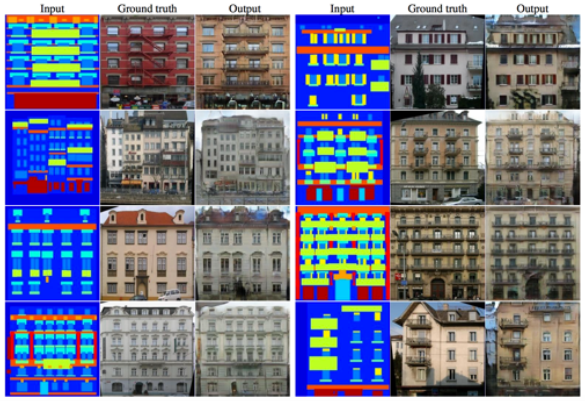
Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." 2014

Example: Pix2Pix



Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." 2016

Example: Pix2Pix

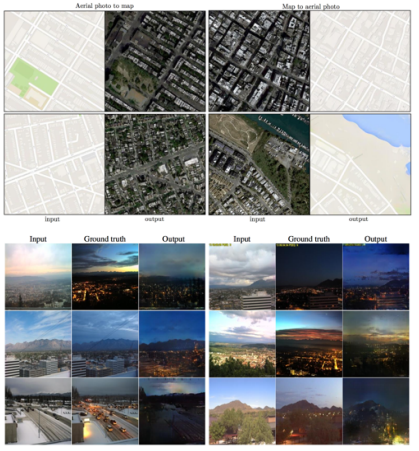


Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." 2016

Example: Pix2Pix

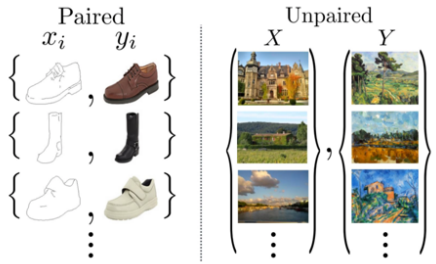
Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." 2016

Example: Pix2Pix



Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." 2016

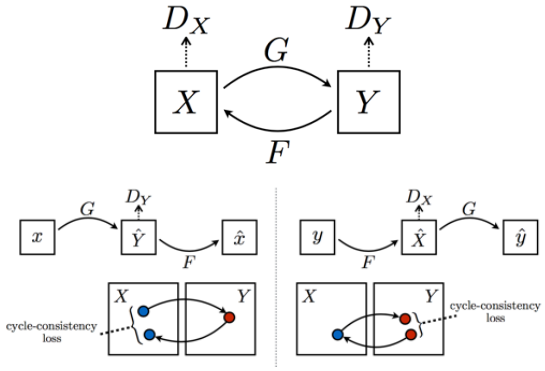
Example: CycleGAN



Pix2Pix needs paired data
However, in real world it is hard to find that many paired data

Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." 2017

Example: CycleGAN

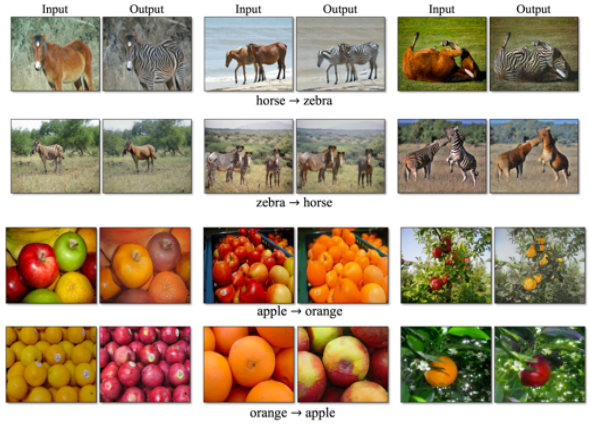


Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." 2017

Example: CycleGAN

Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." 2017

Example: CycleGAN



Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." 2017

Example: InfoGAN

Given a noise vector ***z*** and a latent code ***c***
How do we find G(*z*,*c*) ?

Chen, Xi, et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." NIPS2016

Example: InfoGAN

Given a noise vector ***z*** and a latent code ***c***
How do we find G(*z*,*c*) ?

Avoiding trivial solution $P_G(x \mid c) = P_G(x)$
Need high mutual information I(*c*; G(*z*,*c*))

Chen, Xi, et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." NIPS2016

Example: InfoGAN

I(X; Y): the amount of information learned from knowledge of Y about X, H denotes entropy:

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

I is the uncertainty in *X* when *Y* is observed

Chen, Xi, et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." NIPS2016

Example: InfoGAN

Information-regularized minimax objective:

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$

z: Noise vector
c: Latent code for salient structured semantic features
I(X; Y): the amount of information learned from knowledge of Y about X

Chen, Xi, et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." NIPS2016

Example: InfoGAN

Variational Lower bound L1(G,Q):

$$
\begin{aligned}
L_I(G,Q) &= E_{c\sim P(c),x\sim G(z,c)}[\log Q(c|x)] + H(c) \\
&= E_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) \\
&\leq I(c; G(z,c))
\end{aligned}
$$

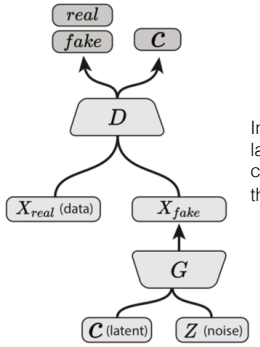
Final minimax game objective:

$$
\min_{G,Q}\max_{D} V_{\text{InfoGAN}}(D,G,Q) = V(D,G) - \lambda L_I(G,Q)
$$

Q: Auxiliary Distribution, parameterized as an NN

Chen, Xi, et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." NIPS2016

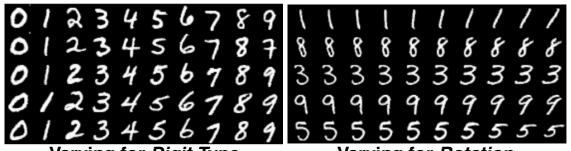
Example: InfoGAN



In Practice, ***Q*** shares all layers and one final fully connected layer to output the conditional distribution

Image credit: Odena, Augustus, Christopher Olah, and Jonathon Shlens. "Conditional image synthesis with auxiliary classifier gans." 2016
Chen, Xi, et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." NIPS2016

Example: InfoGAN

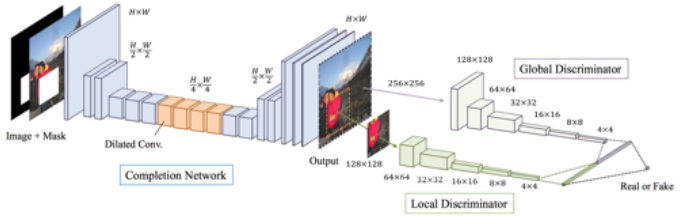


Varying for *Digit Type* **Varying for *Rotation***

Varying for *Rotation* **Varying for *Width***

Chen, Xi, et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." NIPS2016

Example: Image Completion



Iizuka, Satoshi et al. "Globally and locally consistent image completion." *ACM Transactions on Graphics (TOG)* 36.4 (2017): 107.

