THE UNIVERSITY OF BRITISH COLUMBIA

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

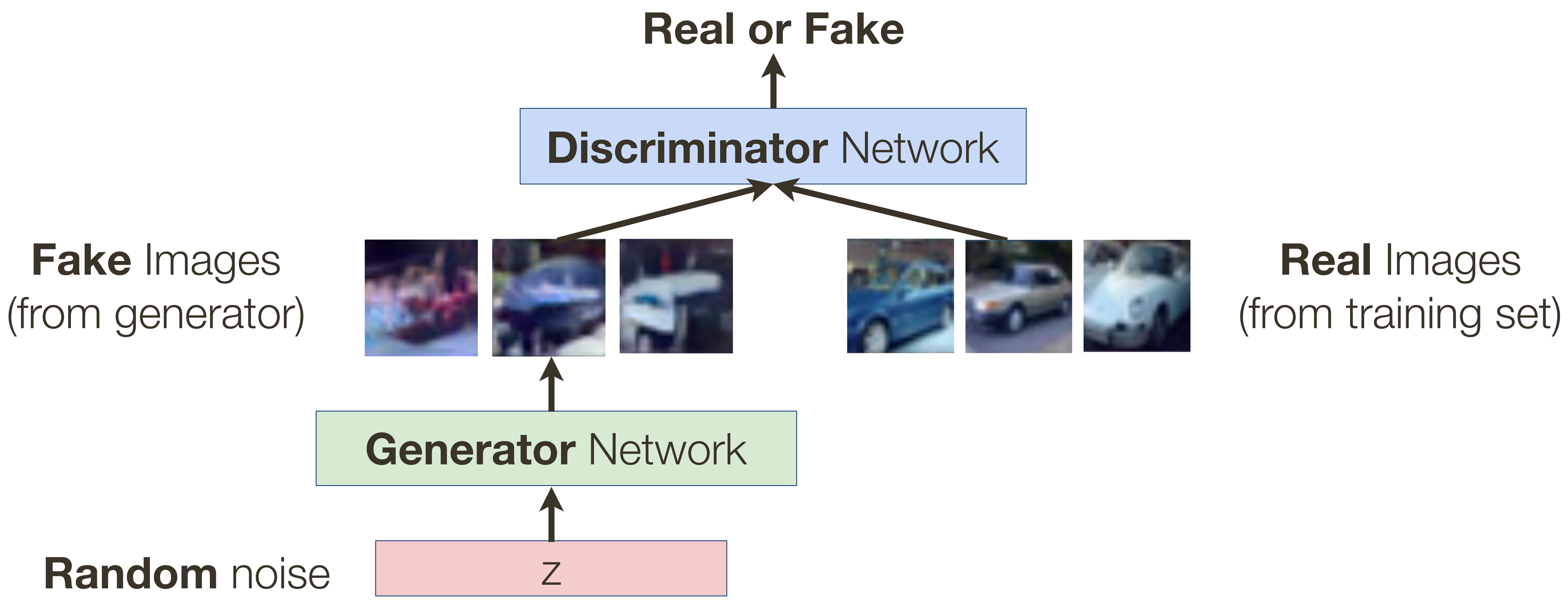
Lecture 18: GANs [cont]

Generative Adversarial Networks (GANs)

Training GANs: Two-player Game

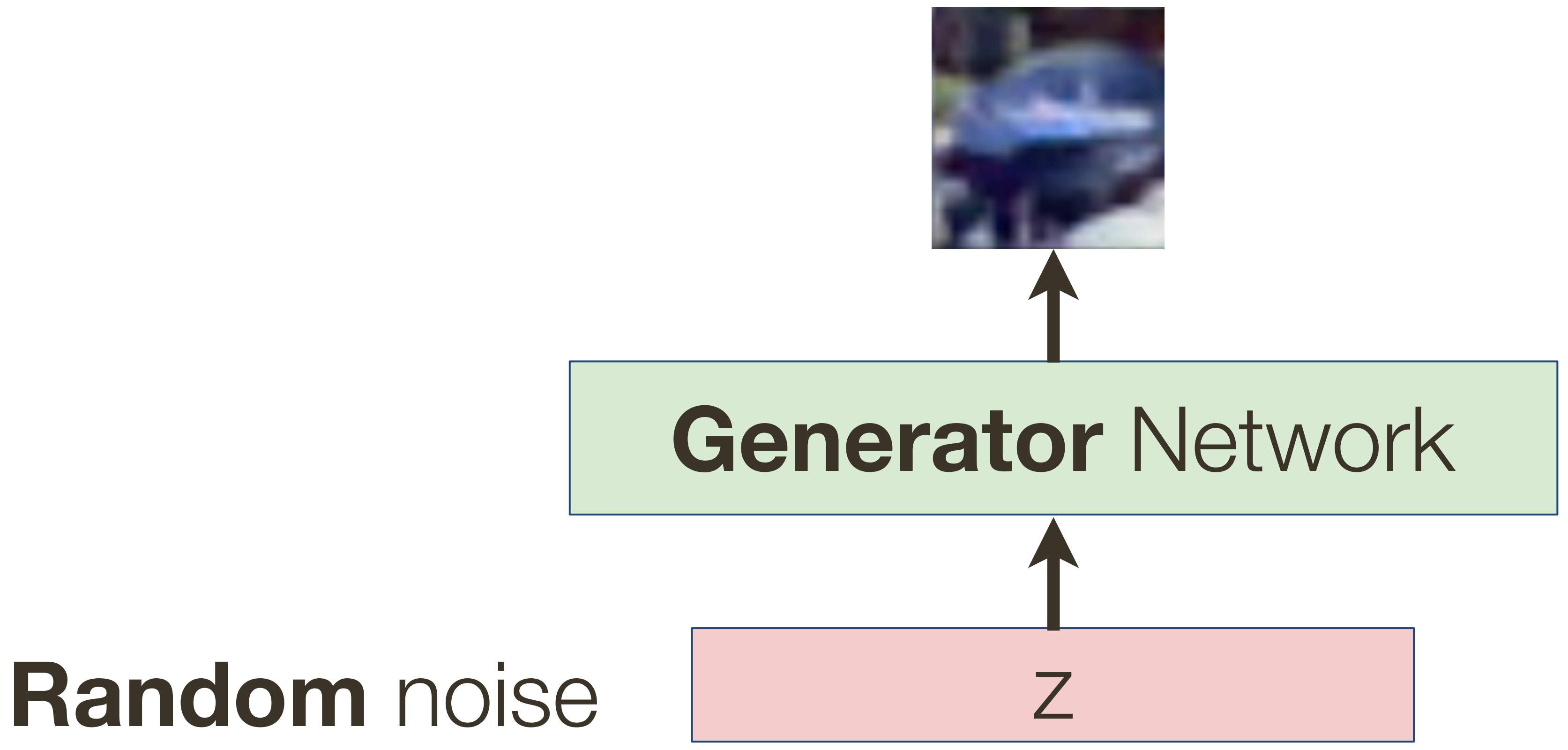
[Goodfellow et al., 2014]

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



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

Sampling **GAN**s

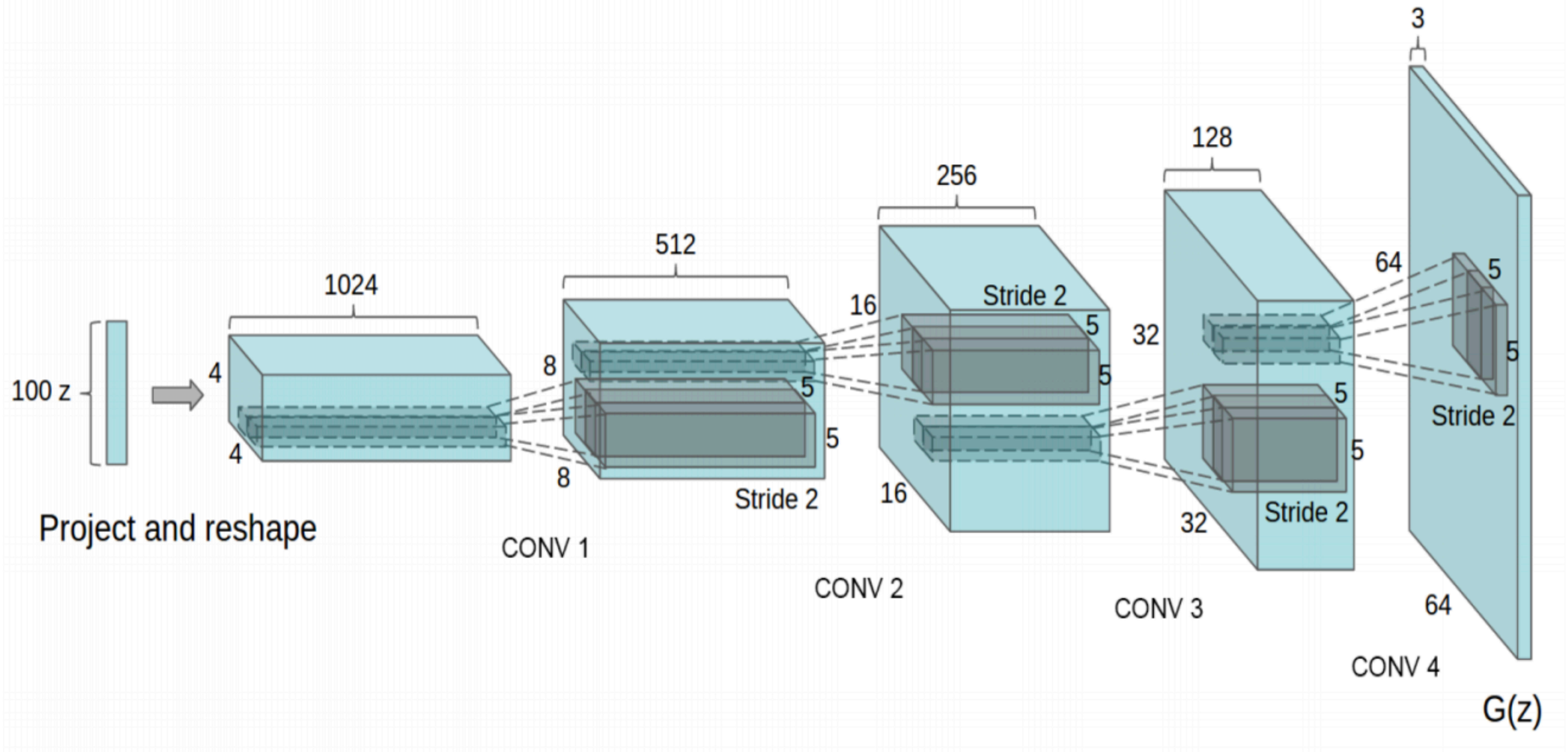


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

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

Conditional GAN: Text-to-Image Synthesis

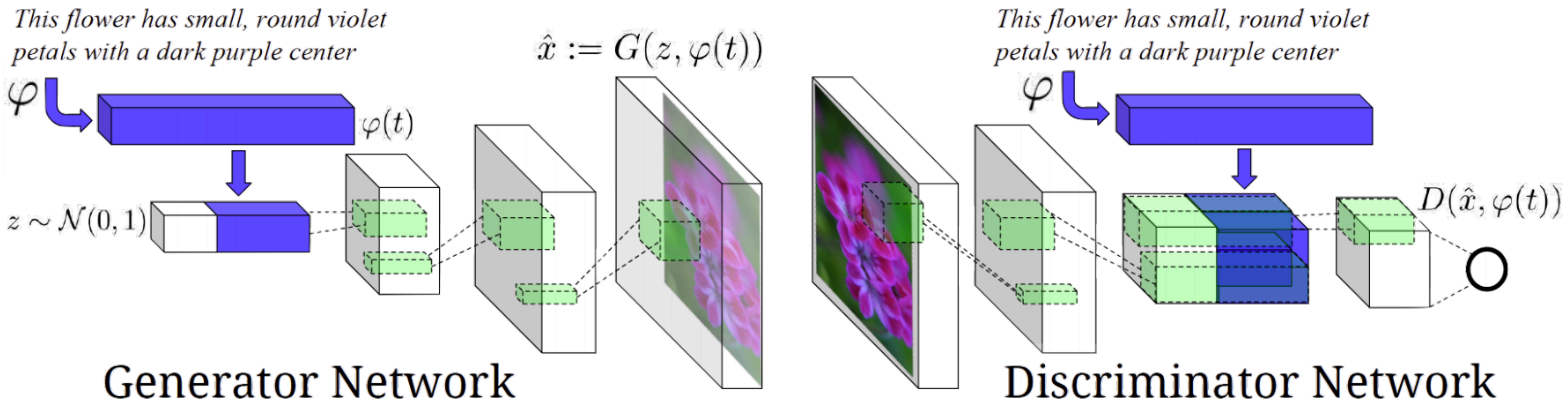


Figure 2 in the original paper.

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

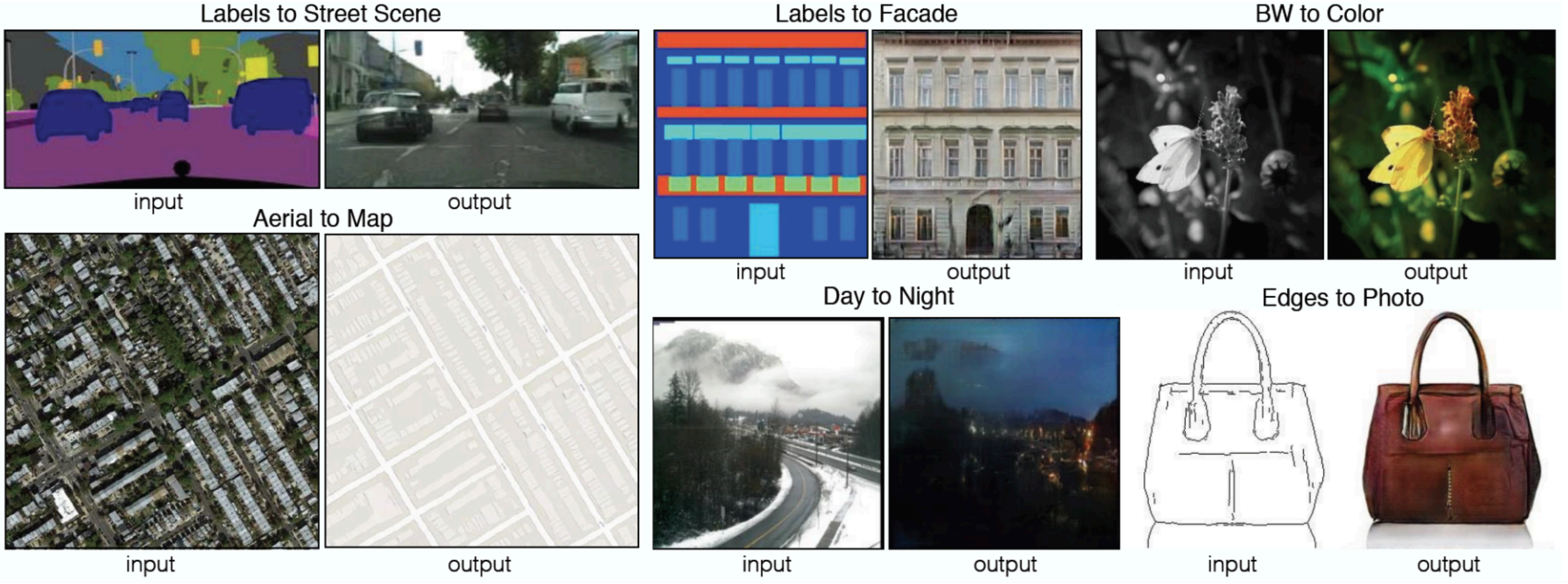


Figure 1 in the original paper.

[Isola et al., 2016]

