CRIPAC 智能感知与计算研究中心 Center for Research on Intelligent Perception and Computing

中国科学院自动化研究所 Institute of Automation Chinese Academy of Sciences

2022

"Deep Learning Lecture"

Lecture 8：Deep Generative Model (2)

Yan Huang

Center for Research on Intelligent Perception and Computing (CRIPAC)

National Laboratory of Pattern Recognition (NLPR)

Institute of Automation, Chinese Academy of Science (CASIA)

Outline

1. Course Review
2. Generative Adversarial Networks
3. Variational Autoencoder
4. Other Generation Methods

Review: Generative Model

- We want to build a probabilistic model of the input $P(\mathbf{x})$
- Like before, we are interested in latent factors $\mathbf{h}$ that explain $\boldsymbol{x}$
- We then care about the marginal:

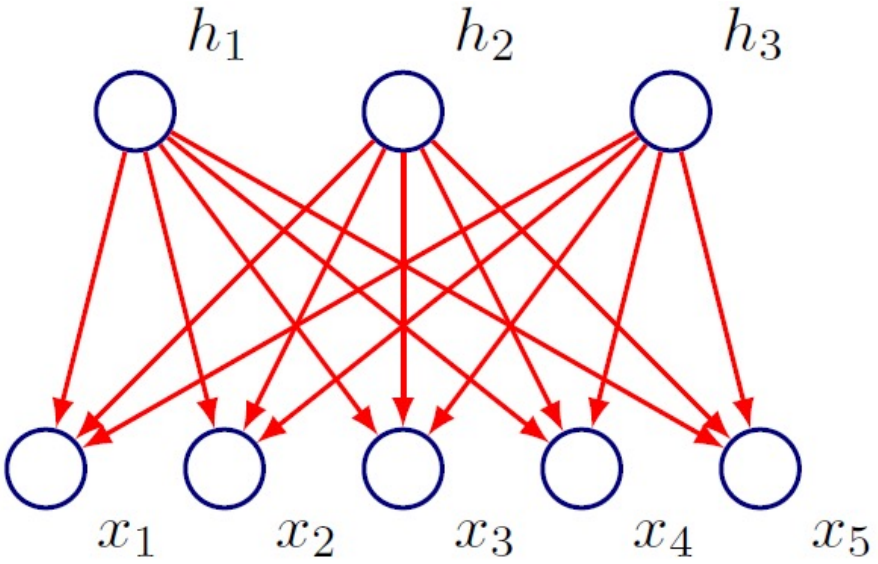
$$P(\mathbf{x}) = E_{\mathbf{h}} P(\mathbf{x}|\mathbf{h})$$

- The latent factor $\mathbf{h}$ is an encoding of the data

Unsupervised Learning ⟷ Probabilistic Modeling

Review: Linear Factor Model

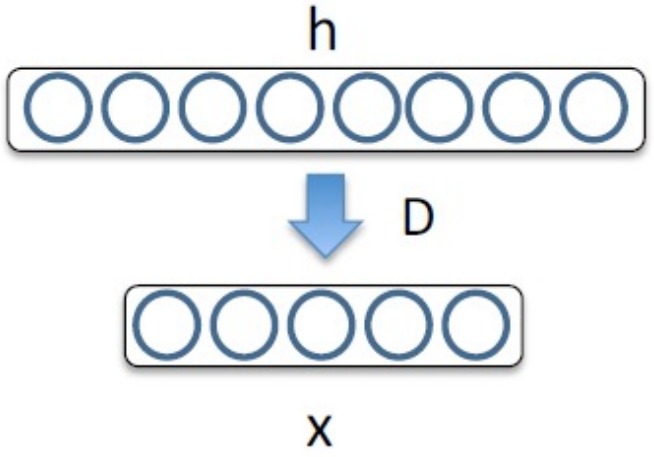
- Simplest decoding model: Get $\mathbf{x}$ after a linear transformation of $\mathbf{h}$ with some noise
- Formally: Suppose we sample the latent factors from a distribution $\mathbf{h} \sim P(\mathbf{h})$
- Then: $\mathbf{x} = W\mathbf{h} + \mathbf{b} + \boldsymbol{\varepsilon}$



$$\mathbf{x} = W\mathbf{h} + \mathbf{b} + \epsilon$$

Review: Sparse Coding

- Sparse coding (Olshausen & Field, 1996). Originally developed to explain early visual processing in the brain (edge detection)
- For each input $\mathbf{x}^{(t)}$ find a latent representation $\mathbf{h}^{(t)}$ such that:
  - it is sparse: the vector $\mathbf{h}^{(t)}$ has many zeros
  - we can reconstruct the original input $\mathbf{x}^{(t)}$



Review: Sparse Coding

- For each $\mathbf{x}^{(t)}$ find a latent representation $\mathbf{h}^{(t)}$ such that:
  - it is sparse: the vector $\mathbf{h}^{(t)}$ has many zeros
  - we can reconstruct the original input $\mathbf{x}^{(t)}$

- In other words:

Reconstruction: $\widehat{\mathbf{x}}^{(t)}$

Sparsity vs. reconstruction control

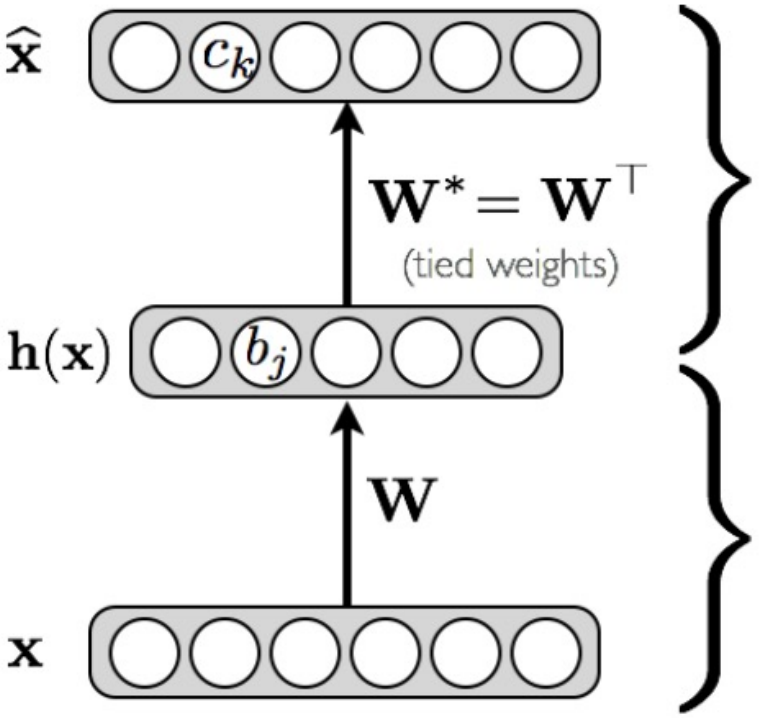
$$\min_{\mathbf{D}} \frac{1}{T} \sum_{t=1}^{T} \min_{\mathbf{h}^{(t)}} \frac{1}{2} ||\mathbf{x}^{(t)} - \mathbf{D}\,\mathbf{h}^{(t)}||_2^2 + \lambda ||\mathbf{h}^{(t)}||_1$$

Reconstruction error

Sparsity penalty

Review: Autoencoder

- Feed-forward neural network trained to reproduce its input at the output layer



Decoder

$$\hat{\mathbf{x}} = o(\hat{\mathbf{a}}(\mathbf{x})) = \underbrace{\text{sigm}(\mathbf{c} + \mathbf{W}^*\mathbf{h}(\mathbf{x}))}_{\text{For binary units}}$$

Encoder

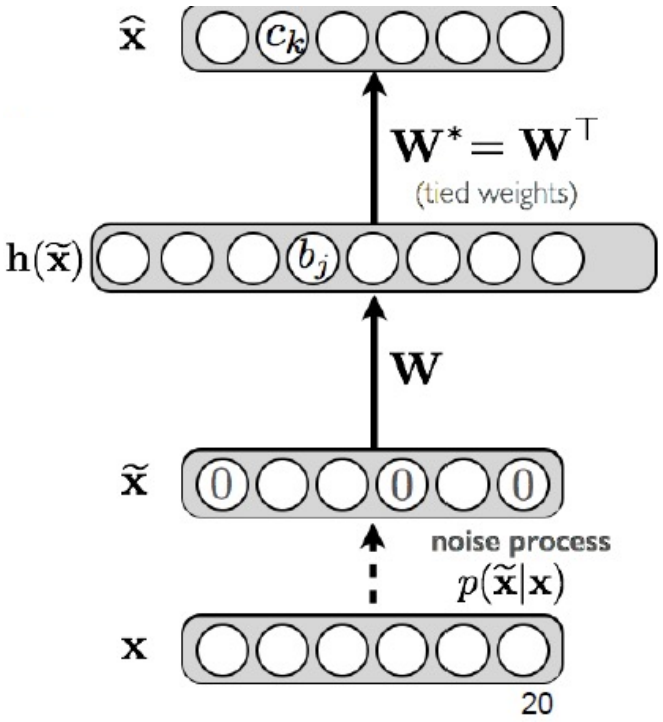
$$\mathbf{h}(\mathbf{x}) = g(\mathbf{a}(\mathbf{x})) = \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})$$

Review: Denoising Autoencoder

- Idea: representation should be robust to introduction of noise:
  - Random assignment of subset of inputs to 0, with probability
  - Similar to dropouts on the input layer
  - Gaussian additive noise

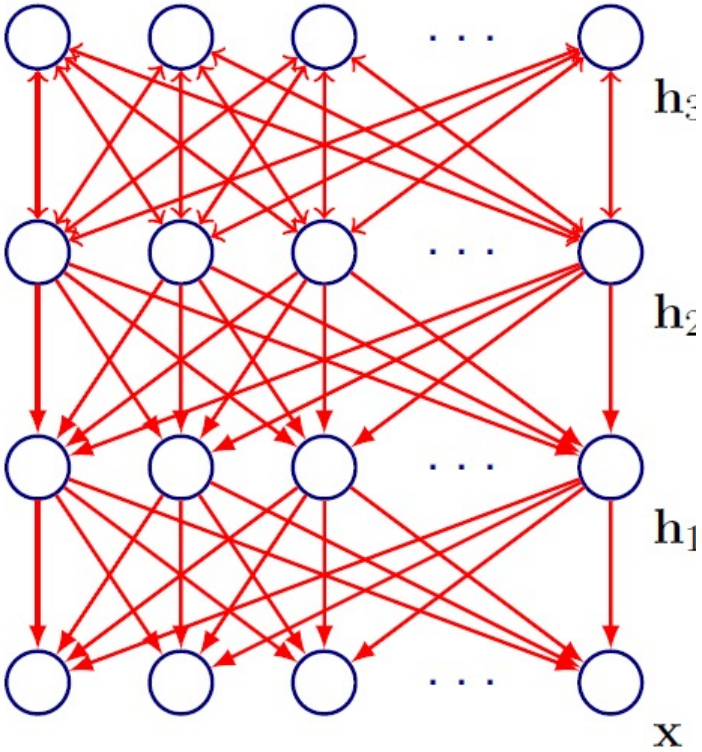
- Reconstruction $\widehat{\mathbf{x}}$ computed from the corrupted input $\widetilde{\mathbf{x}}$

- Loss function compares $\widehat{\mathbf{x}}$ reconstruction with the noiseless input $\mathbf{x}$



(Vincent et.al., ICML 2008)

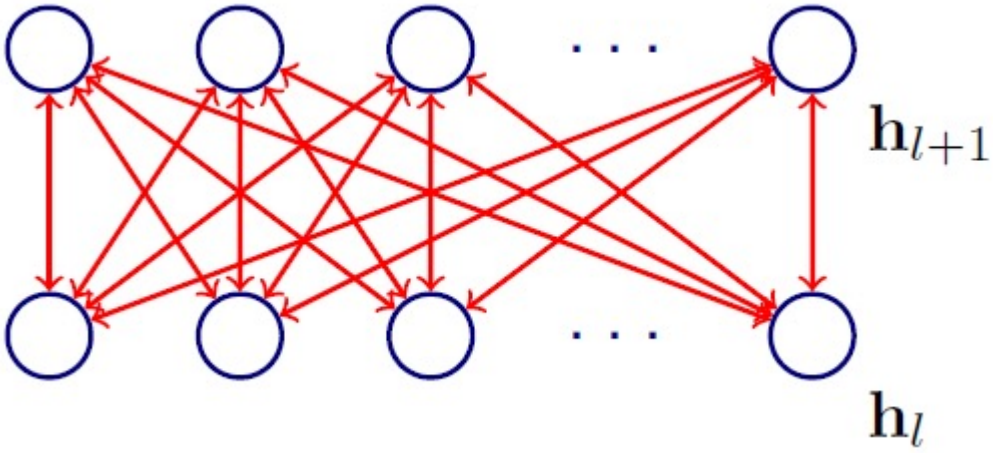
Review: Deep Belief Networks



- The top two layers have undirected edges
- The joint probability:

$$P(\mathbf{x}, \mathbf{h}^1, \ldots, \mathbf{h}^l) = P(\mathbf{h}^l, \mathbf{h}^{l-1}) \Big(\prod_{k=1}^{l-2} P(\mathbf{h}^k | \mathbf{h}^{k+1}) \Big) P(\mathbf{x} | \mathbf{h}^1)$$

Review: Restricted Boltzmann Machines



- As seen before, the Free Energy can be computed efficiently:

$$\text{FreeEnergy}(\mathbf{x}) = -\mathbf{b}^T\mathbf{x} - \sum_i \log \sum_{\mathbf{h}_i} \exp^{\mathbf{h}_i(\mathbf{c}_i + W_i\mathbf{x})}$$

- The conditional probability:

$$P(\mathbf{h}|\mathbf{x}) = \frac{\exp\left(\mathbf{b}^T\mathbf{x} + \mathbf{c}^T\mathbf{h} + \mathbf{h}^T W\mathbf{x}\right)}{\sum_{\tilde{\mathbf{h}}} \exp\left(\mathbf{b}^T\mathbf{x} + \mathbf{c}^T\tilde{\mathbf{h}} + \tilde{\mathbf{h}}^T W\mathbf{x}\right)} = \prod_i P(\mathbf{h}_i|\mathbf{x})$$

Outline

1. Course Review
2. Generative Adversarial Networks
3. Variational Autoencoder
4. Other Generation Methods

Why Generative Models?

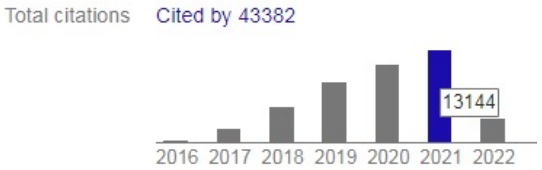
- **Discriminative models**
  - Given an image **X**, predict a label **Y**
  - Estimates **P(Y|X)**

- **Discriminative models have several key limitations**
  - **Can't model P(X)**, i.e. the probability of seeing a certain image
  - Can't sample from **P(X)**, i.e. **can't generate new images**

- **Generative models (in general) cope with all of above**
  - Can model **P(X)**
  - Can generate new images

Generative Adversarial Networks

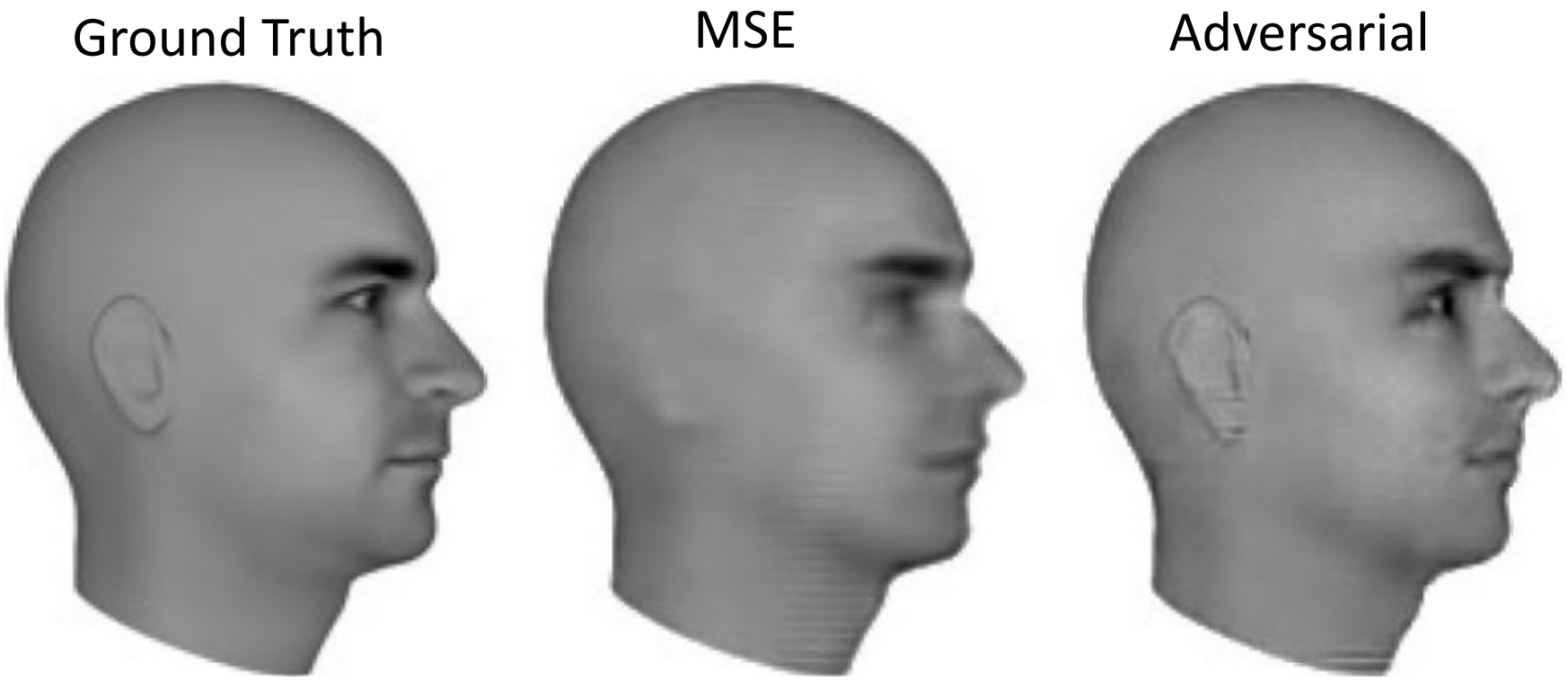
- **Generative**
  - Learn a generative model
- **Adversarial**
  - Trained in an adversarial setting
- **Networks**
  - Use Deep Neural Networks



Why Use GANs for Generation?

- Can be trained using back-propagation for neural network based Generator/Discriminator functions
- Sharper images can be generated
- Faster to sample from the model distribution: *single* forward pass generates a *single* sample

Magic of GANs



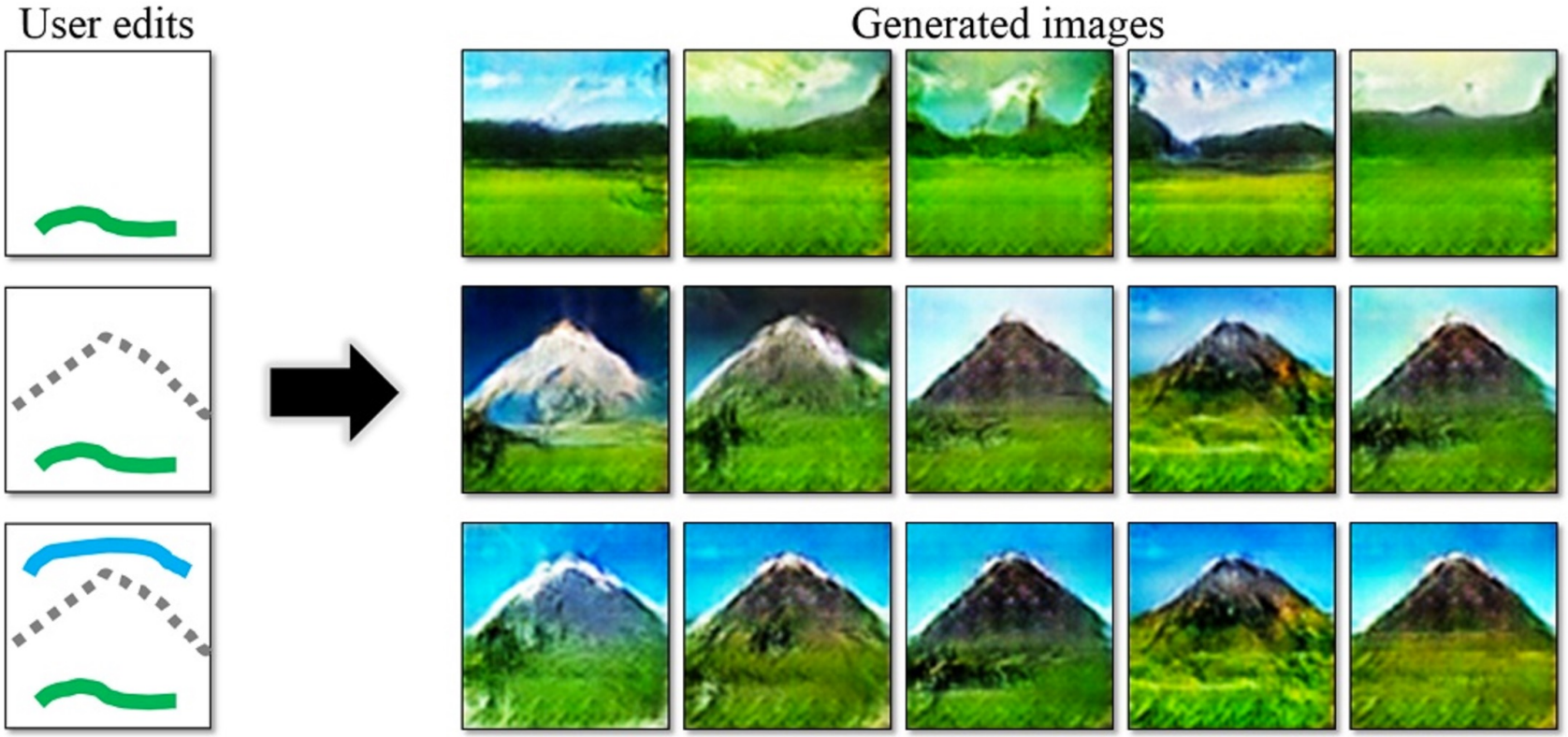
Lotter, William, Gabriel Kreiman, and David Cox. "Unsupervised learning of visual structure using predictive generative networks." *arXiv preprint arXiv:1511.06380* (2015).

Magic of GANs

Which one is generated by computer?

Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." *arXiv preprint arXiv:1609.04802* (2016).

