



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

2018

## “Deep Learning Lecture”

# Lecture 9 : Deep Generative Model (2)

Wei Wang

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

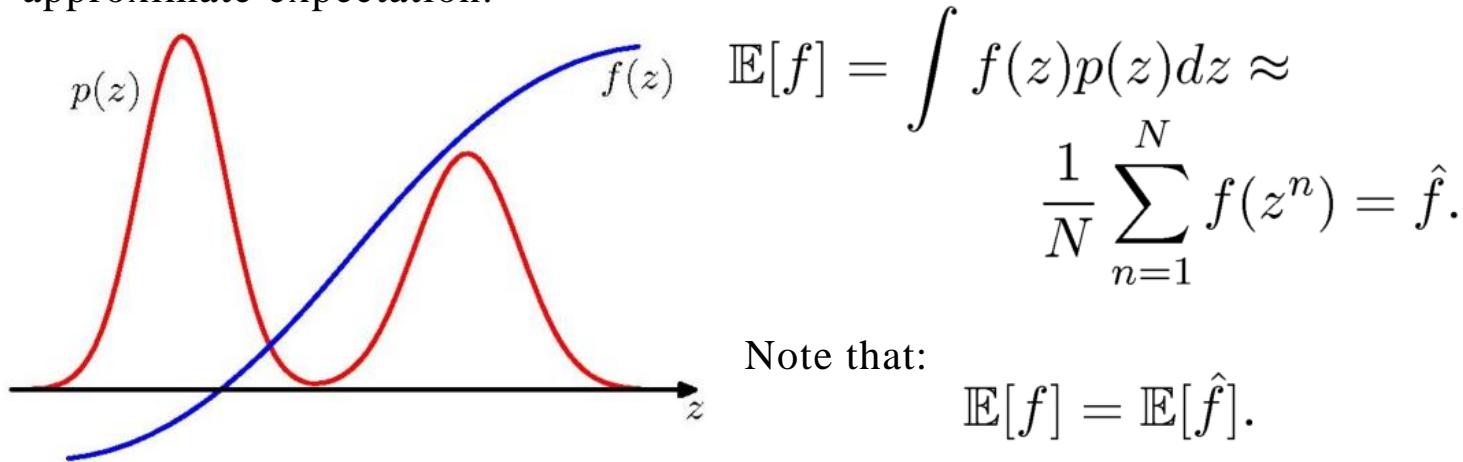
**2** Generative Adversarial Networks

**3** Variational Autoencoder

**4** Advanced Generation Methods

# Review: Simple Monte Carlo

- **General Idea:** Draw independent samples  $\{z_1, \dots, z_N\}$  from distribution  $p(z)$  to approximate expectation:



so the estimator has correct mean (**unbiased**).

- The **variance**:  $\text{var}[\hat{f}] = \frac{1}{N} \mathbb{E}[(f - \mathbb{E}[f])^2]$ .
- Variance decreases as  $1/N$ .
- **Remark:** The accuracy of the estimator **does not depend on dimensionality** of  $z$ .

# Review: Markov Chains

---

- We want to sample from the **target distribution** (e.g. posterior distribution, or a Markov Random Field):  
$$\pi(z) = \tilde{\pi}(z)/\mathcal{Z}.$$
- Obtaining independent samples is difficult.
  - Set up a Markov chain with transition kernel  $T(z' \leftarrow z)$  that **leaves our target distribution**  $\pi(z)$  invariant.
  - If the **chain is ergodic**, then the chain will converge to this unique equilibrium distribution.
  - We obtain **dependent samples drawn approximately** from  $\pi(z)$  by simulating a Markov chain for some time.
- **Ergodicity** requires: There exists  $K$ , such that for any starting  $z$ , we have

$$T^K(z' \leftarrow z) > 0 \text{ for all } \pi(z') > 0.$$

# Review: Variational Bound

- Given a joint distribution  $p(\mathbf{Z}, \mathbf{X} | \theta)$  over observed and latent variables governed by parameters  $\theta$ , the goal is to maximize the likelihood function  $p(\mathbf{X} | \theta)$  with respect to  $\theta$ :

$$p(\mathbf{X} | \theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \theta).$$

- We will assume that  $\mathbf{Z}$  is discrete, although derivations are identical if  $\mathbf{Z}$  contains continuous, or a combination of discrete and continuous variables.
- For any distribution  $q(\mathbf{Z})$  over latent variables we can derive the following variational lower bound:

$$\begin{aligned} \ln p(\mathbf{X} | \theta) &= \ln \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \theta) = \ln \sum_{\mathbf{Z}} q(\mathbf{Z}) \frac{p(\mathbf{X}, \mathbf{Z} | \theta)}{q(\mathbf{Z})} \\ \text{Jensen's inequality } \rightarrow &\geq \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \frac{p(\mathbf{X}, \mathbf{Z} | \theta)}{q(\mathbf{Z})} = \mathcal{L}(q, \theta). \end{aligned}$$

# Review: Factorized Gaussian

---

- Consider a problem of approximating a general distribution by a factorized distribution.
- To get some insight, let us look at the problem of **approximating a Gaussian distribution using a factorized Gaussian distribution**.
- Consider a Gaussian distribution over two correlated variables  $\mathbf{z} = (z_1, z_2)$ .

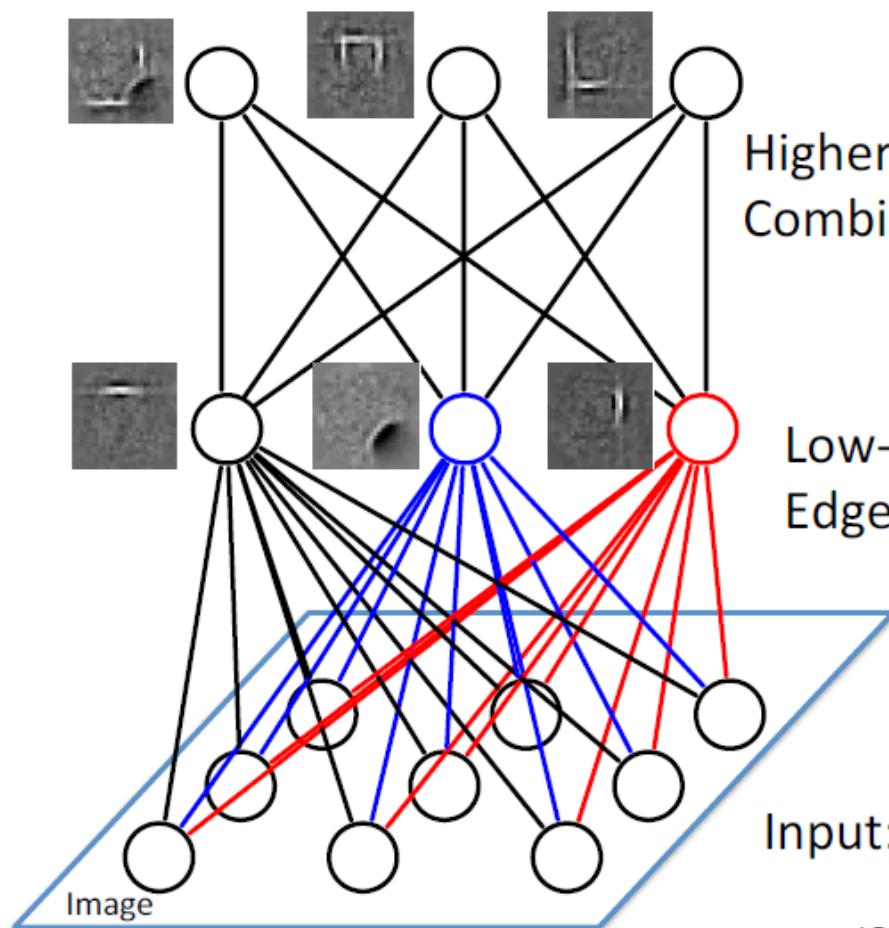
$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}),$$

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \quad \boldsymbol{\Lambda} = \begin{pmatrix} \beta_{11} & \beta_{12} \\ \beta_{12} & \beta_{22} \end{pmatrix}$$

- Let us approximate this distribution using a **factorized Gaussian** of the form:

$$q(\mathbf{z}) = q_1(z_1)q_2(z_2).$$

# Review: Deep Boltzmann Machines



Learn simpler representations,  
then compose more complex ones

Higher-level features:  
Combination of edges

Low-level features:  
Edges

Built from **unlabeled** inputs.

Input: Pixels

(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)

# Outline

---

1 Course Review

2 Generative Adversarial Networks

3 Variational Autoencoder

4 Advanced Generation Methods

# Generative Adversarial Networks

---

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

# Why Generative Models?

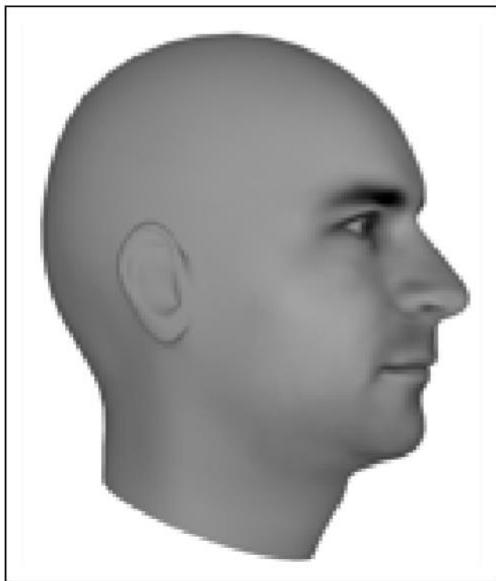
---

- **We've only seen discriminative models so far**
  - Given an image  $\mathbf{X}$ , predict a label  $\mathbf{Y}$
  - Estimates  $\mathbf{P}(\mathbf{Y}|\mathbf{X})$
- **Discriminative models have several key limitations**
  - Can't model  $\mathbf{P}(\mathbf{X})$ , i.e. the probability of seeing a certain image
  - Thus, can't sample from  $\mathbf{P}(\mathbf{X})$ , i.e. **can't generate new images**
- **Generative models (in general) cope with all of above**
  - Can model  $\mathbf{P}(\mathbf{X})$
  - Can generate new images

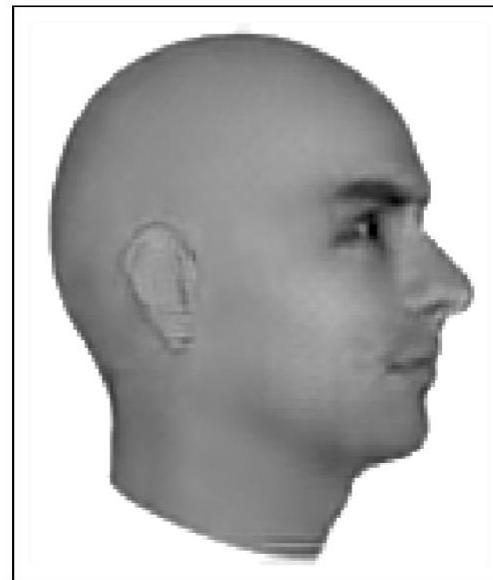
# Magic of GANs

---

Ground Truth



Adversarial



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

---

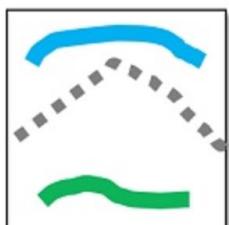
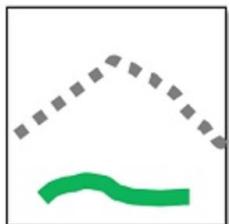
Which one is Computer generated?



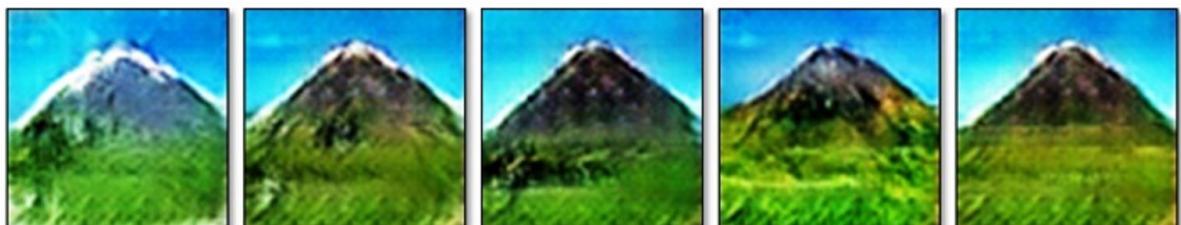
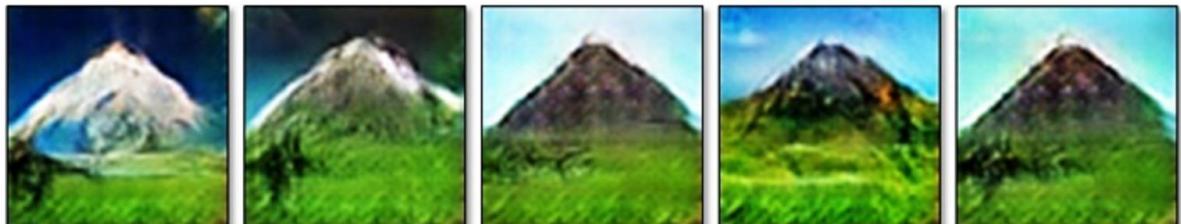
Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." *arXiv preprint arXiv:1609.04802* (2016).

# Magic of GANs

User edits



Generated images



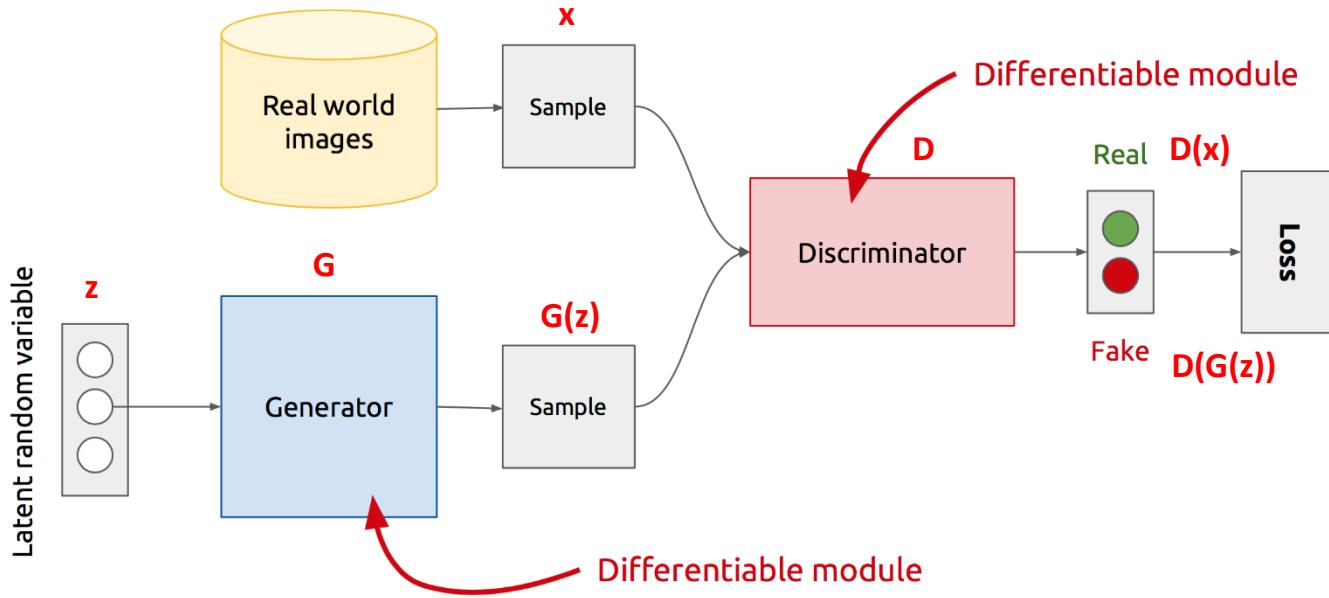
<http://people.eecs.berkeley.edu/~junyanz/projects/gvm/>