Conditional GAN: Image-to-Image translation

Architecture: DCGAN-based

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

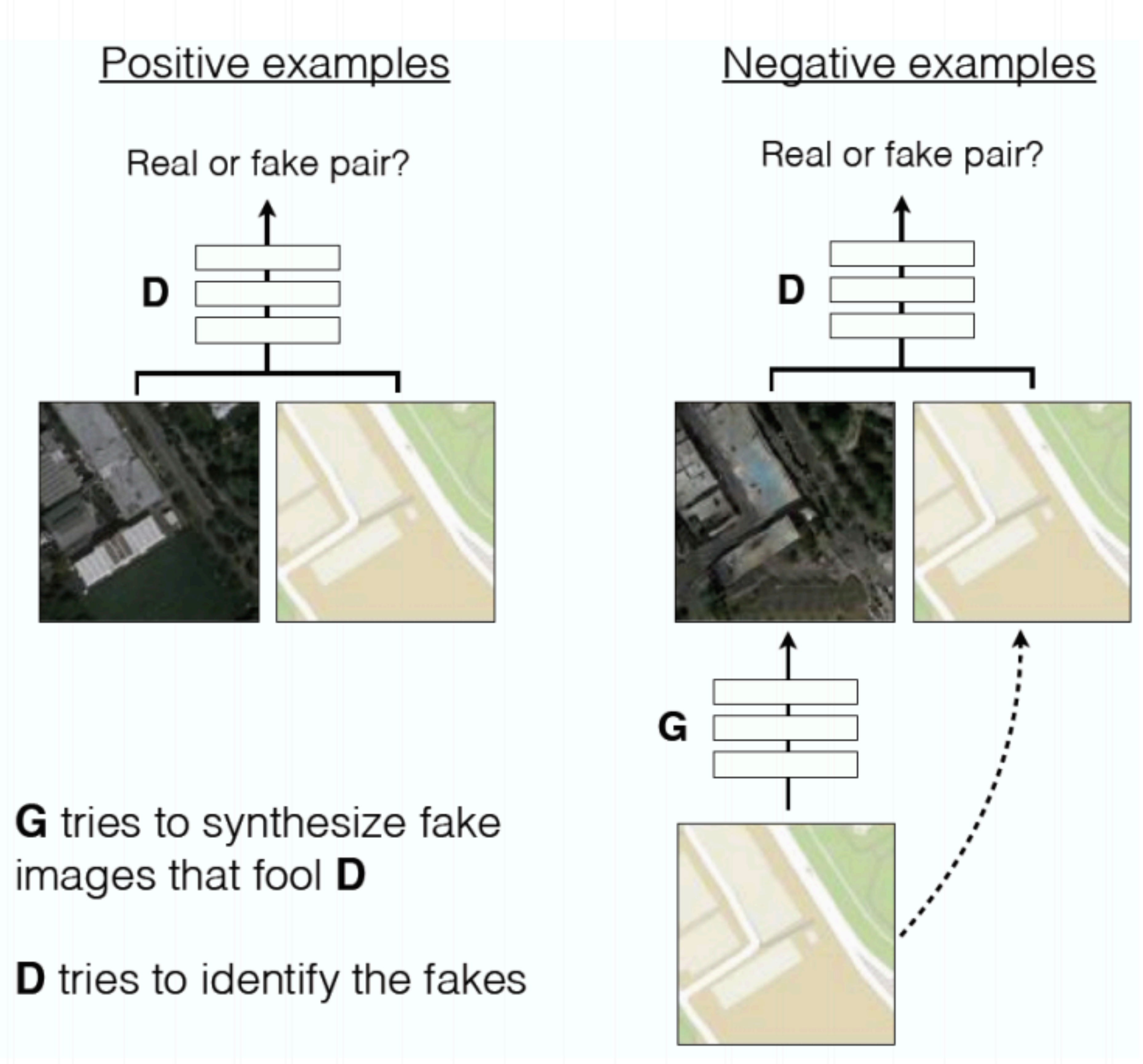
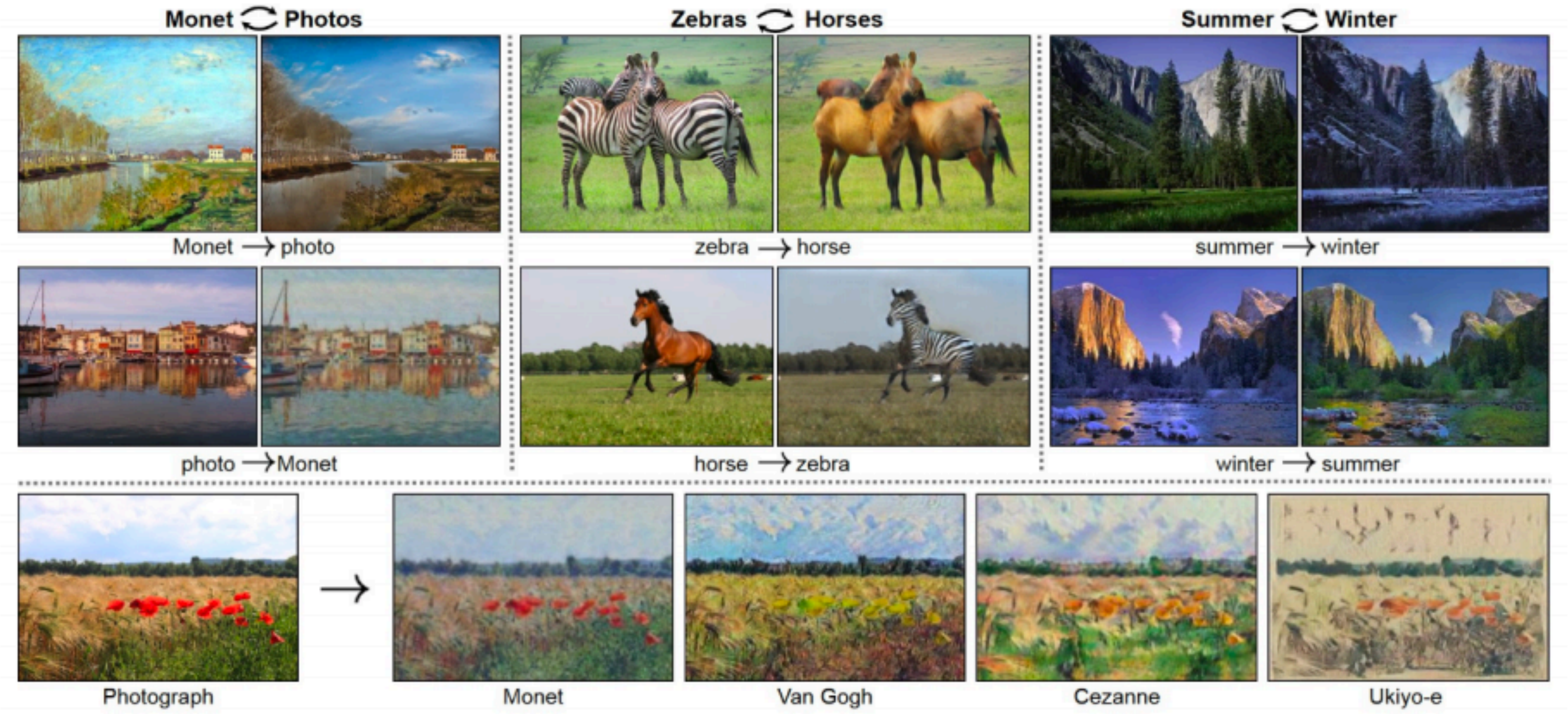


Figure 2 in the original paper.

[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

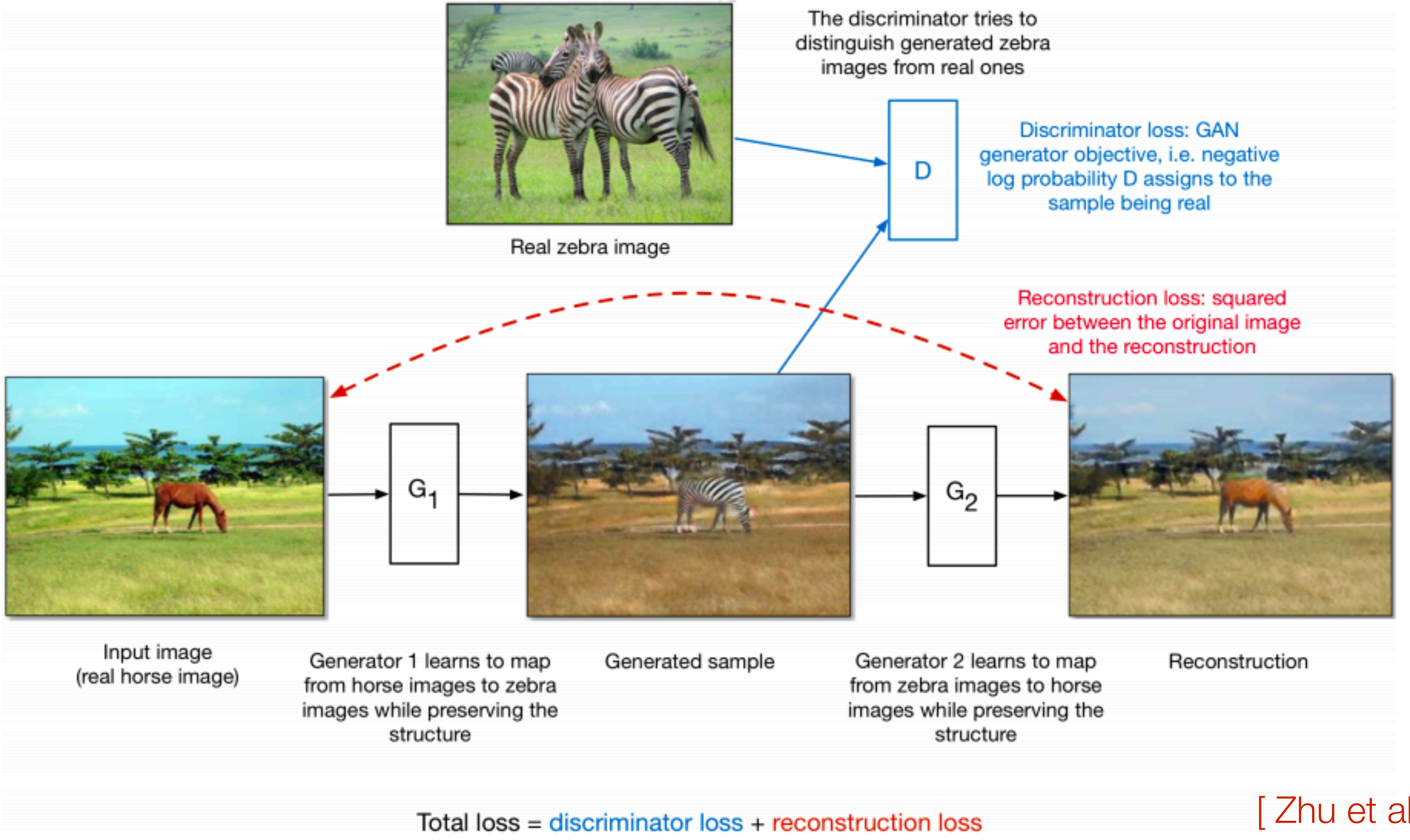
— Make sure the generated (translated) samples of Style 2 are indistinguishable from real images of Style 2 by a discriminator network

— 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]

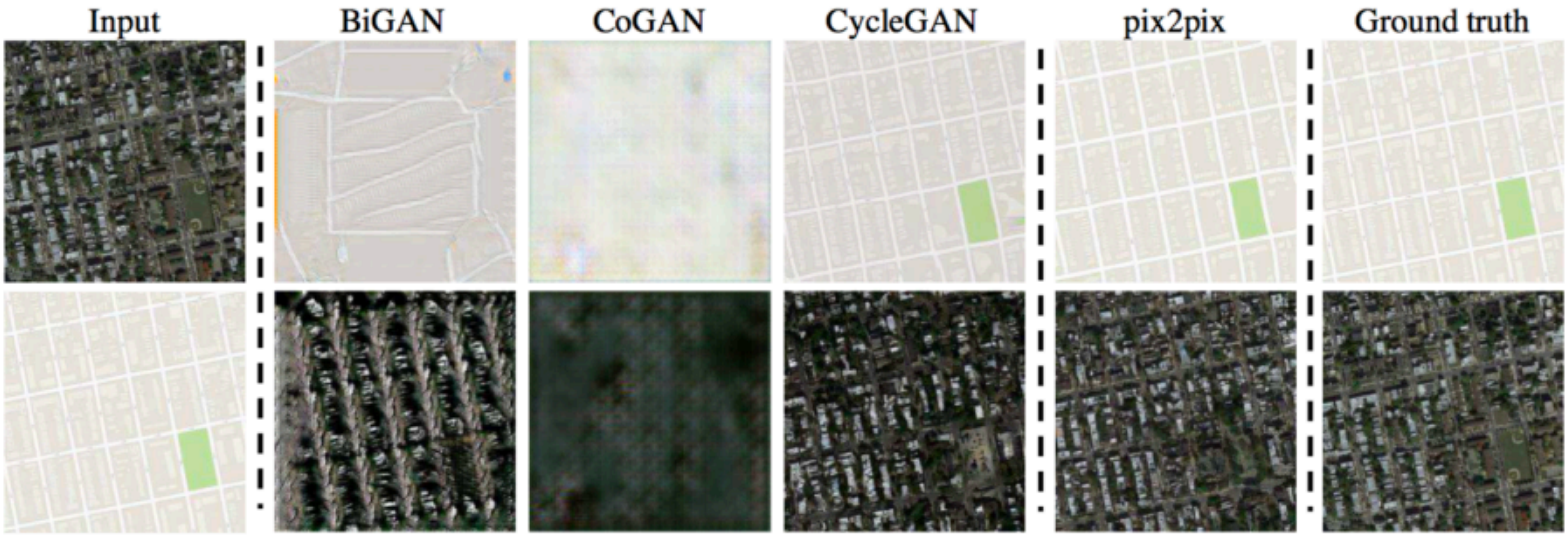
CycleGAN: Unpaired Image-to-Image translation



[Zhu et al., 2017]

CycleGAN: Unpaired Image-to-Image translation

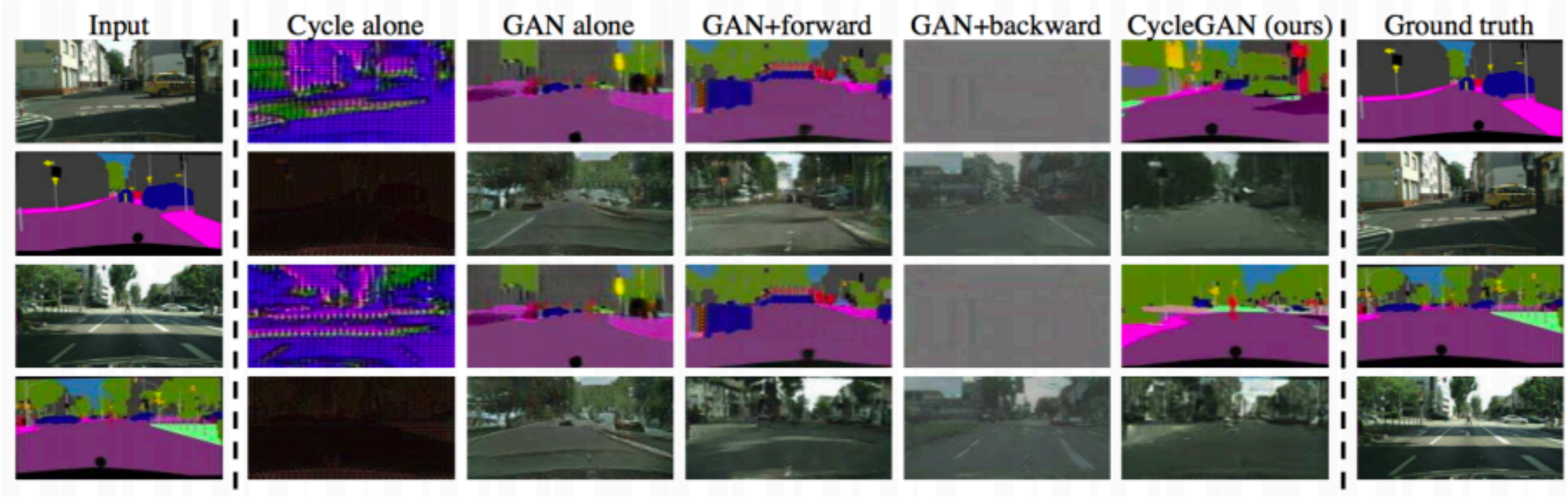
Ariel photos to **maps:**



[Zhu et al., 2017]

CycleGAN: Unpaired Image-to-Image translation

Images to **semantic segmentation:**



[Zhu et al., 2017]

Laplacian Pyramid GAN

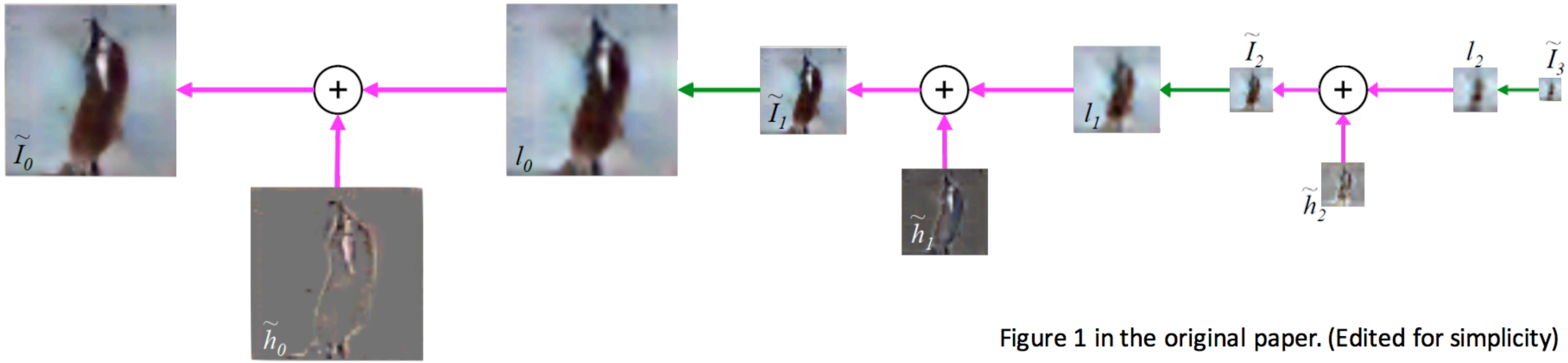


Figure 1 in the original paper. (Edited for simplicity)

— 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

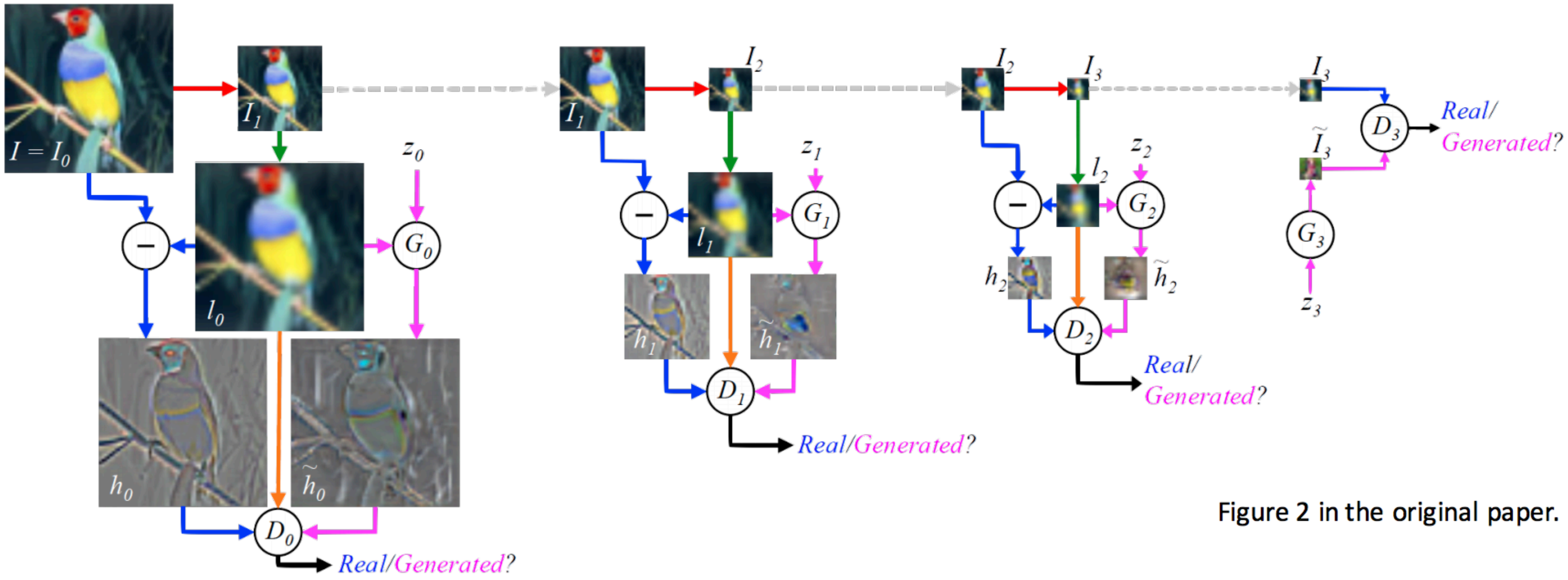


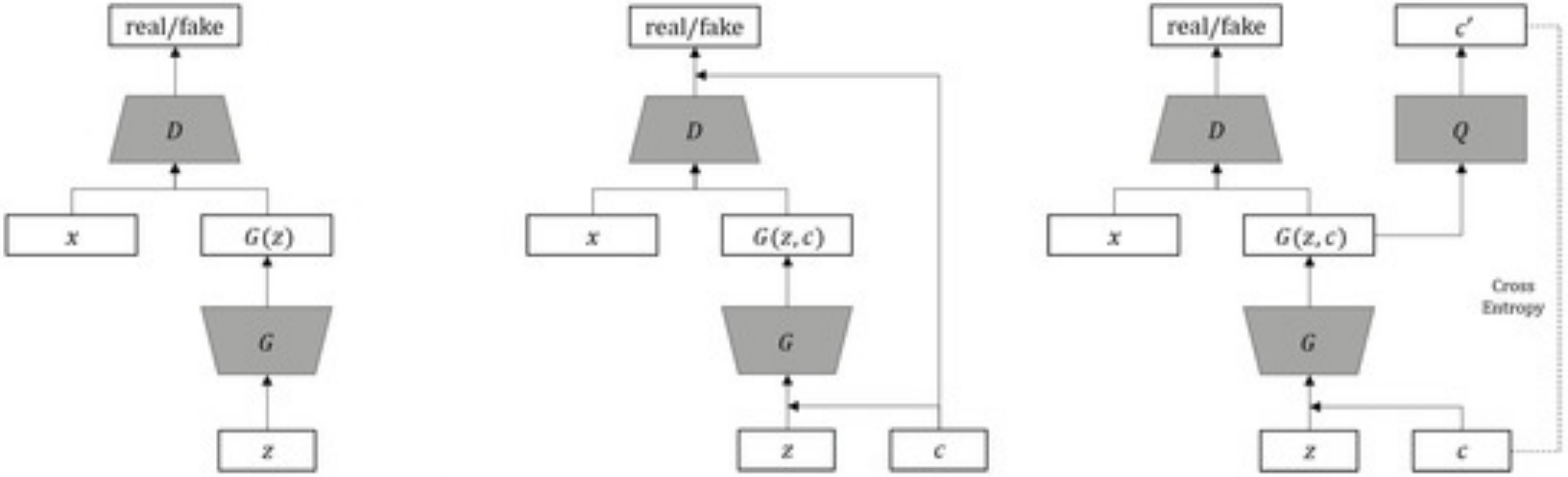
Figure 2 in the original paper.

