深度学习

第十讲

王胤

Kullback-Leibler Divergence

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

$$= \mathbb{E}_{X \sim p(x)} \left[\log \frac{p(x)}{q(x)} \right]$$

$$\int P(x) \log \frac{P(x)}{q(x)} dx$$

- Always ≥ 0, where the equality holds iff $p = q$
- Asymmetric $D(p \parallel q) \neq D(q \parallel p)$
- Does not satisfy triangle inequality

Jensen-Shannon Divergence

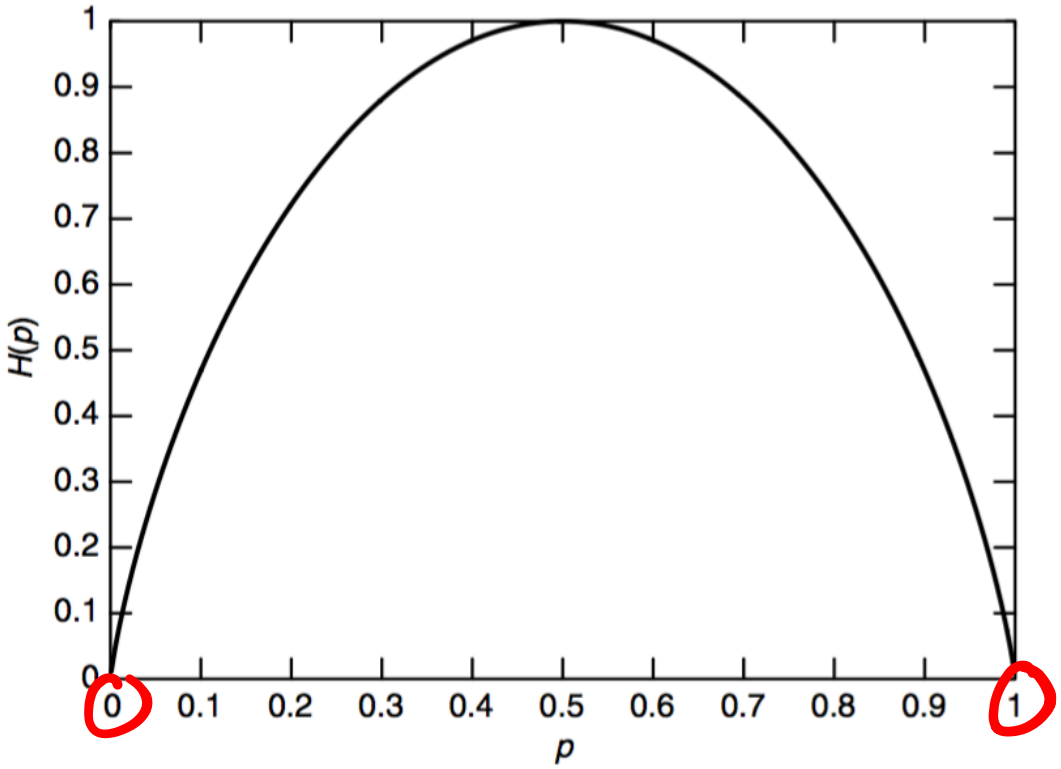
$$D_{JS}(p \parallel q) = \frac{1}{2} D_{KL}(p \parallel m) + \frac{1}{2} D_{KL}(q \parallel m)$$

$$where\ m = \frac{1}{2}(p + q)$$

$$D_{JS}(p \parallel q) = D_{JS}(q \parallel p)$$

Entropy

$$
\begin{aligned}
H(X) &= -\sum_x p(x) \log p(x) \\
&= \sum_x -p(x) \log p(x) \\
&= \sum_x p(x) \log \frac{1}{p(x)} \\
&= \mathbb{E}_{X \sim p(x)} \left[\log \frac{1}{p(x)} \right]
\end{aligned}
$$



Cross Entropy

$$H(p,q) = H(p) + D(p||q)$$

$$= \mathbb{E}_{X\sim p(x)}\left[\log \frac{1}{p(x)}\right] + \mathbb{E}_{X\sim p(x)}\left[\log \frac{p(x)}{q(x)}\right]$$

$$= \mathbb{E}_{X\sim p(x)}\left[\log \frac{1}{q(x)}\right]$$

Maximum Likelihood Estimation:

Density Estimation

$x_0, x_1, \ldots x_m = X_{example}$

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}}\, p_{model}(X_{example}; \theta)$$

$$= \underset{\theta}{\operatorname{argmax}} \prod_{i=0}^{n} p_{model}(x_i; \theta)$$

$$= \operatorname{argmax} \log(\Pi \ldots)$$

$$= \operatorname{argmax} \sum_i \log p_{model}(x_i; \theta)$$

$$= \operatorname{argmax} \sum_i \frac{1}{m} \log p_{model}(x_i; \theta)$$

$$= \operatorname{argmin} E_{x \sim x_{example}} -\log p_{model}(x_i; \theta)$$

Maximum Likelihood Estimation:

Classification (x_0, y_0) (x_1, y_1) ... (x_m, y_m)

$$\theta = \arg\max_{\theta} P_{model}(y_i \mid x_i ; \theta)$$

y_i $[0, 0, \dots 1, \dots 0]$

$\hat{y}_i = P_{model}(j \mid x_i ; \theta)$ $\hat{y}_i$ $[0.1, 0.05, \dots 0.8, \dots 0.001]$

$$= \arg\max_{\theta} (\log P_{model}(y_i \mid x_i ; \theta)$$

$$= \arg\max_{\theta} \sum_{j=1}^{k} y_i^j \cdot \log \hat{y}_i^j$$

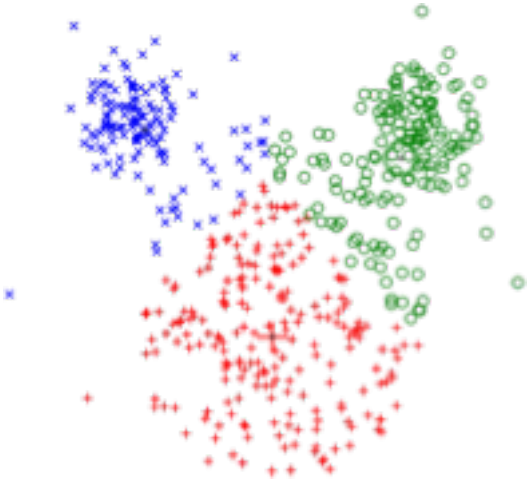
Supervised vs Unsupervised Learning

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



K-means clustering

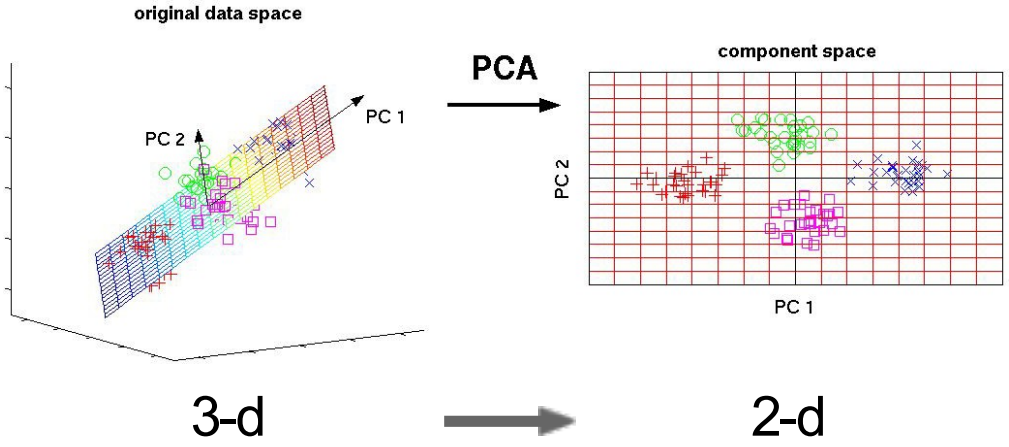
Supervised vs Unsupervised Learning

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

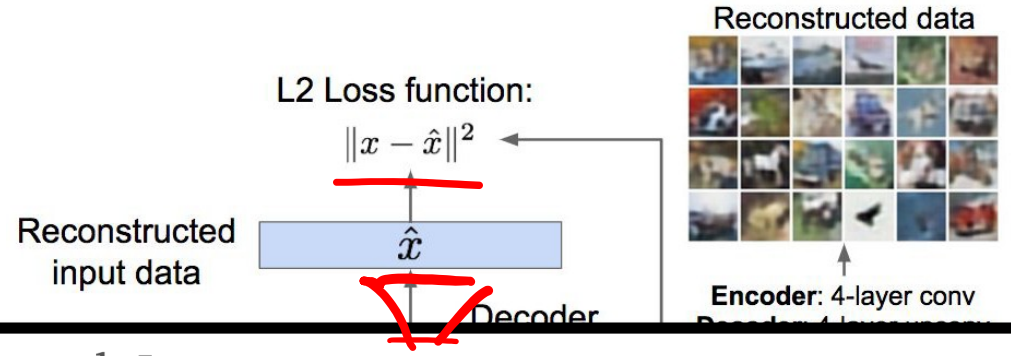


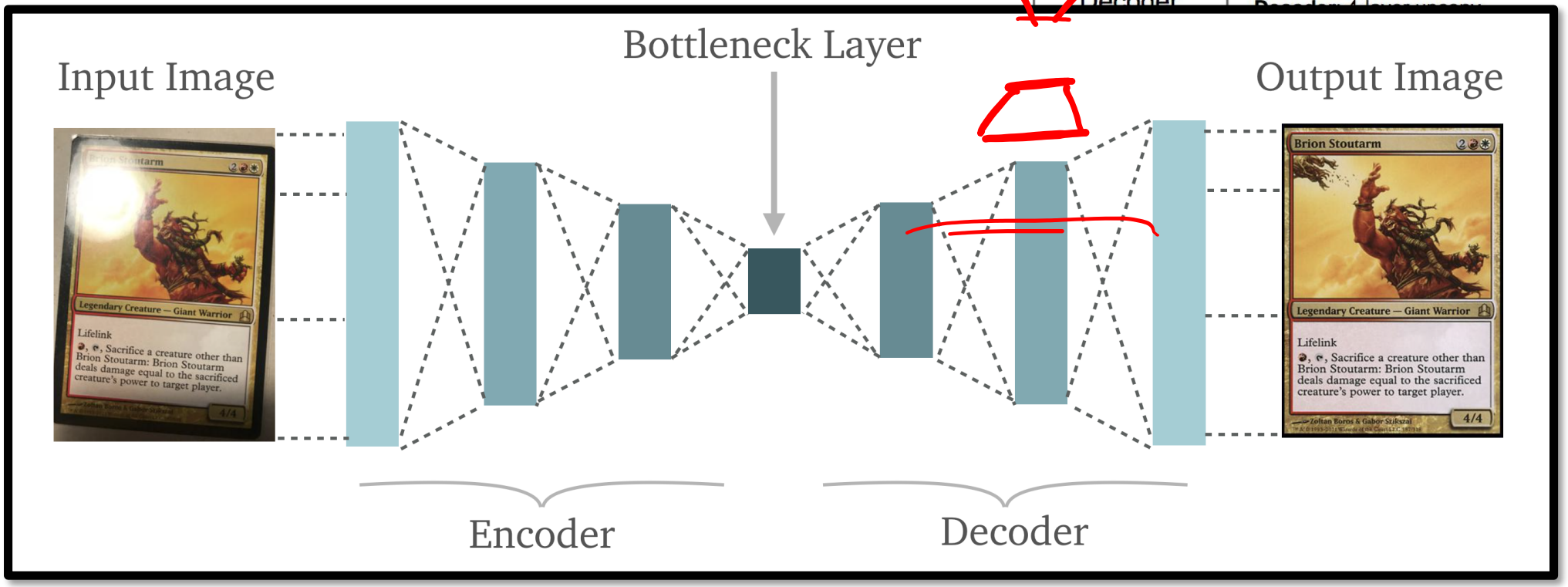
Principal Component Analysis
(Dimensionality reduction)

Supervised vs Unsupervised Learning

Unsupervised Learning

Data: x
Just data, no labels!





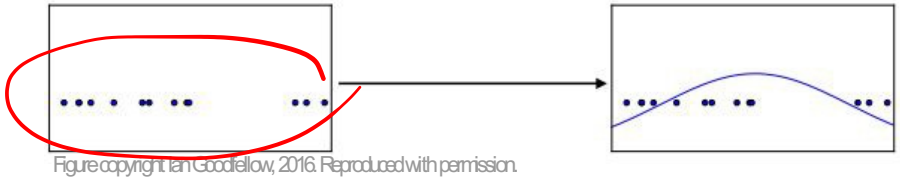
Supervised vs Unsupervised Learning

Unsupervised Learning

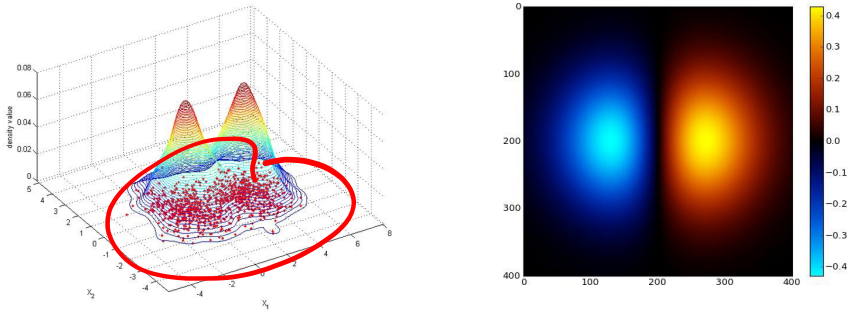
Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



1-d density estimation



2-d density estimation

Why Unsupervised Learning?

