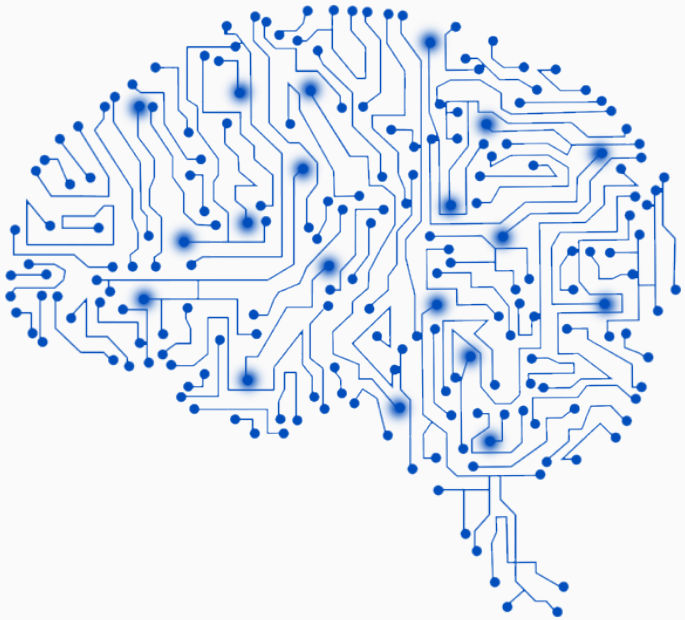
Collège doctoral Centre-Val de Loire

Introduction to Deep Learning

Course IV – Introduction to Artificial Neural Networks: Generative Models

Bruno Galerne

2024-2025



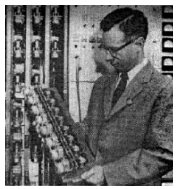
Credits

Most of the slides from **Charles Deledalle's** course "UCSD ECE285 Machine learning for image processing" (30 $\times$ 50 minutes course)

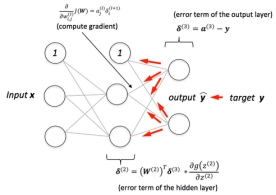
`www.charles-deledalle.fr/`

`https://www.charles-deledalle.fr/pages/teaching.php#learning`

Timeline of (deep) learning



1958 Perceptron



1974 Backpropagation

Convolution Neural Networks for Handwritten Recognition

1998

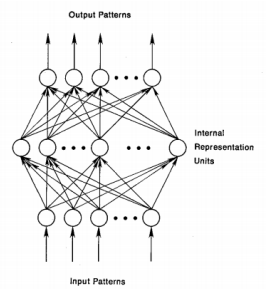
Google Brain Project on 16k Cores

2012

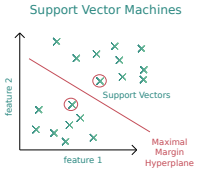
awkward silence (AI winter)

1969 Perceptrons book

~1980 Multilayer network



1995 SVM reigns

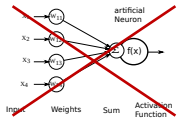


2006 Restricted Boltzmann Machine

2012 AlexNet wins ImageNet

IMAGENET

Perceptron criticized

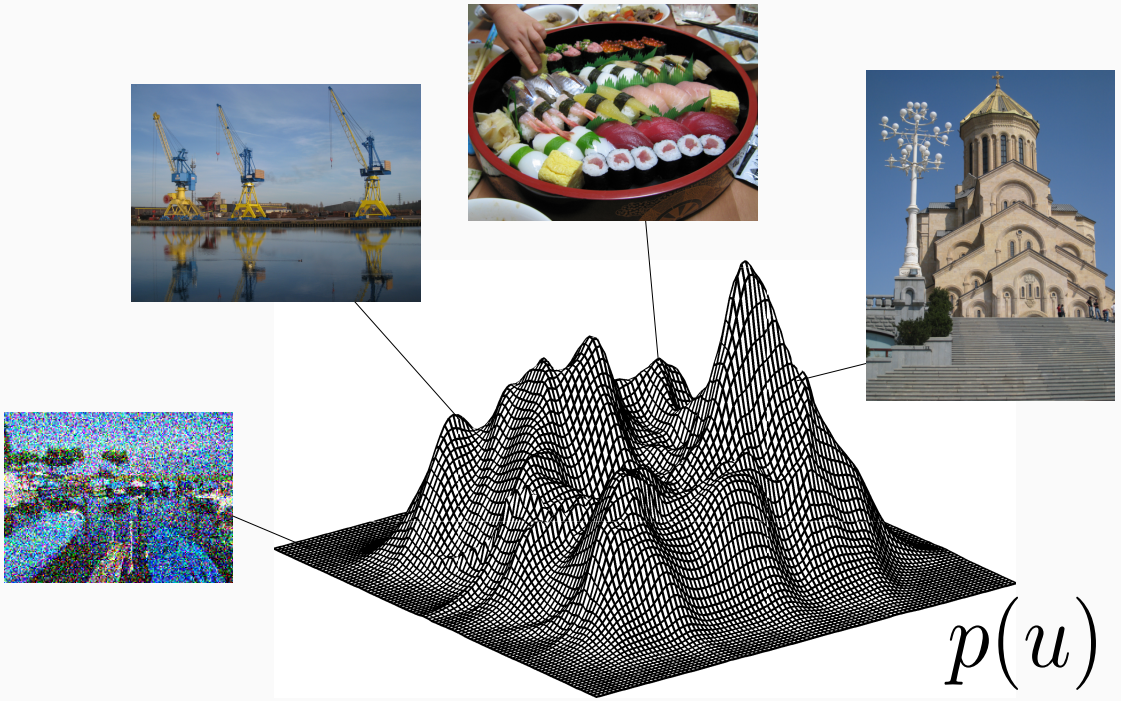


(Source: Lucas Masuch & Vincent Lepetit)

Introduction on generative models

Generative models

1. Model and/or learn a distribution $p(u)$ on the space of images.



(source: Charles Deledalle)

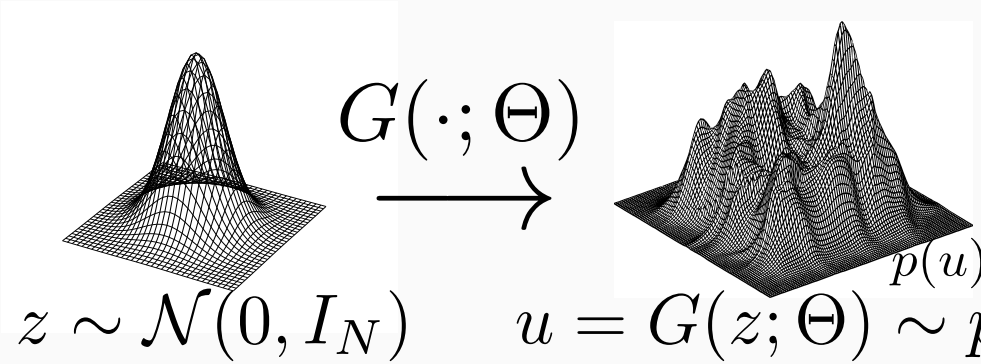
The images may represent:

- different instances of the same texture image,
- all images naturally described by a dataset of images,
- any image

2. Generate samples from this distribution.

Generative models

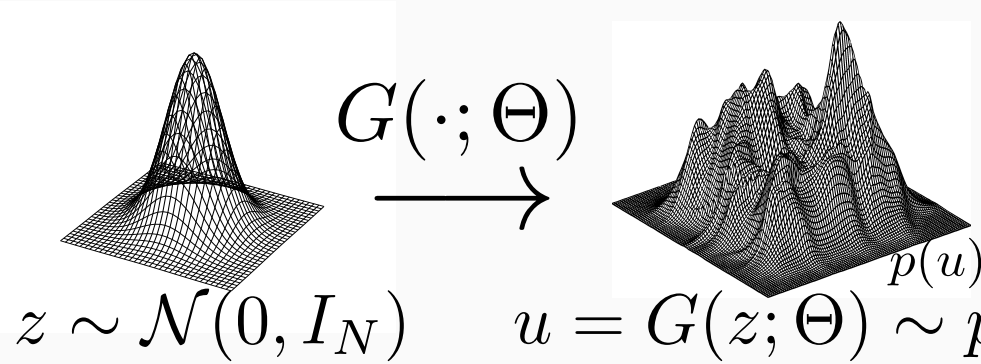
1. Model and/or learn a distribution $p(u)$ on the space of images.
2. Generate samples from this distribution.



- z is a generic source of randomness, often called the latent variable.
- If $G(\cdot;\Theta)$ is known, then $p = G(\cdot;\Theta)_{\#}\mathcal{N}(0, I_n)$ is the push-forward of the latent distribution.

Generative models

1. Model and/or learn a distribution $p(u)$ on the space of images.
2. Generate samples from this distribution.



- z is a generic source of randomness, often called the latent variable.
- If $G(\cdot;\Theta)$ is known, then $p = G(\cdot;\Theta)_{\#}\mathcal{N}(0, I_n)$ is the push-forward of the latent distribution.

The generator $G(\cdot;\Theta)$ can be:

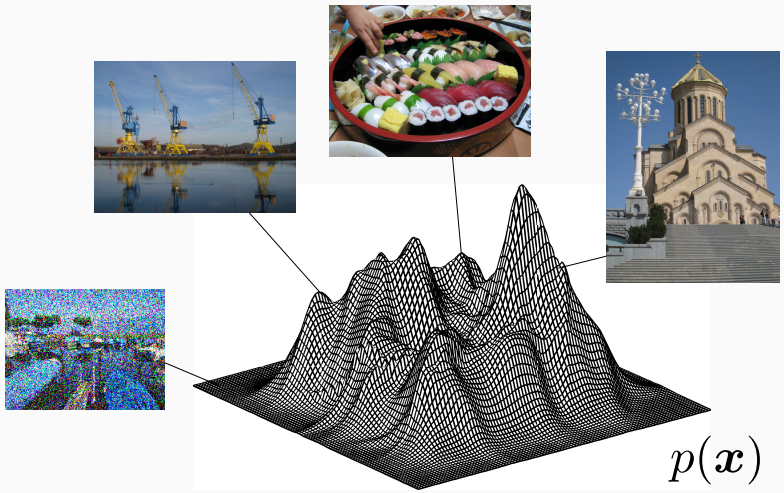
- A deterministic function (e.g. convolution operator),
- A neural network with learned parameter,
- An iterative optimization algorithm (gradient descent,...),
- A stochastic sampling algorithm (e.g. MCMC, Langevin diffusion,...).

Image generation: Gaussian model

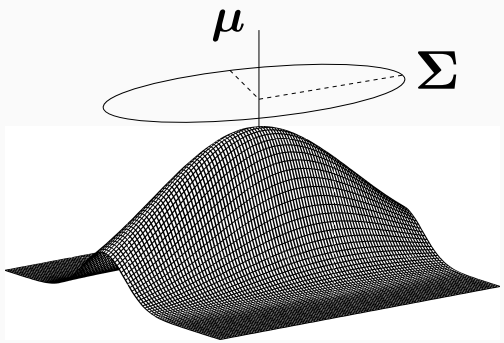
- Consider a **Gaussian model** for the distribution of images $\boldsymbol{x}$ with d pixels:

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left[-(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right]$$

  - $\boldsymbol{\mu}$: mean image,
  - $\boldsymbol{\Sigma}$: covariance matrix of images.



$\approx$



Gaussian prior $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

(source: Charles Deledalle)

Image generation: Gaussian model

- Take a training dataset $\mathcal{D}$ of images:

$$\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$$

$$= \left\{ \text{  }, \text{  }, \text{  }, \text{  }, \text{  }, \text{  }, \ldots \right\}_{\times N}$$

- Estimate the mean

$$\hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_i \boldsymbol{x}_i = \text{  }$$

- Estimate the covariance matrix: $\hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_i (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})^T = \hat{\boldsymbol{E}} \hat{\boldsymbol{\Lambda}} \hat{\boldsymbol{E}}^T$

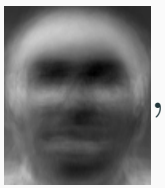
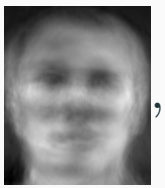
$$\hat{\boldsymbol{E}} = \underbrace{\left\{ \text{  }, \text{  }, \text{  }, \text{  }, \text{  }, \text{  }, \ldots \right\}_{\times N}}_{\text{eigenvectors of } \hat{\boldsymbol{\Sigma}}, \textit{i.e.}, \text{ main variation axis}}$$

Image generation: Gaussian model

You now have learned a **generative model**:

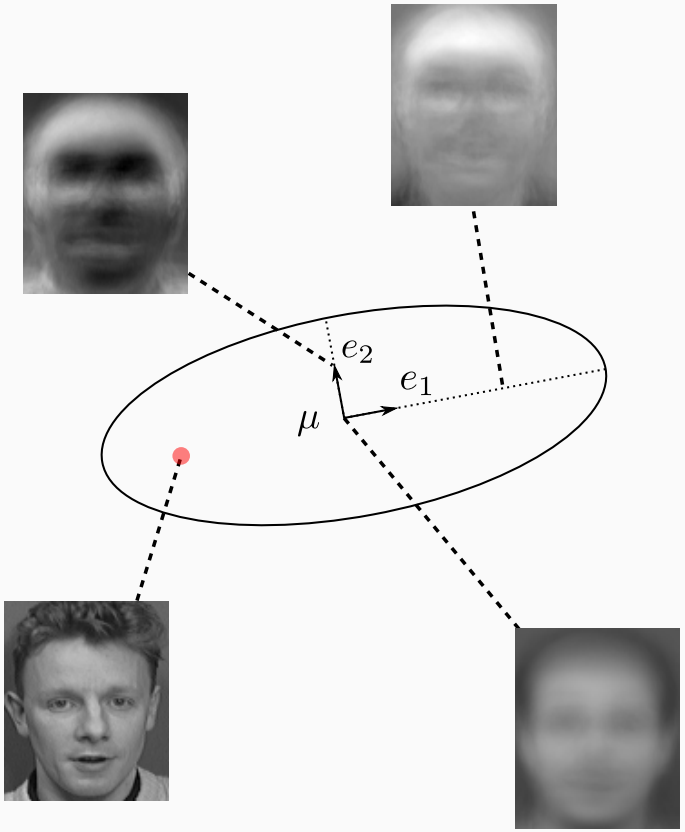


Image generation: Gaussian model

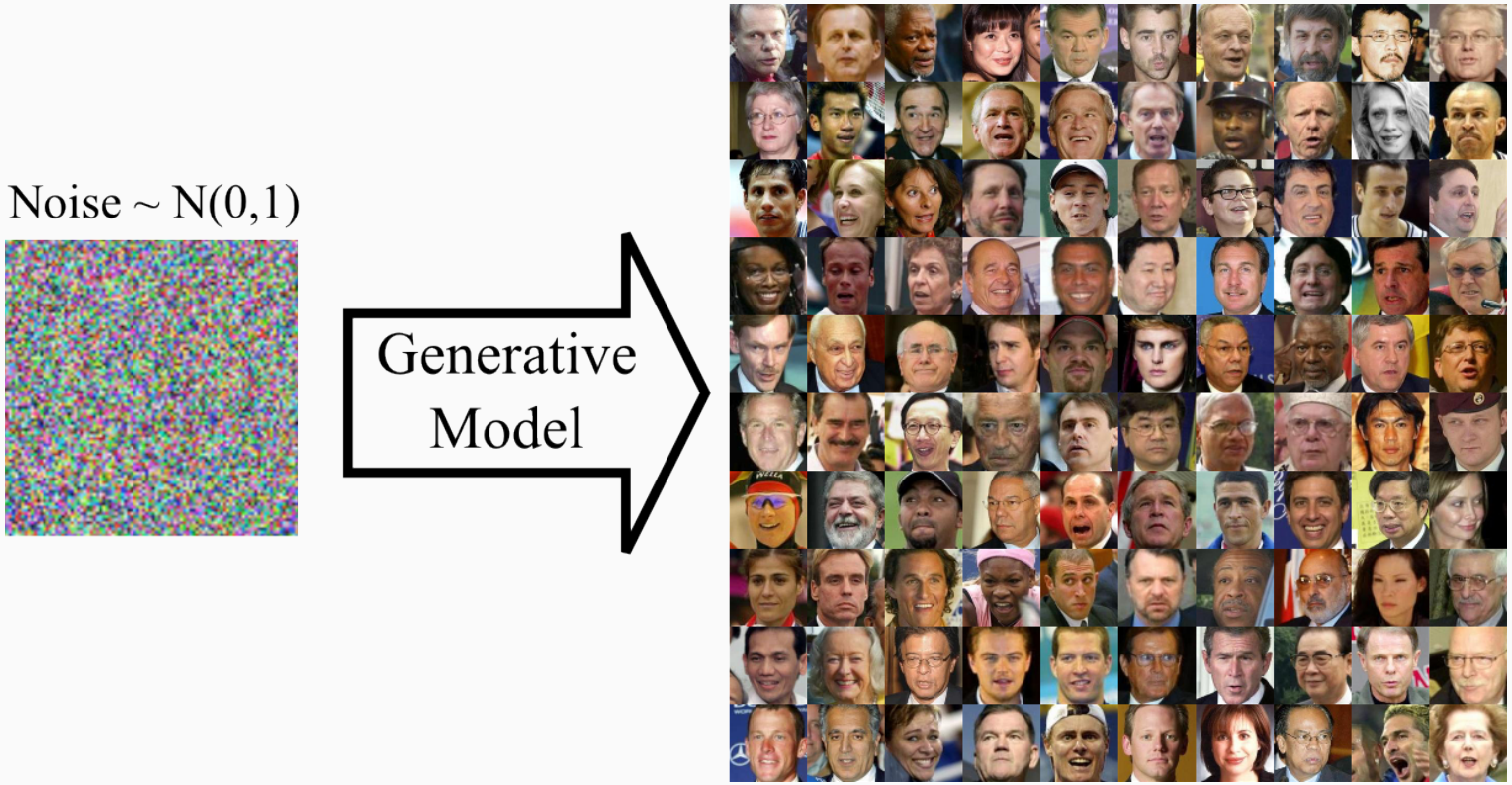
How to generate samples from $\mathcal{N}(\hat{\mu}, \hat{\Sigma})$?

$$\begin{cases} \boldsymbol{z} & \sim \mathcal{N}(0, \boldsymbol{I}_d) \quad \leftarrow \text{Generate random latent variable} \\ \boldsymbol{x} & = \hat{\boldsymbol{\mu}} + \hat{\boldsymbol{E}}\hat{\boldsymbol{\Lambda}}^{1/2}\boldsymbol{z} \end{cases}$$

The model does not generate realistic faces.

- The Gaussian distribution assumption is too simplistic.
- Each generated image is just a linear random combination of the eigenvectors (with independence !).
- The generator corresponds to a one layer liner neural network (without non-linearities).