— 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

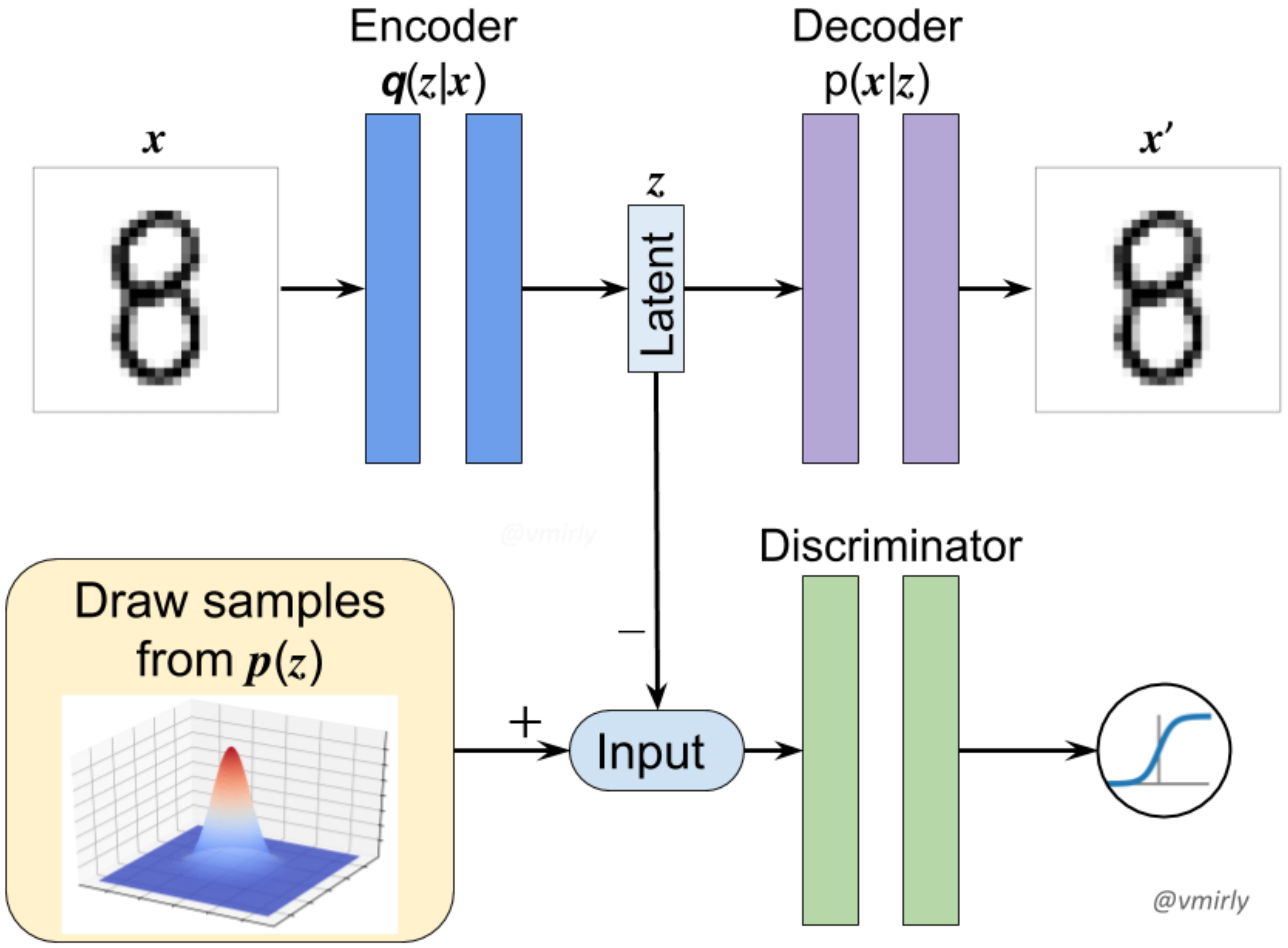


(a) GAN, DCGAN, LSGAN, WGAN (b) CGAN (c) InfoGAN

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

[Chen et al., 2016]

Adversarial Autoencoder (GAN + VAE)



[Makhzani et al., 2015]

CVPR LONG BEACH CALIFORNIA June 16-20, 2019

Image Generation from Layout

Bo Zhao

Lili Meng

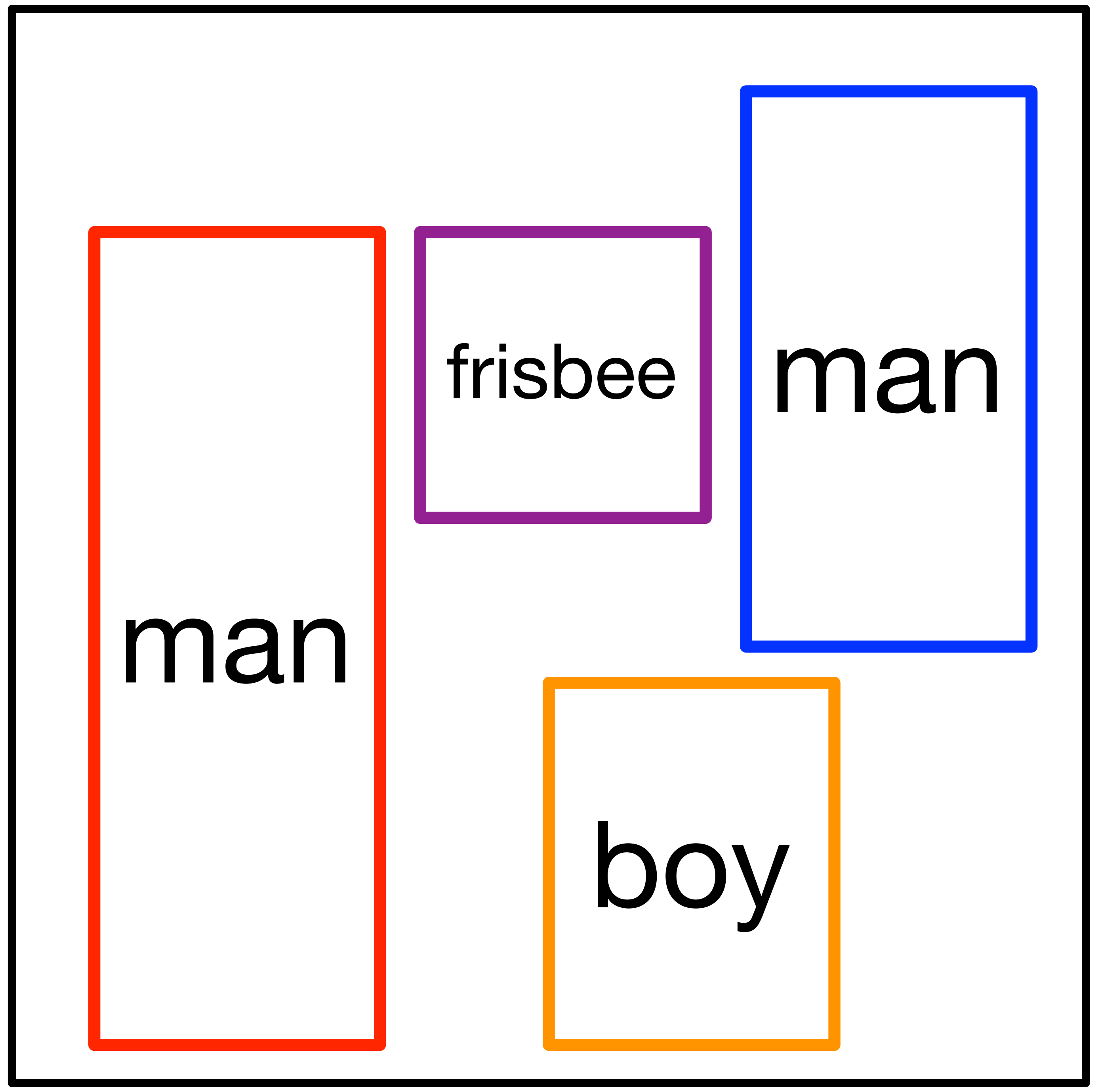
Weidong Yin

Leonid Sigal

UBC

Image Generation from Layout

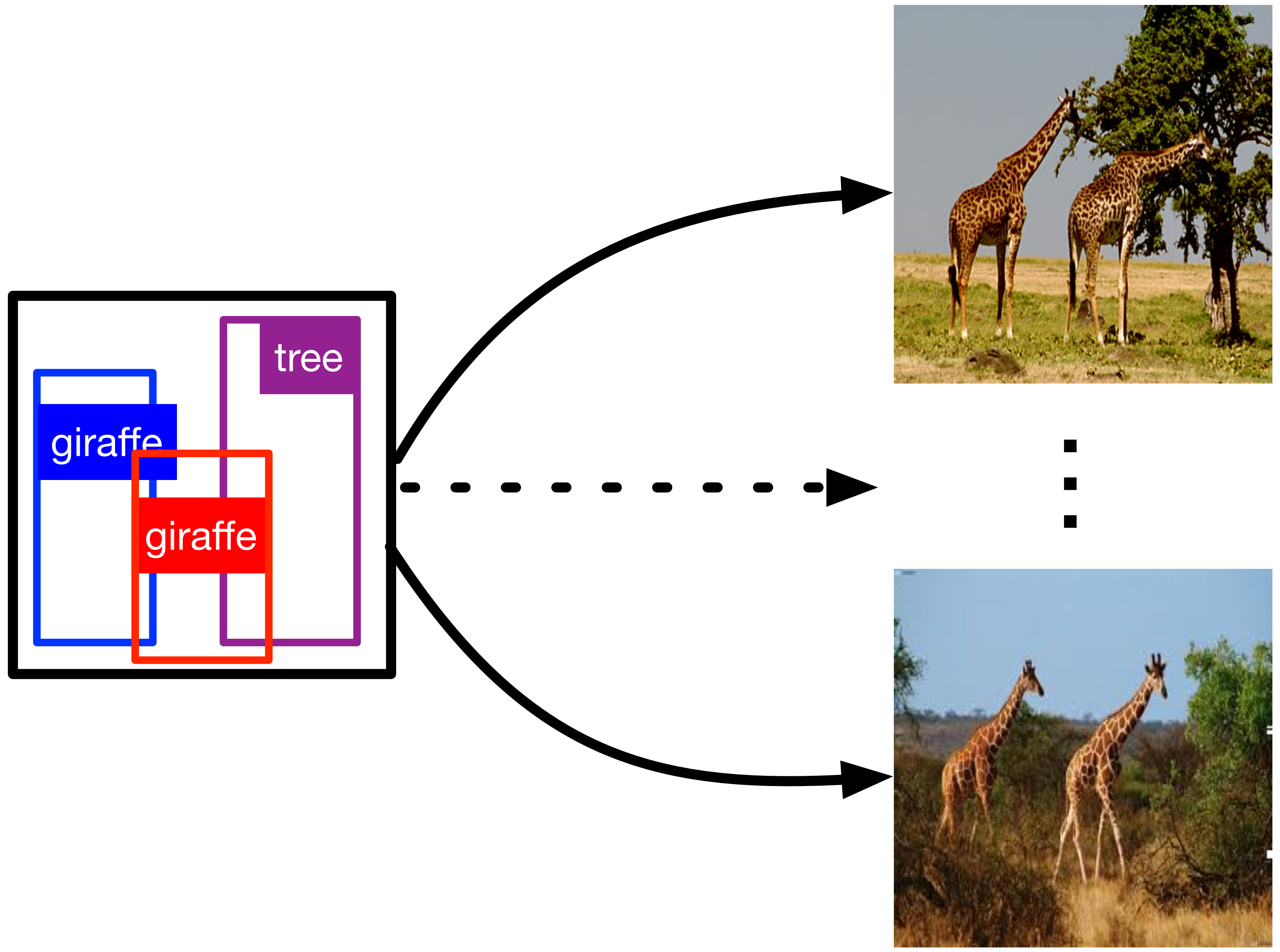
Image Generation from Layout



Layout

Results

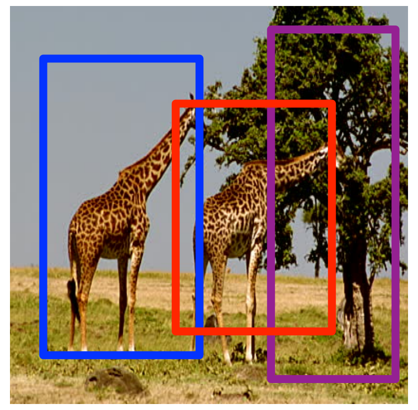
Image Generation from Layout: **Challenges**



- One-to-many mapping
- Information in layout is limited (but important)
- Important interactions between objects in overlap regions and with scene