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

Training data is cheap

Data: x
Just data, no labels!

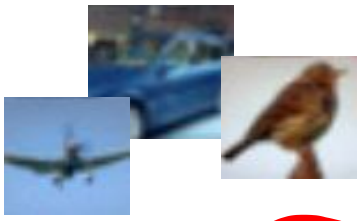
Holy grail: Solve unsupervised learning => understand structure of visual world

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Generative Models

Given training data, generate new samples from same distribution



Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

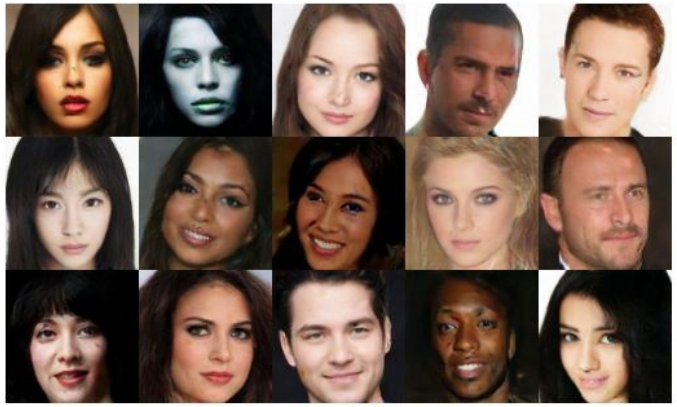
Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Addresses density estimation, a core problem in unsupervised learning

Several flavors:

- Explicit density estimation: explicitly define and solve for $p_{model}(x)$
- Implicit density estimation: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it

Why Generative Models?



Taxonomy of Generative Models

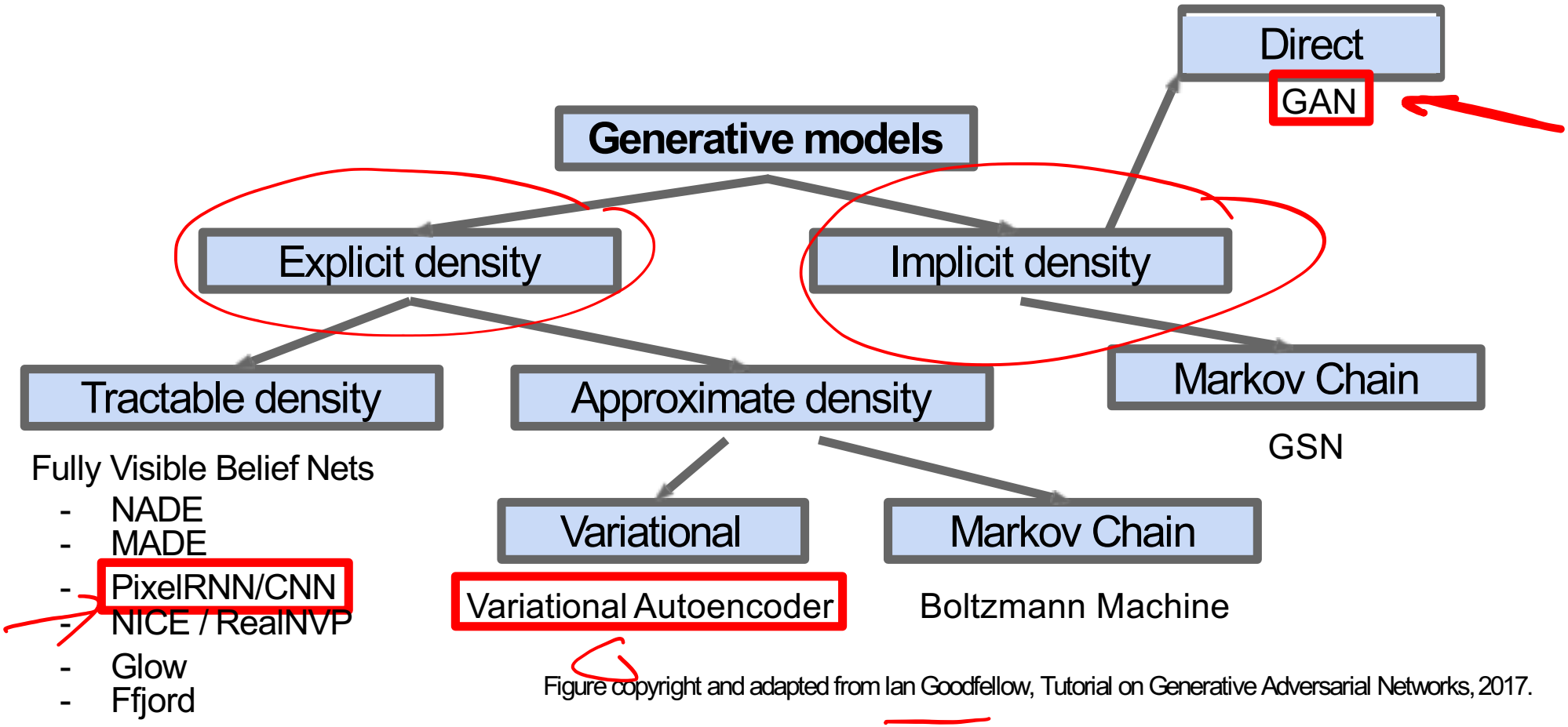


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

PixelCNN *[van der Oord et al. 2016]*

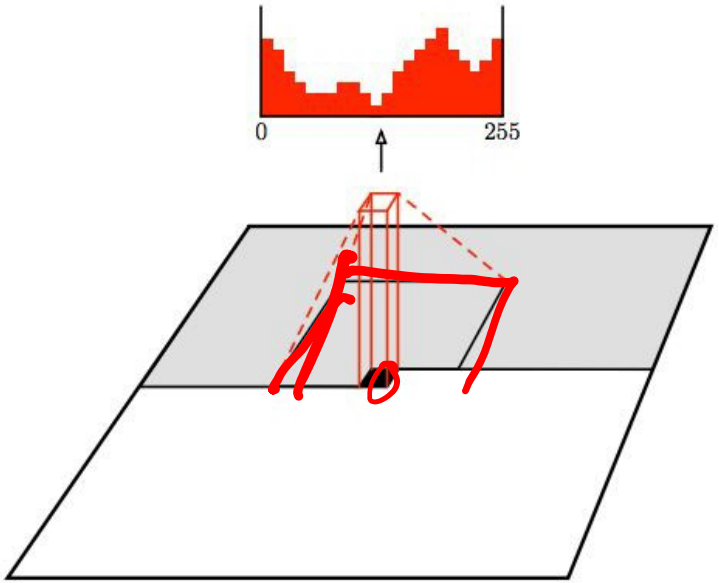
Still generate image pixels starting from corner

Dependency on previous pixels now modeled using a CNN over context region

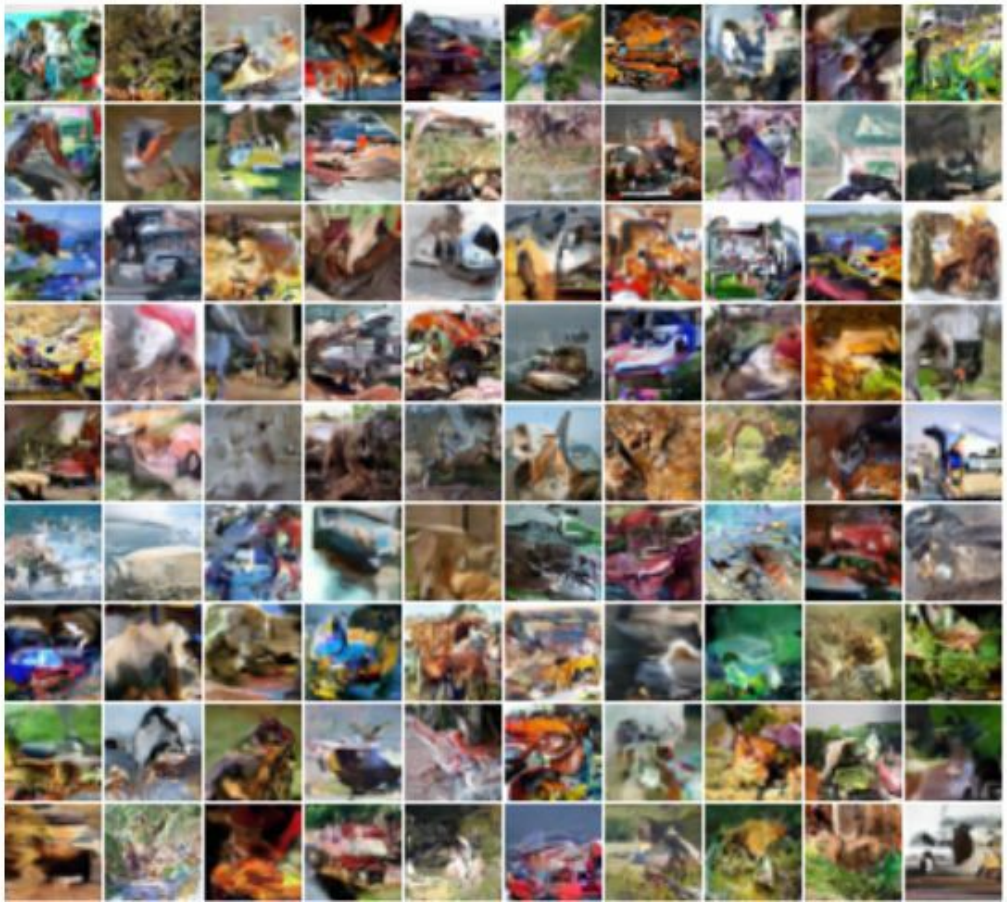
Training: maximize likelihood of training images

