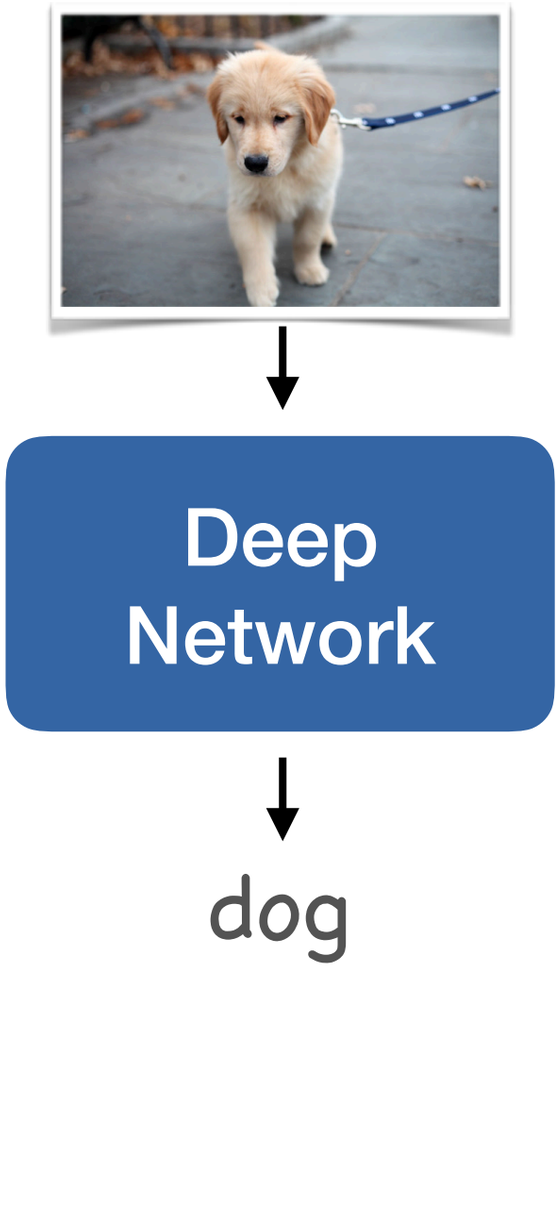
# Image generation

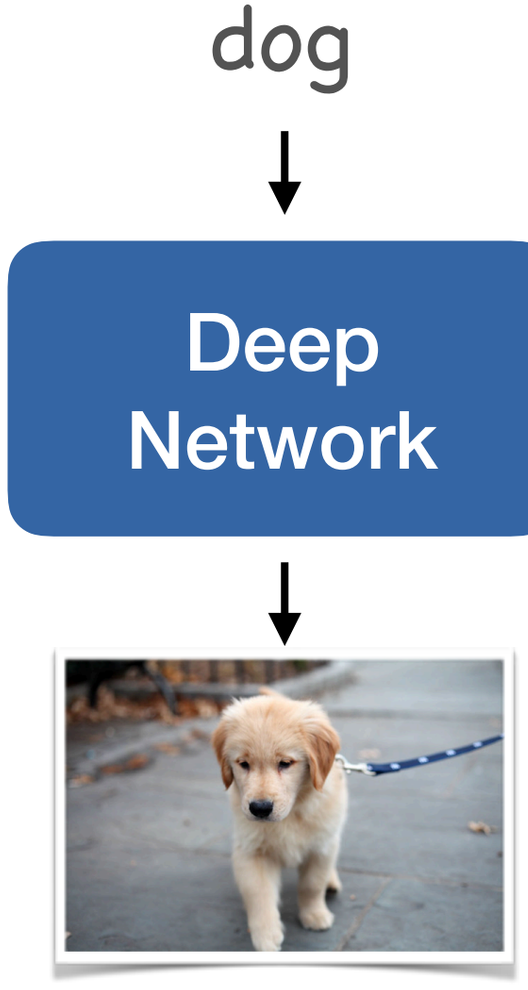
# Image recognition

- Input: Image
  - High dimensional
  - Structured
- Output: Label
  - Low dimensional
- Easy



# Generation

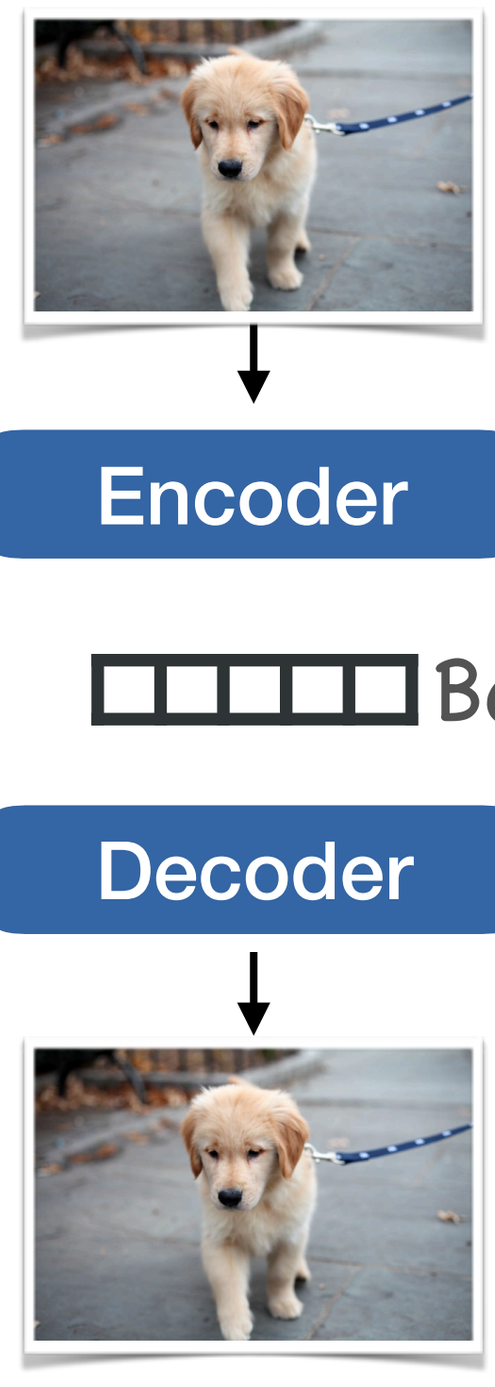
- Input: Label / nothing
  - Low dimensional
- Output: Image
  - High dimensional
  - Many possibilities
- Very hard



# Autoencoder

# Autoencoder

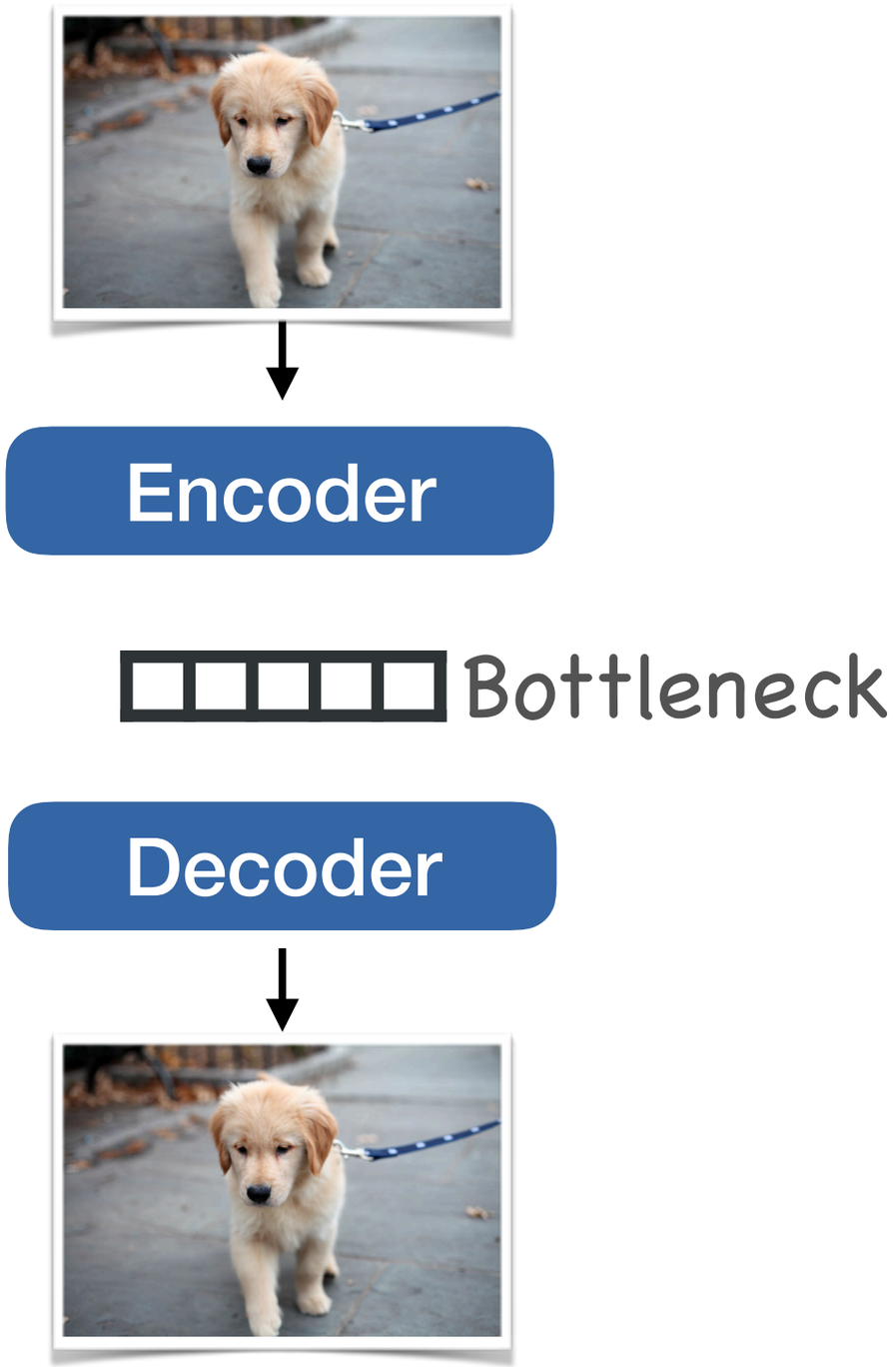
- Image to image
  - with bottleneck




Reducing the Dimensionality of Data with Neural Networks, Hinton and Salakhutdinov, Science, 2006

# What does an autoencoder learn?

- Compression
- Invertible mapping
- Does it learn to understand the image?
  - Only in the limit / best compression

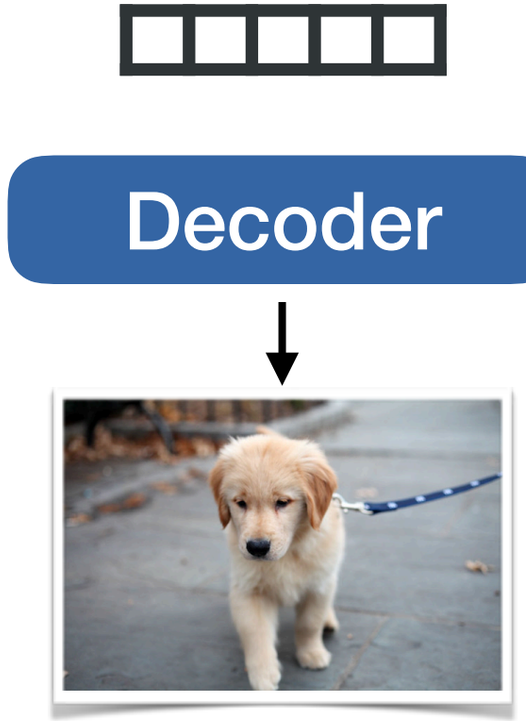


# How do we sample from an autoencoder?

- Random samples?
- Like never seen during training
- Might produce garbage

# Alternative that works

- Deep image prior
  - Learn decoder of autoencoder
  - Fixed random input
  - Learns to denoise

