Deep Learning

Generative models

Danijel Skočaj
University of Ljubljana
Faculty of Computer and Information Science

Academic year: 2022/23

Discriminative vs. Generative models

- Discriminative models

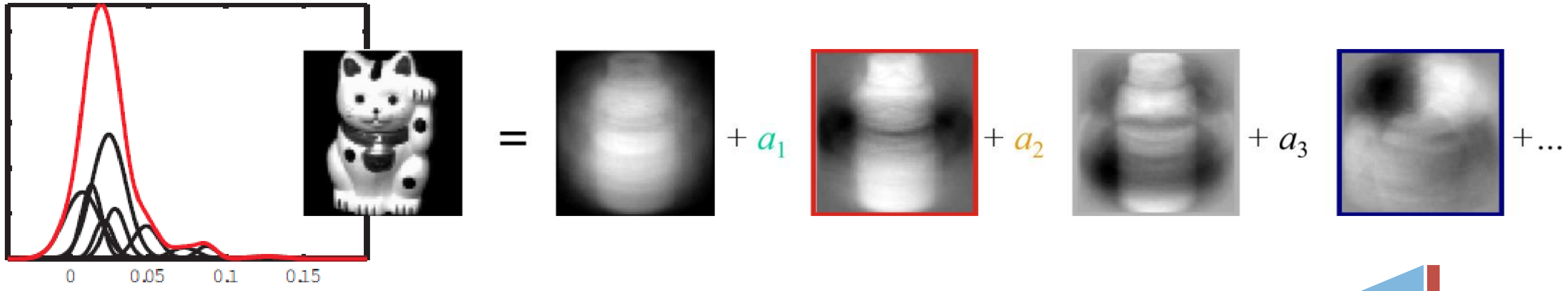
discriminative model
model, typically built with supervised learning, that models the conditional probability distribution of the target predictive value given the input instance, for example by finding a decision boundary between different classes, it is also used for classification or regression. $P(y|x)$

- Generative models

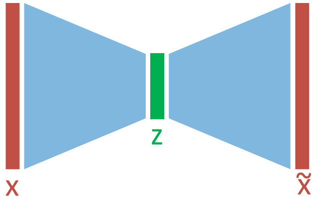
generative model
model built with machine learning that models the distribution of training examples, thereby predicting the probability of occurrence for each individual sample, it is also used for generating new samples similar to the training examples. $P(x), P(x,y)$

Generative models

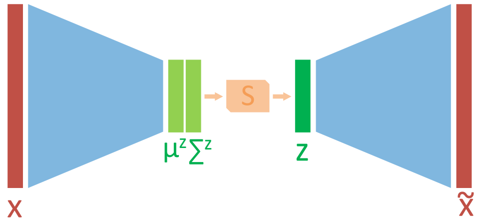
- Simple models
  - GMM, PCA



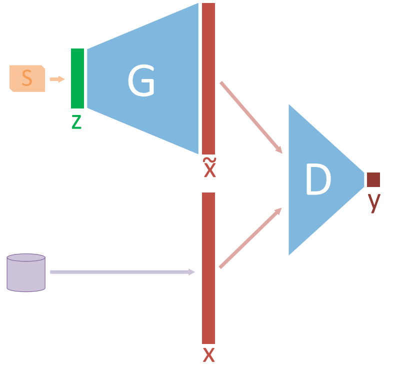
- Autoencoders



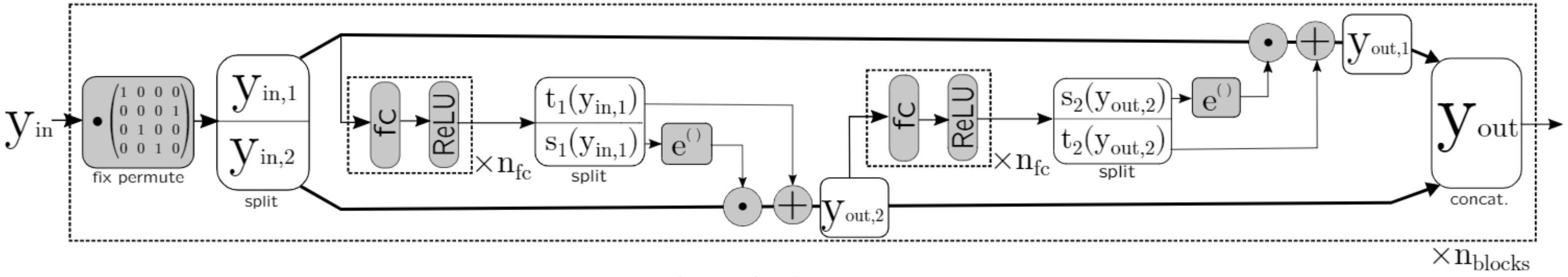
- Variational Autoencoders



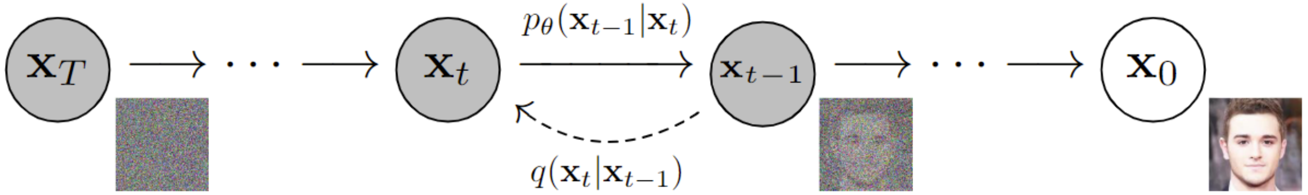
- Generative Adversarial Networks



- Normalizing Flows



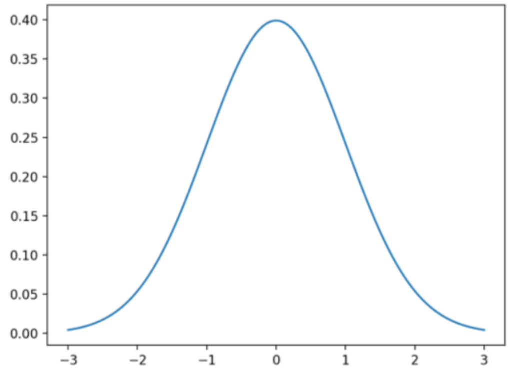
- Diffusion Models



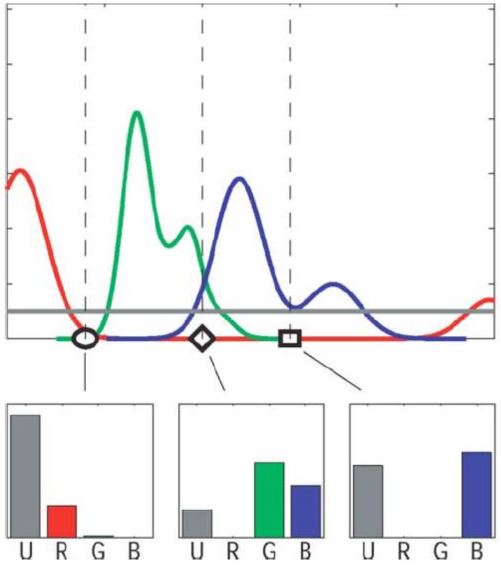
Gaussian, GMM

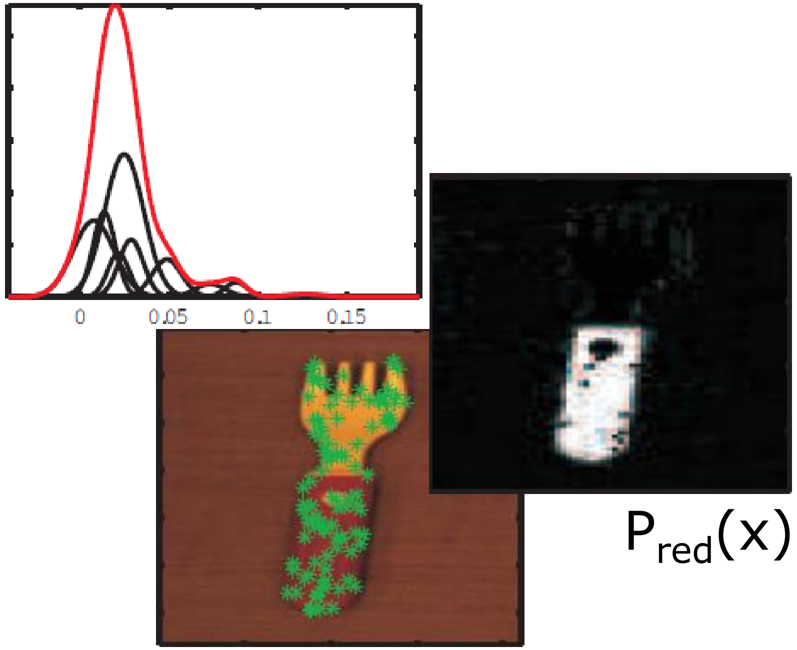
- Normal distribution

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



- Gaussian mixture models

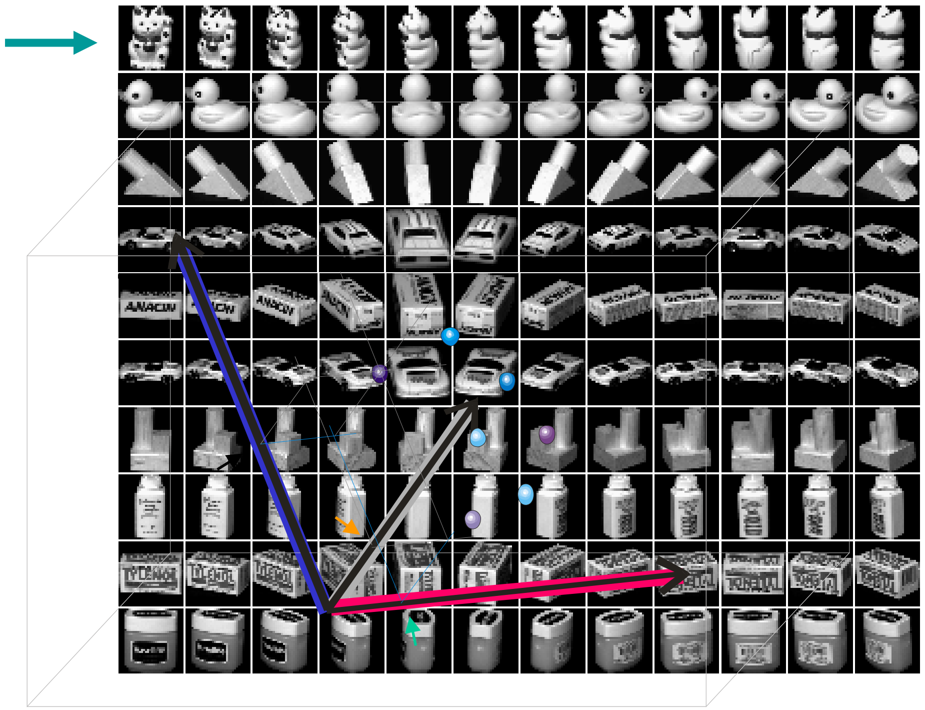


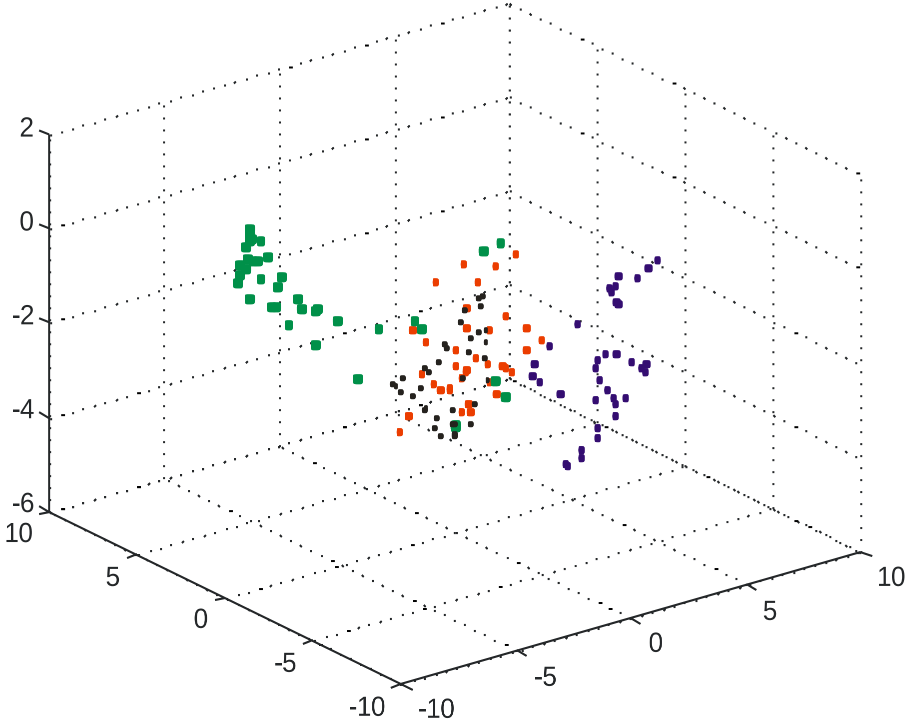


Wyatt et al., 2009

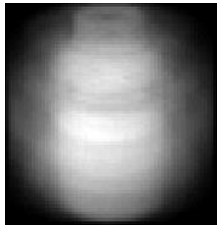
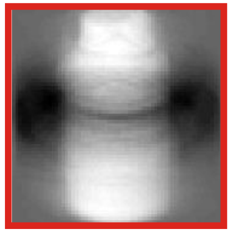
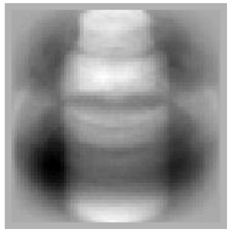
Kristan et al., 2009

Principal Component Analysis





- Features: PCA coefficients

 =  + a_1  + a_2  + a_3  + ...

Skočaj, 2003

PCA algorithm

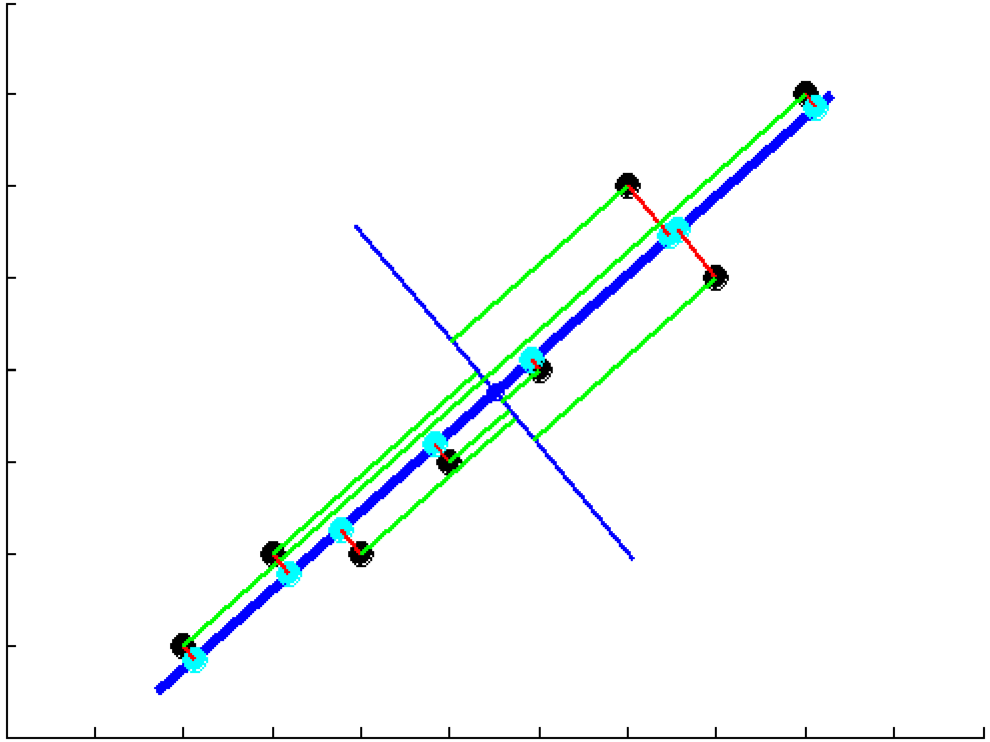
Input: data matrix $\mathbf{X}$

Output: mean value $\boldsymbol{\mu}$, eigenvectors $\mathbf{U}$, eigenvalues $\boldsymbol{\lambda}$.

1. Estimate the mean vector: $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$.
2. Center the input data around the mean: $\hat{\mathbf{X}} = \mathbf{X} - \boldsymbol{\mu}\mathbf{1}_{1\times N}$.
3. **if** $M \leq N$ **then**
4. Estimate the covariance matrix : $\mathbf{C} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^\top$.
5. Perform SVD on $\mathbf{C}$. Obtain eigenvectors $\mathbf{U}$ and eigenvalues $\boldsymbol{\lambda}$.
6. **else**
7. Estimate the inner product matrix: $\mathbf{C}' = \frac{1}{N}\hat{\mathbf{X}}^\top\hat{\mathbf{X}}$.
8. Perform SVD on $\mathbf{C}'$. Obtain eigenvectors $\mathbf{U}'$ and eigenvalues $\boldsymbol{\lambda}'$.
9. Determine the eigenvectors $\mathbf{U}$: $\mathbf{u}_i = \frac{\hat{\mathbf{X}}\mathbf{u}'_i}{\sqrt{N\lambda'_i}}\ ,\ i = 1\ldots N$.
10. Determine the eigenvalues $\boldsymbol{\lambda} = \boldsymbol{\lambda}'$.
11. **end if**

Projection and reconstruction

- A subset of principal components suffices for a good approximation of the input data.
  - Use only k, k<<N principal axes



- Projection: $\mathbf{U}^\top : \mathbb{R}^M \rightarrow \mathbb{R}^k$

$$\mathbf{a} \;=\; \mathbf{U}^\top \hat{\mathbf{x}} = \mathbf{U}^\top (\mathbf{x} - \boldsymbol{\mu})$$

$$a_j \;=\; \langle \hat{\mathbf{x}}, \mathbf{u}_j \rangle = \sum_{i=1}^{M} u_{ij} \hat{x}_i = \sum_{i=1}^{M} u_{ij} (x_i - \mu_i) \;\;,\;\; j = 1 \ldots k$$

- Reconstruction: $\mathbf{U} : \mathbb{R}^k \rightarrow \mathbb{R}^M$

$$\hat{\mathbf{y}} \;=\; \mathbf{U}\mathbf{a} = \sum_{j=1}^{k} a_j \mathbf{u}_j \qquad\qquad \mathbf{y} \;=\; \hat{\mathbf{y}} + \boldsymbol{\mu}$$

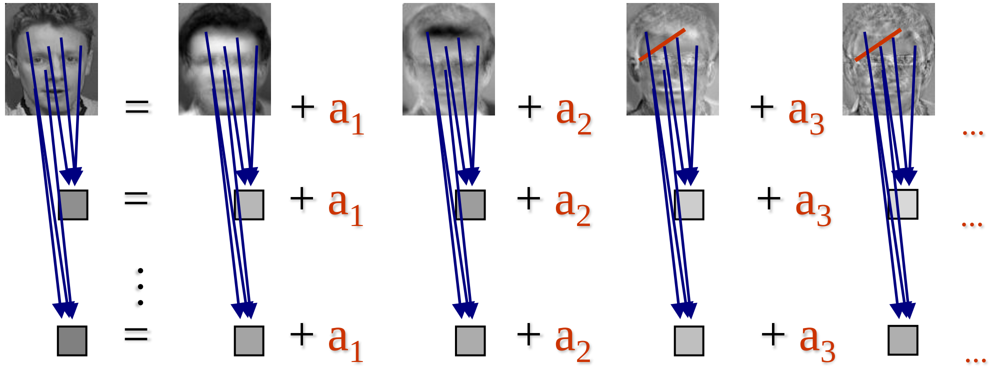
- PCA minimizes the squared reconstruction error:

$$\mathbf{e} = \hat{\mathbf{x}} - \hat{\mathbf{y}} = \sum_{i=1}^{N} a_i \mathbf{u}_i - \sum_{i=1}^{k} a_i \mathbf{u}_i = \sum_{i=k+1}^{N} a_i \mathbf{u}_i \qquad\qquad e = \|\mathbf{e}\|^2 = \left\| \sum_{i=k+1}^{N} a_i \mathbf{u}_i \right\|^2 = \sum_{i=k+1}^{N} a_i^2$$

Robust projection

- Reconstruction error: $e = \sum_{i=1}^{M} \sum_{j=1}^{N} \left(\hat{x}_{ij} - \sum_{l=1}^{k} u_{il} a_{lj} \right)^2$
- Instead of projection solve a system of linear equations!

$$
\begin{aligned}
x^1 &= a_1 u_1^1 + a_2 u_2^1 + \cdots + a_k u_k^1 \\
x^2 &= a_1 u_1^2 + a_2 u_2^2 + \cdots + a_k u_k^2 \\
&\vdots \\
x^m &= a_1 u_1^m + a_2 u_2^m + \cdots + a_k u_k^m
\end{aligned}
$$