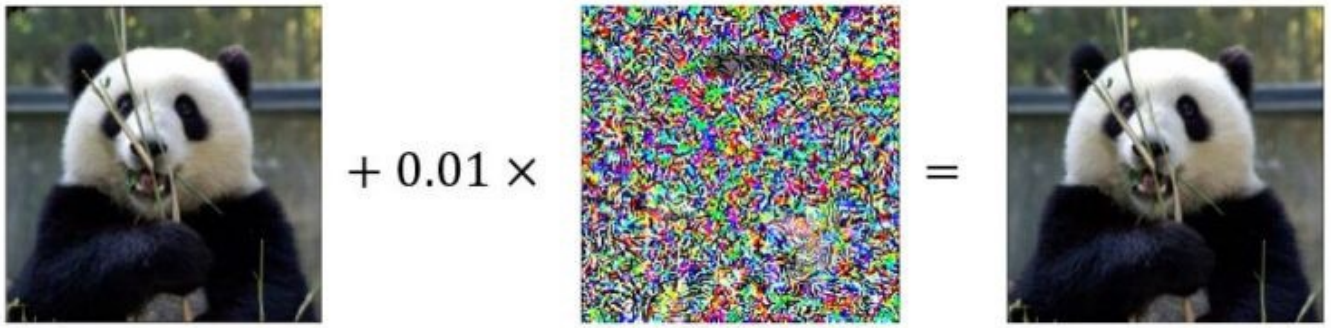
Magic of GANs



http://people.eecs.berkeley.edu/~junyanz/projects/gvm/

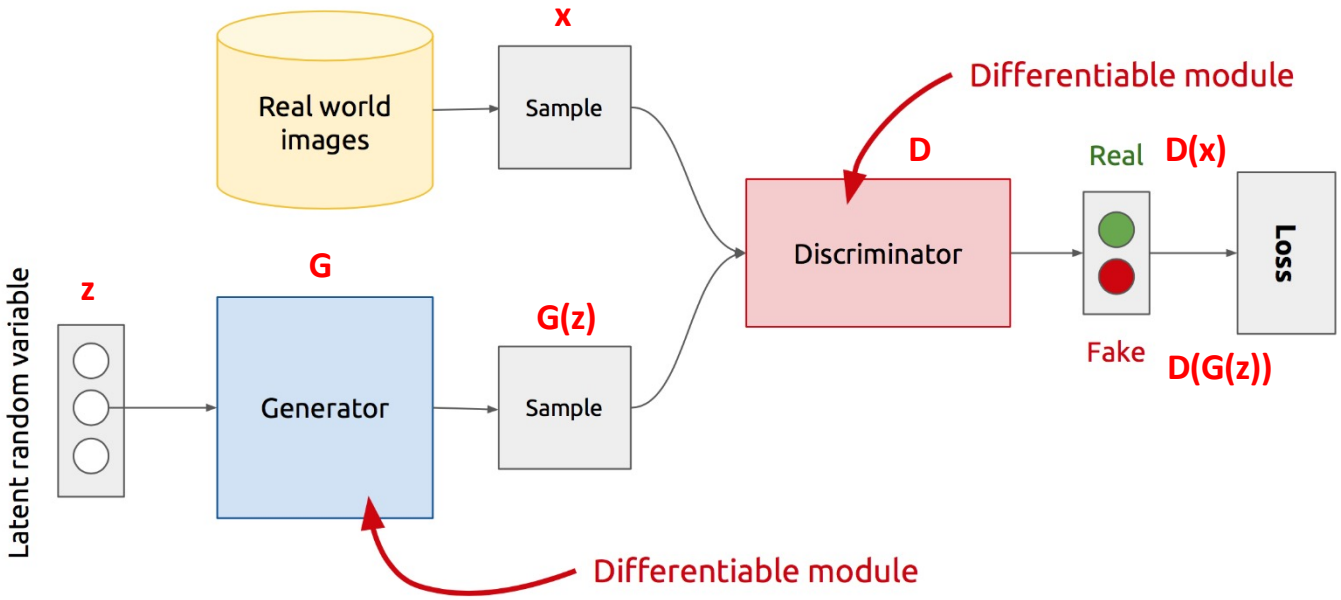
Adversarial Training

- **Adversarial Samples**
  - We can generate adversarial samples to **fool a discriminative model**
  - We can use those adversarial samples to **make models robust**
  - Repeat this and we get better discriminative model



- **GANs extend that idea to generative models**
  - **Generator: generate fake samples, tries to fool the Discriminator**
  - **Discriminator: try to distinguish between real and fake samples**
  - Train them against each other
  - Repeat this and we get better Generator and Discriminator

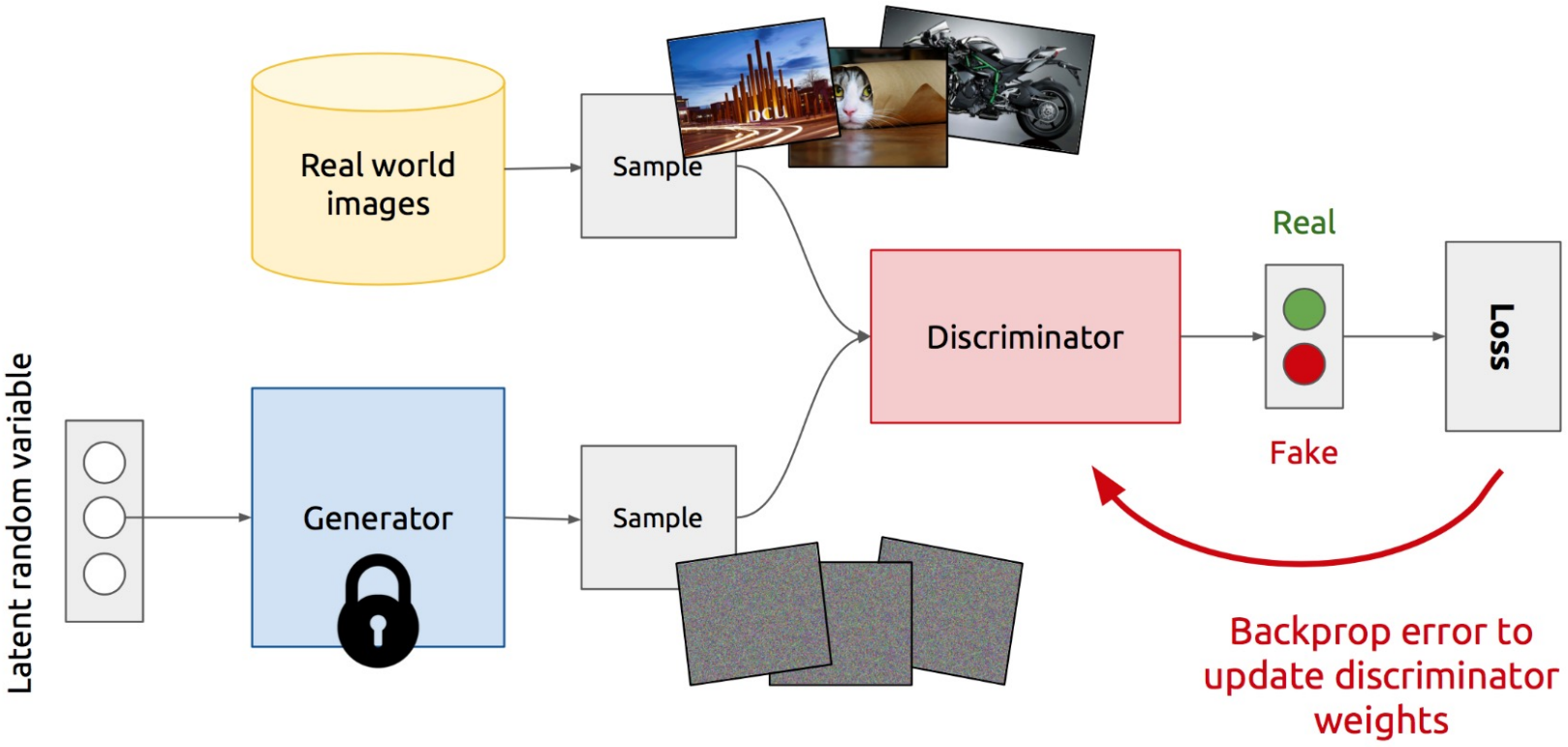
GAN's Architecture



- **Z** is some random noise (Gaussian/Uniform)
- **Z** can be thought as the latent representation of the image

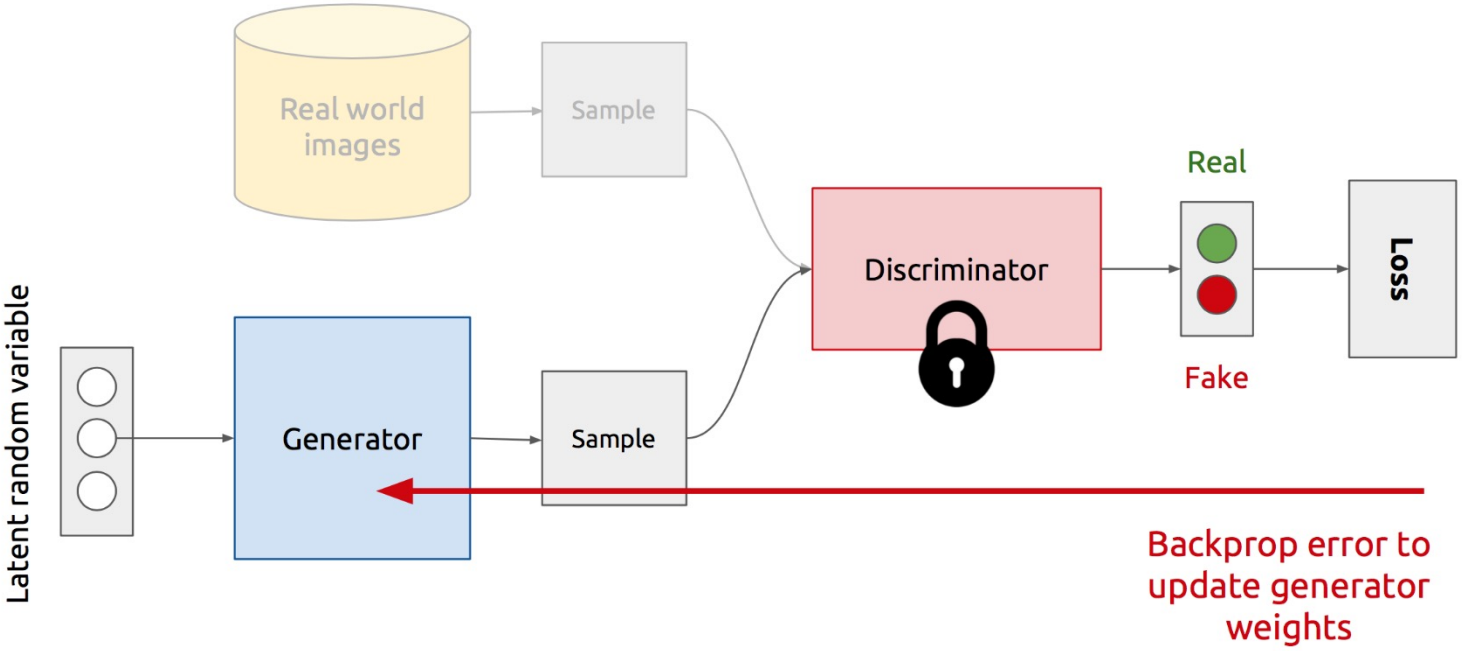
https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Training Discriminator



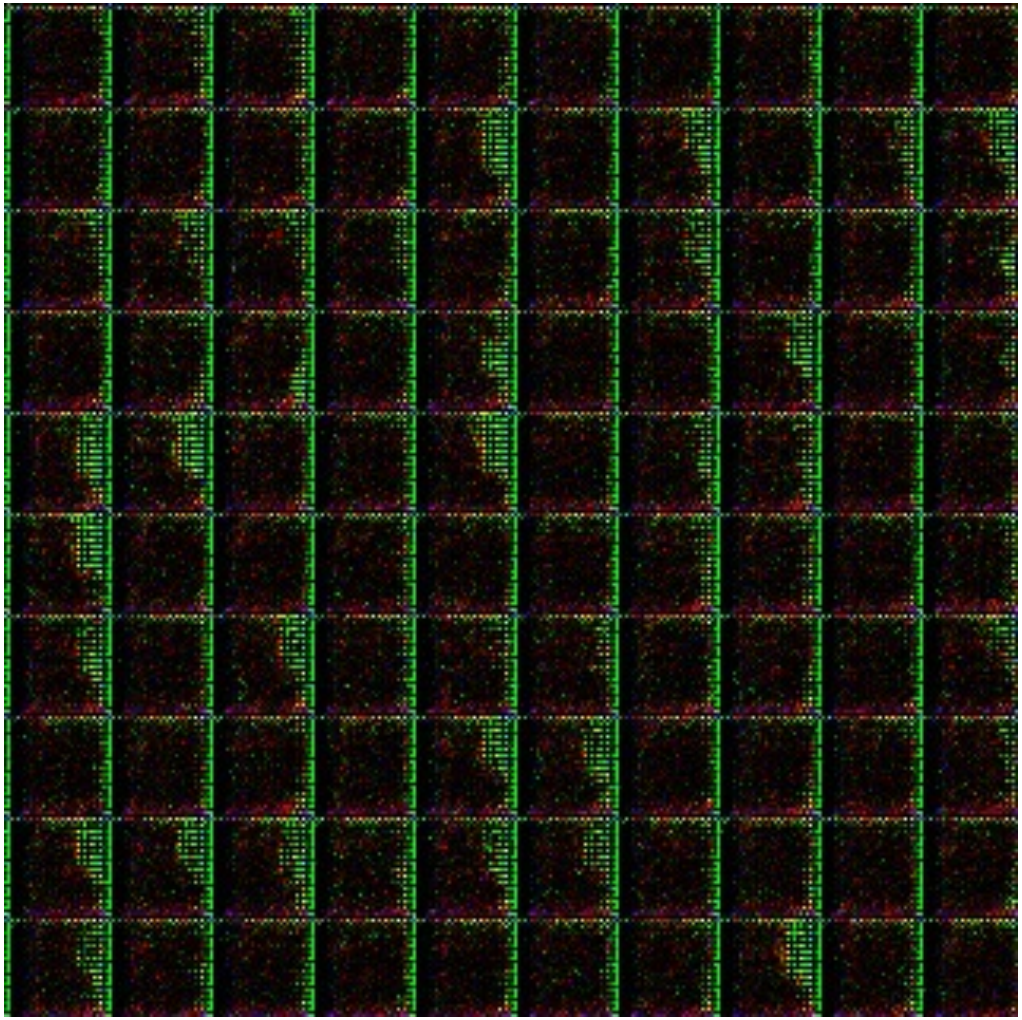
https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Training Generator



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

Generator in Action



https://openai.com/blog/generative-models/

GAN's Formulation

$$\min_G \max_D V(D,G)$$

- It is formulated as a **minimax game**, where:
  - The Discriminator is trying to maximize $V(D,G)$
  - The Generator is trying to minimize $V(D,G)$
  - $D(x)$ should be 1 for real examples, 0 for fake examples

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

- The Nash equilibrium of this particular game is achieved at:
  - $P_{data}(x) = P_{gen}(x) \;\; \forall x$
  - $D(x) = \frac{1}{2} \;\; \forall x$

Adversarial Training

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**

Discriminator updates

- 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

Generator updates

- 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.

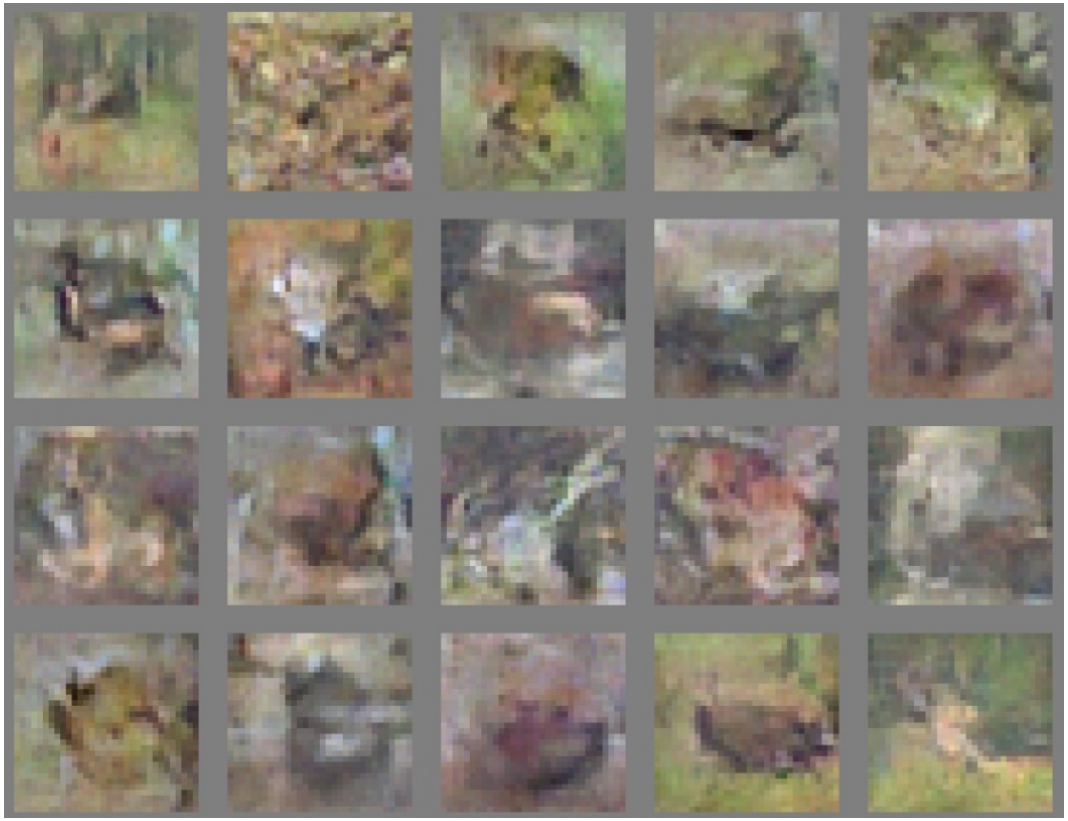
Applications

Faces

Goodfellow, Ian, et al. **"Generative adversarial nets."** *Advances in neural information processing systems*. 2014.

Applications

CIFAR



Goodfellow, Ian, et al. **"Generative adversarial nets."** *Advances in neural information processing systems*. 2014.

Advantages of GANs

- **Plenty of existing work on Deep Generative Models**
  - Boltzmann Machines
  - Deep Belief Nets
  - Variational AutoEncoders (VAE)

- **Why GANs?**
  - Sampling (or generation) is straightforward
  - Training doesn't involve Maximum Likelihood Estimation (MLE)
  - Robust to overfitting since Generator never sees training data
  - Empirically, GANs are good at capturing the modes of the distribution

Goodfellow, Ian. **"NIPS 2016 Tutorial: Generative Adversarial Networks."** arXiv preprint arXiv:1701.00160 (2016).

Controversy of GANs

Related **What happened with Jurgen Schmidhuber at NIPS 2016 during the GAN tutorial?**

Jurgen Schmidhuber interrupted the lecture on GANs by Ian Goodfellow claiming that he did very similar work earlier which was overlooked. This caused outrage in the media, massive discussions on Reddit and Twitter, and some serious words were thrown by some very famous people. The whole dispute about recognition shows how cutthroat the academic world at the top really is and how much of it is about credit.

Controversy of GANs

Juergen Schmidhuber

FOLLOW

King Abdullah University of Science and Technology / The Swiss AI Lab, IDSIA / University of Lugano
Verified email at kaust.edu.sa - Homepage

computer science artificial intelligence reinforcement learning neural networks physics

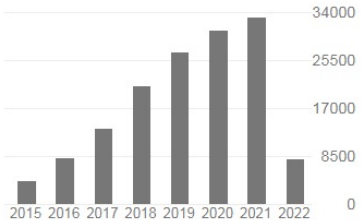
GET MY OWN PROFILE

TITLE	CITED BY	YEAR
Long short-term memory S Hochreiter, J Schmidhuber Neural computation 9 (8), 1735-1780	63949	1997
Deep learning in neural networks: An overview J Schmidhuber Neural networks 61, 85-117	15569	2015
Learning to forget: Continual prediction with LSTM FA Gers, J Schmidhuber, F Cummins Neural computation 12 (10), 2451-2471	5427	2000
Multi-column deep neural network for traffic sign classification D Cireşan, U Meier, J Masci, J Schmidhuber Neural networks 32, 333-338	5367 *	2012
Multi-column deep neural networks for image classification D Ciregan, U Meier, J Schmidhuber 2012 IEEE conference on computer vision and pattern recognition, 3642-3649	~~5327~~	2012
LSTM: A search space odyssey K Greff, RK Srivastava, J Koutník, BR Steunebrink, J Schmidhuber IEEE transactions on neural networks and learning systems 28 (10), 2222-2232	4497	2016
Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks A Graves, S Fernández, F Gomez, J Schmidhuber Proceedings of the 23rd international conference on Machine learning, 369-376	4109	2006
Framewise phoneme classification with bidirectional LSTM and other neural network architectures A Graves, J Schmidhuber Neural networks 18 (5-6), 602-610	4020	2005
A novel connectionist system for unconstrained handwriting recognition A Graves, M Liwicki, S Fernández, R Bertolami, H Bunke, J Schmidhuber IEEE transactions on pattern analysis and machine intelligence 31 (5), 855-868	2107	2008
Gradient flow in recurrent nets: the difficulty of learning long-term dependencies S Hochreiter, Y Bengio, P Frasconi, J Schmidhuber A field guide to dynamical recurrent neural networks. IEEE Press	1963	2001
Highway networks RK Srivastava, K Greff, J Schmidhuber arXiv preprint arXiv:1505.00387	1905	2015

Cited by

VIEW ALL

	All	Since 2017
Citations	160903	132972
h-index	108	84
i10-index	391	244



Public access

VIEW ALL

3 articles — not available
147 articles — available

Based on funding mandates

Co-authors

VIEW ALL

- Sepp Hochreiter — Institute for Machine Learning, J…
- Dan Ciresan — Conndera Research
- Faustino Gomez — CEO NNAISENSE SA. Former S…
- Alex Graves — University of Toronto
- Felix Gers — Professor of Computer Science, …
- luca maria gambardella — Faculty of informatics, USI, IDSI

Controversy of GANs

nature

Explore content · About the journal · Publish with us · Subscribe

nature › review articles › article

Published: 27 May 2015

Deep learning

Yann LeCun ✉, Yoshua Bengio & Geoffrey Hinton

Nature **521**, 436–444 (2015) | Cite this article

704k Accesses | **31338** Citations | **1125** Altmetric | Metrics

Abstract

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

Problems with GANs

- **Probability distribution is implicit**
  - Not straightforward to compute P(X)
  - Thus vanilla GANs are only good for sampling/generation
- **Training is hard**
  - Gradient vanishing
  - Non-convergence
  - Mode-collapse

Goodfellow, Ian. **"NIPS 2016 Tutorial: Generative Adversarial Networks."** arXiv preprint arXiv:1701.00160 (2016).