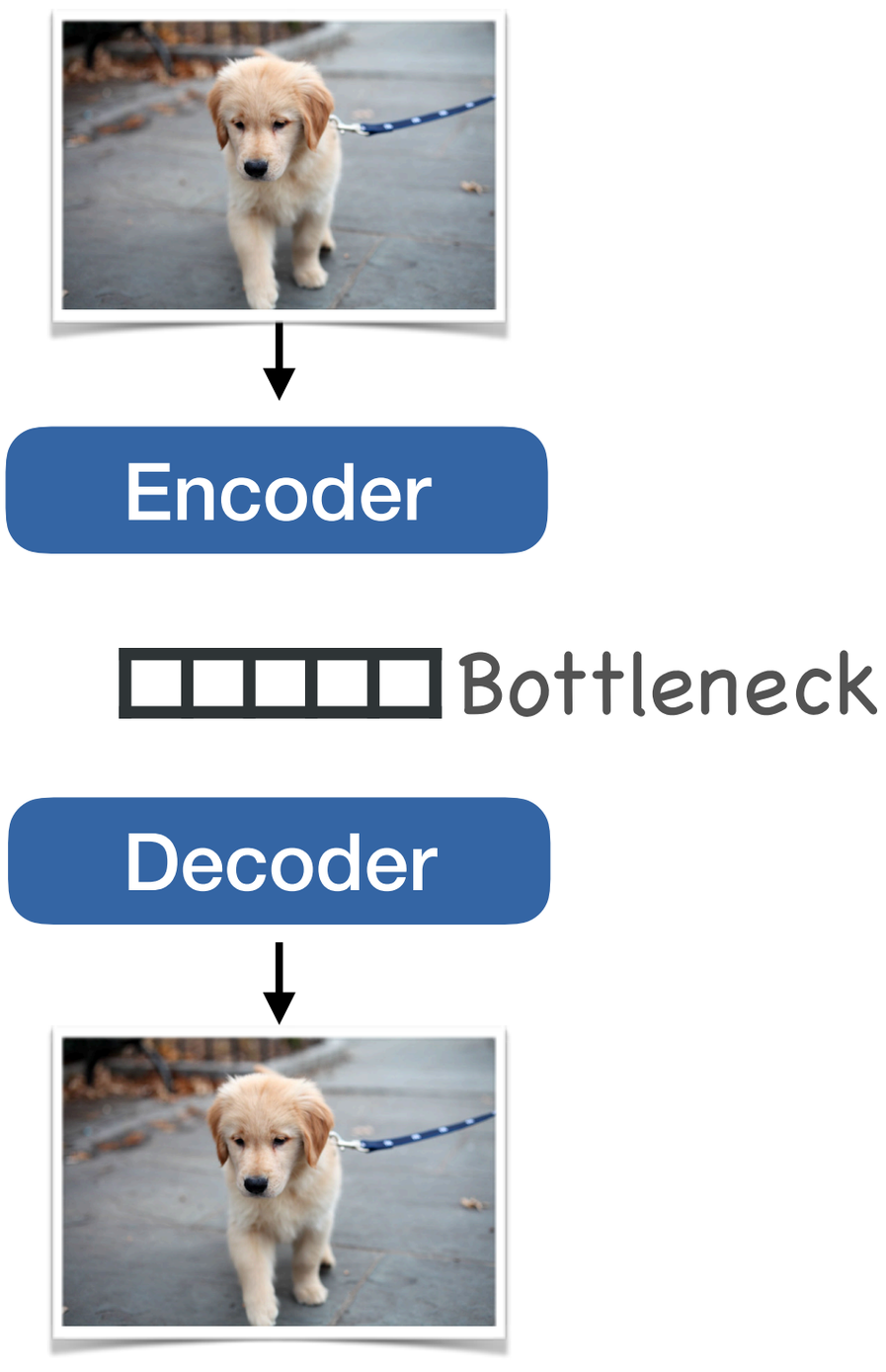



Deep image prior, Ulyanov et al., CVPR 2018

# Variational autoencoder

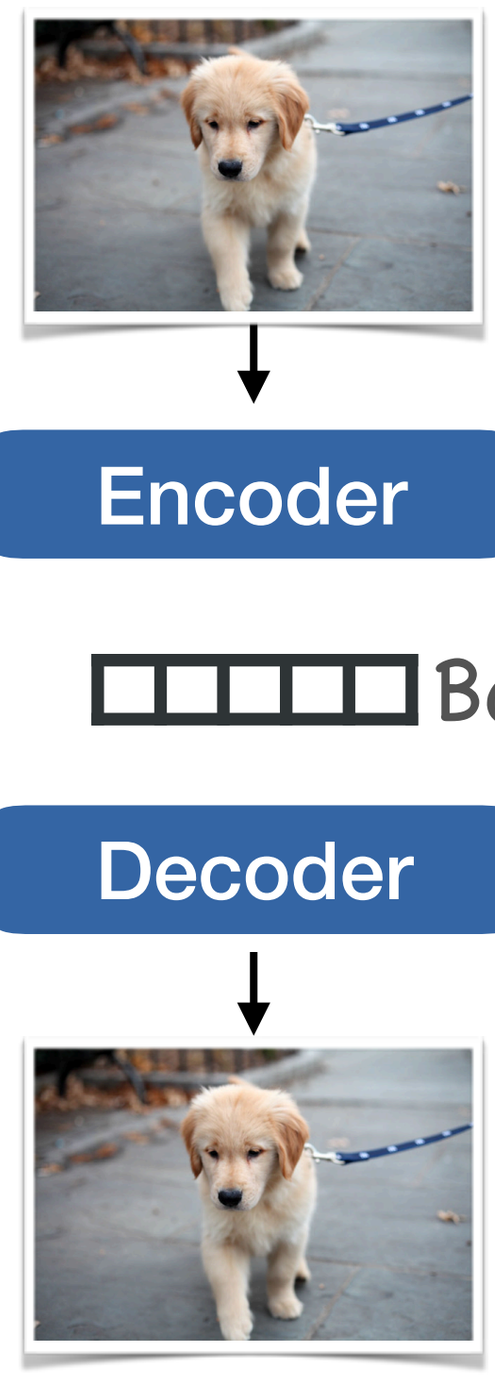
# Autoencoders - Issues

- No sampling
- Learns just compression



# Variational autoencoder

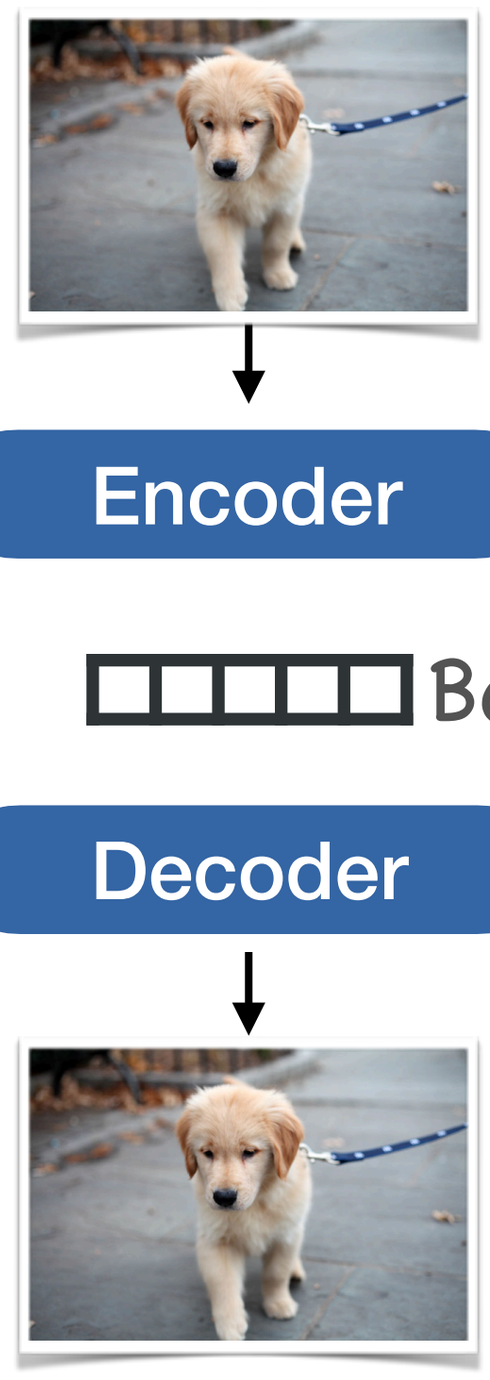
- Autoencoder
  - with noise in bottleneck



# Why does noise help?

# Variational autoencoder - formal definition

- Encoder
  - $q(\mathbf{z}\,|\,\mathbf{x}) = \mathcal{N}\left(\mathbf{z}; \mu_\theta(\mathbf{x}), \sigma^2_\theta(\mathbf{x})\mathbf{I}\right)$
- Sampling $\mathbf{f} \sim q(\mathbf{z}\,|\,\mathbf{x})$
- Decoder
  - $P(\mathbf{x}\,|\,\mathbf{f})$
- Approximately learns $P(\mathbf{x})$
  - Variational lower bound



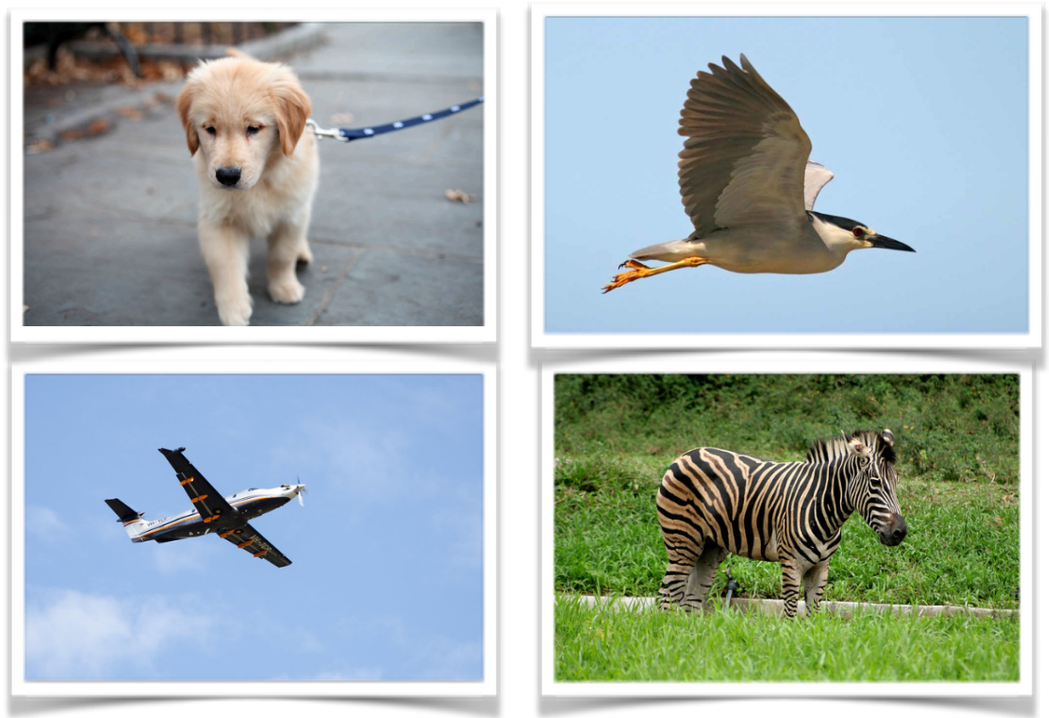
# Variational autoencoder - Issues

- Fails in high dimensions
  - Hard to embed spherical distributions
- Blurry outputs
  - Pixel-distance

# Transforming noise

# Modeling the distribution of images

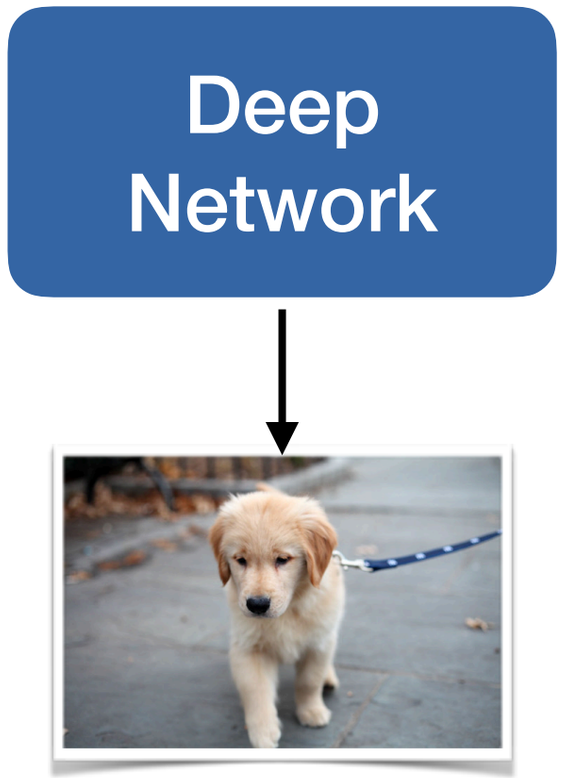
- Goal
  - Model $P(\mathbf{x})$
  - As a sample distribution $X \sim P$
    - Only generate images



# Sampling distributions

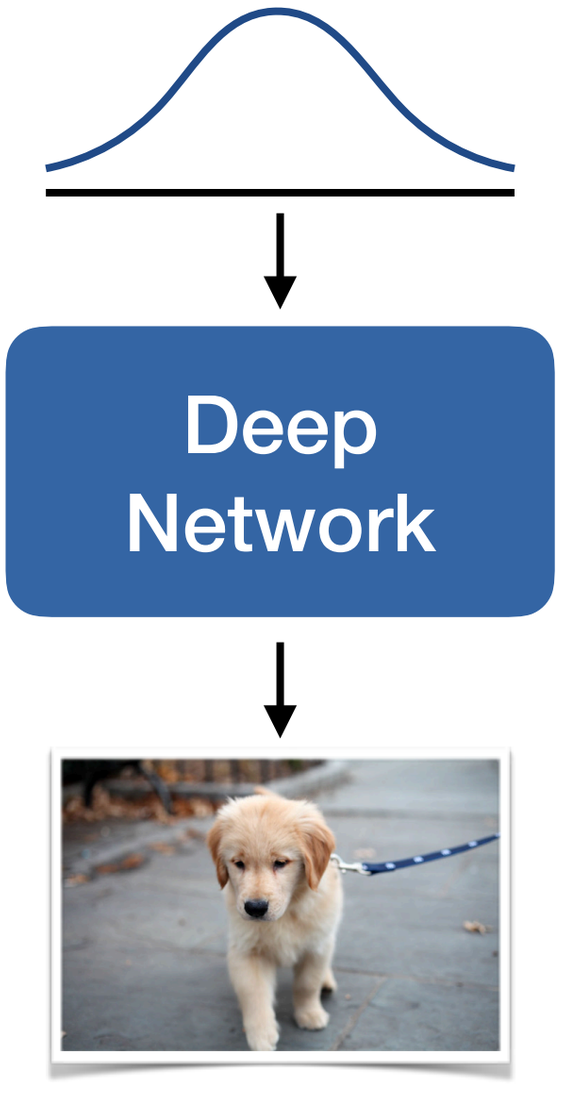
- Model $X \sim P$ as network

- What is the input?



