EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Computer Graphics

# Machine Learning in Graphics and Vision

## - Generative Adversarial Networks -

SoSe 2018

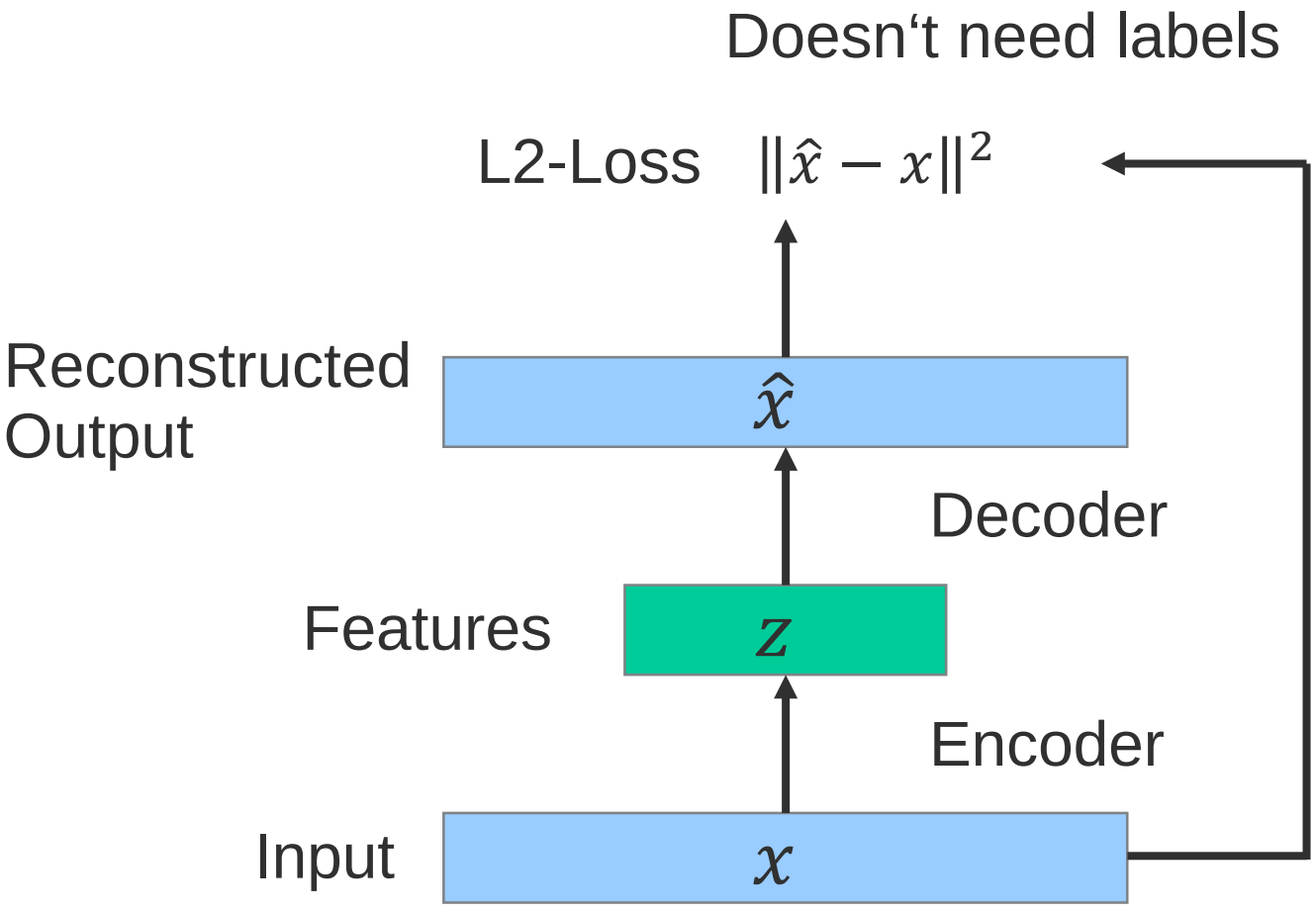
Hendrik Lensch

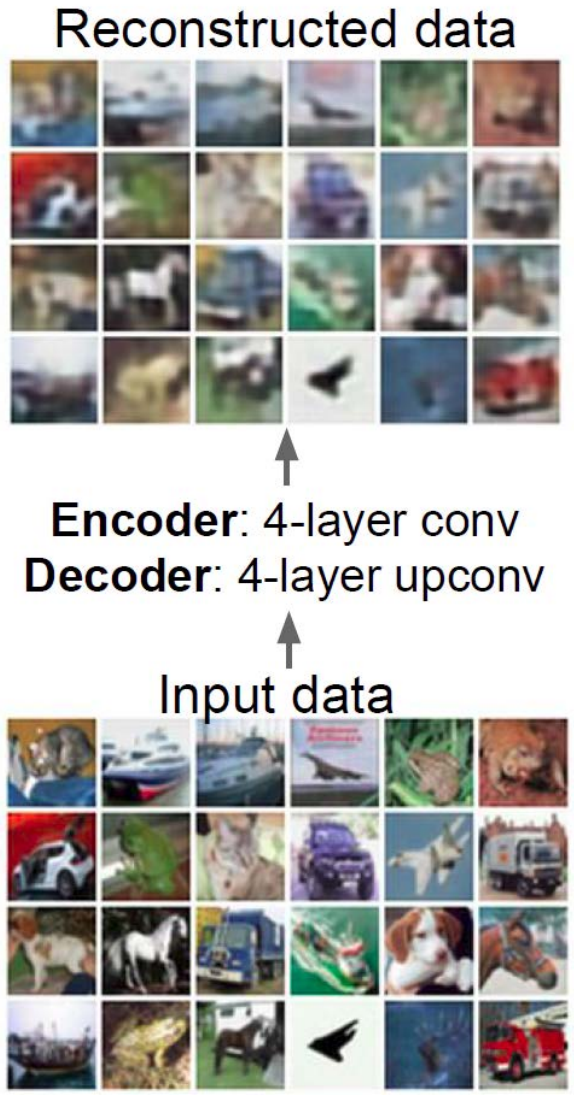
# Variational Autoencoder

[slides based on FeiFei et al. **Generative Models - CS231n**]

# Autoencoders – How to train?

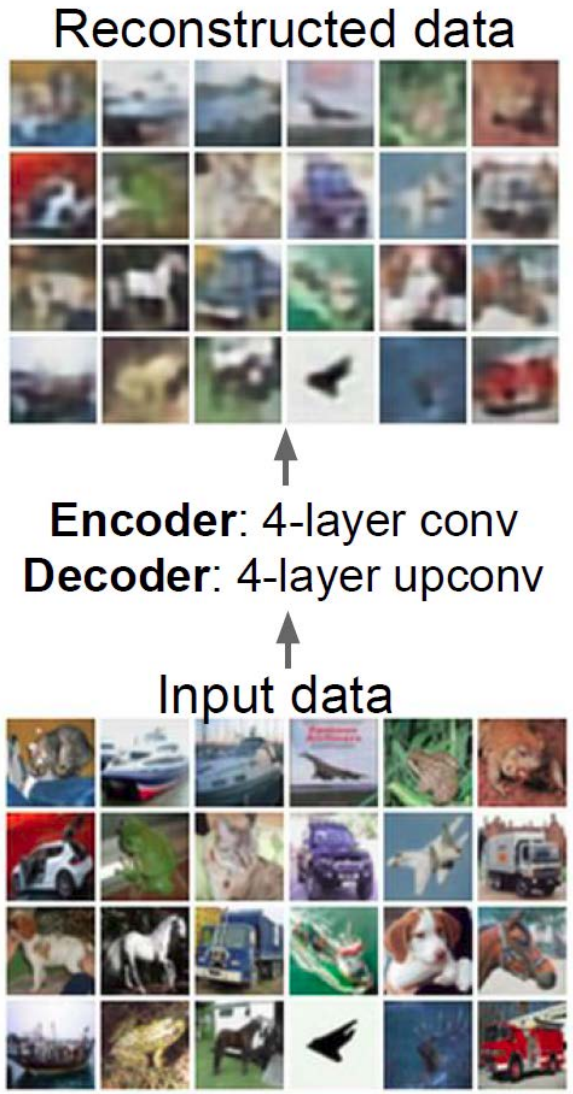
- Train such that that features can be used to reconstruct original data

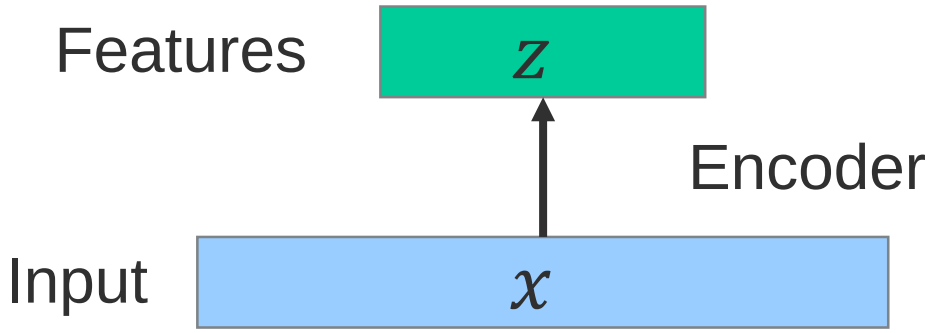


# Autoencoders – How to train?

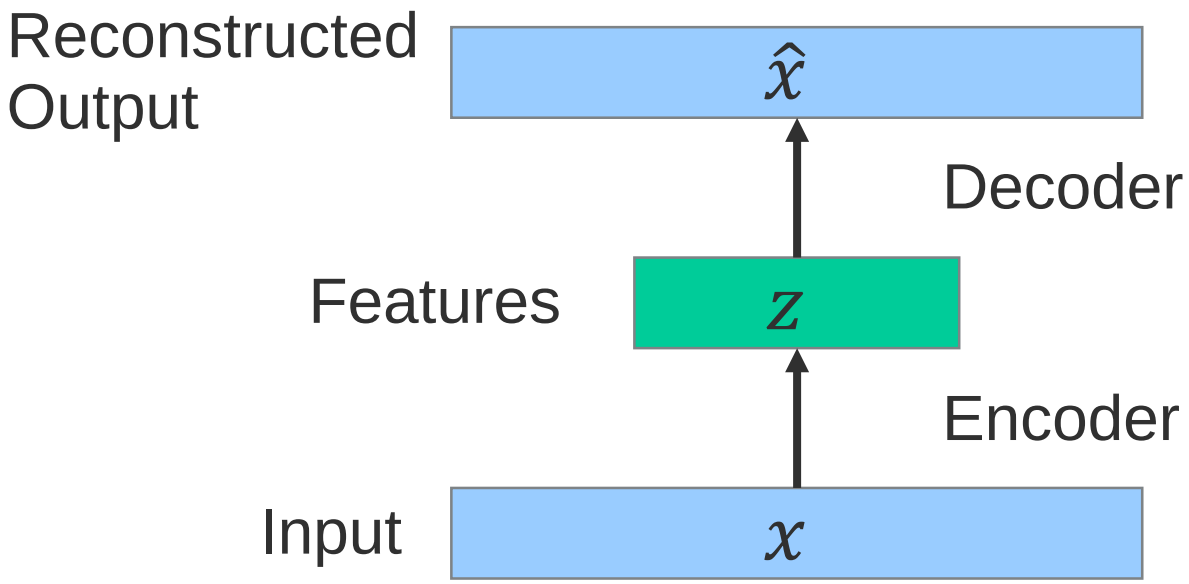
- Train such that that features can be used to reconstruct original data
- For compression: forget about the decoder after training



# Autoencoders – How to train?

- Autoencoders can reconstruct data
- Features capture factors of variation in training data.
- Can we generate new images from an autoencoder?

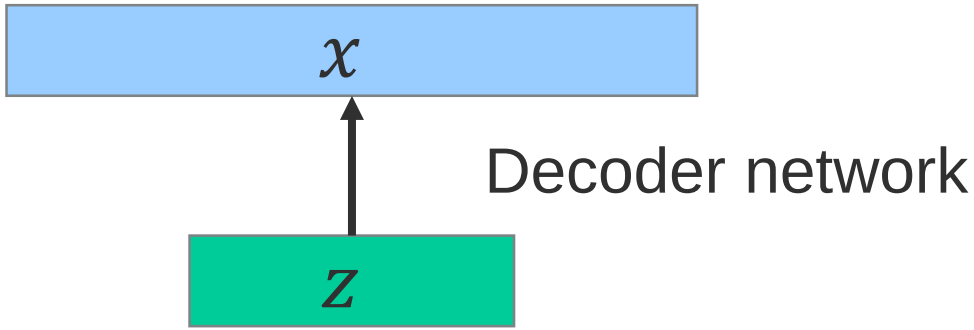
# Variational Autoencoders

- **Goal:** We want to estimate the true parameters $\theta^*$ of this generative model.