Gradient Vanishing

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

$$\nabla_{\theta_G} V(D,G) = \nabla_{\theta_G} \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$

- $\nabla_a \log\left(1-\sigma(a)\right) = \frac{-\nabla_a\, \sigma(a)}{1-\sigma(a)} = \frac{-\sigma(a)\left(1-\sigma(a)\right)}{1-\sigma(a)} = -\sigma(a) = -D\left(G(z)\right)$
- $\sigma(.)$ is the activation function in the D
- Gradient goes to 0 if D is confident, i.e. $D\left(G(z)\right) \to 0$

- Minimize $-\mathbb{E}_{z\sim q(z)}\left[\log D\left(G(z)\right)\right]$ for **Generator** instead (keep Discriminator as it is)

Non-Convergence

- **Deep learning models (in general) involve a single player**
  - The player tries to minimize its loss
  - Use SGD (with backpropagation) to find the optimal parameters, and SGD has convergence guarantees (under certain conditions)
  - **Problem:** with non-convexity, we might converge to local optima

$$\min_G L(G)$$

- **GANs instead involve two (or more) players**
  - Discriminator is trying to maximize its reward
  - Generator is trying to minimize Discriminator's reward

$$\min_G \max_D V(D, G)$$

  - SGD was not designed to find the Nash equilibrium of a game
  - **Problem:** we might not converge to the Nash equilibrium at all

Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

Non-Convergence

$$\min_{x} \max_{y} \; xy$$

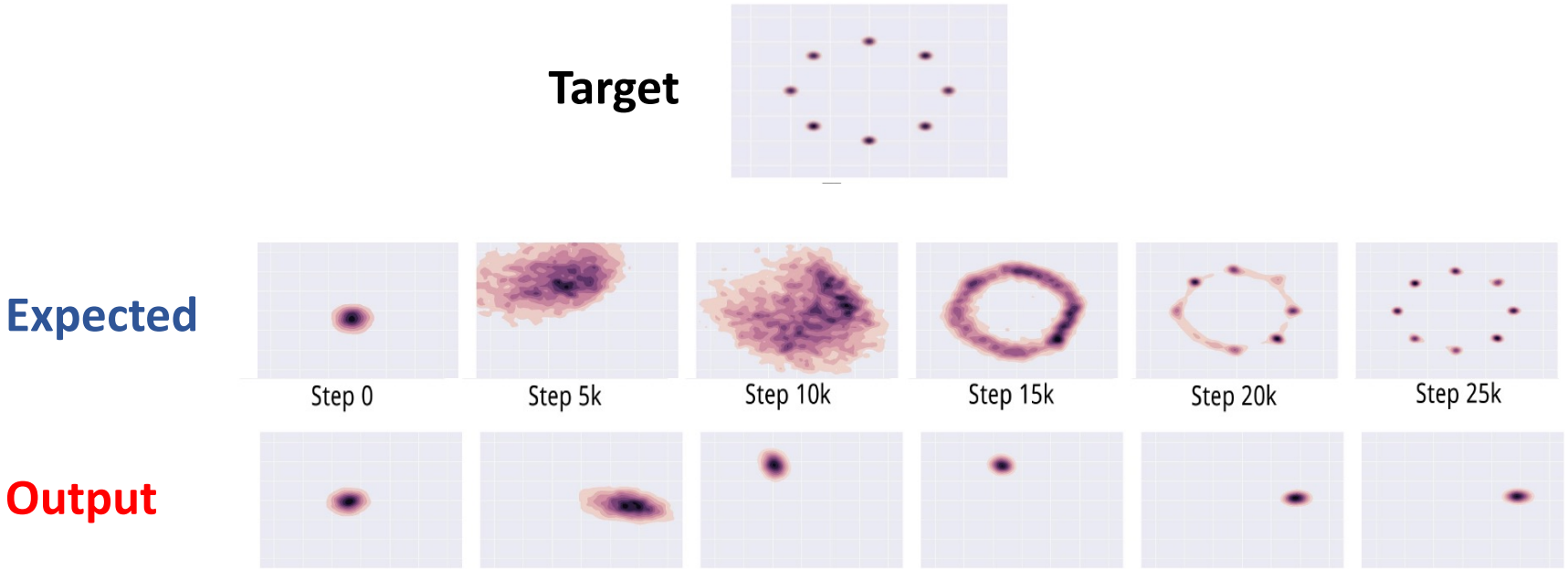
- $\frac{\partial}{\partial x} = -y \quad \ldots \quad \frac{\partial}{\partial y} = x$
- $\frac{\partial^2}{\partial y^2} = \frac{\partial}{\partial x} = -y$
- Even with a small learning rate, it will not converge

– State 1:	x > 0	y > 0	V > 0	Increase y	Decrease x
– State 2:	x < 0	y > 0	V < 0	Decrease y	Decrease x
– State 3:	x < 0	y < 0	V > 0	Decrease y	Increase x
– State 4 :	x > 0	y < 0	V < 0	Increase y	Increase x
– State 5:	x > 0	y > 0	V > 0	Increase y	Decrease x

== State 1

Mode-Collapse

- **Generator fails to output diverse samples**, and maps several different input ***z*** values to the same output point



Metz, Luke, et al. **"Unrolled Generative Adversarial Networks."** arXiv preprint arXiv:1611.02163 (2016).

Some Real Examples

- **Generator makes multiple images that contain the same color or texture themes**
  - multiple images containing different views of the same person

Reed, S., et al. ***Generating interpretable images with controllable structure.*** Technical report, 2016. 2, 2016.

Some Solutions

- **Mini-Batch GANs**
- **Supervision with labels**
- **Some recent attempts**
  - Unrolled GANs
  - W-GANs

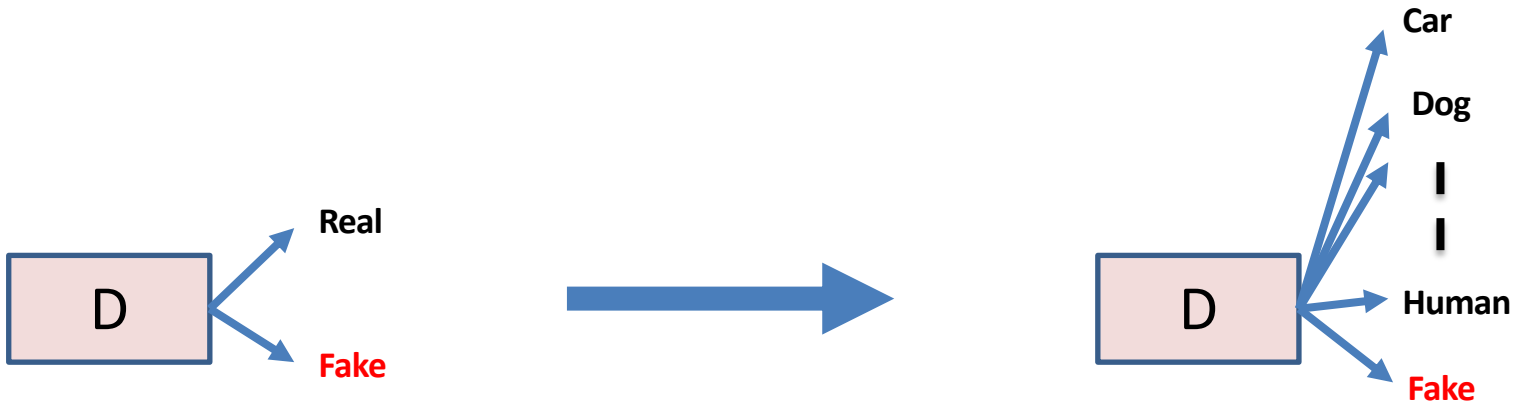
Mini-Batch GANs

- **To address the problem**
  - Let the Discriminator **look at the entire batch instead of single examples**
  - If there is lack of diversity, it will mark the examples as fake
- **Extract features that capture diversity in the mini-batch**
  - L2 norm of the difference between all pairs from the batch
  - Feed those features to the discriminator along with the image
- **Feature values will differ b/w diverse and non-diverse batches**
  - Discriminator will rely on those features for classification
- **This in turn**
  - Will force the Generator to match those feature values with the real data
  - Will generate diverse batches

Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

Supervision with Labels

- Label information of the real data might help



- Empirically generates much better samples

Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

Alternate View of GANs

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]$$

$$D^* = \arg\max_D V(D,G)$$

- Discriminator's strategy was **Real = 1, Fake = 0**, i.e., $D(x) \to 1, \quad D\left(G(z)\right) \to 0$

- Alternatively, we can flip the binary classification labels, i.e., **Fake = 1, Real = 0**

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

- In this new formulation, Discriminator's strategy will be $D(x) \to 0, \; D\left(G(z)\right) \to 1$

Zhao, Junbo, Michael Mathieu, and Yann LeCun. **"Energy-based generative adversarial network."** arXiv preprint arXiv:1609.03126 (2016)

Alternate View of GANs

- If all we want to encode is $D(x) \rightarrow 0, \quad D\big(G(z)\big) \rightarrow 1$

We can use this

$$D^* = argmin_D \, \mathbb{E}_{x \sim p(x)} \log\big(D(x)\big) + \mathbb{E}_{z \sim q(z)} \left[\log\Big(1 - D\big(G(z)\big)\Big)\right]$$

- Now, we can replace cross-entropy with any loss function **(Hinge Loss)**

$$D^* = argmin_D \, \mathbb{E}_{x \sim p(x)} D(x) + \mathbb{E}_{z \sim q(z)} \max\Big(0, m - D\big(G(z)\big)\Big)$$

- And thus, instead of outputting probabilities, Discriminator just has to output:
  - High values for fake samples
  - Low values for real samples

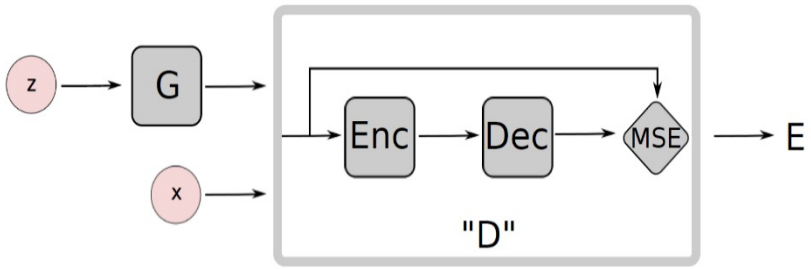
Zhao, Junbo, Michael Mathieu, and Yann LeCun. **"Energy-based generative adversarial network."** arXiv preprint arXiv:1609.03126 (2016)

Energy-Based GANs

- Modified game plans
  - **Generator** will try to generate samples with low values
  - **Discriminator** will try to assign high values to fake samples

- **Use AutoEncoder inside the Discriminator**

- **Use Mean-Squared Reconstruction error as $D(x)$**
  - High reconstruction error for fake samples
  - Low reconstruction error for real samples

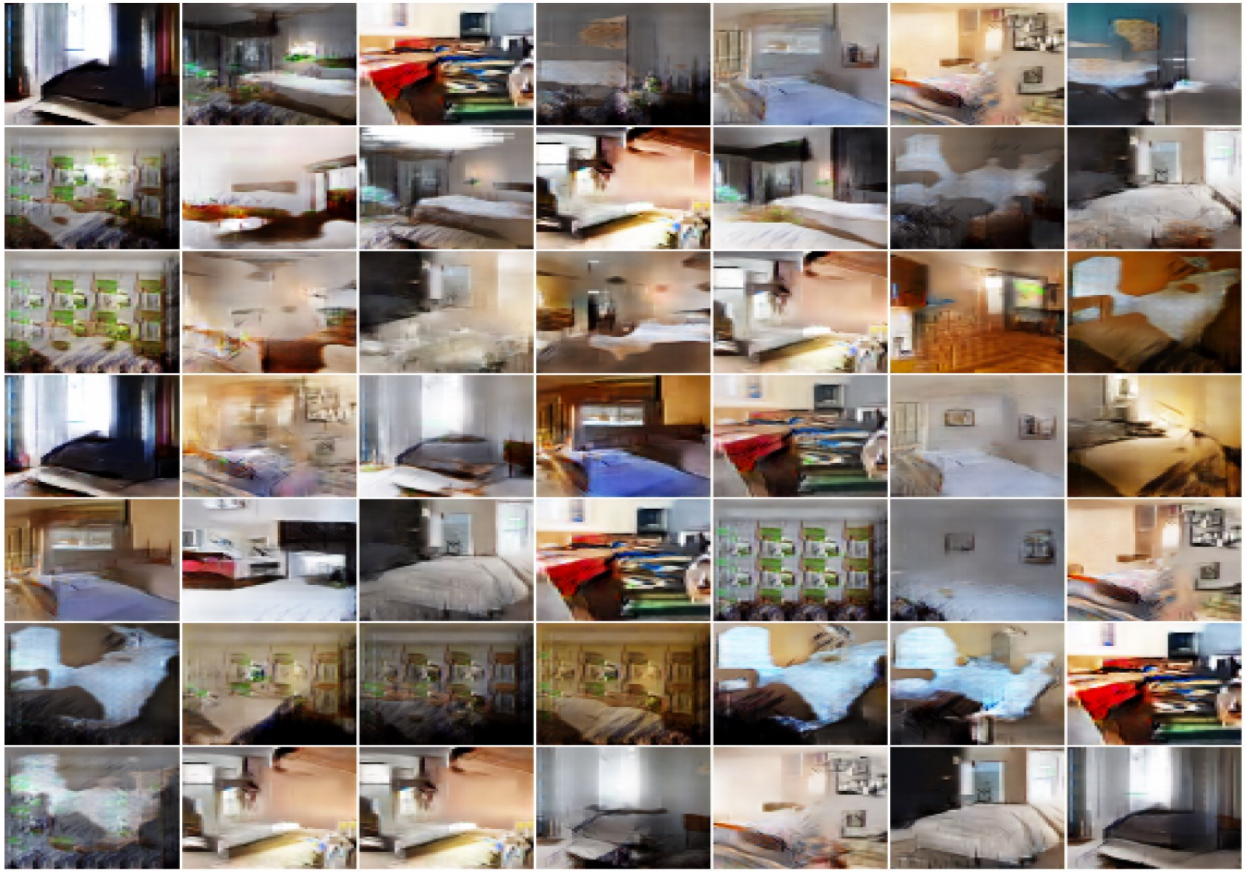


$$D(x) = ||Dec(Enc(x)) - x||_{MSE}$$

Zhao, Junbo, Michael Mathieu, and Yann LeCun. **"Energy-based generative adversarial network."** arXiv preprint arXiv:1609.03126 (2016)

Examples

Bedrooms



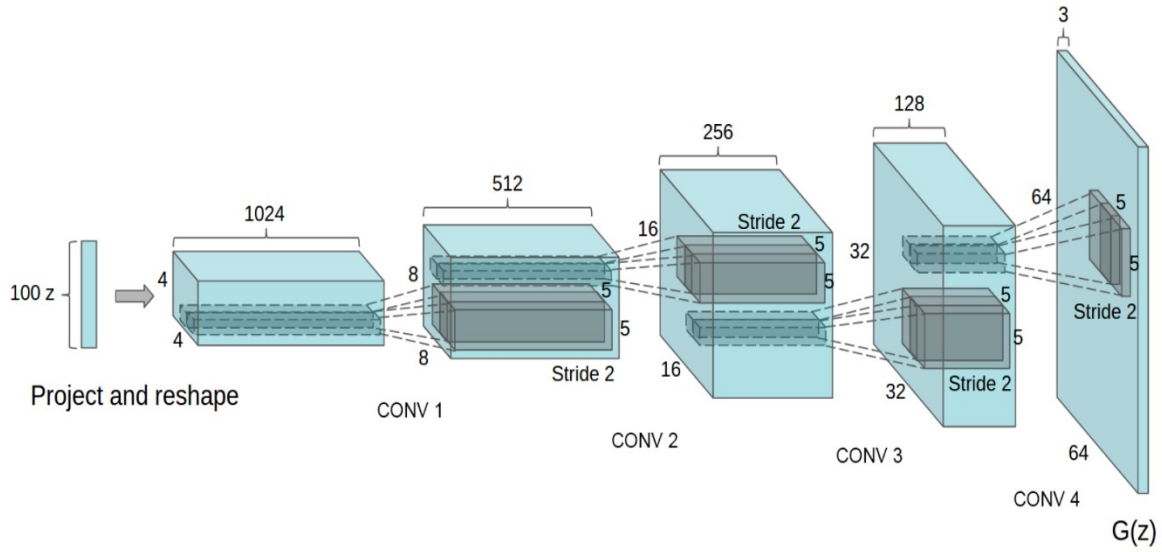
Zhao, Junbo, Michael Mathieu, and Yann LeCun. **"Energy-based generative adversarial network."** arXiv preprint arXiv:1609.03126 (2016)

Examples

Celebs

Deep Convolutional GANs (DCGANs)

Generator Architecture

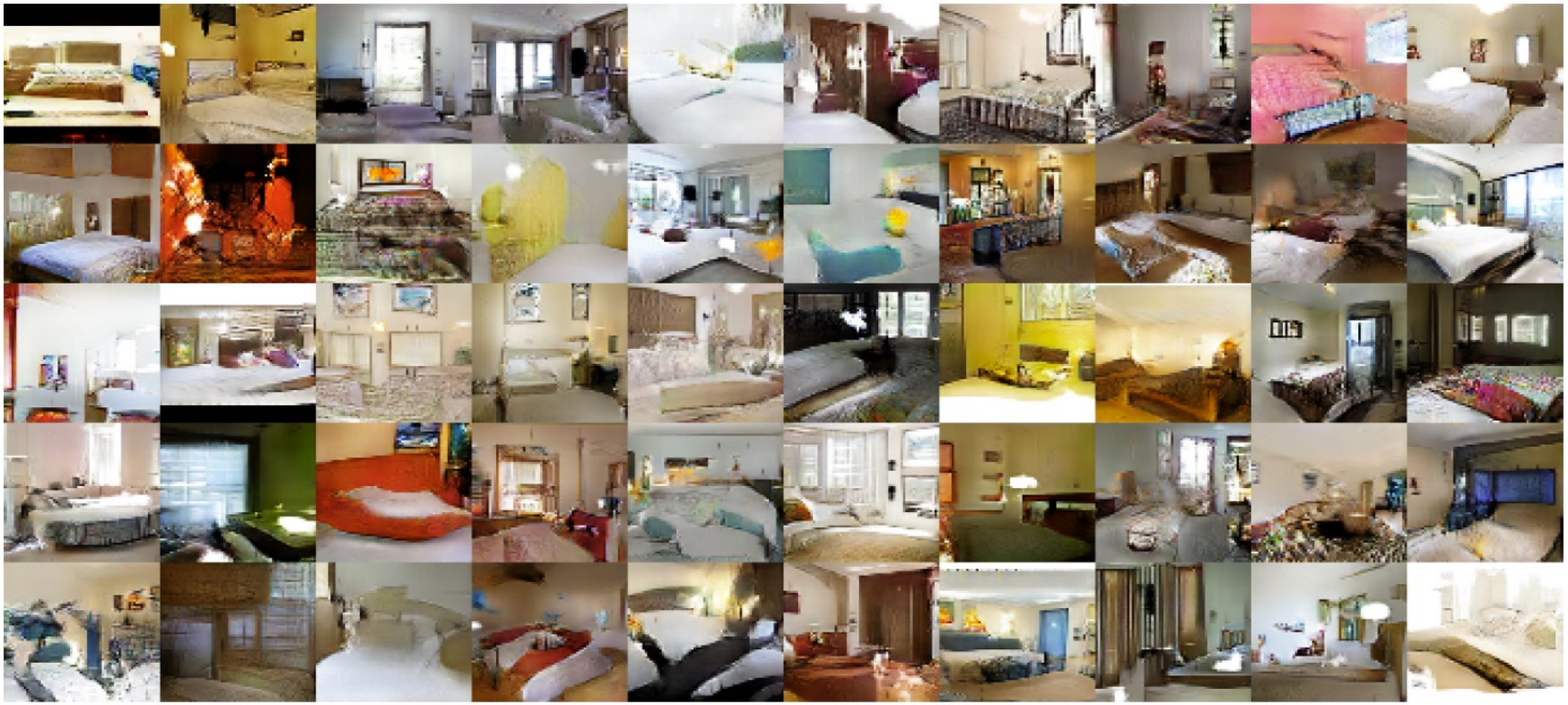


Key ideas:

- Replace fully-connected hidden layers with **convolutions**
- Use Batch Normalization after each layer
- **Inside Generator**
  - Use ReLU for hidden layers
  - Use Tanh for output layer

Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).

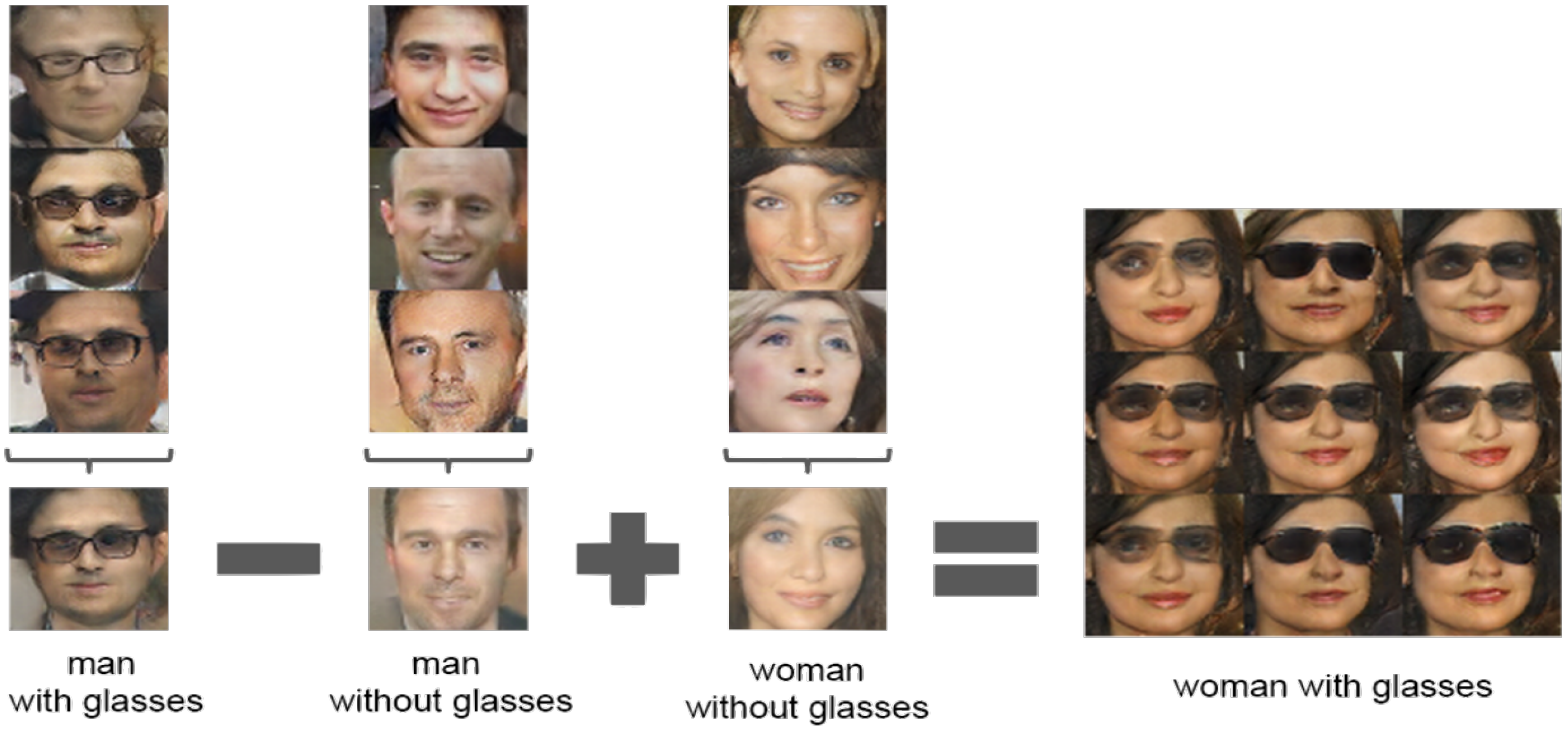
DCGAN: Bedroom Images



Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).

DCGANs: Latent Vectors

Latent vectors capture interesting patterns



Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." arXiv:1511.06434 (2015).

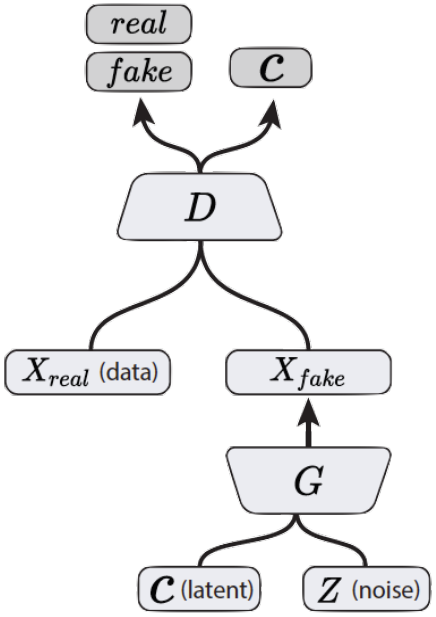
How to Reward Disentanglement?