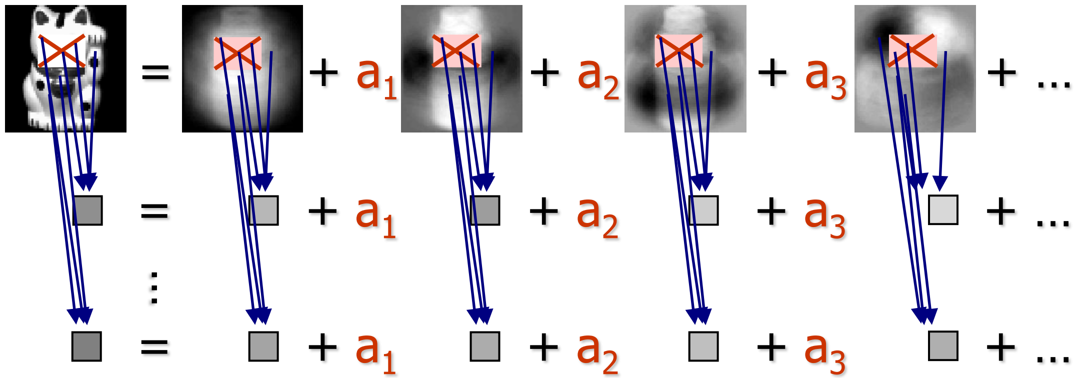


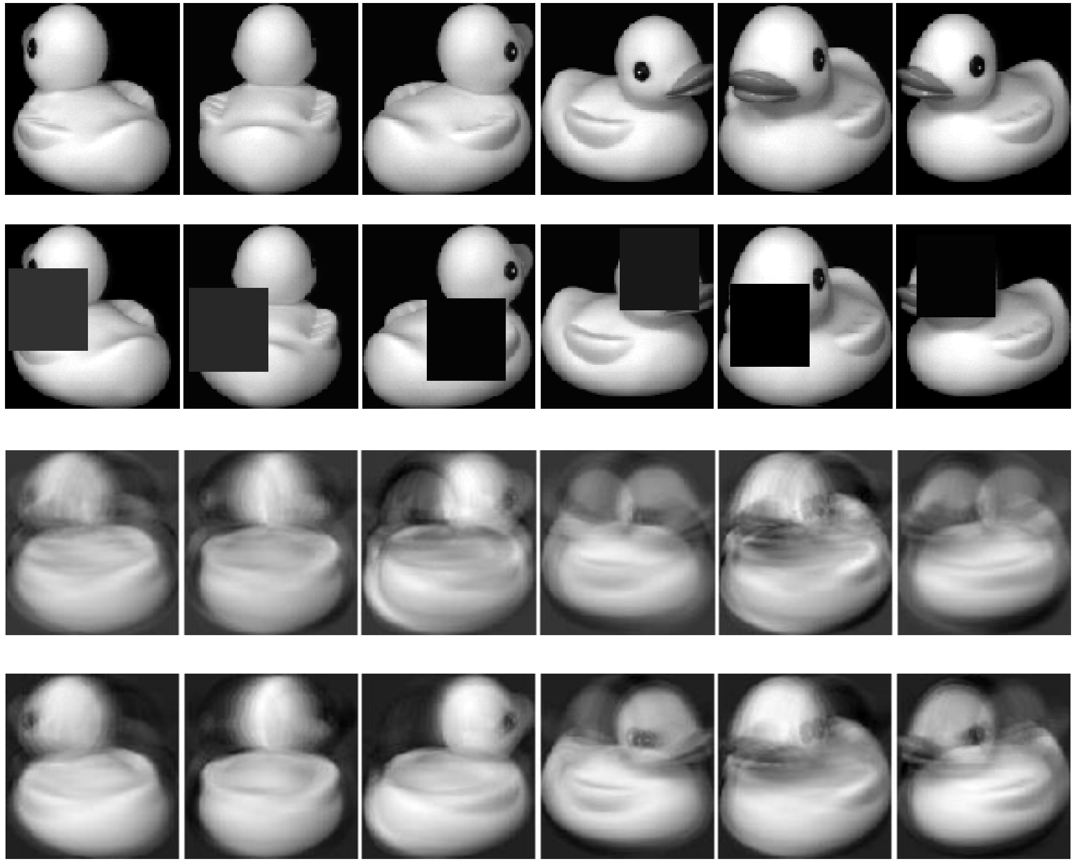
- Only a subset of pixels can be used
  - Projection in the presence of missing data!

Leonardis & Bischof, 2000

Robust projection

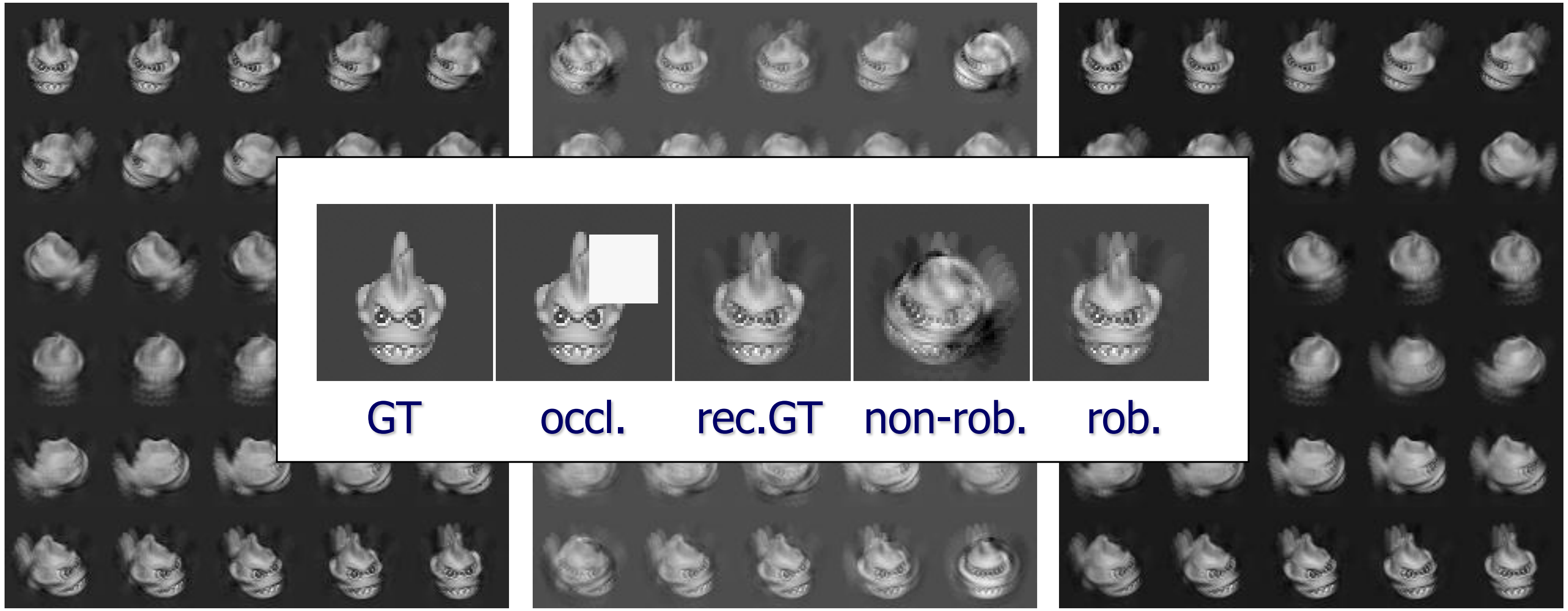
- Instead of using the standard approach for projection:
  - Select a subset of pixels
  - Find a robust solution of equations.
  - Perform multiple hypotheses.





  - Hypothesize-and-test paradigm
  - Competing hypotheses are subject to a selection procedure based on the MDL principle.
- Robust algorithm is able to reconstruct missing/corrupted pixels

Robust projection



reconstructed
non-occluded images

training images

non-robustly
reconstructed
occluded images

occluded images

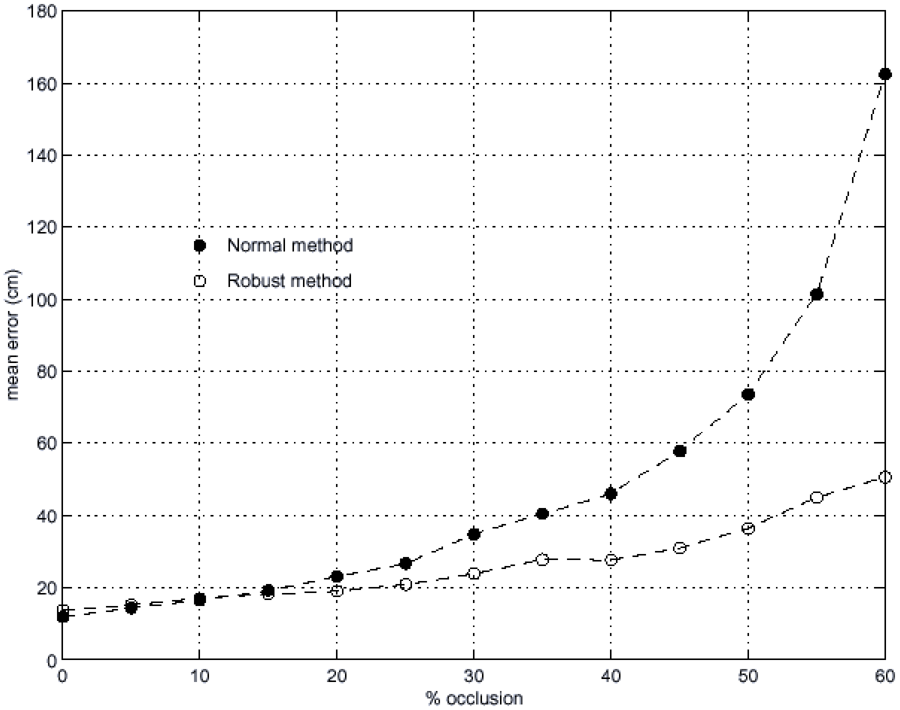
robustly
reconstructed
occluded images

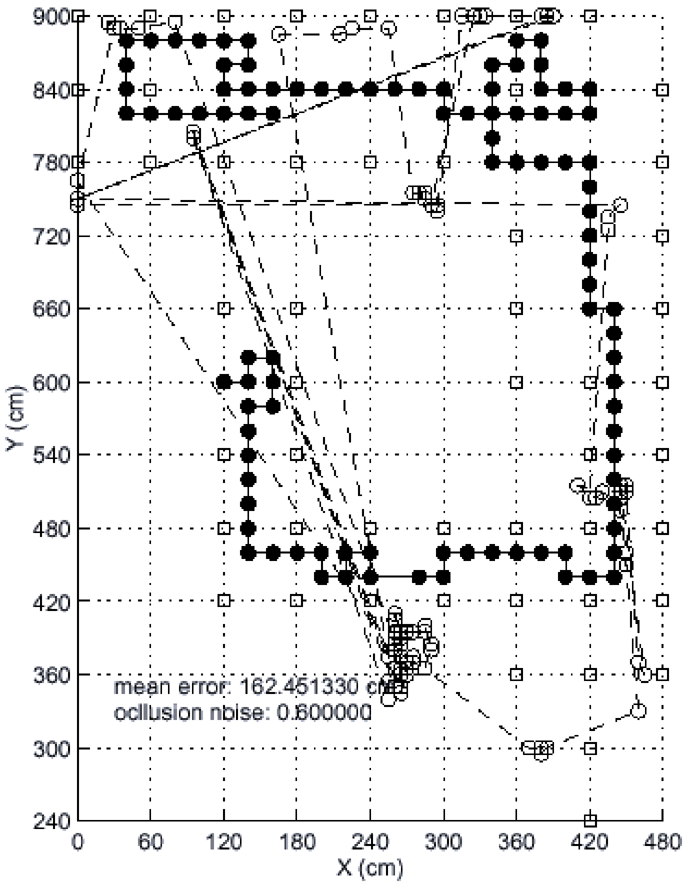
Fidler et al., 2006

Robust projection

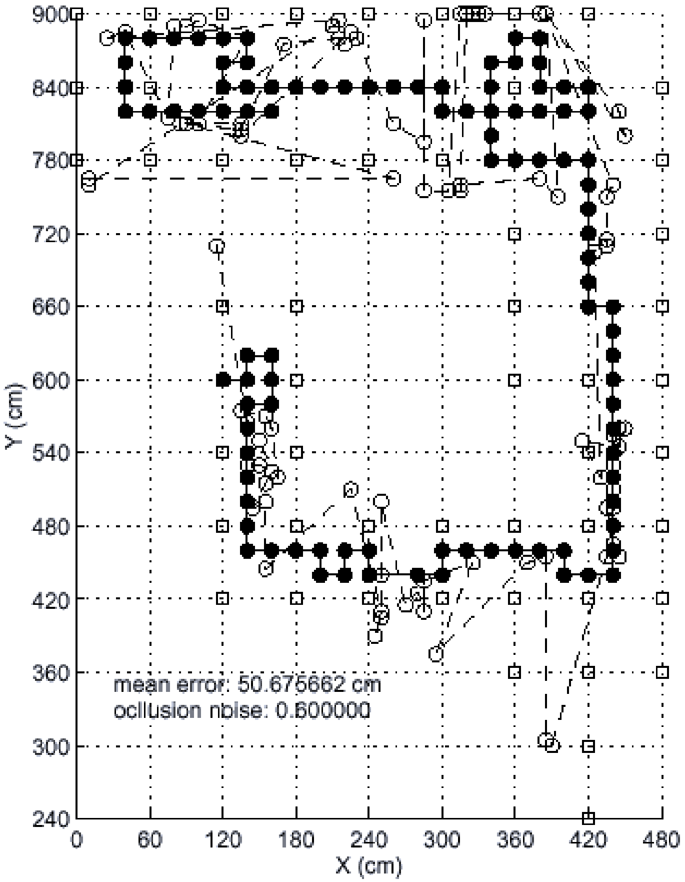
- Appearance-based localisation

[Jogan & Leonardis, 2003]



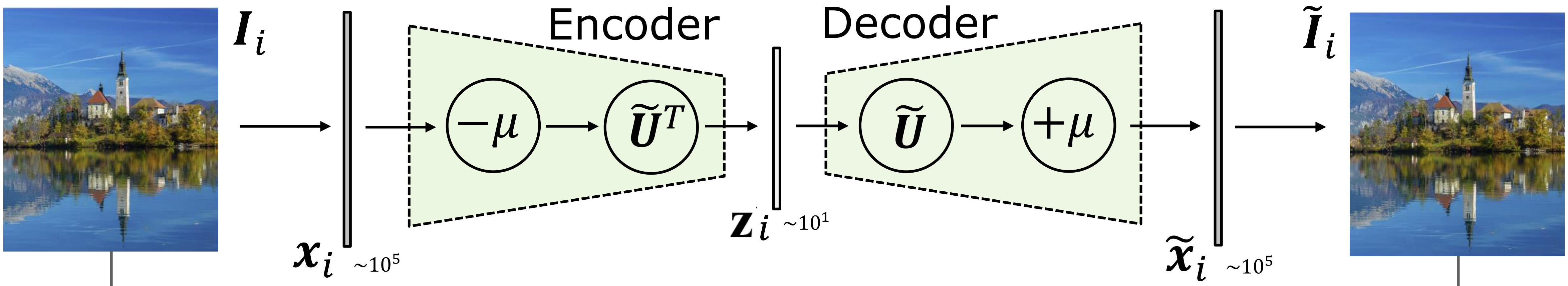


standard



robust

PCA is a linear autoencoder



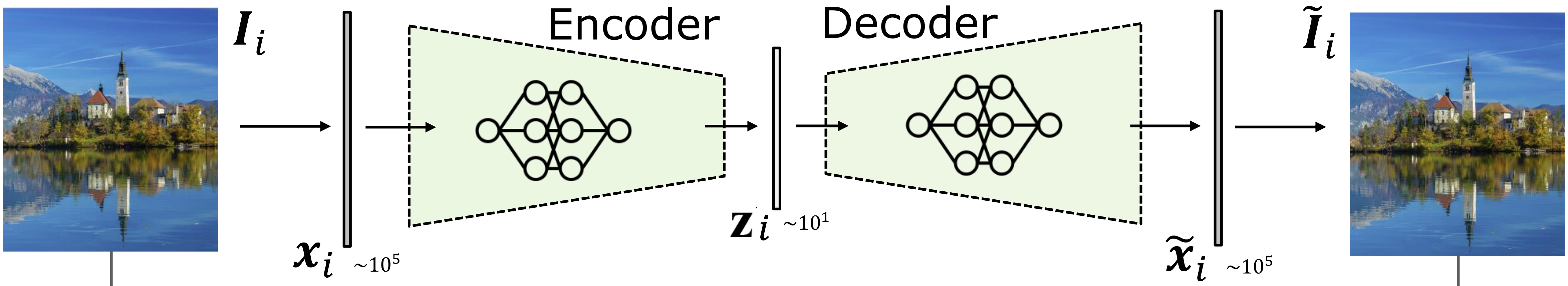
$$\mathcal{L}(\mu, \widetilde{U}) = \left\| \boldsymbol{I}_i - \tilde{\boldsymbol{I}}_i \right\|^2$$

$$\mu, \widetilde{U} = \underset{\mu^*, \widetilde{U}^*}{\operatorname{argmin}} \mathcal{L}(\mu^*, \widetilde{U}^*)$$

$$\mathbf{z} \;=\; \mathrm{U}^\top \hat{\mathbf{x}} = \mathrm{U}^\top (\mathbf{x} - \boldsymbol{\mu})$$

$$\widetilde{\boldsymbol{x}} \;=\; \mathrm{U}\mathbf{z} = \sum_{j=1}^{k} \mathrm{z}_j \mathbf{u}_j$$

Autoencoder

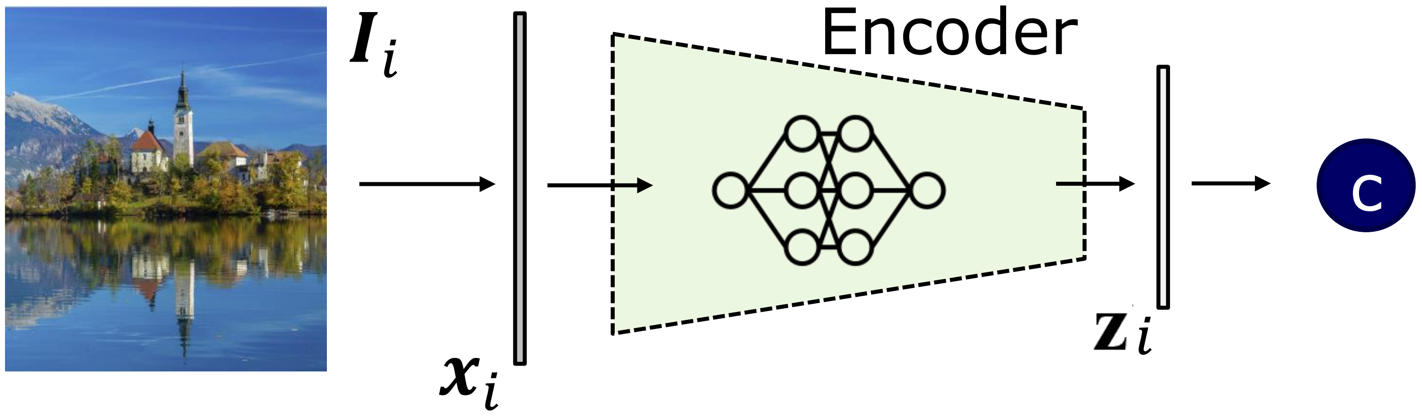


$$\mathcal{L}(\mu, \widetilde{U}) = \left\| \boldsymbol{I}_i - \tilde{\boldsymbol{I}}_i \right\|^2$$

$$\mu, \widetilde{U} = \underset{\mu^*, \widetilde{U}^*}{\operatorname{argmin}} \mathcal{L}(\mu^*, \widetilde{U}^*)$$

- $\boldsymbol{z}$ - Latent representation
- Self-supervised learning – reconstruction loss with no labels required

Autoencoder



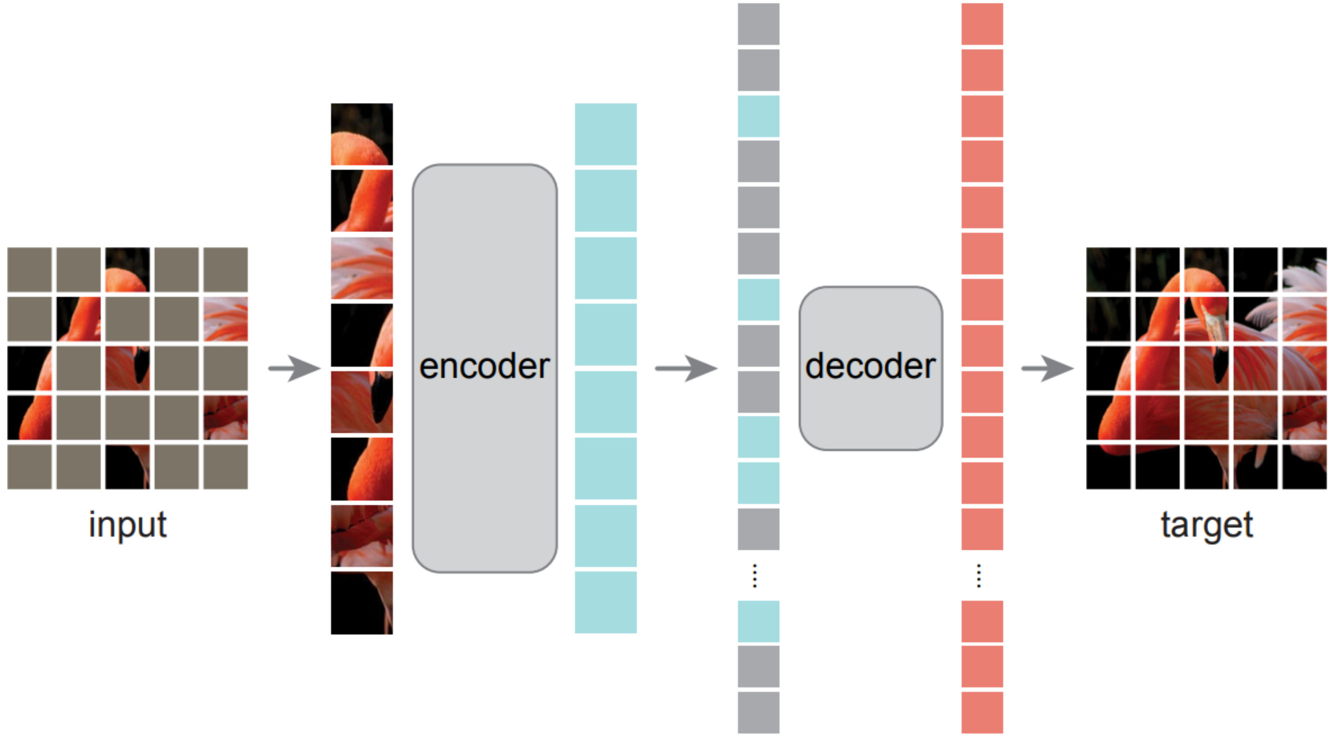
Fine-tune encoder on a down-stream task
(e.g., classification)