# Adversarial Training

---

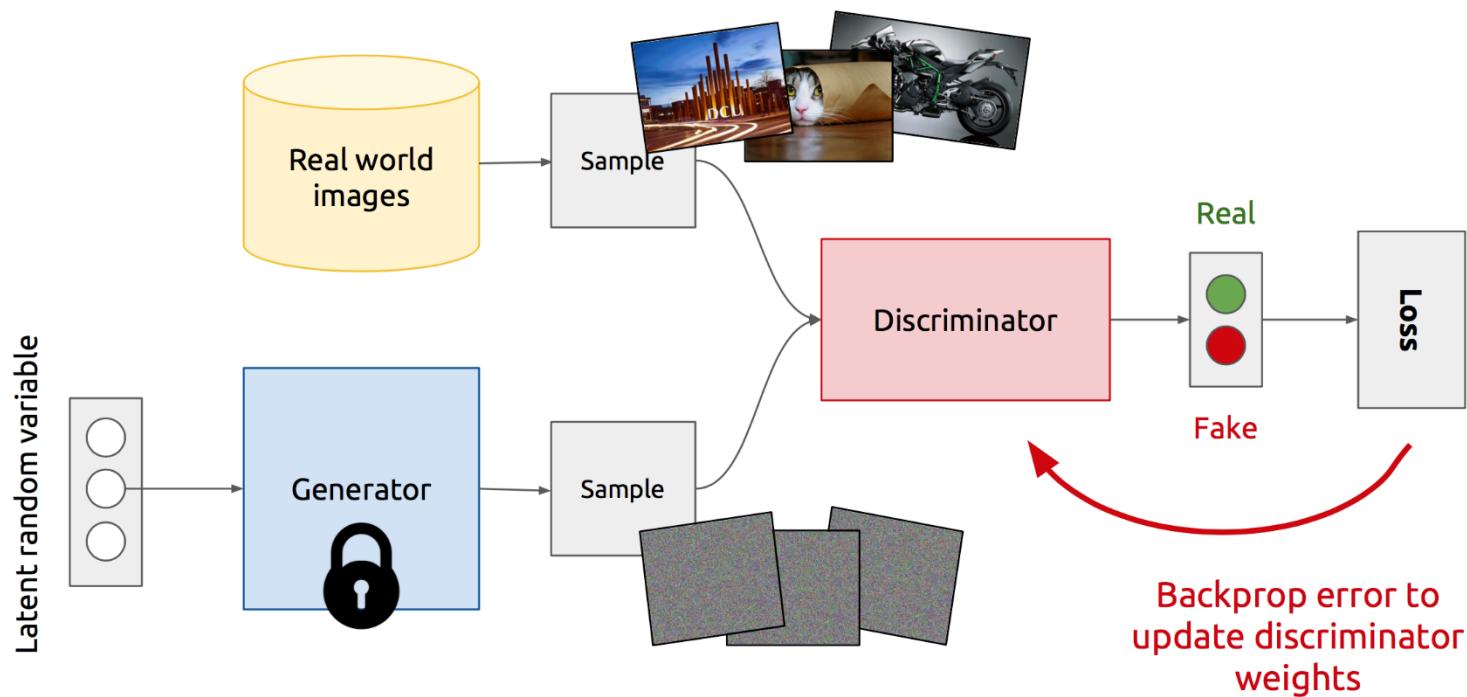
- **Adversarial Samples:**
  - We can generate adversarial samples to fool a discriminative model
  - We can use those adversarial samples to make models robust
  - We then require more effort to generate adversarial samples
  - Repeat this and we get better discriminative model
- **GANs extend that idea to generative models:**
  - Generator: generate fake samples, tries to fool the Discriminator
  - Discriminator: tries to distinguish between real and fake samples
  - Train them against each other
  - Repeat this and we get better Generator and Discriminator

# GAN's Architecture

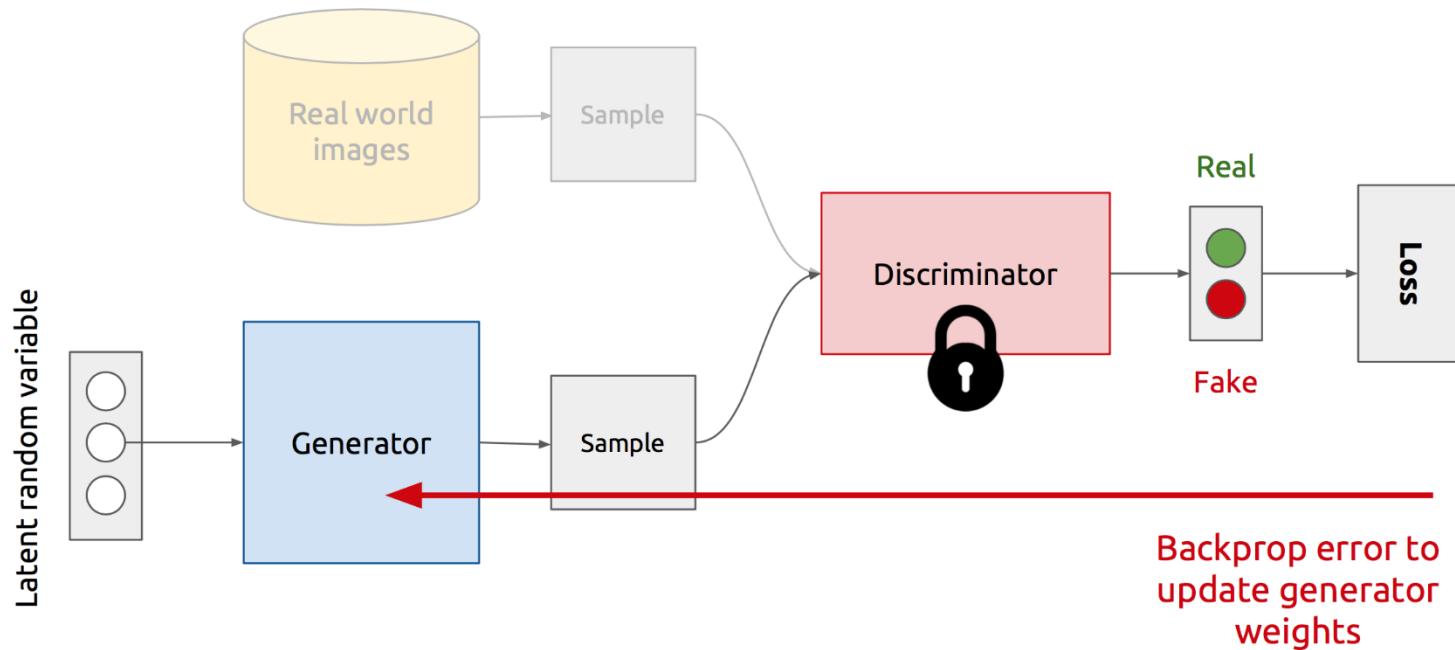


- $Z$  is some random noise (Gaussian/Uniform).
- $Z$  can be thought as the latent representation of the image.

# Training Discriminator



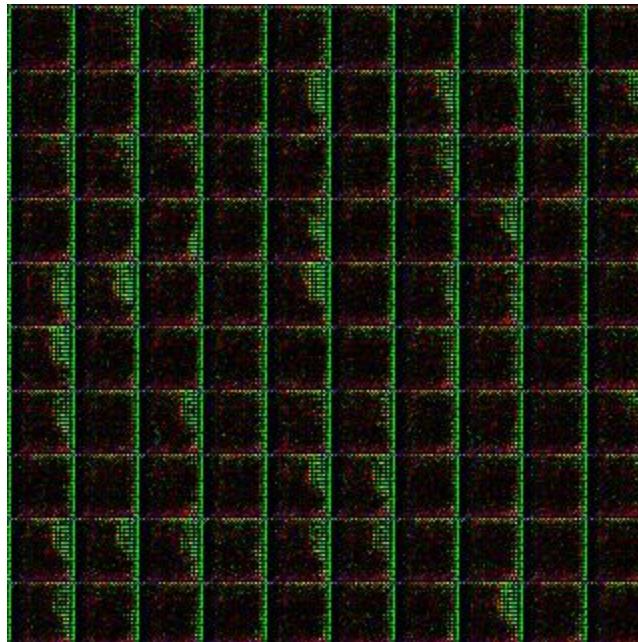
# Training Generator



<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 its reward  $V(D, G)$
  - The Generator is trying to minimize Discriminator's reward (or maximize its loss)

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

- 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.

---

Discriminator  
updates

```
for number of training iterations do
    for  $k$  steps do
        • Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ .
        • Sample minibatch of  $m$  examples  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  from data generating distribution  $p_{\text{data}}(\mathbf{x})$ .
        • Update the discriminator by ascending its stochastic gradient:
```

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

```
end for
```

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

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

```
end for
```

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

---

Generator  
updates

# Vanishing gradient strikes back again

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

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

- $\nabla_a \log(1 - \sigma(a)) = \frac{-\nabla_a \sigma(a)}{1 - \sigma(a)} = \frac{-\sigma(a)(1 - \sigma(a))}{1 - \sigma(a)} = -\sigma(a) = -D(G(z))$
- Gradient goes to 0 if  $D$  is confident, i.e.  $D(G(z)) \rightarrow 0$
- Minimize  $-\mathbb{E}_{z \sim q(z)}[\log D(G(z))]$  for **Generator** instead (keep Discriminator as it is)

# Applications

---

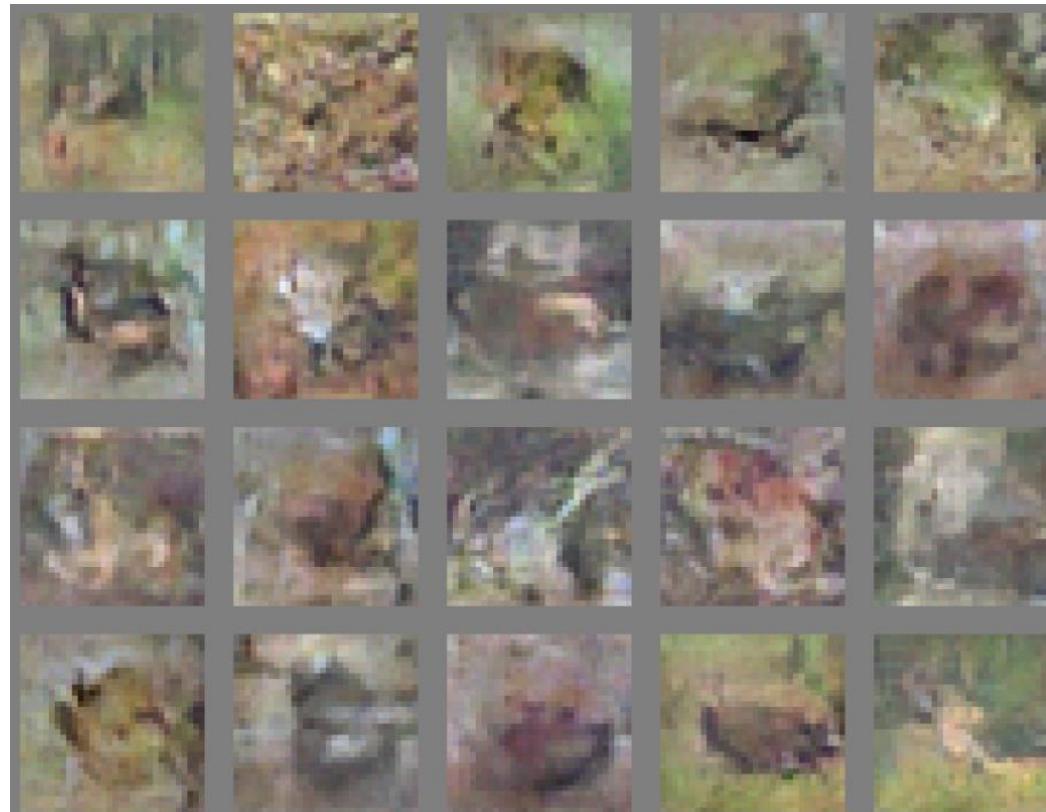
## Faces



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.

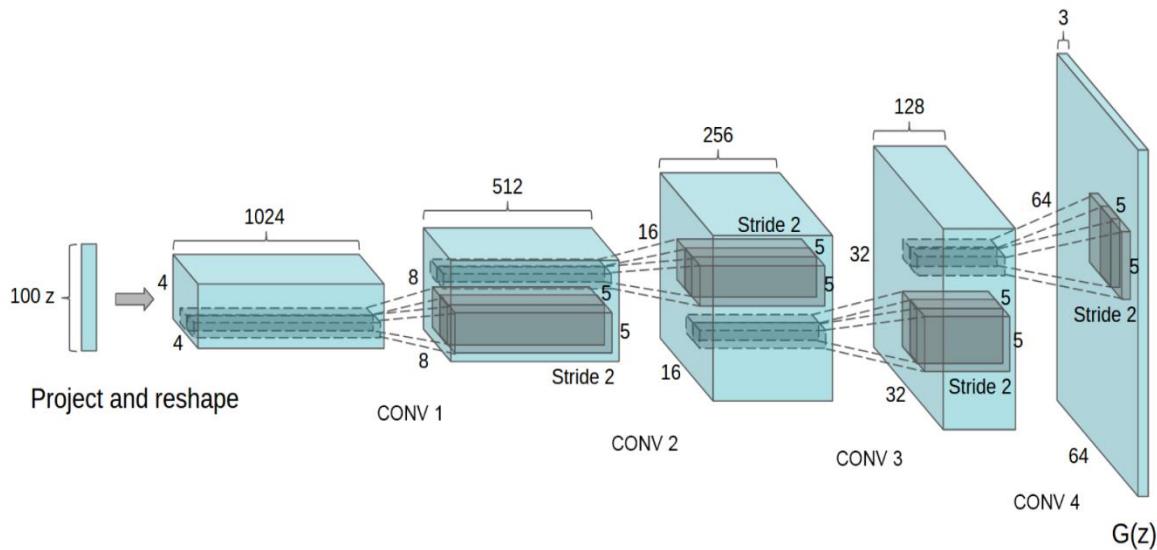
# DCGAN: Bedroom images



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

# Deep Convolutional GANs (DCGANs)

## Generator Architecture



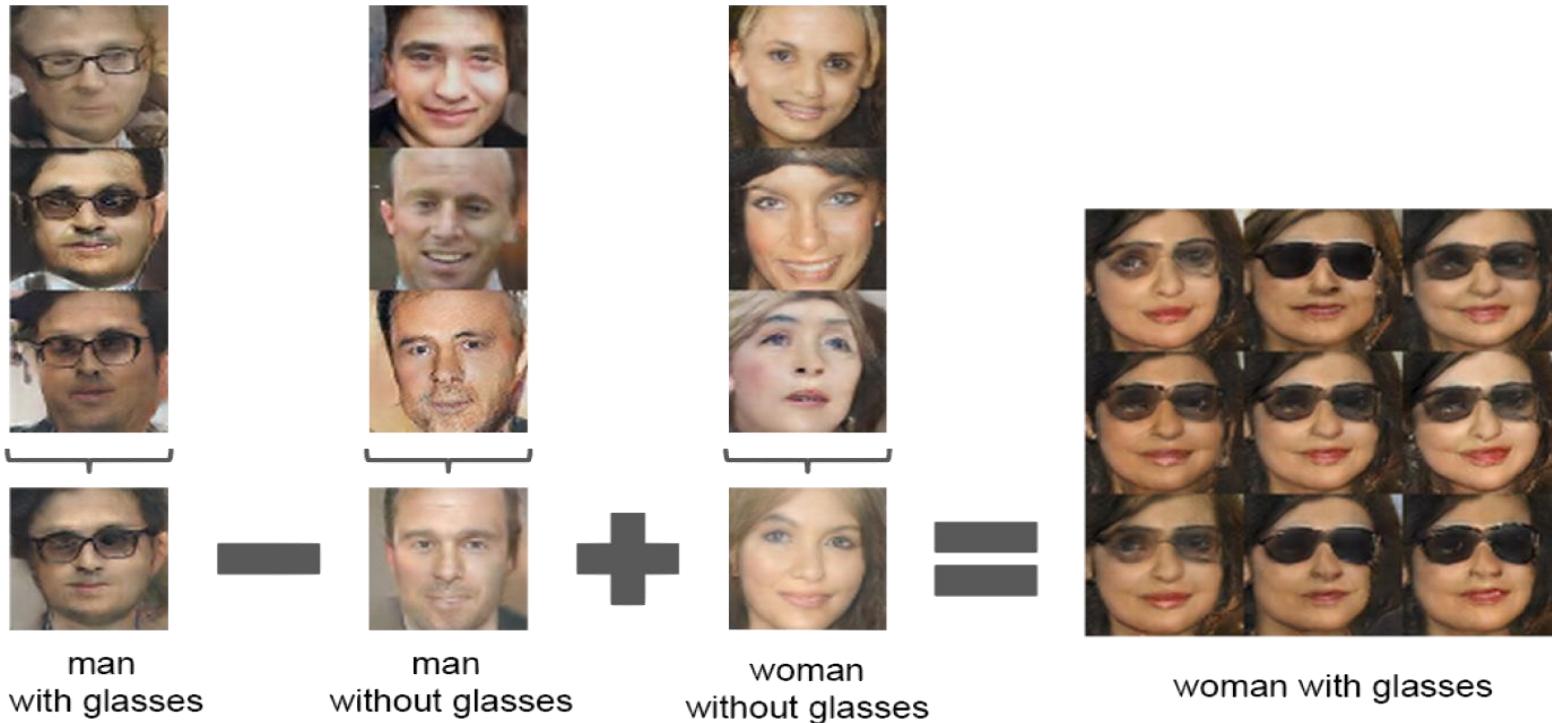
### Key ideas:

- Replace FC hidden layers with Convolutions
  - **Generator:** Fractional-Strided convolutions
- Use Batch Normalization after each layer
- **Inside Generator**
  - Use ReLU for hidden layers
  - Use Tanh for the output layer

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

# Deep Convolutional GANs (DCGANs)

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).

