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

2019

"Deep Learning Lecture"

Lecture 8: 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: Generative Model

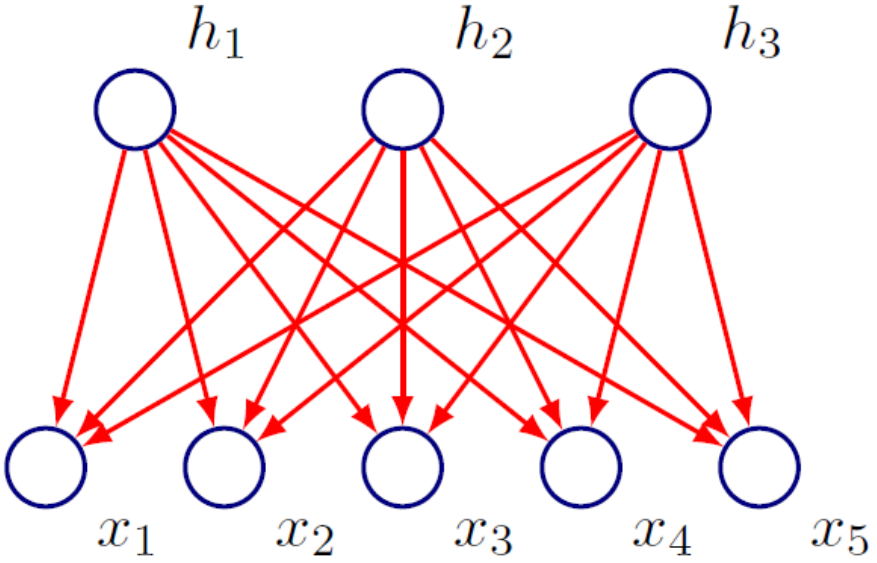
- We want to build a probabilistic model of the input $\tilde{P}(\mathbf{x})$
- Like before, we are interested in latent factors **h** that explain **x**
- We then care about the marginal:

$$\tilde{P}(\mathbf{x}) = \mathbb{E}_{\mathbf{h}} \tilde{P}(\mathbf{x}|\mathbf{h})$$

- **h** is a representation of the data

Review: Linear Factor Model

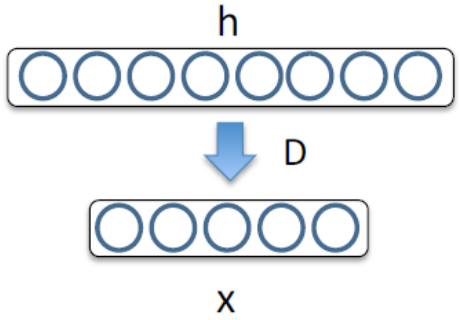
- The latent factor **h** is an encoding of the data
- Simplest decoding model: Get **x** after a linear transformation of **h** with some noise
- Formally: Suppose we sample the latent factors from a distribution $\mathbf{h} \sim P(\mathbf{h})$
- Then: **x** = *W***h** + **b** + ε



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

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 good 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

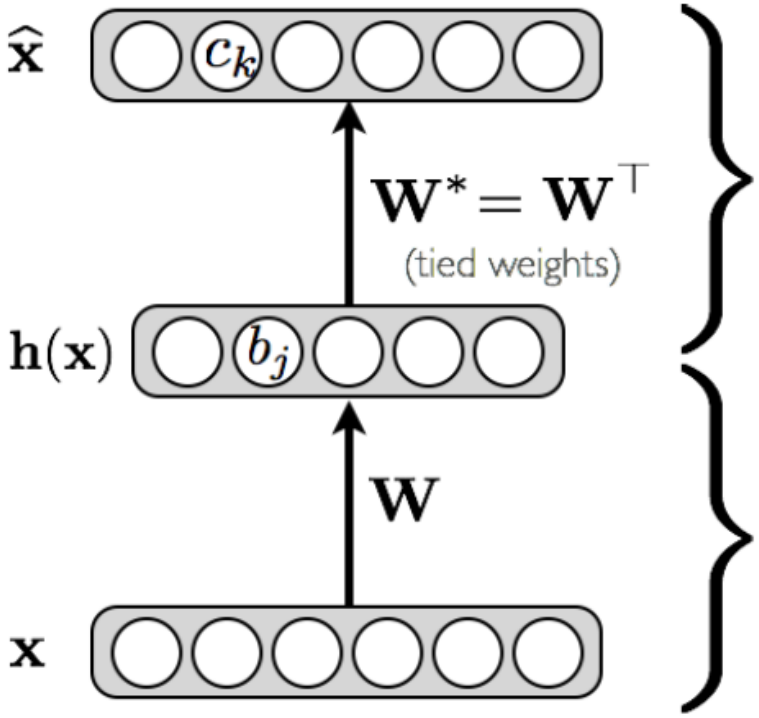
$$\min_{\mathbf{D}} \frac{1}{T} \sum_{t=1}^{T} \min_{\mathbf{h}^{(t)}} \frac{1}{2} ||\mathbf{x}^{(t)} - \mathbf{D}\,\mathbf{h}^{(t)}||_2^2 + \lambda ||\mathbf{h}^{(t)}||_1$$

Reconstruction error

Sparsity penalty

Review: Autoencoder

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



Decoder

$$\hat{\mathbf{x}} = o(\hat{\mathbf{a}}(\mathbf{x})) = \mathrm{sigm}(\mathbf{c} + \mathbf{W}^*\mathbf{h}(\mathbf{x}))$$

For binary units

Encoder

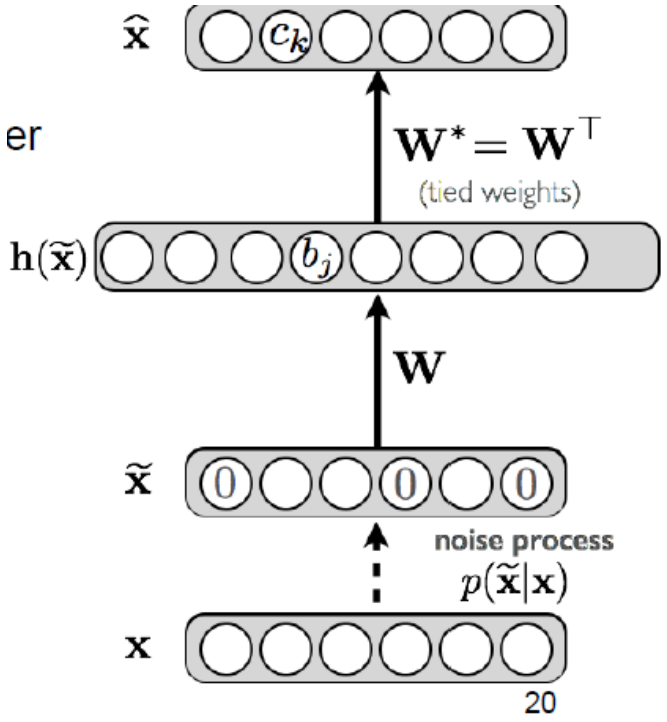
$$\mathbf{h}(\mathbf{x}) = g(\mathbf{a}(\mathbf{x})) = \mathrm{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})$$

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)

Outline

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

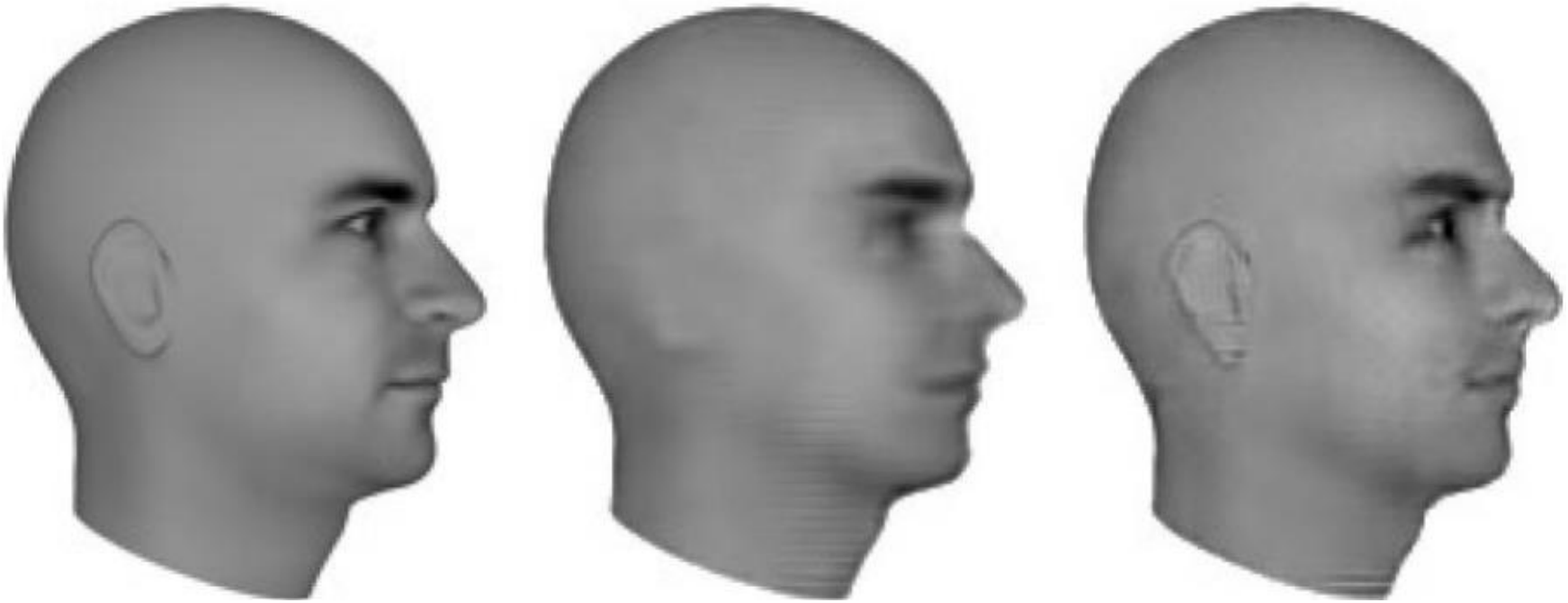
Generative Adversarial Networks

- **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 **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
  - Thus, 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

Magic of GANs



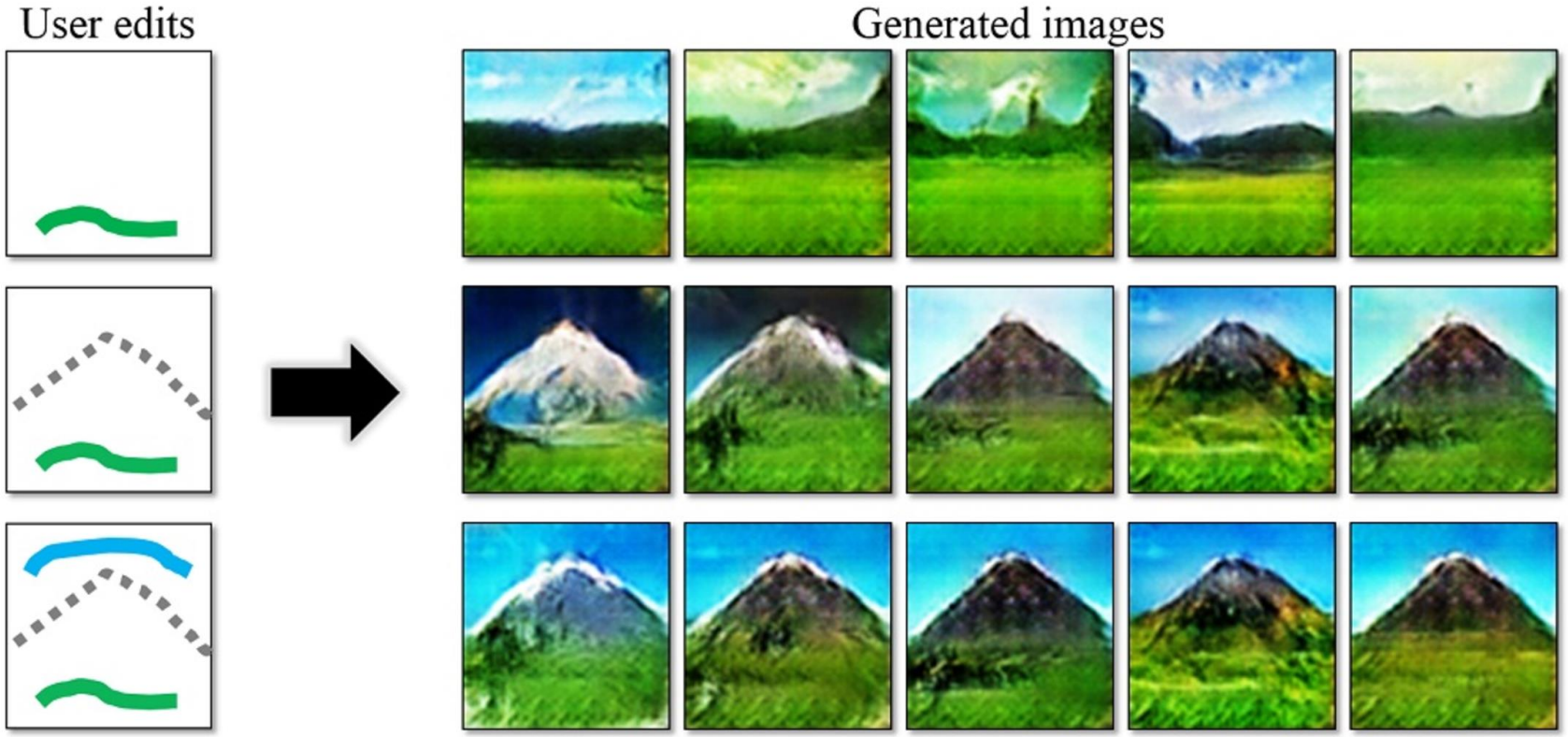
Lotter, William, Gabriel Kreiman, and David Cox. "Unsupervised learning of visual structure using predictive generative networks." *arXiv preprint arXiv:1511.06380* (2015).