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

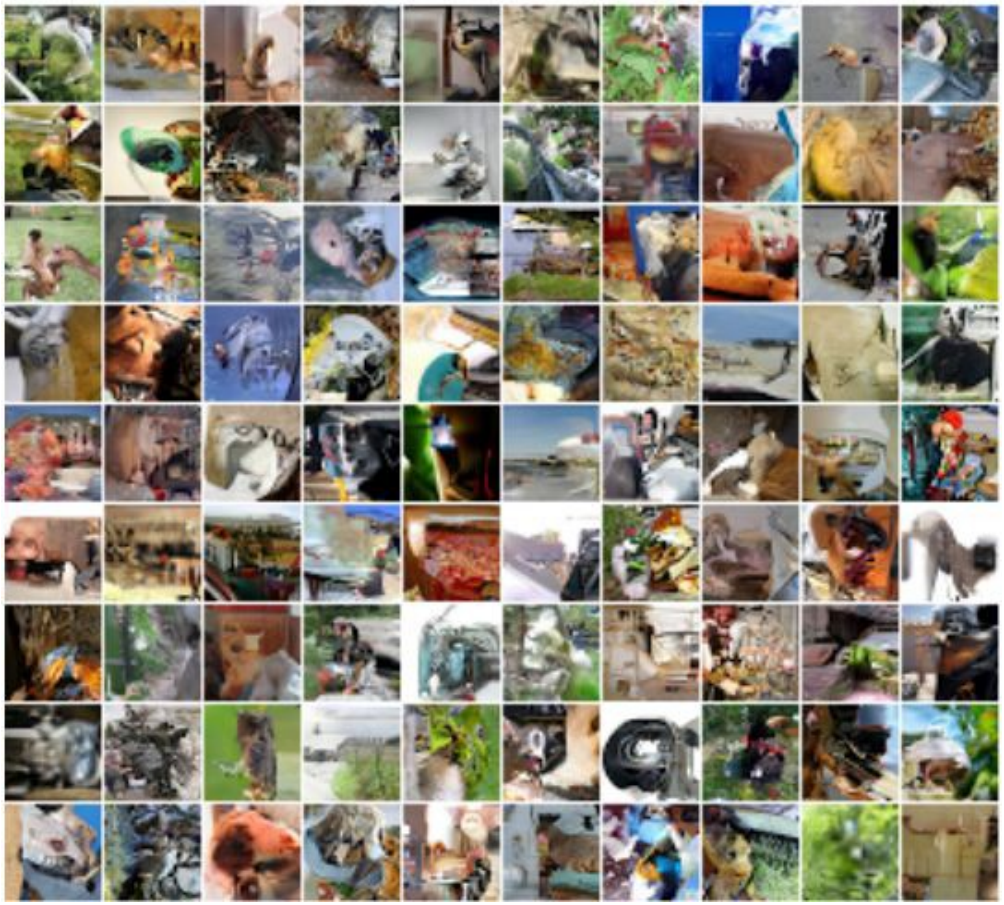
Softmax loss at each pixel



Generation Samples

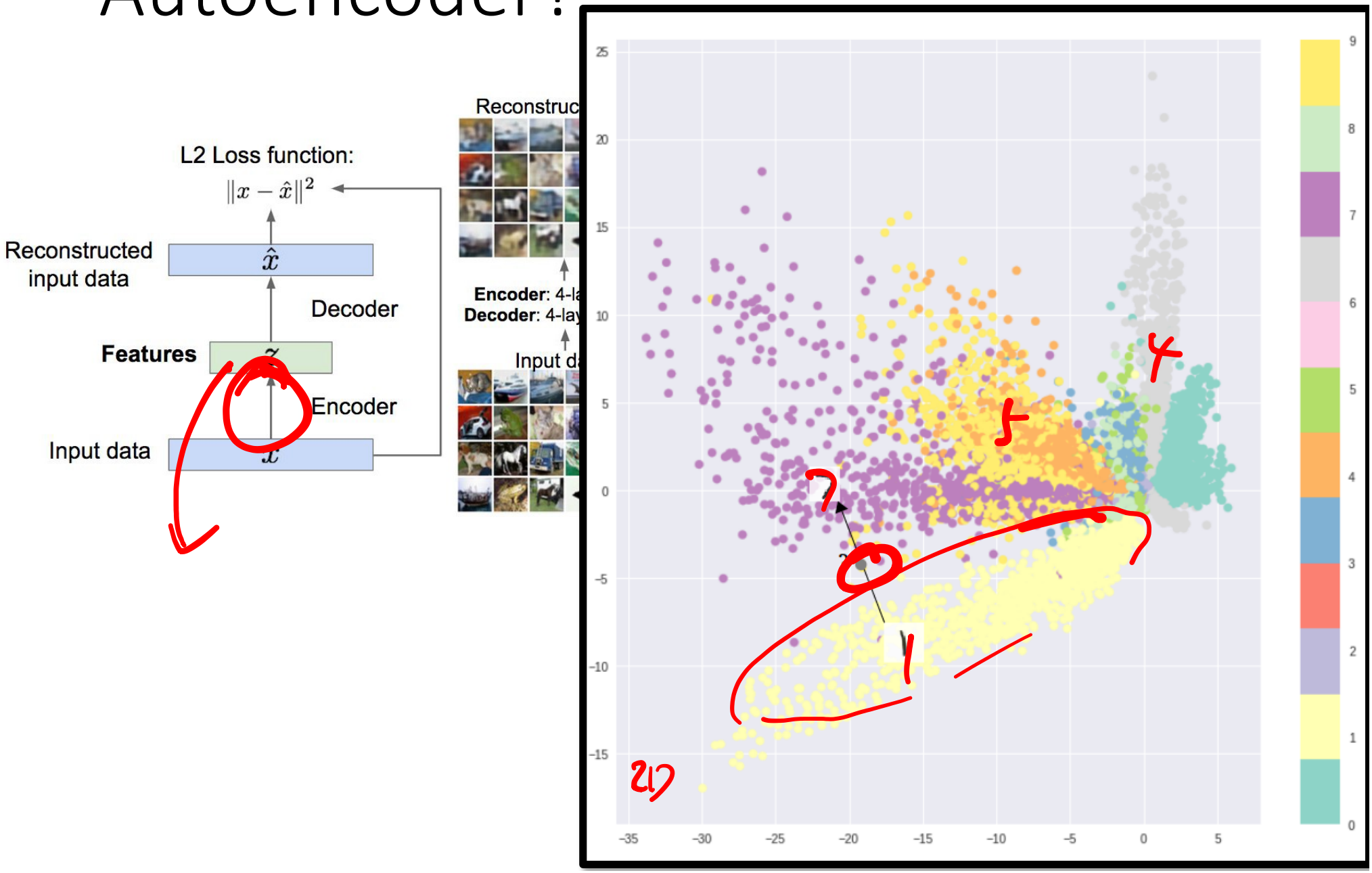


32x32 CIFAR-10

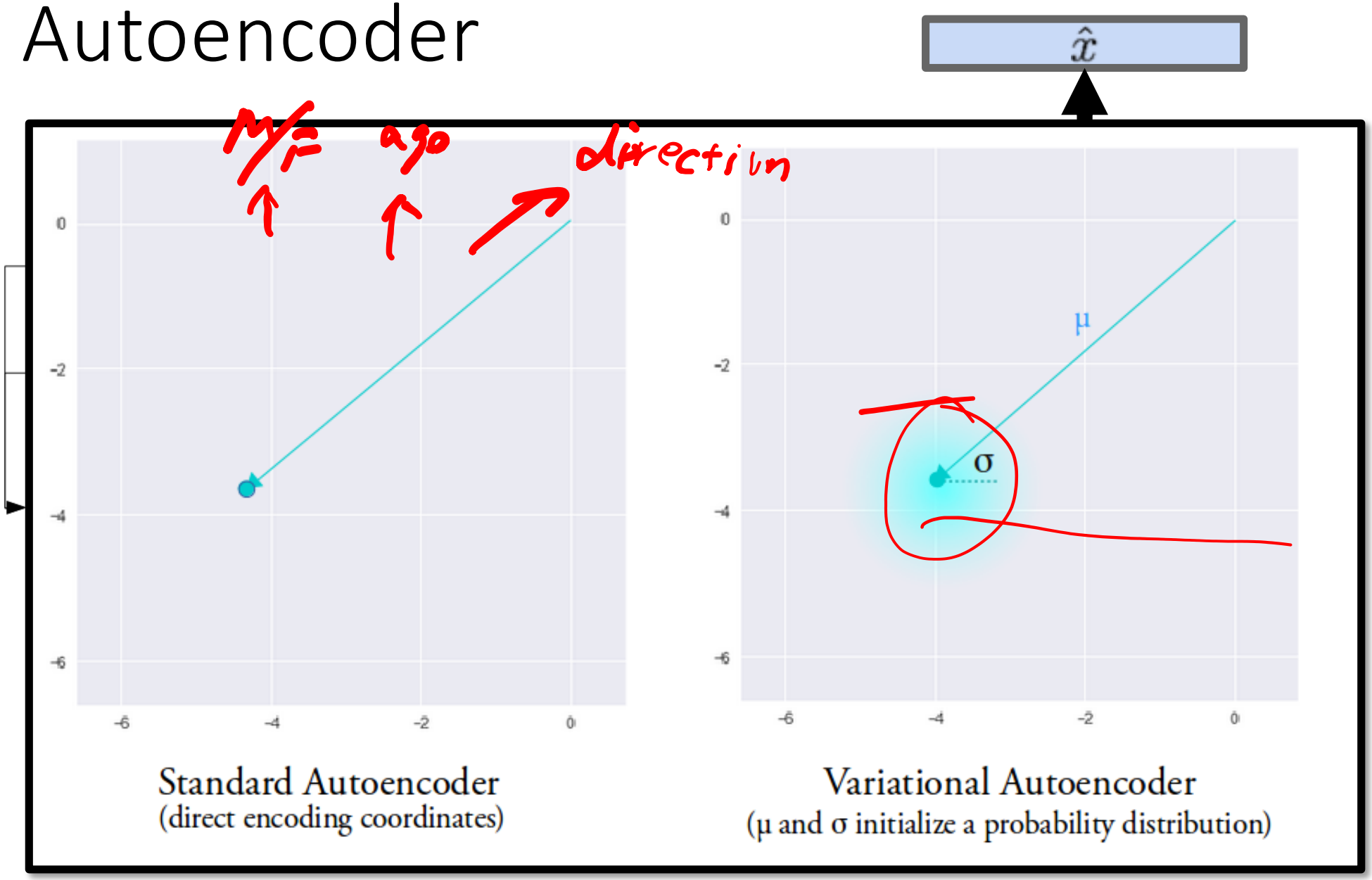


32x32 ImageNet

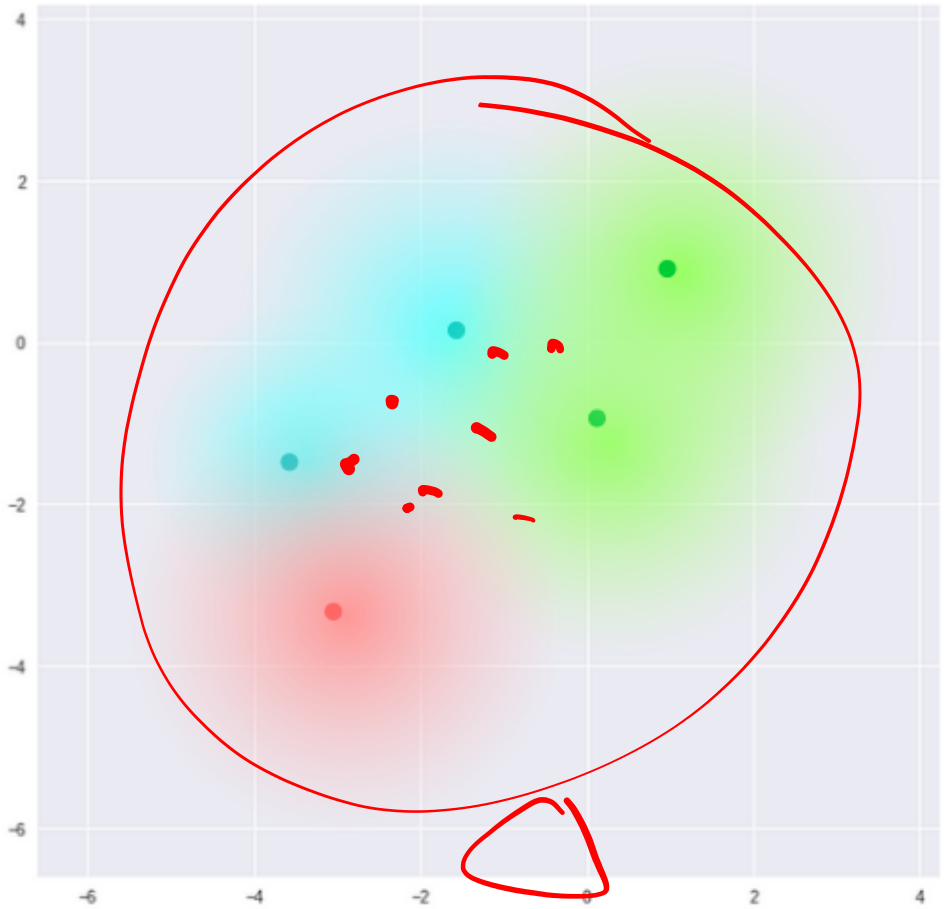
Autoencoder?