Image generation: Gaussian model



- Deep generative modeling consists in learning non-linear generative models to reproduce complex data such as realistic images.
- It relies on deep neural networks and several solutions have been proposed since the "Deep learning revolution" (2012).

Generative models: Examples

Generative Adversarial Networks: [Goodfellow et al., 2014]

- Data: A database of images.
- Inferred distribution: Not explicit, push-forward measure given by generator.
- z is a Gaussian array in a latent space.
- $G(\cdot;\Theta)$ is a (convolutional) neural network with parameters Θ learned using an adversarial discriminator network $D(\cdot;\Theta_D)$.

Data

MNIST: handwritten digits

Generated images

Fake images (100 epochs)

Image size:
28×28 px

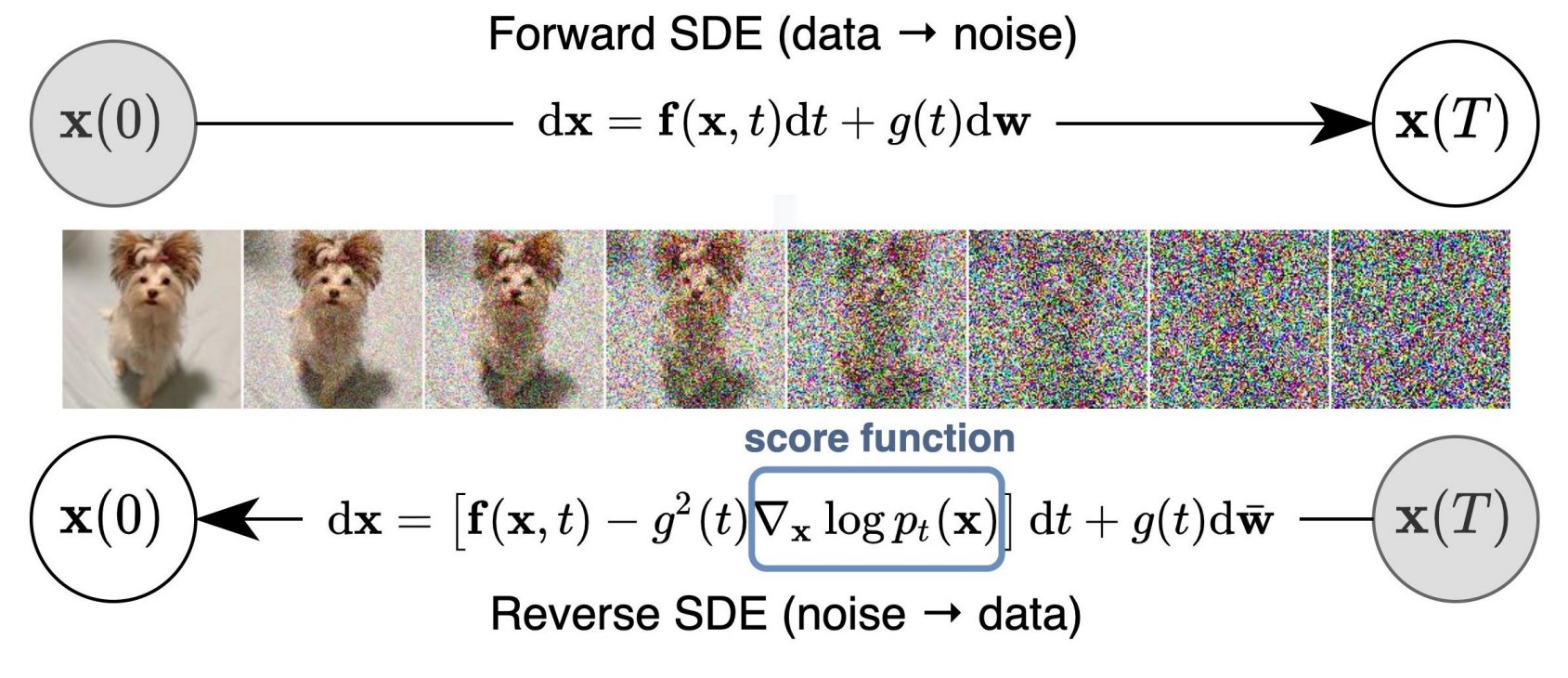
Generative models: Examples

Generative Adversarial Networks: Style GAN [Karras et al., 2019]

Image size:
1024× 1024 px
(source: Karras et al.)

Denoising diffusion probabilistic models

- Learn to revert a degradation process: Add more and more noise to an image.
- First similar model [Sohl-Dickstein et al., 2015]



(source: Yang Song)

- Probably the most promising framework these days... but things change very quickly in this field!

Diffusion models

[Ho et al., 2020]: Denoising Diffusion Probabilistic Models (DDPM): One of the first paper producing images with reasonable resolution.

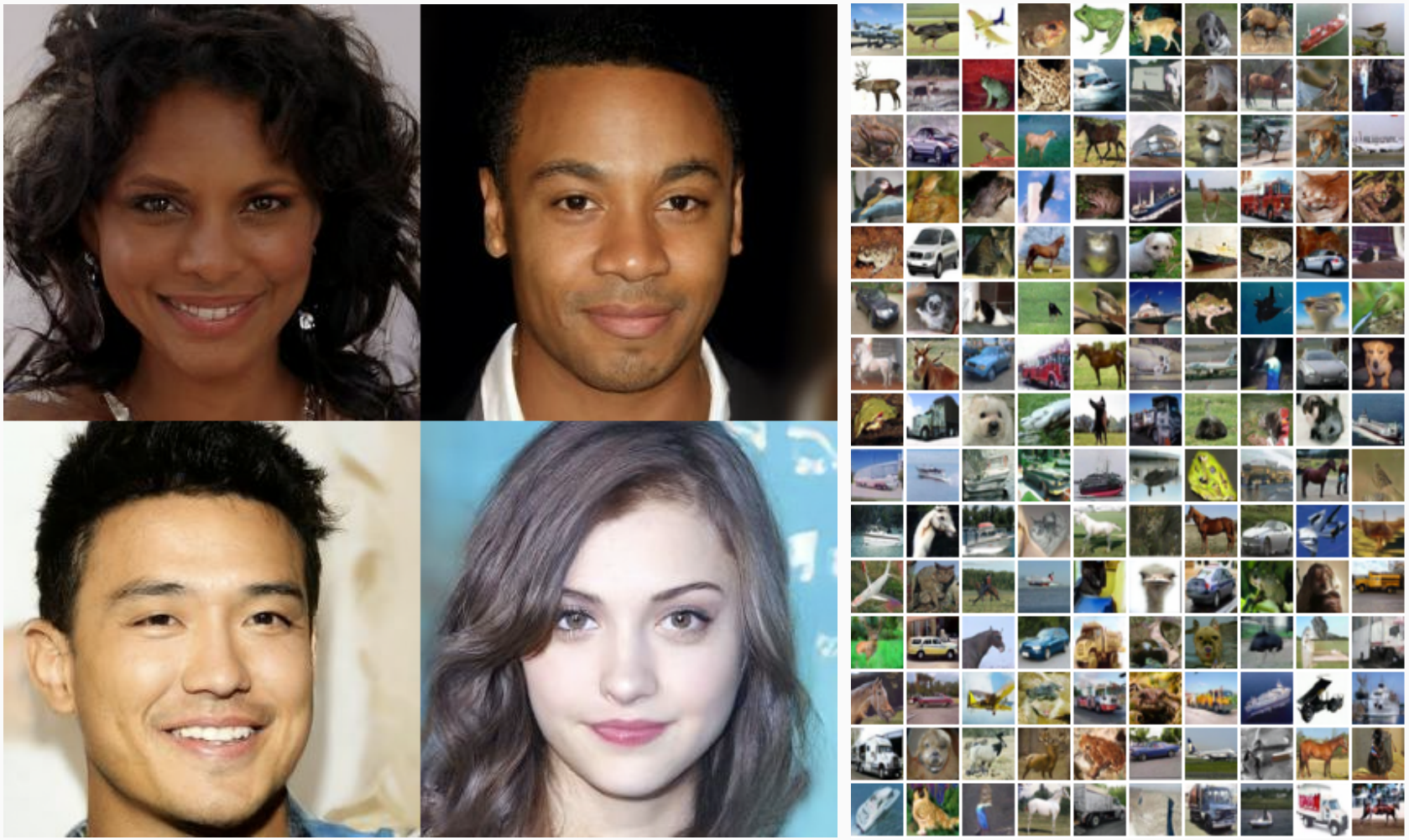


Figure 1: Generated samples on CelebA-HQ 256×256 (left) and unconditional CIFAR10 (right)

Generative models: Motivations

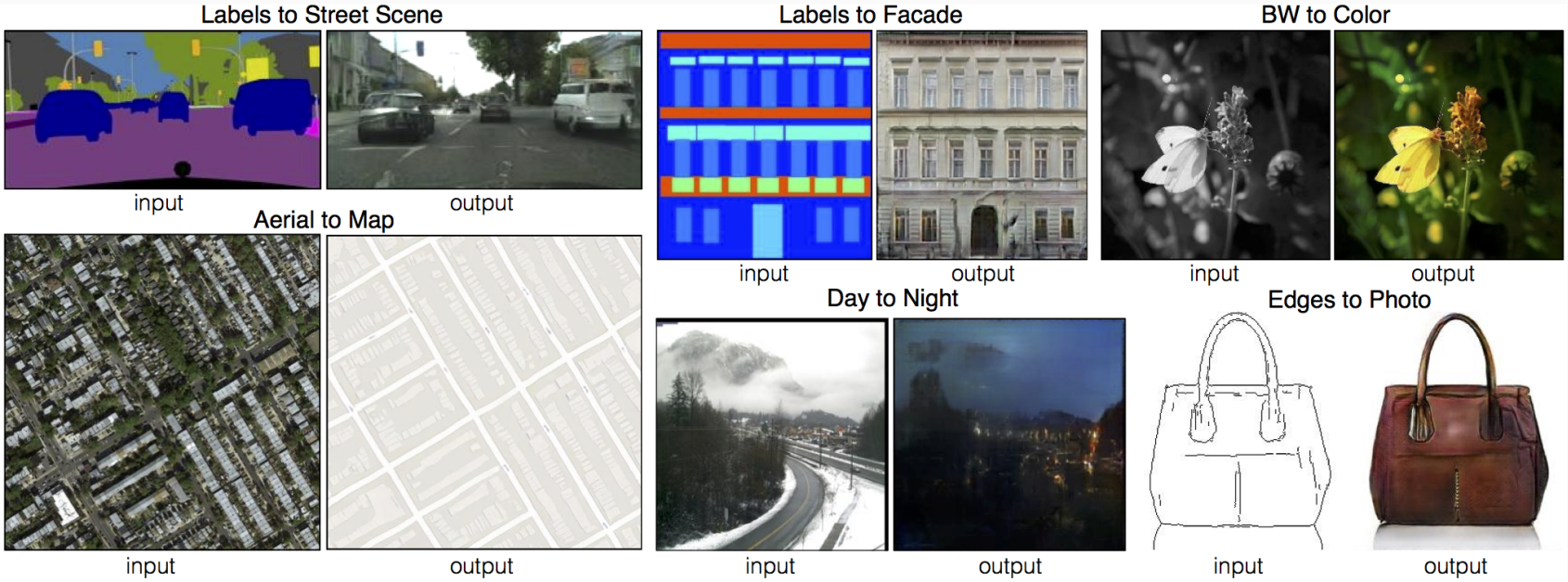
Why generative models are interesting ?

- **Generating realistic images is important by itself** for entertainment industry (visual effects, video games, augmented reality...), design, advertising industry,...
- **Good image model leads to good image processing:** Generative models can be used as a parametric space for solving inverse problems. Example: Inpainting of a portrait image.
- Also generative models opens the way to **non trivial image manipulation** using **conditional generative models**.

Conditional generative models: Examples

Pix2pix: Image-to-Image Translation with Conditional Adversarial Nets

[Isola et al., 2017]



- GAN conditioned on input image.
- Generator: U-net architecture
- Discriminator: Patch discriminator applied to each patch
- Opens the way for new creative tools

(source: Isola et al.)

Conditional generative models: Examples

Latest trends using **diffusion models**: Text to image generation

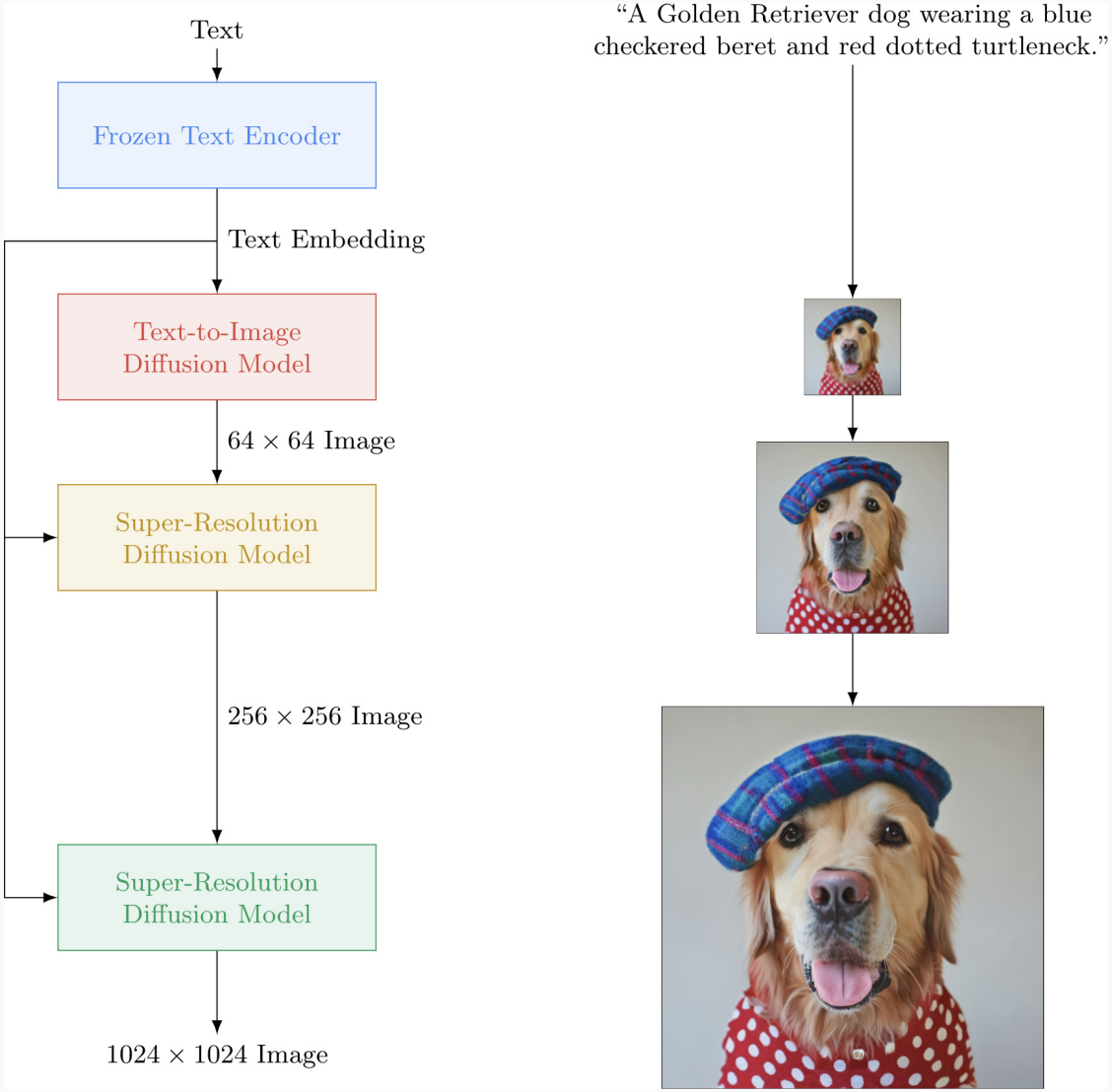
- DALL·E 1 & 2: CreatingImages from Text (Open AI, January 2021 and April 2022)
- Imagen, Google research (May 2022)

DALL·E 2 (Open AI)
Input: An astronaut riding a horse in a photorealistic style

Imagen (Google)
Input: A dog looking curiously in the mirror, seeing a cat.

Conditional generative models: Examples

Imagen pipeline:



(source: [Saharia et al., 2022])

Conditional generative models: Examples

In August 2022, StableDiffusion was released:

- Based on the paper [Rombach et al., 2022]
- **Open source!**

futuristic tree house, hyper realistic, epic composition, cinematic, landscape vista photography by Carr Clifton & Galen Rowell, Landscape veduta photo by Dustin Lefevre & tdraw, detailed landscape painting by Ivan Shishkin, rendered in Enscape, Miyazaki, Nausicaa Ghibli, 4k detailed post processing, unreal engine
Steps: 50, Sampler: PLMS, CFG scale: 9, Seed: 2937258437

(source: Stable diffusion)

Diffusion models in 2023

Diffusion models are considered mature models and have been used in a large variety of frameworks.

- Diffusion models **beyond image generation**: Text to video, motion generation, proteins, soft robots,...
- **Control of (latent) diffusion models**([Ruiz et al., 2023], [Zhang et al., 2023],...)
- **Diffusion models as priors for imaging inverse problems** ([Chung et al., 2023], [Song et al., 2023], lot of applications in medical imaging, etc.)

Diffusion models in 2023

DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation

Nataniel Ruiz[*,1,2] Yuanzhen Li[1] Varun Jampani[1]
Yael Pritch[1] Michael Rubinstein[1] Kfir Aberman[1]
[1] Google Research [2] Boston University

Figure 1. With just a few images (typically 3-5) of a subject (left), *DreamBooth*—our AI-powered photo booth—can generate a myriad of images of the subject in different contexts (right), using the guidance of a text prompt. The results exhibit natural interactions with the environment, as well as novel articulations and variation in lighting conditions, all while maintaining high fidelity to the key visual features of the subject.