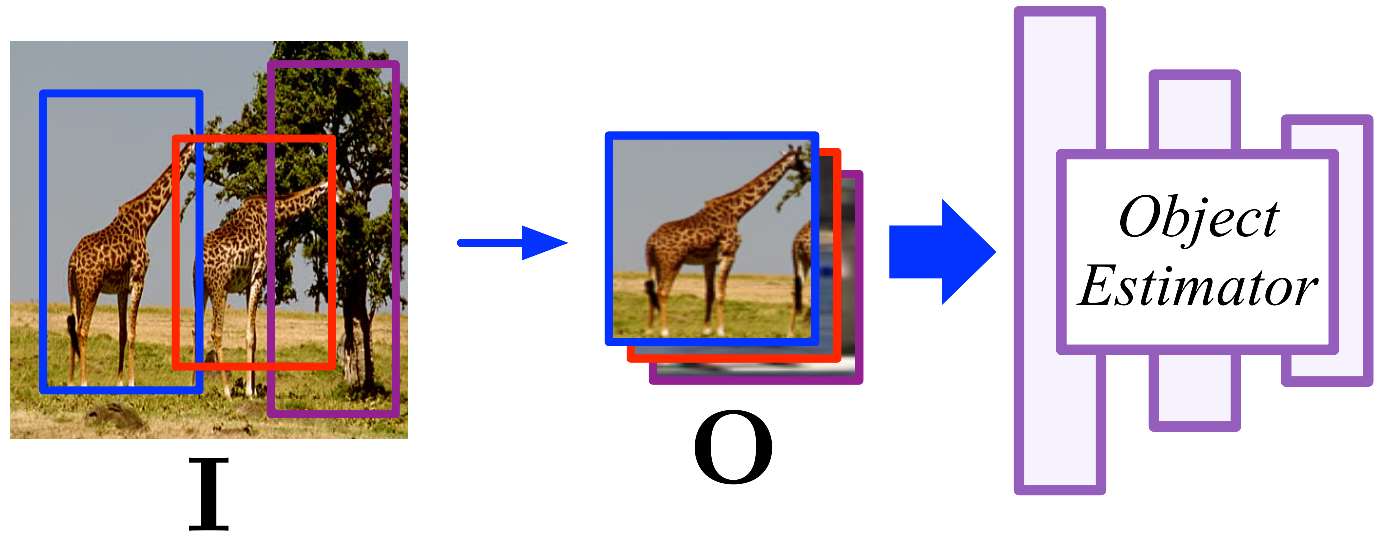
Model **Architecture**: Training

Model **Architecture**: Training

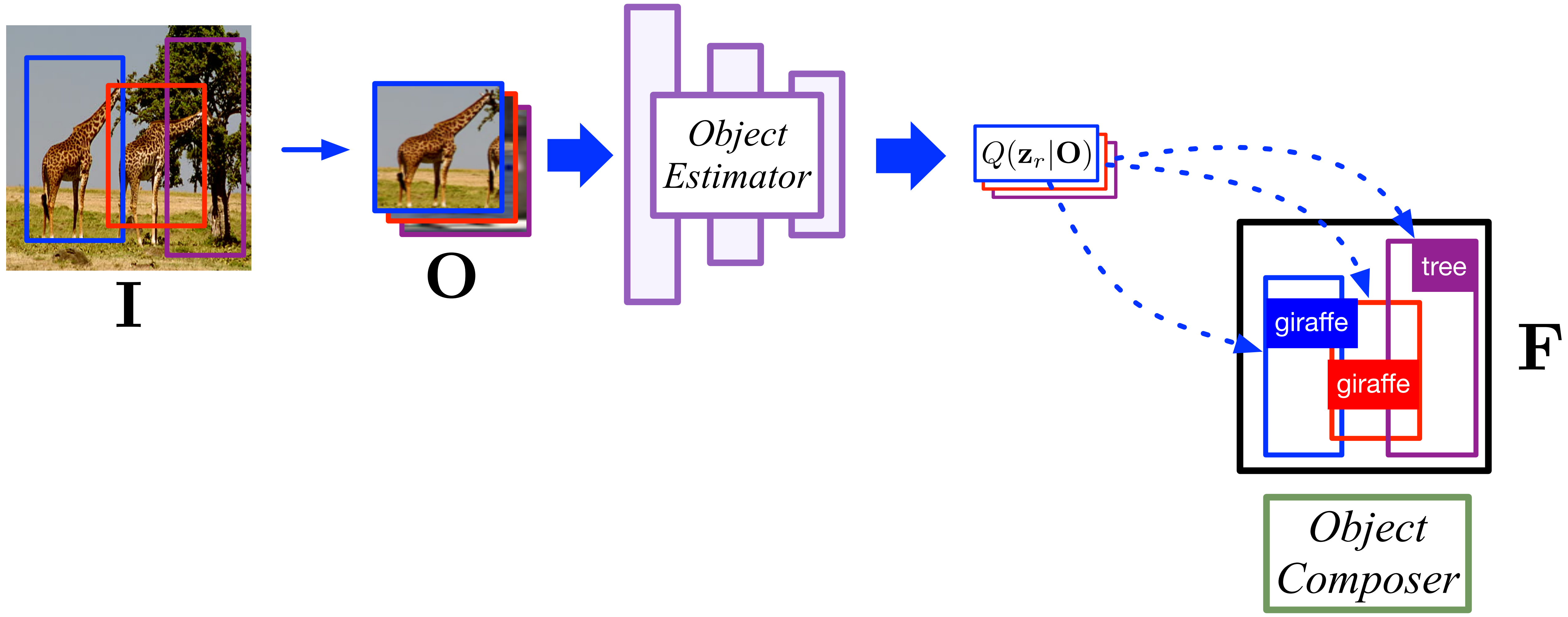


I

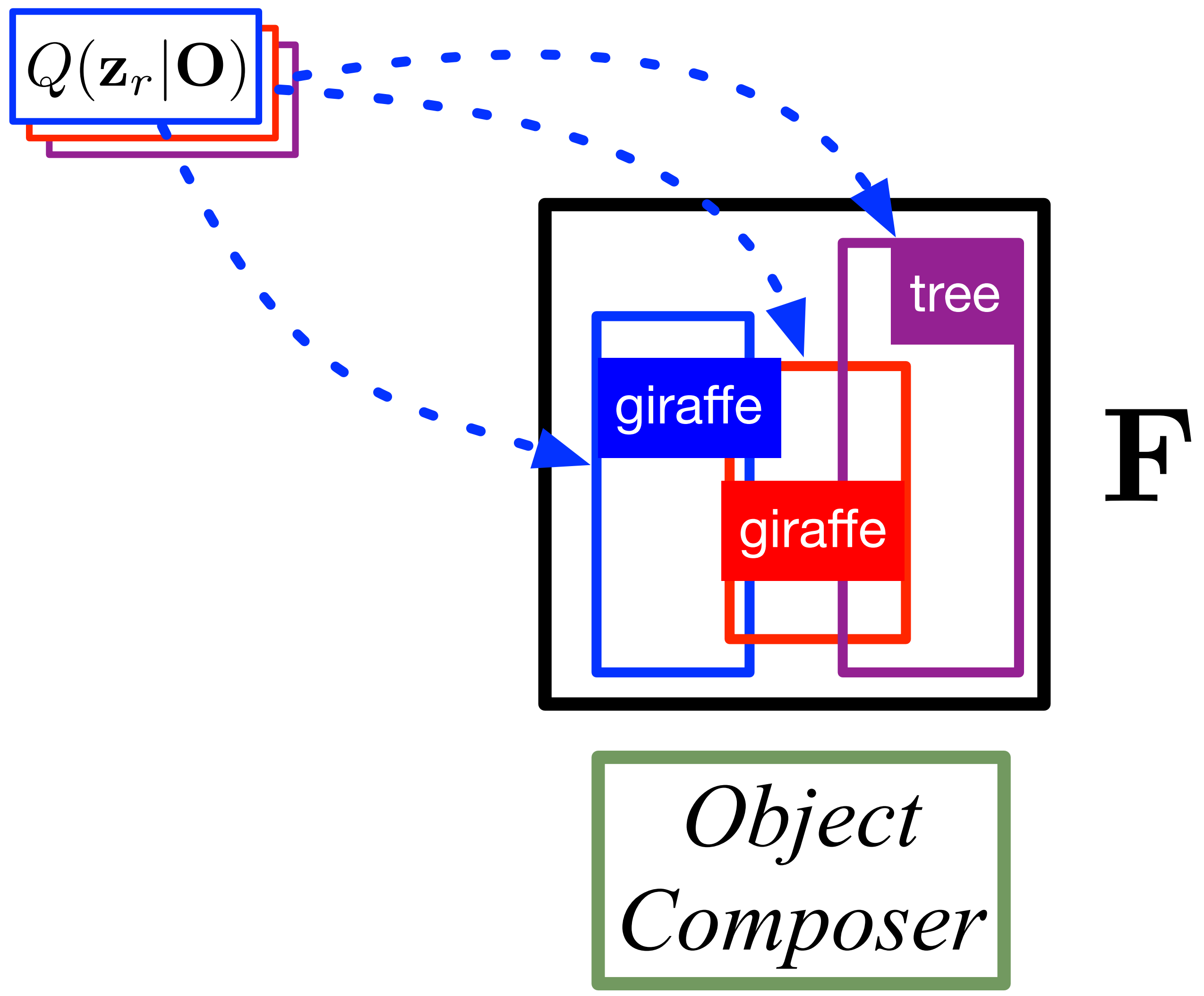
Model **Architecture**: Training



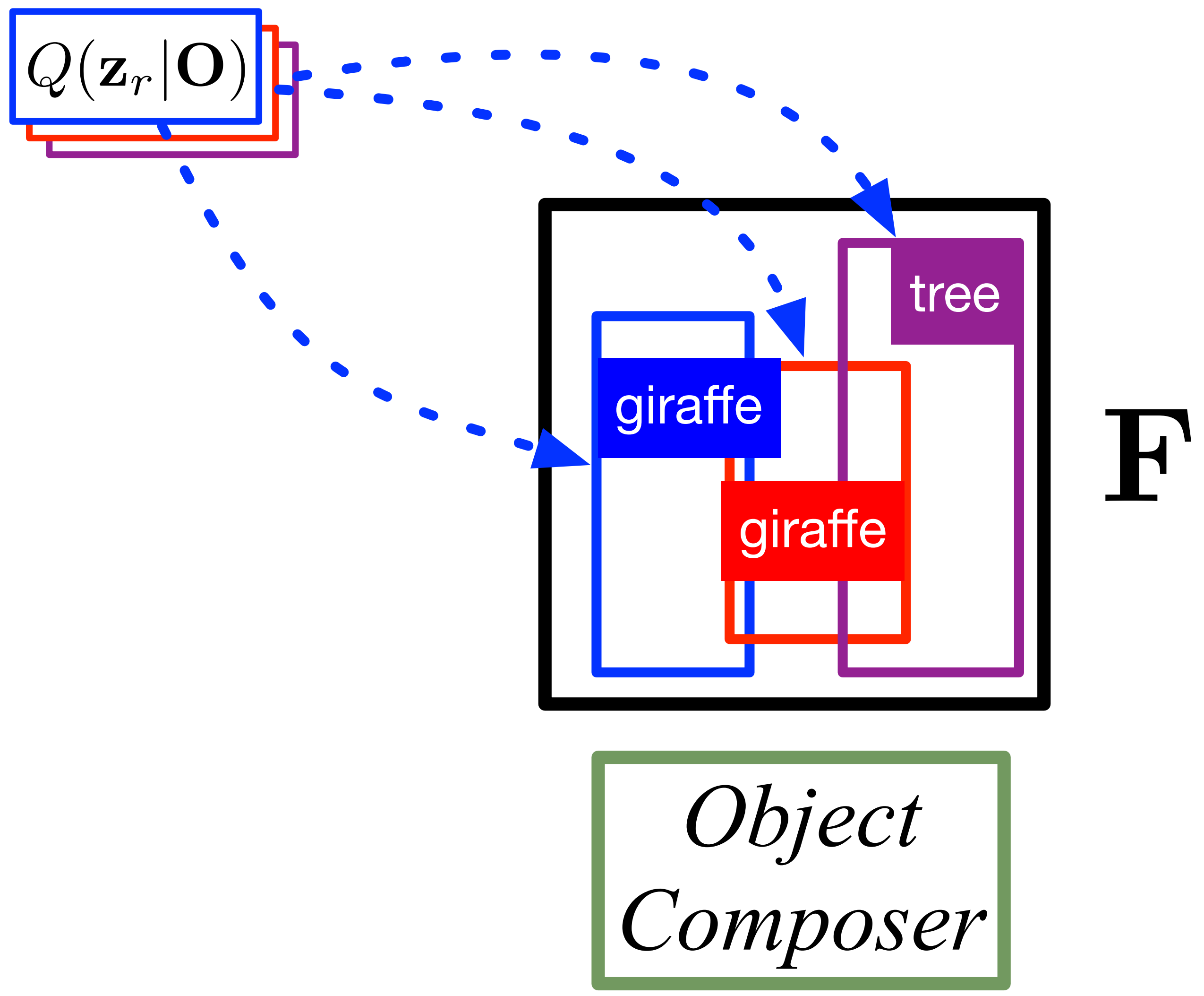
Model **Architecture**: Training

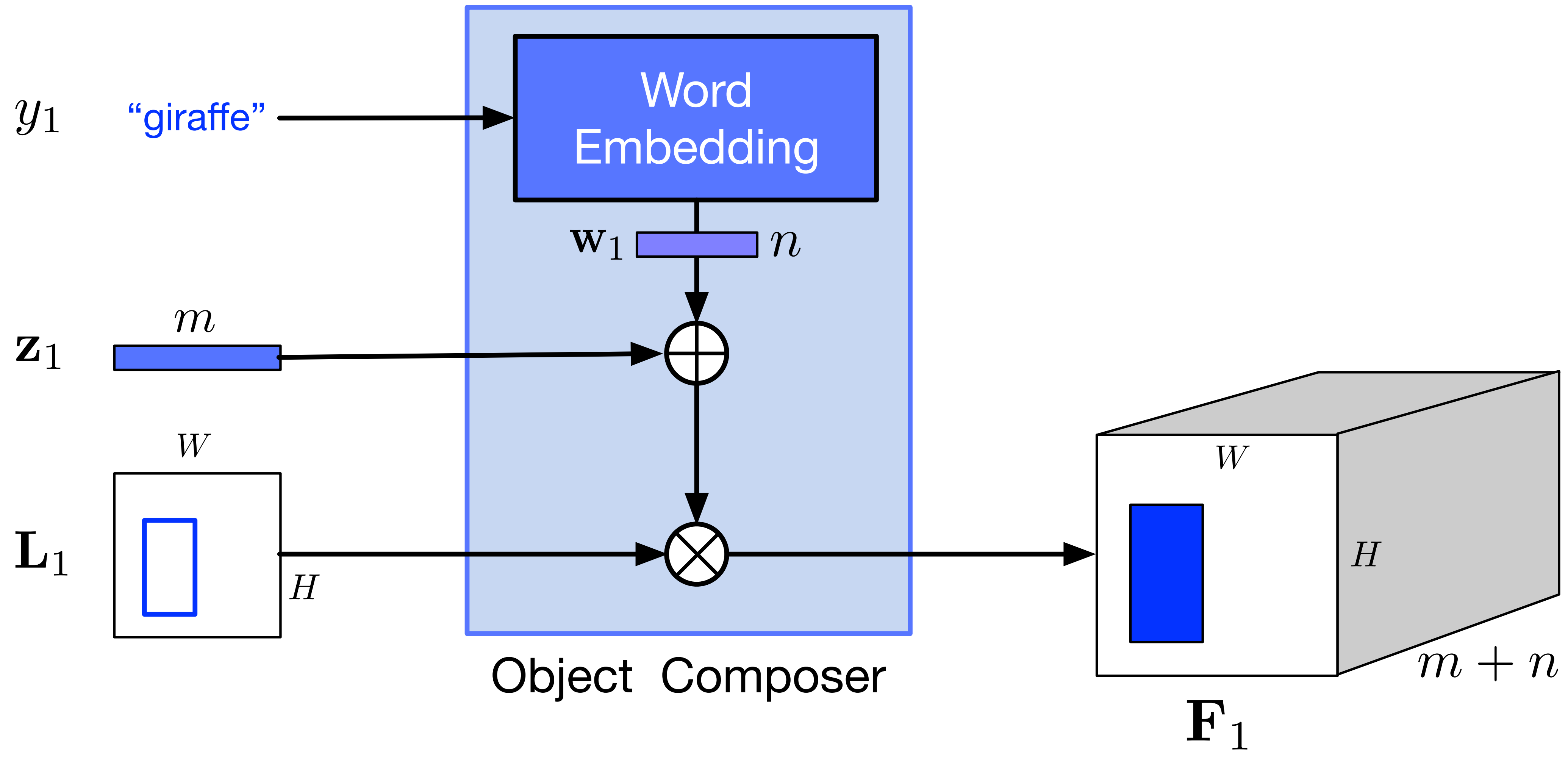


Model **Architecture**: Training

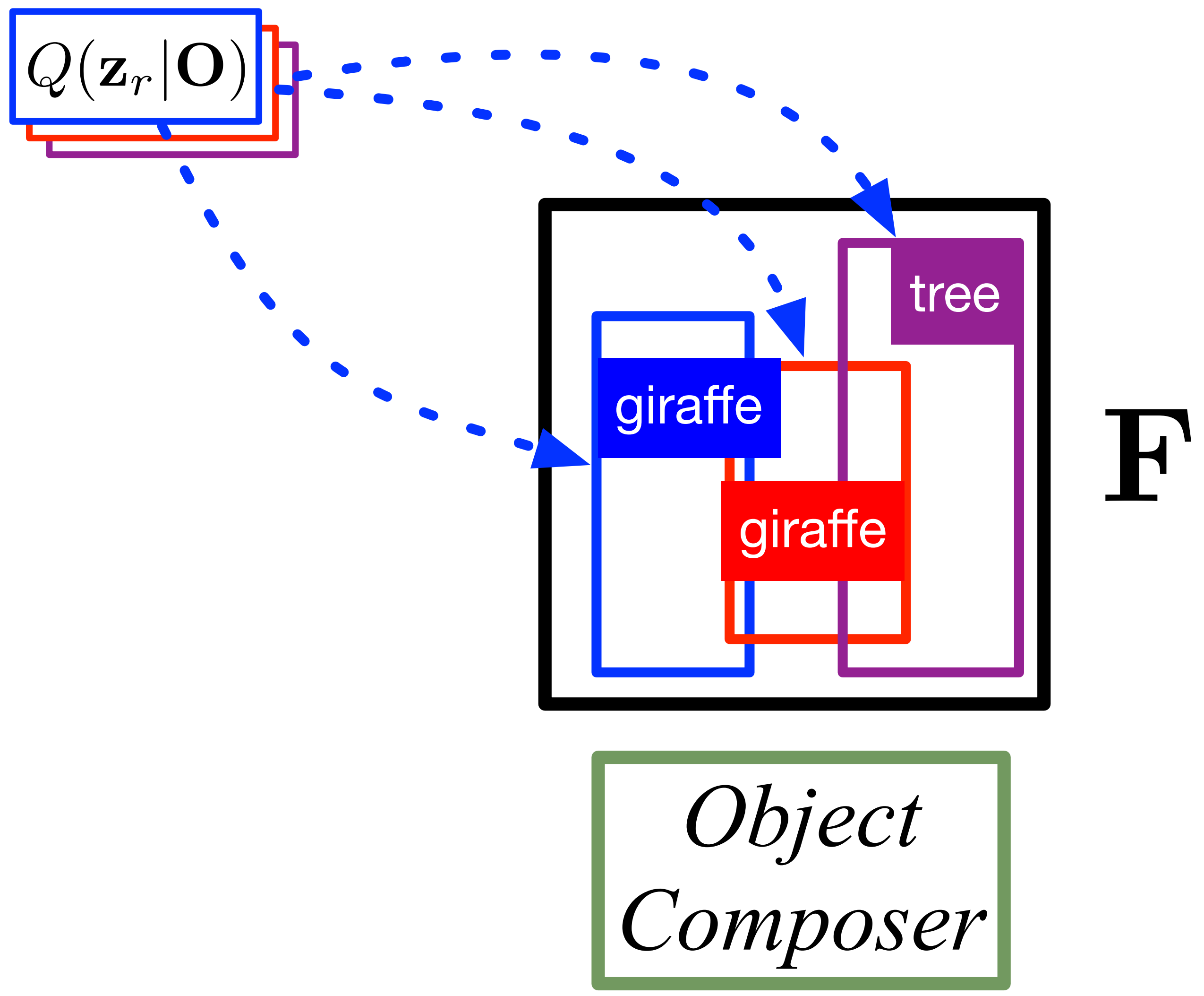


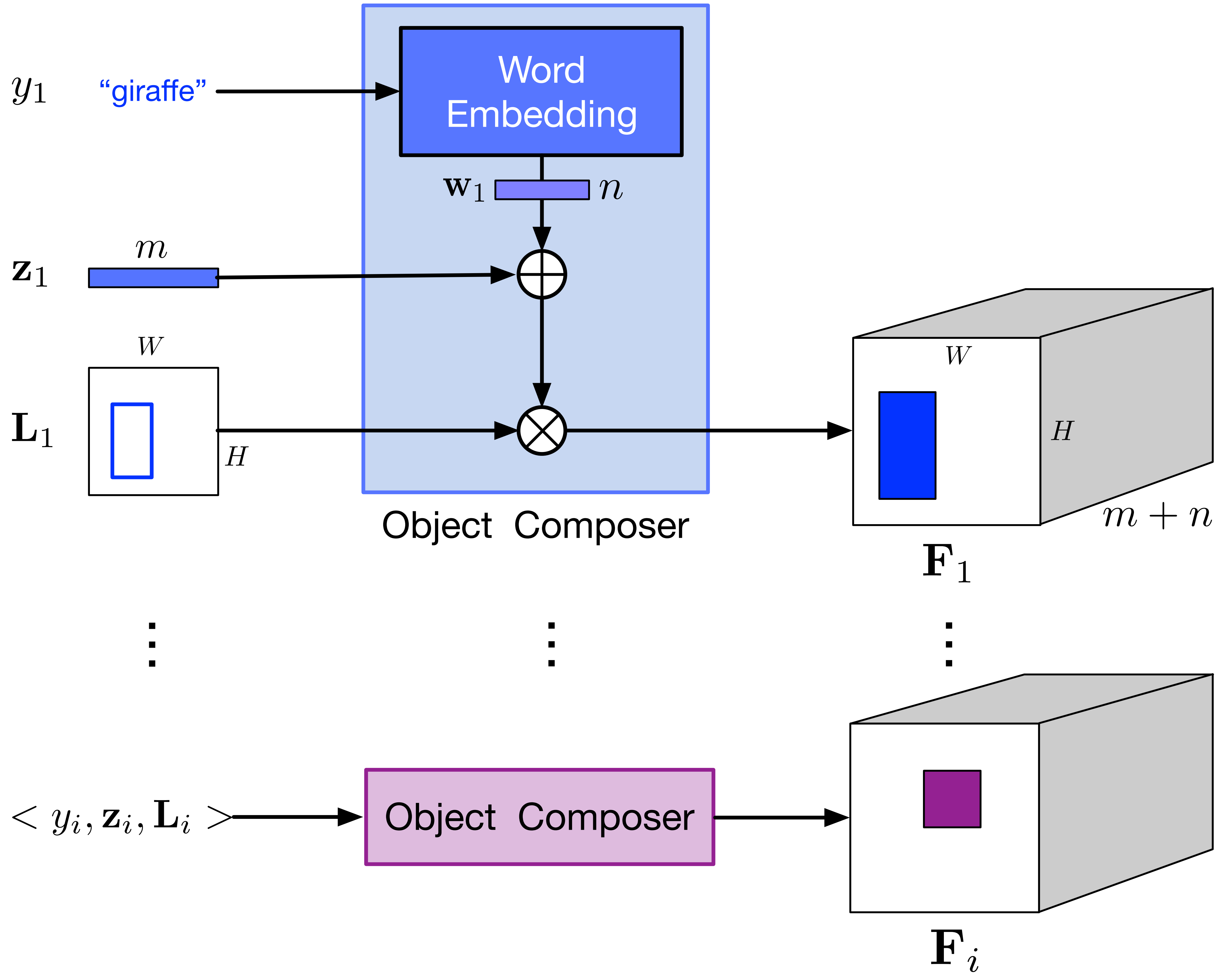
Model **Architecture**: Training



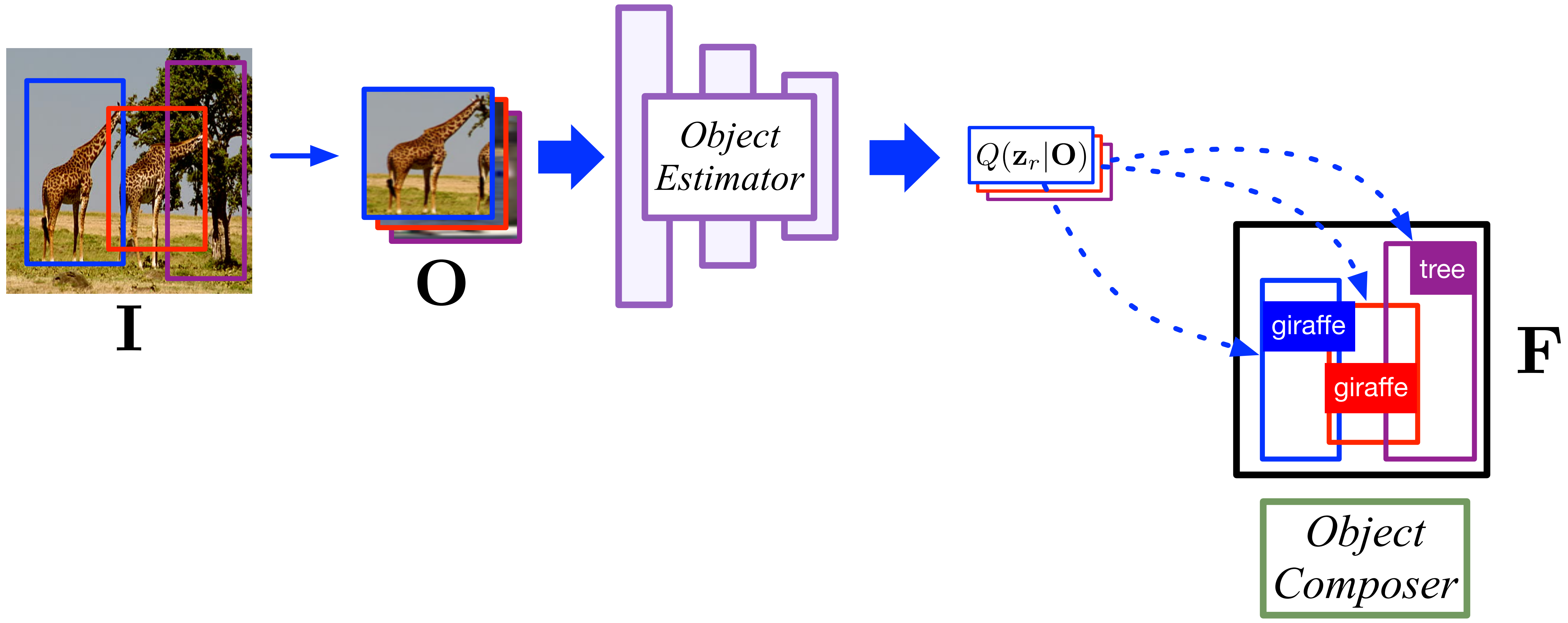


Model **Architecture**: Training

