



Topics in AI (CPSC 532S): Multimodal Learning with Vision, Language and Sound

Lecture 17: Generative Models [part 3], GANs

Logistics

Project Proposals were due Yesterday

- Will try to grade and provide feedback over the weekend

This week:

- Start working on **projects**
- Start thinking about **paper presentations**

Variational Autoencoders (VAE)

So far ...

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

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

So far ...

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

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

VAEs define intractable density function with latent variables z (that we need to marginalize):

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

cannot optimize directly, derive and optimize lower bound of likelihood instead

Variational Autoencoder: Learning

Putting it all together:

maximizing the likelihood lower bound

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

Lets look at **computing the bound** (forward pass)
for a given mini batch of input data

Input Data

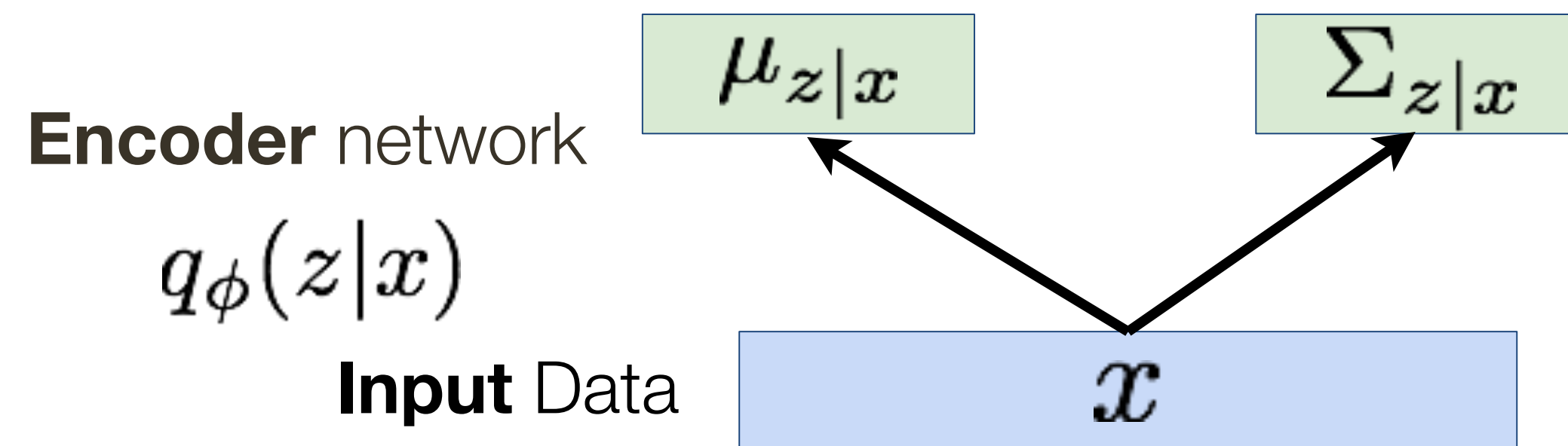
\mathcal{X}

Variational Autoencoder: Learning

Putting it all together:

maximizing the likelihood lower bound

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



Variational Autoencoder: Learning

Putting it all together:

maximizing the likelihood lower bound

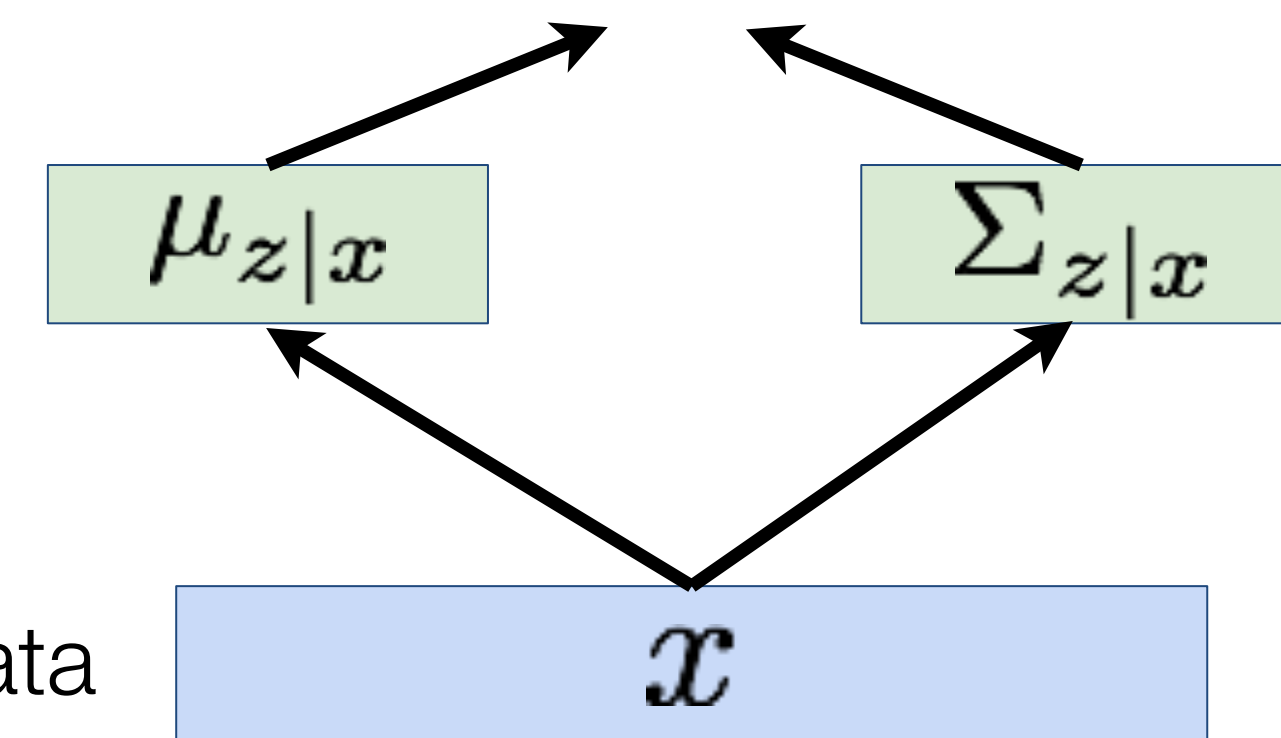
$$\underbrace{\mathbf{E}_z \left[\log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Make approximate posterior distribution close to prior

Encoder network

$$q_\phi(z|x)$$

Input Data



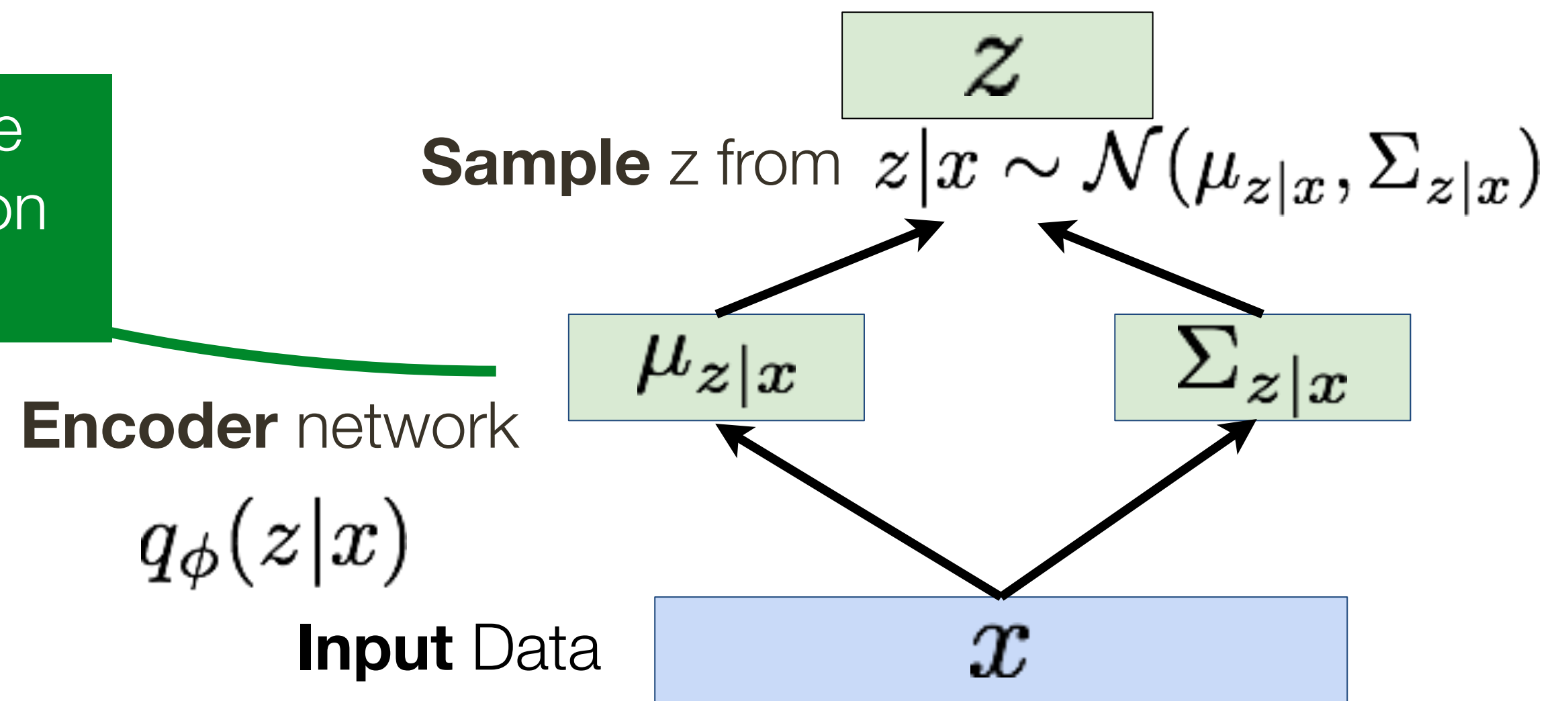
Variational Autoencoder: Learning

Putting it all together:

maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior



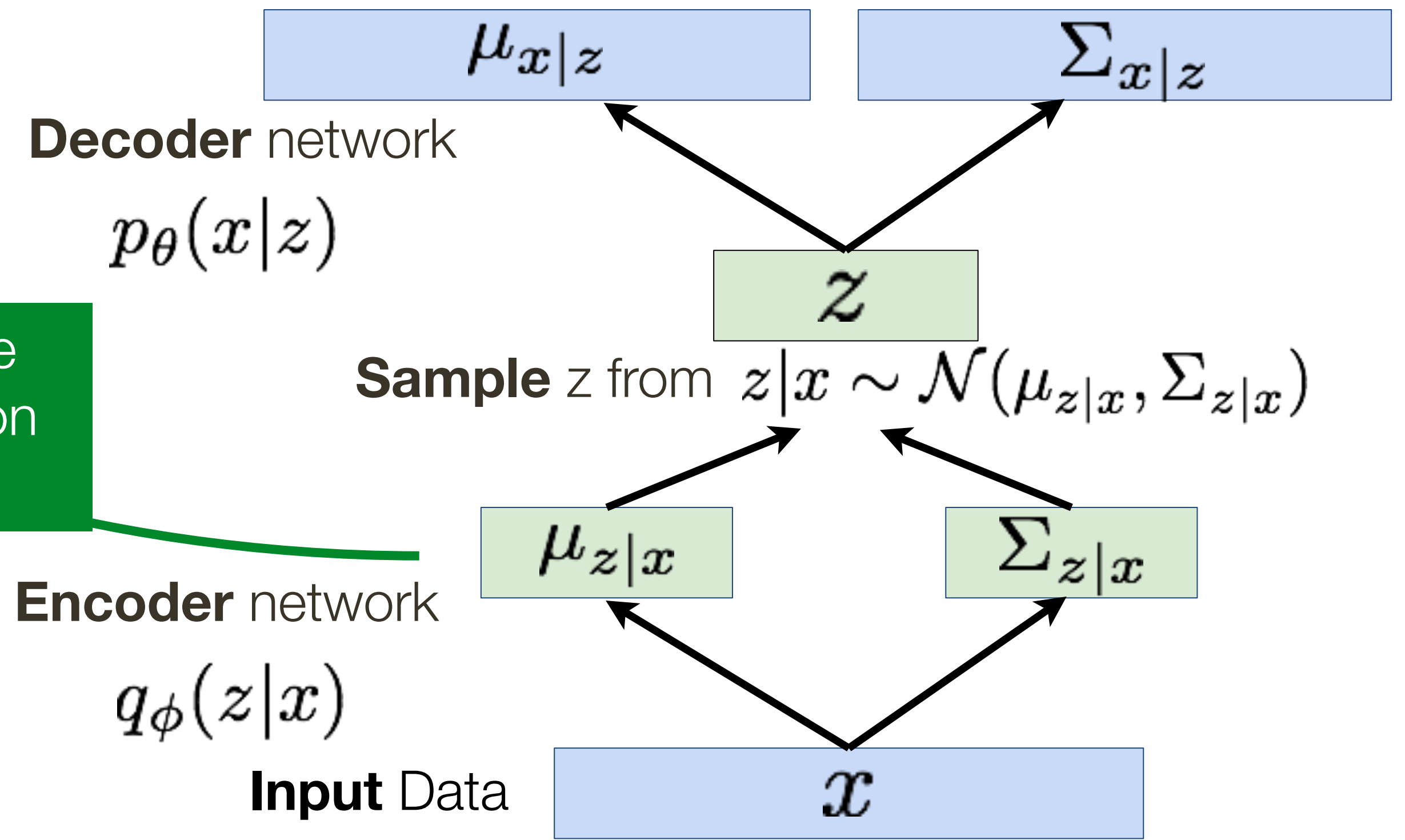
Variational Autoencoder: Learning

Putting it all together:

maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior



Variational Autoencoder: Learning

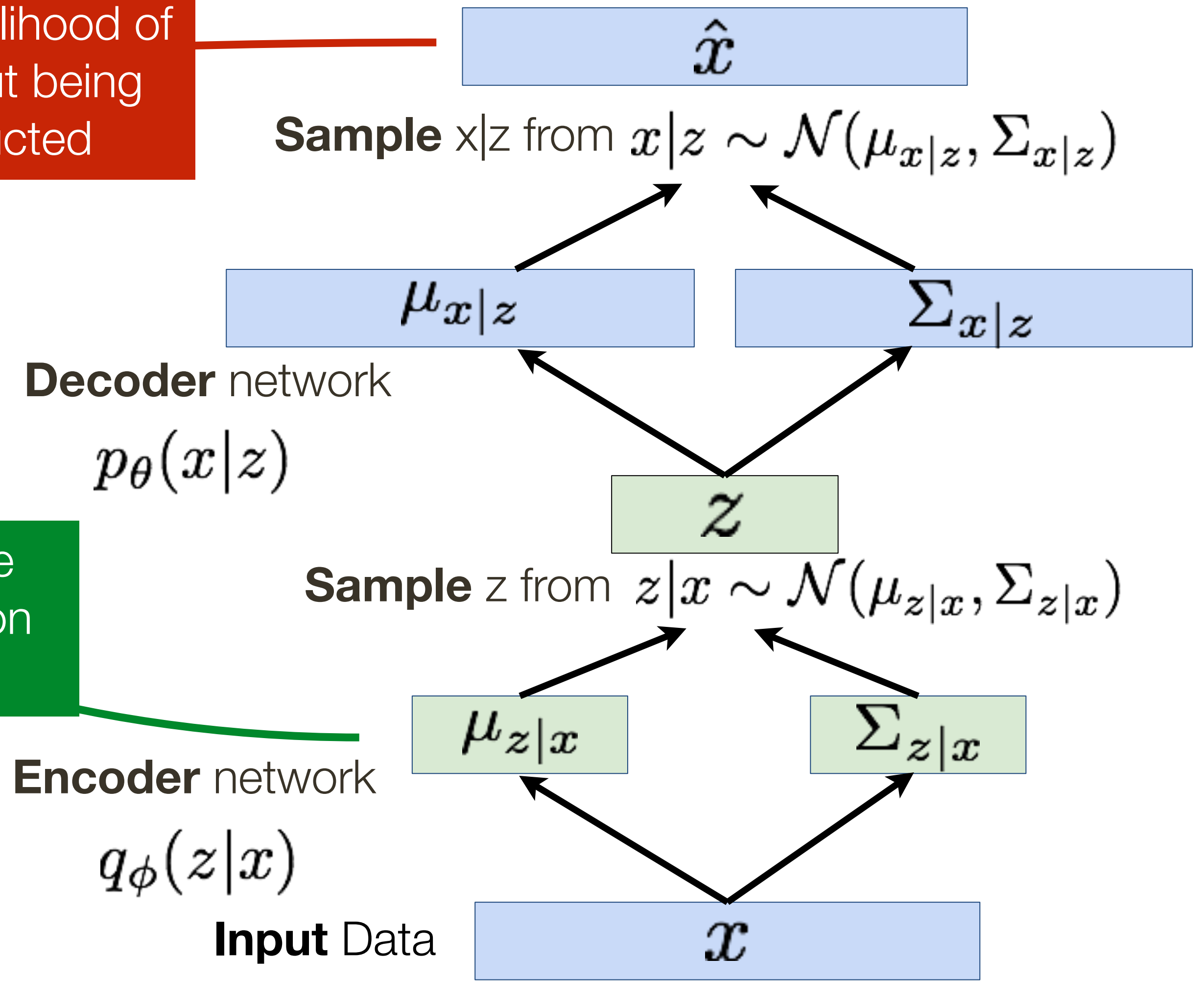
Putting it all together:

maximizing the likelihood lower bound

Maximize likelihood of original input being reconstructed

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

Make approximate posterior distribution close to prior



Variational Autoencoder: Learning

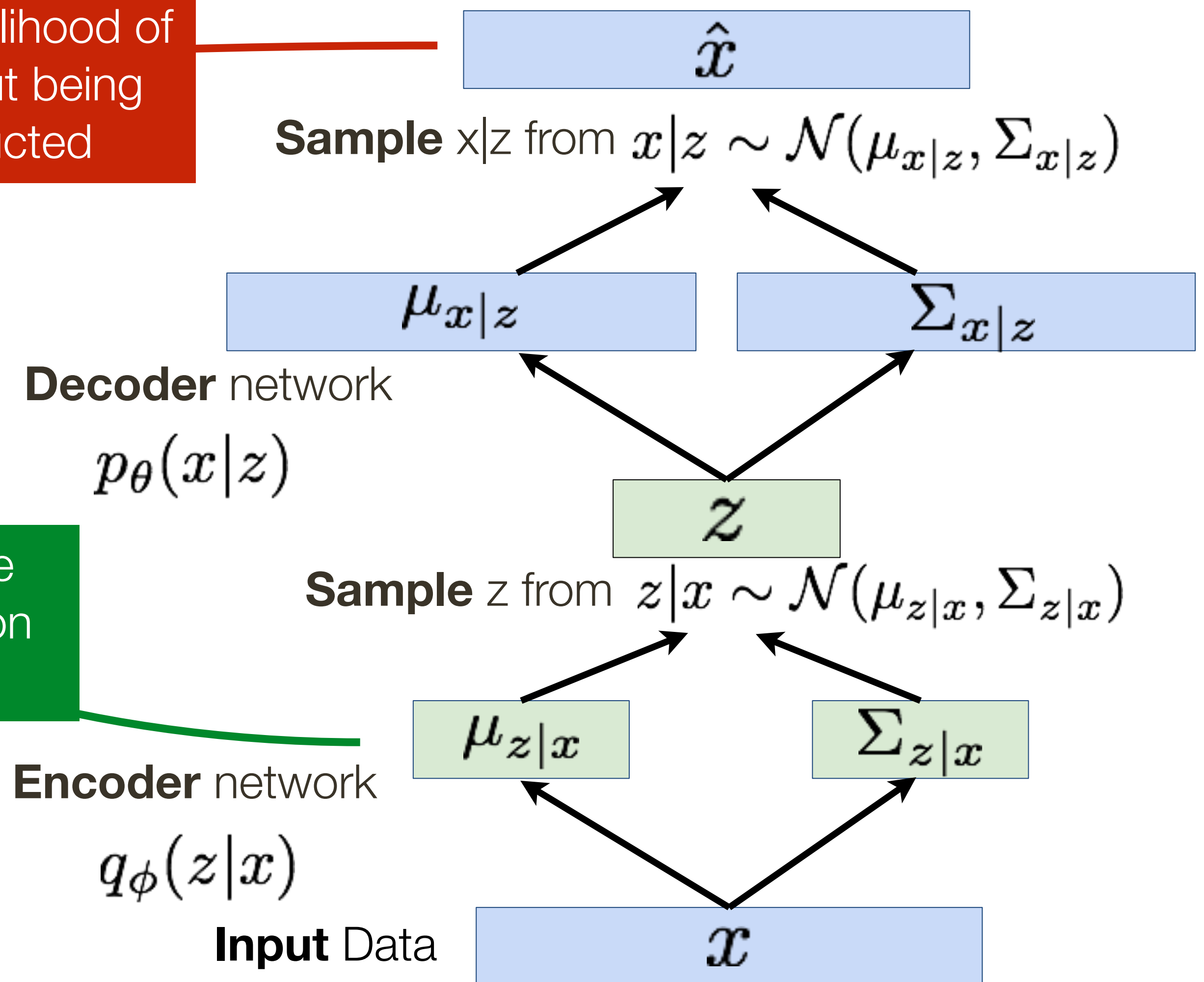
Putting it all together:

maximizing the likelihood lower bound

Maximize likelihood of original input being reconstructed

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

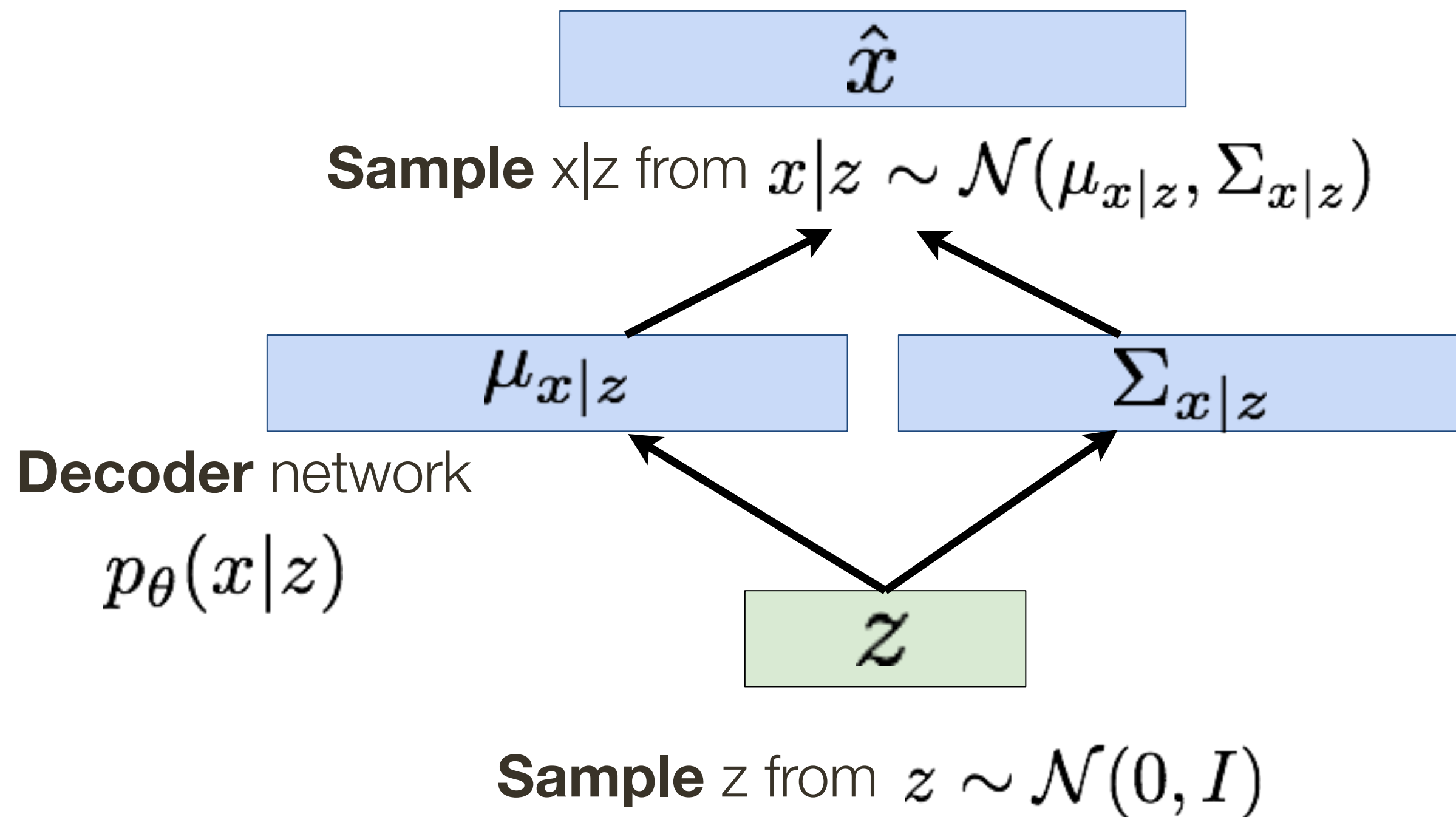
Make approximate posterior distribution close to prior



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

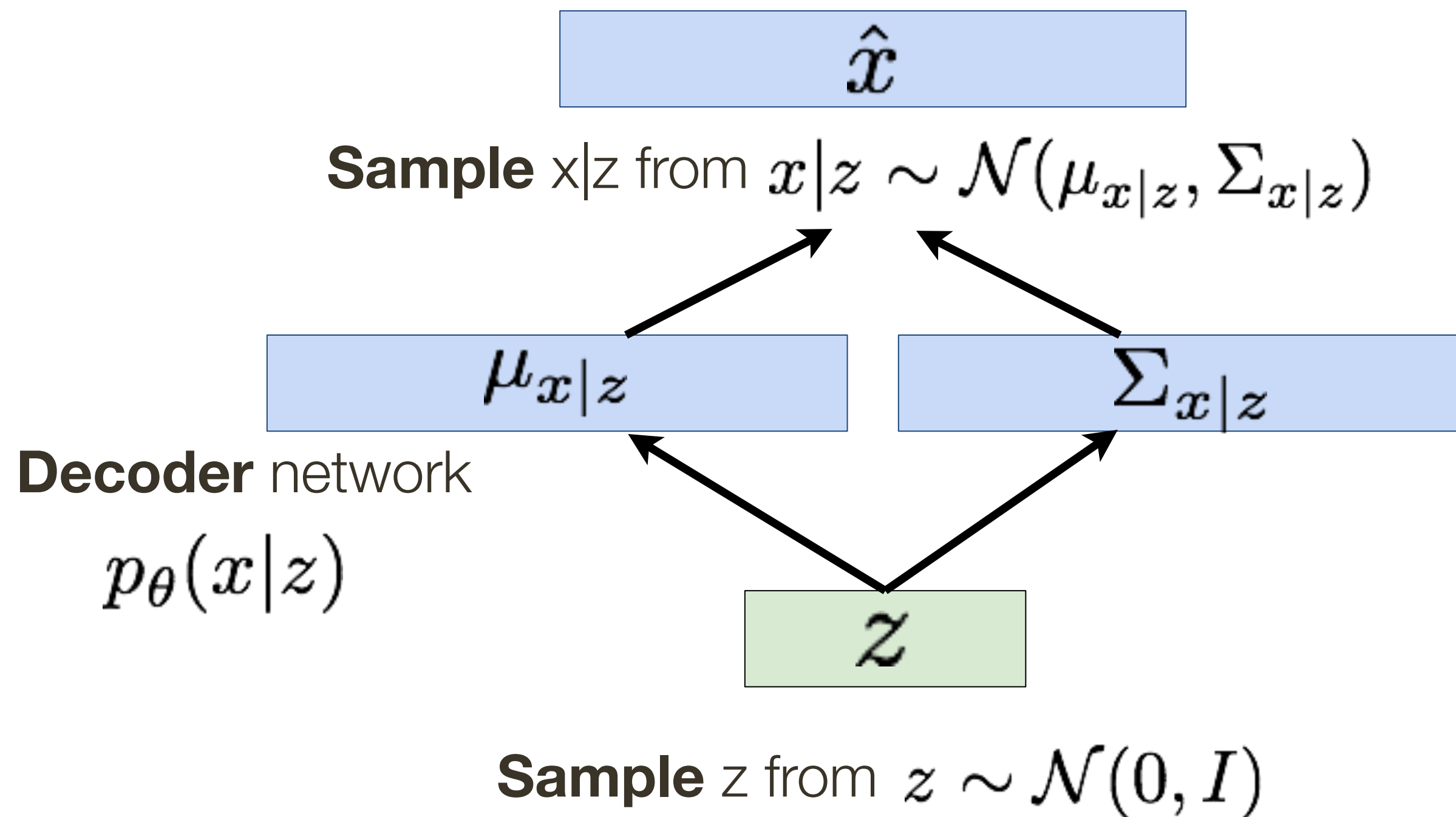
Variational Autoencoder: Generating Data