Magic of GANs

Which one is 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



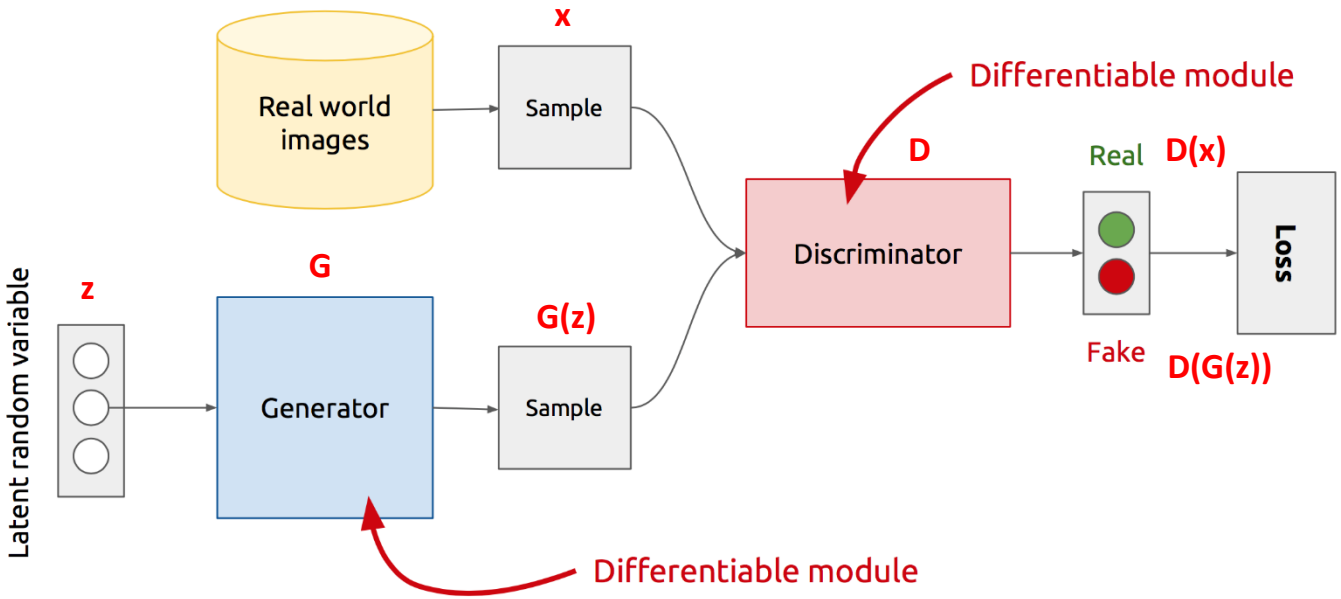
http://people.eecs.berkeley.edu/~junyanz/projects/gvm/

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

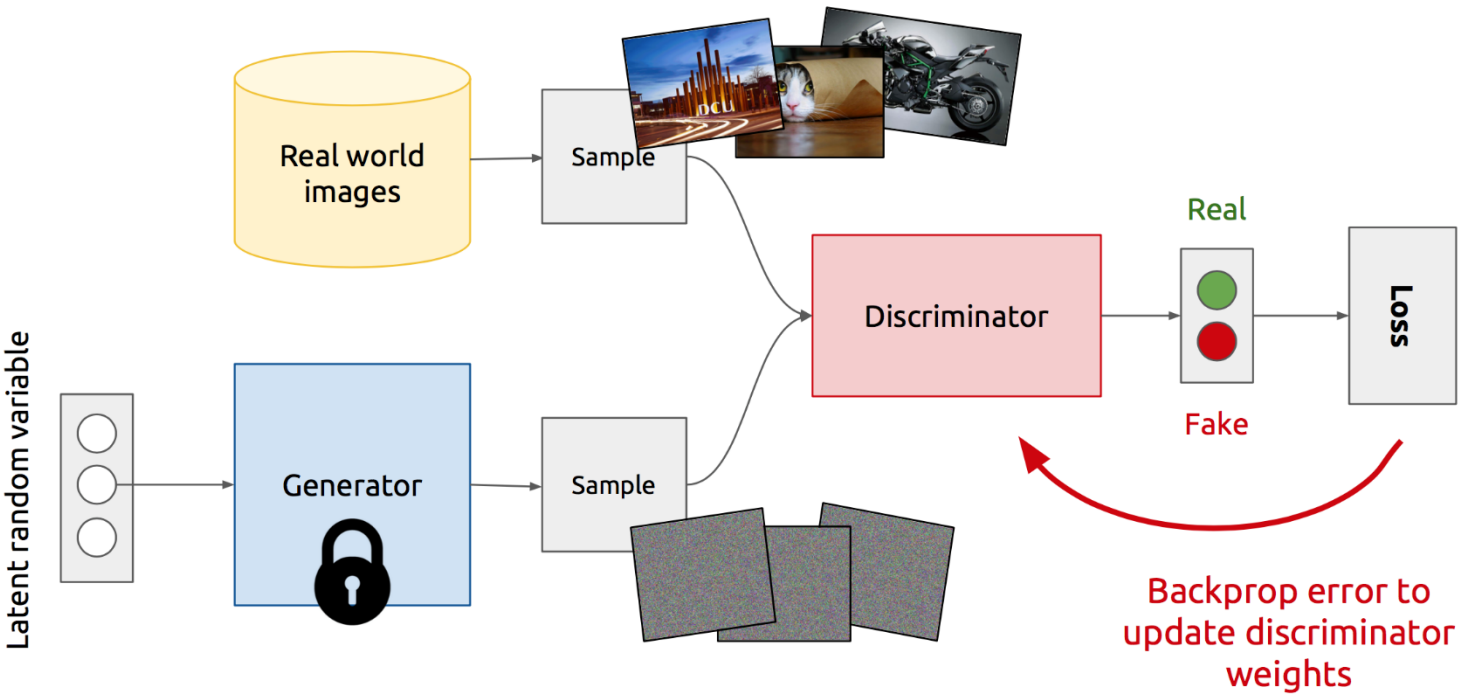
GAN's Architecture



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

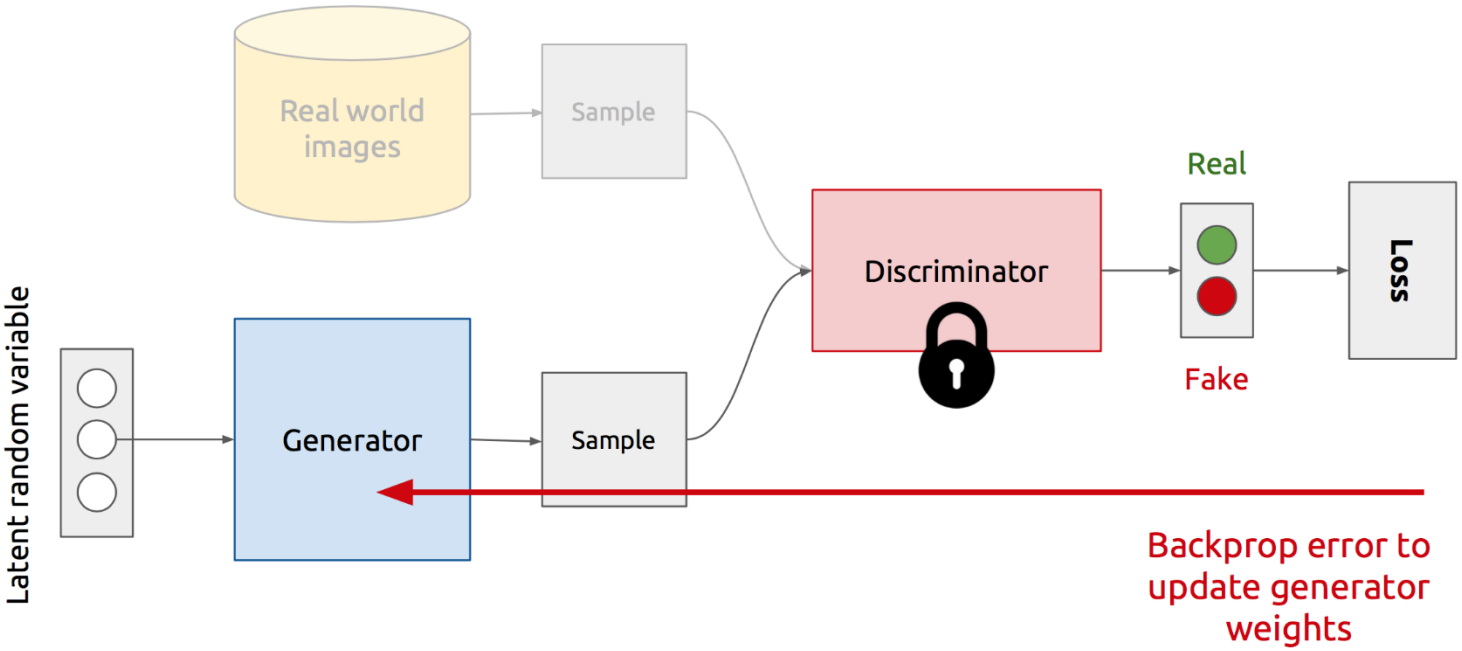
https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

Training Discriminator



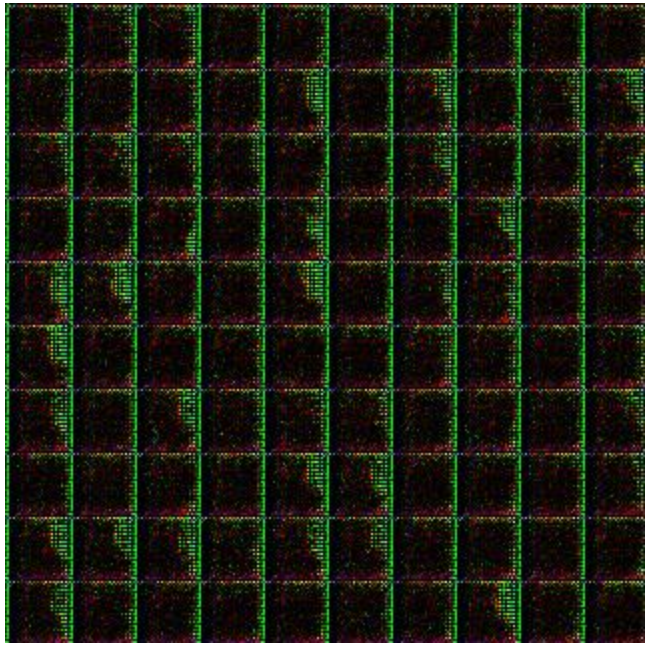
https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

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

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

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

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.

Vanishing gradient strikes back again

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

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

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

- $\nabla_a \log\left(1-\sigma(a)\right) = \frac{-\nabla_a \sigma(a)}{1-\sigma(a)} = \frac{-\sigma(a)\left(1-\sigma(a)\right)}{1-\sigma(a)} = -\sigma(a) = -D\left(G(z)\right)$
- 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)

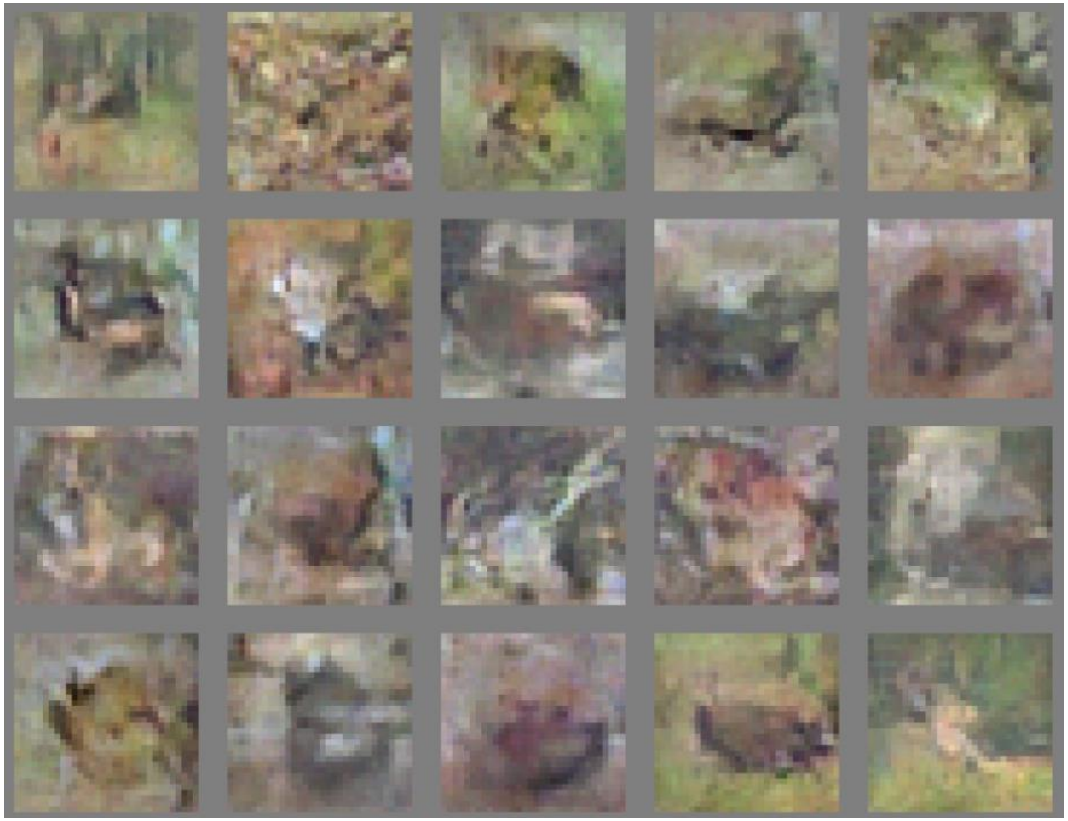
Applications

Faces

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

Applications

CIFAR



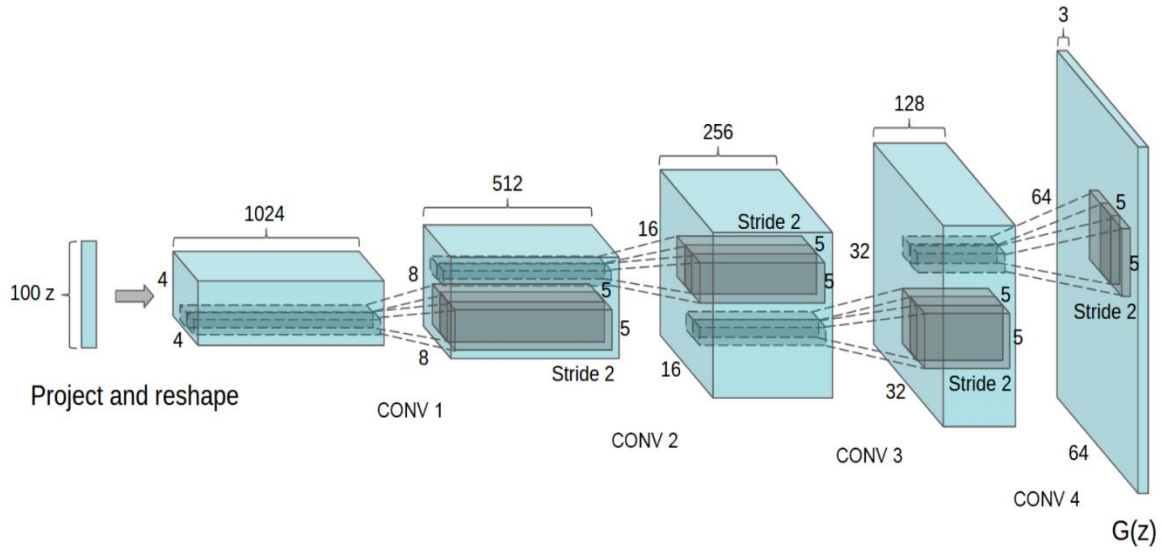
Goodfellow, Ian, et al. **"Generative adversarial nets."** *Advances in neural information processing systems*. 2014.