Example: Image Completion

Iizuka, Satoshi et al. "Globally and locally consistent image completion." *ACM Transactions on Graphics (TOG)* 36.4 (2017): 107.

The ***GAN Zoo***

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

How to train your **GAN**?

https://github.com/soumith/ganhacks

Image credit: http://www.atlanticfasteners.com/tech-tips/

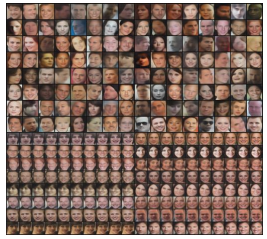
Break Time

See you in 10 mins!

Today's agenda

- Sponsor Talk: Neudesic
- Sample midterm
- Part 1: Generative Adversarial Networks
- Part 2: Variational Autoencoders
- Part 3: PixelRNN and PixelCNN

Variational Autoencoder

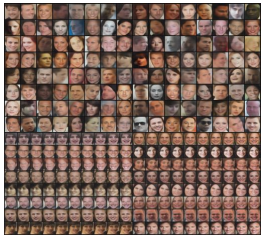


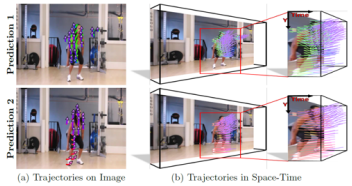
https://github.com/davidsandberg/facenet/wiki/Variational-autoencoder

Walker et. al. The Pose Knows: Video Forecasting by Generating Pose Futures. ICCV 2017

Walker et. al. An Uncertain Future: Forecasting from Static Images using Variational Autoencoders. ECCV 2016

Variational Autoencoder



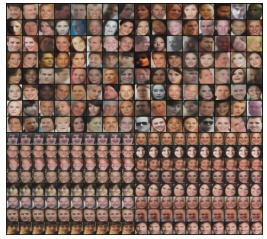


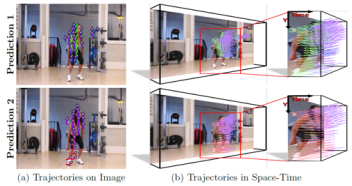
https://github.com/davidsandberg/facenet/wiki/Variational-autoencoder

Walker et. al. The Pose Knows: Video Forecasting by Generating Pose Futures. ICCV 2017

Walker et. al. An Uncertain Future: Forecasting from Static Images using Variational Autoencoders. ECCV 2016

Variational Autoencoder



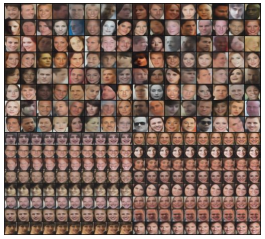


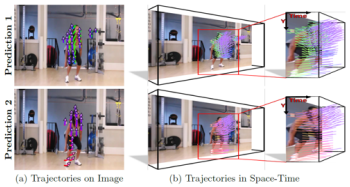
https://github.com/davidsandberg/facenet/wiki/Variational-autoencoder

Walker et. al. The Pose Knows: Video Forecasting by Generating Pose Futures. ICCV 2017

Walker et. al. An Uncertain Future: Forecasting from Static Images using Variational Autoencoders. ECCV 2016

Variational Autoencoder



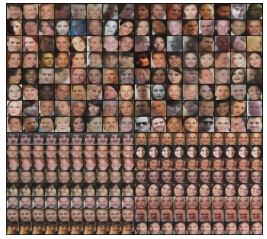


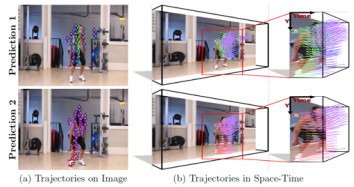
https://github.com/davidsandberg/facenet/wiki/Variational-autoencoder

Walker et. al. The Pose Knows: Video Forecasting by Generating Pose Futures. ICCV 2017

Walker et. al. An Uncertain Future: Forecasting from Static Images using Variational Autoencoders. ECCV 2016

Variational Autoencoder



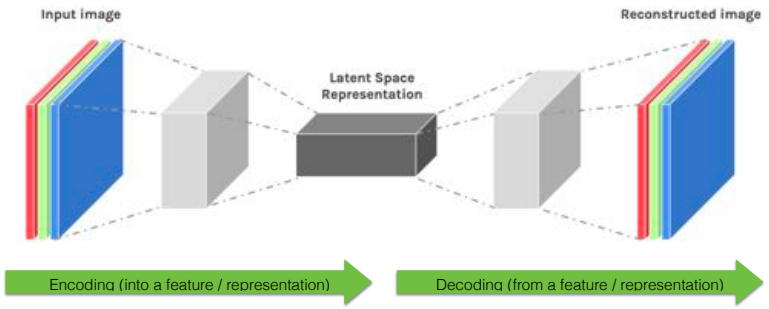


https://github.com/davidsandberg/facenet/wiki/Variational-autoencoder

Walker et. al. The Pose Knows: Video Forecasting by Generating Pose Futures. ICCV 2017

Walker et. al. An Uncertain Future: Forecasting from Static Images using Variational Autoencoders. ECCV 2016

Autoencoder



https://hackernoon.com/autoencoders-deep-learning-bits-1-11731e200694

Autoencoder

- **Unsupervised** approach
- Learn a lower-dimensional **feature representation** from unlabeled training data

Autoencoder

- Encoder
  - "**Encode**" the information

Autoencoder

- Encoder
  - "**Encode**" the information

Input Data x

Autoencoder

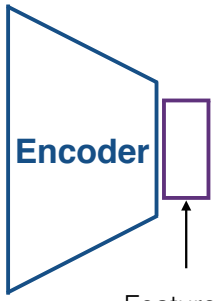
- Encoder
  - "**Encode**" the information