# Advantages of GANs

---

- **Plenty of existing work on Deep Generative Models**
  - Boltzmann Machine
  - Deep Belief Nets
  - Variational AutoEncoders (VAE)
- **Why GANs?**
  - Sampling (or generation) is straightforward.
  - Training doesn't involve Maximum Likelihood estimation.
  - Robust to Overfitting since Generator never sees the 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).

# 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**
  - Non-Convergence
  - Mode-Collapse

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

# Problems with GANs

---

- **Non-Convergence**
- Mode-Collapse

# Formulation

---

- Deep Learning models (in general) involve a single player
  - The player tries to maximize its reward (minimize its loss).
  - Use SGD (with Backpropagation) to find the optimal parameters.
  - 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

$$\min_x \max_y V(x, y)$$

Let  $V(x, y) = xy$

- 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
-------	-------	-------

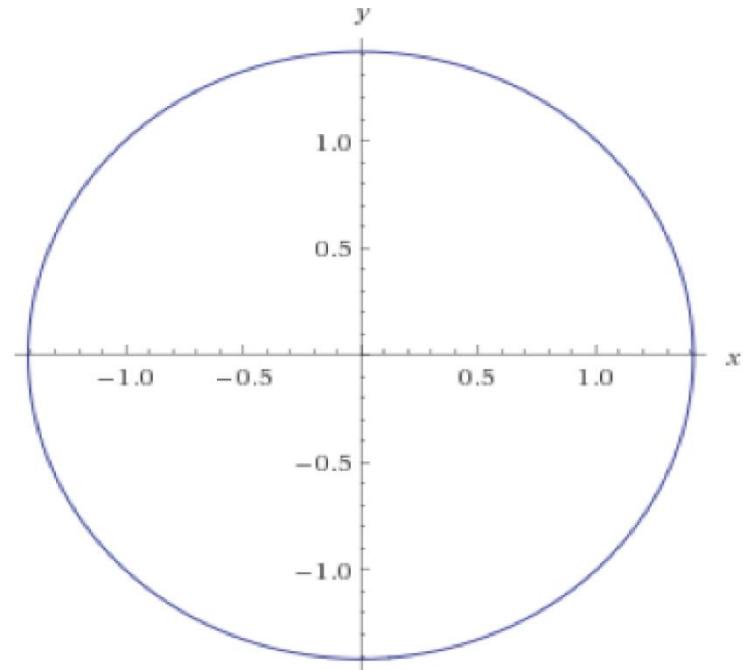
 == State 1 

Increase y	Decrease x
------------	------------

# Non-Convergence

$$\min_x \max_y xy$$

- $\frac{\partial}{\partial x} = -y$  ...  $\frac{\partial}{\partial y} = x$
- $\frac{\partial^2}{\partial y^2} = \frac{\partial}{\partial x} = -y$
- Differential equation's solution has sinusoidal terms
- Even with a small learning rate, it will not converge



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

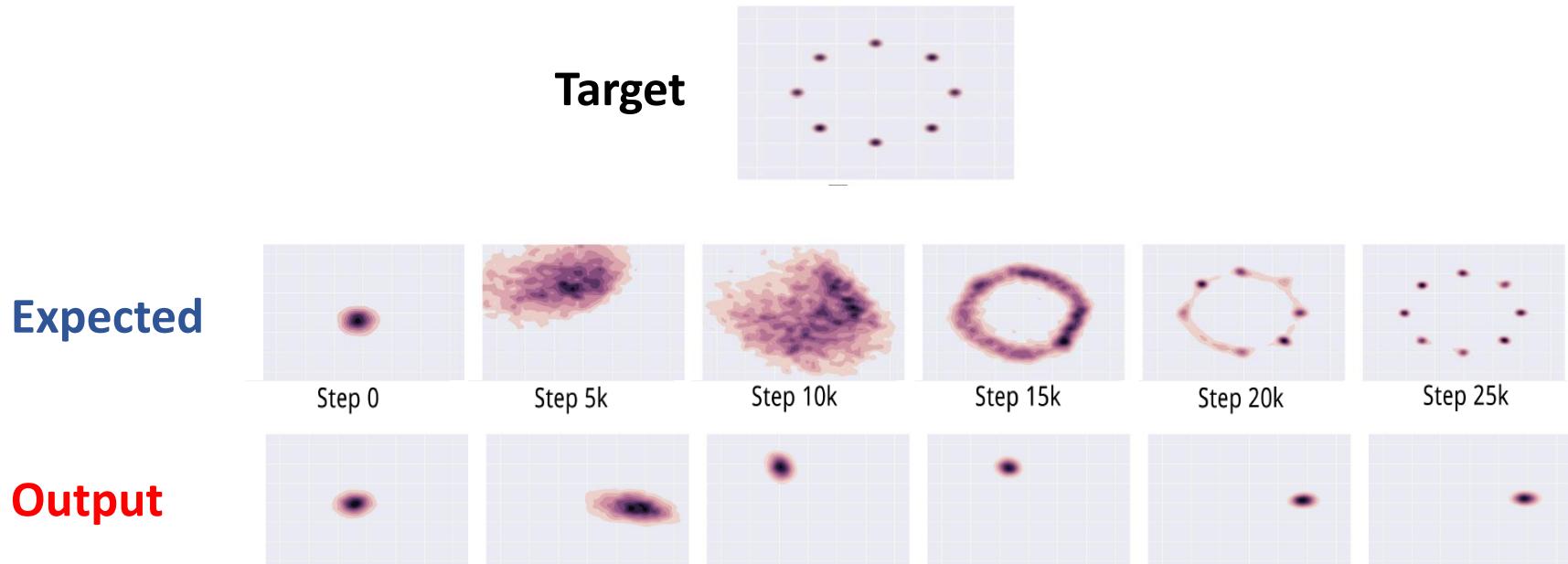
# Problems with GANs

---

- Non-Convergence
- **Mode-Collapse**

# Mode-Collapse

- Generator fails to output diverse samples



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

# Some Real Examples



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](#)

# Basic (Heuristic) Solutions

---

- **Mini-Batch GANs**
- Supervision with labels

# How to reward sample diversity?

---

- **At Mode Collapse,**
  - Generator produces good samples, but a very few of them.
  - Thus, Discriminator can't tag them as fake.
- **To address this problem,**
  - Let the Discriminator know about this edge-case.
- **More formally,**
  - 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
- **Thus,**
  - Generator will be forced to produce diverse samples.

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

# Mini-Batch GANs

---

- **Extract features that capture diversity in the mini-batch**
  - For e.g. 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**
  - Thus, 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.

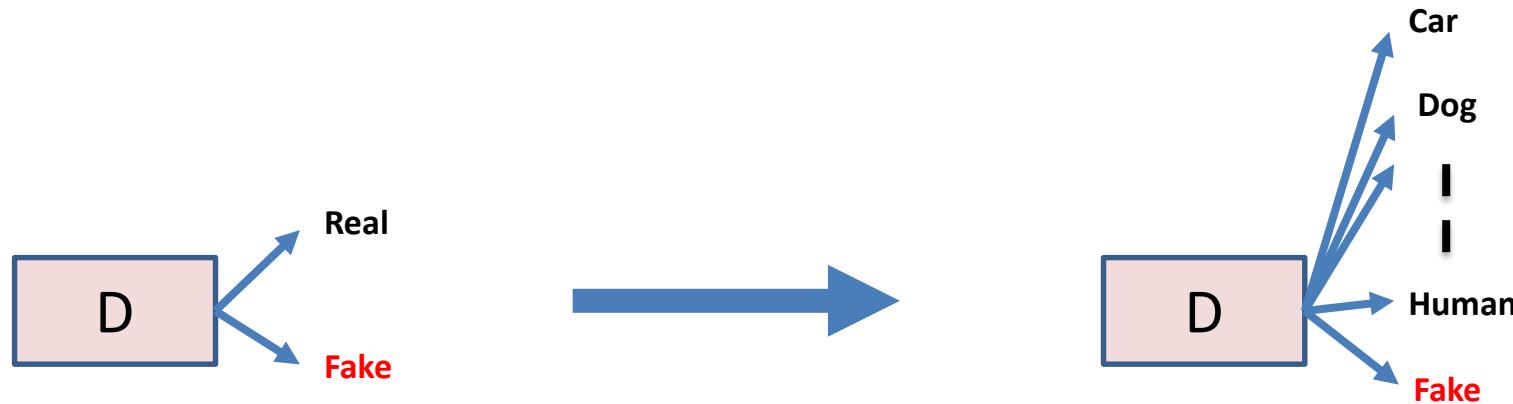
# Basic (Heuristic) Solutions

---

- Mini-Batch GANs
- **Supervision with labels**

# 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

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

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

$$D^* = \underset{D}{\operatorname{argmax}} V(D, G)$$

$$G^* = \underset{G}{\operatorname{argmin}} V(D, G)$$

- In this formulation, Discriminator's strategy was  $D(x) \rightarrow 1, D(G(z)) \rightarrow 0$
- Alternatively, we can flip the binary classification labels i.e. **Fake = 1, Real = 0**

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

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

# Alternate view of GANs (Contd.)

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

$$D^* = \operatorname{argmax}_D \mathbb{E}_{x \sim p(x)} [\log(1 - D(x))] + \mathbb{E}_{z \sim q(z)} [\log(D(G(z)))]$$

We can use this

$$D^* = \operatorname{argmin}_D \mathbb{E}_{x \sim p(x)} \log(D(x)) + \mathbb{E}_{z \sim q(z)} [\log(1 - D(G(z)))]$$

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

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

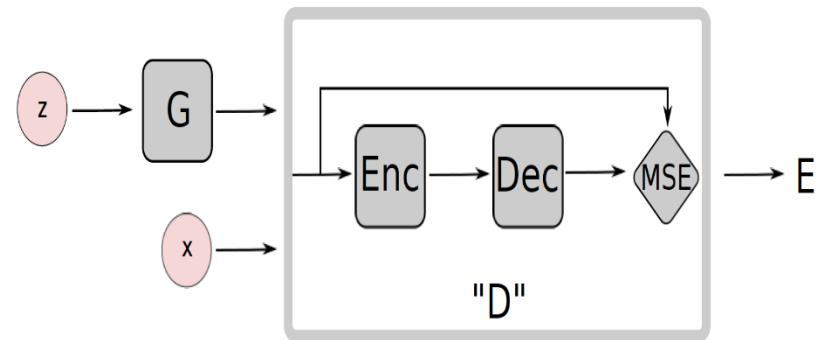
- And thus, instead of outputting probabilities, Discriminator just has to output :-
  - High values for fake samples
  - Low values for real samples

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 scores to fake values
- 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

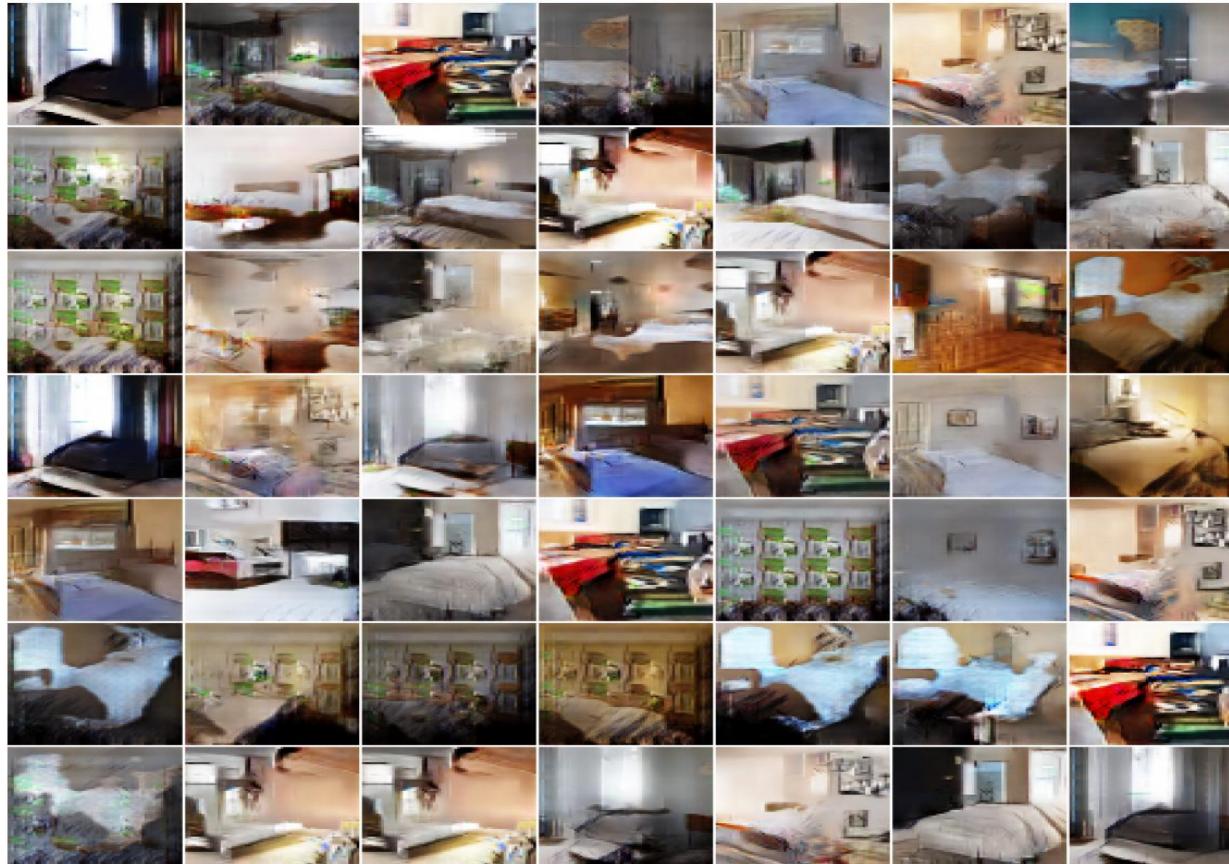
$$D(x) = ||Dec(Enc(x)) - x||_{MSE}$$



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

# Examples

More Bedrooms...