- z - Latent representation
- Self-supervised learning – reconstruction loss with no labels required
- Self-supervised pretraining and supervised fine-tunning

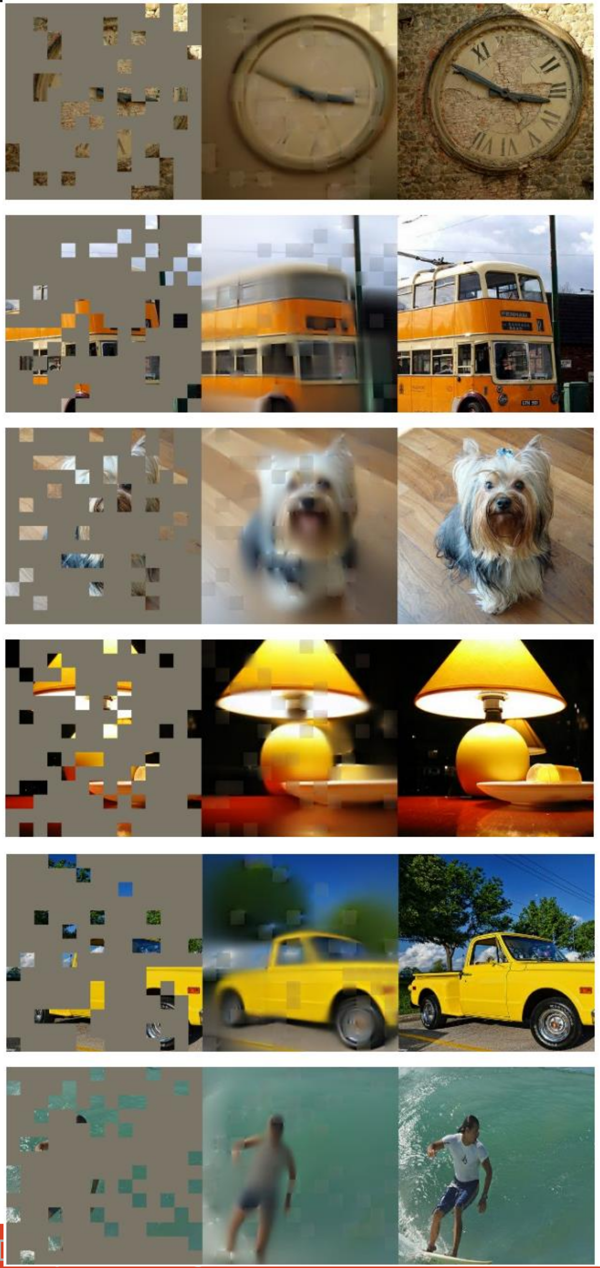
Masked autoencoders

- Masked Autoencoders Are Scalable Vision Learners
- Self-supervised learning of representtaions (pre-training)
- Scalable architecture for vision learning tasks
- Autoregressive modelling
- Improves downstream tasks
  - classification
  - detection
  - segmentation
- Partial fine-tunning
- Encoder
  - on unmasked patches only
- Decoder
  - on full set of tokens
  - Lightweight
- MSE on masked patches

He et al., 2022

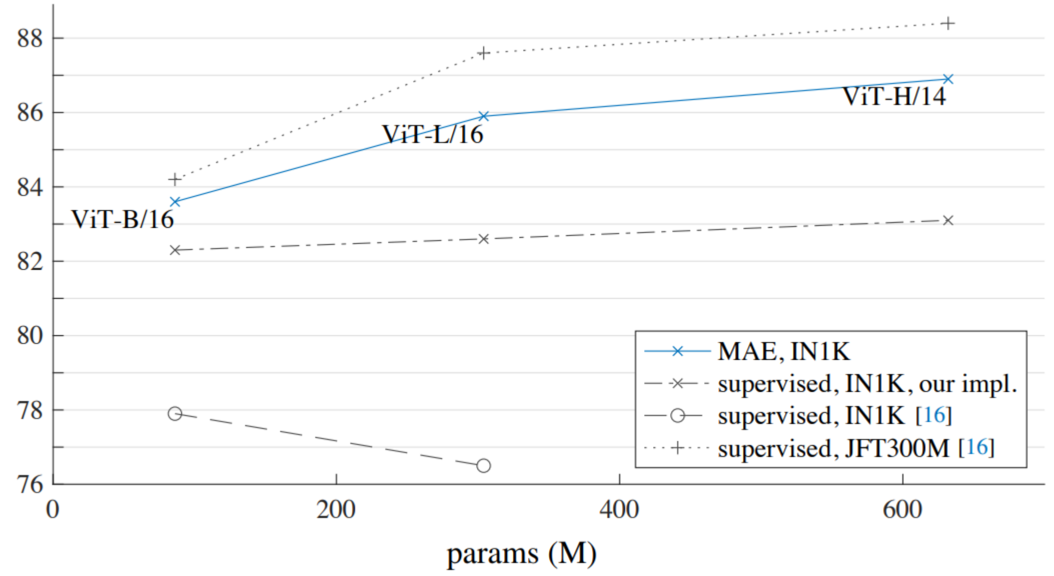


MAE results



original mask 75% mask 85% mask 95%

MAE results



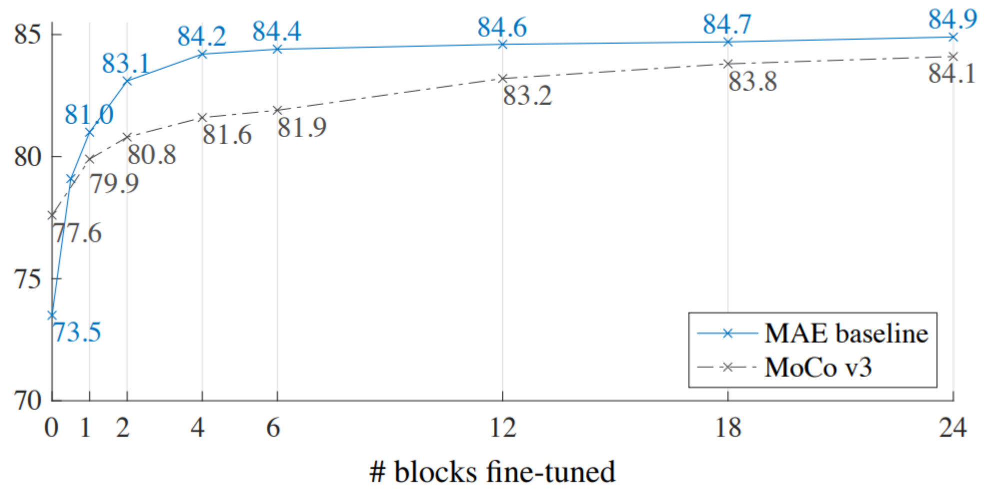
ImageNet1k classification

method	pre-train data	AP^box ViT-B	AP^box ViT-L	AP^mask ViT-B	AP^mask ViT-L
supervised	IN1K w/ labels	47.9	49.3	42.9	43.9
MoCo v3	IN1K	47.9	49.3	42.7	44.0
BEiT	IN1K+DALLE	49.8	**53.3**	44.4	47.1
MAE	IN1K	**50.3**	**53.3**	**44.9**	**47.2**

COCO object detection and segmentation

method	pre-train data	ViT-B	ViT-L
supervised	IN1K w/ labels	47.4	49.9
MoCo v3	IN1K	47.3	49.1
BEiT	IN1K+DALLE	47.1	53.3
MAE	IN1K	**48.1**	**53.6**

ADE20K semantic segmentation



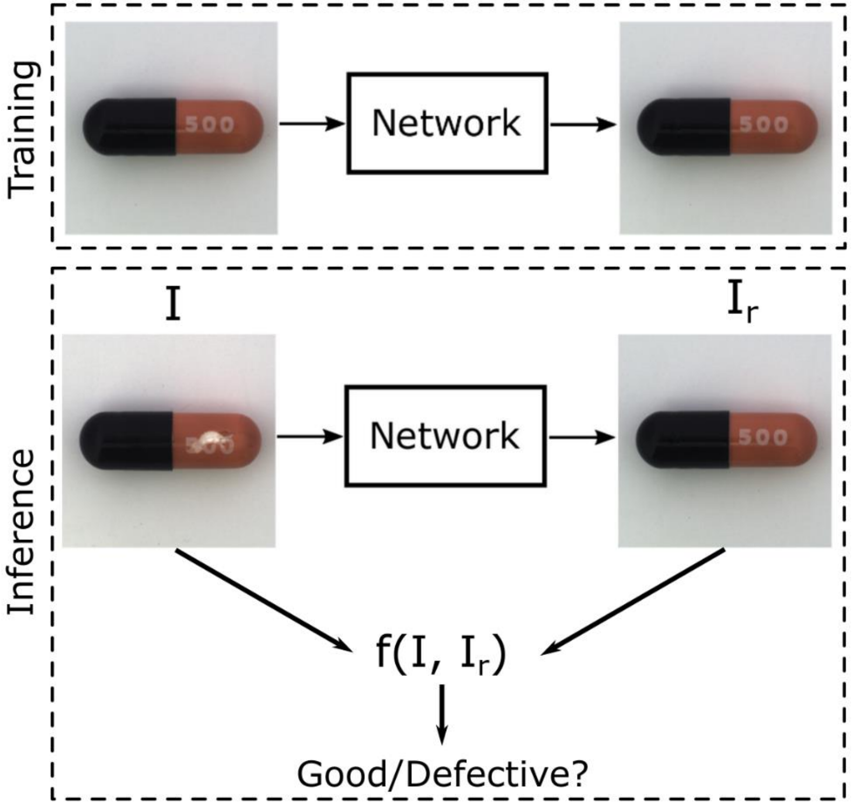
Partial
fine-tunning

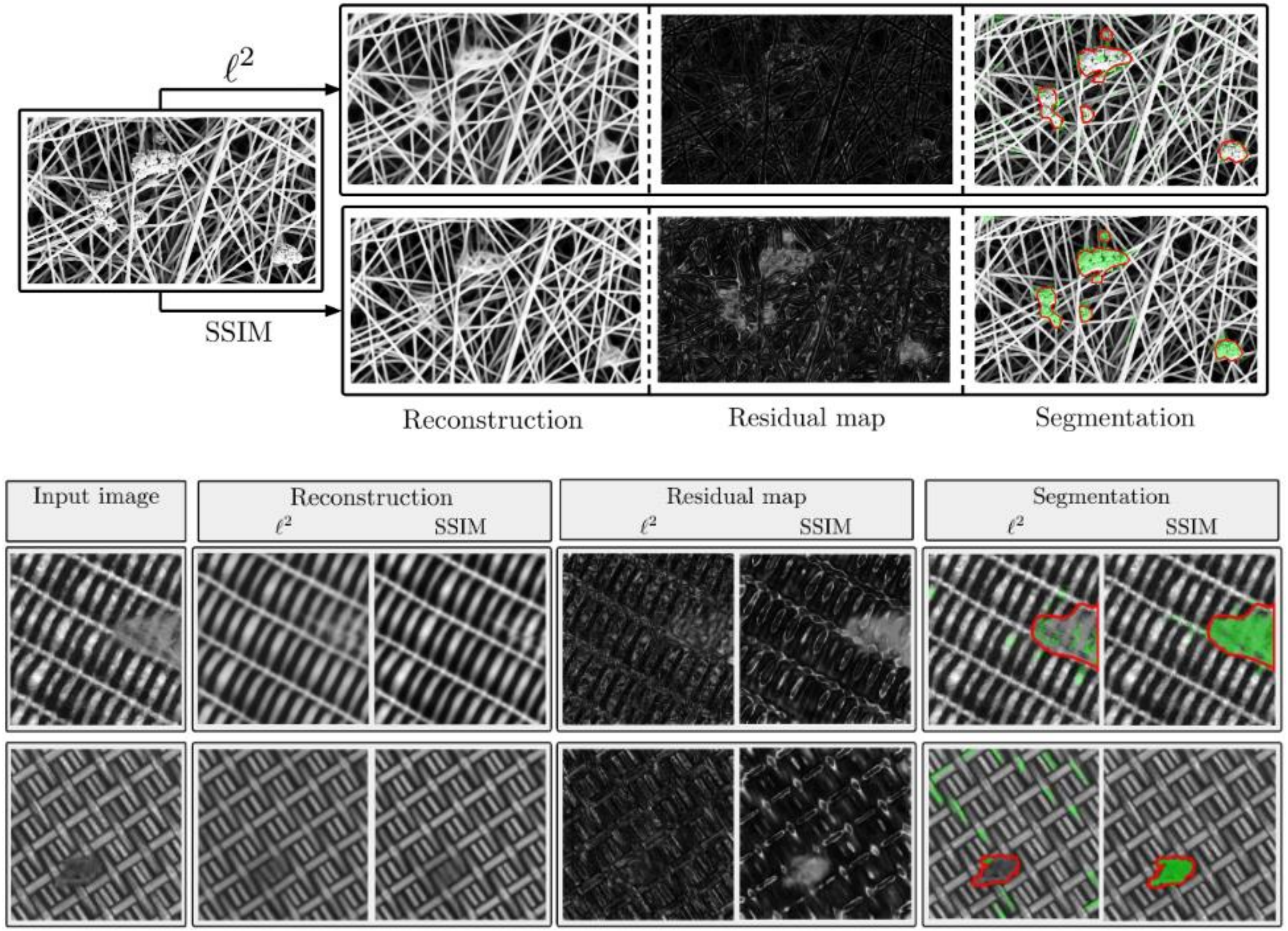
He et al., 2022

Autoencoders for anomaly detection

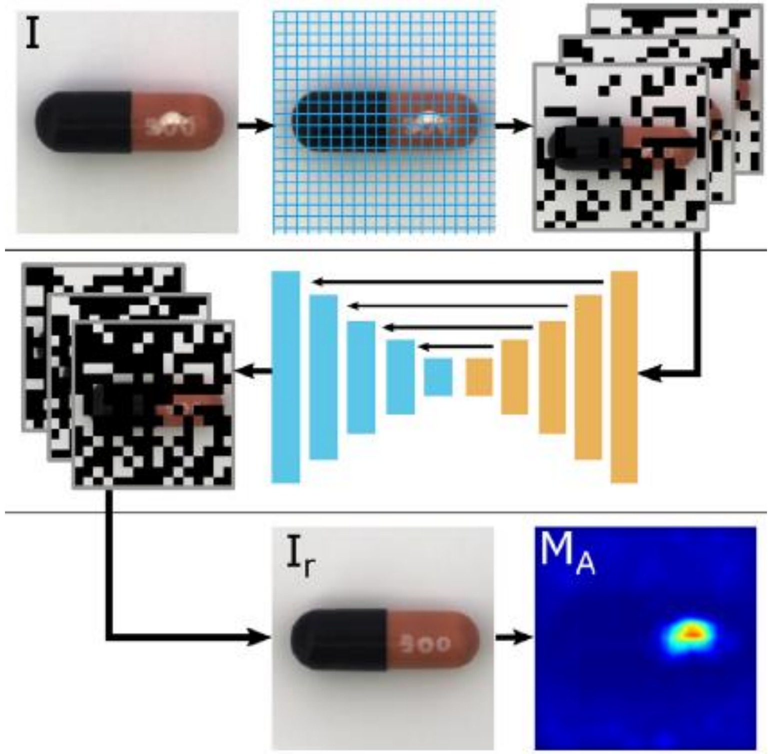
- Train only on good samples
- Unable to reconstruct anomalies
- Check the reconstruction error

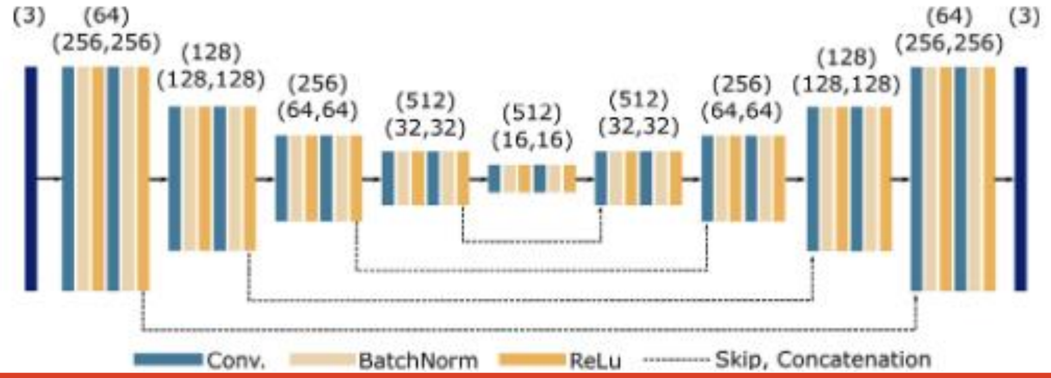
[Bergman et al., 2018](#)

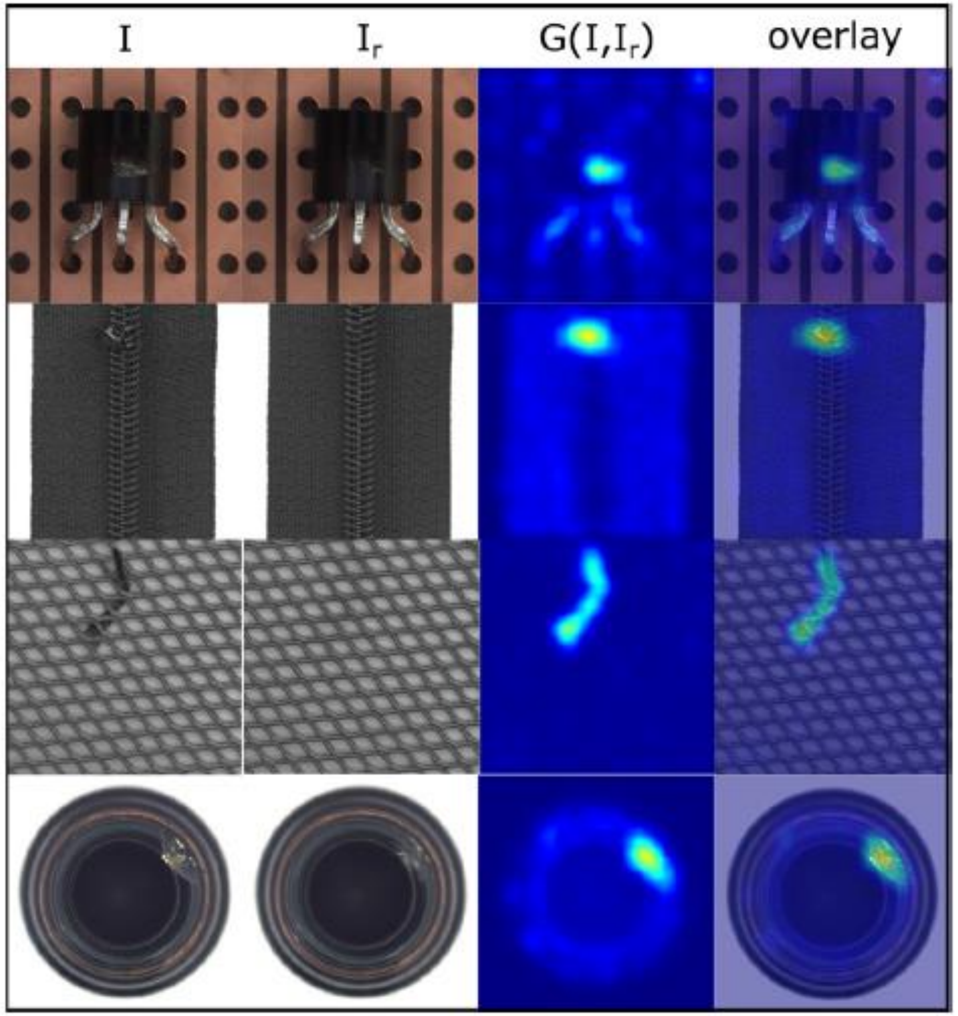




Autoencoders for inpainting and anomaly detection

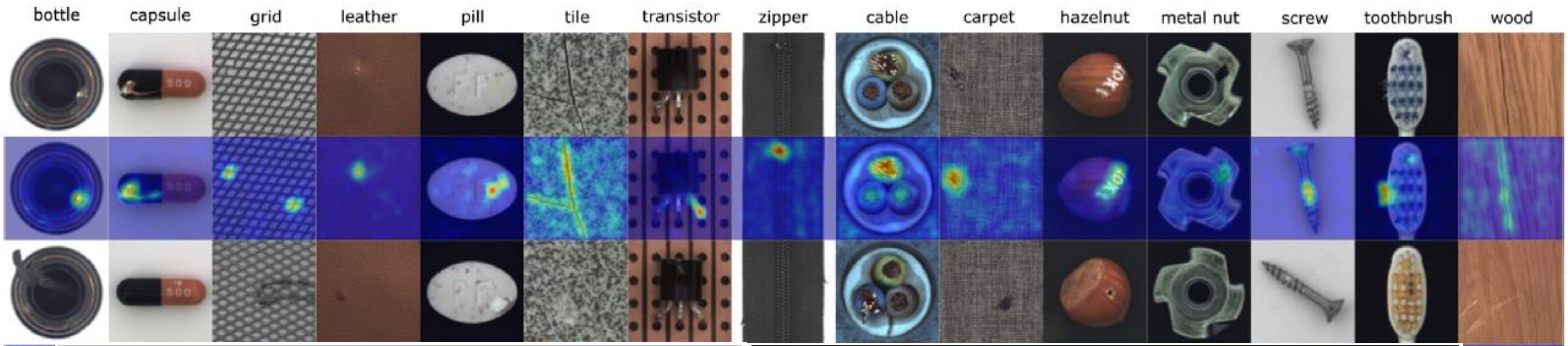






[Zavrtanik et al., 2021]