(source: [Ruiz et al., 2023])

Diffusion models in 2023

Adding Conditional Control to Text-to-Image Diffusion Models

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala
Stanford University
{lvmin, anyirao, maneesh}@cs.stanford.edu

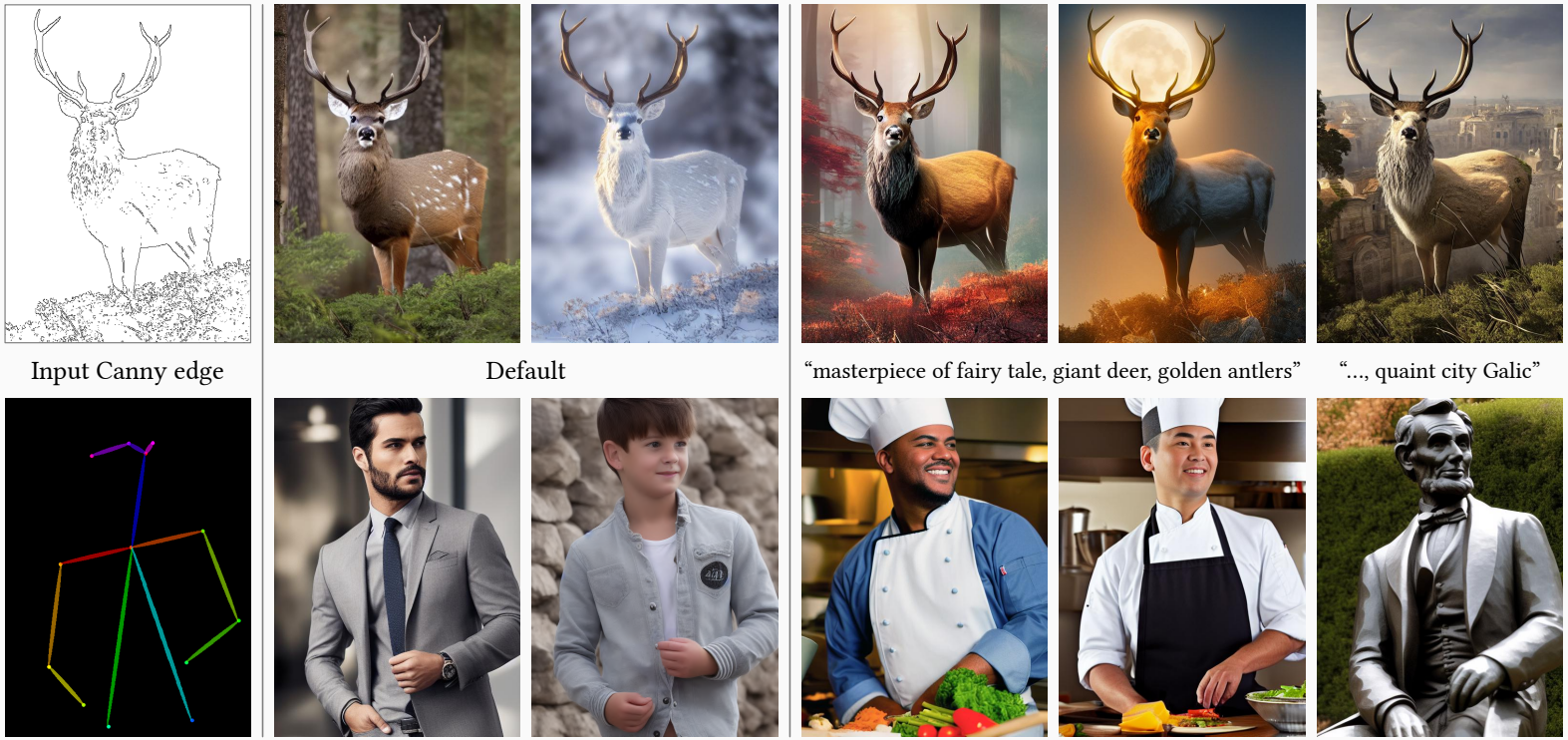


Figure 1: Controlling Stable Diffusion with learned conditions. ControlNet allows users to add conditions like Canny edges (top), human pose (bottom), *etc*., to control the image generation of large pretrained diffusion models. The default results use the prompt "a high-quality, detailed, and professional image". Users can optionally give prompts like the "chef in kitchen".

(source: ControlNet [Zhang et al., 2023])

Diffusion models in 2023

Diffusion posterior sampling for general noisy inverse problems
[Chung et al., 2023]

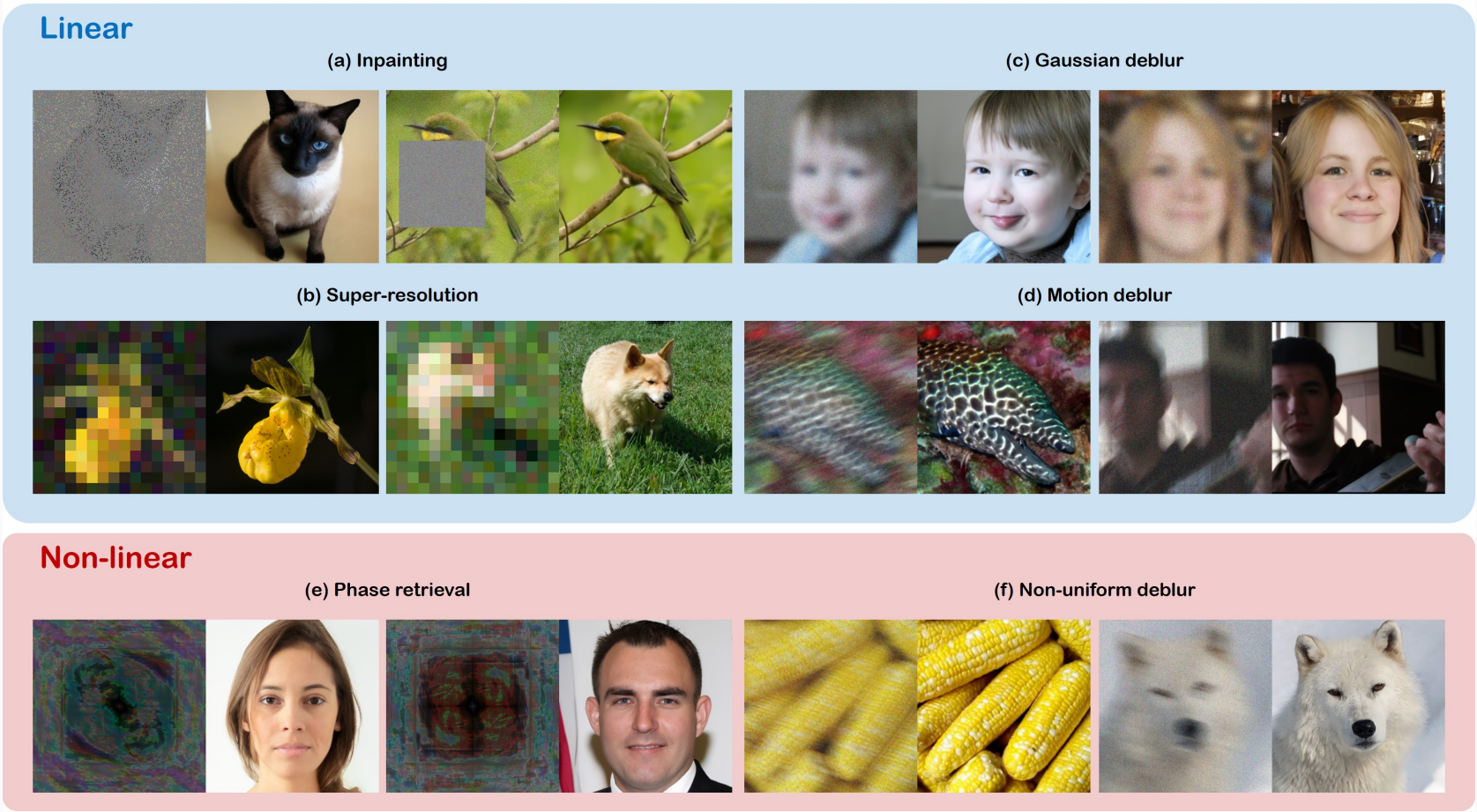


Figure 1: Solving noisy linear, and nonlinear inverse problems with diffusion models. Our reconstruction results (right) from the measurements (left) are shown.

(source: [Chung et al., 2023])

Diffusion models in 2024

- Text-to-video generation.
- Image generation broadly available via conversational agent (ChatGPT, Le Chat by Mistral (based on Flux Pro),...).
- Diffusion models used as "world model".

DiffusionLight: Light Probes for Free by Painting a Chrome Ball

Pakkapon Phongthawee[*1] Worameth Chinchuthakun[*1,2] Nontaphat Sinsunthithet[1]
Amit Raj[3] Varun Jampani[4] Pramook Khungurn[5] Supasorn Suwajanakorn[1]
[1] VISTEC [2] Tokyo Tech [3] Google Research [4] Stability AI [5] Pixiv

https://diffusionlight.github.io/

Figure 1. We leverage a pre-trained diffusion model (Stable Diffusion XL) for light estimation by rendering an HDR chrome ball. In each scene, we show our normally exposed chrome ball on top and our underexposed version, which reveals bright light sources, on the bottom.

(source: [?])

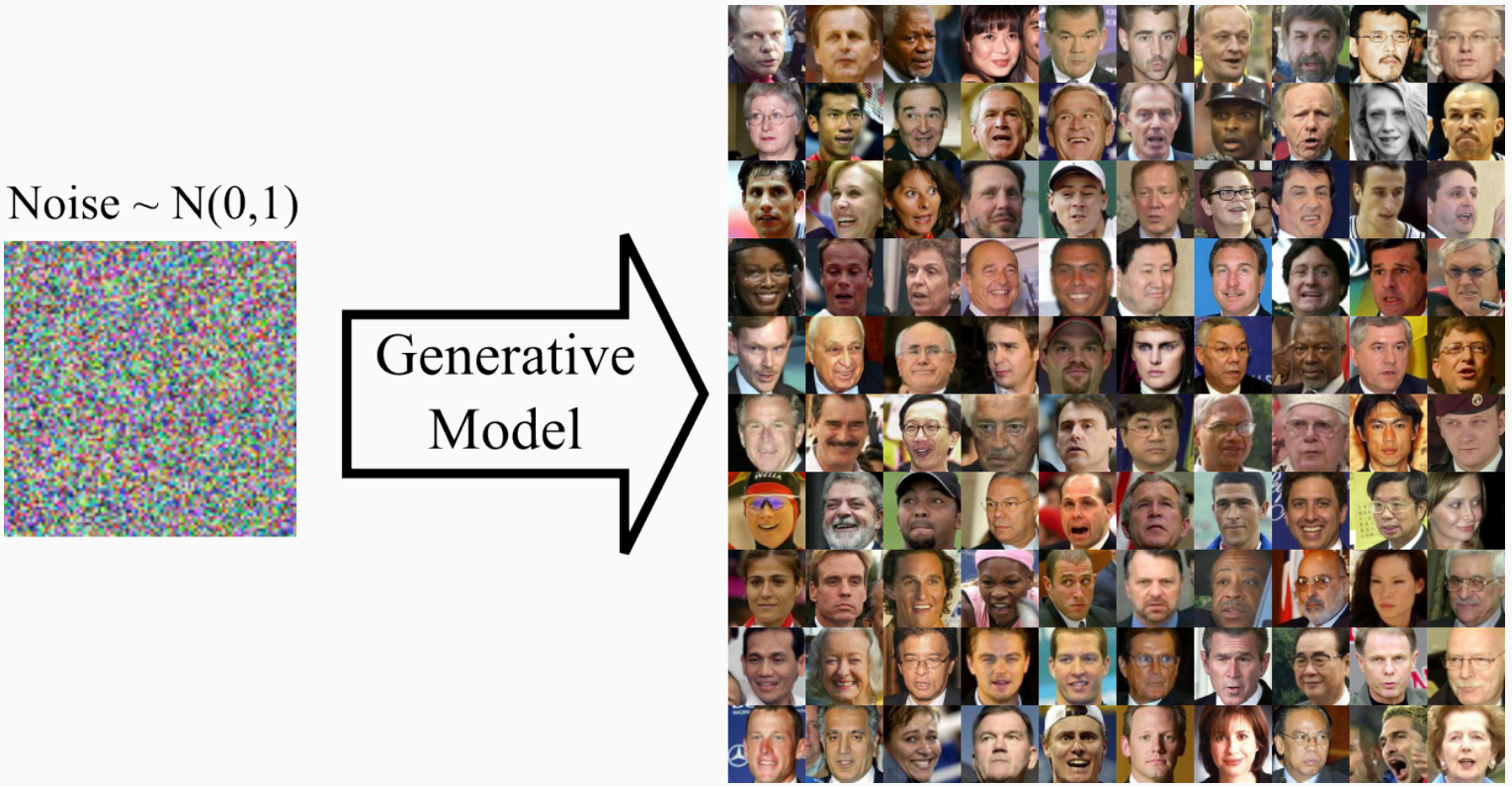
Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GAN)

Main references:

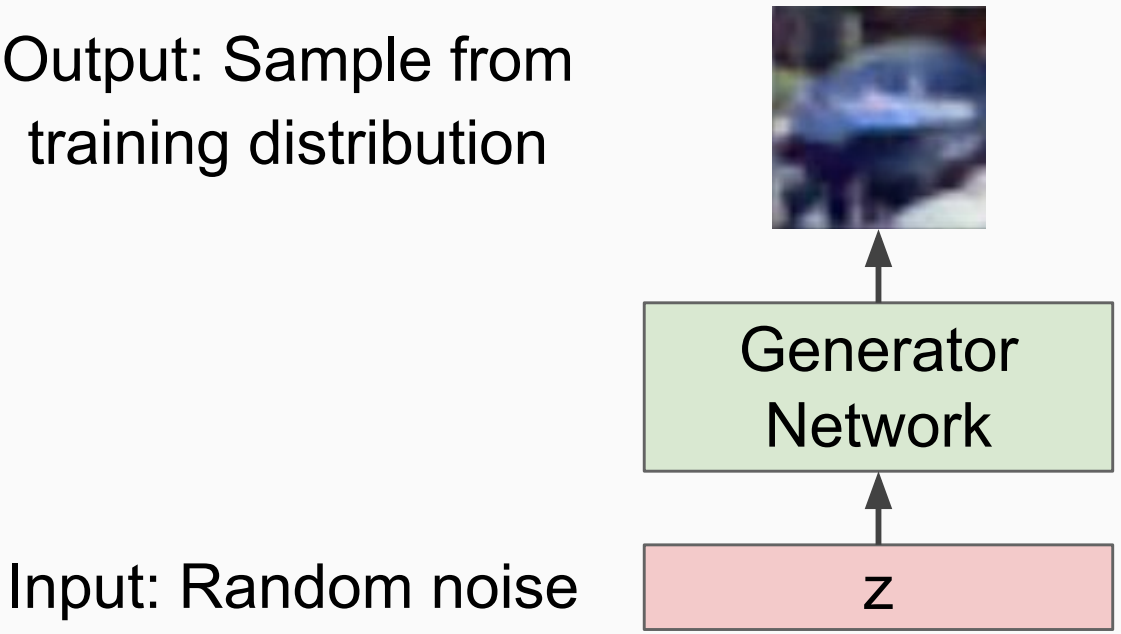
1. Original paper: [Goodfellow et al., 2014]
2. NIPS 2016 tutorial: [Goodfellow, 2017]

Image generation – Beyond Gaussian models



Generative Adversarial Networks (GAN)

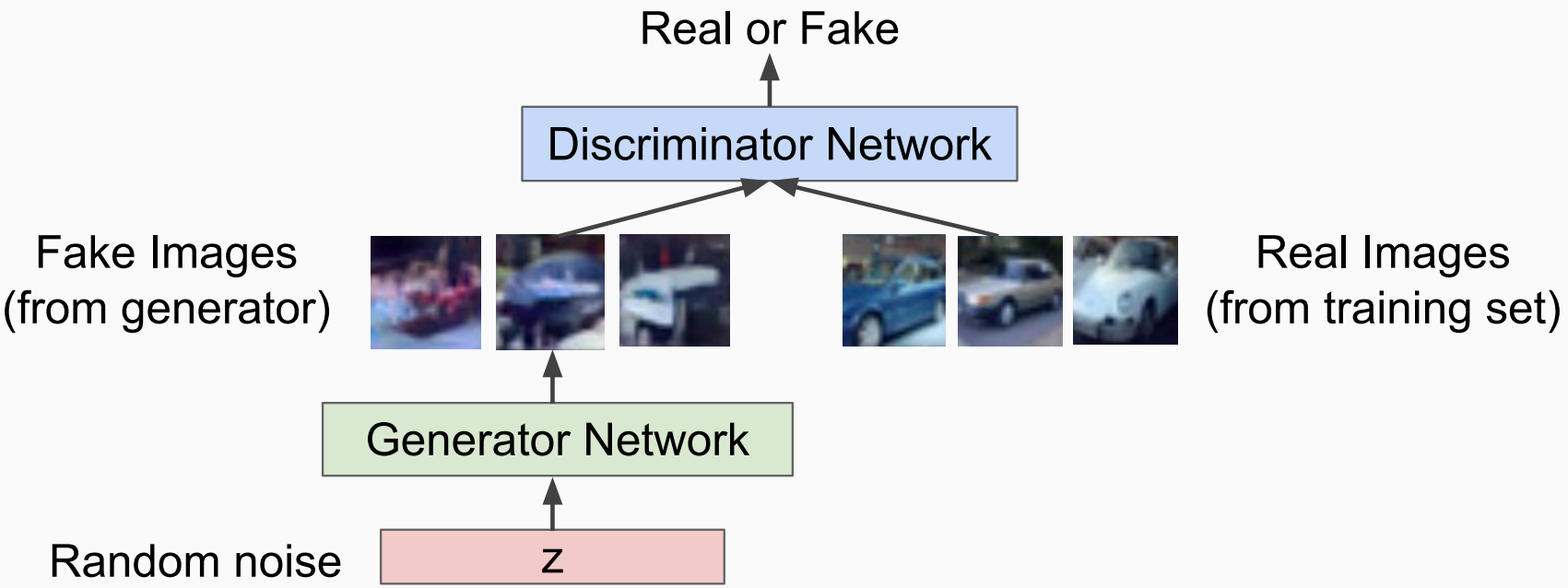
- **Goal:** design a complex model with high capacity able to map latent random noise vectors $\boldsymbol{z} \in \mathbb{R}^k$ to a realistic image $\boldsymbol{x} \in \mathbb{R}^d$.
- **Idea:** Take a deep neural network



- **What about the loss?** Measure if the generated image is **photo-realistic**.

Generative Adversarial Networks (GAN)

Define a loss measuring how much you can fool a classifier that has learned to distinguish between real and fake images.



- **Discriminator network:** Try to distinguish between real and fake images.
- **Generator network:** Fool the discriminator by generating realistic images.