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



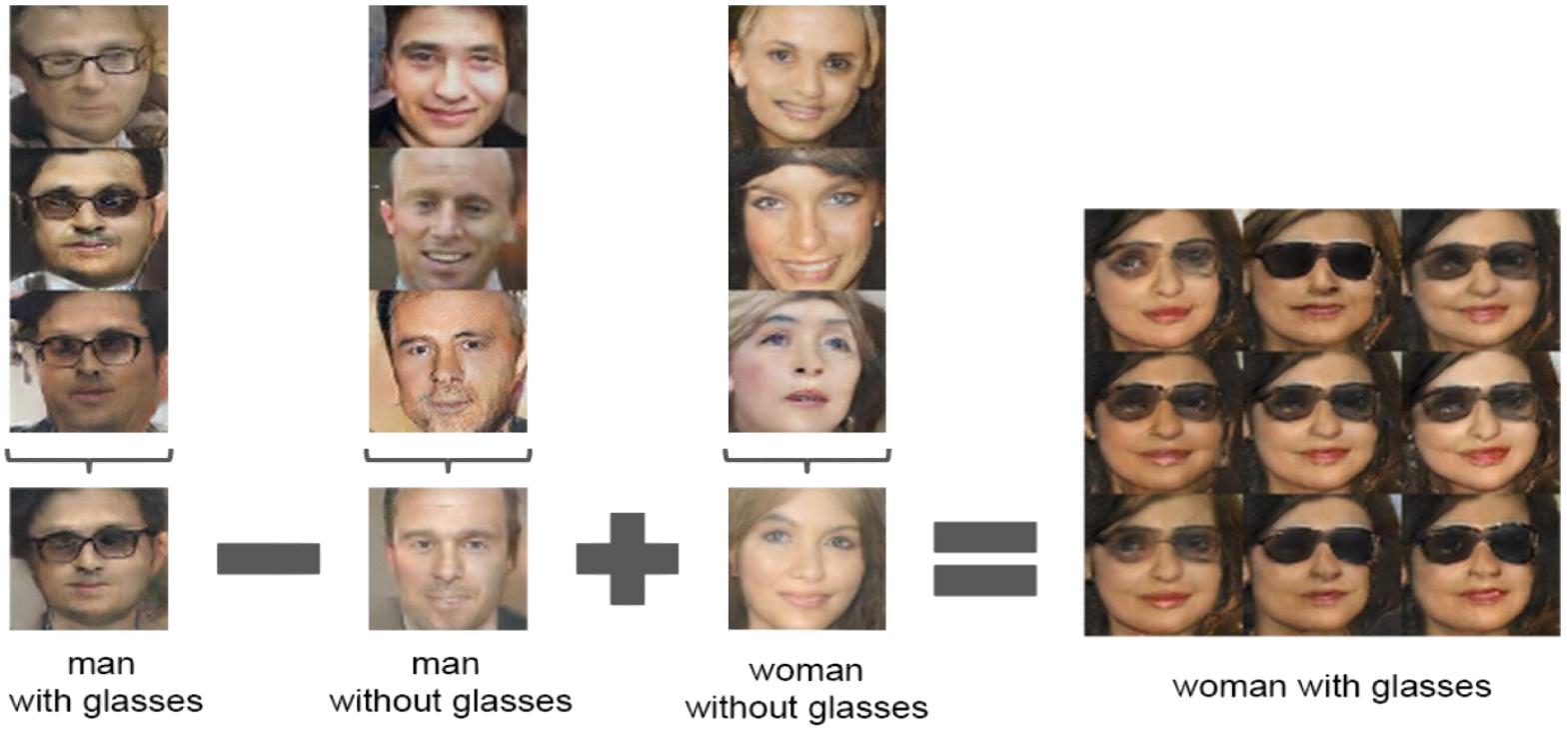
Key ideas:

- 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 $V(x,y) = xy$

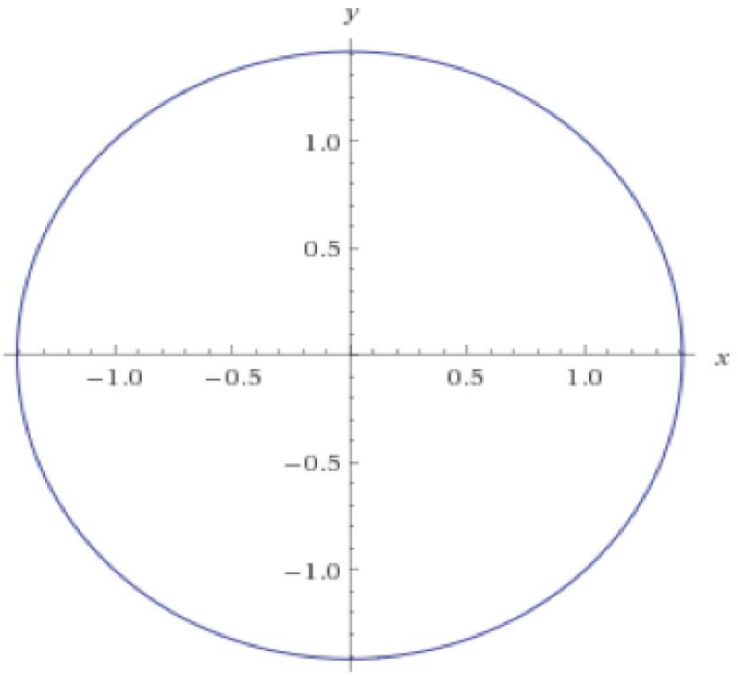
– State 1	x > 0	y > 0	V > 0	Increase y	Decrease x
– State 2	x < 0	y > 0	V < 0	Decrease y	Decrease x
– State 3	x < 0	y < 0	V > 0	Decrease y	Increase x
– State 4	x > 0	y < 0	V < 0	Increase y	Increase x
– State 5	x > 0	y > 0	V > 0	Increase y	Decrease x

== State 1

Non-Convergence

$$\min_x \max_y \; xy$$

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



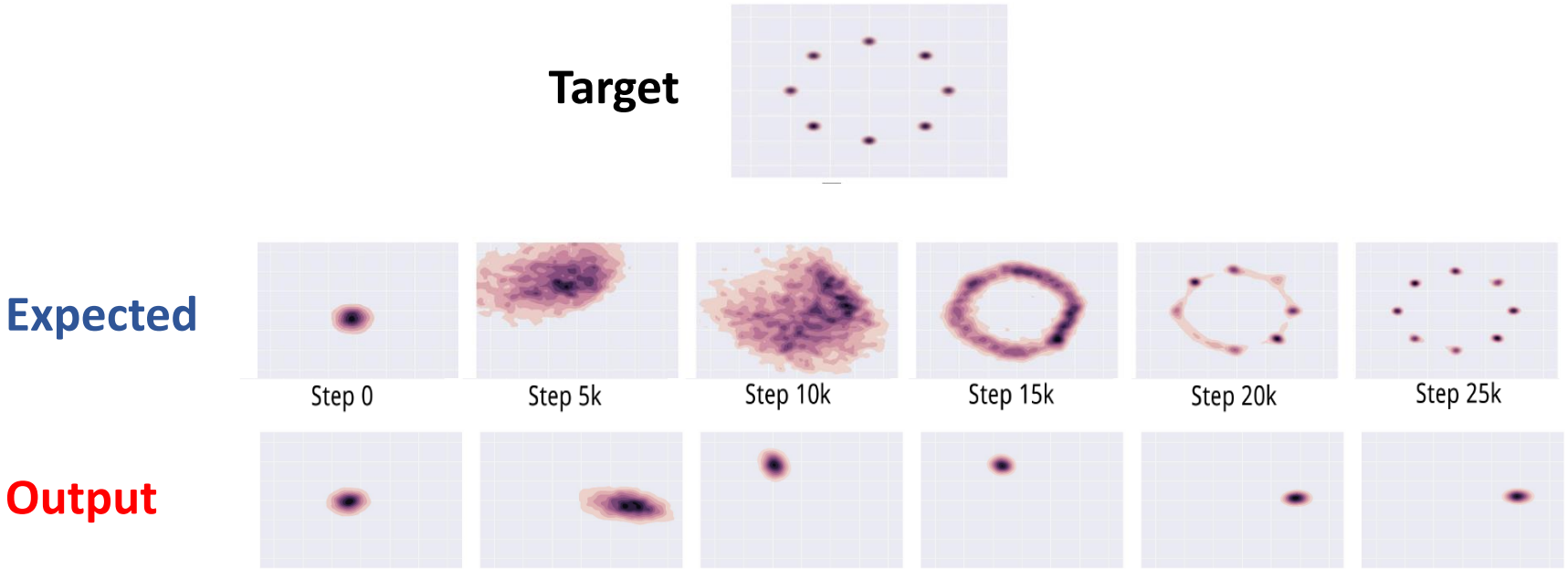
Goodfellow, Ian. **"NIPS 2016 Tutorial: Generative Adversarial Networks."** arXiv preprint arXiv:1701.00160 (2016).

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

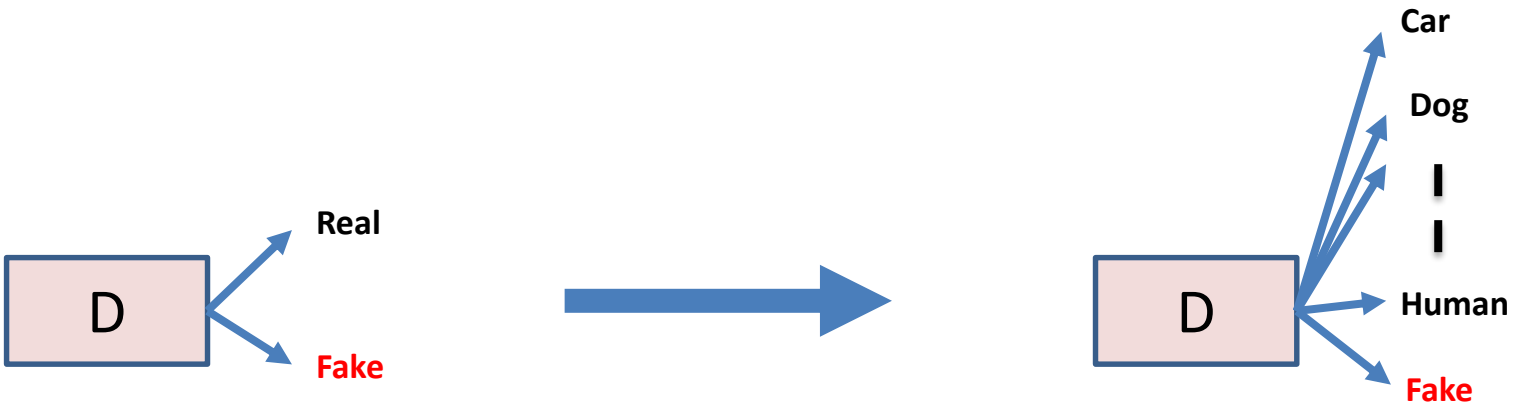
Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

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

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

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

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

- In this formulation, Discriminator's strategy was $D(x) \to 1, \quad D\left(G(z)\right) \to 0$

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

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

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

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

Alternate view of GANs (Contd.)

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

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

We can use this

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

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

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

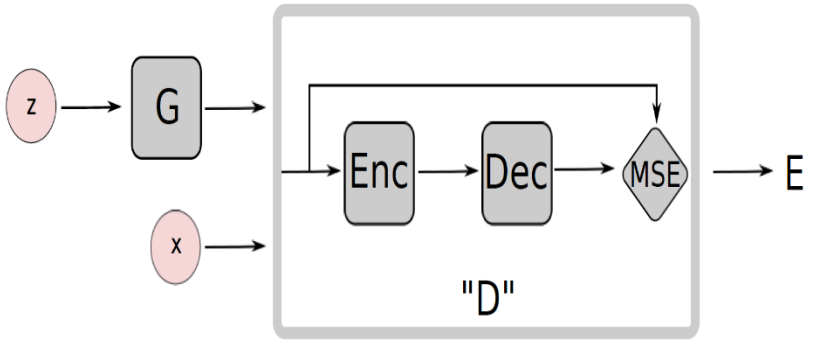
- And thus, instead of outputting probabilities, Discriminator just has to output :-
  - High values for fake samples
  - Low values for real samples