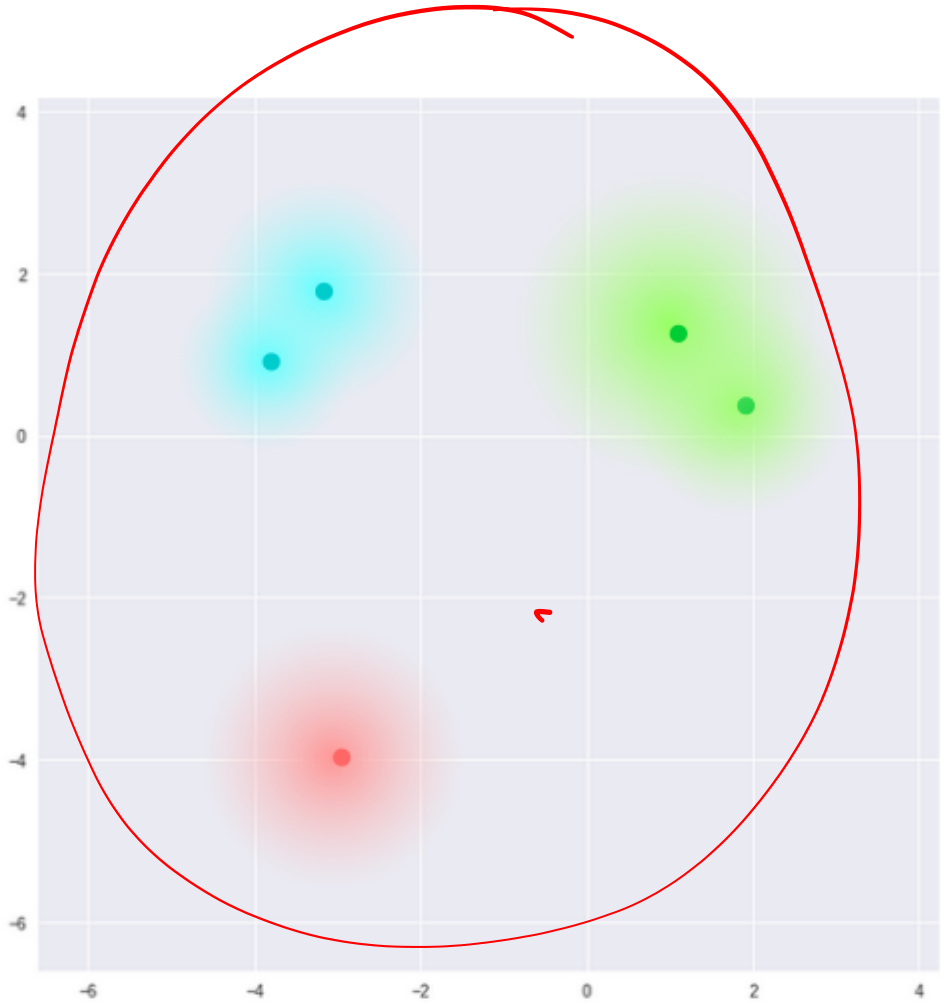
Variational Autoencoder



Variational Autoencoder



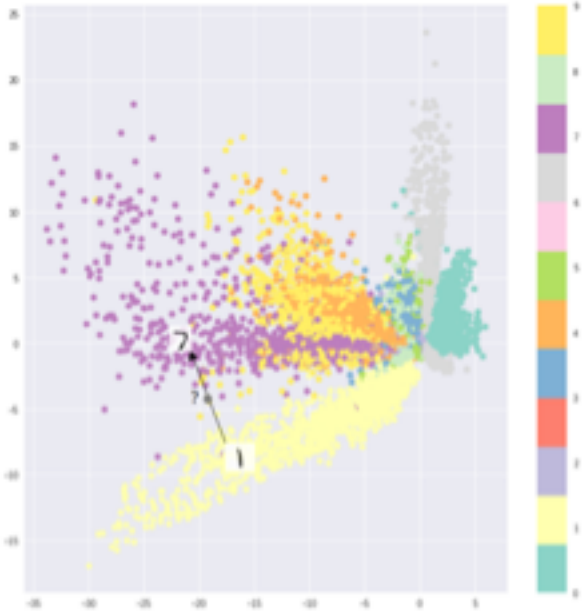
What we require



What we may inadvertently end up with

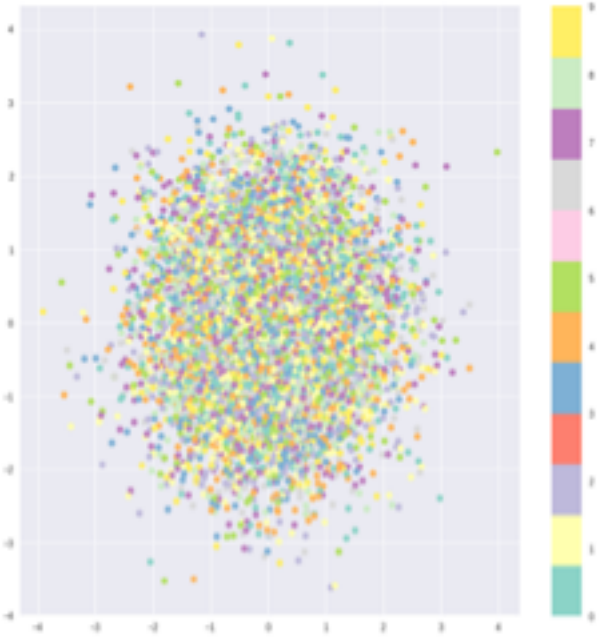
Variational Autoencoder

Only reconstruction loss

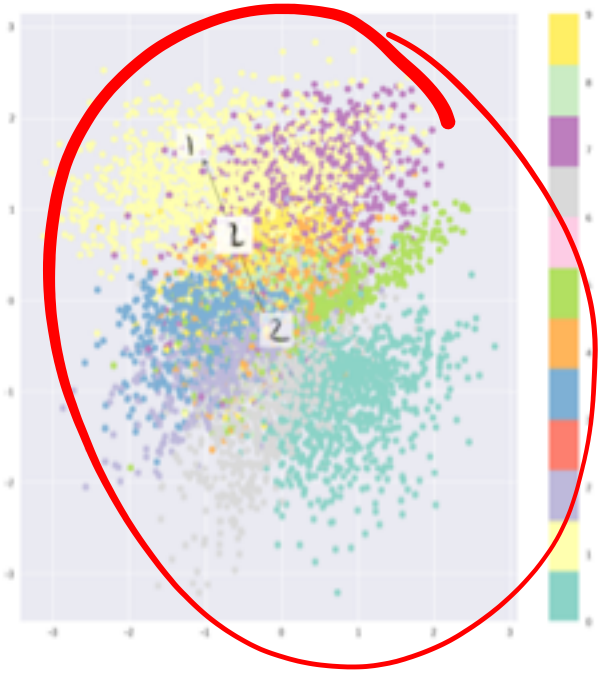


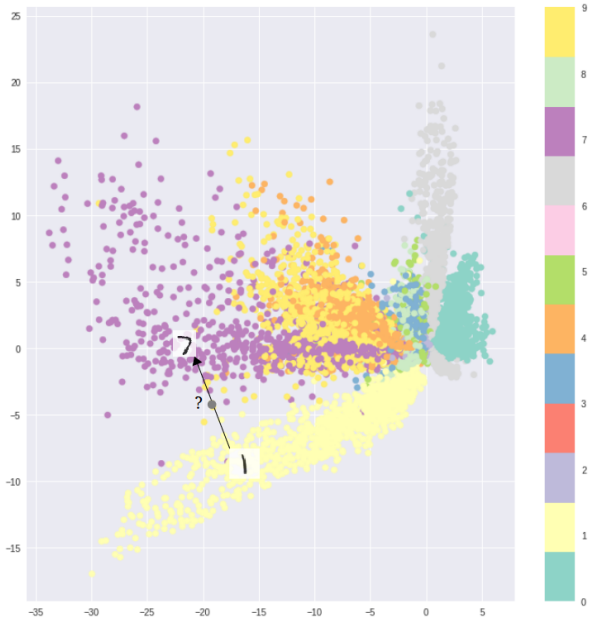
+

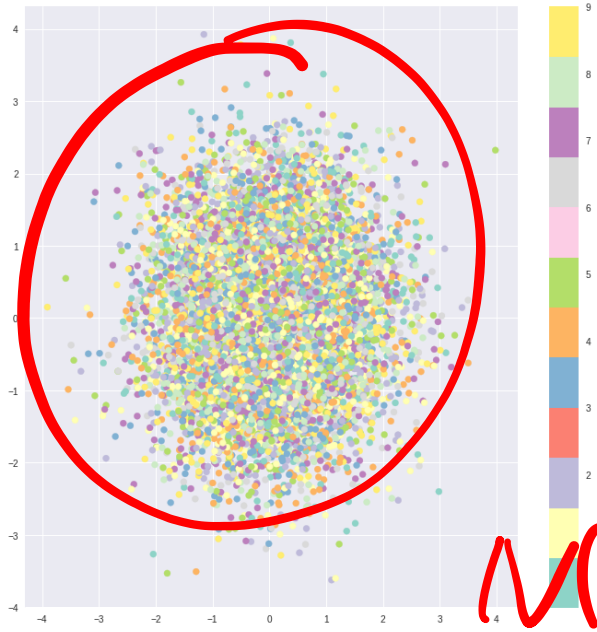
Only KL divergence



Combination







$\hat{x}$

θ

z

Sample z from $z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$

$\mu_{z|x}$ $\Sigma_{z|x}$

ϕ

Input Data

x

Variational Autoencoders

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

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

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

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

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

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

We want to maximize the data likelihood

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

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

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

CE

$$-D_{KL}((q_\phi(\mathbf{z})||p_\theta(\mathbf{z})) = \int q_\theta(\mathbf{z})\left(\log p_\theta(\mathbf{z}) - \log q_\theta(\mathbf{z})\right) d\mathbf{z}$$

$$= \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j)^2) - (\mu_j)^2 - (\sigma_j)^2\right)$$

Variational Autoencoders

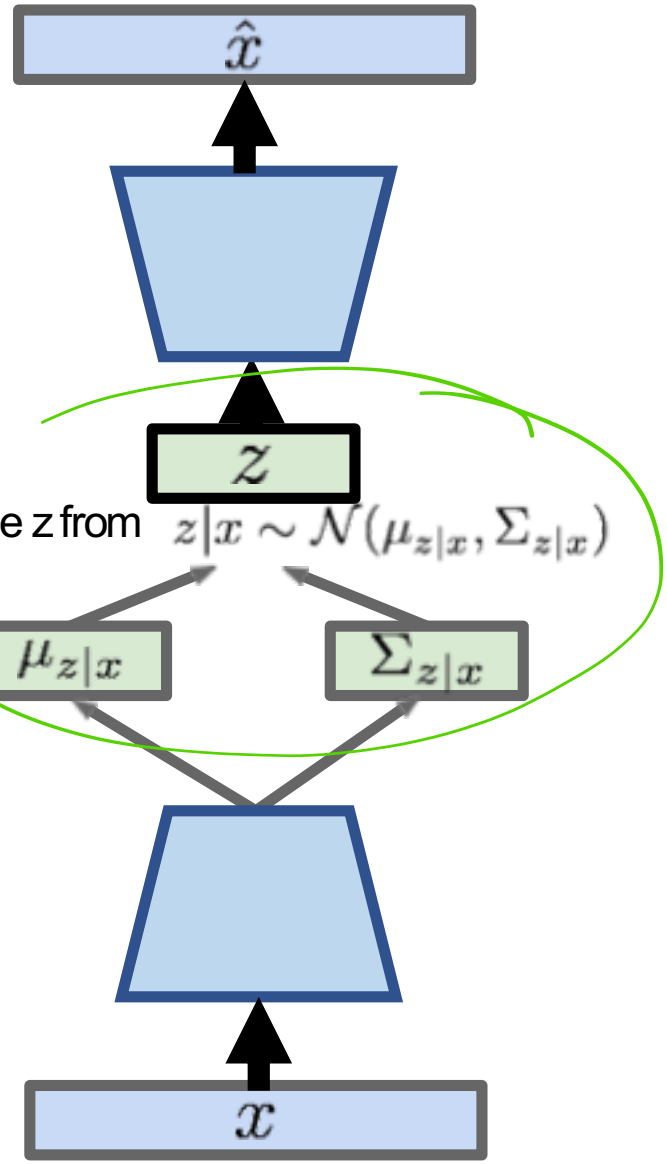
Putting it all together: maximizing the likelihood lower bound

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

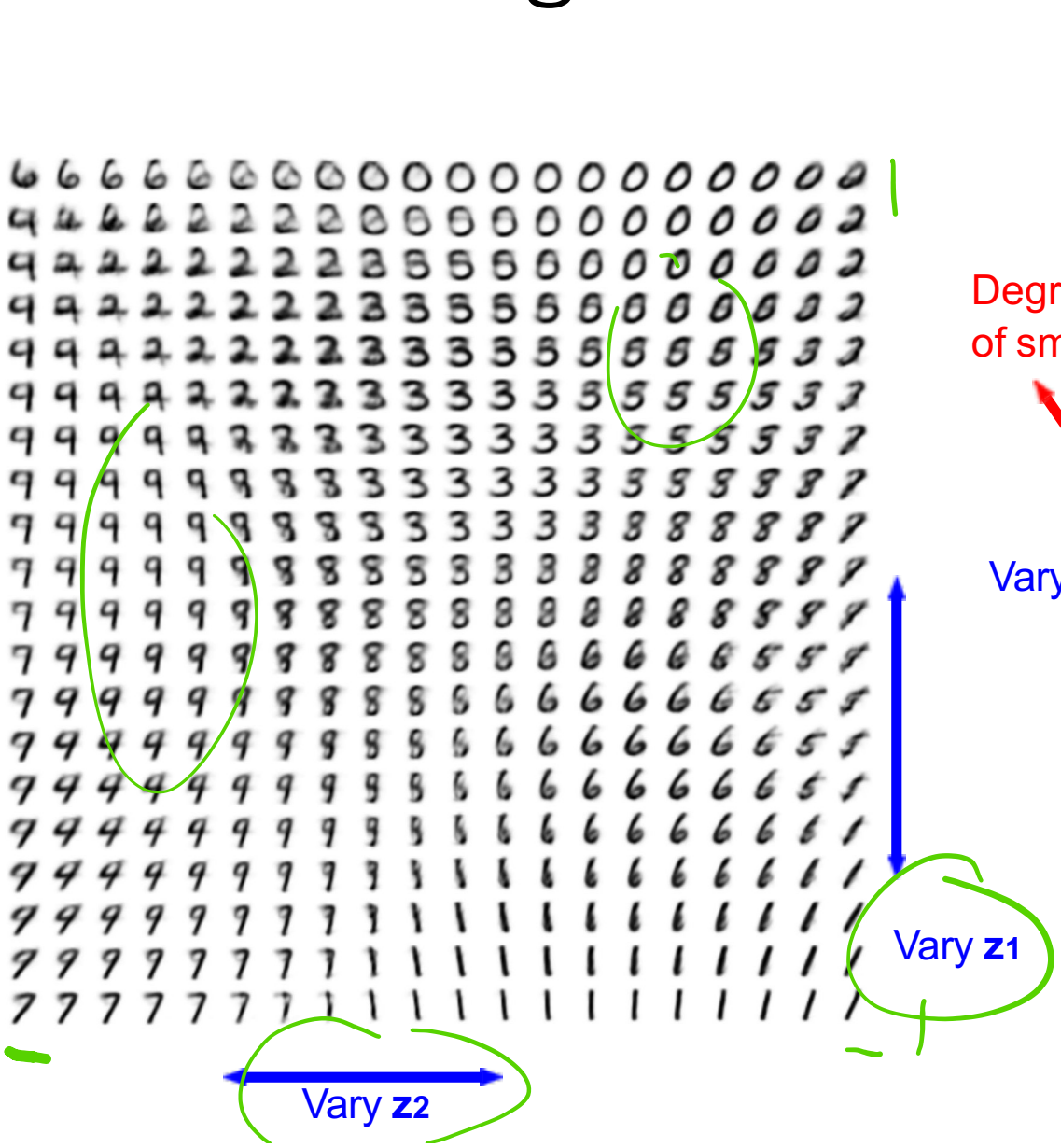
Maximize likelihood of original input being reconstructed