Recap on binary classification

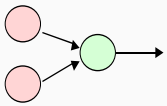
- Given a labeled dataset

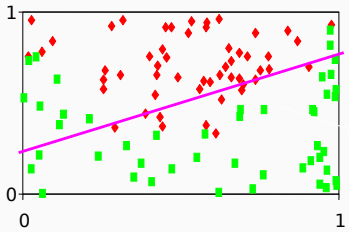
$$\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)}),\ i = 1, \dots, N\} \subset \mathbb{R}^d \times \{0, 1\}$$

  with **binary labels** $y^{(i)} \in \{0, 1\}$ that corresponds to two classes C_0 and C_1.
- A **parametric classifier** $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to [0, 1]$ outputs a probability such that

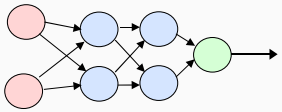
$$\boldsymbol{p} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbb{P}(\boldsymbol{x} \in C_1) \quad \text{and} \quad 1 - \boldsymbol{p} = 1 - f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbb{P}(\boldsymbol{x} \in C_0)$$

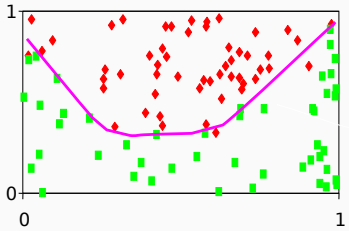
1 neuron



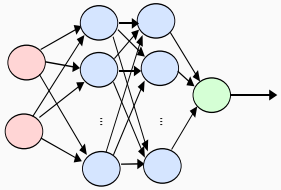


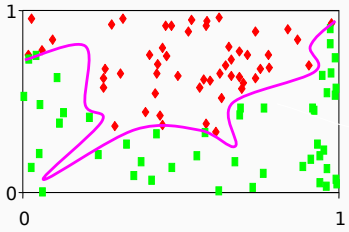
2+2+1 neurons





10+10+1 neurons





Estimated decision regions:
$\hat{C}_1 = \{\boldsymbol{x} \in \mathbb{R}^d,\ f_{\boldsymbol{\theta}}(\boldsymbol{x}) \geqslant \frac{1}{2}\}$ and
$\hat{C}_0 = \mathbb{R}^d \setminus \hat{C}_1$.

Complexity/capacity of the network

$\Rightarrow$

Trade-off between generalization and overfitting.

- **Training:** Logistic regression for binary classification: Maximum likelihood of the dataset (opposite of binary crosss-entropy : `BCELoss` in PyTorch):

$$\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \left[y^{(i)} \log f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + (1 - y^{(i)}) \log \left(1 - f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) \right) \right]$$

- For neural networks, the probability $f_{\boldsymbol{\theta}}$ is obtained using the **sigmoid function** $\sigma(t) = \dfrac{e^t}{1+e^t} = \dfrac{1}{1+e^{-t}}$ as the activation function of the last layer.
- Beware that $y^{(i)} = 0$ or 1 so only one term is non-zero.
- One could instead regroup the terms of the sum according to the label values:

$$\max_{\boldsymbol{\theta}} \sum_{\substack{(\boldsymbol{x}^{(i)}, y^{(i)}) \in \mathcal{D} \\ \text{s.t. } y^{(i)}=1}}^{N} \log f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \sum_{\substack{(\boldsymbol{x}^{(i)}, y^{(i)}) \in \mathcal{D} \\ \text{s.t. } y^{(i)}=0}}^{N} \log \left(1 - f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) \right)$$

Generative Adversarial Networks (GAN)

- **Discriminator network:** Consider two sets
  - $\mathcal{D}_{\text{real}}$: a dataset of n real images **(real = labeled with $y^{(i)} = 1$)**,
  - $\mathcal{D}_{\text{fake}}$: a dataset of m fake images $\boldsymbol{x} = G_{\theta_g}(\boldsymbol{z})$ **(fake = labeled with $y^{(i)} = 0$)**.

- **Goal:** Find the parameters θ_d of a binary classification network $\boldsymbol{x} \mapsto D_{\theta_d}(\boldsymbol{x})$ meant to classify real and fake images.
  Minimize the binary cross-entropy, or maximize its negation

$$\max_{\theta_d} \underbrace{\sum_{\boldsymbol{x}_{\text{real}} \in \mathcal{D}_{\text{real}}} \log D_{\theta_d}(\boldsymbol{x}_{\text{real}})}_{\substack{\text{force predicted labels to be 1} \\ \text{for real images}}} + \underbrace{\sum_{\boldsymbol{x}_{\text{fake}} \in \mathcal{D}_{\text{fake}}} \log(1 - D_{\theta_d}(\boldsymbol{x}_{\text{fake}}))}_{\substack{\text{force predicted labels to be 0} \\ \text{for fake images}}}$$

- **How:** Use gradient ascent (Adam).

Generative Adversarial Networks (GAN)

- **Generator network:** Consider a given discriminative model $\boldsymbol{x} \mapsto D_{\theta_d}(\boldsymbol{x})$ and consider $\mathcal{D}_{\text{rand}}$ a set of m random latent vectors.
- **Goal:** Find the parameters θ_g of a network $\boldsymbol{z} \mapsto G_{\theta_g}(\boldsymbol{z})$ generating images from random vectors $\boldsymbol{z}$ such that it fools the discriminator

$$\min_{\theta_g} \underbrace{\sum_{\boldsymbol{z} \in \mathcal{D}_{\text{rand}}} \log(1 - D_{\theta_d}(G_{\theta_g}(\boldsymbol{z})))}_{\substack{\text{force the discriminator to think that} \\ \text{our generated fake images are not fake (away from 0)}}} \tag{1}$$

or alternatively (works better in practice)

$$\max_{\theta_g} \underbrace{\sum_{\boldsymbol{z} \in \mathcal{D}_{\text{rand}}} \log D_{\theta_d}(G_{\theta_g}(\boldsymbol{z}))}_{\substack{\text{force the discriminator to think that} \\ \text{our generated fake images are real (close to 1)}}} \tag{2}$$

- **How:** Gradient descent for (1) or gradient ascent for (2) (Adam)

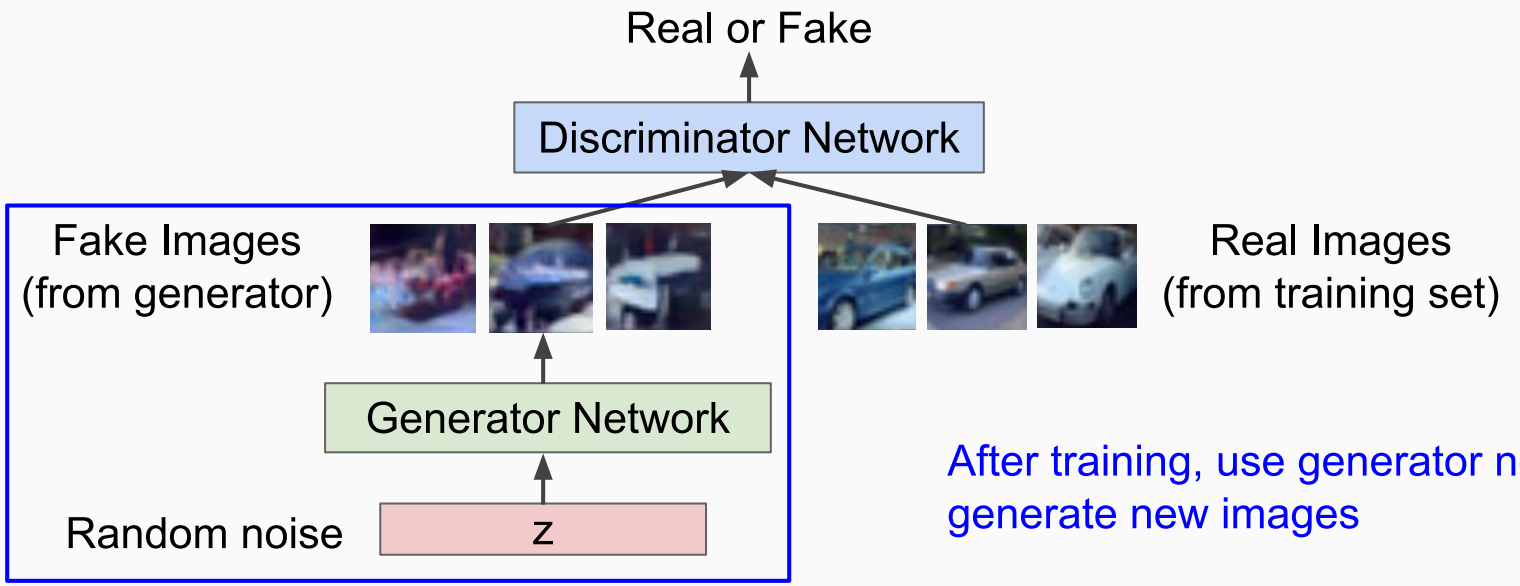
Generative Adversarial Networks (GAN)

- **Train both networks jointly**.
- Minimax loss in a two player game (each player is a network):

$$\min_{\theta_g} \max_{\theta_d} \sum_{\boldsymbol{x} \in \mathcal{D}_{\text{real}}} \log D_{\theta_d}(\boldsymbol{x}) + \sum_{\boldsymbol{z} \in \mathcal{D}_{\text{rand}}} \log(1 - D_{\theta_d}(\underbrace{G_{\theta_g}(\boldsymbol{z})}_{\text{fake}}))$$

- **Training algorithm:** Repeat until convergence
  1. Fix θ_g, update θ_d with one step of gradient ascent,
  2. Fix θ_d, update θ_g with one step of gradient descent for (1), (or one step of gradient ascent for (2).)

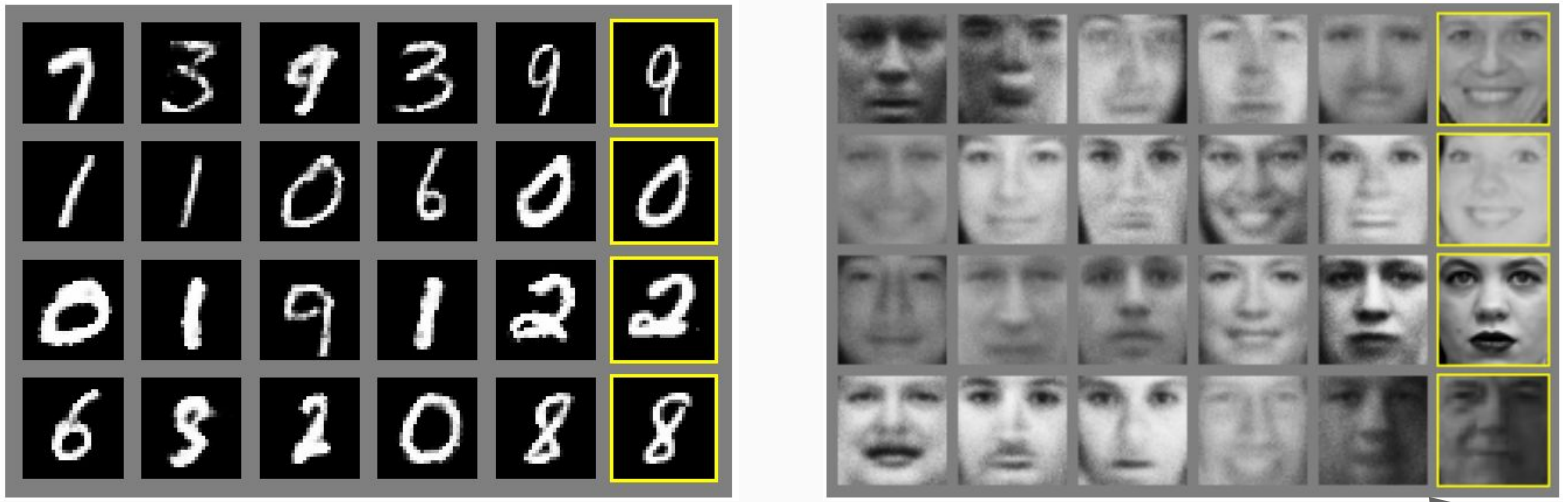
Generative Adversarial Networks (GAN)



After training, use generator network to generate new images

Generative Adversarial Networks (GAN)

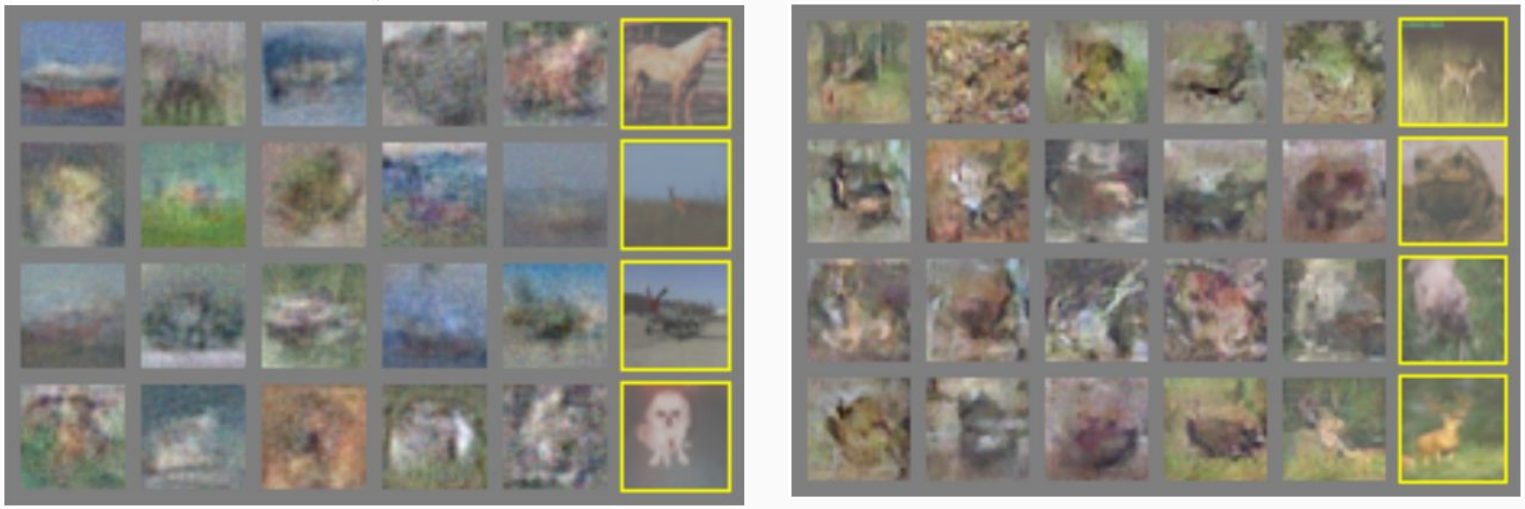
Generated samples



Nearest neighbor from training set

Generative Adversarial Networks (GAN)

Generated samples (CIFAR-10)

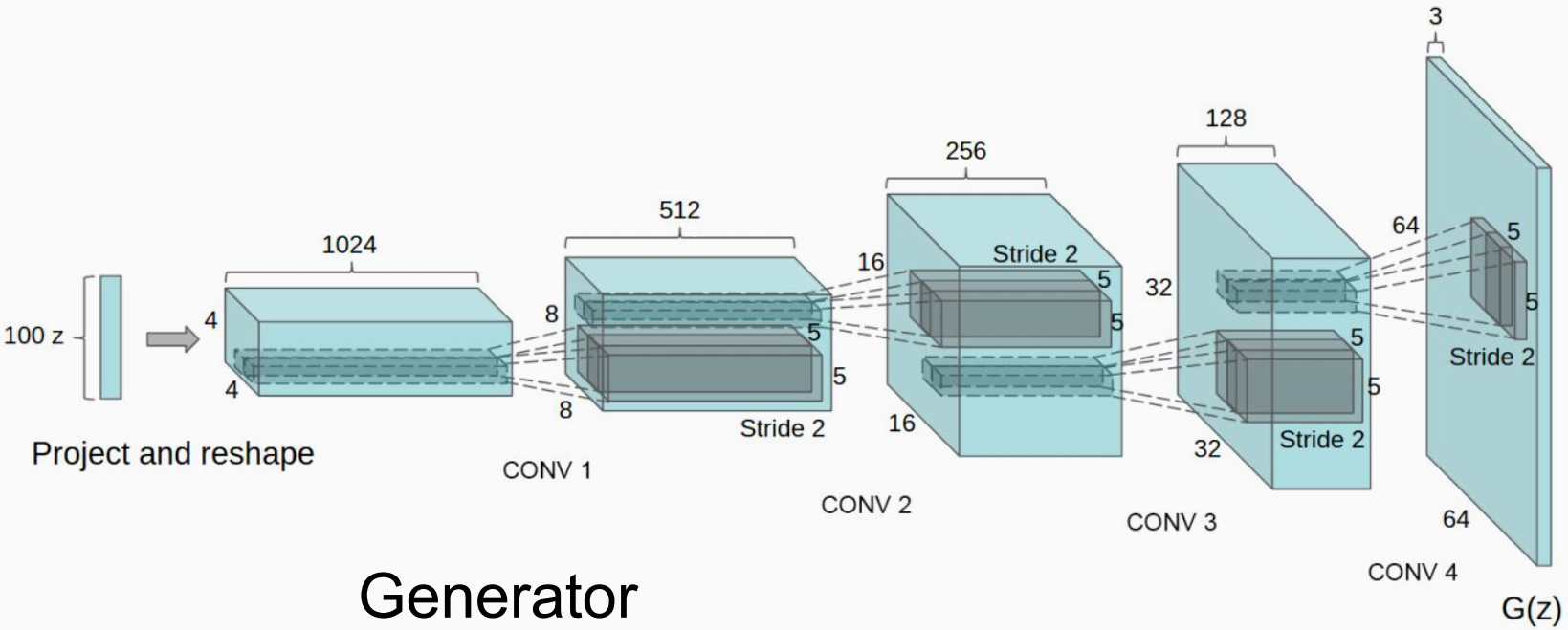


Nearest neighbor from training set

Convolutional GAN

[Radford et al., 2016]

- **Generator:** upsampling network with **fractionally strided convolutions** (i.e. the transpose operator of convolution+subsampling , called `ConvTranspose2d` in PyTorch),
- **Discriminator:** convolutional network with strided convolutions.

