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

Lecture 17: Generative Models [part 3]

Logistics

Project Proposal Document due **Today, 11:59pm**

Assignment 5 due **December 6 (last day of classes)**

Logistics

Research Paper Presentations:

— List of 35 papers and Quiz published

— Quiz is due **tomorrow, 11:59pm**

— Paper assignments by Thursday

— Presentations by Friday, **November 25th**

Research Paper Readings:

— We will not have many of these. I expect 4 papers ~ 3.5 weeks

— First paper reading will be for next class (in prep for Diffusion Models)

Variational Autoencoders (VAE)

So far …

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

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

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

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

Variational Autoencoder: Inference

Input Data x

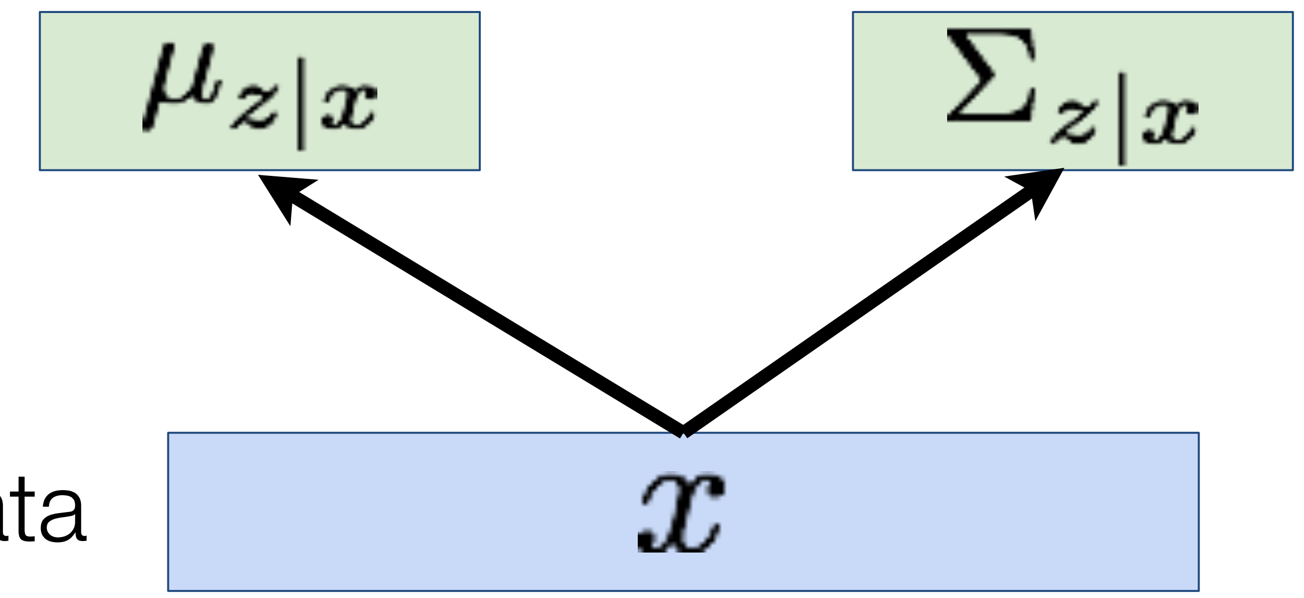
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder: Inference

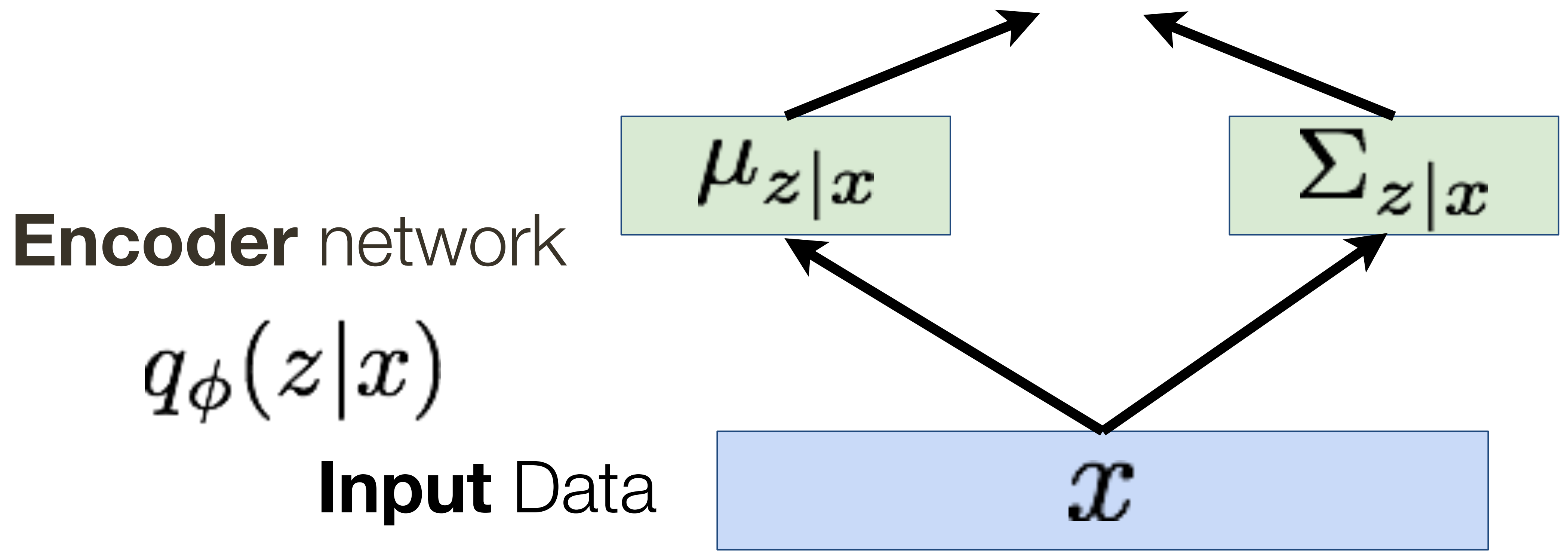
Encoder network

$q_\phi(z|x)$

Input Data

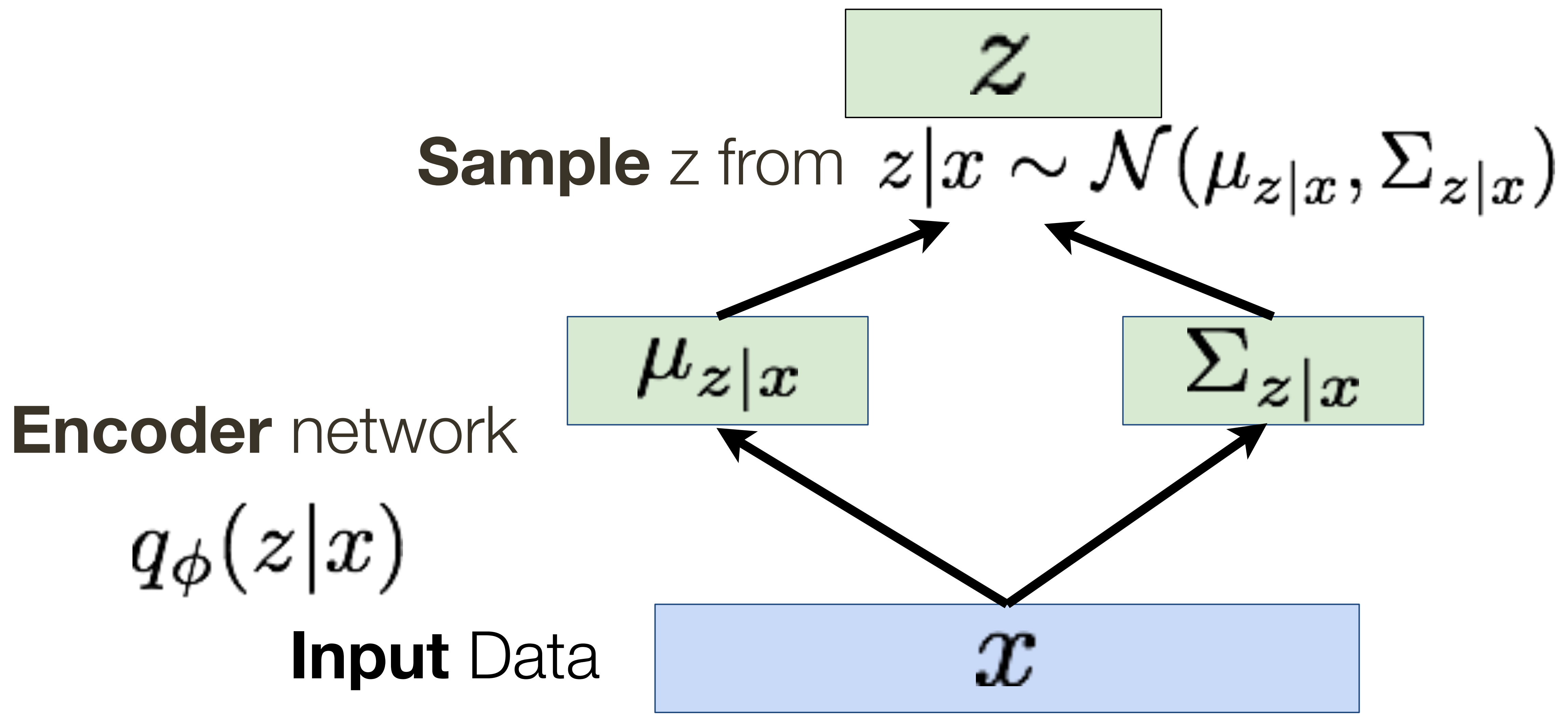


Variational Autoencoder: Inference

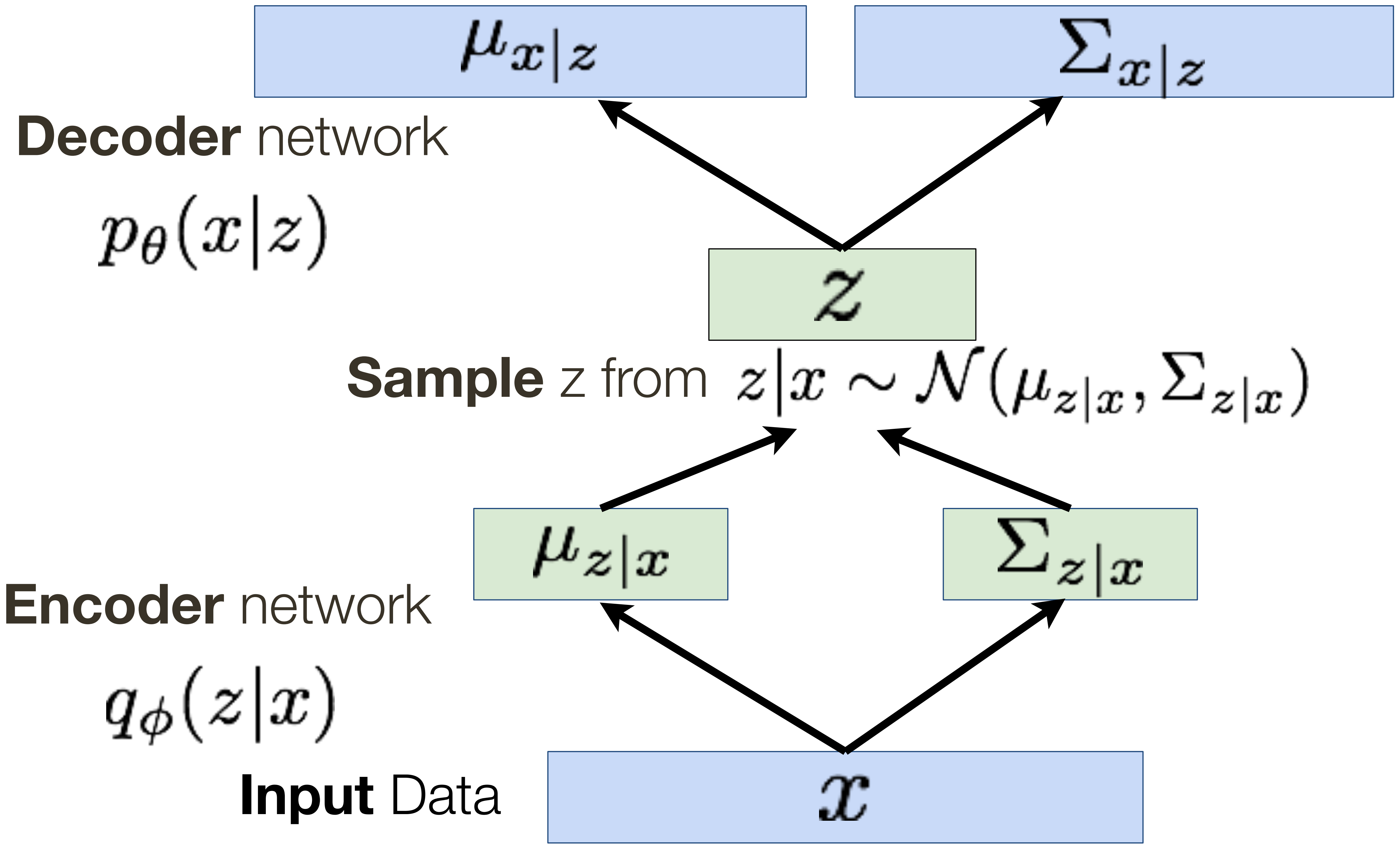


* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder: Inference

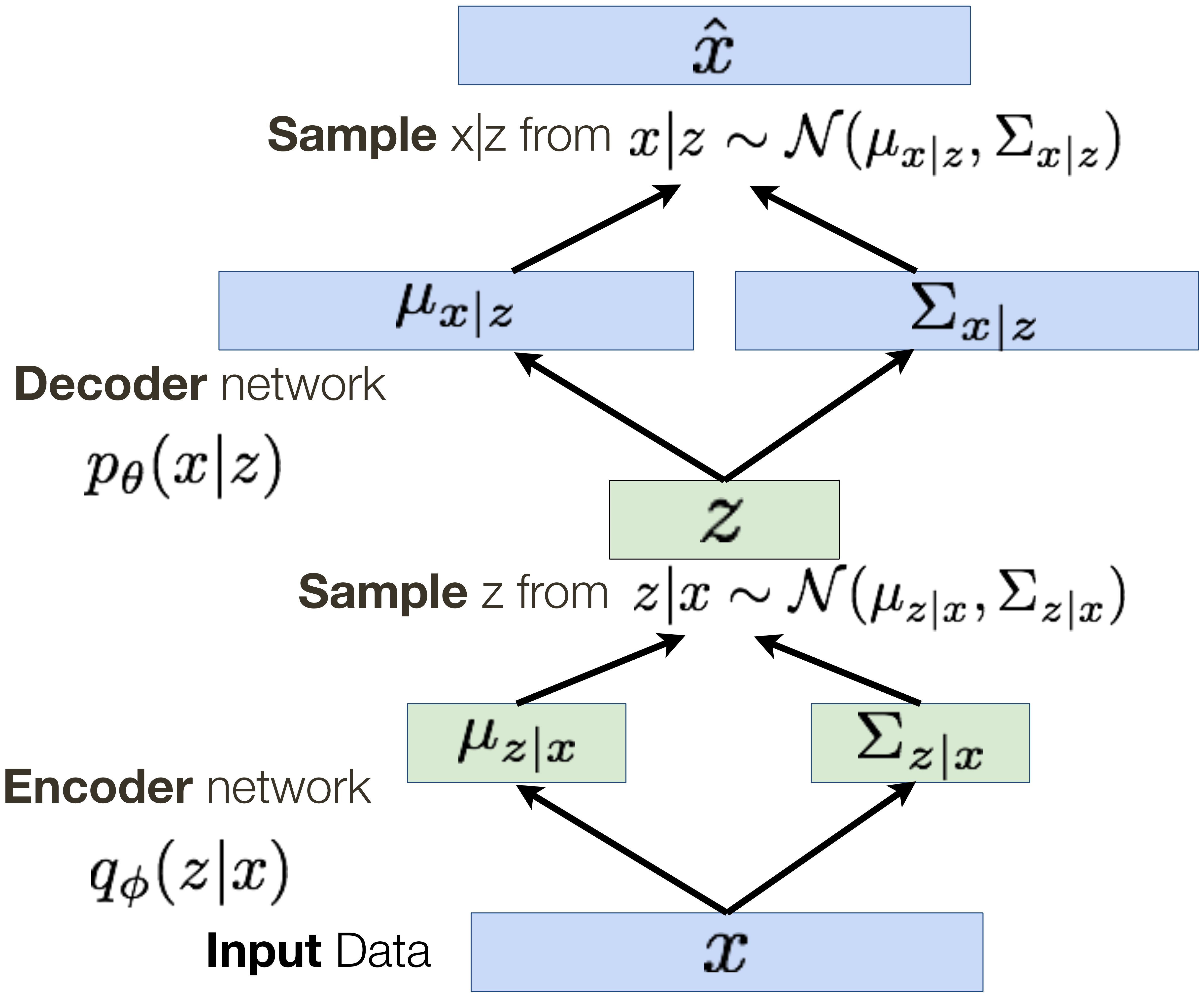


Variational Autoencoder: Inference



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

Variational Autoencoder: Inference



Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower
bound

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

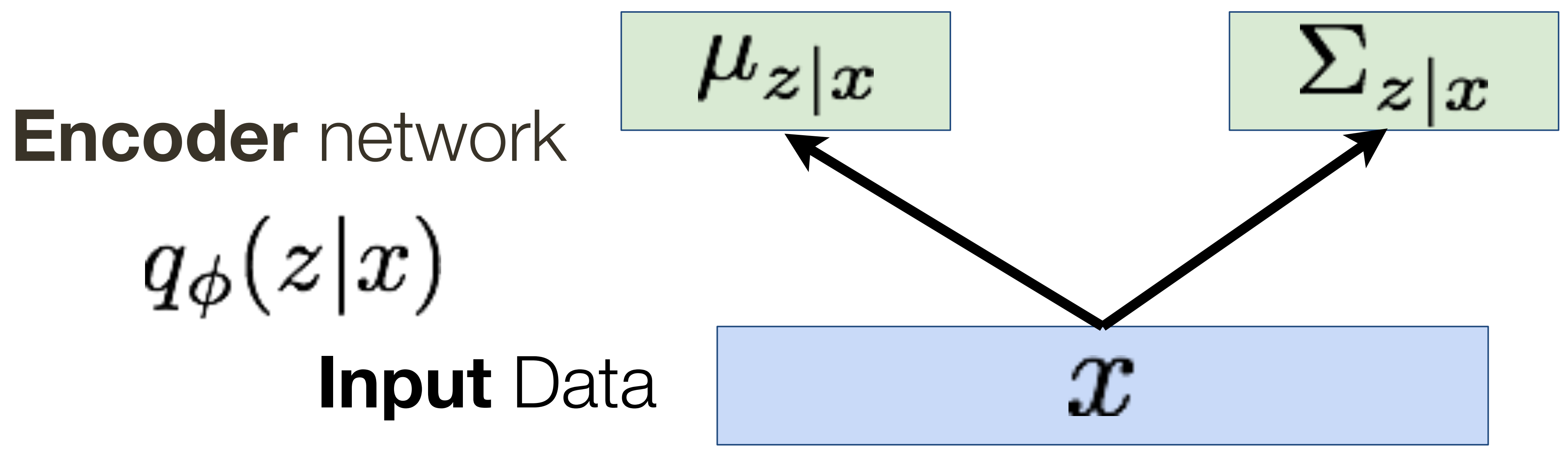
Lets look at **computing the bound** (forward pass)
for a given mini batch of input data

Input Data x

Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

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



Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

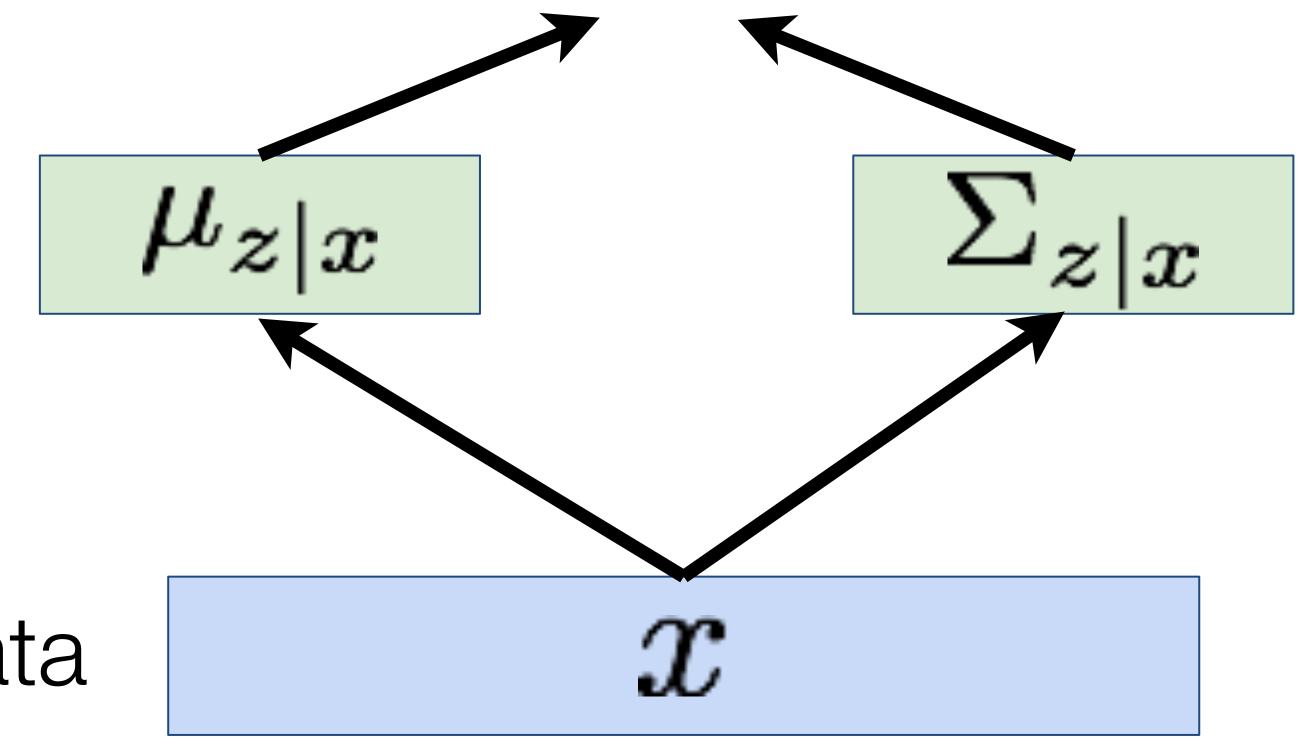
$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)}$$

Make approximate posterior distribution close to prior

Encoder network

$q_\phi(z|x)$

Input Data



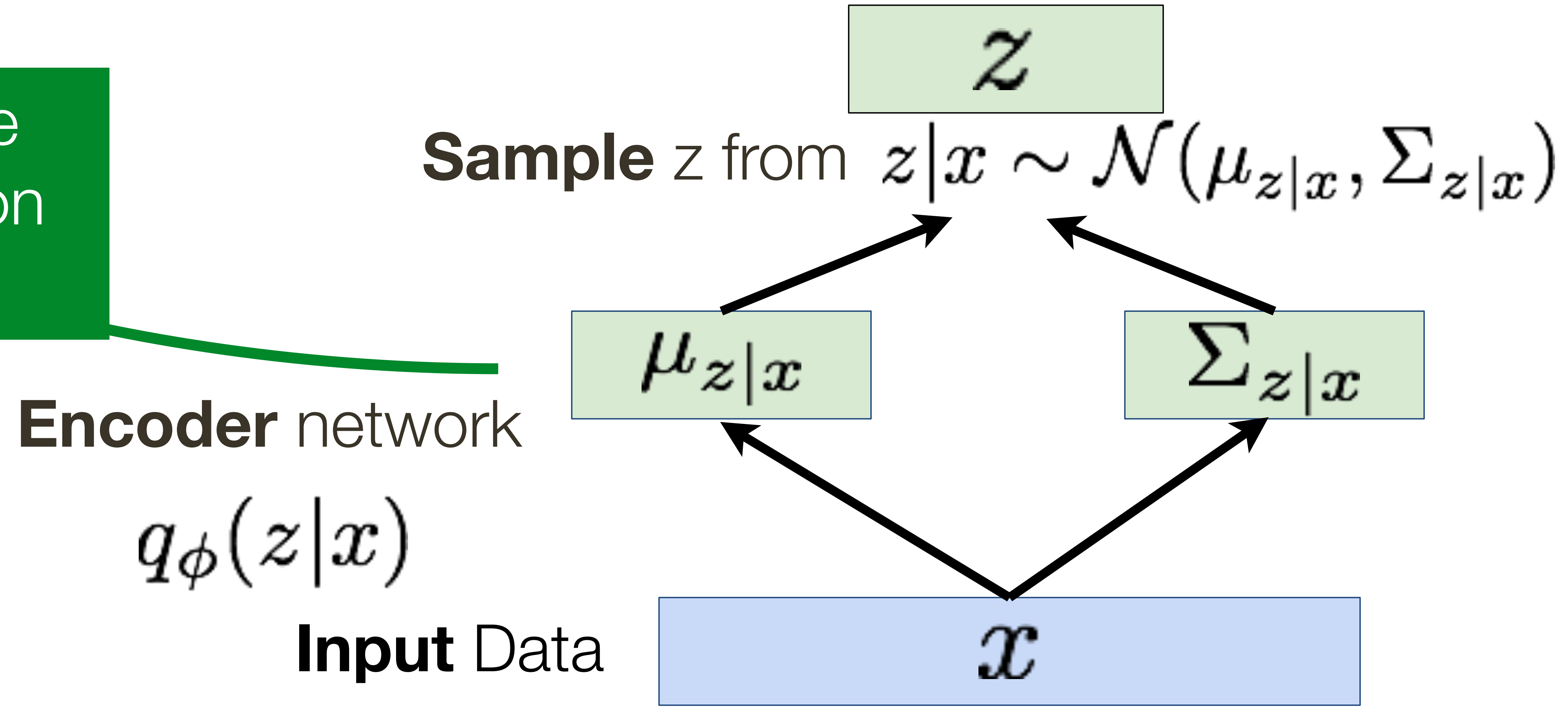
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior



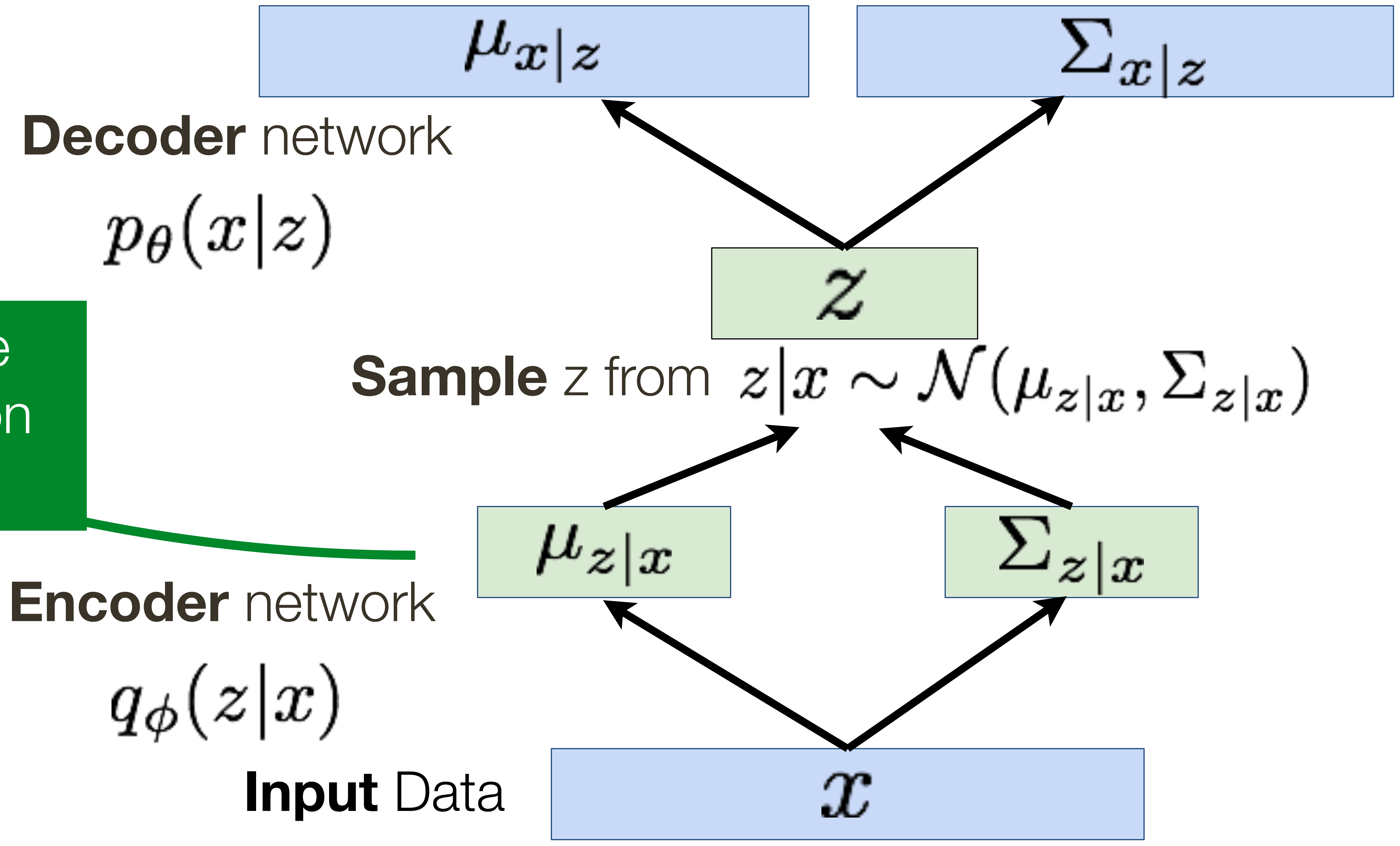
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior



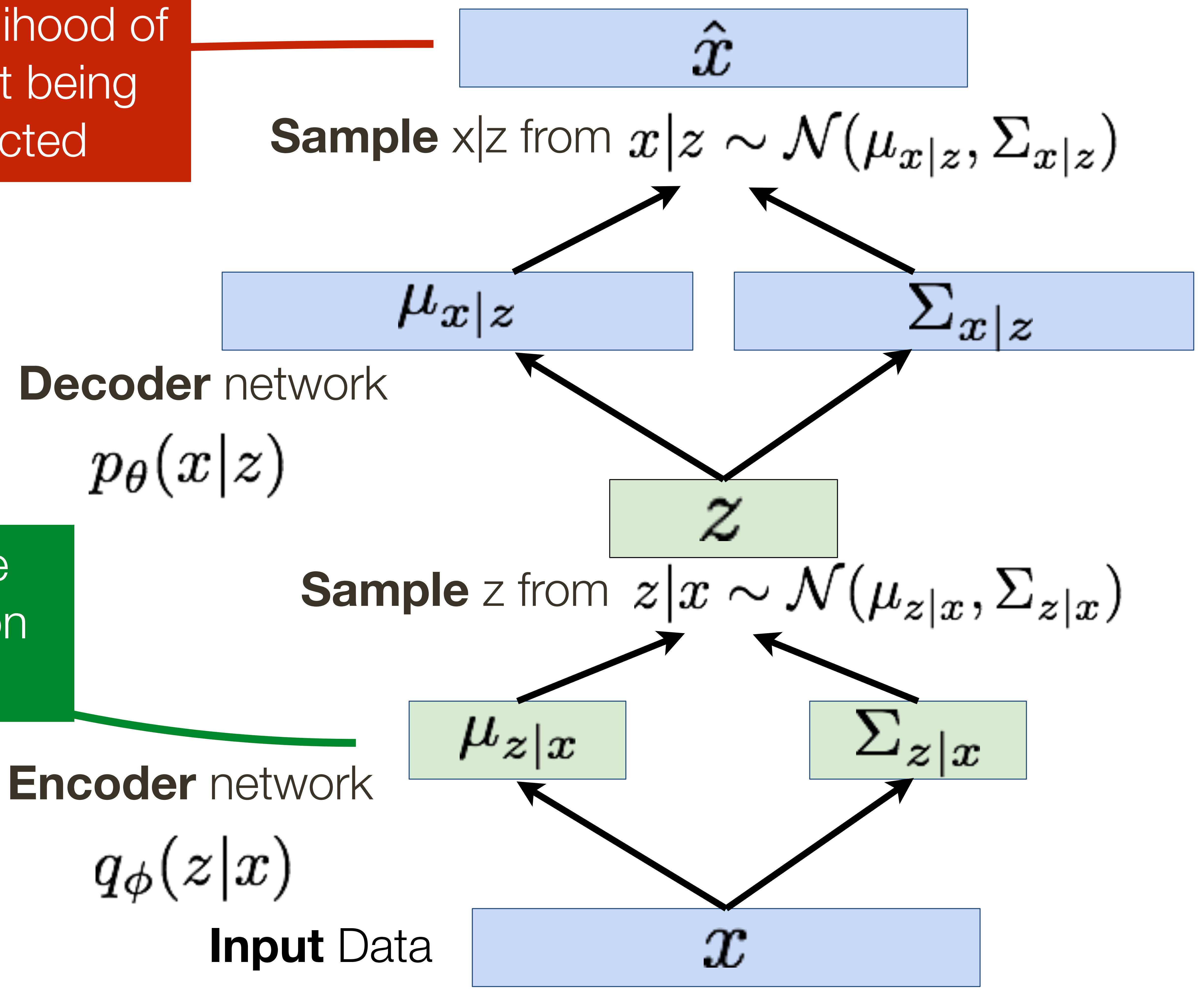
Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

Maximize likelihood of original input being reconstructed

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

Make approximate posterior distribution close to prior



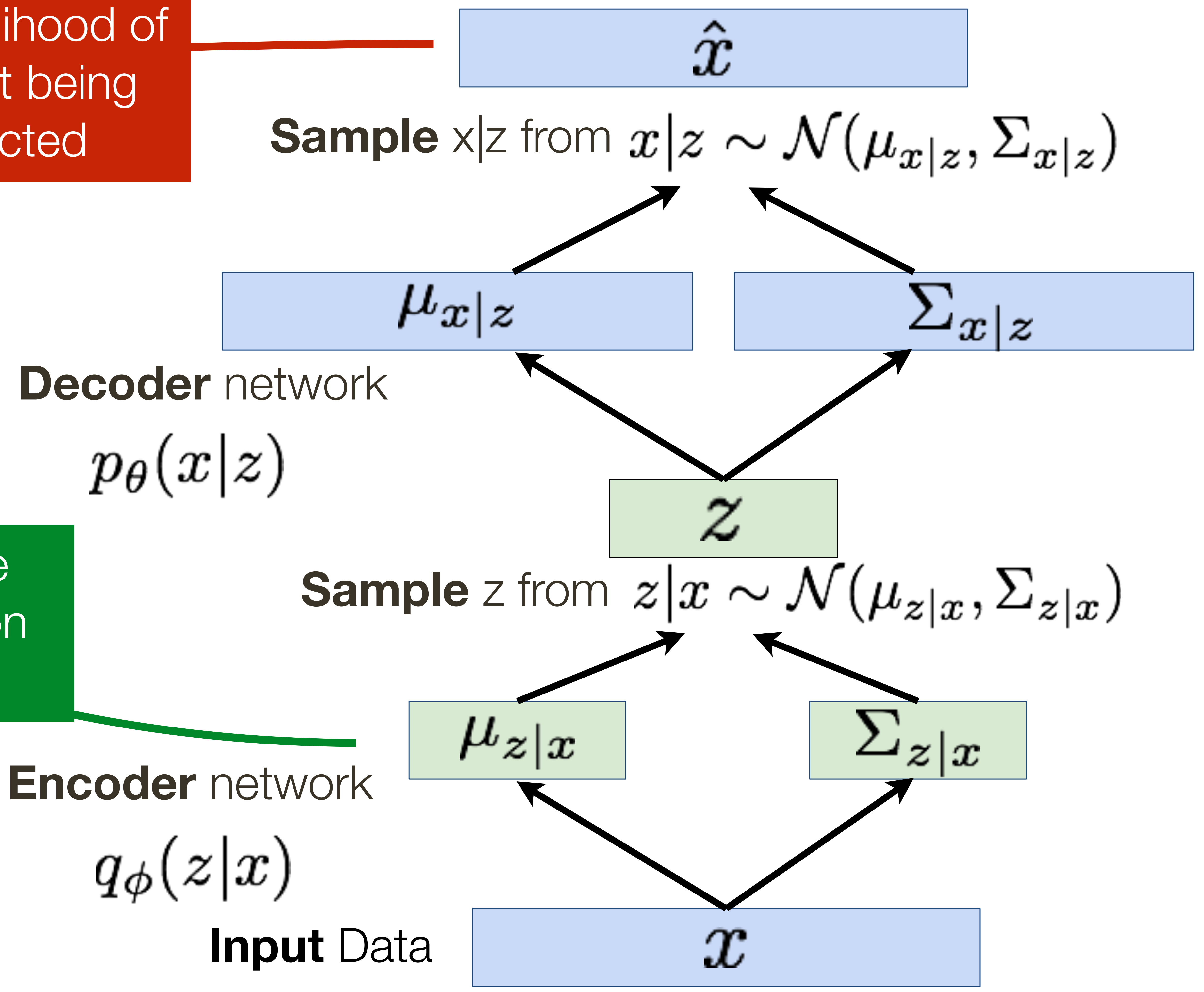
Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

Maximize likelihood of original input being reconstructed

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

Make approximate posterior distribution close to prior



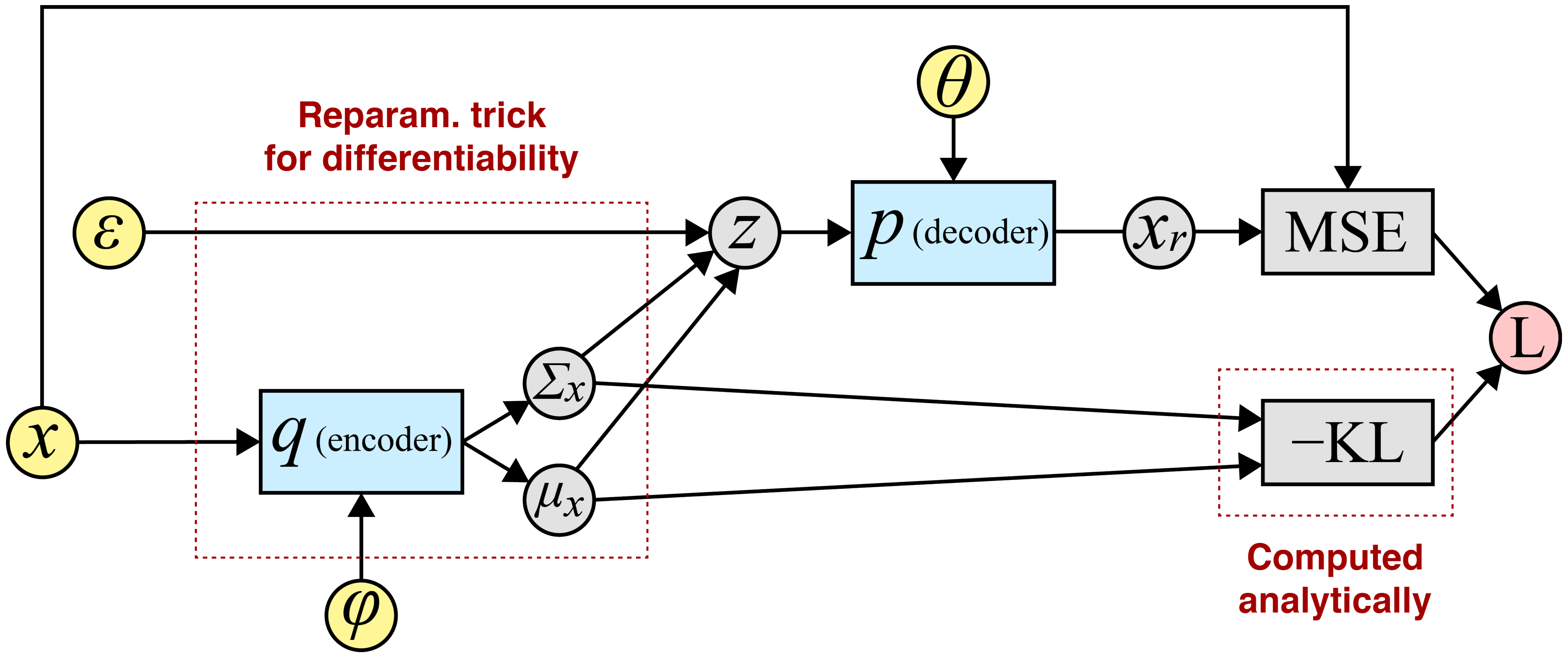
For every minibatch of input data: compute this forward pass, and then backprop!