- How should we represent this model?
- Choose prior p(z) to be simple, e.g. Gaussian.
- Conditional p(x|z) is complex (generates image) → represent with neural network

Sample from true conditional $p_{\theta^*}(x|z^{(i)})$

Sample from true prior $p_{\theta^*}(z)$




[Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014]

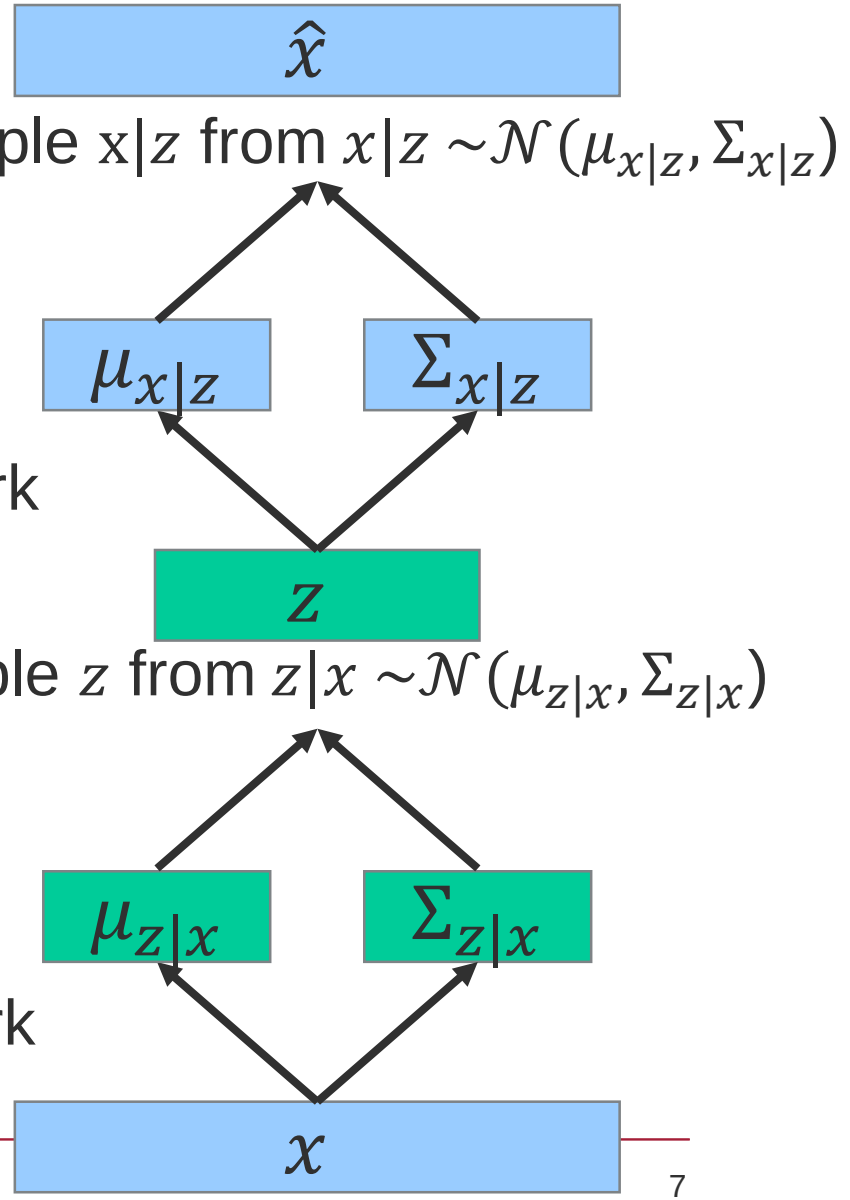
# Variational Autoencoders – Training – full cycle

- Maximize the likelihood lower bound
- **For every minibatch of input data: compute this forward pass, and then backprop!**

Maximize likelihood of original input being reconstructed

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

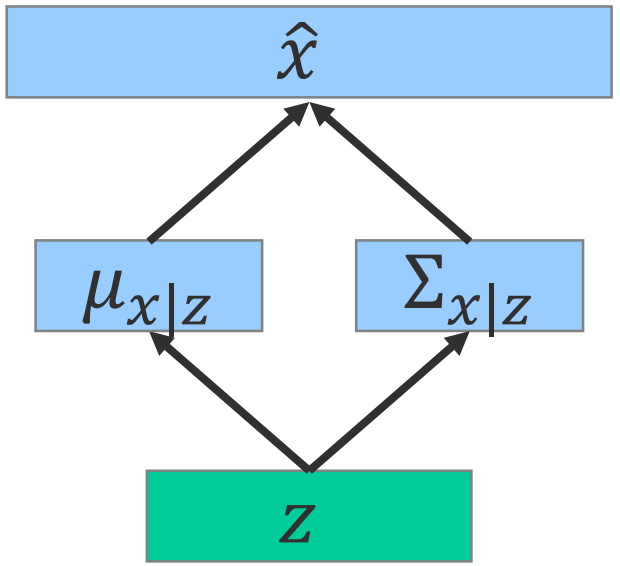
Make approximate posterior distribution close to prior, e.g. normal



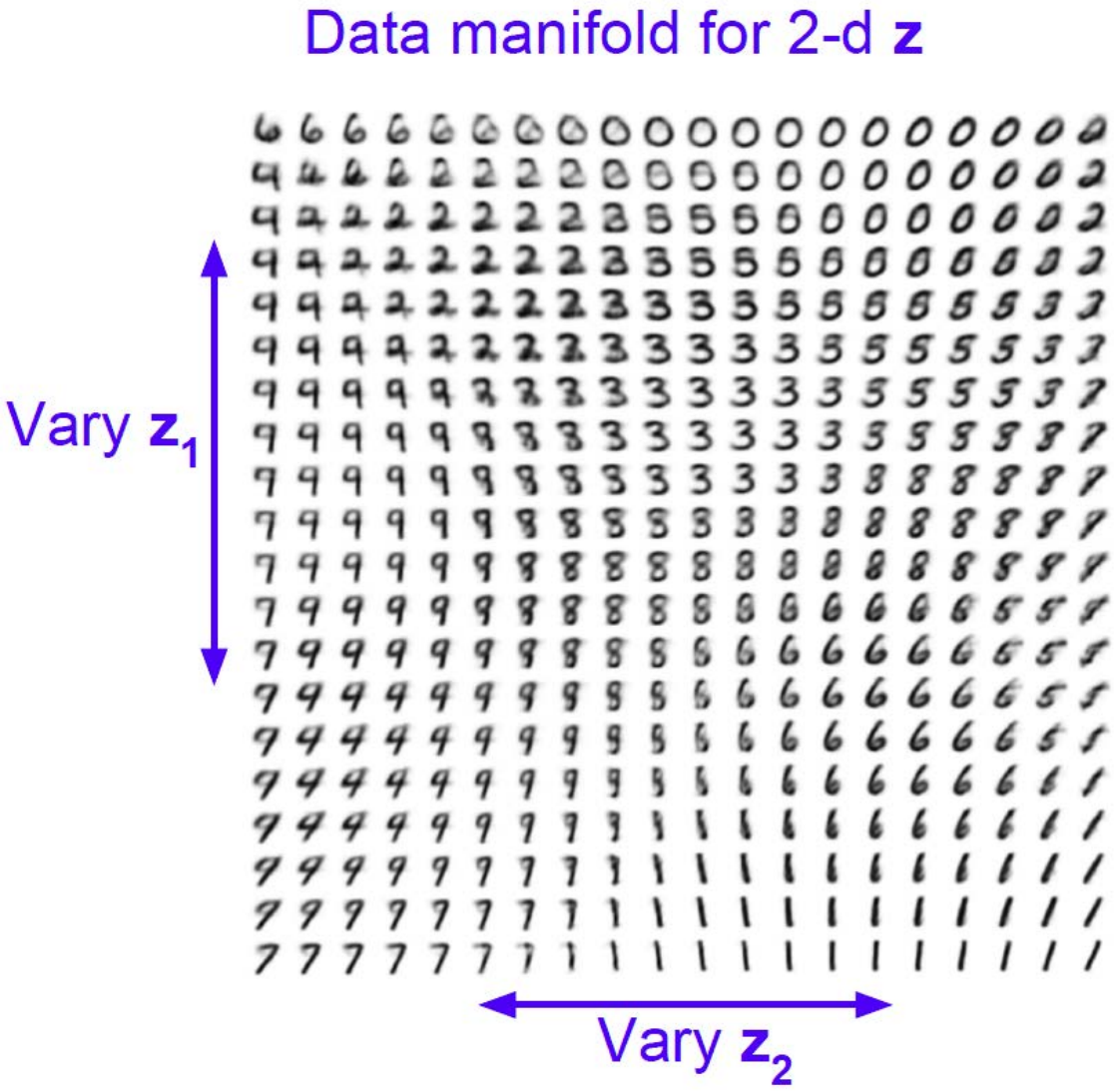
# Variational Autoencoders – Generating Data

- Sample $z$ from prior, then apply decoder

Decoder network
$p_\theta(x|z)$



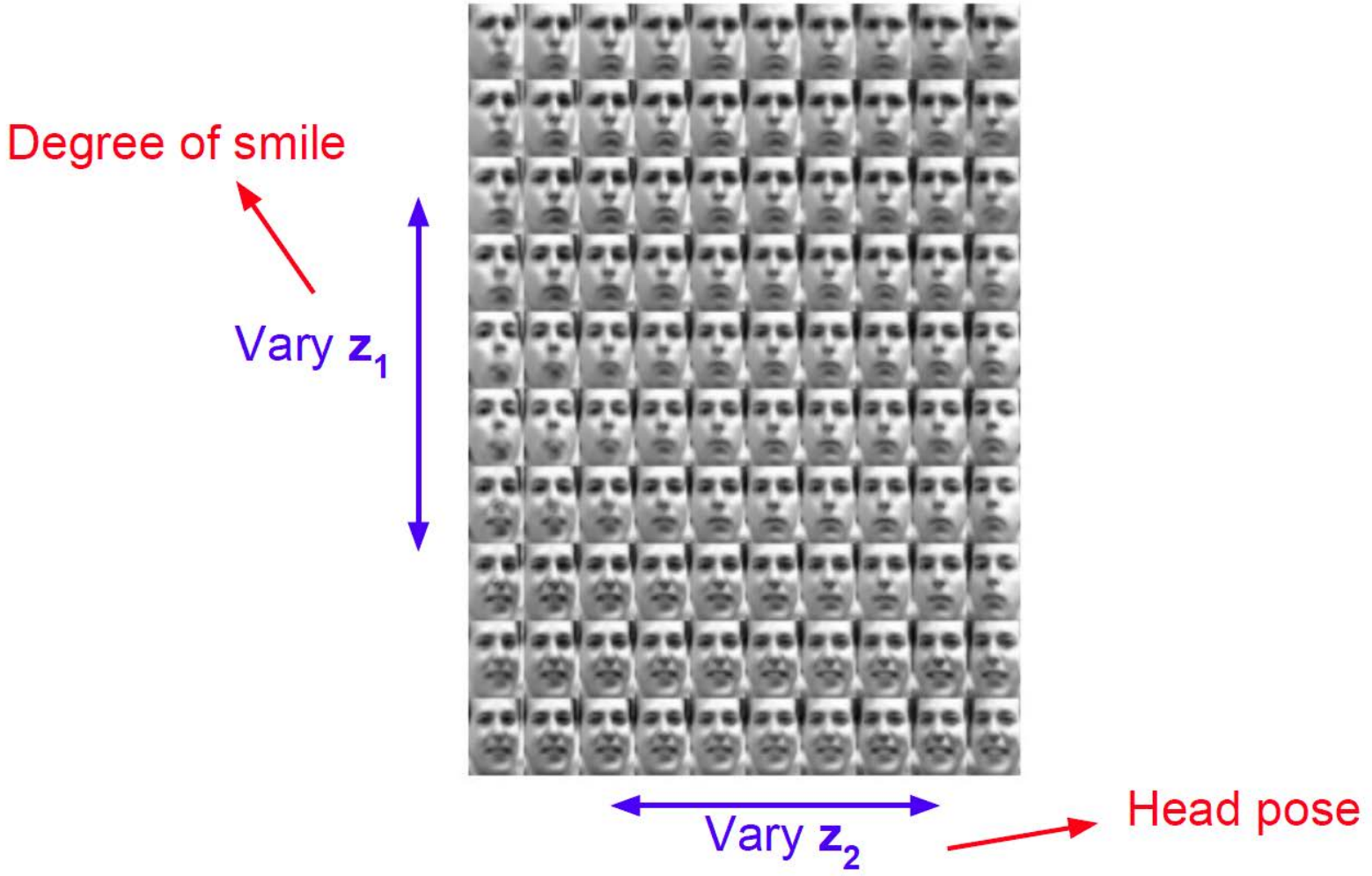

Sample $z$ from $\mathcal{N}(0,1)$



# VAE - Results

- Diagonal prior on $z \rightarrow$ independent latent variables

- Different dimensions of $z$ encode interpretable factors of variation



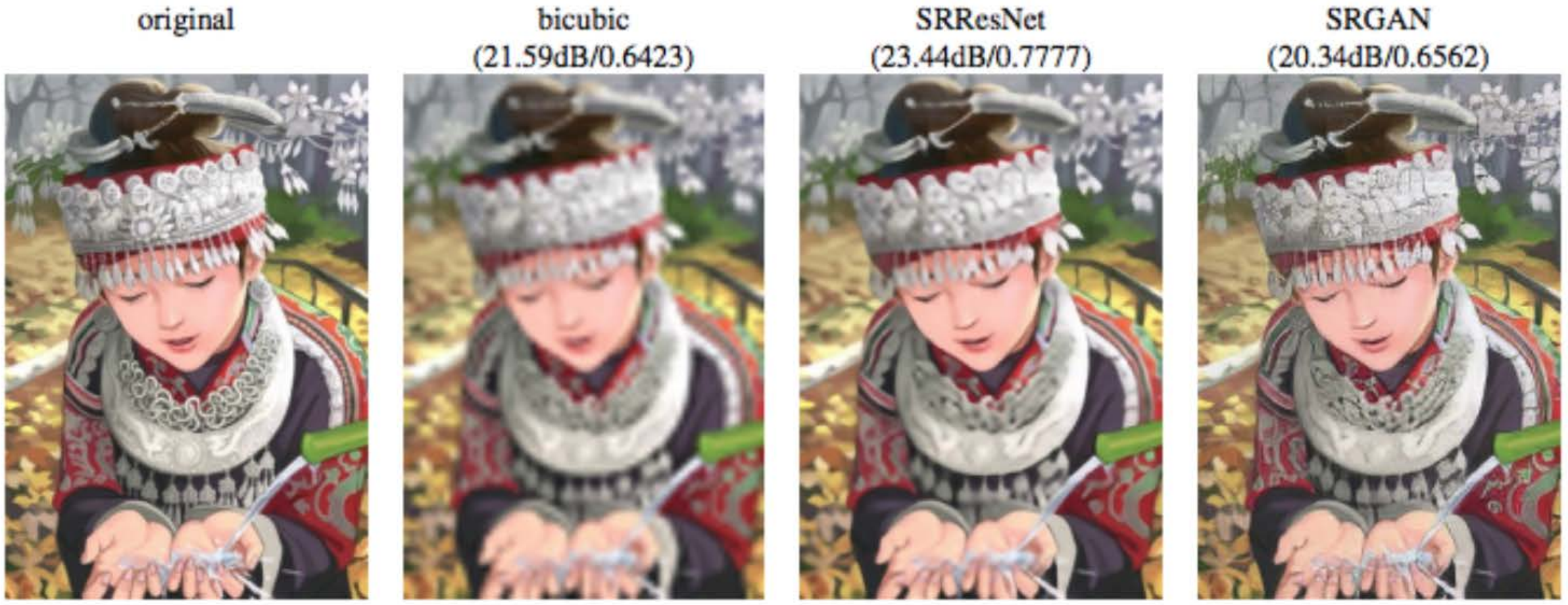
# GANs

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

Some slides from FeiFei et al., Goodfellow,

# GAN – Single Image Superresolution






[Ledig 2016]

# GAN: Image to Image Translation



[Isola et al. 2016]