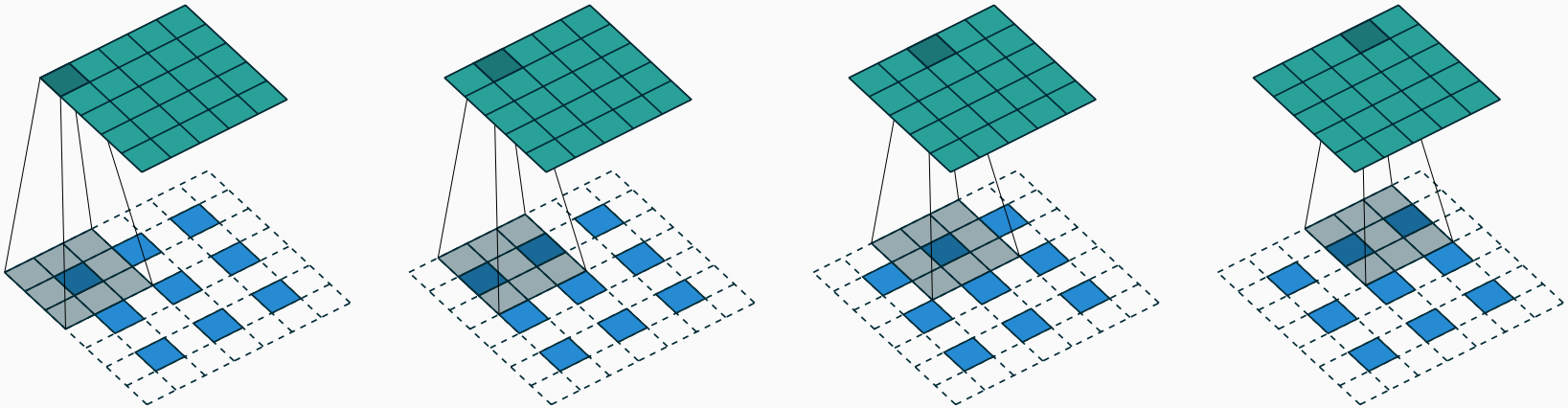


Generator

Transposed convolution arithmetic

Fractionally strided convolutions:

- This is the transpose operator of convolution+subsampling (convolution with stride).
- Called `ConvTranspose2d` in PyTorch

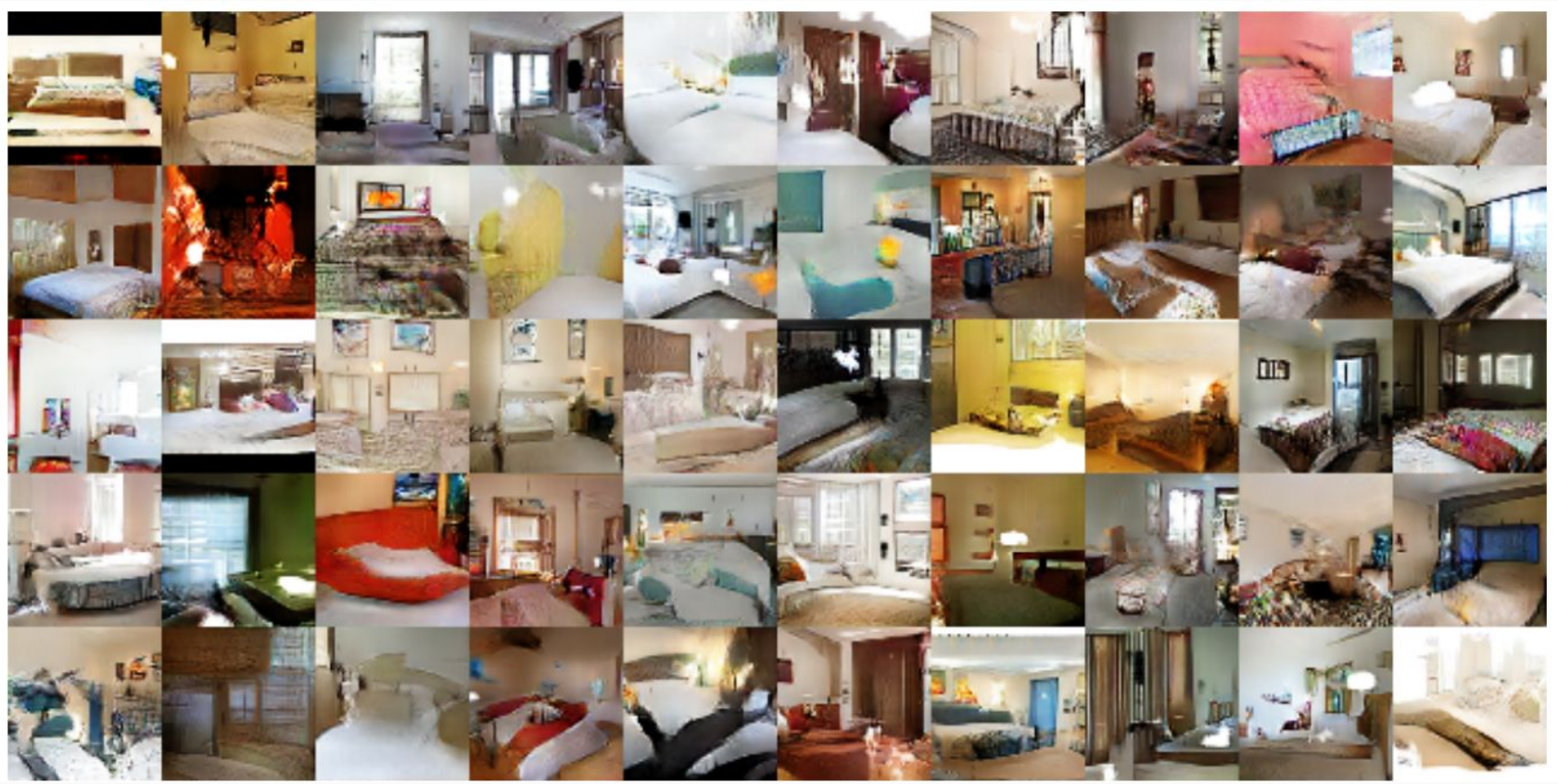


The transpose of convolving a 3×3 kernel over a 5×5 input padded with a 1×1 border of zeros using 2×2 strides (i.e., $i = 5$, $k = 3$, $s = 2$ and $p = 1$). It is equivalent to convolving a 3×3 kernel over a 3×3 input (with 1 zero inserted between inputs) padded with a 1×1 border of zeros using unit strides (i.e., $i' = 3$, $\tilde{i}' = 5$, $k' = k$, $s' = 1$ and $p' = 1$).

(source: From [Dumoulin and Visin, 2016])

Convolutional GAN

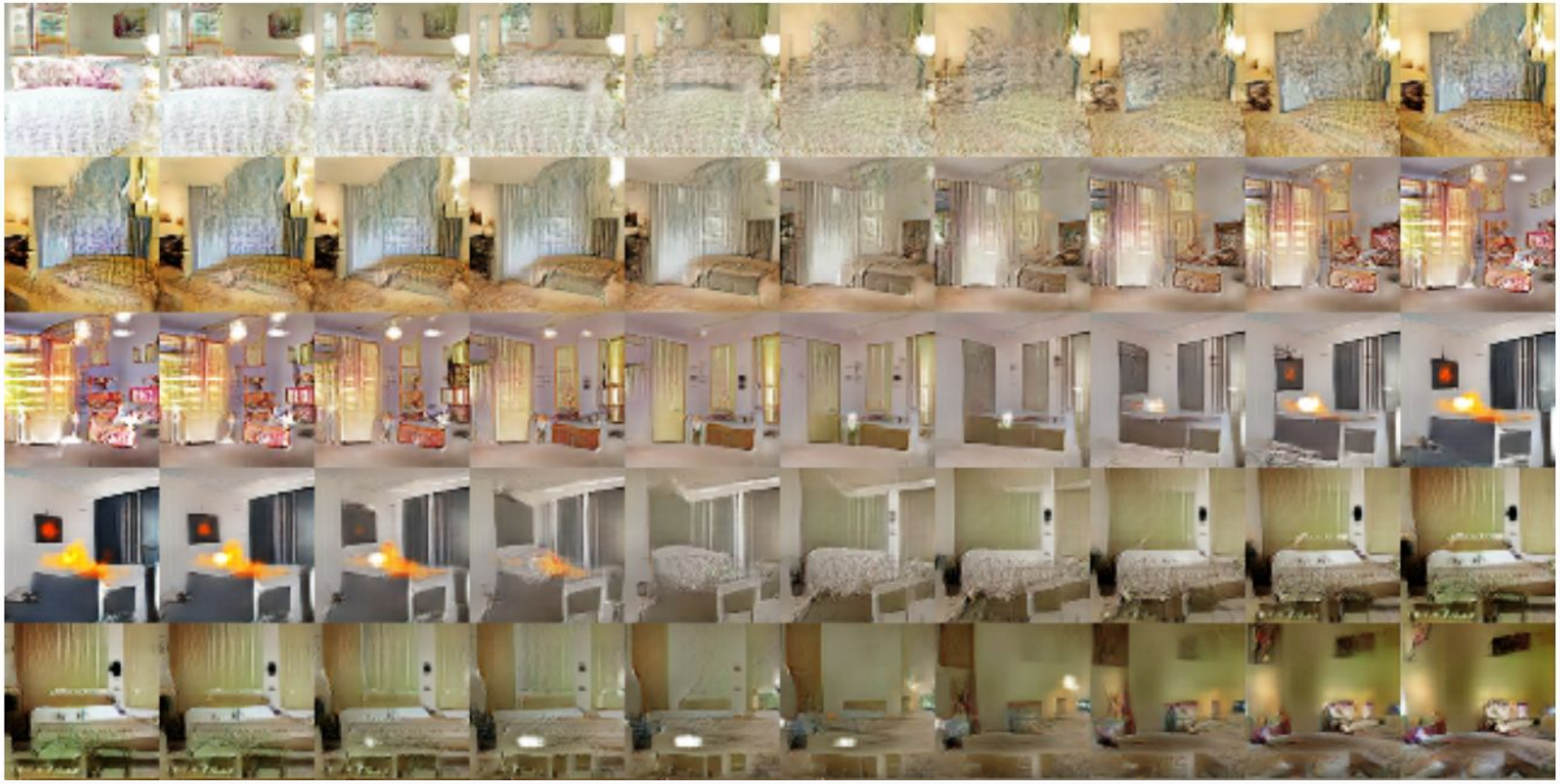
[Radford et al., 2016]



Generations of realistic bedrooms pictures, from randomly generated latent variables.

Convolutional GAN

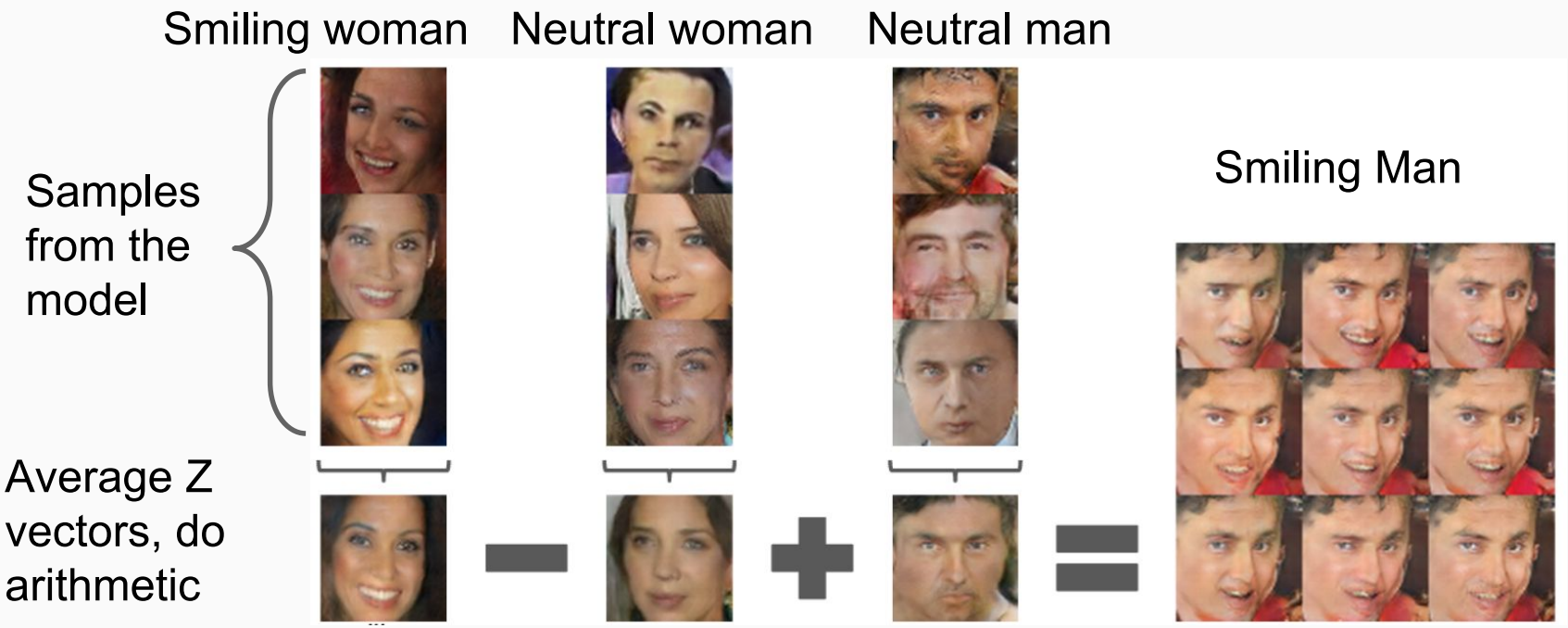
[Radford et al., 2016]



Interpolation in between points in latent space.

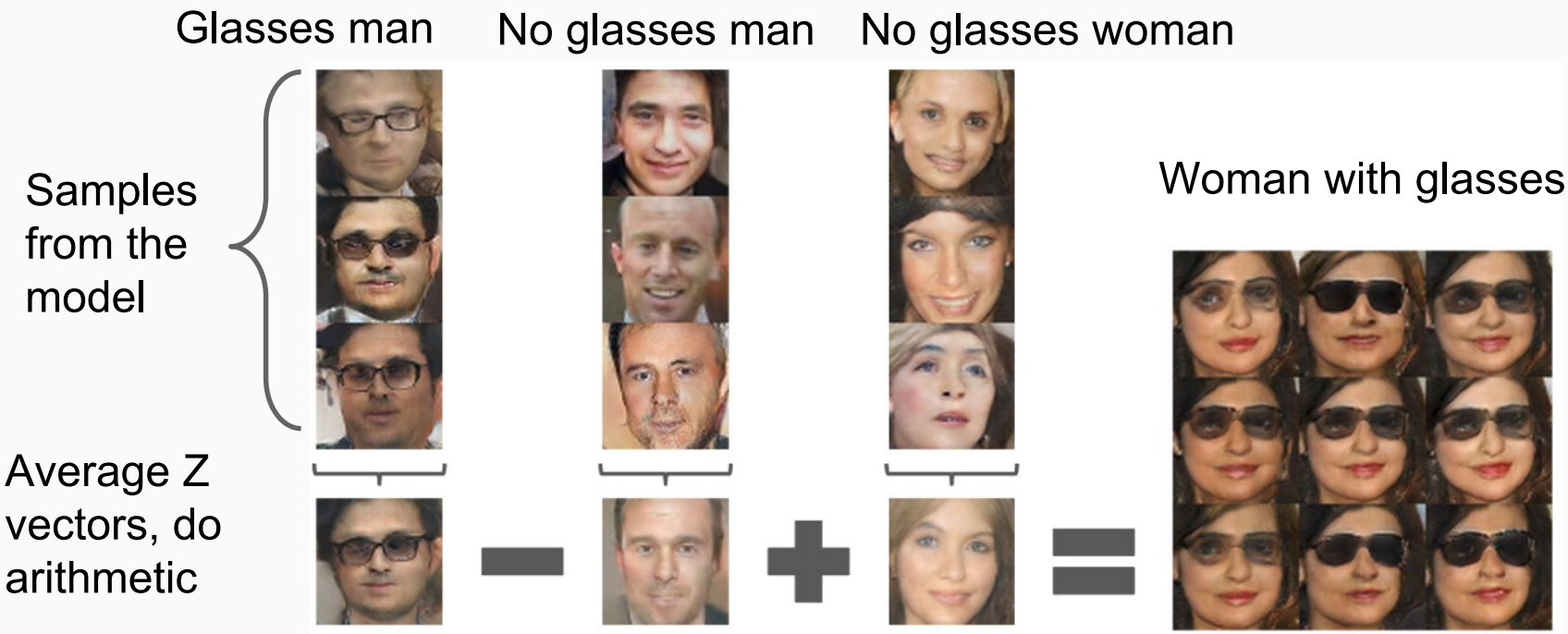
Convolutional GAN – Arithmetic

[Radford et al., 2016]



Convolutional GAN – Arithmetic

[Radford et al., 2016]



Generative Aversarial Networks: Style GAN [Karras et al., 2019]

Image size:
1024× 1024 px
(source: Karras et al.)

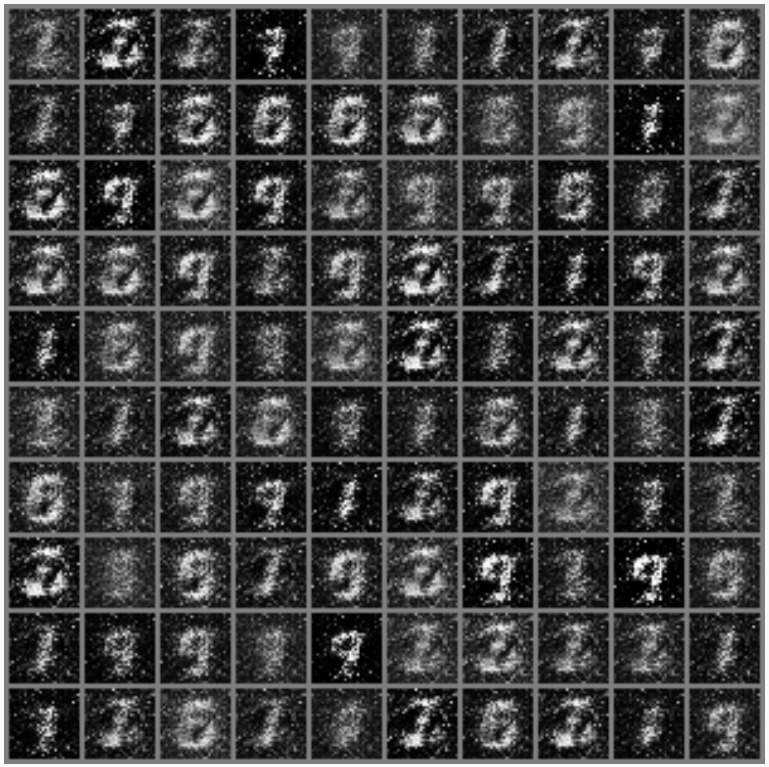
GAN Training

Training with MNIST (60 000 images)

- Adam optimizer
- Learning rate 0.0002 for both the discriminator and the generator

Real images:

Fake images, epoch 1:



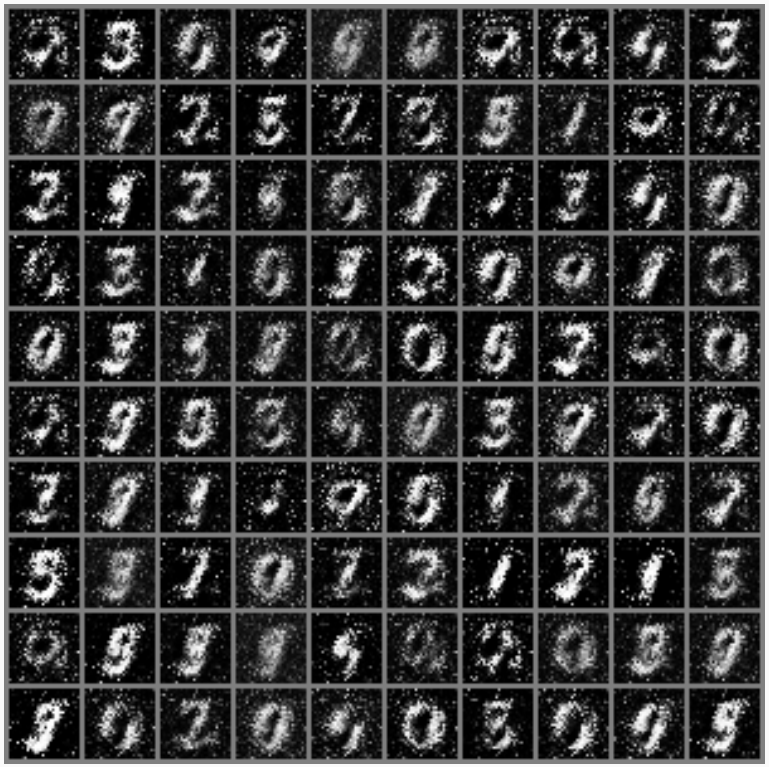
GAN Training

Training with MNIST (60 000 images)

- Adam optimizer
- Learning rate 0.0002 for both the discriminator and the generator

Real images:

Fake images, epoch 2:



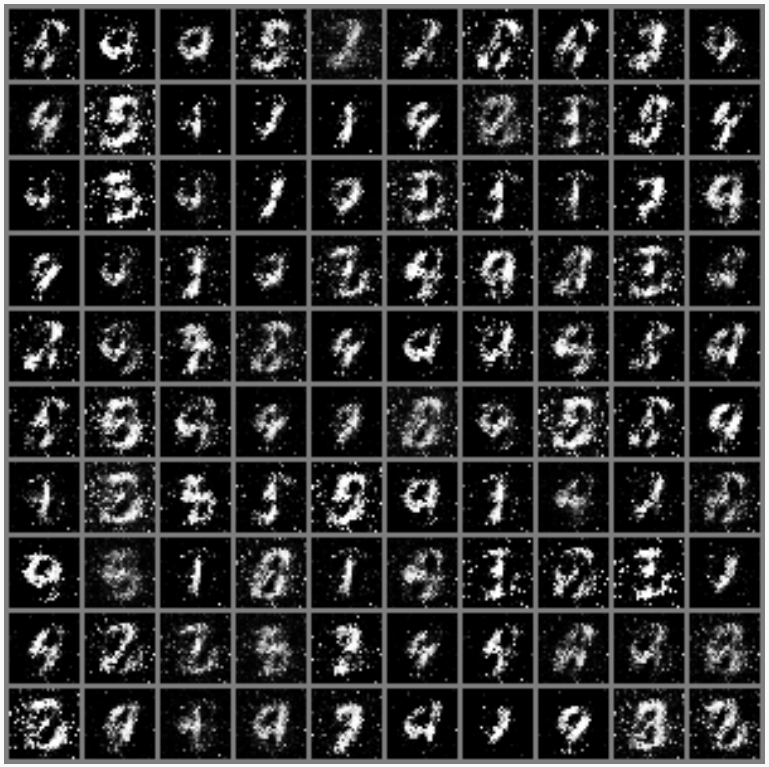
GAN Training

Training with MNIST (60 000 images)

- Adam optimizer
- Learning rate 0.0002 for both the discriminator and the generator

Real images:

Fake images, epoch 3:



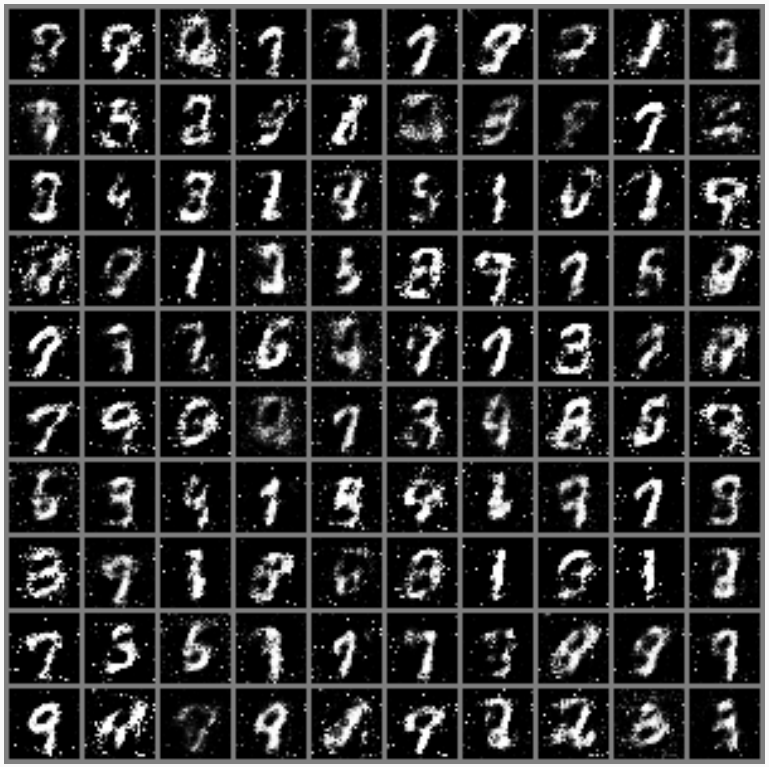
GAN Training

Training with MNIST (60 000 images)

- Adam optimizer
- Learning rate 0.0002 for both the discriminator and the generator

Real images:

Fake images, epoch 10:



GAN Training

Training with MNIST (60 000 images)

- Adam optimizer
- Learning rate 0.0002 for both the discriminator and the generator

Real images:

Fake images, epoch 100:

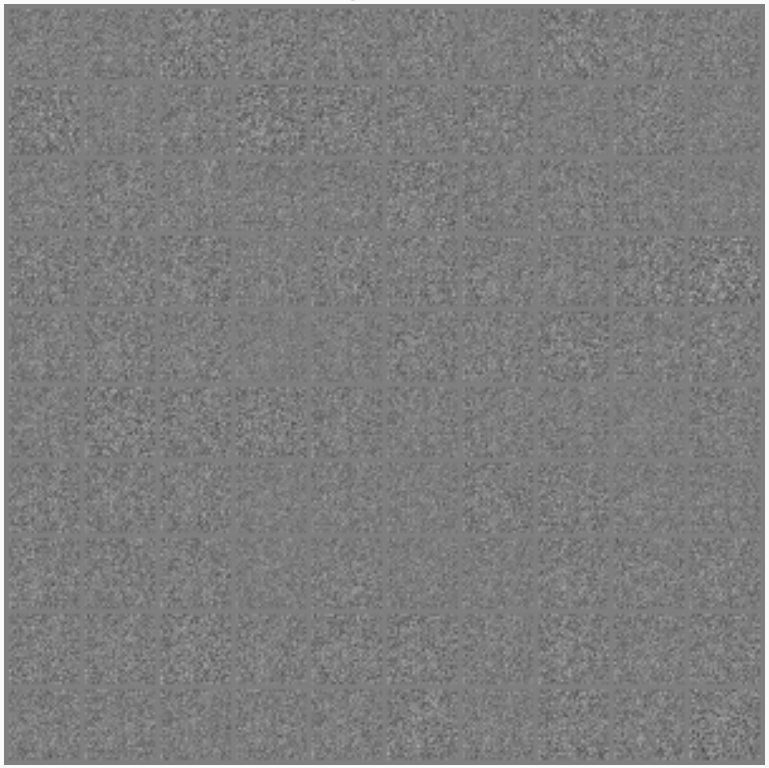
Training GANs is quite unstable!

The generator can suffer *mode collapse*: It always produces the same image (one mode only).

Same as before **but with SGD instead of Adam**.

Real images:

Fake images, epoch 1:



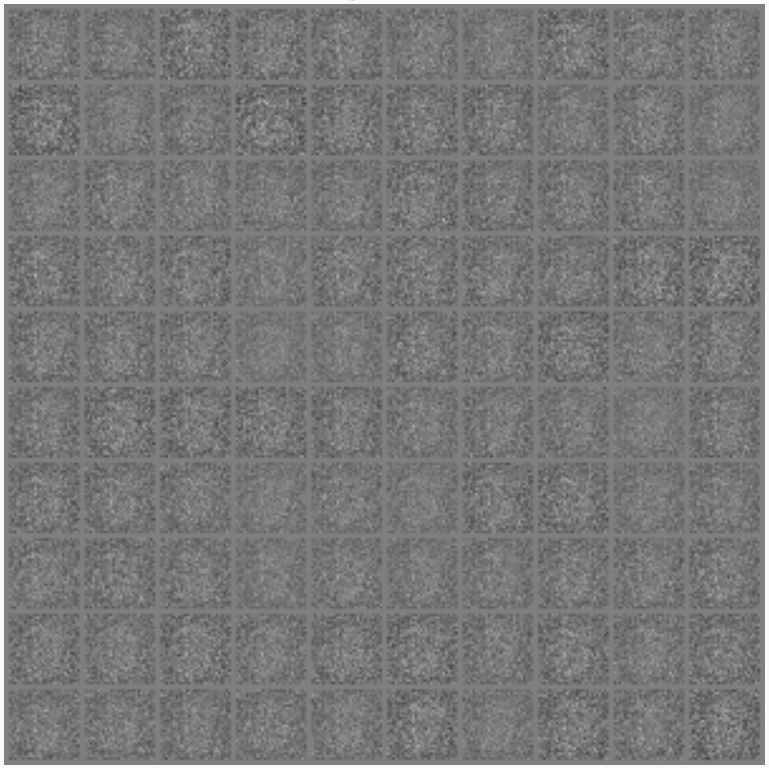
Training GANs is quite unstable!

The generator can suffer *mode collapse*: It always produces the same image (one mode only).

Same as before **but with SGD instead of Adam**.

Real images:

Fake images, epoch 2:



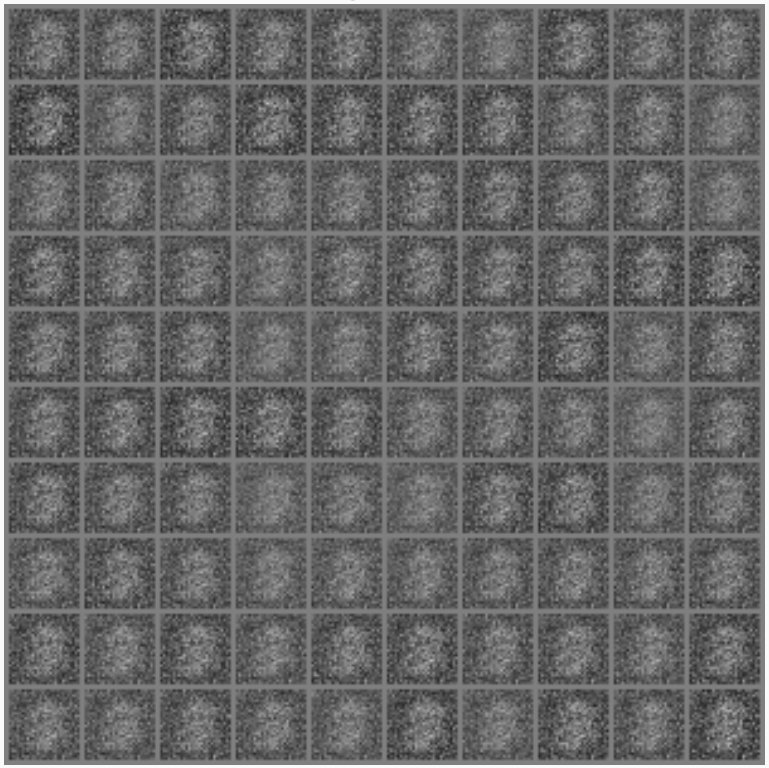
Training GANs is quite unstable!

The generator can suffer *mode collapse*: It always produces the same image (one mode only).

Same as before **but with SGD instead of Adam**.

Real images:

Fake images, epoch 3:



GAN Training

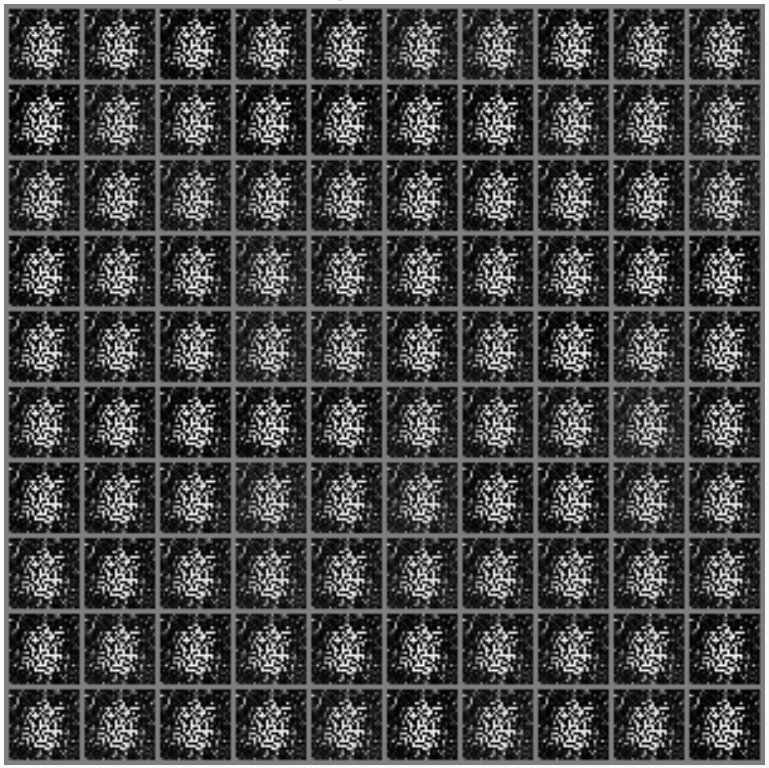
Training GANs is quite unstable!

The generator can suffer *mode collapse*: It always produces the same image (one mode only).

Same as before **but with SGD instead of Adam**.

Real images:

Fake images, epoch 10:



GAN Training

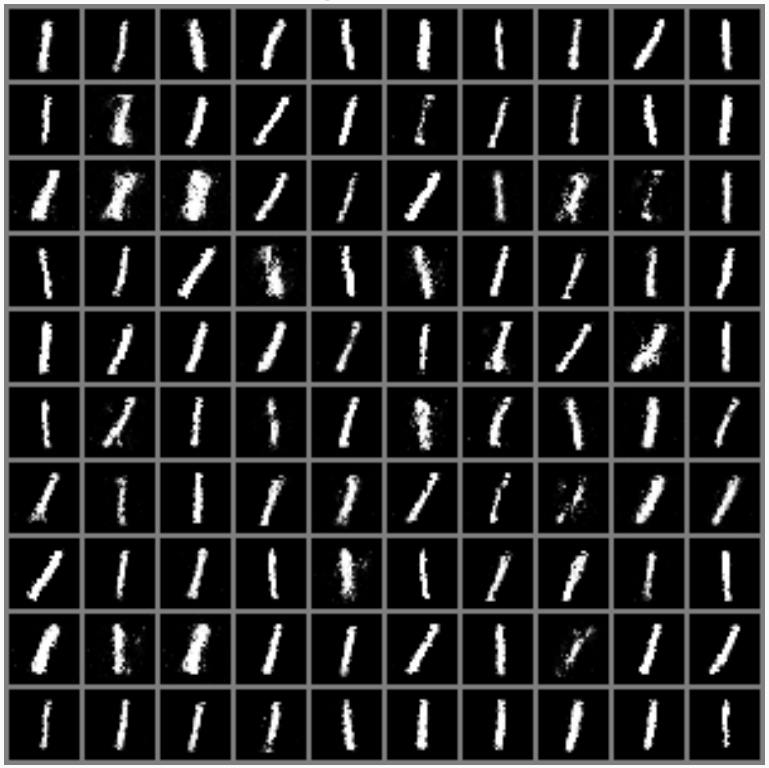
Training GANs is quite unstable!

The generator can suffer *mode collapse*: It always produces the same image (one mode only).

Same as before **but with SGD instead of Adam**.

Real images:

Fake images, epoch 100:



Conditional Generative Models

Conditional GANs

- Conditional GANs: Train the generator and the discriminator by passing some information about the images.

Example: Class conditional generator and discriminator

- **Generator:** Generate a fake "3".
- **Discriminator:** Is it a real or a fake "3"?

Unconditional training:

$$\min_{\theta_g} \max_{\theta_d} \sum_{\boldsymbol{x} \in \mathcal{D}_{\text{real}}} \log D_{\theta_d}(\boldsymbol{x}) + \sum_{\boldsymbol{z} \in \mathcal{D}_{\text{rand}}} \log(1 - D_{\theta_d}(\underbrace{G_{\theta_g}(\boldsymbol{z})}_{\text{fake}})$$

Class conditional training:

$$\min_{\theta_g} \max_{\theta_d} \sum_{(\boldsymbol{x}, c) \in \mathcal{D}_{\text{real}}} \log D_{\theta_d}(\boldsymbol{x}, c) + \sum_{(\boldsymbol{z}, c) \in \mathcal{D}_{\text{rand}}} \log(1 - D_{\theta_d}(\underbrace{G_{\theta_g}(\boldsymbol{z}, c)}_{\text{fake}}, c)$$

Conditional GANs

- Conditional GANs: Train the generator and the discriminator by passing some information about the images.

Example: Class conditional generator and discriminator

- **Generator:** Generate a fake "3".
- **Discriminator:** Is it a real or a fake "3"?

Unconditional training:

$$\min_{\theta_g} \max_{\theta_d} \sum_{\boldsymbol{x} \in \mathcal{D}_{\text{real}}} \log D_{\theta_d}(\boldsymbol{x}) + \sum_{\boldsymbol{z} \in \mathcal{D}_{\text{rand}}} \log(1 - D_{\theta_d}(\underbrace{G_{\theta_g}(\boldsymbol{z})}_{\text{fake}})$$

Class conditional training:

$$\min_{\theta_g} \max_{\theta_d} \sum_{(\boldsymbol{x}, c) \in \mathcal{D}_{\text{real}}} \log D_{\theta_d}(\boldsymbol{x}, c) + \sum_{(\boldsymbol{z}, c) \in \mathcal{D}_{\text{rand}}} \log(1 - D_{\theta_d}(\underbrace{G_{\theta_g}(\boldsymbol{z}, c)}_{\text{fake}}, c)$$

- **Discussion:** How to do this in practice?

Conditional GANs

- Conditional GANs: Train the generator and the discriminator by passing some information about the images.

Example: Class conditional generator and discriminator

- **Generator:** Generate a fake "3".
- **Discriminator:** Is it a real or a fake "3"?