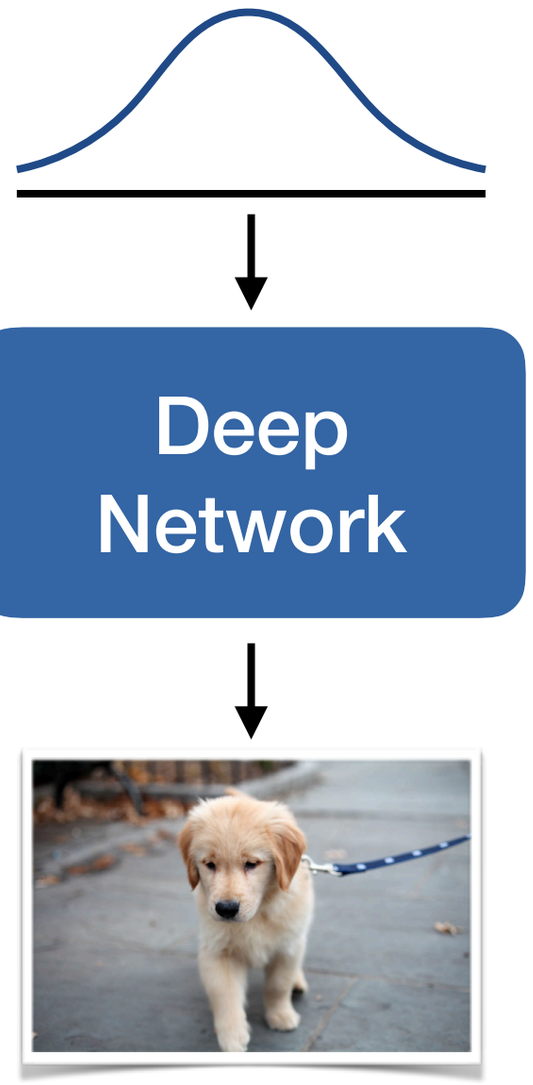
# Transforming noise

- Input: Random noise
  - e.g. $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Output: Image



# How do we train this?

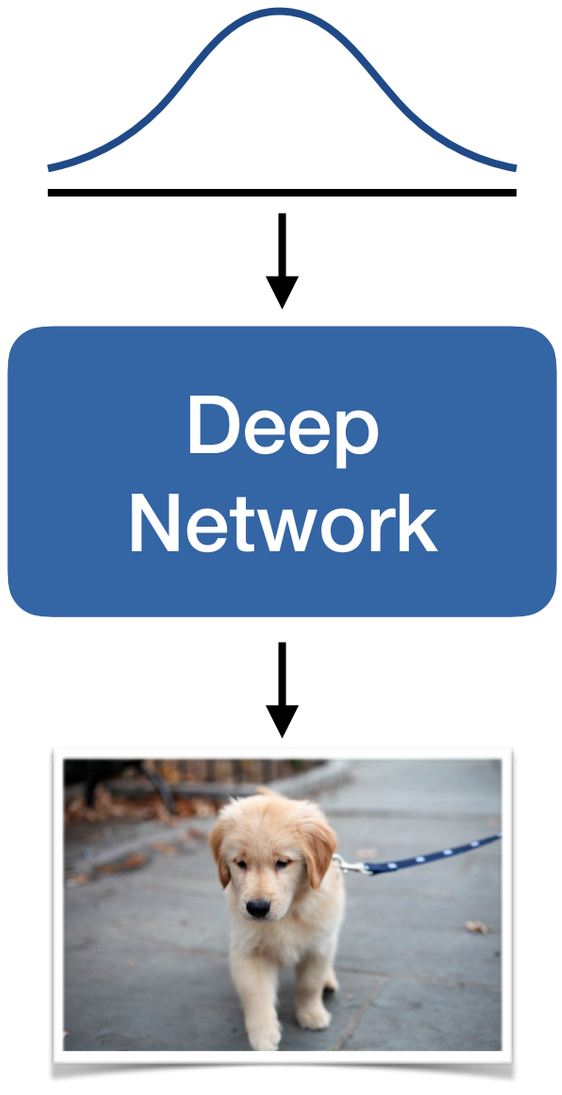
- How do we assign $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to the corresponding image?



# Generative adversarial networks

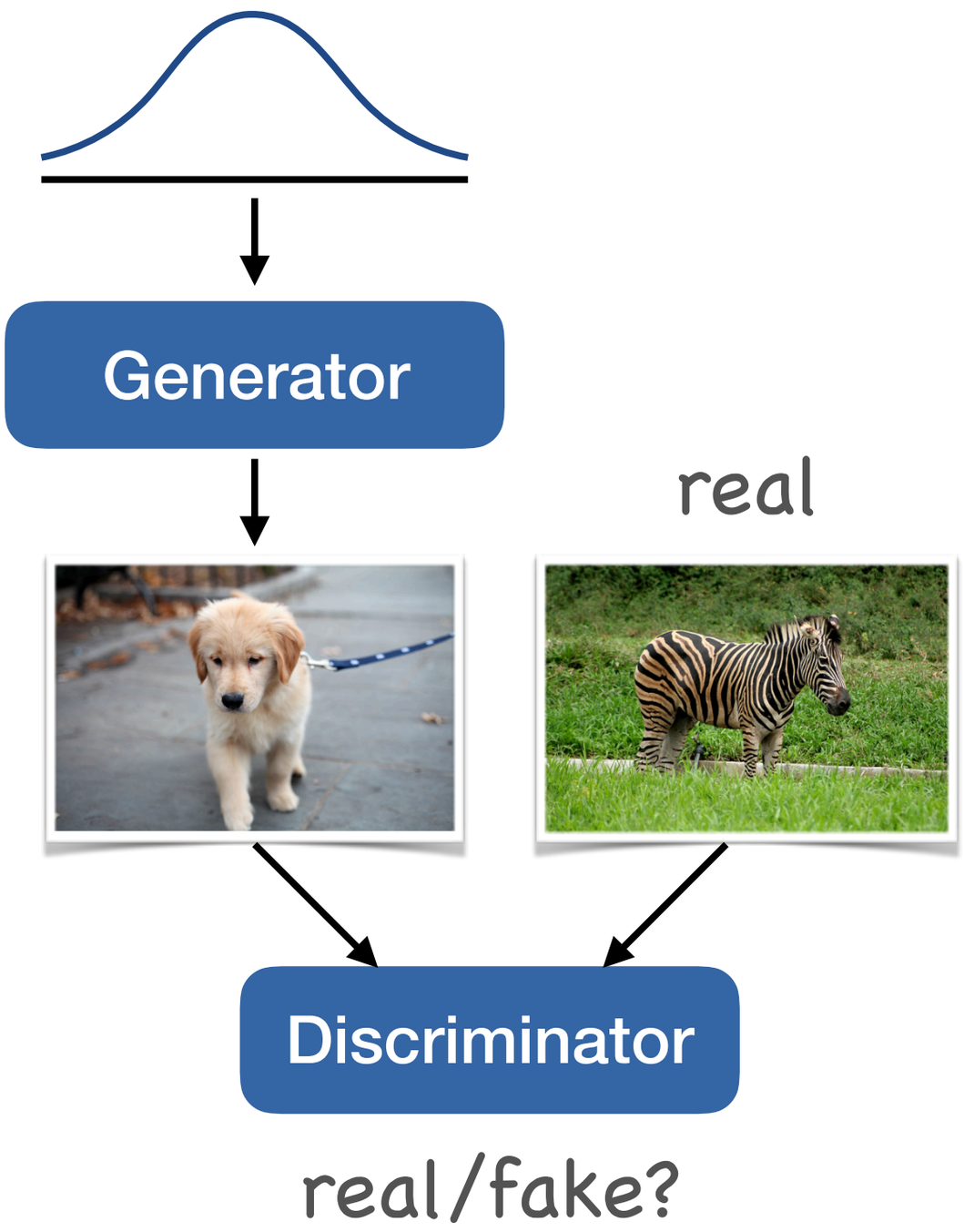
# Transforming noise

- Input: Random noise
  - e.g. $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Output: Image
- Objective
  - Output should look good



# Judging output quality

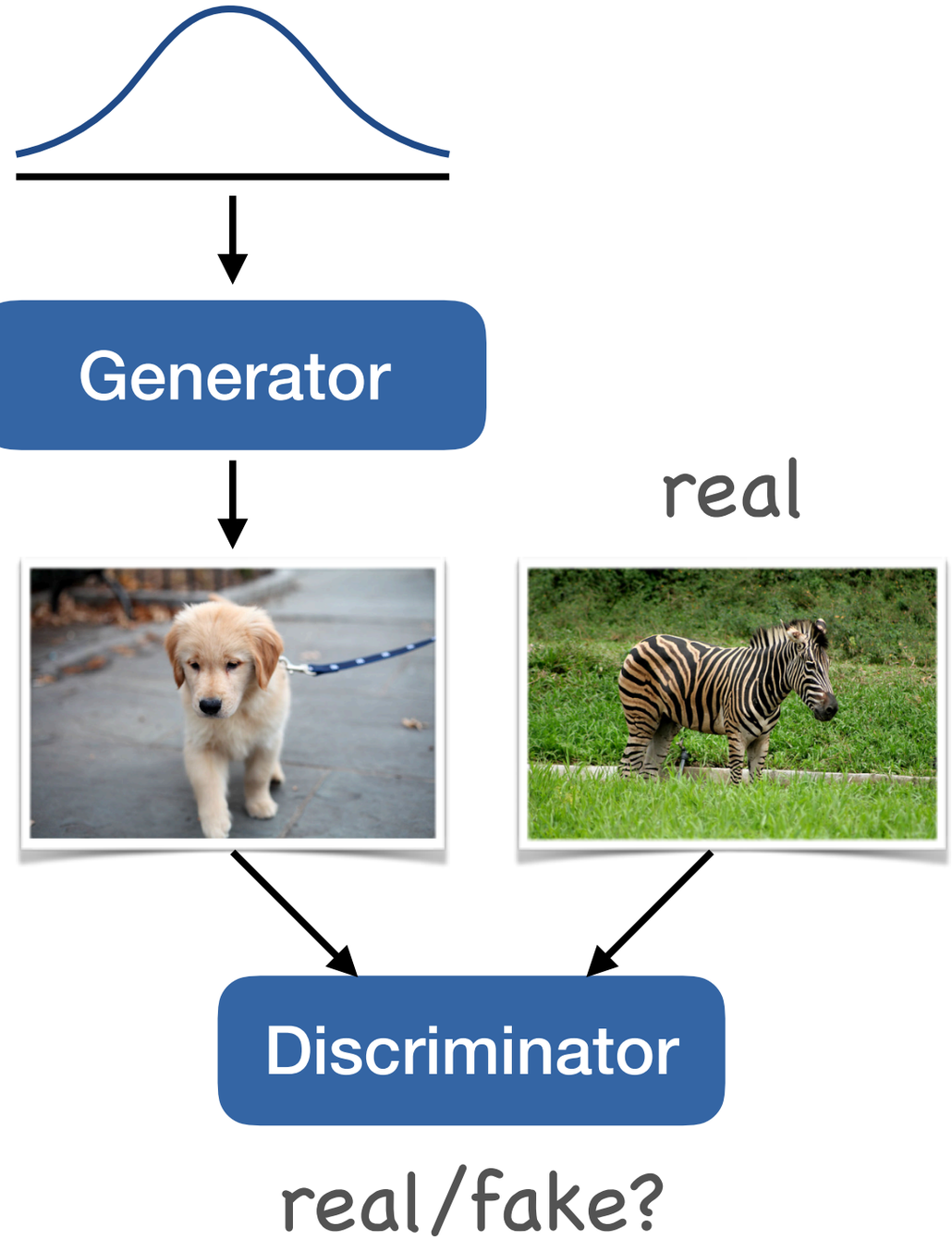
- A second network
- Output looks like training data or not?




Generative Adversarial Nets, Goodfellow et al., NIPS 2014

# Adversarial objective

- Generator
  - Produce an image that fools discriminator
- Discriminator
  - Tell difference between generation and training data



# What does this optimize?

- Discriminator log-likelihood
- Jensen-Shanon divergence data distribution and generator distribution

# GANs work!

- Sampling is easy
- Learn pixel-distance
- Loss on distributions

Large Scale GAN Training for High Fidelity Natural Image Synthesis, Brock et al., ICLR 2019

# Applications - Super resolution

- Learn to up-sample images
- Input: Low-res image
- Output: HD image
- Loss
  - Reconstruct HD
  - GAN for sharp reconstruction

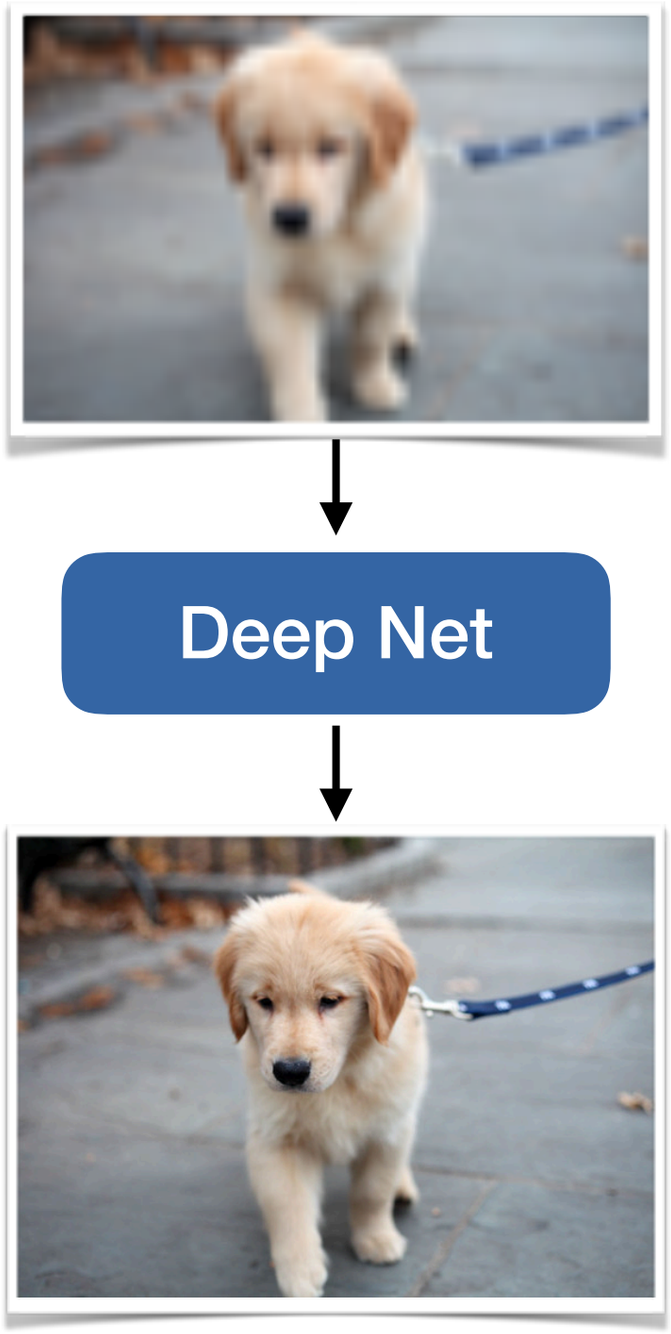



Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network, Ledig et al., CVPR 2017

# Applications - Text to image

- Input: text
- Output: Image
- Loss:
  - GAN
    - Does text and image fit or not?