Use decoder network and sample z from **prior**

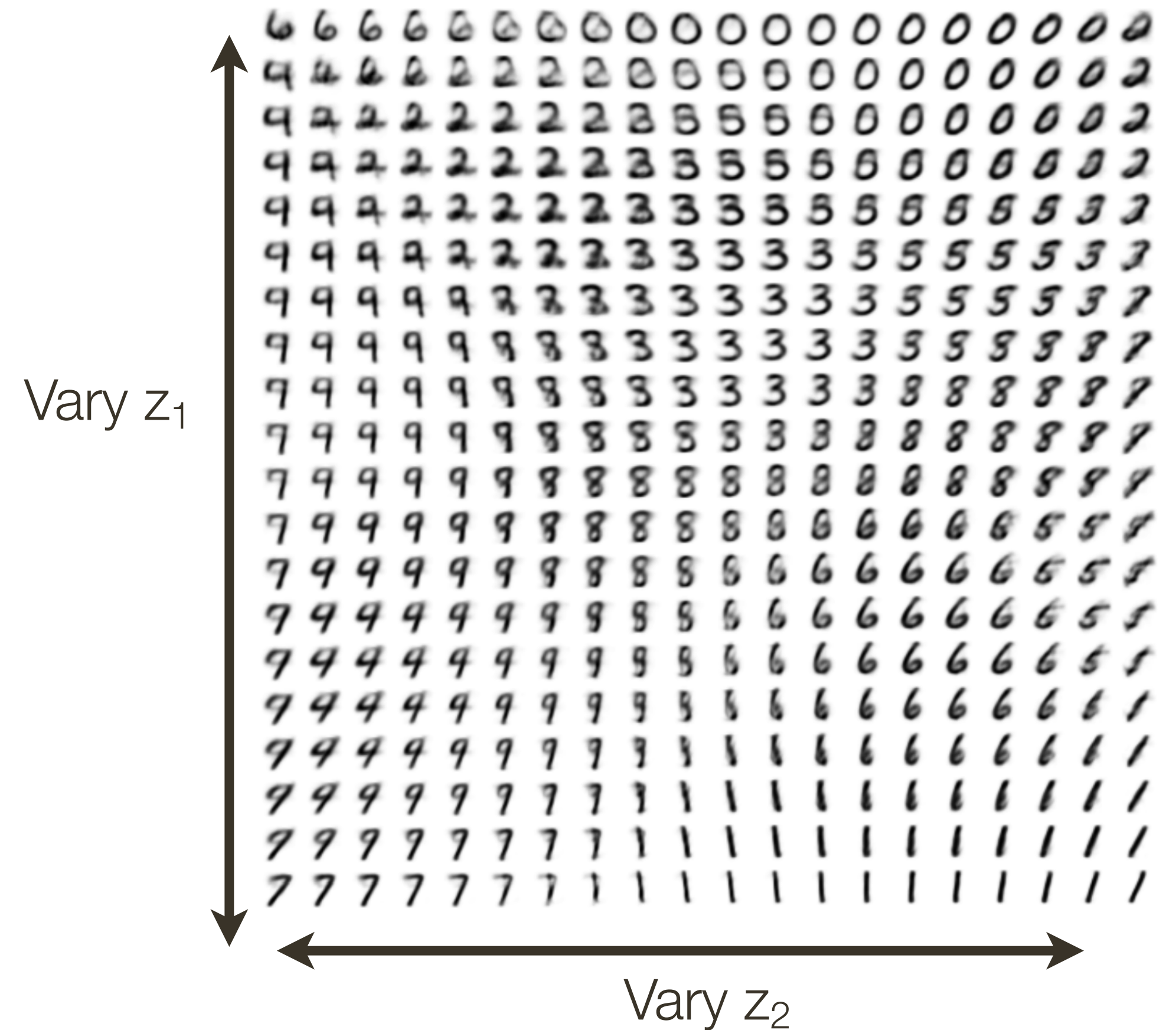


Variational Autoencoder: Generating Data

Use decoder network and sample z from **prior**



Data manifold for 2-d z

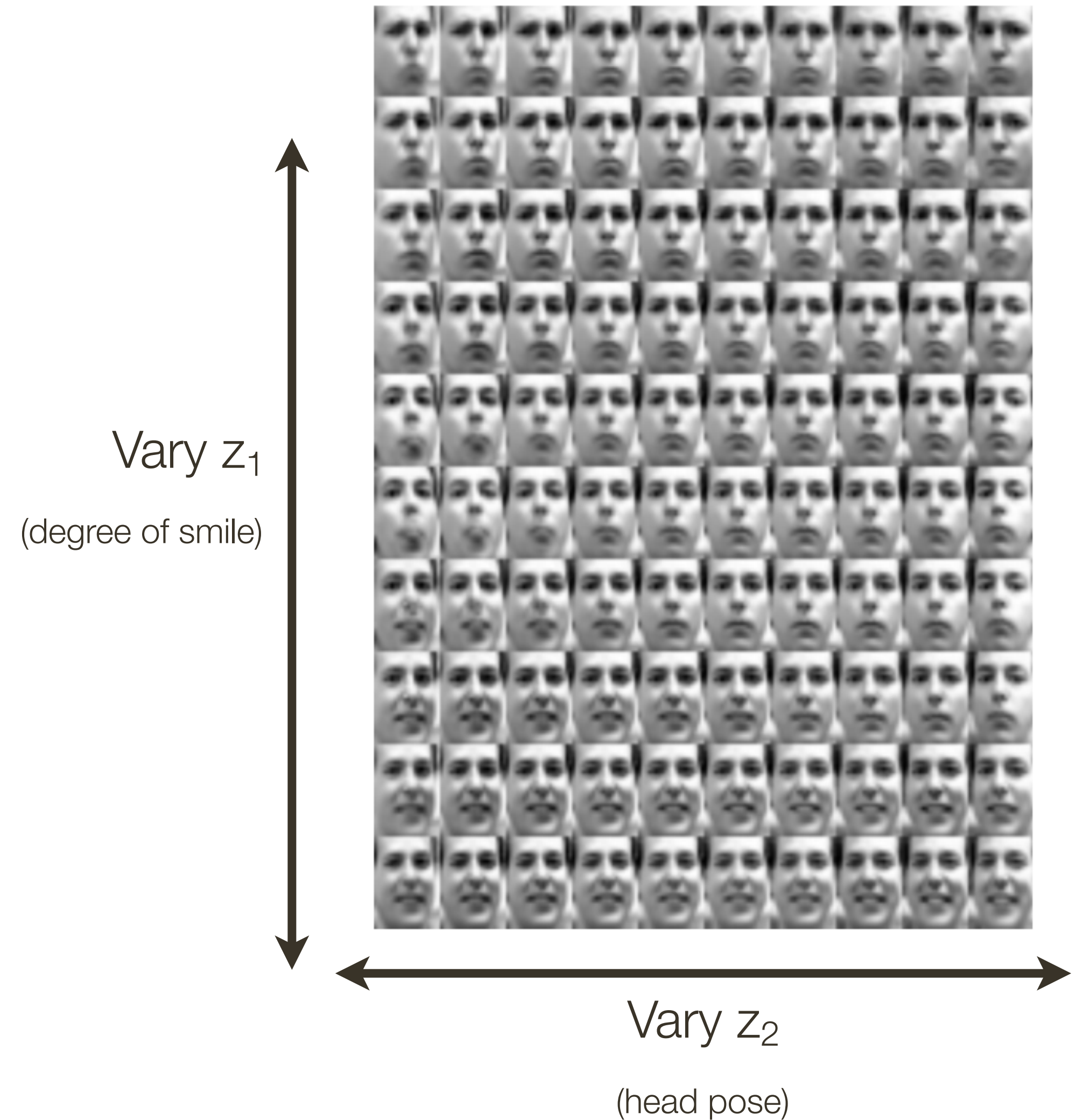


Variational Autoencoder: Generating Data

Diagonal prior on $z \Rightarrow$
independent latent variables

Different dimensions of z encode
interpretable factors of variation

Data manifold for 2-d z



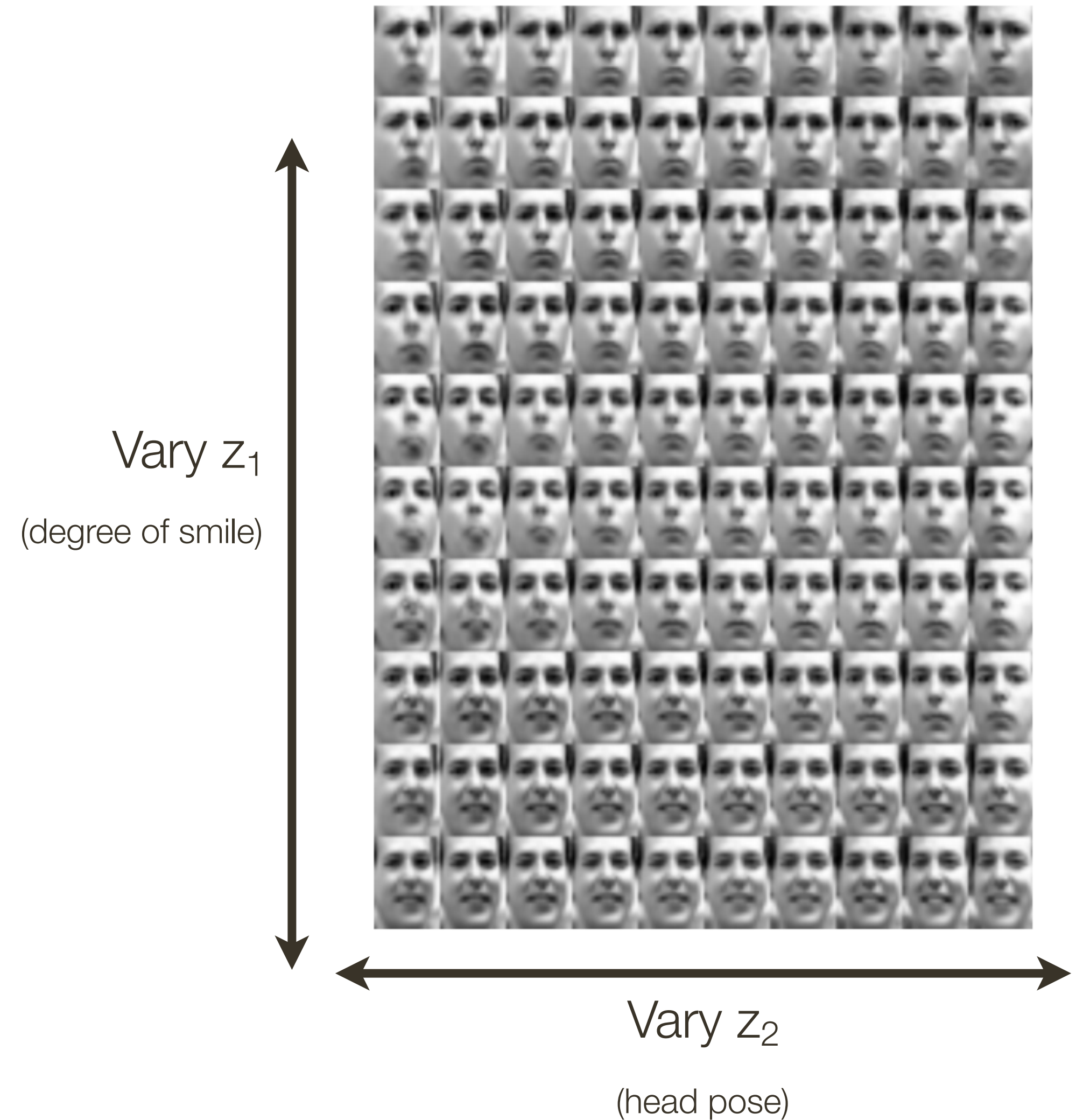
Variational Autoencoder: Generating Data

Diagonal prior on $z \Rightarrow$
independent latent variables

Different dimensions of z encode
interpretable factors of variation

Also good feature representation that can
be computed using $q_\phi(z|x)$!

Data manifold for 2-d z



Variational Autoencoder: Generating Data

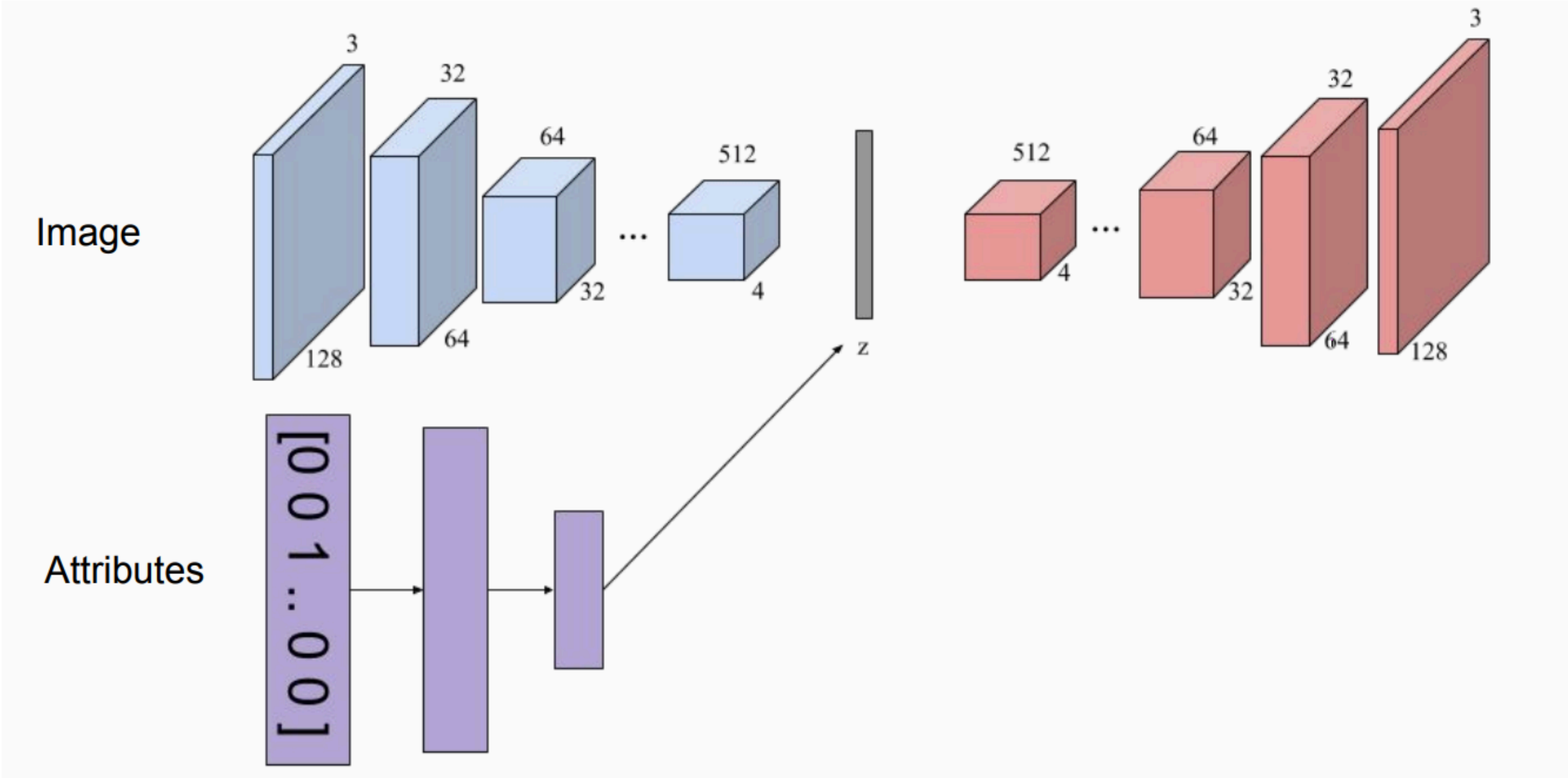


32x32 CIFAR-10

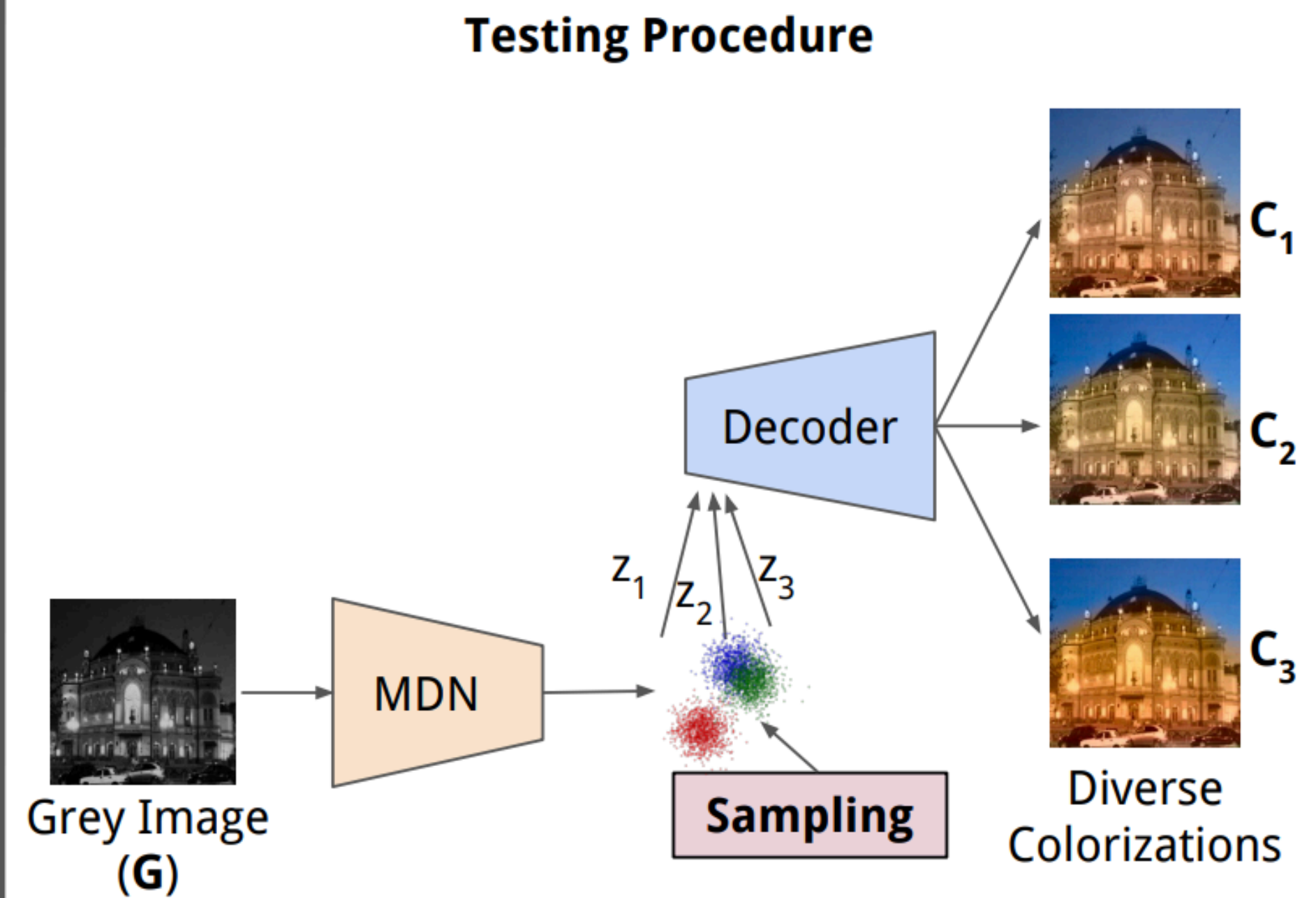
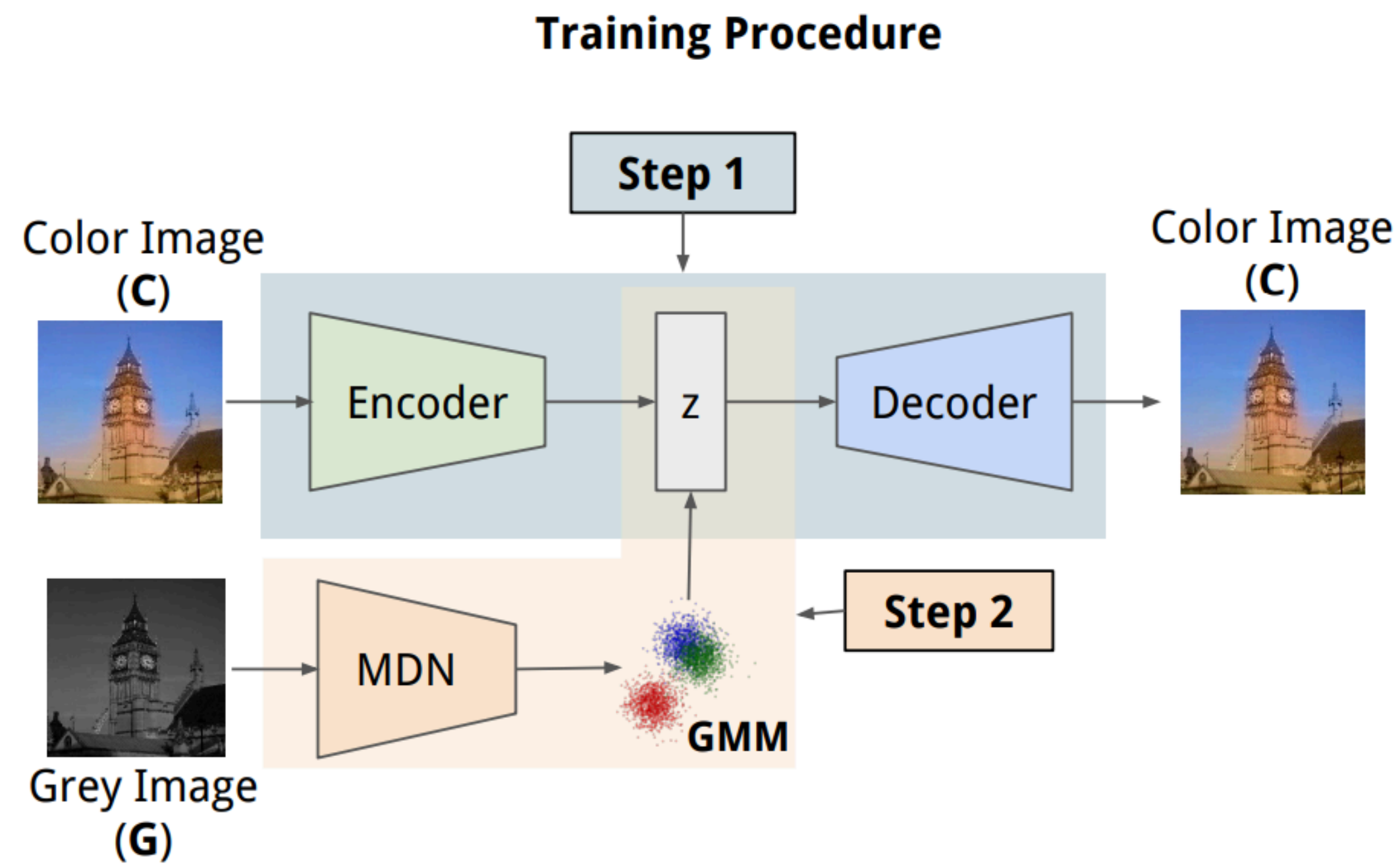