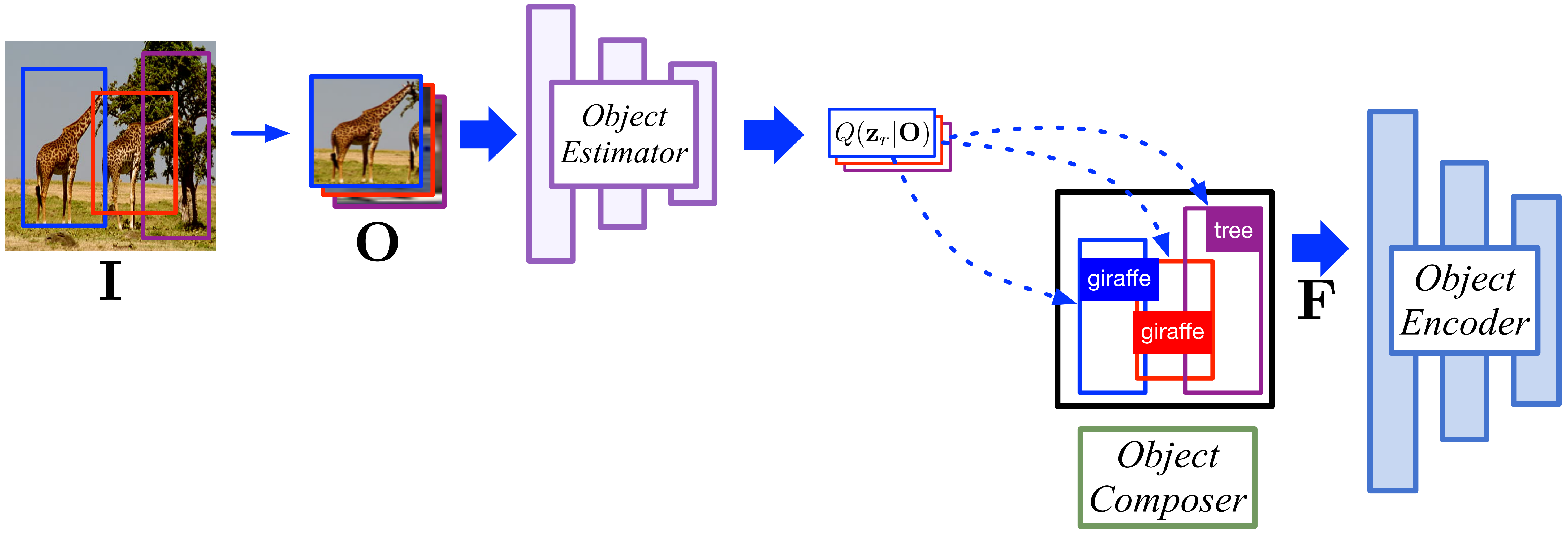




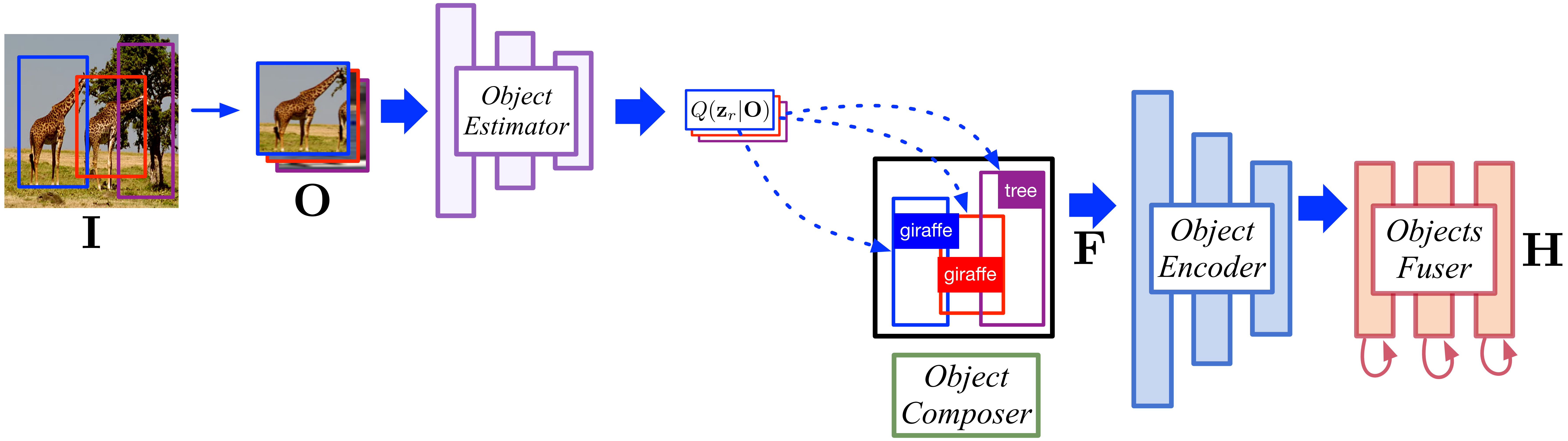
Model **Architecture**: Training



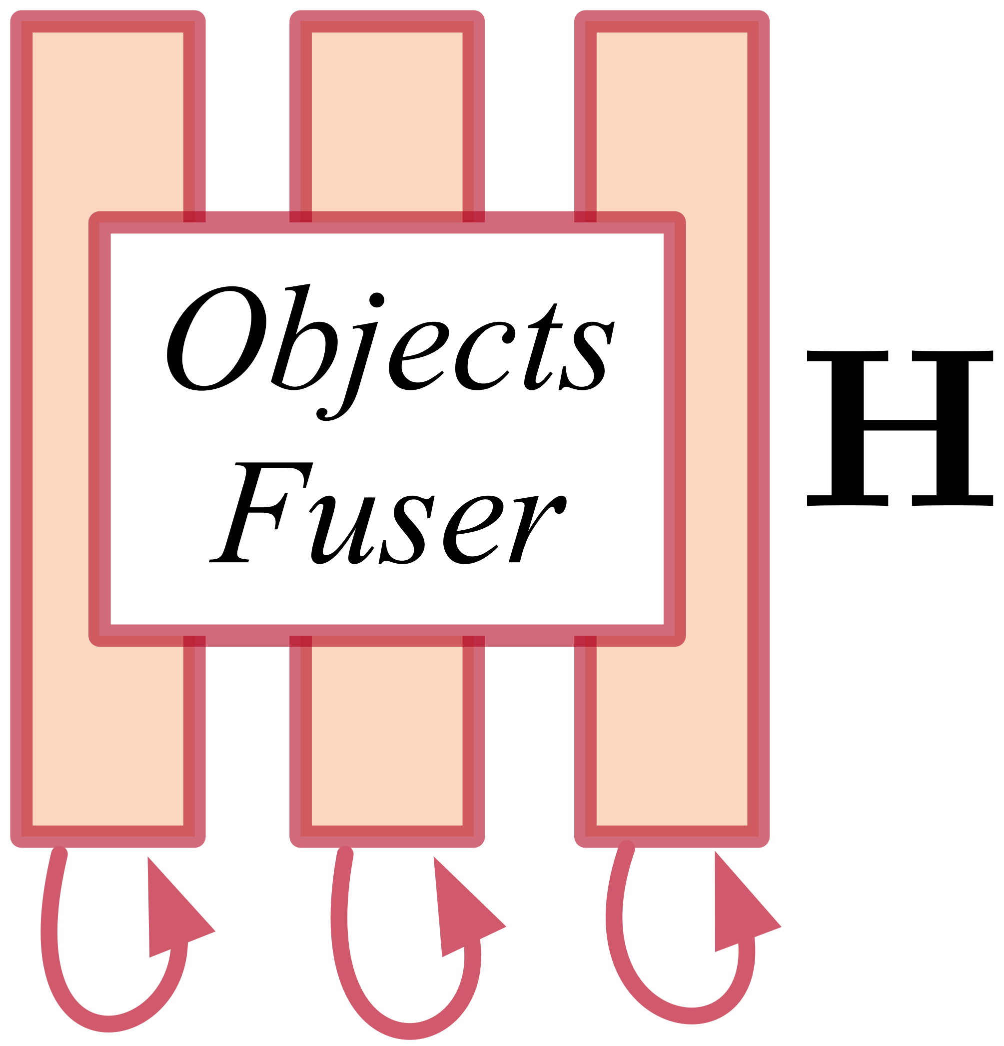
Model **Architecture**: Training

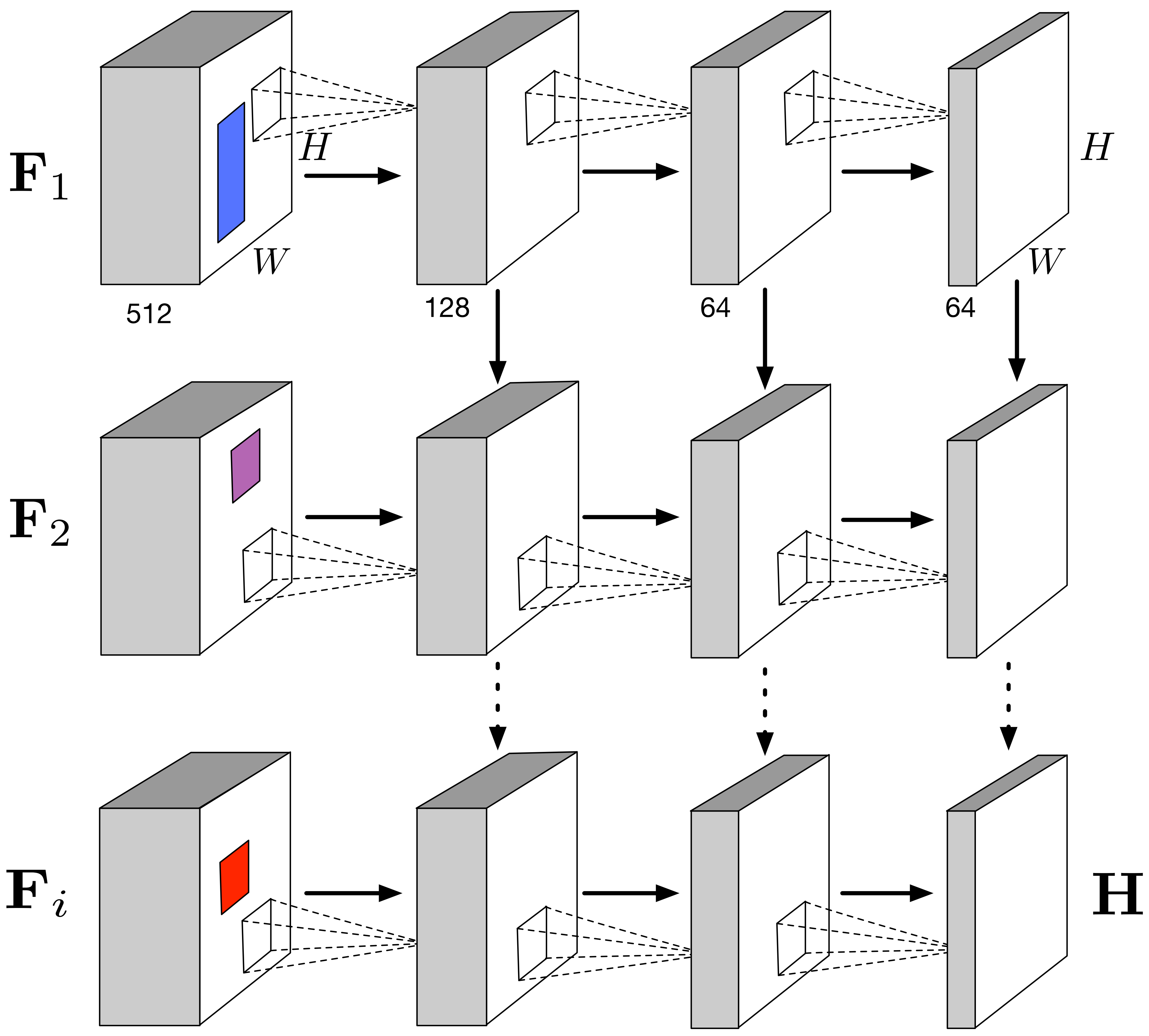


Model **Architecture**: Training

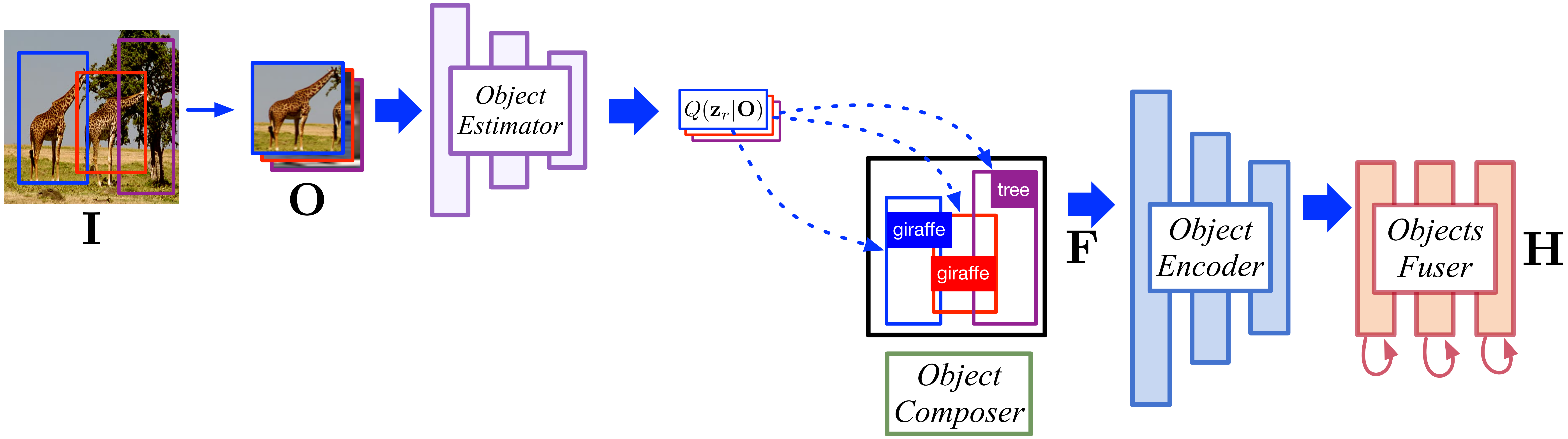


Model **Architecture**: Training

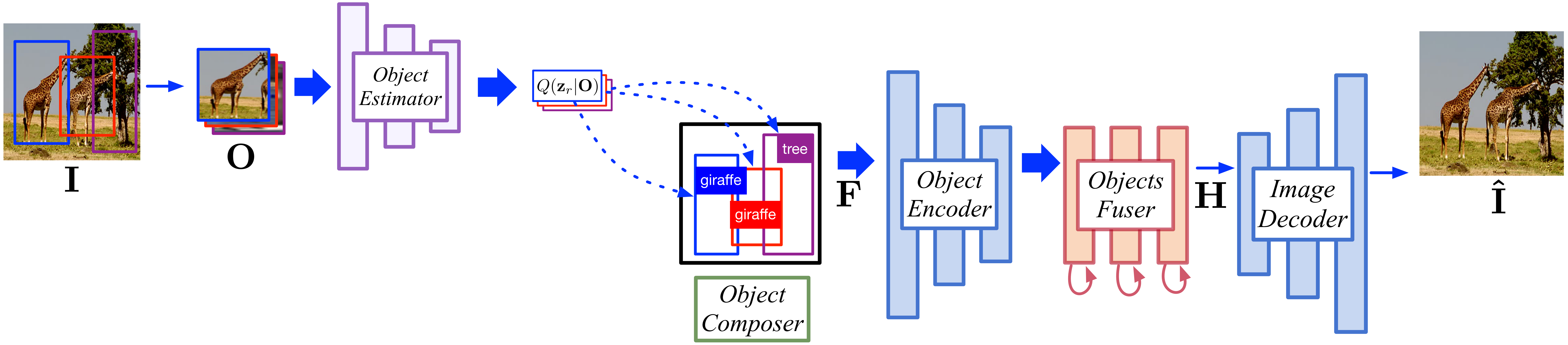




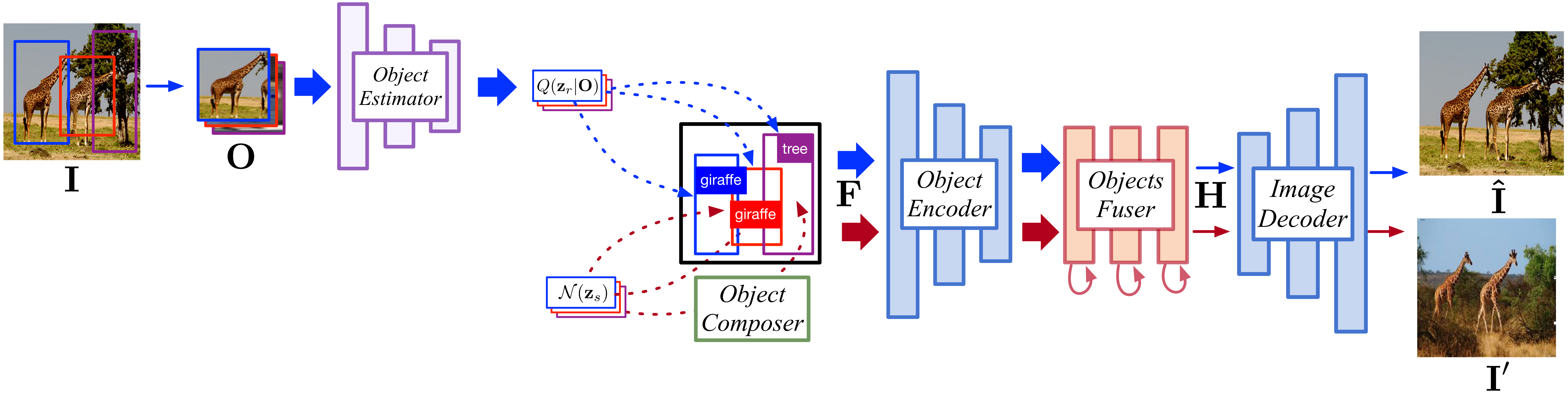
Model **Architecture**: Training



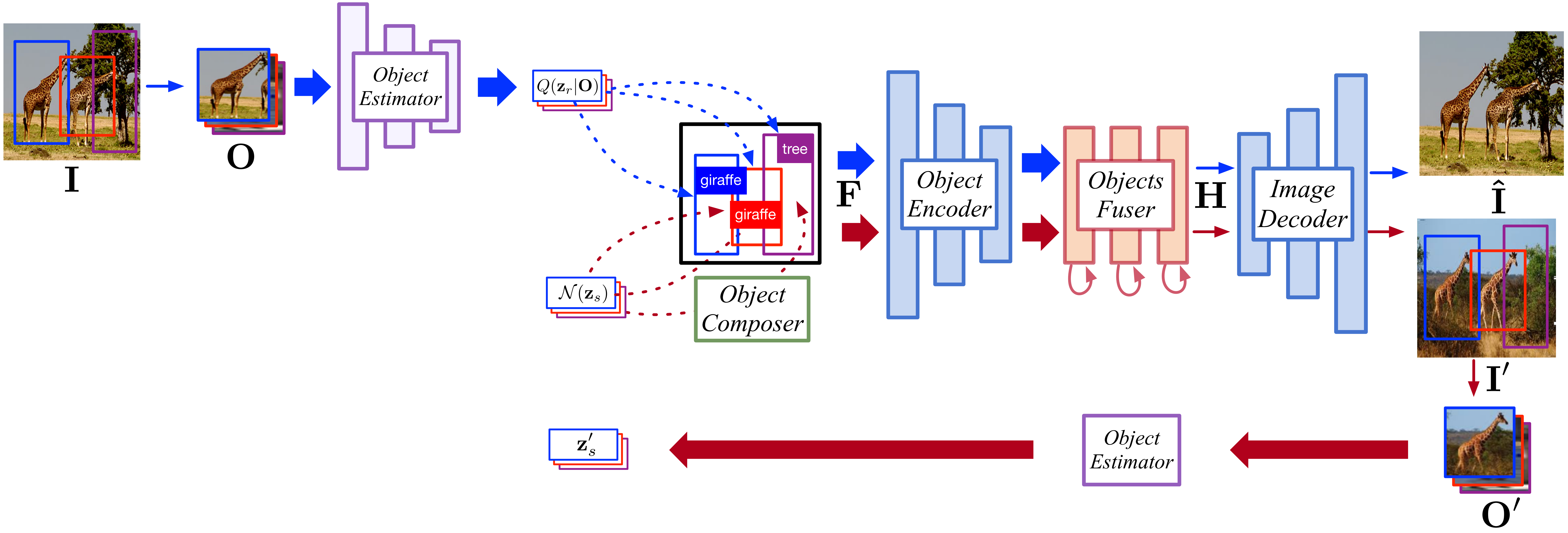
Model **Architecture**: Training