Input Data x

Autoencoder

- Encoder
  - "**Encode**" the information

Input Data x



Features $\mathbf{z}$

Autoencoder

- Encoder
  - Architecture: Multilayer Perceptrons

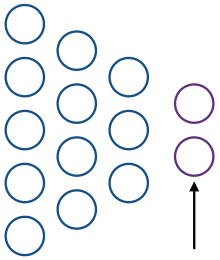
Input Data x

Features $\mathbf{z}$

Autoencoder

- Encoder
  - Architecture: Multilayer Perceptrons

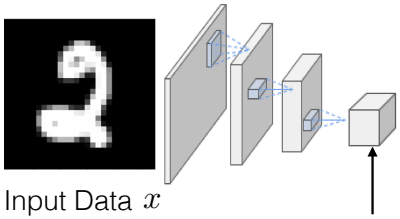
Input Data x



Features $\mathbf{z}$

Autoencoder

- Encoder
  - Architecture: Convolutional layers

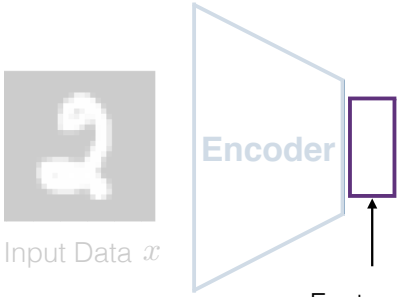


Input Data x

Features $\mathbf{z}$

Autoencoder

- Decoder
  - "**Decode**" the codes

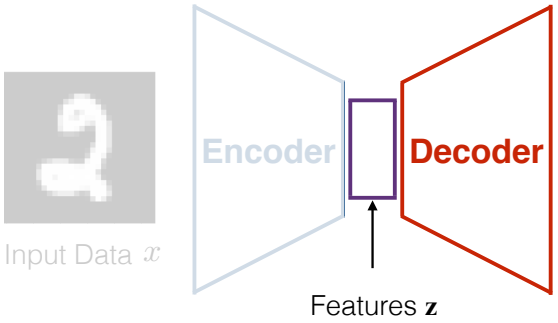


Input Data x

Features $\mathbf{z}$

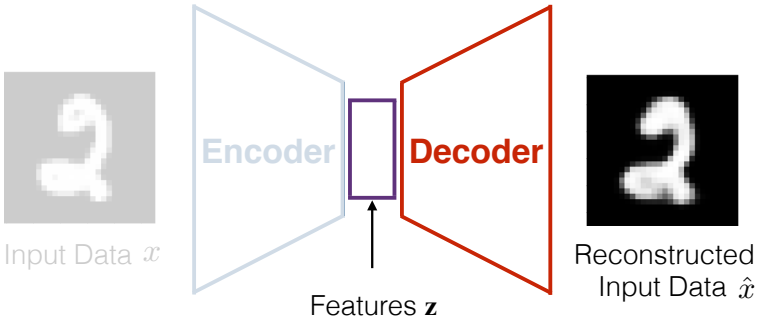
Autoencoder

- Decoder
  - "**Decode**" the codes



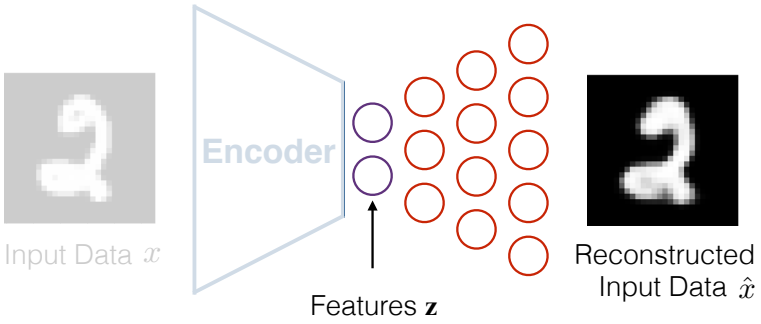
Autoencoder

- Decoder
  - "**Decode**" the codes



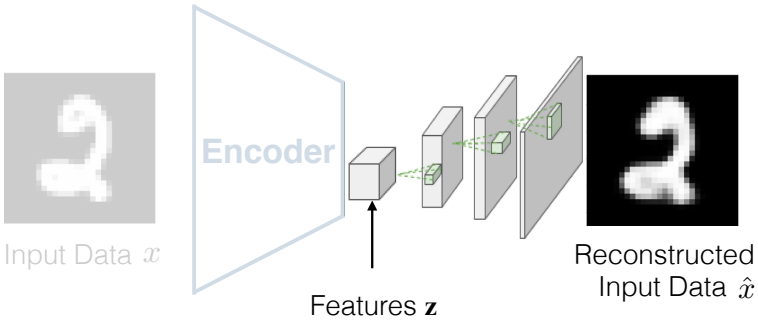
Autoencoder

- Decoder
  - Architecture: Multilayer Perceptrons



Autoencoder

- Decoder
  - Architecture: Deconvolutional layers



Autoencoder

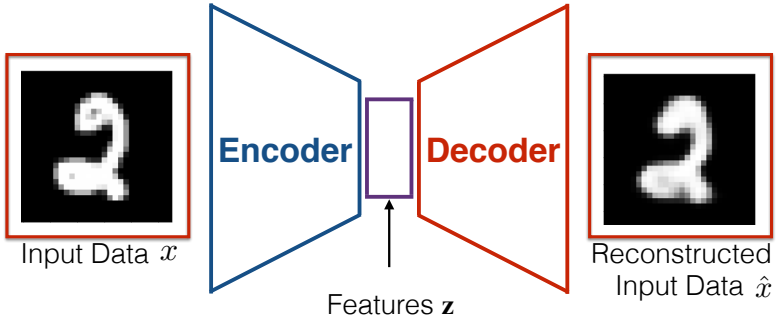
How do we learn the parameters?