Labeled Faces in the Wild

Conditional VAEs

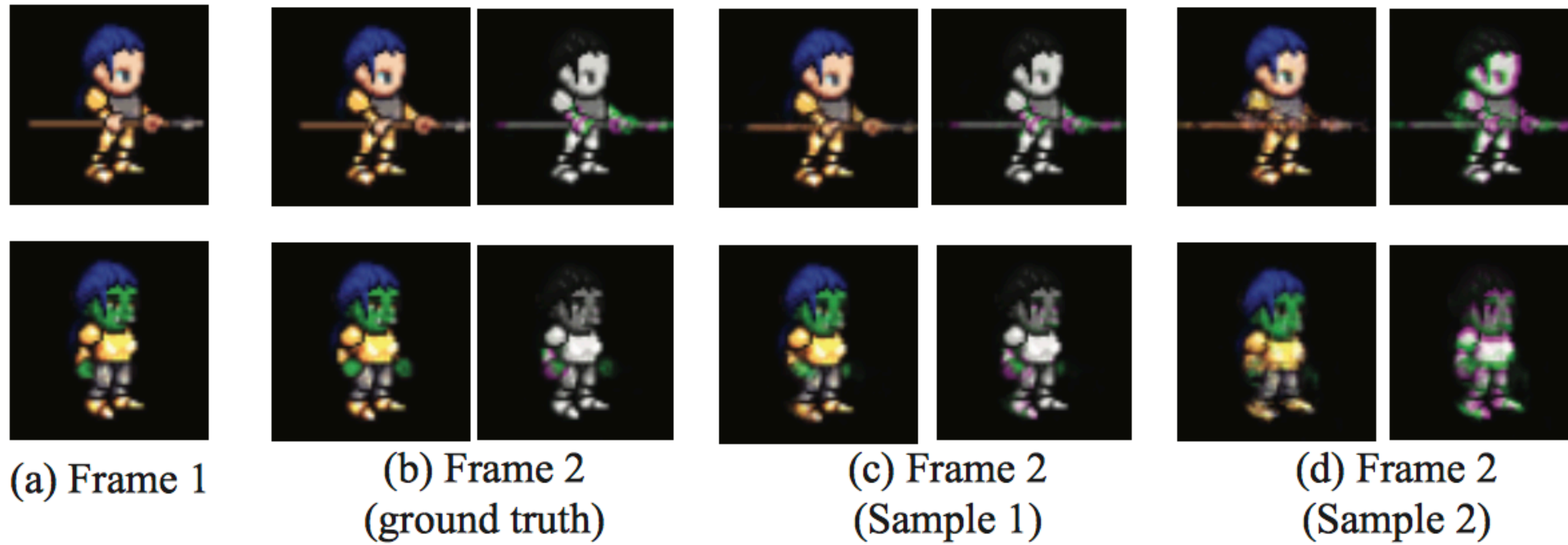


Conditional VAE: Diverse Image Colorization



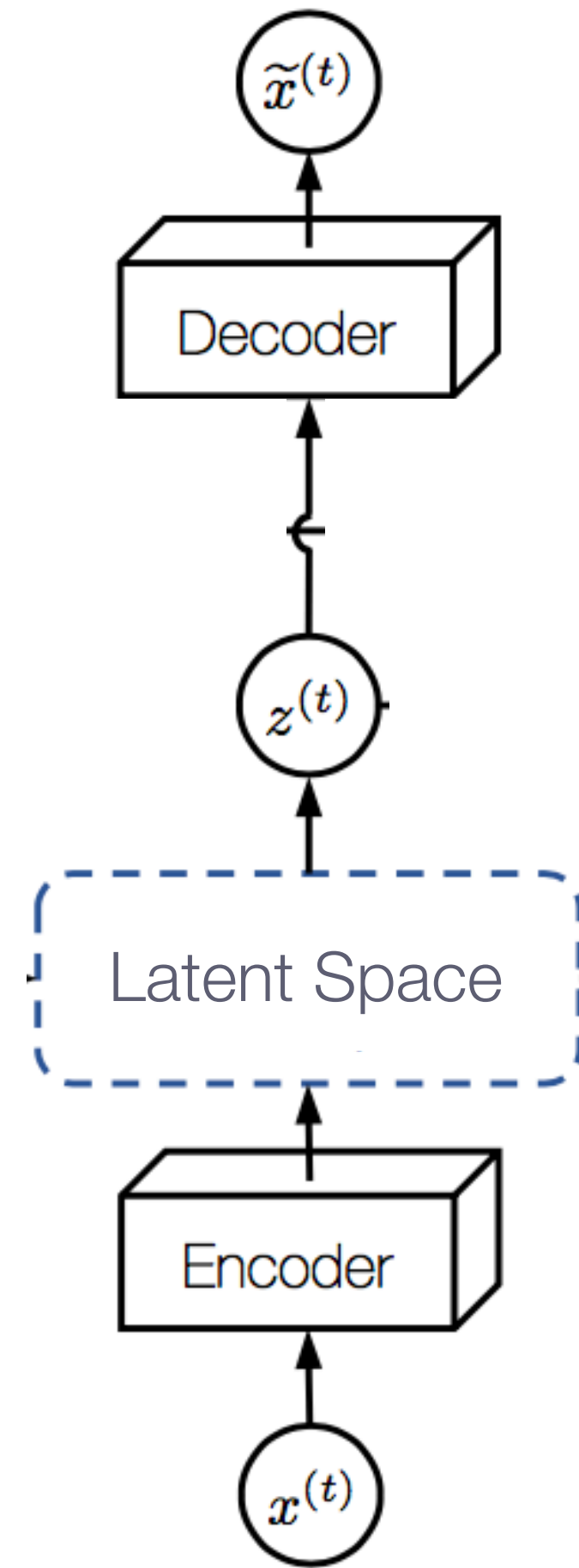
Conditional VAE: Temporal Predictions

[Xue et al., 2016]



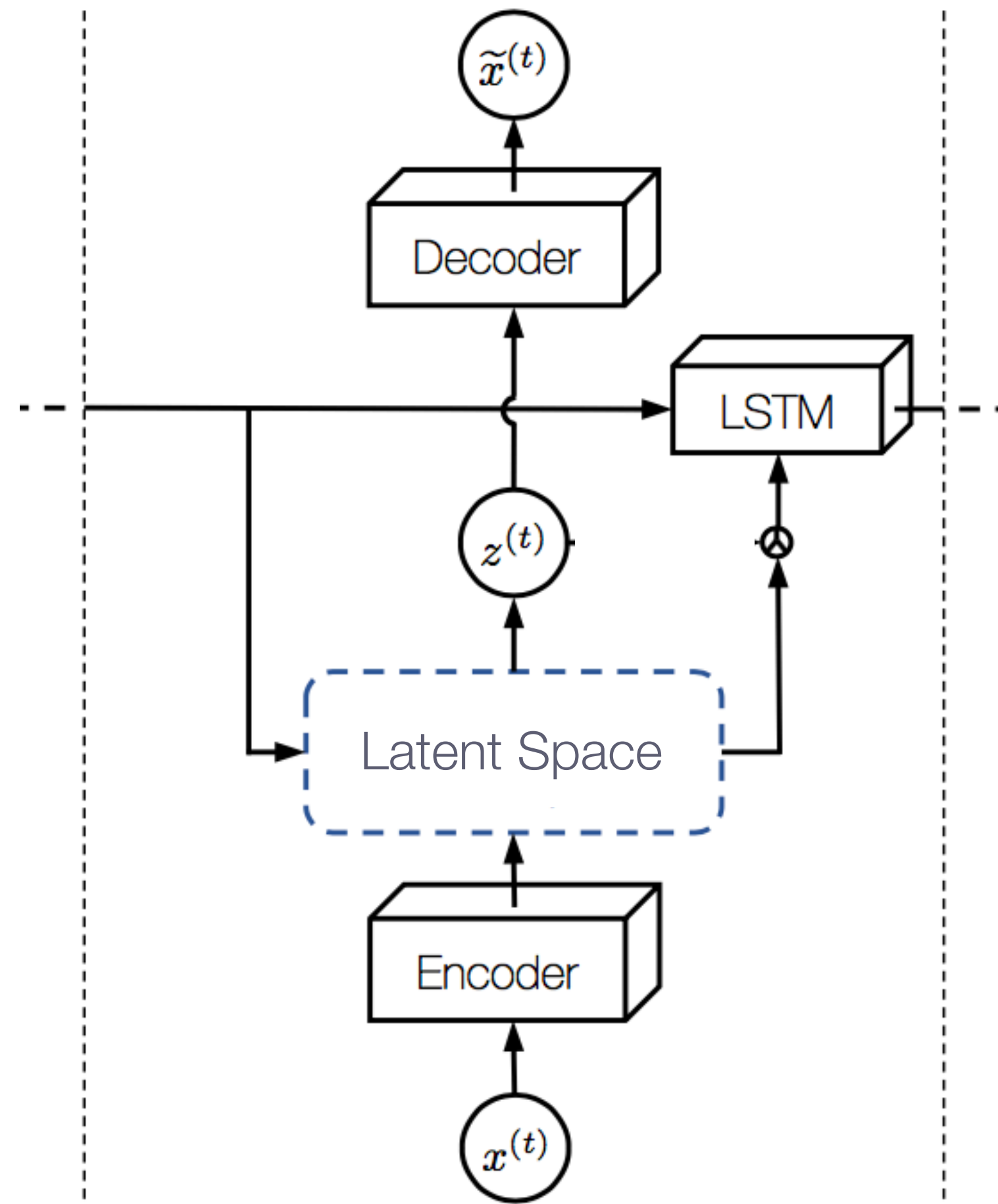
Variational Autoencoder (VAE)

[He et al., 2018]



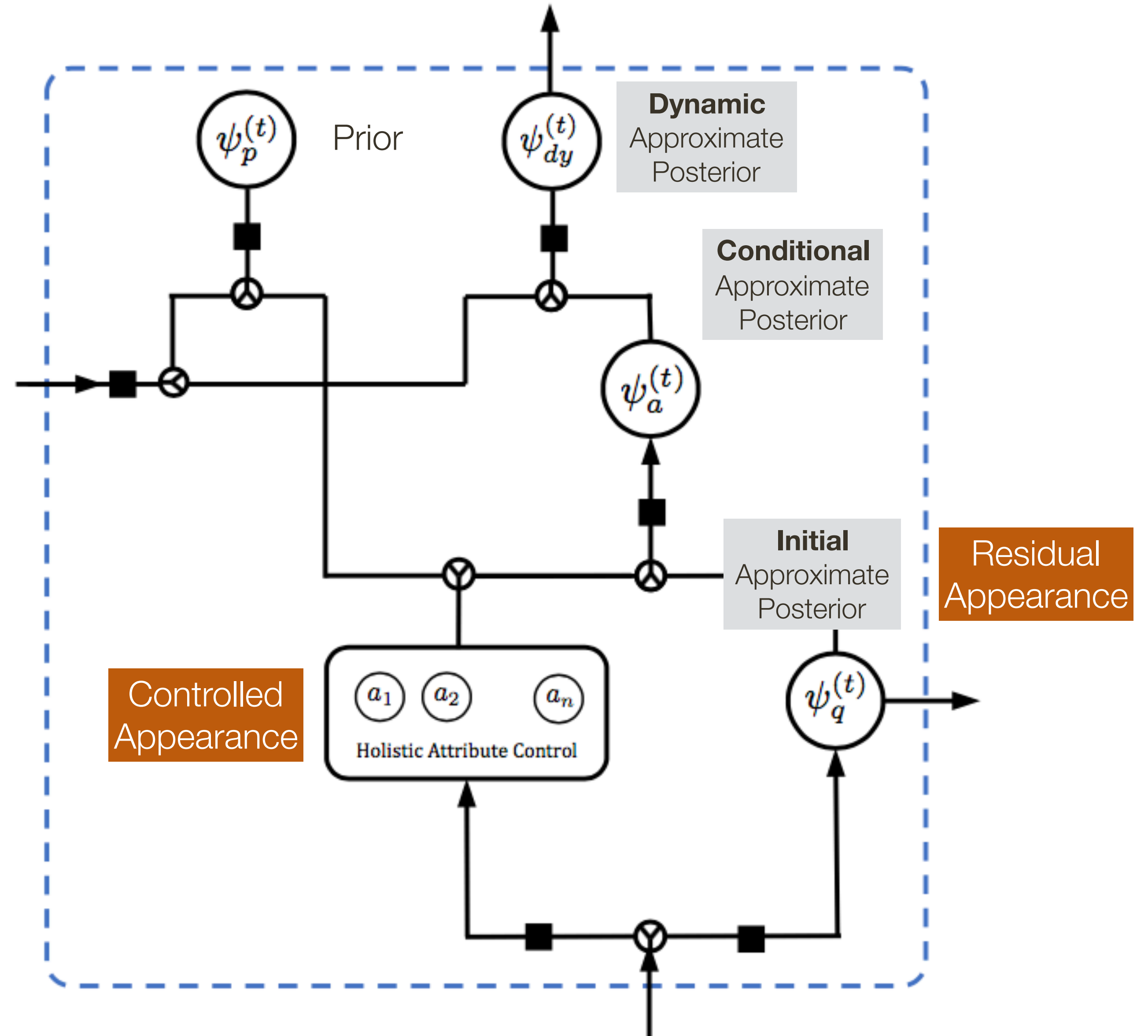
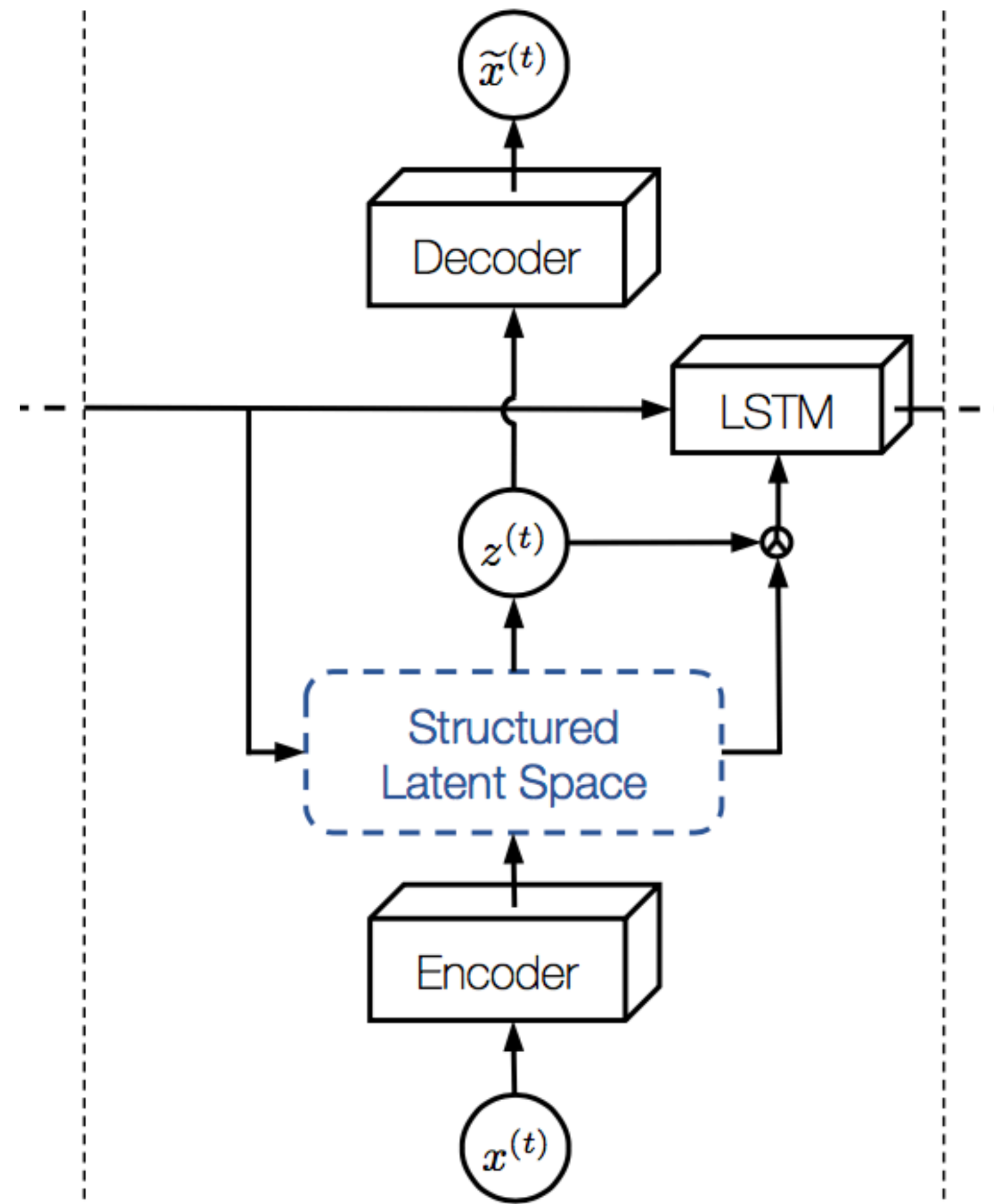
Variational Autoencoder (VAE) + LSTM

[He et al., 2018]