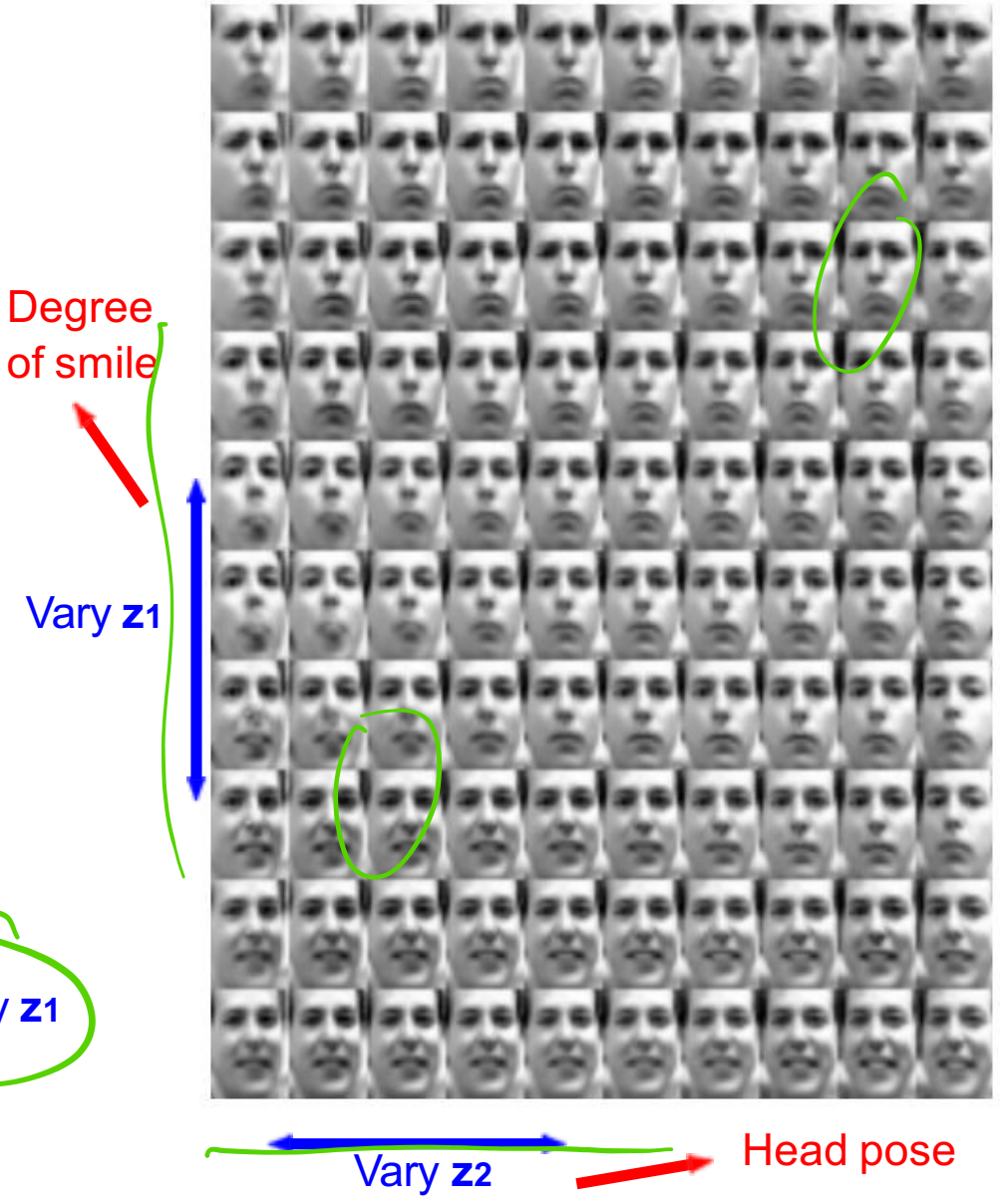
Make approximate posterior distribution close to prior

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

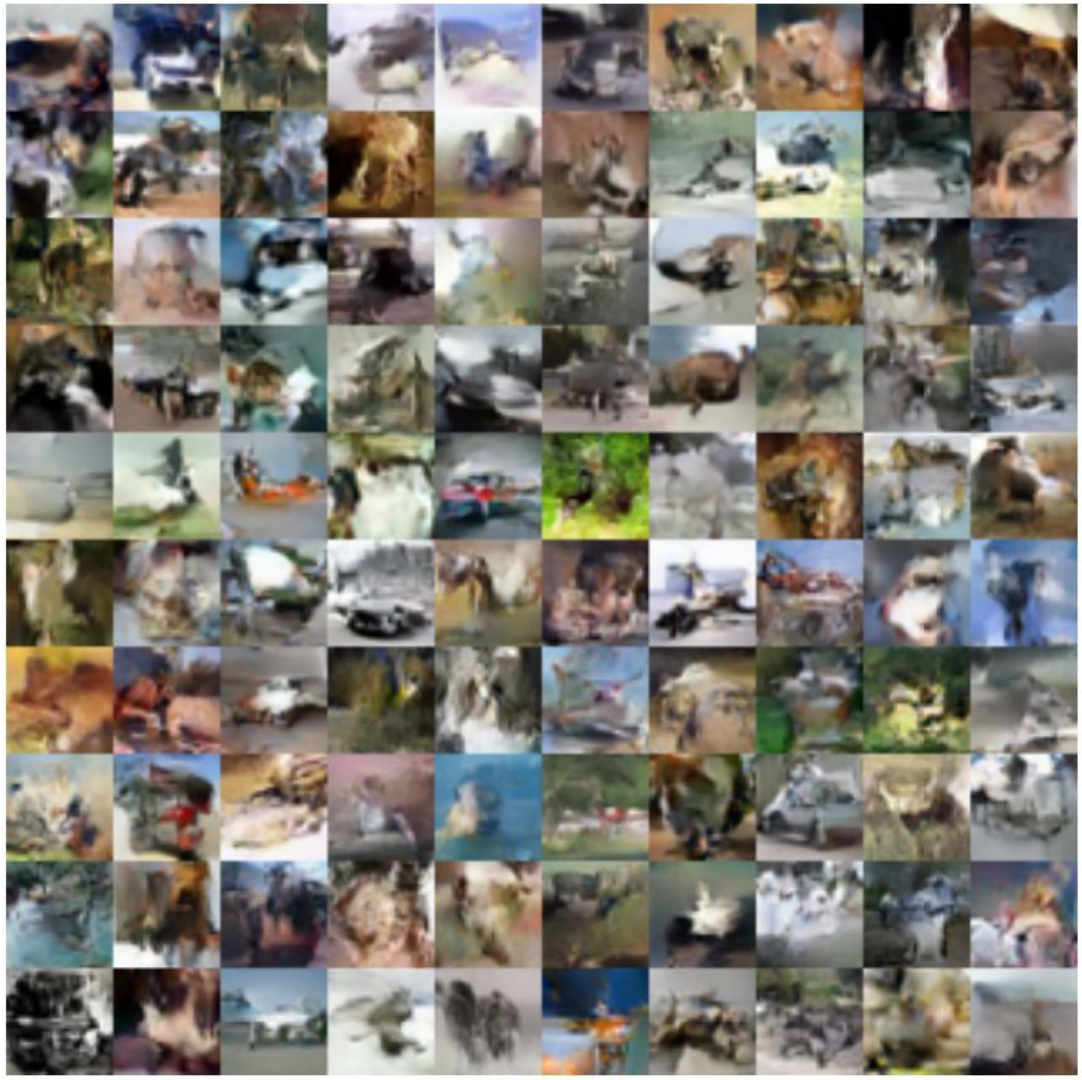


Variational Autoencoders: Generating Data!





Variational Autoencoders: Generating Data!



32x32 CIFAR-10

Labeled Faces in the Wild

So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

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

VAEs define intractable density function with latent **z**:

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

Cannot optimize directly, derive and optimize lower bound on likelihood instead

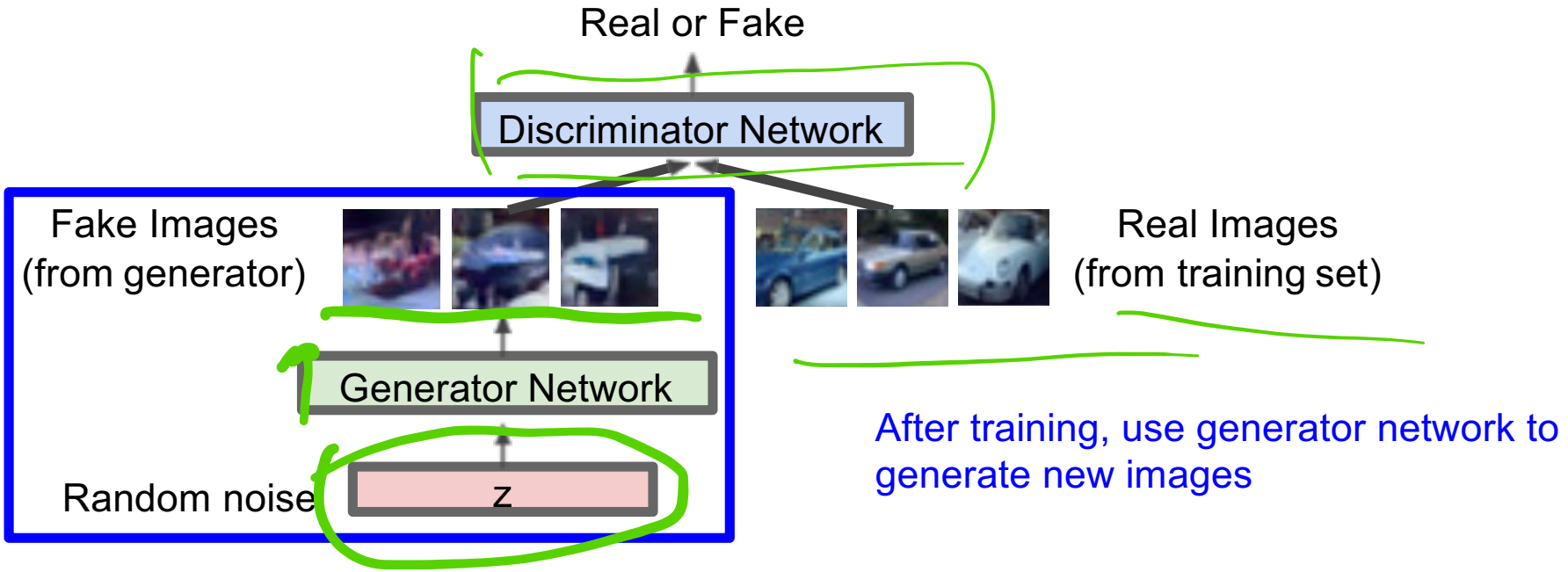
What if we give up on explicitly modeling density, and just want ability to sample?

GANs: don't work with any explicit density function!
Instead, take game-theoretic approach: learn to generate from training distribution through 2-player game

GAN

Training GANs: Two-player game

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

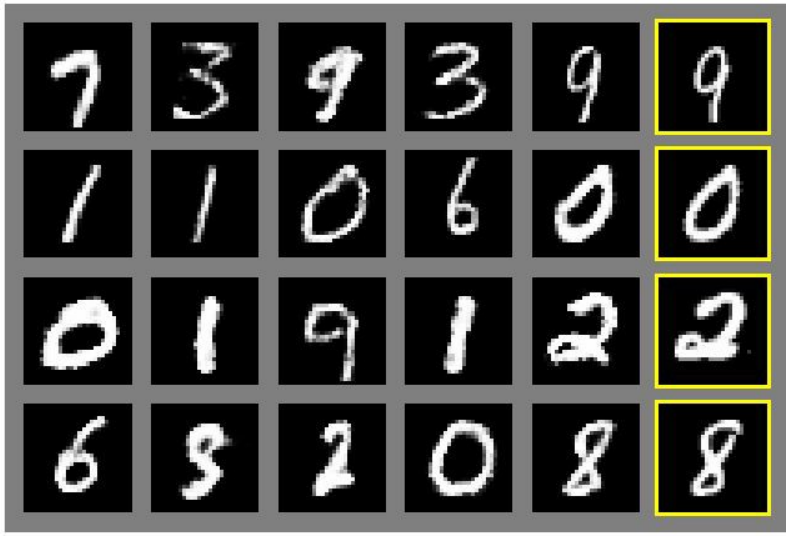


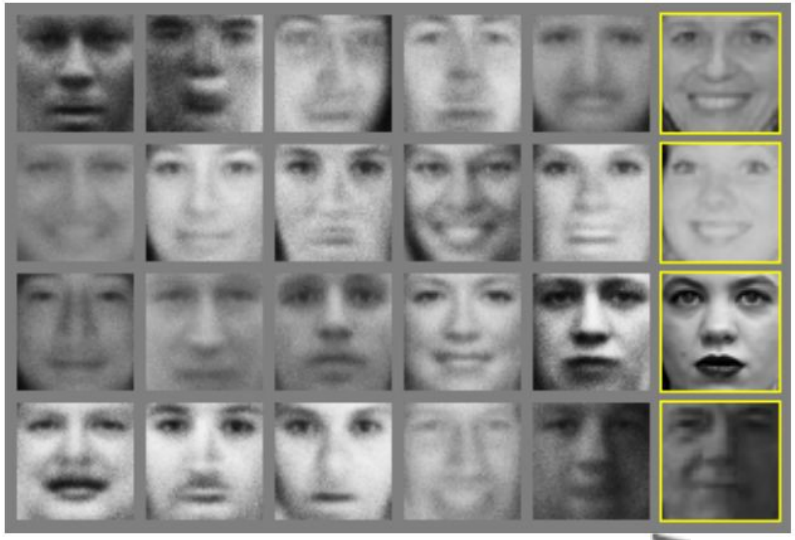
Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Generative Adversarial Nets