- Encoder(x; θ_d)
- Decoder(z; θ_e)

Autoencoder

Jointly train the encoder and the decoder

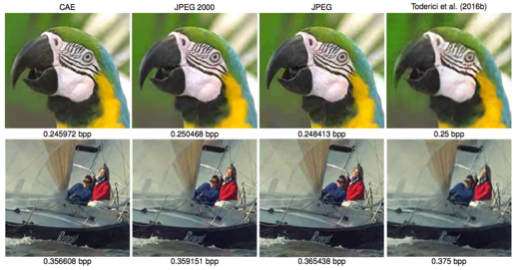
- Reconstruction loss: $distance(x, \hat{x})$



Autoencoder

Applications

- Compression

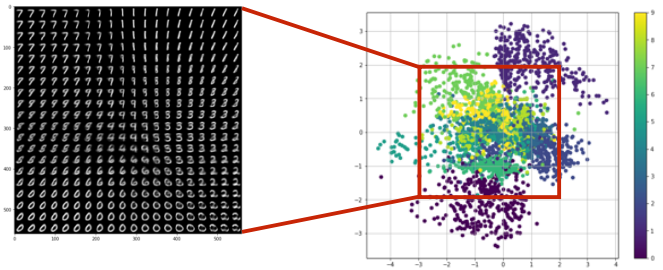


Theis et. al., Lossy Image Compression with Compressive Autoencoders, ICLR 2017

Autoencoder

Applications

- Capture meaningful feature representations

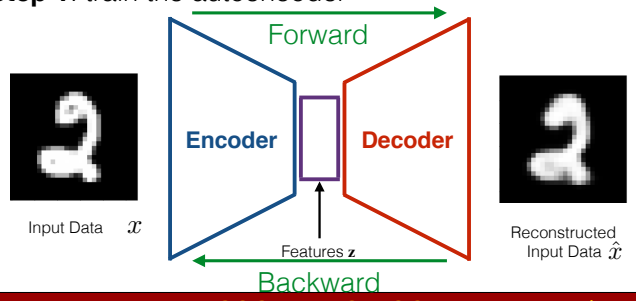


Autoencoder

Applications

- Capture meaningful feature representations

Step 1: train the autoencoder

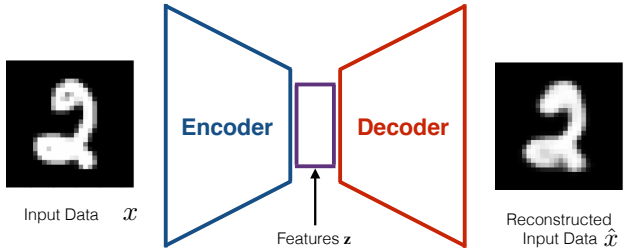


Autoencoder

Applications

- Capture meaningful feature representations

Step 2: throw away the decoder

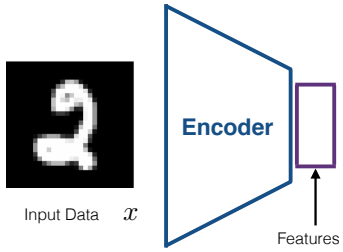


Autoencoder

Applications

- Capture meaningful feature representations

Step 2: throw away the decoder

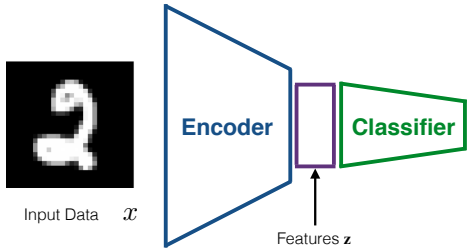


Autoencoder

Applications

- Capture meaningful feature representations

Step 3: add additional layers

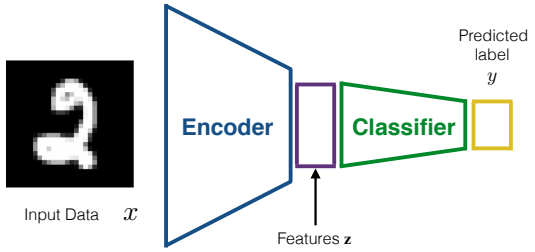


Autoencoder

Applications

- Capture meaningful feature representations

Step 3: add additional layers

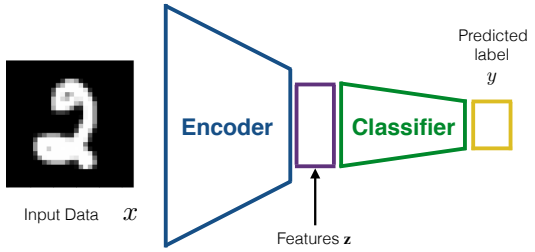


Autoencoder

Applications

- Capture meaningful feature representations

Step 4: jointly train the classifier and fine-tune the encoder

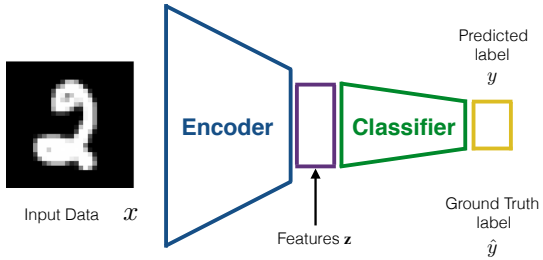


Autoencoder

Applications

- Capture meaningful feature representations

Step 4: jointly train the classifier and fine-tune the encoder

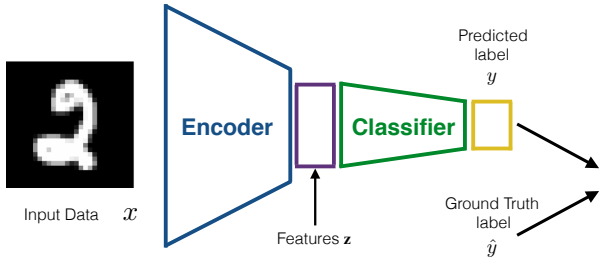


Autoencoder

Applications

- Capture meaningful feature representations

Step 4: jointly train the classifier and fine-tune the encoder

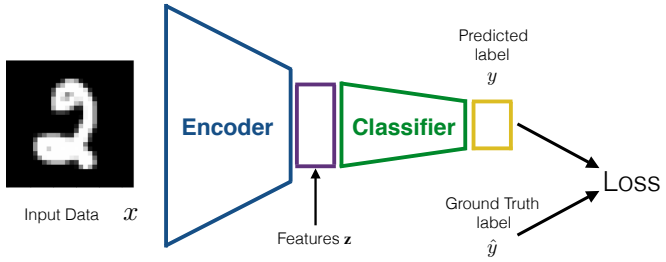


Autoencoder

Applications

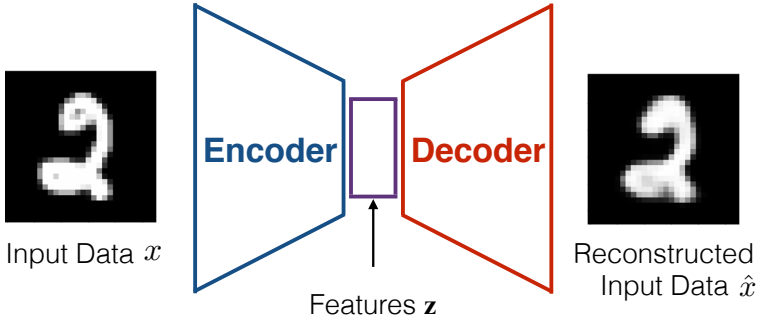
- Capture meaningful feature representations

Step 4: jointly train the classifier and fine-tune the encoder



Autoencoder

Looks pretty good! Doesn't it?