Autoencoders for inpainting and anomaly detection



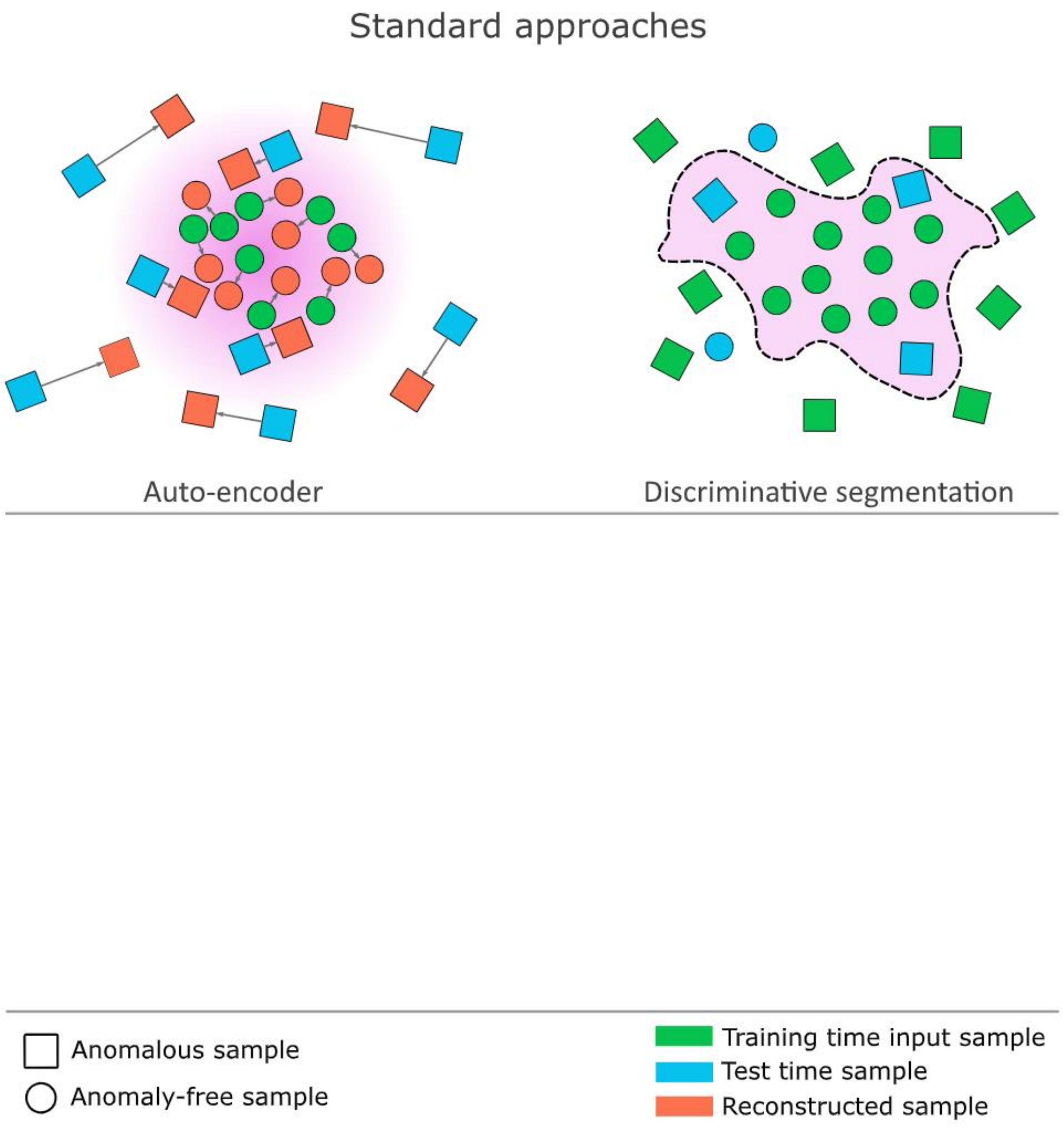
Class	GeoTrans [6]	GANomaly [1]	ITAE [4]	US [16]	RIAD
bottle	74.4	89.2	94.1	99.0	**99.9**
capsule	67.0	73.2	68.1	86.1	**88.4**
grid	61.9	70.8	88.3	81.0	**99.6**
leather	84.1	84.2	86.2	88.2	**100**
pill	63.0	74.3	78.6	**87.9**	83.8
tile	41.7	79.4	73.5	**99.1**	98.7
transistor	86.9	79.2	84.3	81.8	**90.9**
zipper	82.0	74.5	87.6	91.9	**98.1**
cable	78.3	75.7	83.2	**86.2**	81.9
carpet	43.7	69.9	70.6	**91.6**	84.2
hazelnut	35.9	78.5	85.5	**93.1**	83.3
metal nut	81.3	70.0	66.7	82.0	**88.5**
screw	50.0	74.6	**100**	54.9	84.5
toothbrush	97.2	65.3	**100**	95.3	**100**
wood	61.1	83.4	92.3	**97.7**	93.0
avg_{tex}	58.5	76.5	82.2	91.5	**95.1**
avg_{obj}	71.6	75.4	84.8	85.8	**89.9**
avg	67.2	76.2	83.9	87.7	**91.7**

Class	AE-SSIM [34]	AnoGAN [11]	VEVAE [35]	US [16]	RIAD
bottle	93.0	86.0	87.0	97.8	**98.4**
capsule	94.0	84.0	74.0	**96.8**	92.8
grid	94.0	58.0	73.0	89.9	**98.8**
leather	78.0	64.0	95.0	97.8	**99.4**
pill	91.0	87.0	83.0	**96.5**	95.7
tile	59.0	50.0	80.0	**92.5**	89.1
transistor	80.0	80.0	**93.0**	73.7	87.7
zipper	88.0	78.0	78.0	95.6	**97.8**
cable	82.0	78.0	90.0	**91.9**	84.2
carpet	87.0	54.0	78.0	93.5	**96.3**
hazelnut	97.0	87.0	98.0	**98.2**	96.1
metal nut	89.0	76.0	94.0	**97.2**	92.5
screw	96.0	80.0	97.0	97.4	**98.8**
toothbrush	92.0	90.0	94.0	97.9	**98.9**
wood	73.0	62.0	77.0	**92.1**	85.8
avg_{tex}	78.2	57.7	80.6	93.2	**93.9**
avg_{obj}	90.2	82.6	88.8	**94.3**	**94.3**
avg	86.2	74.3	86.1	93.9	**94.2**

Generative vs. discriminative approaches

- Generative models
  - Good approximation of data
  - Unsupervised learning
  - General, task-independent
  - Not very compact, redundant representations (for a particular task)
  - ⇨ Enable reconstruction and outlier detection
- Discriminative models
  - Supervised learning
  - Task-dependent
  - Compact representations
  - No reconstruction (low dimensional projections)
  - ⇨ Do not enable reconstruction and detection of outliers

⇨ Combine reconstructive model and discriminative classifier

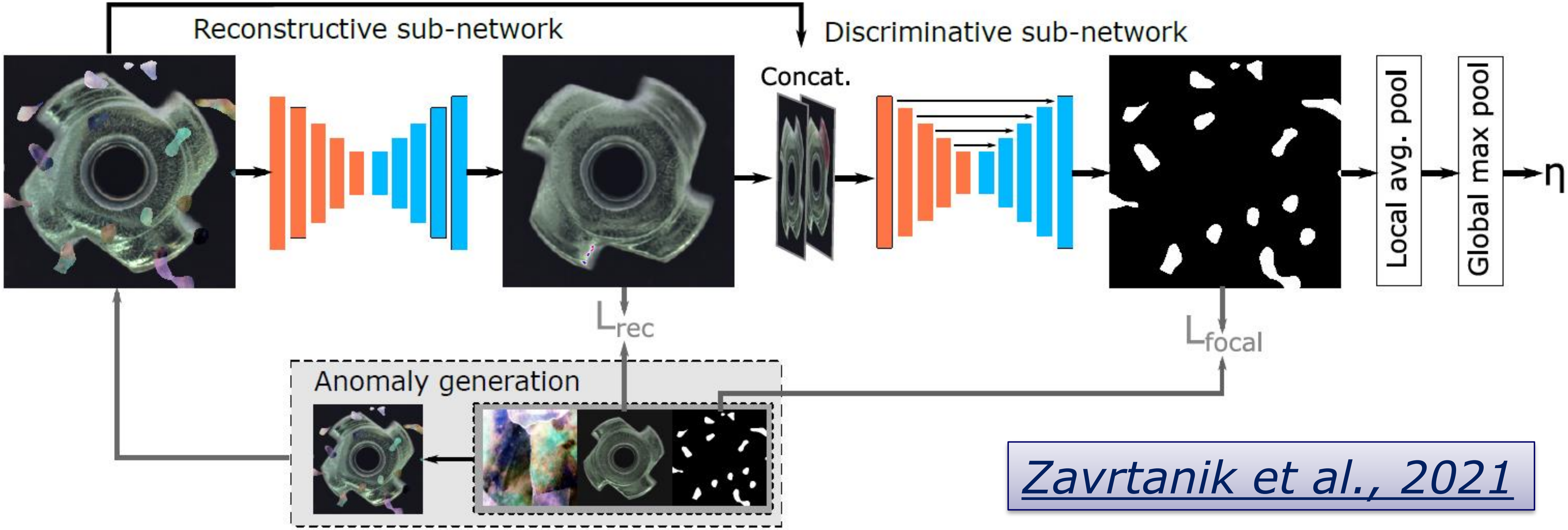


Fidler et al., 2006

Zavrtanik et al., 2021

Combining reconstruction and discrimination

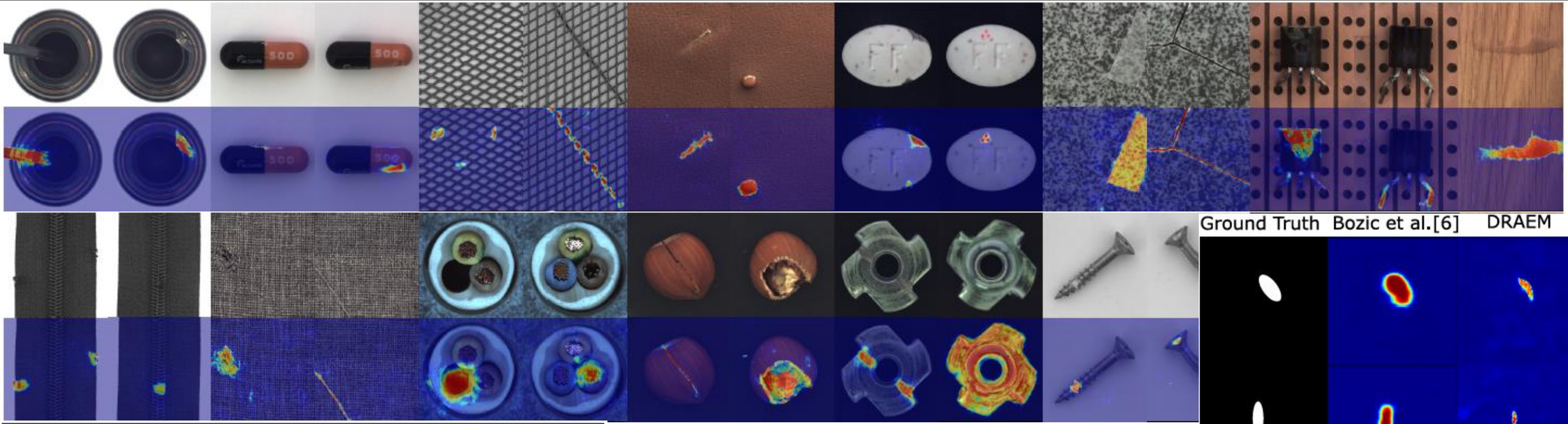
- Combining reconstruction and discrimination improves results



[Zavrtanik et al., 2021]

	Architecture		Anomaly Generation						Results	
Method	Recon. Net.	Discr. Net.	Augmentation	β	ImageNet	DTD	Perlin	Rectangle	Det.	Loc.
Disc.		✓	✓	✓		✓	✓		93.9	92.7 / 62.5
Recon.-AE	✓		✓	✓		✓	✓		83.9	89.7 / 47.5
Recon.-AE$_{MSGMS}$	✓		✓	✓		✓	✓		90.7	93.4 / 50.9
DRÆM	✓	✓	✓	✓		✓	✓		**98.0**	**97.3 / 68.4**

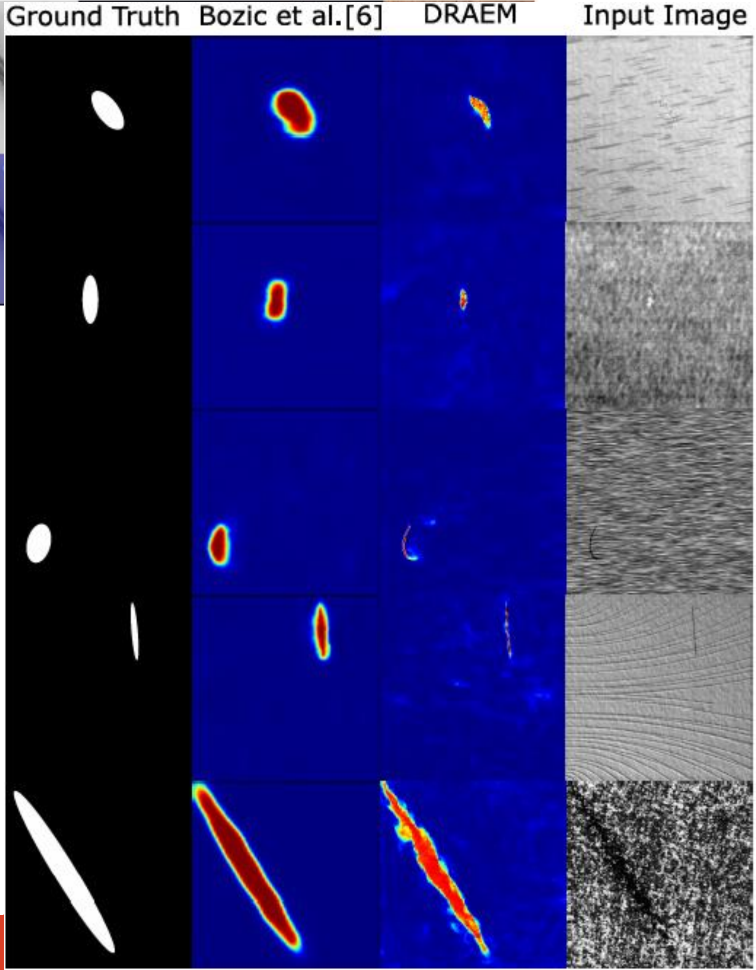
Combining reconstruction and discrimination



Class	[1]	[26]	[4]	[31]	[20]	[11]	DRÆM
bottle	79.4	98.3	99.0	99.9	**100**	99.9	99.2
capsule	72.1	68.7	86.1	88.4	92.3	91.3	**98.5**
grid	74.3	86.7	81.0	99.6	92.9	96.7	**99.9**
leather	80.8	94.4	88.2	**100**	**100**	**100**	**100**
pill	67.1	76.8	87.9	83.8	83.4	93.3	**98.9**
tile	72.0	96.1	99.1	98.7	97.4	98.1	**99.6**
transistor	80.8	79.4	81.8	90.9	95.9	**97.4**	93.1
zipper	74.4	78.1	91.9	98.1	97.9	90.3	**100**
cable	71.1	66.5	86.2	81.9	**94.0**	92.7	91.8
carpet	82.1	90.3	91.6	84.2	95.5	**99.8**	97.0
hazelnut	87.4	100	93.1	83.3	98.7	92.0	**100.0**
metal nut	69.4	81.5	82.0	88.5	93.1	**98.7**	**98.7**
screw	**100**	**100**	54.9	84.5	81.2	85.8	93.9
toothbrush	70.0	95.0	95.3	**100**	95.8	96.1	**100**
wood	92.0	97.9	97.7	93.0	97.6	**99.2**	99.1
avg	78.2	87.3	87.7	91.7	94.4	95.5	**98.0**

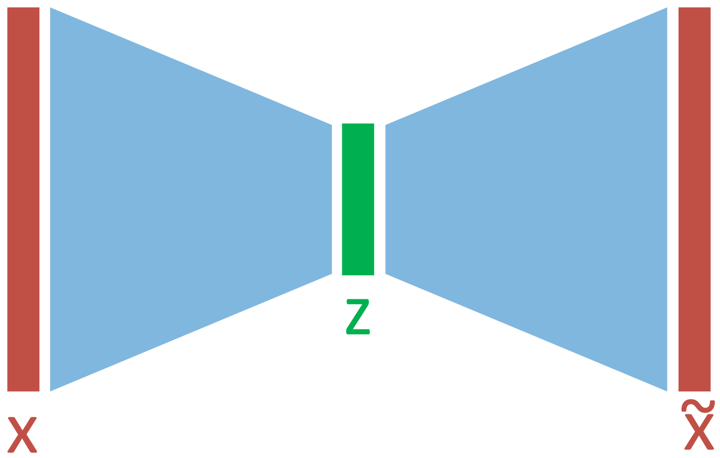
[Zavrtanik et al., 2021]

	Methods	AUROC	TPR	TNR	CA
Unsup.	RIAD [31]	78.6	79.2	69.1	70.4
	US [4]	72.5	72.6	65.3	66.2
	MAD [20]	82.4	78.7	85.7	66.2
	PaDim [11]	95.0	83.3	97.5	95.7
	DRÆM	**99.0**	**96.5**	**99.4**	**98.5**
Sup.	CADN [32]	-	-	-	89.1
	Rački *et al.* [19]	99.6	99.9	99.5	-
	Lin *et al.* [15]	99.0	99.4	99.9	-
	Božič *et al.* [6]	**100**	**100**	**100**	**100**



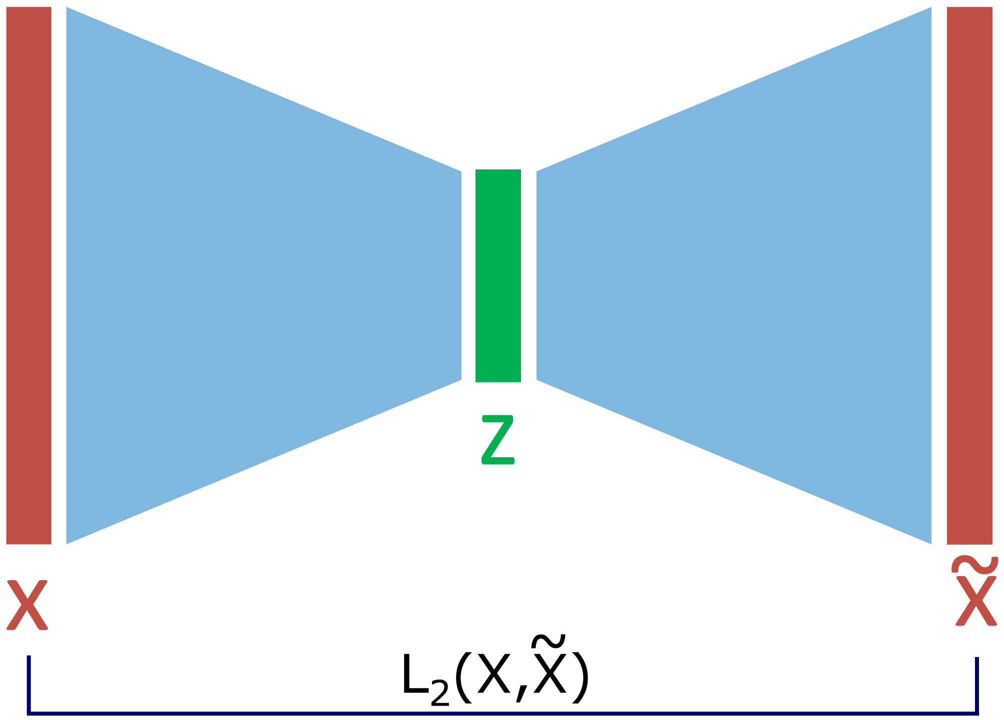
AE recap

- Neural network architecture for unsupervised learning and dimensionality reduction.
- Comprises of an encoder and a decoder.
- Encoder compresses input data into a lower-dimensional latent representation.
- Decoder reconstructs the original input from the latent space.
- Reconstruction loss is used to train the model to minimize the difference between the input and the output.
- Autoencoders can learn meaningful representations and denoise data.
- They are used for data compression, feature extraction, and anomaly detection.
- Variants include sparse autoencoders, denoising autoencoders, and convolutional autoencoders.
- Widely used in various domains, including image processing, natural language processing, and recommendation systems.