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

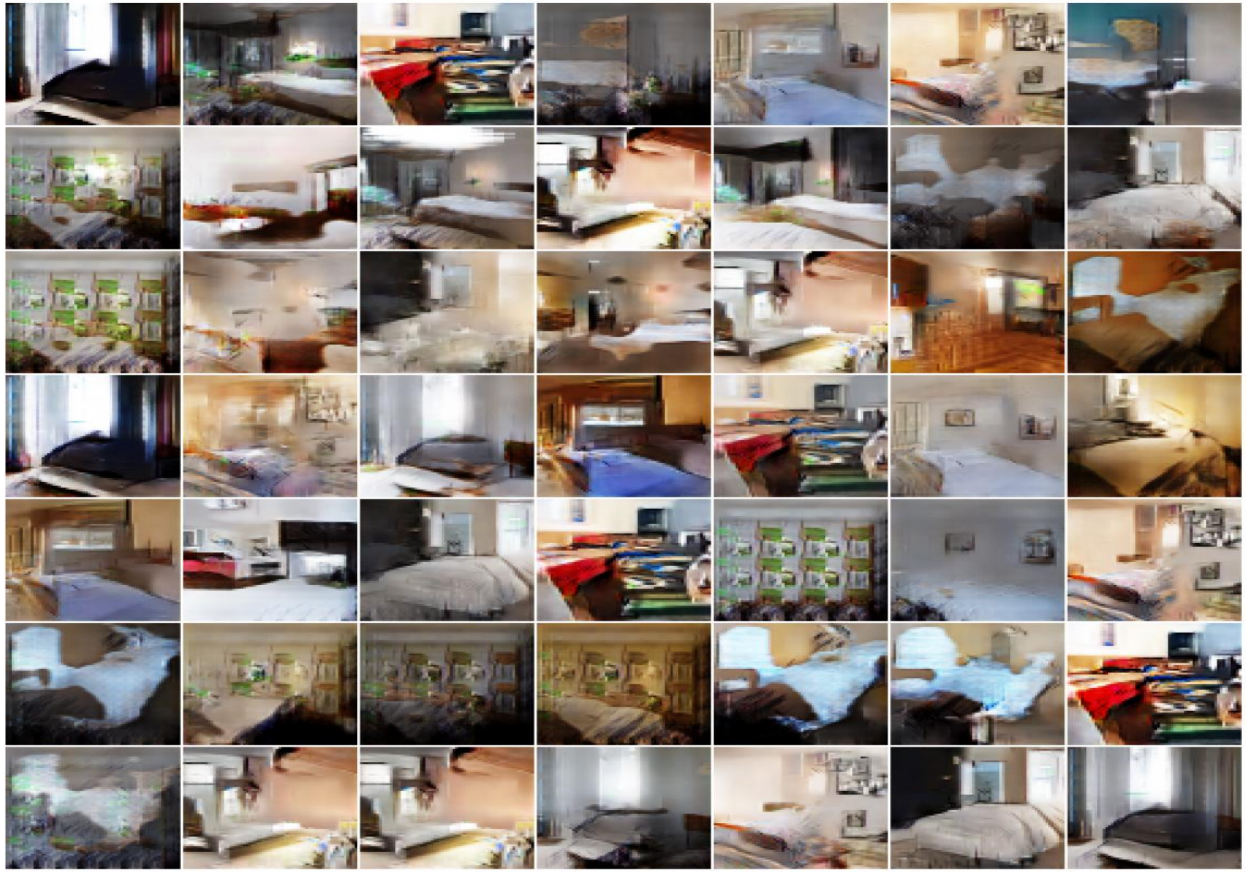
$$D(x) = ||Dec(Enc(x)) - x||_{MSE}$$



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

Examples

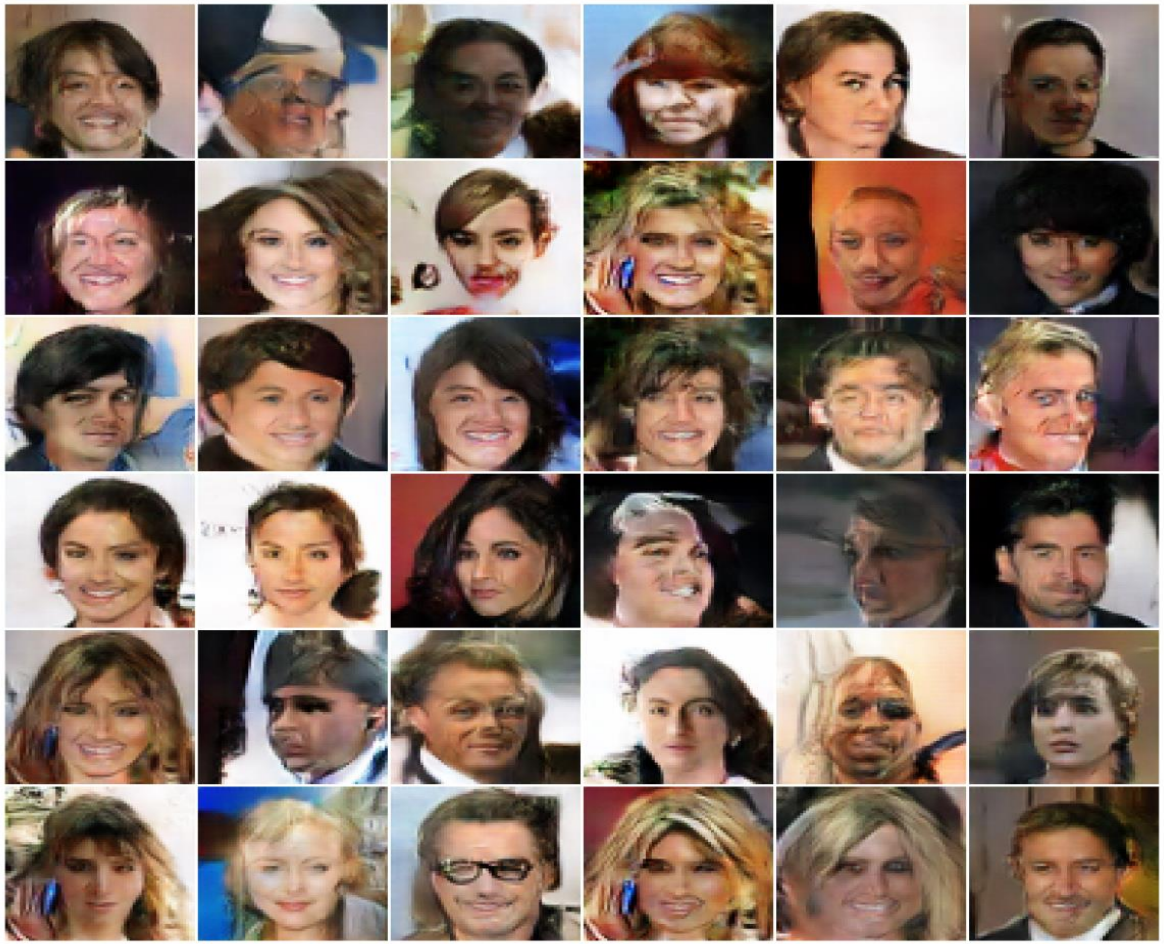
More Bedrooms…



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

Examples

Celebs…



Examples

Cool Stuff (contd.)

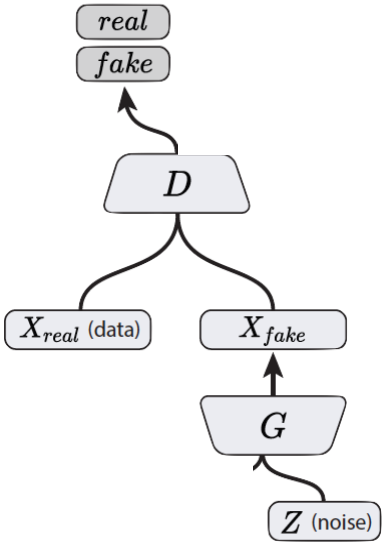
3D Chairs

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?

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

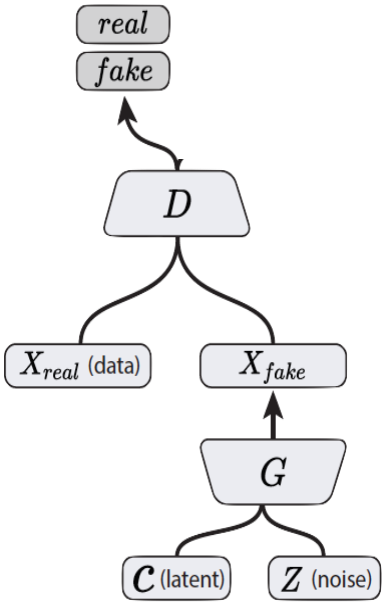
- Mutual Information captures the mutual dependence between two variables
- Mutual information between two variables X, Y is defined as:

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

$$I(X;Y) = H(X) - H(X|Y) = \mathrm{H(Y)} - \mathrm{H(Y|X)}$$

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.



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

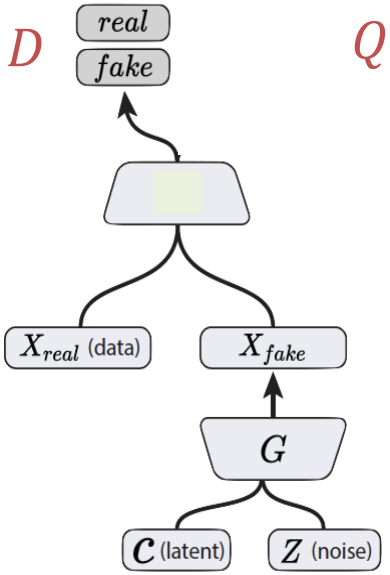
Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. InfoGAN: Interpretable Representation Learning by Information Maximization Generative Adversarial Nets, NIPS (2016).

InfoGAN

Mutual Information's Variational Lower bound

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

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



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.

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

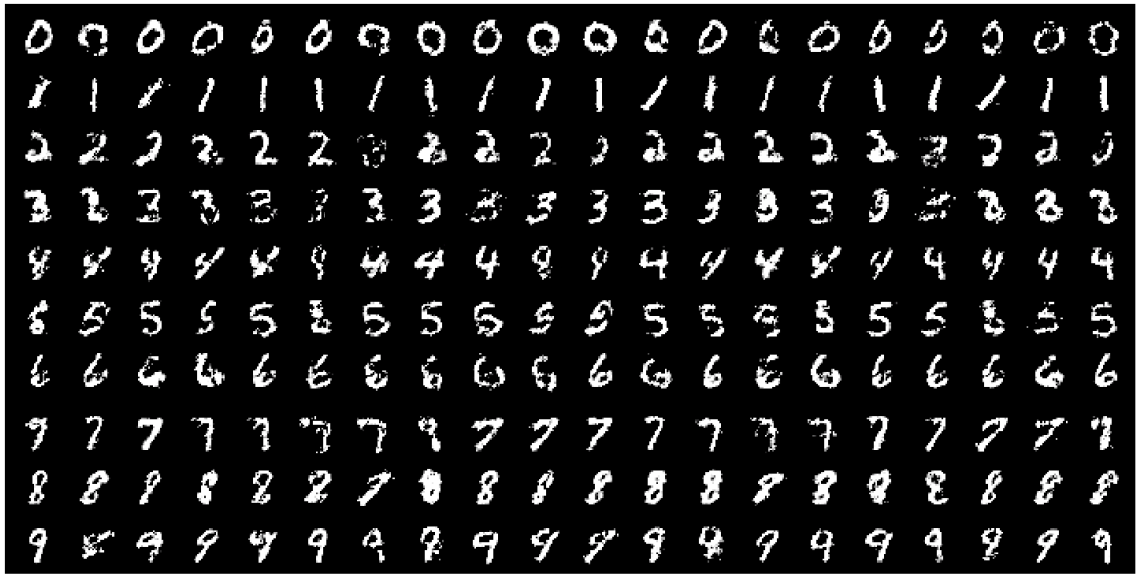


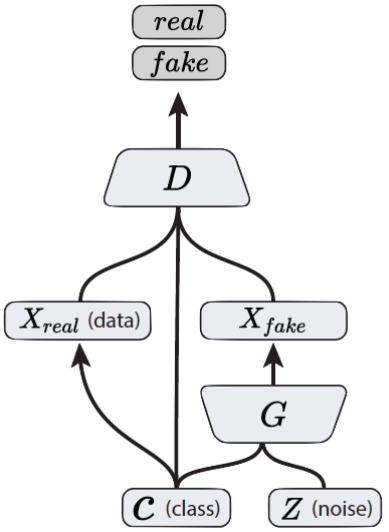
Figure 2 in the original paper.

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 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

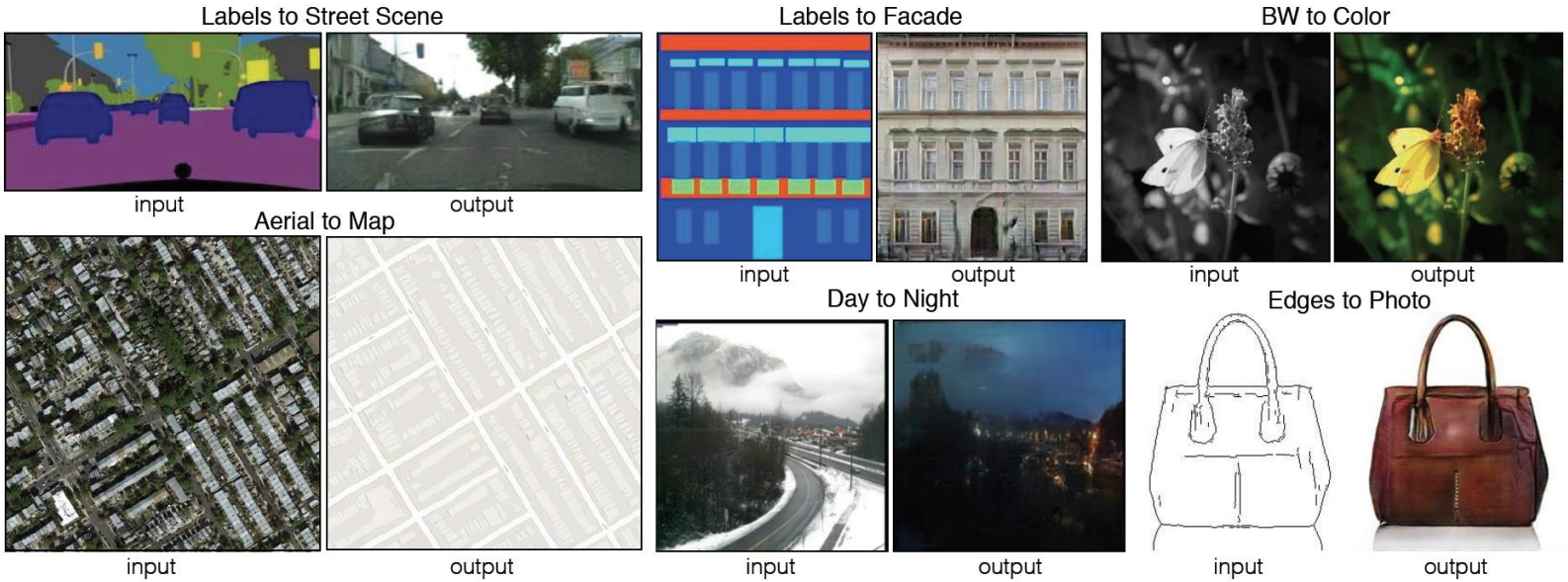


Figure 1 in the original paper.

[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.

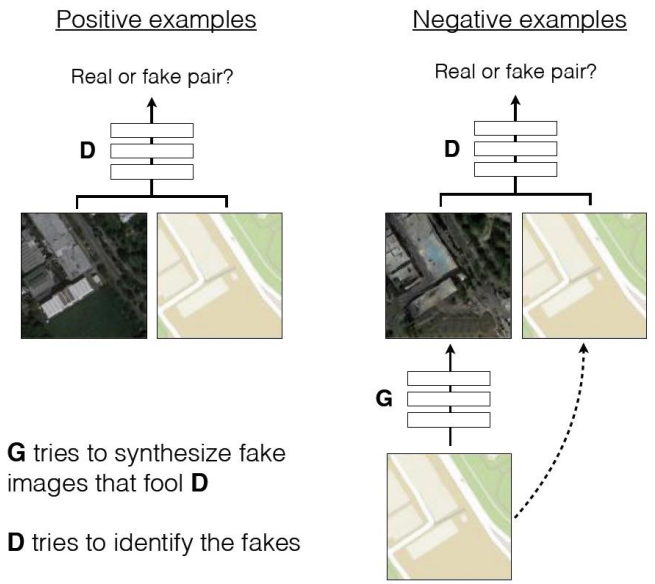


Figure 2 in the original paper.

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.

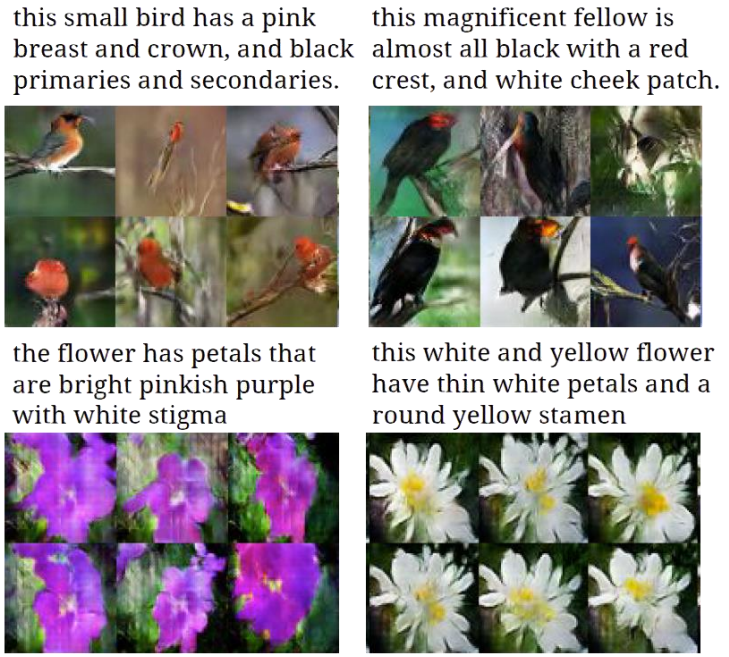


Figure 1 in the original paper.

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

Text-to-Image Synthesis

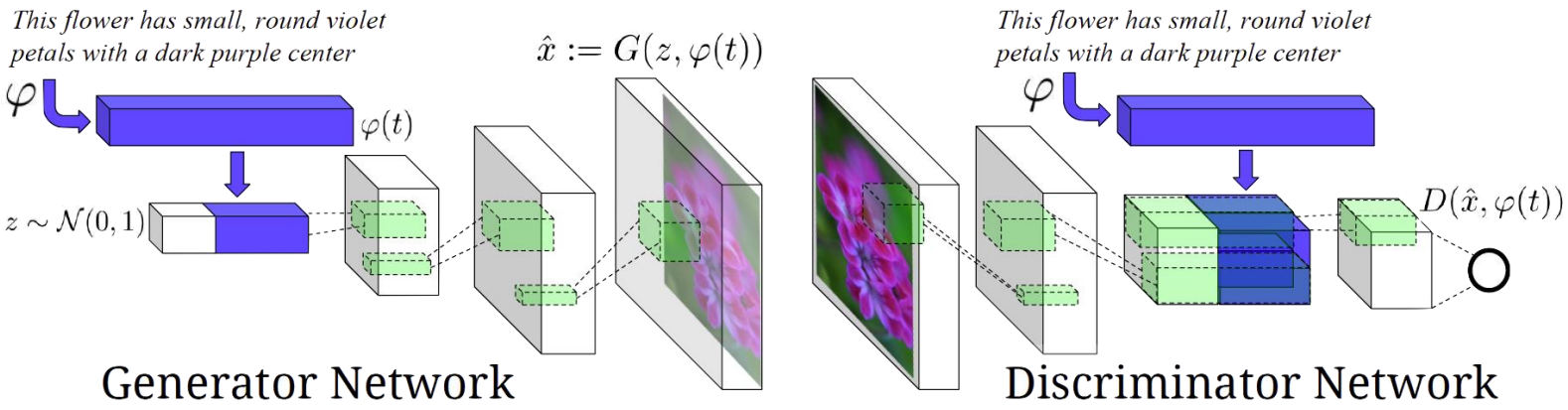


Figure 2 in the original paper.