Reparametrization Trick

$$\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x = M(\mathbf{x}), \Sigma(\mathbf{x})$$ Push $\mathbf{x}$ through encoder

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, 1)$$ Sample noise

$$\mathbf{z} = \boldsymbol{\epsilon}\boldsymbol{\sigma}_x + \boldsymbol{\mu}_x$$ Reparameterize



Source: https://gregorygundersen.com/blog/2018/04/29/reparameterization/

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)

VAE /w (powerful) PixelCNN Decoder

Problem: If the decoder is too powerful, it may just ignore the latent variables (i.e. posterior collapse). This happens when the decoder can make the reconstruction loss incredibly small, such that the regularization term dominates the loss function. In such a case, the encoder will learn to reduce the regularization term, and produce meaningless latents to match $p(z) = N(0, 1)$.

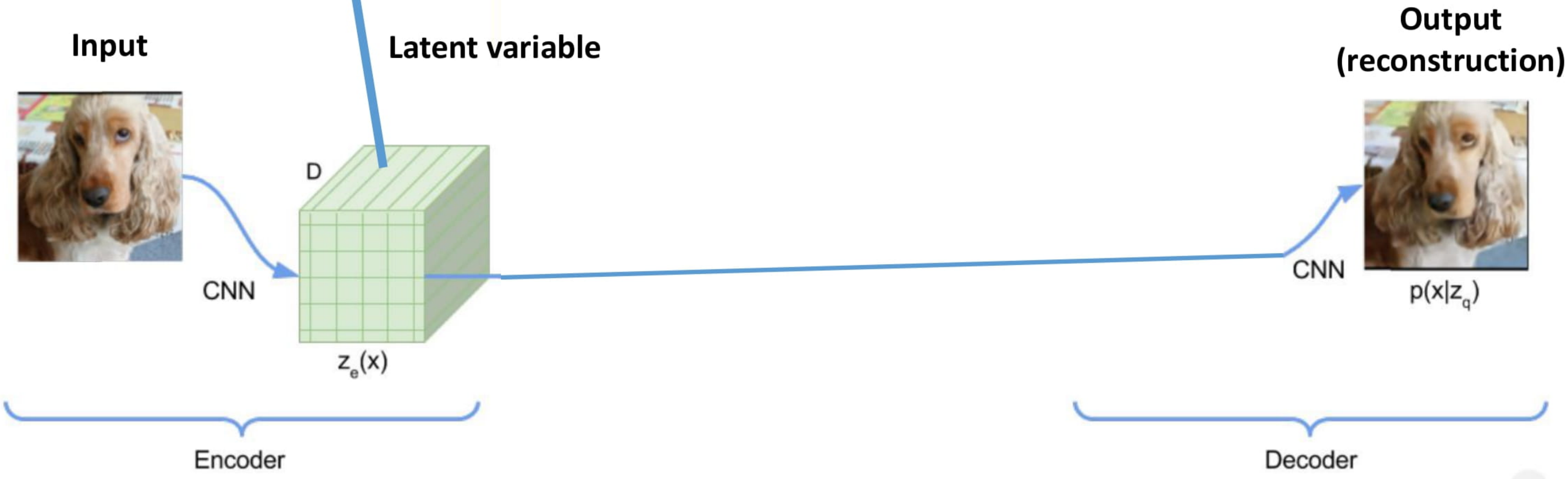
Vector Quantized Variational Autoencoders (VQ-VAE)

Autoencoders Reminder …

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

How to discretize?

For the example:
We take this to be a 4 x 4 image
with 2 channels.

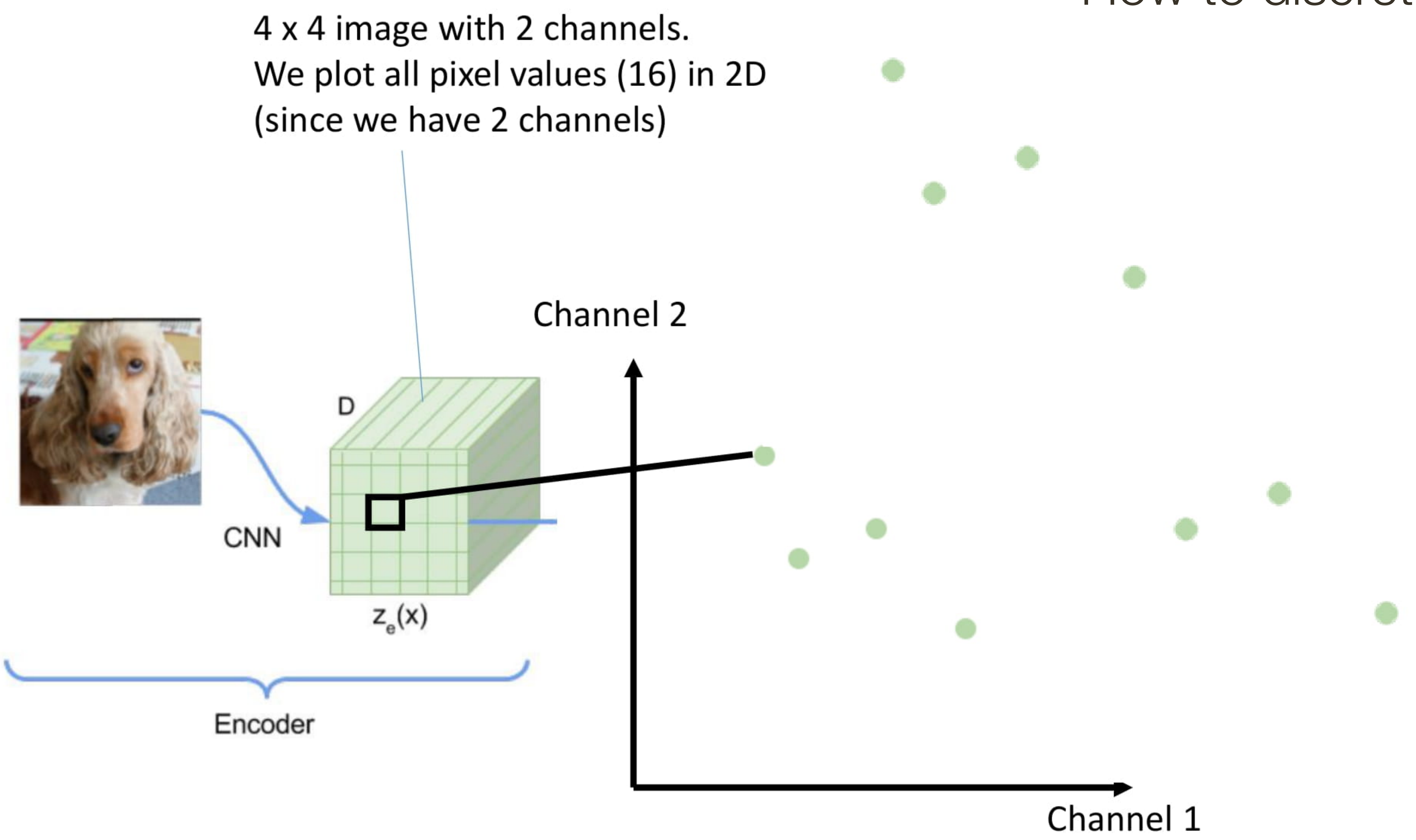


We can train this system end-to-end
using MSE (reconstruction loss)

Autoencoders Reminder …

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

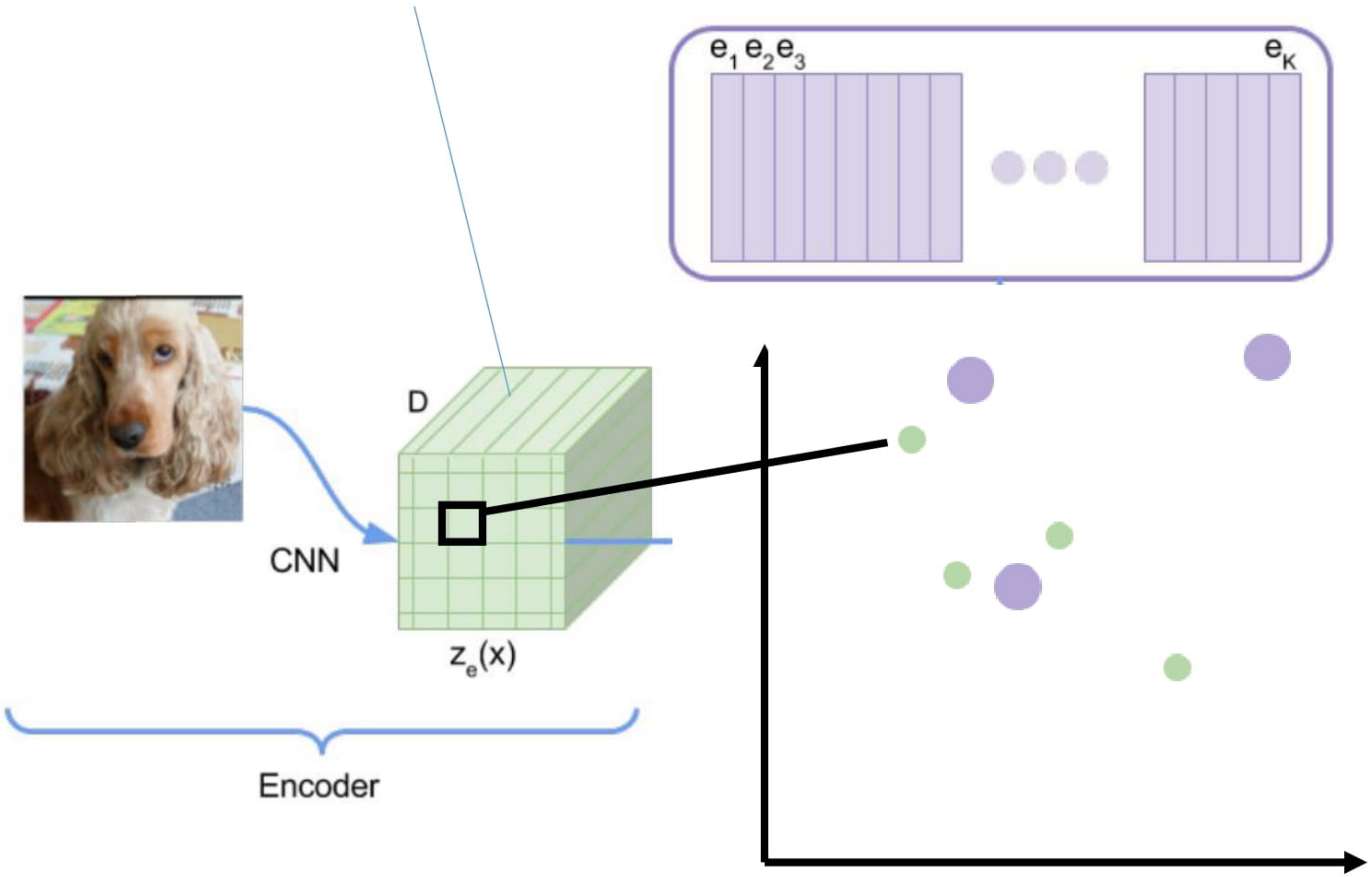
How to discretize?



Vector Quantized - VAE

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

4 x 4 image with 2 channels.

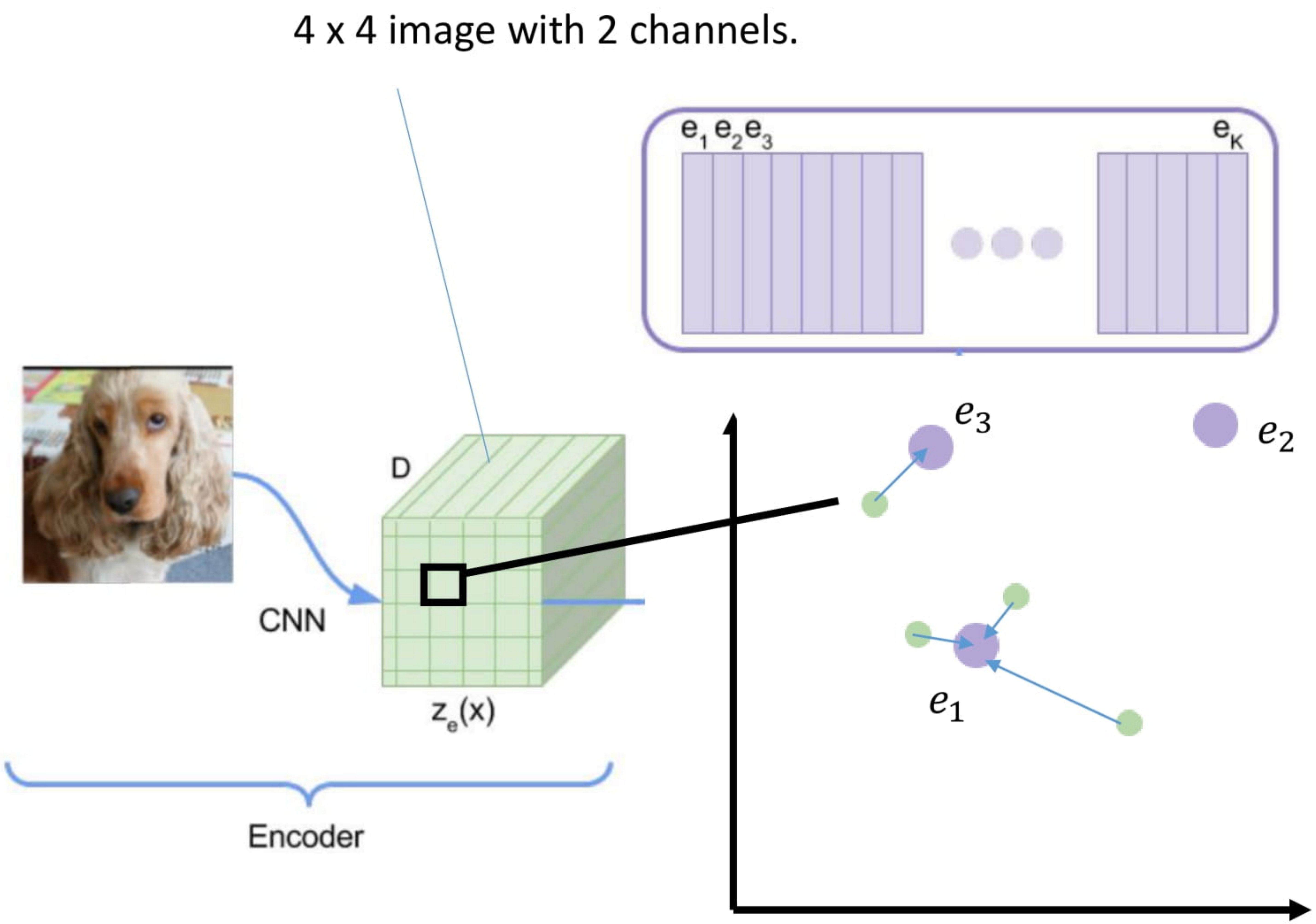


Make dictionary of vectors
$e_1, \ldots, e_K$

Each e_i has 2 dimensions.

Vector Quantized - VAE

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf



Make dictionary of vectors $e_1, \ldots, e_K$

Each e_i has 2 dimensions.

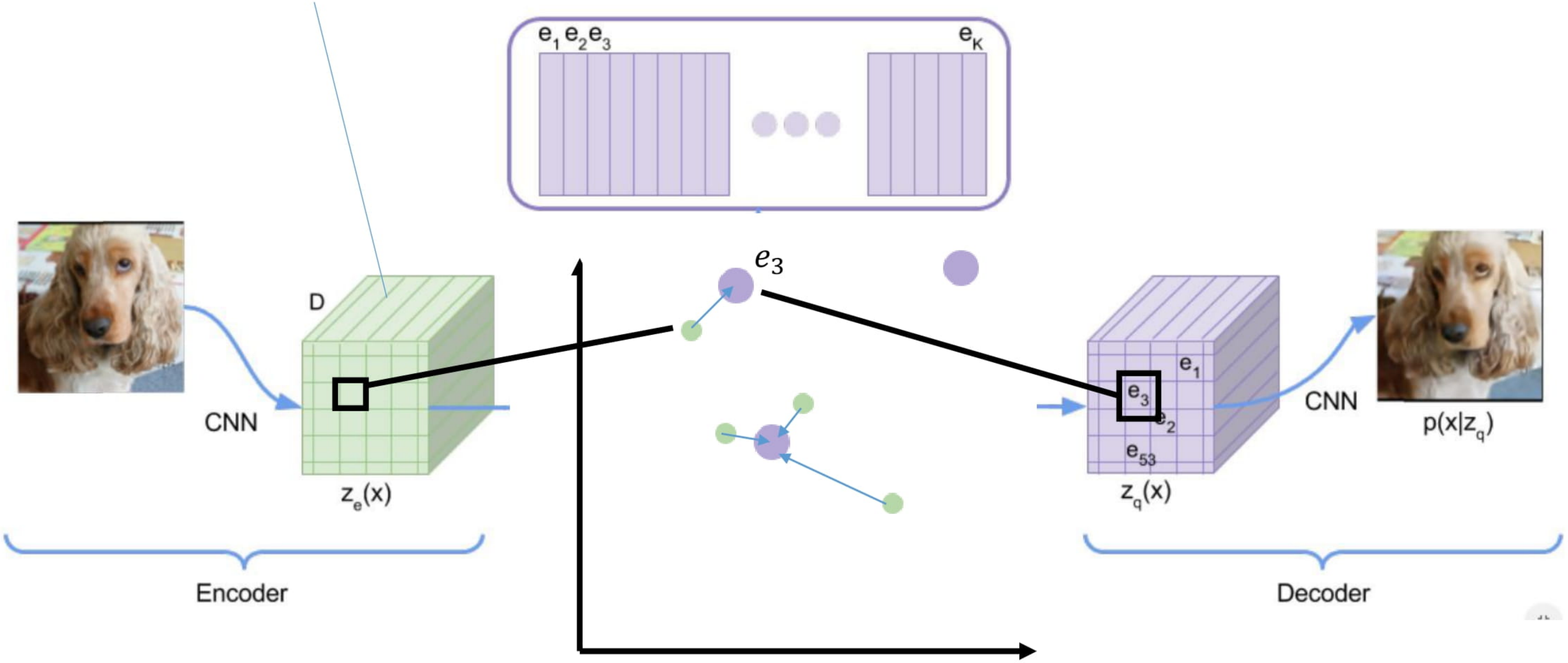
For each latent pixel, look up nearest dictionary element e

Vector Quantized - VAE

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

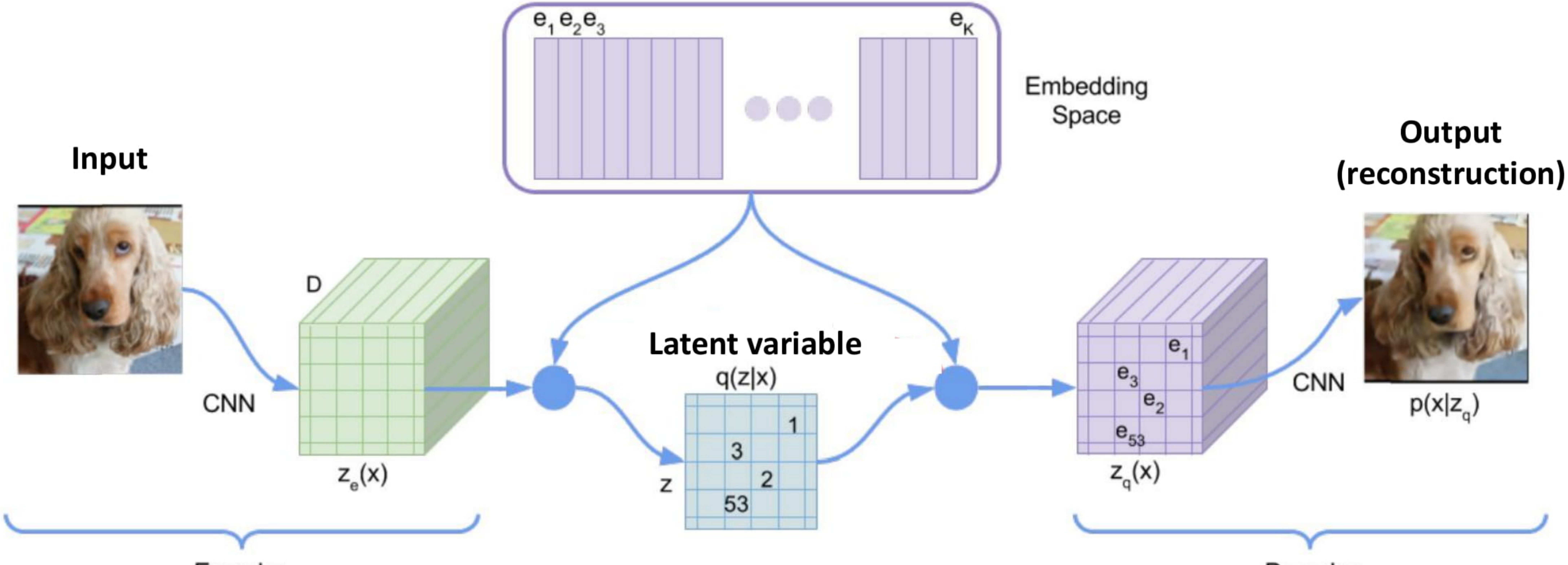
4 x 4 image with 2 channels.

Each e_i has 2 dimensions.



Vector Quantized - VAE

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

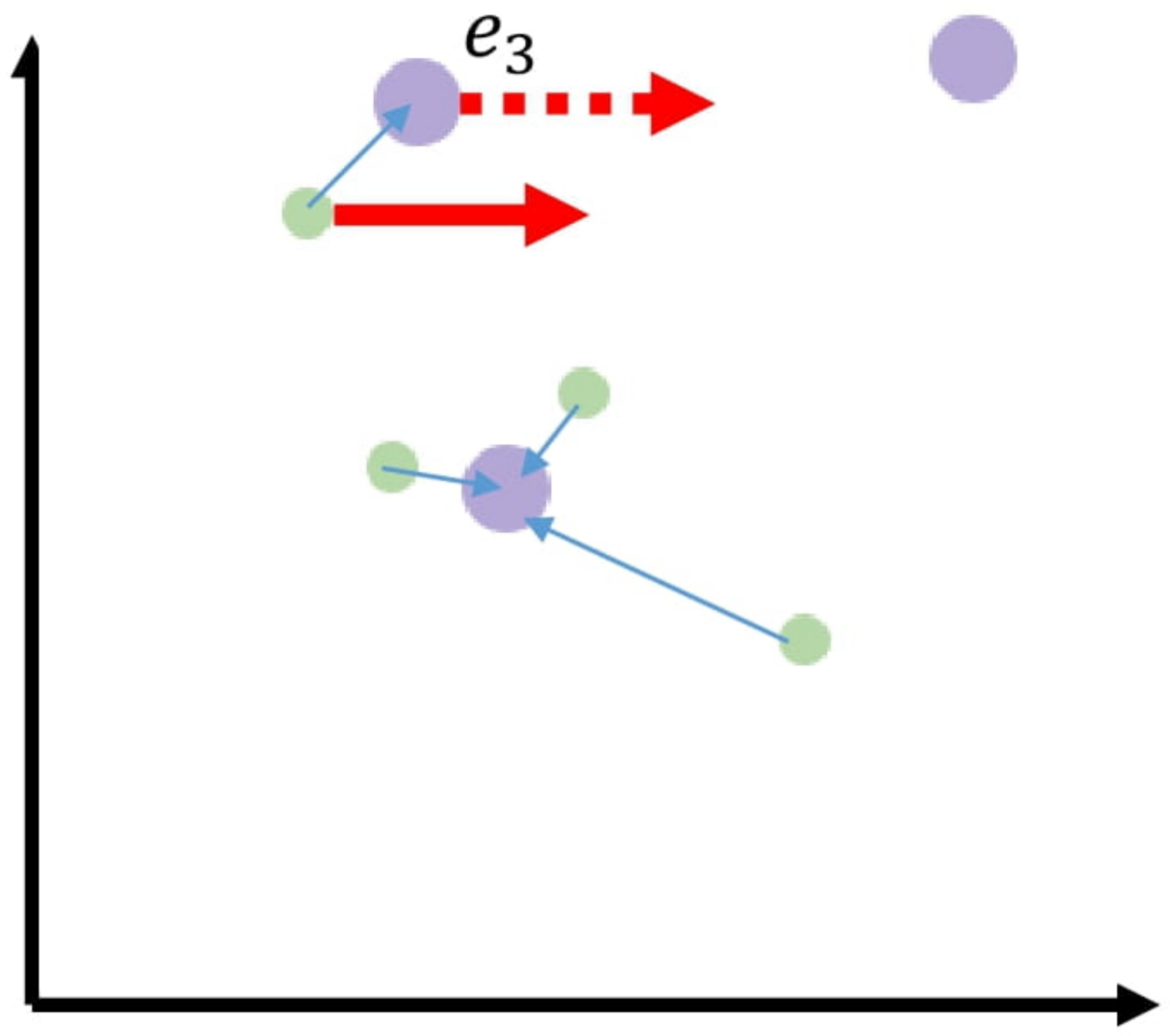


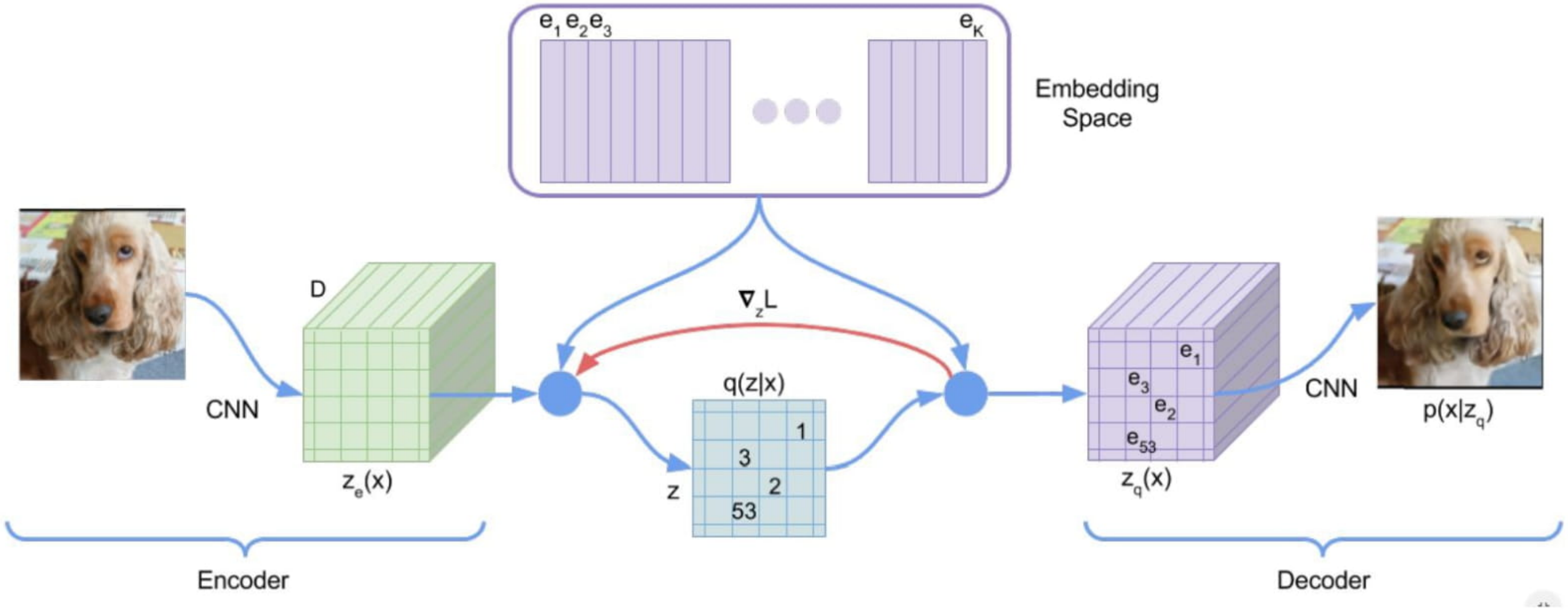
Latent is 1 channel image and contains the id of each e for each pixel (**discrete**).