"This bird has a yellow belly and tarsus, grey back, wings, and brown throat, nape with a black face"

Image source: StackGAN paper

StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, Zhang et al., ICCV 2017

# Pix2Pix

# Image to image translation

- Input: Image
- Output: Image
- Objective:
  - Map input to output

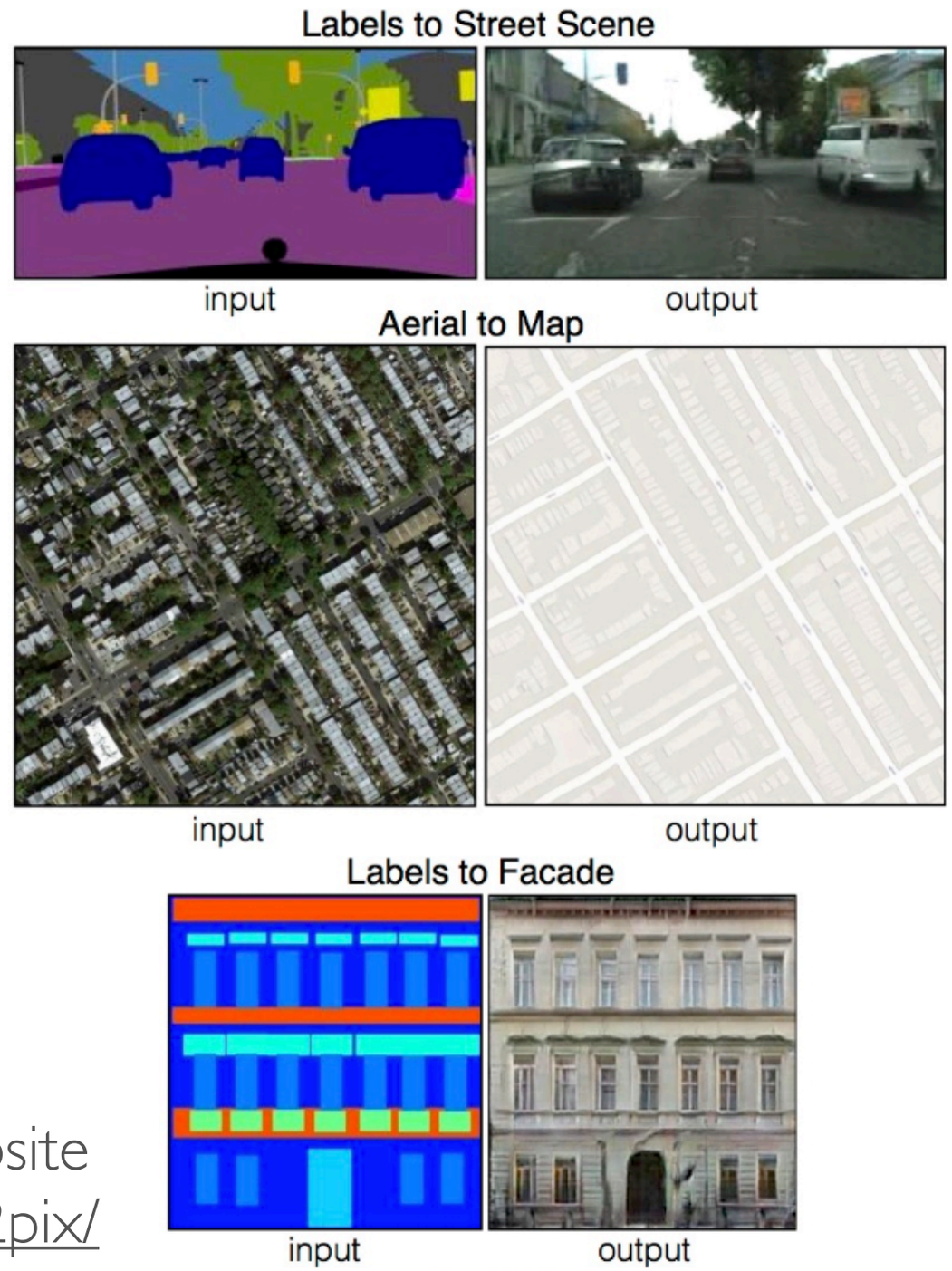
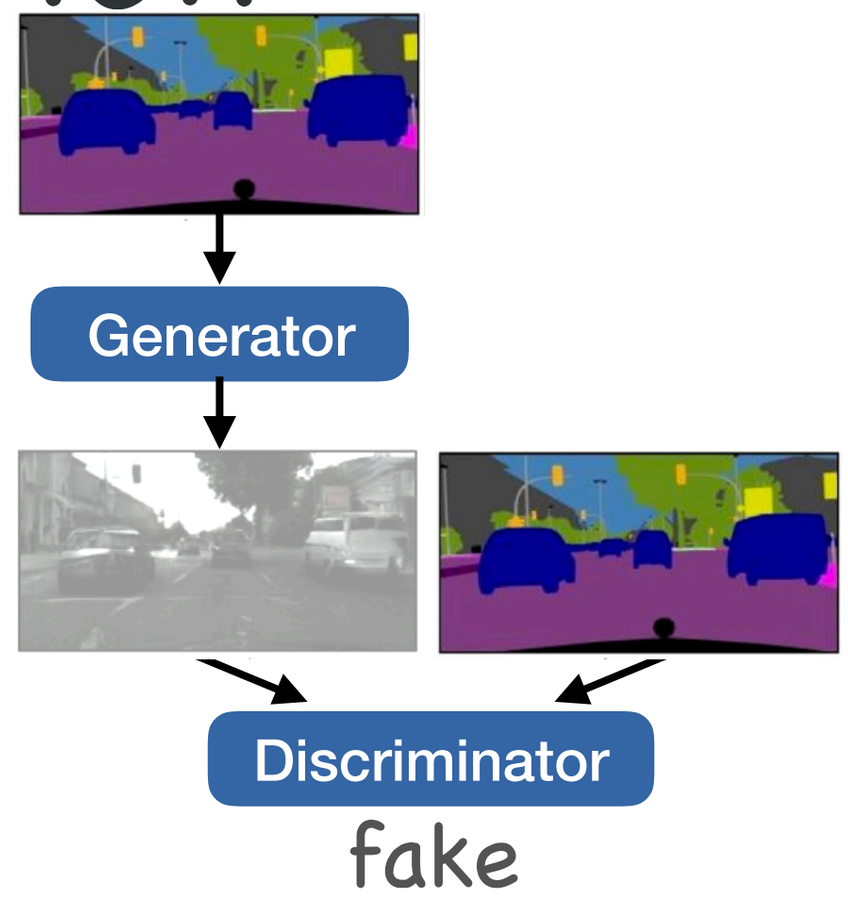



Image source: pix2pix website
https://phillipi.github.io/pix2pix/

# Pix2Pix - Image2Image translation

- "Autoencoder"
  - Different input and output
- GAN loss
  - High fidelity reconstruction




real image


Image-to-Image Translation with Conditional Adversarial Networks, Isola et al., CVPR 2017

# Pix2Pix - examples

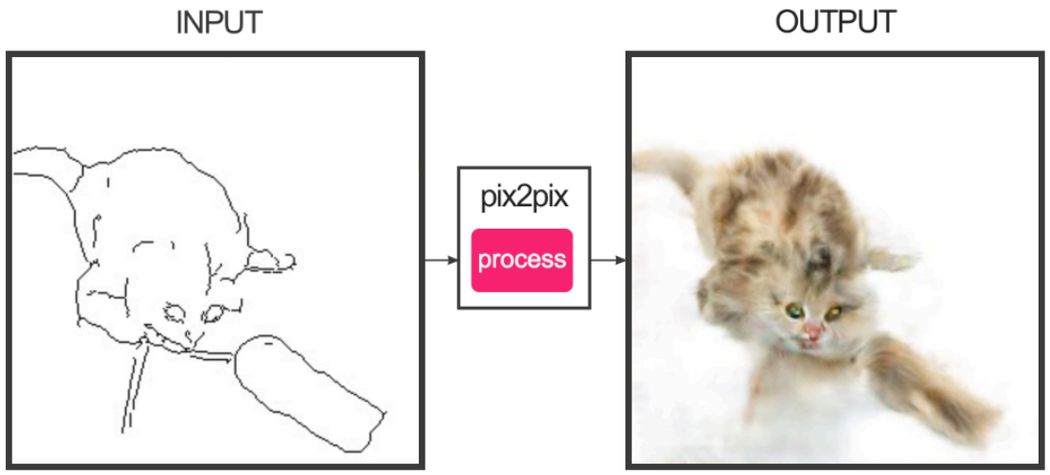
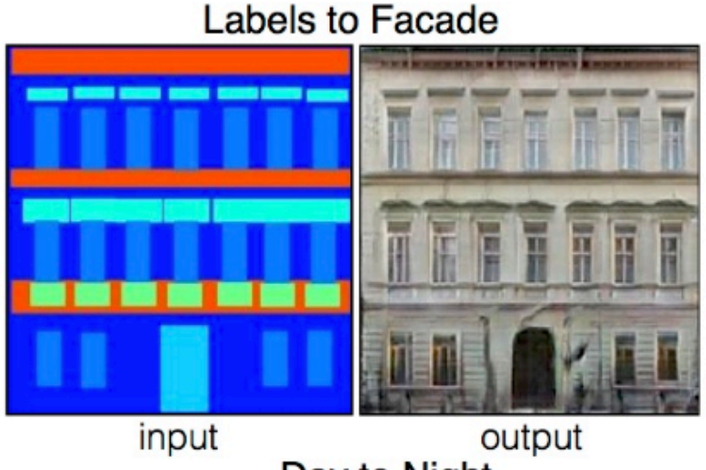
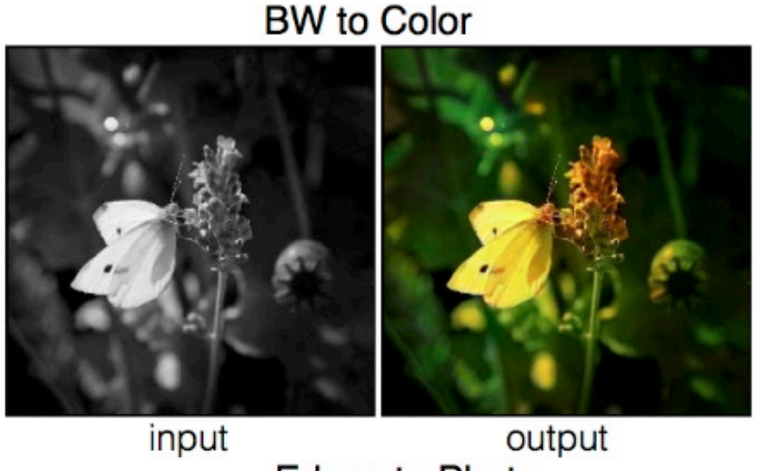
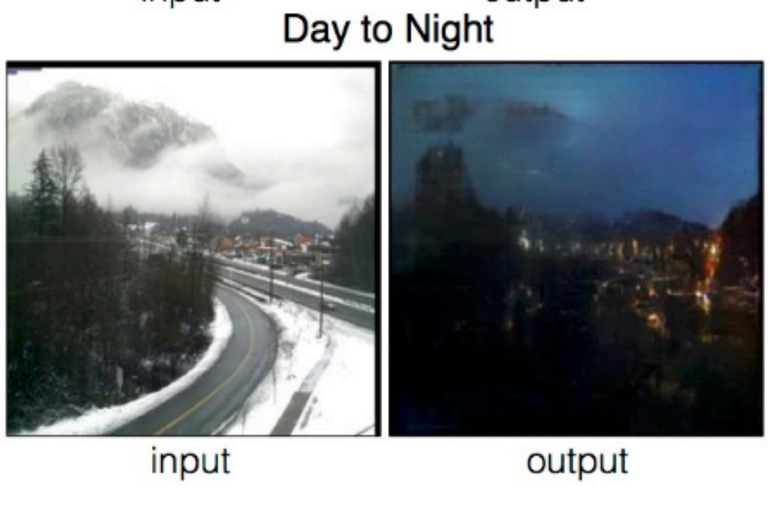