- **Disentanglement means individual dimensions independently capturing key attributes of the image**

- **Let's partition the noise vector into 2 parts**
  - **z** vector will capture slight variations in the image
  - **c** vector will capture key attributes of the image
    - Digit, angle and thickness of images in MNIST dataset

- If **c** vector captures the key attributes in the image,

  Will c and x_{fake} be highly correlated or weakly correlated?



InfoGAN
(Chen, et al., 2016)

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. InfoGAN: Interpretable Representation Learning by Information Maximization Generative Adversarial Nets

Recap: Mutual Information

- Mutual Information captures the **mutual dependence** between two variables
- Mutual information between two variables X, Y is defined as:

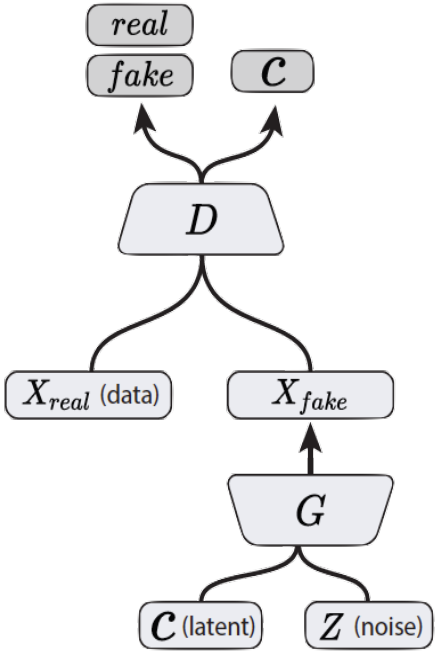
$$I(X;Y) = \sum_{x,y} p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

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

InfoGAN

- We want to maximize the mutual information I between $\boldsymbol{c}$ and $\mathbf{x} = \boldsymbol{G}(\boldsymbol{z}, \boldsymbol{c})$
- Incorporate in the value function of the minimax game

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



InfoGAN
(Chen, et al., 2016)

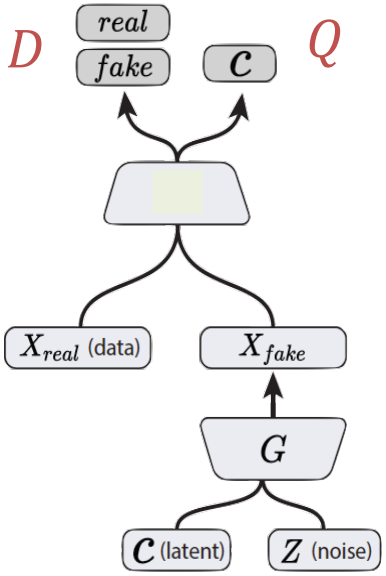
Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. InfoGAN: Interpretable Representation Learning by Information Maximization Generative Adversarial Nets, NIPS (2016).

InfoGAN

Mutual Information's Variational Lower Bound

$$I\big(c; G(z,c)\big) = H(c) - H(c|G(z,c))$$
$$= \mathbb{E}_{x \sim G(z,c)}\Big[\mathbb{E}_{c' \sim P(c|x)}[\log P(c'|x)]\Big] + H(c)$$
$$= \mathbb{E}_{x \sim G(z,c)}\Big[D_{KL}(P||Q) + \mathbb{E}_{c' \sim P(c|x)}[\log Q(c'|x)]\Big] + H(c)$$
$$\geq \mathbb{E}_{x \sim G(z,c)}\Big[\mathbb{E}_{c' \sim P(c|x)}[\log Q(c'|x)]\Big] + H(c)$$

$$\geq \mathbb{E}_{c \sim P(c),\ x \sim G(z,c)}[\boldsymbol{\log Q(c|x)}] + H(c)$$

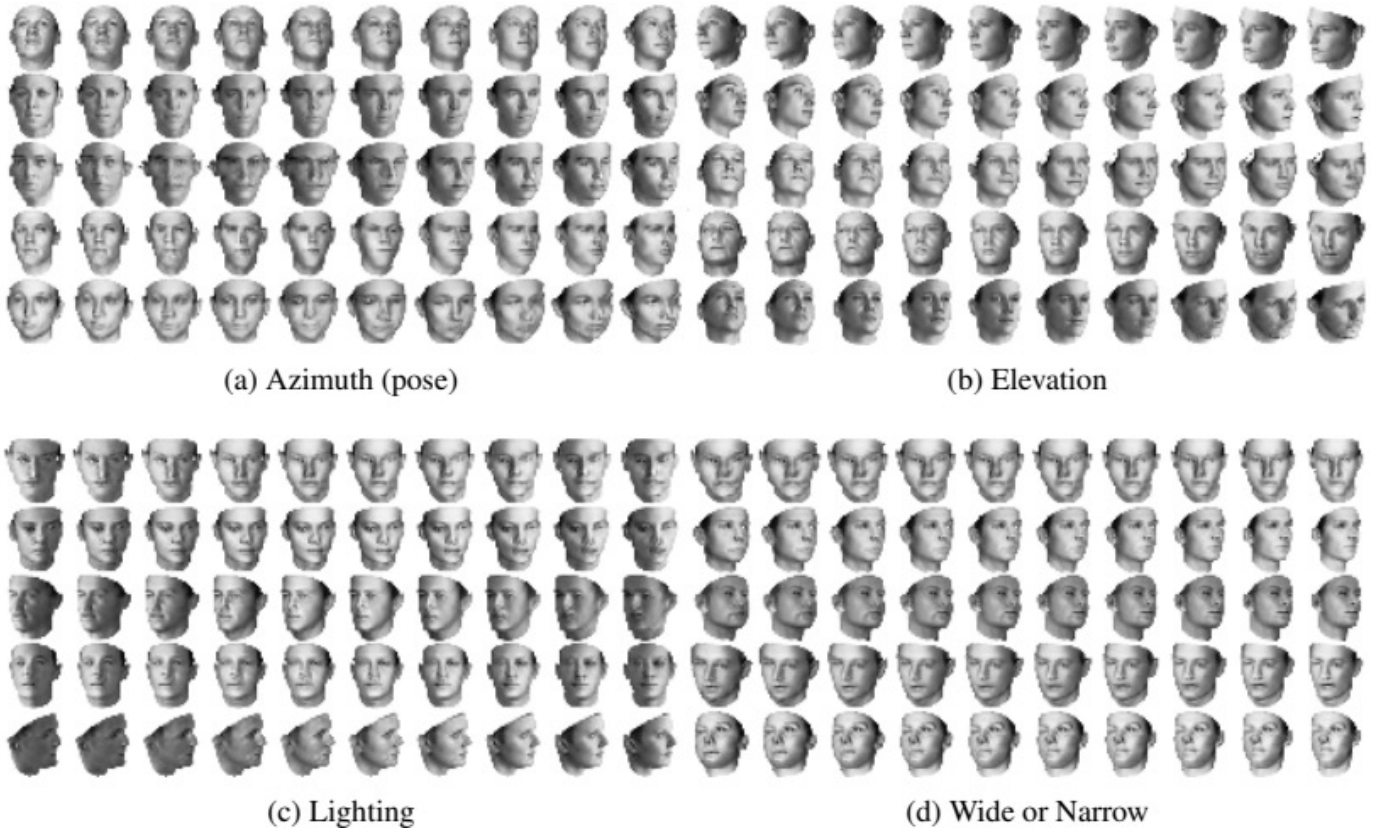


InfoGAN
(Chen, et al., 2016)

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. InfoGAN: Interpretable Representation Learning by Information Maximization Generative Adversarial Nets, NIPS (2016).

InfoGAN: Generate 3D Face Images

Show the effect of the learned continuous latent factors on the outputs as their values vary from −1 to 1



Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. InfoGAN: Interpretable Representation Learning by Information Maximization Generative Adversarial Nets, NIPS (2016).

Conditional GANs

MNIST digits generated conditioned on their class label rather than noises

$[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ →
$[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$ →
$[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]$ →
$[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$ →
$[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]$ →
$[0, 0, 0, 0, 0, 1, 0, 0, 0, 0]$ →
$[0, 0, 0, 0, 0, 0, 1, 0, 0, 0]$ →
$[0, 0, 0, 0, 0, 0, 0, 1, 0, 0]$ →
$[0, 0, 0, 0, 0, 0, 0, 0, 1, 0]$ →
$[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]$ →

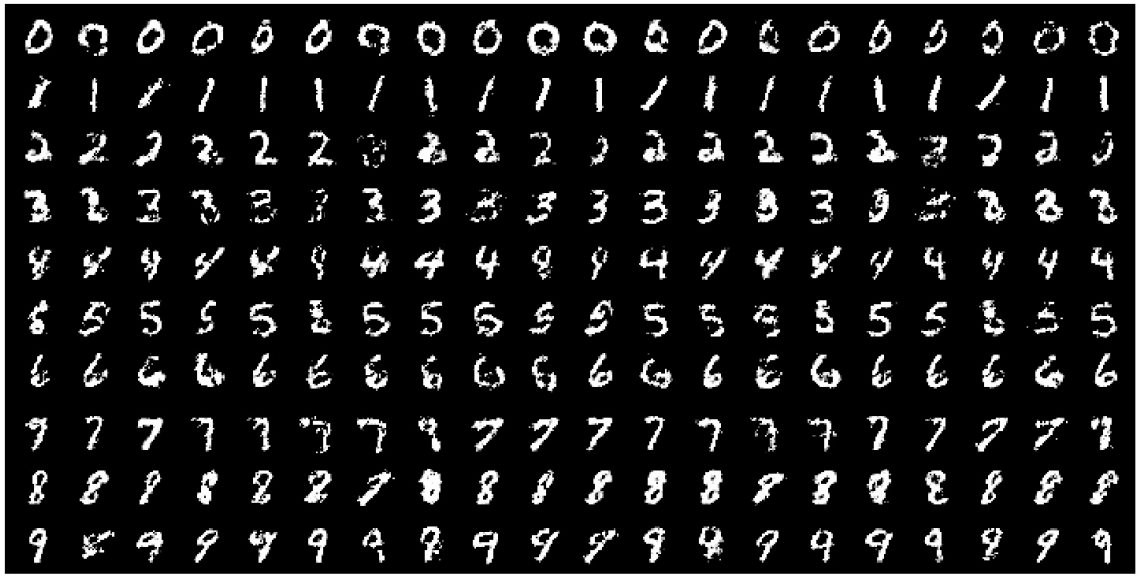
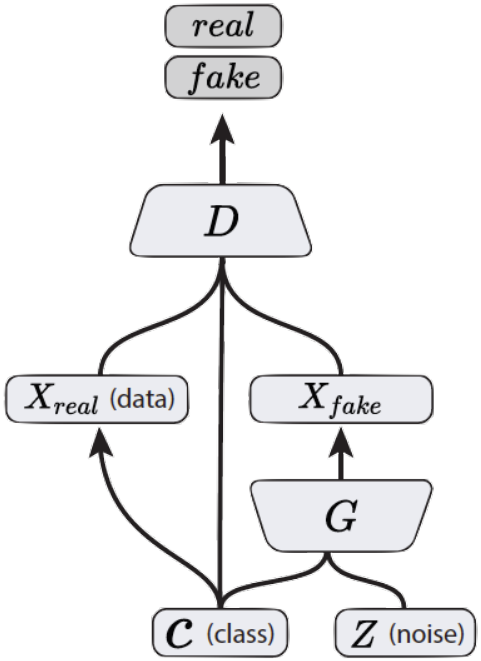


Figure 2 in the original paper.

Mirza, Mehdi, and Simon Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014).

Conditional GANs

- Simple modification to the original GAN framework that conditions the model on *additional information* for better multimodal learning
- Lend to many practical applications of GANs when we have explicit *supervision* available



Conditional GAN
(Mirza & Osindero, 2014)

Mirza, Mehdi, and Simon Osindero. "**Conditional generative adversarial nets**". arXiv preprint arXiv:1411.1784 (2014).

Odena, A., Olah, C. and Shlens, J., 2016. **Conditional image synthesis with auxiliary classifier GANs**. arXiv preprint arXiv:1610.09585.

Image-to-Image Translation

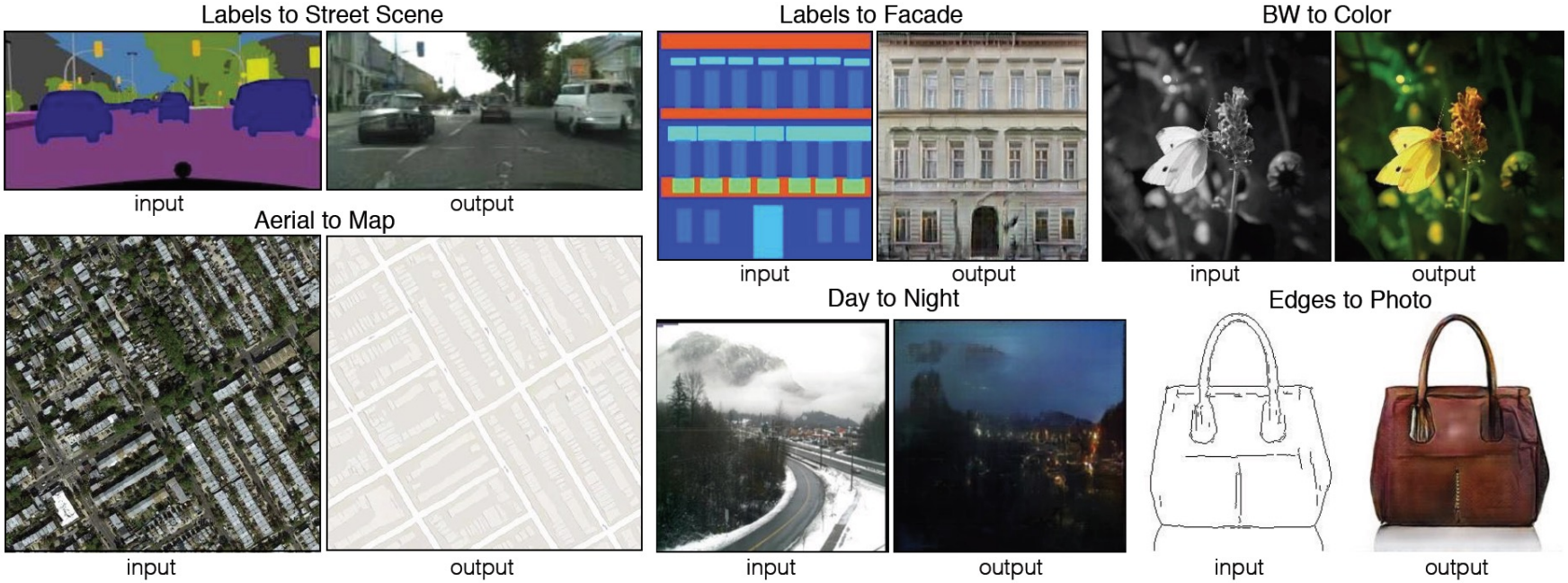


Figure 1 in the original paper.

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. "**Image-to-image translation with conditional adversarial networks**". arXiv preprint arXiv:1611.07004. (2016).

Image-to-Image Translation

- Architecture: *DCGAN*-based architecture
- Training is conditioned on the images from the source domain

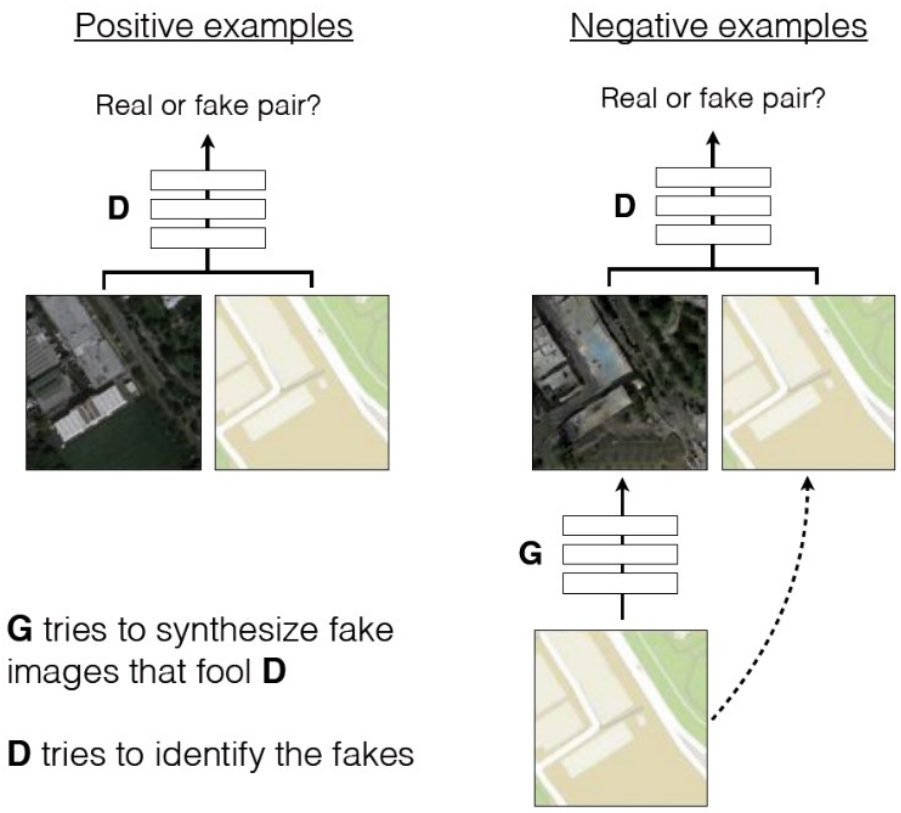


Figure 2 in the original paper.

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. "**Image-to-image translation with conditional adversarial networks**". arXiv preprint arXiv:1611.07004. (2016).

Text-to-Image Synthesis

Motivation

Given a **text description**, generate images closely associated

Uses a conditional GAN with the generator and discriminator being **condition on "dense" text embedding**

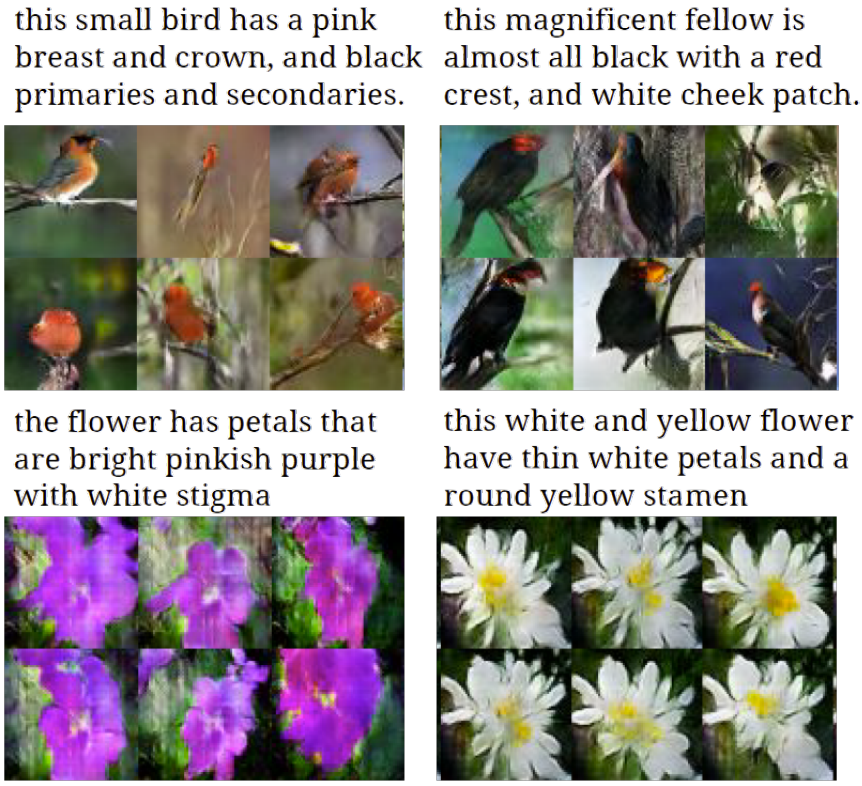


Figure 1 in the original paper.

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

Text-to-Image Synthesis

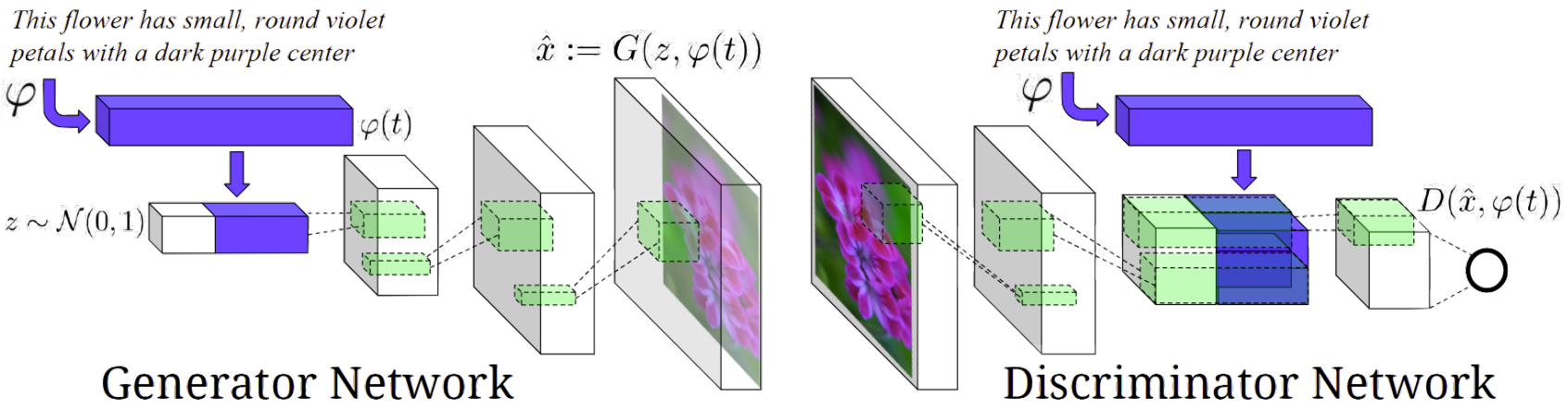


Figure 2 in the original paper.

Positive Example:
Real Image, Right Text

Negative Examples:
Real Image, Wrong Text
Fake Image, Right Text

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

Face Aging with Conditional GANs

- Uses an *Identity Preservation Optimization* based on an auxiliary network to get a better approximation of the latent code for an input image
- Latent code is then conditioned on a discrete (one-hot) embedding of age categories

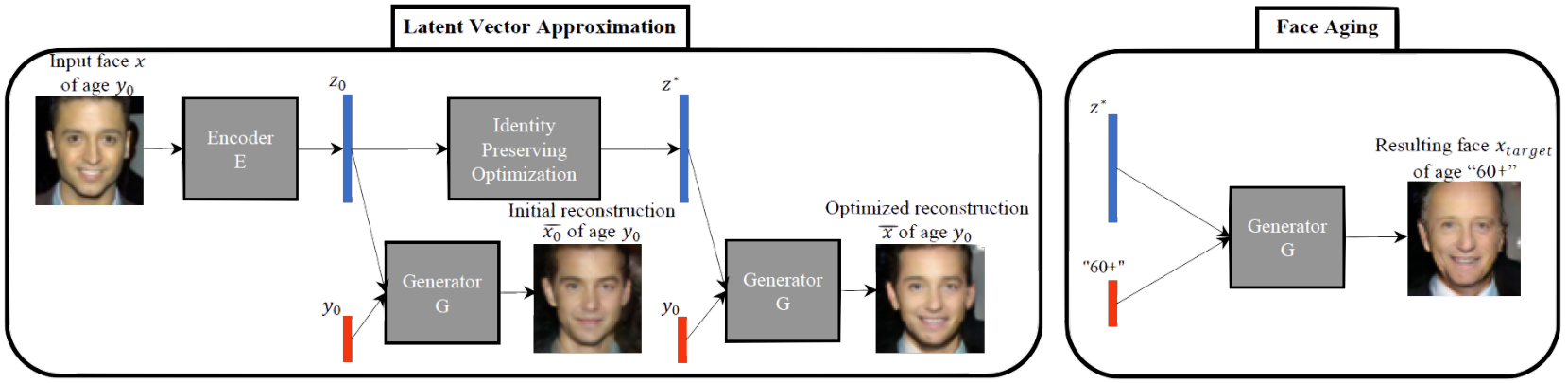


Figure 1 in the original paper.