Autoencoder

What about generation?

Autoencoder

What about generation?

Images in a dataset

Autoencoder

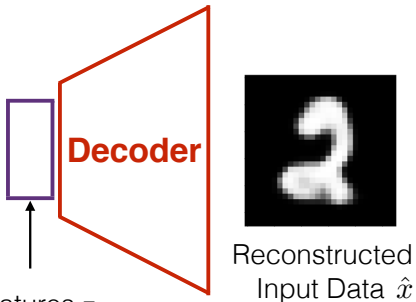
What about generation?

Images in a dataset

Generated images

Autoencoder

What about generation?



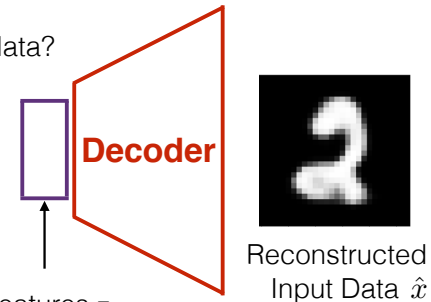
Features $\mathbf{z}$

Reconstructed Input Data $\hat{x}$

Autoencoder

What about generation?

- Given feature vectors
- Can we generate data?



Features $\mathbf{z}$

Reconstructed Input Data $\hat{x}$

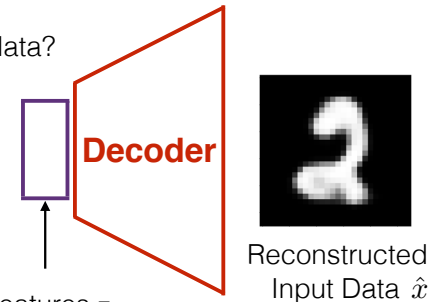
Autoencoder

What about generation?

- Given feature vectors
- Can we generate data?

No!

- What are right feature vectors?