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

# Examples

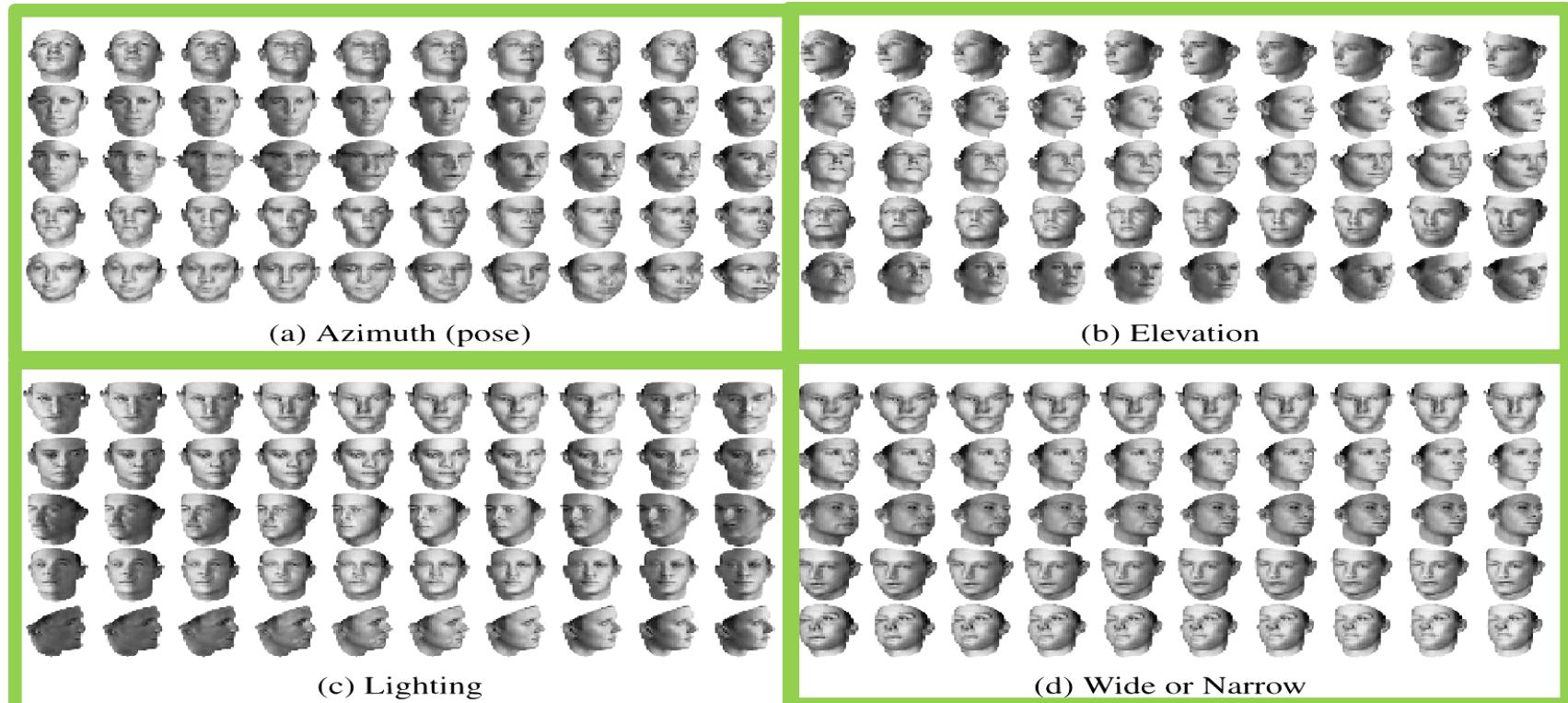
Celebs...



# Examples

## The Cool Stuff...

3D Faces

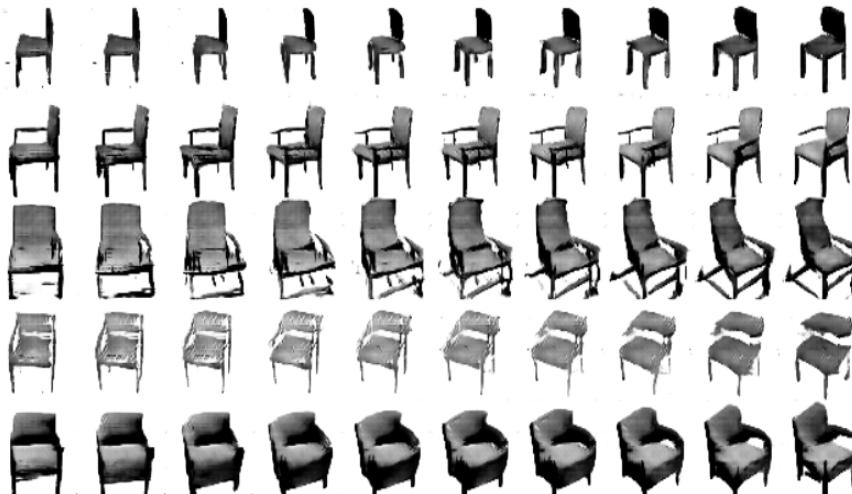


Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. InfoGAN: Interpretable Representation Learning by Information Maximization Generative Adversarial Nets, NIPS (2016).

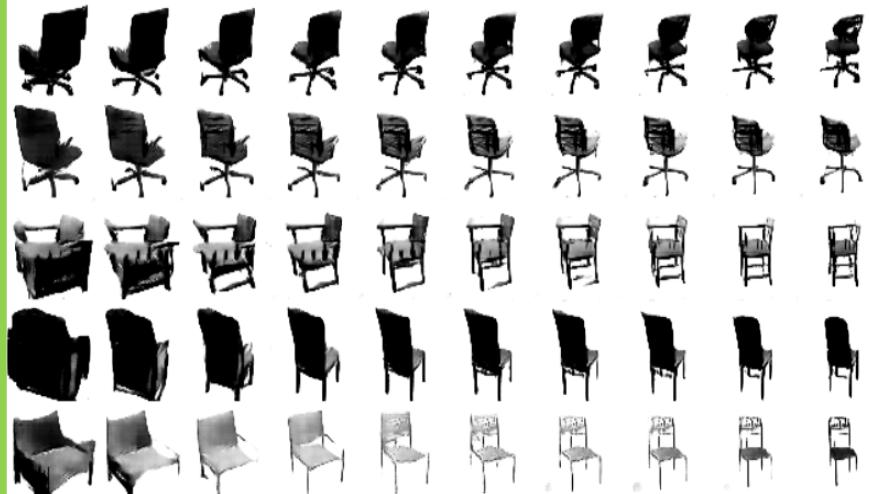
# Examples

## Cool Stuff (contd.)

3D Chairs



(a) Rotation

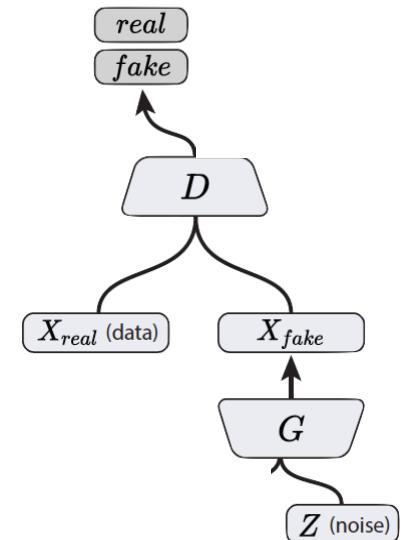


(b) Width

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

# 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 the main attributes of the image
    - For e.g. **Digit**, **Angle** and **Thickness** of images in MNIST
- If **c** vector captures the key variations in the image,  
**Will c and  $x_{fake}$  be highly correlated or weakly correlated?**



# Recap: Mutual Information

---

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

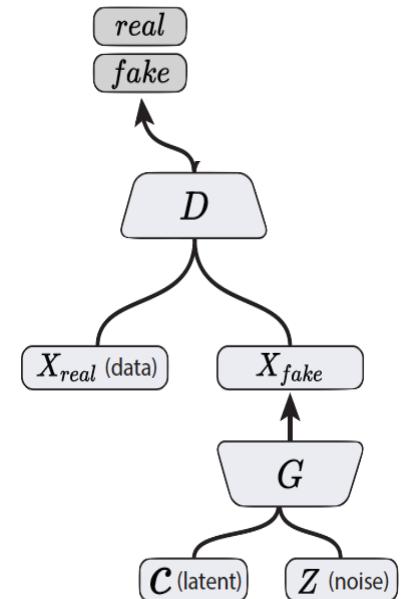
$$I(X; Y) = \sum_{x,y} p(x, y) \log \left( \frac{p(x, y)}{p(x)p(y)} \right)$$

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

# InfoGAN

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

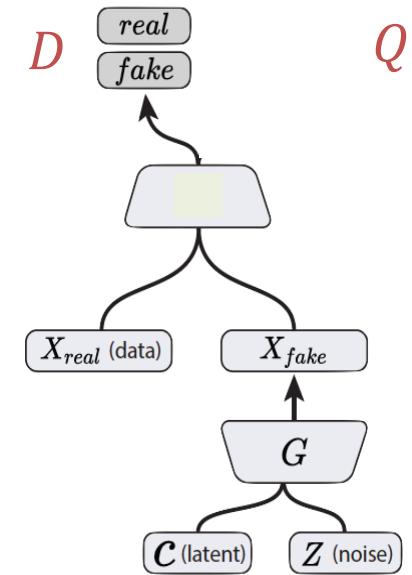
$$\min_G \max_D V_I(D, G) = V(D, G) - \lambda I(c; G(z, c))$$



# InfoGAN

## Mutual Information's Variational Lower bound

$$\begin{aligned} I(c; G(z, c)) &= H(c) - H(c|G(z, c)) \\ &= \mathbb{E}_{x \sim G(z, c)} \left[ \mathbb{E}_{c' \sim P(C|x)} [\log P(c'|x)] \right] + H(c) \\ &= \mathbb{E}_{x \sim G(z, c)} \left[ D_{KL}(P||Q) + \mathbb{E}_{c' \sim P(C|x)} [\log Q(c'|x)] \right] + H(c) \\ &\geq \mathbb{E}_{x \sim G(z, c)} \left[ \mathbb{E}_{c' \sim P(C|x)} [\log Q(c'|x)] \right] + H(c) \\ &\geq \mathbb{E}_{c \sim P(c), x \sim G(z, c)} [\log Q(c|x)] + H(c) \end{aligned}$$



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.

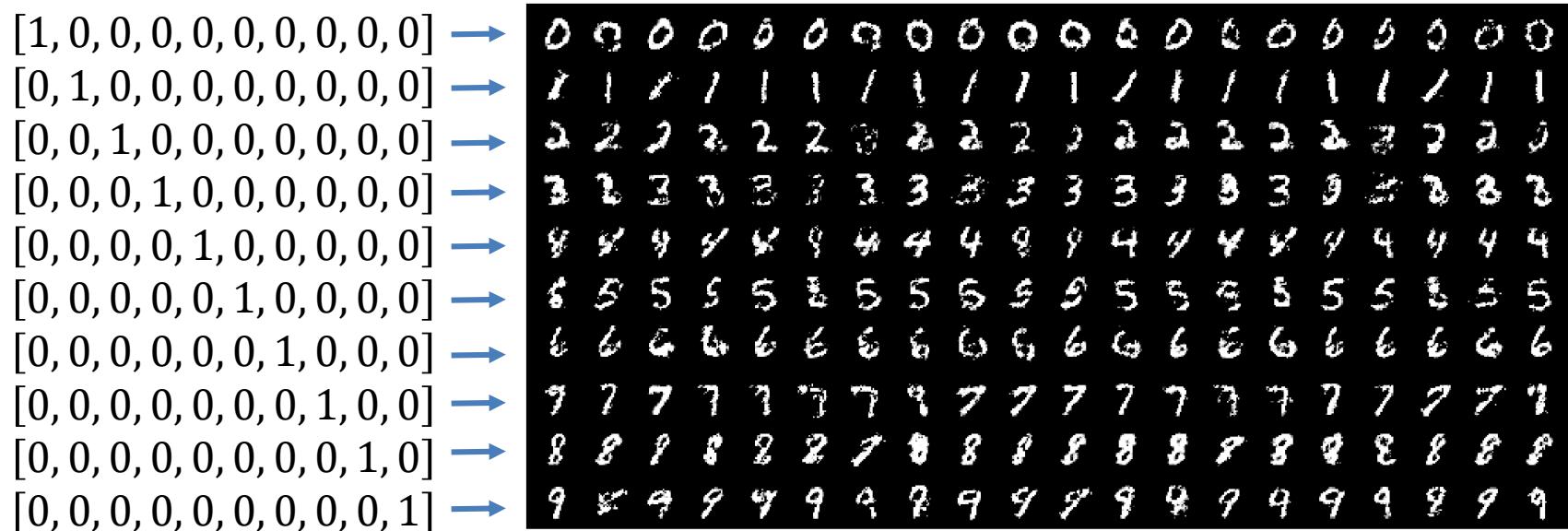
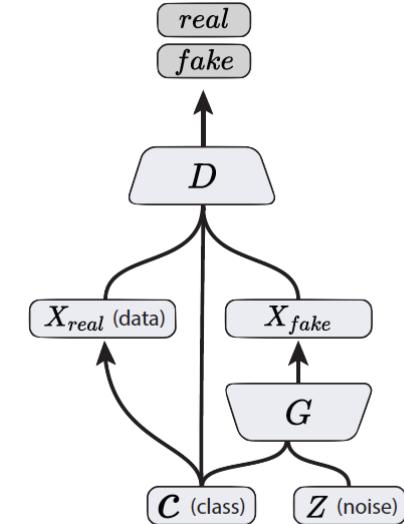


Figure 2 in the original paper.

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 multi-modal learning.
- Lends to many practical applications of GANs when we have explicit *supervision* available.



Conditional GAN  
(Mirza & Osindero, 2014)

Image Credit: Figure 2 in Odena, A., Olah, C. and Shlens, J., 2016. Conditional image synthesis with auxiliary classifier GANs. *arXiv preprint arXiv:1610.09585*.

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

# Image-to-Image Translation

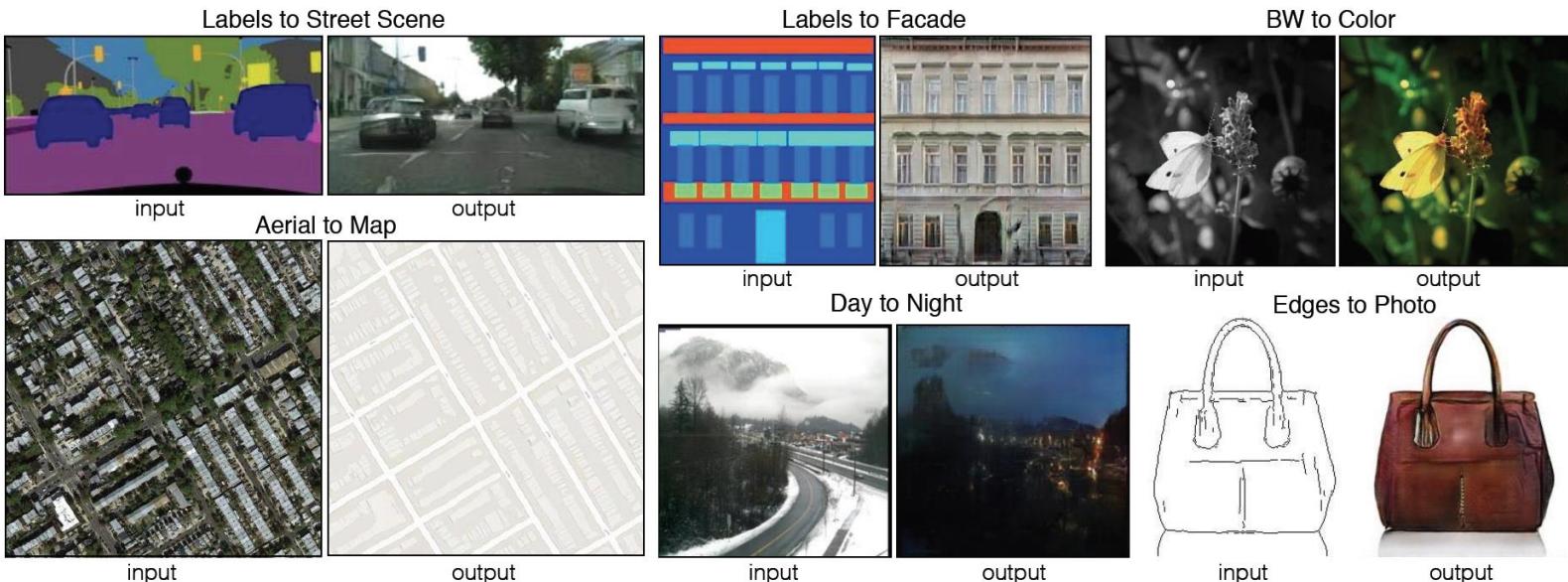


Figure 1 in the original paper.