Autoencoder latent representation

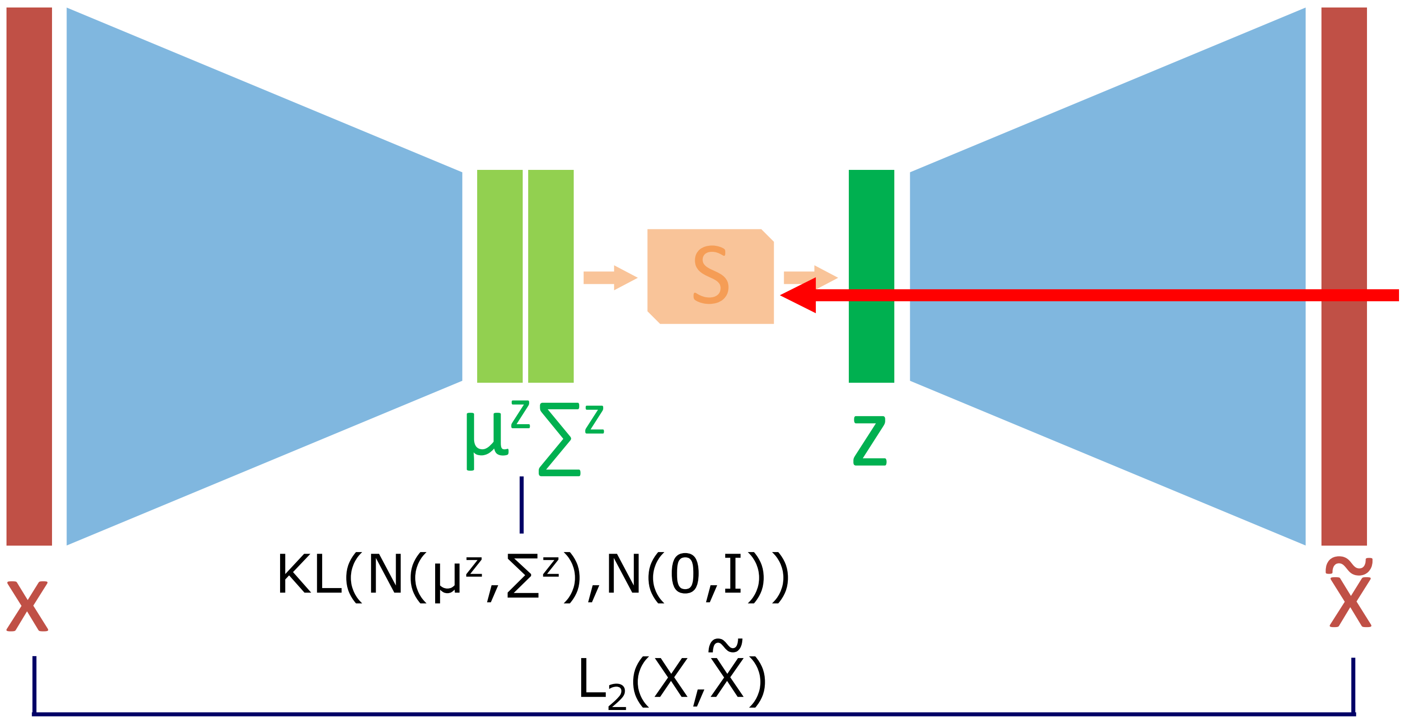
- Autoencoder minimizes the reconstruction error

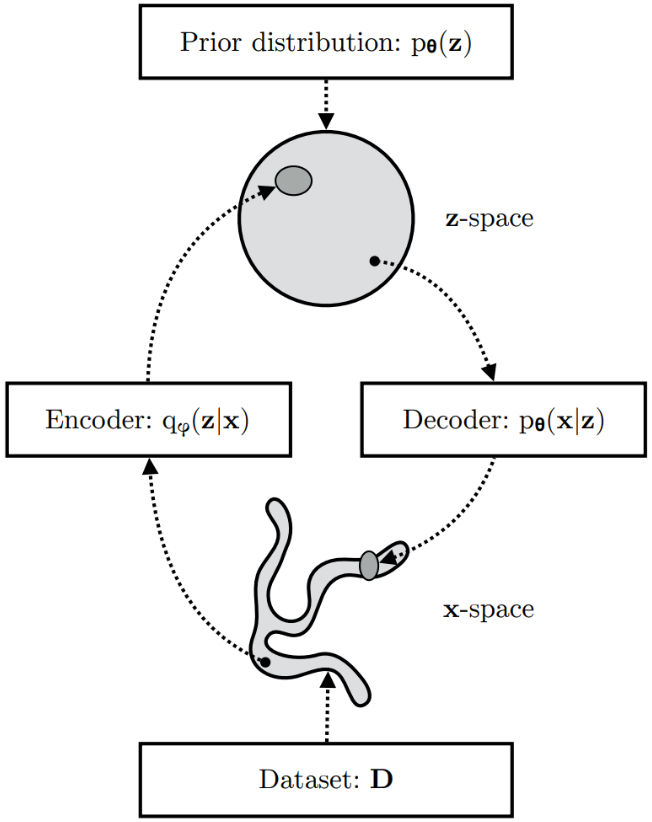


- No constraints for the distribution of z

Variational autoencoder

- Autoencoder + variational inference





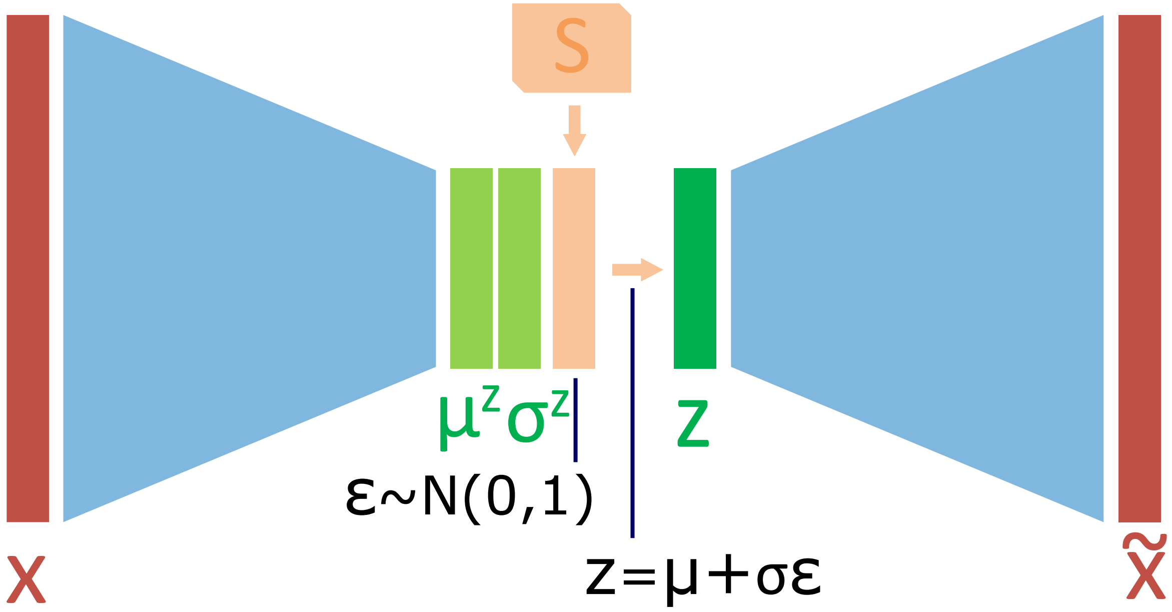
[Kingma & Welling, 2014]

[Kingma, 2016]

- Probabilistic interpretation of the latent space
  - learned latent space well-behaved and structured

Variational autoencoder

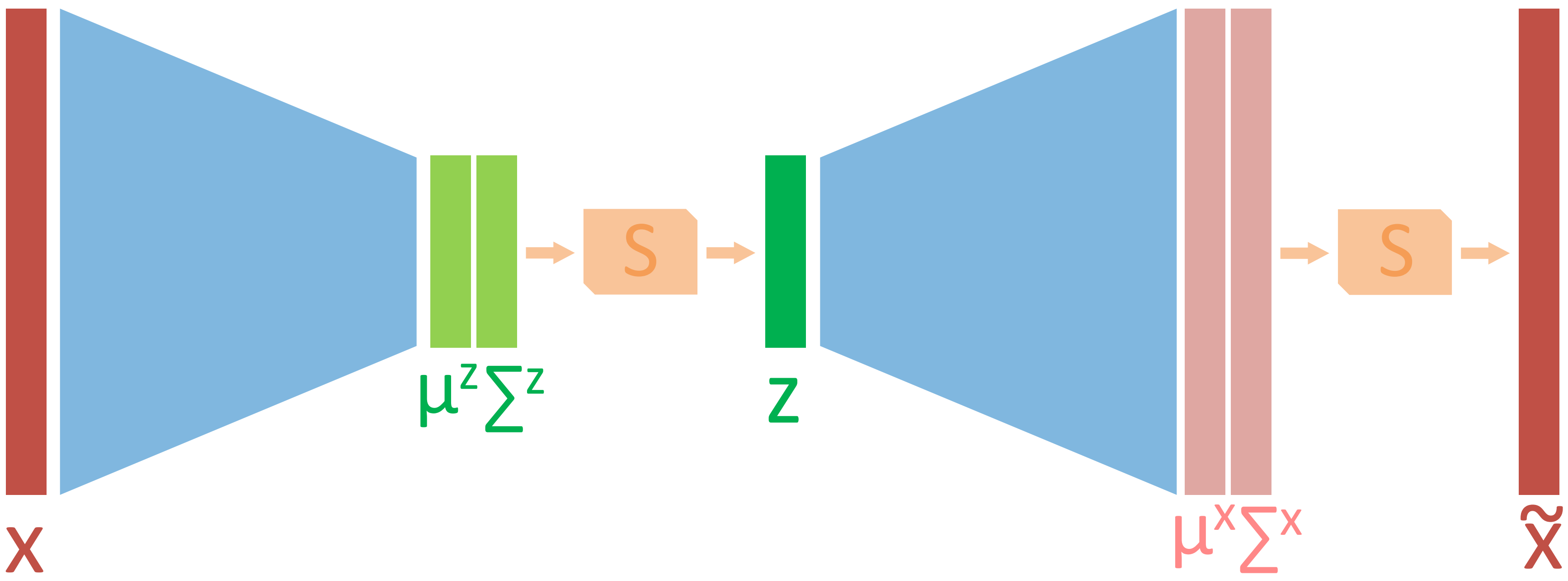
- Reparametrisation trick
  - to deal with nondifferentiable sampling function
  - enables backpropagation



[Kingma & Welling, 2014]

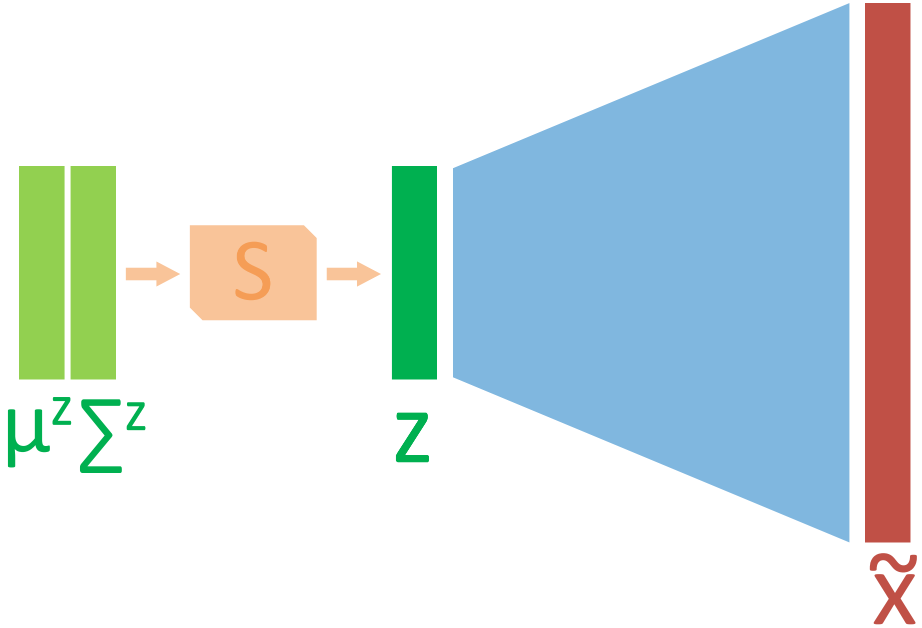
Variational autoencoder

- Generating distribution in the output space as well
- Sampling
  - in the latent space
  - in the output space

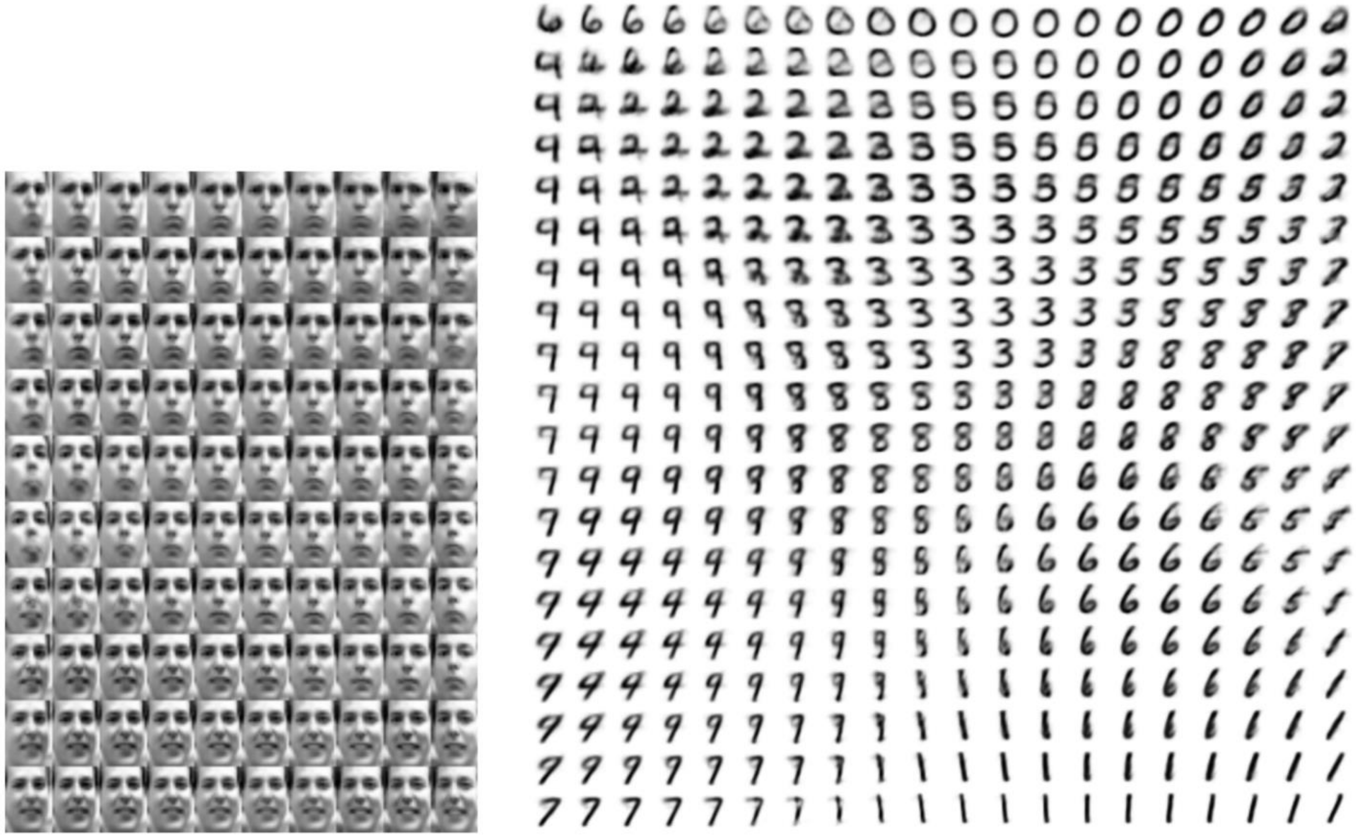


Variational autoencoder

- Generating new samples:



Kingma & Welling, 2014



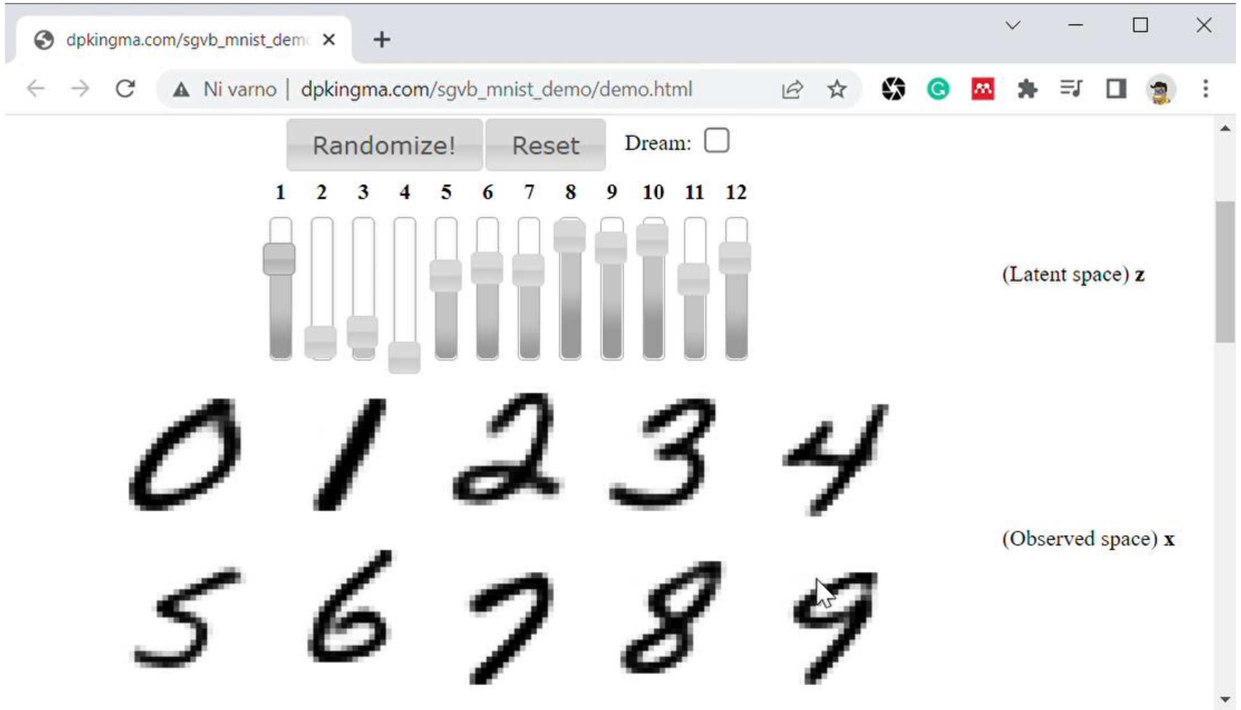
- Randomly sampling z
- Modifying z
- Smooth changes of the generated image when moving sligtly in the latent space

VAE examples

- Faces generated by the model trained on Labeled Faces in the Wild (LFW) dataset
- Digits using MNIST dataset

[Kingma, 2016]

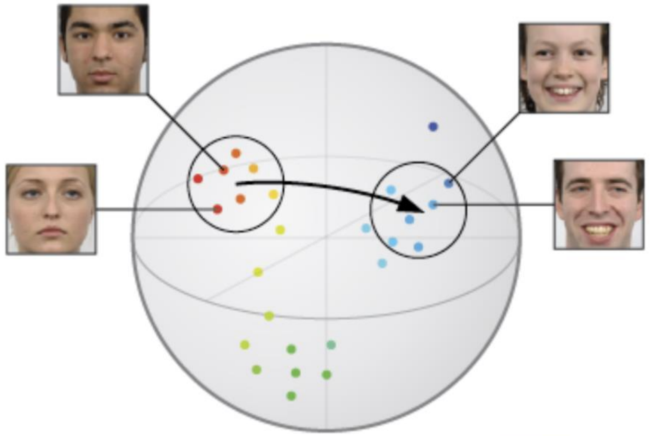
Variational Autoencoder: Generating images



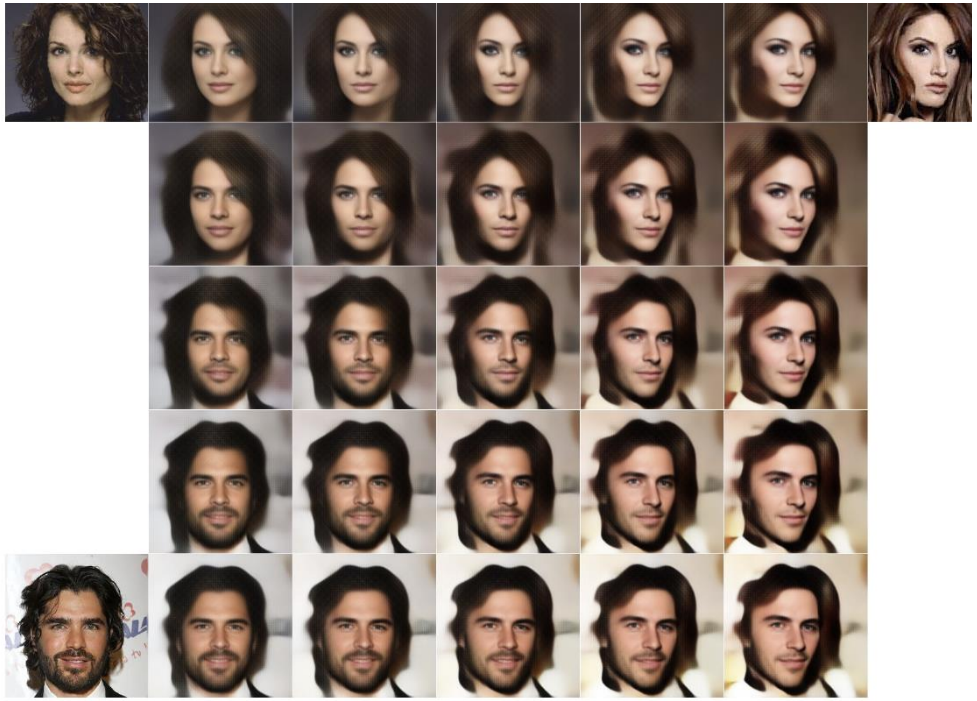
http://dpkingma.com/sgvb_mnist_demo/demo.html