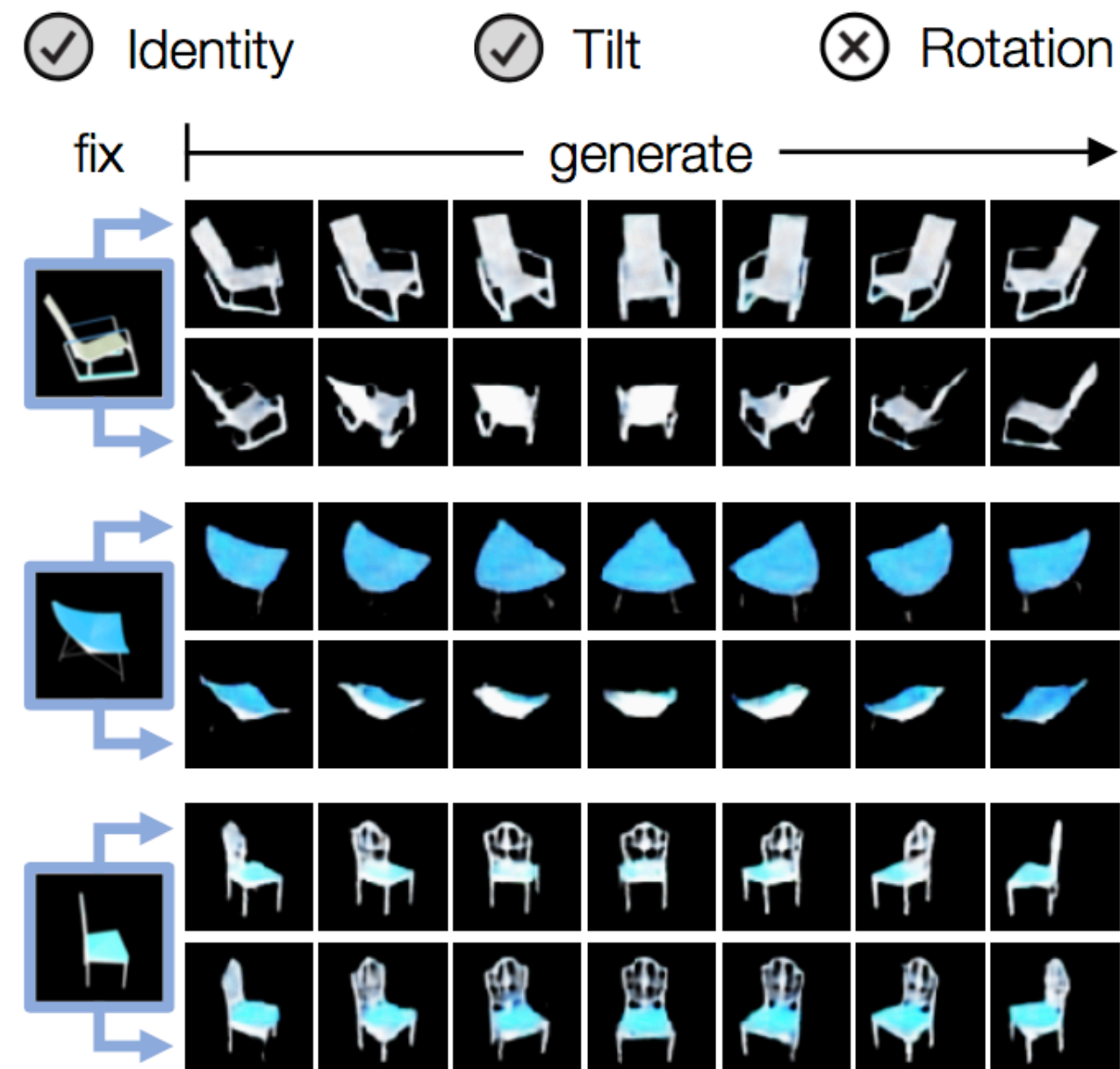
VAE + LSTM with Structured Latent Space

[He et al., 2018]

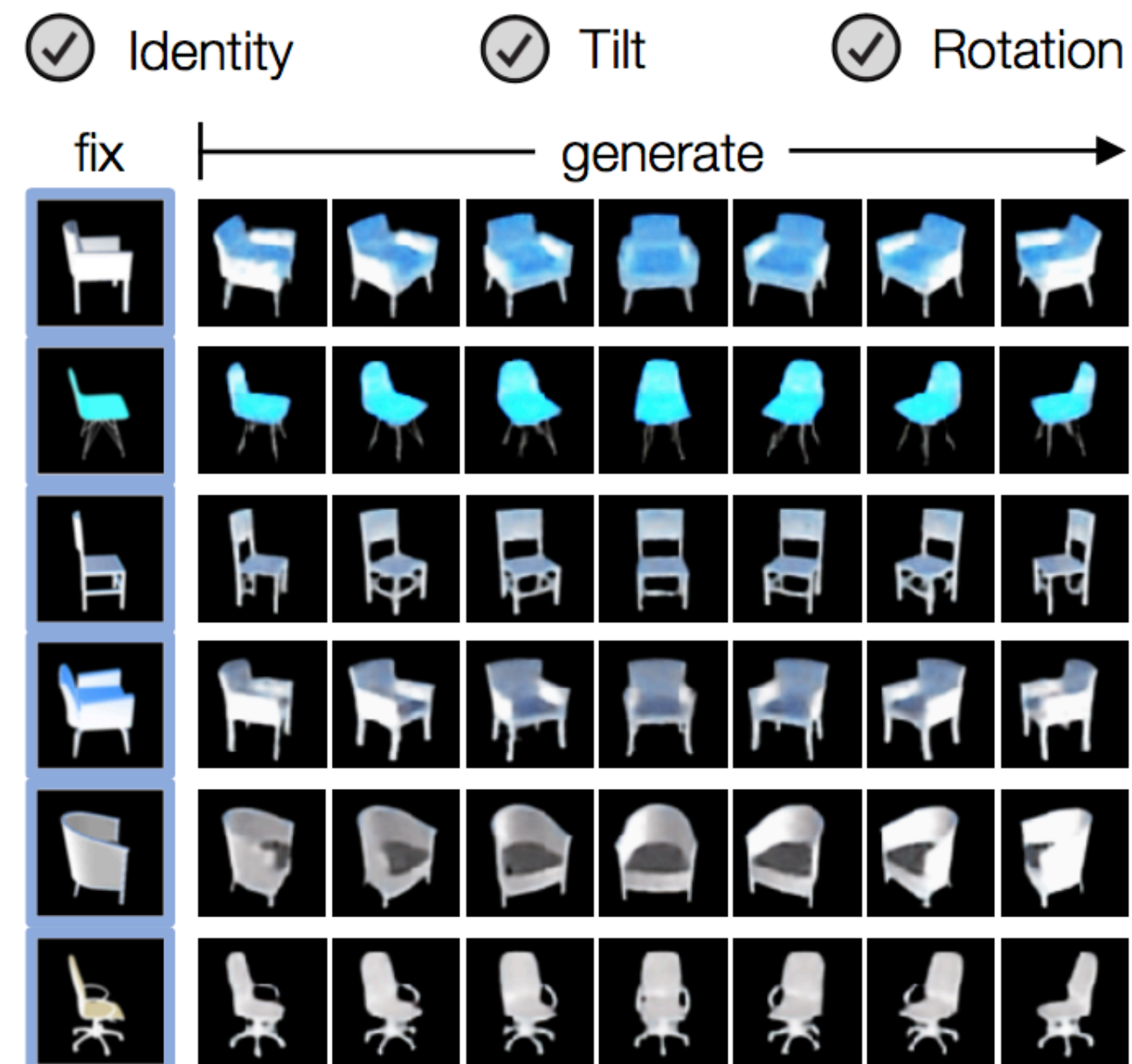


Results: Chair CAD dataset

[He et al., 2018]



(a) Partial control.



(b) Full control.

Ablation

	Bound	Static	$-C$		$+C$	
			$-S$	$+S$	$-S$	$+S$
Intra-E ↓	1.98	40.33	17.64	7.79	14.81	5.50
Inter-E ↑	1.39	0.42	0.73	1.35	1.02	1.37
I-Score ↑	4.01	1.28	1.83	3.63	2.56	3.94

Quantitative

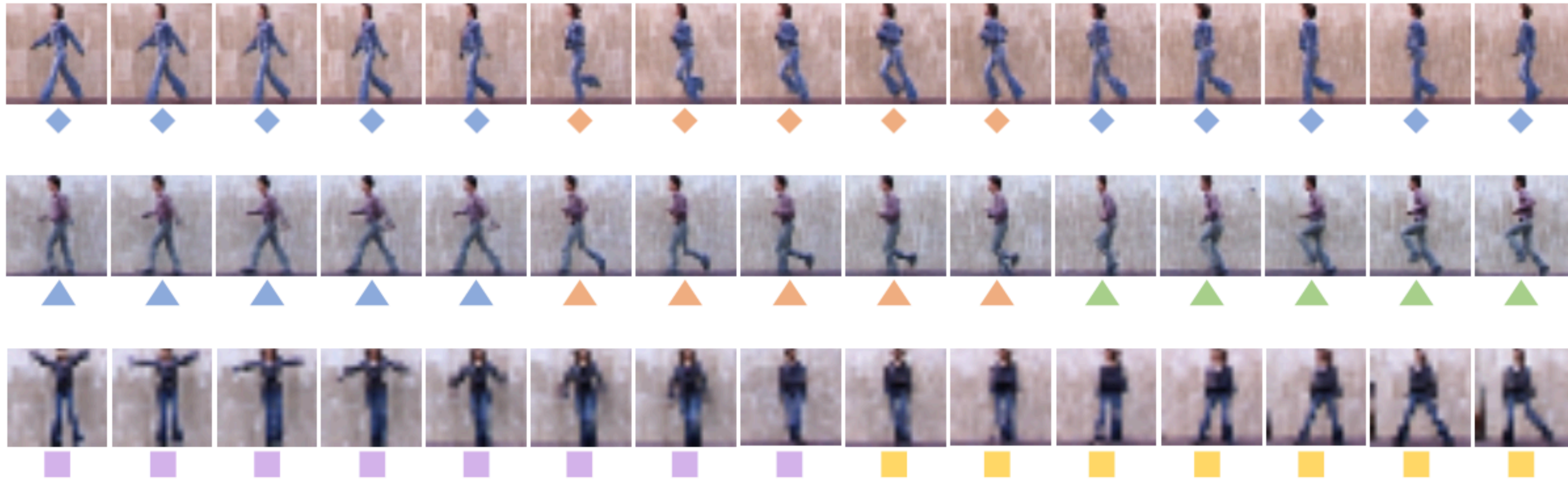
Chair CAD [1, 40]			
	Bound	Deep Rot. [40]	VideoVAE (ours)
		●	●
Intra-E ↓	1.98	14.68	5.50
Inter-E ↑	1.39	1.34	1.37
I-Score ↑	4.01	3.39	3.94

Results: Weizmann Human Action dataset

[He et al., 2018]

⊙ Identity = \blacklozenge | \blacktriangle | \blacksquare ⊙ Action = \bullet walking | \bullet running | \bullet skipping | \bullet jumping jack | \bullet side step

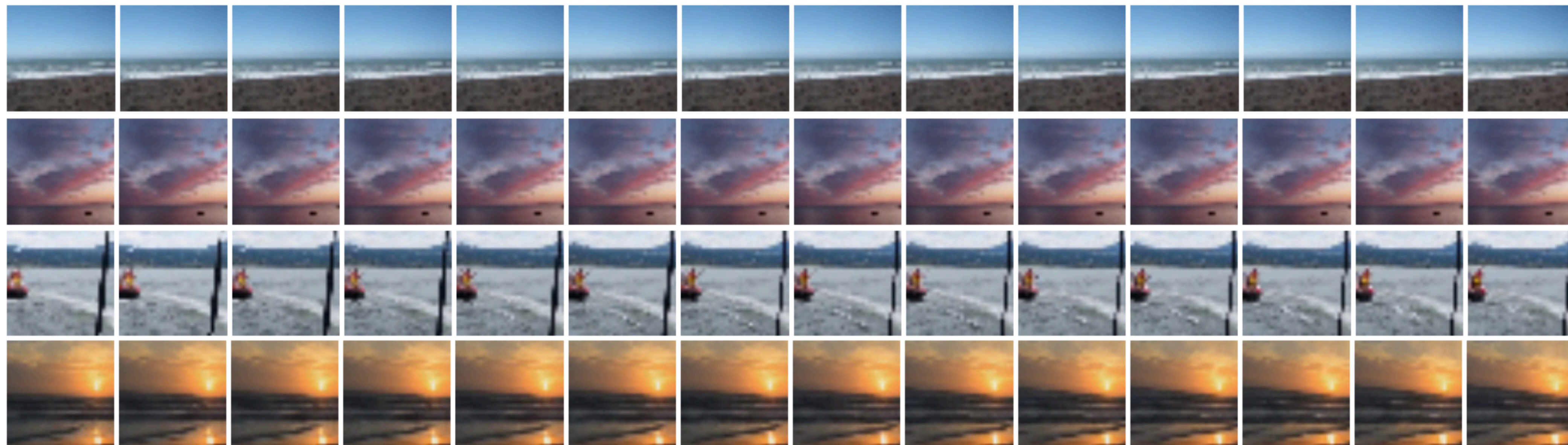
generate \longrightarrow



Weizmann Human Action [2]				
	Bound	MoCoGAN [32]	VideoVAE (ours)	
		○	○	●
Intra-E	↓ 0.63	23.58	9.53	9.44
Inter-E	↑ 4.49	2.91	4.37	4.37
I-Score	↑ 89.12	13.87	69.55	70.10

Results: MIT Flickr

[He et al., 2018]



YFCC [31] — MIT Flickr [34]				
	Bound	VGAN [34]	VideoVAE (ours)	
		○	○	●
Intra-E	↓ 30.34	46.96	44.03	38.20
Inter-E	↑ 0.693	0.692	0.691	0.692
I-Score	↑ 1.87	1.58	1.62	1.81

Variational Autoencoders

Probabilistic spin to traditional autoencoders => allows generating data

Defines an intractable density => derive and optimize a (variational) lower bound

Pros:

- Principled approach to generative models
- Allows inference of $q(z|x)$, can be useful feature representation for other tasks

Cons:

- Maximizes lower bound of likelihood: okay, but not as good evaluation as PixelRNN/PixelCNN
- Samples blurrier and lower quality compared to state-of-the-art (GANs)

Active area of research:

- More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian
- Incorporating structure in latent variables (our submission to CVPR)

So far ...

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

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

VAEs define intractable density function with latent variables z (that we need to marginalize):

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

cannot optimize directly, derive and optimize lower bound of likelihood instead

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

So far ...

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

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

VAEs define intractable density function with latent variables z (that we need to marginalize):

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

cannot optimize directly, derive and optimize lower bound of likelihood instead

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

GANs: don't work with any explicit density function

Generative Adversarial Networks (GANs)

Generative Adversarial Networks

[Goodfellow et al., 2014]

Problem: Want to sample from complex, high-dimensional training distribution. There is no direct way to do this!

Generative Adversarial Networks

[Goodfellow et al., 2014]

Problem: Want to sample from complex, high-dimensional training distribution. There is no direct way to do this!

Solution: Sample from a simple distributions, e.g., random noise. Learn transformation to the training distribution

Generative Adversarial Networks

[Goodfellow et al., 2014]

Problem: Want to sample from complex, high-dimensional training distribution. There is no direct way to do this!

Solution: Sample from a simple distributions, e.g., random noise. Learn transformation to the training distribution

Question: What can we use to represent complex transformation function?

Generative Adversarial Networks

[Goodfellow et al., 2014]

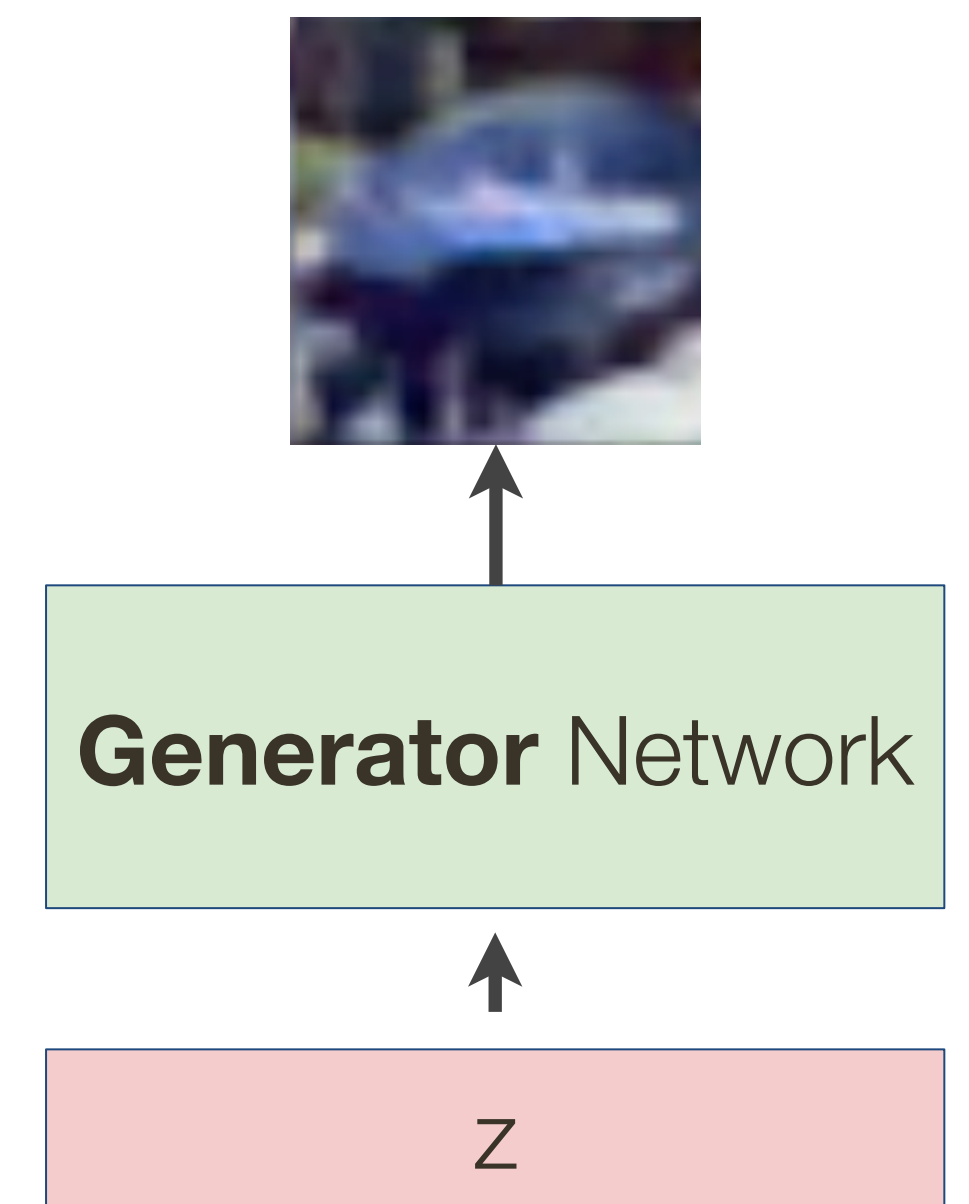
Problem: Want to sample from complex, high-dimensional training distribution. There is no direct way to do this!

Solution: Sample from a simple distributions, e.g., random noise. Learn transformation to the training distribution

Question: What can we use to represent complex transformation function?