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Generated samples



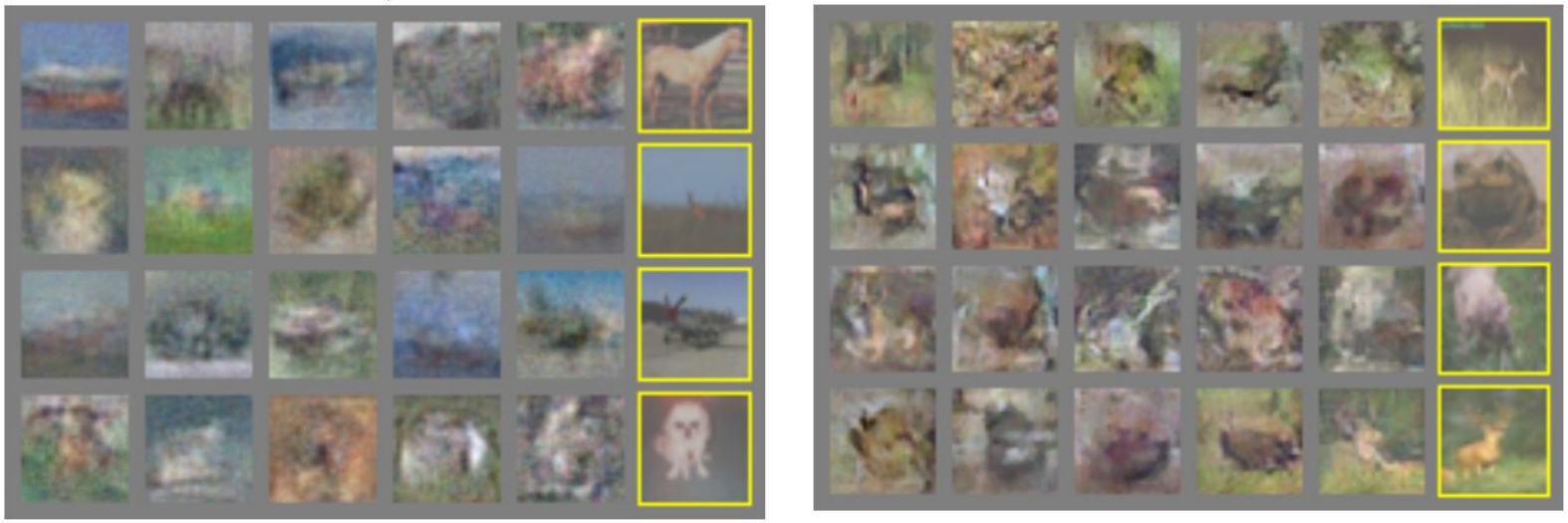


Nearest neighbor from training set

Generative Adversarial Nets

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Generated samples (CIFAR-10)



Nearest neighbor from training set

Generative Adversarial Nets: Convolutional Architectures

Generator is an upsampling network with fractionally-strided convolutions
Discriminator is a convolutional network

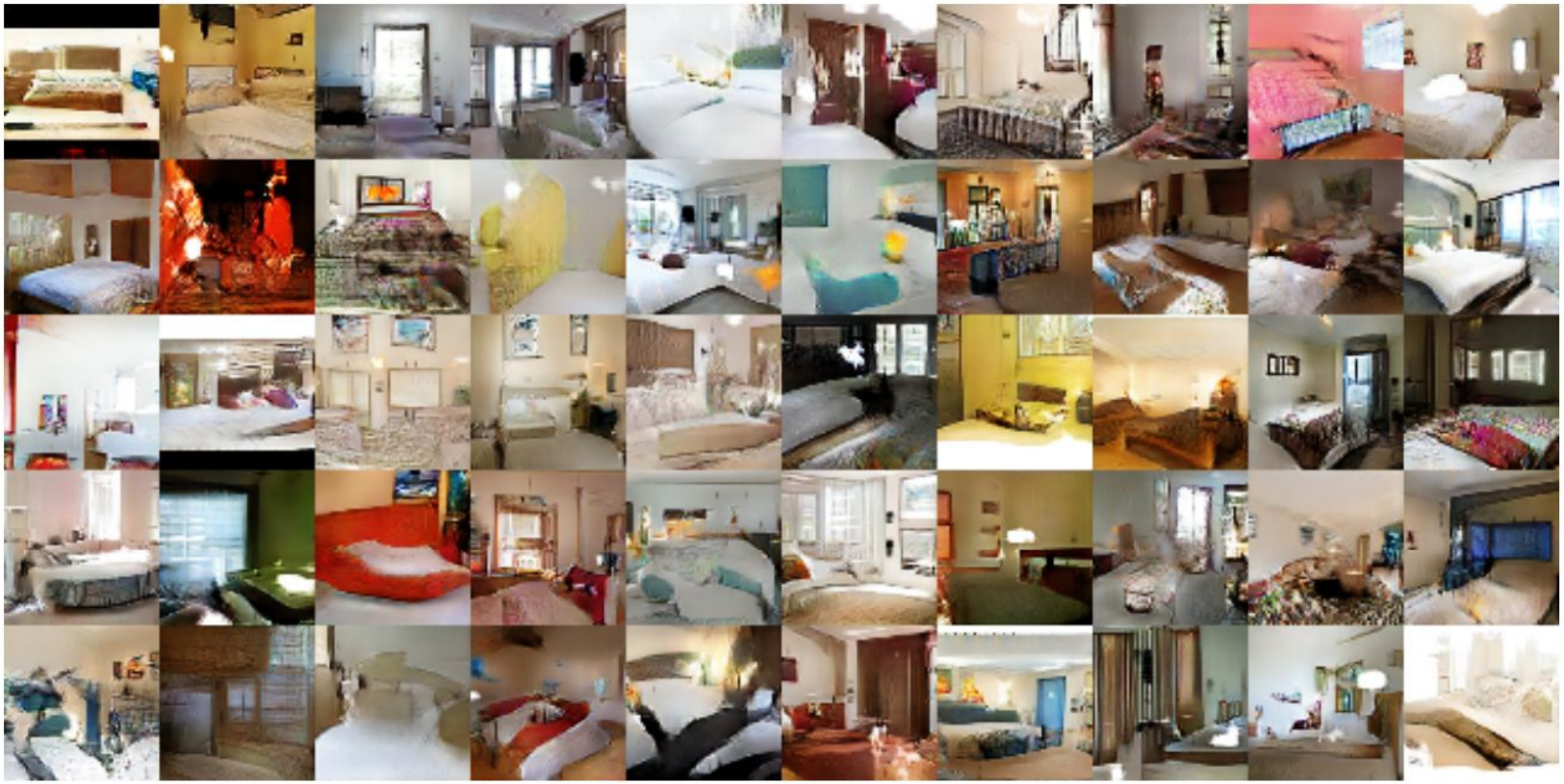
Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

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

Generative Adversarial Nets: Convolutional Architectures

Samples from the model look much better!



Radford et al, ICLR 2016

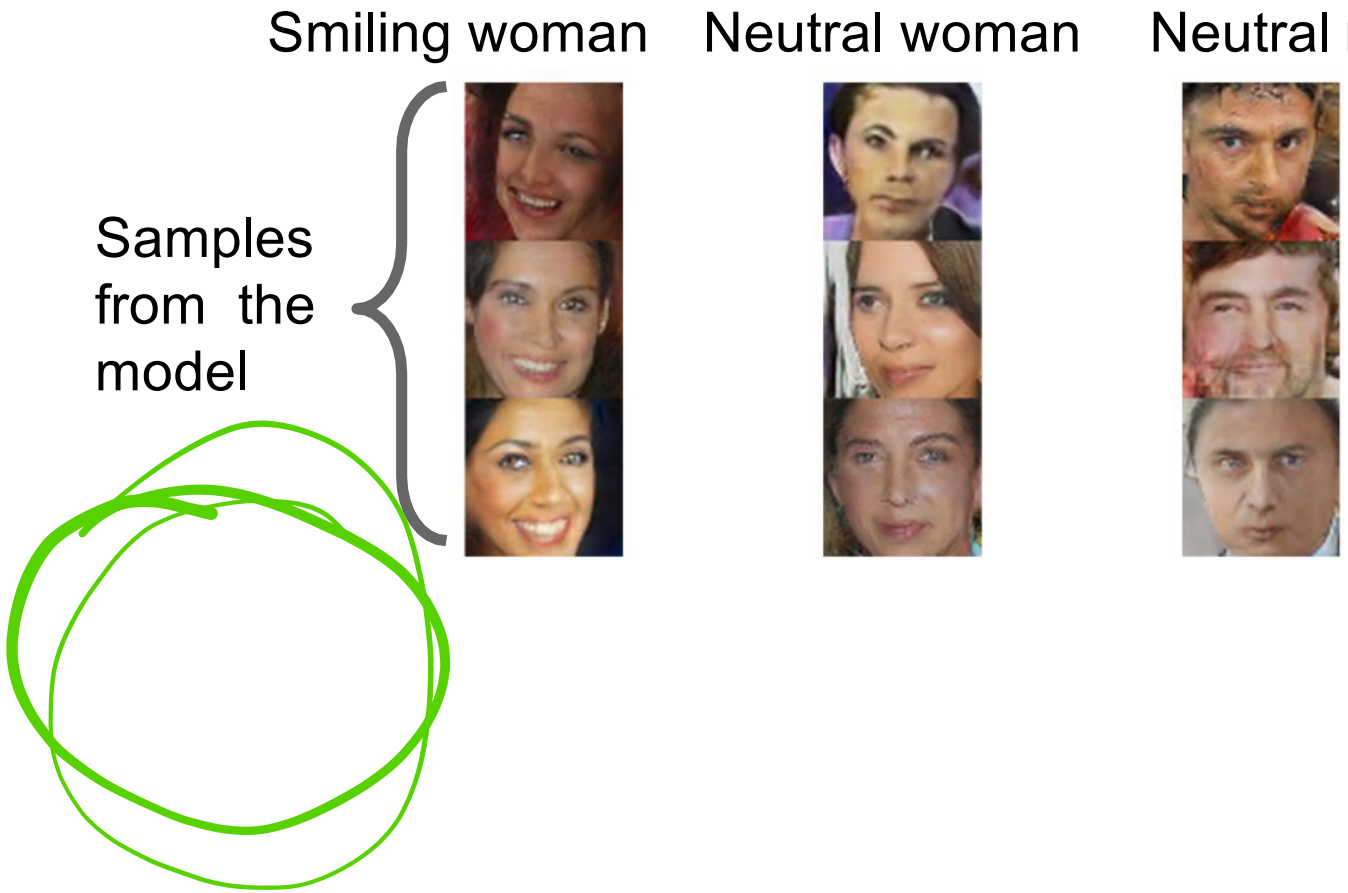
Generative Adversarial Nets: Convolutional Architectures