Antipov, G., Baccouche, M., & Dugelay, J. L. (2017). "**Face Aging With Conditional Generative Adversarial Networks**". arXiv preprint arXiv:1702.01983.

Face Aging with Conditional GANs

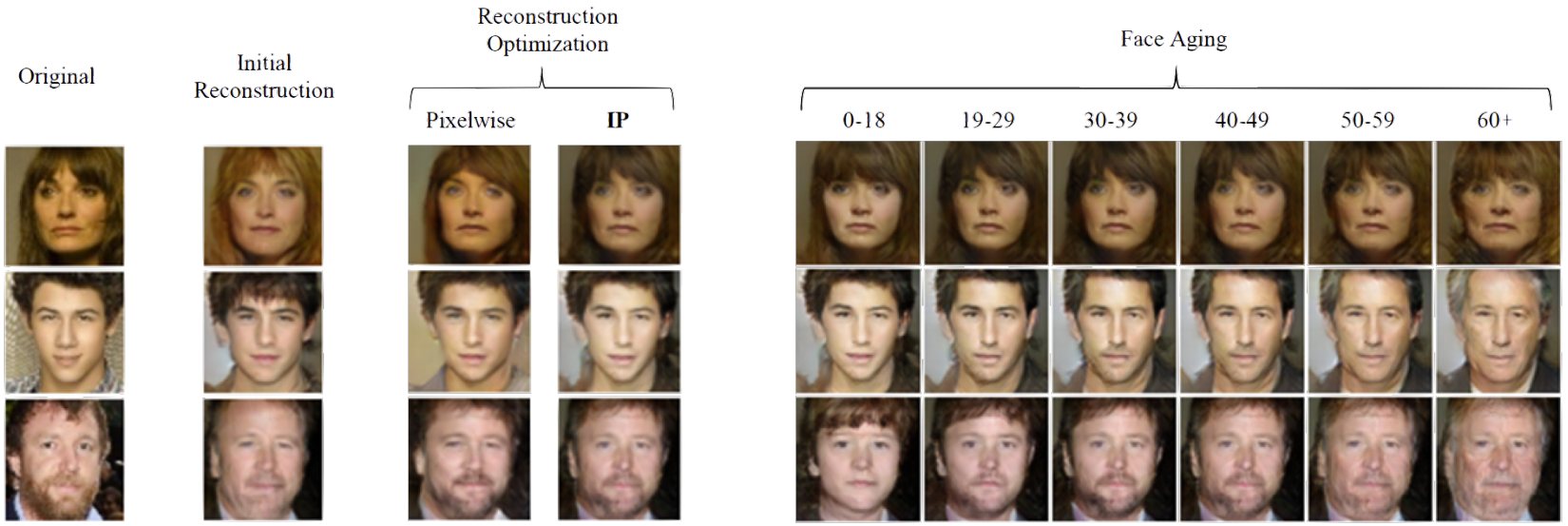


Figure 3 in the original paper.

Antipov, G., Baccouche, M., & Dugelay, J. L. (2017). "**Face Aging With Conditional Generative Adversarial Networks**". arXiv preprint arXiv:1702.01983.

Laplacian Pyramid of Adversarial Networks

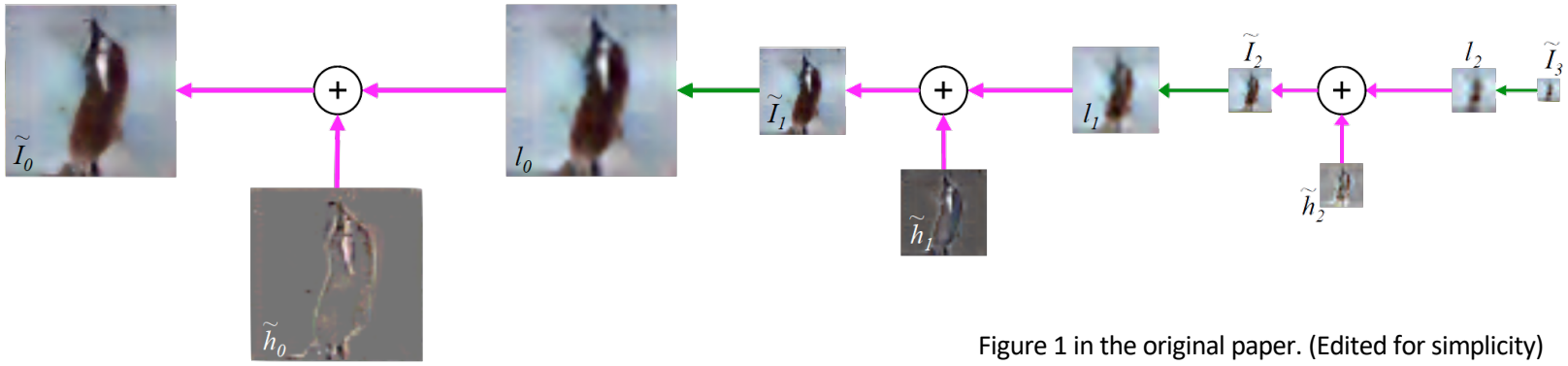


Figure 1 in the original paper. (Edited for simplicity)

- Based on the Laplacian Pyramid representation of images (1983)
- Generate high resolution (dimension) images by using a hierarchical system of GANs
- Iteratively increase image resolution and quality

Denton, E.L., Chintala, S. and Fergus, R., 2015. "**Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks**". NIPS (2015)

Laplacian Pyramid of Adversarial Networks

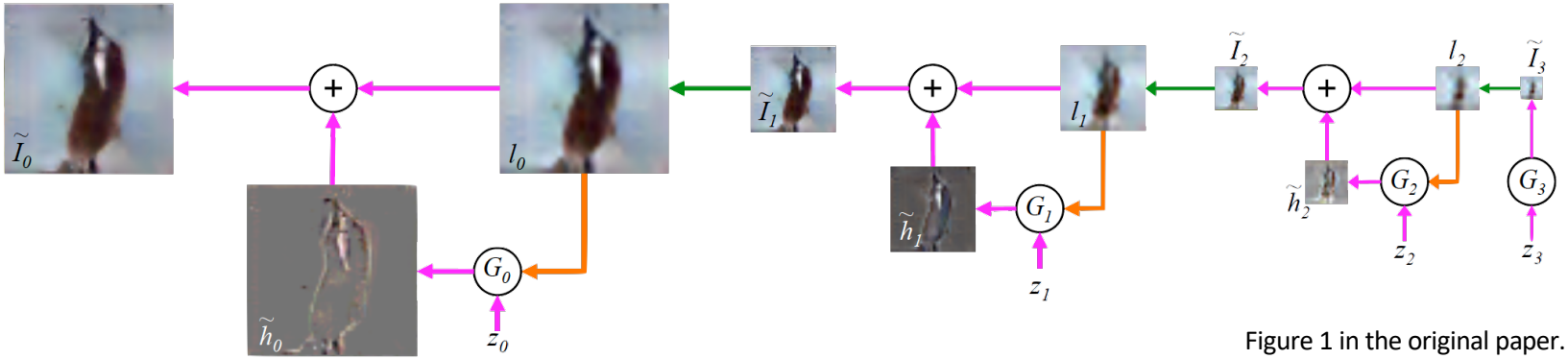


Figure 1 in the original paper.

Image generation using a LAPGAN

- Generator G_3 generates the base image $\tilde{I}_3$ from random noise input z_3
- Generators (G_2, G_1, G_0) iteratively generate the *difference image* $\hat{h}$ conditioned on previous small image l
- This *difference image* is added to an up-scaled version of previous smaller image

Denton, E.L., Chintala, S. and Fergus, R., 2015. "**Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks**". NIPS (2015)

Laplacian Pyramid of Adversarial Networks

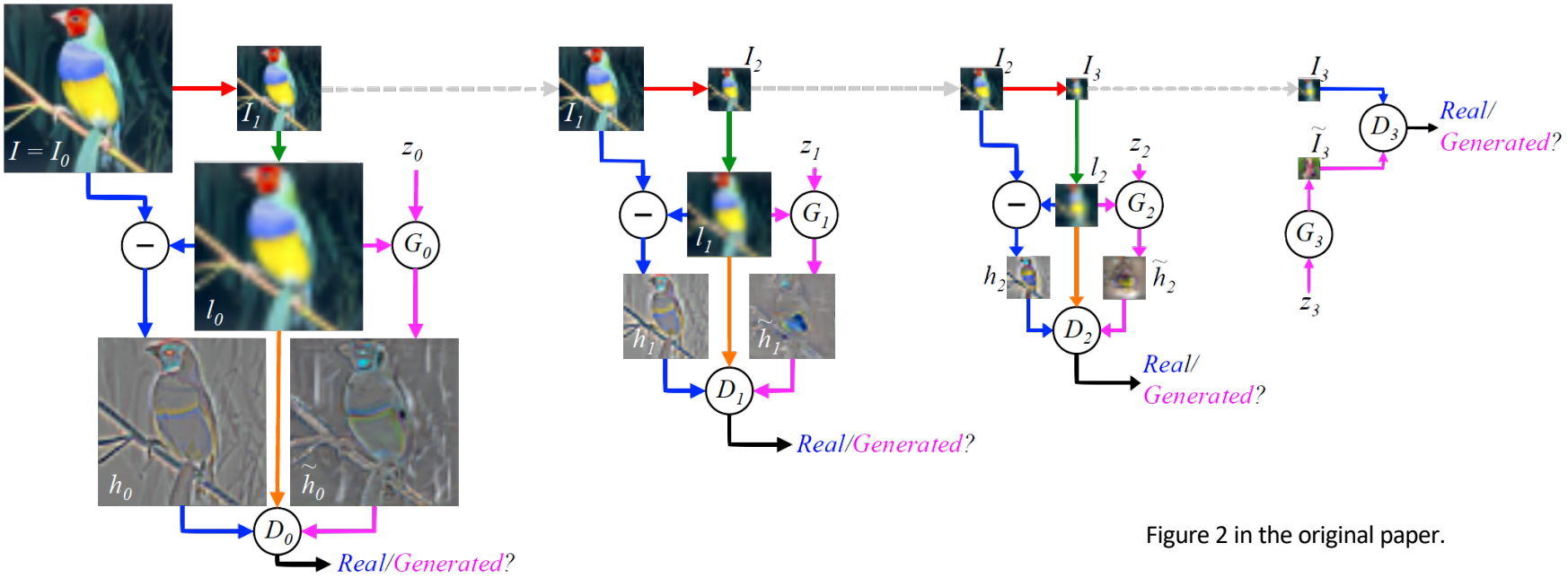


Figure 2 in the original paper.

Training Procedure:

Models at each level are trained independently to learn the required representation

Denton, E.L., Chintala, S. and Fergus, R., 2015. "**Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks**". NIPS (2015)

Adversarially Learned Inference

- Basic idea is to learn an encoder/inference network along with the generator network

- Consider the following joint distributions over x (image) and z (latent variables):

$$q(x, z) = q(x)\, q(z|x)$$ **encoder distribution**

$$p(x, z) = p(z)\, p(x|z)$$ **generator distribution**

Dumoulin, Vincent, et al. "**Adversarially learned inference**". *arXiv preprint arXiv:1606.00704* (2016).

Adversarially Learned Inference

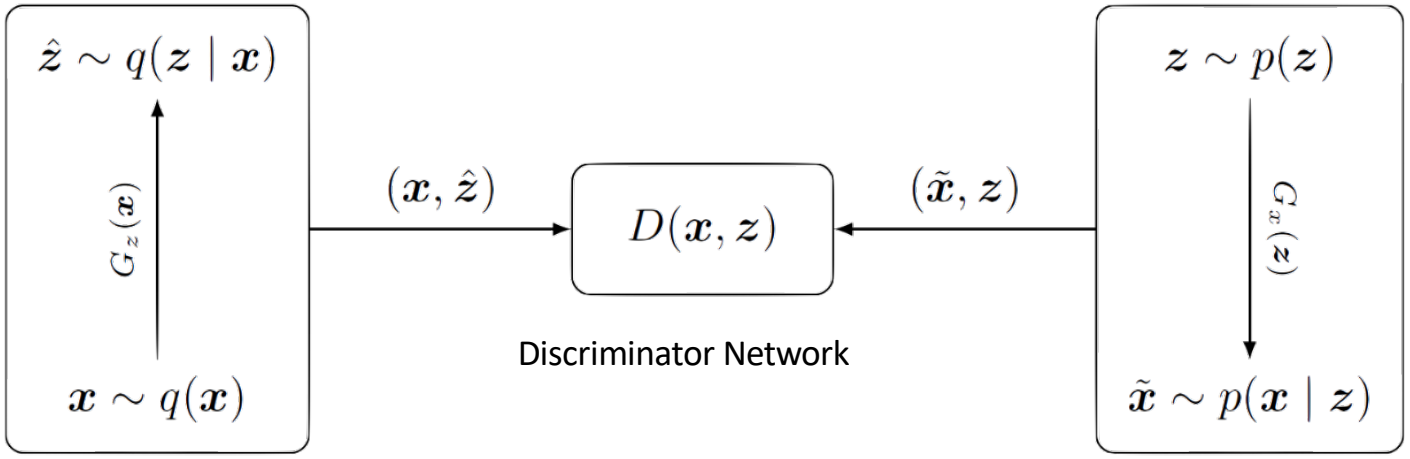


Figure 1 in the original paper.

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

Dumoulin, Vincent, et al. "**Adversarially learned inference**". *arXiv preprint arXiv:1606.00704* (2016).

Summary

- GANs are generative models that are implemented using two stochastic neural network modules: **Generator** and **Discriminator**
- **Generator** tries to generate samples from random noise as input
- **Discriminator** tries to distinguish the samples from Generator and samples from the real data distribution
- Both networks are trained **adversarially** to fool the other component. In this process, both models become better at their respective tasks

Outline

1. Course Review
2. Generative Adversarial Networks
3. Variational Autoencoder
4. Other Generation Methods

Variational Autoencoder

1

Auto-Encoding Variational Bayes

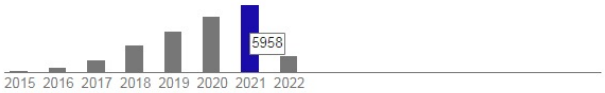
Authors: Diederik P Kingma, Max Welling

Publication date: 2013/12/20

Journal: arXiv preprint arXiv:1312.6114

Description: How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

Total citations: Cited by 20124



2

Stochastic backpropagation and approximate inference in deep generative models

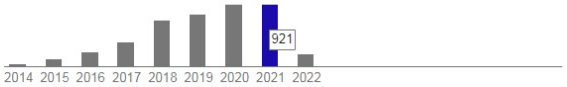
Authors: Danilo Jimenez Rezende, Shakir Mohamed, Daan Wierstra

Publication date: 2014/1/16

Conference: International Conference on Machine Learning, 2014

Description: We marry ideas from deep neural networks and approximate Bayesian inference to derive a generalised class of deep, directed generative models, endowed with a new algorithm for scalable inference and learning. Our algorithm introduces a recognition model to represent an approximate posterior distribution and uses this for optimisation of a variational lower bound. We develop stochastic backpropagation–rules for gradient backpropagation through stochastic variables–and derive an algorithm that allows for joint optimisation of the parameters of both the generative and recognition models. We demonstrate on several real-world data sets that by using stochastic backpropagation and variational inference, we obtain models that are able to generate realistic samples of data, allow for accurate imputations of missing data, and provide a useful tool for high-dimensional data visualisation.

Total citations: Cited by 4208



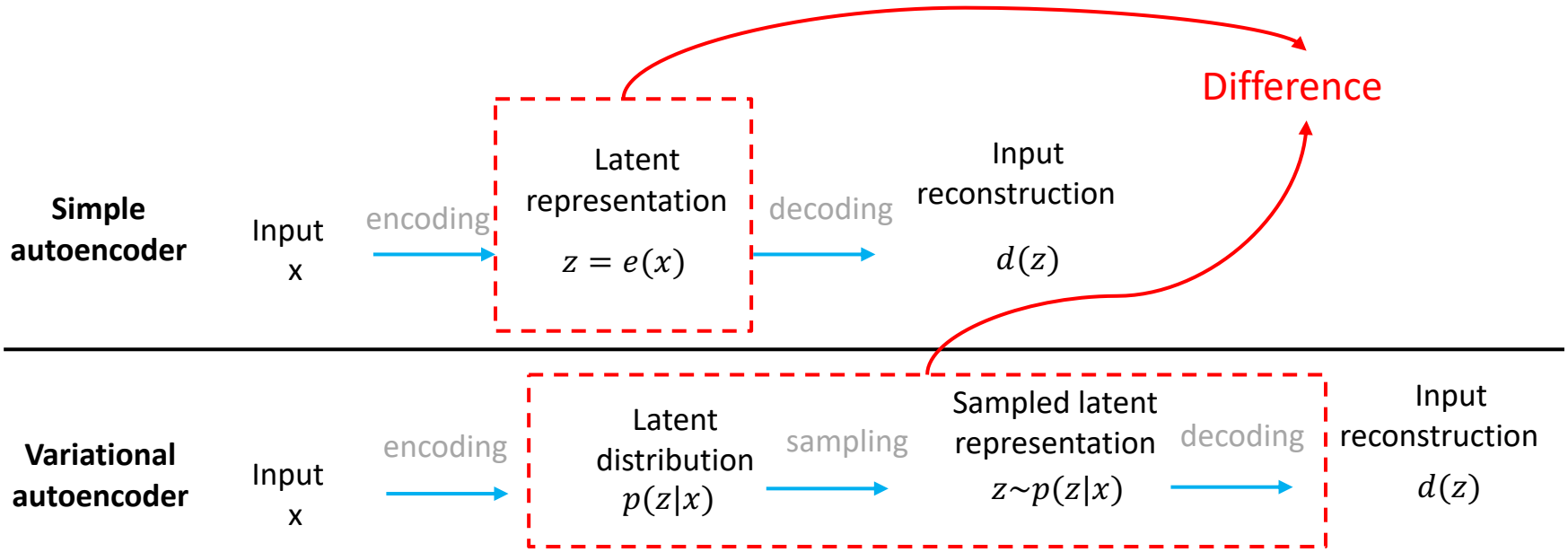
1

The recently proposed DARN method [GMW13], also learns a directed probabilistic model using an auto-encoding structure, however their method applies to binary latent variables. Even more recently, [RMW14] also make the connection between auto-encoders, directed proabilistic models and stochastic variational inference using the reparameterization trick we describe in this paper. Their work was developed independently of ours and provides an additional perspective on AEVB.

2

eter learning in large neural networks. Concurrently with this paper, Kingma & Welling (2014) present an alternative discussion of stochastic backpropagation. Our approaches were developed simultaneously and provide complementary perspectives on the use and derivation of stochastic backpropagation rules.

Autoencoder vs Variational Autoencoder



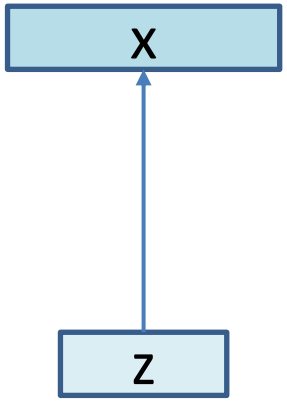
Difference: autoencoder (**deterministic**), variational autoencoder (**probabilistic**)

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoders

- Probabilistic spin on autoencoders - will let us sample from the model to generate data
- Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from underlying unobserved (latent) representation z

Sample from true conditional $p_{\theta^*}(x|z^{(i)})$



Sample from true prior $p_{\theta^*}(z)$

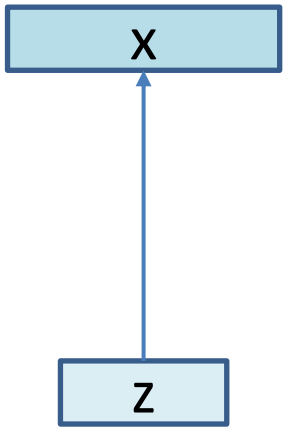
Intuition: x is an image, z is latent factors used to generate

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

- We want to estimate the true parameters θ^* of this generative model

Sample from true conditional $p_{\theta^*}(x|z^{(i)})$



Sample from true prior $p_{\theta^*}(z)$

- How should we represent this model?
  - Choose prior p(z) to be simple, e.g. Gaussian
  - Conditional p(x|z) is complex (generates image) => represent with neural network

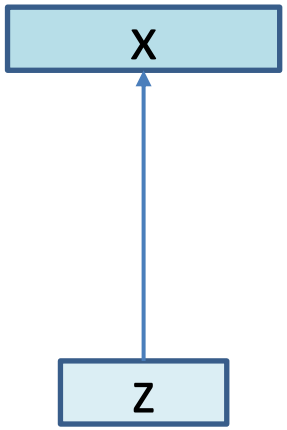
Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

- We want to estimate the true parameters θ^* of this generative model.

Sample from true conditional $p_{\theta^*}(x|z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$



- How to train the model?
  - Remember strategy for training generative models
  - Learn model parameters to maximize likelihood of training data

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

- Q: What is the problem with this?
  - Intractable!

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Intractability

- Data likelihood: $p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$

  - Intractable to compute p(x|z) for every z!
  - Can we sampling from the Posterior?