Output: Sample from training distribution

Input: Random noise



Training GANs: Two-player Game

[Goodfellow et al., 2014]

Generator network: try to fool the discriminator by generating real-looking images

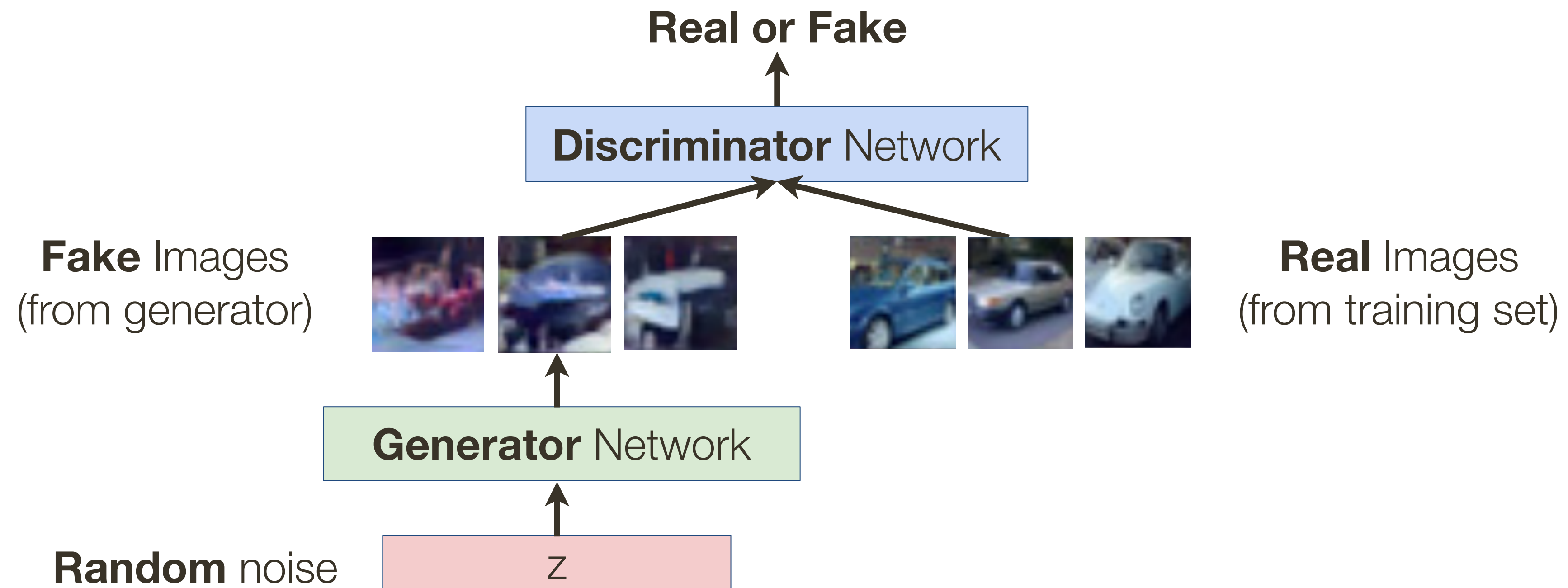
Discriminator network: try to distinguish between real and fake images

Training GANs: Two-player Game

[Goodfellow et al., 2014]

Generator network: try to fool the discriminator by generating real-looking images

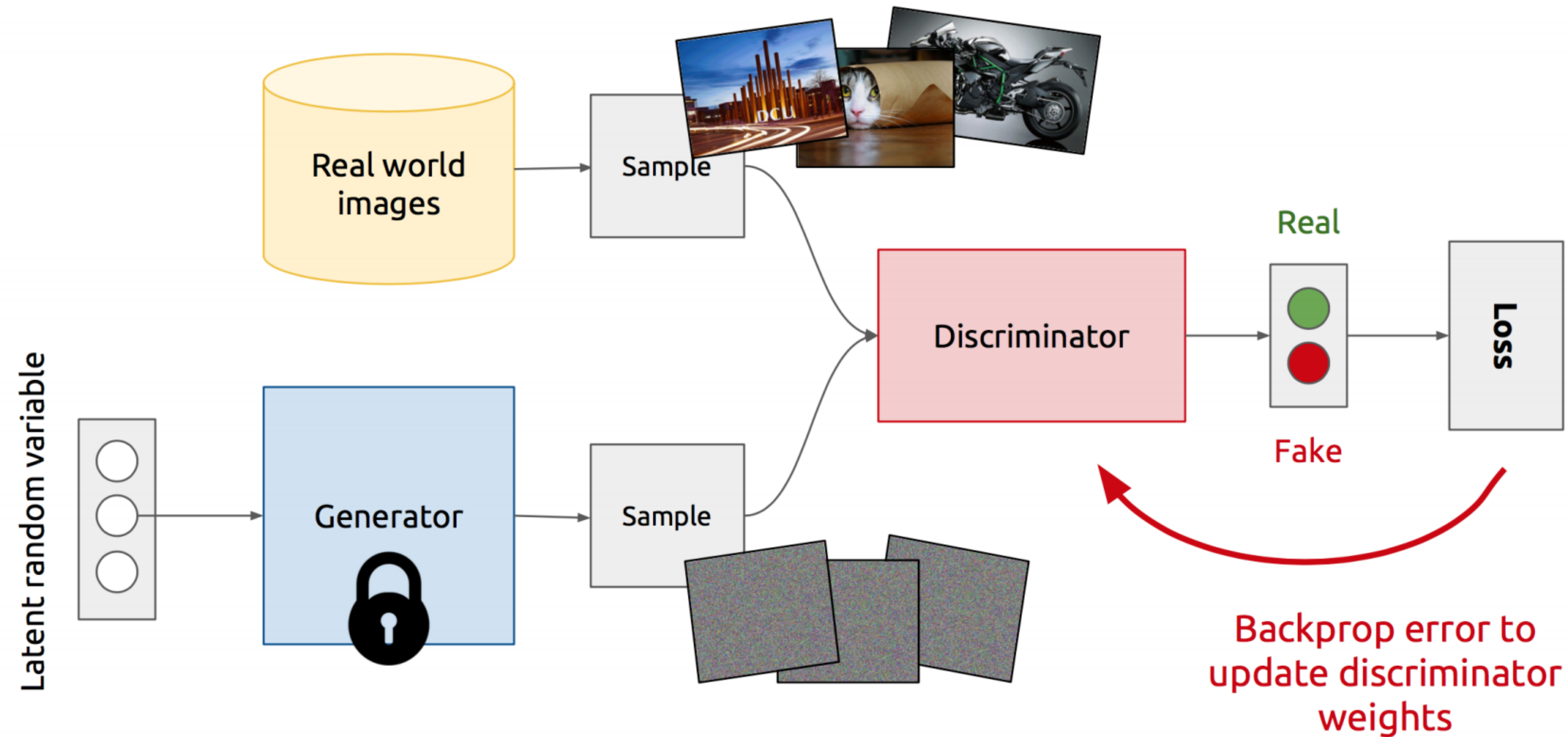
Discriminator network: try to distinguish between real and fake images



Training GANs: Two-player Game

Generator network: try to fool the discriminator by generating real-looking images

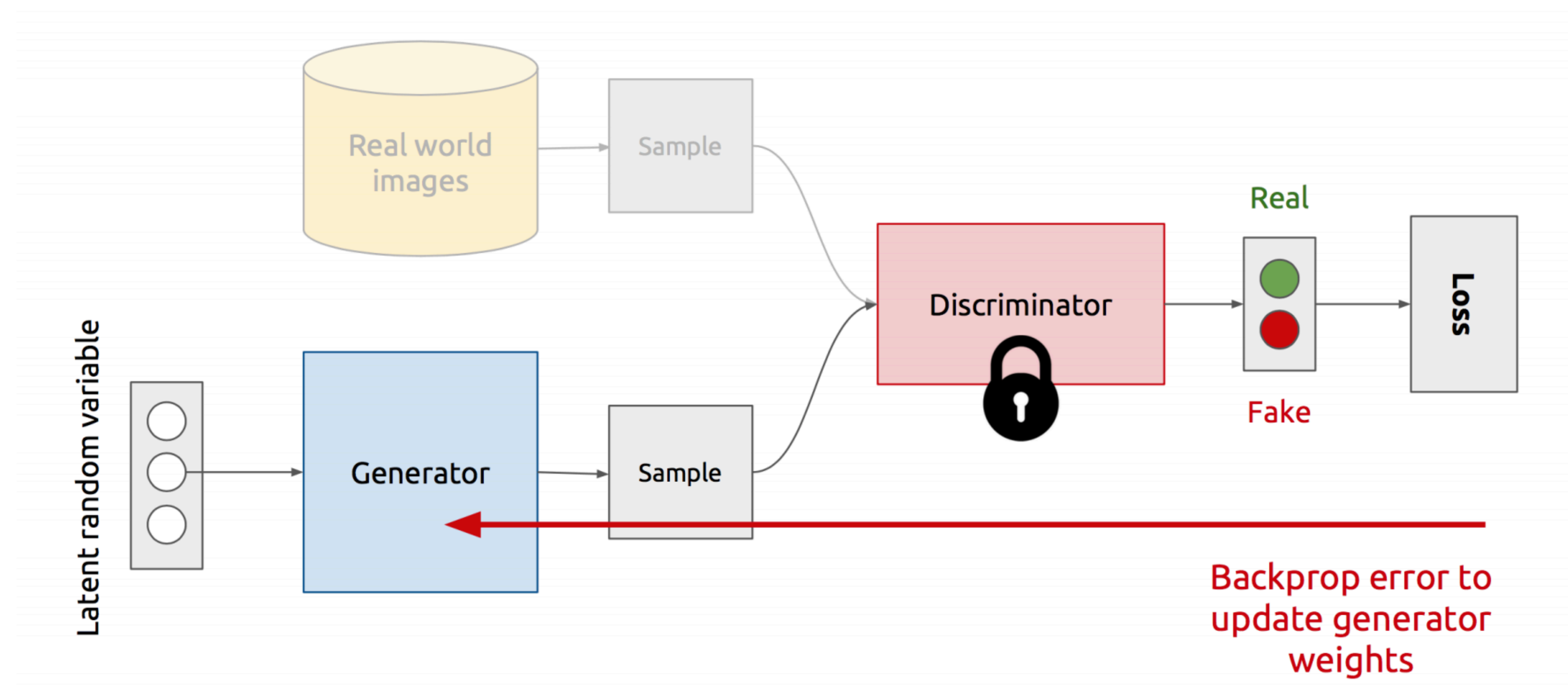
Discriminator network: try to distinguish between real and fake images



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



Training GANs: Two-player Game

[Goodfellow et al., 2014]

Generator network: try to fool the discriminator by generating real-looking images

Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Minimax objective function:

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

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

Discriminator output
for real data x

Discriminator output for
generated fake data G(z)

- **Discriminator** (θ_d) wants to maximize objective such that $D(x)$ is close to 1 (real) and $D(G(z))$ is close to 0 (fake)
- **Generator** (θ_g) wants to minimize objective such that $D(G(z))$ is close to 1 (discriminator is fooled into thinking generated $G(z)$ is real)

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

Train jointly in **minimax game**

Minimax objective function:

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

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

Discriminator output
for real data x

Discriminator output for
generated fake data $G(z)$

The **Nash equilibrium** of this particular game is achieved when:

$$p_{data}(x) = p_{gen}(G_{\theta_g}(z)), \quad \forall x$$

$$D_{\theta_d}(x) = 0.5, \quad \forall x$$

Training GANs: Two-player Game

[Goodfellow et al., 2014]

Minimax objective function:

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

Alternate between:

1. Gradient **ascent** on discriminator

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

2. Gradient **descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Training GANs: Two-player Game

[Goodfellow et al., 2014]

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k , is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Sample minibatch of m examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
- Update the discriminator by ascending its stochastic gradient:

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

end for

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

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

end for

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

**Discriminator
updates**

**Generator
updates**

Training GANs: Two-player Game

[Goodfellow et al., 2014]

Minimax objective function:

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

Alternate between:

1. Gradient **ascent** on discriminator

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

2. Gradient **descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

Training GANs: Two-player Game

[Goodfellow et al., 2014]

Minimax objective function:

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

Alternate between:

1. Gradient **ascent** on discriminator

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

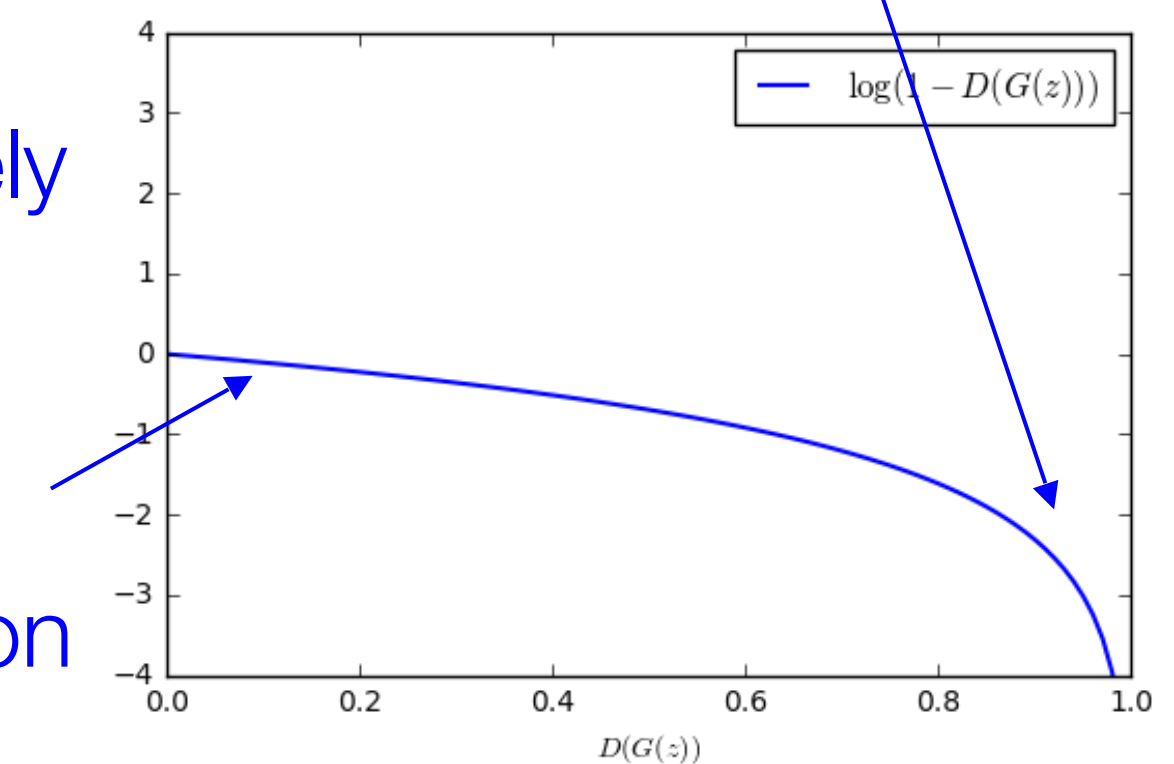
2. Gradient **descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

Gradient signal dominated by region where sample is already good

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



Training GANs: Two-player Game

[Goodfellow et al., 2014]

Minimax objective function:

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

Alternate between:

1. Gradient **ascent** on discriminator

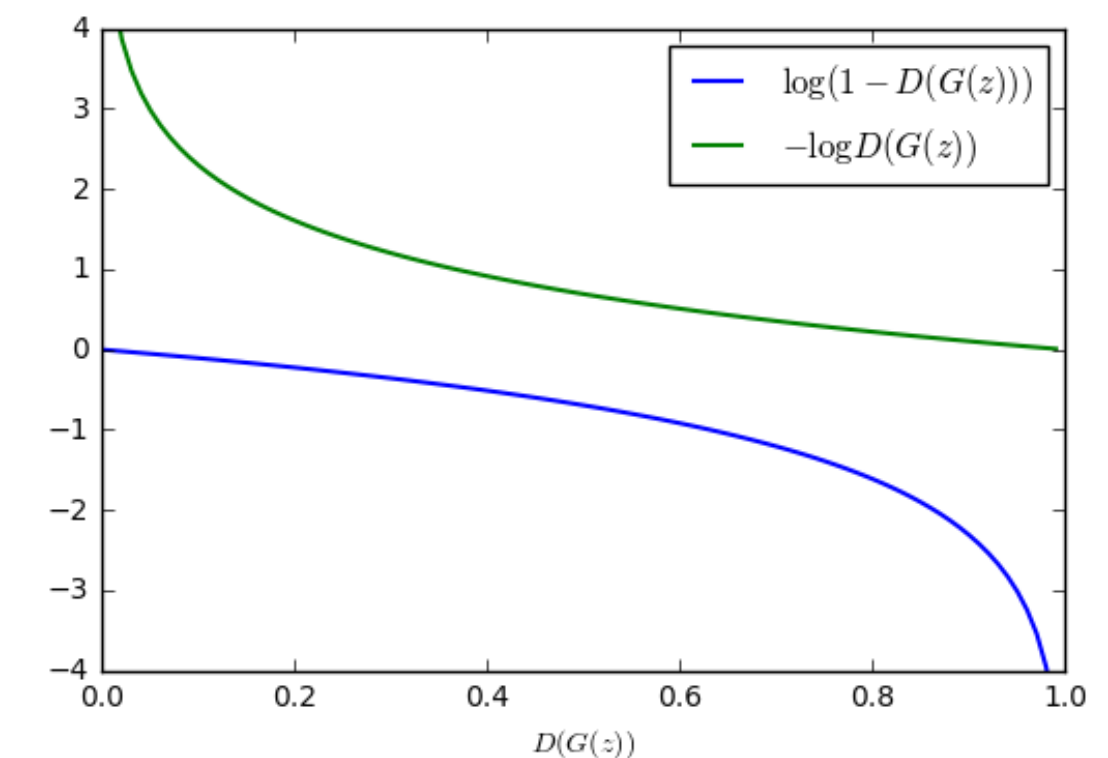
$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. Instead, gradient **ascent** on generator, different objective