# Taxonomy of Generative Modesl

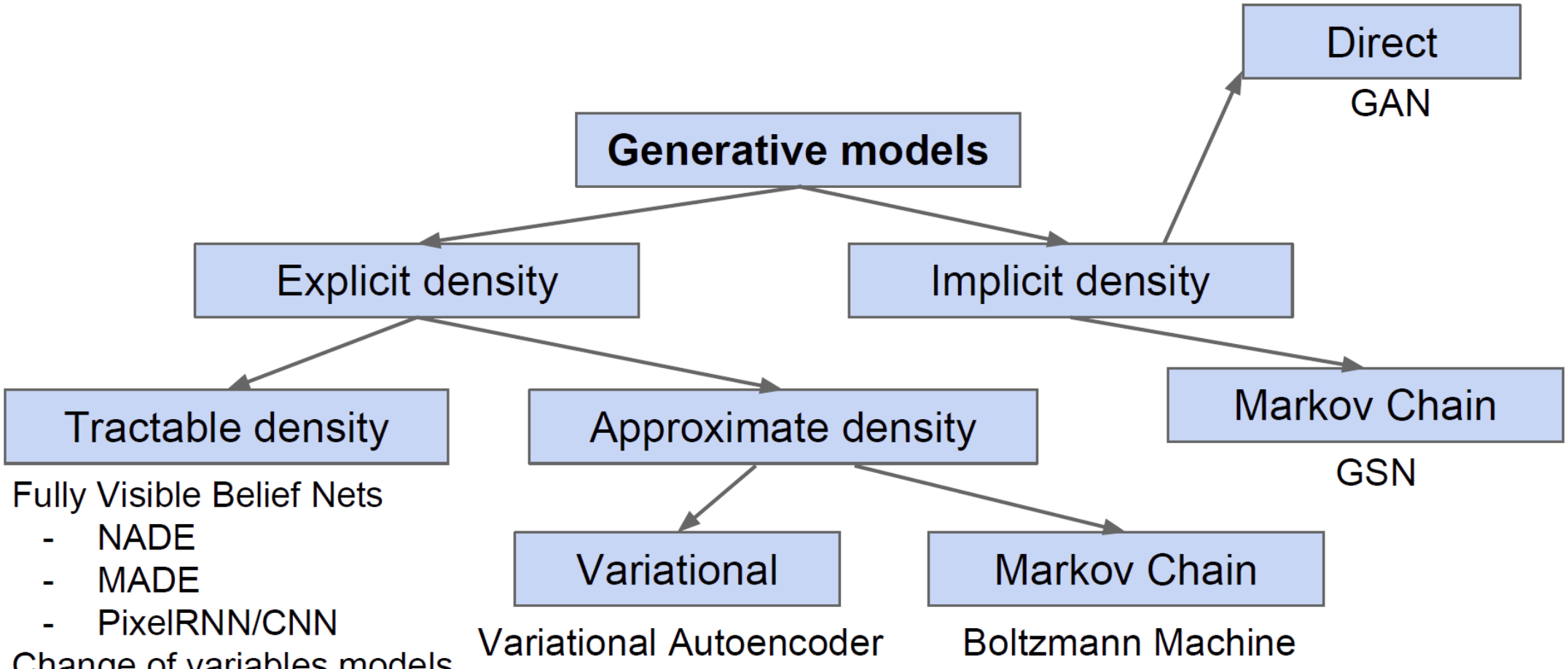
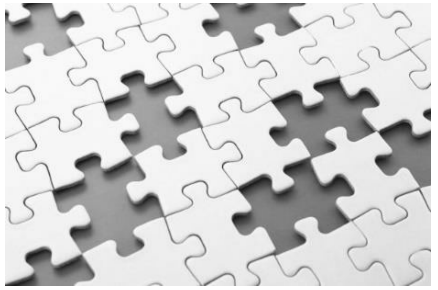



Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

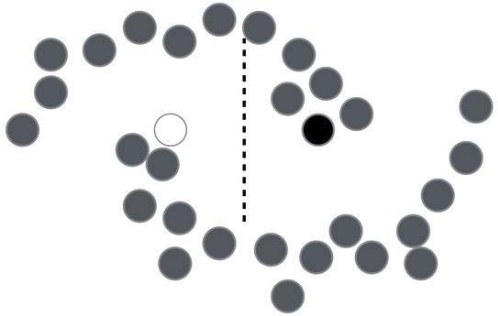
# Motivation: Generative Models

- Theoretical Reason:
- Excellent test of our ability to deal with high-dimensional diistributions


- Applications:
  - Simulate possible futures for planning
  - Missing data
  - Semi-supervised learning
  - Multi-modal outputs
  - Realistic generation takss







[images by Goodfellow]

# GANs idea

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- VAEs define intractable density function with latent z:

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

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

- What if we give up on explicitly modeling density, and just want ability to sample?
- **GANs: don't work with any explicit density function!**
- **Instead, take game-theoretic approach: learn to generate from training distribution through 2-player game**

- Use a latent code
- Asymptotically consistent (unlike variational methods)

# GANs idea

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

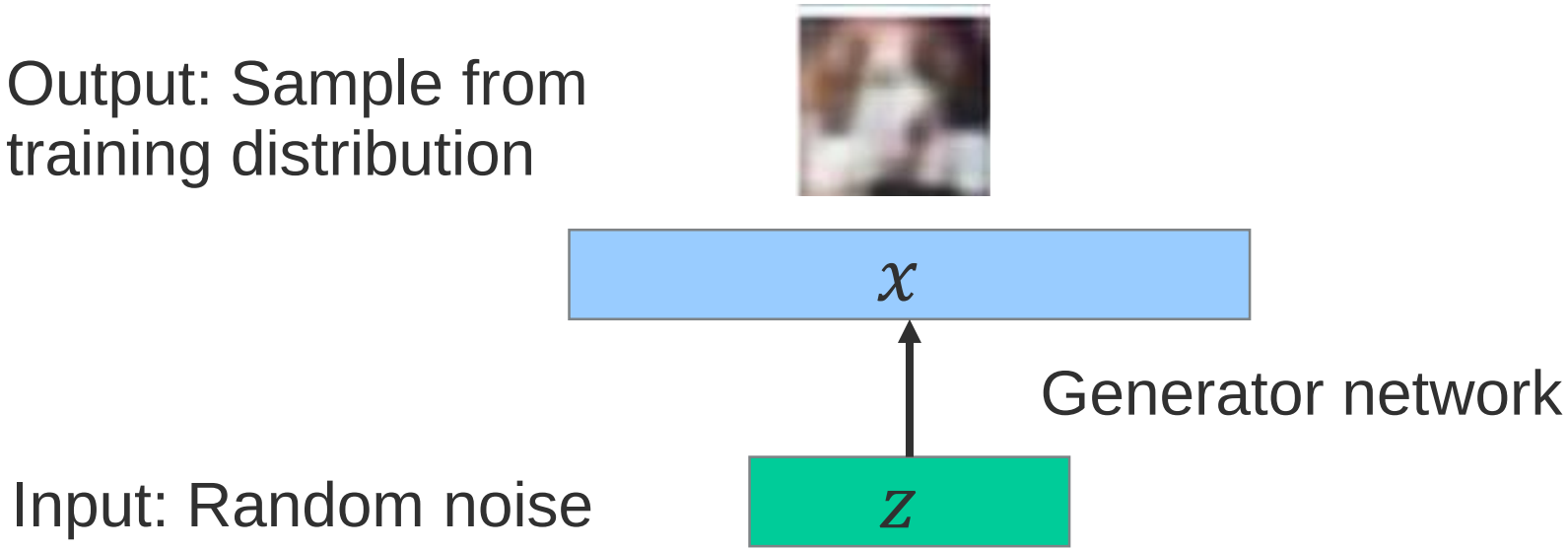
- Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!
- Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

- Q: What can we use to represent this complex transformation?

# GANs idea

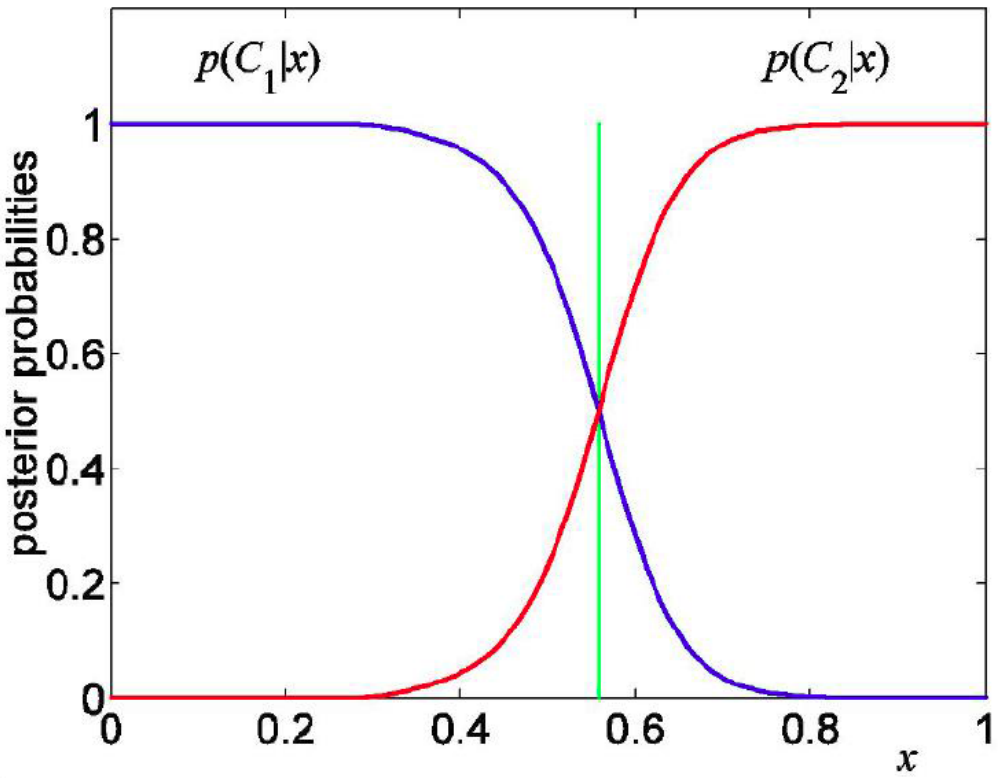
- Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!
- Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

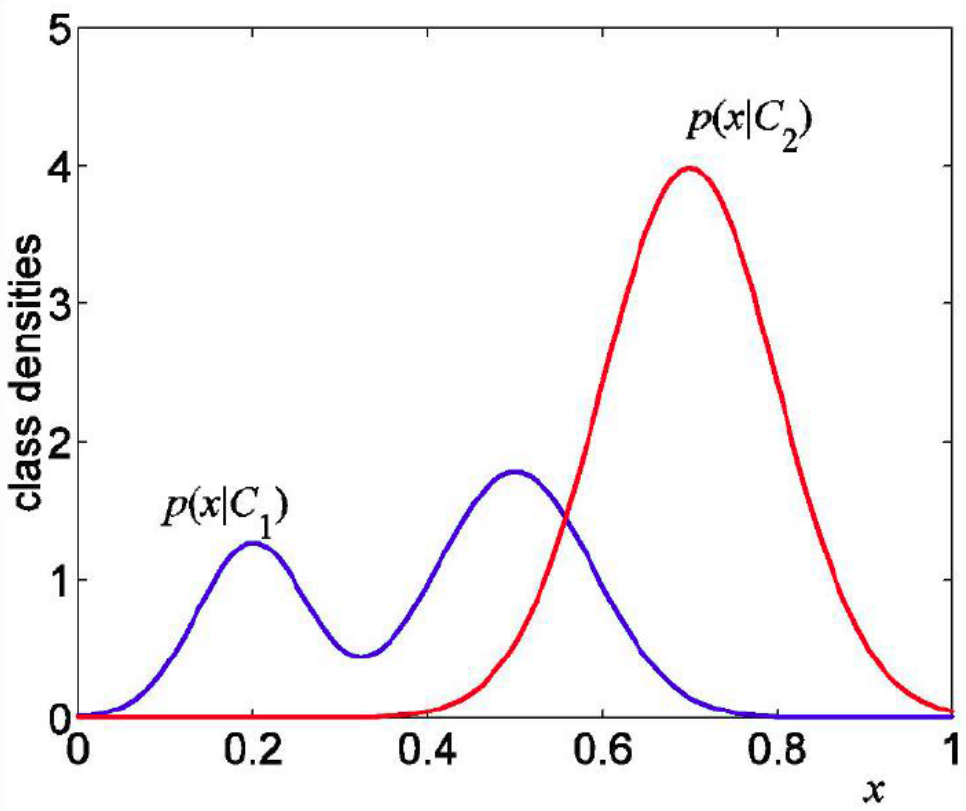
- Q: What can we use to represent this complex transformation?