Interpolating between random points in latent space

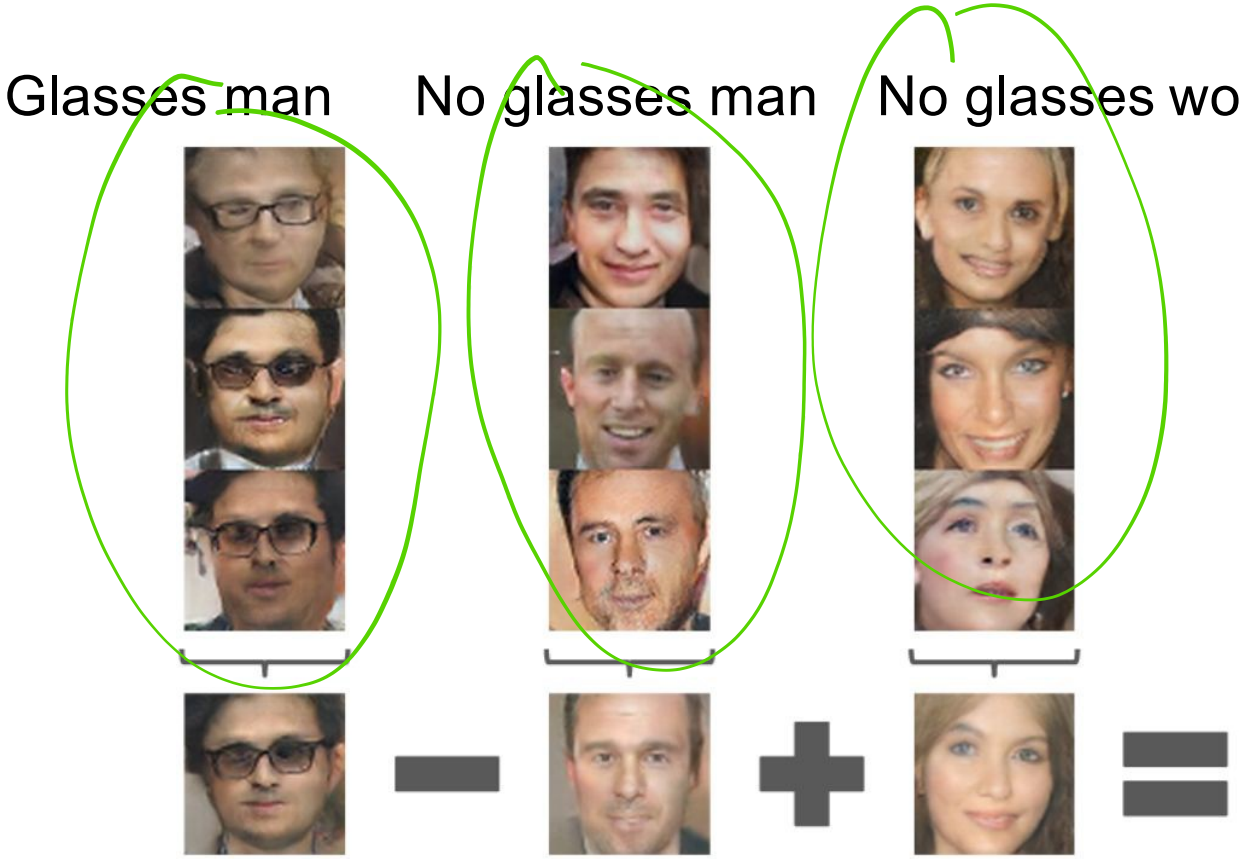
Radford et al, ICLR 2016

Generative Adversarial Nets: Interpretable Vector Math

Radford et al, ICLR 2016



Generative Adversarial Nets: Interpretable Vector Math



Radford et al, ICLR 2016

2017: Explosion of GANs

See also: https://github.com/soumith/ganhacks for tips and tricks for trainings GANs

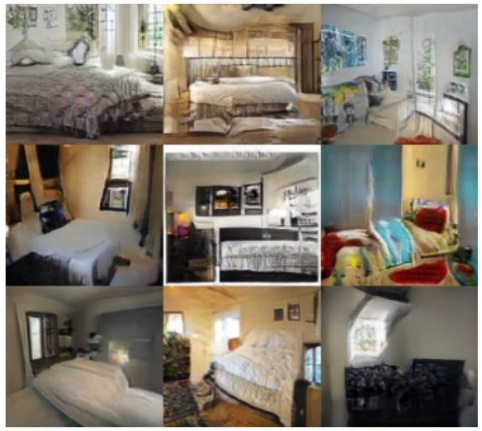
"The GAN Zoo"

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

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

2017: Explosion of GANs

Better training and generation



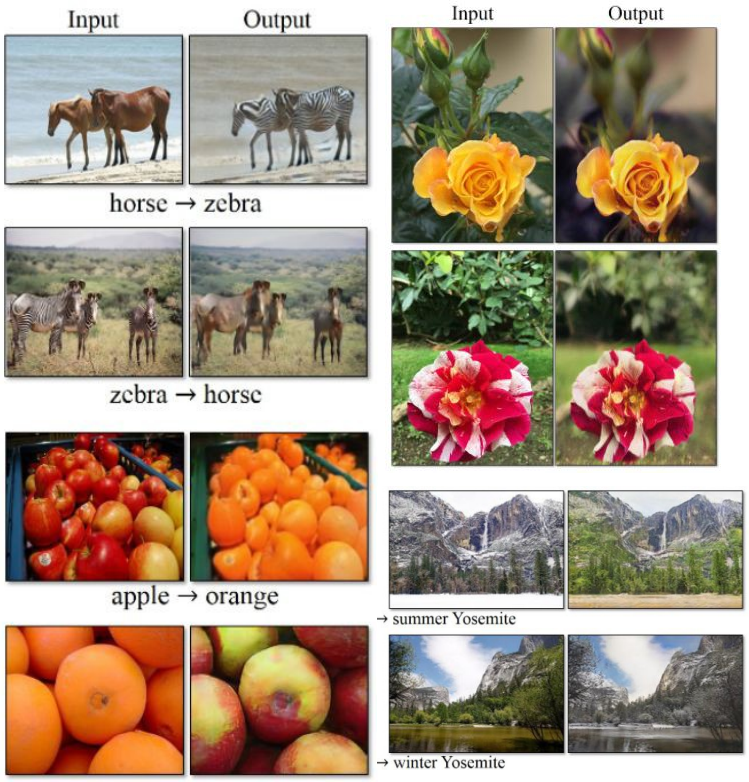
LSGAN, Zhu 2017.

Wasserstein GAN, Arjovsky 2017. Improved Wasserstein GAN, Gulrajani 2017.

Progressive GAN, Karras 2018.

2017: Explosion of GANs

Source->Target domain transfer



CycleGAN. Zhu et al. 2017.

Text -> Image Synthesis

Reed et al. 2017.

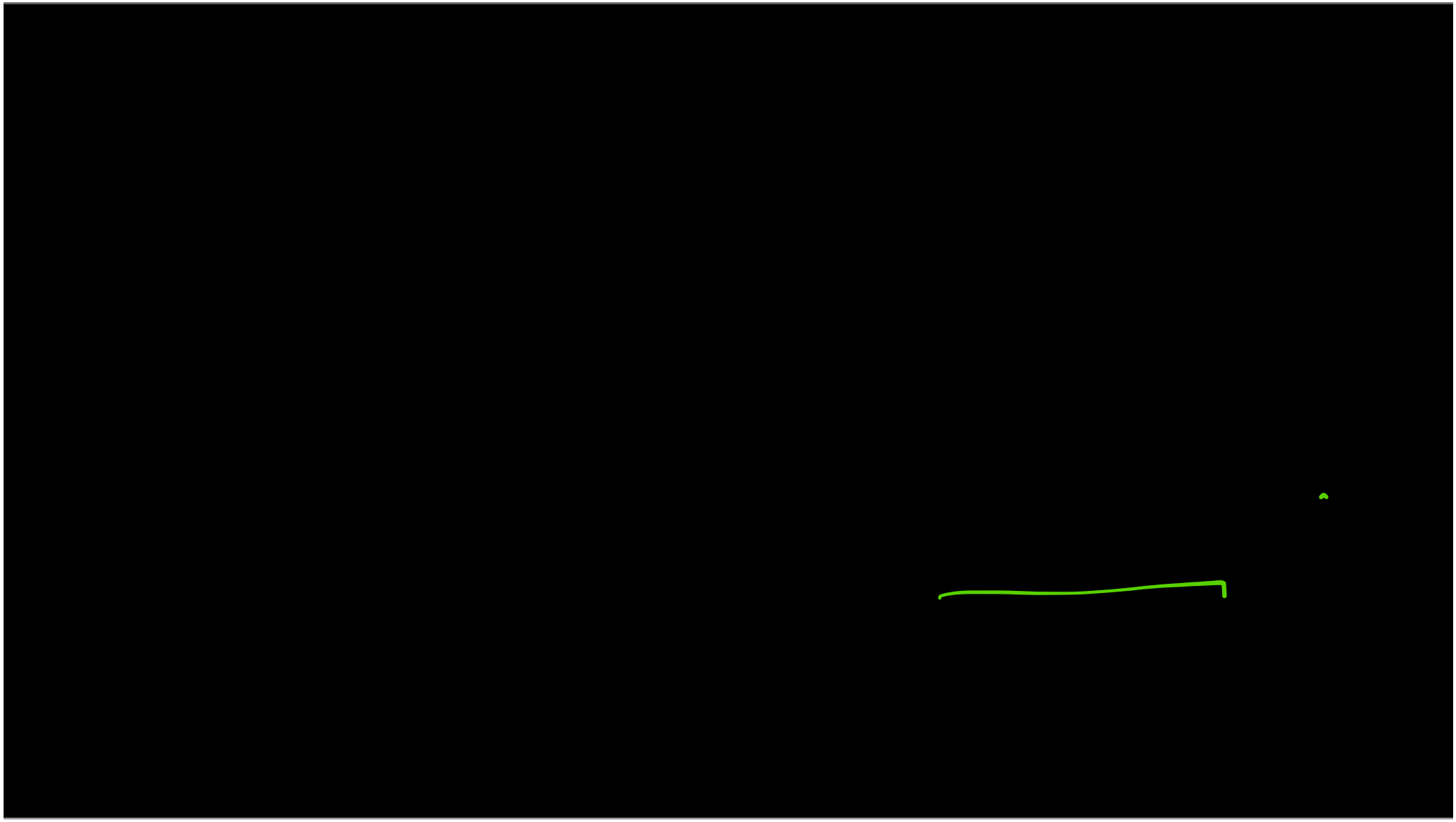
Many GAN applications

Pix2pix. Isola 2017. Many examples at https://phillipi.github.io/pix2pix/

2019: BigGAN

Brock et al., 2019

DeepFake



Theory of GAN

- Generator:
  minimize distance $P_{data}(x),\ P_z(x;\theta)$
- Two player minimax game

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

$$C(\omega) = \frac{1}{m}\sum_{i=1}^{m} Cost(y_\omega(x), t)$$

$$Cost(y_\omega(x), t) = \begin{cases} -\log\left(1 - y_\omega(x)\right) & \text{if } t = 0 \\ -\log\left(y_\omega(x)\right) & \text{if } t = 1 \end{cases}$$

$$C(\omega) = -\frac{1}{m}\sum_{i=1}^{m} t^{(i)} \log\left(y_\omega\left(x^{(i)}\right)\right) + \left(1 - t^{(i)}\right) \log\left(1 - y_\omega\left(x^{(i)}\right)\right)$$

Proposition 1. *For G fixed, the optimal discriminator D is*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

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

$$V(G,D) = \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_{z} p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz$$

$$= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx$$

$$\arg\max_y \; a \log y + b \log(1-y)$$

$$= \frac{a}{a+b}$$

$$\frac{a}{y} - \frac{b}{1-y}$$

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

$$= \mathbb{E}_{x\sim p_{data}}\left[\log \frac{p_{data}(x)}{\frac{1}{2}(p_{data}(x) + p_z(x))}\right] +$$

$$\mathbb{E}_{z\sim p_z(z)}\left[\log \frac{p_z(x)}{\frac{1}{2}(p_{data}(x) + p_z(x))}\right]$$

$$- 2\log 2$$

$$= KL\left(p_{data} \,\middle\|\, \frac{p_{data}+p_z}{2}\right) +$$

$$KL\left(p_z \,\middle\|\, \frac{p_{data}+p_z}{2}\right) - 2\log 2$$

$$= JS(p_{data} \,\|\, p_z) - 2\log 2$$

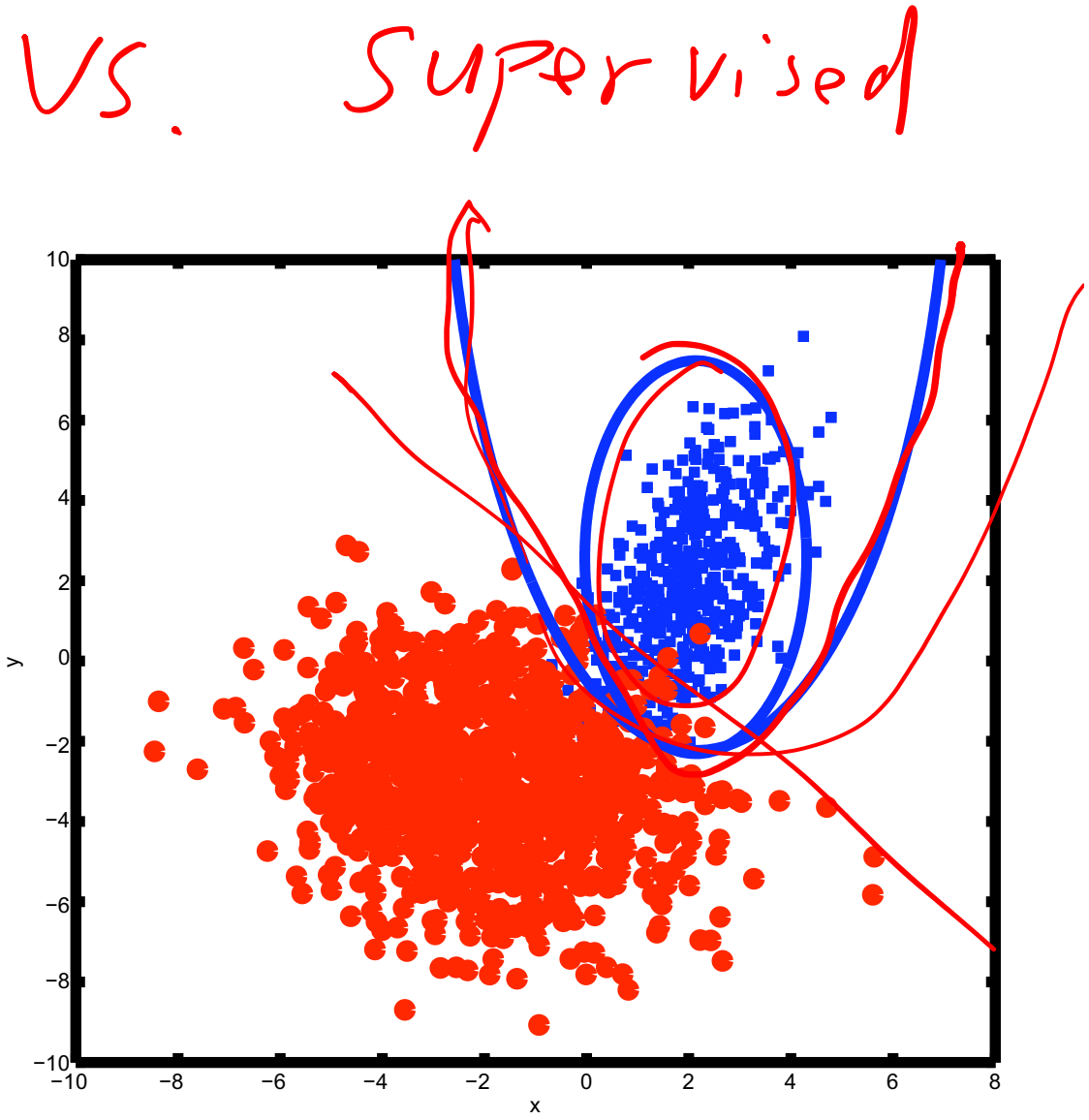
Useful Resources on Generative Models

CS 236: Deep Generative Models (Stanford)