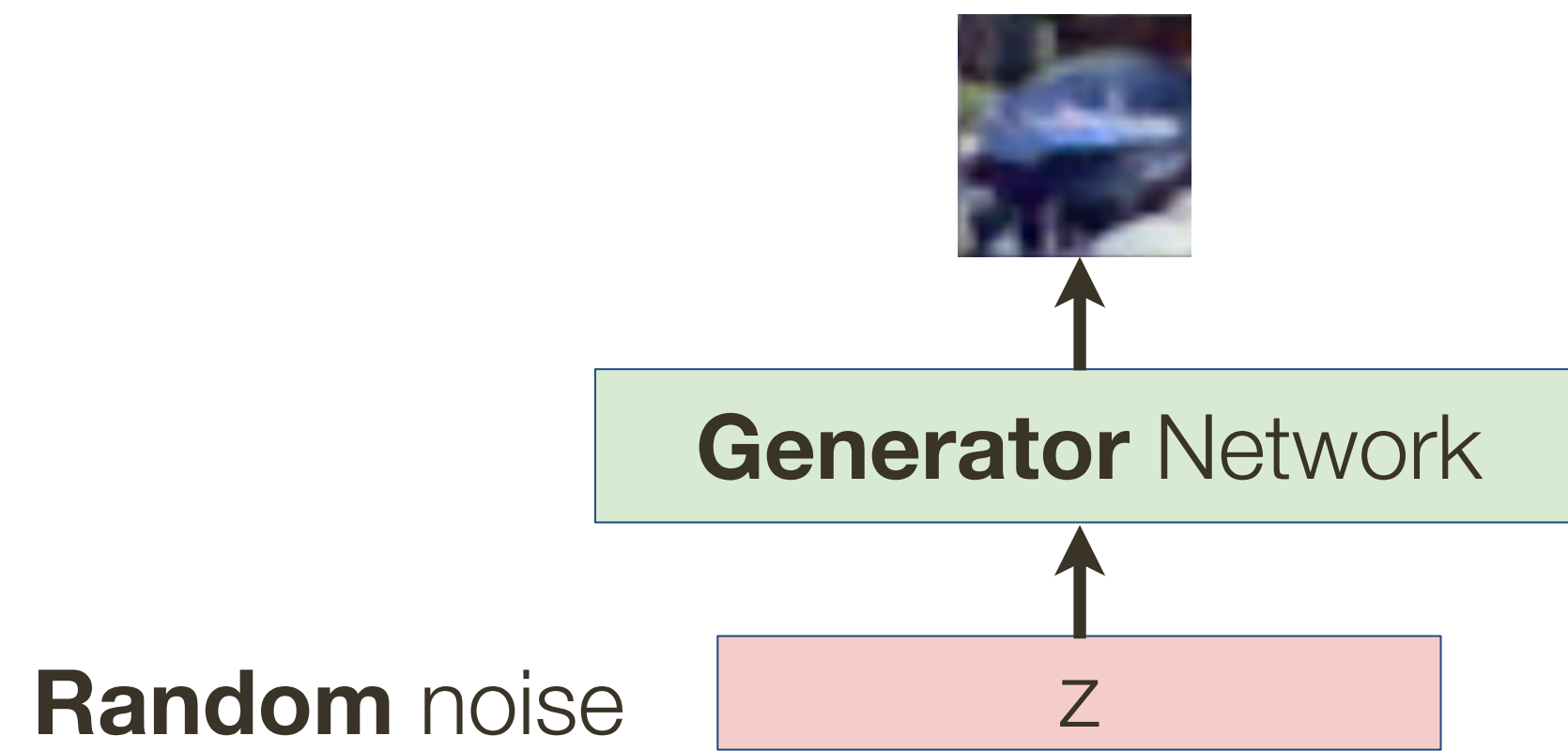
$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.

Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.

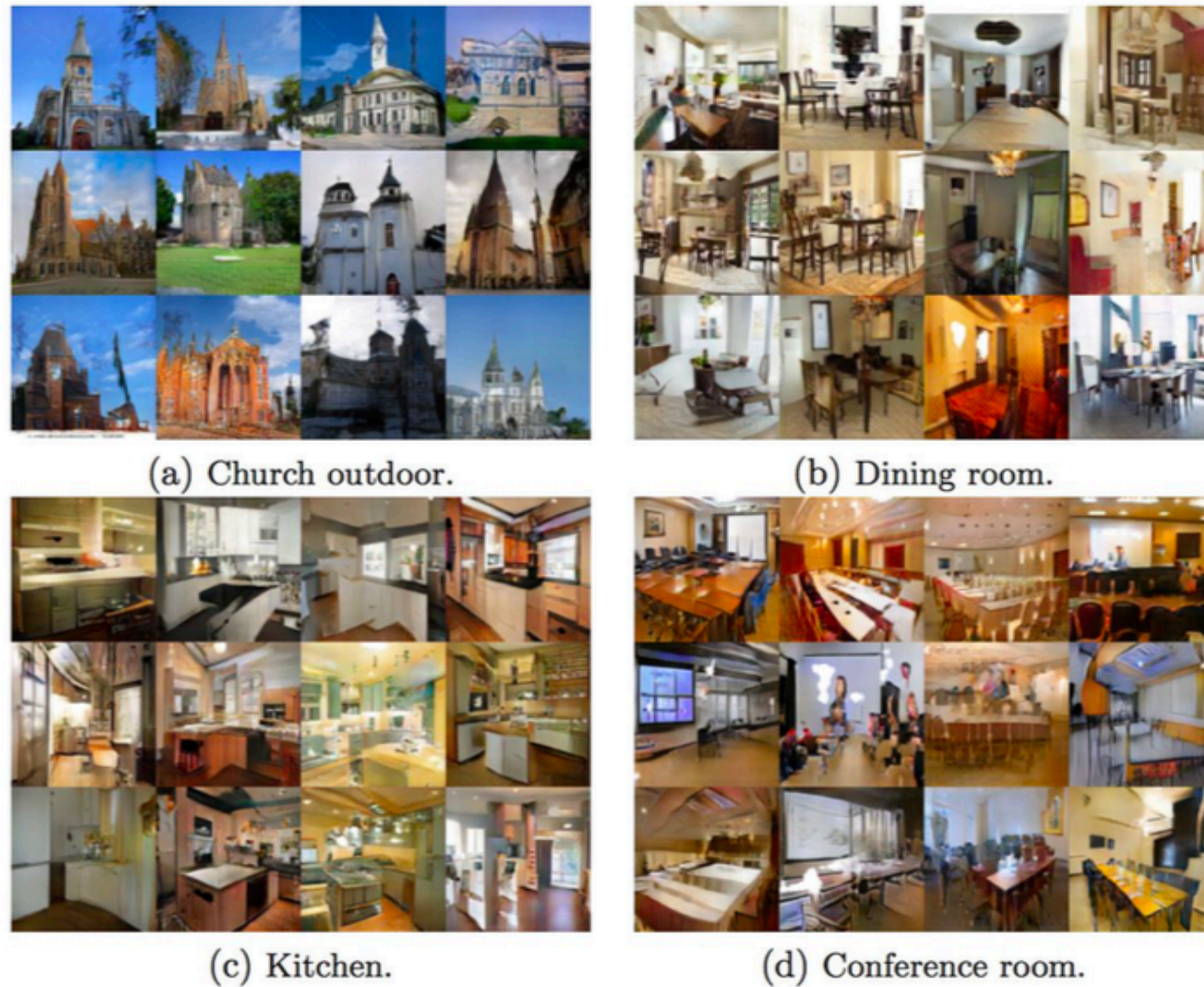


Sampling **GANs**

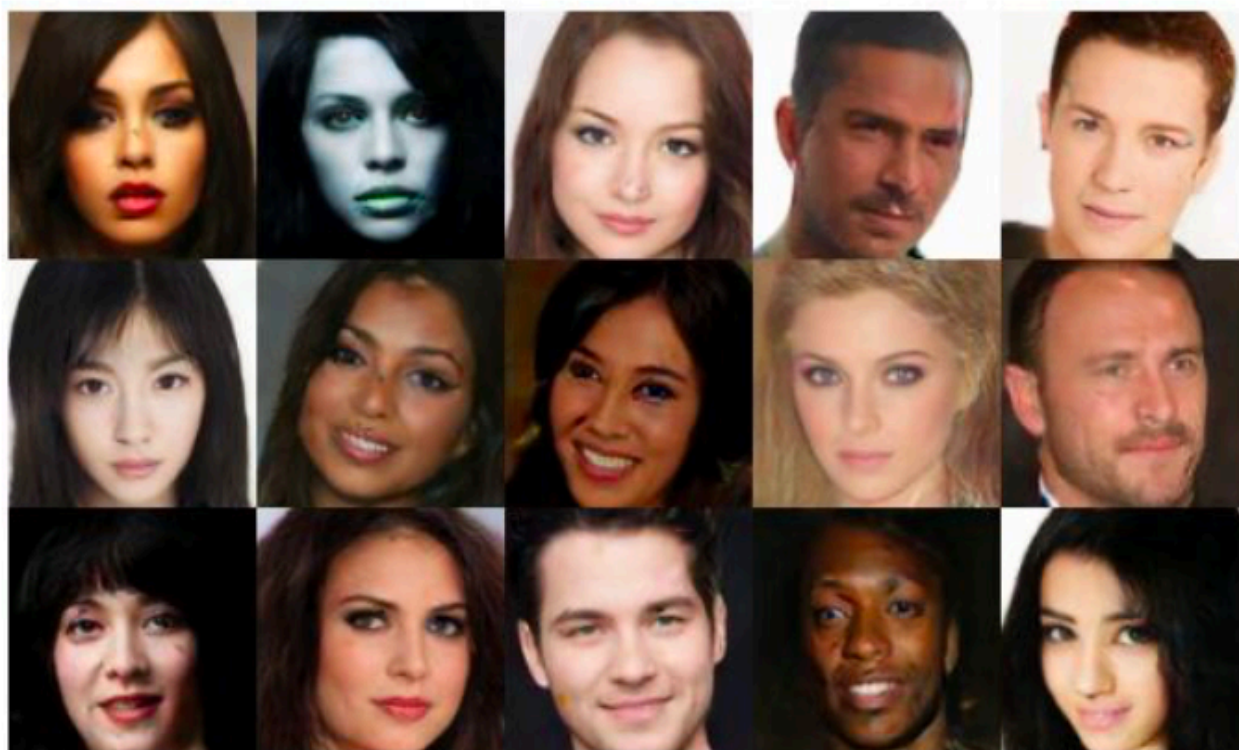


Year of the GAN

Better training and generation



LSGAN. Mao et al. 2017.



BEGAN. Bertholet et al. 2017.

Source->Target domain transfer



CycleGAN. Zhu et al. 2017.

Text -> Image Synthesis

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.



Reed et al. 2017.

Many GAN applications



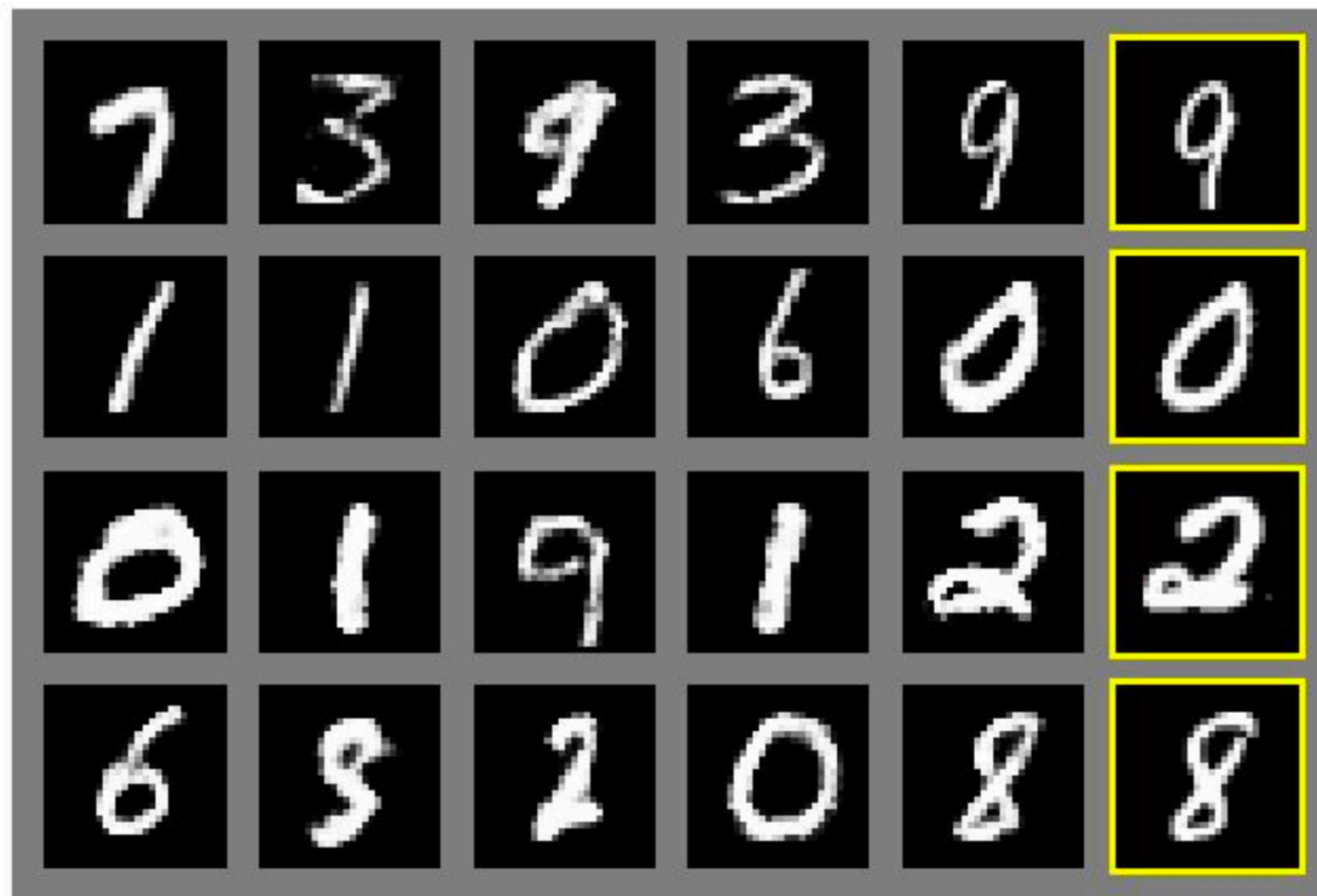
Pix2pix. Isola 2017. Many examples at <https://phillipi.github.io/pix2pix/>

Year of the GAN

- 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 Categorical 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

Generative Adversarial Nets

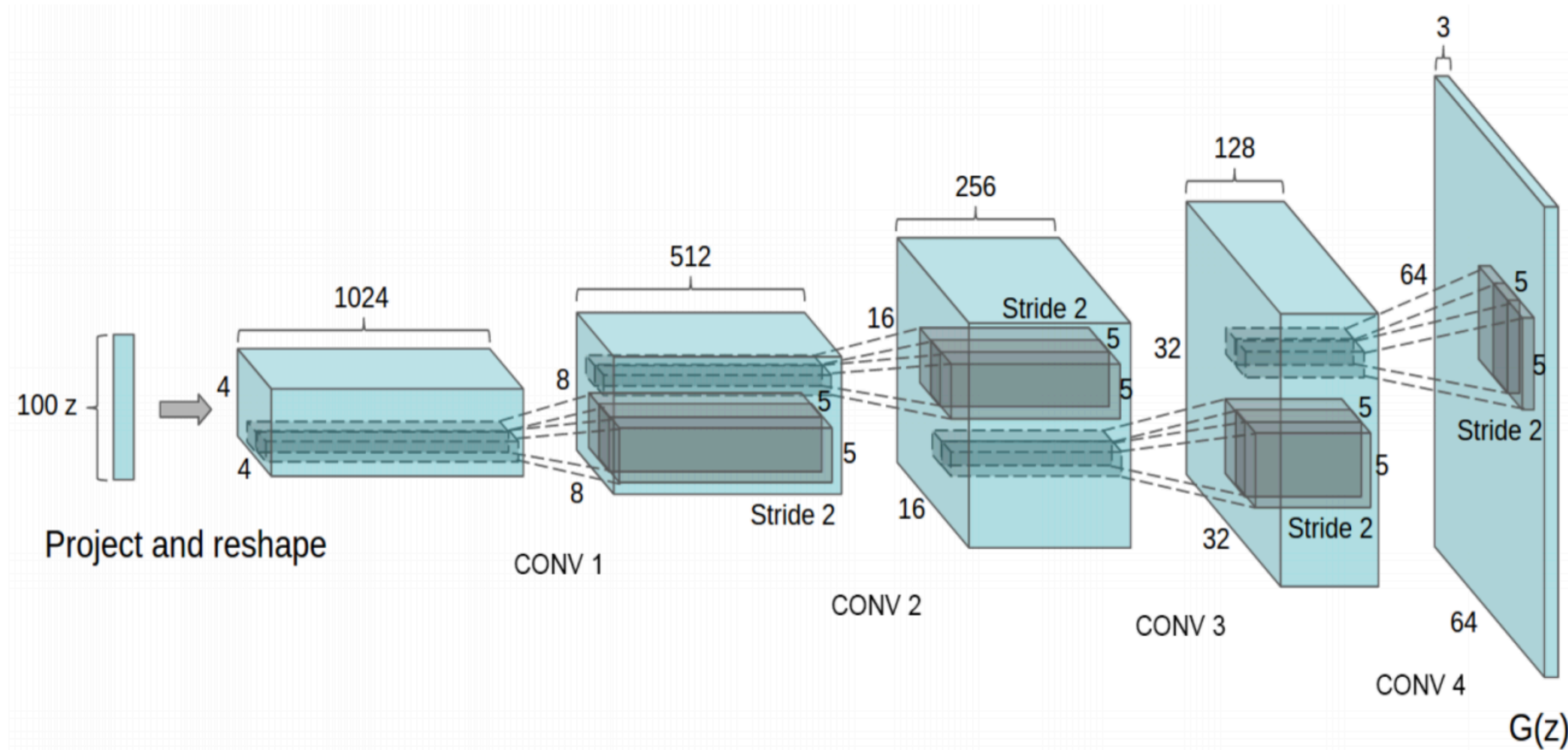
Generated Samples



Deep Convolutional GANs (DCGANs)