[Link to an interactive demo of this 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.
- Conditional GANs provide an effective way to handle many complex domains without worrying about designing *structured loss* functions explicitly.

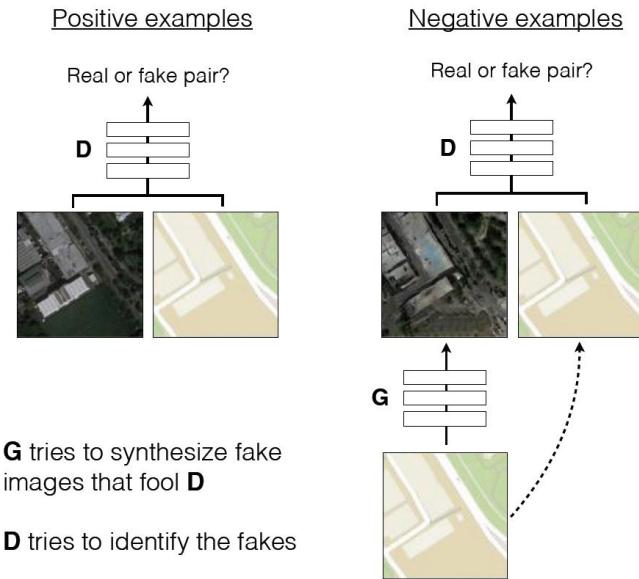


Figure 2 in the original paper.

# 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.

this small bird has a pink breast and crown, and black primaries and secondaries.



this magnificent fellow is almost all black with a red crest, and white cheek patch.



the flower has petals that are bright pinkish purple with white stigma



this white and yellow flower have thin white petals and a round yellow stamen



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

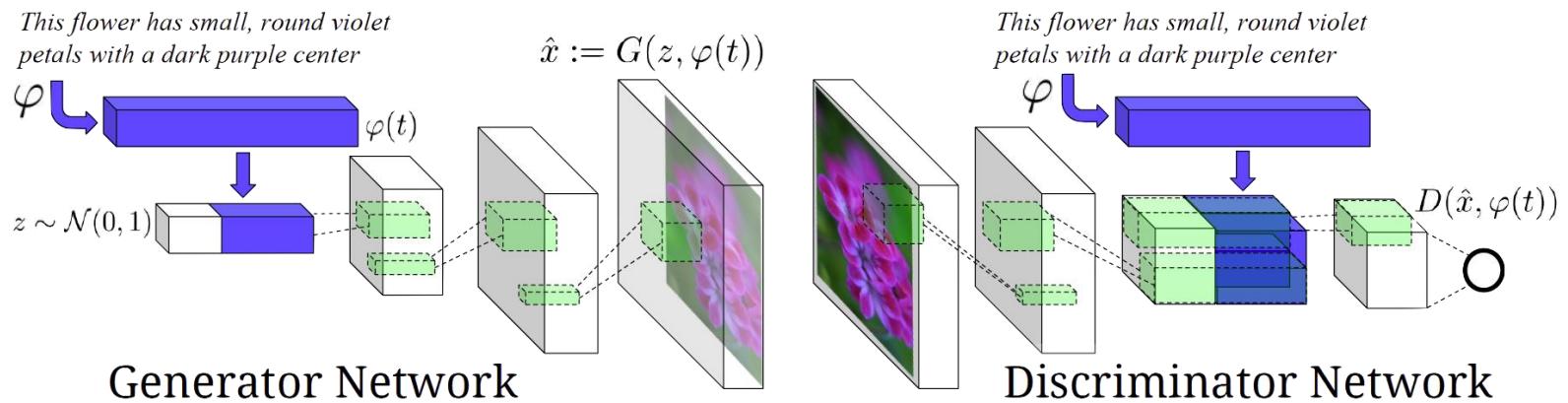


Figure 2 in the original paper.

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

- Differentiating Feature: Uses an *Identity Preservation Optimization* using an auxiliary network to get a better approximation of the latent code ( $z^*$ ) for an input image.
- Latent code is then conditioned on a discrete (one-hot) embedding of age categories.

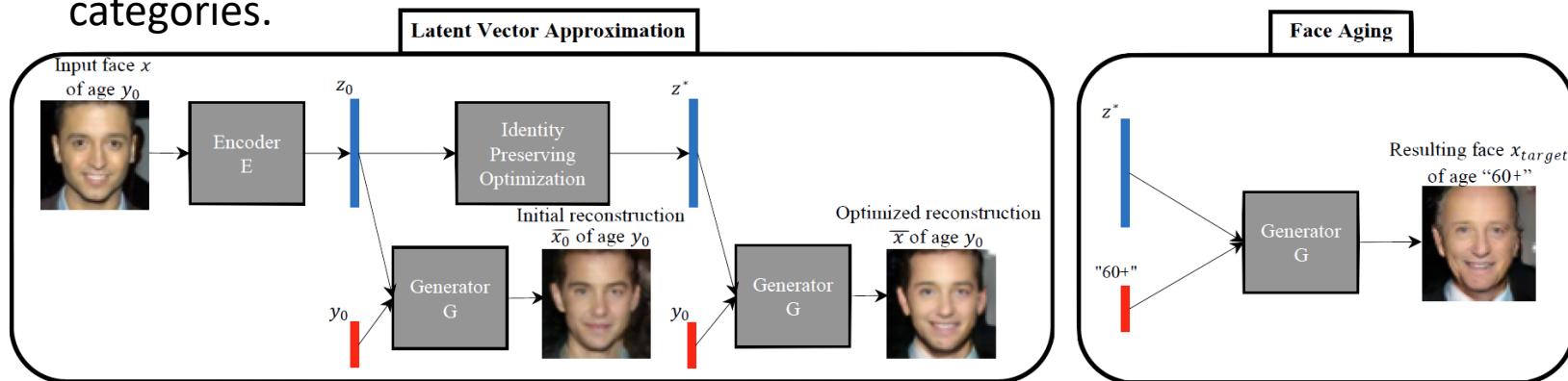


Figure 1 in the original paper.

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

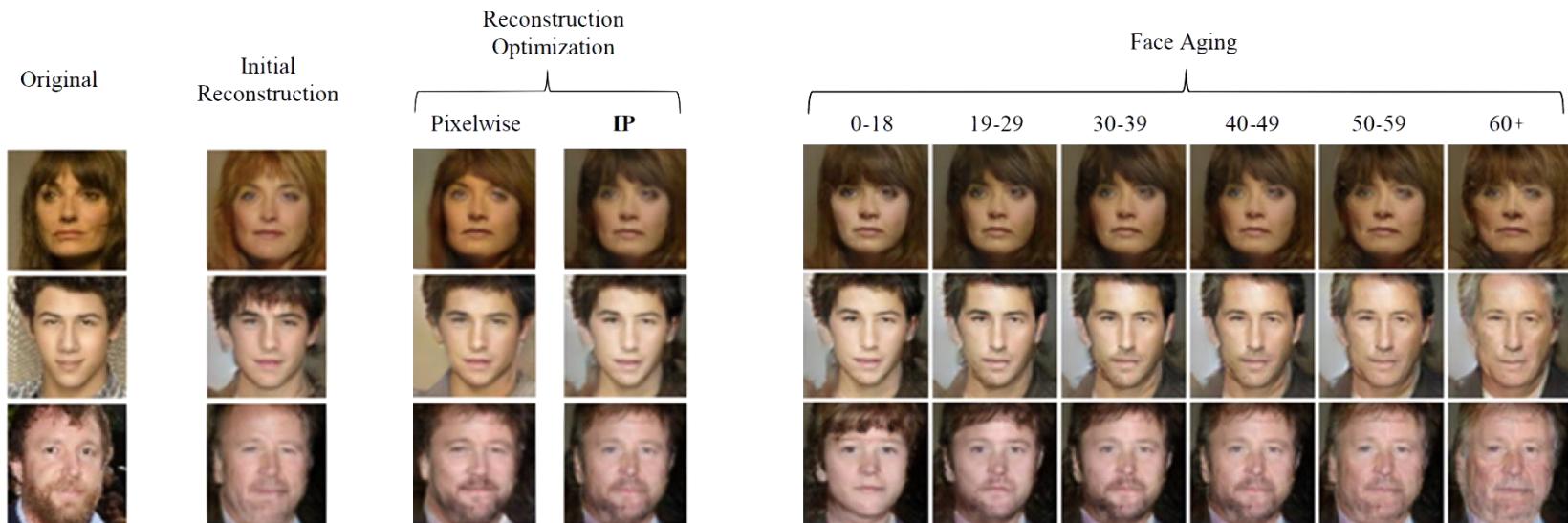


Figure 3 in the original paper.

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

# Conditional GANs

## Conditional Model Collapse

- Scenario observed when the Conditional GAN starts *ignoring* either the code ( $c$ ) or the noise variables ( $z$ ).
- This limits the diversity of images generated.



Credit?

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

# Coupled GAN

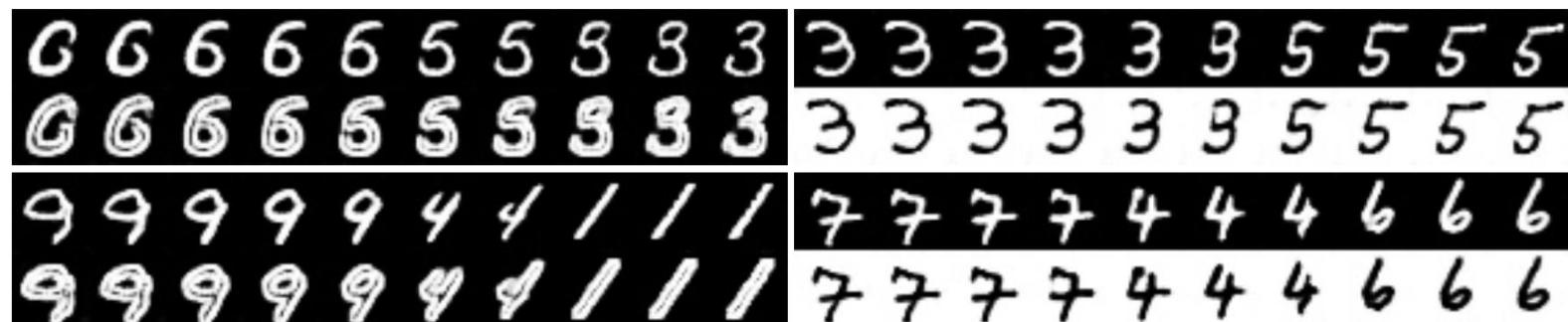


Figure 2 in the original paper.

- Learning a *joint distribution* of *multi-domain* images.
- Using GANs to learn the joint distribution with samples drawn from the marginal distributions.
- Direct applications in domain adaptation and image translation.

Liu, Ming-Yu, and Oncel Tuzel. “Coupled generative adversarial networks”. NIPS (2016).

# Coupled GANs

- Architecture

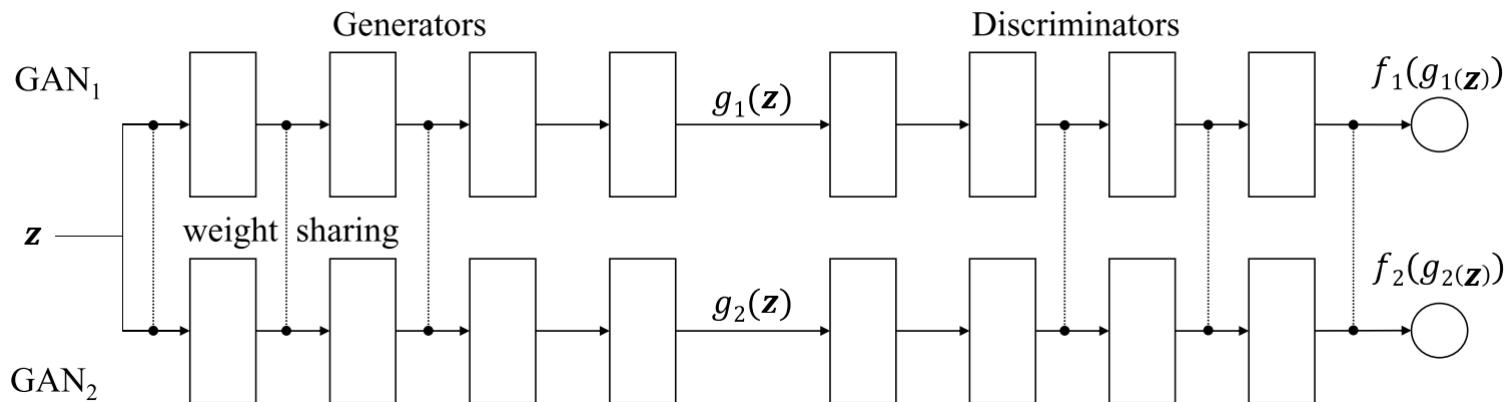


Figure 1 of the original paper.

Weight-sharing constraints the network to learn a *joint distribution* without corresponding supervision.

Liu, Ming-Yu, and Oncel Tuzel. “Coupled generative adversarial networks”. NIPS (2016).

# Coupled GANs

- Some examples of generating facial images across different feature domains.
- Corresponding images in a column are generated from the same latent code  $z$



Figure 4 in the original paper.

Liu, Ming-Yu, and Oncel Tuzel. "Coupled generative adversarial networks". NIPS (2016).

# Laplacian Pyramid of Adversarial Networks

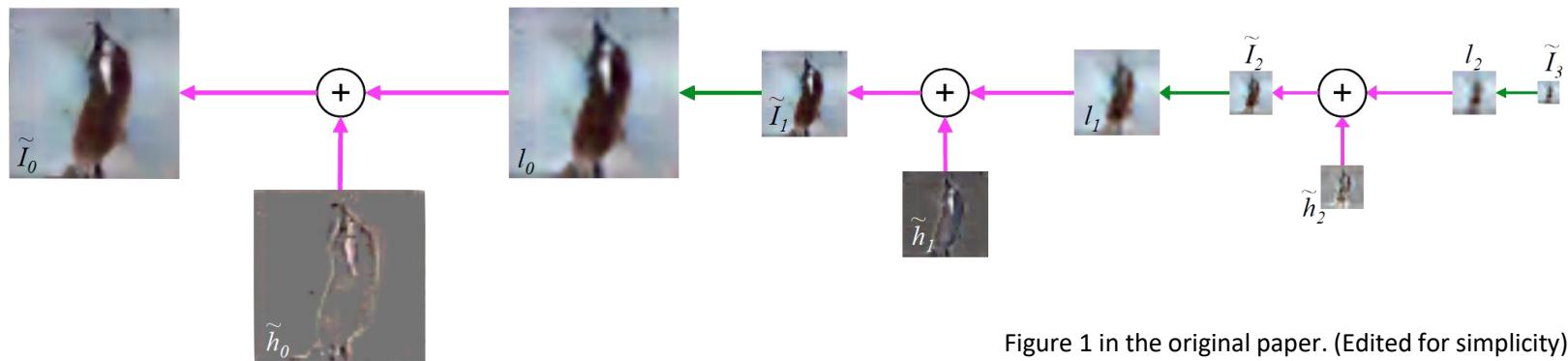


Figure 1 in the original paper. (Edited for simplicity)

- 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

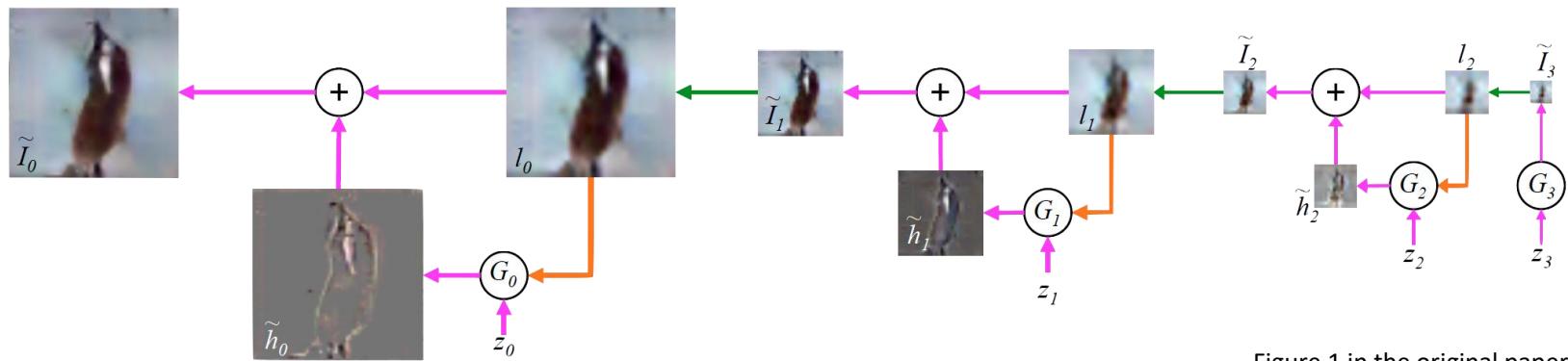


Figure 1 in the original paper.