Model **Architecture**: Training

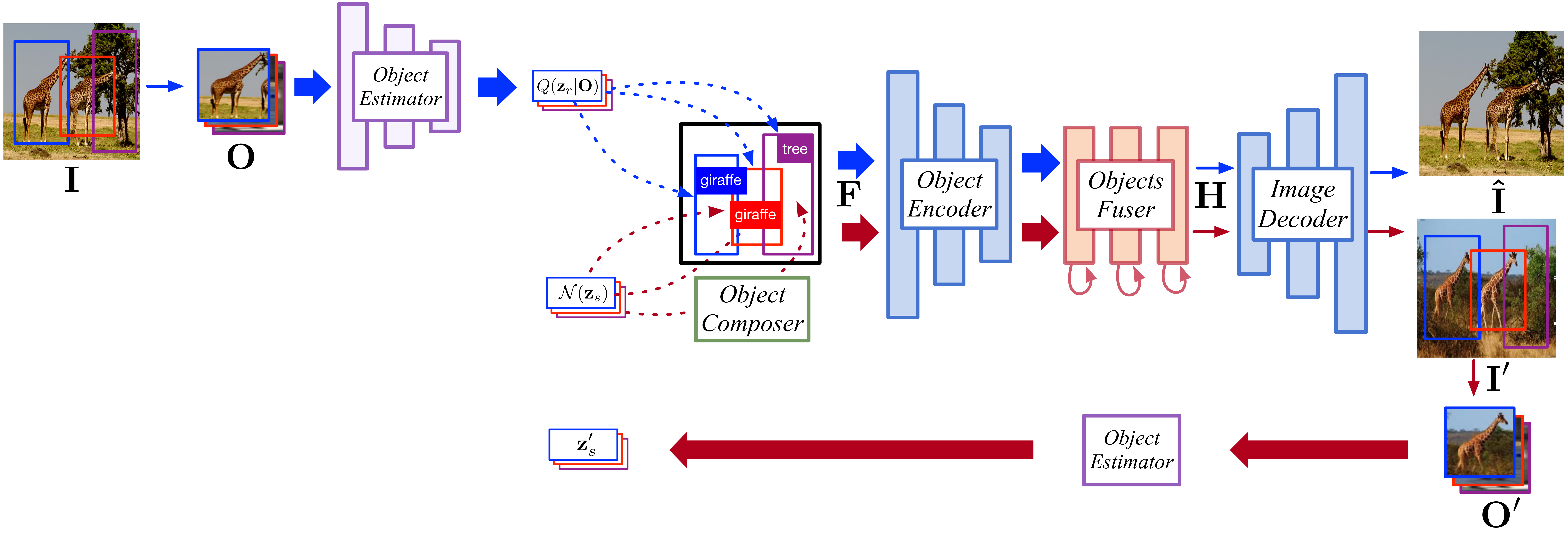


Model **Architecture**: Training



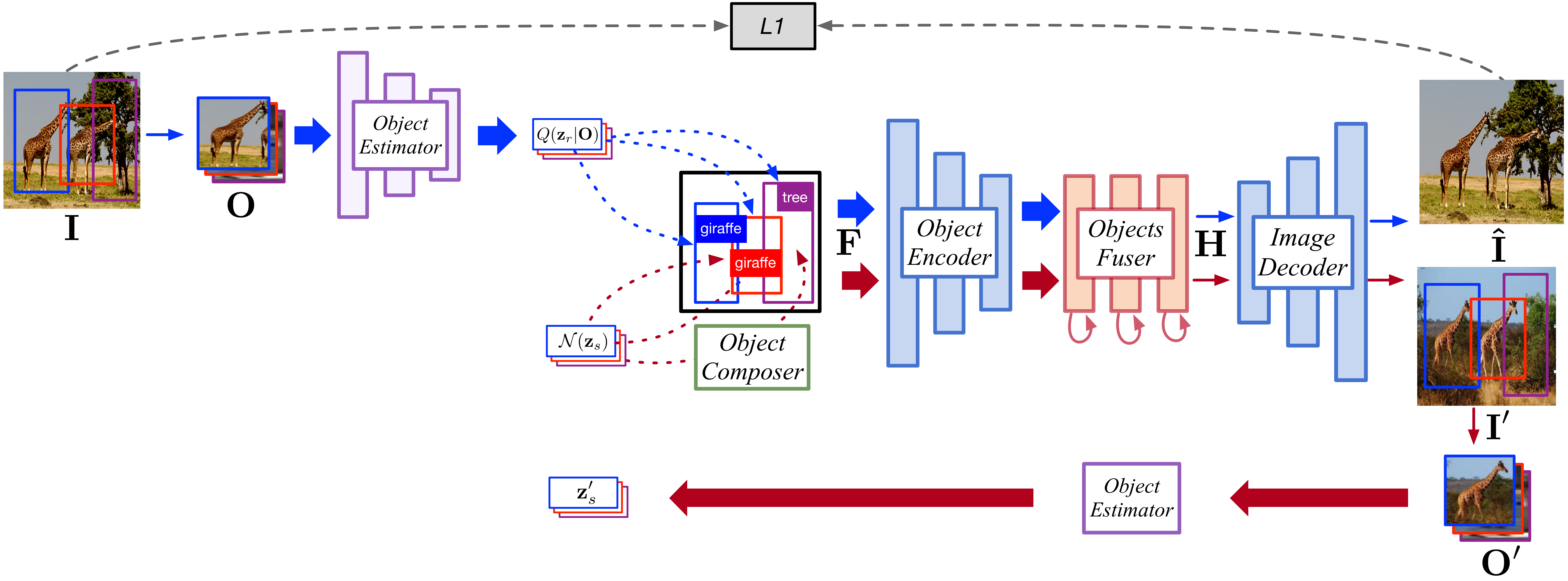
Model **Architecture**: Training

Losses



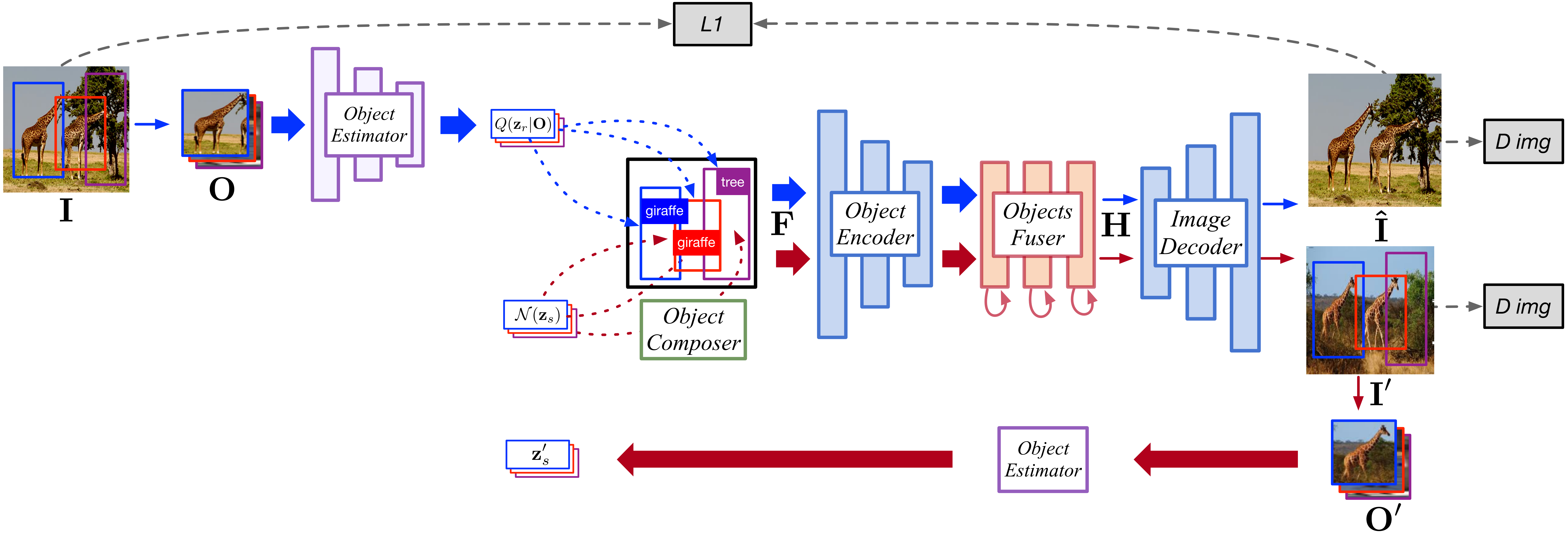
Model **Architecture**: Training

Losses



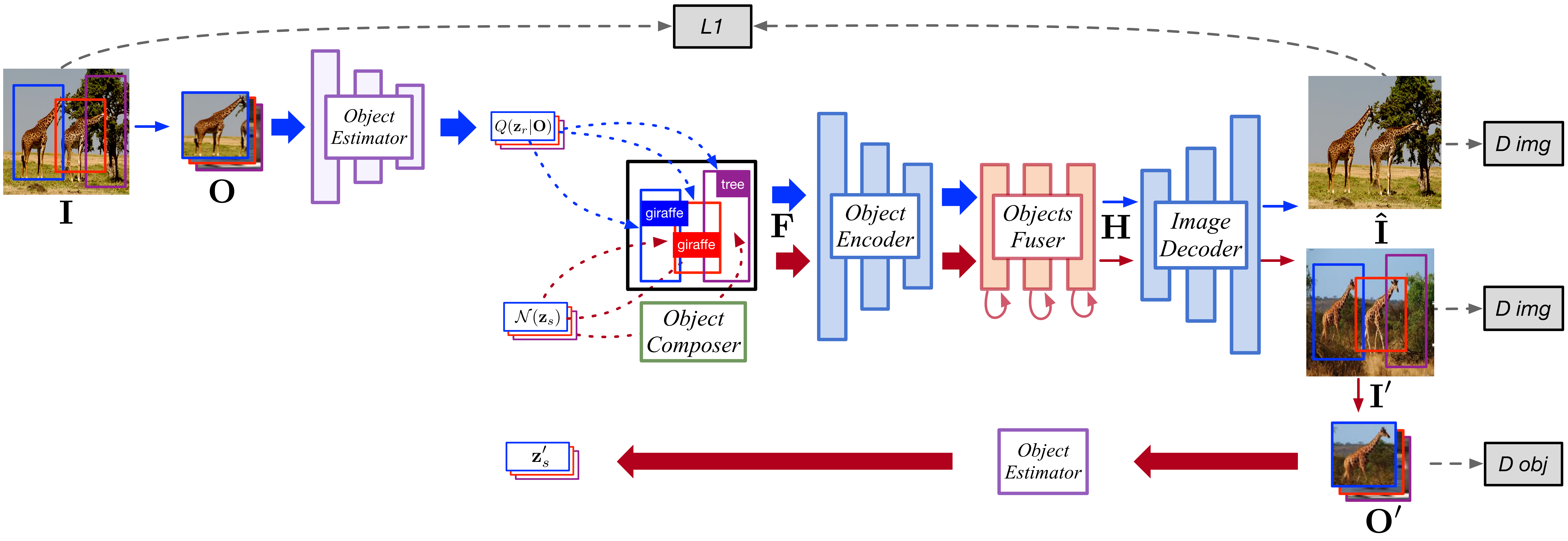
Model **Architecture**: Training

Losses



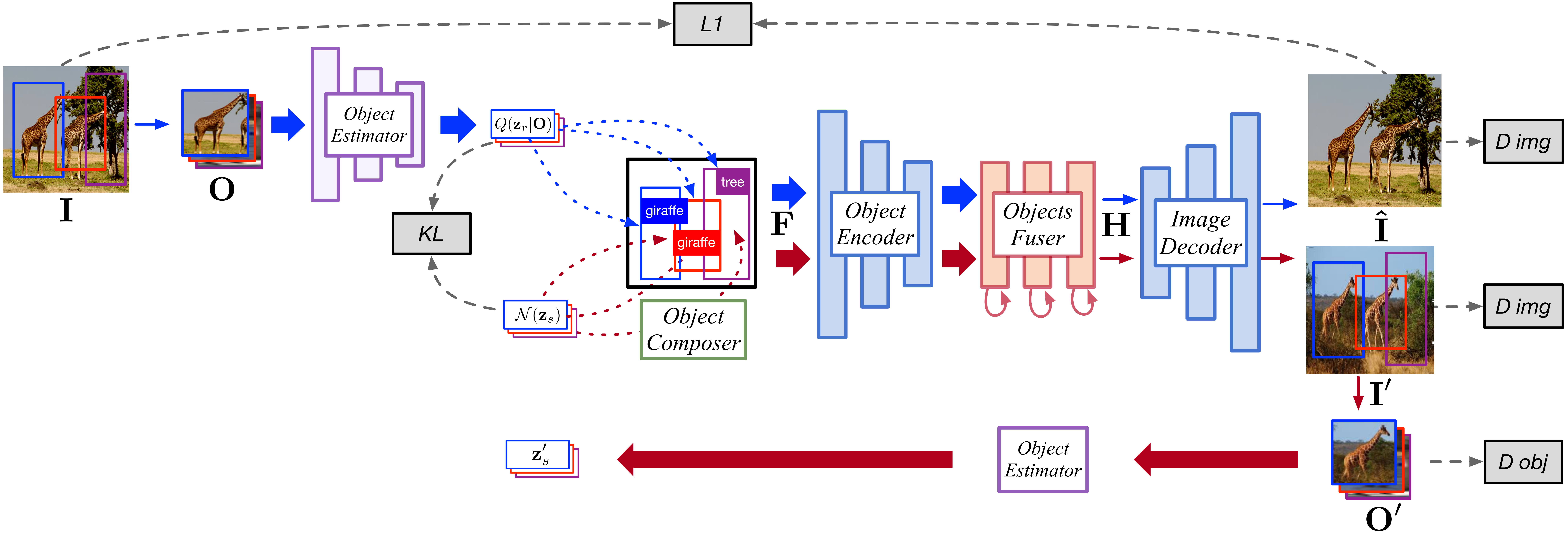
Model **Architecture**: Training

Losses



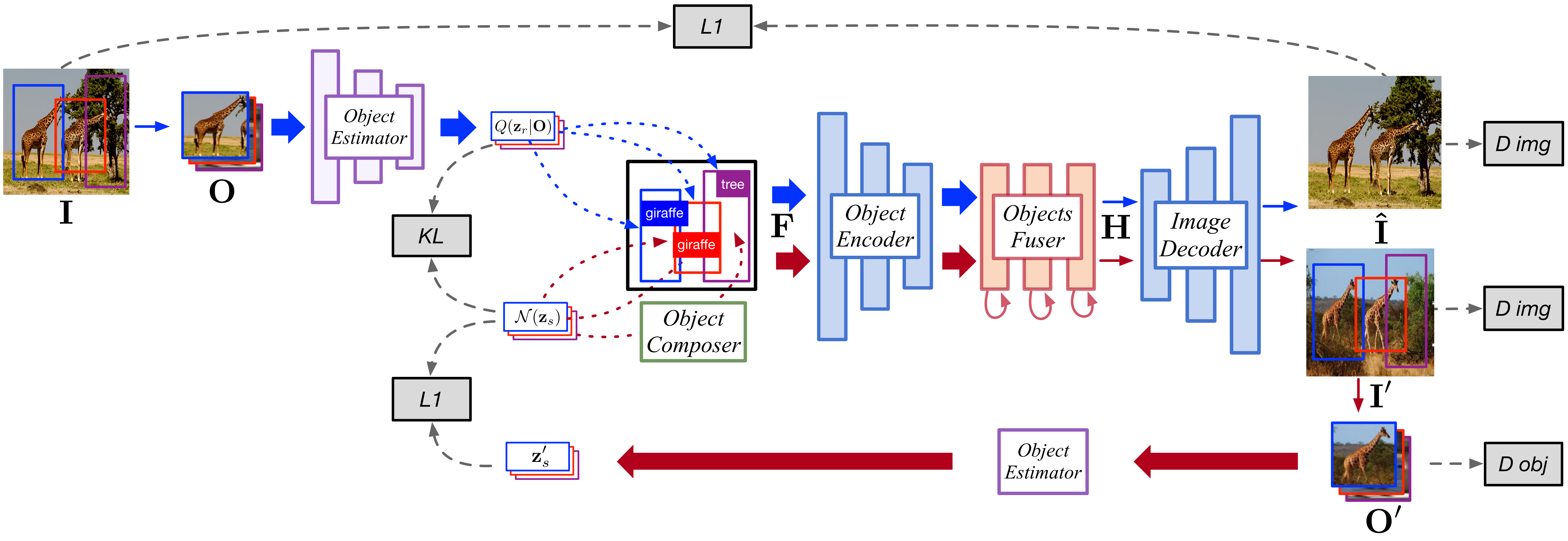
Model **Architecture**: Training

Losses



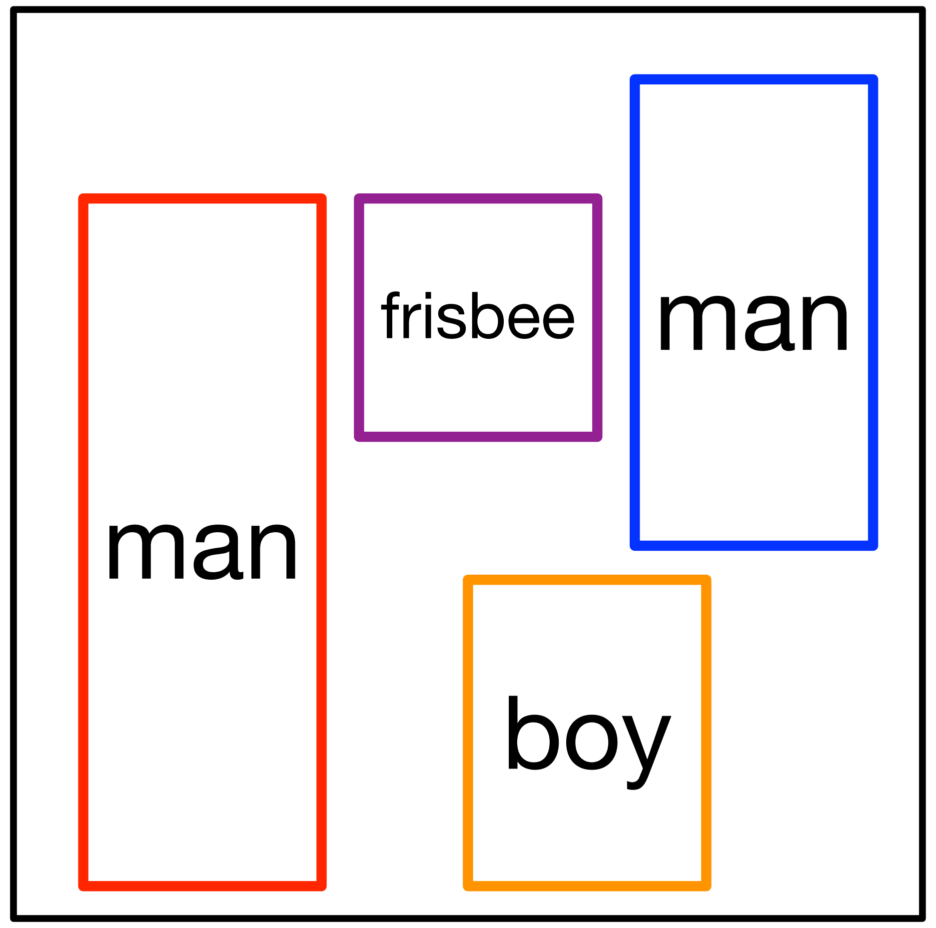