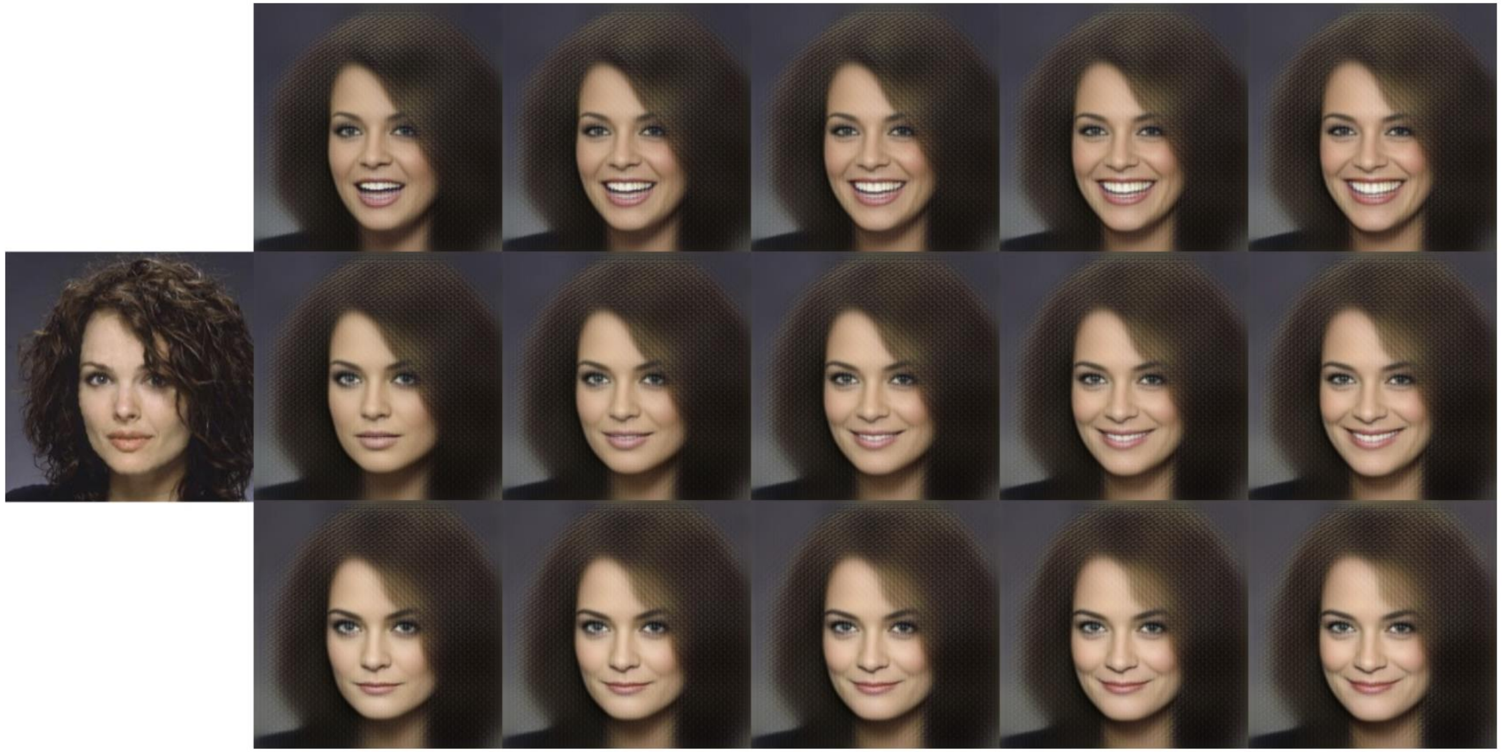
[Kingma & Welling, 2014]

Sampling VAE latent space



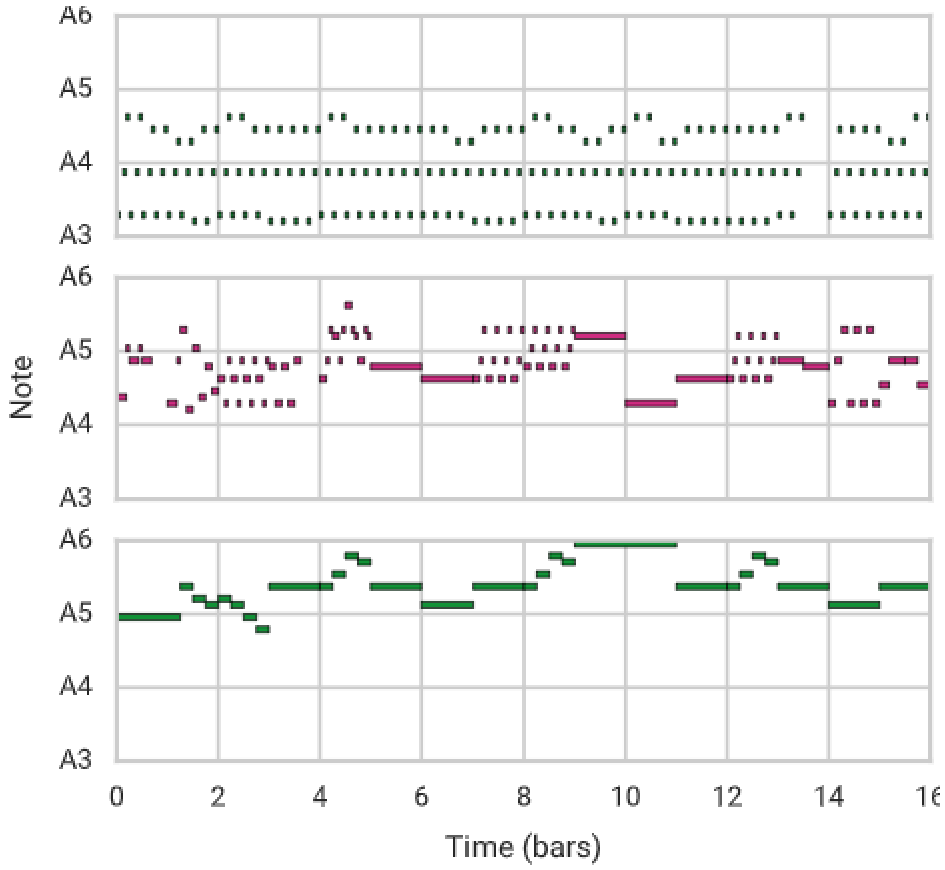
White, 2016

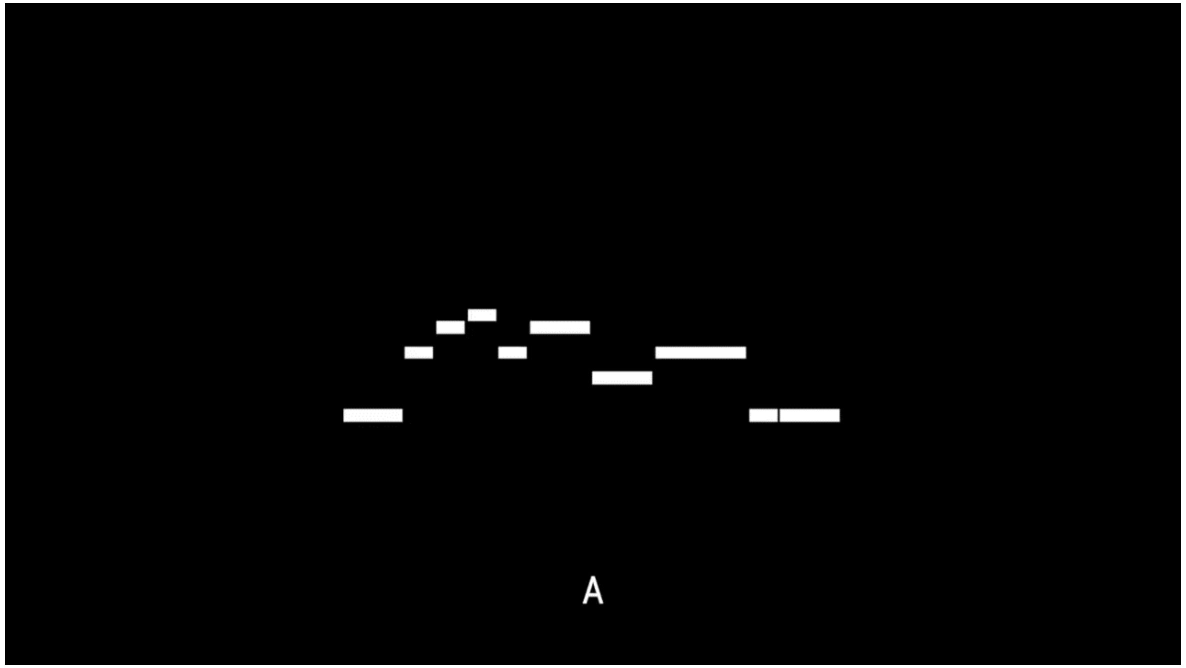




Variational autoencoders in music

- A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music
- VAE on sequential data
- Recurrent encoder and decoder



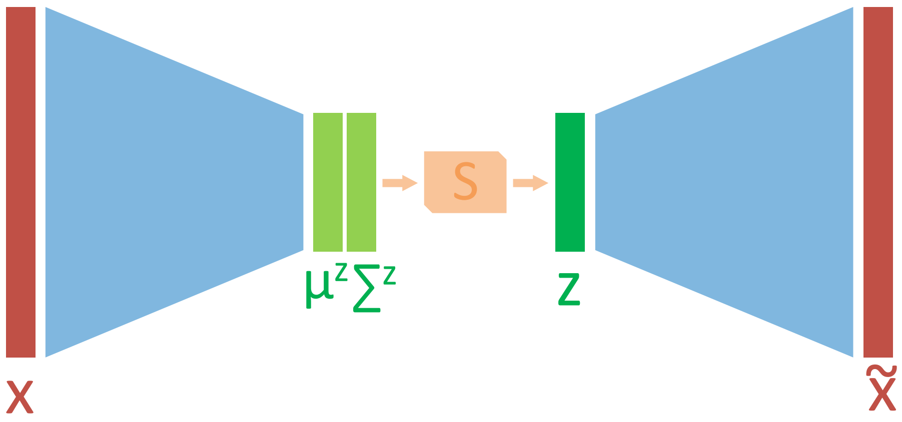


https://magenta.tensorflow.org/music-vae

[Roberts et al., 2019]

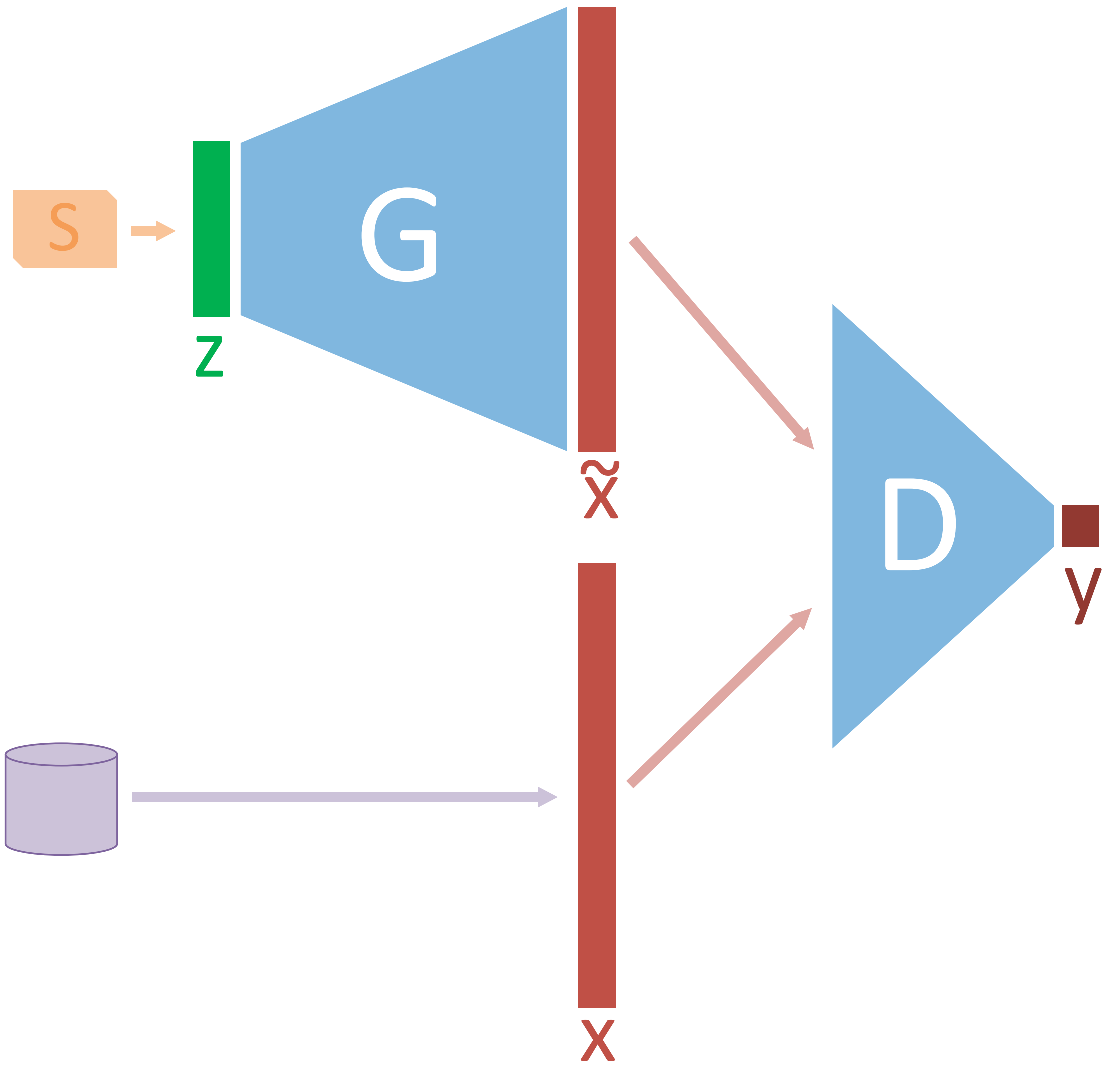
VAE recap

- Generative model for unsupervised learning and dimensionality reduction.
- Combines autoencoder with variational inference.
- Learns compressed representation in a lower-dimensional latent space.
- Encoder maps input data to the latent space, modeling a distribution.
- Decoder reconstructs input data from latent space samples.
- Introduces probabilistic interpretation for flexible generation.
- Training objectives:
  - reconstruction loss
  - KL divergence.
- Reparameterization trick enables backpropagation.
- Applications: data compression, anomaly detection, generating new samples.
- Widely used in deep learning research.



Generative Adversarial Networks

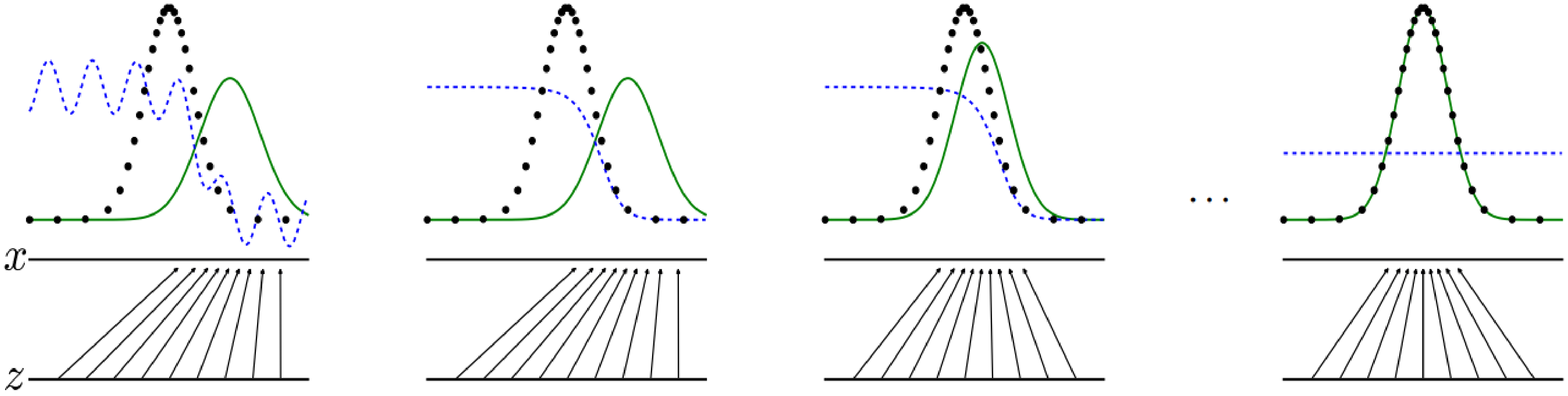
- No explicit probability modelling
- Minimax game:
  - Generator generates syntetic images
    - input: random noise
    - tries to fool the discriminator
  - Discriminator classifies wheter an image is real or fake
    - tries to catch the generator`s fakes
    - it is fed with generated and real samples



Goodfellow et al., 2014

GAN loss function

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})} \underbrace{[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})} [\underbrace{\log(1 - D(G(\boldsymbol{z})))}_{\text{generator loss}}]}_{\text{discriminator loss}}$$



GAN training algorithm

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  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 descending its stochastic gradient:

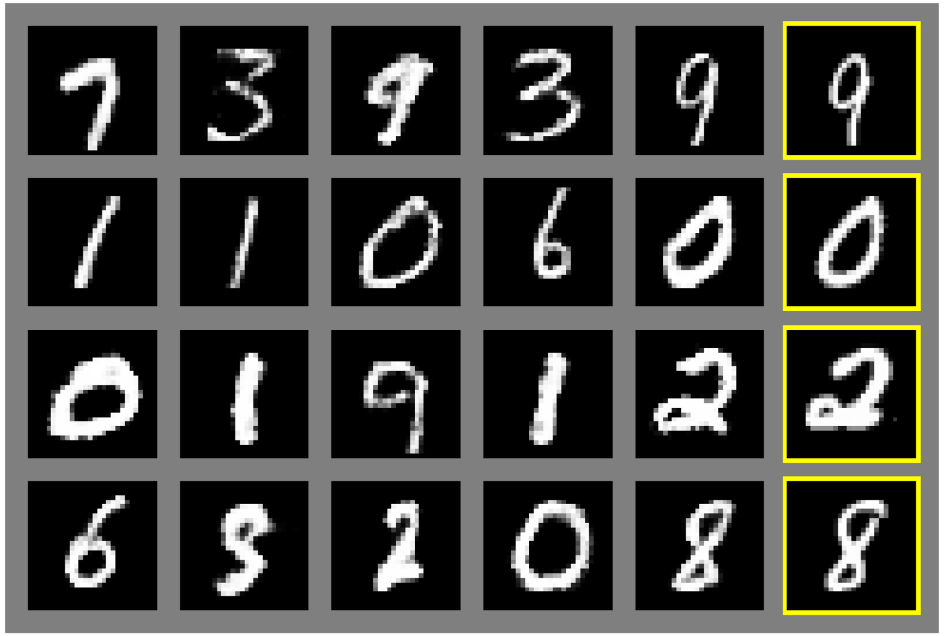
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

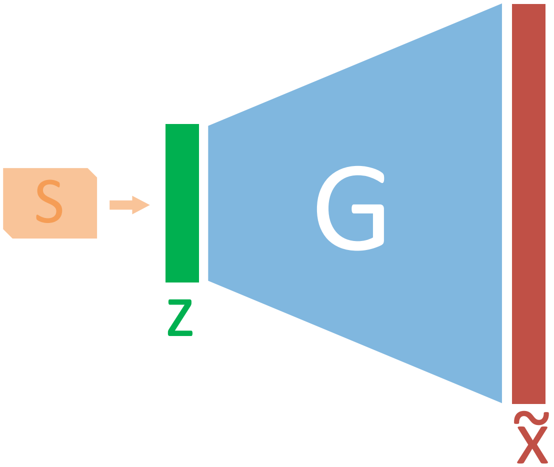
end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

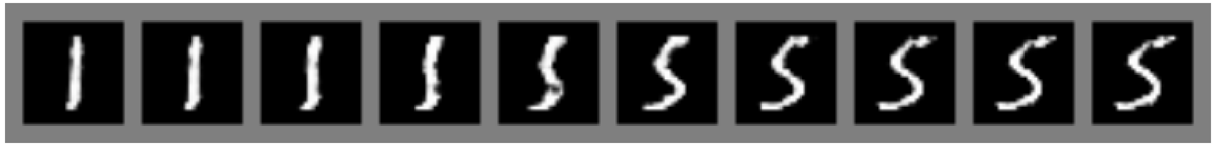
Generating synthetic samples

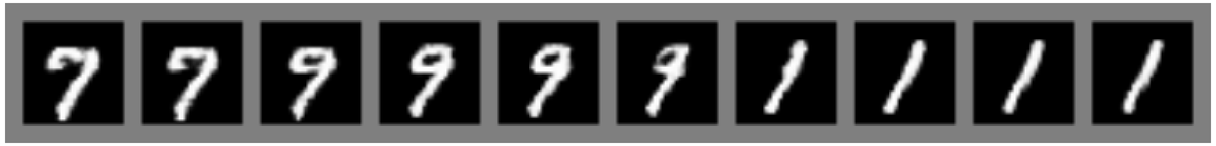
- Keep the generator only
- Randomly sample z