Image source: pix2pix website
https://phillipi.github.io/pix2pix/

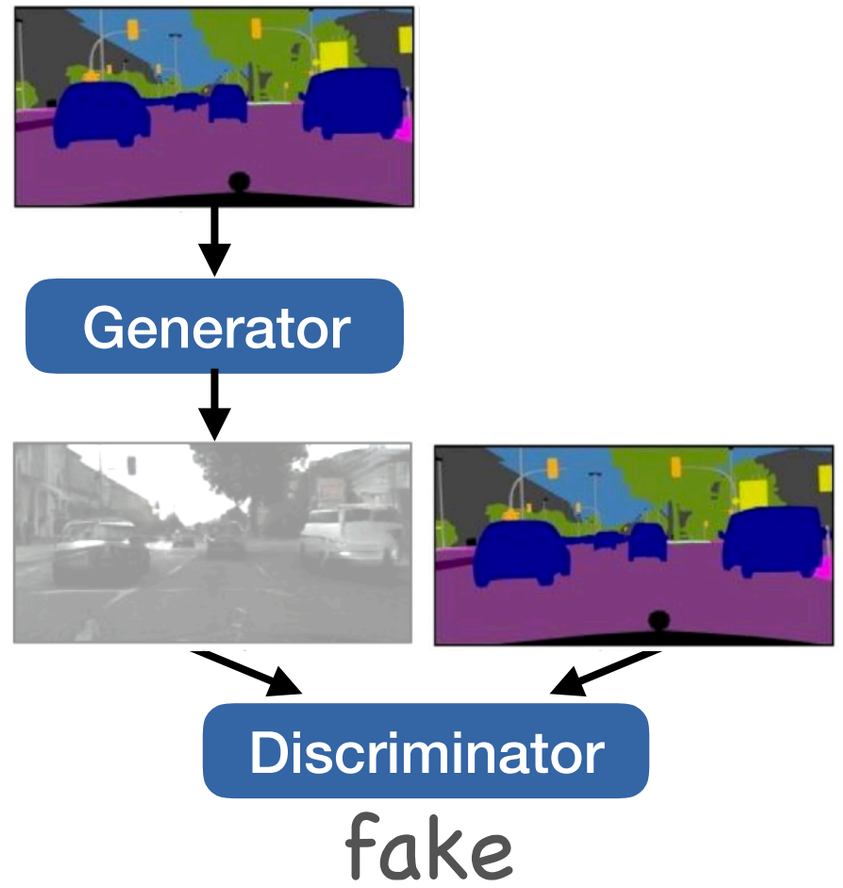








# Pix2Pix - limitations

- Requires a dataset of paired images

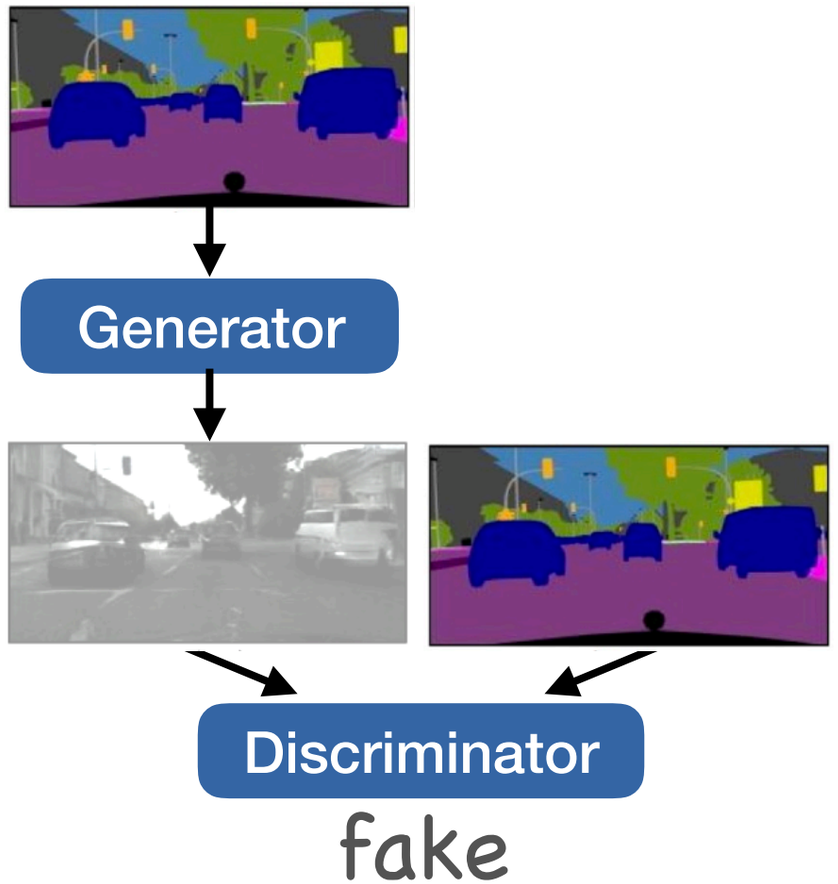



# CycleGAN

# Pix2Pix

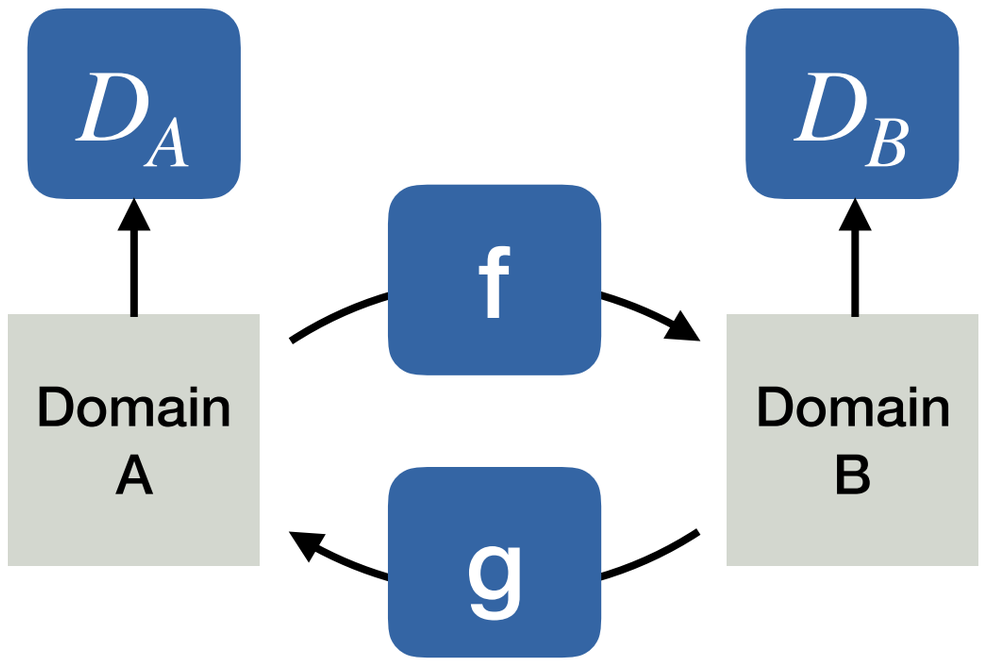
- Translates images from paired input / output data

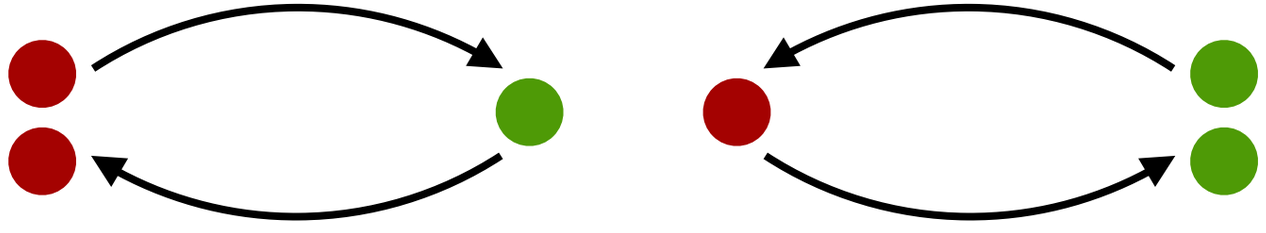
- How do we this without pairing?




# CycleGAN

- Train two networks f and g
  - Map between domains
- Loss
  - GAN
  - Reconstruction



Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, Zhu et al., ICCV 2017

# What does CycleGAN do?

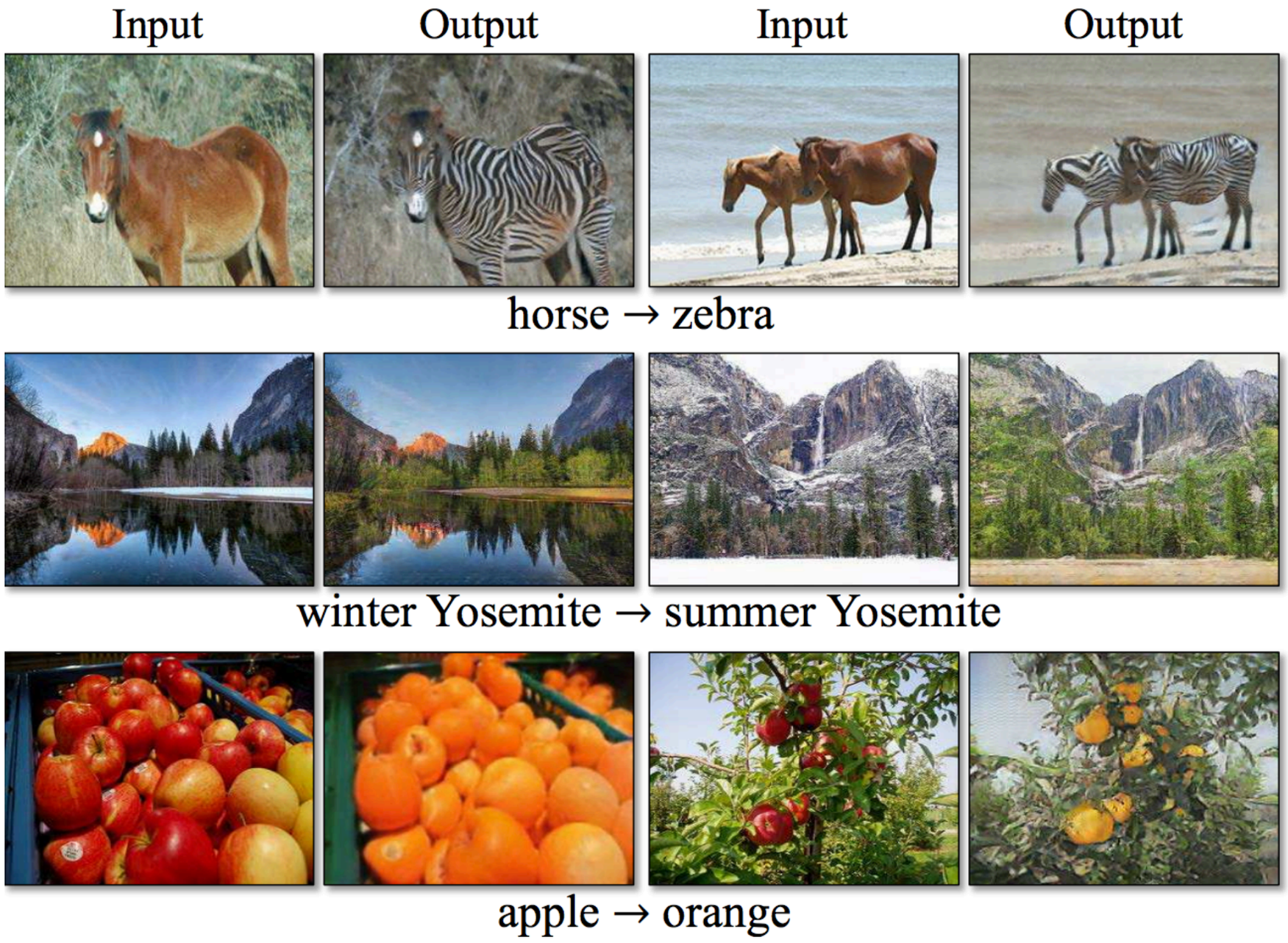



Figure source: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, Zhu et al., ICCV 2017

# What does CycleGAN do?

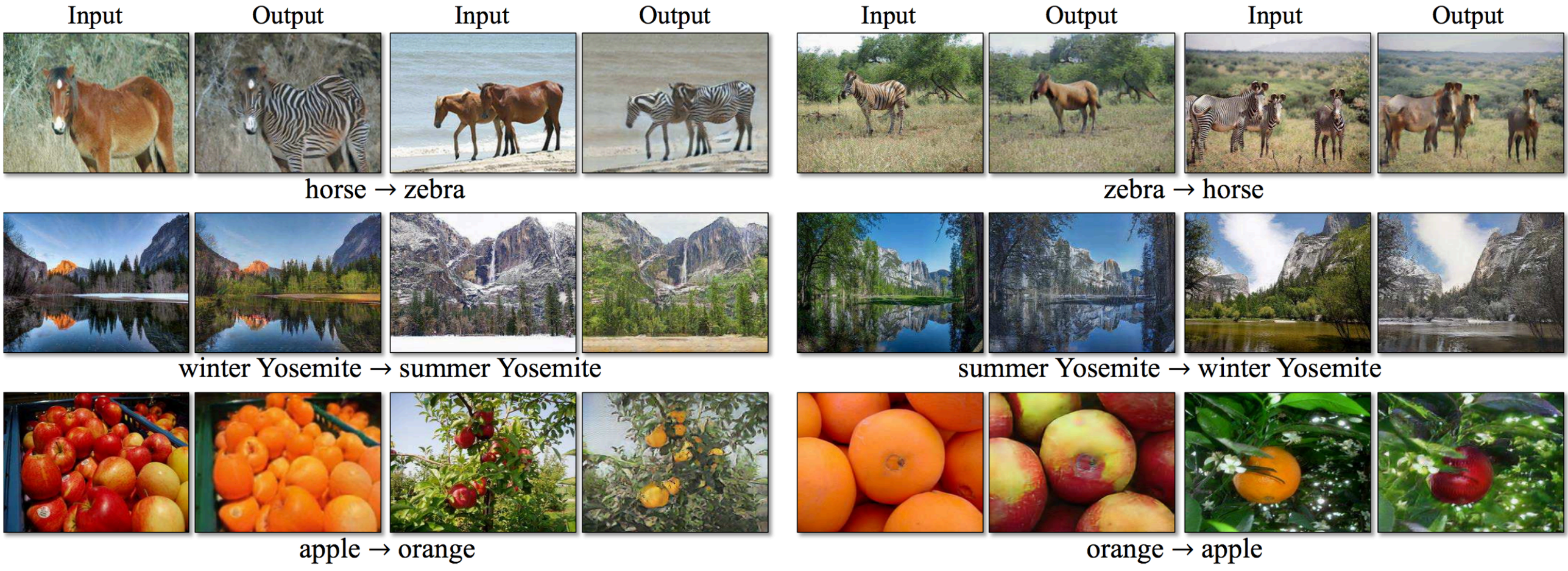
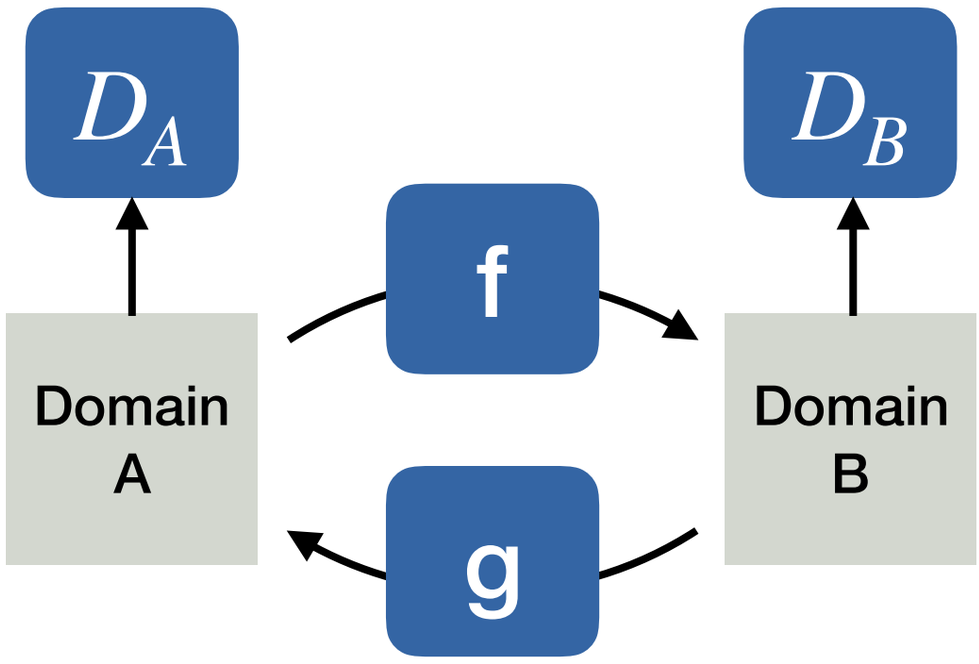
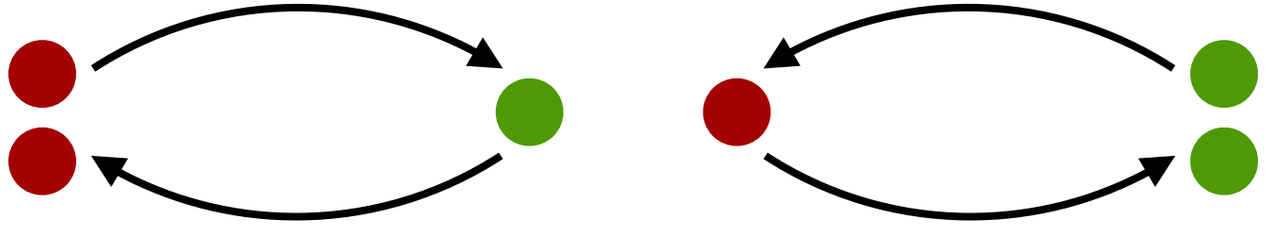