VQ-VAE — **Training**

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

- How to backpropegate through the discretization?
  - Lets say a gradient is incoming to a dictionary vector
  - We do not update the dictionary vector (fixed)
  - Instead we apply the gradient of e to the non-discretized vector



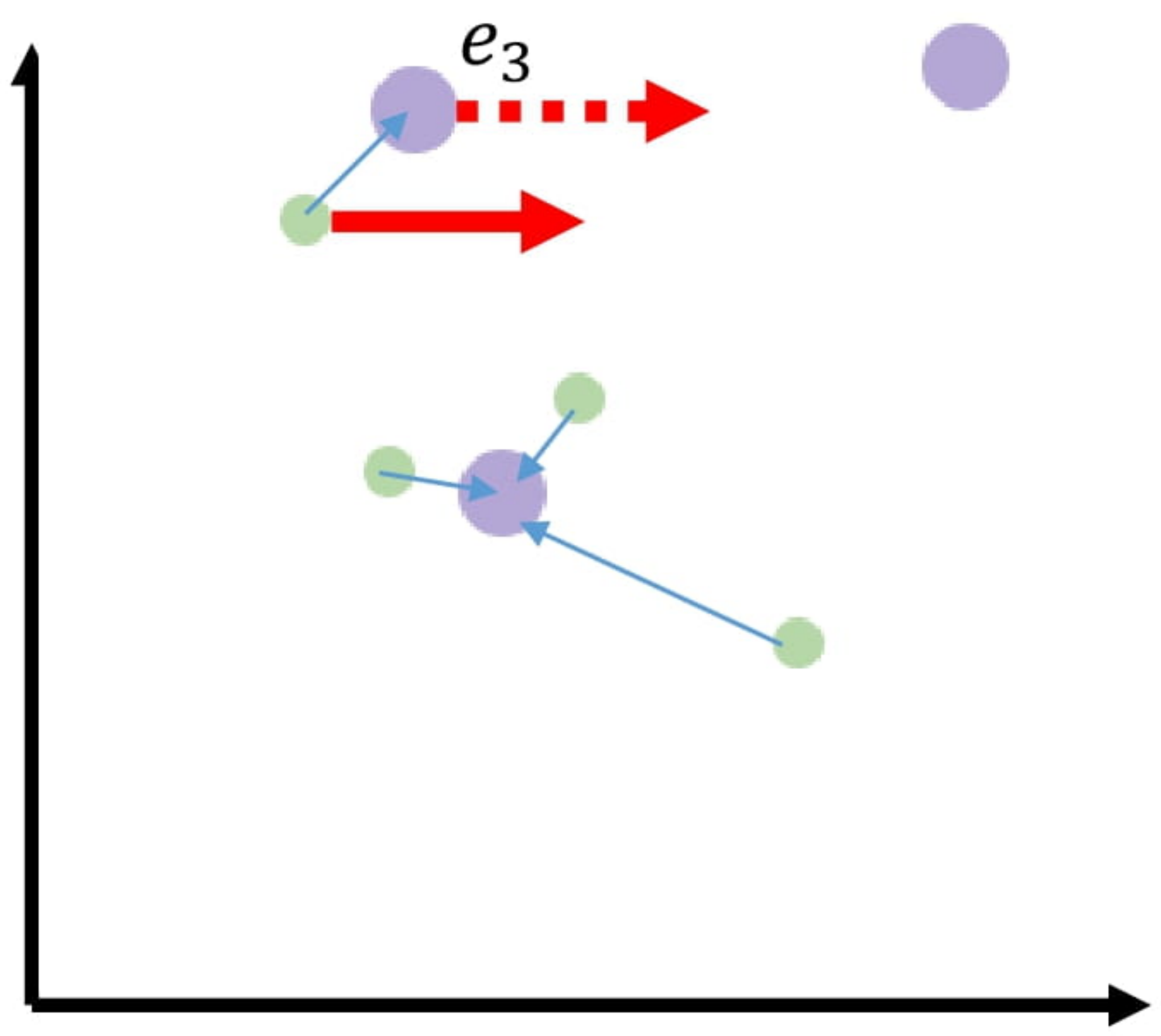


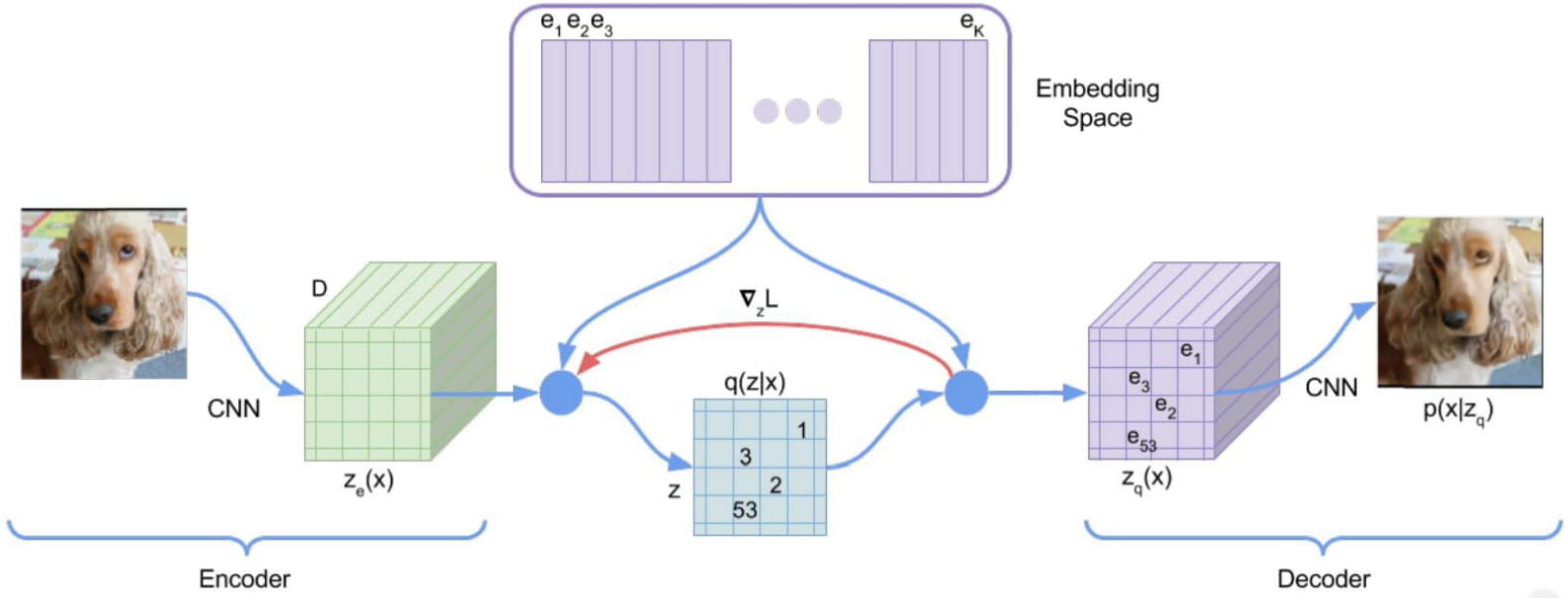
VQ-VAE — **Training**

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \text{sg}[e]\|_2^2,$$

- How to backpropegate through the discretization?
  - Lets say a gradient is incoming to a dictionary vector
  - We do not update the dictionary vector (fixed)
  - Instead we apply the gradient of e to the non-discretized vector





VQ-VAE — **Training**

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

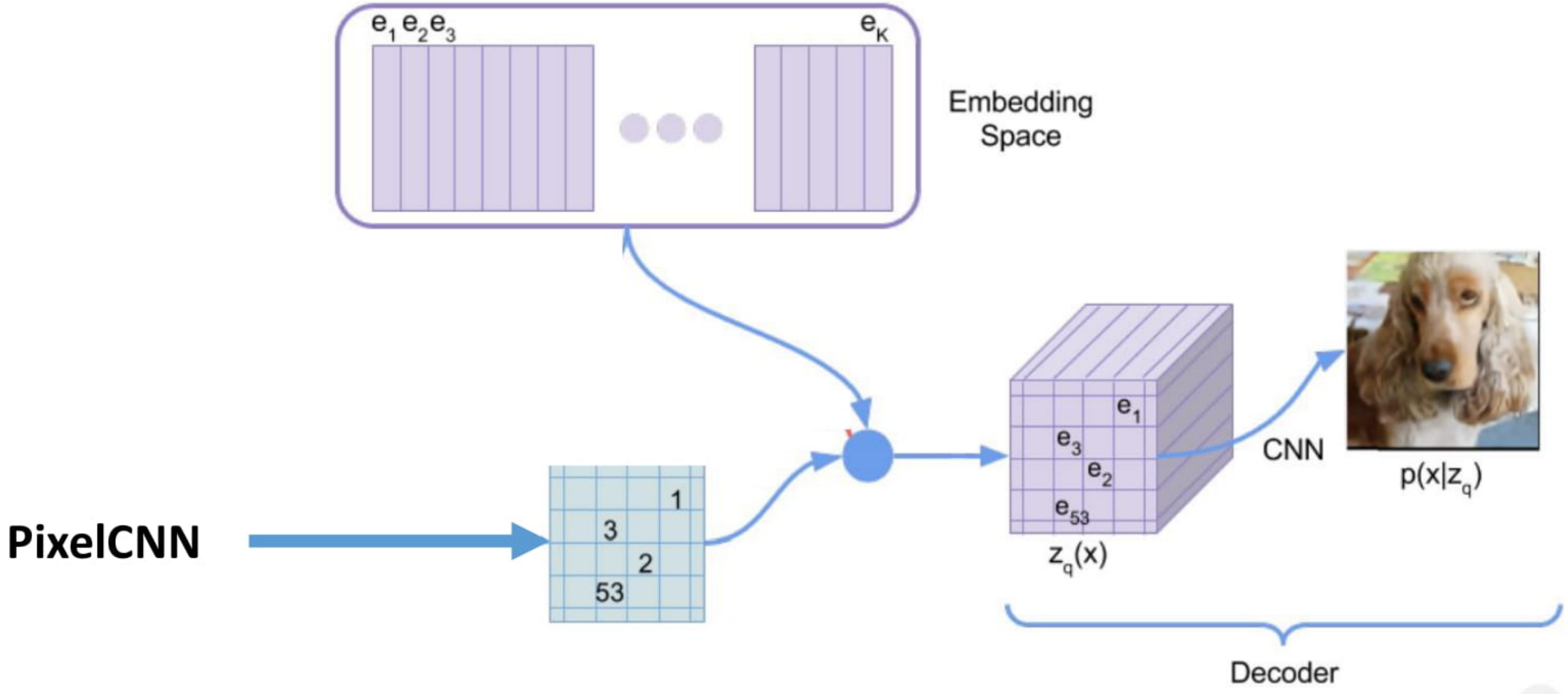
$$L = \log p(x|z_q(x)) + \|\mathrm{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \mathrm{sg}[e]\|_2^2,$$

Reconstruction loss, which optimizes both the encoder and decoder

Regularization, ensures that encoder does not grow arbitrarily

VQ loss, which moves the embedding vectors towards encoder outputs

VQ-VAE — **Sampling / Generation**



Class: pickup

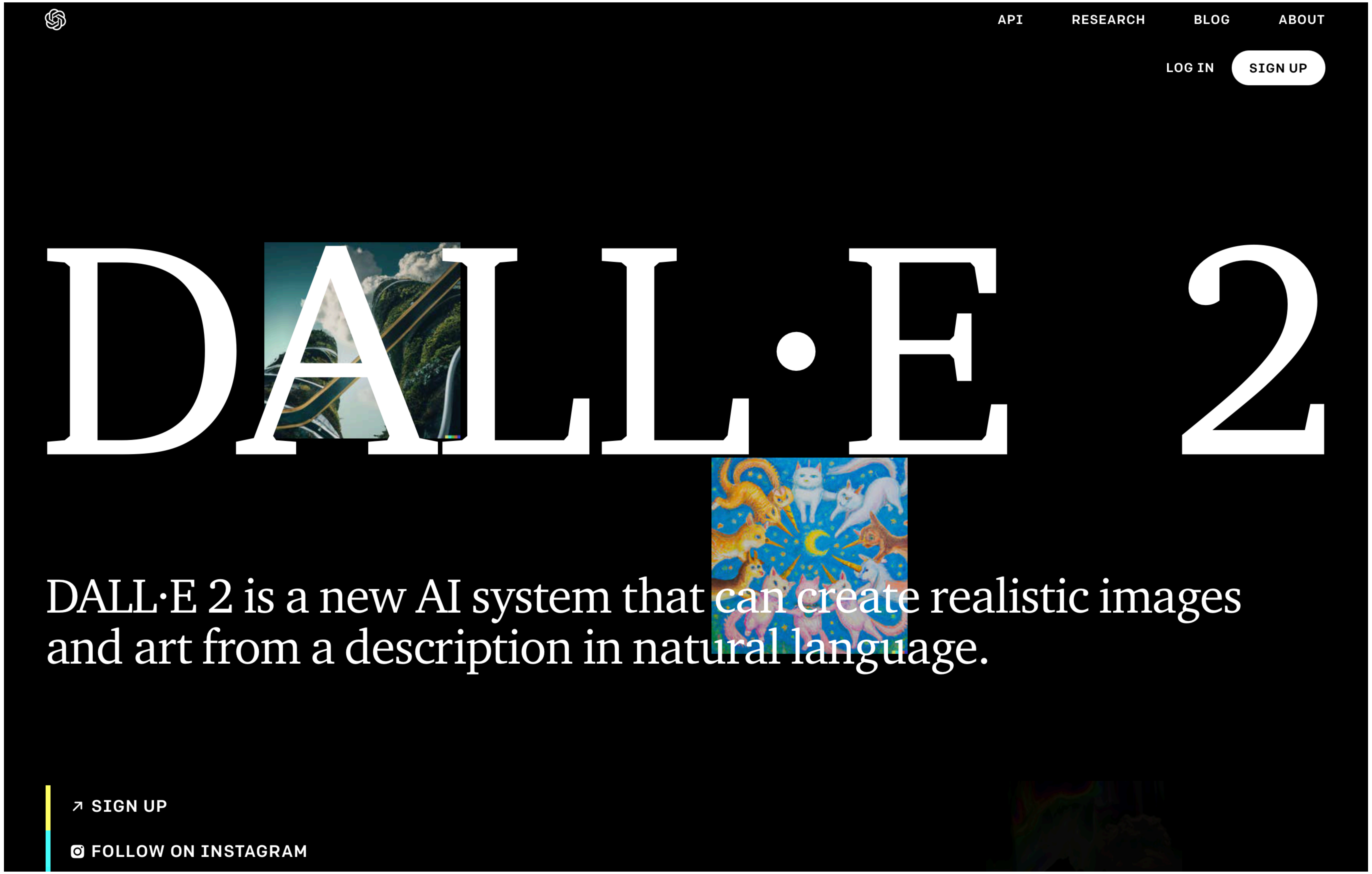
VQ-VAE — **Sampling / Generation**

- Comparable with VAE on CIFAR-10 in terms of density estimation
- Reconstructions on ImageNet are very good

Figure 2: Left: ImageNet 128x128x3 images, right: reconstructions from a VQ-VAE with a 32x32x1 latent space, with K=512.

VQ-VAE vs. GAN

VQ-VAE (Proposed)　　　　BigGAN deep


API
RESEARCH
BLOG
ABOUT
LOG IN
SIGN UP
DALL·E 2
DALL·E 2 is a new AI system that can create realistic images and art from a description in natural language.
SIGN UP
FOLLOW ON INSTAGRAM

Relationship of VQ-VAE to VAE

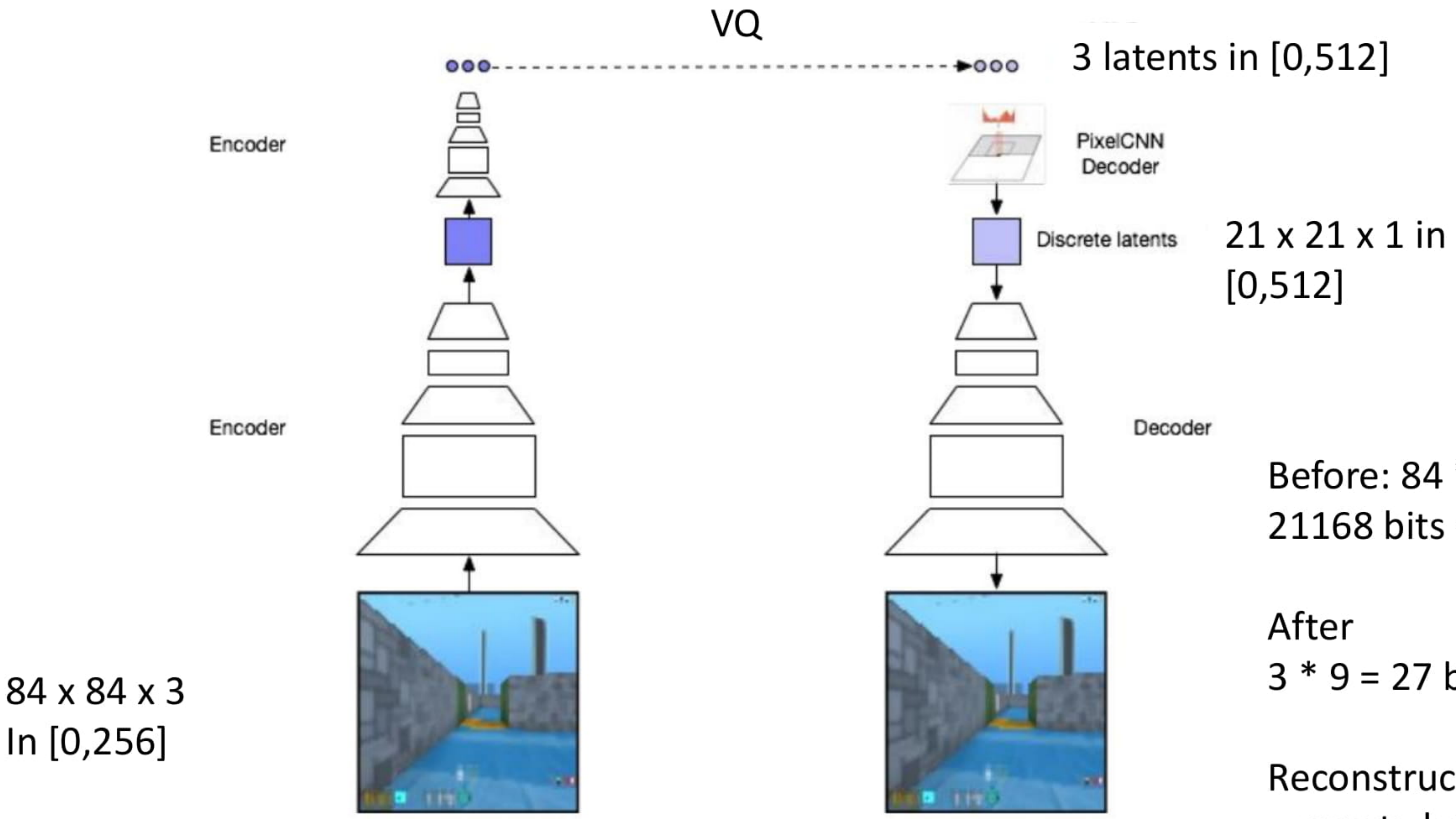
VAE: Assumes Gaussian prior over continuous latent space

VQ-VAE: Assumes uniform categorical distribution over discrete keywords (all keywords are equally likely)

Comparison

	GAN	Variational Autoencoder	Pixel CNN	VQ-VAE (This talk)
Compute exact likelihood p(x)	✘	✘	✓	✘
Has latent variable z	✓	✓	✘	✓
Compute latent variable z (inference)	✘	✓	✘	✓
Discrete latent variable	✘	✘	✘	✓
Stable training?	✘	✓	✓	✓
Sharp images?	✓	✘	✓	✓?

Multi-stage VQ-VAE



Before: 84 * 84 * 3 * 8 =
21168 bits = 3 Kb

After
3 * 9 = 27 bits

Reconstruction not very accurate but powerful representation

So far …

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

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

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?

So far …

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

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

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

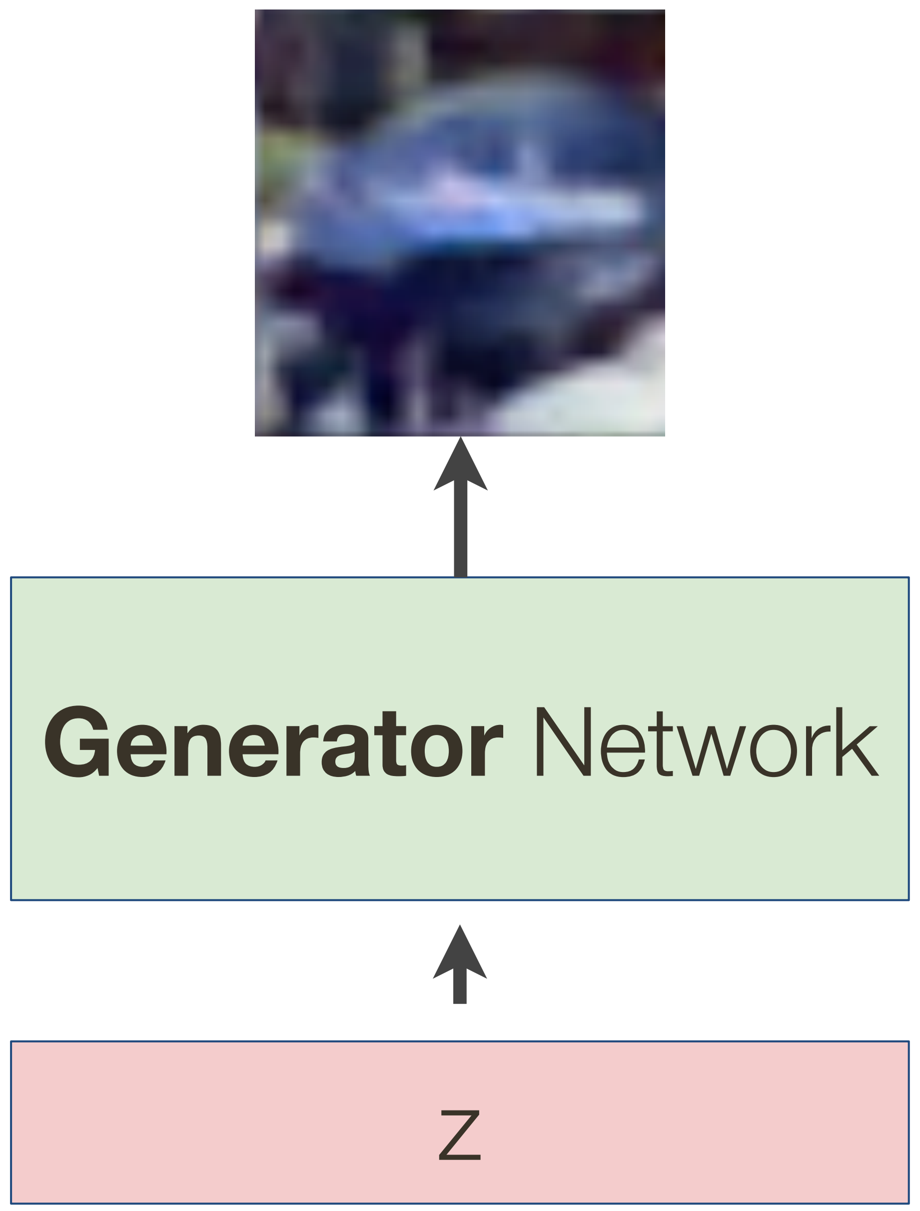
[Goodfellow et al., 2014]

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

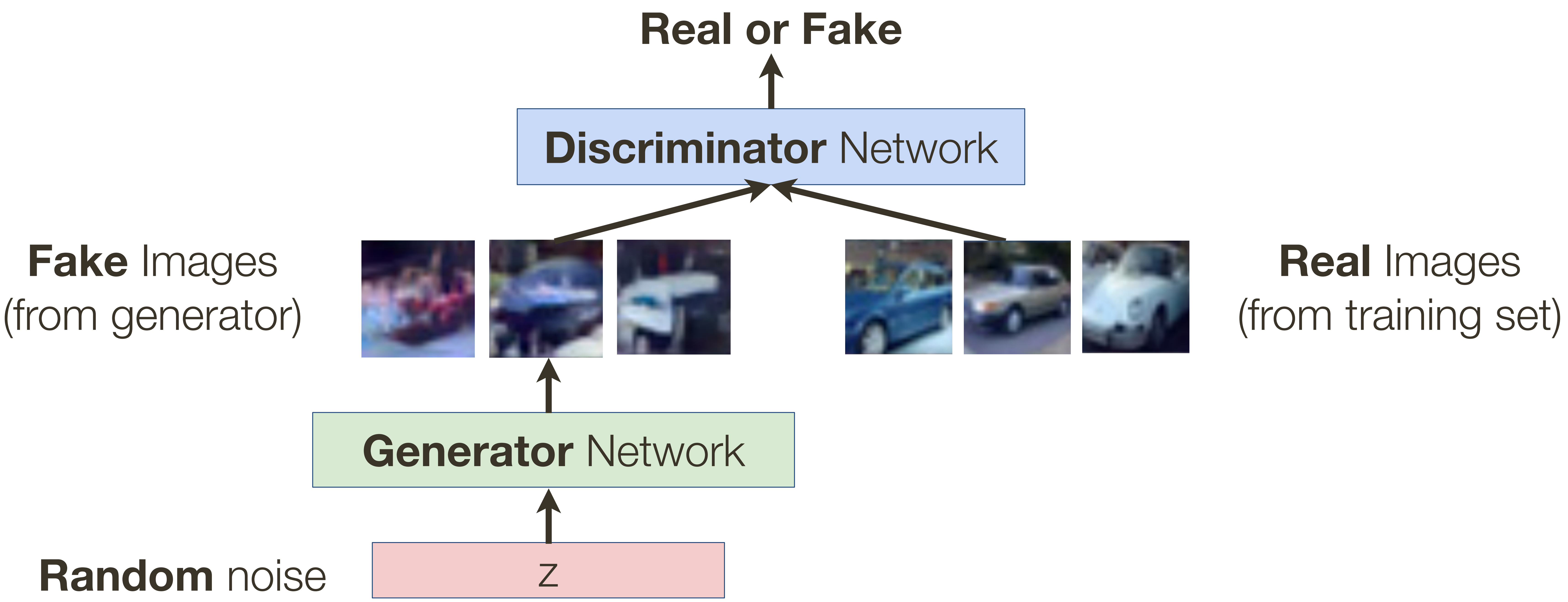
[Goodfellow et al., 2014]

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

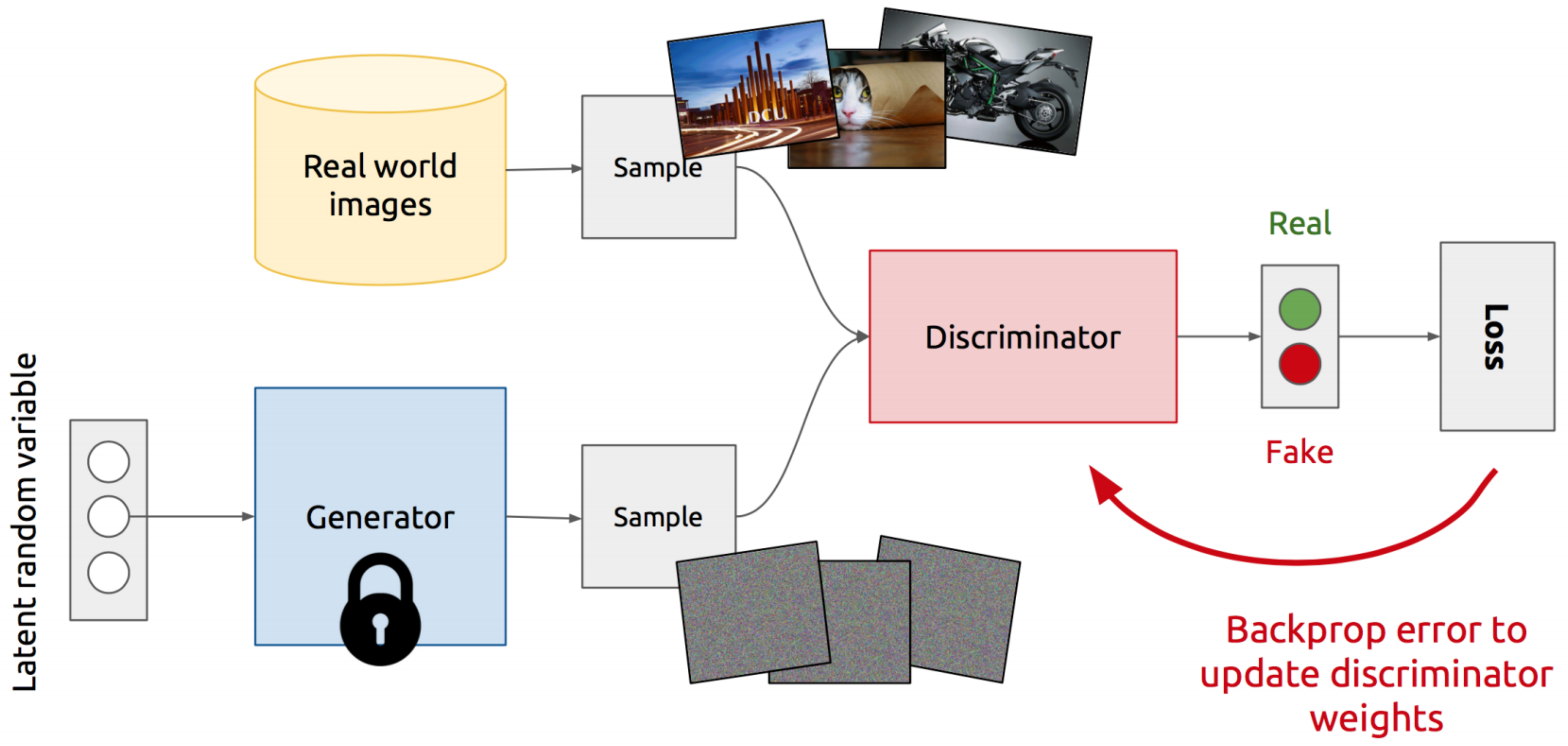
[Goodfellow et al., 2014]

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

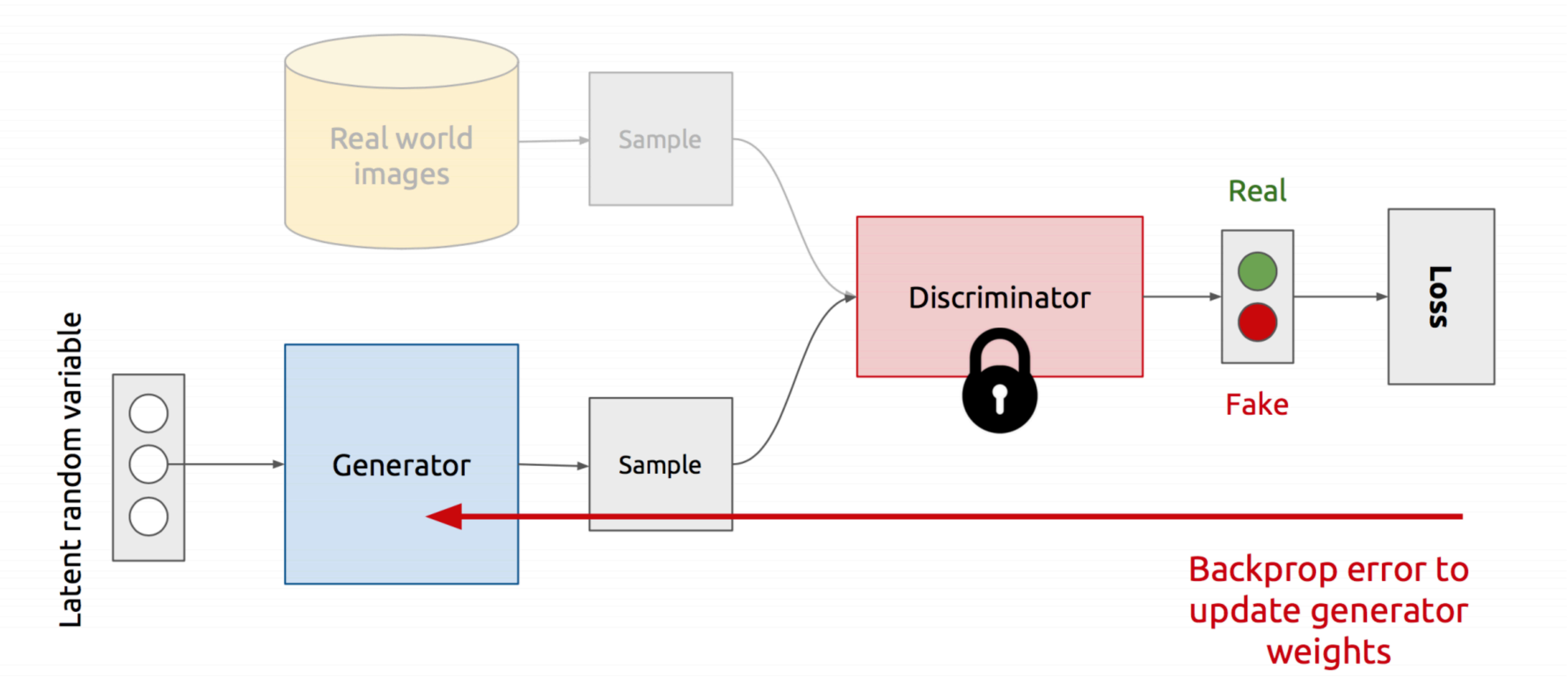
Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



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

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



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

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

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

- **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:

$$p_{data}(x) = p_{gen}(G_{\theta_g}(z)), \quad \forall x \qquad\qquad D_{\theta_d}(x) = 0.5, \quad \forall x$$

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.

Discriminator updates

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ 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 $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ 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.

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

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

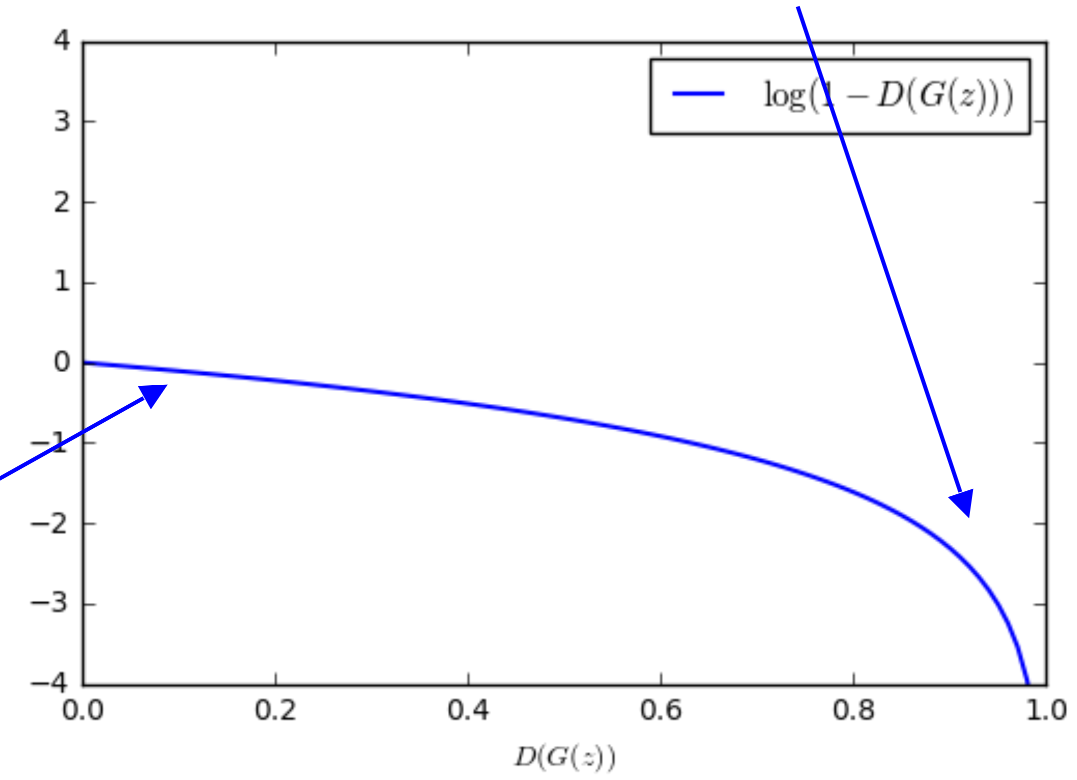
2. Gradient **descent** on generator

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

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!



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

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

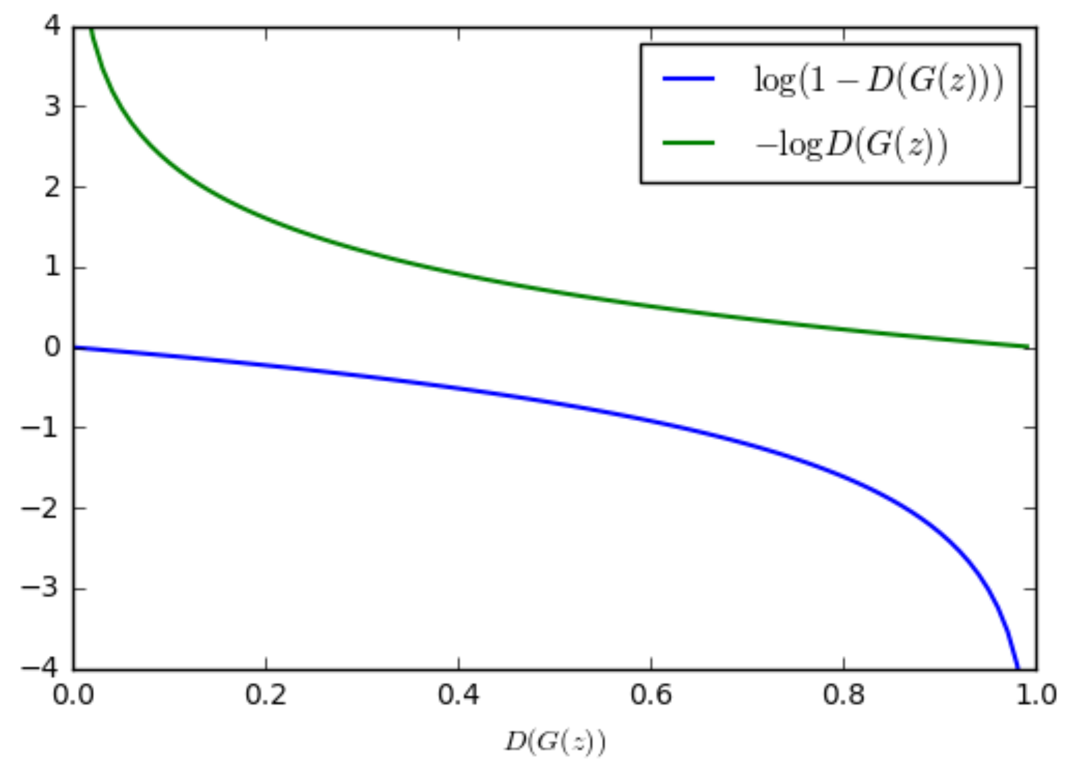
$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

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

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

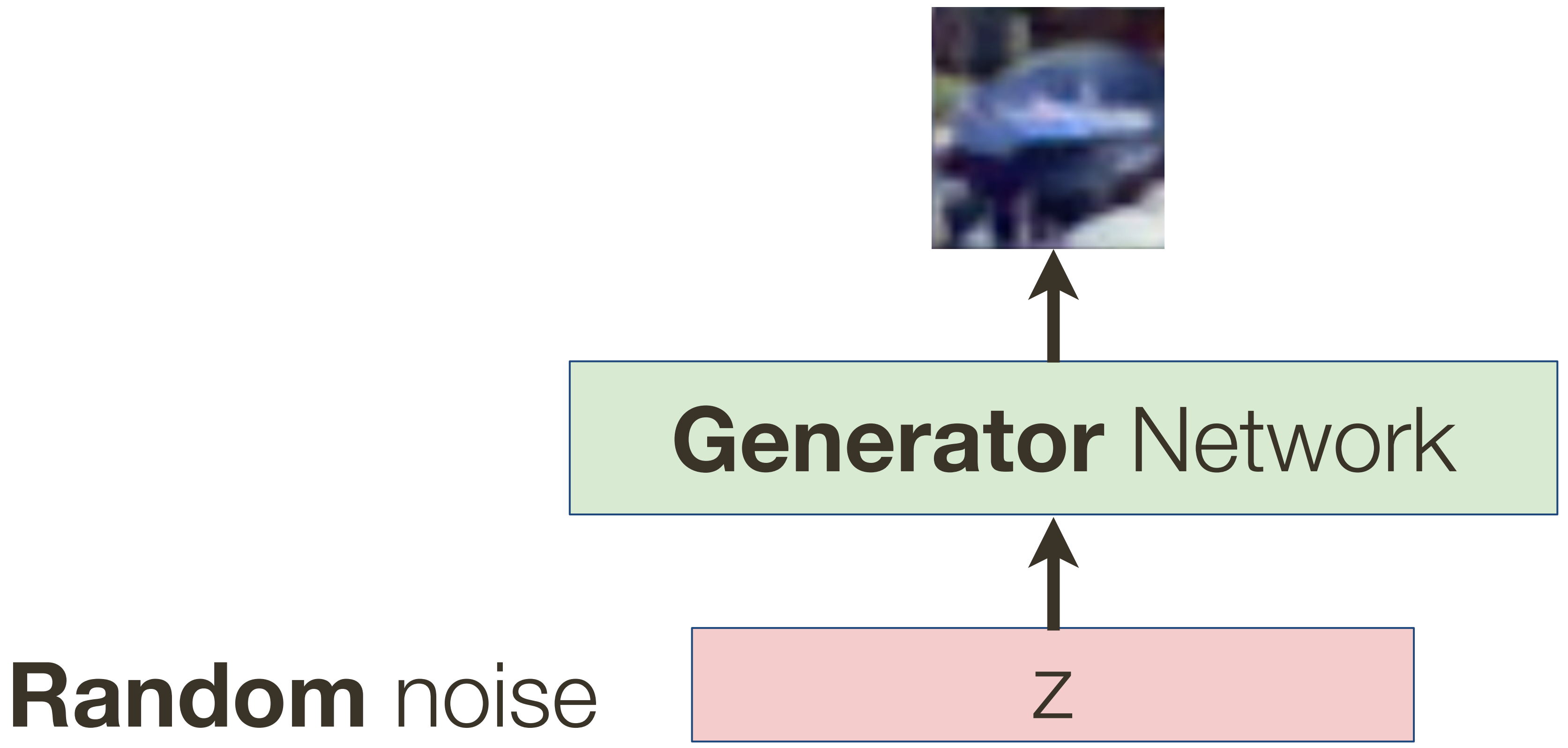
Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.