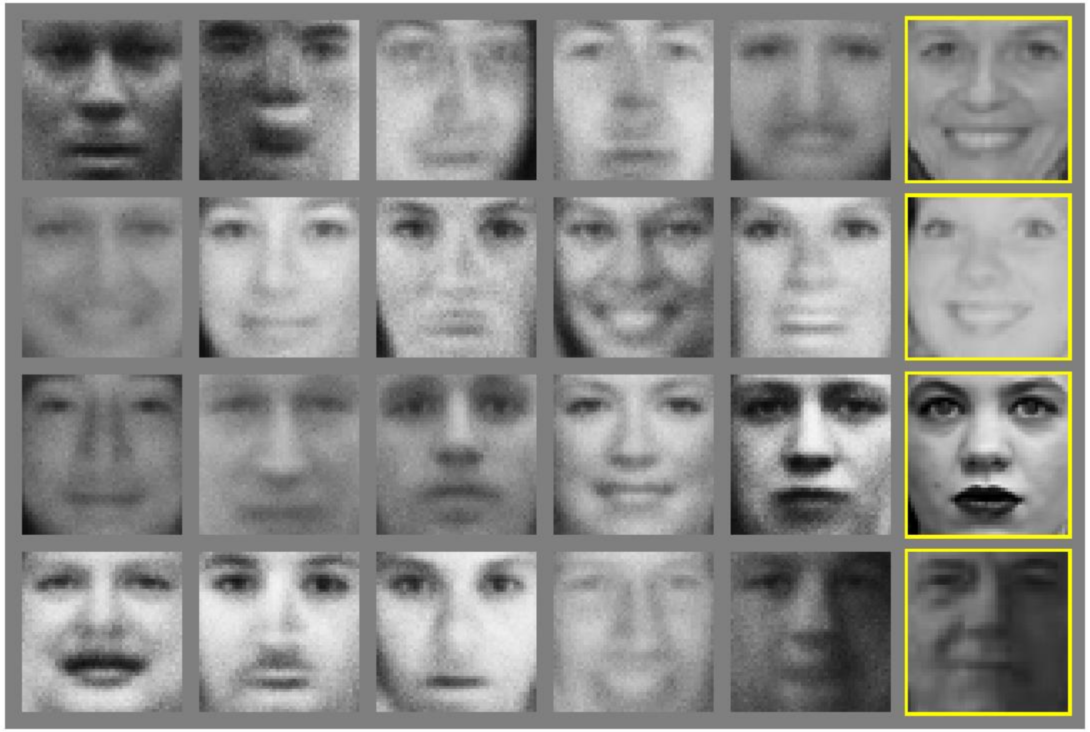


- Linear interpolation in the latent space:





Generating synthetic samples



Goodfellow et al., 2014

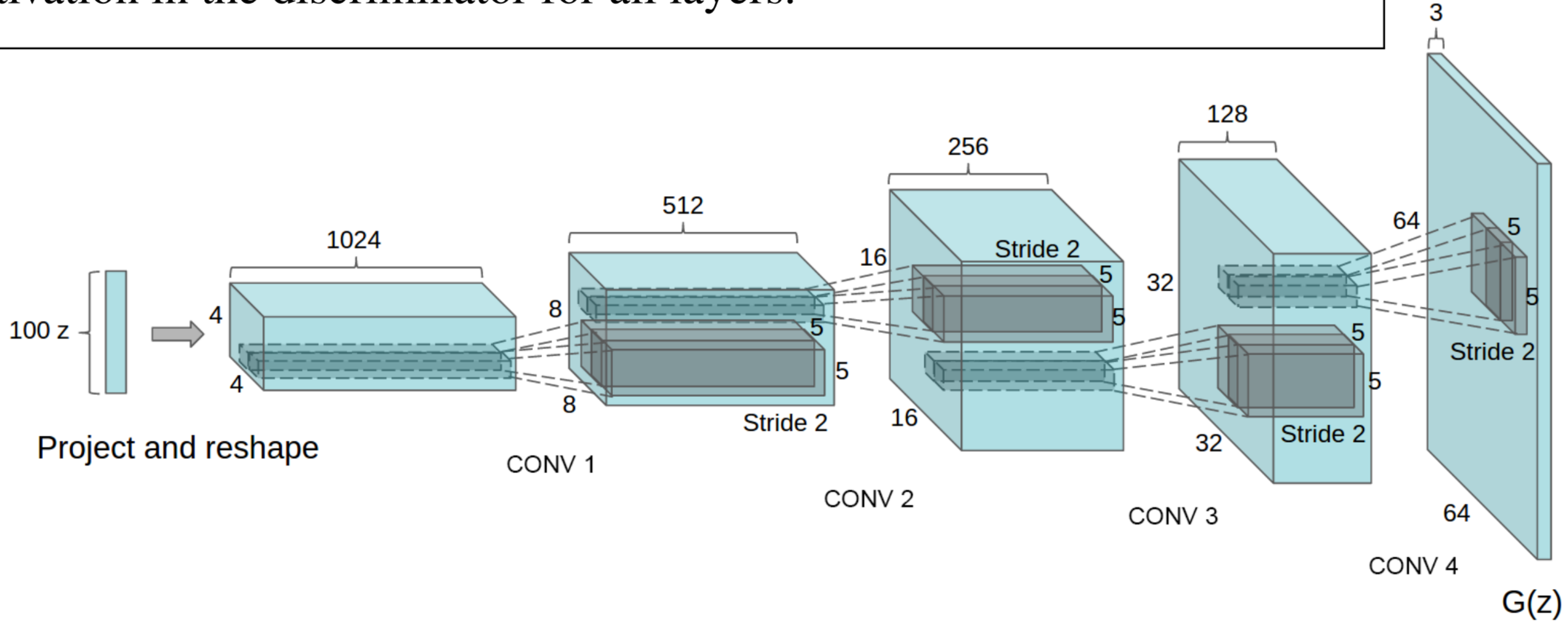
DCGAN

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.

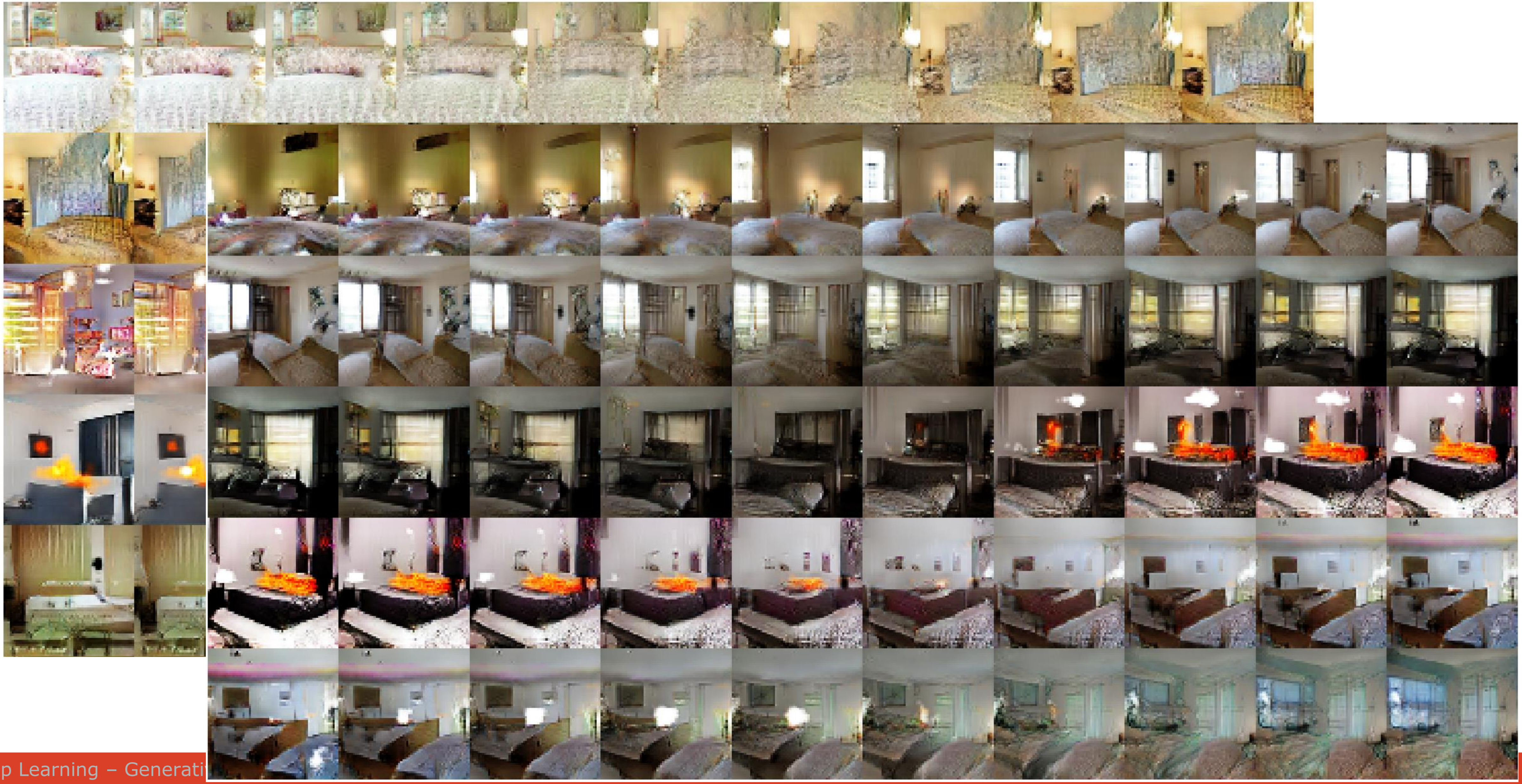
- Unsupervised representation learning
- Deep Convolutional Generative Adversarial Networks

Radford et al., 2016

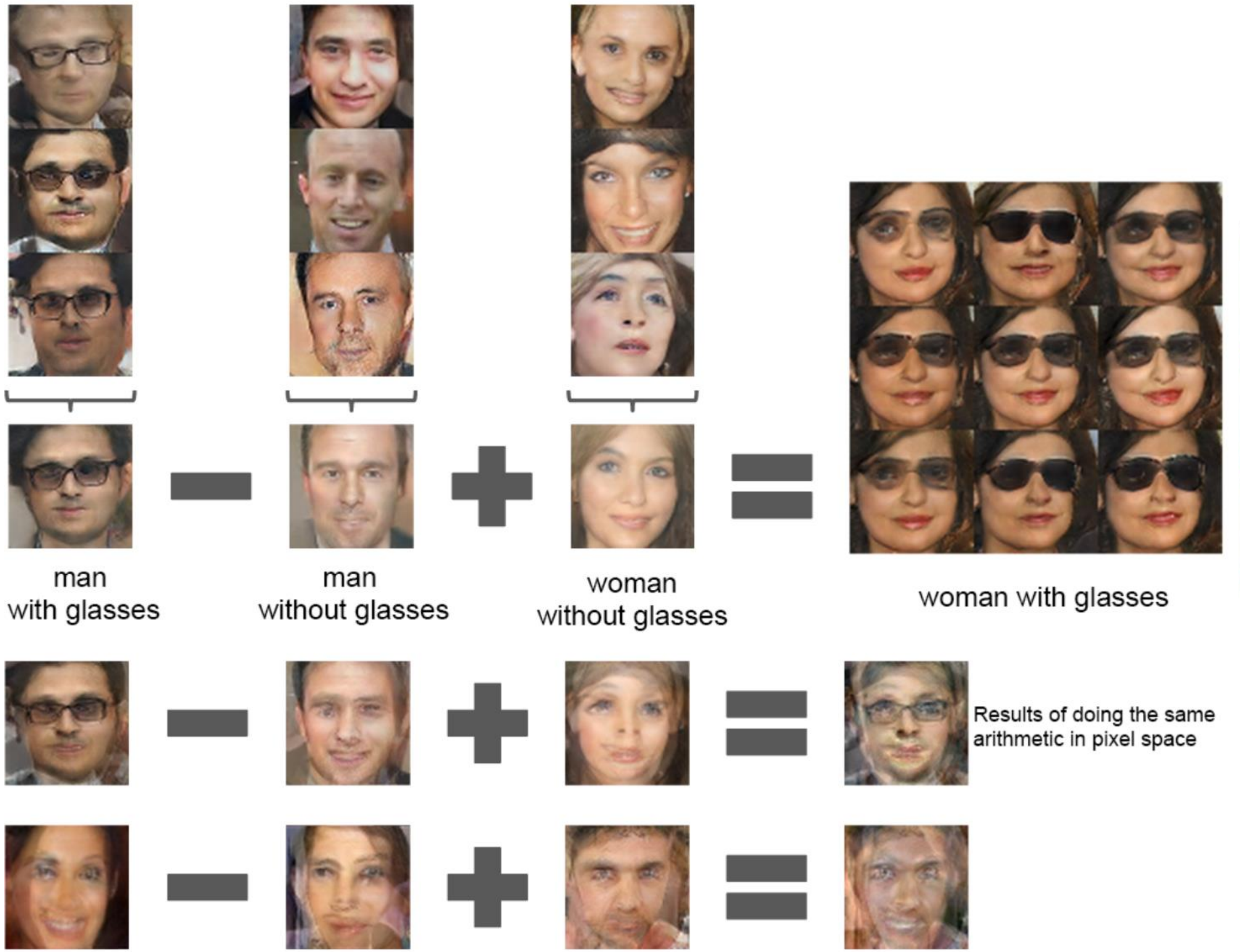


DCGAN – generated examples

DCGAN – interplolating in the latent space



DCGAN – vector arithmetic



Radford et al., 2016

LSGAN

- Least Squares Generative Adversarial Networks
- Higher quality images and more stable training

Mao et al., 2016

$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right]$$

$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right]$$

z, 1024
↓
fc, 7×7×256, BN
↓
3×3, deconv, 256, stride=2, BN
↓
3×3 deconv, 256, stride=1, BN
↓
3×3 deconv, 256, stride=2, BN
↓
3×3 deconv, 256, stride=1, BN
↓
3×3 deconv, 128, stride=2, BN
↓
3×3 deconv, 64, stride=2, BN
↓
3×3 deconv, 3, stride=1

5×5 conv, 64, stride=2
↓
5×5 conv, 128, stride=2, BN
↓
5×5 conv, 256, stride=2, BN
↓
5×5 conv, 512, stride=2, BN
↓
fc, 1
↓
least squares loss

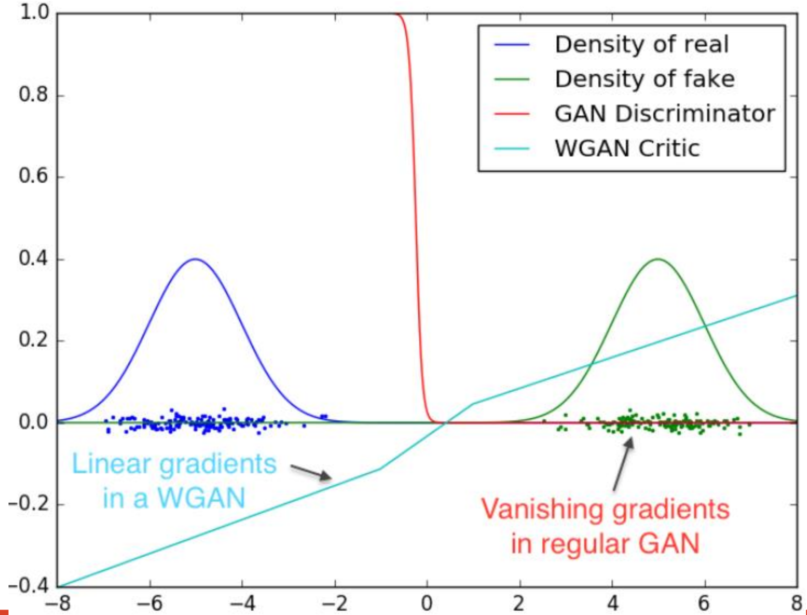
Wasserstain GAN

Arjovsky *et al.*, 2017

- Wasserstain distance more suitable than Jensen-Shannon divergence (GAN):
  - More stable
  - The critic estimates the Wasserstain distance between distributions of fake and real data
  - The generator minimizes the Wasserstain distance between the dist. of fake and real data

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big]$$

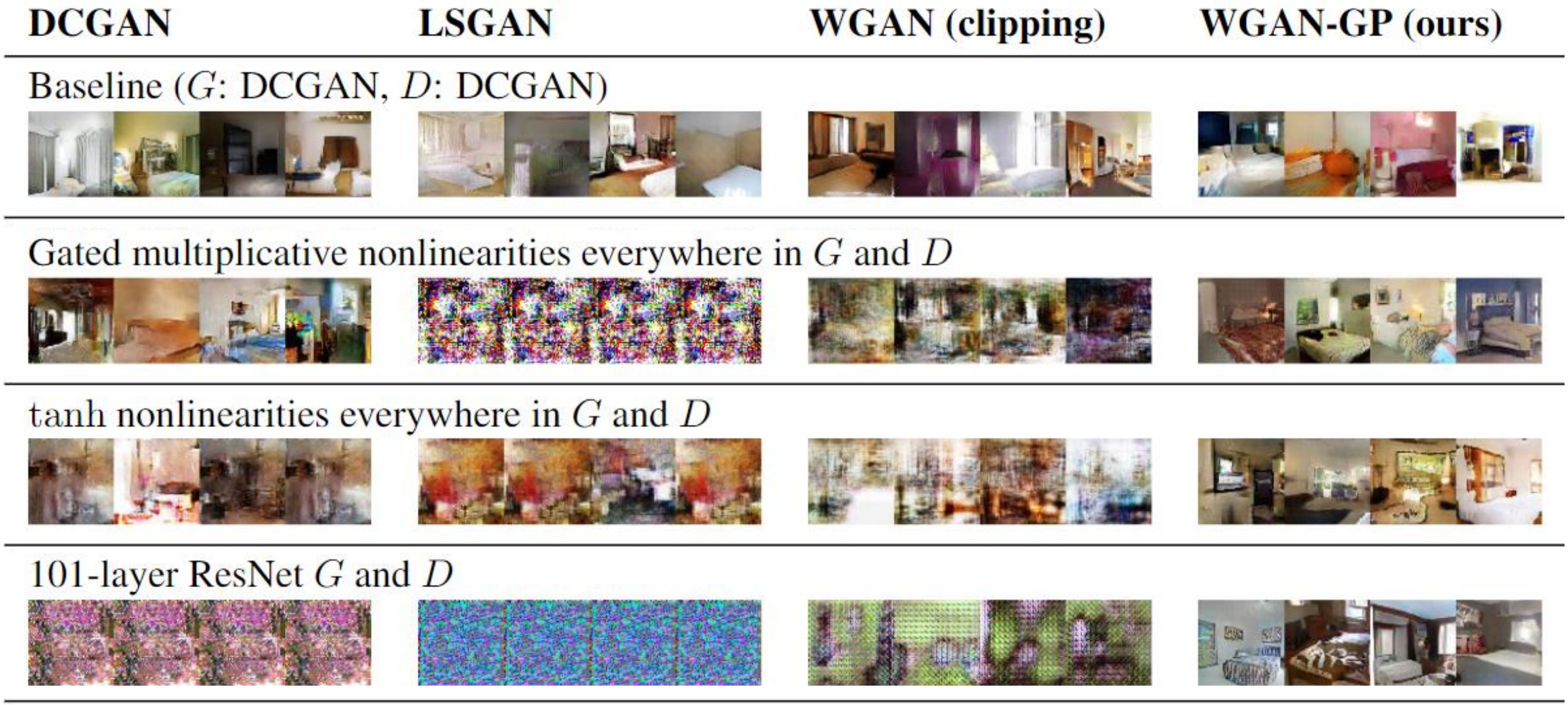
$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$