[Radford et al., 2016]

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

GANs with Convolutional Architectures

[Radford et al., 2016]



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GANs with Convolutional Architectures

[Radford et al., 2016]

Interpolating between points in latent space



GANs: Interpretable Vector Math

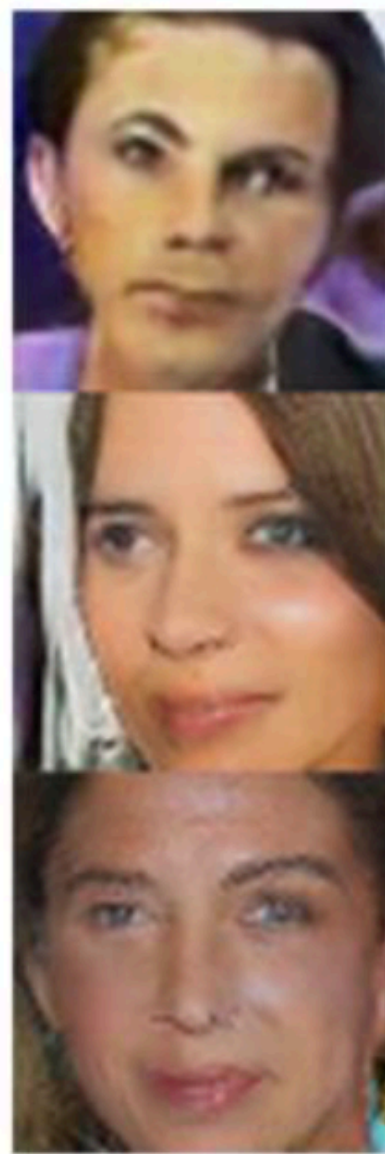
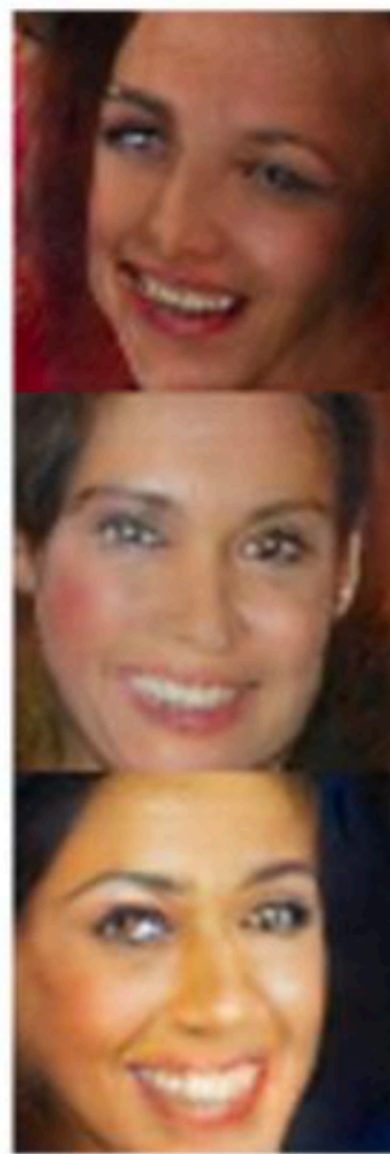
[Radford et al., 2016]

Smiling woman

Neutral woman

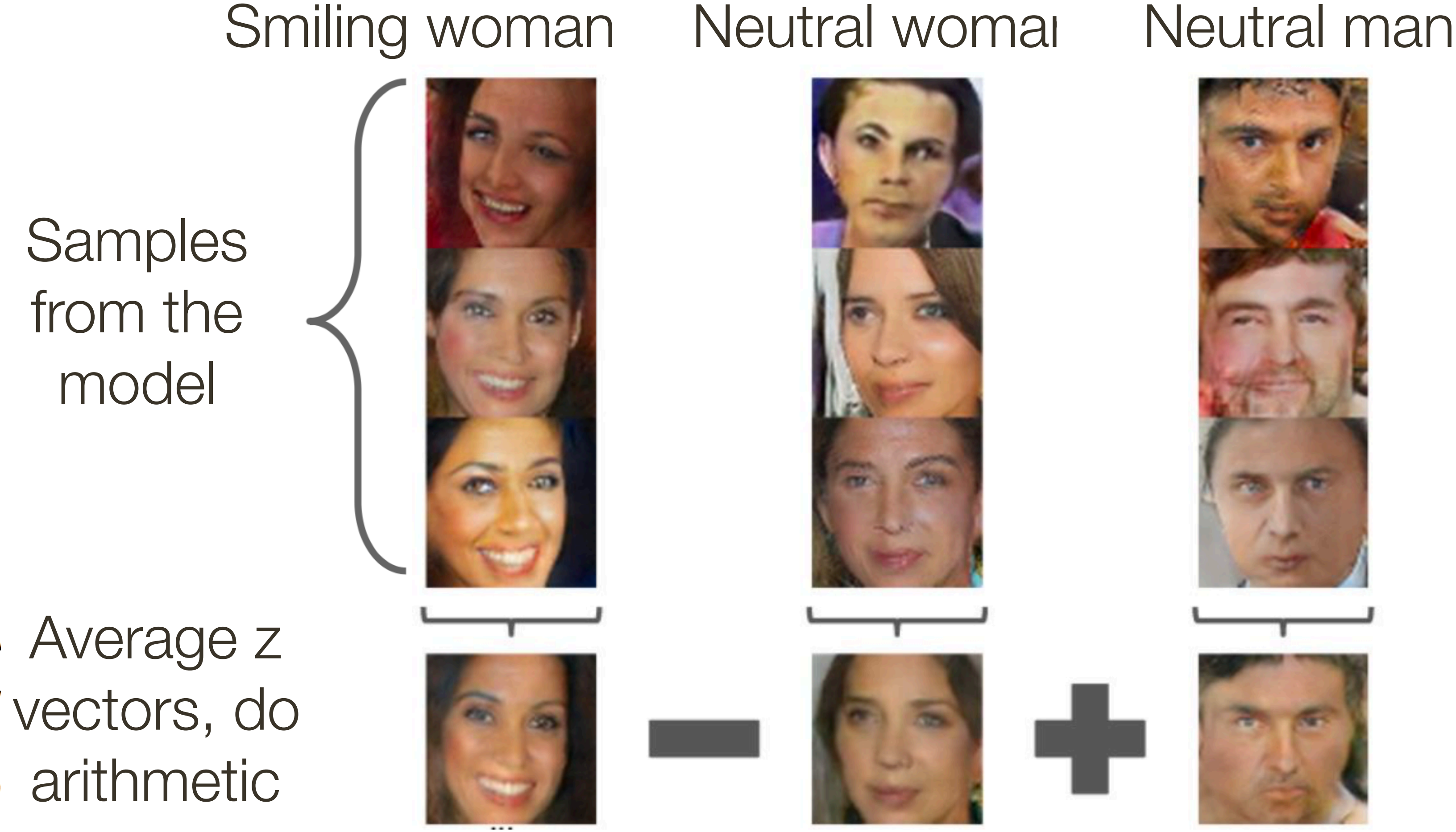
Neutral man

Samples
from the
model



GANs: Interpretable Vector Math

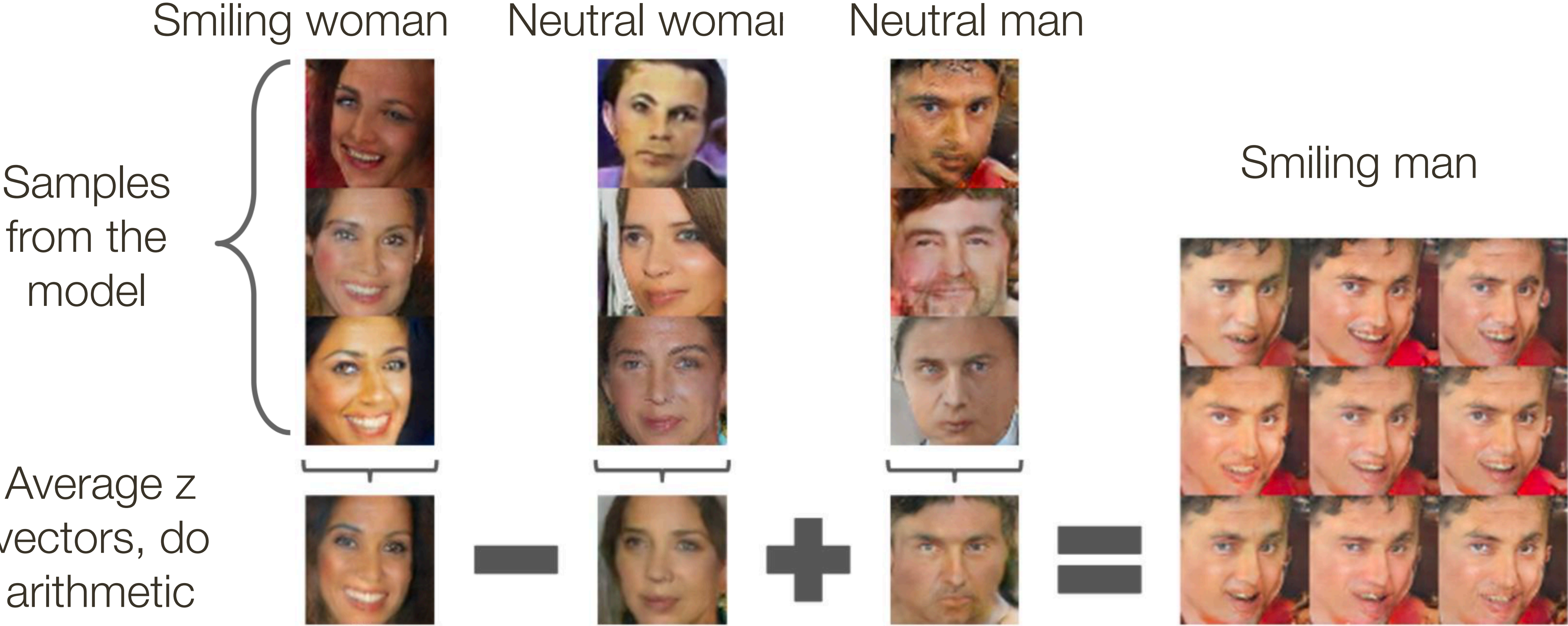
[Radford et al., 2016]



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GANs: Interpretable Vector Math

[Radford et al., 2016]



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GANs: Interpretable Vector Math

[Radford et al., 2016]

Glasses Man

No Glasses Man

No Glasses Woman

Samples
from the
model

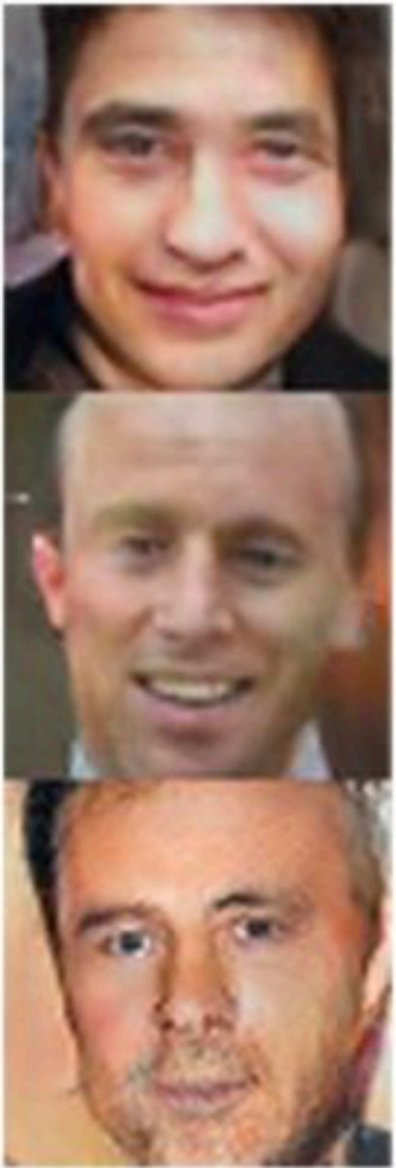


GANs: Interpretable Vector Math

[Radford et al., 2016]

Glasses Man No Glasses Man No Glasses Woman

Samples from the model



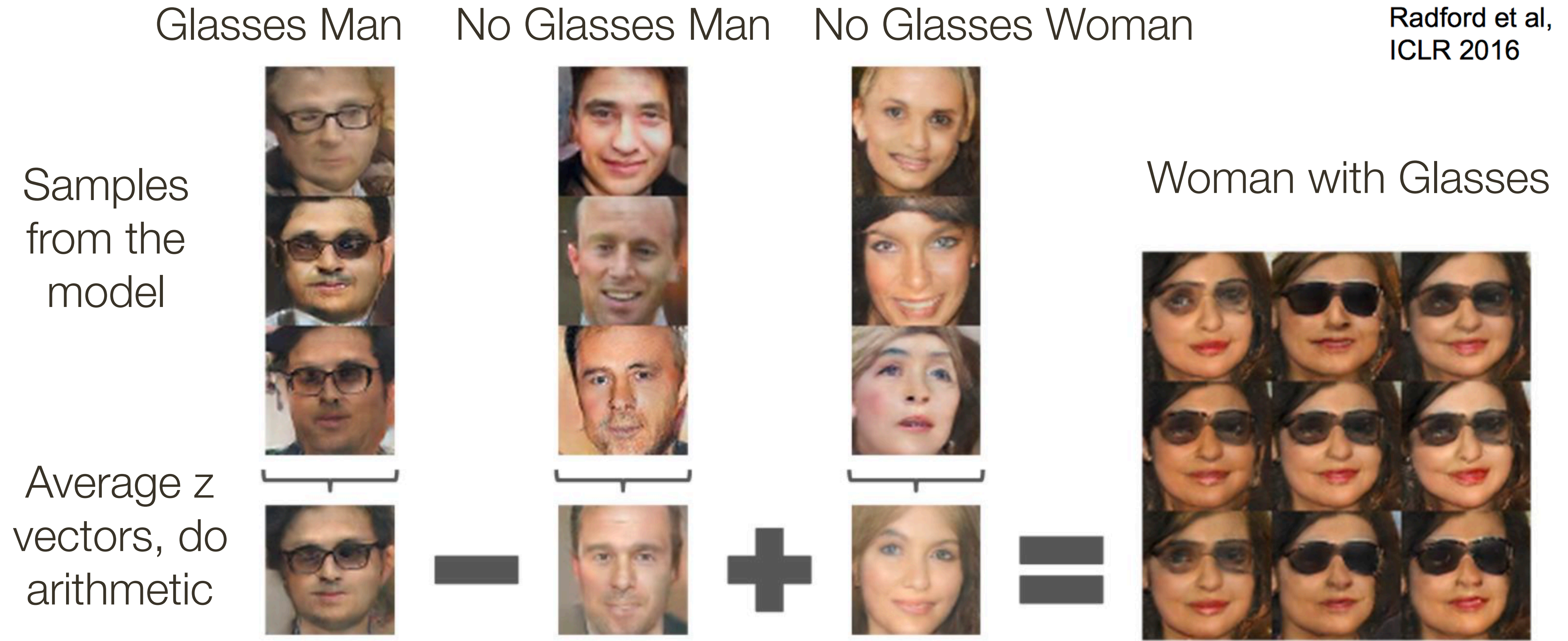
Average z vectors, do arithmetic



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GANs: Interpretable Vector Math

[Radford et al., 2016]



Conditional GAN: Text-to-Image Synthesis

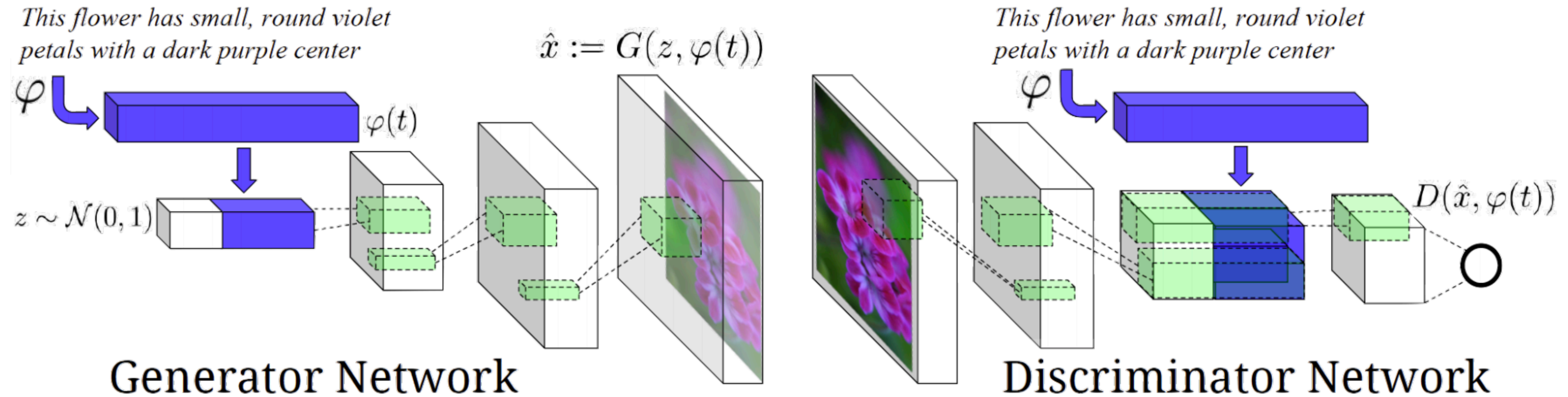


Figure 2 in the original paper.

Positive Example:
Real Image, Right Text

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