Unconditional training:

$$\min_{\theta_g} \max_{\theta_d} \sum_{\boldsymbol{x} \in \mathcal{D}_{\text{real}}} \log D_{\theta_d}(\boldsymbol{x}) + \sum_{\boldsymbol{z} \in \mathcal{D}_{\text{rand}}} \log(1 - D_{\theta_d}(\underbrace{G_{\theta_g}(\boldsymbol{z})}_{\text{fake}}))$$

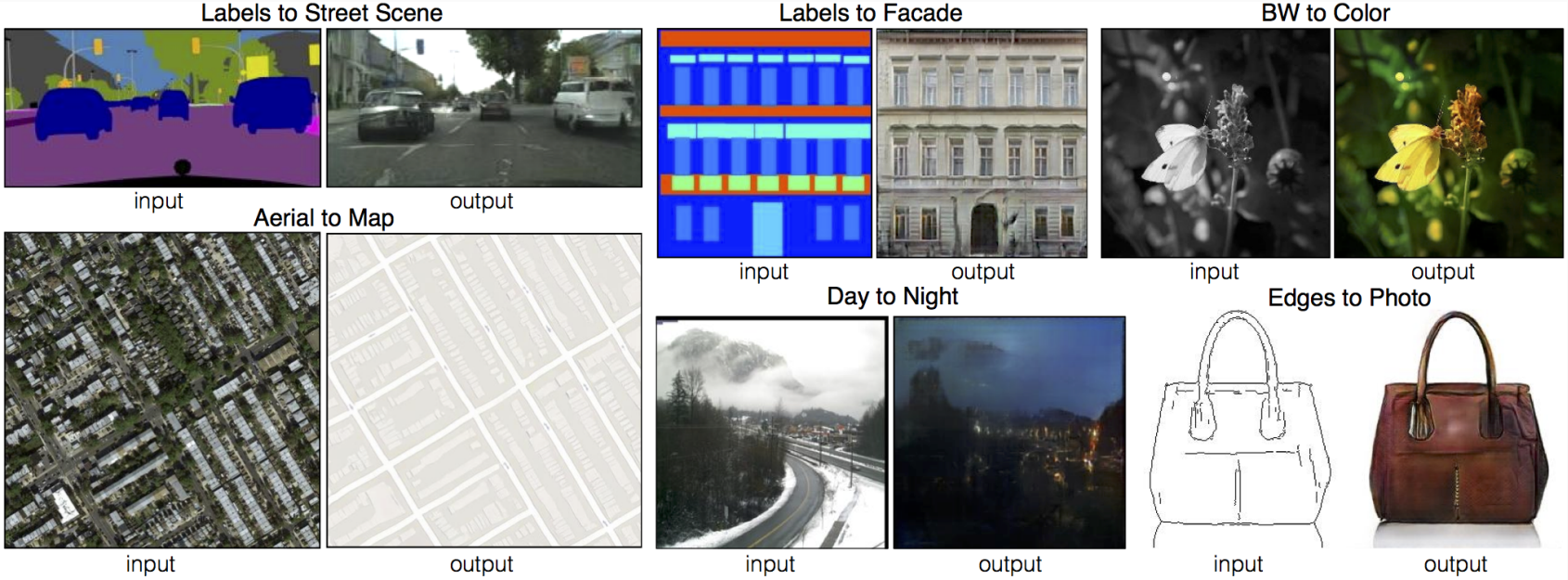
Class conditional training:

$$\min_{\theta_g} \max_{\theta_d} \sum_{(\boldsymbol{x}, c) \in \mathcal{D}_{\text{real}}} \log D_{\theta_d}(\boldsymbol{x}, c) + \sum_{(\boldsymbol{z}, c) \in \mathcal{D}_{\text{rand}}} \log(1 - D_{\theta_d}(\underbrace{G_{\theta_g}(\boldsymbol{z}, c)}_{\text{fake}}, c)$$

- **Discussion:** How to do this in practice? Use `torch.nn.Embedding`.

Conditional GANs: image-to-image translation

Pix2pix: Image-to-Image Translation with Conditional Adversarial Nets [Isola et al., 2017]



(source: From [Isola et al., 2017])

- Training using a set of image pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$
- GAN conditioned on input image $\boldsymbol{x}$ to produce $\boldsymbol{y} = G(\boldsymbol{x})$.
- Opens the way for new creative tools

Conditional GANs: image-to-image translation

Pix2pix: Image-to-Image Translation with Conditional Adversarial Nets [Isola et al., 2017]

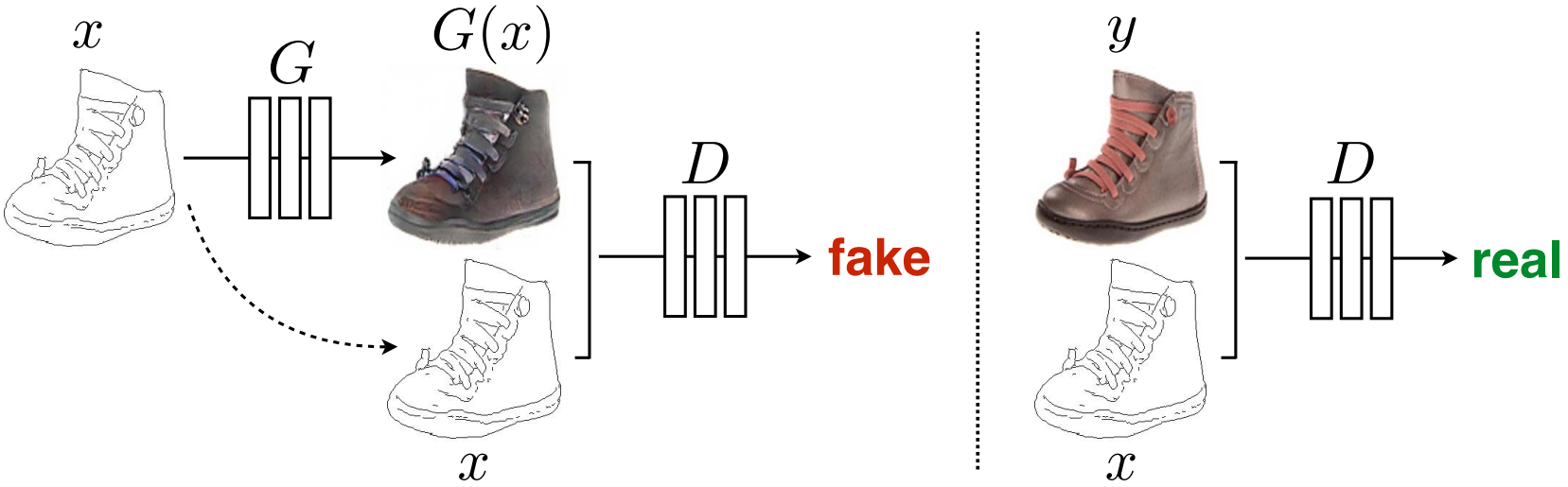
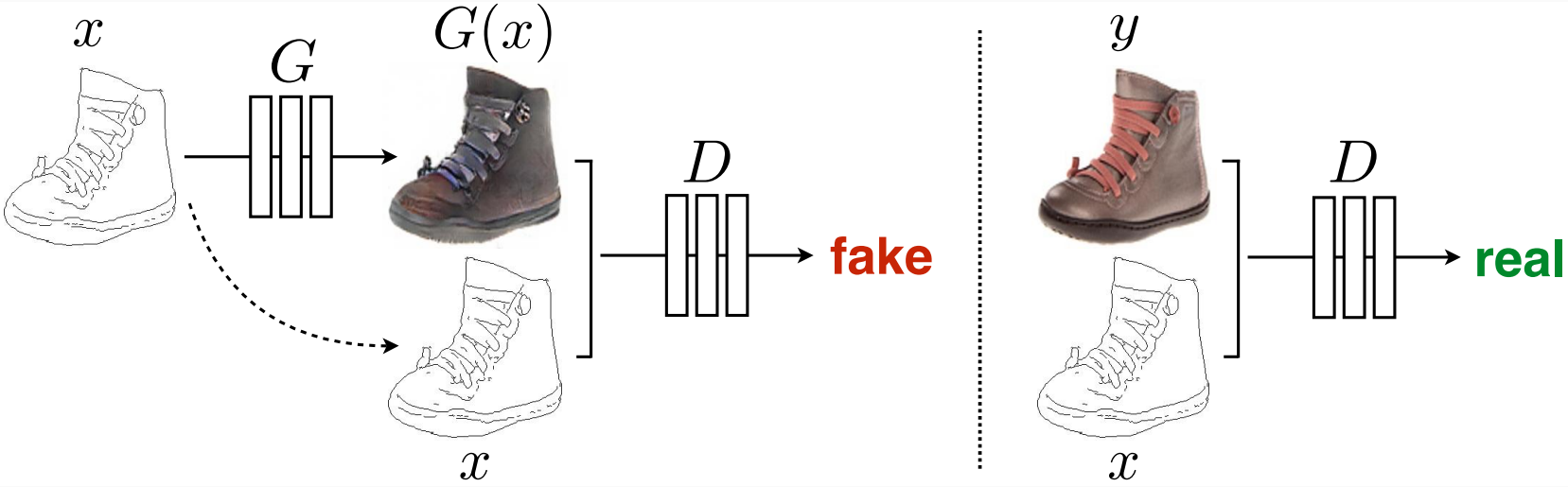


Figure 2: Training a conditional GAN to map edges→photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

(source: From [Isola et al., 2017])

Conditional GANs: image-to-image translation

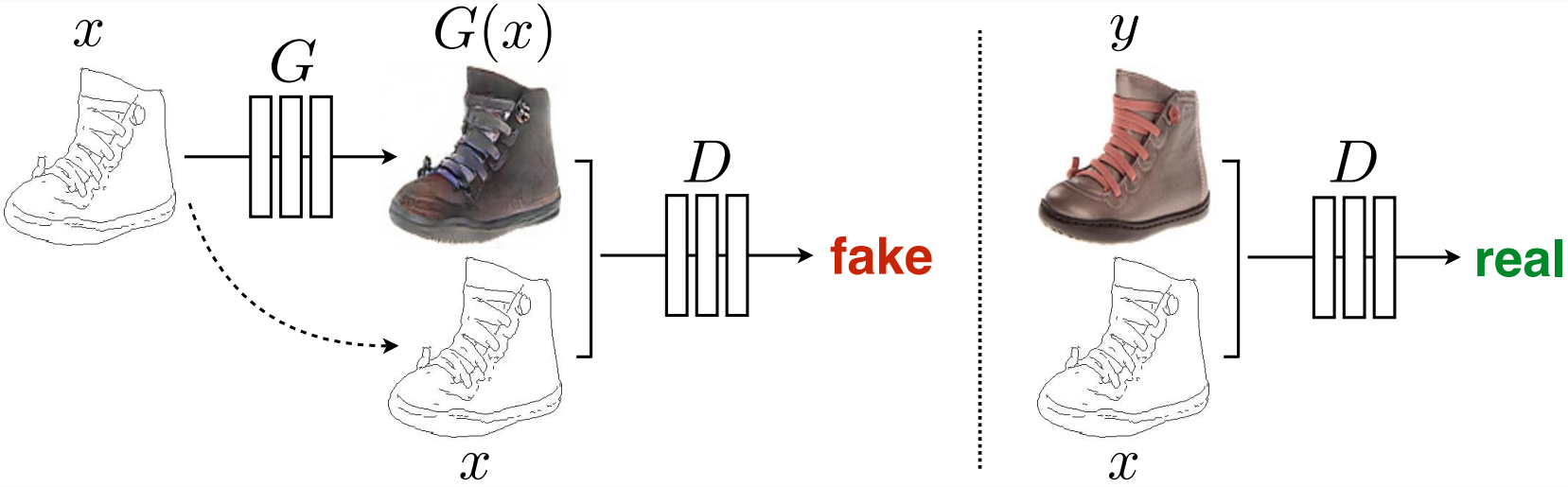


(source: From [Isola et al., 2017])

Architecture details:

- Generator: **U-net architecture** (see recap later)
- Discriminator: Patch discriminator applied to each 70×70 patch, and the score is spatially average.
- Both are fully convolutional so larger images can be used at test time.

Conditional GANs: image-to-image translation



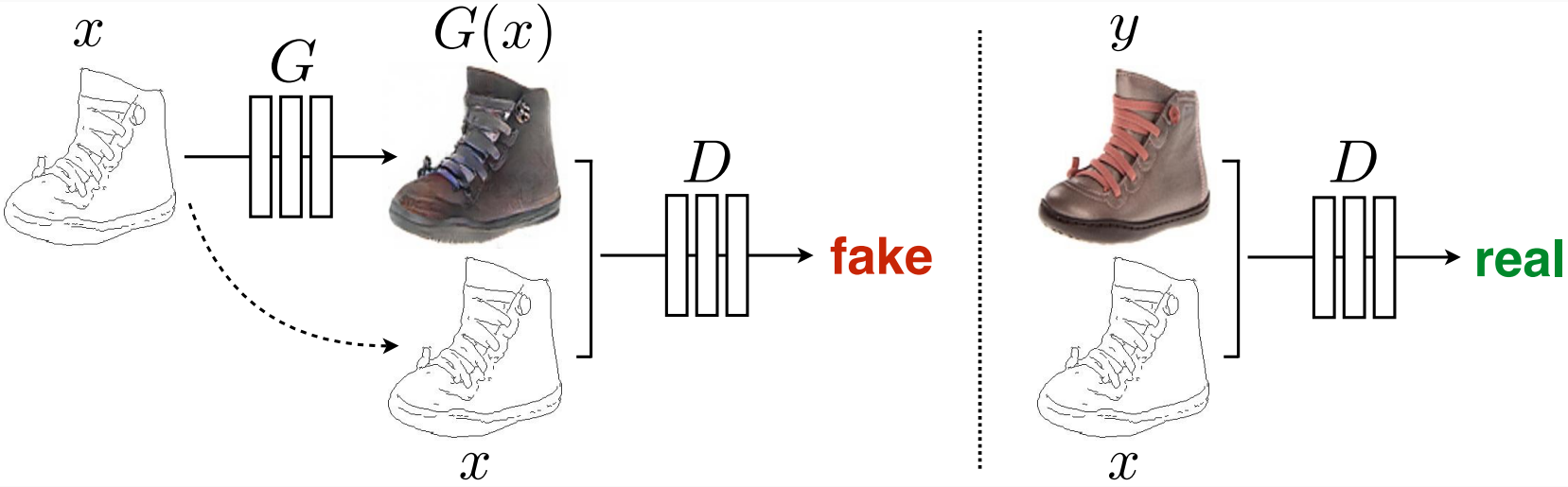
(source: From [Isola et al., 2017])

Architecture details:

- Generator: **U-net architecture** (see recap later)
- Discriminator: Patch discriminator applied to each 70×70 patch, and the score is spatially average.
- Both are fully convolutional so larger images can be used at test time.

Question: What is missing here?

Conditional GANs: image-to-image translation



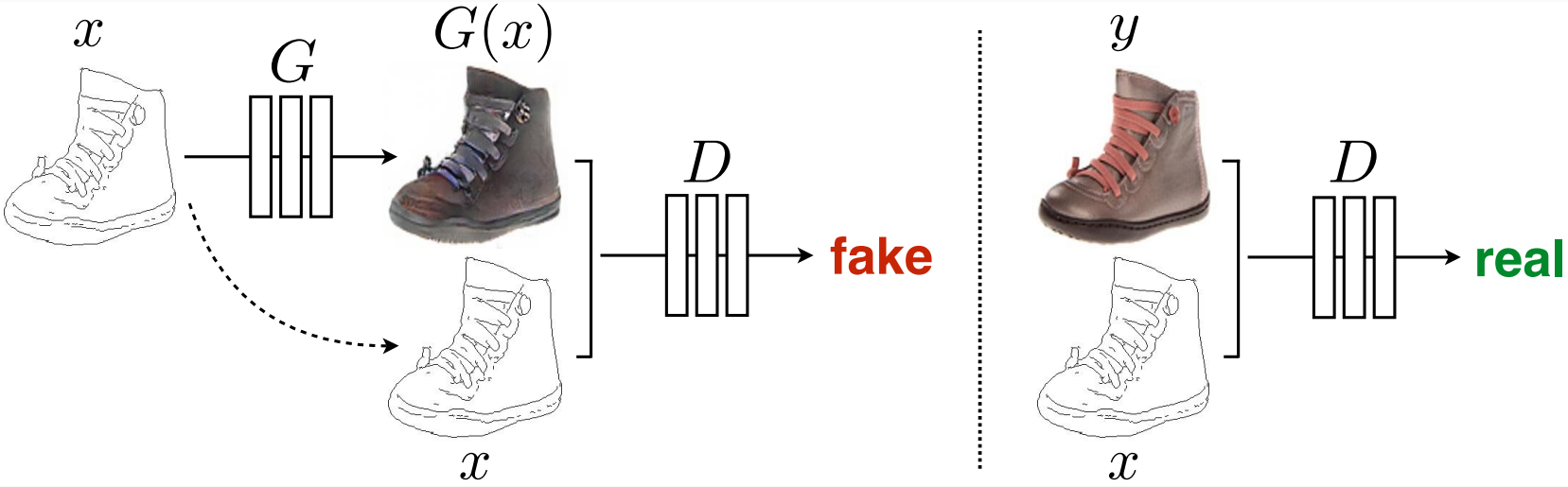
(source: From [Isola et al., 2017])

Architecture details:

- Generator: **U-net architecture** (see recap later)
- Discriminator: Patch discriminator applied to each 70×70 patch, and the score is spatially average.
- Both are fully convolutional so larger images can be used at test time.

Question: What is missing here? No latent code z in the generator, but randomness thanks to dropout in the network. Still stochasticity is limited given an input x.

Conditional GANs: image-to-image translation



(source: From [Isola et al., 2017])

Training loss:

- The training of generator combines a GAN loss and an ℓ_1 loss:

$$\min_{\theta_g} \max_{\theta_d} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \log D_{\theta_d}(\boldsymbol{y},\boldsymbol{x}) + \log(1 - D_{\theta_d}(\underbrace{G_{\theta_g}(\boldsymbol{x})}_{\text{fake}},\boldsymbol{x}) + \|\underbrace{G_{\theta_g}(\boldsymbol{x})}_{\text{fake}} - \boldsymbol{y}\|_1$$

- The discriminator looks at generated patches while the ℓ_1 loss is global.
- This is a mixed loss between classical supervised training and unsupervised GAN training.

Conditional GANs: image-to-image translation

Figure 4: Different losses induce different quality of results. Each column shows results trained under a different loss. Please see `https://phillipi.github.io/pix2pix/` for additional examples.