## 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

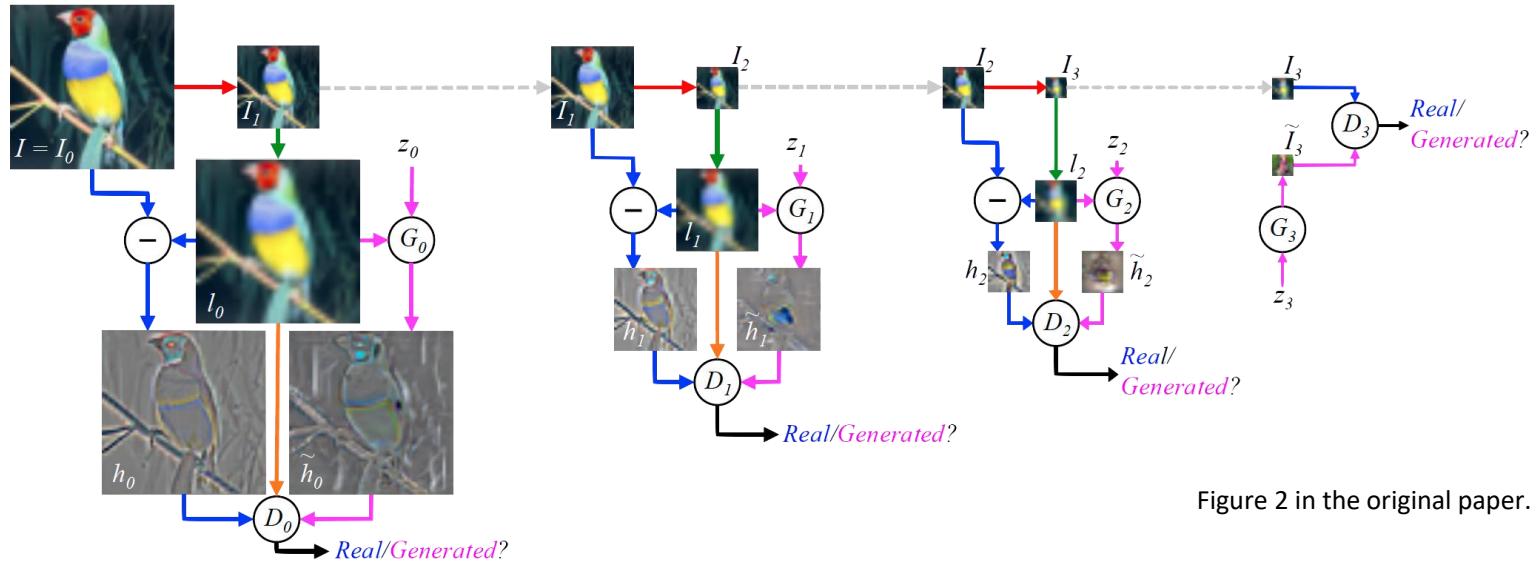


Figure 2 in the original paper.

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) :

$$q(x, z) = q(x) q(z|x) \quad \text{encoder distribution}$$

$$p(x, z) = p(z) p(x|z) \quad \text{generator distribution}$$

# Adversarially Learned Inference

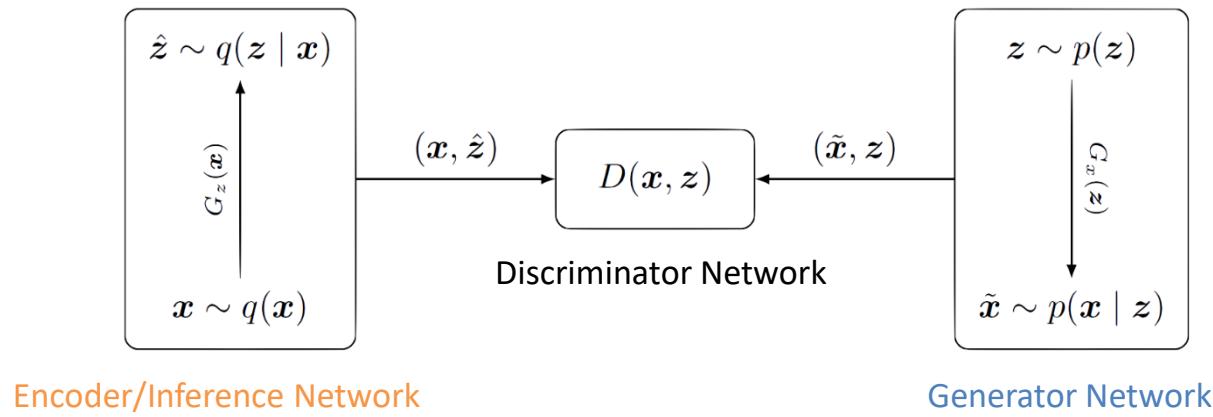


Figure 1 in the original paper.

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

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

# Adversarially Learned Inference

---

- Nash equilibrium yields
  - **Joint:**  $p(x, z) \sim q(x, z)$
  - **Marginals:**  $p(x) \sim q(x)$  and  $p(z) \sim q(z)$
  - **Conditionals:**  $p(x|z) \sim q(x|z)$  and  $p(z|x) \sim q(z|x)$
- Inferred latent representation successfully reconstructed the original image.
- Representation was useful in the downstream semi-supervised task.

# 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 (in tandem) to fool the other component. In this process, both models become better at their respective tasks.

# 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.

# Outline

---

**1** Course Review

**2** Generative Adversarial Networks

**3** Variational Autoencoder

**4** Advanced Generation Methods

# Outline

---

**1** Course Review

**2** Generative Adversarial Networks

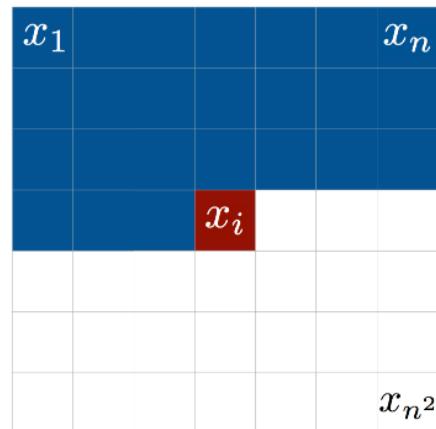
**3** Variational Autoencoder

**4** Advanced Generation Methods

# Pixel-by-pixel generation

---

A simple way to iterate, employ feedback  
and capture pixel dependencies



# Pixel-by-pixel Generation

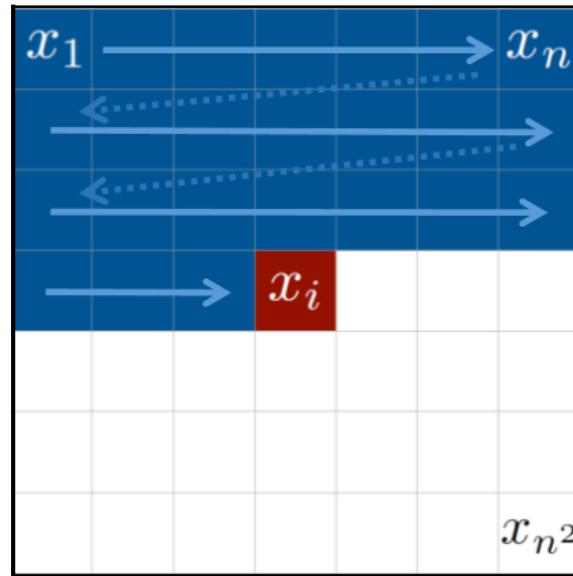
---

How to include statistical dependencies over hundreds of pixels?



Image Source: <http://wallpapers.ae/wp-content/uploads/2016/01/Tom-And-Jerry-Funny-Wallpaper.jpg>

# Intuition



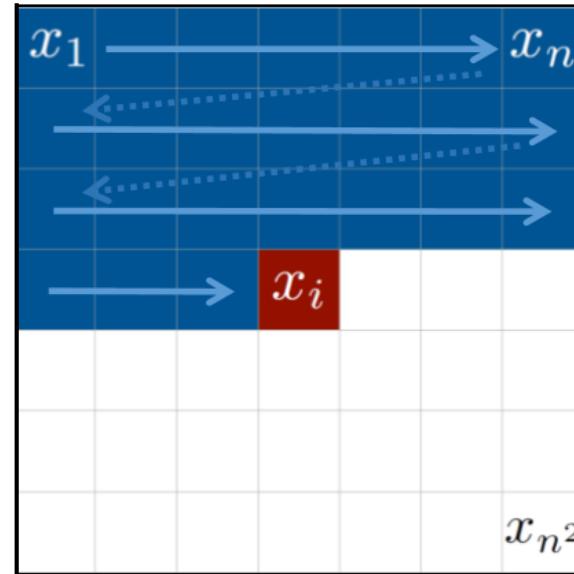
[Pixel recurrent neural networks](#), ICML 2016

# Intuition

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

Bayes Theorem:

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



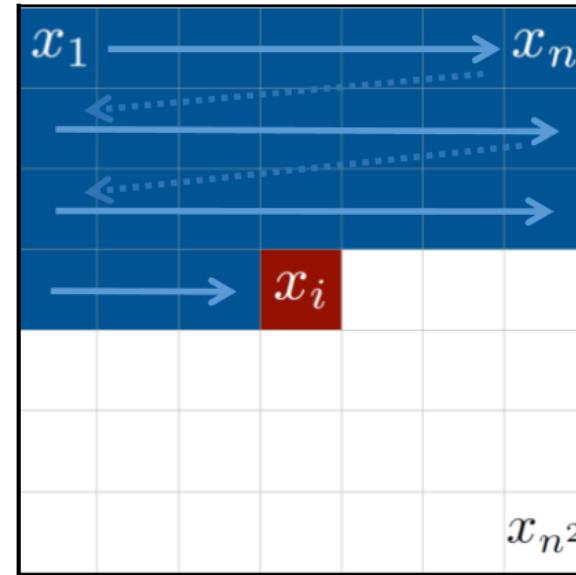
# Intuition

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

Bayes Theorem:

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

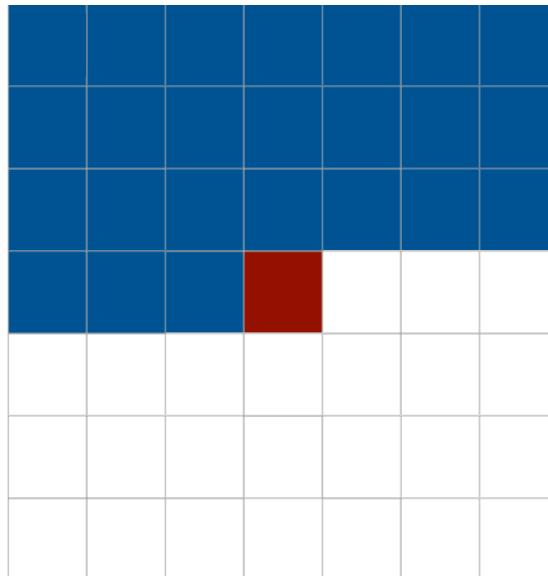
A sequential model!



# Intuition

---

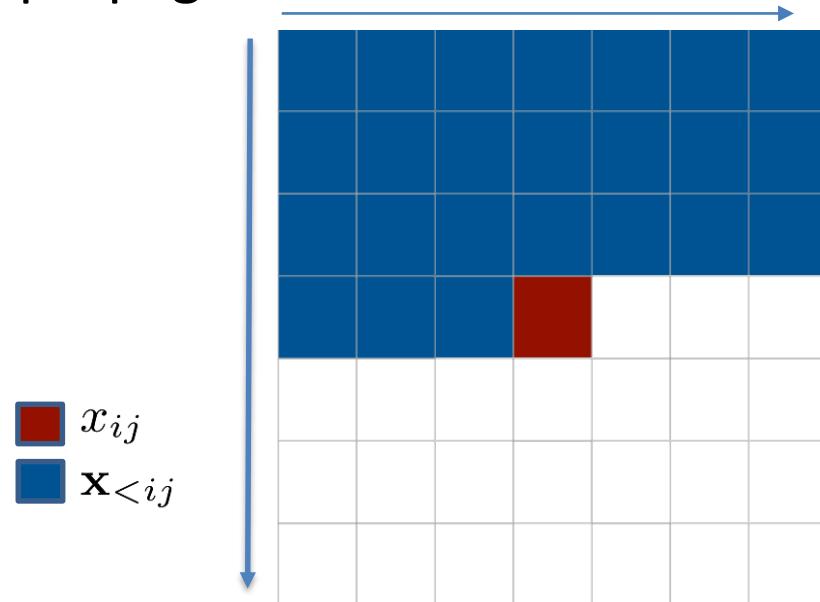
- Question: Can we use plain-LSTM to generate images pixels by pixels?