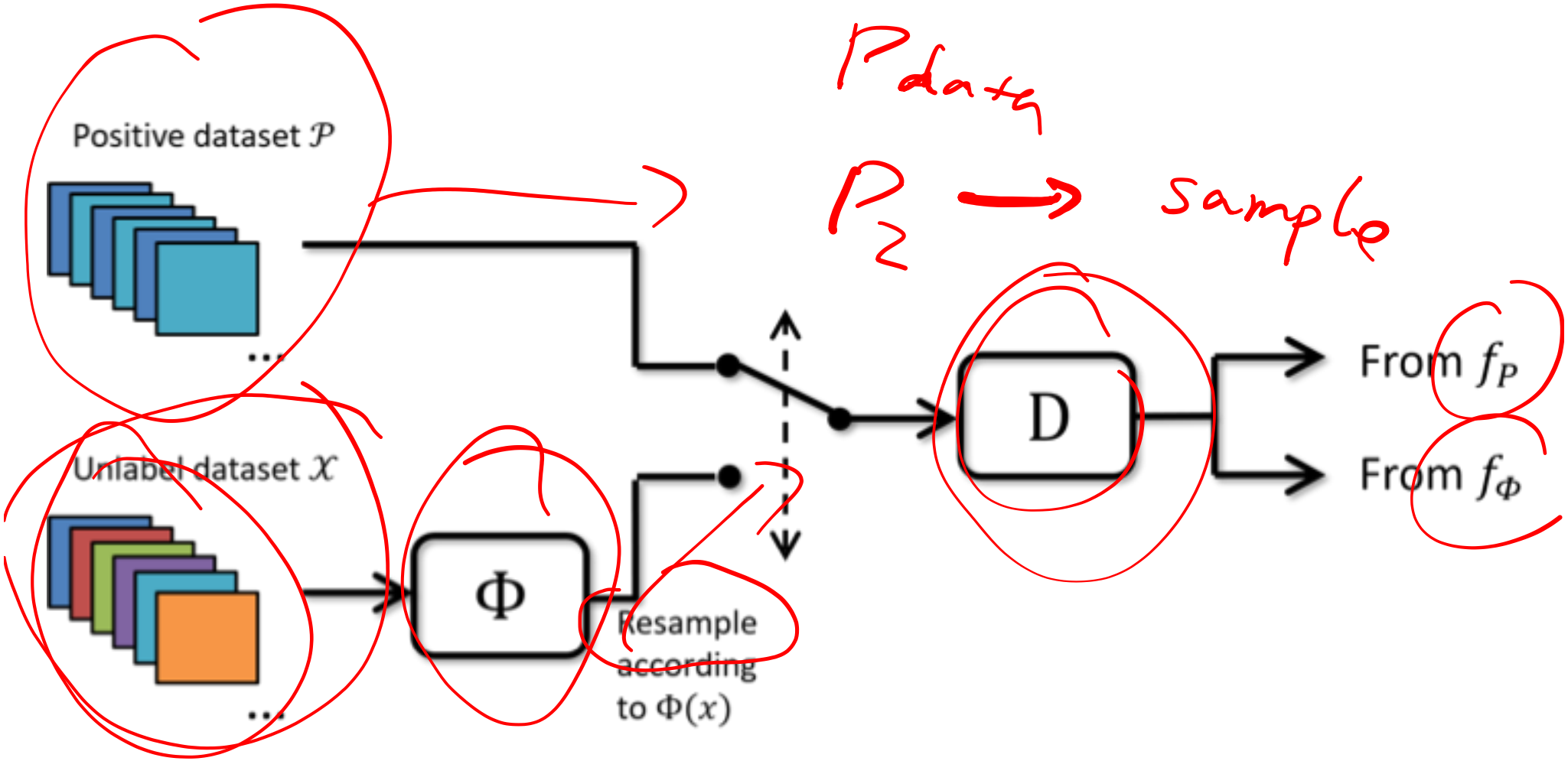
CS 294-158 Deep Unsupervised Learning (Berkeley)

PU Learning

- P: positive
- U: unlabeled



PU Learning with Adversarial Nets



PU Learning Loss

$$\min_{\Phi} \max_{D} J_{\mathcal{PU}}(\Phi, D) = \mathbb{E}_{p_{\mathcal{P}}}[\log D(x)]$$

$$+ \mathbb{E}_{p_{\Phi}}[\log(1 - D(x))]$$

$$\approx \frac{\sum_{x \in \mathcal{P}} \log D(x)}{M} + \frac{\sum_{x \in \mathcal{U}} \Phi(x) \log(1 - D(x))}{\sum_{x' \in \mathcal{U}} \Phi(x')}$$

Experiments: 2-class

Avila 10430 train 10437 test labels A-I

Positive: [A,F] Negative: the rest

P,N,U:2000,2000,10430

SETTING	ACCURACY(%) (MEAN ± STD)
PU	87.1 ± 0.29
PNU	91.6 ± 0.19
PN (2000+2000)	86.4 ± 0.24
PN (all labeled)	94.1 ± 0.05

Experiments: 3-class

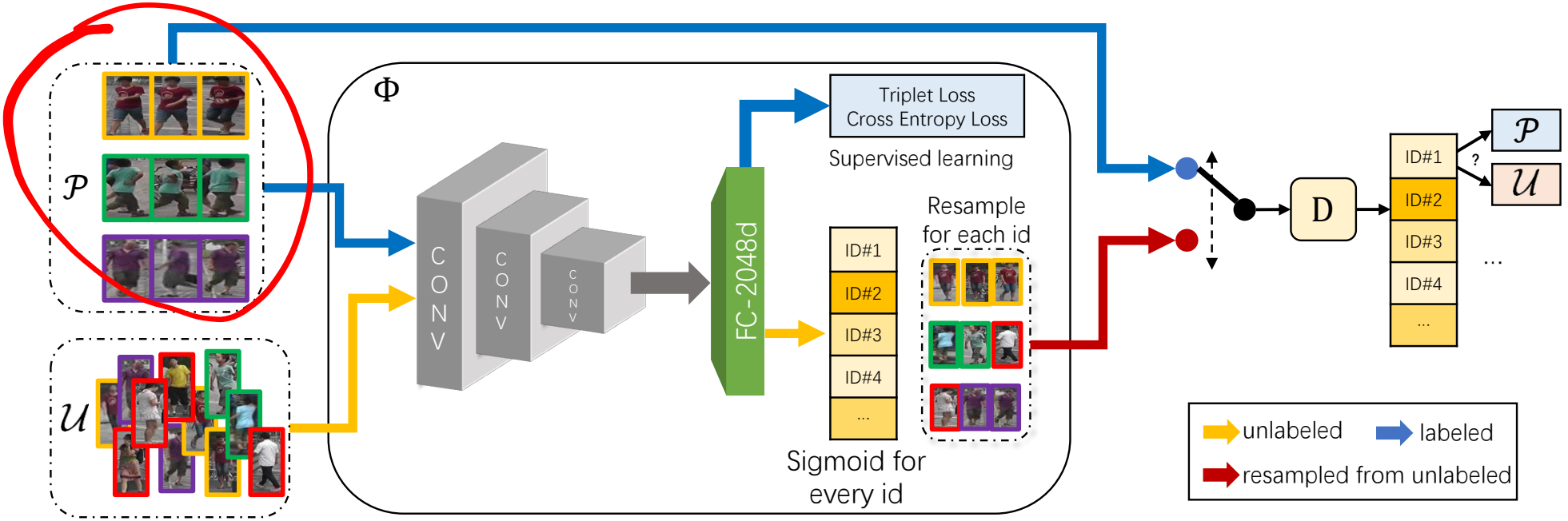
Avila 10430 train 10437 test labels A-I

class 0: [A] class 1: [E,F] class 2: the rest

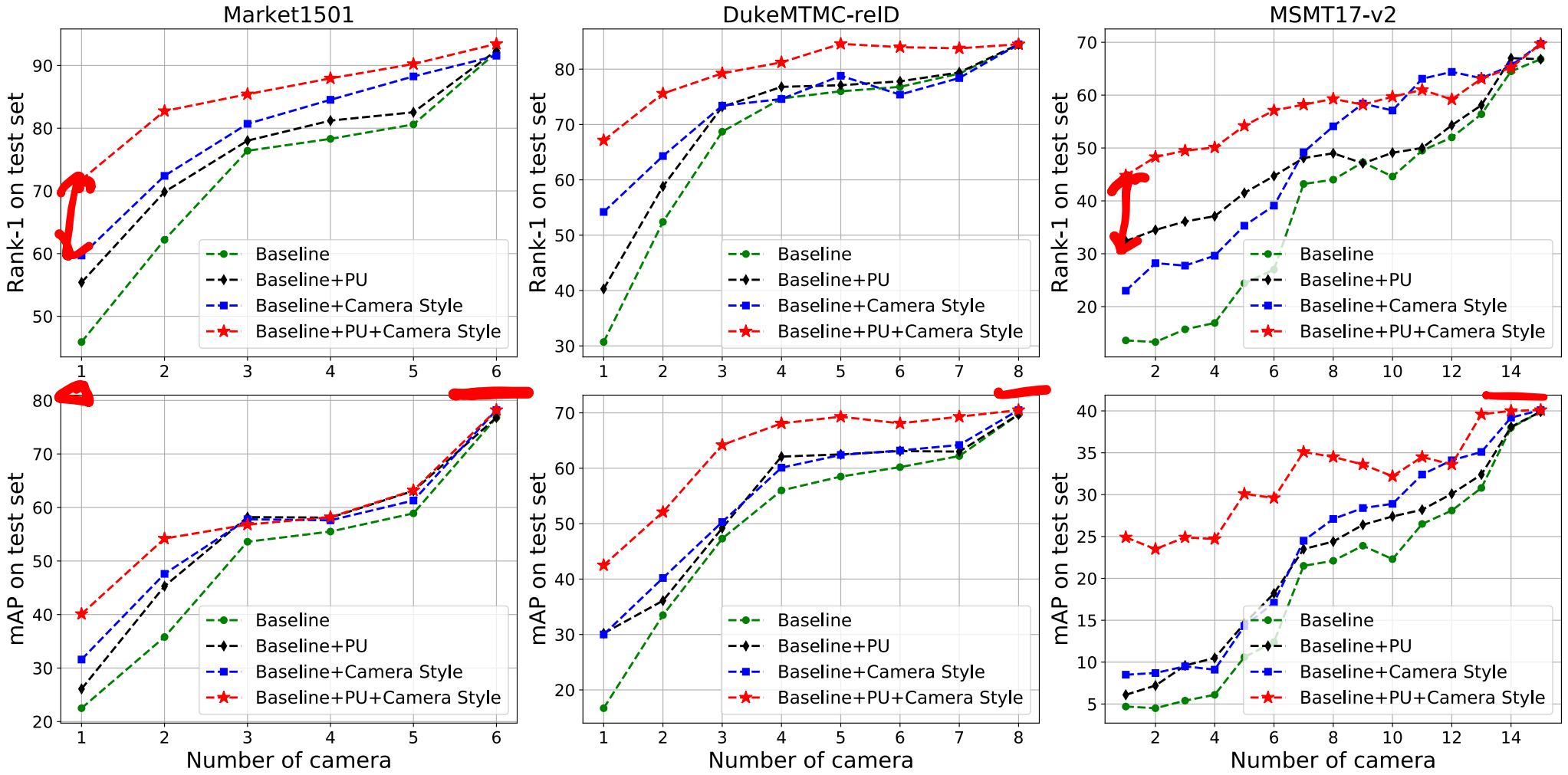
0,1,2,U:1000,1000,1000,10430

SETTING	ACCURACY(%) (MEAN ± STD)
0,1,2,U	77.2 ± 0.24
0,1,2 (1000*3)	67.3 ± 0.48
0,1,2 (all labeled)	81.7 ± 1.38

PU Learning for Person ReID



PU Learning for Person ReID




Q&A