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



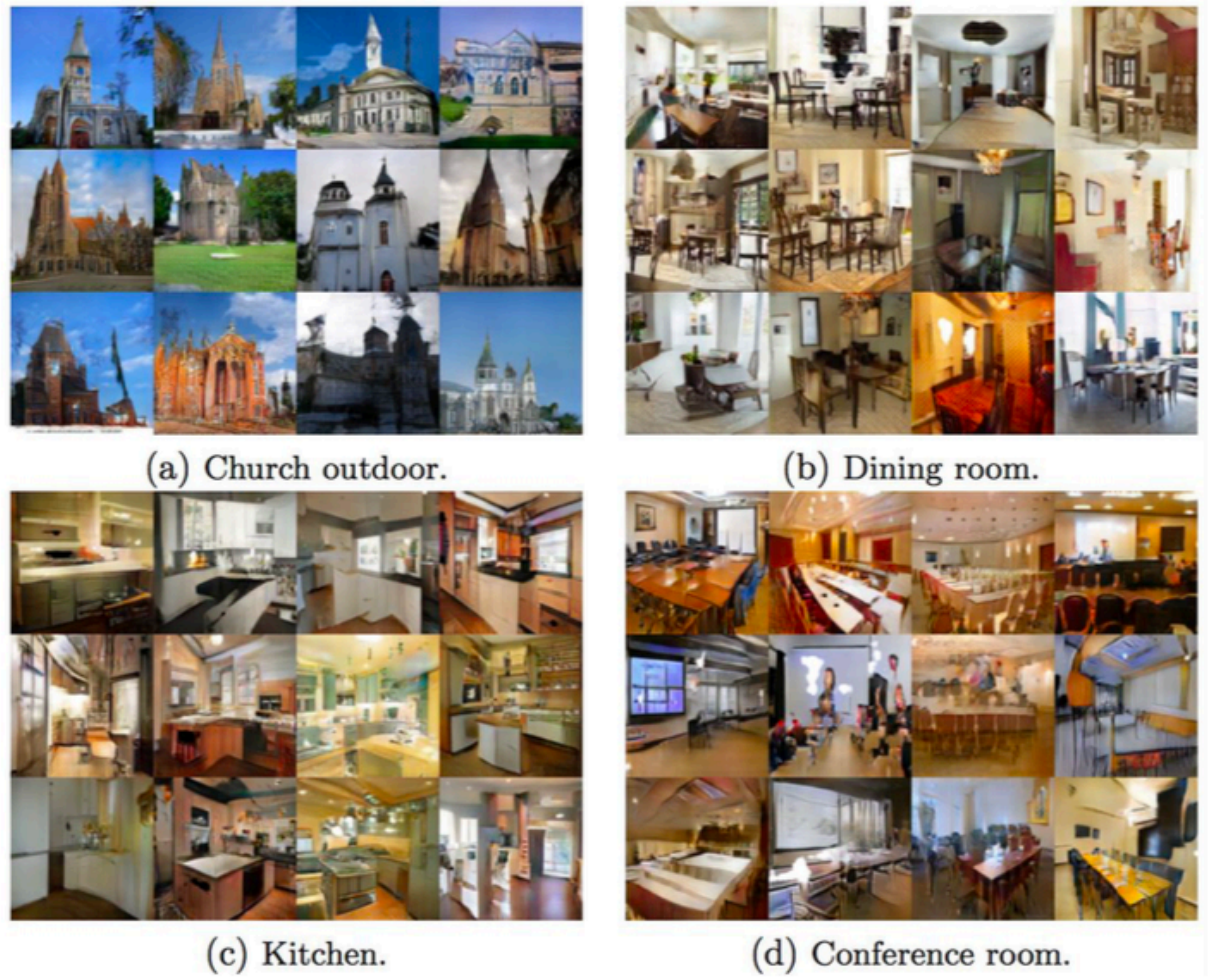
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Sampling **GAN**s

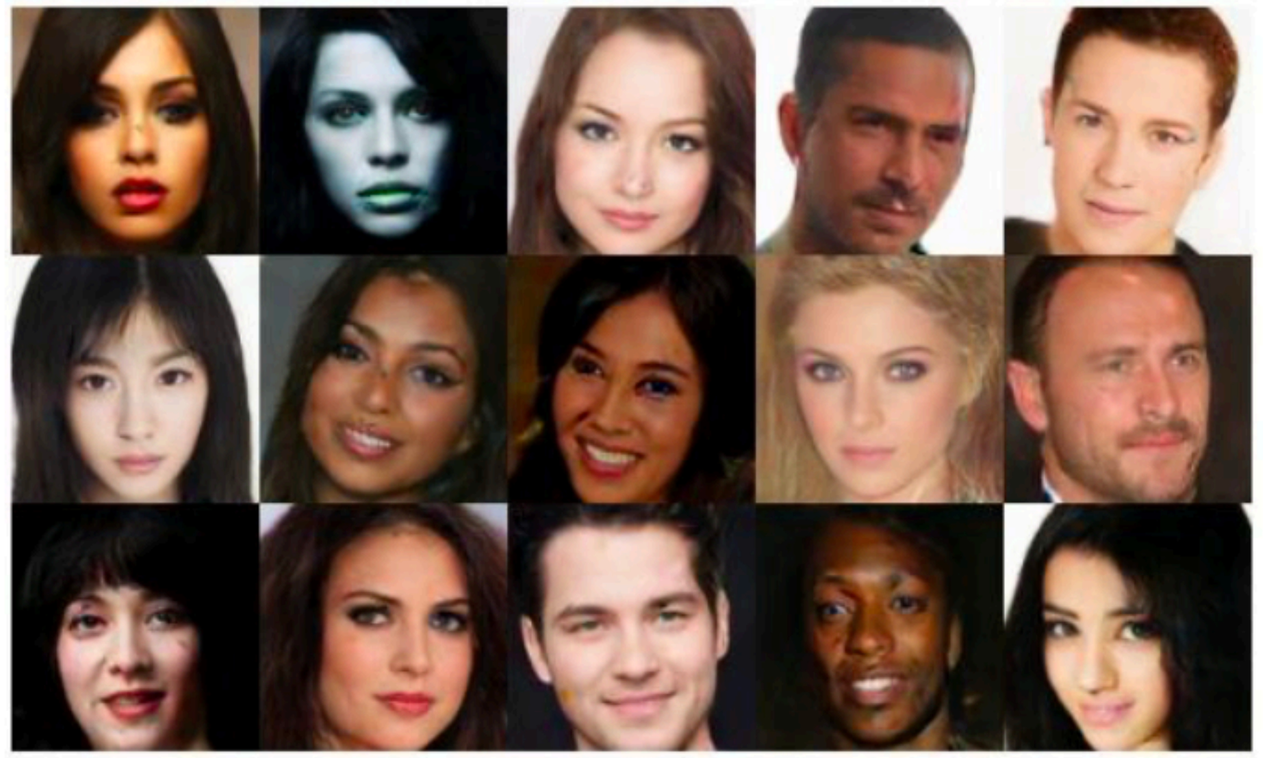


Year of the **GAN**

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

Reed et al. 2017.

Many GAN applications

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

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

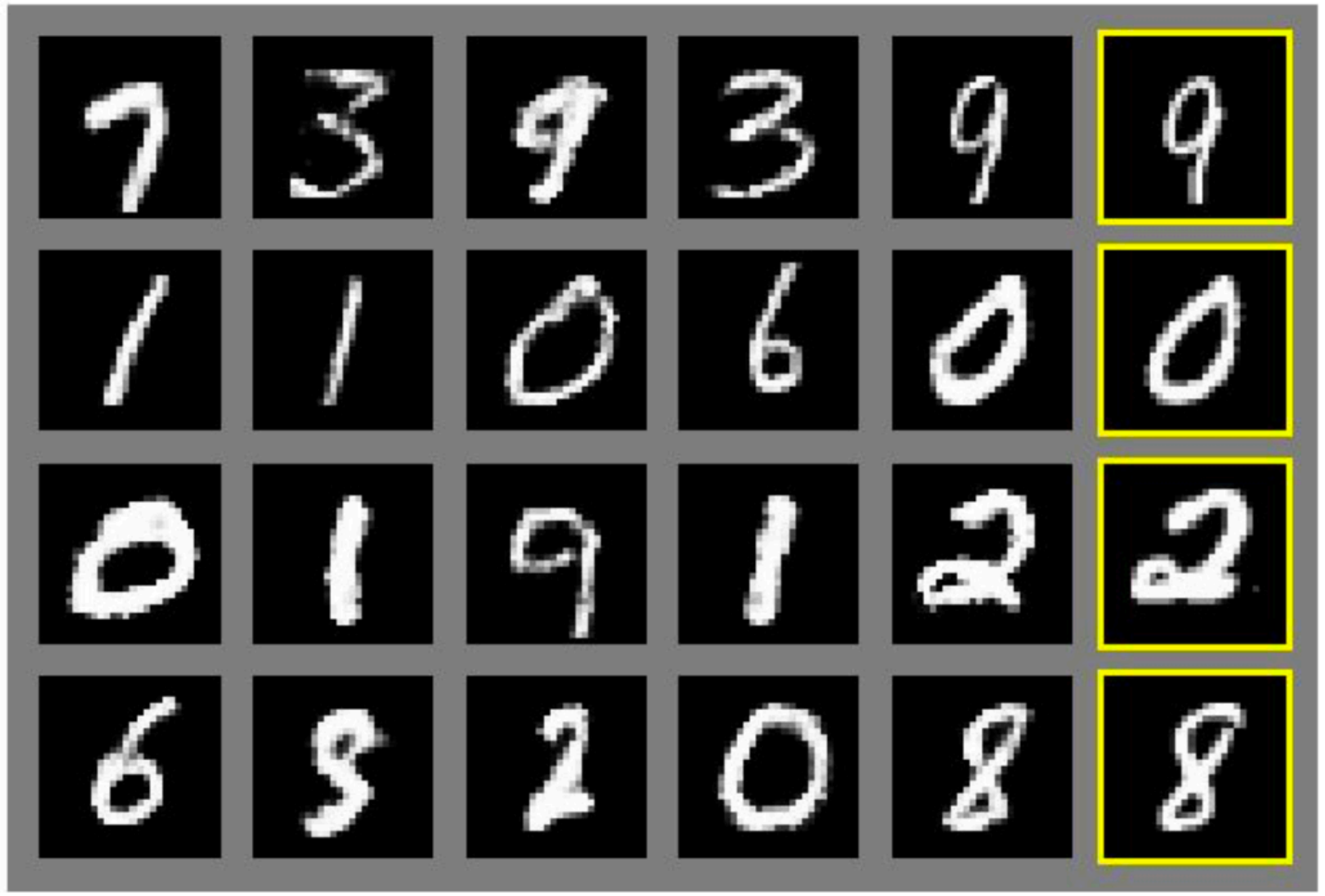
Year of the **GAN**

- 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 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

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

Generative Adversarial Nets

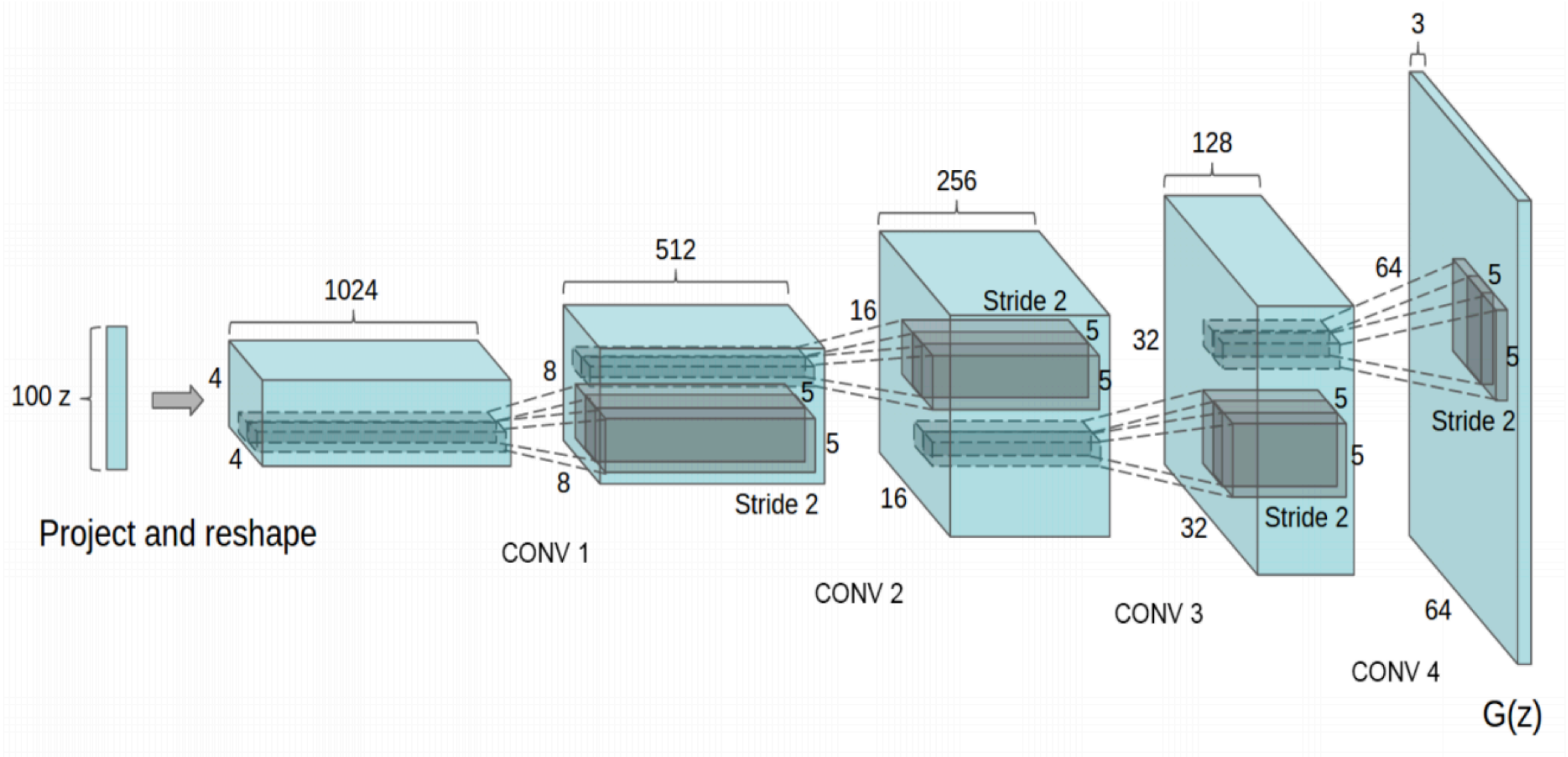
Generated Samples



Deep Convolutional GANs (DCGANs)

[Radford et al., 2016]

Generator Architecture

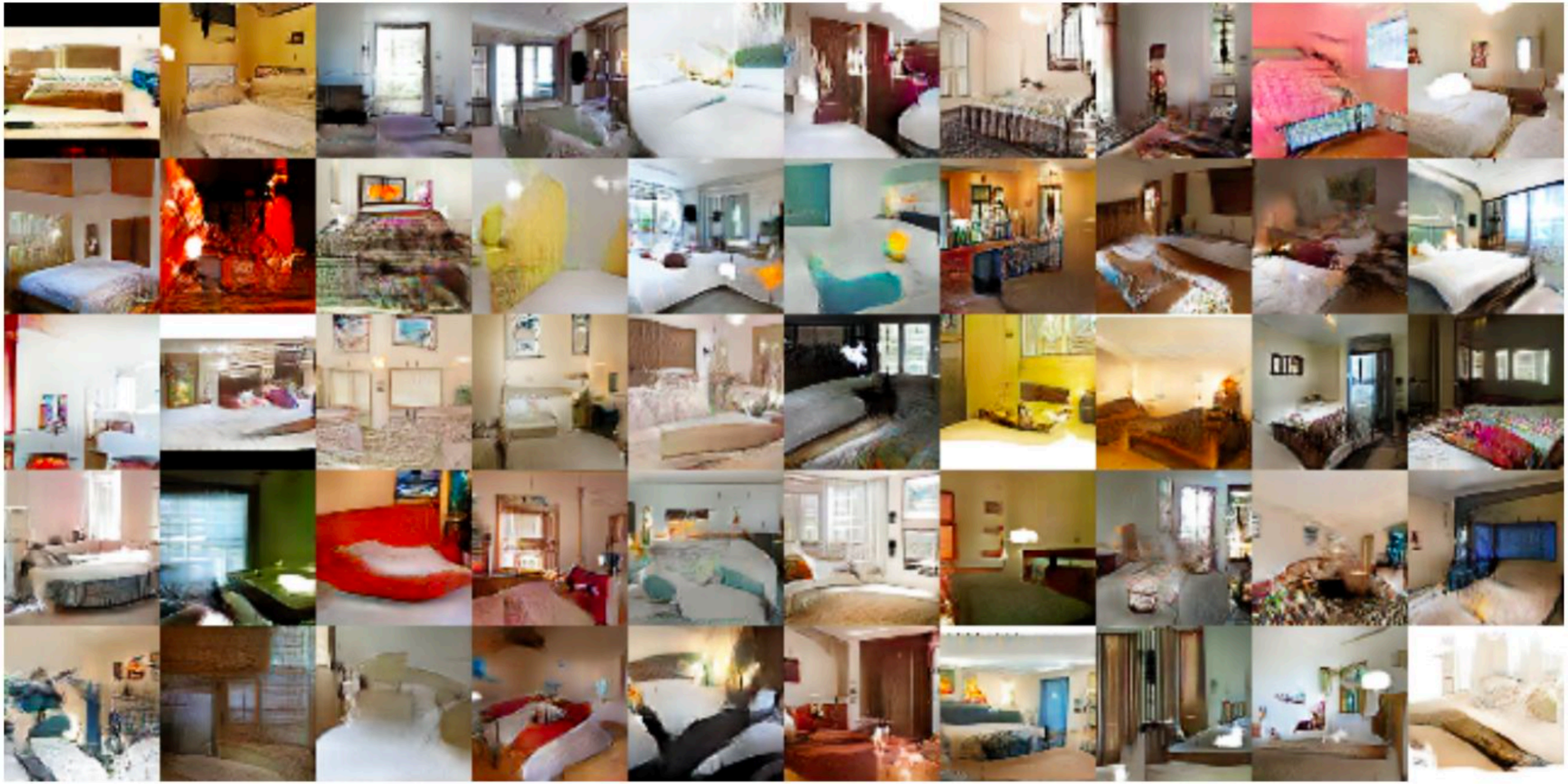


Key ideas:

- 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]

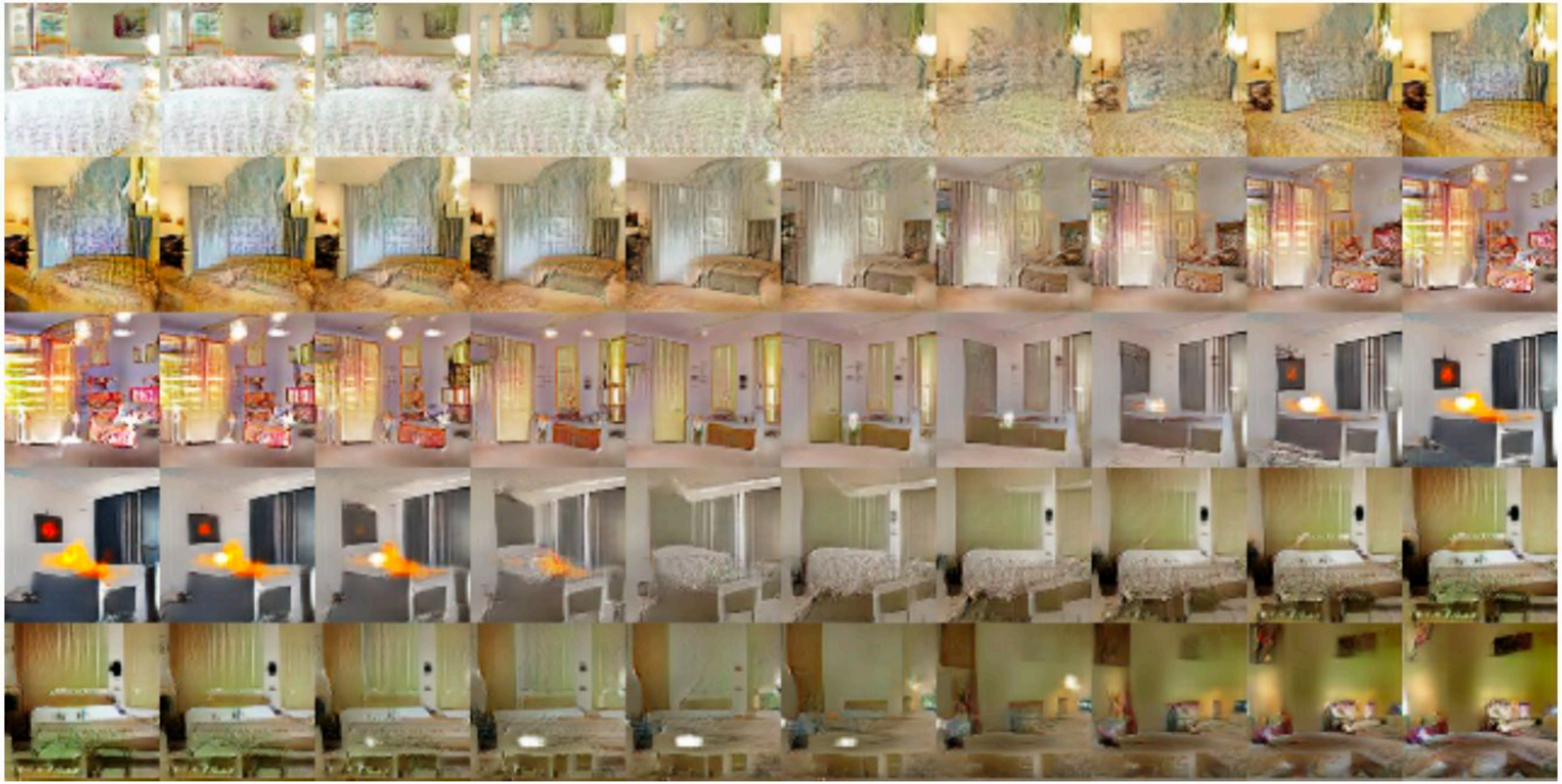


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GANs with Convolutional Architectures

[Radford et al., 2016]

Interpolating between points in latent space



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

GANs: Interpretable Vector Math

[Radford et al., 2016]

Smiling woman | Neutral womai | Neutral man

Samples from the model

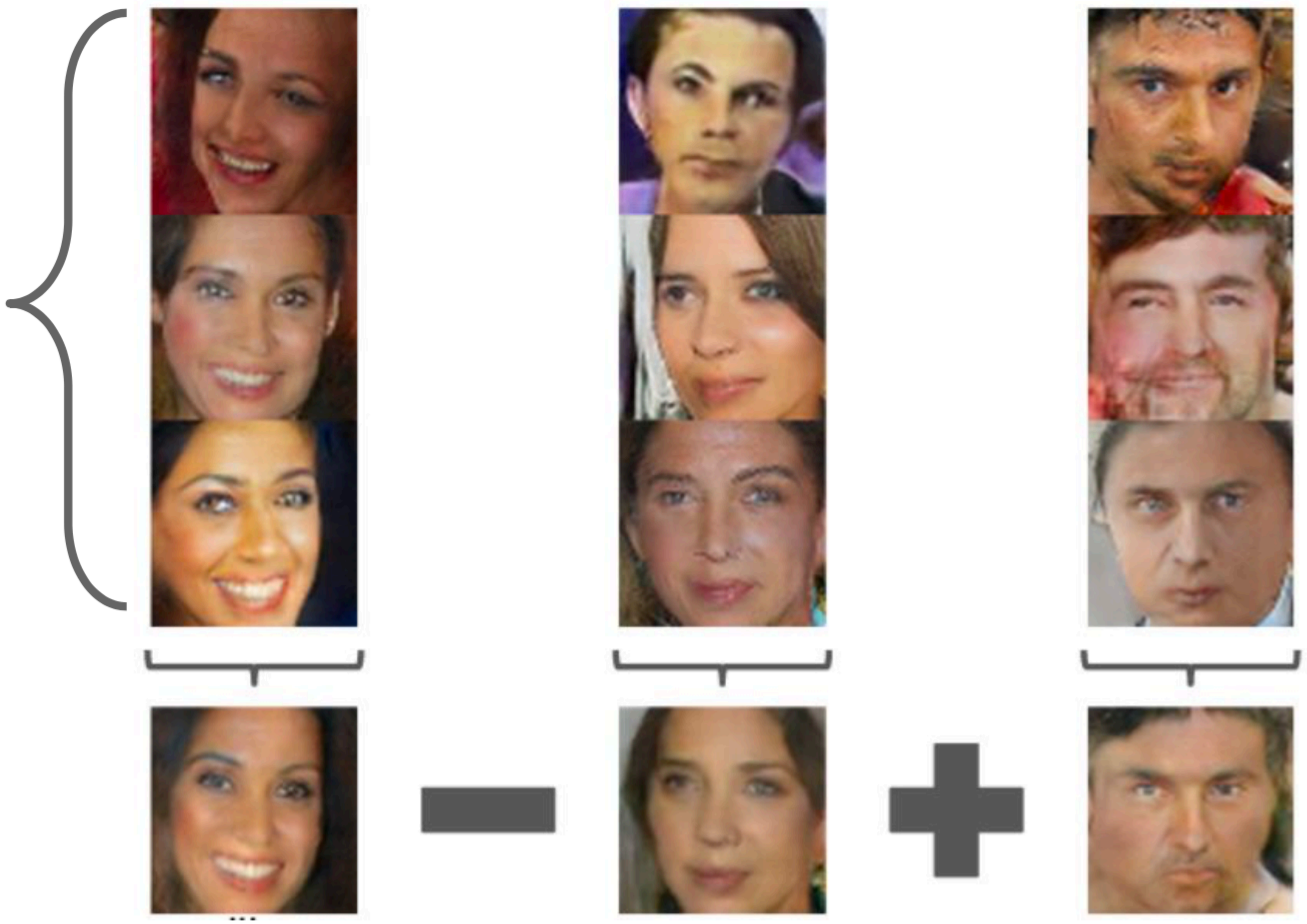
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GANs: Interpretable Vector Math

[Radford et al., 2016]

Smiling woman   Neutral woman   Neutral man

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]

Smiling woman Neutral womai Neutral man

Smiling man

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]

Glasses Man | No Glasses Man | No Glasses Woman

Samples from the model

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

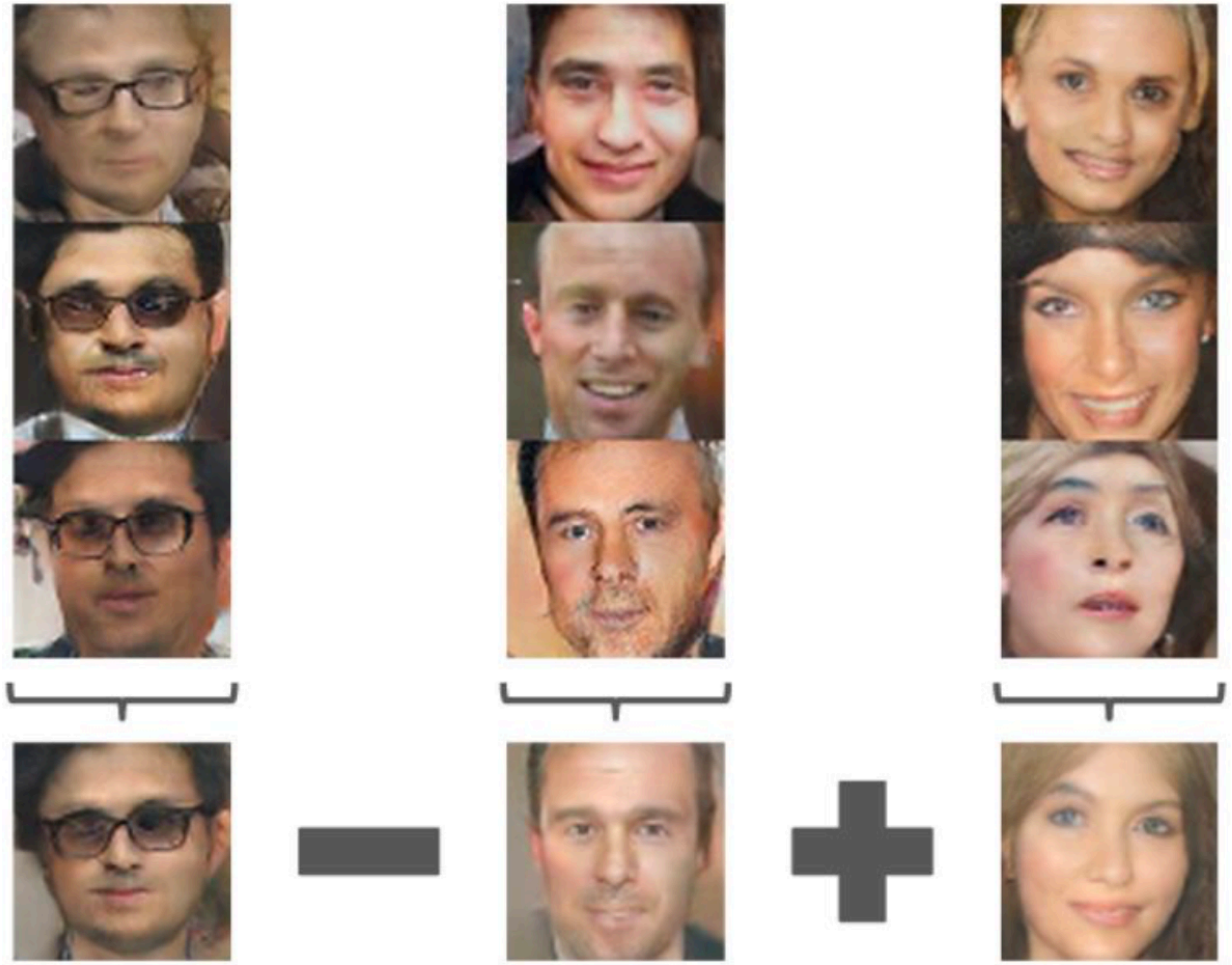
GANs: Interpretable Vector Math

[Radford et al., 2016]

Glasses Man No Glasses Man No Glasses Woman

Samples from the model

Average z vectors, do arithmetic



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

GANs: Interpretable Vector Math

[Radford et al., 2016]

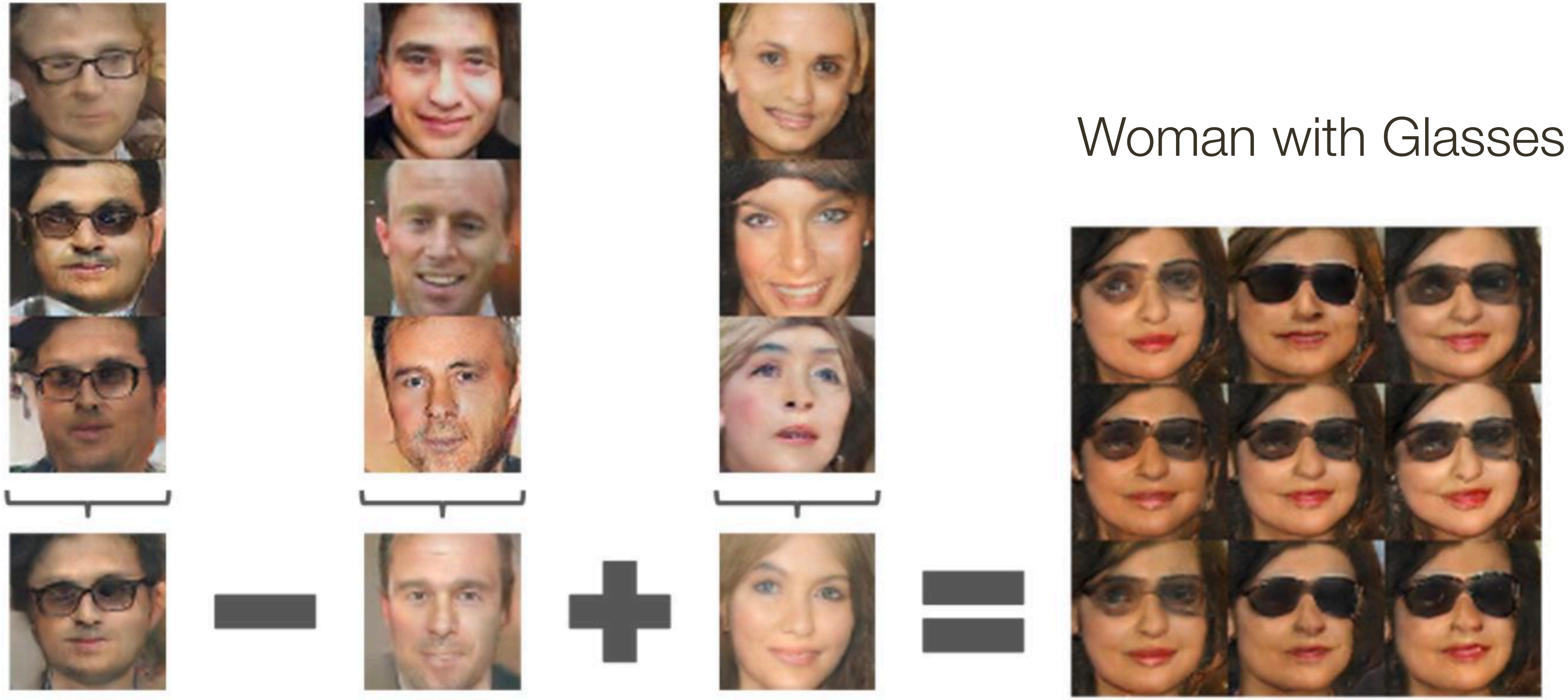
Radford et al,
ICLR 2016

Glasses Man   No Glasses Man   No Glasses Woman

Samples from the model

Average z vectors, do arithmetic

Woman with Glasses



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

Conditional GAN: Text-to-Image Synthesis

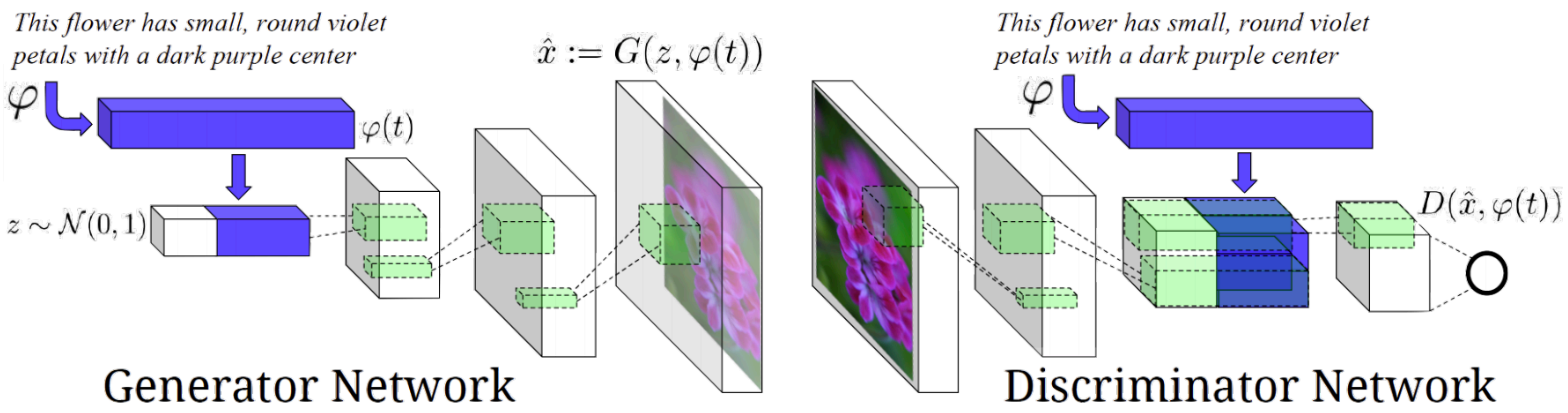


Figure 2 in the original paper.