[Pixel recurrent neural networks](#), ICML 2016

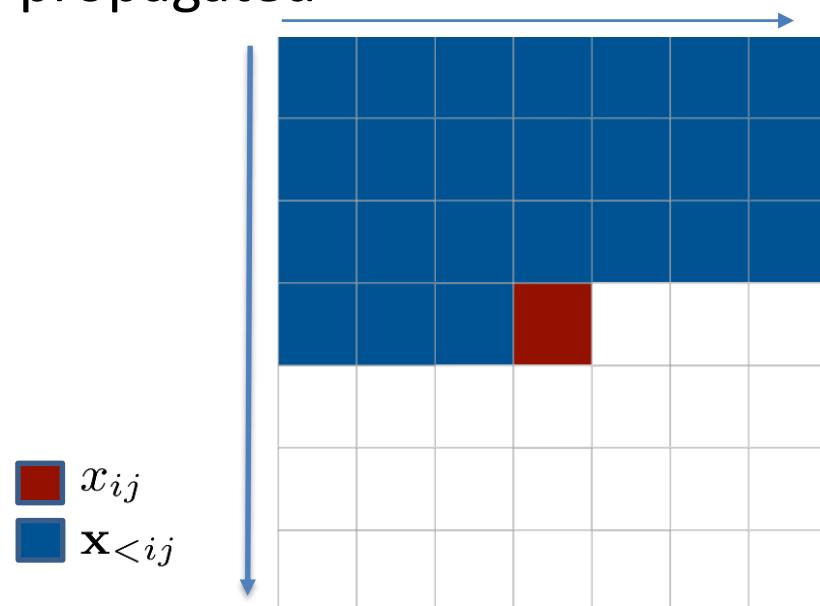
# Intuition

- Question: Can we use plain-LSTM to generate images pixels by pixels?
- Ensure information is well propagated in two dimensions

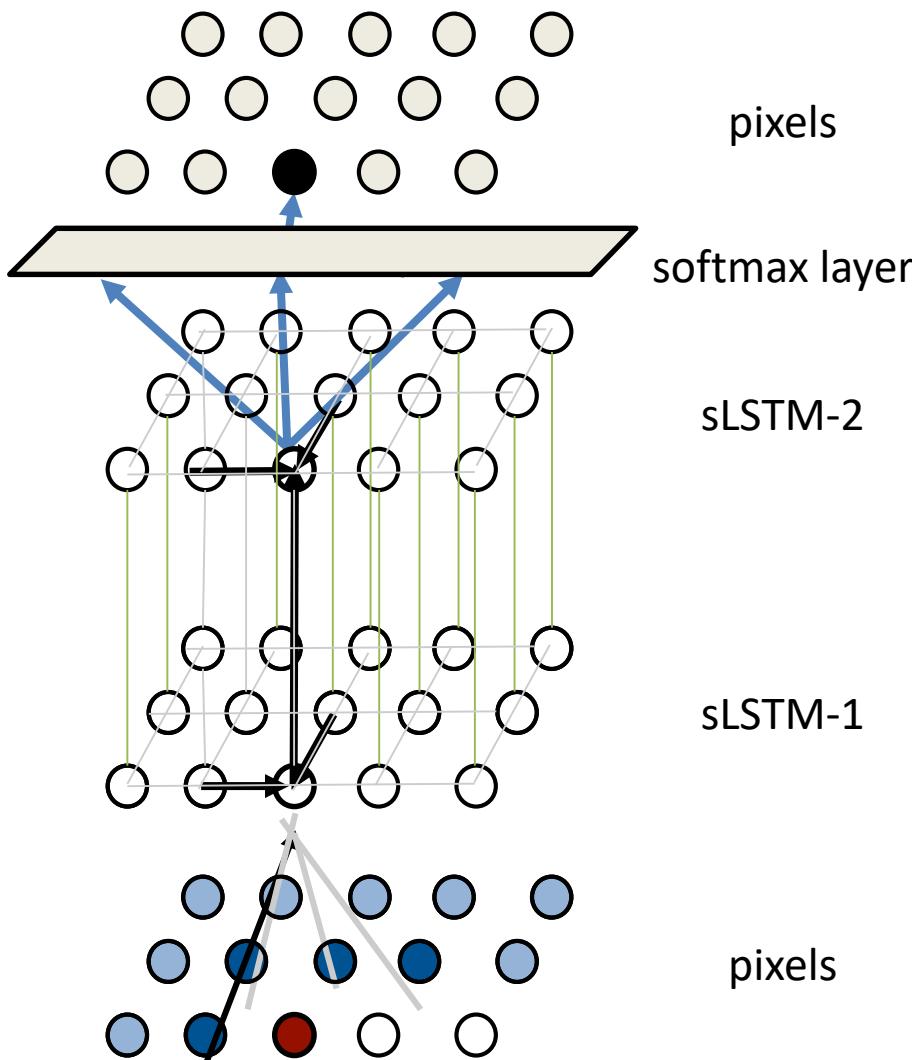


# Intuition

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

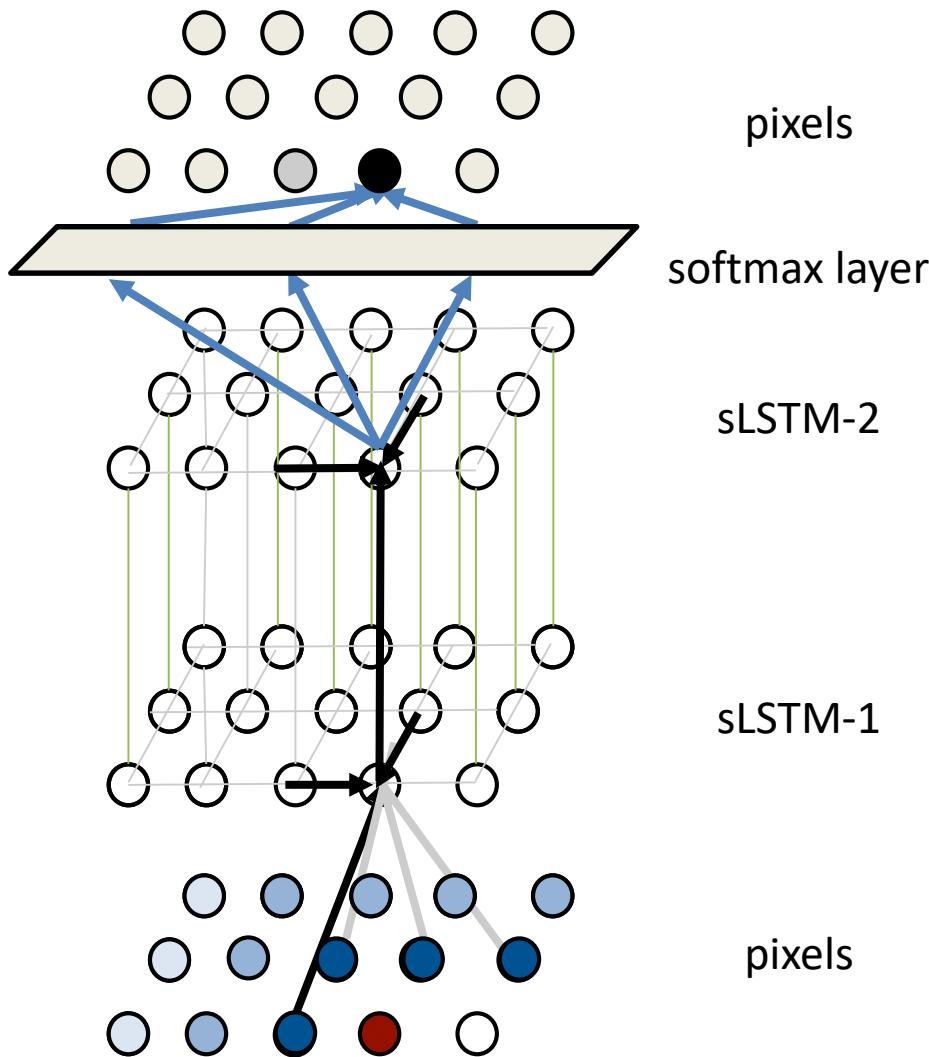


# Spatial LSTM



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

- Treat pixels as discrete variables:
  - To estimate a pixel value, do classification in every channel (256 classes indicating pixel values 0-255)
  - Implemented with a final softmax layer

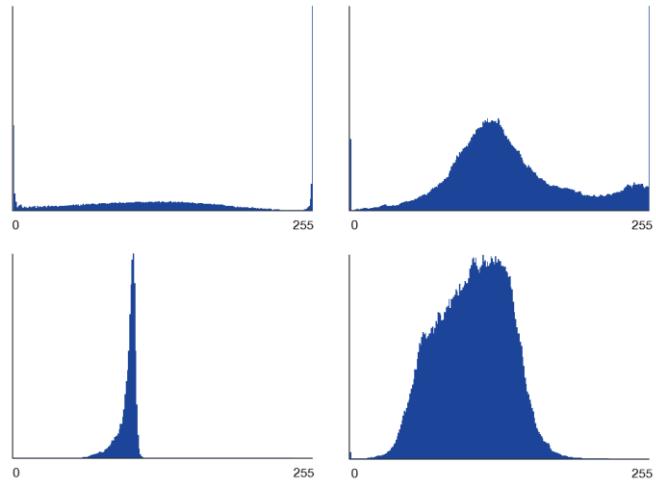
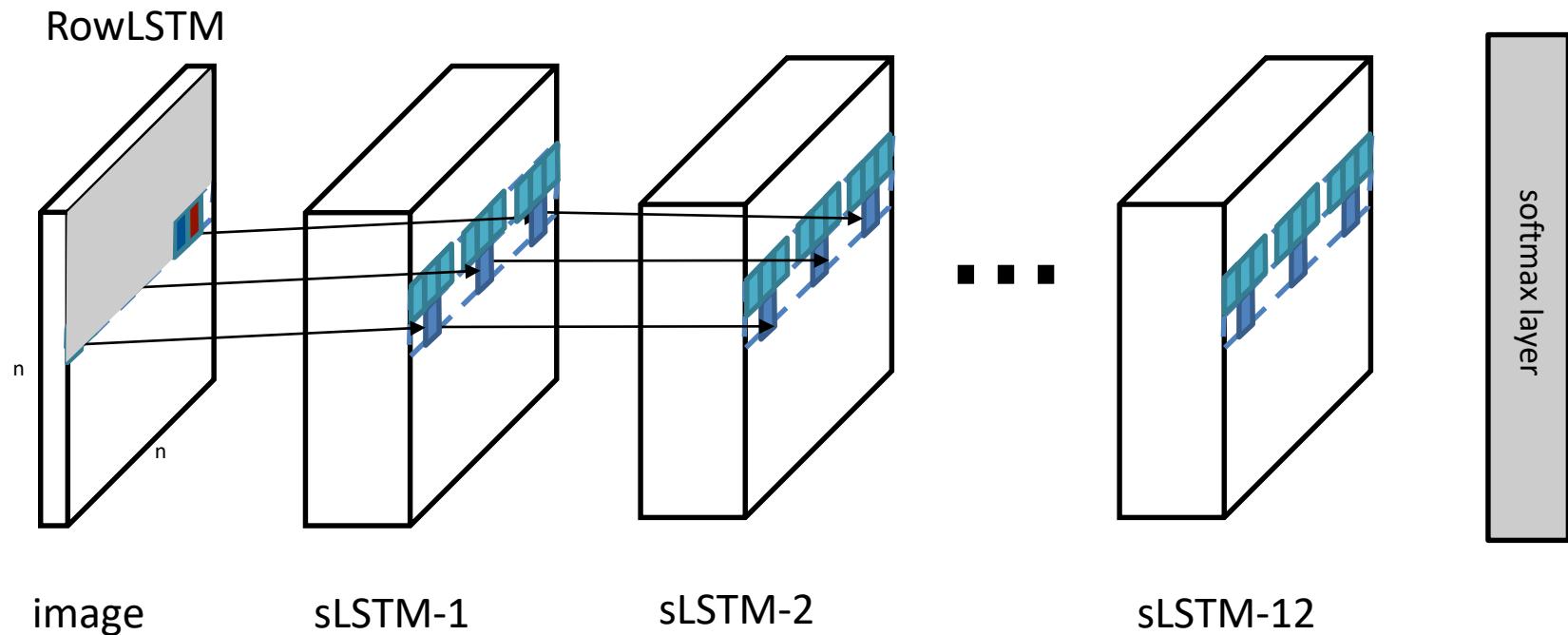


Figure: Example softmax outputs in the final layer, representing probability distribution over 256 classes.

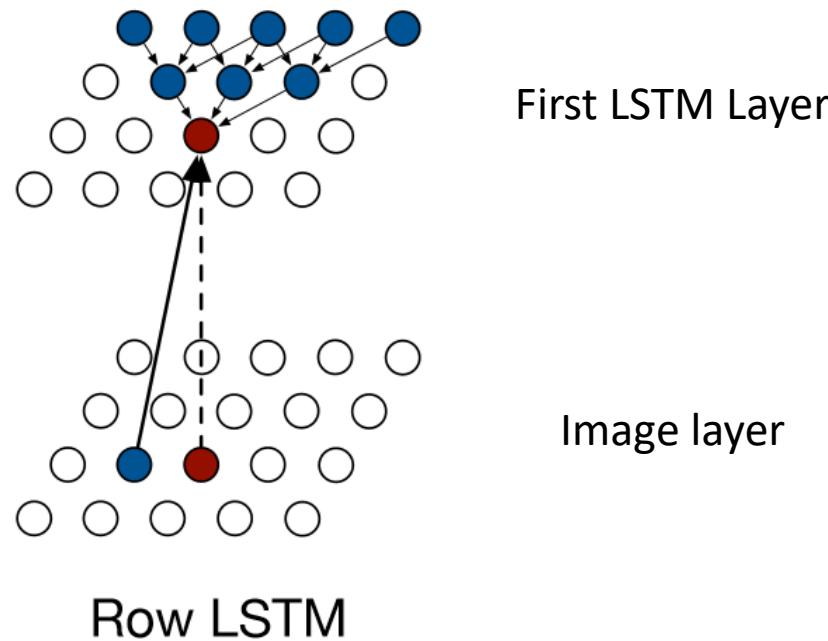
Figure from: [Oord et al.](#)

# PixelRNN: A specific Multidimensional LSTM



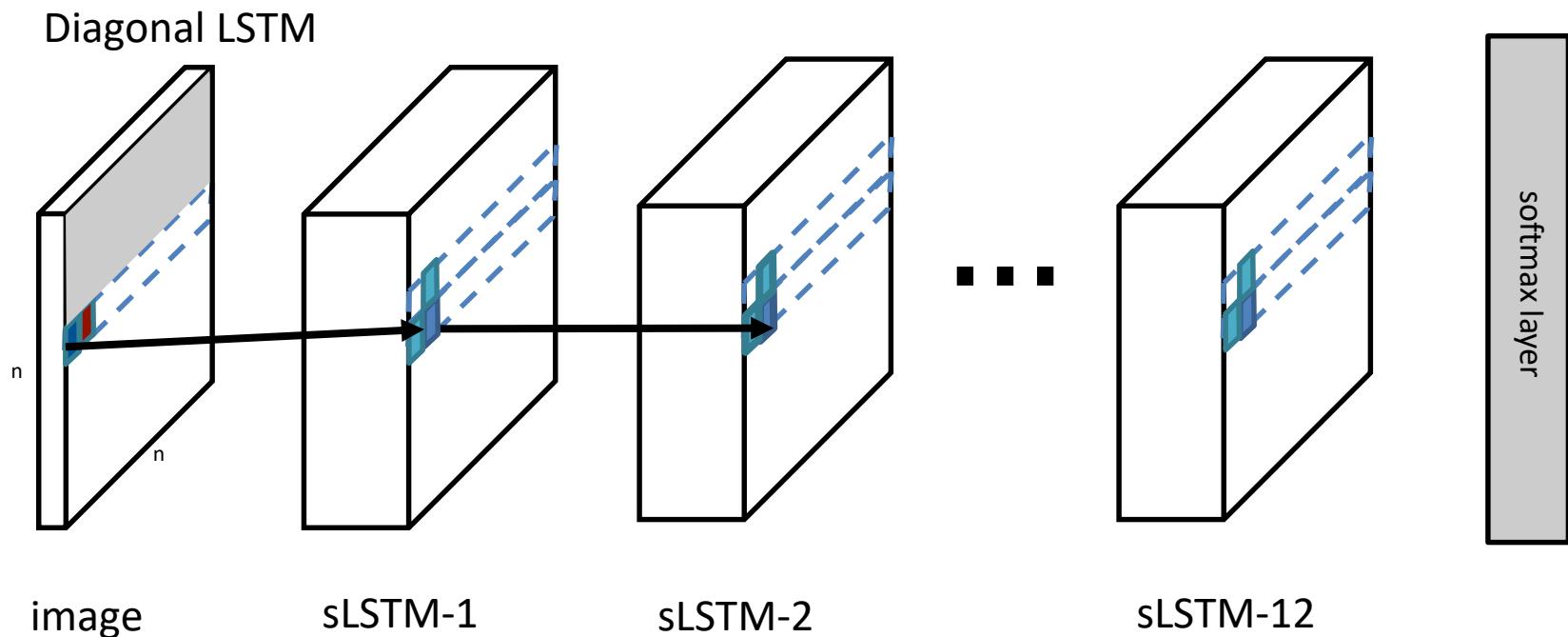
[Pixel recurrent neural networks](#), ICML 2016

# PixelRNN: A specific Multidimensional LSTM



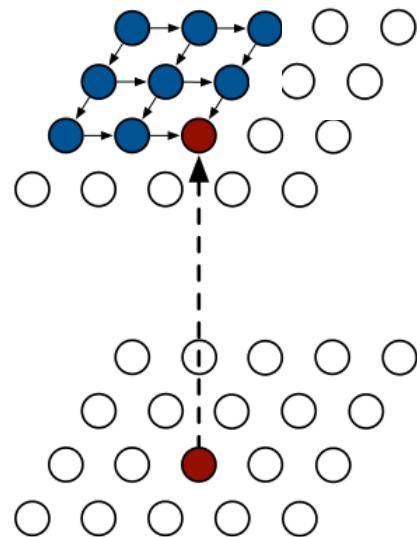
[Pixel recurrent neural networks](#), ICML 2016