Figure source: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, Zhu et al., ICCV 2017

# Why does it work?

- Why learn the right mapping?
  - Simplest mapping



# Image editing

# Image editing

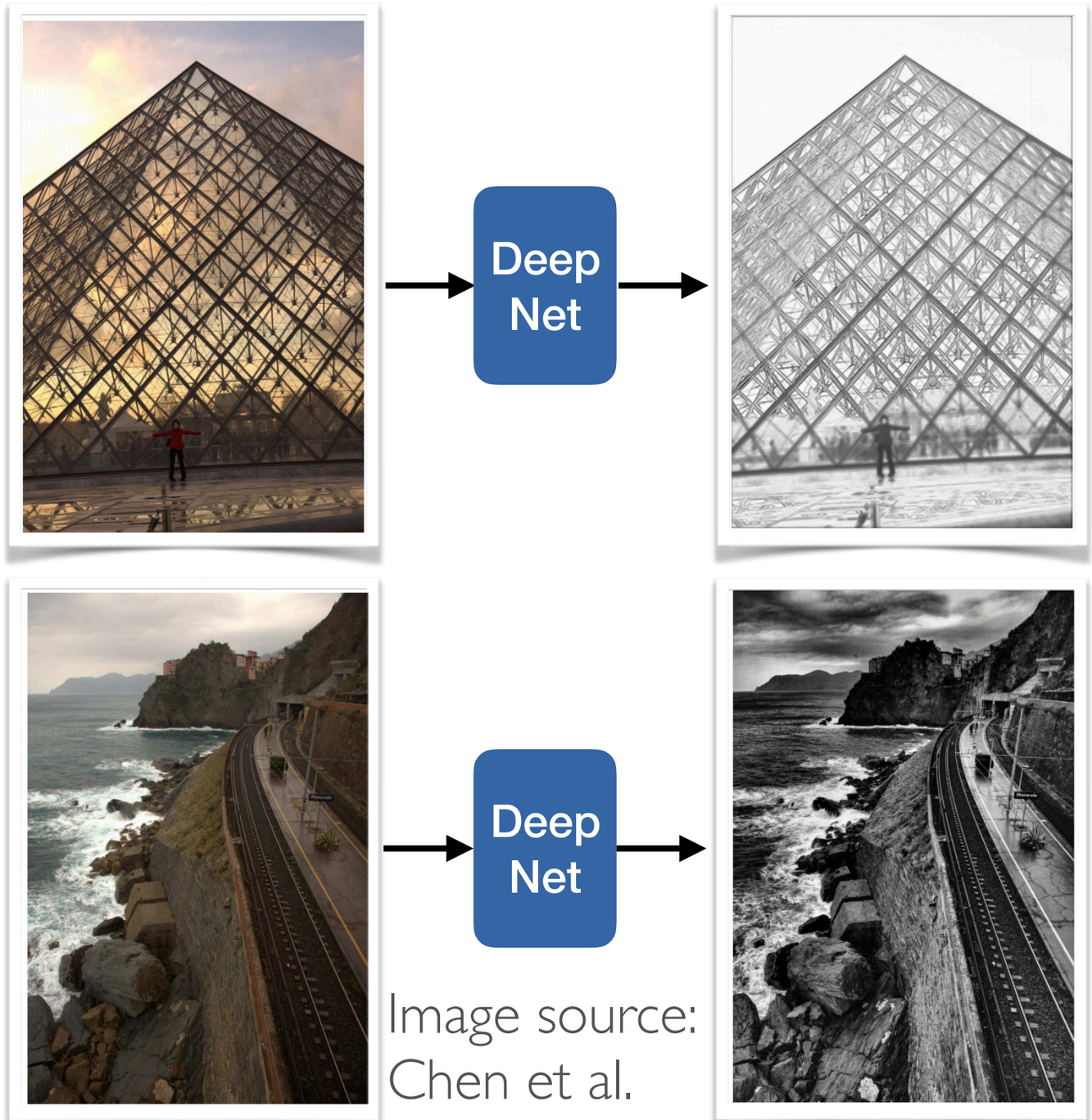
- Computer graphics tasks
  - Image restoration
    - Inpainting
  - Denoising
  - Matting / compositing
  - Style transfer

source: Xu et al., https://arxiv.org/pdf/1703.03872.pdf

source: Gatys et al., https://arxiv.org/abs/1508.06576

# How are many image processing tasks done?

- Deep network
- Input: Original
- Output: Desired output




Fast Image Processing with Fully-Convolutional Networks, Chen et al., ICCV 2017

Image source: Chen et al.

# Style transfer

+

- Content of source
- Style of target

Image generative by https://github.com/pytorch/examples/tree/master/fast_neural_style

# Style transfer

# Style transfer

- Content of source
- Style of target
- How do we measure style and content?

+

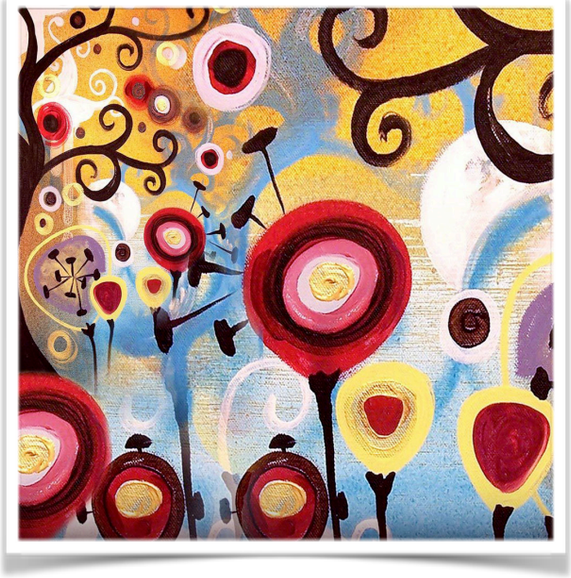
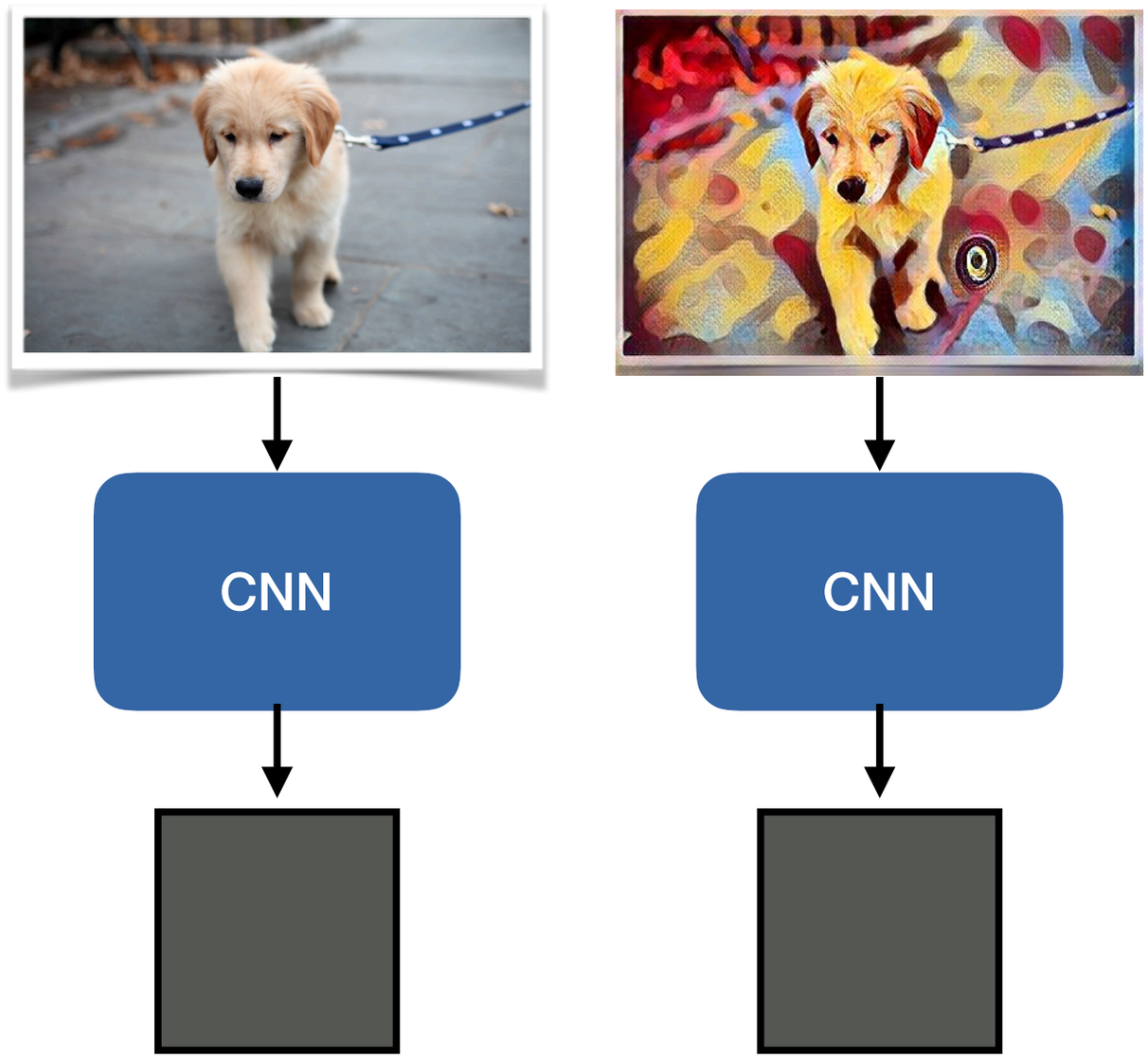


Image generative by https://github.com/pytorch/examples/tree/master/fast_neural_style

# Content

- High level activations of a CNN
  - e.g. VGG



# Style

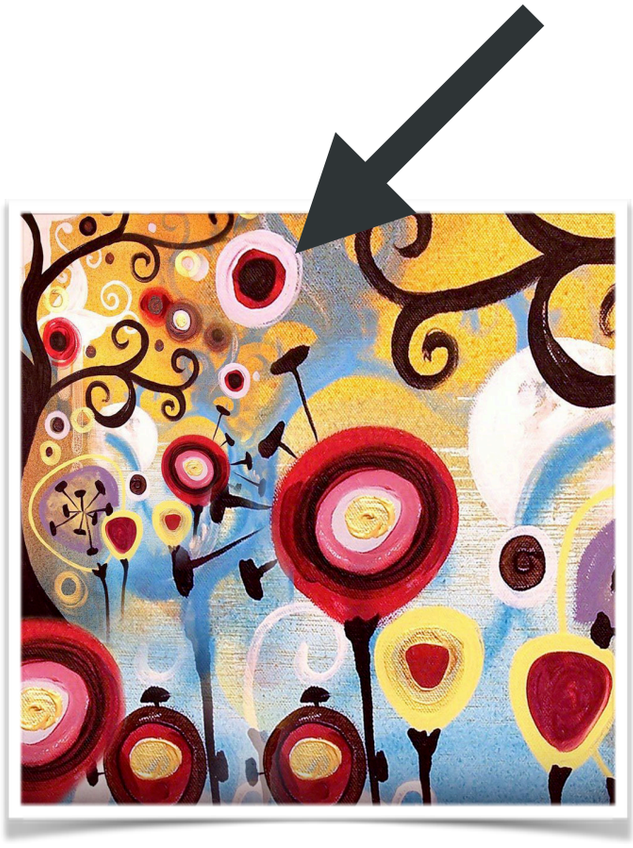
- Low level activations?
  - Contain both style and content
    - Spatial arrangement

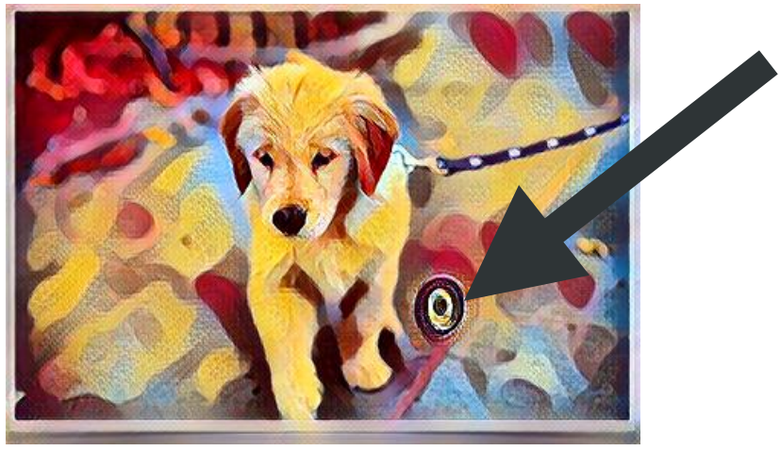
# Style

- Solution: Statistics of low level activations

- No (global) spatial information

- Local patterns only





# Neural style transfer

- Optimize for image $\mathbf{x}$
  - With high level activations $f(\mathbf{y}_1)$
  - and Gram statistic of low level activation $g(\mathbf{y}_2)$