Positive Example:

Real Image, Right Text

Negative Examples:

Real Image, Wrong Text

Fake Image, Right Text

[Reed et al., ICML 2016]

Conditional GAN: Image-to-Image translation

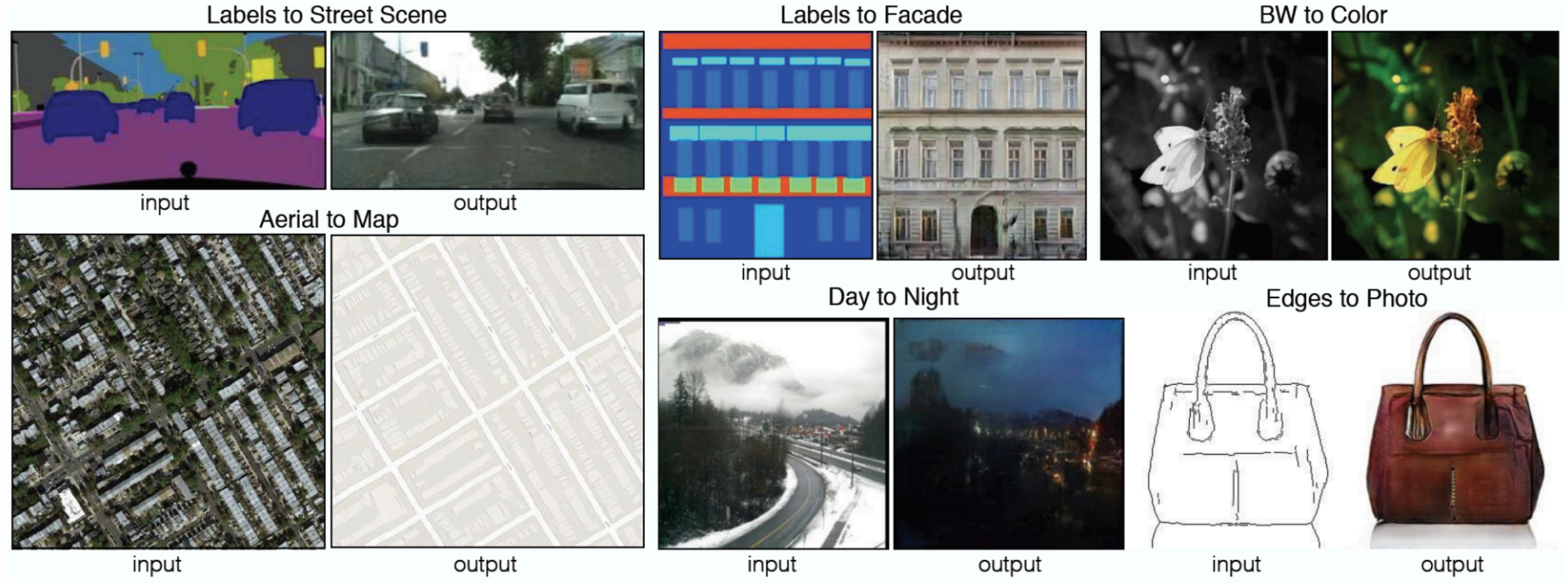


Figure 1 in the original paper.

[Isola et al., 2016]

Conditional GAN: Image-to-Image translation

Architecture: DCGAN-based

Training is conditioned on the **images from the source domain**

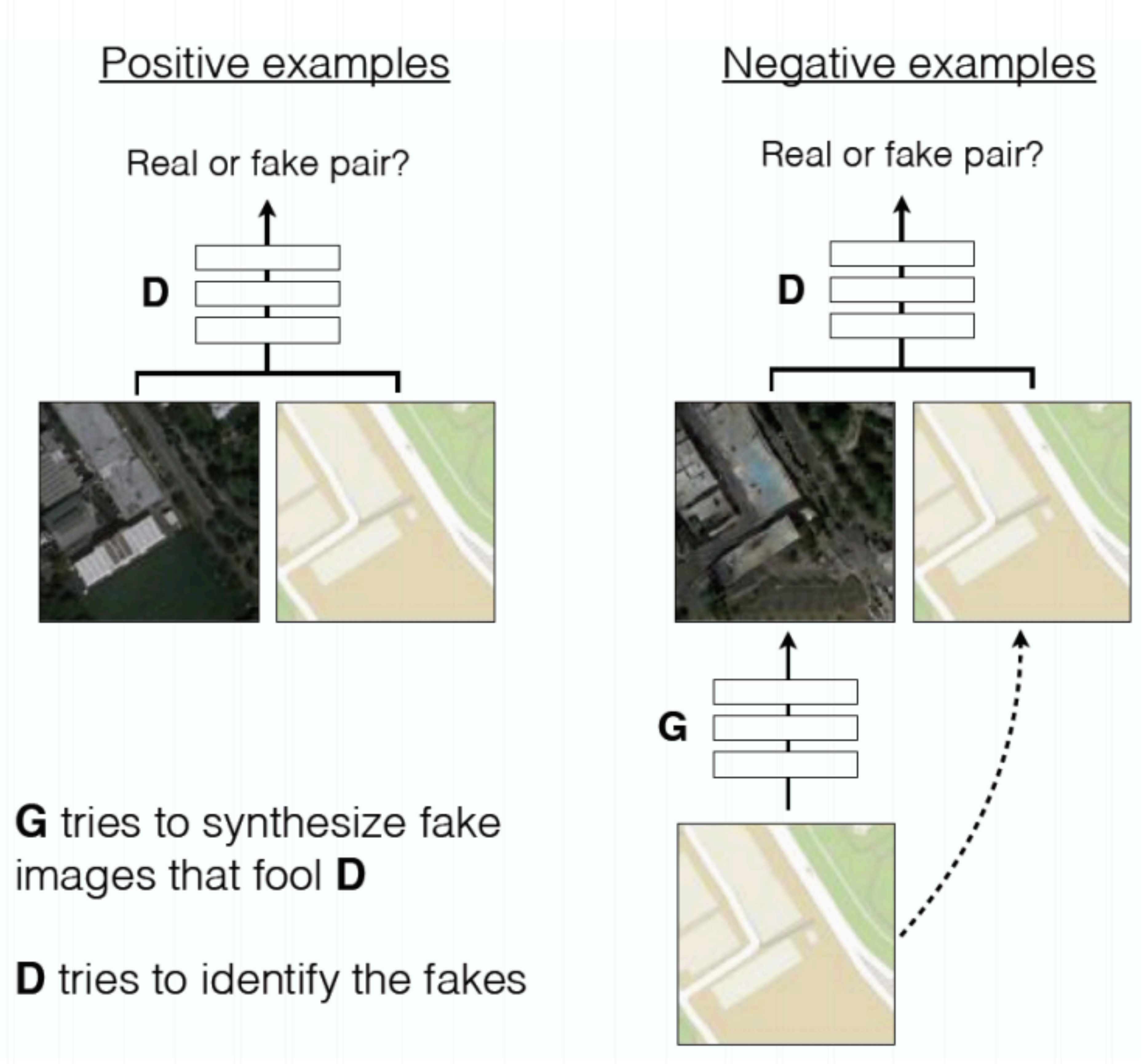
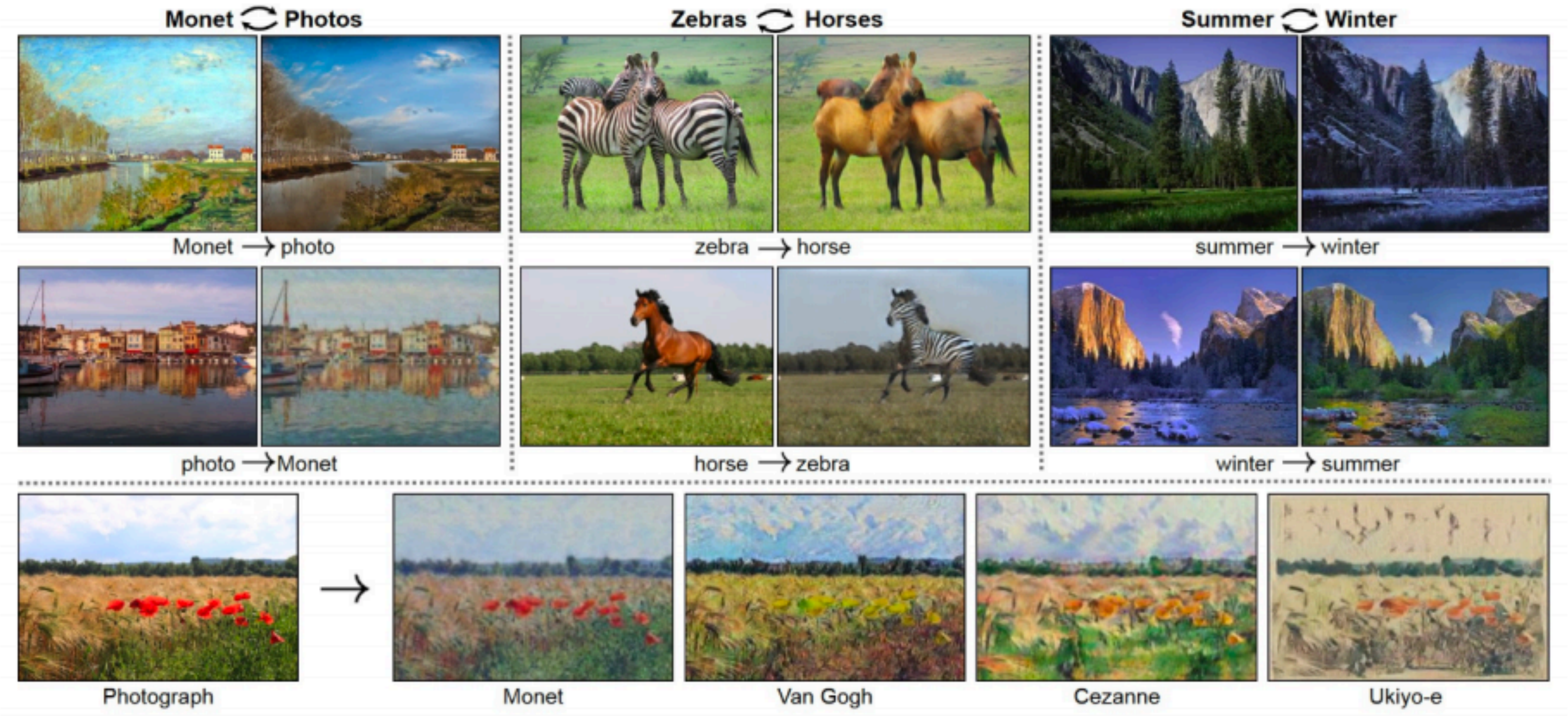


Figure 2 in the original paper.

[Isola et al., 2016]

CycleGAN: Unpaired Image-to-Image translation

Style transfer: change the style of an image while preserving the content



Data: two unrelated collections of image, one for each style

[Zhu et al., 2017]

CycleGAN: Unpaired Image-to-Image translation

Style transfer: change the style of an image while preserving the content

— Train **two different generator networks** to go from Style 1 to Style 2 and vice versa

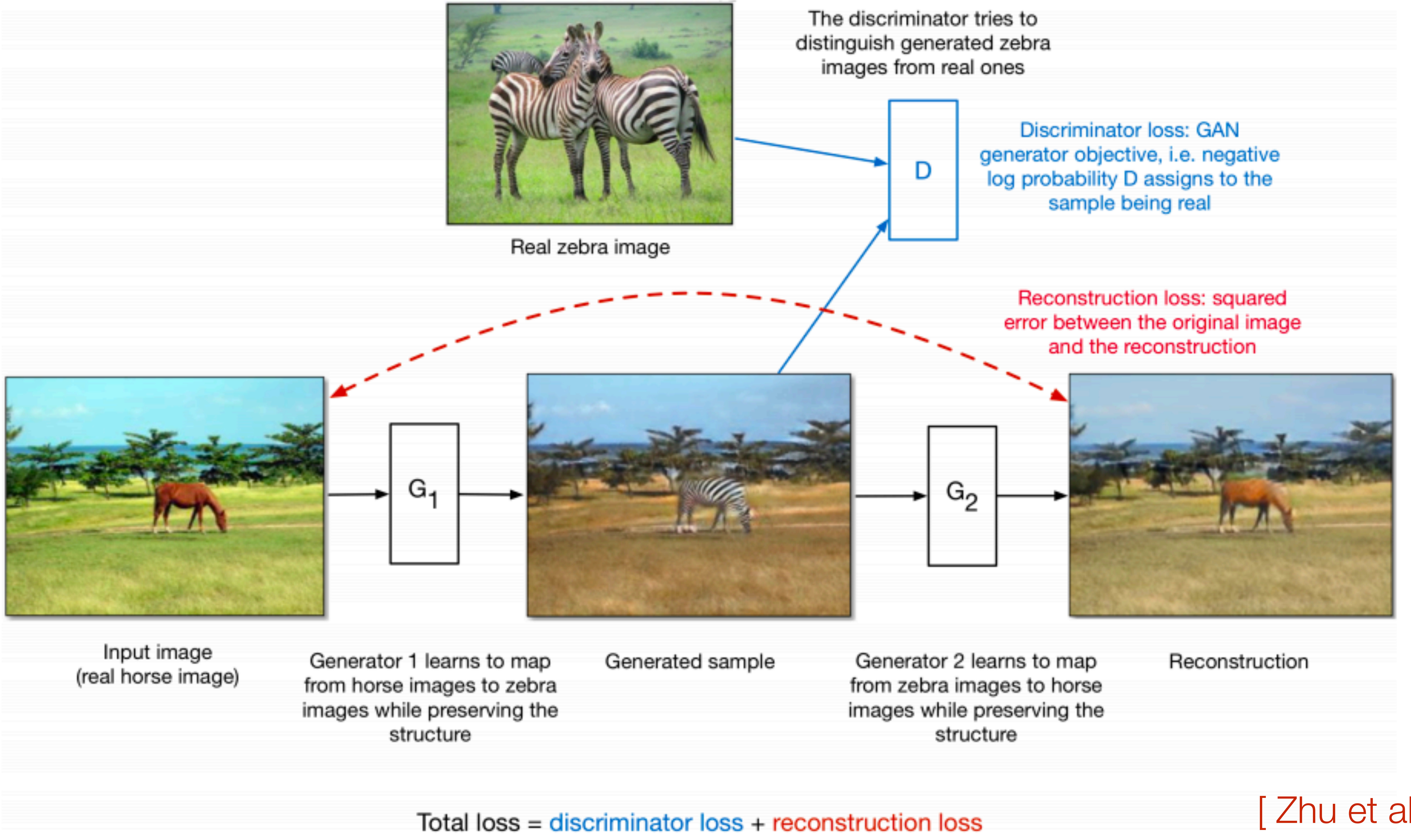
— Make sure the generated (translated) samples of Style 2 are indistinguishable from real images of Style 2 by a discriminator network

— Make sure the generated (translated) samples of Style 1 are indistinguishable from real images of Style 1 by a discriminator network

— Make sure the generators are **cycle-consistent**: mapping Style1 -> Style 2 -> Style 1 should give close to the original image

[Zhu et al., 2017]

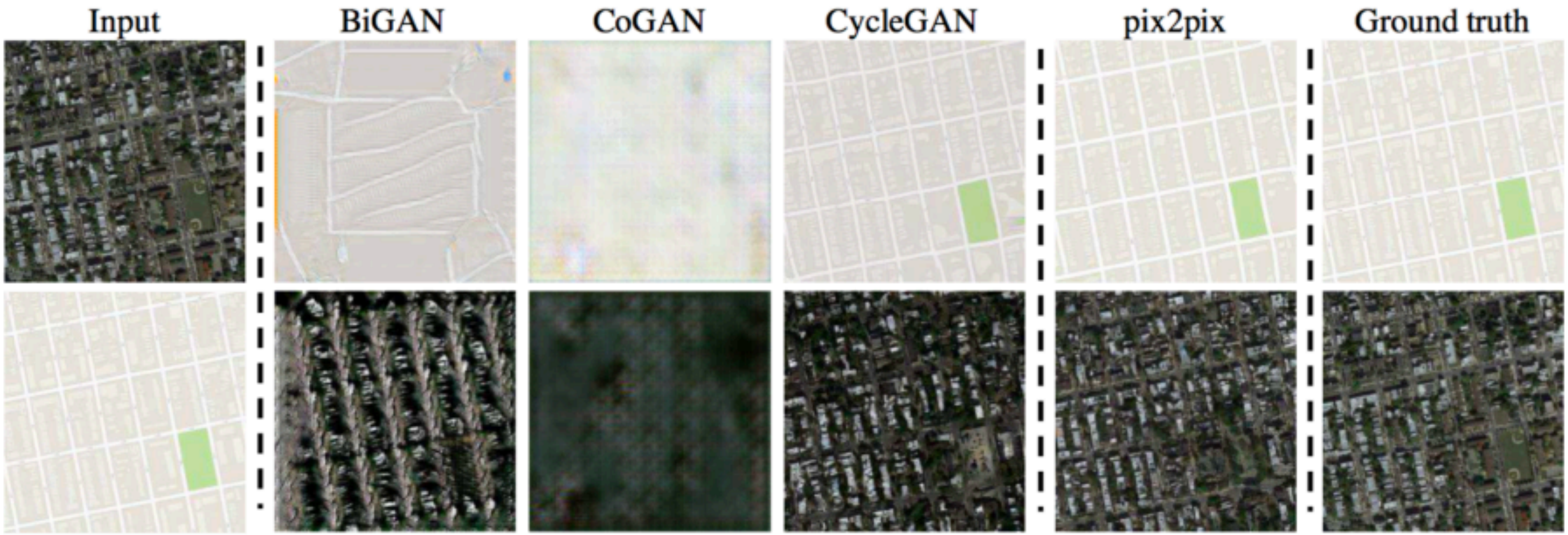
CycleGAN: Unpaired Image-to-Image translation



[Zhu et al., 2017]

CycleGAN: Unpaired Image-to-Image translation

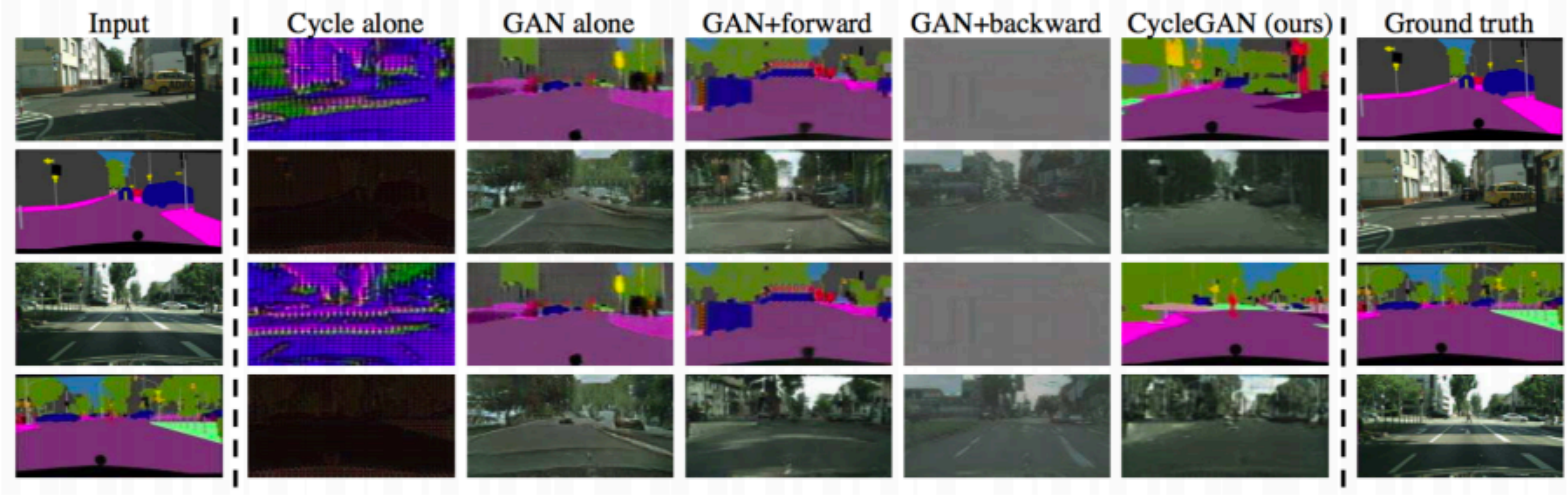
Ariel photos to **maps:**



[Zhu et al., 2017]

CycleGAN: Unpaired Image-to-Image translation

Images to **semantic segmentation:**



[Zhu et al., 2017]

Laplacian Pyramid GAN

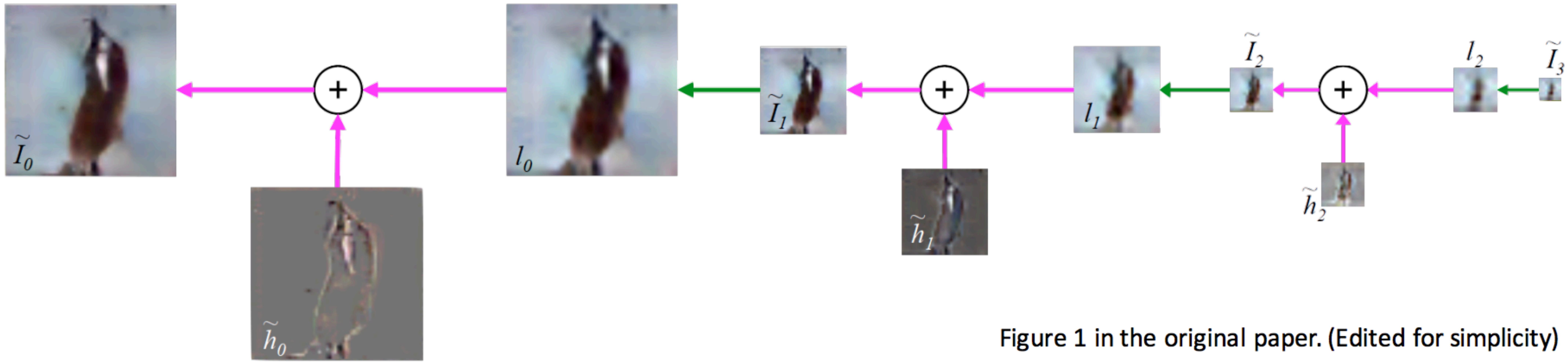


Figure 1 in the original paper. (Edited for simplicity)

— Based on the **Laplacian Pyramid** representation of images

— Generates high resolution images by using **hierarchical set of GANs** by iteratively increasing image resolution and quality

[Denton et al., 2015]

Laplacian Pyramid GAN

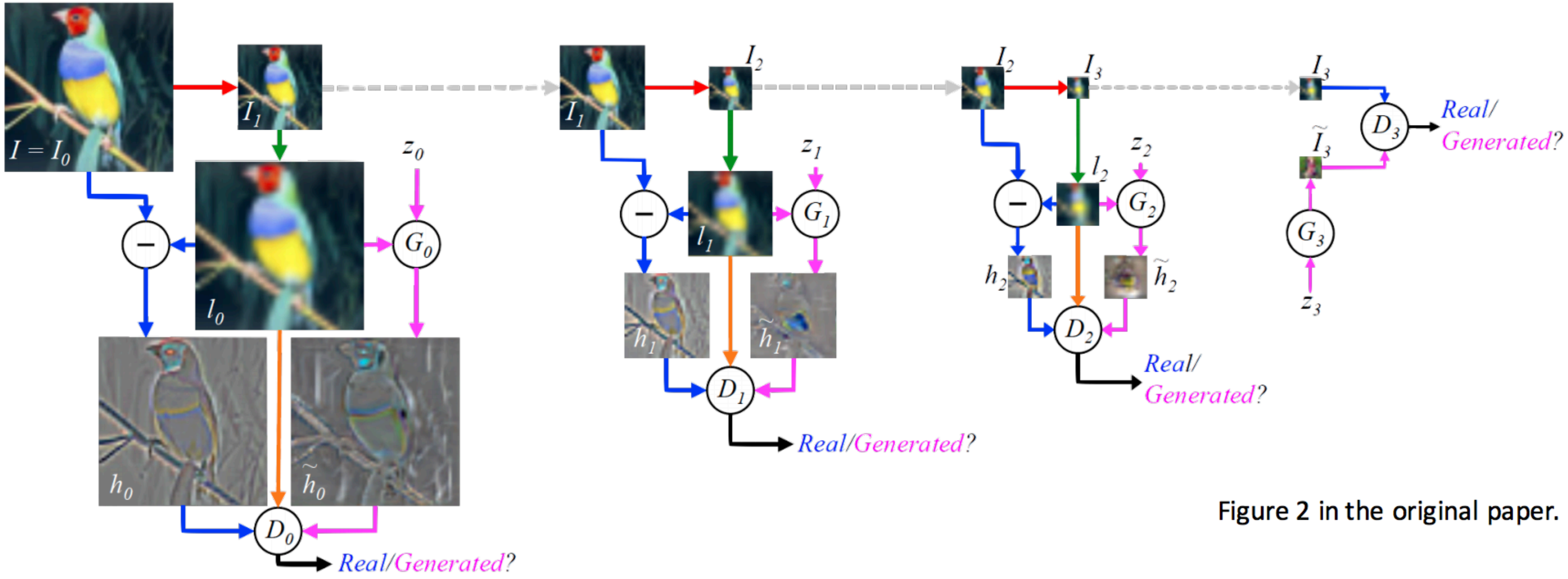


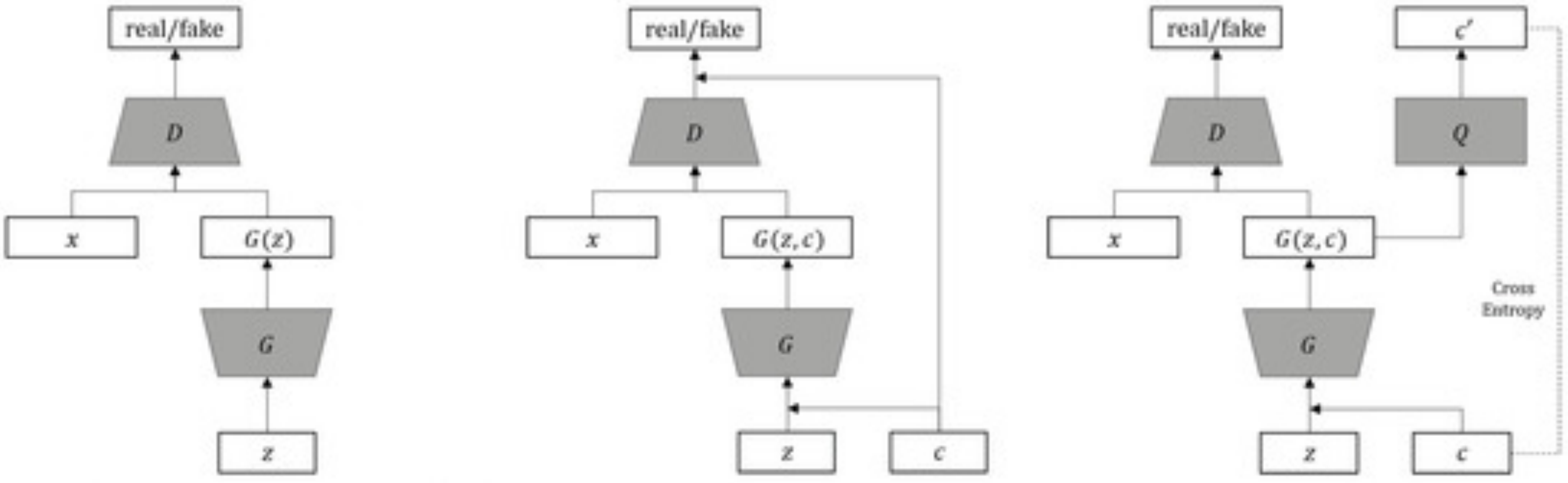
Figure 2 in the original paper.

— Based on the **Laplacian Pyramid** representation of images

— Generates high resolution images by using **hierarchical set of GANs** by iteratively increasing image resolution and quality

[Denton et al., 2015]

InfoGAN

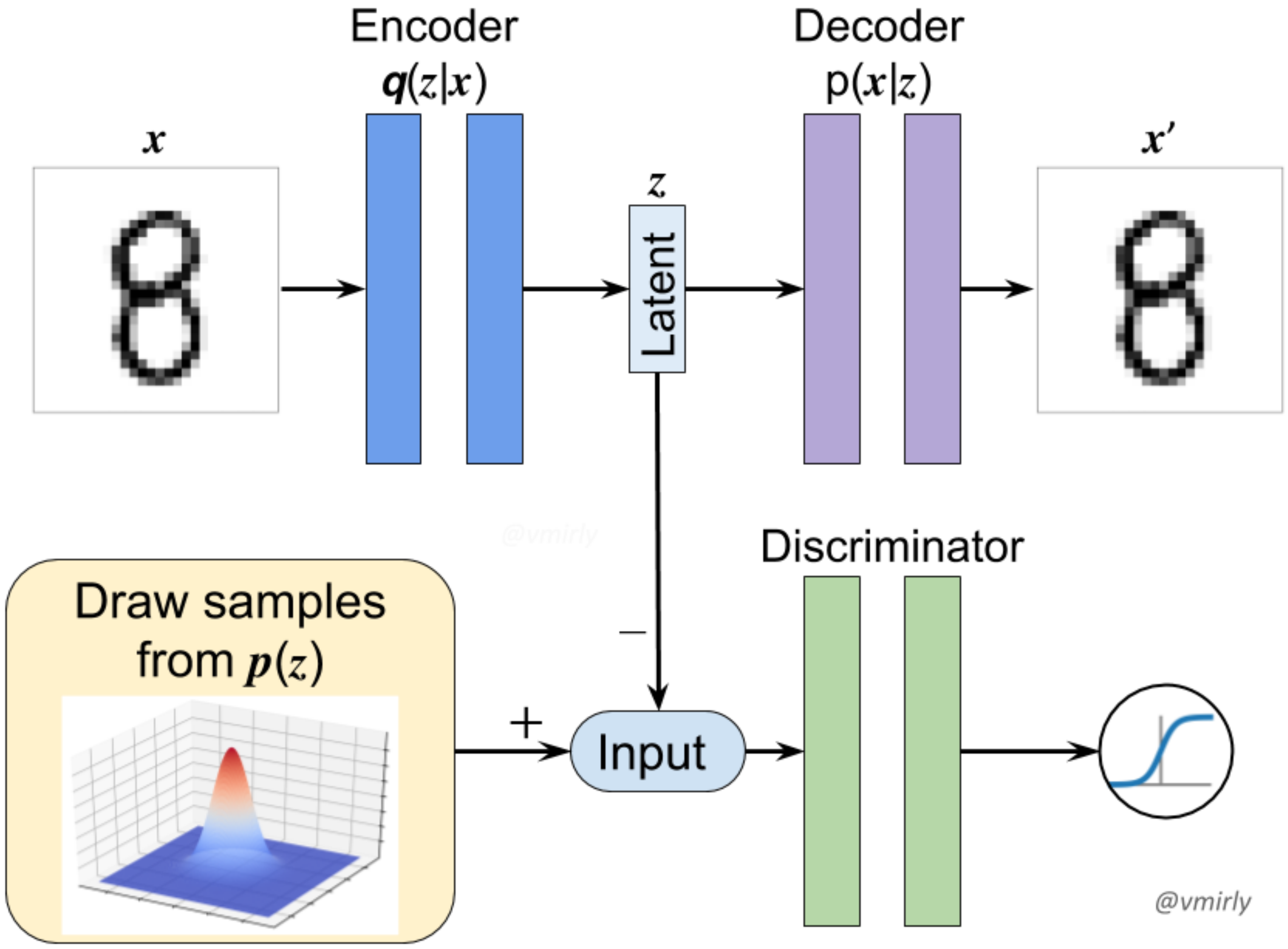


(a) GAN, DCGAN, LSGAN, WGAN (b) CGAN (c) InfoGAN

Maximizes **mutual information** between **latent code** and the **generated sample**

[Chen et al., 2016]

Adversarial Autoencoder (GAN + VAE)



[Makhzani et al., 2015]