(source: From [Isola et al., 2017])

Adversarial loss for inverse problems

- With the pix2pix example we see that an adversarial loss is used to produce realistic images.
- Using only ℓ_1 or ℓ_2 loss leads to blurry results. This is known as "regression to the mean" issue.
- Adversarial losses helps to improve the visual aspect, but can also hallucinate details. Use with care in scientific context.
- Introducing generative capabilities to networks is especially important for tasks where content has to be inferred with few to no available information, such as **super-resolution**.

SRGAN [Ledig et al., 2017]

Adversarial loss: Same as GAN but replace the latent code by the LR image

$$\min_{\theta_{\text{SR}}} \max_{\theta_{\text{D}}} \sum \log D_{\theta_D}(\boldsymbol{x}^{\text{HR}}) + \log(1 - D_{\theta_D}(\boldsymbol{x}^{\text{SR}})) + \lambda L_{\text{content}}(\boldsymbol{x}^{\text{SR}}, \boldsymbol{x}^{\text{HR}})$$

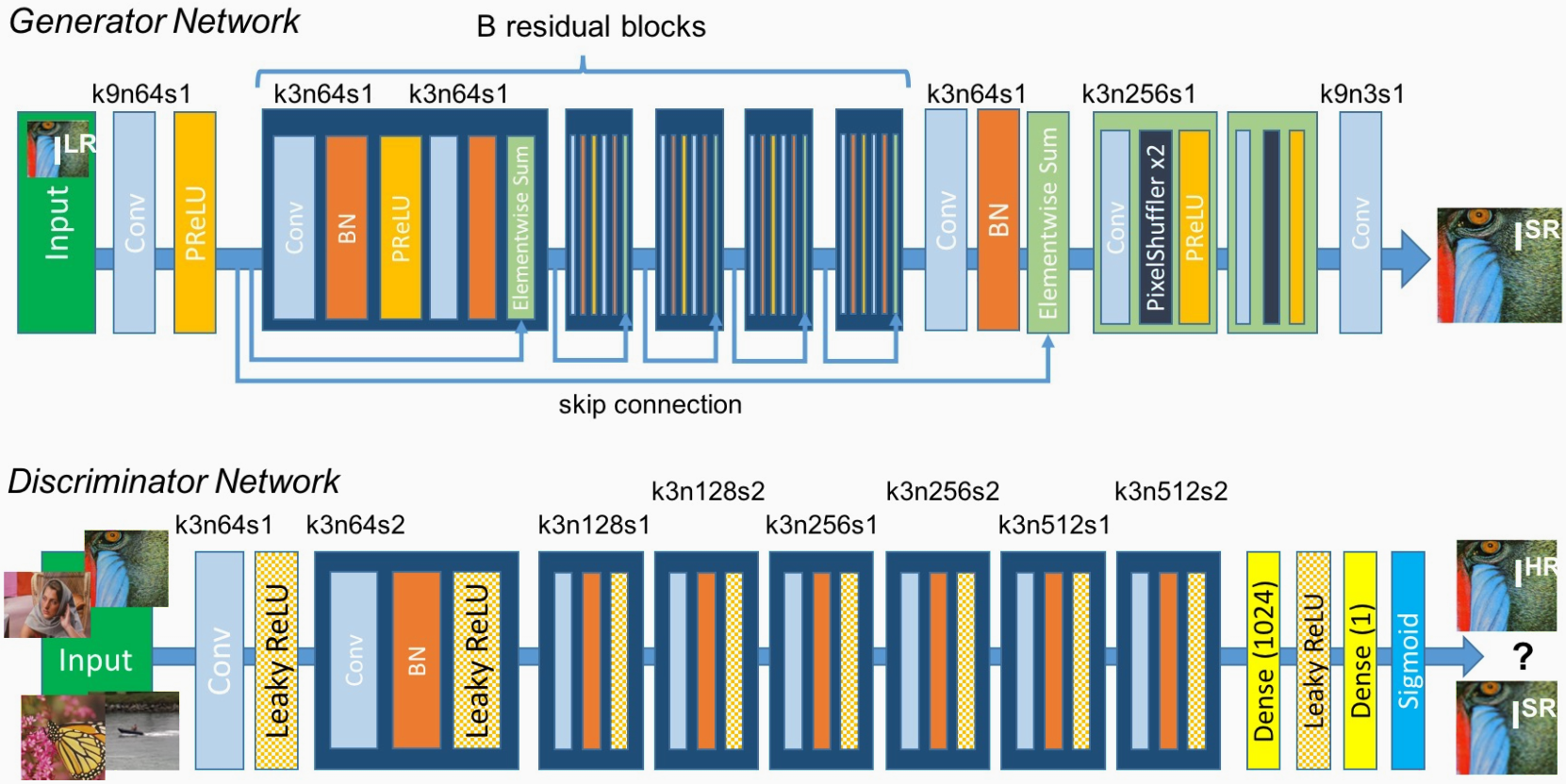
$$\text{where} \quad \boldsymbol{x}^{\text{SR}} = G_{\theta_{\text{SR}}}(\boldsymbol{x}^{\text{LR}}) \quad \text{and} \quad \boldsymbol{x}^{\text{LR}} = \boldsymbol{H}(\boldsymbol{x}^{\text{HR}})$$

Content loss: Euclidean distance between the L^{th} feature tensors obtained with VGG for the SR and HR images, respectively:

$$L_{\text{content}}(\boldsymbol{x}^{\text{SR}}, \boldsymbol{x}^{\text{HR}}) = \|\boldsymbol{h}^{\text{SR}} - \boldsymbol{h}^{\text{HR}}\|_2^2 \quad \text{with} \quad \begin{cases} \boldsymbol{h}^{\text{SR}} = \text{VGG}^L(\boldsymbol{x}^{\text{SR}}) \\ \boldsymbol{h}^{\text{HR}} = \text{VGG}^L(\boldsymbol{x}^{\text{HR}}) \\ \boldsymbol{x}^{\text{SR}} = \text{G}_{\theta_{\text{SR}}}(\boldsymbol{x}^{\text{LR}}) \end{cases}$$

- Force images to have similar high level feature tensors.
- Supposed to be closer to perceptual similarity [Johnson et al., 2016].

SRGAN [Ledig et al., 2017]

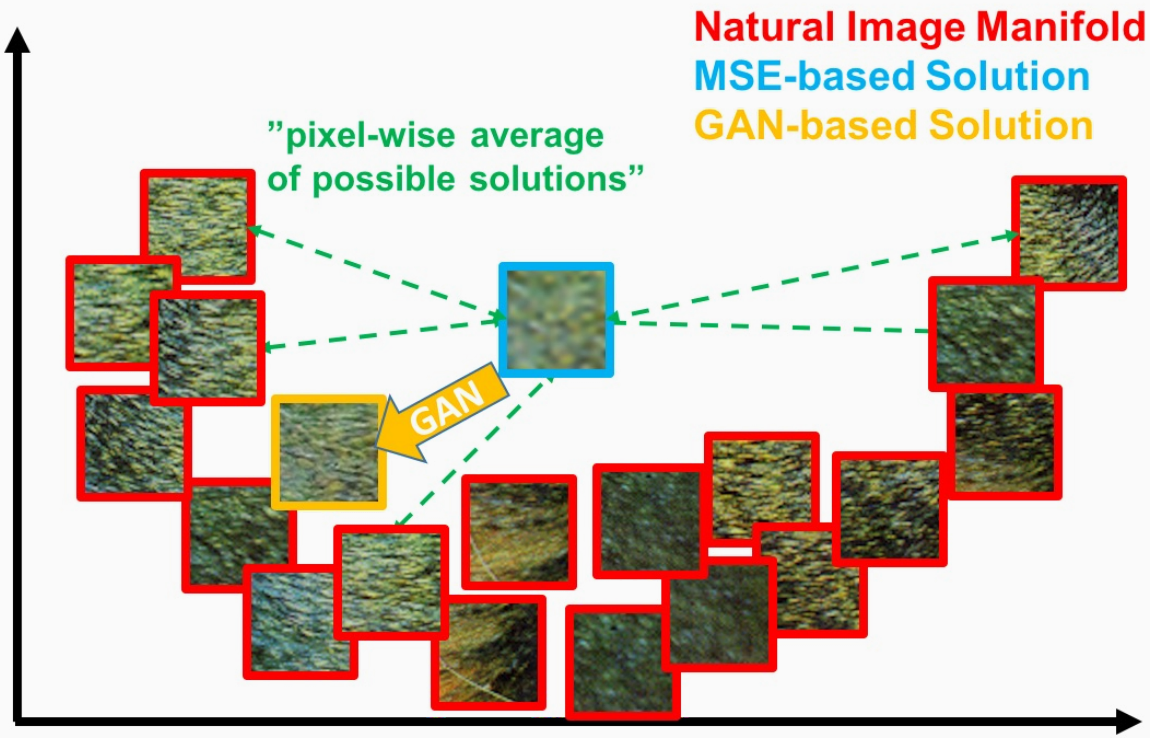


Architecture of Generator and Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer.

(source: From [Ledig et al., 2017])

Both networks are trained by alternating their gradient based updates.

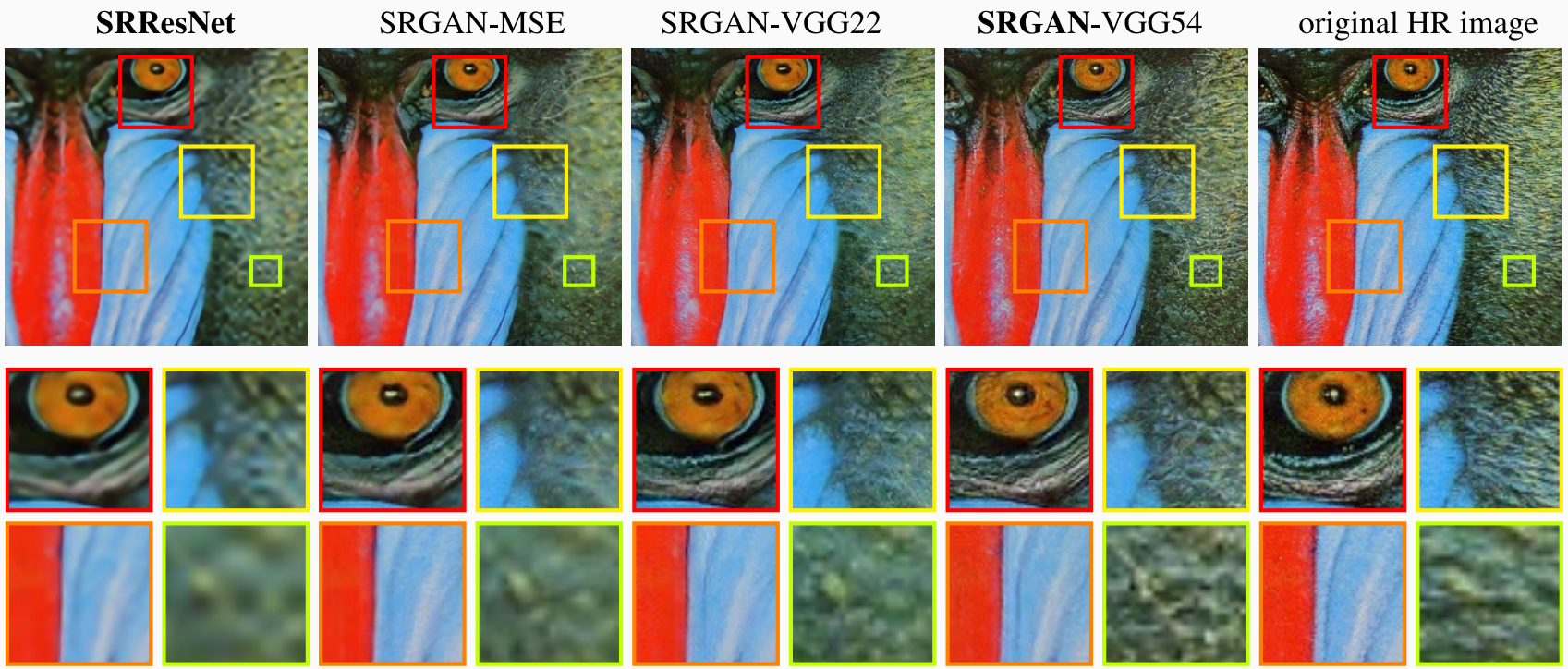
SRGAN [Ledig et al., 2017]



(source: From [Ledig et al., 2017])

- The SR problem is ill-posed → infinite number of solutions,
- MSE promotes a pixel-wise average of them → over-smooth,
- GAN drives reconstruction towards the "natural image manifold".

Super-resolution

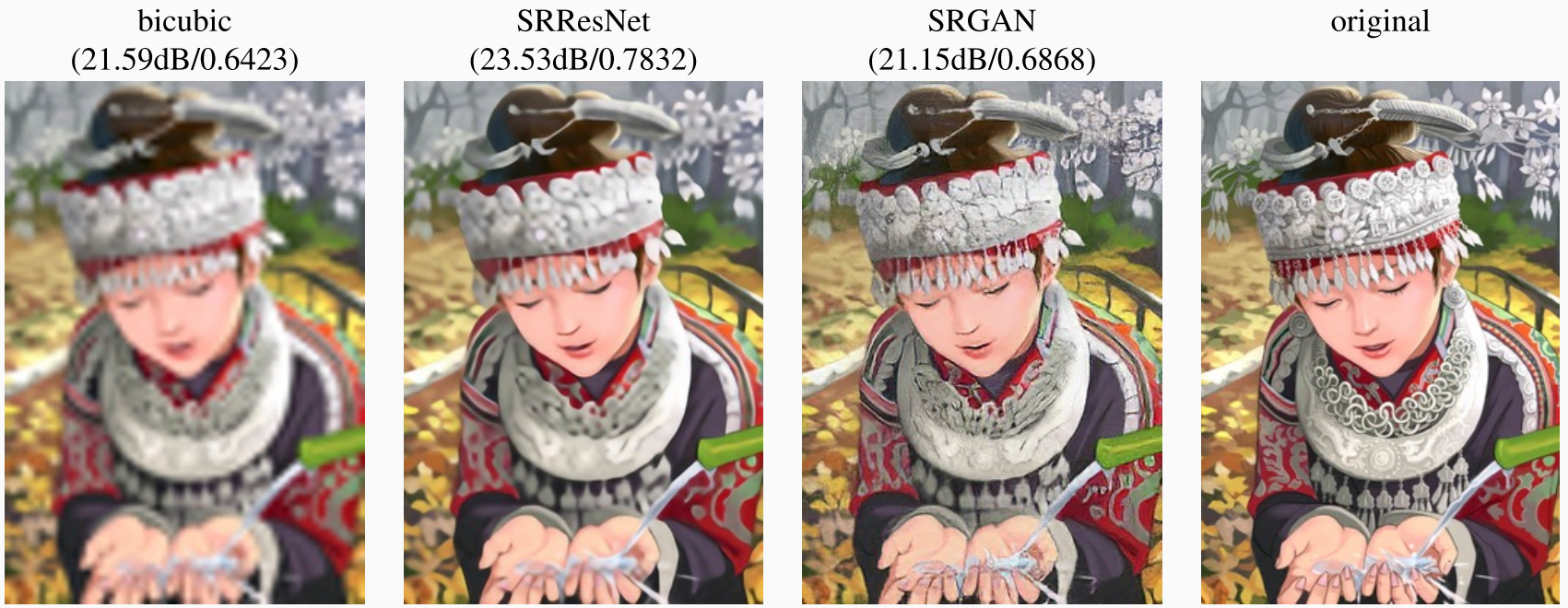


(source: From [Ledig et al., 2017])

×4 upsampling (16× more pixels)

- SRResNet: ResNet SR generator trained with MSE,
- SRGAN-MSE: generator and discriminator with MSE content loss,
- SRGAN-VGG22: generator and discriminator with VGG22 content loss,
- SRGAN-VGG54: generator and discriminator with VGG54 content loss.

Super-resolution



(source: From [Ledig et al., 2017])

×4 upsampling (16× more pixels)

- Even though some details are lost, they are replaced by "fake" but photo-realistic objects (instead of blurry ones).
- Remark that SRResNet is blurrier but achieves better PSNR.

Other approaches for generative modeling

To go further, there are different approaches for generative modeling:

- Variation Autoencoders (VAE)
  [Kingma and Welling, 2014, Kingma and Welling, 2019]
- Normalizing Flows [Dinh et al., 2017, Kingma and Dhariwal, 2018]
- Diffusion models [Song et al., 2021]

References

References i


Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. (2023).
Diffusion posterior sampling for general noisy inverse problems.
In *The Eleventh International Conference on Learning Representations*.


Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017).
Density estimation using real NVP.
In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.
OpenReview.net.


Dumoulin, V. and Visin, F. (2016).
A guide to convolution arithmetic for deep learning.
ArXiv e-prints.


Goodfellow, I. (2017).
Nips 2016 tutorial: Generative adversarial networks.

References ii


Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014).
Generative adversarial nets.
In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.


Ho, J., Jain, A., and Abbeel, P. (2020).
Denoising diffusion probabilistic models.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc.


Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017).
Image-to-image translation with conditional adversarial networks.
In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

References iii


Johnson, J., Alahi, A., and Fei-Fei, L. (2016).
Perceptual losses for real-time style transfer and super-resolution.
In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 694–711, Cham. Springer International Publishing.


Karras, T., Laine, S., and Aila, T. (2019).
A style-based generator architecture for generative adversarial networks.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.


Kingma, D. P. and Dhariwal, P. (2018).
Glow: Generative flow with invertible 1x1 convolutions.
In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

References iv


Kingma, D. P. and Welling, M. (2014).
Auto-encoding variational Bayes.
In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.


Kingma, D. P. and Welling, M. (2019).
An introduction to variational autoencoders.
Foundations and TrendsÂ® in Machine Learning, 12(4):307–392.


Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., and Shi, W. (2017).
Photo-realistic single image super-resolution using a generative adversarial network.
In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.


Radford, A., Metz, L., and Chintala, S. (2016).
Unsupervised representation learning with deep convolutional generative adversarial networks.
In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.


Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022).
High-resolution image synthesis with latent diffusion models.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695.


Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., and Aberman, K. (2023).
Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22500–22510.


Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. (2022).
Photorealistic text-to-image diffusion models with deep language understanding.


Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015).
Deep unsupervised learning using nonequilibrium thermodynamics.
In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France. PMLR.


Song, J., Vahdat, A., Mardani, M., and Kautz, J. (2023).
Pseudoinverse-guided diffusion models for inverse problems.
In *International Conference on Learning Representations*.


Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021).
Score-based generative modeling through stochastic differential equations.
In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net.


Zhang, L., Rao, A., and Agrawala, M. (2023).
Adding conditional control to text-to-image diffusion models.
In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3836–3847.

Questions?

(Most) Slides from Charles Deledalle

Sources, images courtesy and acknowledgment

K. Chatfield
P. Gallinari,
C. Hazırbaş
A. Horodniceanu
Y. LeCun
V. Lepetit
L. Masuch
A. Ng
M. Ranzato