# Training GANs: Two-Player Game

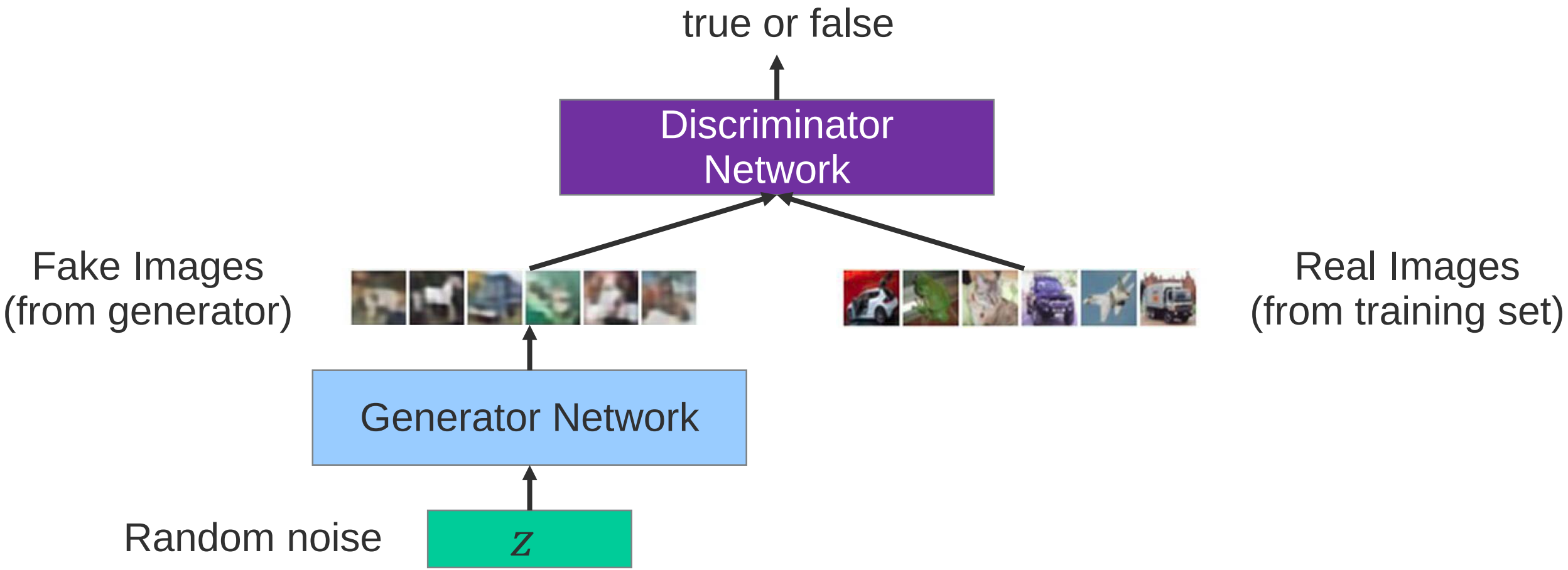
- **Discriminator modeling:** $\boldsymbol{p(C|X)}$
- **Generative modeling:** $p(x|C)$ and $p(x)$



# Training GANs: Two-Player Game

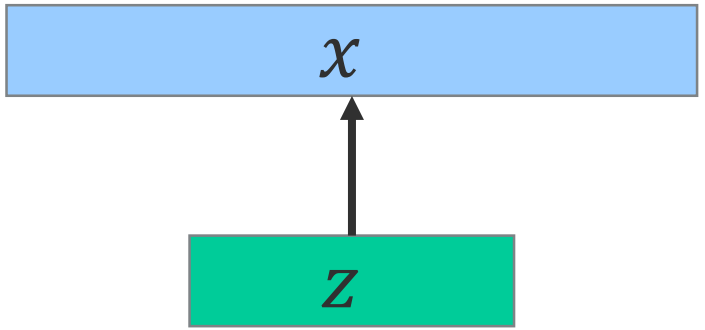
- **Generator network**: try to fool the discriminator by generating real-looking images
- **Discriminator network**: try to distinguish between real and fake images



# GAN – Generator Network

$$x = G(z; \theta^G)$$

- Must be differentiable
- No invertibility requirement
- Trainable for any size of z
- Can make x conidtionally Gaussian given z but need not do so



# Training GANs: Two-Player Game

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- **Generator network**: try to fool the discriminator by generating real-looking images
- **Discriminator network**: try to distinguish between real and fake images
  - Discriminator outputs likelihood in (0,1) of real image

- Train jointly in **minimax game**
- Minimax objective function:

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

- Discriminator ($\theta_D$) wants to maximize objective such that $D(x)$ is close to 1 (real) and $D(G(z))$ is close to 0 (fake)
- Generator ($\theta_G$) 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

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Minimax objective function:

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

- Alternate between:

1. **Gradient ascent** on discriminator

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

2. **Gradient descent** on generator

$$\min_{\theta_G} \left[ \mathbb{E}_{z \sim p(z)} \log\left(1 - D_{\theta_D}(G_{\theta_G}(z))\right) \right]$$

# Training GANs: Two-Player Game

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Minimax objective function:

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

- Alternate between:

1. **Gradient ascent** on discriminator

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

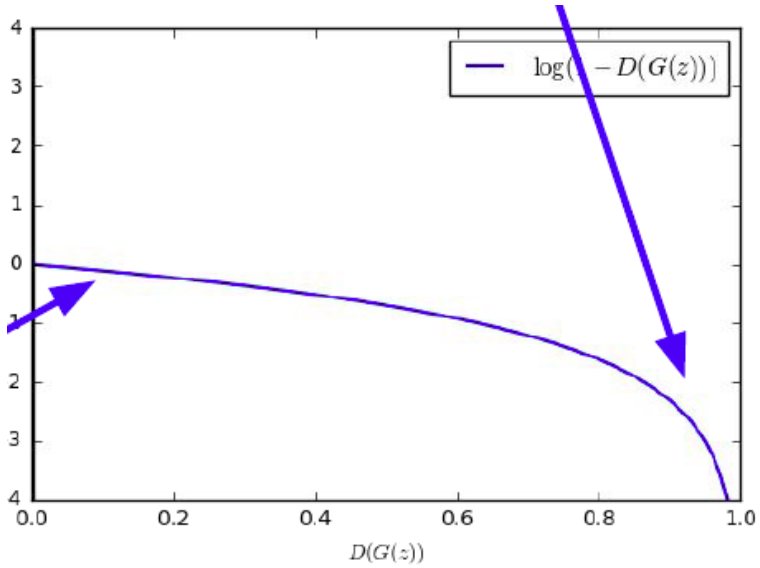
Gradient signal dominated by region where sample is already good

2. **Gradient descent** on generator

$$\min_{\theta_G} \left[ \mathbb{E}_{z \sim p(z)} \log\left(1 - D_{\theta_D}(G_{\theta_G}(z))\right) \right]$$

in practise, optimizing this generator objective does not work well!

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

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Minimax objective function:

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

- Alternate between:

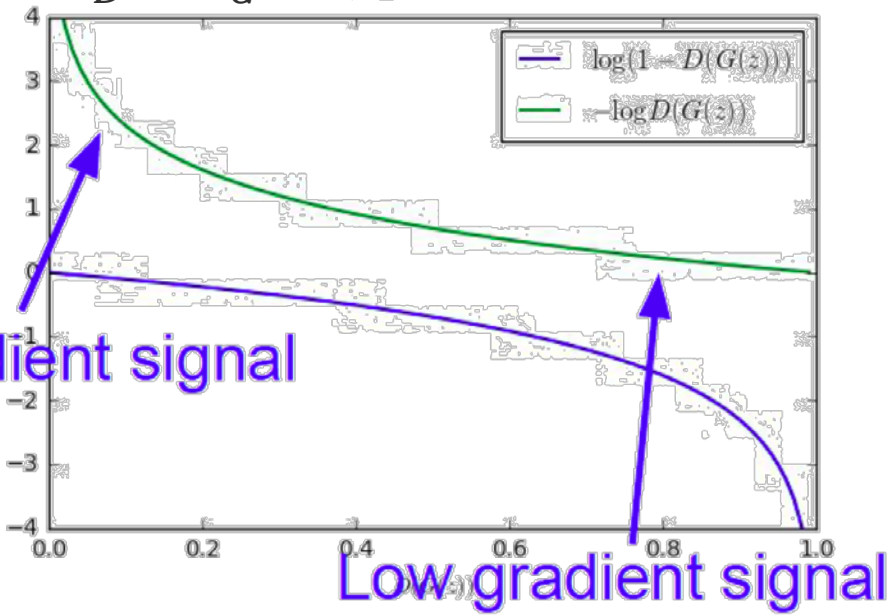
1. **Gradient ascent** on discriminator

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

2. **Gradient ascent** on generator, different objective

$$\max_{\theta_G} \left[ \mathbb{E}_{z \sim p(z)} \log\left(D_{\theta_D}(G_{\theta_G}(z))\right) \right]$$

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.



# GAN Training Algorithm



**for** number of training iterations **do**

**for** $k$ steps **do**

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

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

**end for**

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

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

**end for**

# GAN Challenges

- Jointly training two networks is challenging, can be unstable. Choosing objectives with better loss landscapes helps training, is an active area of research.

- Balancing discriminator and generator is difficult
  - Selecting size of minibatch

- Recent work (e.g. Wasserstein GAN) changes the objective even further

# Non-convergence

- Optimization allgorithms often approach a saddle point or local minimum rather than a global minimum
- Game solving algroithms may not approach an equilibrium at all
- Instead of modifying density the loss functions are modified
- D and G are represented as highly non-convex parametric functions
- „Osscillation“ can train vor a very long time, generating very many different categories of samples without cleearly generating better samples.

# Mode Collapse in GANs

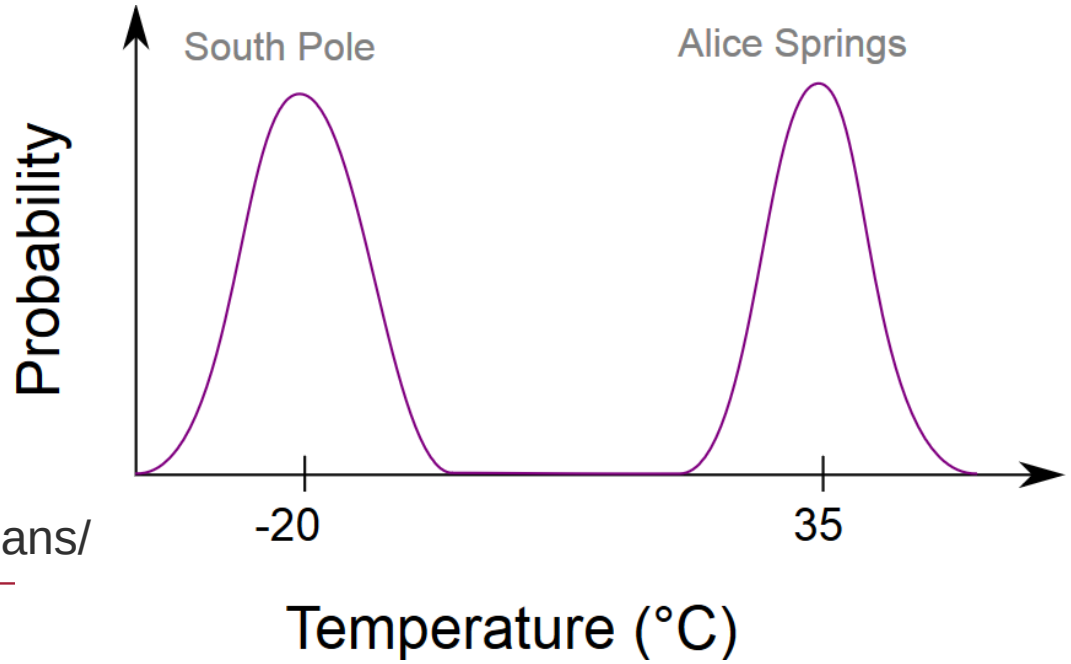
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Commonly encountered failure case for GANs: the generator learns to produce samples with extremely low variety.

Example:

1. The generator learns that it can fool the discriminator into thinking that it is outputting realistic temperatures by producing values close to Antarctic temperatures.
2. The discriminator counters by learning that all Australian temperatures are real (not produced by the generator), and essentially guesses whether Antarctic temperatures are real or fake since they are indistinguishable.
3. The generator exploits the discriminator by switching modes to produce values close to Australian temperatures instead, abandoning the Antarctic mode.
4. The discriminator now assumes that all Australian temperatures are fake and Antarctic temperatures are real.
5. Return to step 1.

http://aiden.nibali.org/blog/2017-01-18-mode-collapse-gans/



# Mode Collapse in GANs

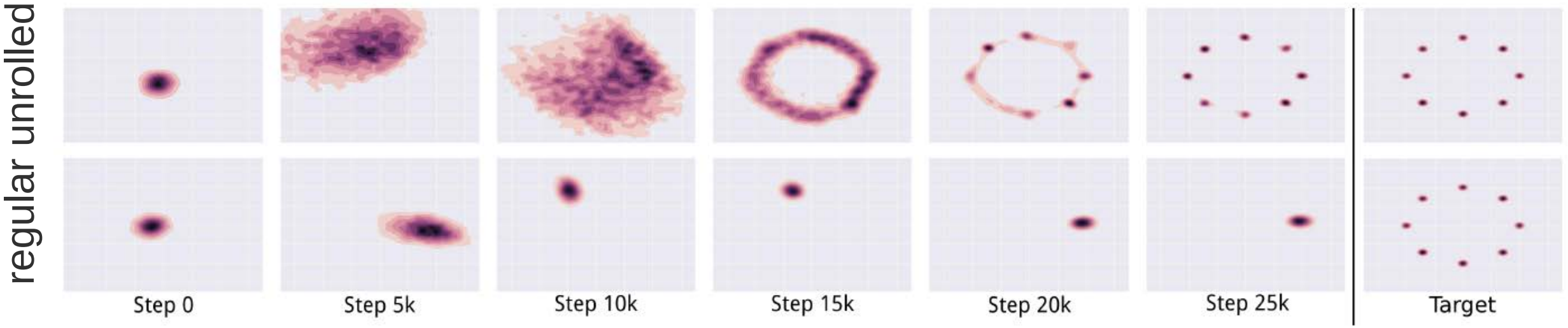
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Directly encourage diversity
  - Minibatch discrimination gives the discriminator the power of comparing samples across a batch to help determine whether the batch is real or fake.
- Anticipate counterplay
  - Unrolled GANs take this kind of approach by allowing the generator to “unroll” updates of the discriminator in a fully differentiable way. Now instead of the generator learning to fool the current discriminator, it learns to maximally fool the discriminator *after it has a chance to respond*
- Use experience replay
  - Hopping back and forth between modes can be minimized by showing old fake samples to the discriminator every so often.
- Use multiple GANs
  - The individual GANs might collapse to different modes.

http://aiden.nibali.org/blog/2017-01-18-mode-collapse-gans/

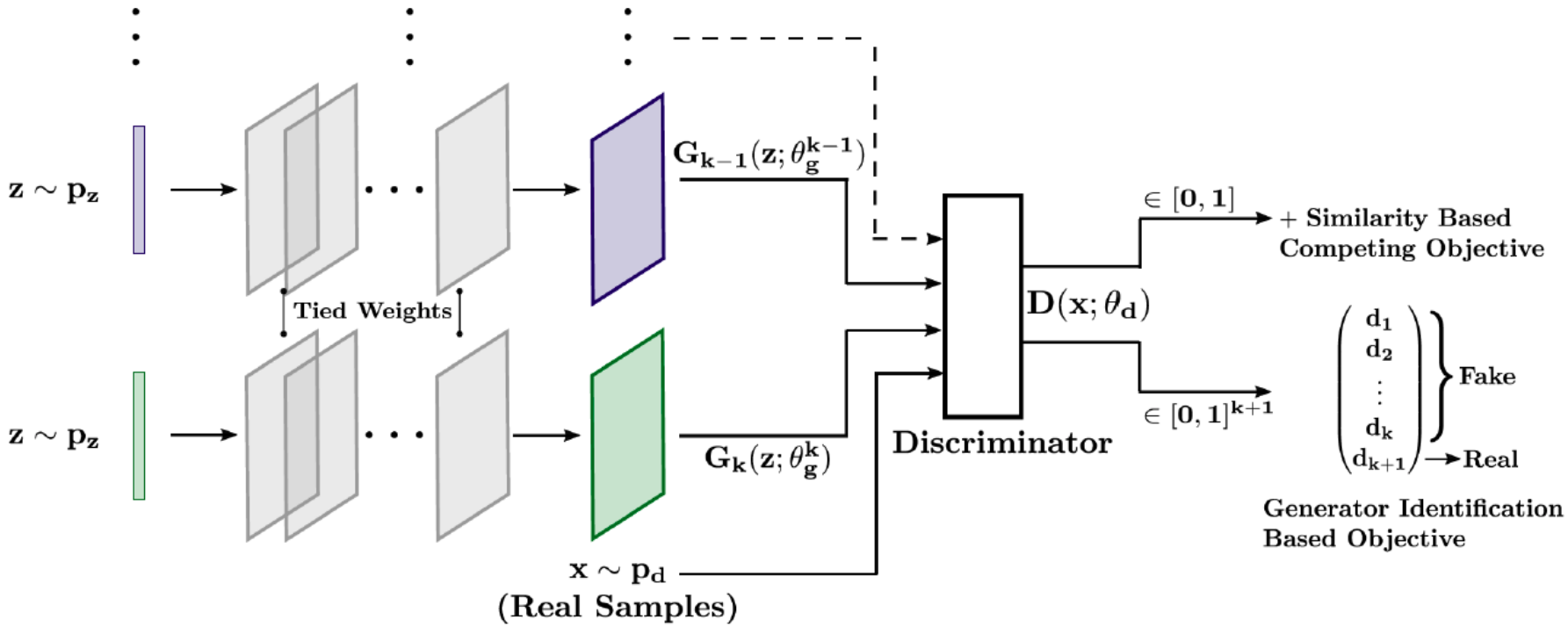
# Mode Collapse

- D in inner loop: convergence to correct distribution
- G in inner loop: place all mass on most likely point




[Metz et al. 2017 – Unrolled Generative Adversarial Networks]

# Use Multiple GANs




- multiple parallel generators
- share parameters up to layer l
- applies a diversity loss on different generators
- alternatively, have D predict which generator the fake sample came from!