LONG BEACH
CALIFORNIA
June 16-20, 2019

Image Generation from Layout

Bo Zhao

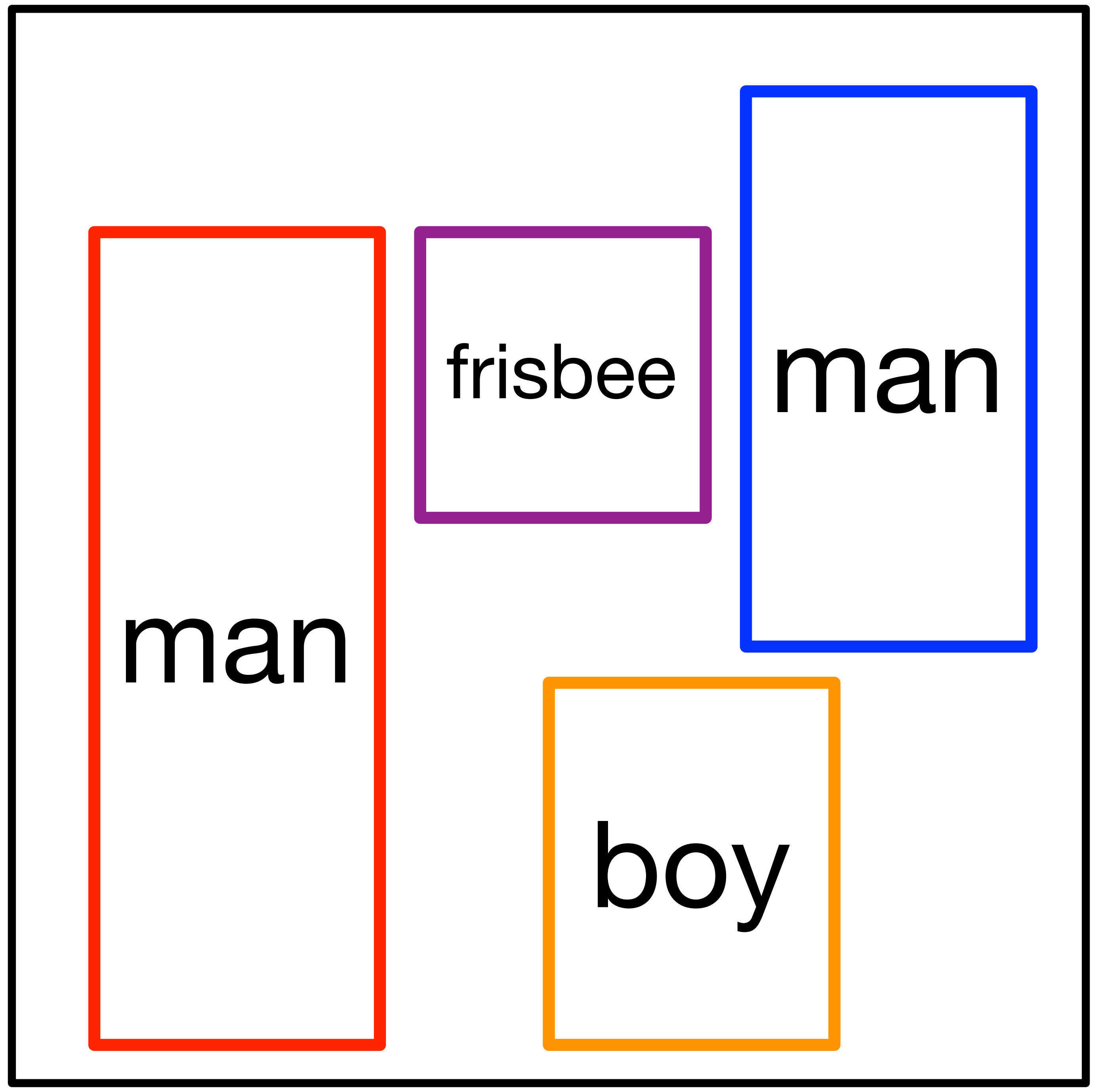
Lili Meng

Weidong Yin

Leonid Sigal

Image Generation from Layout

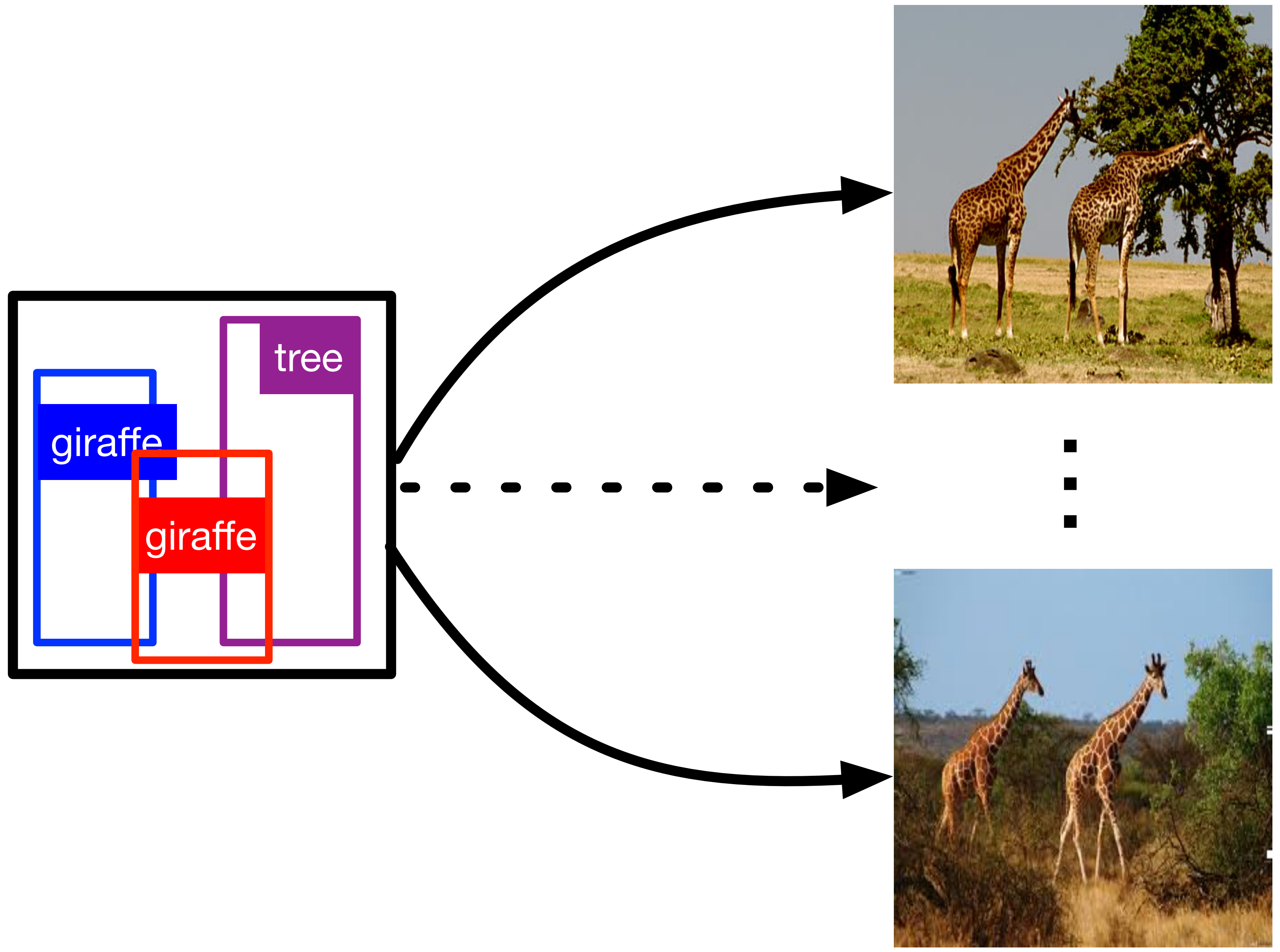
Image Generation from Layout



Layout

Results

Image Generation from Layout: **Challenges**



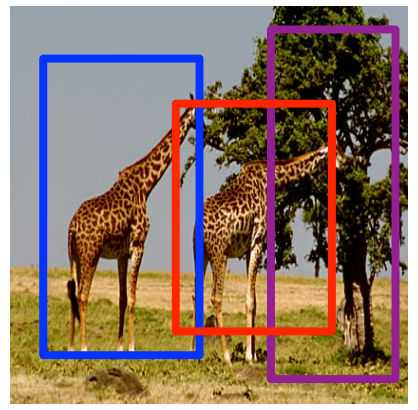
— One-to-many mapping

— Information in layout is limited (but important)