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

- 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.

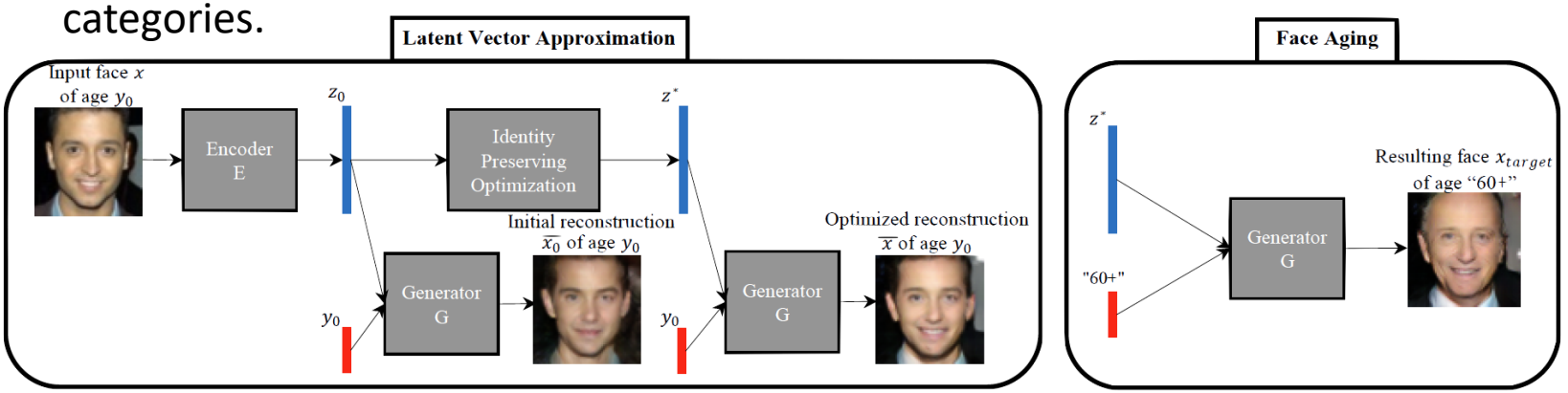


Figure 1 in the original paper.

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

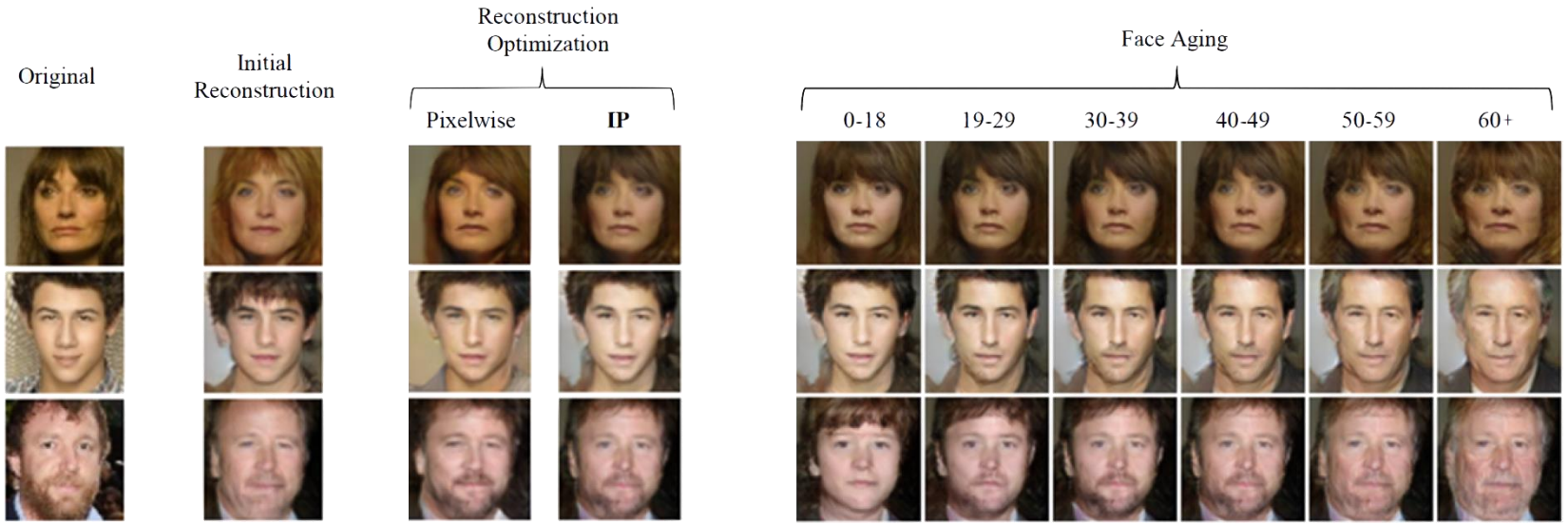


Figure 3 in the original paper.

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

Coupled GAN

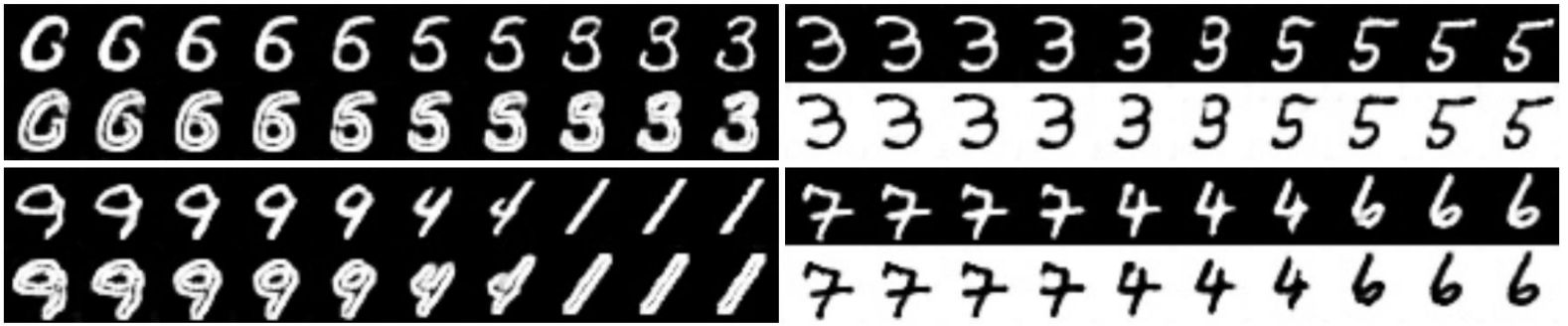


Figure 2 in the original paper.

- 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

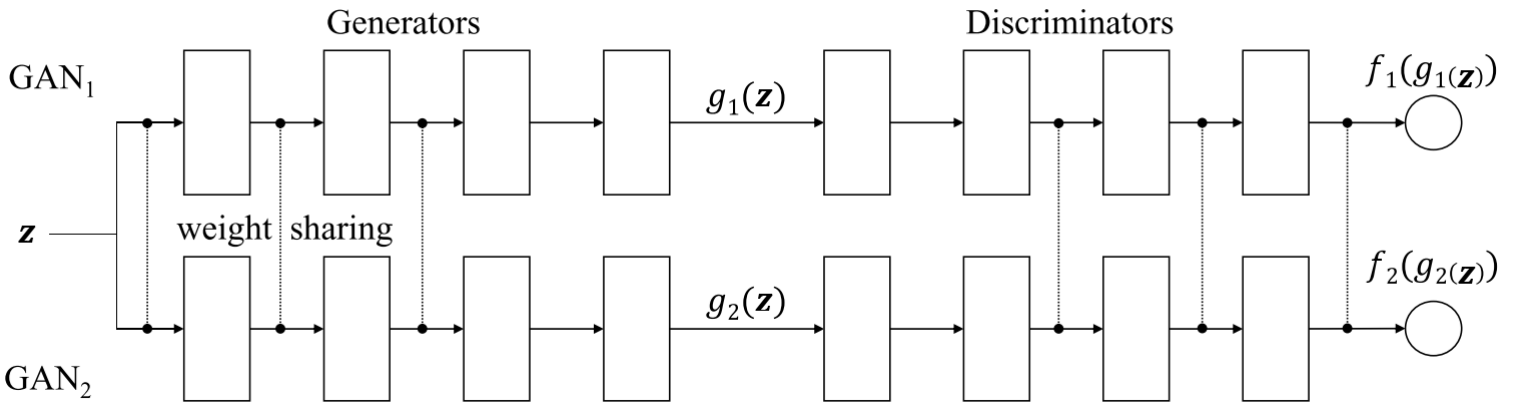


Figure 1 of the original paper.

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 generate from the same latent code z

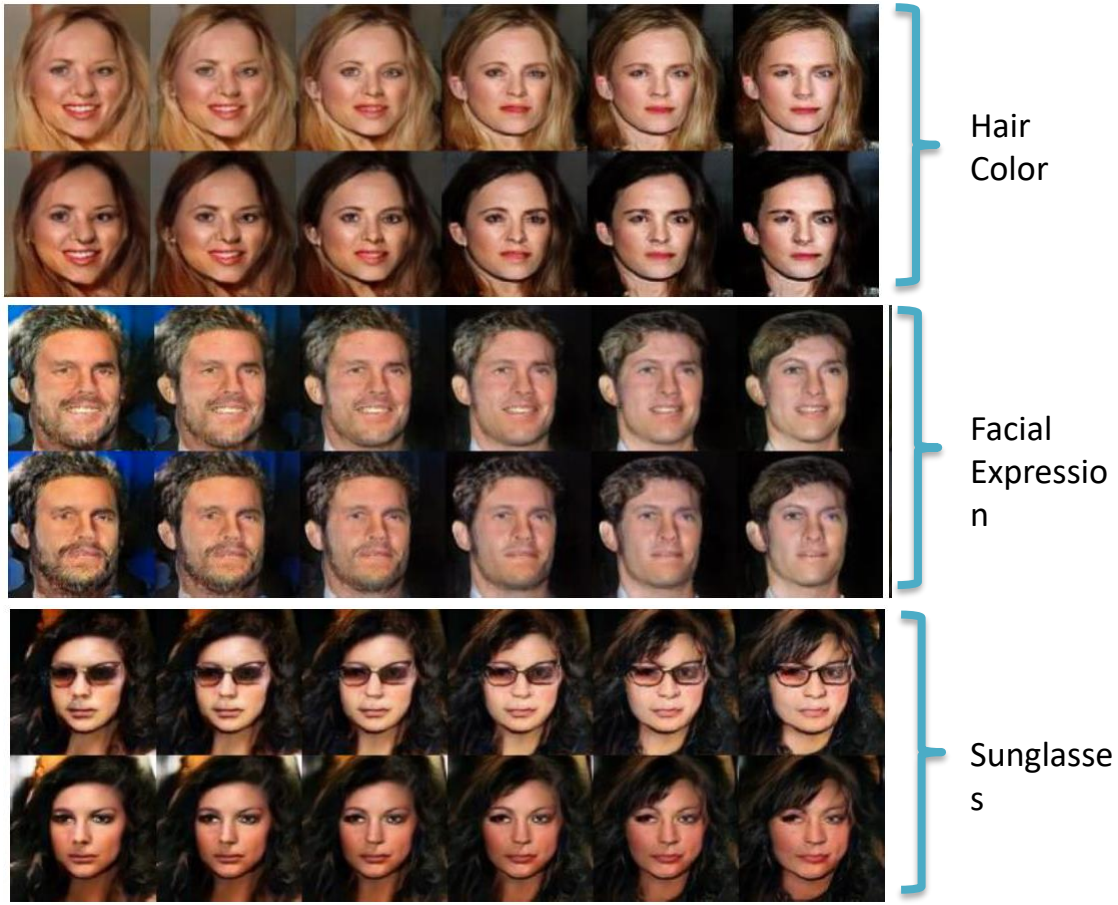


Figure 4 in the original paper.

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

Laplacian Pyramid of Adversarial Networks

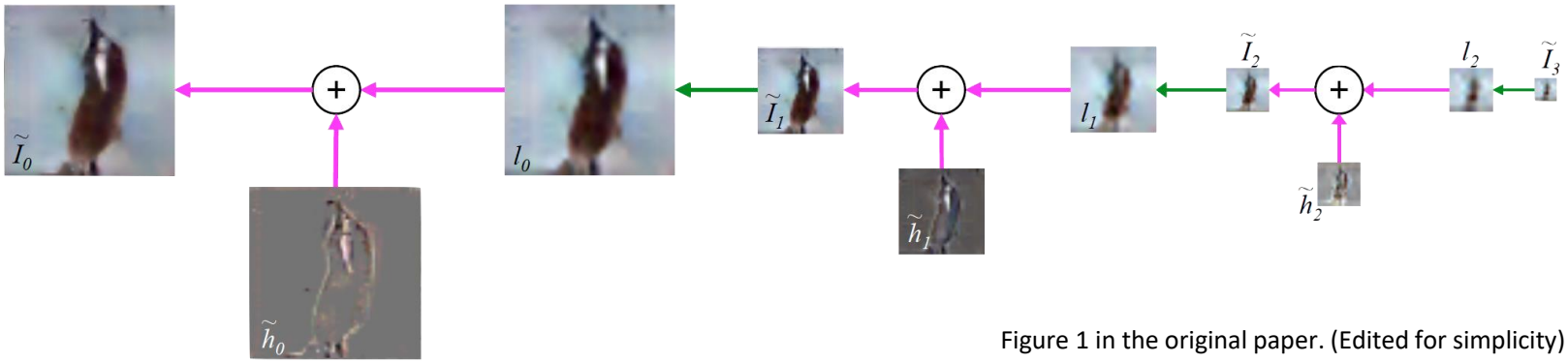


Figure 1 in the original paper. (Edited for simplicity)

- 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

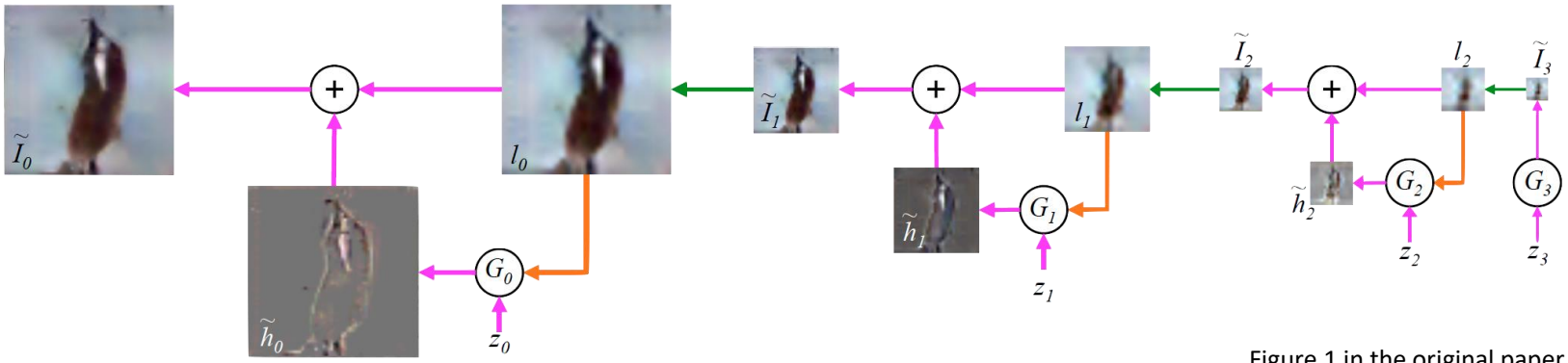


Figure 1 in the original paper.