[Ghosh et al. Multi-Agent Diverse Generative Adversarial Networks (2017) ]

# Generative Adversarial Nets: Convolutional Architectures



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

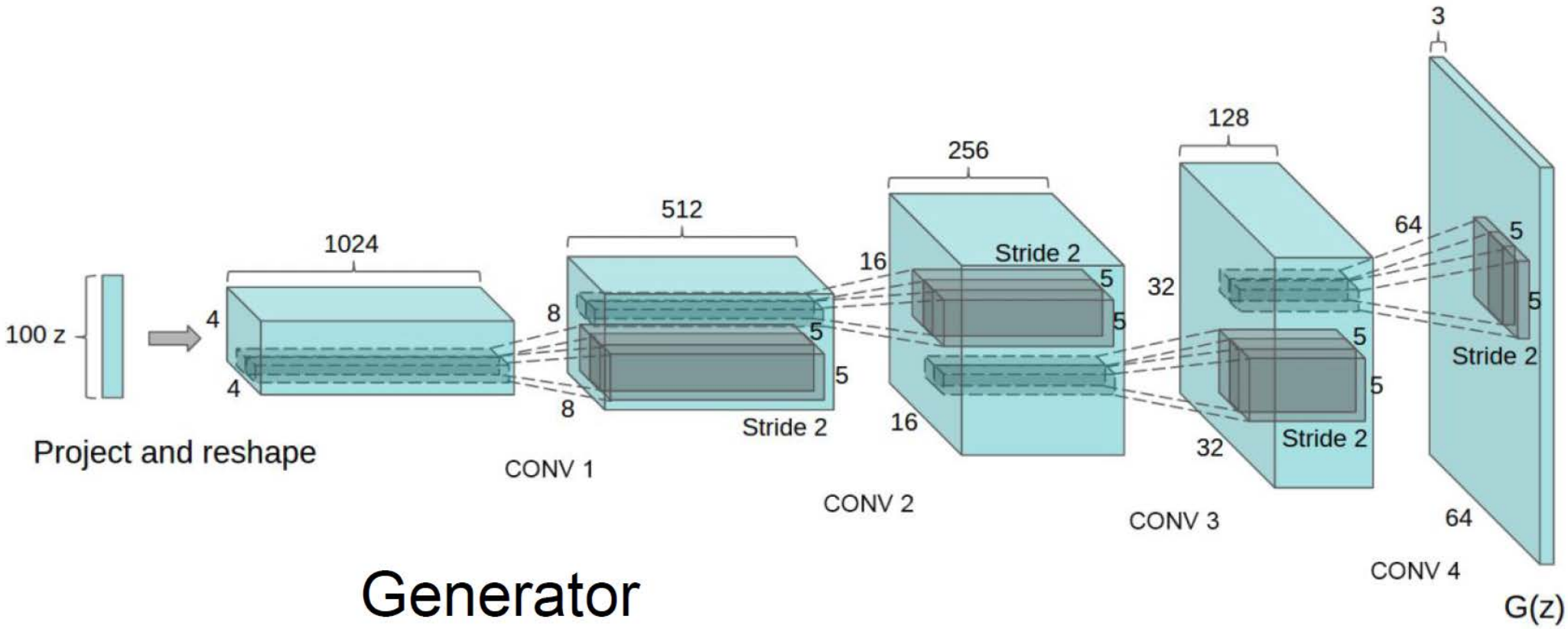
Architecture guidelines for stable Deep Convolutional GANs

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

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

# Generative Adversarial Nets: Convolutional Architectures

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




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

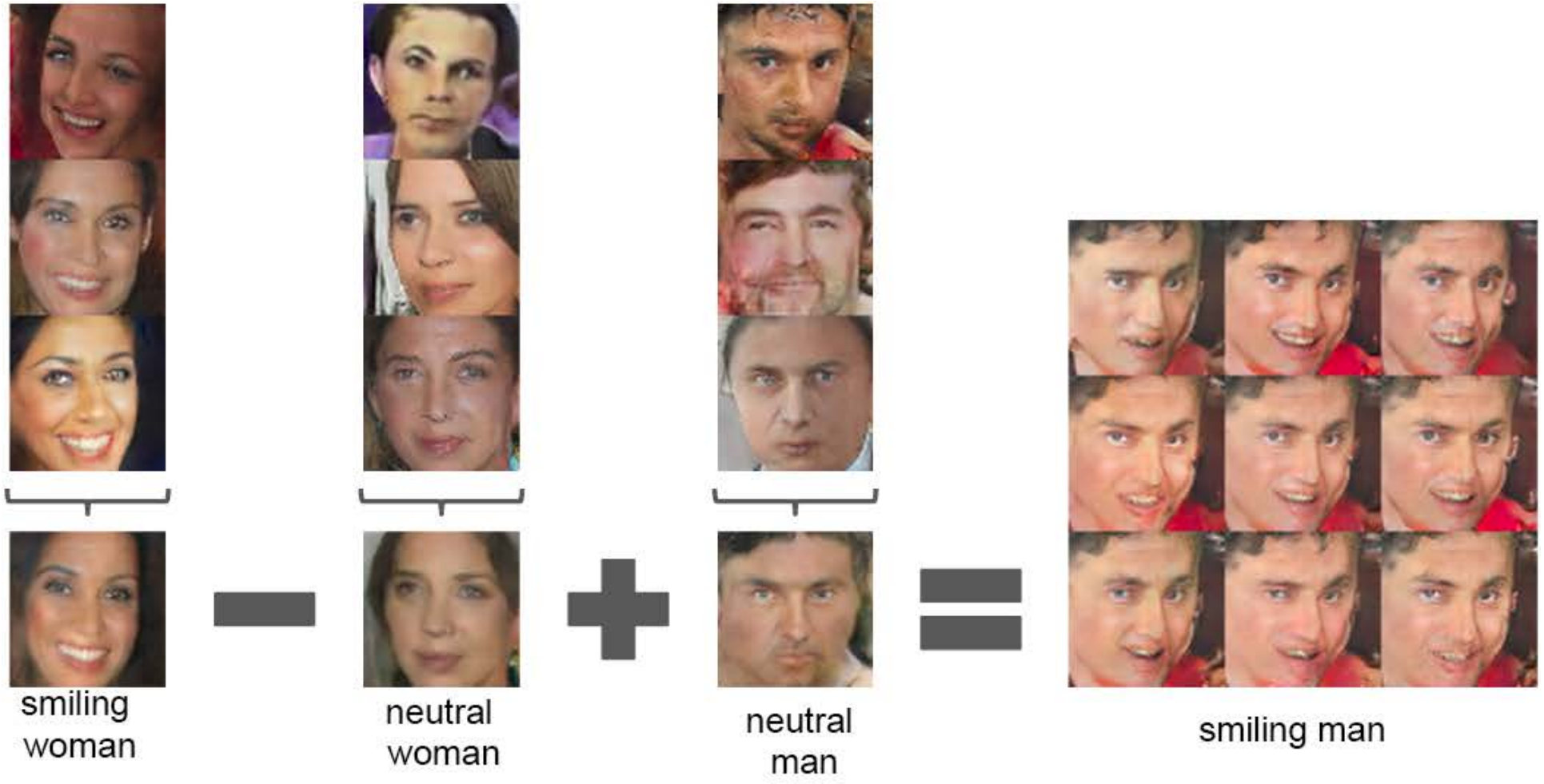
# GAN: Latent Space

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

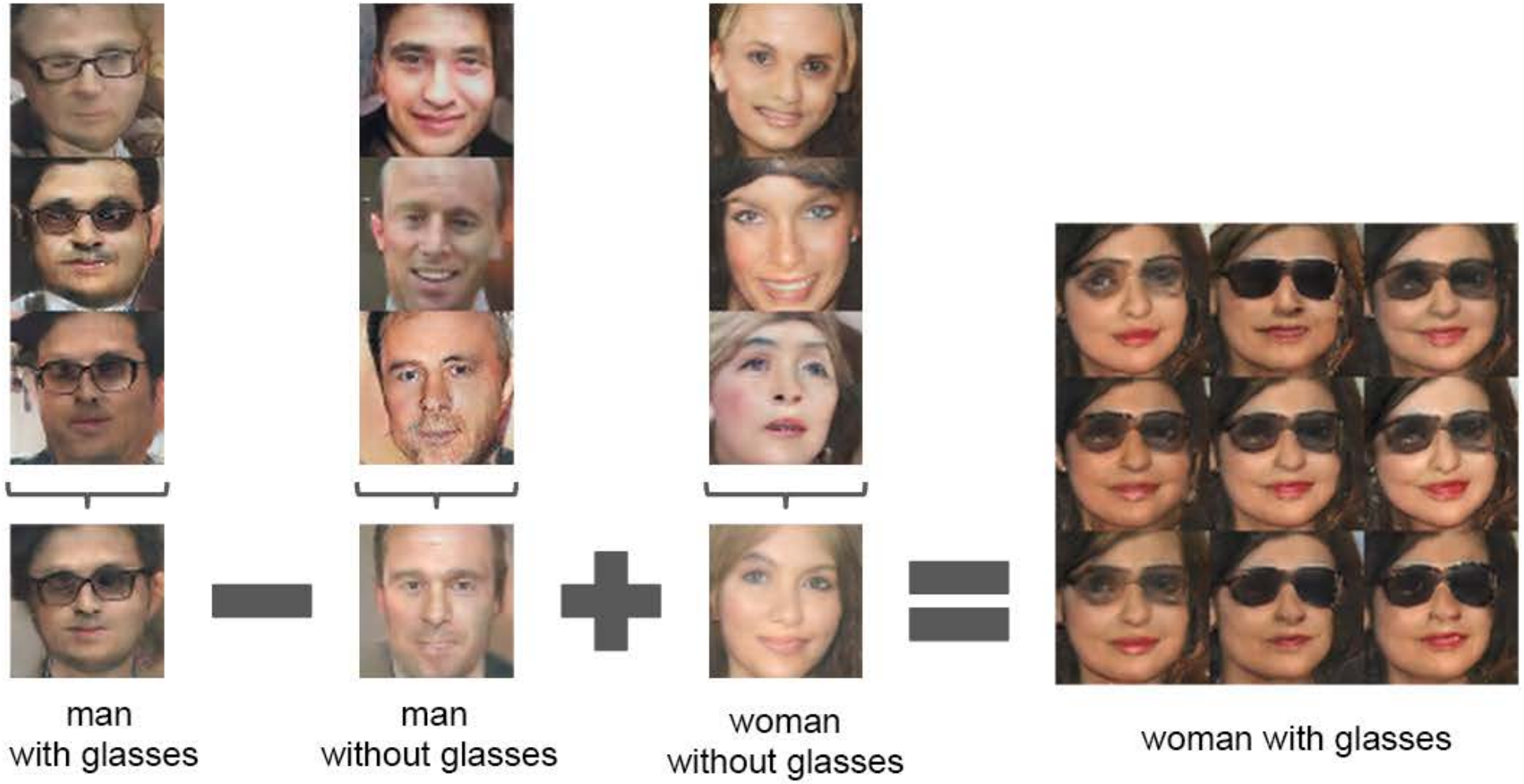
- Interpolating between random points in latent space

Radford et al, 2016

# Interpretable Vector Math



# Interpretable Vector Math



# Info GAN

[Chen et al. 2016]

# Info-GAN

- Create loss function such that the latent parameters become independent of each other
- Latent vector $(z, c)$ consisting of random noise $z$ and latent codes $c$
- Maximize the mutual information (as differences of entropies)