— Important interactions between objects in overlap regions and with scene

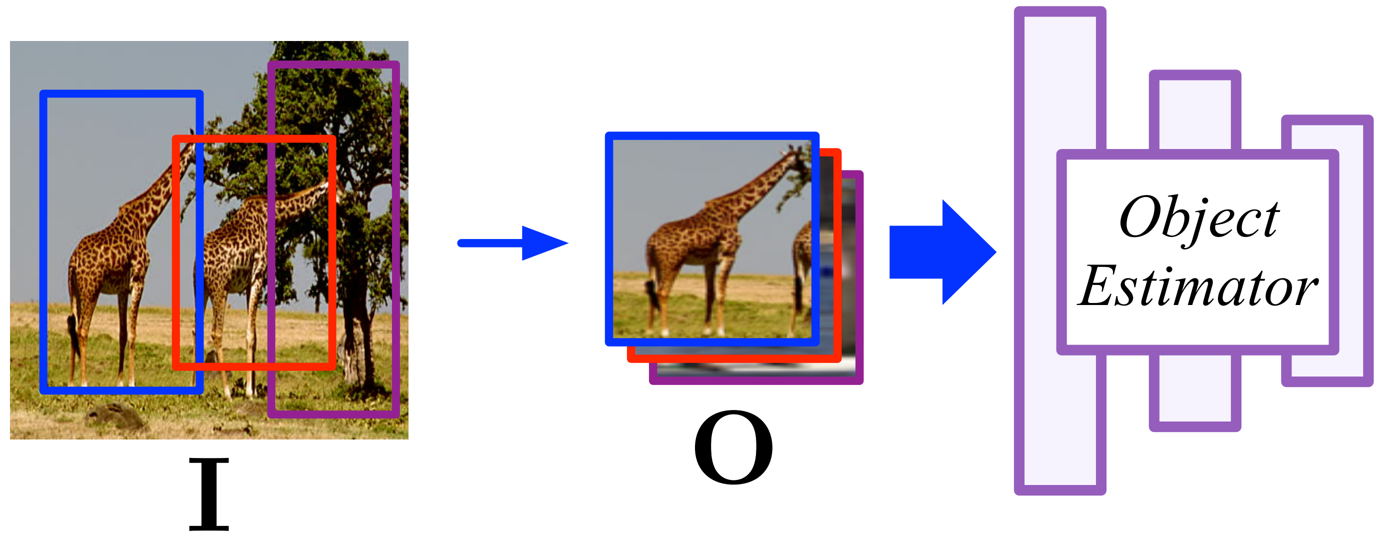
Model **Architecture**: Training

Model **Architecture**: Training

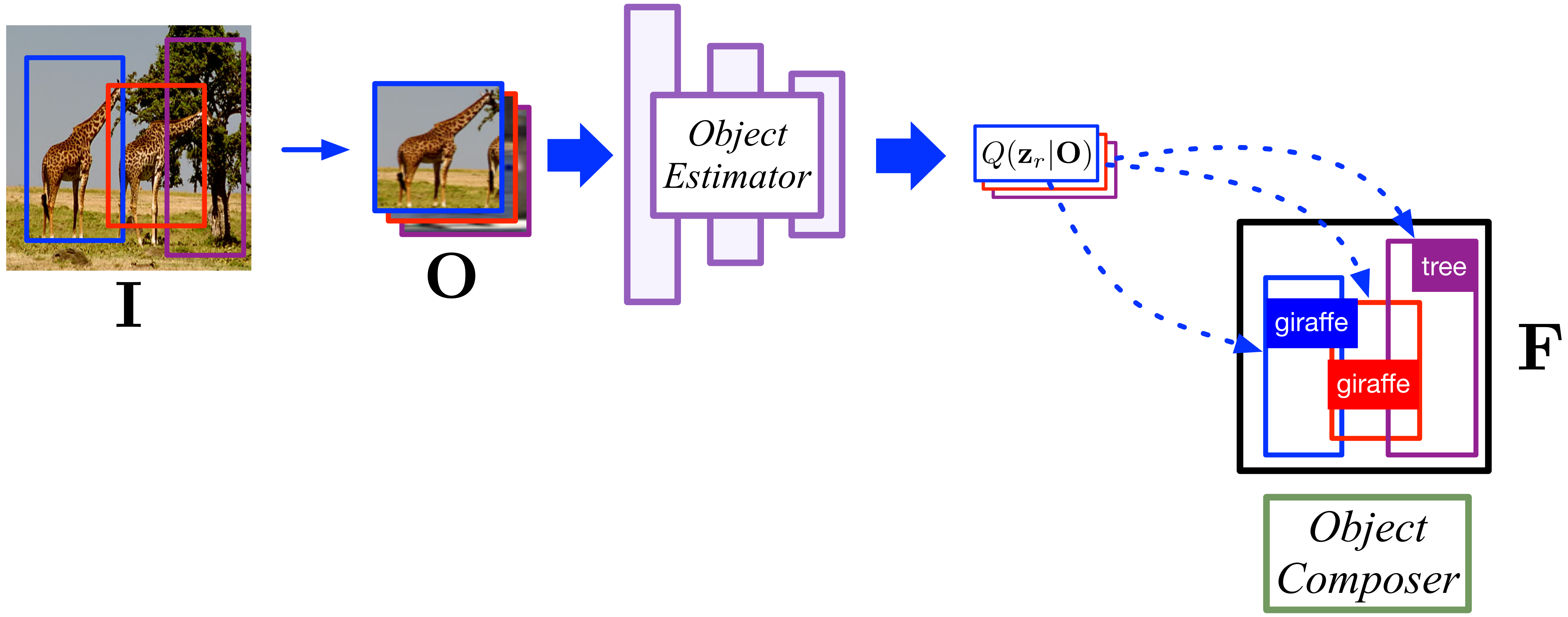


$\mathbf{I}$

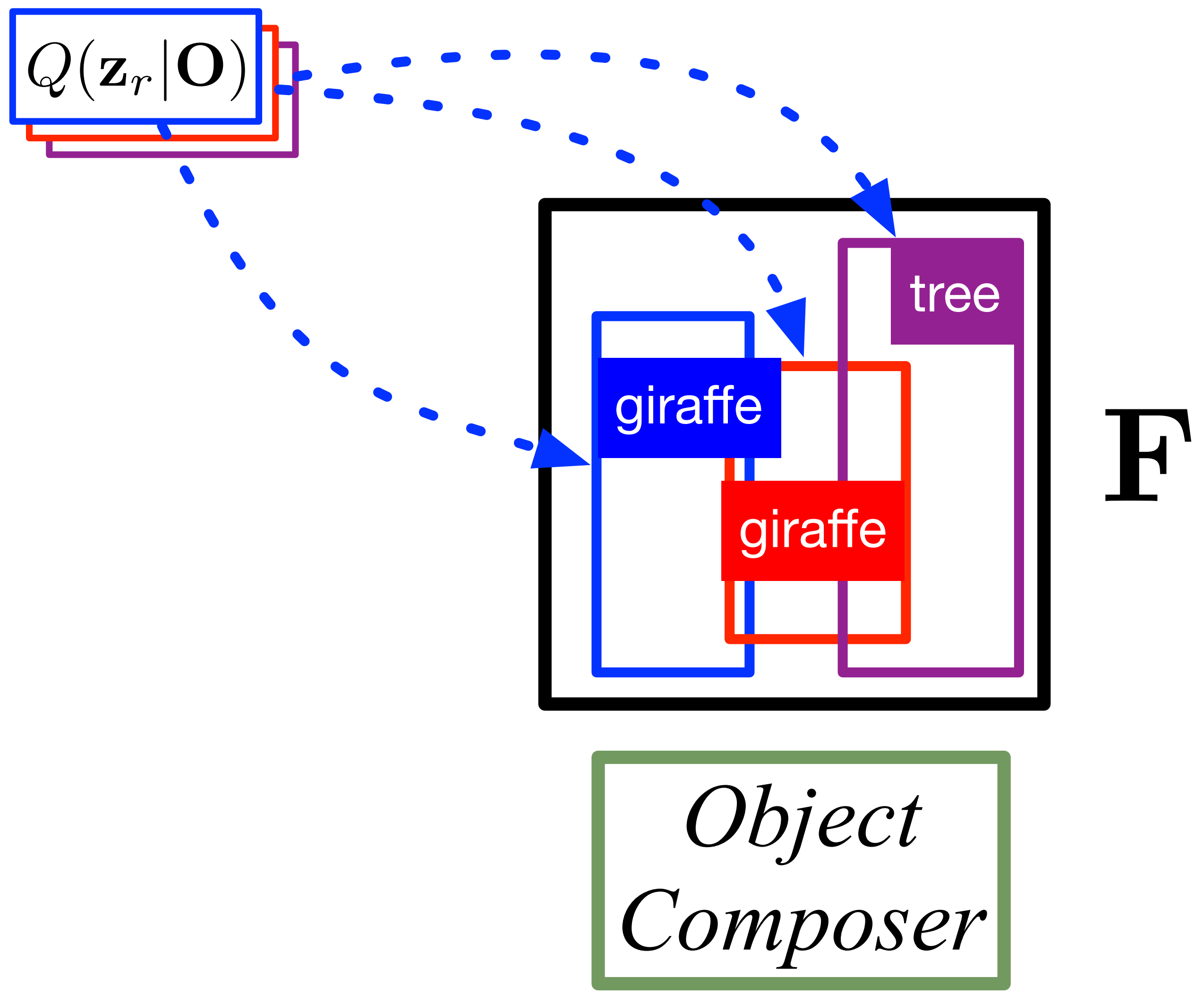
Model **Architecture**: Training



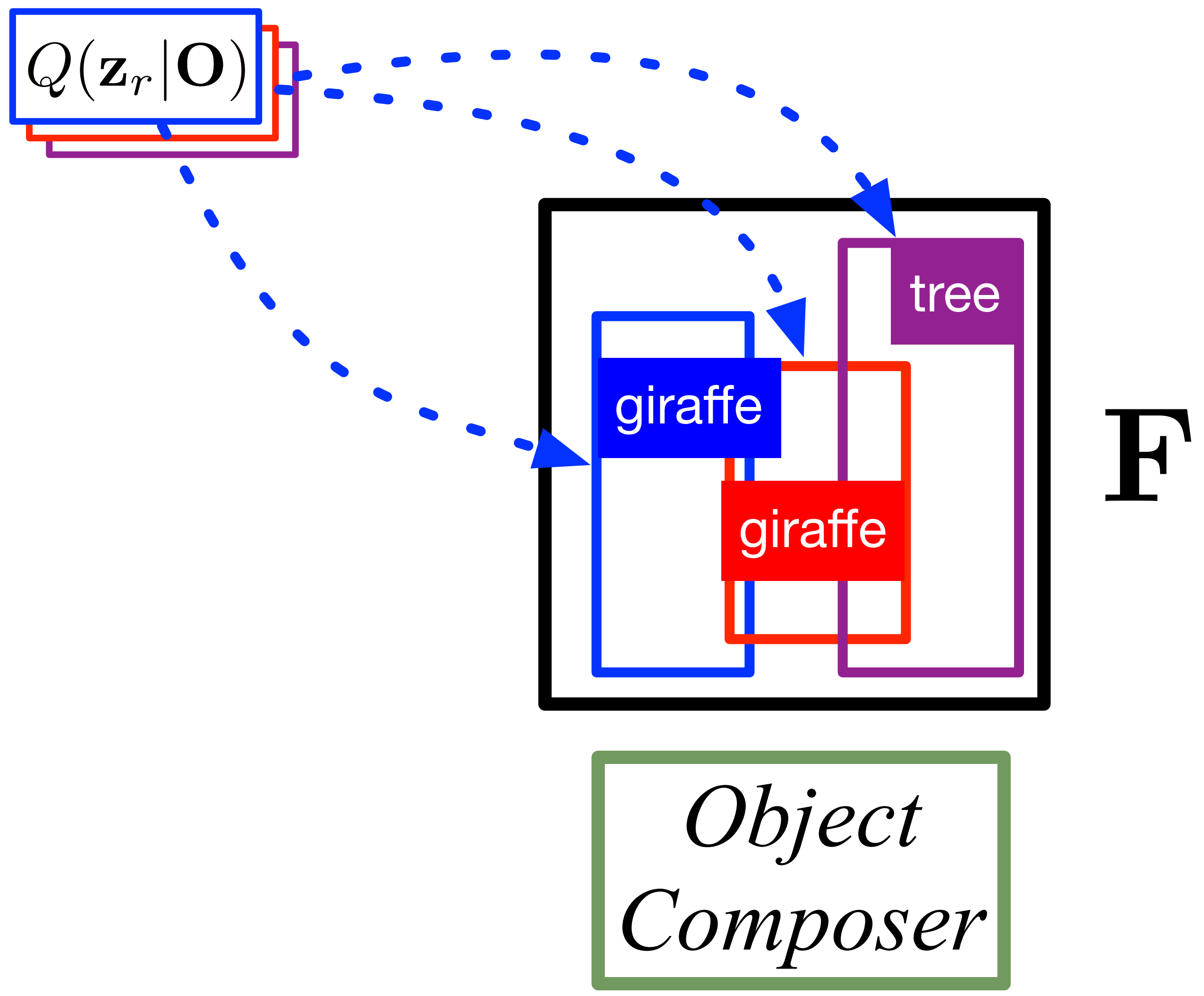
Model **Architecture**: Training

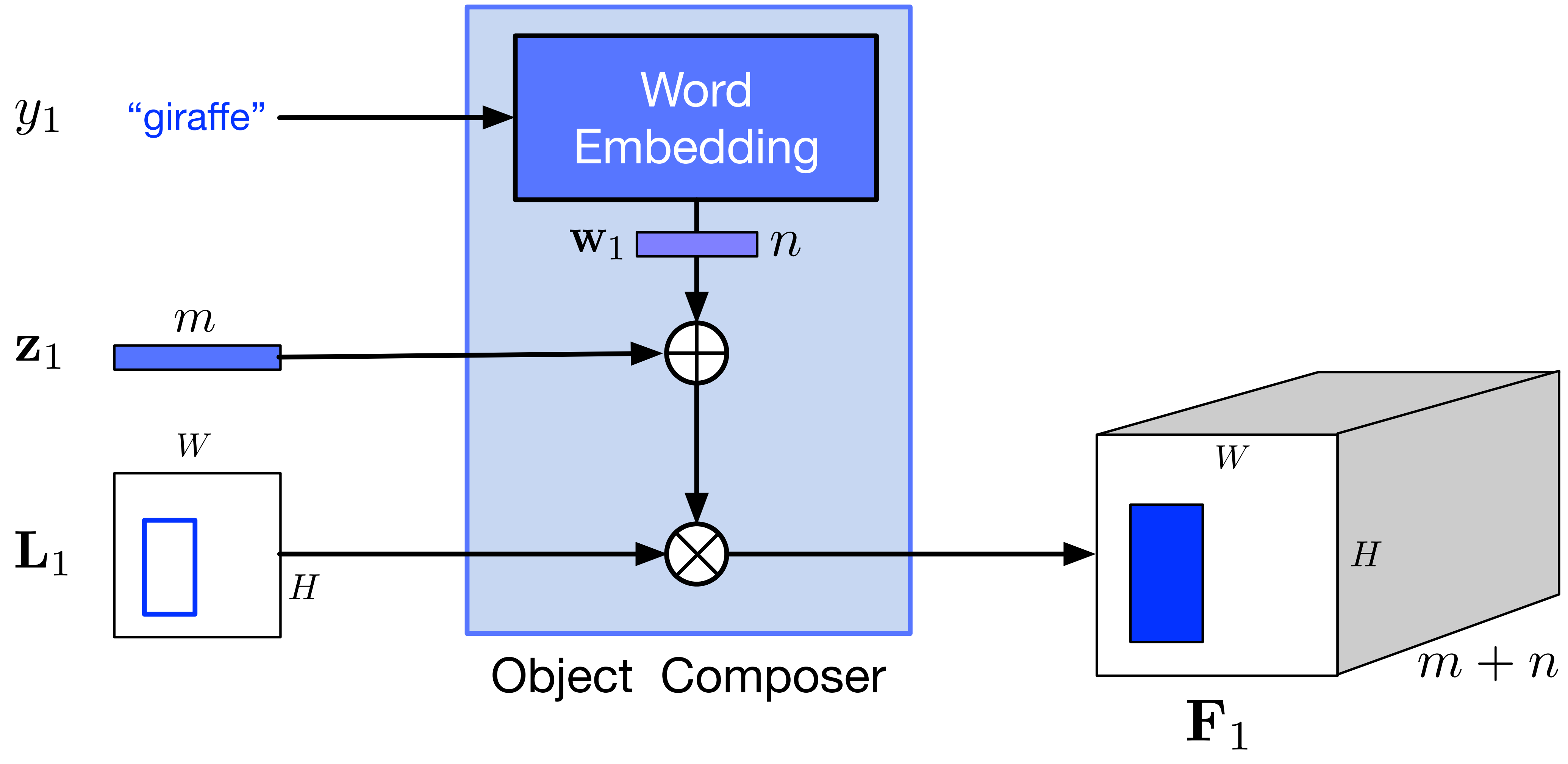


Model **Architecture**: Training

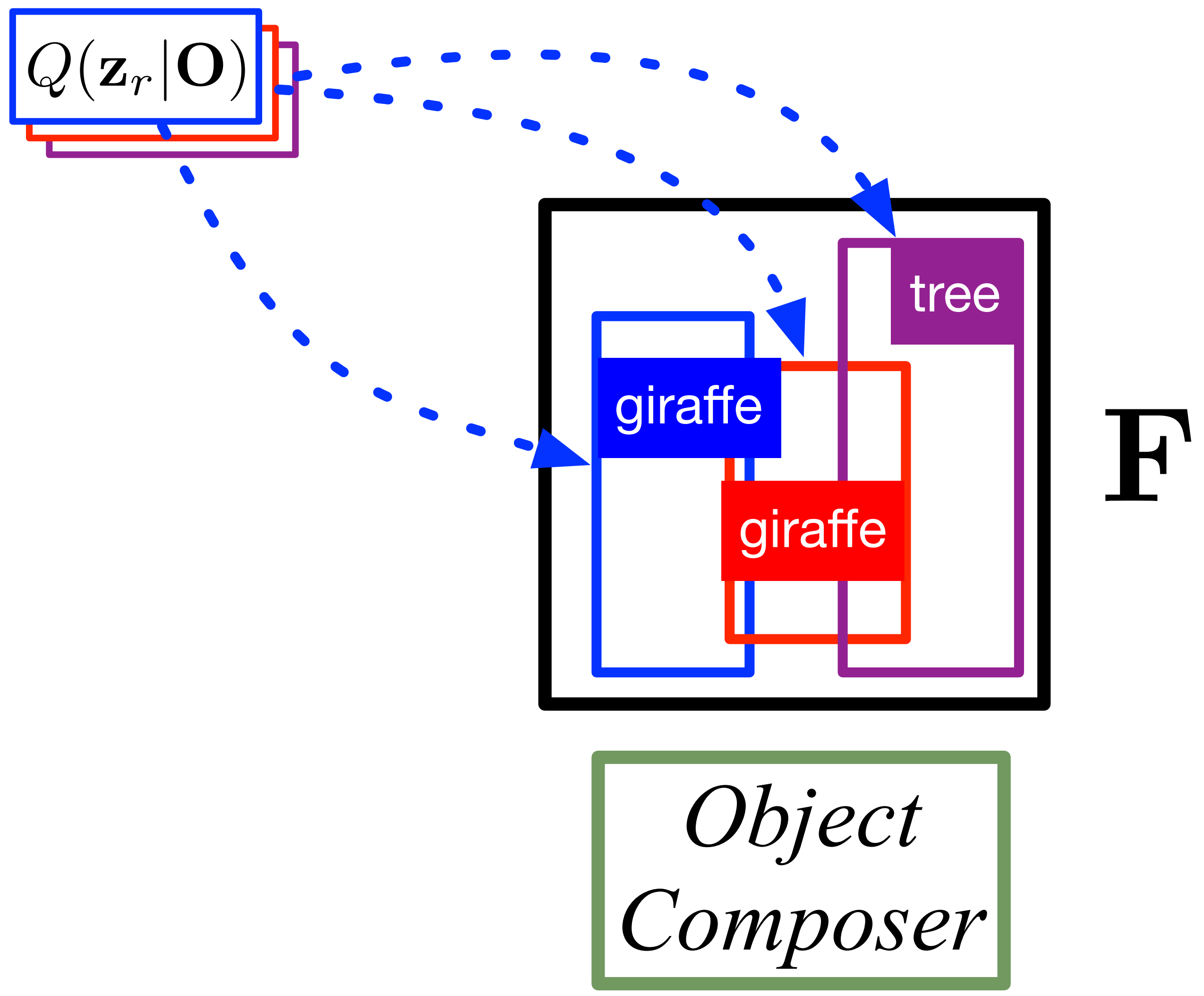


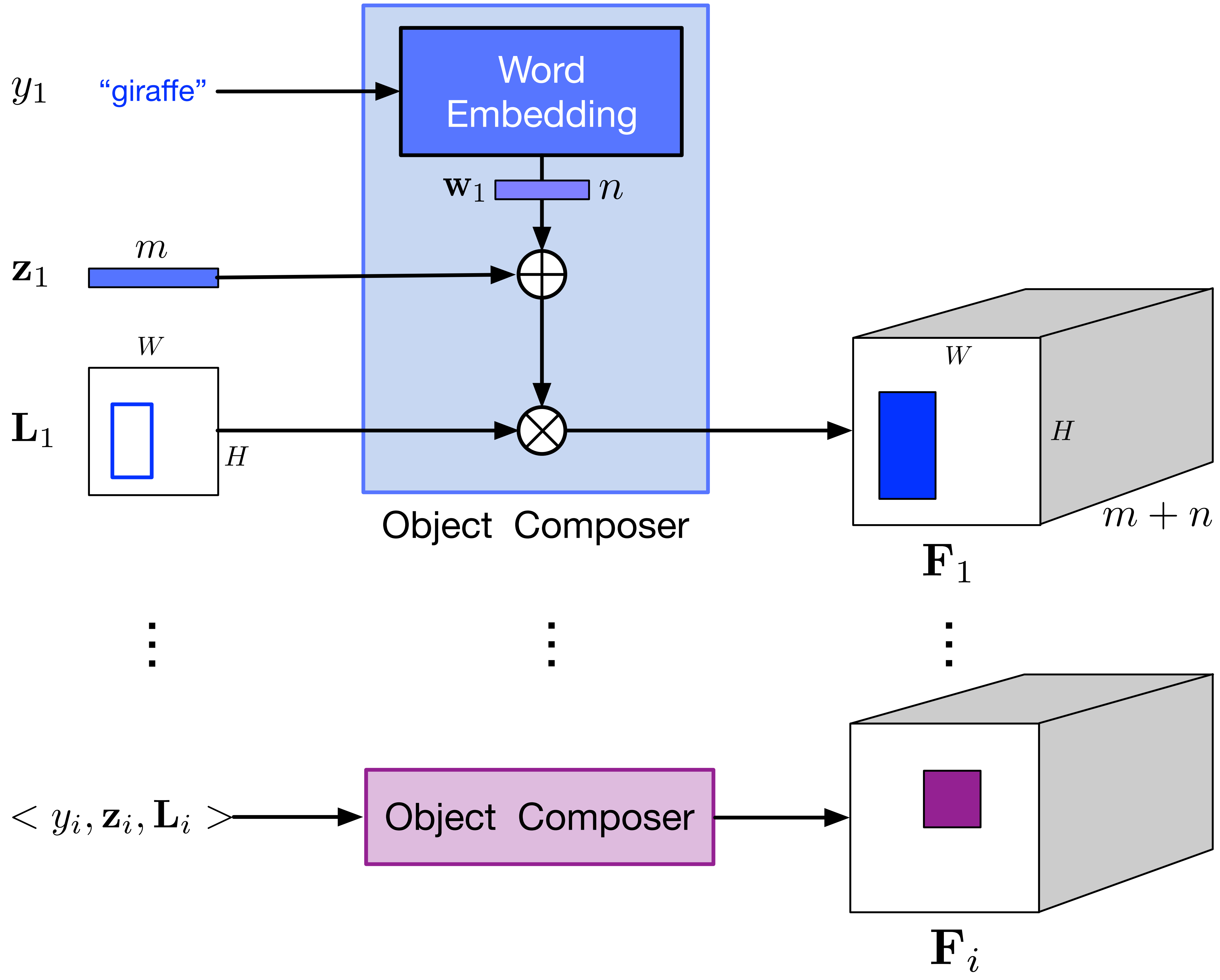
Model **Architecture**: Training



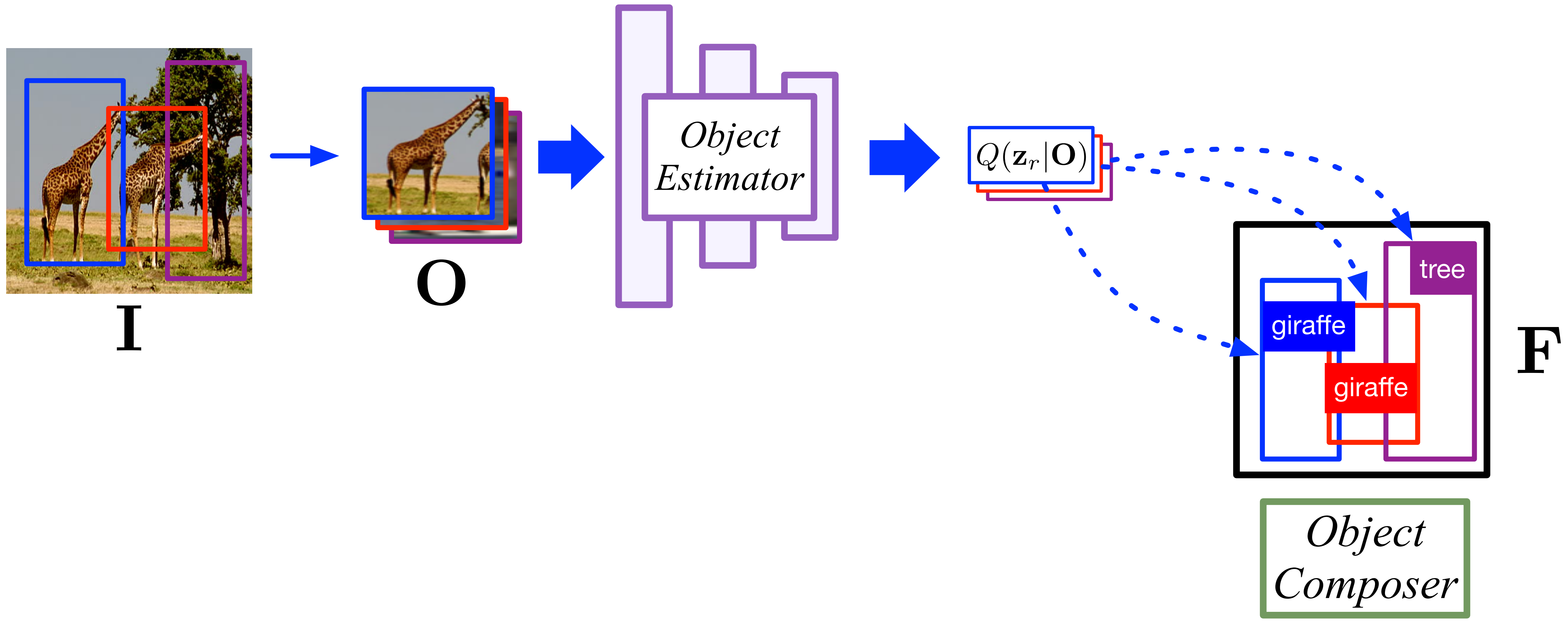


Model **Architecture**: Training

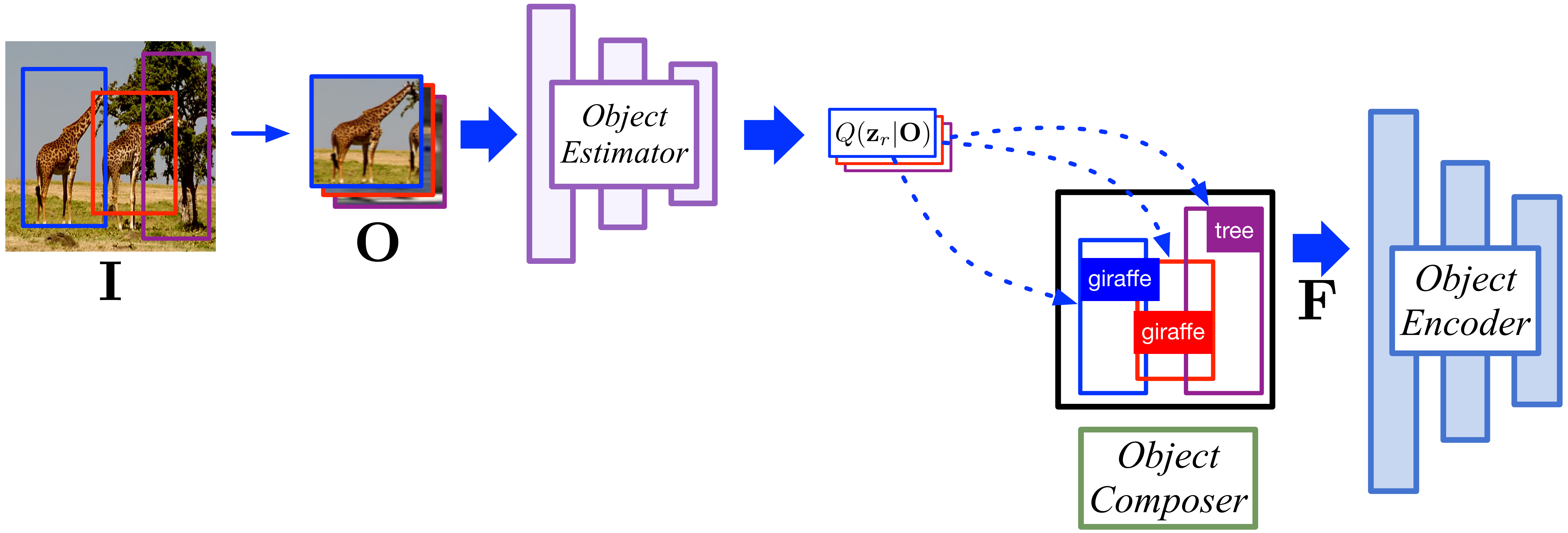




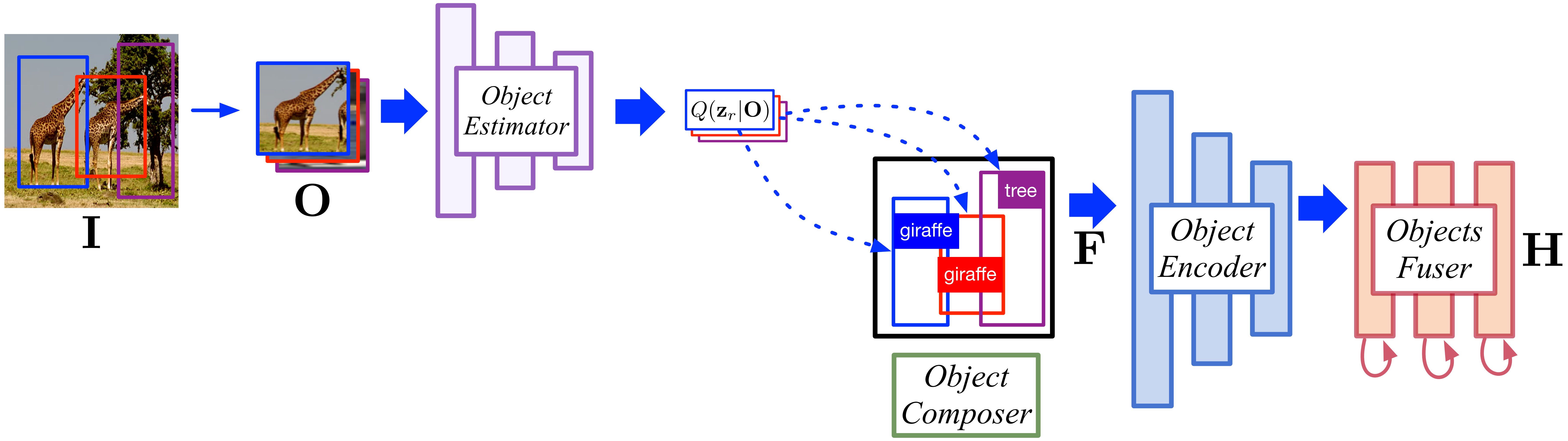
Model **Architecture**: Training



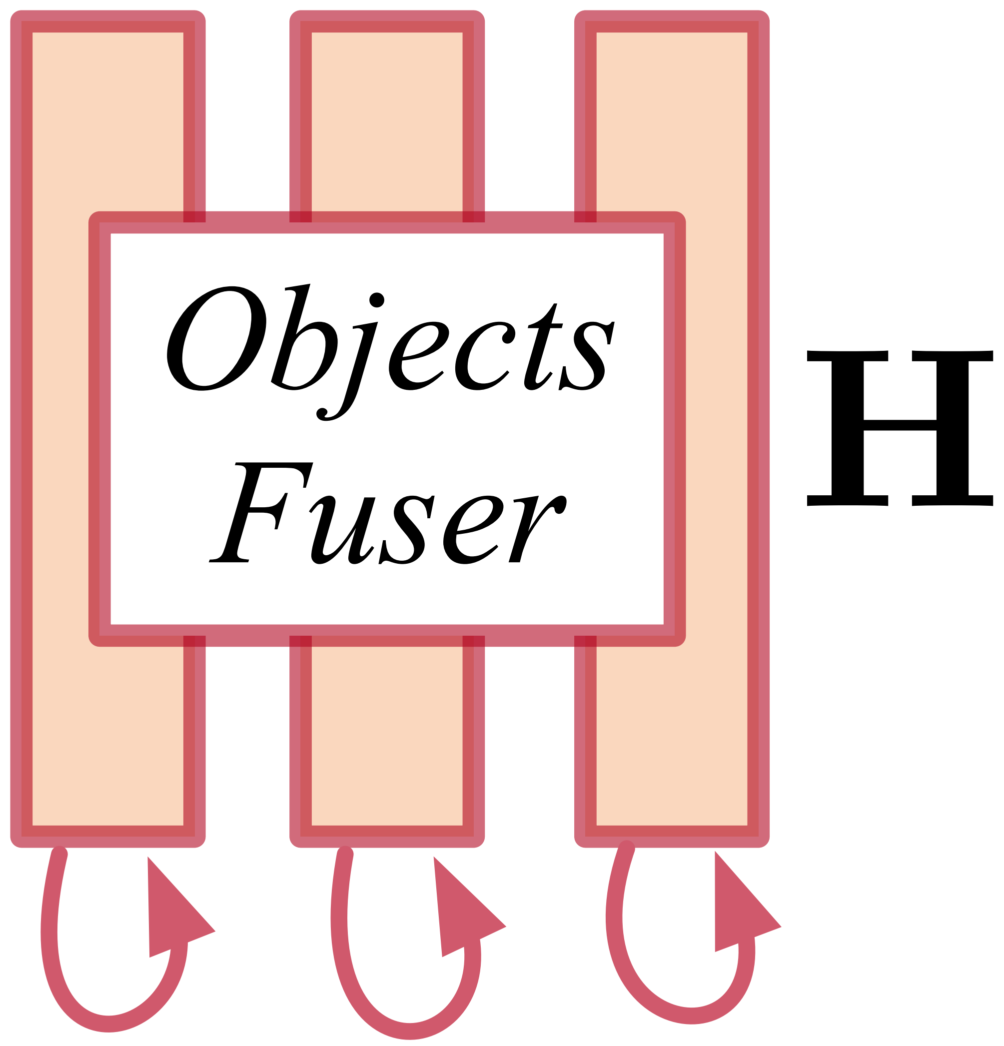
Model **Architecture**: Training

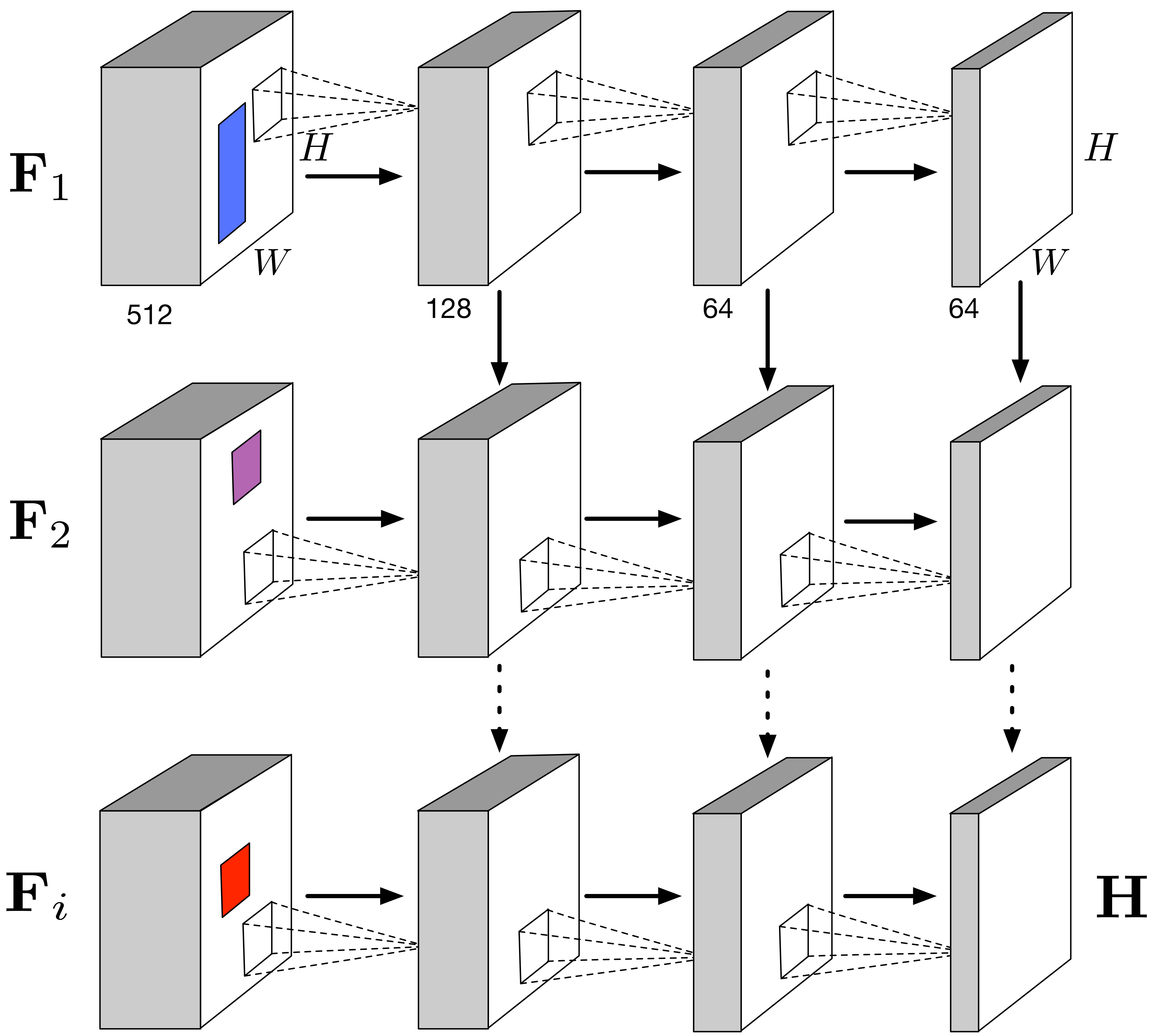


Model **Architecture**: Training

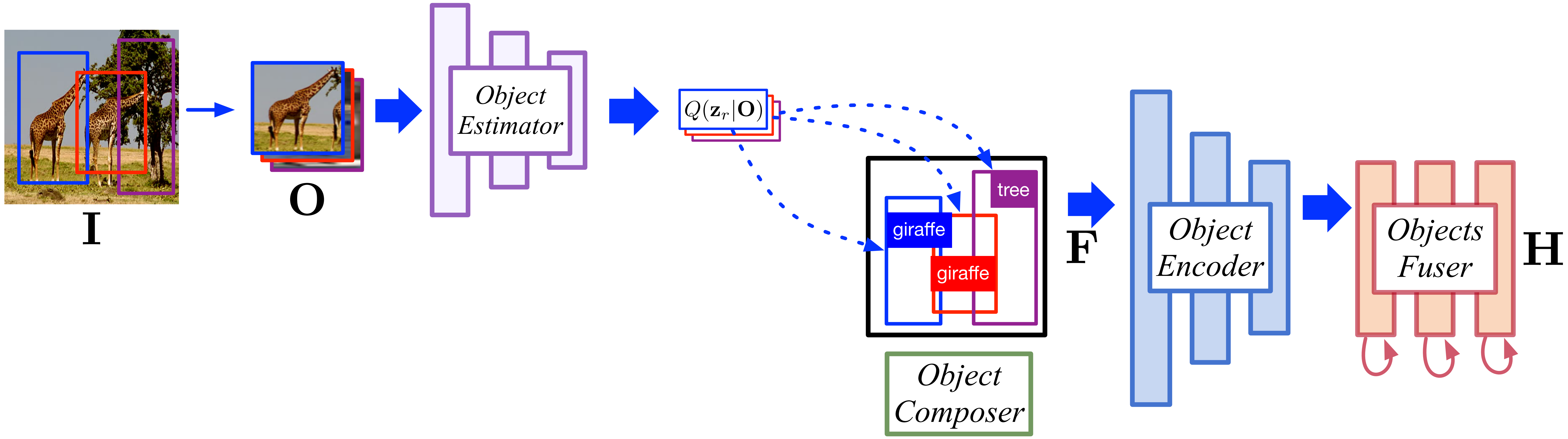


Model **Architecture**: Training

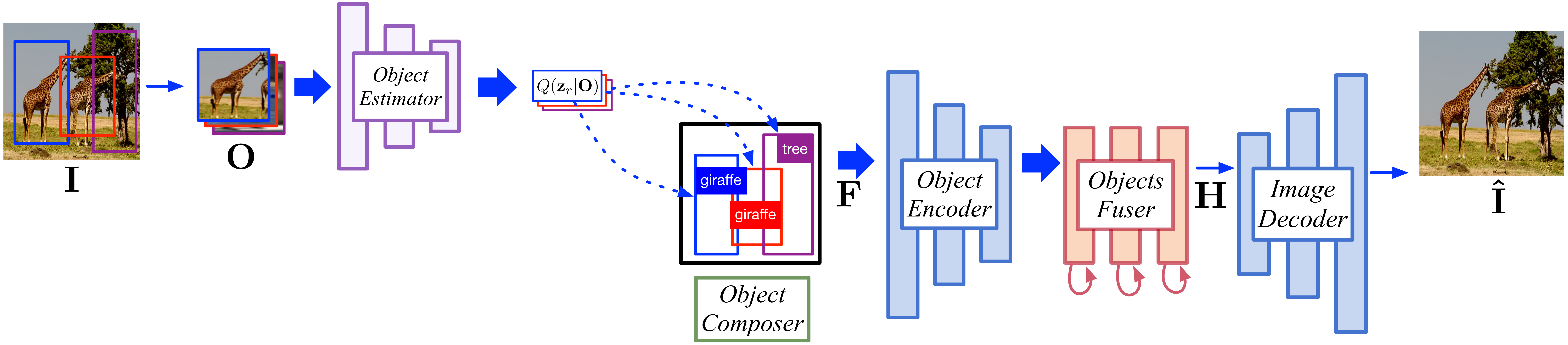




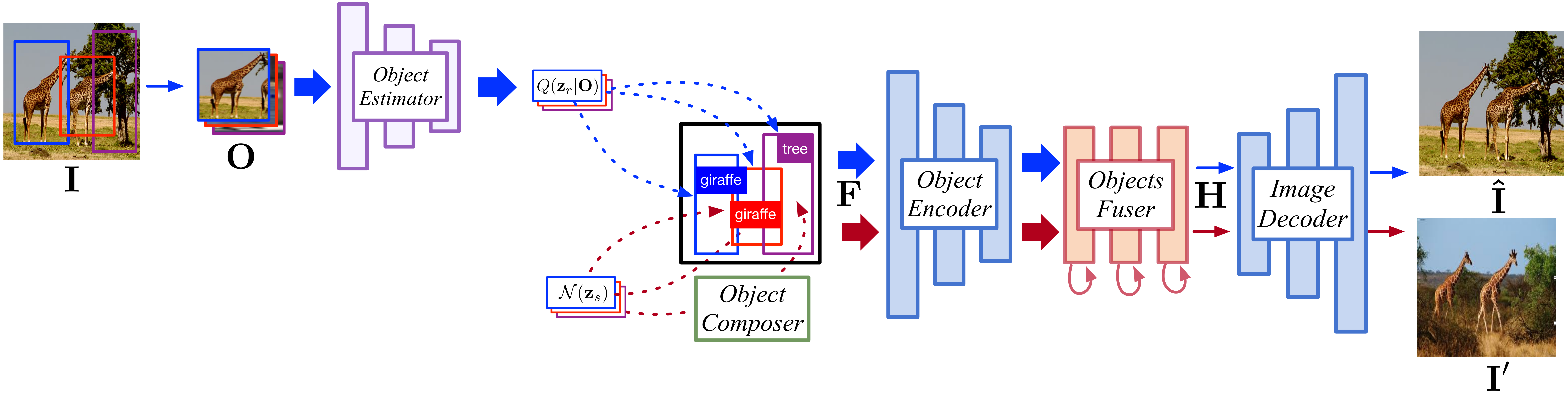
Model **Architecture**: Training



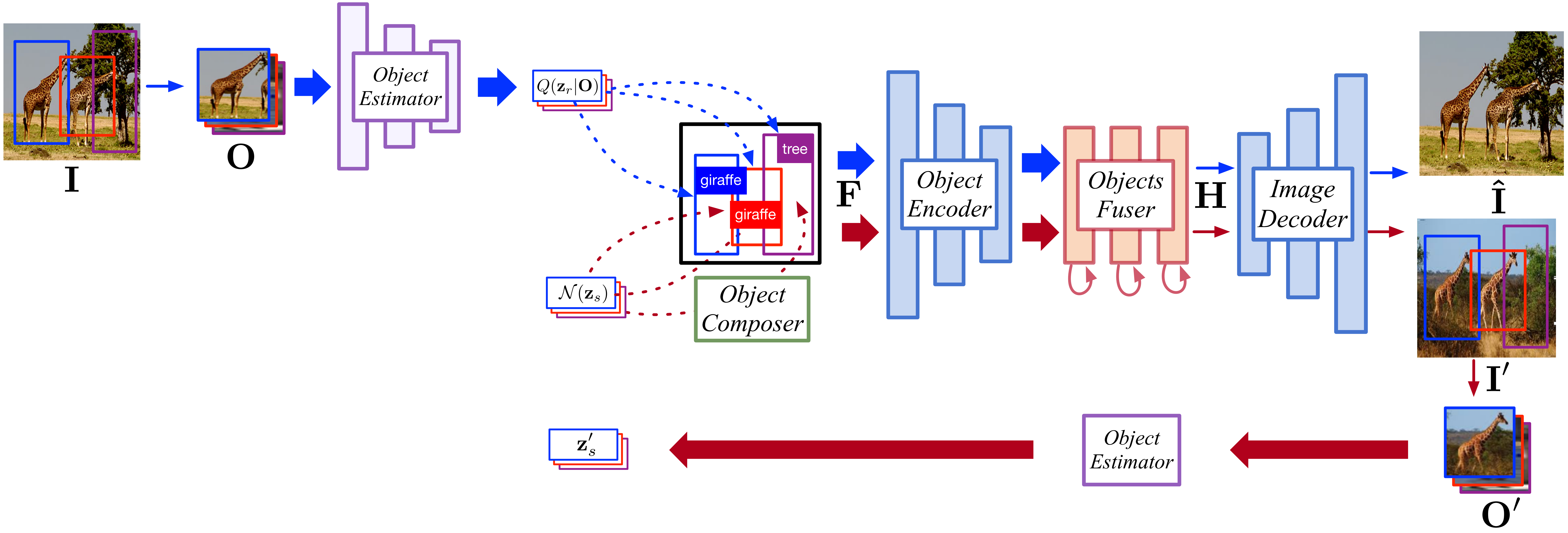
Model **Architecture**: Training



Model **Architecture**: Training

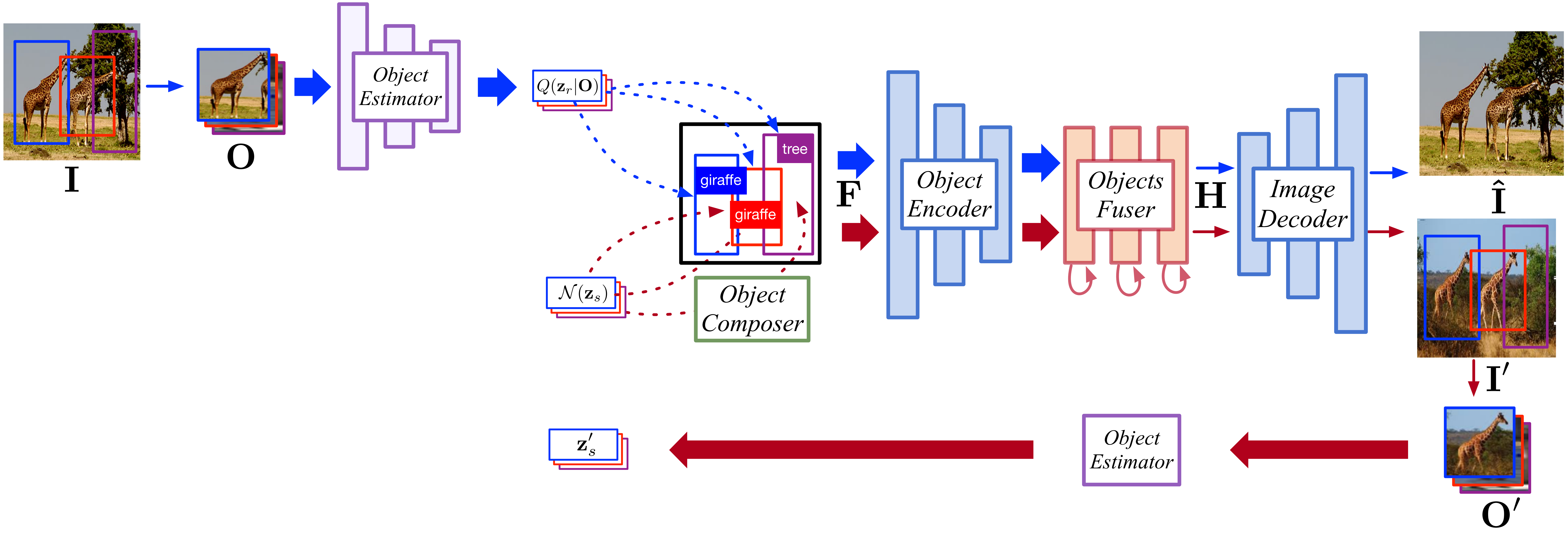


Model **Architecture**: Training



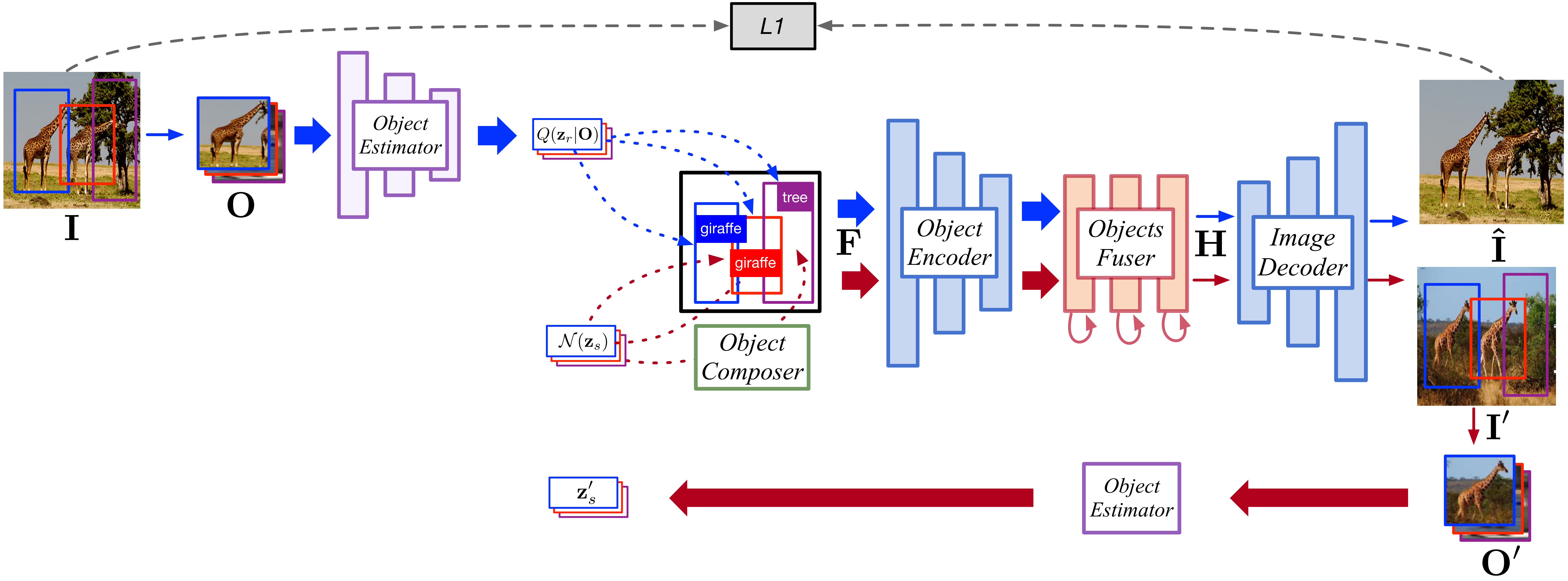
Model **Architecture**: Training

Losses



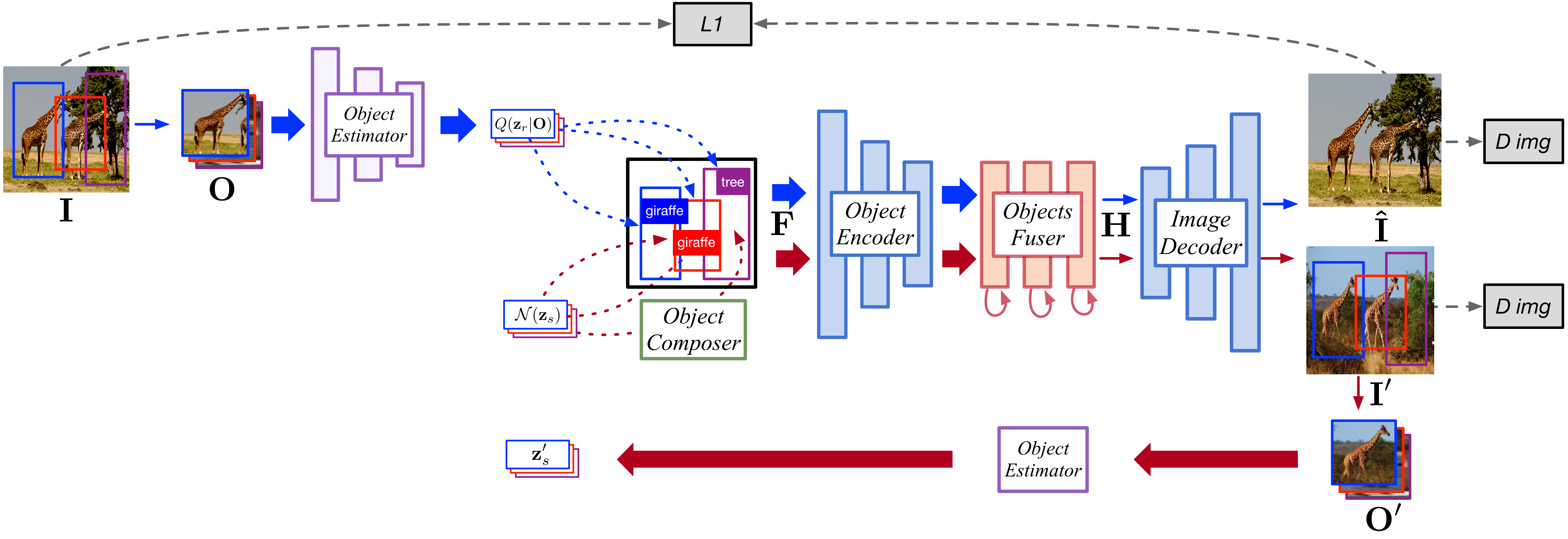
Model **Architecture**: Training

Losses



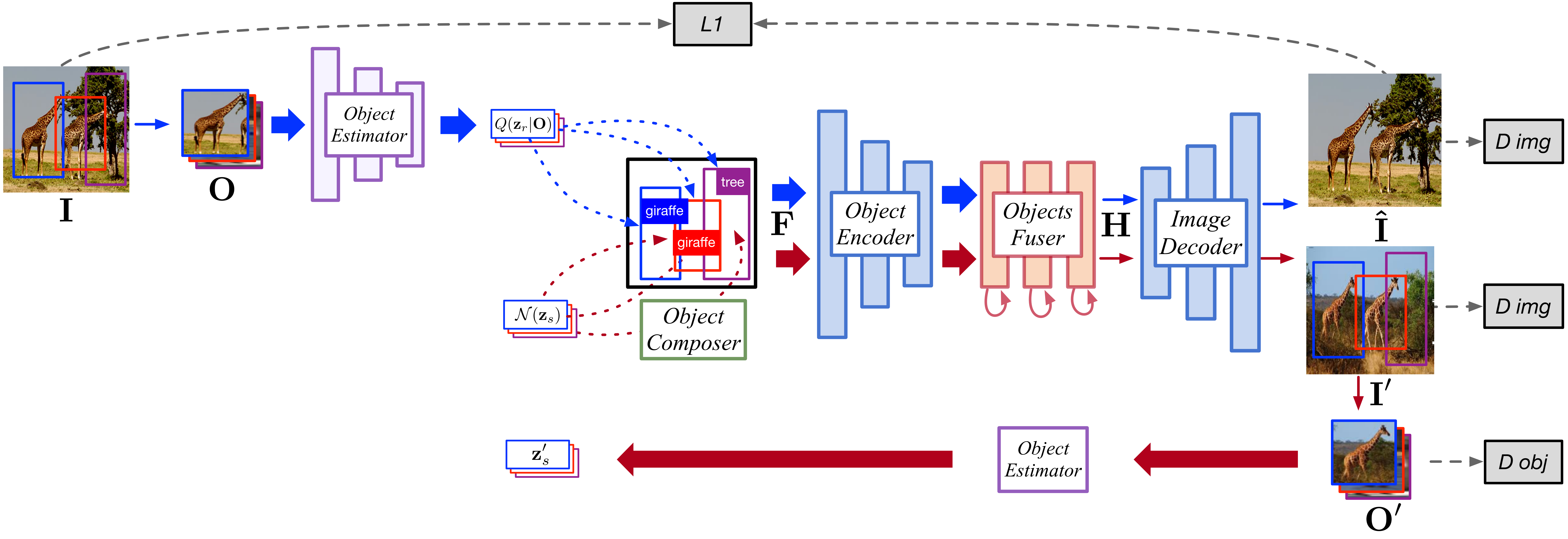
Model **Architecture**: Training

Losses



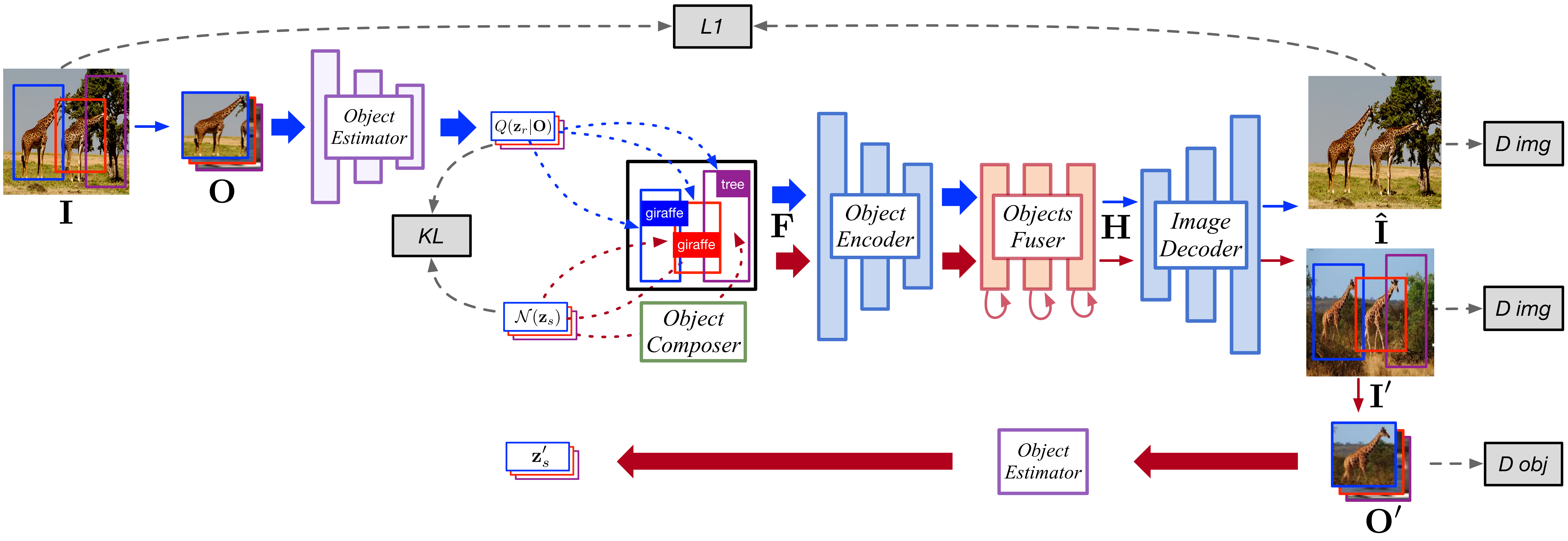
Model **Architecture**: Training

Losses



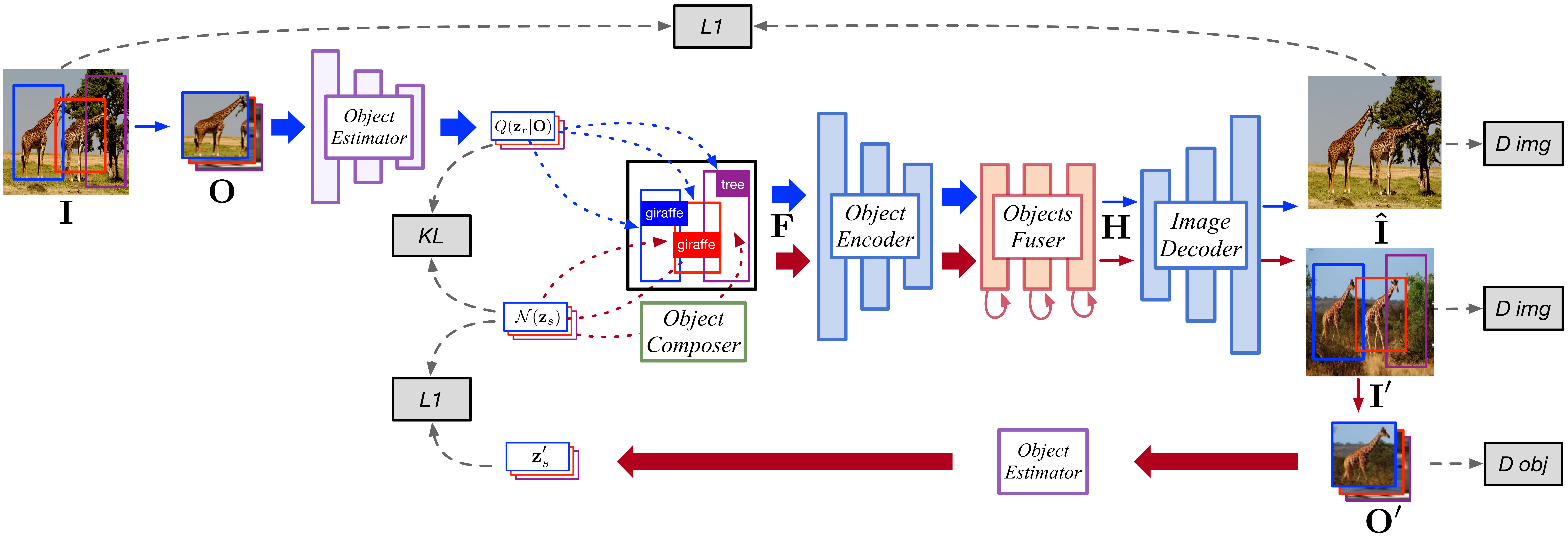
Model **Architecture**: Training

Losses



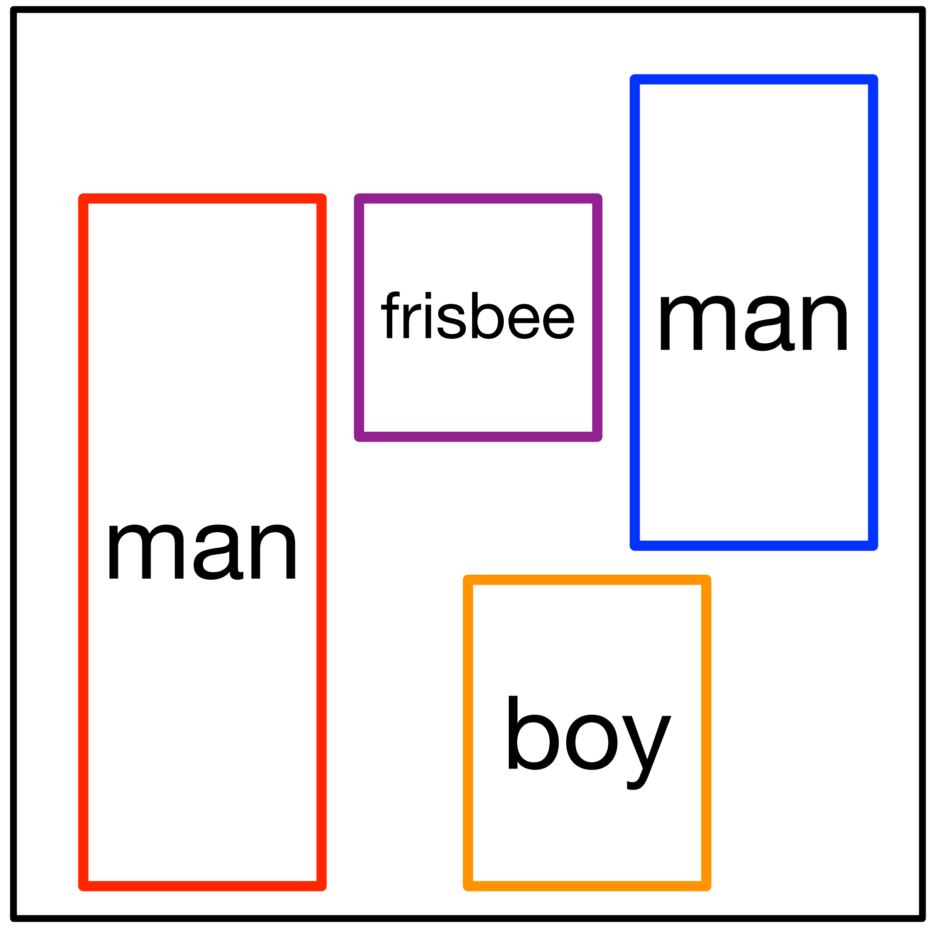
Model **Architecture**: Training

Losses

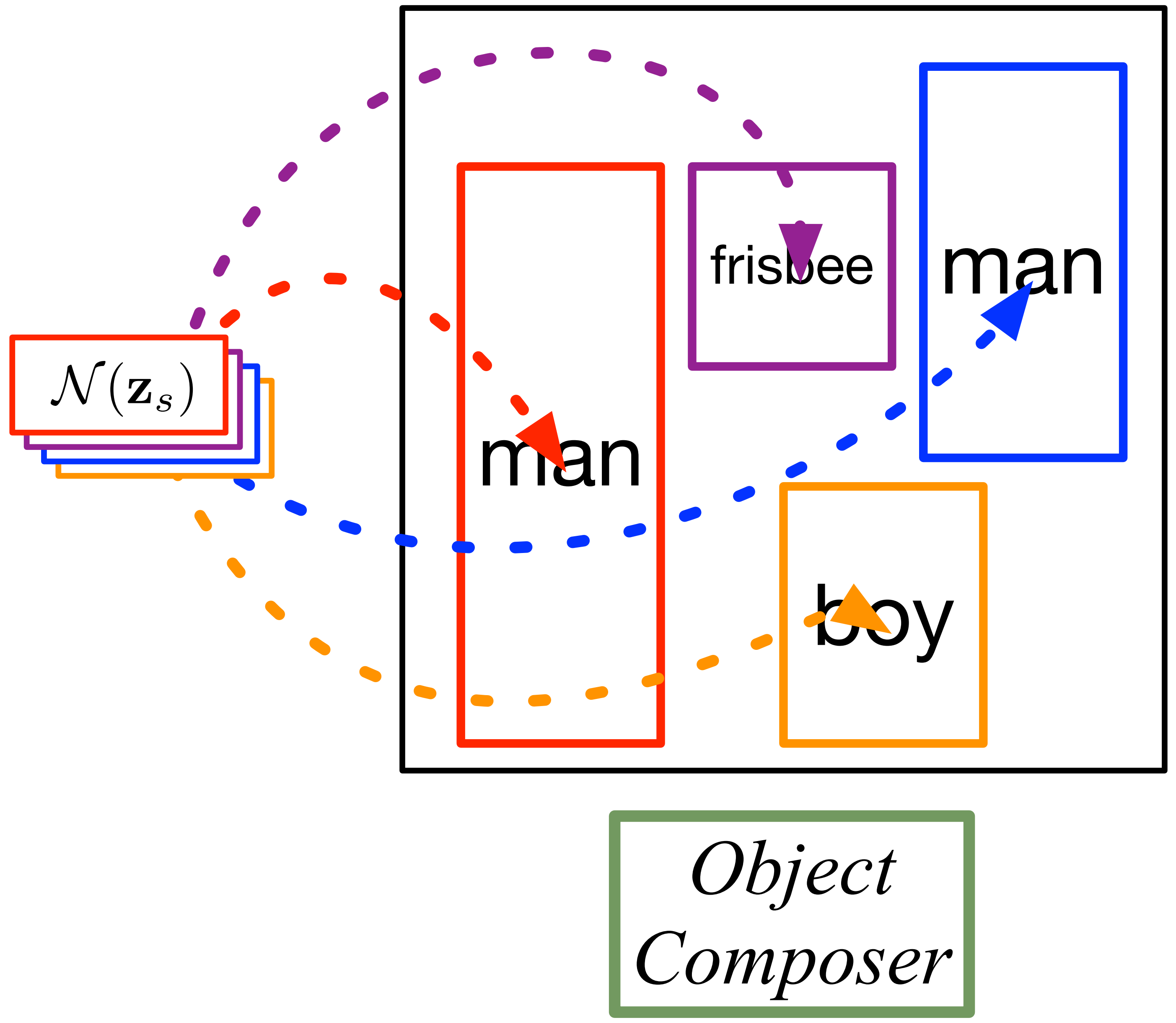


Model **Architecture**: Runtime

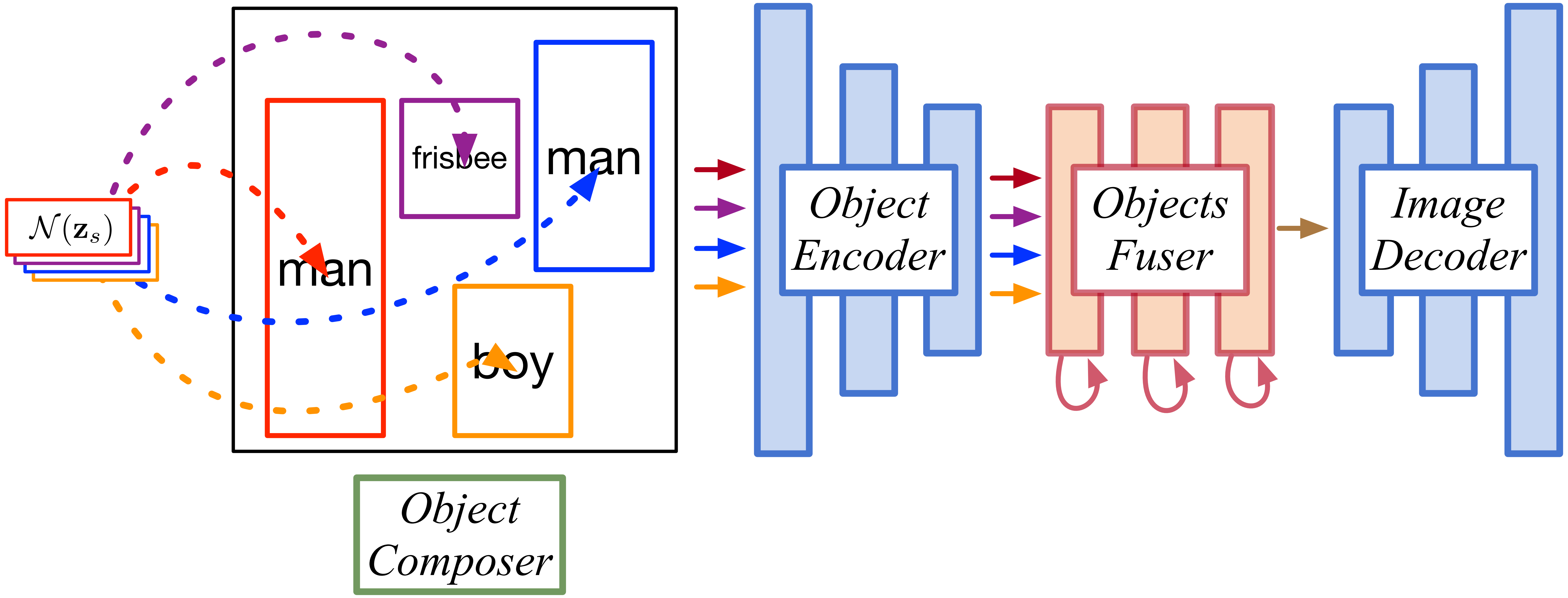
Model **Architecture**: Runtime



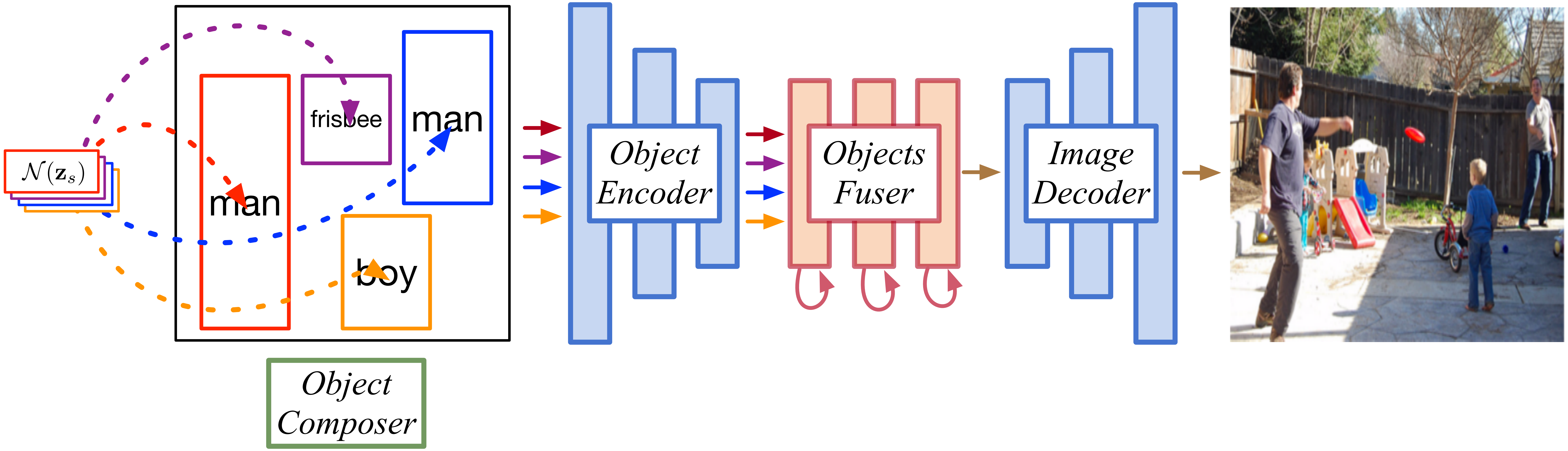
Model **Architecture**: Runtime



Model **Architecture**: Runtime



Model **Architecture**: Runtime



Experiments: Quantitative Results

Datasets:

Dataset	Train	Val.	Test	# Obj.	# Obj. in Image
COCO [1]	24,972	1,024	2,048	171	$3 \sim 8$
VG [18]	62,565	5,506	5,088	178	$3 \sim 30$

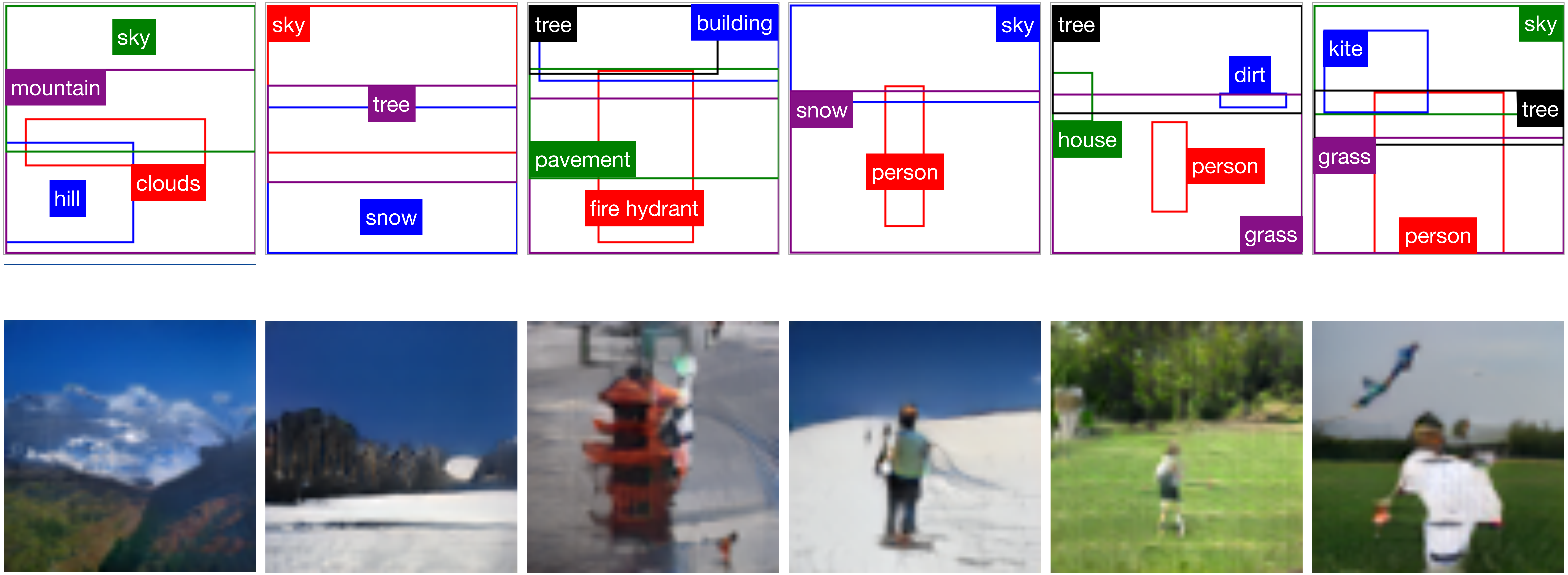
Evaluation:

	Inception Score		Object **Classification** Score		**Diversity** Score	
Method	**COCO**	**VG**	**COCO**	**VG**	**COCO**	**VG**
Real Images (64×64)	16.3 ± 0.4	13.9 ± 0.5	55.16	49.13	-	-
pix2pix [12]	3.5 ± 0.1	2.7 ± 0.02	12.06	9.20	0	0
sg2im (GT Layout) [13]	7.3 ± 0.1	6.3 ± 0.2	30.04	40.29	0.02 ± 0.01	0.15 ± 0.12
Ours	$\mathbf{9.1 \pm 0.1}$	$\mathbf{8.1 \pm 0.1}$	**50.84**	**48.09**	$\mathbf{0.15 \pm 0.06}$	$\mathbf{0.17 \pm 0.09}$

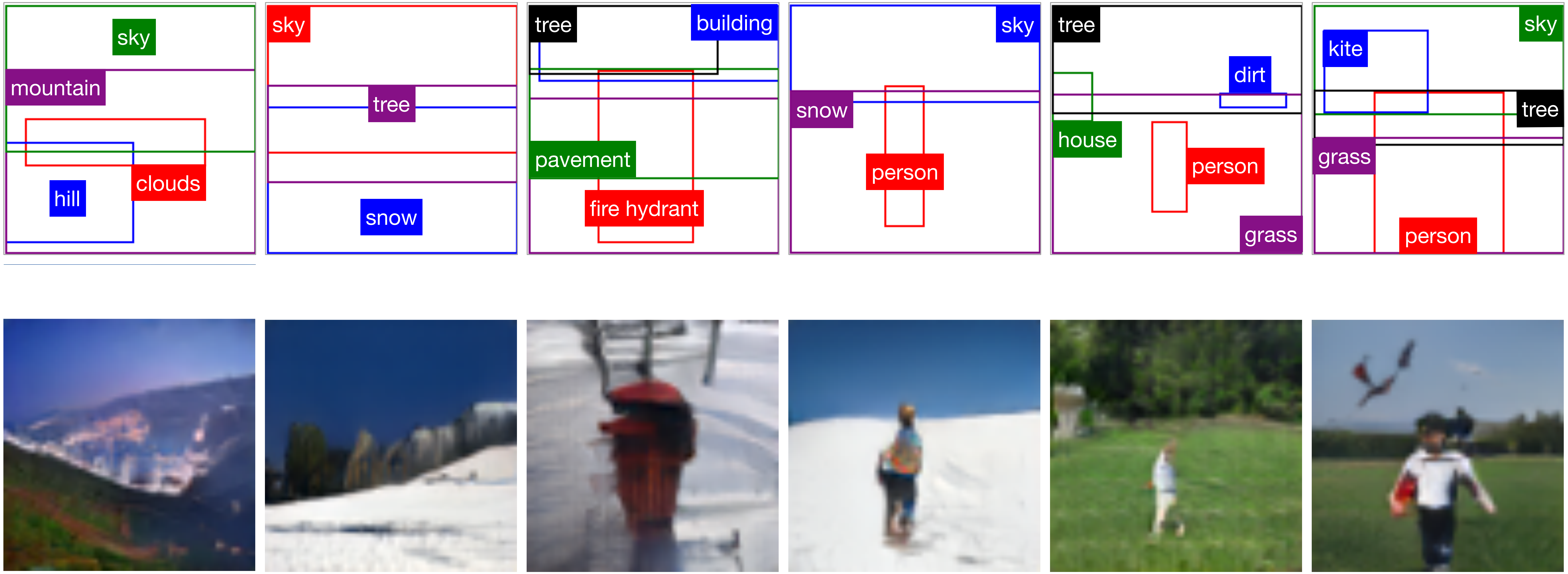