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

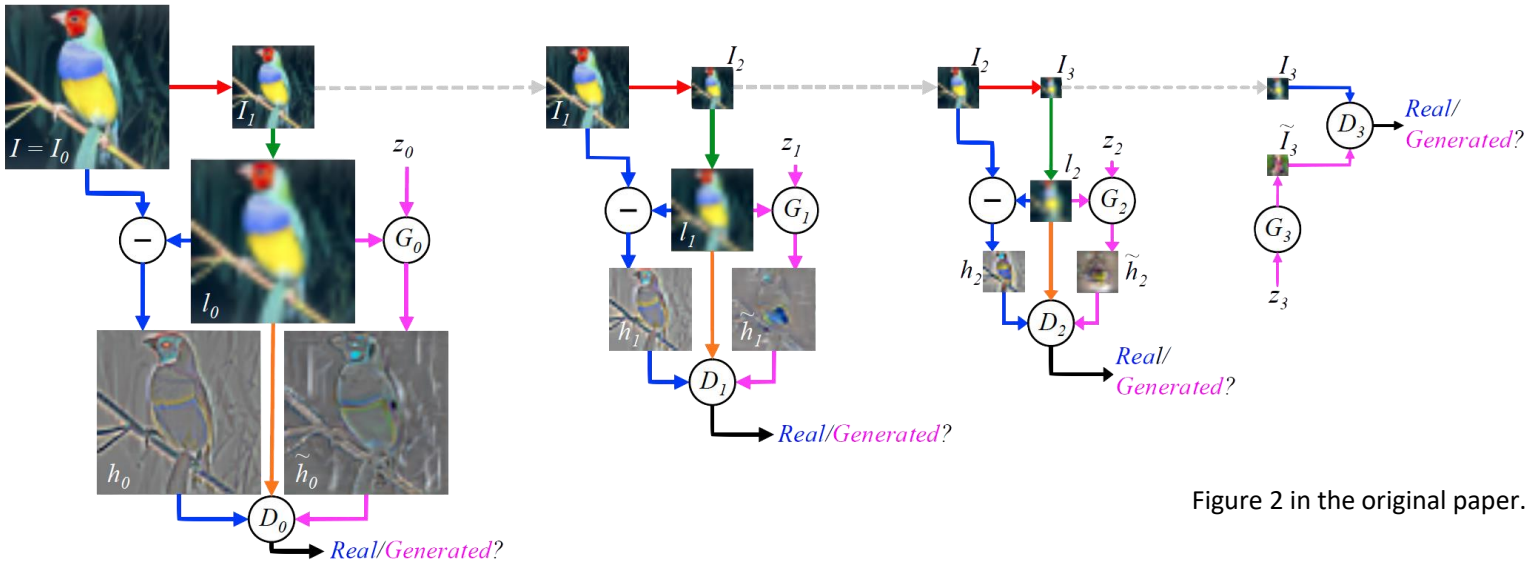


Figure 2 in the original paper.

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.

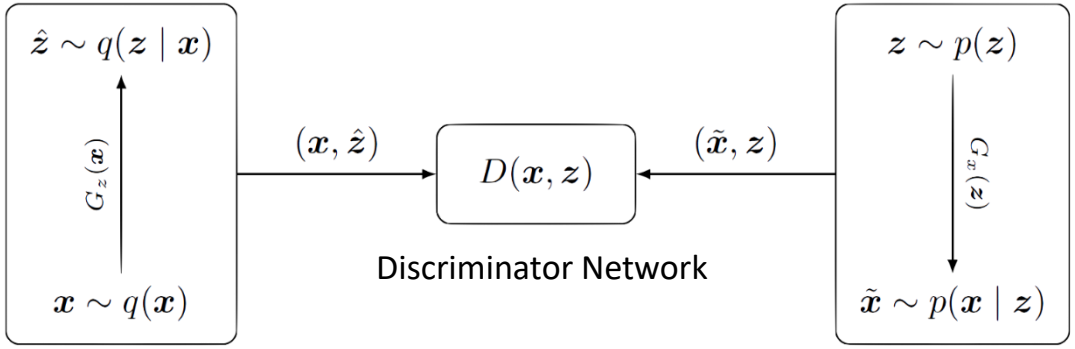
- Consider the following joint distributions over x (image) and z (latent variables) :

$$q(x, z) = q(x)\, q(z|x)$$ 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



Encoder/Inference Network

Generator Network

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.

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 (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

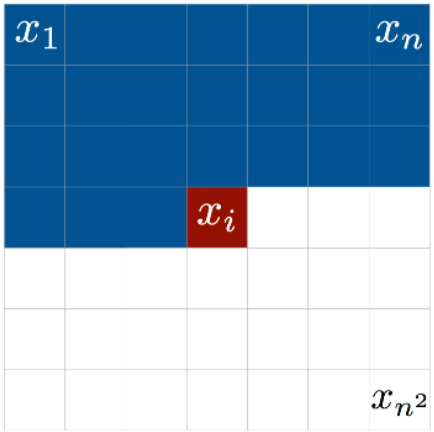
1. Course Review
2. Generative Adversarial Networks
3. Variational Autoencoder
4. Advanced Generation Methods

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

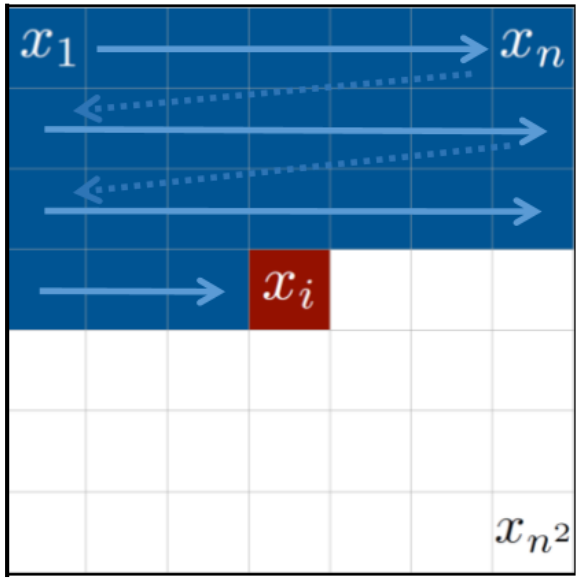


Pixel-by-pixel Generation

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



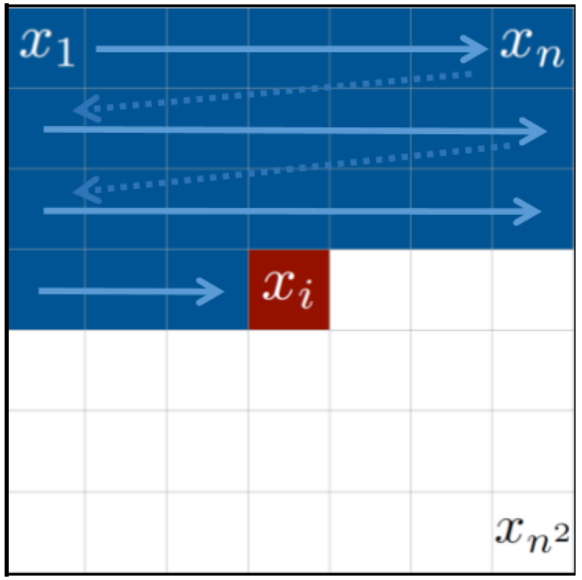
Pixel recurrent neural networks, ICML 2016

Intuition

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

Bayes Theorem:

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



Pixel recurrent neural networks, ICML 2016

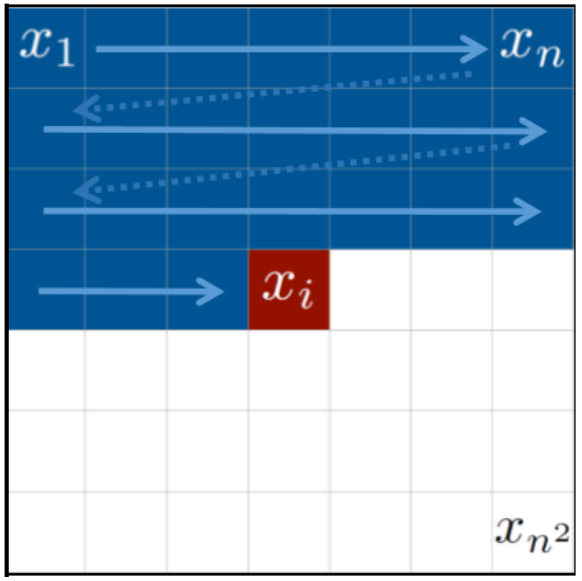
Intuition

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

Bayes Theorem:

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

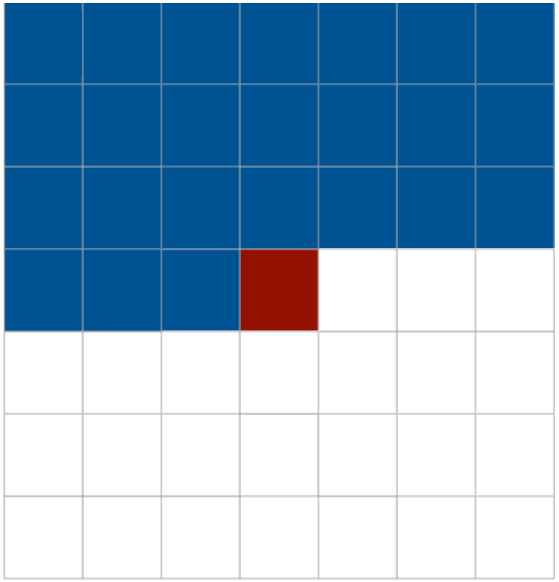
A sequential model!



Pixel recurrent neural networks, ICML 2016

Intuition

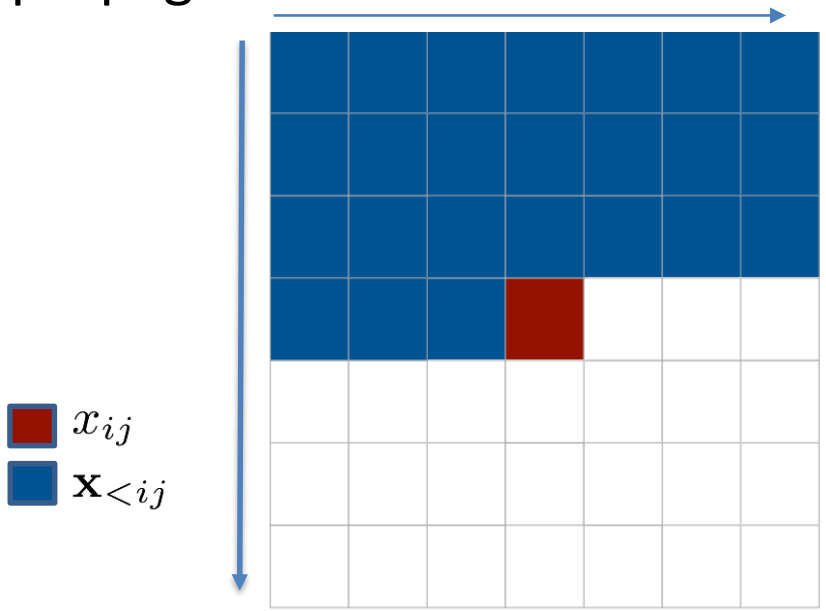
- Question: Can we use plain-LSTM to generate images pixels by pixels?



Pixel recurrent neural networks, ICML 2016

Intuition

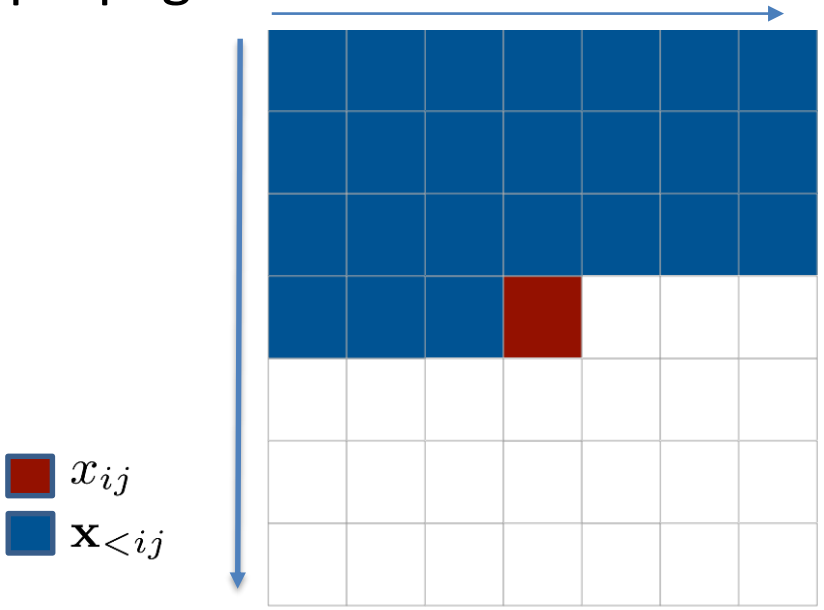
- Question: Can we use plain-LSTM to generate images pixels by pixels?
- Ensure information is well propagated in two dimensions