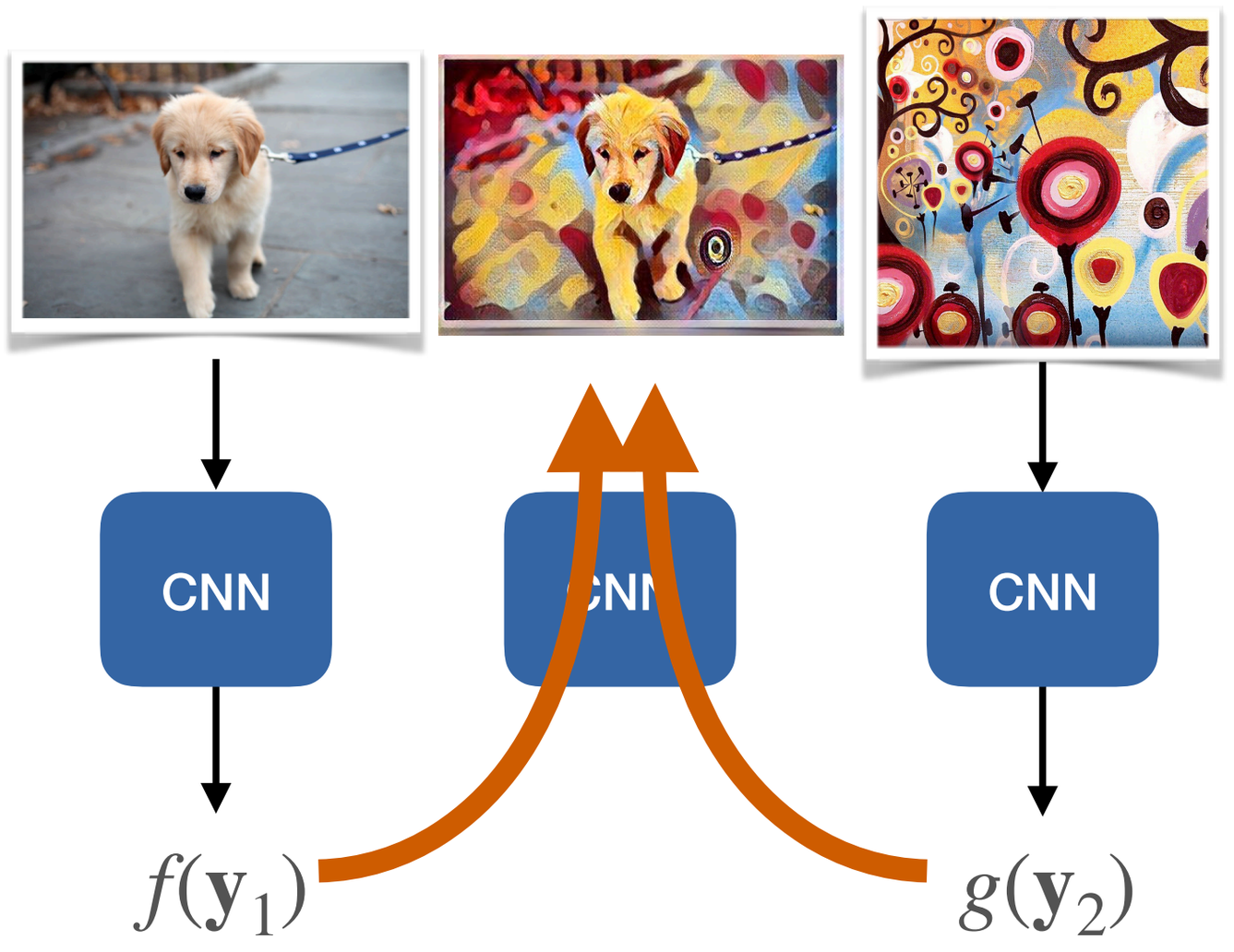
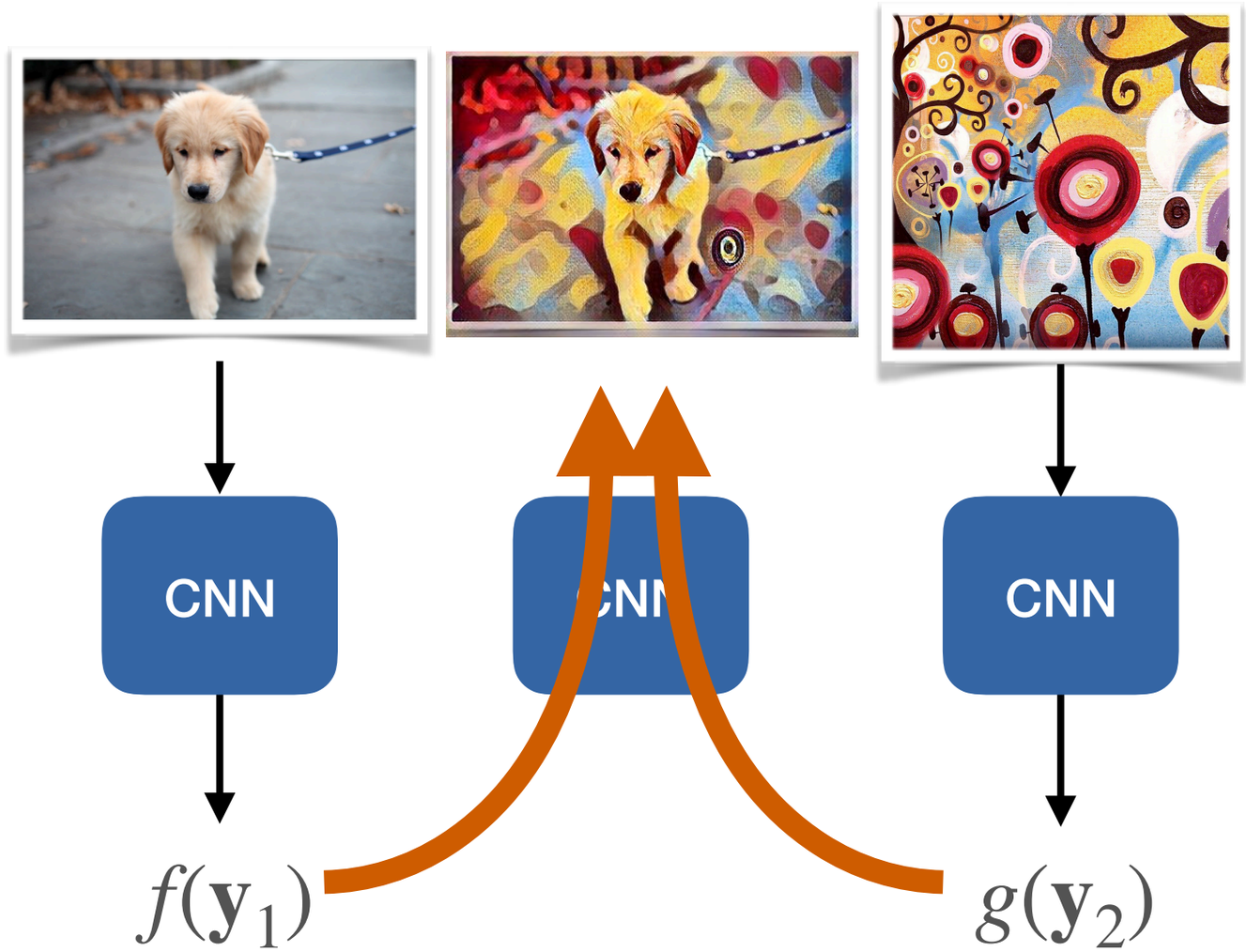



Image Style Transfer Using Convolutional Neural Networks, Gatys et al., CVPR 2016

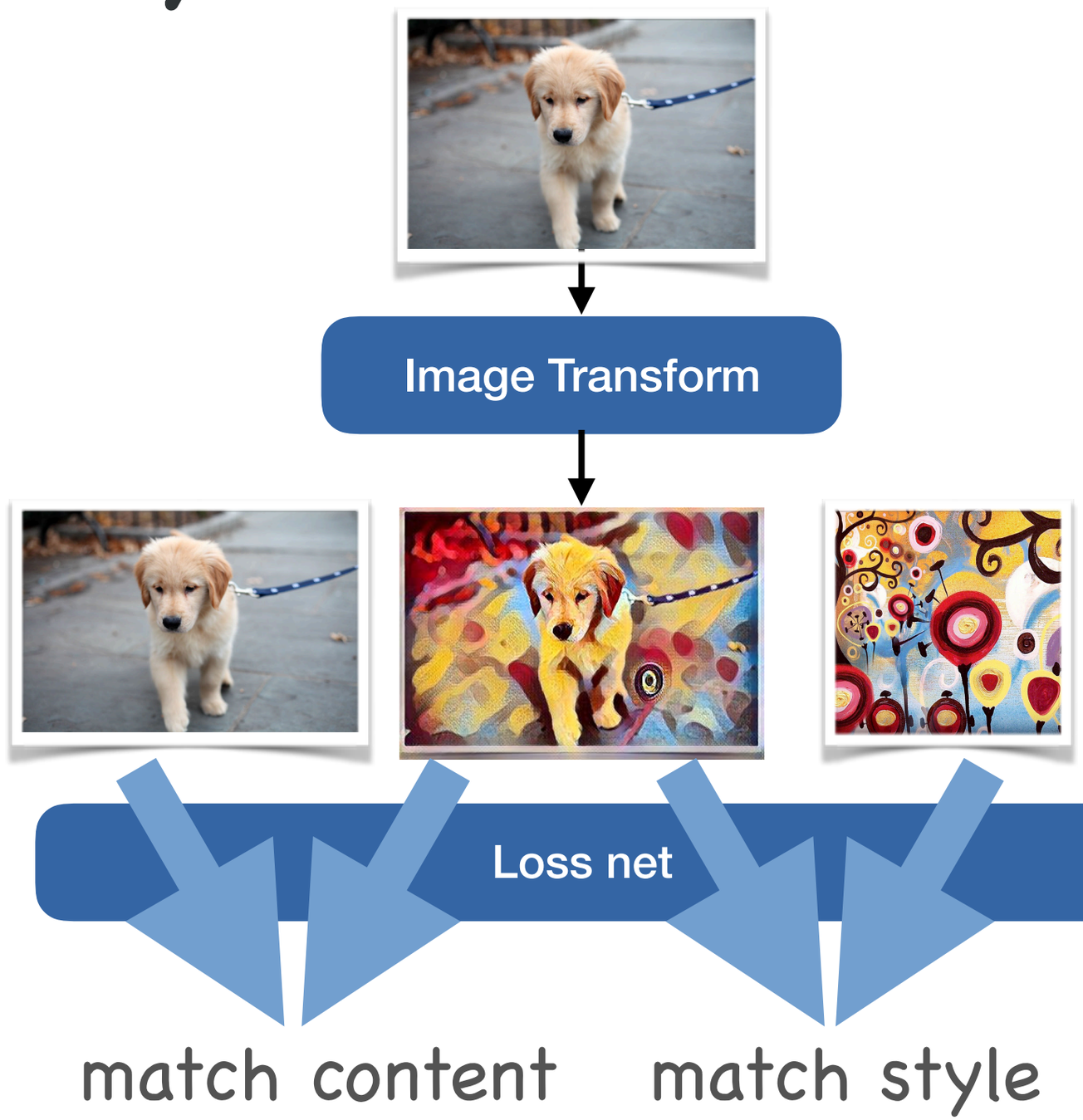
# Neural style transfer - Issues

- Optimization is slow
- 100-1000 backpropagation steps per image



# Fast neural style transfer

- Train a network to do style transfer
  - Objective
    - Match activations
    - Match Gram statistics
- Fast inference
- Slower training

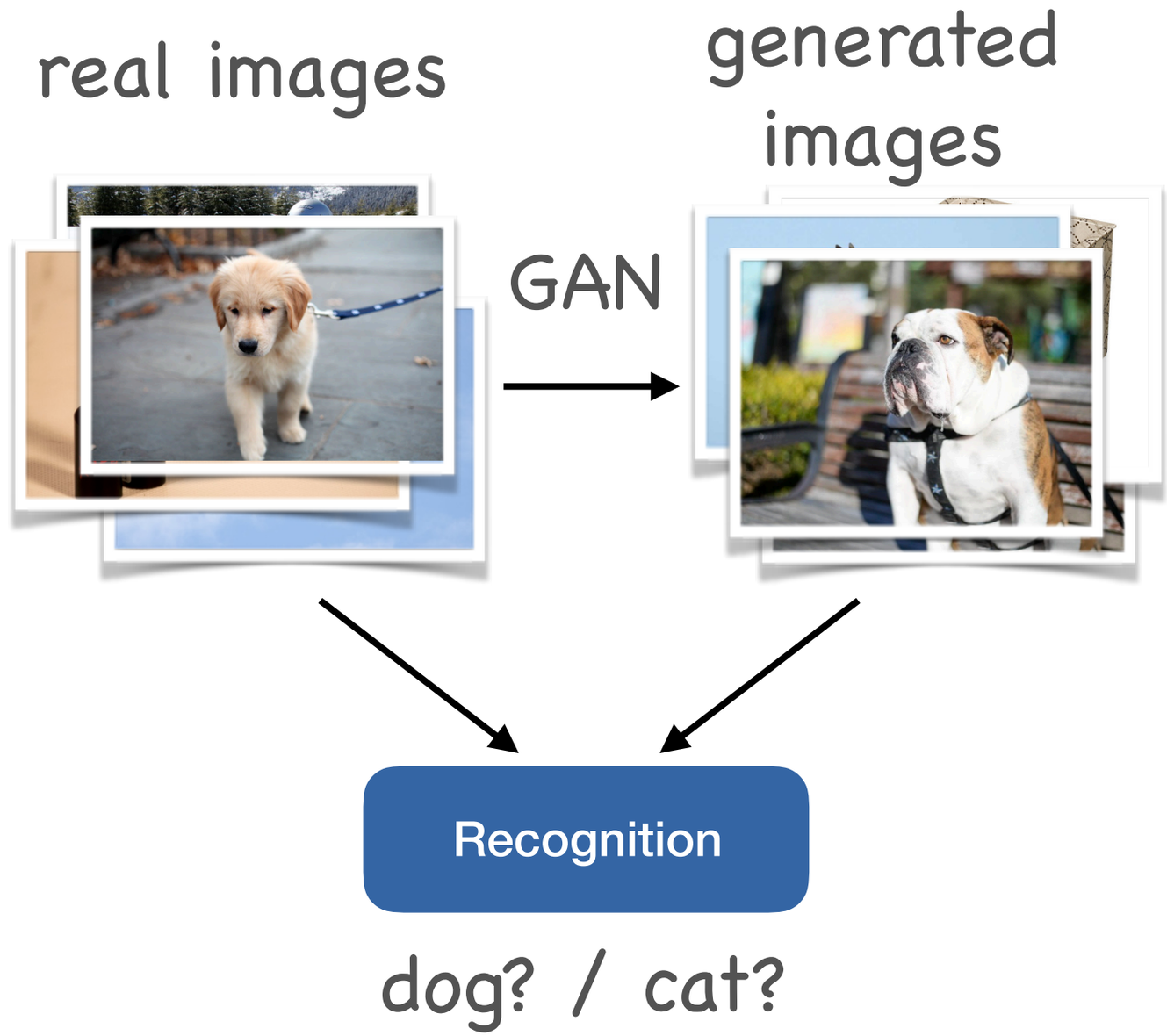



Perceptual Losses for Real-Time Style Transfer and Super-Resolution, Johnson et al., ECCV 2016