Model **Architecture**: Training

Losses

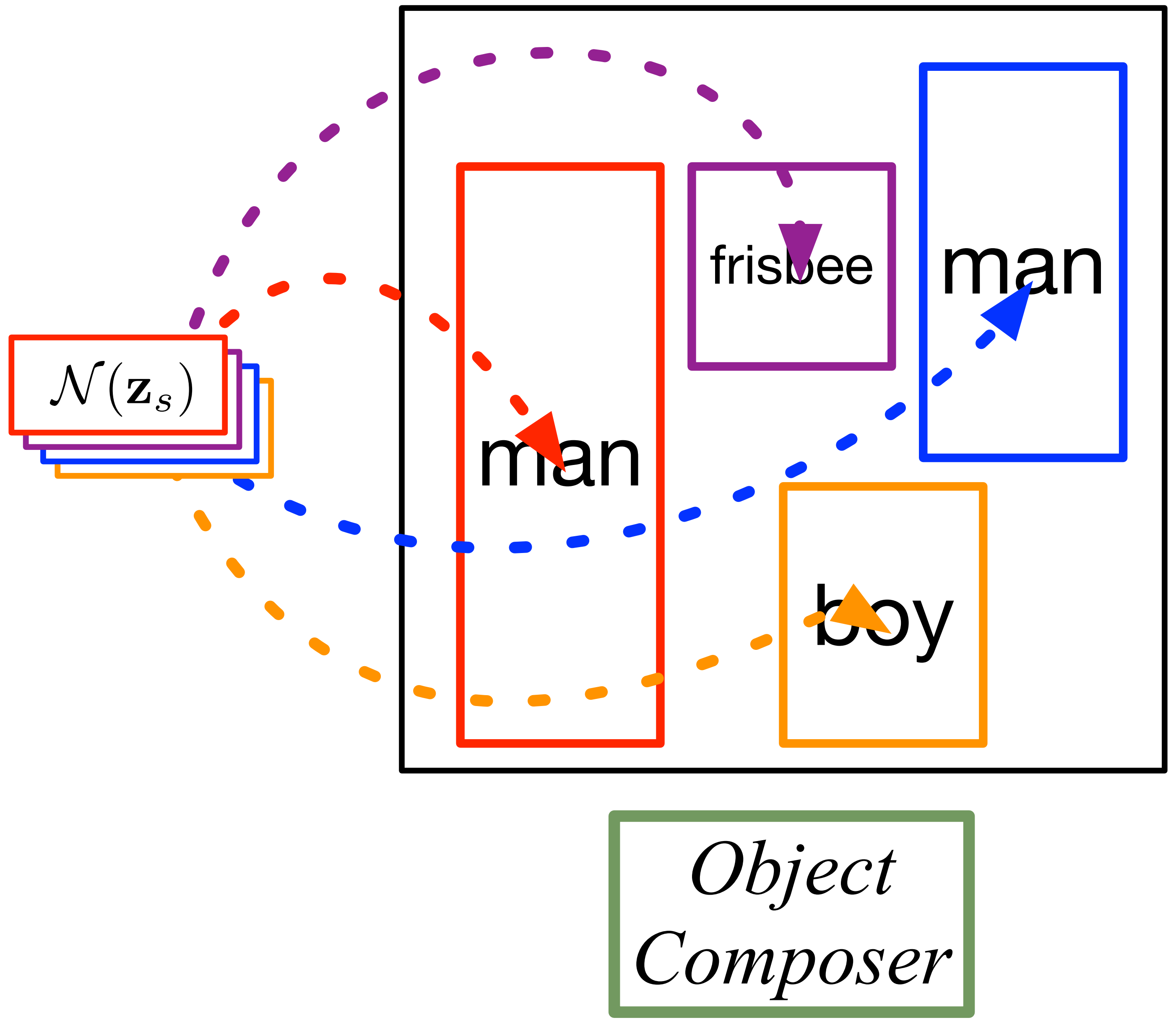


Model **Architecture**: Runtime

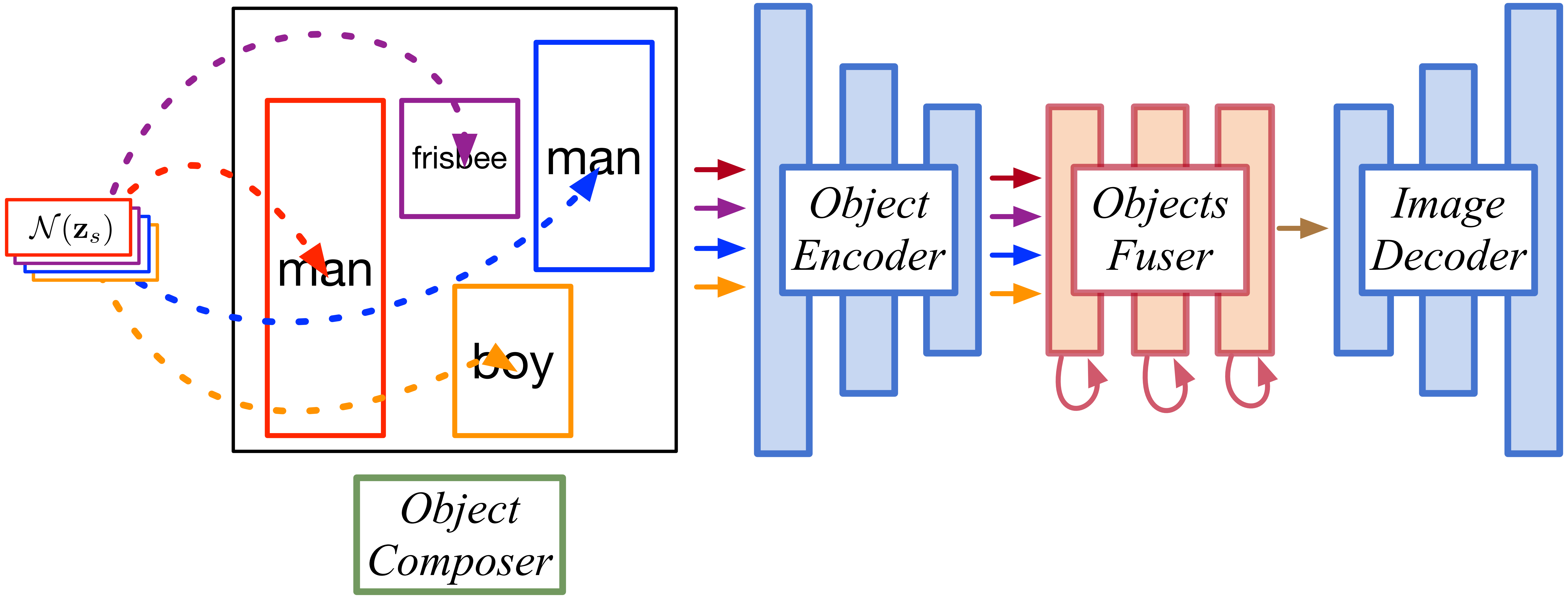
Model **Architecture**: Runtime



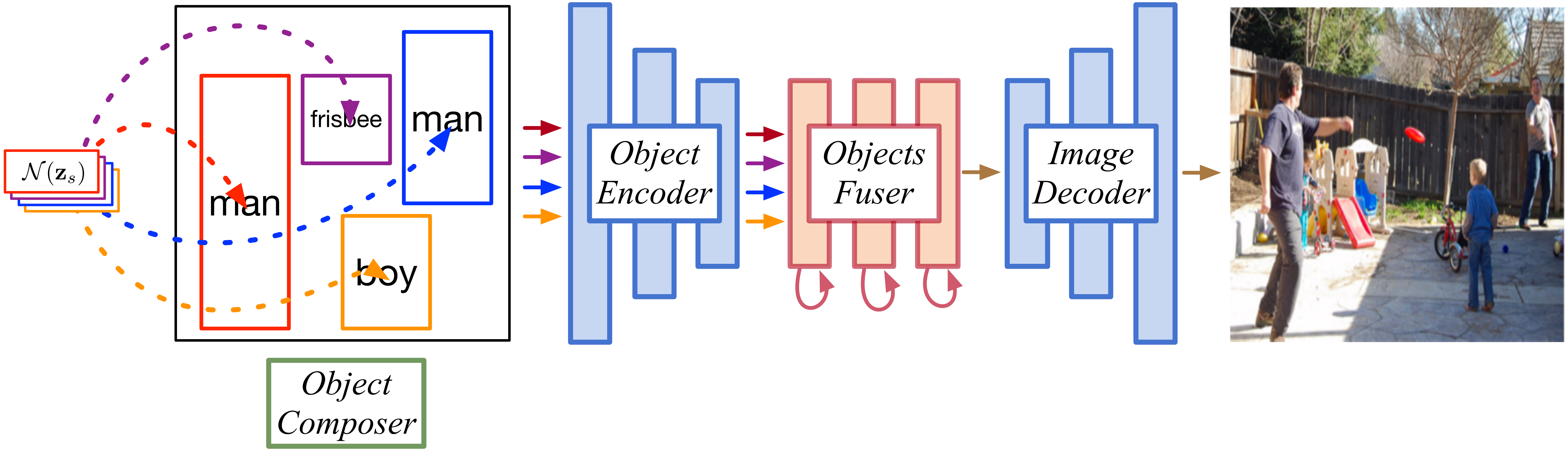
Model **Architecture**: Runtime



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	$3 \sim 30$

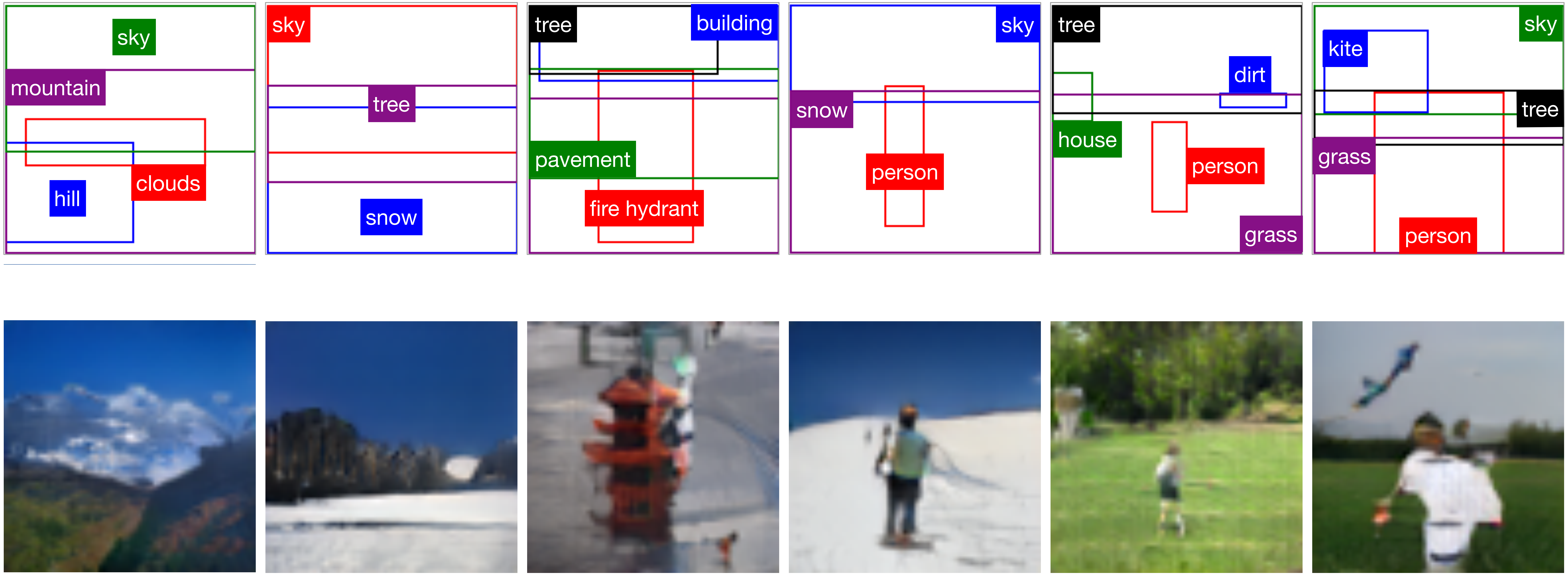
Evaluation:

Method	**Inception** Score		Object **Classification** Score		**Diversity** Score	
	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**	$\mathbf{0.15 \pm 0.06}$	$\mathbf{0.17 \pm 0.09}$