Pixel recurrent neural networks, ICML 2016

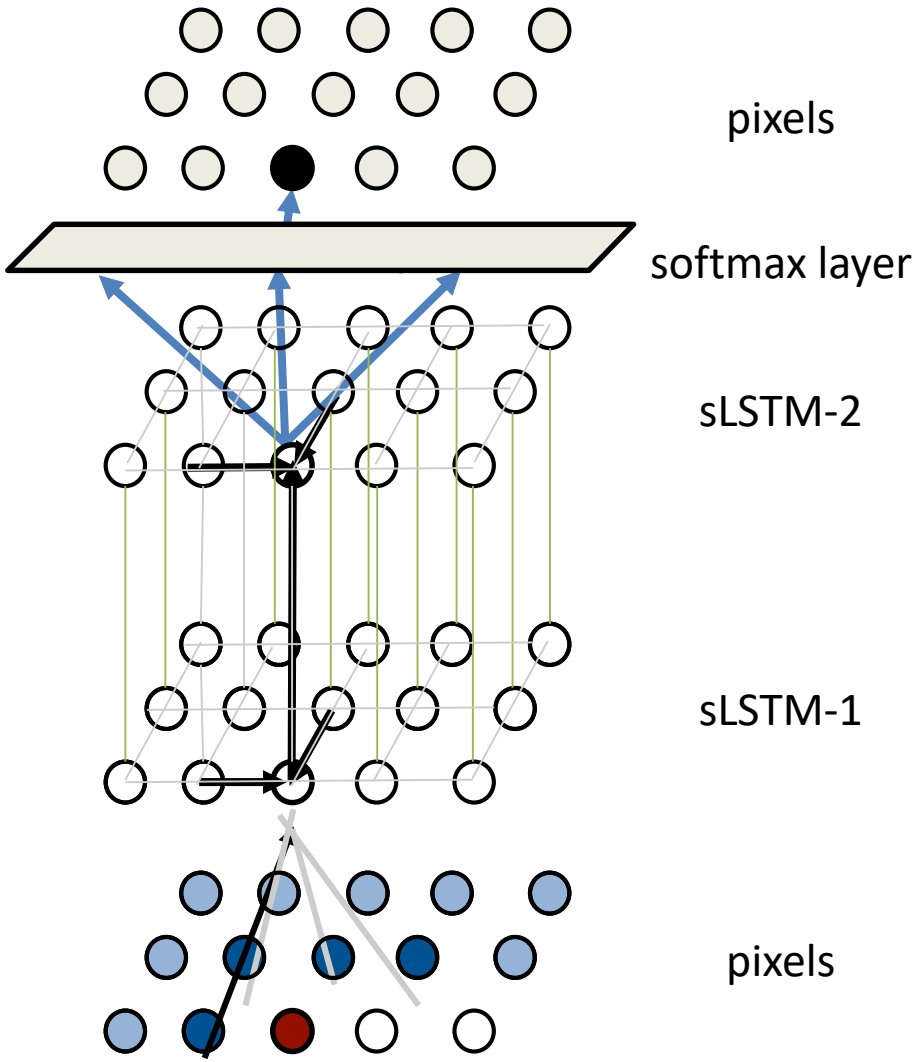
Intuition

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



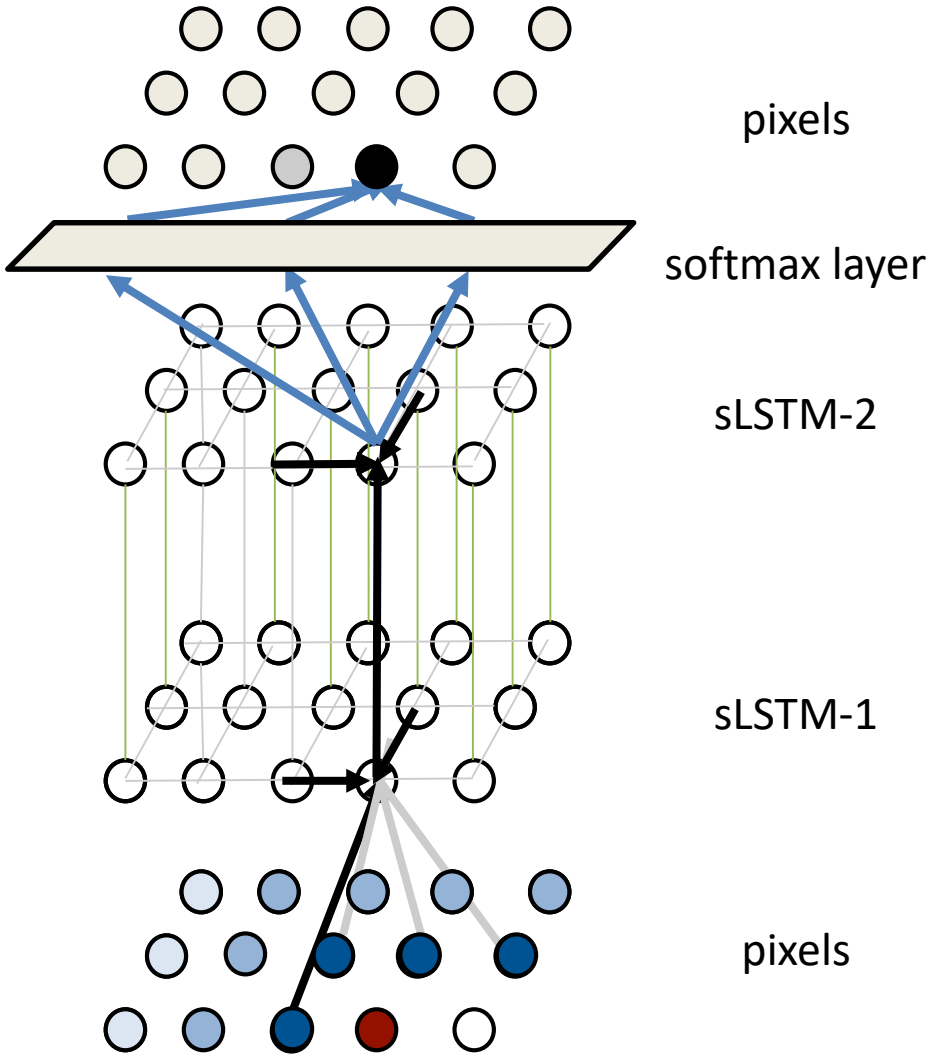
Pixel recurrent neural networks, ICML 2016

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

- 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

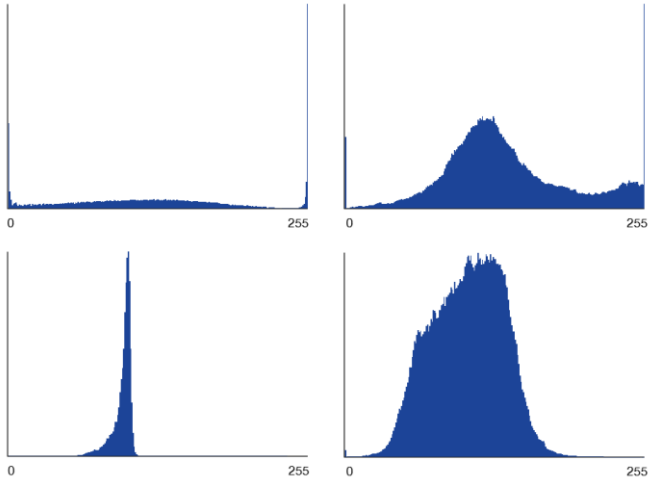
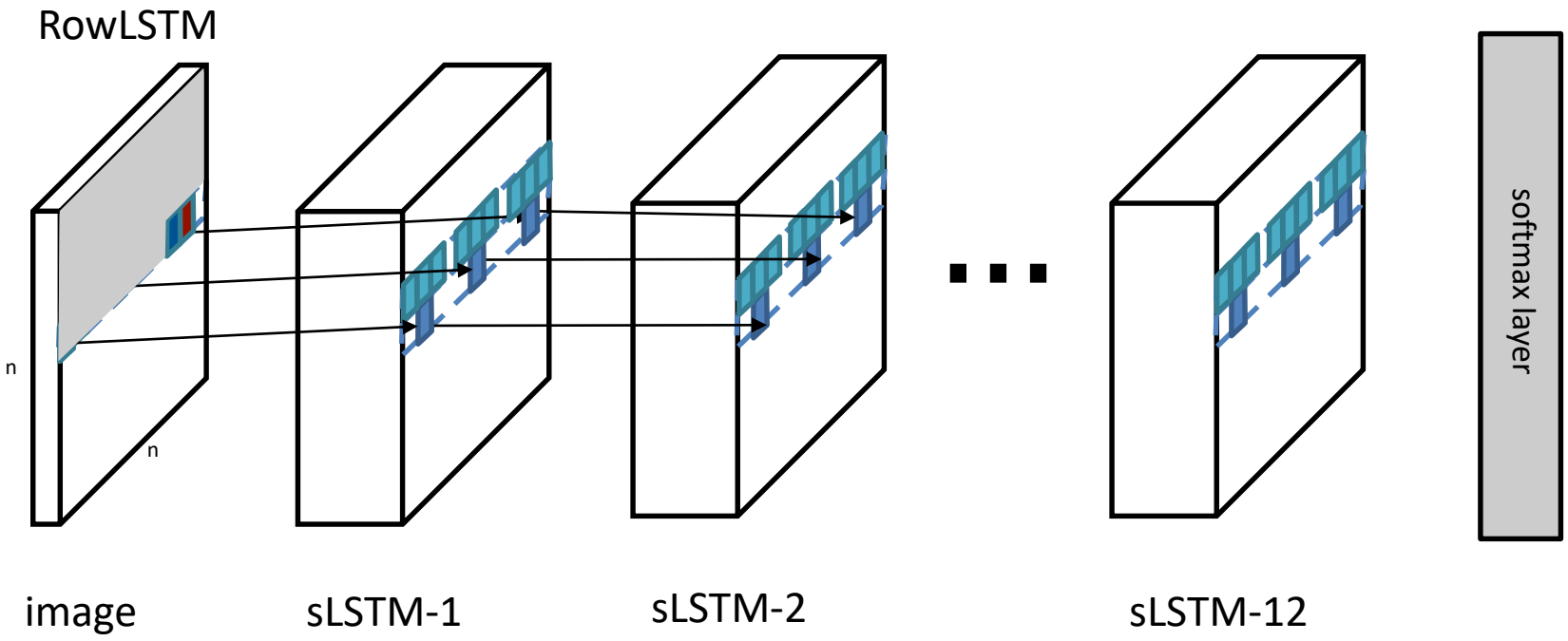


Figure: Example softmax outputs in the final layer, representing probability distribution over 256 classes.

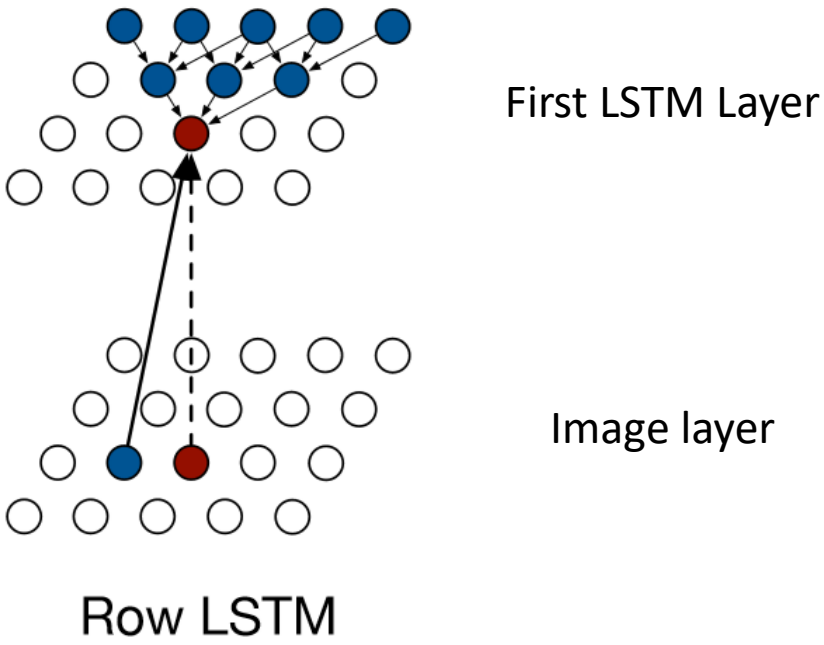
Figure from: Oord et al.

PixelRNN: A specific Multidimensional LSTM



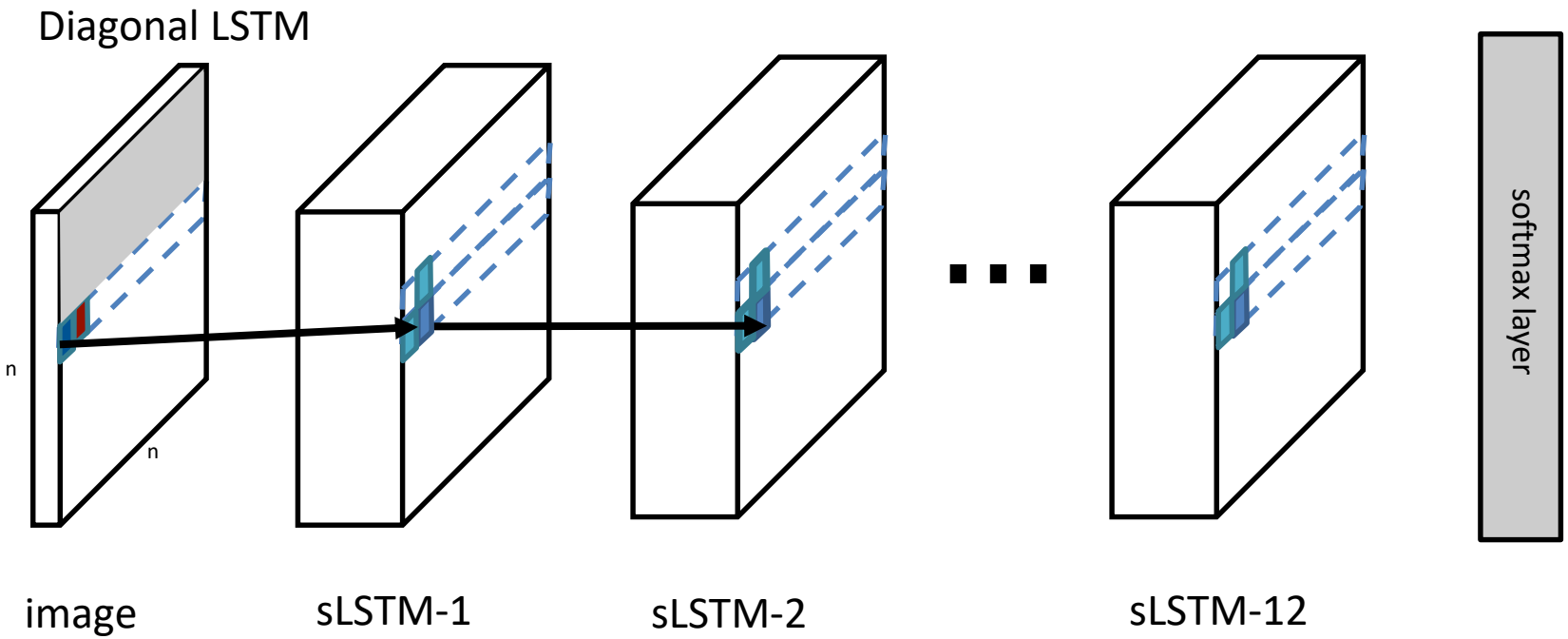
Pixel recurrent neural networks, ICML 2016

PixelRNN: A specific Multidimensional LSTM



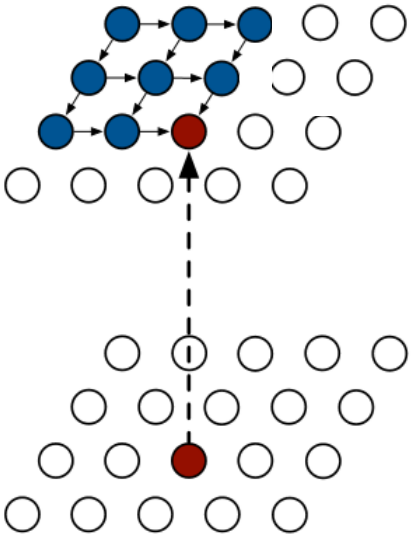
Pixel recurrent neural networks, ICML 2016