  Simple Gaussian prior

  Decoder neural network

- Posterior density also intractable: $p_\theta(z|x) = p_\theta(x|z)p_\theta(z)/p_\theta(x)$

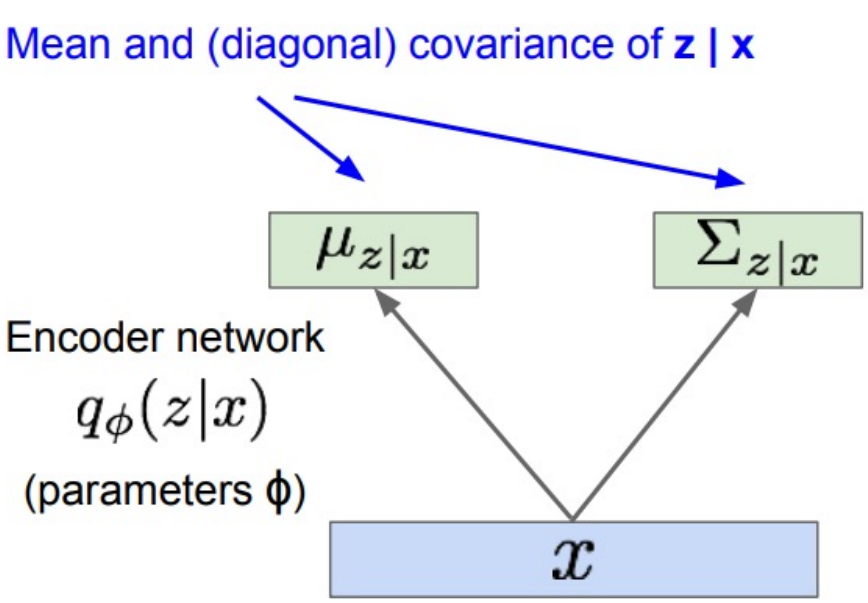
  Intractable data likelihood

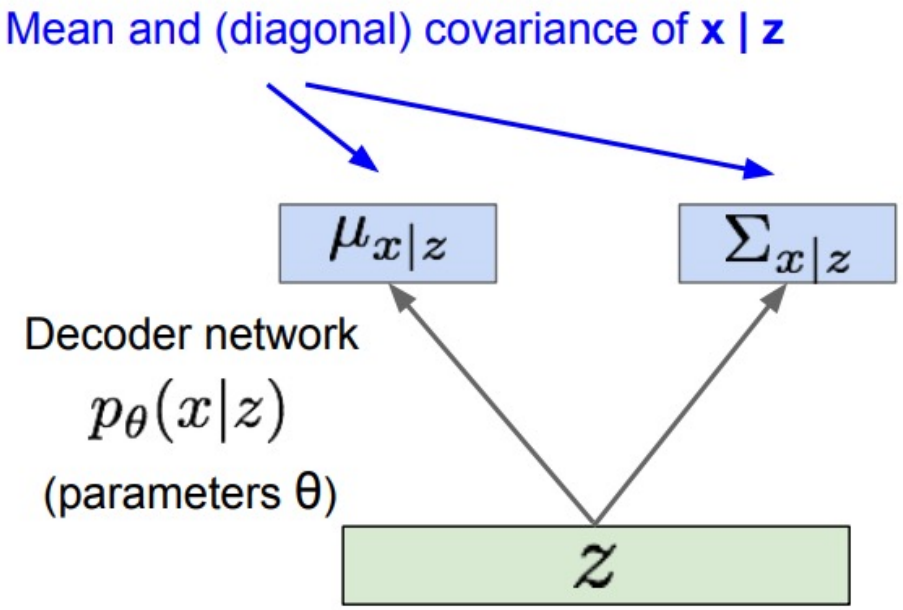
- Solution: in addition to decoder network modeling $p_\theta(x|z)$, define additional encoder network $q_\phi(z|x)$ that approximates $p_\theta(z|x)$

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Encoder + Decoder

- Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic

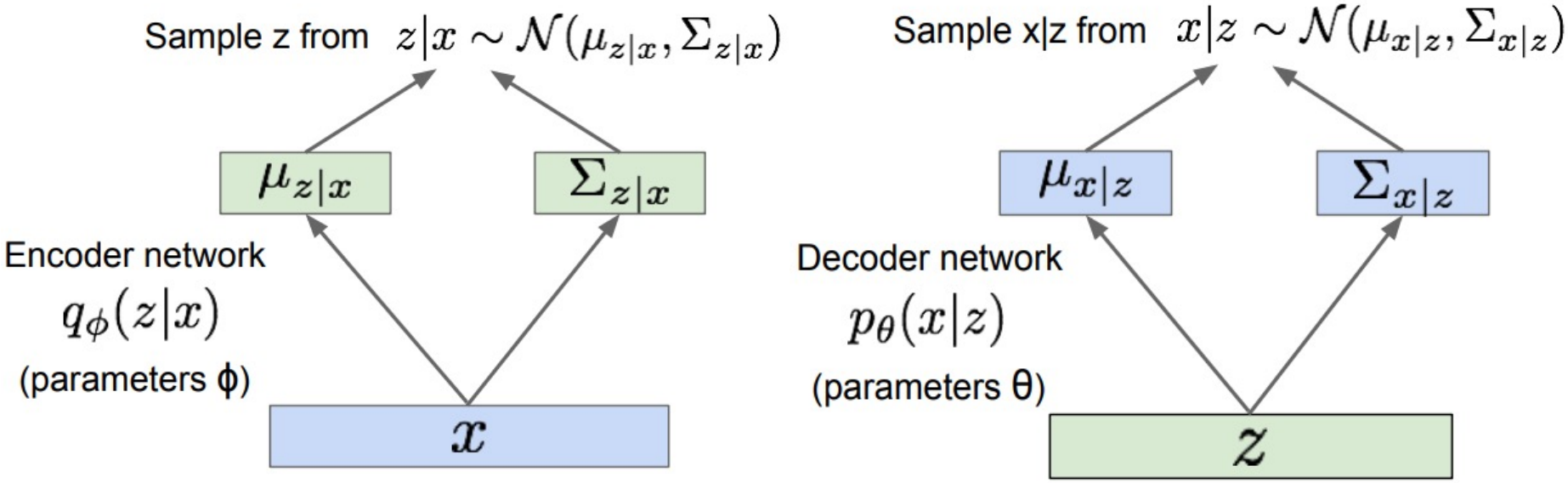




Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Encoder + Decoder

- Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic



Encoder and decoder networks also called "recognition"/"inference" and "generation" networks

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Lower Bound

- Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

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

Taking expectation wrt. z (using encoder network) will come in handy later

Variational Lower Bound

- Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

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

The expectation wrt. z (using encoder network) let us write nice KL terms

Variational Lower Bound

- Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

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

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

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

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling.

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :(But we know KL divergence always >= 0.

Variational Lower Bound

- Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

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

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}$$

Tractable lower bound which we can take gradient of and optimize! ($p_\theta(x|z)$ differentiable, KL term differentiable)

Variational Lower Bound

- Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

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

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{>0}$$

Reconstruct the input data

Make approximate posterior distribution close to prior

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

Variational lower bound ("ELBO")

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)},\theta,\phi)$$

Training: Maximize lower bound

Maximize Lower Bound

Putting it all together: maximizing the likelihood lower bound

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

Maximize Lower Bound

Putting it all together: maximizing the likelihood lower bound

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

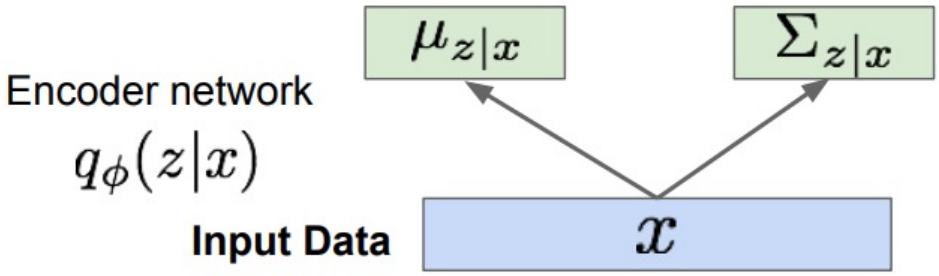
Let's look at computing the bound (forward pass) for a given minibatch of input data

Input Data x

Maximize Lower Bound

Putting it all together: maximizing the likelihood lower bound

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

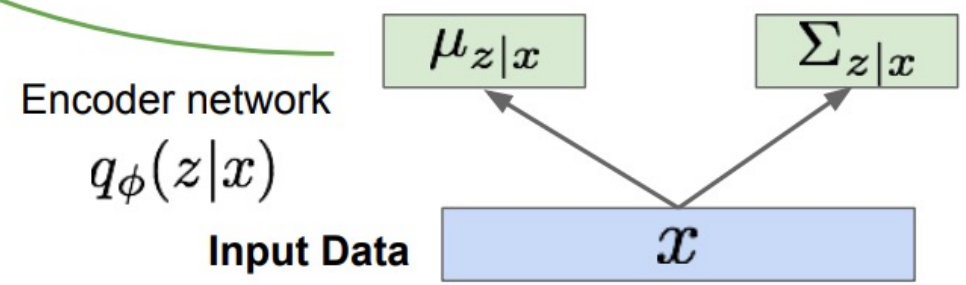


Maximize Lower Bound

Putting it all together: maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior

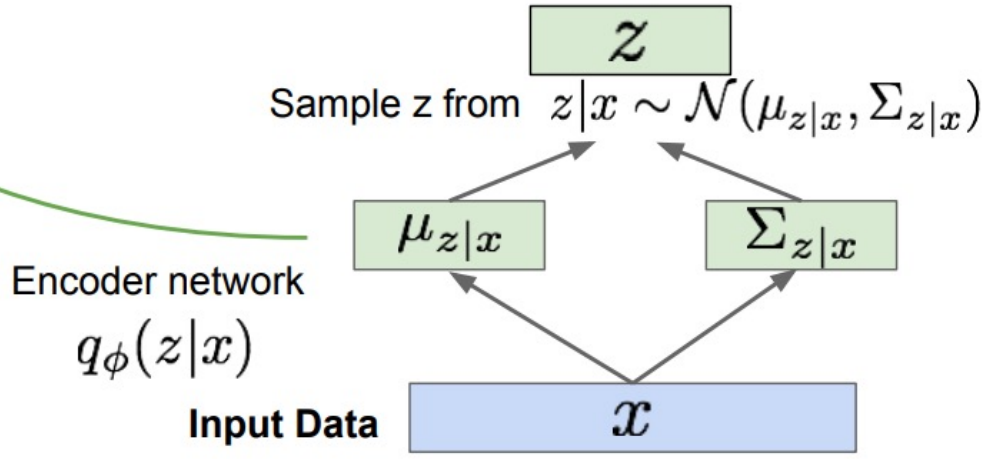


Maximize Lower Bound

Putting it all together: maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior

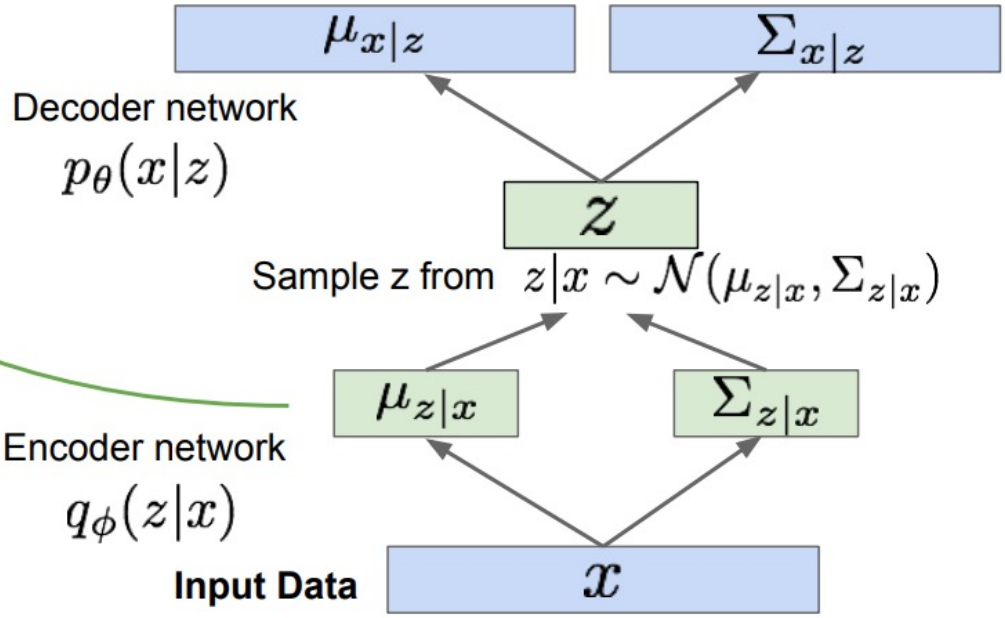


Maximize Lower Bound

Putting it all together: maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior



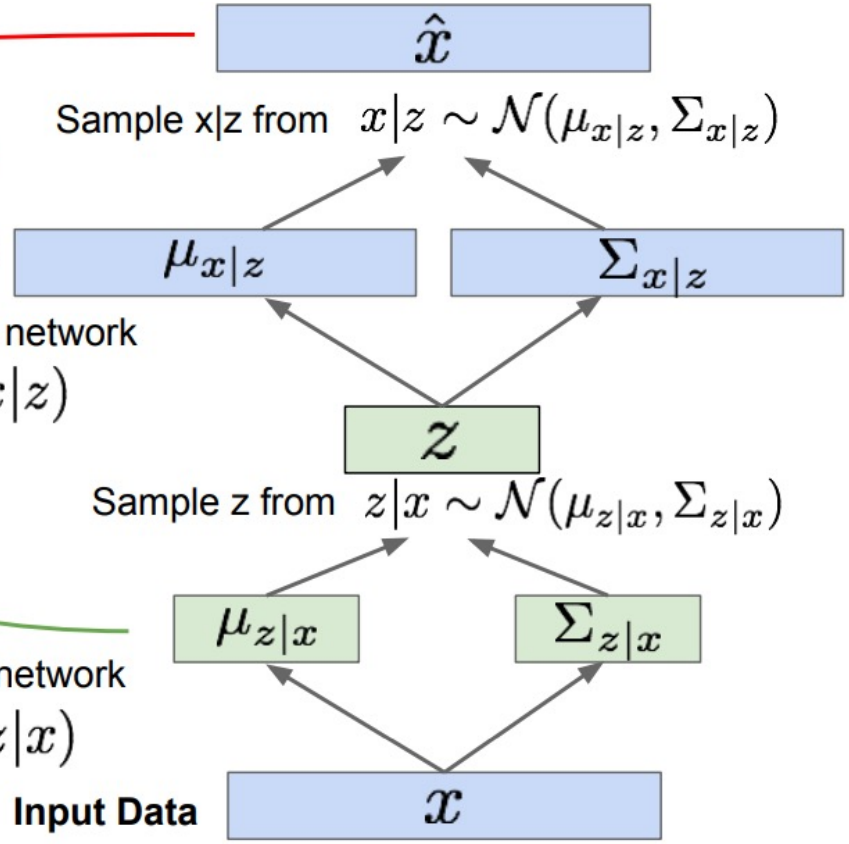
Maximize Lower Bound

Putting it all together: maximizing the likelihood lower bound

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

Maximize likelihood of original input being reconstructed

Make approximate posterior distribution close to prior



Maximize Lower Bound

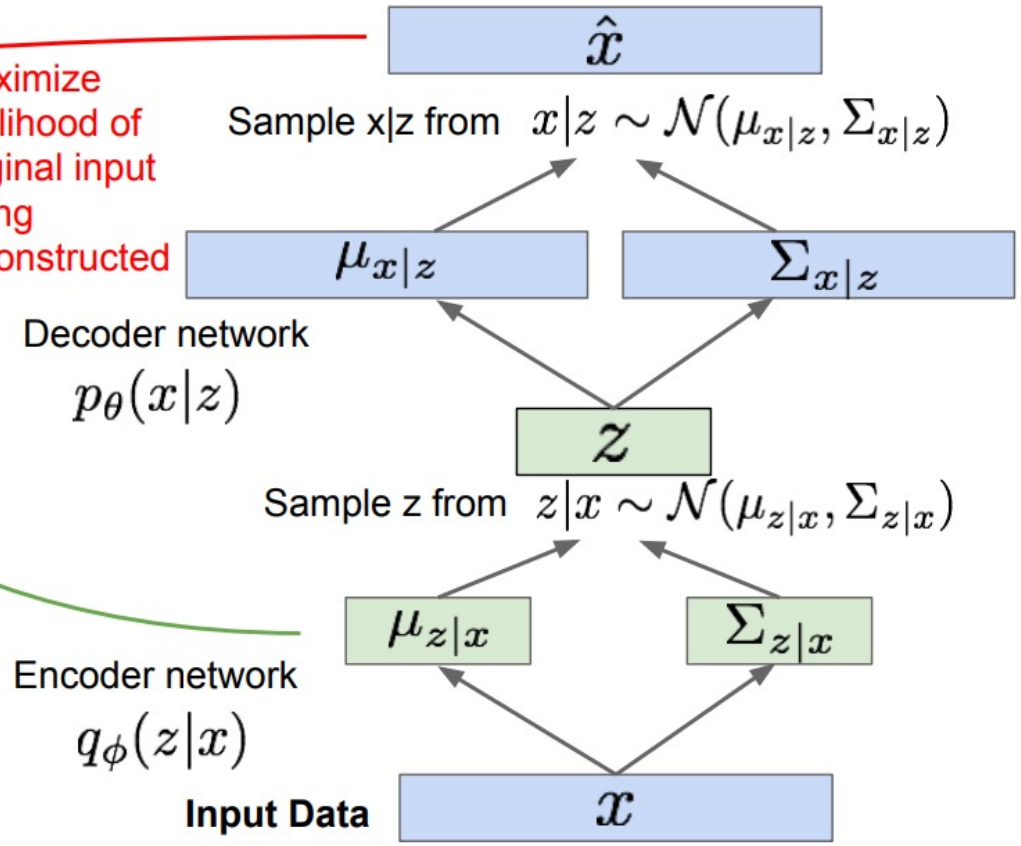
Putting it all together: maximizing the likelihood lower bound

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

Maximize likelihood of original input being reconstructed

Make approximate posterior distribution close to prior

For every minibatch of input data: compute this forward pass, and then backprop!

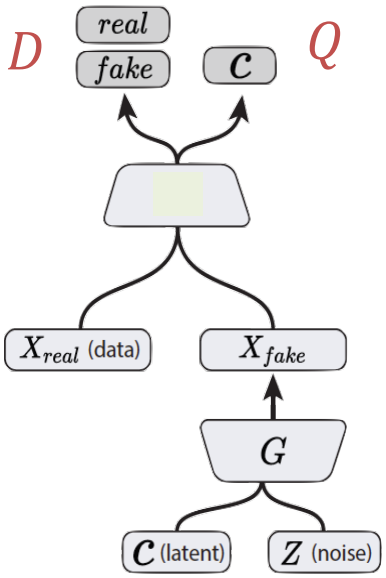


Recall: InfoGAN

Mutual Information's Variational Lower Bound

$$I\big(c; G(z,c)\big) = H(c) - H(c|G(z,c))$$
$$= \mathbb{E}_{x \sim G(z,c)}\Big[\mathbb{E}_{c' \sim P(c|x)}[\log P(c'|x)]\Big] + H(c)$$
$$= \mathbb{E}_{x \sim G(z,c)}\Big[D_{KL}(P||Q) + \mathbb{E}_{c' \sim P(c|x)}[\log Q(c'|x)]\Big] + H(c)$$
$$\geq \mathbb{E}_{x \sim G(z,c)}\Big[\mathbb{E}_{c' \sim P(c|x)}[\log Q(c'|x)]\Big] + H(c)$$

$$\geq \mathbb{E}_{c \sim P(c),\ x \sim G(z,c)}[\boldsymbol{\log Q(c|x)}] + H(c)$$

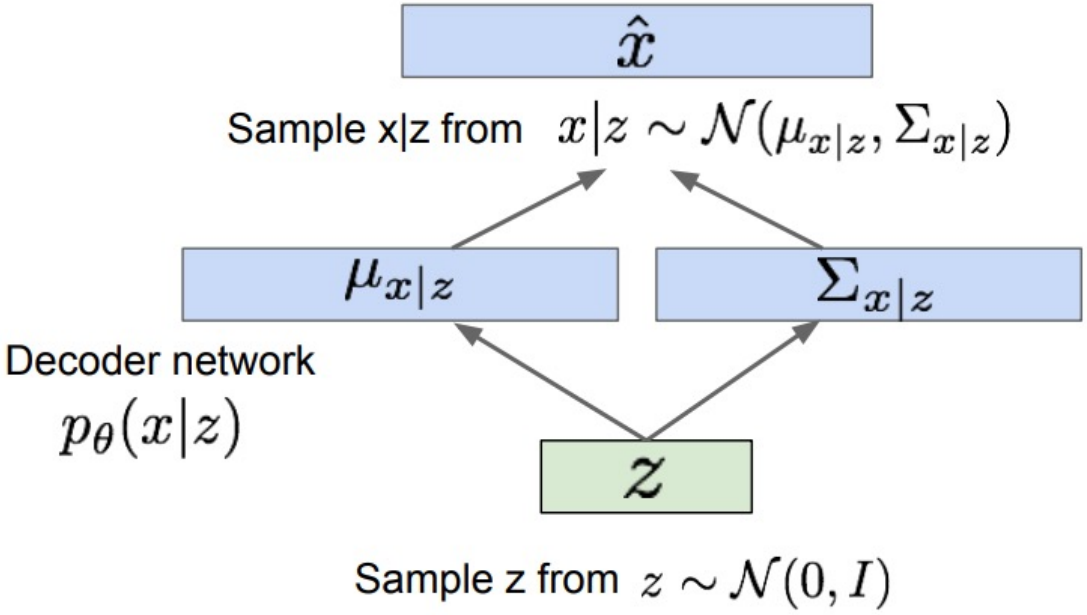


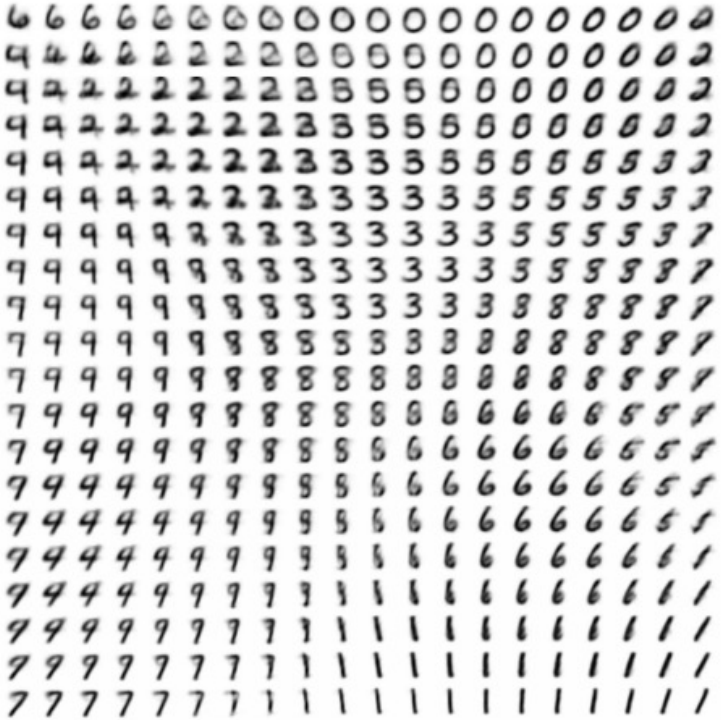
InfoGAN
(Chen, et al., 2016)

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. InfoGAN: Interpretable Representation Learning by Information Maximization Generative Adversarial Nets, NIPS (2016).