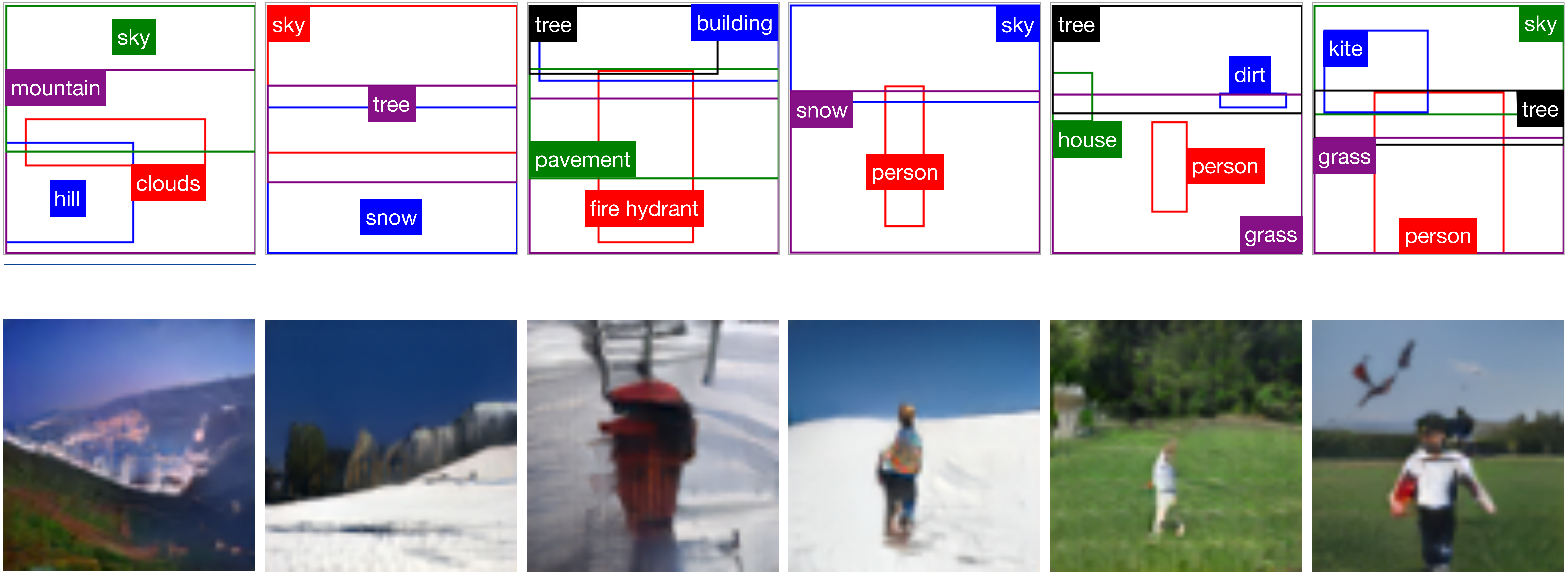
Results on COCO

Results on Visual Genome

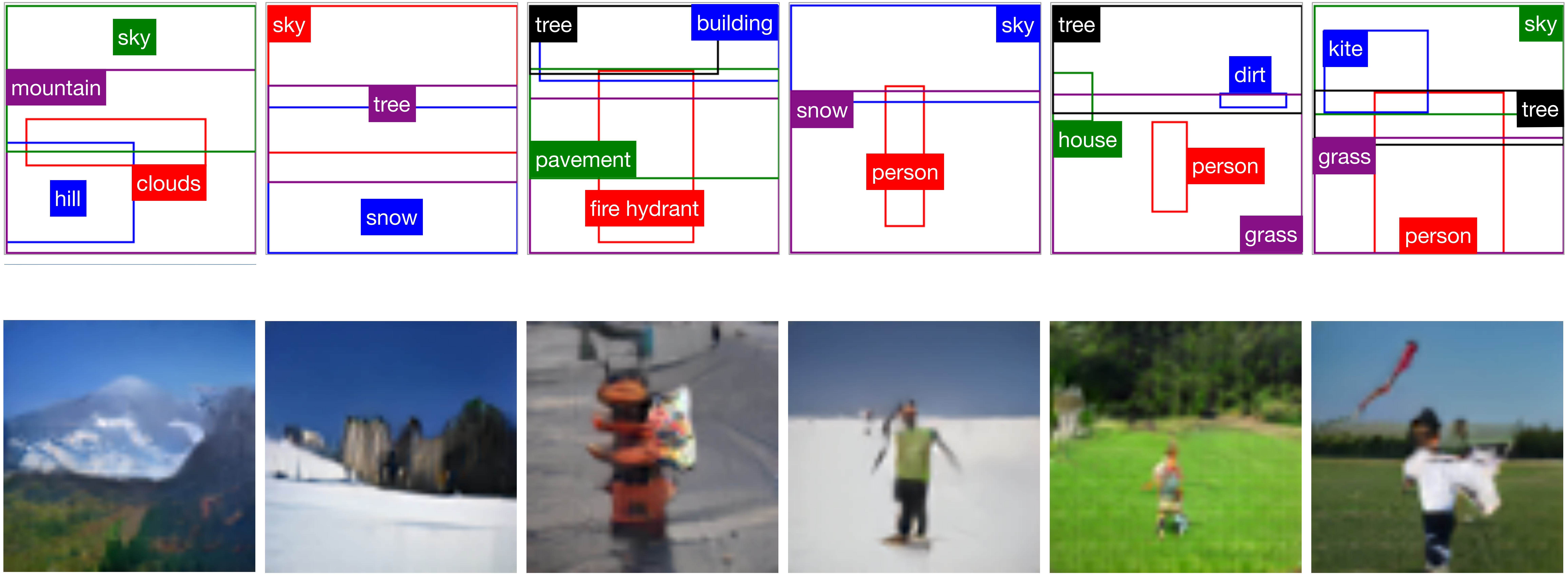
Results: Diversity



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

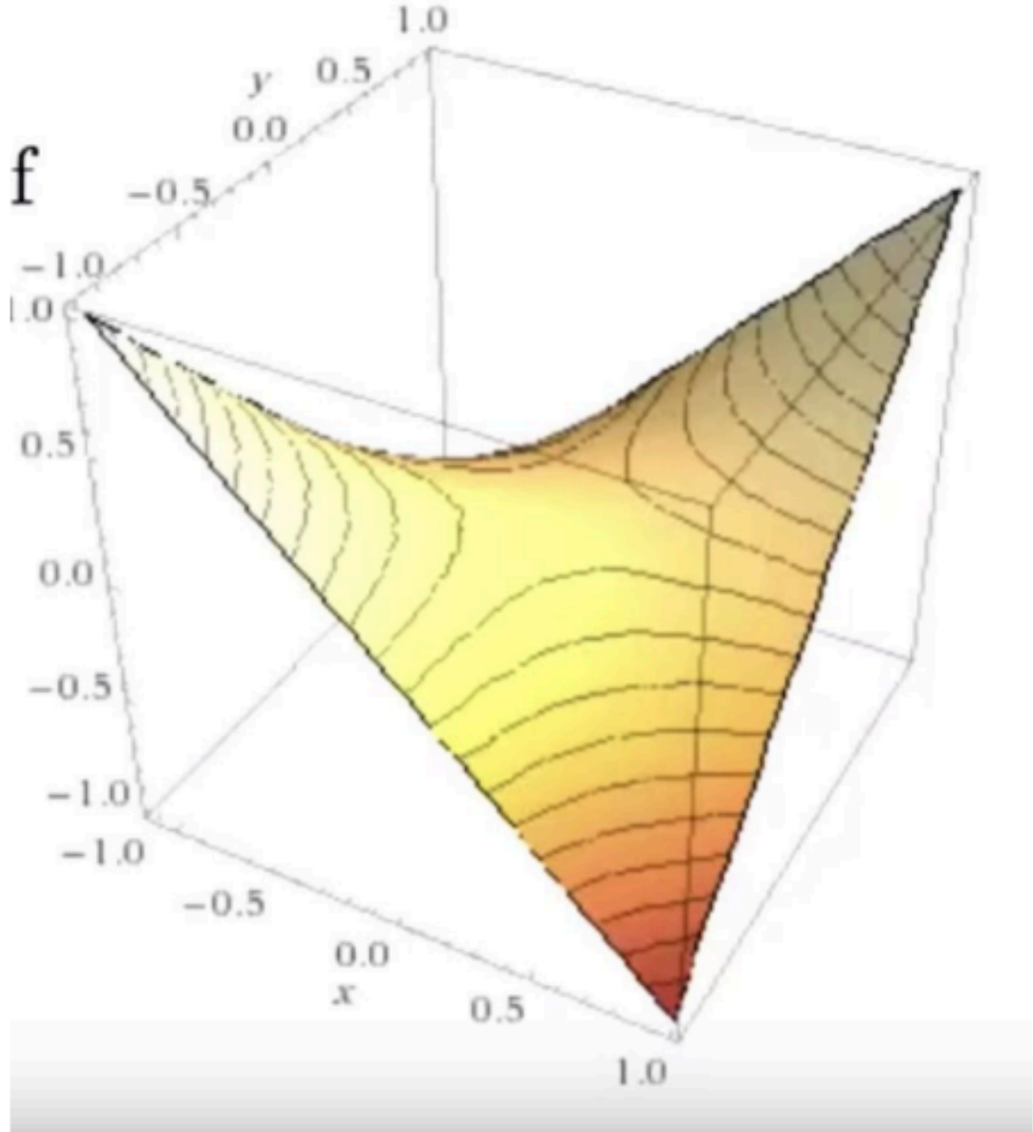
Non-Convergence

D & G nullifies each others learning in every iteration

Train for a long time – without generating good quality samples

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

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



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

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

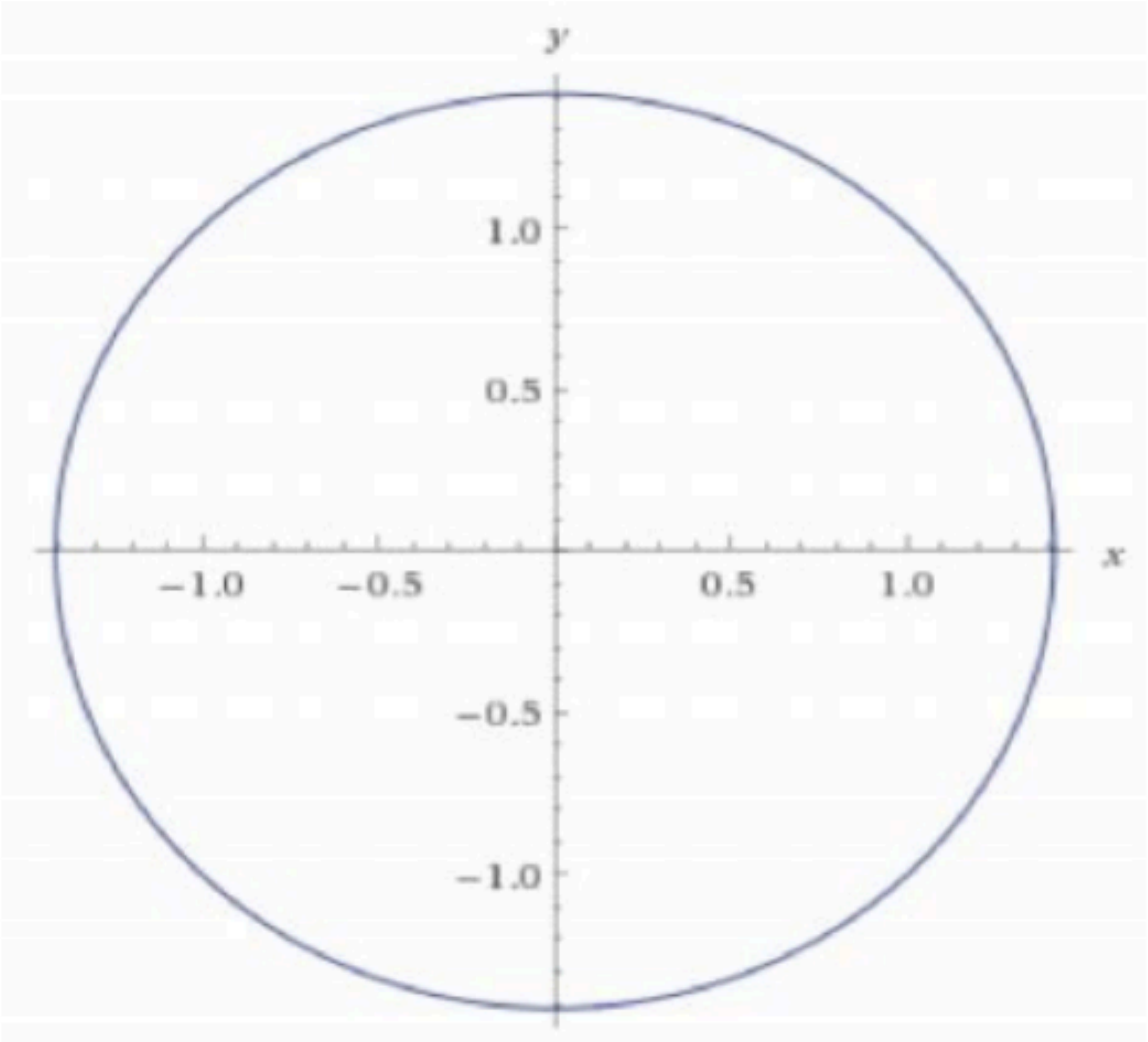
$$\frac{\partial y}{\partial t} = x(t)$$

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

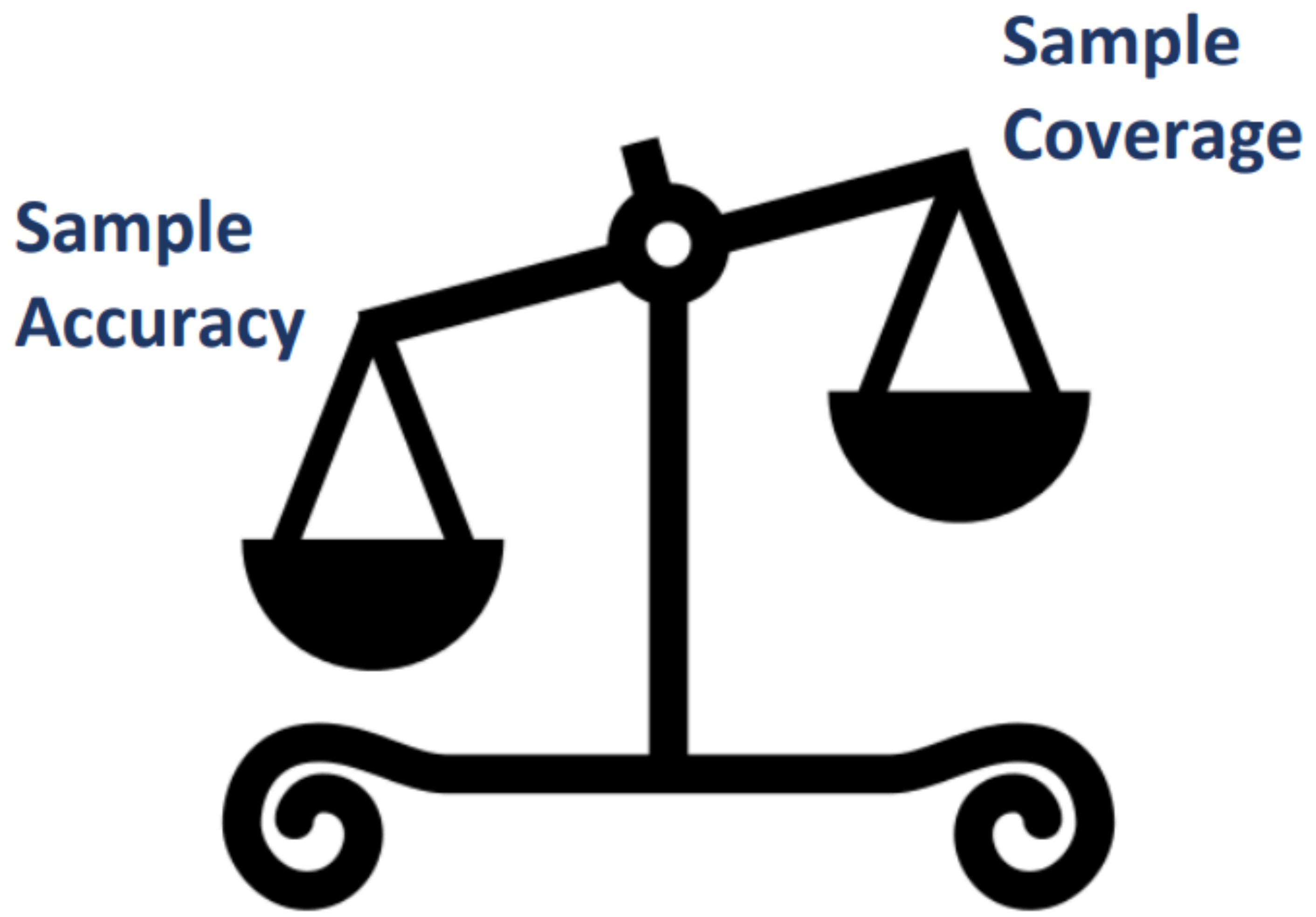
$$x(t) = x(0)cost(t) - y(0)\sin(t)$$

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

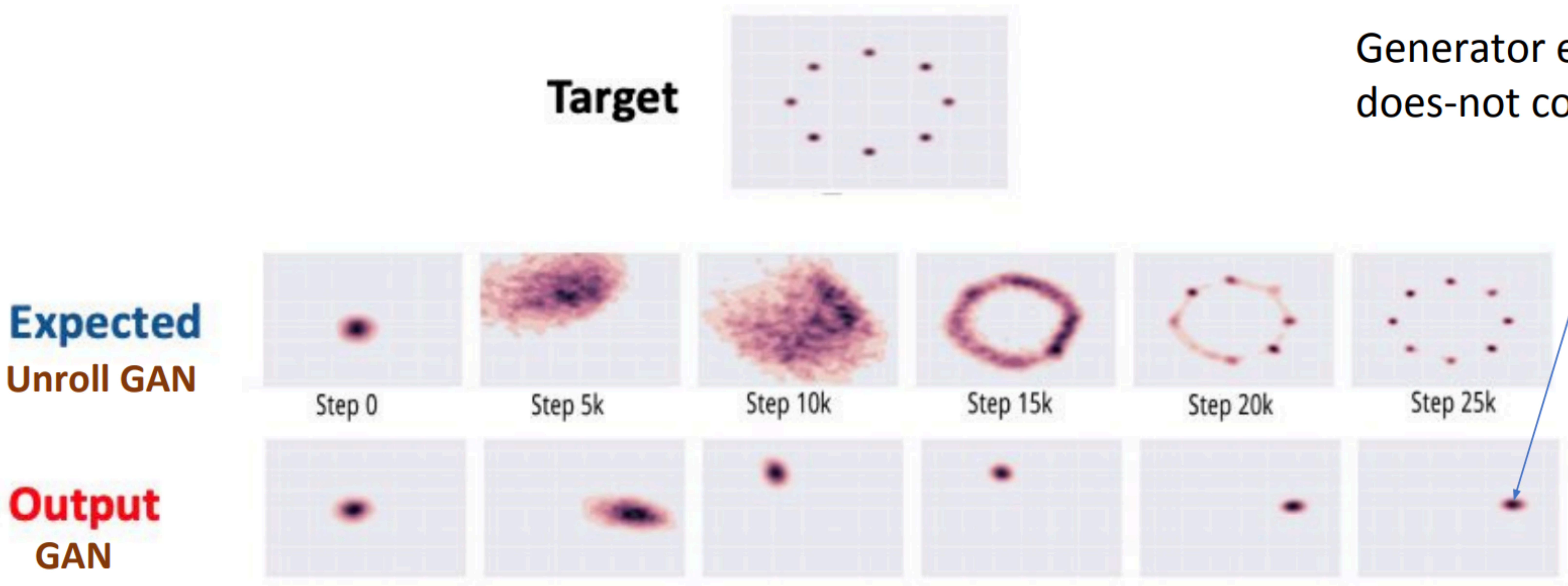
- Differential Equation's solution has sinusoidal terms
- Even with a small learning rate, it will not converge
- Discrete time gradient descent can spiral outward for large step size



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