PixelRNN: A specific Multidimensional LSTM



Pixel recurrent neural networks, ICML 2016

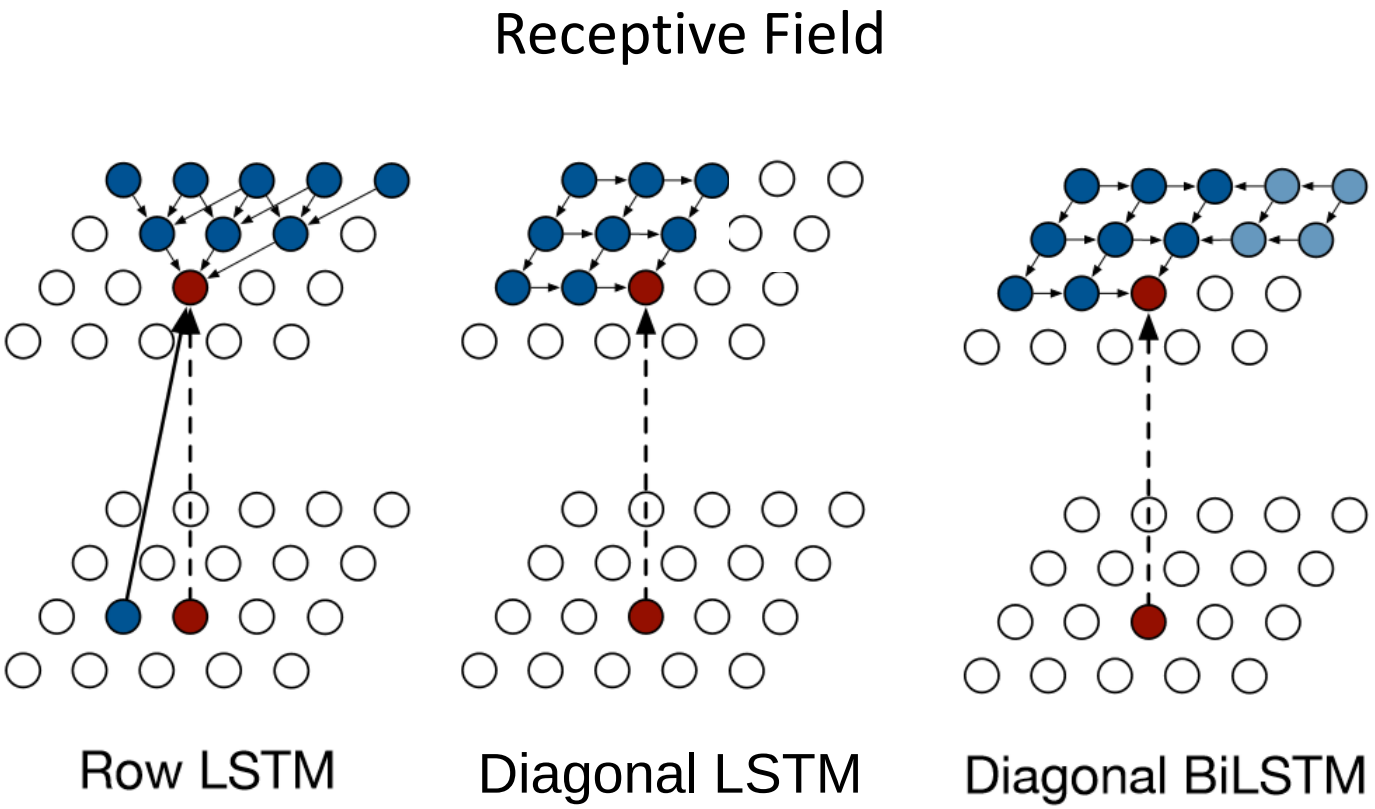
PixelRNN: A specific Multidimensional LSTM



Diagonal LSTM

Pixel recurrent neural networks, ICML 2016

PixelRNN: A specific Multidimensional LSTM

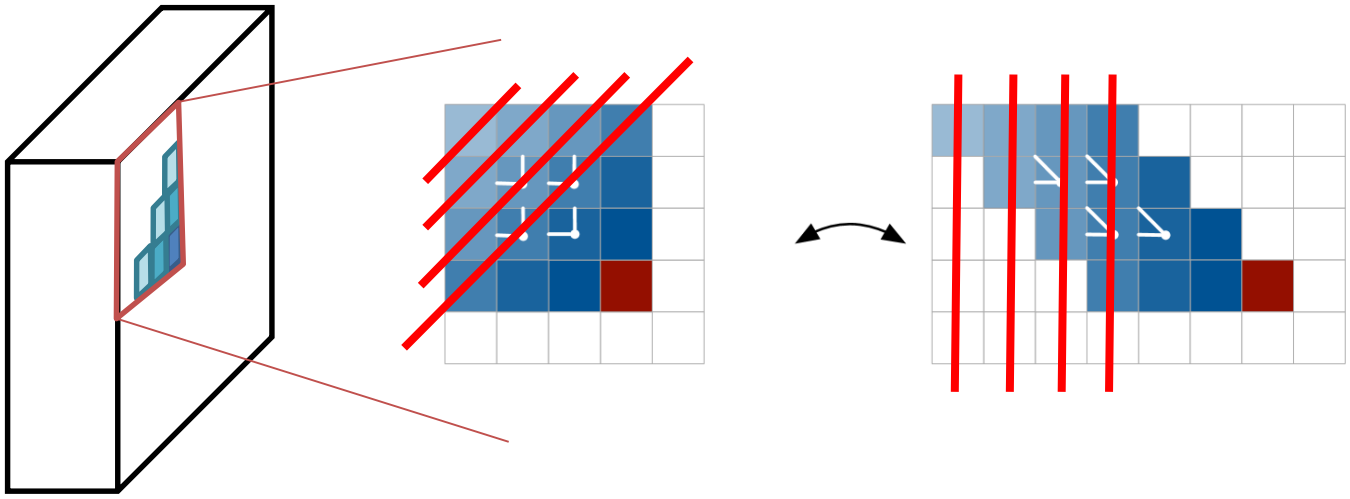


Pixel recurrent neural networks, ICML 2016

PixelRNN: A specific Multidimensional LSTM

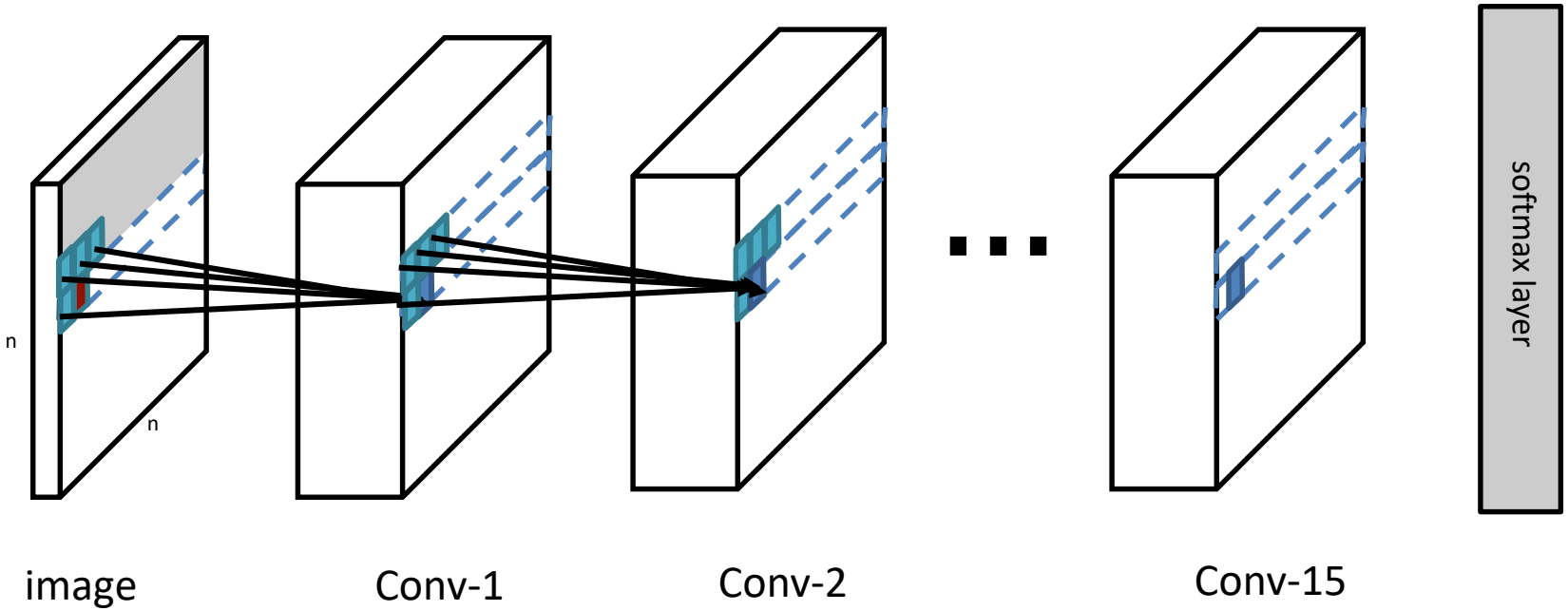
Diagonal LSTM

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



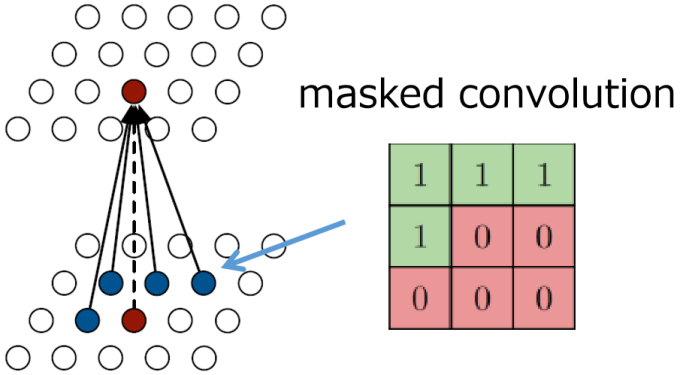
Pixel recurrent neural networks, ICML 2016

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

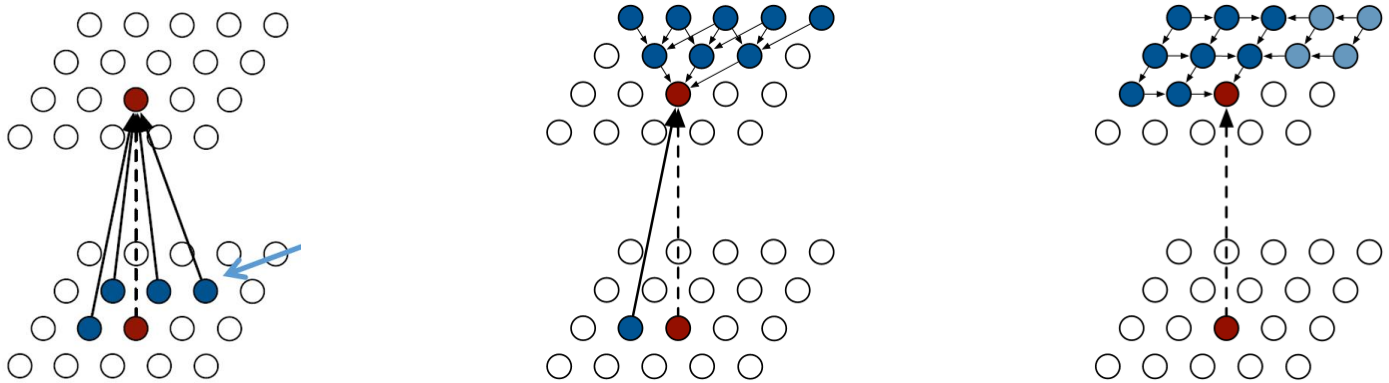
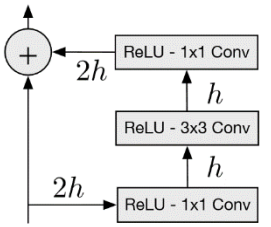


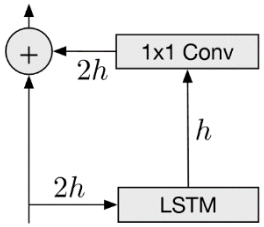
Figure from: Oord et al.

Architecture

- Residual connections

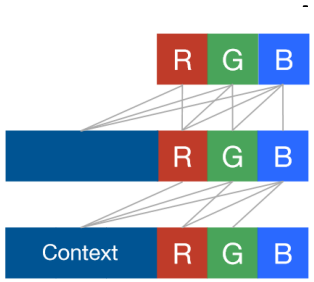


PixelCNN



PixelRNN

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



Mask B
- Channels are connected to themselves
- Used in all other subsequent layers

Mask A
- Channels are **not** connected to themselves
- Only used in first layer

Figure from: Oord et al.

Results

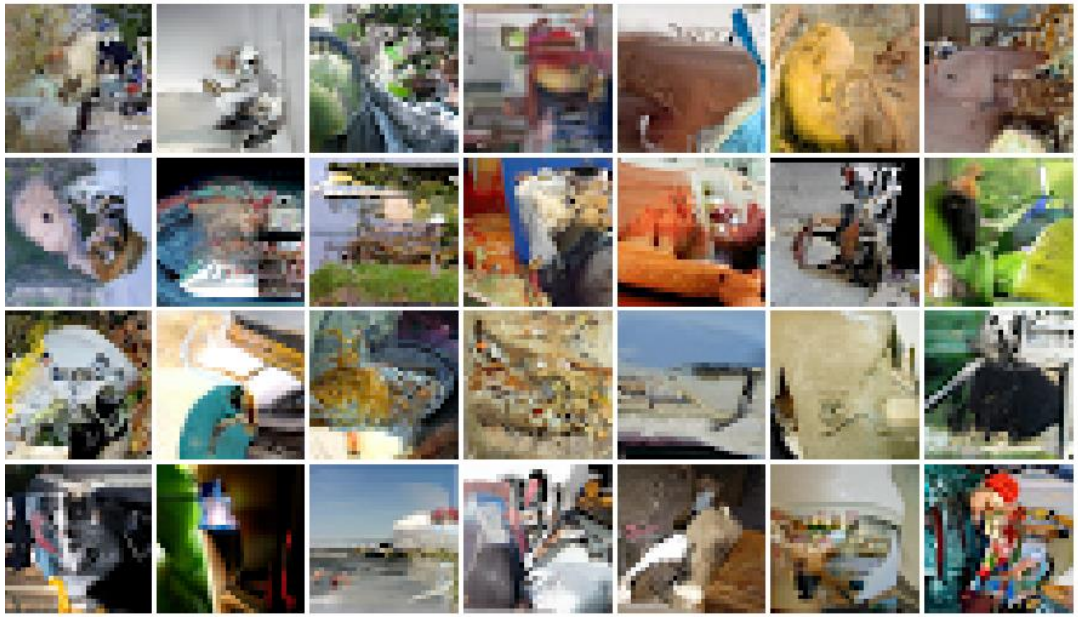


Figure: 32x32 ImageNet results from Diagonal BiLSTM model.

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

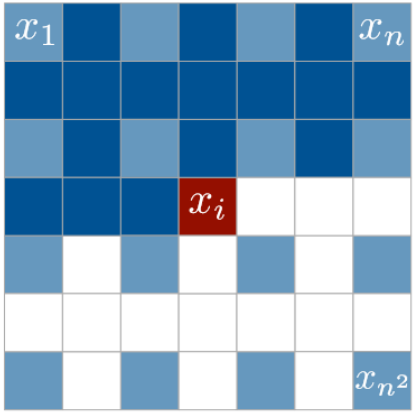


Figure from: Oord et al.

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

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

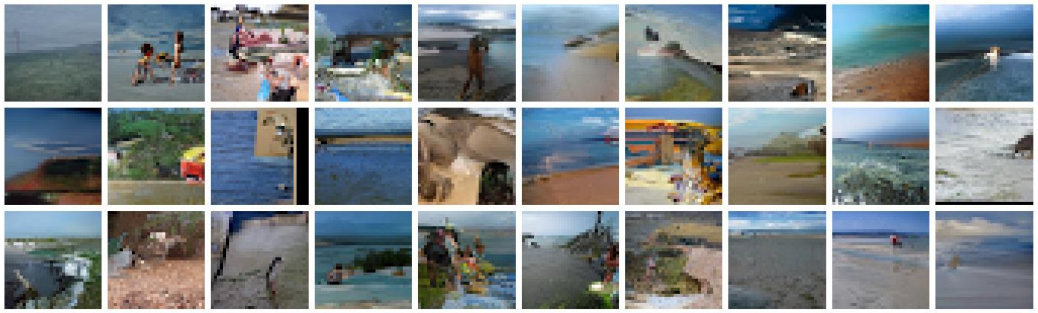
$$\mathbf{y} = \tanh(W_{k,f} * \mathbf{x} + V_{k,f} * \mathbf{s}) \odot \sigma(W_{k,g} * \mathbf{x} + V_{k,g} * \mathbf{s})$$

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 process (e.g. softmax loss) • Best log likelihoods so far	• Sharpest images	• Easy to relate image with low-dimensional latent variables
• Inefficient during sampling • Don't easily provide simple low-dimensional codes for imag	• 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

- 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 !