Generate Data

- Use decoder network, and sample z from prior
- Explicit formulation of data distribution, different from GAN





Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Generate Data

32x32 CIFAR-10

Labeled Faces in the Wild

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Summary

- Probabilistic spin to traditional autoencoders
- Generating data defines an intractable density => derive and optimize a (variational) lower bound

- Pros:
  - Principled approach to generative models
  - Allows inference of q(z|x), can be useful feature representation for other tasks
- Cons:
  - Maximizes lower bound of likelihood: okay, but not as good evaluation as PixelRNN/PixelCNN
  - Samples blurrier and lower quality compared to state-of-the-art (GANs)

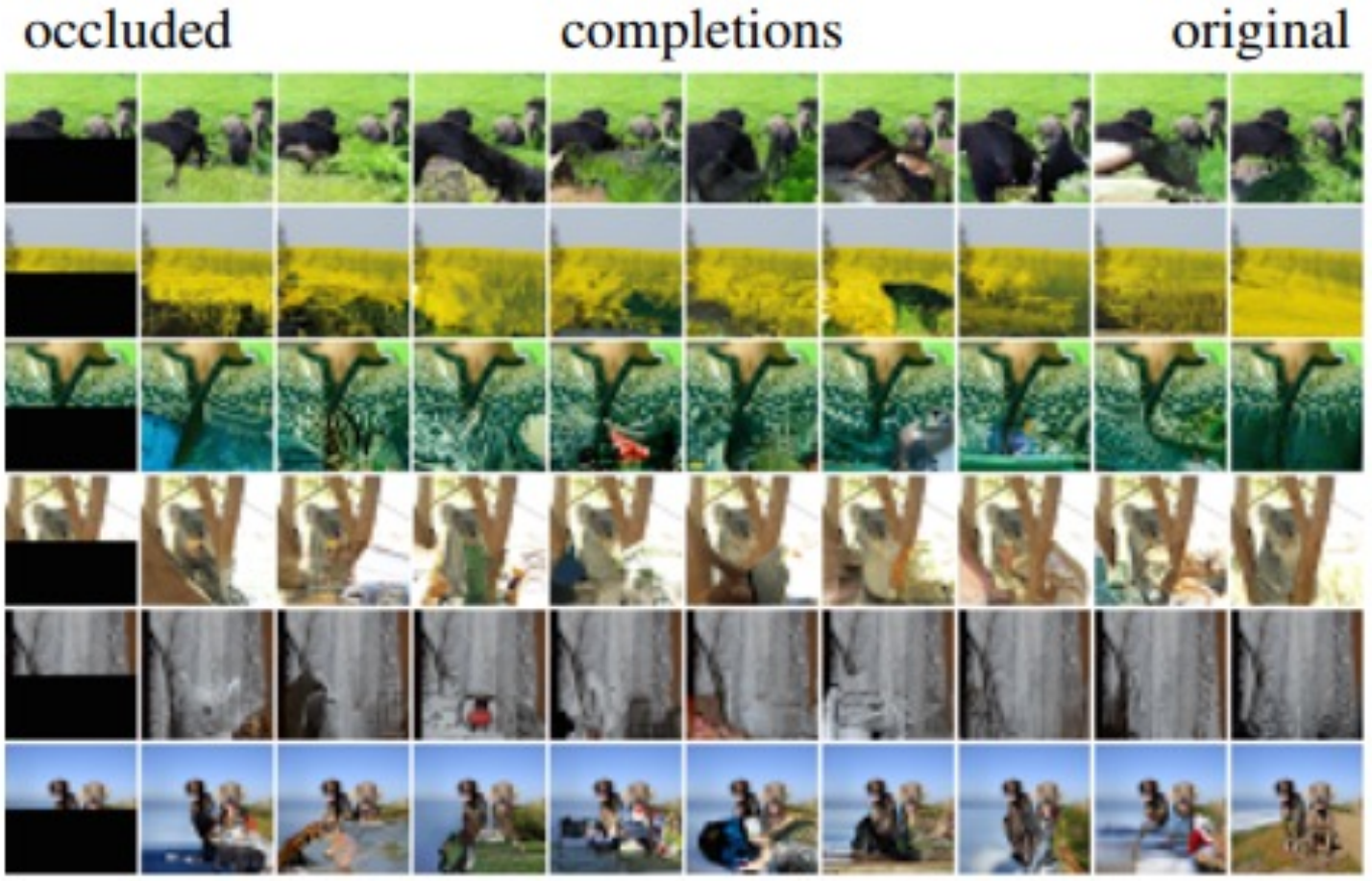
- Active areas of research:
  - More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian
  - Incorporating structure in latent variables

Outline

1. Course Review
2. Generative Adversarial Networks
3. Variational Autoencoder
4. Other Generation Methods

Pixel-by-pixel Generation

How to model statistical dependencies over pixels?



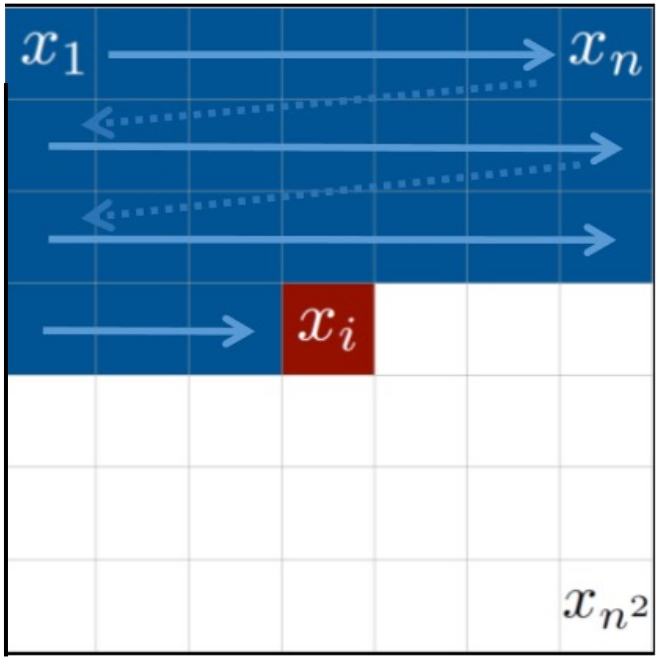
Intuition

$$p(\mathbf{x}) = p(x_1, x_2, ..., x_{n^2})$$

Bayes Theorem:

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

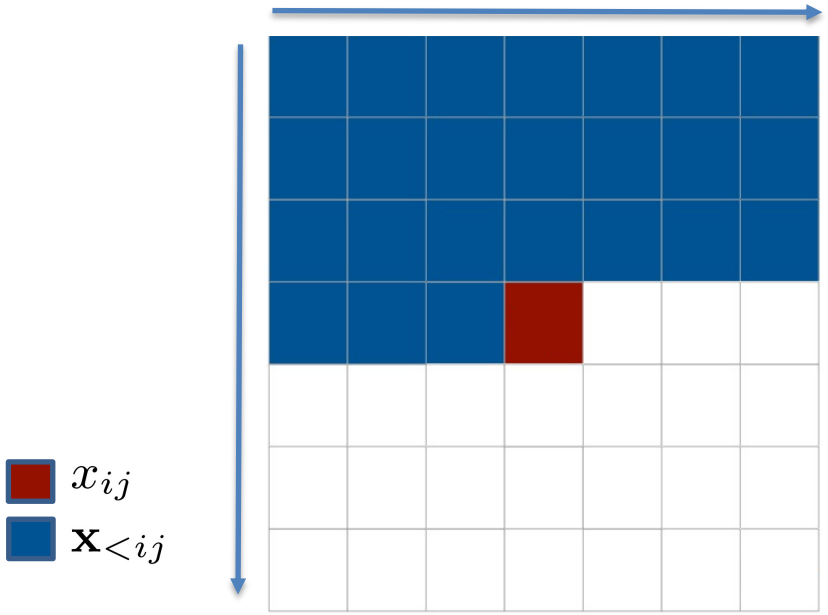
A sequential model!



Pixel recurrent neural networks, ICML 2016

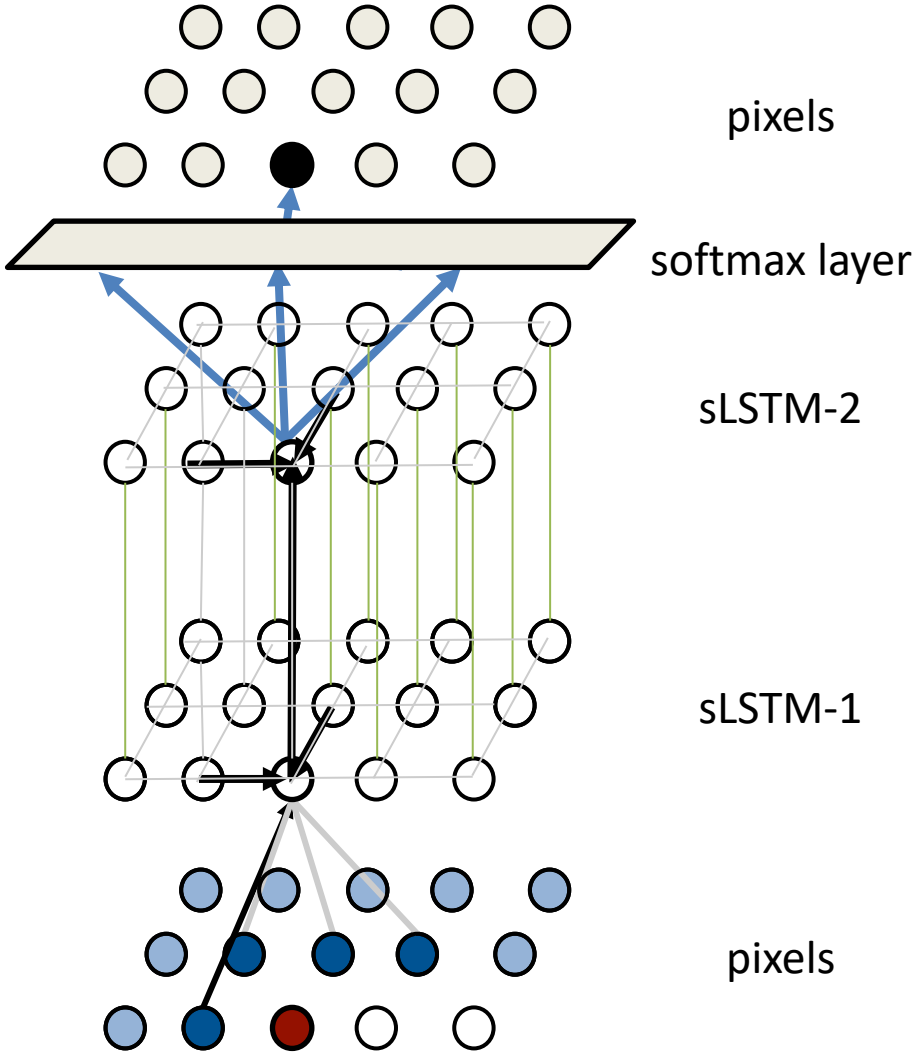
Intuition

- Question: can we use plain-LSTM to generate images pixels by pixels?
- Ensure information is well propagated in two dimensions
- spatial LSTM (sLSTM)



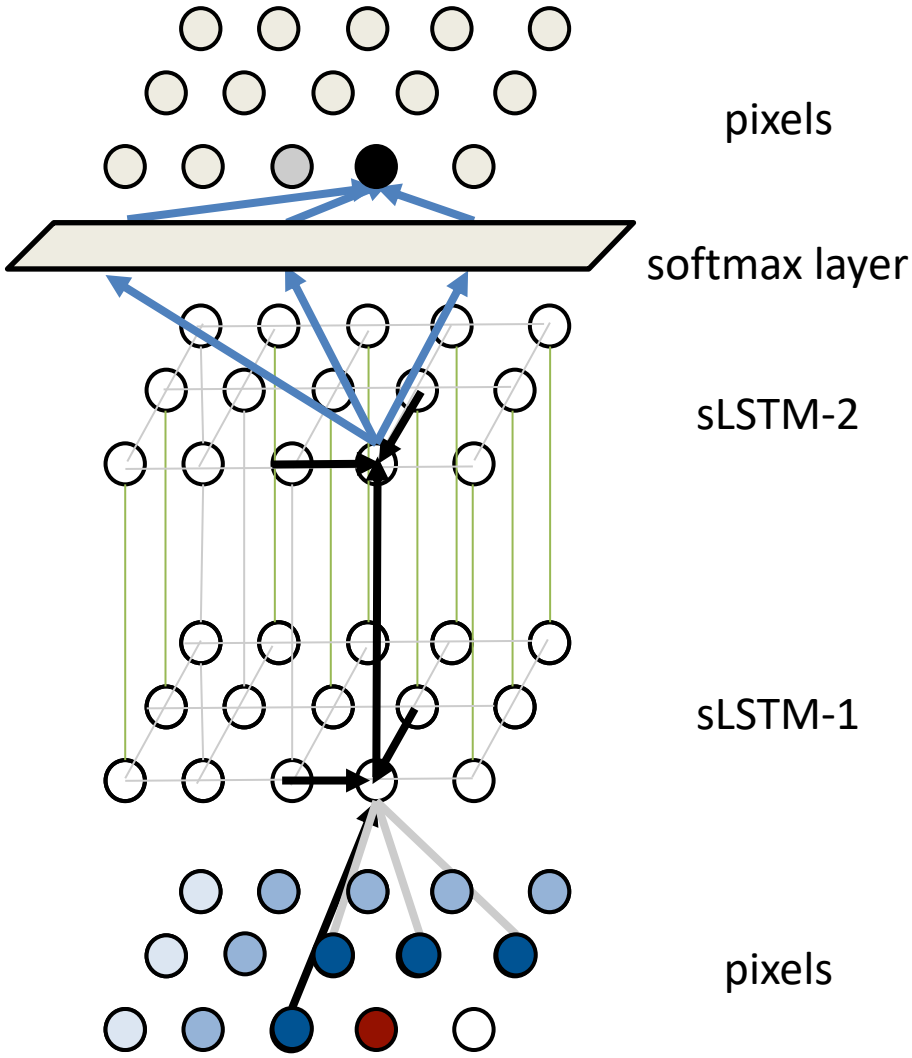
Pixel recurrent neural networks, ICML 2016

Spatial LSTM



Adapted from: Generative image modeling using spatial LSTM. Theis & Bethge, 2015

Spatial LSTM



Adapted from: Generative image modeling using spatial LSTM. Theis & Bethge, 2015

Details about Softmax Layer

- Treat pixels as discrete variables:
  - To estimate a pixel value, do classification (256 classes indicating pixel values 0-255)

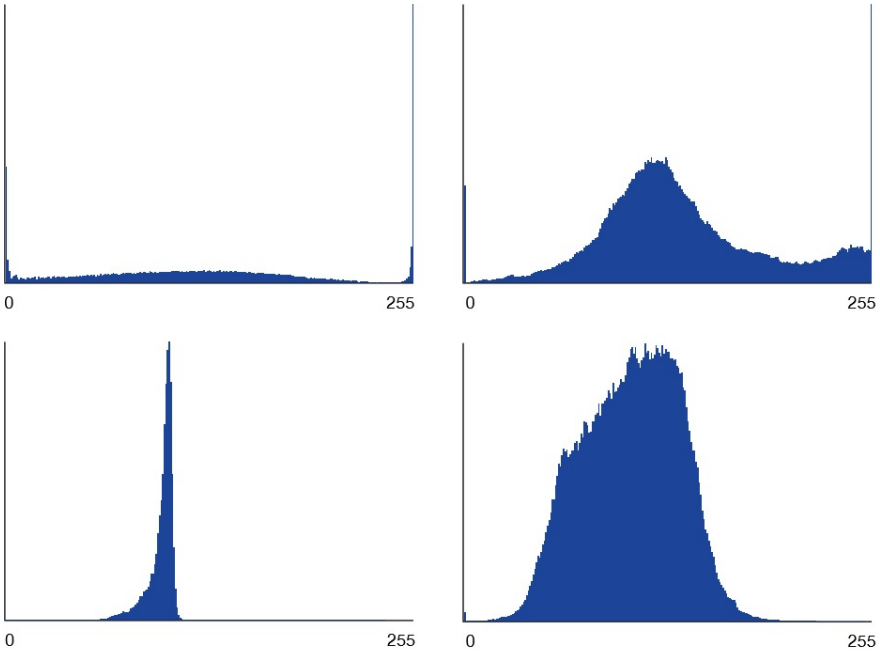
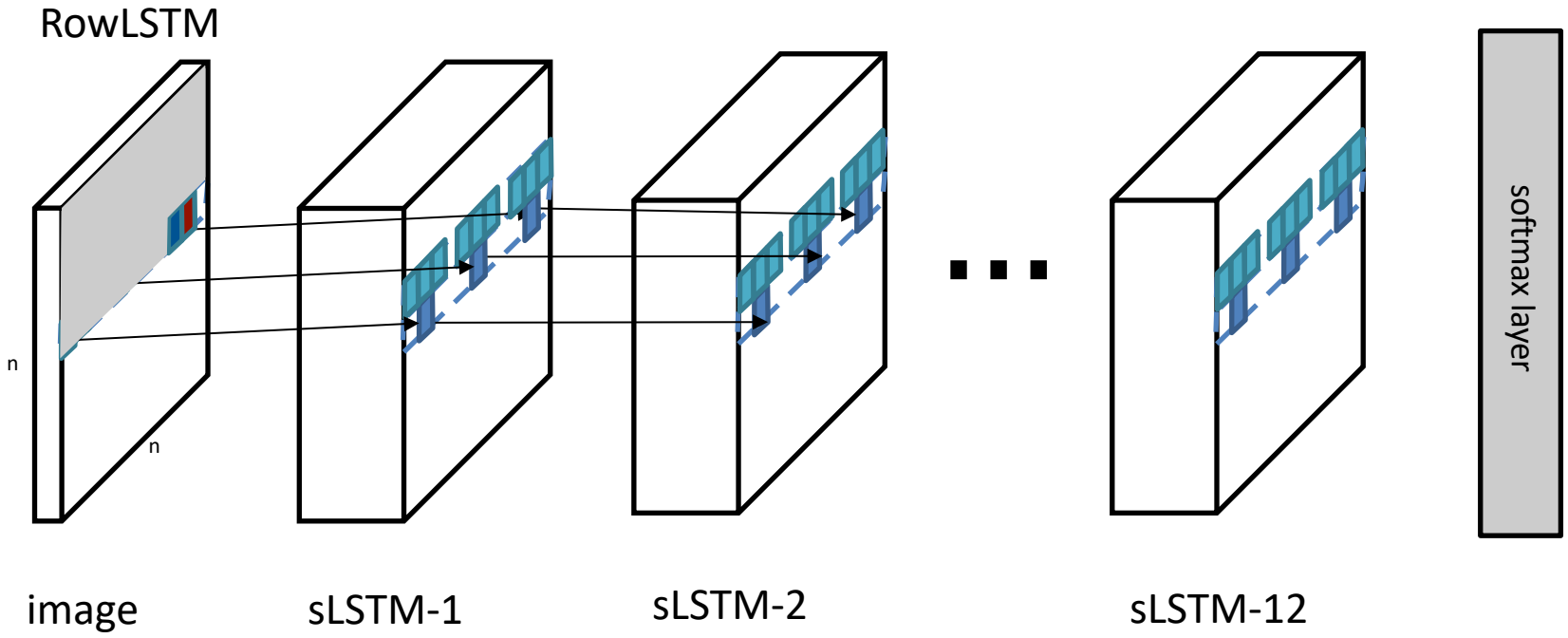


Figure: Example softmax outputs in the final layer, representing probability distribution over 256 classes.

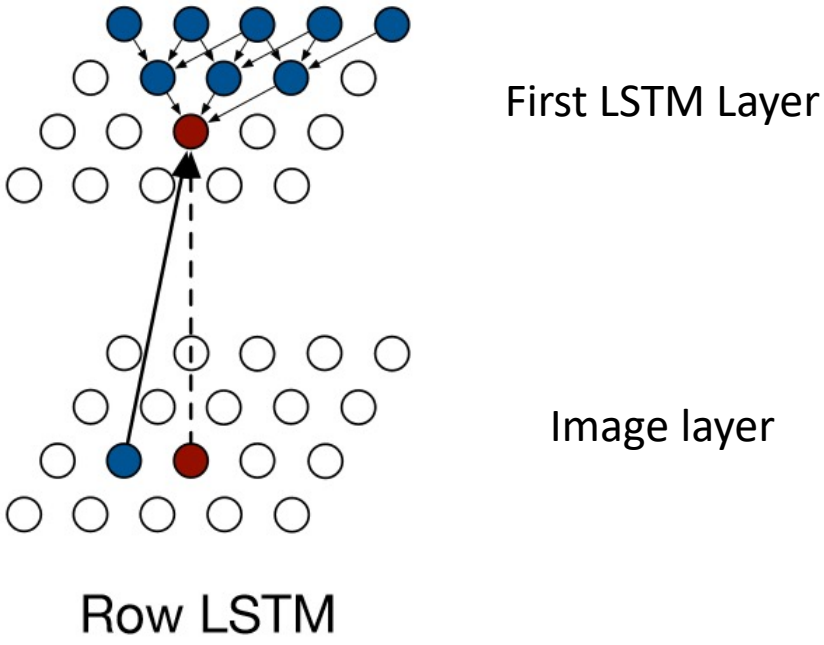
Figure from: Oord et al.

PixelRNN: A Specific Multidimensional LSTM



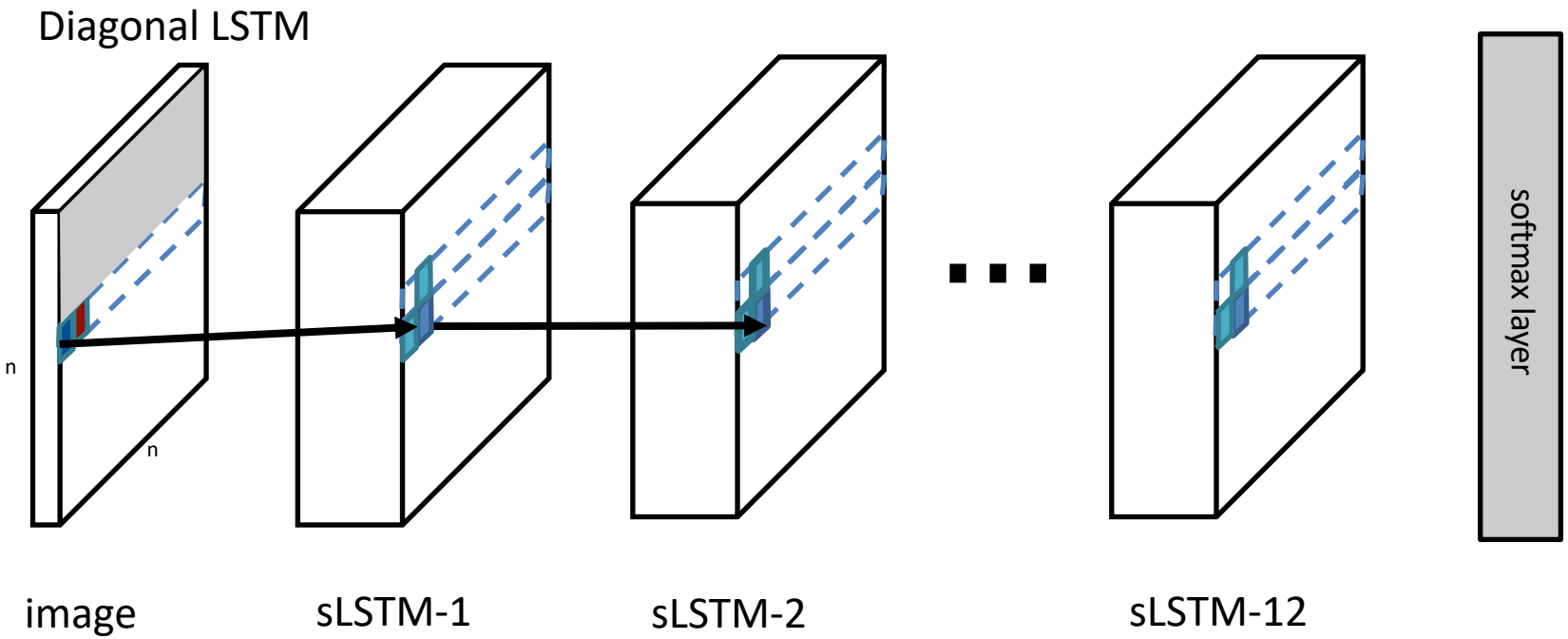
Pixel recurrent neural networks, ICML 2016