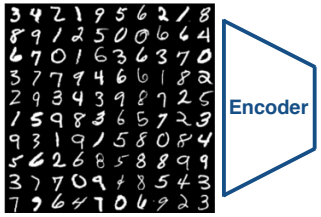
Features $\mathbf{z}$

Reconstructed Input Data $\hat{x}$

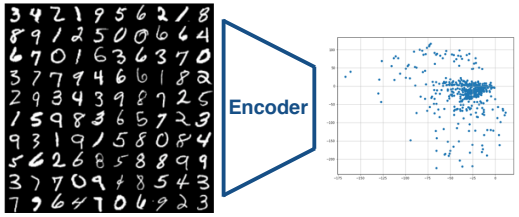
Autoencoder

Autoencoder

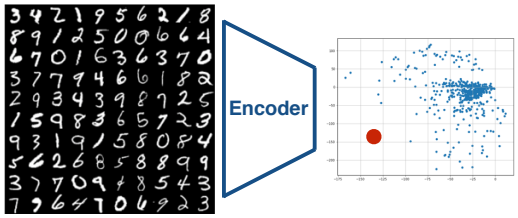
Autoencoder



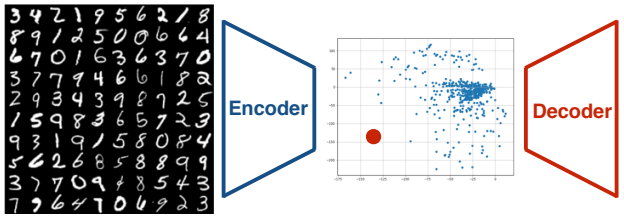
Autoencoder



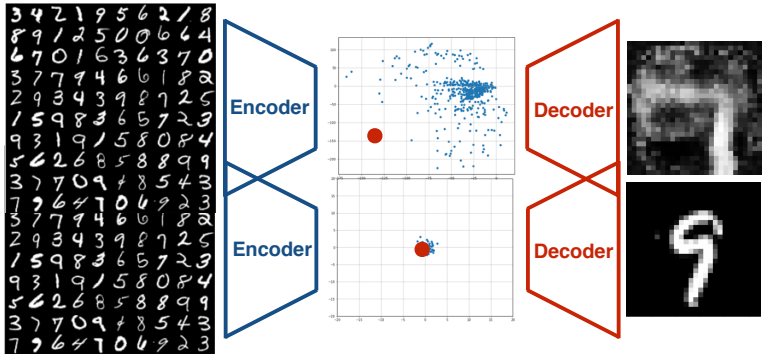
Autoencoder



Autoencoder

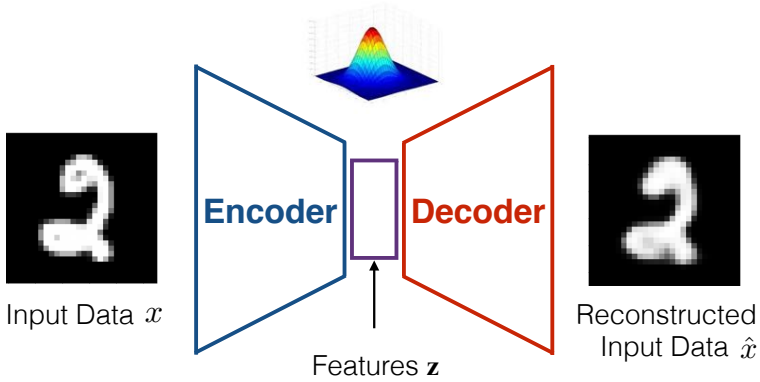


Autoencoder



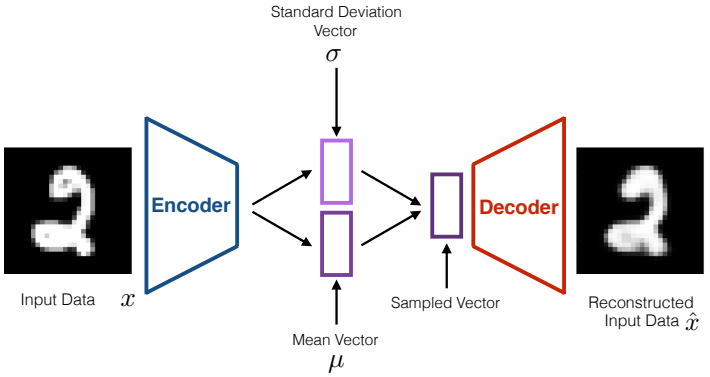
Variational Autoencoder

The trick: constrain the latent distribution



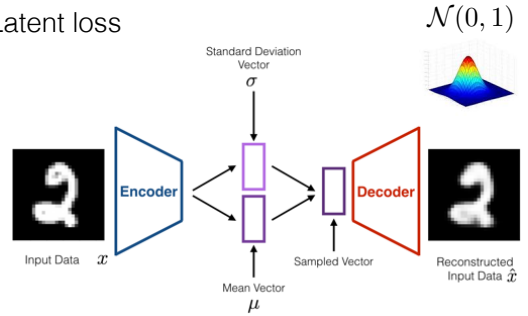
Variational Autoencoder

- Constrain the latent distribution: a Gaussian distribution



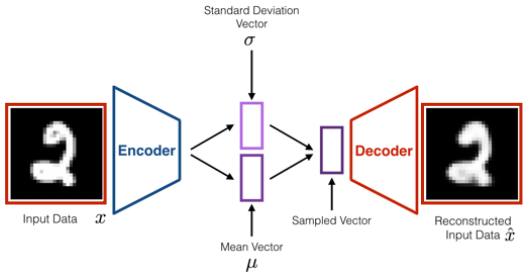
Variational Autoencoder

- Loss
  - Reconstruction loss
  - Latent loss



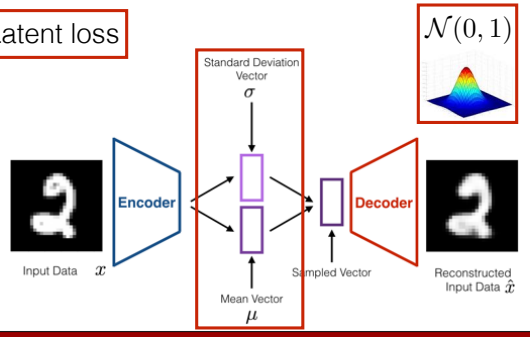
Variational Autoencoder

- Loss
  - Reconstruction loss
  - Latent loss



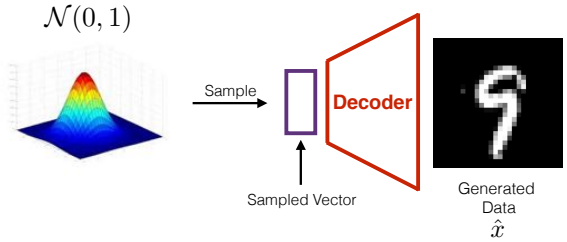
Variational Autoencoder

- Loss
  - Reconstruction loss
  - Latent loss



Variational Autoencoder

- Generation



Autoencoder

Results: Generated from a VAE trained with IAF

Images in the dataset

Generated images

Kingma et. al., Improving Variational Inference with Inverse Autoregressive Flow, NIPS 2016

Variational Autoencoder

Seriously, how does a VAE work?

Following couples of slides: Kingma and Welling, Auto-Encoding Variational Bayes, ICLR 2014

Variational Autoencoder

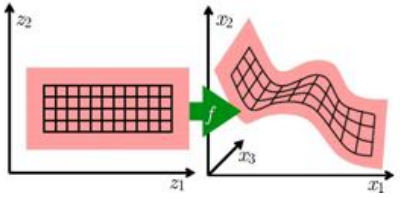
Let's first talk about

how **real data** are generated…

Variational Autoencoder

A dataset: $X = \{x^{(i)}\}_{i=1}^{N}$ is

generated from an unobserved random variable z



$p_\theta(z)$: prior distribution

Variational Autoencoder

A dataset: $X = \{x^{(i)}\}_{i=1}^{N}$ is

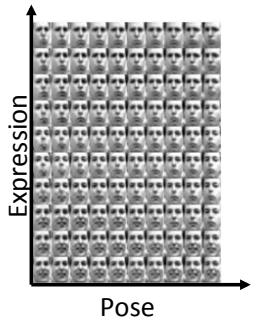
generated from an unobserved random variable z

Example:

A datapoint $\mathbf{x_i}$: 

The latent variable $\mathbf{z_i}$:

{pose: right, expression: sour look}



Variational Autoencoder

The generation process

Step 1: sample from the prior distribution $p_\theta(z)$ to get z

Step 2: generate x

from a conditional distribution $p_\theta(x|z)$

$$p_\theta(z) \rightarrow z \qquad p_\theta(x|z) \rightarrow x$$

{pose: right,
expression: sour look} 

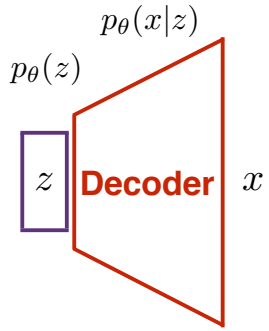
Variational Autoencoder

Explain **observed variables** x

in terms of **latent variables** z

Variational Autoencoder

So, what if we want to generate more data?

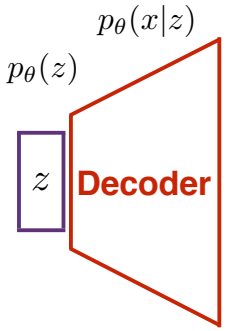


Variational Autoencoder

So, what if we want to generate more data?

How do we **represent** this model?

Choose **prior** as a known distribution, e.g. Gaussian



Represent **likelihood** as a neural network

Variational Autoencoder

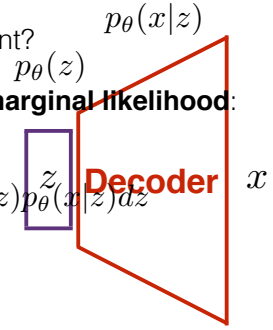
So, what if we want to generate more data?

How do we **train** this model?

What do we want?

Maximize **the marginal likelihood**:

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



Variational Autoencoder

The marginal likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$

Intractable: enumerate every z

Intractable: posterior

The intractability: $p_\theta(z|x) = \frac{p_\theta(x|z) p_\theta(z)}{p_\theta(x)}$

The recognition network: $q_\phi(z|x)$

an approximation of the true posterior

Variational Autoencoder

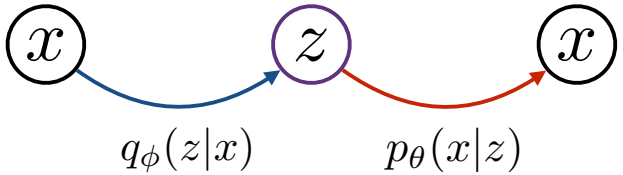
Autoencoder

We learn **deterministic** mappings!