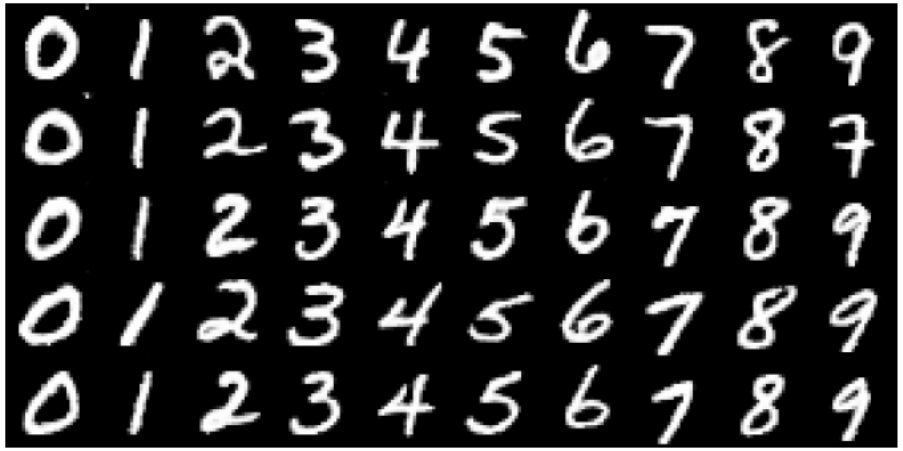
$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

$$\min_G \max_D V_I(D,G) = V(D,G) - \lambda I(c; G(z,c))$$

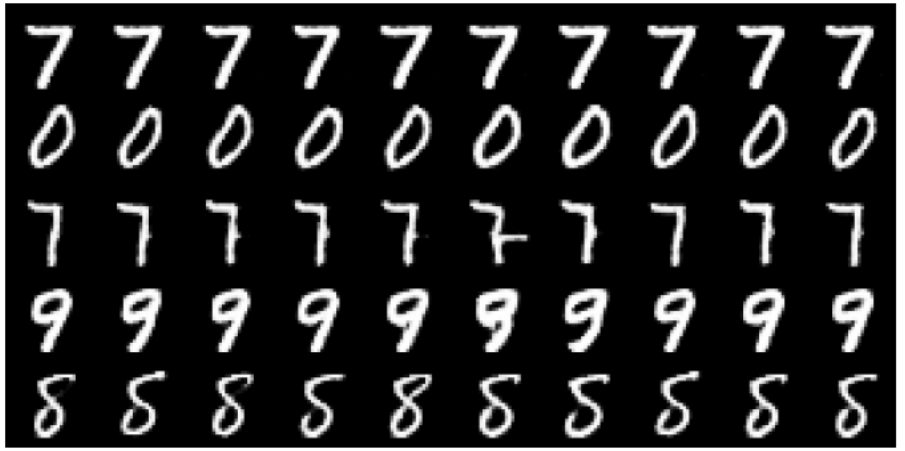
- Often extracts semantically meaningful directions

$$\min_{G,Q} \max_D V_{InfoGAN}(D,G,Q) = V(D,G) - \lambda L_I(G,Q)$$

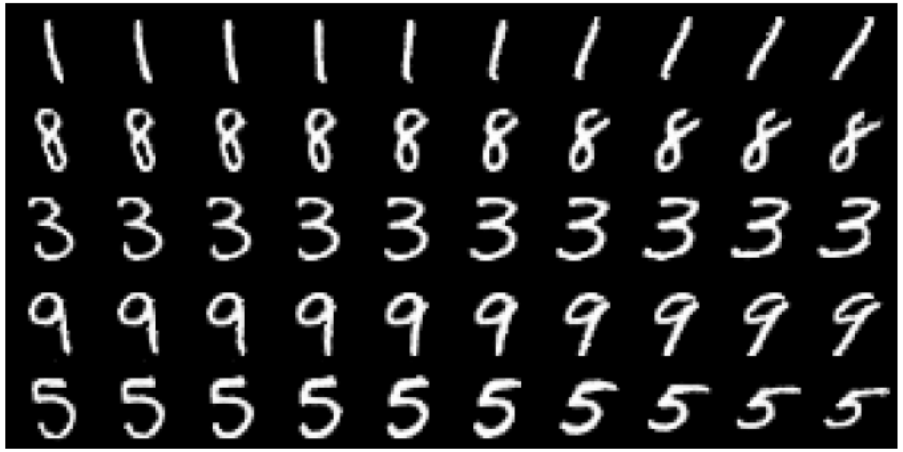
# Effect of Latent Codes



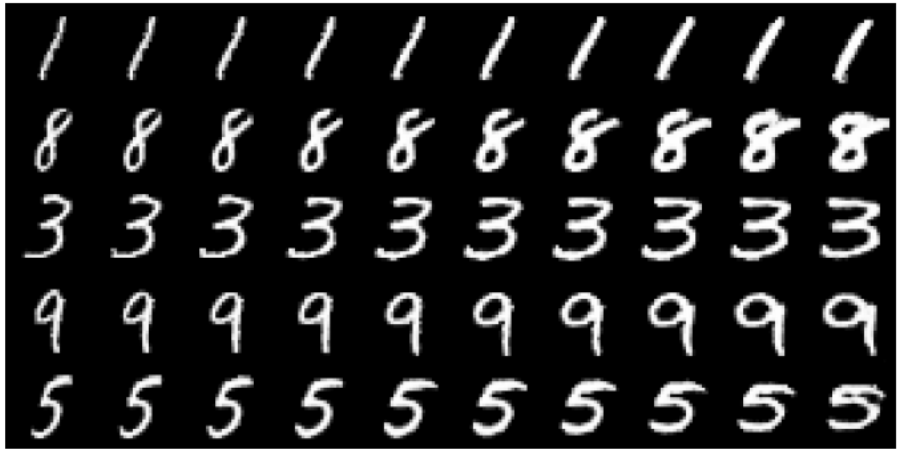
(a) Varying $c_1$ on InfoGAN (Digit type)



(b) Varying $c_1$ on regular GAN (No clear meaning)



(c) Varying $c_2$ from $-2$ to 2 on InfoGAN (Rotation)

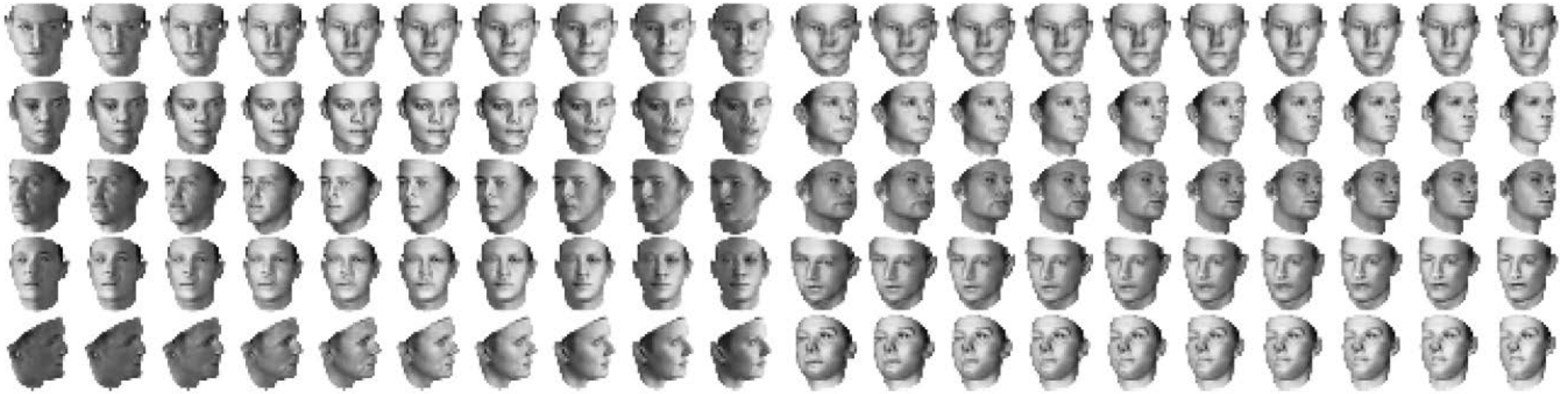


(d) Varying $c_3$ from $-2$ to 2 on InfoGAN (Width)

# Effect of Latent Codes

(a) Azimuth (pose)

(b) Elevation

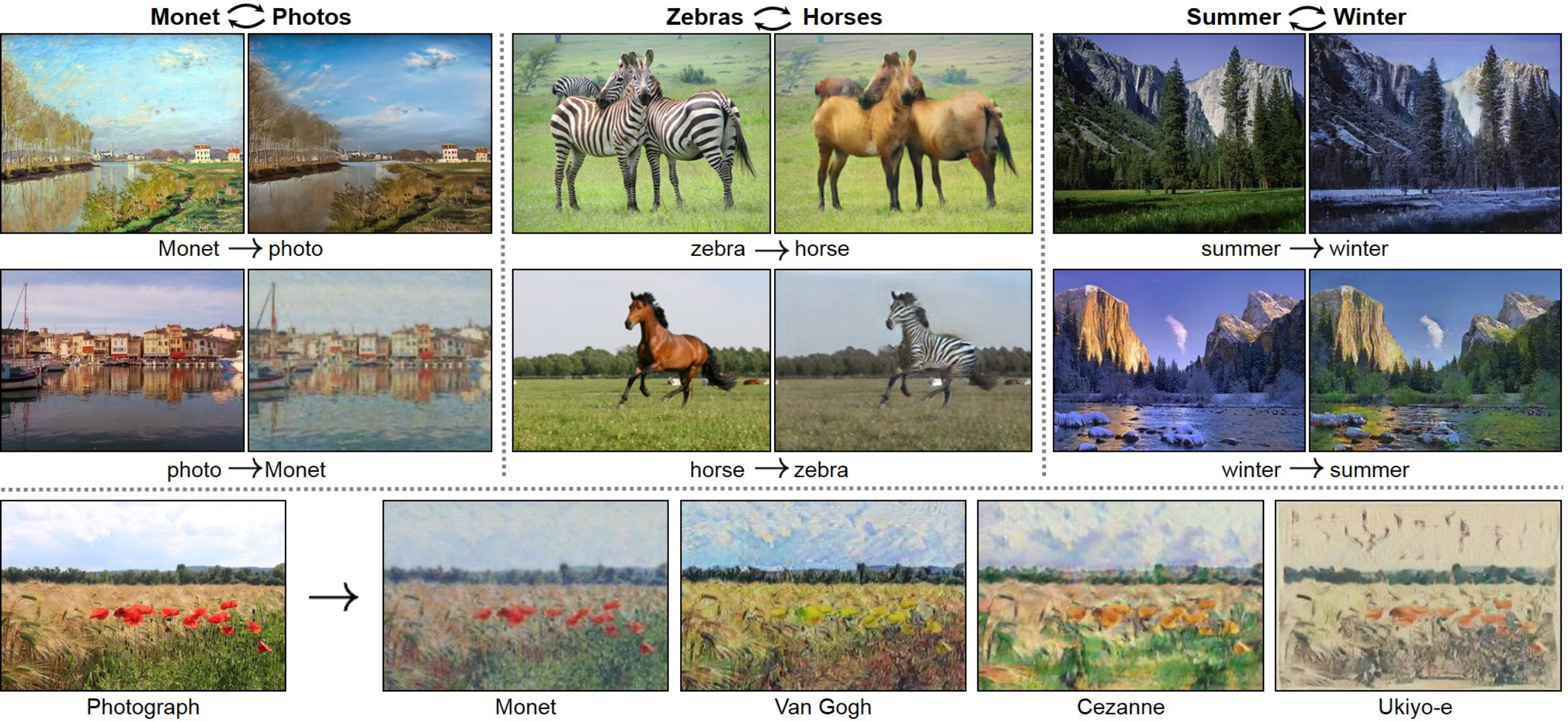


(c) Lighting

(d) Wide or Narrow


# CycleGAN

[Zhu et al. 2017]


Monet ⇄ Photos
Zebras ⇄ Horses
Summer ⇄ Winter
Monet → photo
zebra → horse
summer → winter
photo → Monet
horse → zebra
winter → summer
Photograph
Monet
Van Gogh
Cezanne
Ukiyo-e

# Training with Unpaired Samples

- Goal: translate distribution X to Y, find

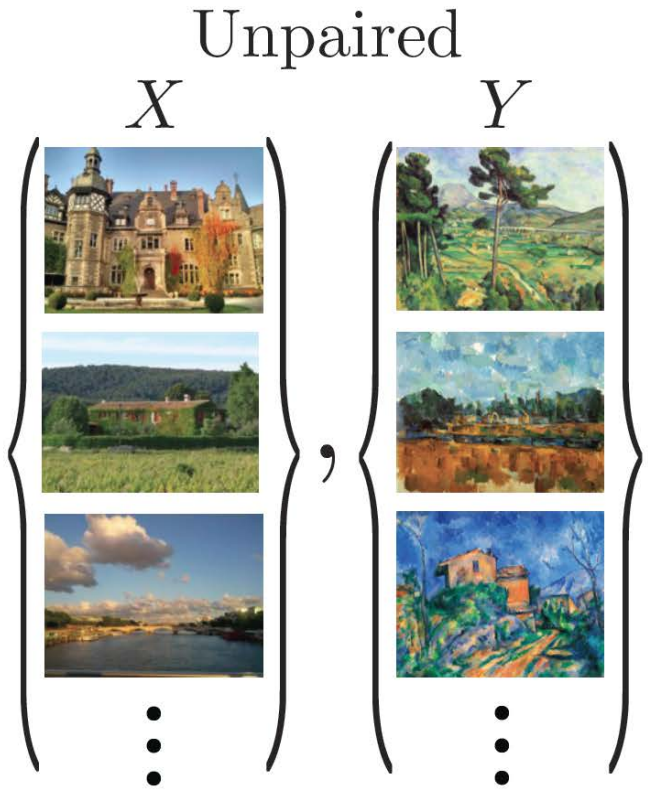
$$G: X \rightarrow Y, \quad \text{or } F: Y \rightarrow X$$

such that the images in G(X) are indistinguishable from images in Y or vice versa.