# Open Problem: Understanding generative models and invariances

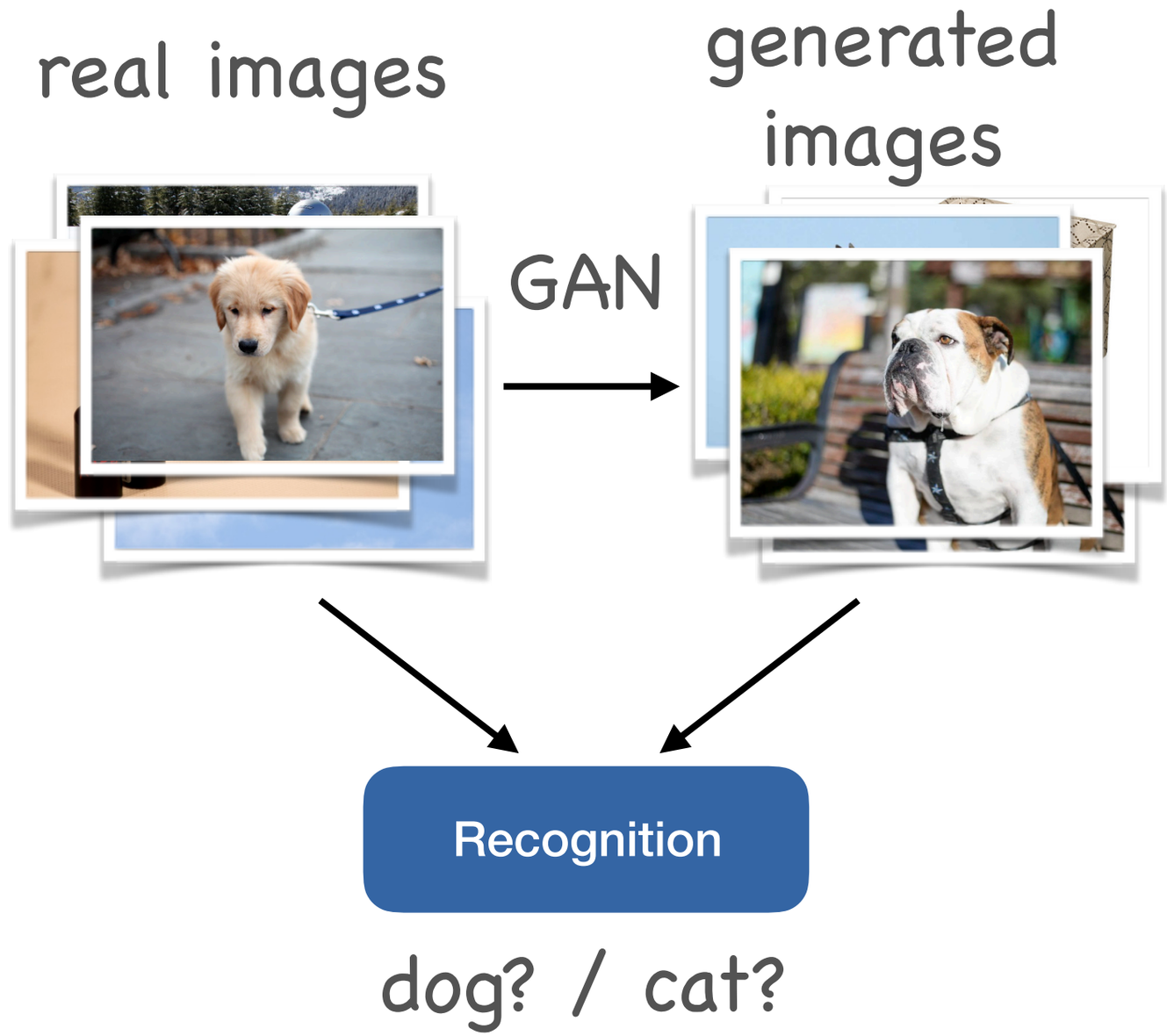
# Can we use generative model to create more training data?

- Step 1: Train a GAN (for each class)
- Step 2: Generate images from that GAN (with free labels)
- Step 3: Train a recognition on generated images



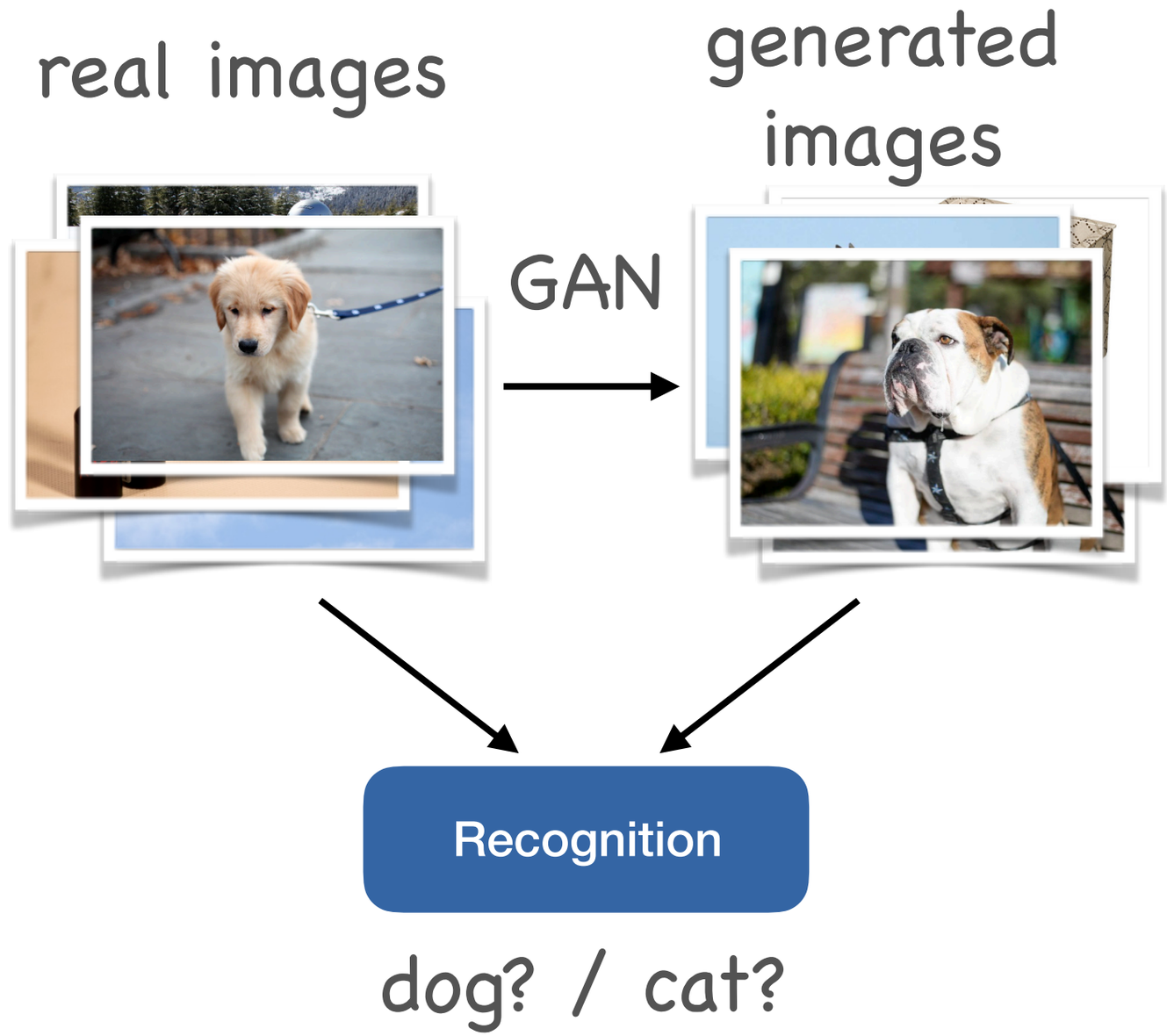
# Does this give a better recognition model?

- Argument in favor
- More data
- Less overfitting
- Better generalization



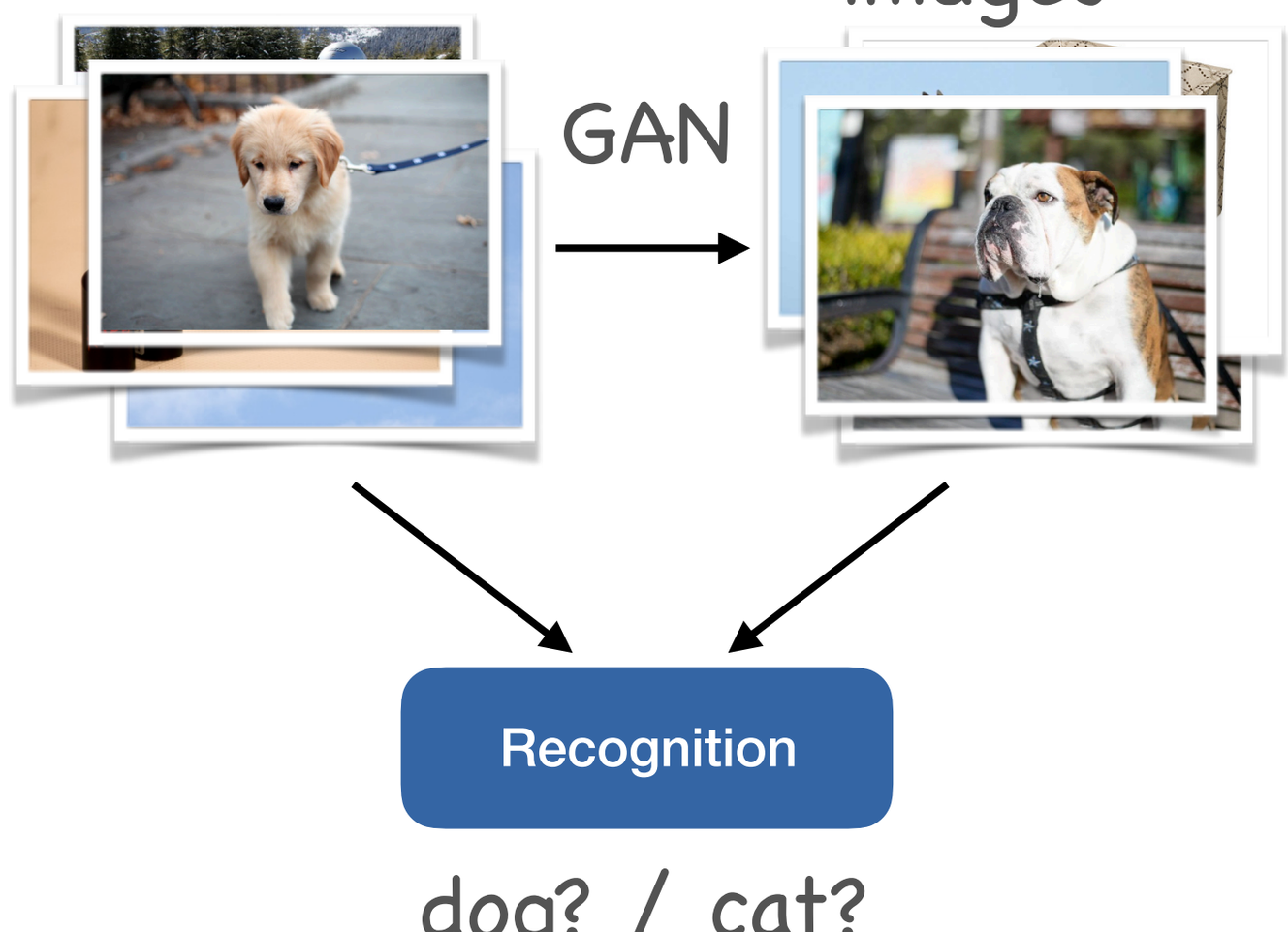
# Does this give a better recognition model?

- Argument against
  - Final training pipeline does not see more data



# S

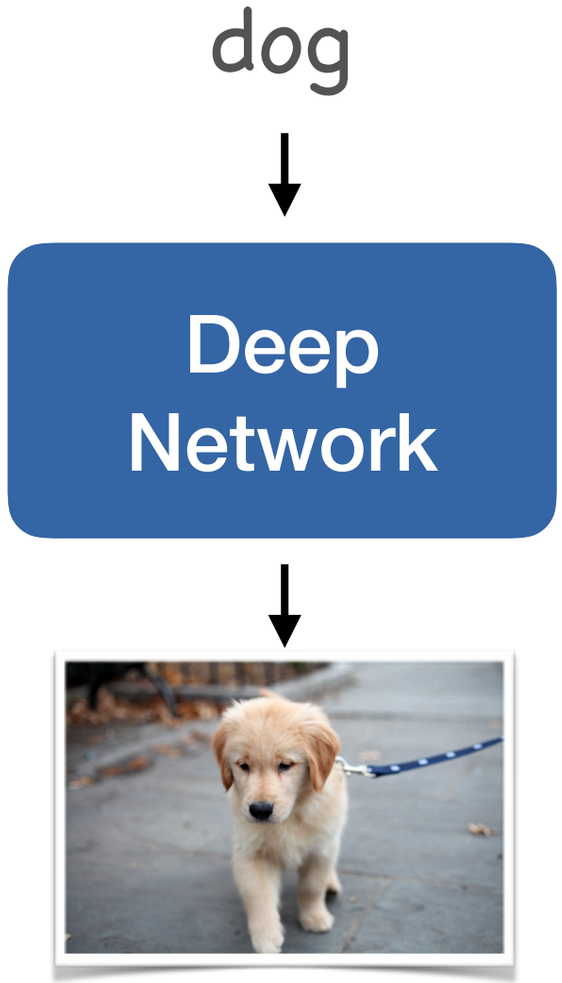
- Counter argument
  - Maybe GAN generalizes better?
- Counter counter argument
  - Why?!



# Summary

# Generative models

- Produce image outputs
  - Quite hard
  - Deep networks and GANs are good tools



# Generative models

- Applications
  - Computer graphics
- What else?