- An encoder f
- A decoder g

Variational Autoencoder

Variational Autoencoder



We learn **probabilistic** mappings!

- A probabilistic encoder $q_\phi(z|x)$
- A probabilistic decoder $p_\theta(x|z)$

Variational Autoencoder

Now we have an encoder and a decoder…

- A probabilistic encoder $q_\phi(z|x)$
- A probabilistic decoder $p_\theta(x|z)$

Variational Autoencoder

Assumptions

- We **don't know** θ, $\varnothing$
- We **don't know** $p_\theta(z|x)$
- Given θ, $\varnothing$, we **know** the **distributions**

$$p_\theta(z), p_\theta(x|z), q_\phi(z|x)$$

- Each latent variable z_i is generated from $p_\theta(z)$
- Each data point x_i is generated from $p_\theta(x|z_i)$

Variational Autoencoder

Now we can work on the marginal likelihood:

Variational Autoencoder

Now we can work on the marginal likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Variational Autoencoder

Now we can work on the marginal likelihood:

$$\begin{aligned}\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] && (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\ &= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] && \text{(Bayes' Rule)}\end{aligned}$$

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Variational Autoencoder

Now we can work on the marginal likelihood:

$$\begin{aligned}\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] && (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\ &= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] && \text{(Bayes' Rule)}\\ &= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] && \text{(Multiply by constant)}\end{aligned}$$

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Variational Autoencoder

Now we can work on the marginal likelihood:

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] && (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] && \text{(Bayes' Rule)}\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] && \text{(Multiply by constant)}\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] && \text{(Logarithms)}
\end{aligned}$$

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Variational Autoencoder

Now we can work on the marginal likelihood:

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] && (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] && \text{(Bayes' Rule)}\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] && \text{(Multiply by constant)}\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] && \text{(Logarithms)}
\end{aligned}$$

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Joseph J. Lim CSCI 599 @ USC Lecture 7

Variational Autoencoder

Now we can work on the marginal likelihood:

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] && (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] && \text{(Bayes' Rule)}\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] && \text{(Multiply by constant)}\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] && \text{(Logarithms)}
\end{aligned}$$

KL divergence: $D_{\mathrm{KL}}(P\|Q) = \sum_i P(i)\,\log\frac{P(i)}{Q(i)}$

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Joseph J. Lim CSCI 599 @ USC Lecture 7

Variational Autoencoder

Now we can work on the marginal likelihood:

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] && (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] && \text{(Bayes' Rule)}\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] && \text{(Multiply by constant)}\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] && \text{(Logarithms)}\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,\|\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,\|\, p_\theta(z \mid x^{(i)}))
\end{aligned}$$

KL divergence: $D_{\mathrm{KL}}(P\|Q) = \sum_i P(i)\,\log\frac{P(i)}{Q(i)}$

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Joseph J. Lim CSCI 599 @ USC Lecture 7

Variational Autoencoder

Now we can work on the marginal likelihood:

$$\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

- Given the parameters of the decoder, we can compute this through sampling
- Given the parameters of the encoder and the Gaussian assumption we made, we can compute this
- We cannot compute this term since is $p_\theta(z|x^{(i)})$ intractable. At least, we know this term is greater than or equal to zeros. (the property of the KL divergence)

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Variational Autoencoder

Now we can work on the marginal likelihood:

$$\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

$$\mathcal{L}(x^{(i)}, \theta, \phi)$$

- Tractable ELBO (evidence lower bound)
- We can optimize it using gradient descent (both the decoder and the KL term are differentiable)
- The goal of the training: maximize ELBO to maximize the marginal likelihood

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Variational Autoencoder

Now we can work on the marginal likelihood:

ELBO

$$\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))$$

- Reconstruct the input data
- Bring the recognition network closer to the prior

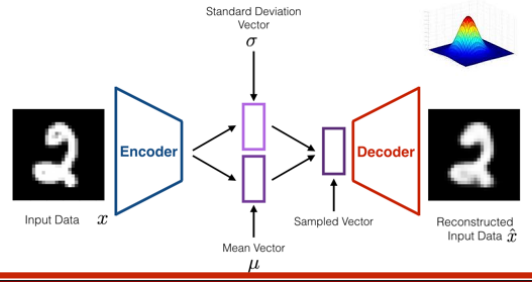
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Variational Autoencoder

- Loss
  - Reconstruction loss
  - Latent loss

Remember This?

$\mathcal{N}(0,1)$



Variational Autoencoder

Now we can work on the marginal likelihood:

ELBO

$$\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))$$

Reconstruction loss Latent loss

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

Variational Autoencoder

Pros

- There is a clear and recognized way to evaluate the quality of the model (log-likelihood)
- Trained recognition networks can learn to produce useful feature

Cons

- Tend to generate blurry results (compared to GANs)

Break Time

See you in 10 mins!

Today's agenda

- Sponsor Talk: Neudesic
- Sample midterm
- Part 1: Generative Adversarial Networks
- Part 2: Variational Autoencoders
- Part 3: PixelRNN and PixelCNN

The Marginal Likelihood

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

The Marginal Likelihood

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

- **Variational Autoencoders** look pretty promising!

The Marginal Likelihood

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

- **Variational Autoencoders** look pretty promising!
- But… we do not directly maximize the marginal likelihood

The Marginal Likelihood

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

- **Variational Autoencoders** look pretty promising!
- But… we do not directly maximize the marginal likelihood
- Instead, we maximize the variational lower bound (ELBO)

The Marginal Likelihood

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

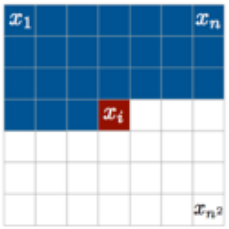
-
-
-

(ELBO)

Why don't we directly maximize the marginal likelihood?

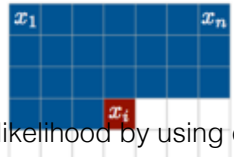
Decompose An Image

Imagine the process of generating images as a **sequential** process.



Following couples of slides: van den Oord et. al., Pixel Recurrent Neural Networks, ICML 2016

Decompose An Image



Decompose the likelihood by using chain rule

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

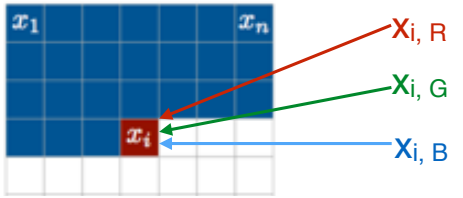
The product of **conditional distributions** over the pixels

Decompose An Image

How do we deal with RGB values?