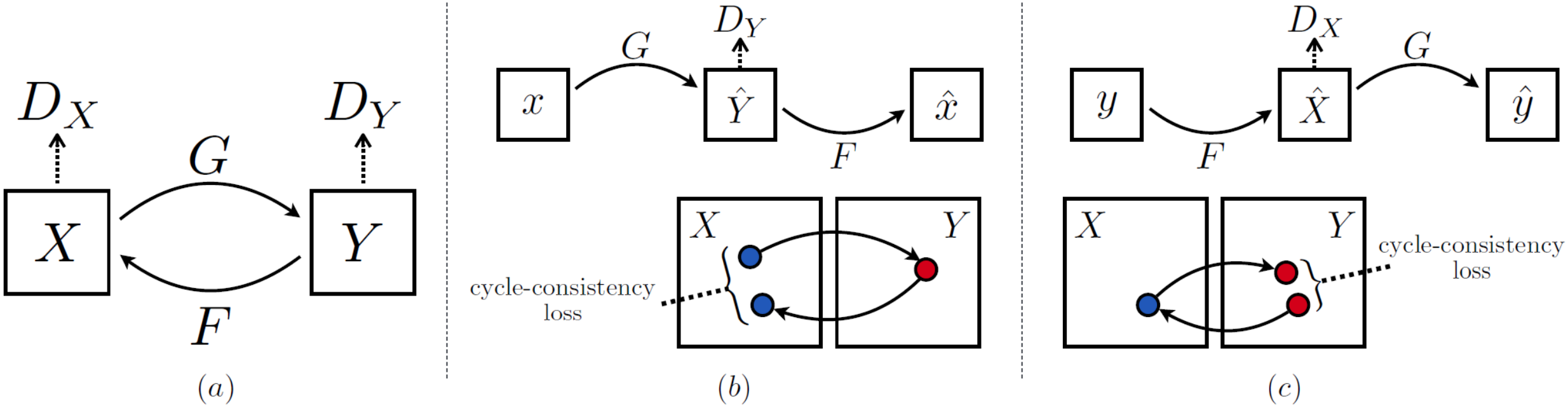
- Hard to train directly

- Instead train cyclic approach:

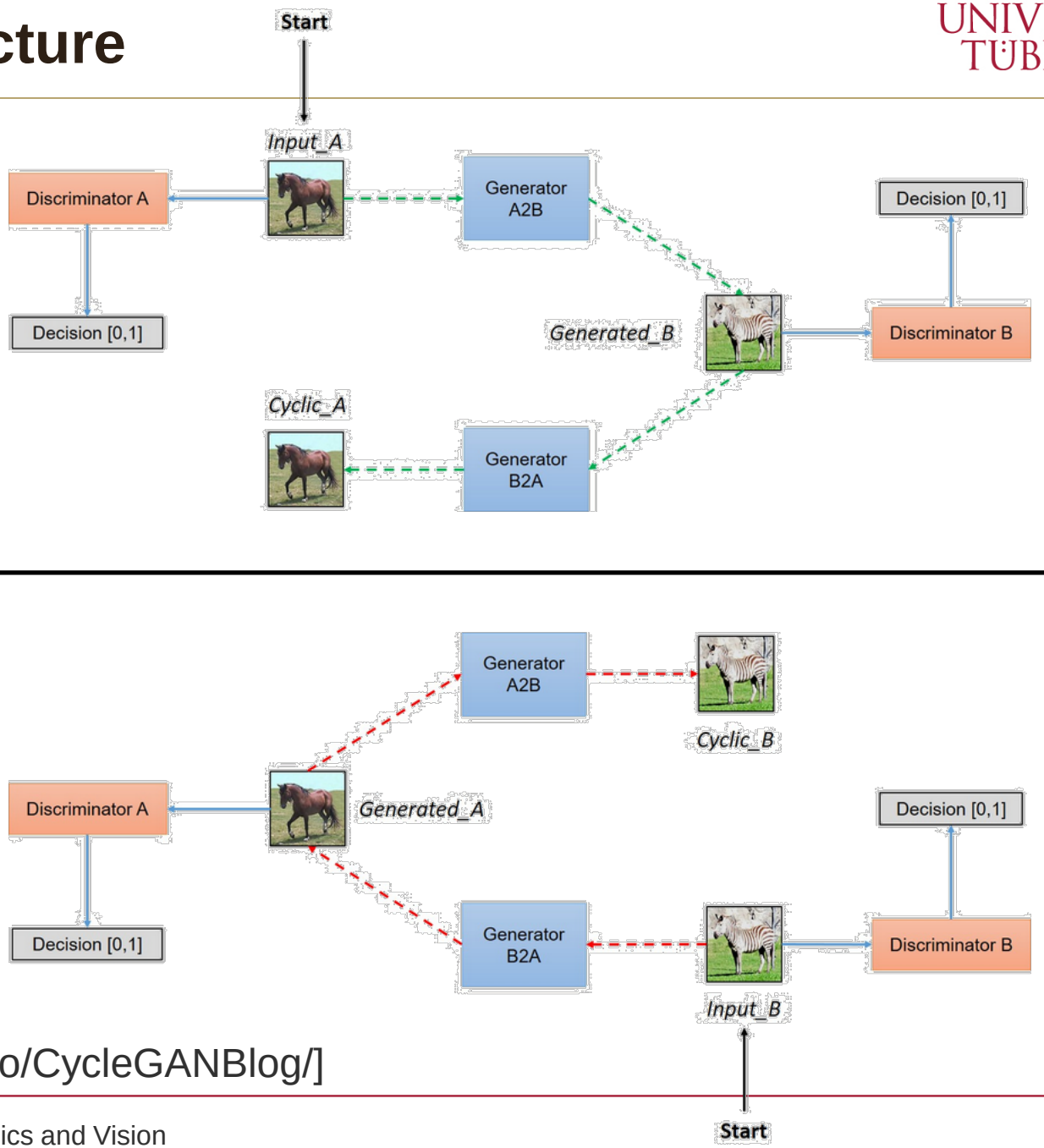
$$Y = G\big(F(Y)\big)$$
$$X = F(G(X))$$

# Cyclic Loss Function



# Simplified Architrecture




[https://hardikbansal.github.io/CycleGANBlog/]

# Results

- video



# Improved IR-Colorization using Adversarial Training and Current Deep Learning Design Patterns

Matthias Limmer, Hendrik P. A. Lensch

BMVC 2017

# Motivation

Goals:

- Improved night vision
- Active illumination without blinding traffic

Problems:

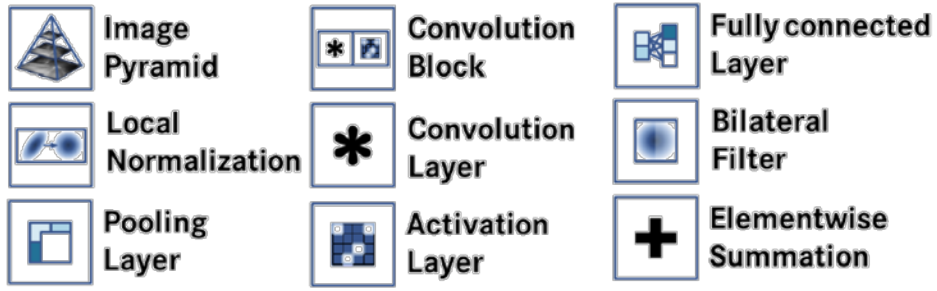
- Single channel → need to reconstruct Lab or RGB
- IR ≠ Luminance

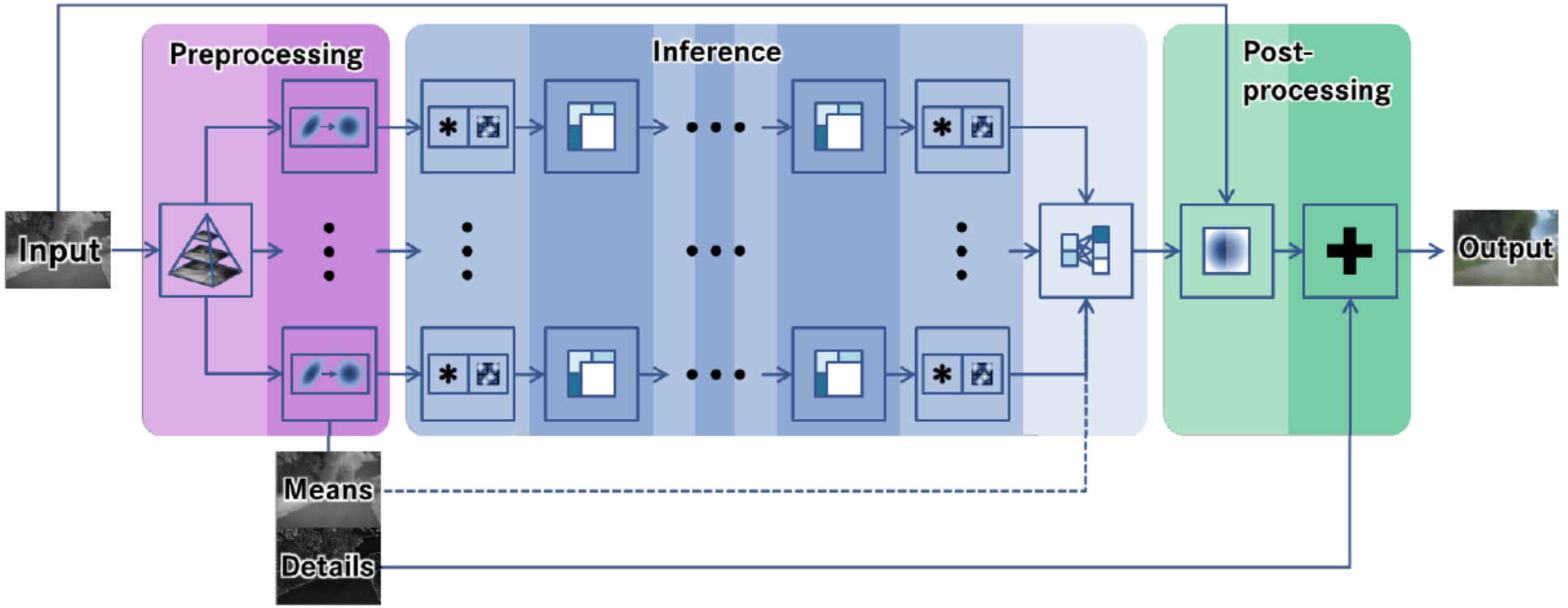
# Early Work

- Infrared Colorization Using Deep Convolutional Neural Networks, Limmer&Lensch – ICMLA 2016

# Early Work

- Multiscale
- Combined at the end



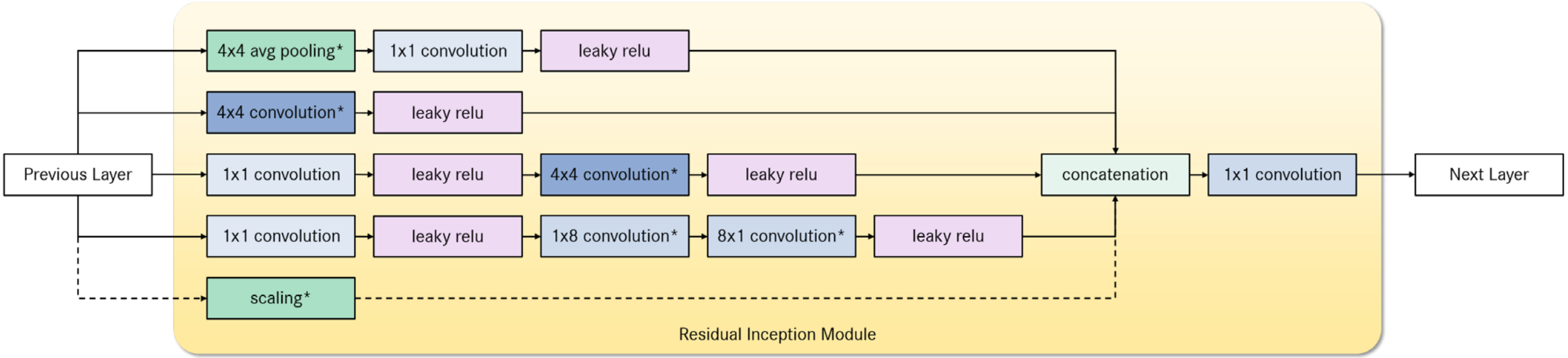
# Training Data

- 38.495 image pairs (IR + RGB – pixel-precise), 768x1024 pixels

# CNN-based Reconstruction - Improvement

- Adversarial Training, conditional GANs
- Inception Modules
- U-Net with multi-scale input

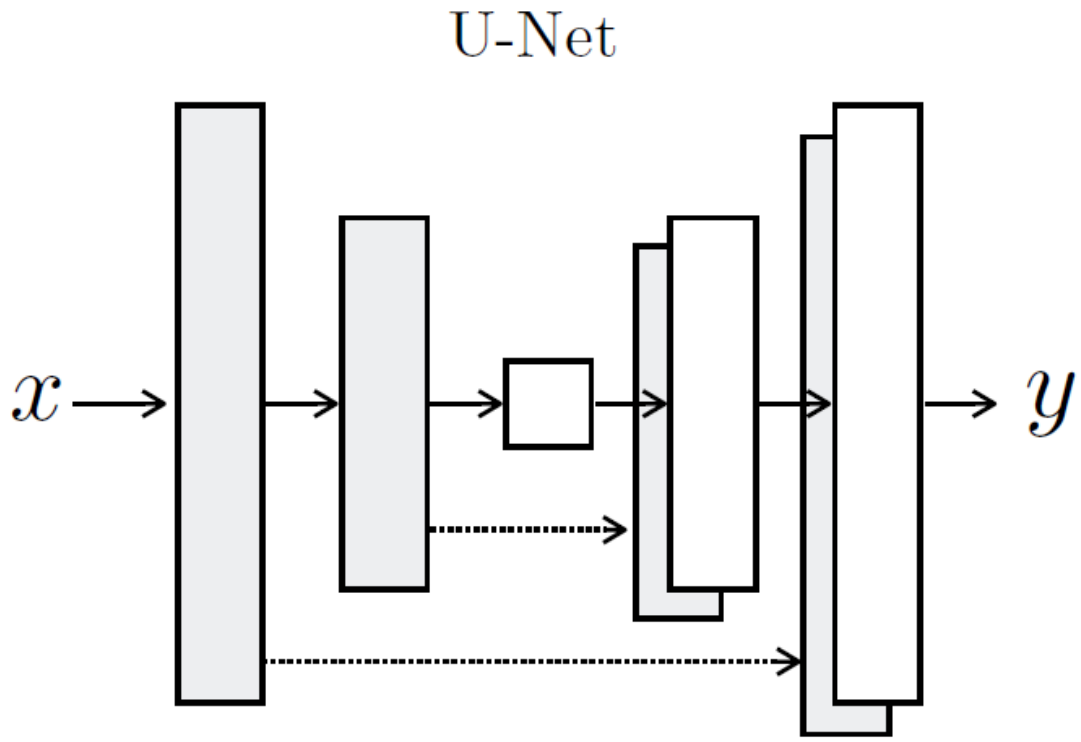
# Residual Inception Module




- Inception [Szegedy et al.] to
  - Increase variety of feature maps
  - increase depth
  - controllable compuational complexity
- Used for up (stride 2) and down sampling (stride ½)