PixelRNN: A Specific Multidimensional LSTM



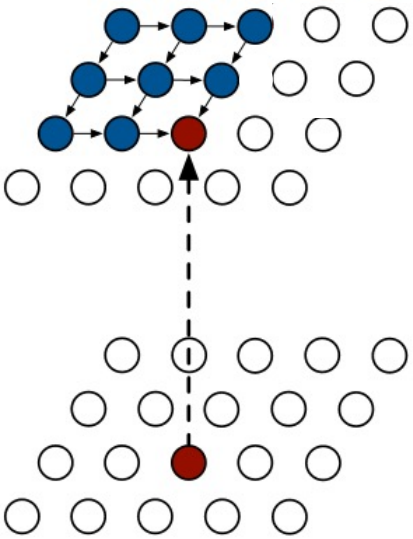
Pixel recurrent neural networks, ICML 2016

PixelRNN: A Specific Multidimensional LSTM



Pixel recurrent neural networks, ICML 2016

PixelRNN: A Specific Multidimensional LSTM

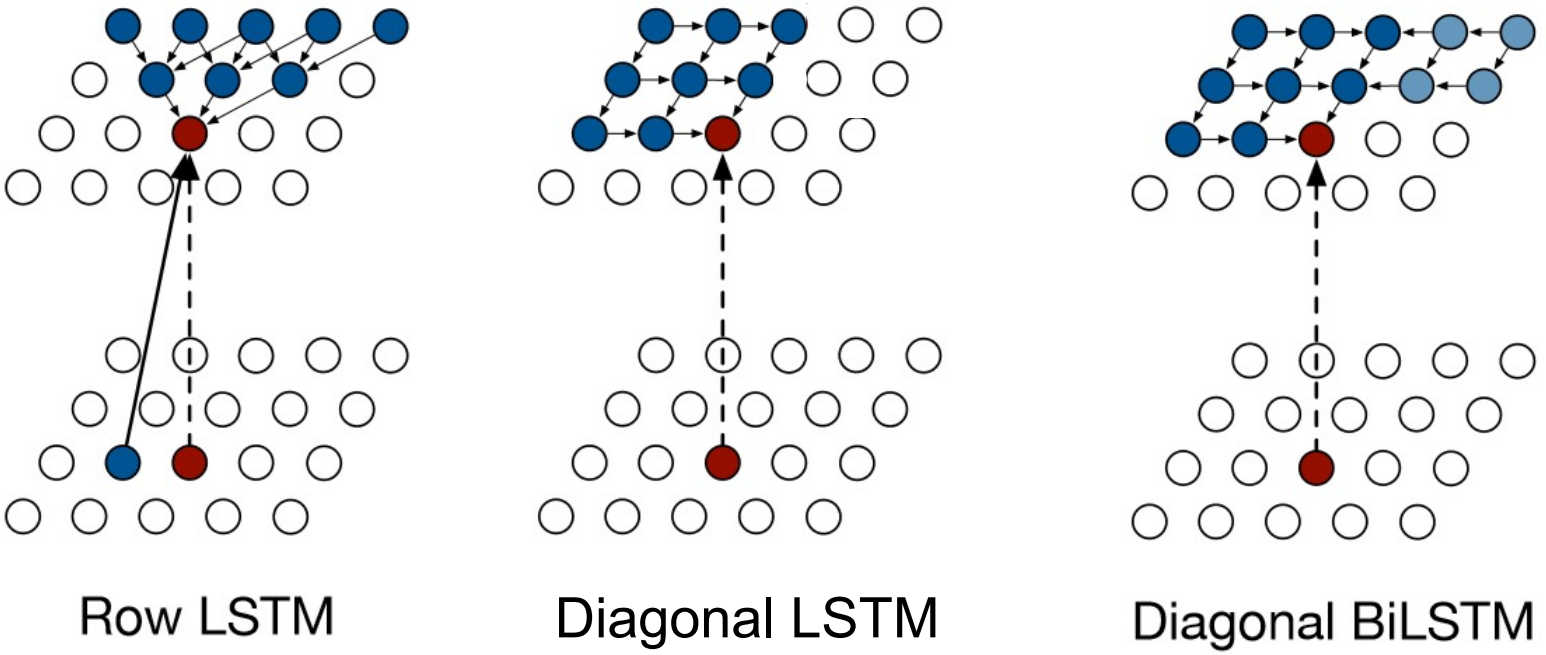


Diagonal LSTM

Pixel recurrent neural networks, ICML 2016

PixelRNN: A Specific Multidimensional LSTM

Receptive Field

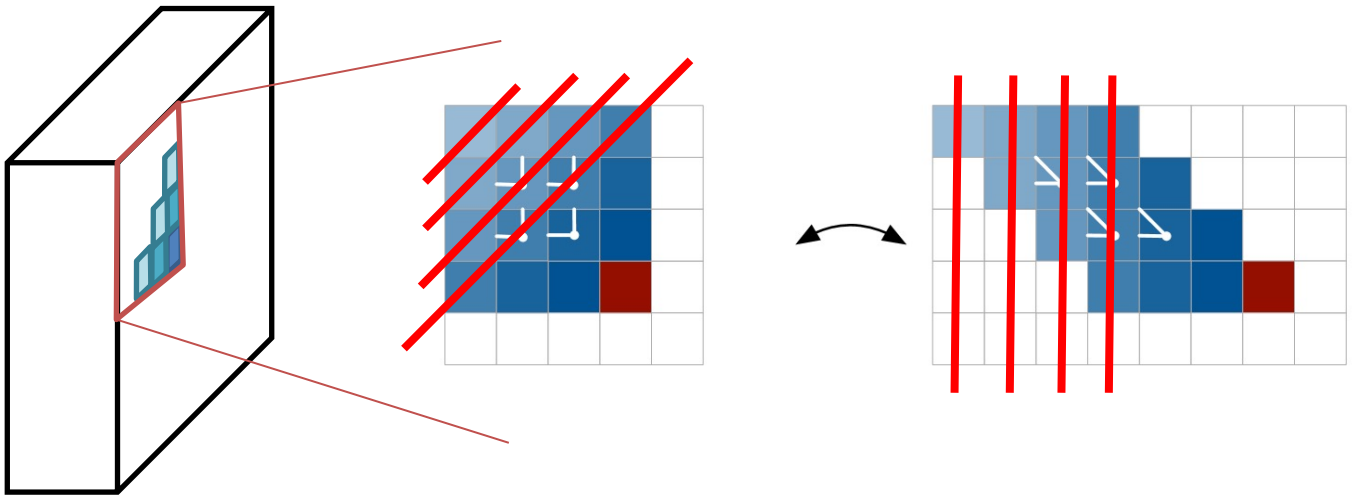


Pixel recurrent neural networks, ICML 2016

PixelRNN: A Specific Multidimensional LSTM

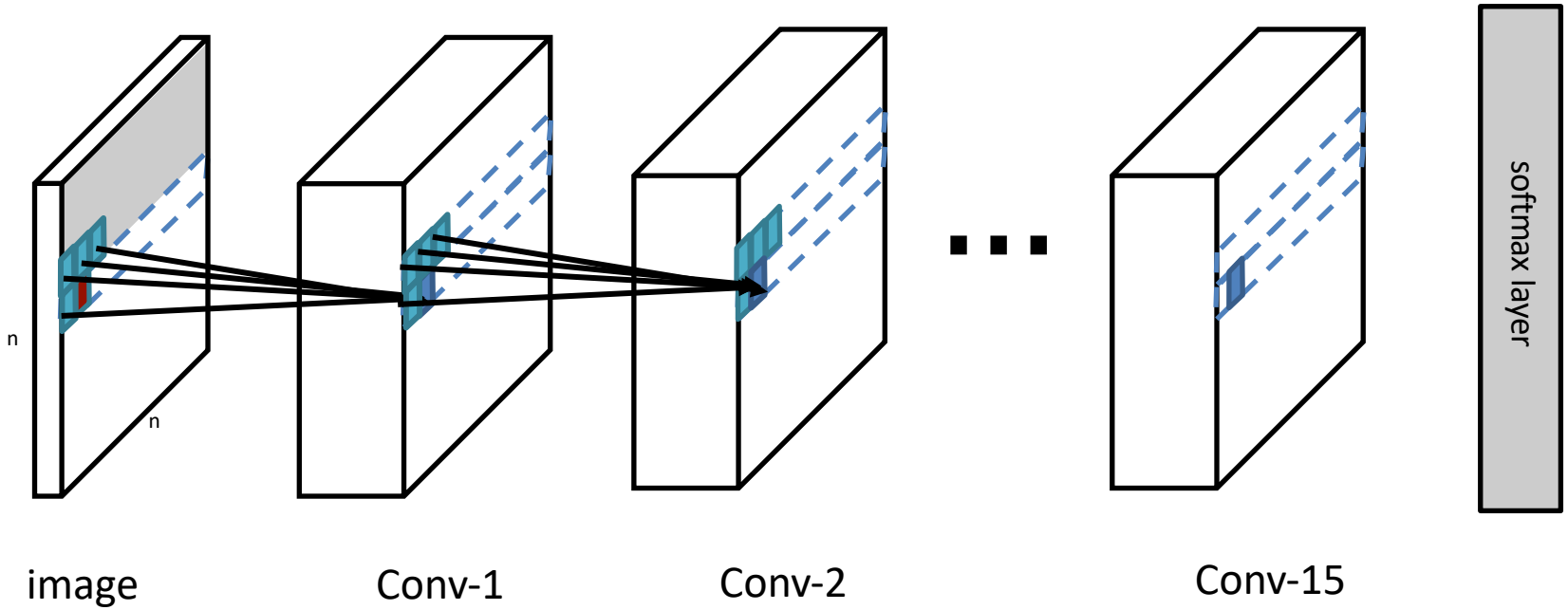
Diagonal LSTM

- To optimize, skew the feature maps so they can be parallelized



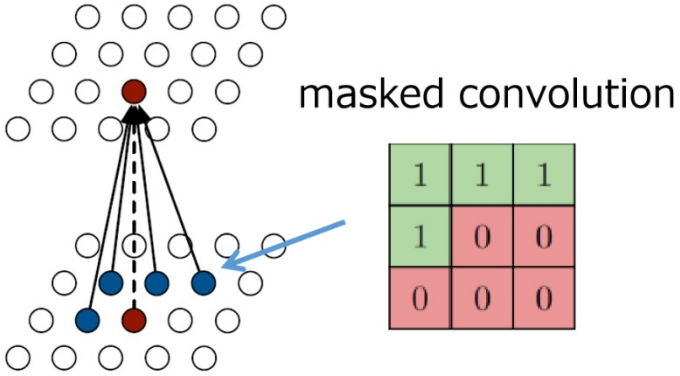
Pixel recurrent neural networks, ICML 2016

PixelCNN



PixelCNN

- 2D convolution on previous layer
- Apply masks so a pixel does not see future pixels (in sequential order)



Pixel recurrent neural networks, ICML 2016

Pixel-by-pixel Generation

Comparison

PixelCNN	PixelRNN – Row LSTM	PixelRNN – Diagonal BiLSTM
Full dependency field	Triangular receptive field	Full dependency field
Fastest	Slow	Slowest
Worst log-likelihood	-	Best log-likelihood

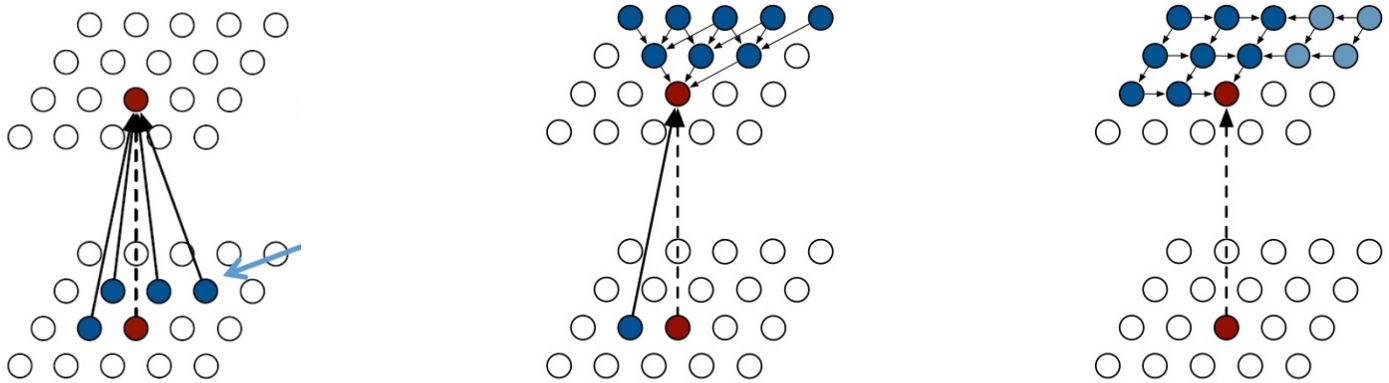


Figure from: Oord et al.

Results

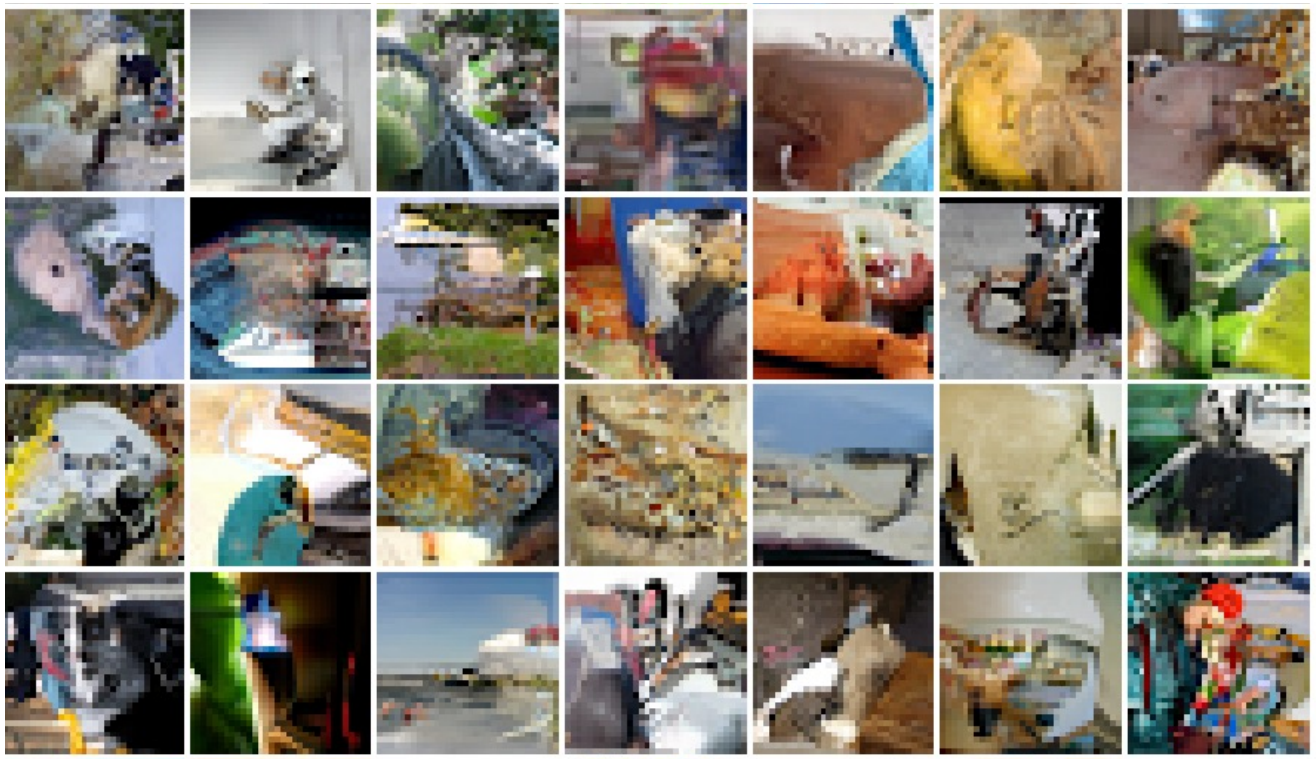


Figure: 32x32 ImageNet results from Diagonal BiLSTM model.

Figure from: Oord et al.

Multi-scale PixelRNN

- Take subsampled pixels as additional input pixels
- Can capture better global information (visually more coherent)
- Performance is similar to normal one

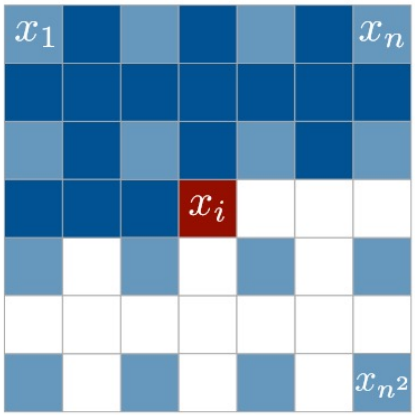


Figure from: Oord et al.

Multi-scale PixelRNN

Figure: 64x64 ImageNet results from normal Diagonal BiLSTM model (left) and multi-scale model (right).

Figure from: Oord et al.

Conditional Image Generation

- Given a high-level **image description vector $\boldsymbol{h}$**

$$p(\mathbf{x}) = p(x_1, x_2, ..., x_{n^2})$$

$$\downarrow$$

$$p(\mathbf{x}|\mathbf{h}) = p(x_1, x_2, ..., x_{n^2}|\mathbf{h})$$

Conditional image generation with pixelcnn decoders. NIPS 2016

Conditional Image Generation

- $\boldsymbol{h}$ is location-independent
  - For example, $p(\mathbf{x}|\mathbf{h}) = \prod_{i=1}^{n^2} p(x_i|x_1, ..., x_{i-1}, \mathbf{h})$ additional input
    - One-hot encoding representing a specific class
    - Latent representation embedding
  - Model joint probability conditioned on $\boldsymbol{h}$

$$\mathbf{y} = \tanh(W_{k,f} * \mathbf{x} + V_{k,f}^T \mathbf{h}) \odot \sigma(W_{k,g} * \mathbf{x} + V_{k,g}^T \mathbf{h})$$

dot product dot product

Conditional image generation with pixelcnn decoders. NIPS 2016

Conditional Image Generation

- $\boldsymbol{h}$ is location-dependent
  - $\boldsymbol{h}$ contains both object and location information
  - Use an additional deconvolutional neural network to estimate $\boldsymbol{s} = m(\boldsymbol{h})$, where $\boldsymbol{s}$ has same size as images

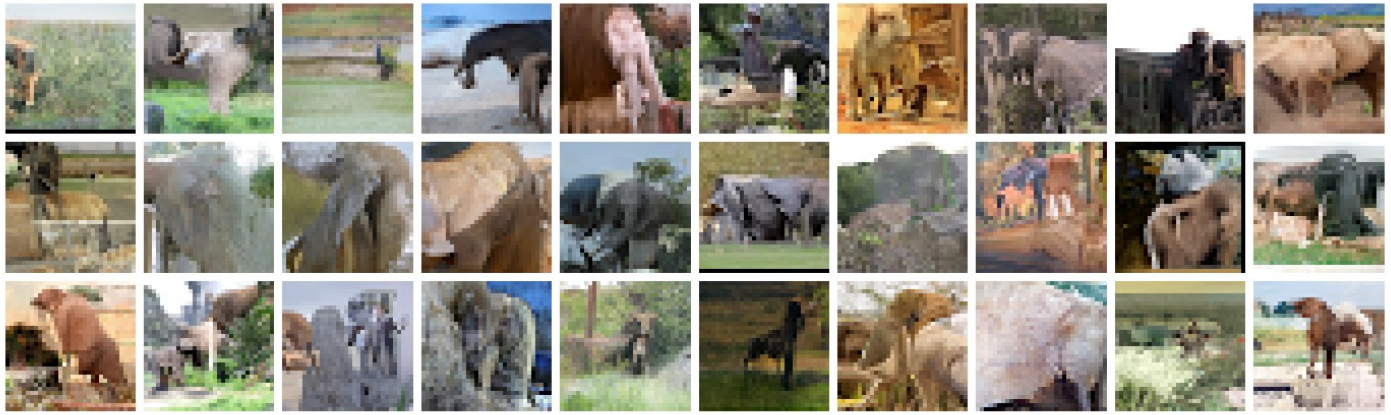
$$p(\mathbf{x}|\mathbf{h}) = \prod_{i=1}^{n^2} p(x_i|x_1, ..., x_{i-1}, \mathbf{h})$$

$$\mathbf{y} = \tanh(W_{k,f} * \mathbf{x} + V_{k,f} * \mathbf{s}) \odot \sigma(W_{k,g} * \mathbf{x} + V_{k,g} * \mathbf{s})$$

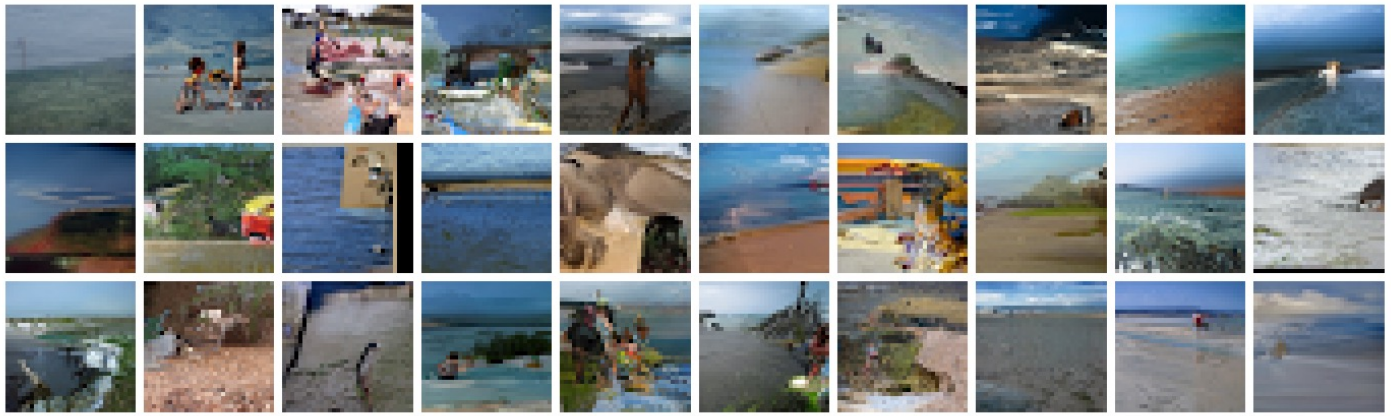
1x1 convolution 1x1 convolution

Conditional image generation with pixelcnn decoders. NIPS 2016

Results



African elephant



Sandbar

Figure from: Oord et al.

Other Recent Improvements

- Gated PixelCNN (Oord et al.)
  - Improve PixelCNN by removing blind spots and replacing ReLU units
- PixelCNN++ (Salimans et al.)
  - Improve PixelCNN by optimization techniques
- Video Pixel Networks (Kalchbrenner et al.)
  - Extend the work to 4 dimension

Comparison with GANs and VAEs

Autoregressive models (PixelRNNs, PixelCNNs)	GAN	VAE
• Simple and stable training (e.g. softmax loss)	• Sharpest images	• Easy to relate image with low-dimensional latent variables
• Inefficient during sampling • Don't easily provide simple low-dimensional codes for images	• Difficult to optimize due to unstable training dynamics	• Tends to have blurry outputs
		

Credit: https://openai.com/blog/generative-models/, Oord et al. and Larsen et al.

Summary

- Generative models can be **put together** in many different innovative ways to obtain interesting results for different applications

- In the future, these techniques are important
  - One-shot generation
  - Video generation
  - Fine-grained generation

Acknowledgement

Some of the materials in these slides are drawn inspiration from:

- Shubhendu Trivedi and Risi Kondor, University of Chicago, Deep Learning Course

- Hung-yi Lee, National Taiwan University, Machine Learning and having it Deep and Structured course

- Xiaogang Wang, The Chinese University of Hong Kong, Deep Learning Course

- Fei-Fei Li, Standord University, CS231n Convolutional Neural Networks for Visual Recognition course

Next time

- Reinforcement Learning

Questions?

Thank You !