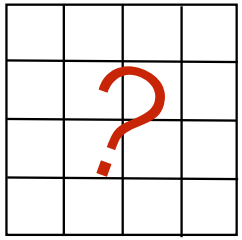
Every value is conditioned on the **other channels** as well as on all the **previously generated pixels**

$$p(x_{i,R}|\mathbf{x}_{<i}) p(x_{i,G}|\mathbf{x}_{<i}, x_{i,R}) p(x_{i,B}|\mathbf{x}_{<i}, x_{i,R}, x_{i,G})$$



PixelRNN

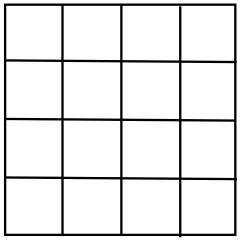
How do we decide the order of pixels?



PixelRNN

How do we decide the order of pixels?

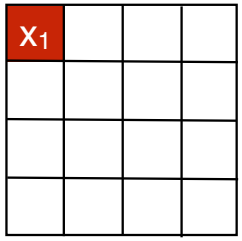
Start from the top-left corner



PixelRNN

How do we decide the order of pixels?

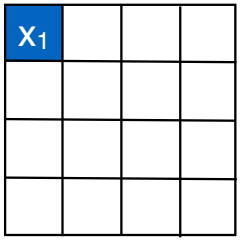
Start from the top-left corner



PixelRNN

How do we decide the order of pixels?

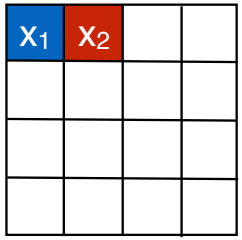
Start from the top-left corner



PixelRNN

How do we decide the order of pixels?

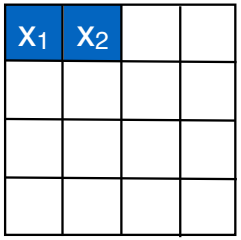
Start from the top-left corner



PixelRNN

How do we decide the order of pixels?

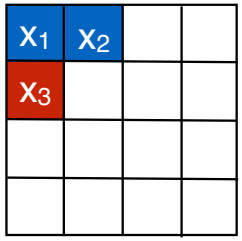
Start from the top-left corner



PixelRNN

How do we decide the order of pixels?

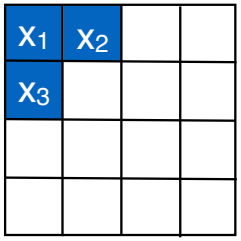
Start from the top-left corner



PixelRNN

How do we decide the order of pixels?

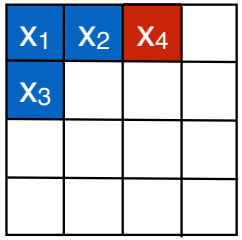
Start from the top-left corner



PixelRNN

How do we decide the order of pixels?

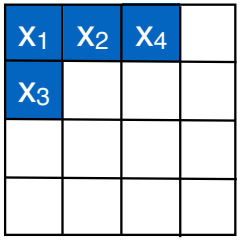
Start from the top-left corner



PixelRNN

How do we decide the order of pixels?

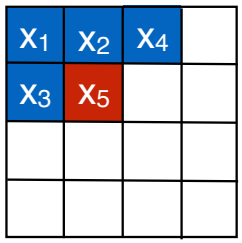
Start from the top-left corner



PixelRNN

How do we decide the order of pixels?

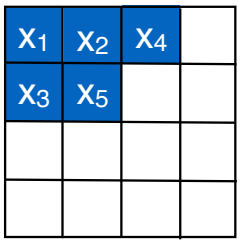
Start from the top-left corner



PixelRNN

How do we decide the order of pixels?

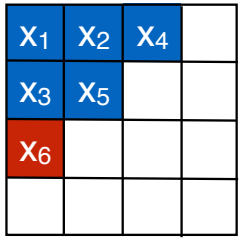
Start from the top-left corner



PixelRNN

How do we decide the order of pixels?

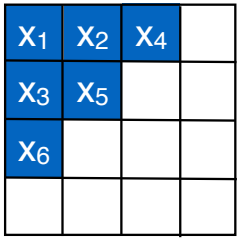
Start from the top-left corner



PixelRNN

How do we decide the order of pixels?

Start from the top-left corner

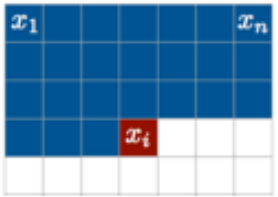


PixelRNN

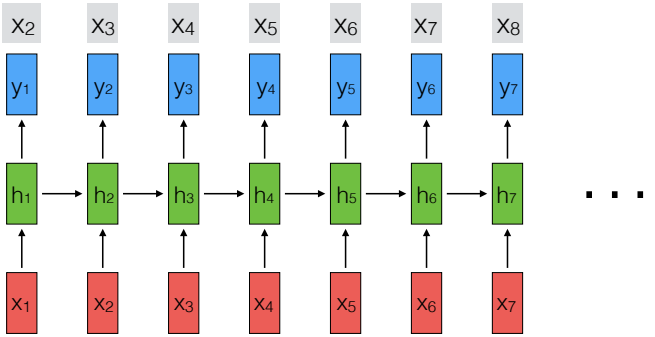
How do we model the pixel value **dependency**?

Remember the architectures we just learned?

RNN (LSTM) !



PixelRNN

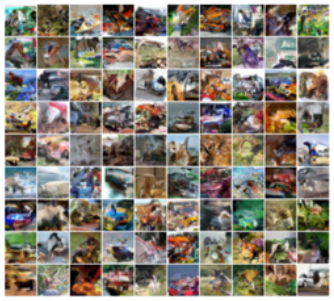


PixelRNN

Results: Image completion

PixelRNN

Results: Image generation



Cifar10

ImageNet (32x32)

PixelRNN

Pros

- Explicitly maximize the likelihood
- There is a clear and recognized way to evaluate the quality of the model (log-likelihood)
- Generate sharper images

Cons

- The sequential generation is slow

PixelRNN

Pros

- Explicitly maximize the likelihood
- There is a clear and recognized way to evaluate the quality of the model (log-likelihood)
- Generate sharper images

Cons

- The sequential generation is slow

PixelCNN [van den Oord et. al. ICML 2016]

Summary

Next week

- Deep Reinforcement Learning

Todo

- Midterm next week (2/26)
- Proposal due in 2 weeks (3/5)
- Projects! :)

Questions?