# PixelRNN: A specific Multidimensional LSTM



[Pixel recurrent neural networks](#), ICML 2016

# PixelRNN: A specific Multidimensional LSTM

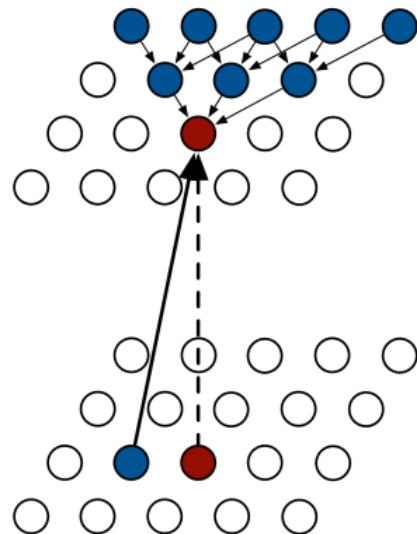


Diagonal LSTM

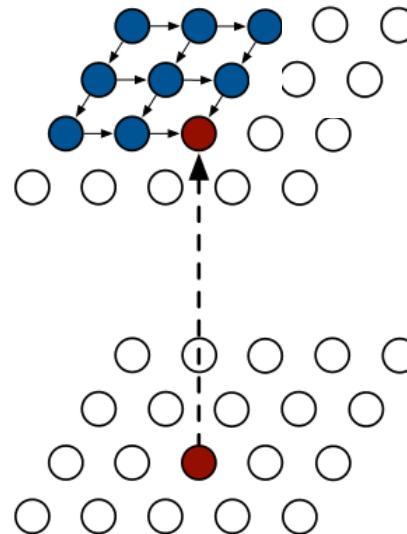
[Pixel recurrent neural networks](#), ICML 2016

# PixelRNN: A specific Multidimensional LSTM

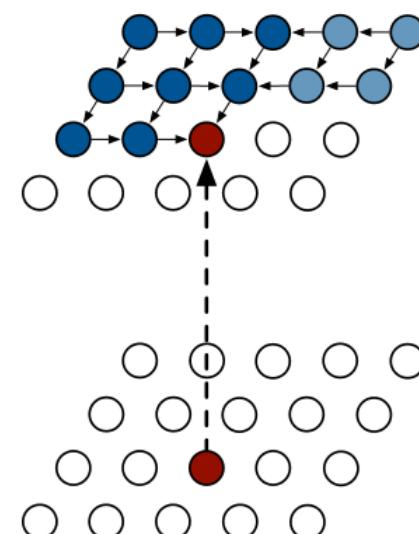
Receptive Field



Row LSTM



Diagonal LSTM



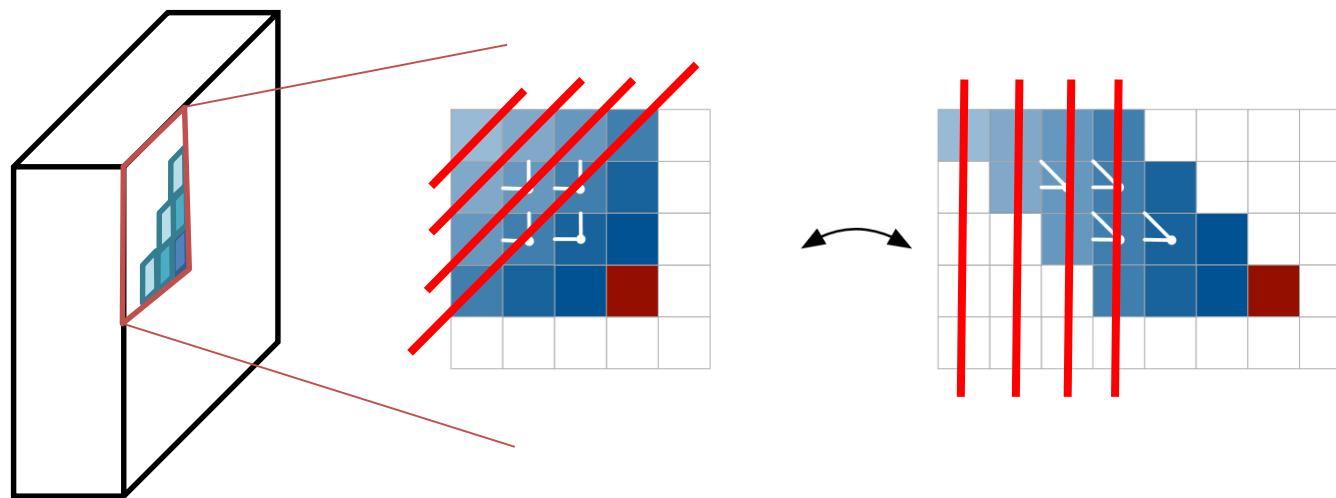
Diagonal BiLSTM

[Pixel recurrent neural networks](#), ICML 2016

# PixelRNN: A specific Multidimensional LSTM

## Diagonal LSTM

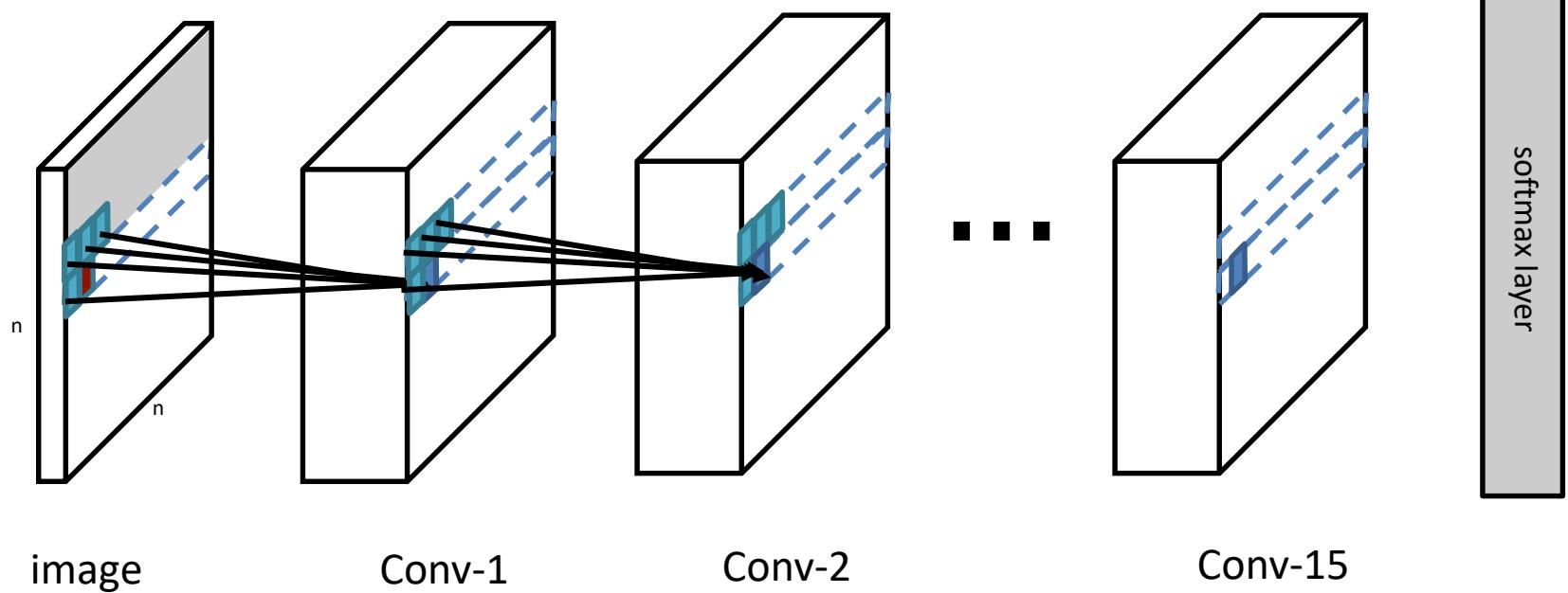
- To optimize, we skew the feature maps so it can be parallelized



[Pixel recurrent neural networks](#), ICML 2016

# PixelCNN

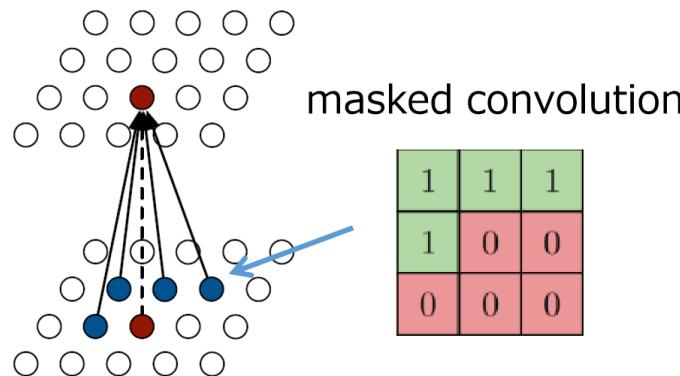
---



# 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

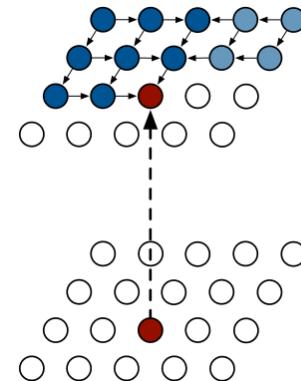
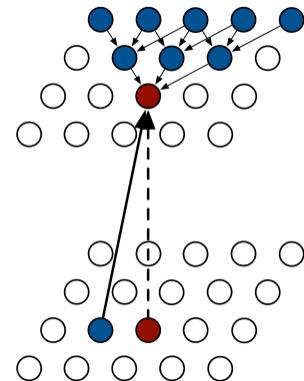
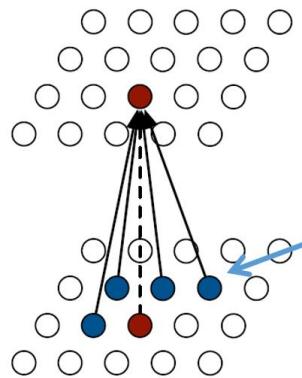
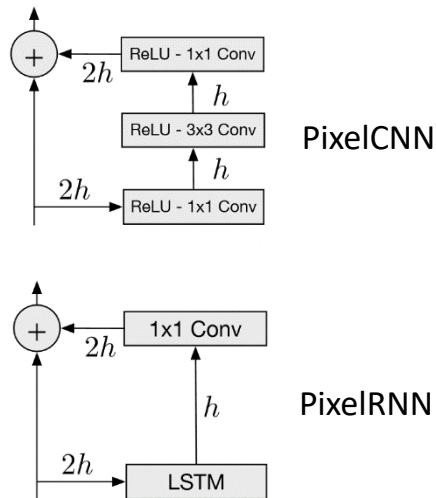


Figure from: [Oord et al.](#)

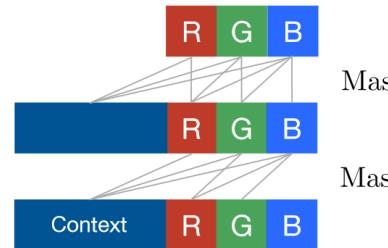
# Architecture

- Residual connections



- Channel masks

- Sequential order: R → G → B
- Used in input-to-state



- Channels are connected to themselves
  - Used in all other subsequent layers
- Channels are **not** connected to themselves
  - Only used in first layer

Figure from: [Oord et al.](#)

# Architecture

PixelCNN	Row LSTM	Diagonal BiLSTM
$7 \times 7$ conv mask A		
Multiple residual blocks: (see fig 5)		
Conv $3 \times 3$ mask B	Row LSTM i-s: $3 \times 1$ mask B s-s: $3 \times 1$ no mask	Diagonal BiLSTM i-s: $1 \times 1$ mask B s-s: $1 \times 2$ no mask
ReLU followed by $1 \times 1$ conv, mask B (2 layers)		
256-way Softmax for each RGB color (Natural images) or Sigmoid (MNIST)		

Figure from: [Oord et al.](#)

# Results

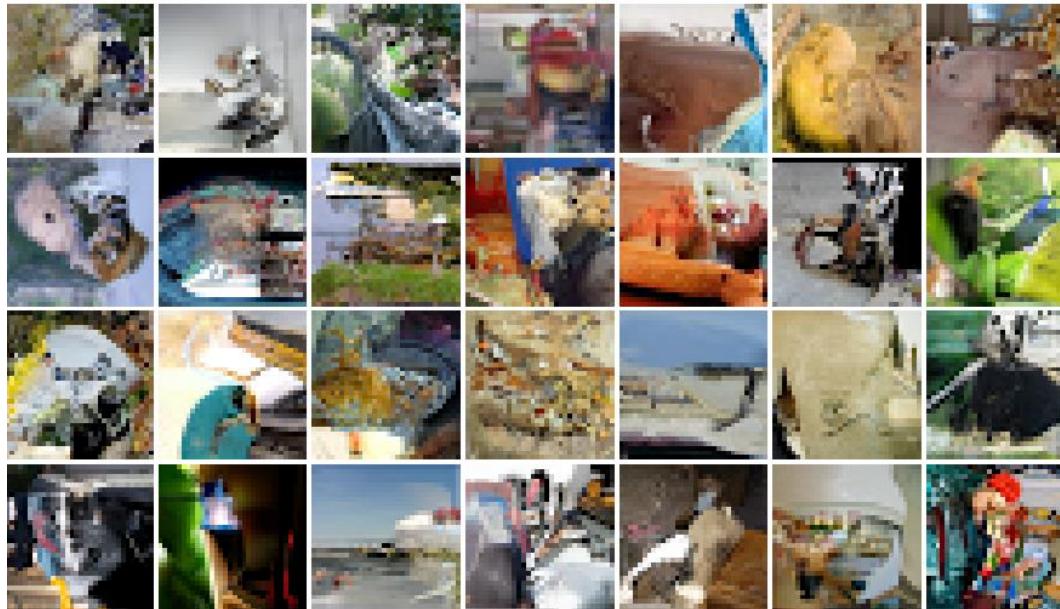


Figure: 32x32 ImageNet results from Diagonal BiLSTM model.

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

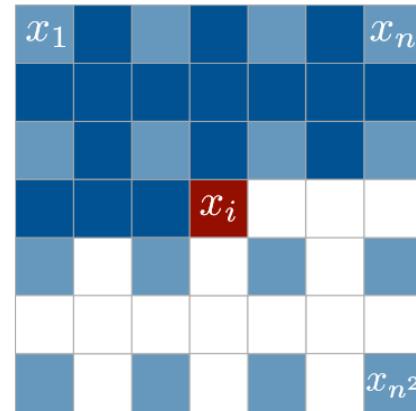


Figure from: [Oord et al.](#)

# 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  $h$**

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



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

[Conditional image generation with pixelcnn decoders](#). NIPS 2016

# Conditional Image Generation

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

Conditional image generation with pixelcnn decoders. NIPS 2016

# Conditional Image Generation

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

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

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

1x1 convolution                          1x1 convolution

[Conditional image generation with pixelcnn decoders](#). NIPS 2016

# Results



African elephant



Sandbar

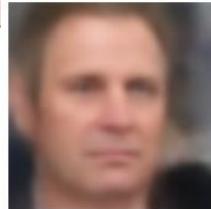
Figure from: [Oord et al.](#)

# 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
<ul style="list-style-type: none"><li>Simple and stable training process (e.g. softmax loss)</li><li>Best log likelihoods so far</li></ul>	<ul style="list-style-type: none"><li>Sharpest images</li></ul>	<ul style="list-style-type: none"><li>Easy to relate image with low-dimensional latent variables</li></ul>
<ul style="list-style-type: none"><li>Inefficient during sampling</li><li>Don't easily provide simple low-dimensional codes for images</li></ul>	<ul style="list-style-type: none"><li>Difficult to optimize due to unstable training dynamics</li></ul>	<ul style="list-style-type: none"><li>Tends to have blurry outputs</li></ul>
		

Credit: <https://openai.com/blog/generative-models/>, Oord et al. and [Larsen et al.](#)

# Summary

---

- Deep learning tools can be put together in many different innovative ways to obtain interesting results for different applications
  - PixelRNN, Deep generative model for conceptual compression
- We essentially studied extensions of “one-shot” image generation techniques to include some feedback in them using RNNs
  - Such extension is possible for GAN-based generation as well

# 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 !



WeChat Group for Deep Learning