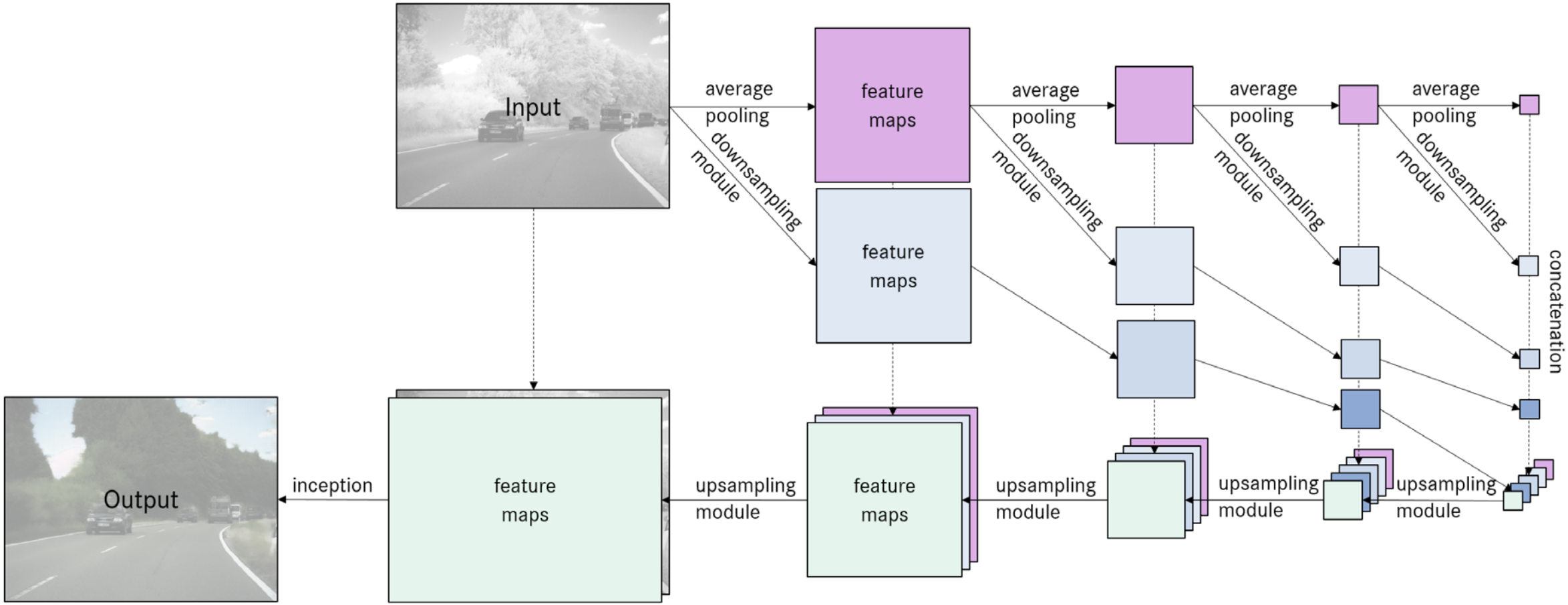
# U-Net

- [Ronneberger et al. 2015]
- Up and down sampling
- Skip connections
- Single input



# Estuary Network


- Input coming in at multiple scales and is combined with U-net structure



# Adverserial Training (GAN)

- Alternating training of a generator and a discriminator to obtain more realistic images
- Generator: produce images
- Discriminator: decide if image is real or synthesized
- Adversarial loss:

$$\mathcal{L}_{AD}(\mathrm{G},D)=\mathbb{E}[\log(\mathrm{D}(\mathrm{y}))]+\mathbb{E}[\log(1-\mathrm{D}(\mathrm{G}(\mathrm{x}))]$$

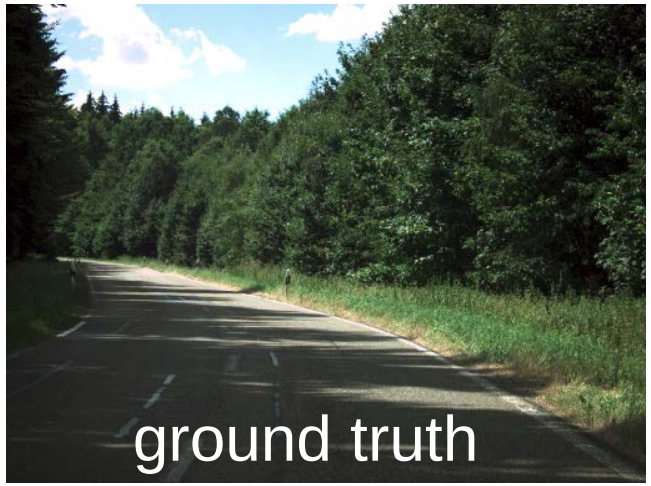
- Discriminator loss:

$$\mathcal{L}_{G}(\mathrm{G},D) = (1-\lambda)\mathcal{L}_{AG}(\mathrm{G},D)+\lambda\mathcal{L}_{L1}(G)$$
$$\mathcal{L}_{AG}(\mathrm{G},D) = \mathbb{E}[\log(D(G(x))]$$
$$\mathcal{L}_{L1}(\mathrm{G}) = \mathbb{E}[\|y-Gx\|_1]$$

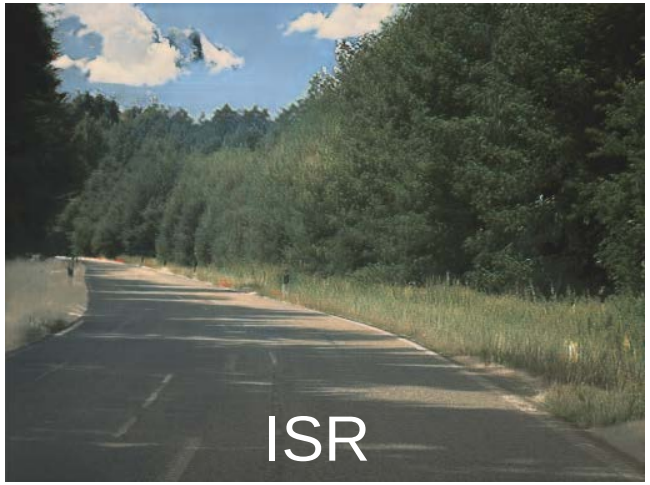
conditional GAN:
We do provide labeled input $(x, y)$

- Blending between $\mathcal{L}_{AD}(\mathrm{G},D)$ and $\mathcal{L}_{L1}(G)$ stabilizes GAN

# Skip / Residual Connections



# Comparison of Network Architectures

- I – inception
- S – skip connection

(a) Input (b) U-Net-64-S [18] (c) Estuary-32-S (d) Honari-64-5 [15] (e) Johnson-6 [19]

(f) Target (g) U-Net-64-IS (h) Estuary-32-IS (i) Honari-64-5-I (j) Lim-3-12-3-bp [26]

# Summary



| Architecture | $L_1$-error | $n_{\text{param}}$ [m] | timing [s] |
|---|---|---|---|
| U-Net-64-S | 0.108879 | 54.4 | 0.089 |
| U-Net-64-IS | 0.103525 | 25.1 | 0.171 |
| Estuary-32-S | 0.100113 | 68.6 | 0.241 |
| Estuary-32-IS | **0.083664** | 28.3 | 0.323 |
| Limmer-3-12-3-bp | 0.103510 | 1.4 | 0.263 |
| Johnson-6 | 0.126073 | 2.0 | 0.128 |
| Honari-64-5 | 0.179843 | 0.5 | **0.077** |
| Honari-64-5-I | 0.120775 | **0.2** | 0.204 |

- Inception modules increase accuracy, but reduce speed
- U-Net better trade-off between accuracy, speed and size
- Adversarial training increases realism, but can introduce harsher visual errors

# High-Res-GAN

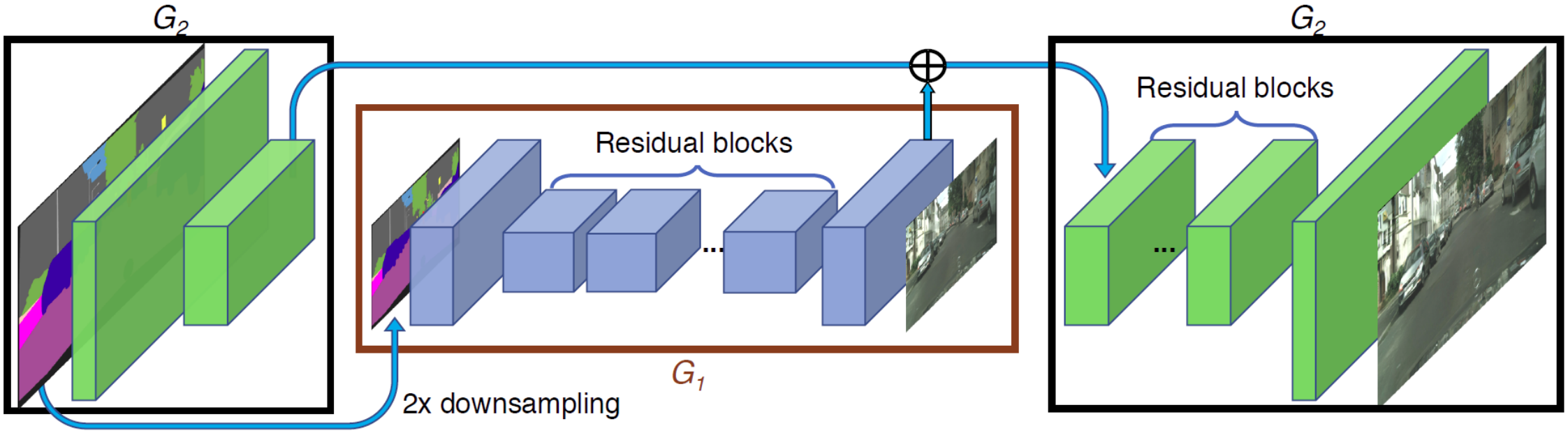
# Addressing High Resolution Image Generation

- Employ different discriminators and generators for different scales:
  - Discriminators $D_1, D_2, D_3$ each with the same network architecture

$$\min_G \max_{D_1,D_2,D_3} \sum_{k=1,2,3} \mathcal{L}_{GAN}(G, D_k)$$

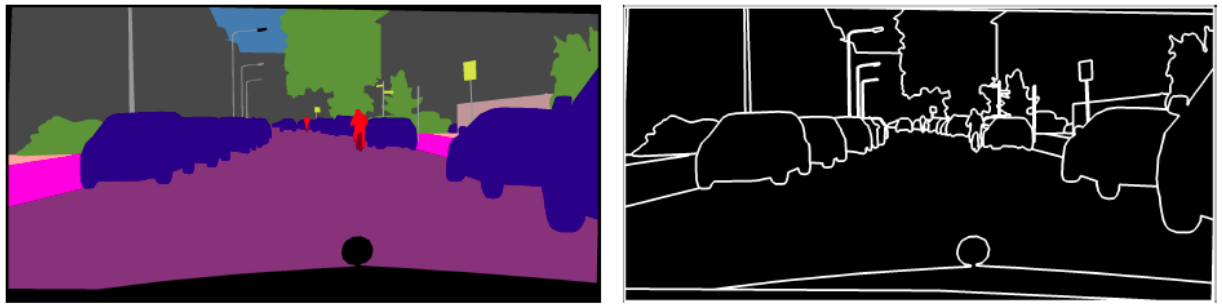
multi-task training

  - Global generator network $G_1$, local enhancer network $G_2$, …

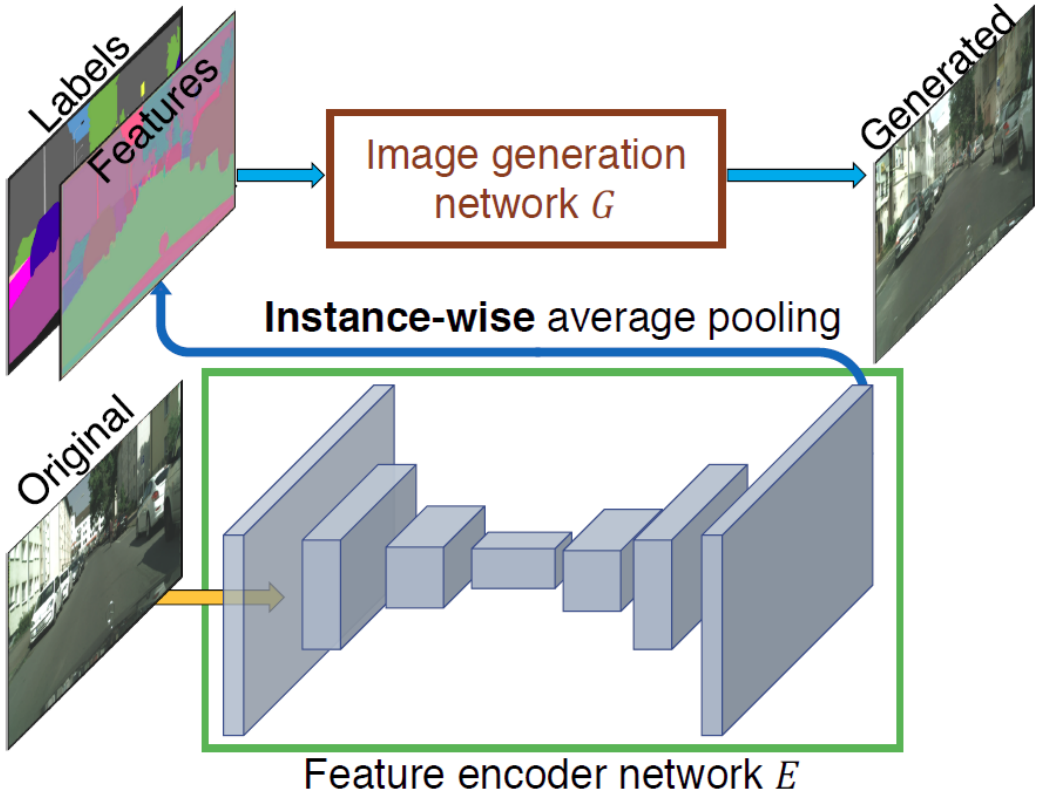


# Conditional GAN

- Use labels and instance-wise features for generating images.
- Allows for controlling the appearance of individual object instances



(a) Semantic labels (b) Boundary map



# Results



- video

# Conclusion - GANs

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- 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)
- Difficult to optain perfect visual quality at high-res

Active areas of research:
- Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)
- Conditional GANs, GANs for all kinds of applications

# References


- Fei-Fei Li & Justin Johnson & Serena Yeung, Generative Models - CS231n, Stanford Slides
- Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio (2014) Generative Adversarial Networks
- Metz, Poole, Pfau, Sohl-Dickstein, (2017) Unrolled Generative Adversarial Networks
- Ghosh et al. Multi-Agent Diverse Generative Adversarial Networks (2017)
- Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, Pieter Abbeel (2016), InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets, NIPS 2016.
- Jun-Yan Zhu, Taesung Park,  Phillip Isola, Alexei A. Efros (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017.
- Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, Honglak Lee (2017). Generative Adversarial Text to Image Synthesis
- Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, Bryan Catanzaro (2018), High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs, CVPR 2018.