Results on COCO

Results on Visual Genome

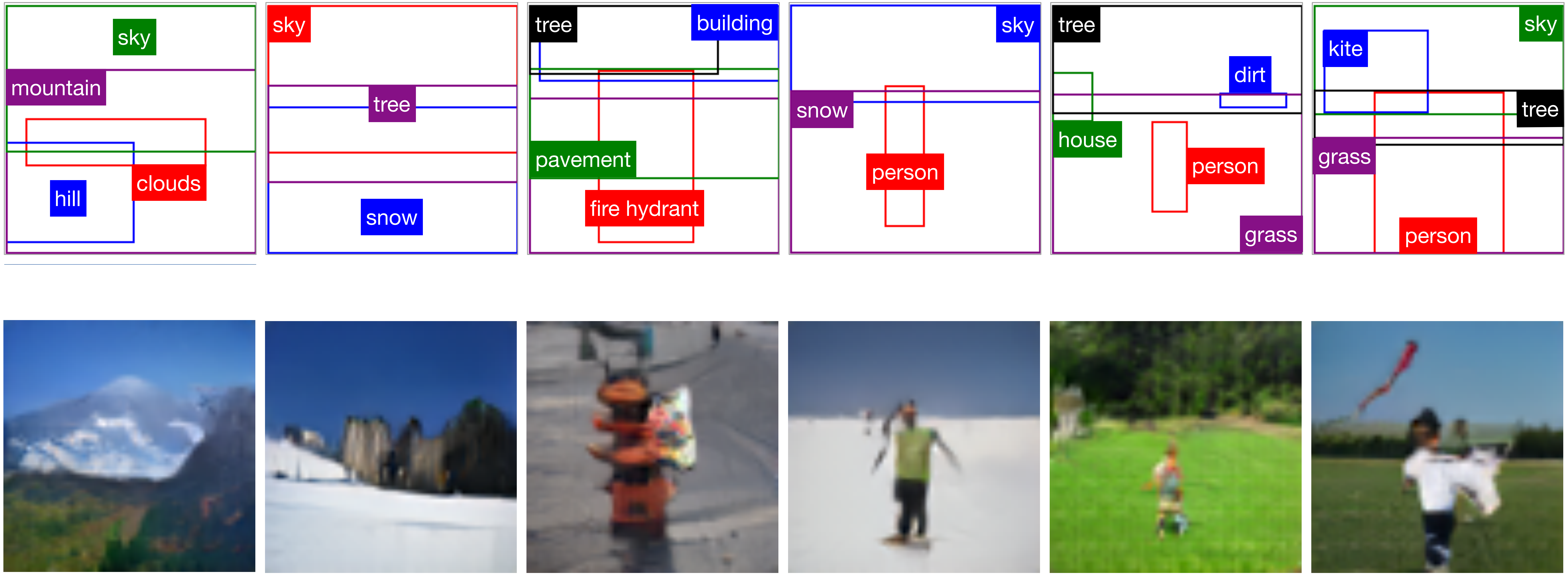
Results: Diversity



Results: Diversity



Results: Diversity



Layout to Image

Drag to draw bounding boxes and assign labels or simply load a pre-defined layout.

PERSON | PERSONS | INDOOR | BEACH | FOOD | BOAT | WINDOW | CAR | COW | MONITOR

Labels

Layout

Images

GENERATE | START OVER

Image Generation from Layout, Bo Zhao, Lili Meng, Weidong Yin and Leonid Sigal, CVPR 2019.

Web Application Developed by Mark (Ke) Ma

Layout to Image

Drag to draw bounding boxes and assign labels or simply load a pre-defined layout.

PERSON | PERSONS | INDOOR | BEACH | FOOD | BOAT | WINDOW | CAR | COW | MONITOR

Labels

Layout

Images

GENERATE

START OVER

Image Generation from Layout, Bo Zhao, Lili Meng, Weidong Yin and Leonid Sigal, CVPR 2019.

Web Application Developed by Mark (Ke) Ma

Conclusions

We propose a novel **layout2image** model, that is able to:

— Generate diverse results by sampling object appearances

— Outperform state of the art methods on COCO and Visual Genome datasets

GANs

Don't work with an explicit density function

Take game-theoretic approach: learn to generate from training distribution through 2-player game

Pros:

— Beautiful, state-of-the-art samples!

Cons:

— Trickier / more unstable to train

— Can't solve inference queries such as p(x), p(z|x)

Active area of research:

— Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)

— Conditional GANs, GANs for all kinds of applications

Non-Convergence

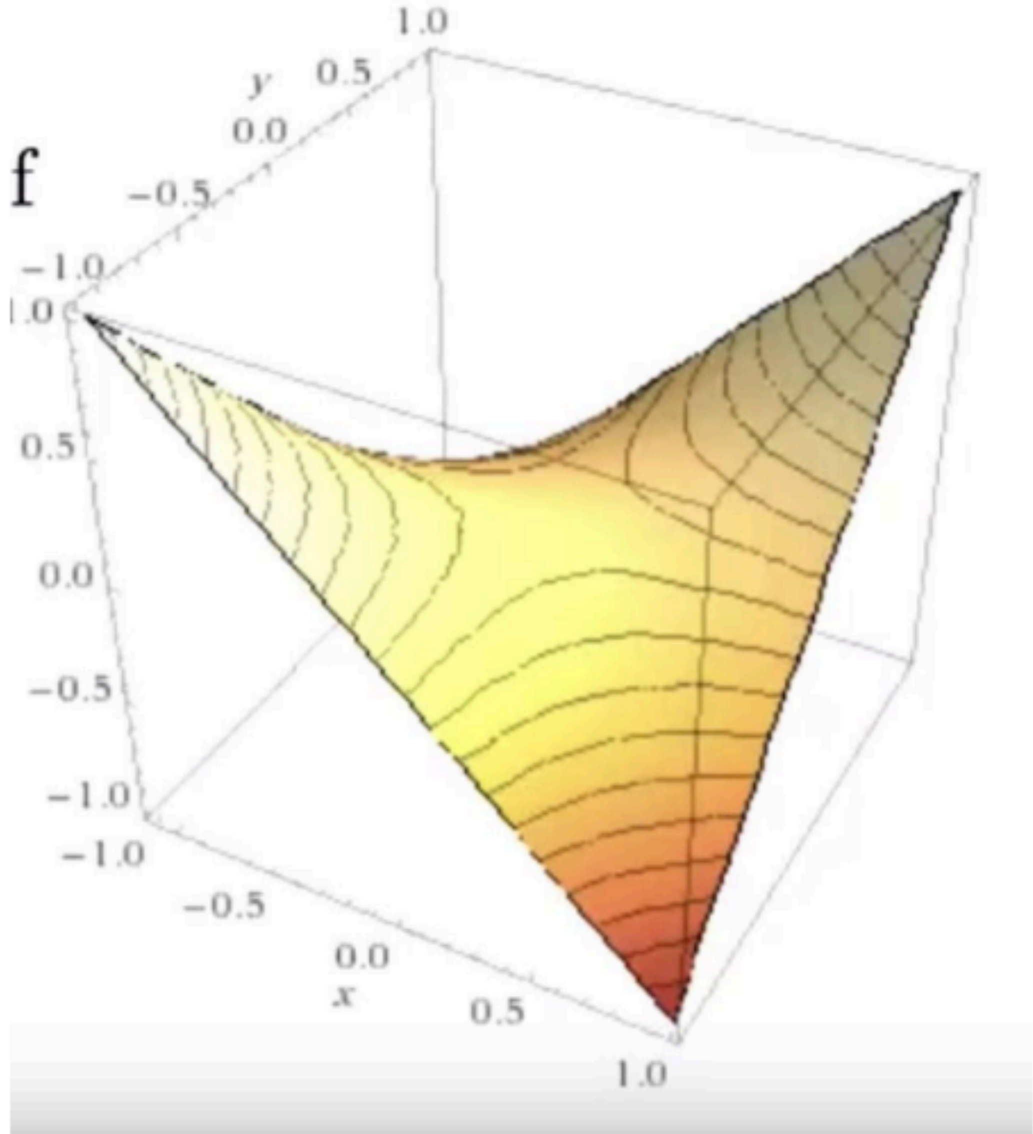
- Differential Equation's solution has sinusoidal terms
- Even with a small learning rate, it will not converge
- Discrete time gradient descent can spiral outward for large step size

D & G nullifies each others learning in every iteration

Train for a long time – without generating good quality samples

$$V(x, y) = xy$$

$$x = 0, \qquad y = 0$$



$$V(x(t), y(t)) = x(t)y(t)$$

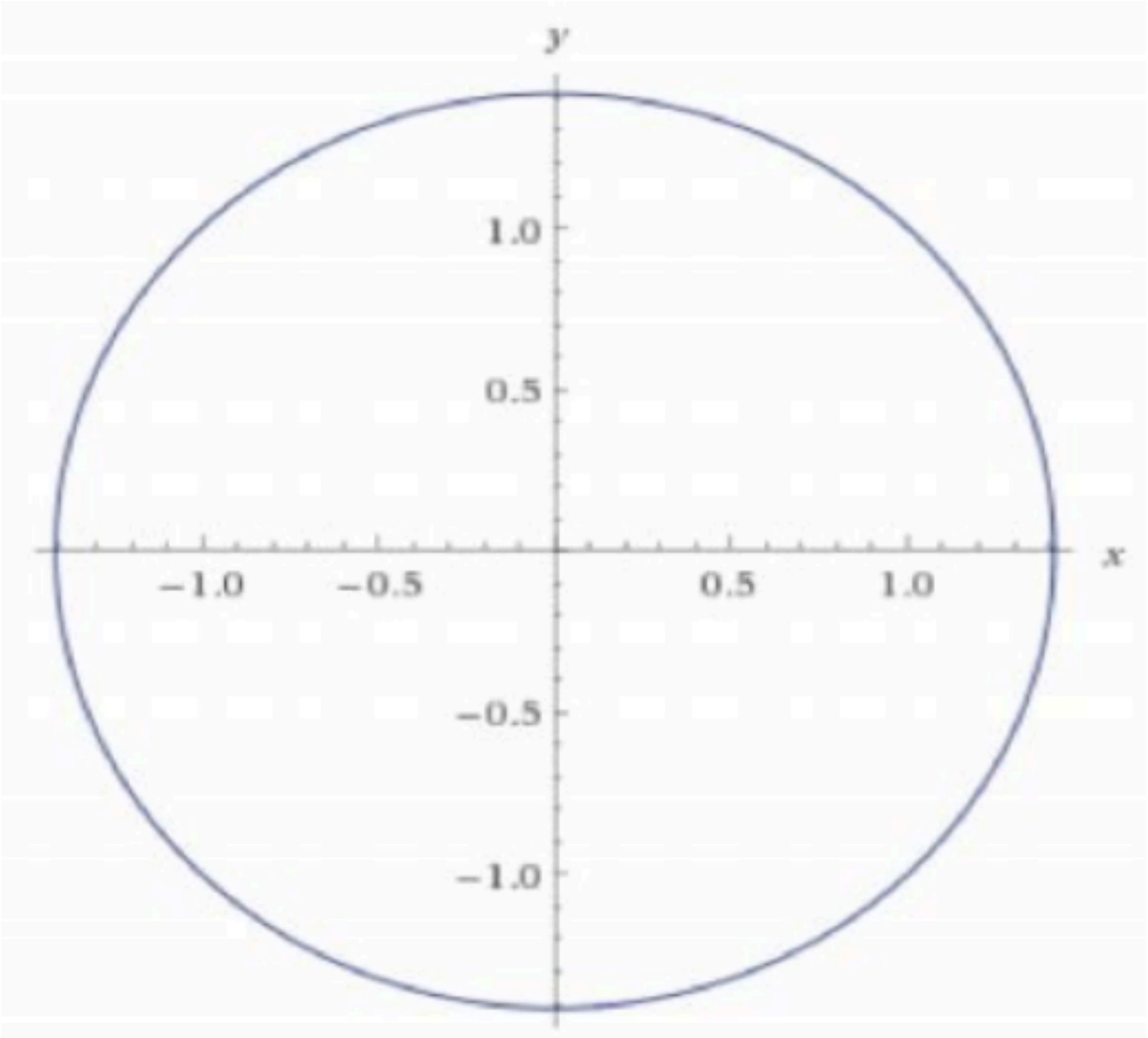
$$\frac{\partial x}{\partial t} = -y(t)$$

$$\frac{\partial y}{\partial t} = x(t)$$

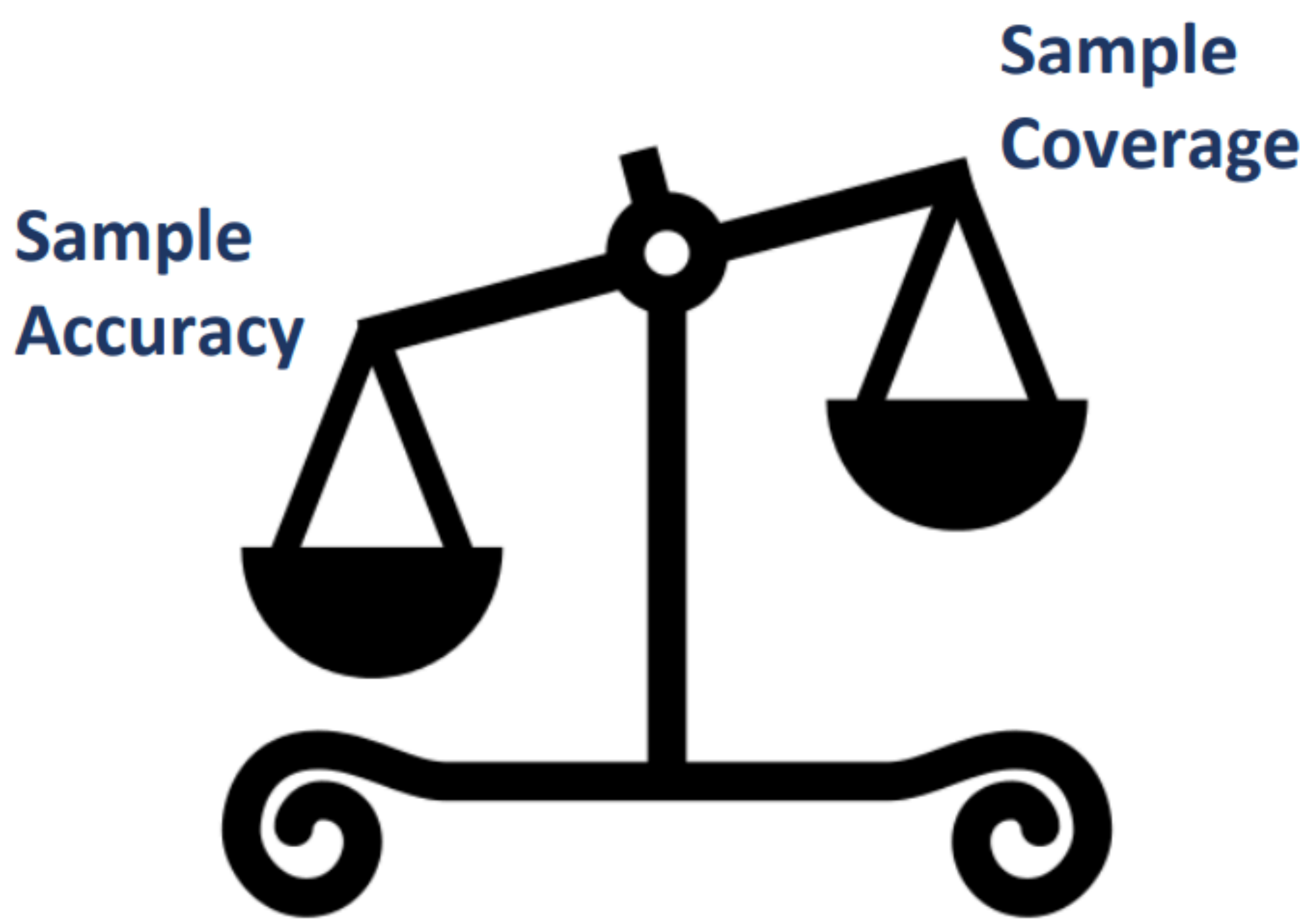
$$\frac{\partial^2 y}{\partial t^2} = \frac{\partial x}{\partial t}. = -y(t)$$

$$x(t) = x(0)cost(t) - y(0)\sin(t)$$

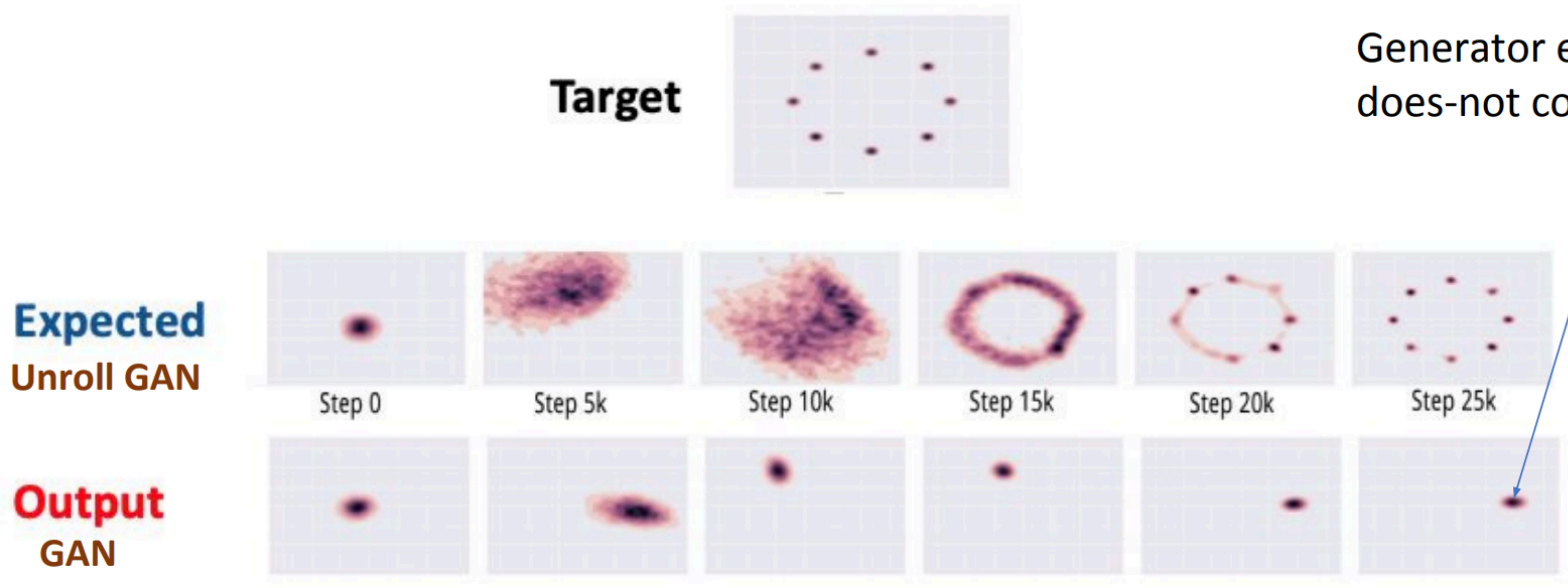
$$\mathrm{y}(t) = x(0)cost(t) - y(0)\sin(t)$$



Mode Collapse



Generator excels in a subspace but does-not cover entire real distribution



Luke et al. 2016

Why **GANs** are hard to train?

- — Generator keeps generating similar images — so nothing to learn
- — Maintain trade-off of generating more **accurate** vs. high **coverage** samples
- — Two learning tasks need to have balance to achieve stability
  - — If the **discriminator** is not sufficiently trained — it can worsen generator
  - — If the **discriminator** is too good — will produce no gradients