Improved Wasserstain GAN

Gulrajani *et al.*, 2017

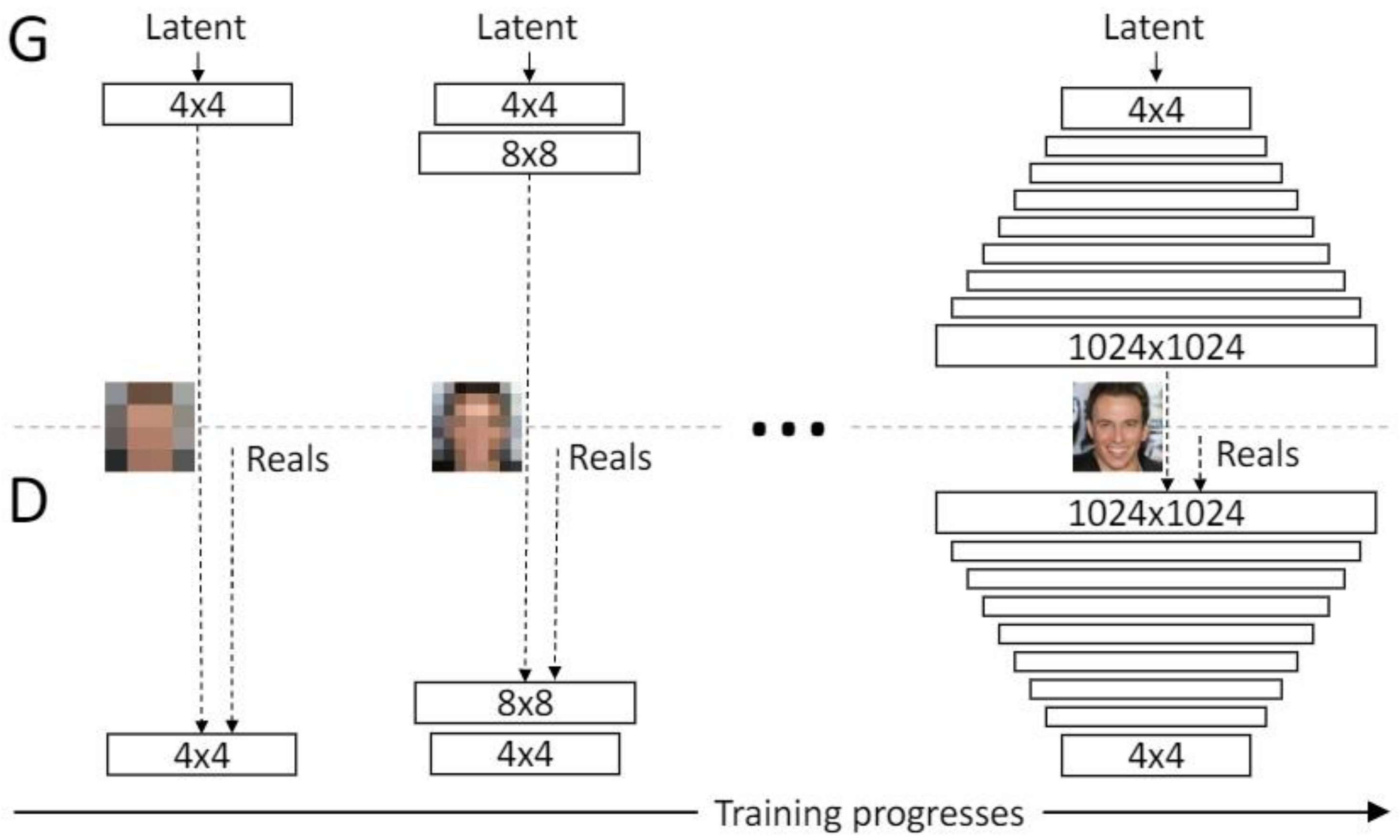
- Avoid weight clipping in WGAN
  - penalize the norm of gradient of the critic with respect to its input



Progressive GAN

Karras *et al.*, 2018

- Progressive growing of GANs for improved quality, stability, and variation
  - Grow generator and discriminator progressively, adding new layers
  - Speeds up and stabilizes learning, produces hi-res images



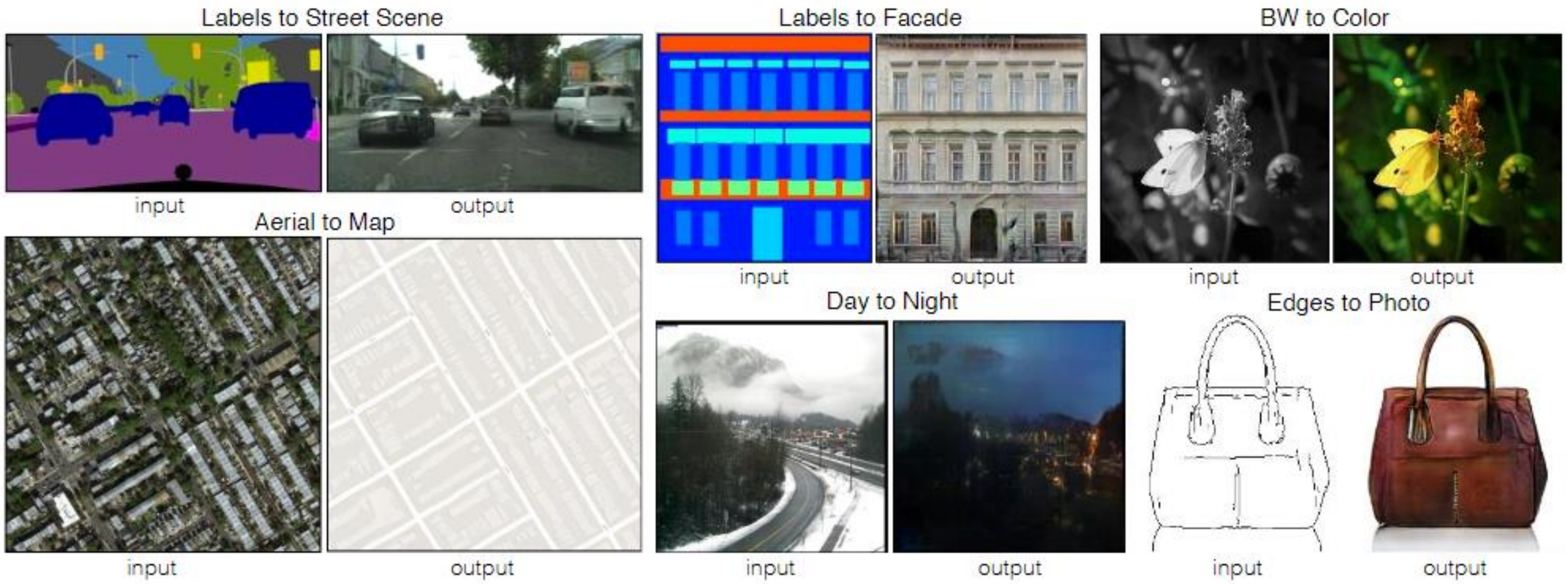
Progressive GAN

https://www.youtube.com/watch?v=G06dEcZ-QTg

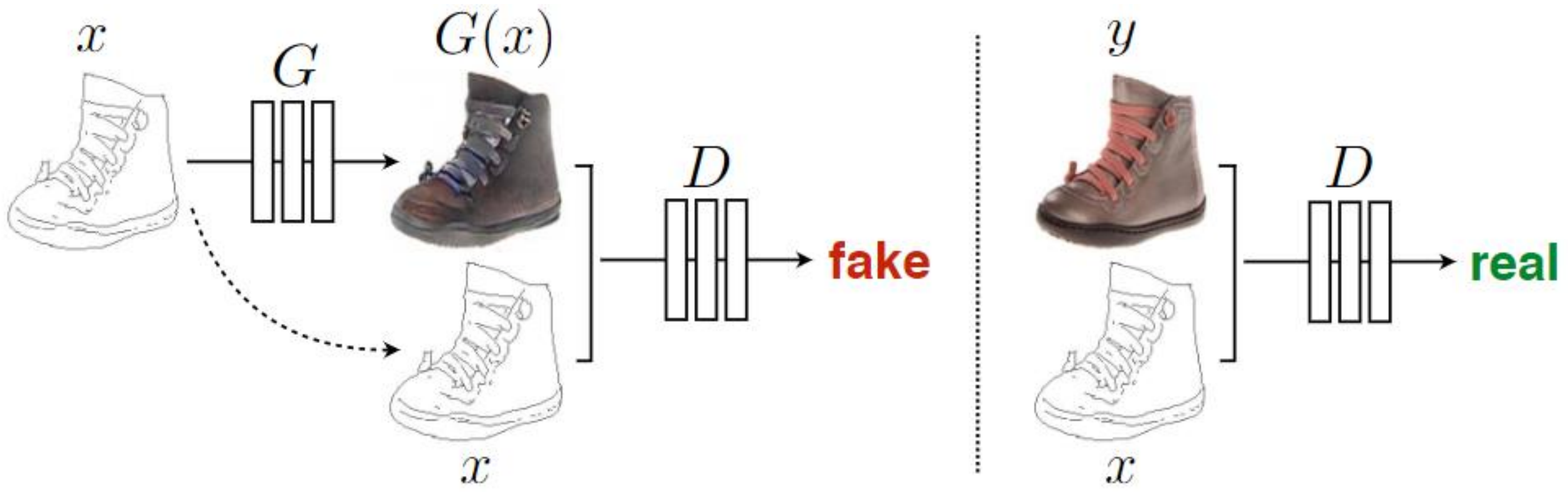
Pix2pix cGAN

Isola *et al.*, 2017

- Image-to-Image Translation with Conditional Adversarial Networks
  - Map the image to another image by aiming at different goals
- Conditional GAN – GAN conditioned on the additional input data



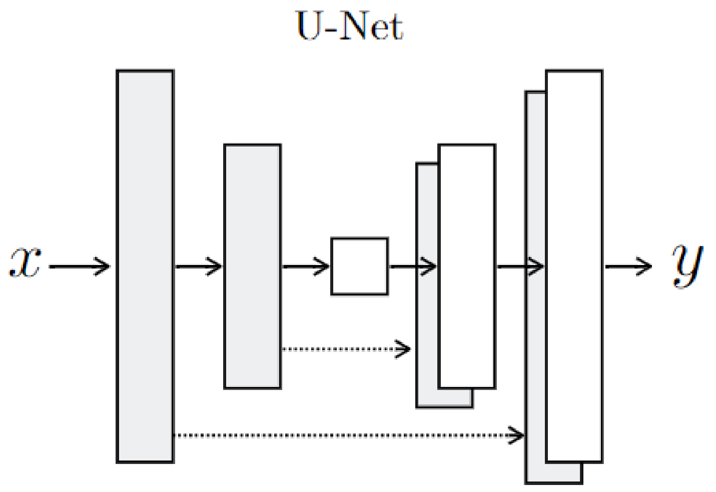
Pix2pix cGAN



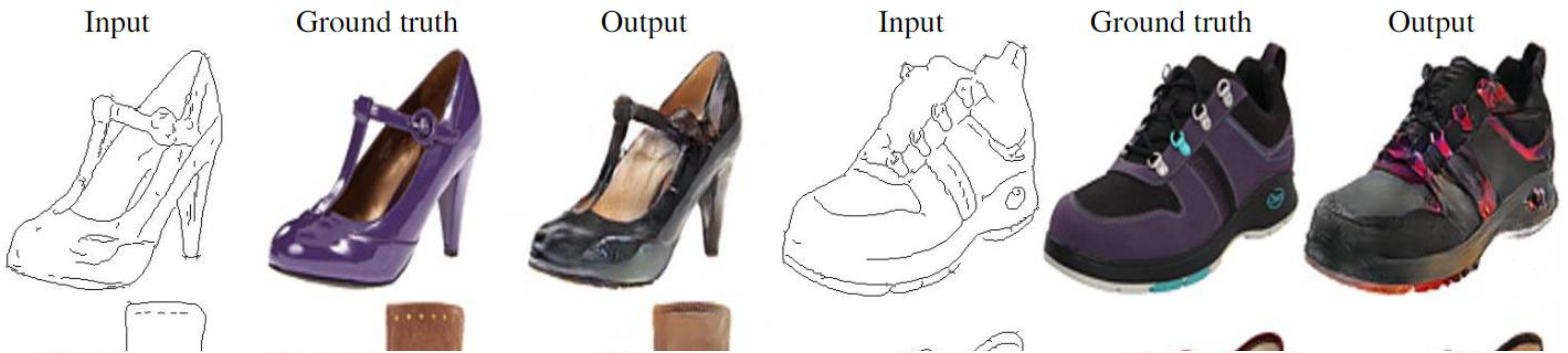
$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x,z)))]$$

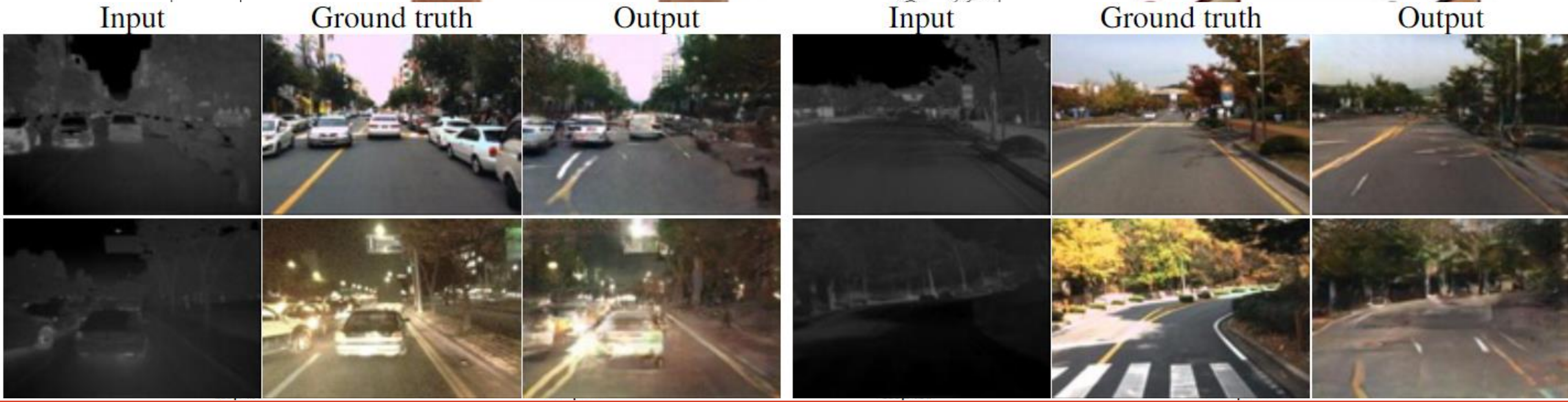
$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1]$$

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G)$$



Pix2pix cGAN

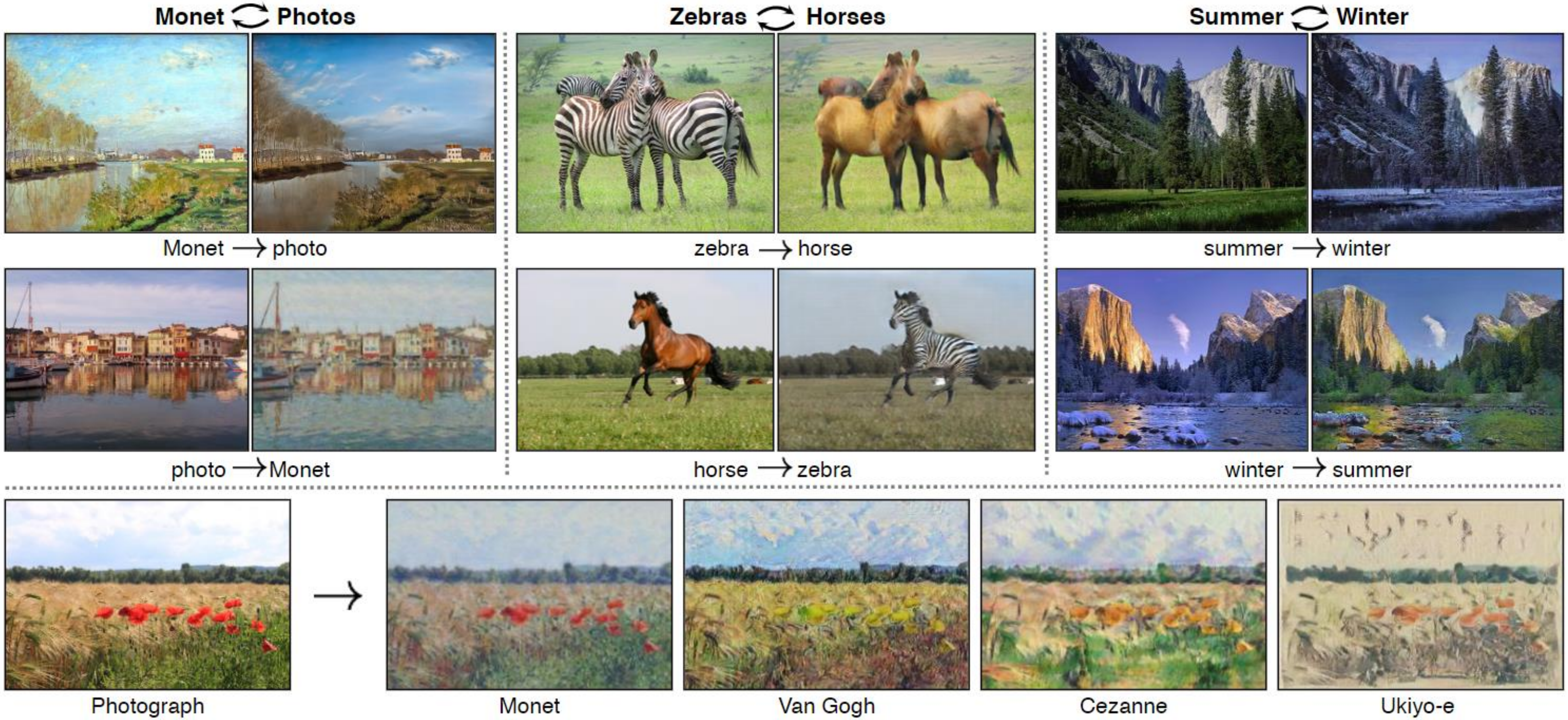




CycleGAN

Zhu *et al.*, 2017

- Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks



CycleGAN

Zhu *et al.*, 2017

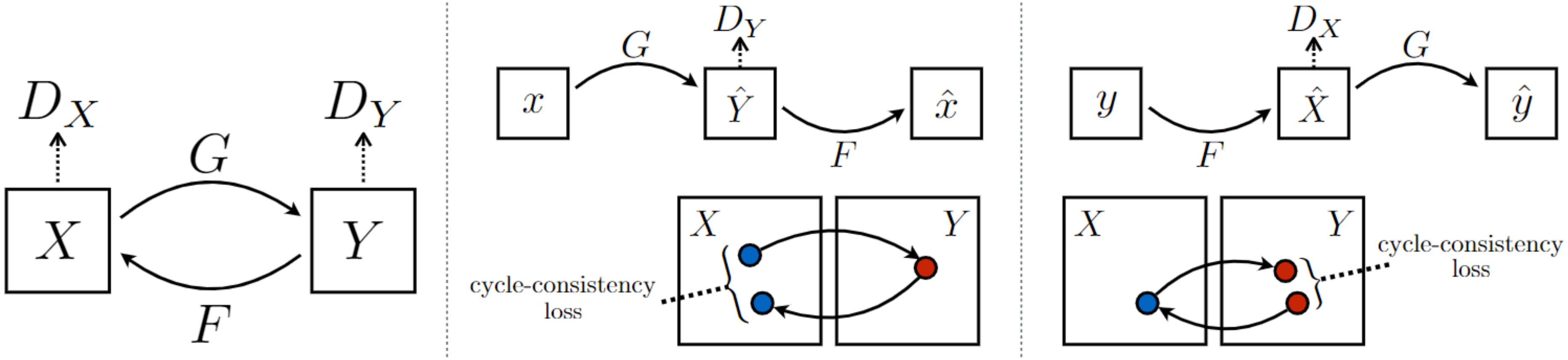
- Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- Cycle consistency loss: forward and backward consistency loss

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x)))]$$

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]$$

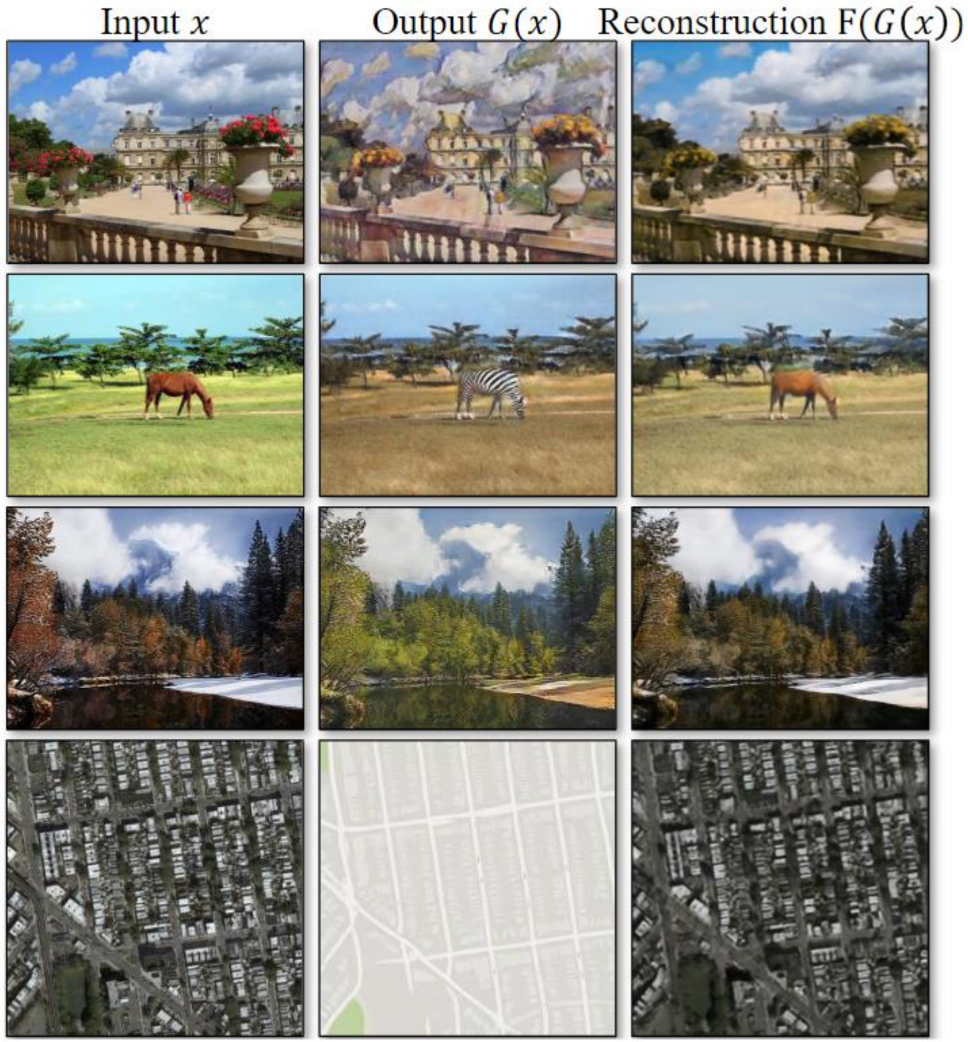
$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F)$$

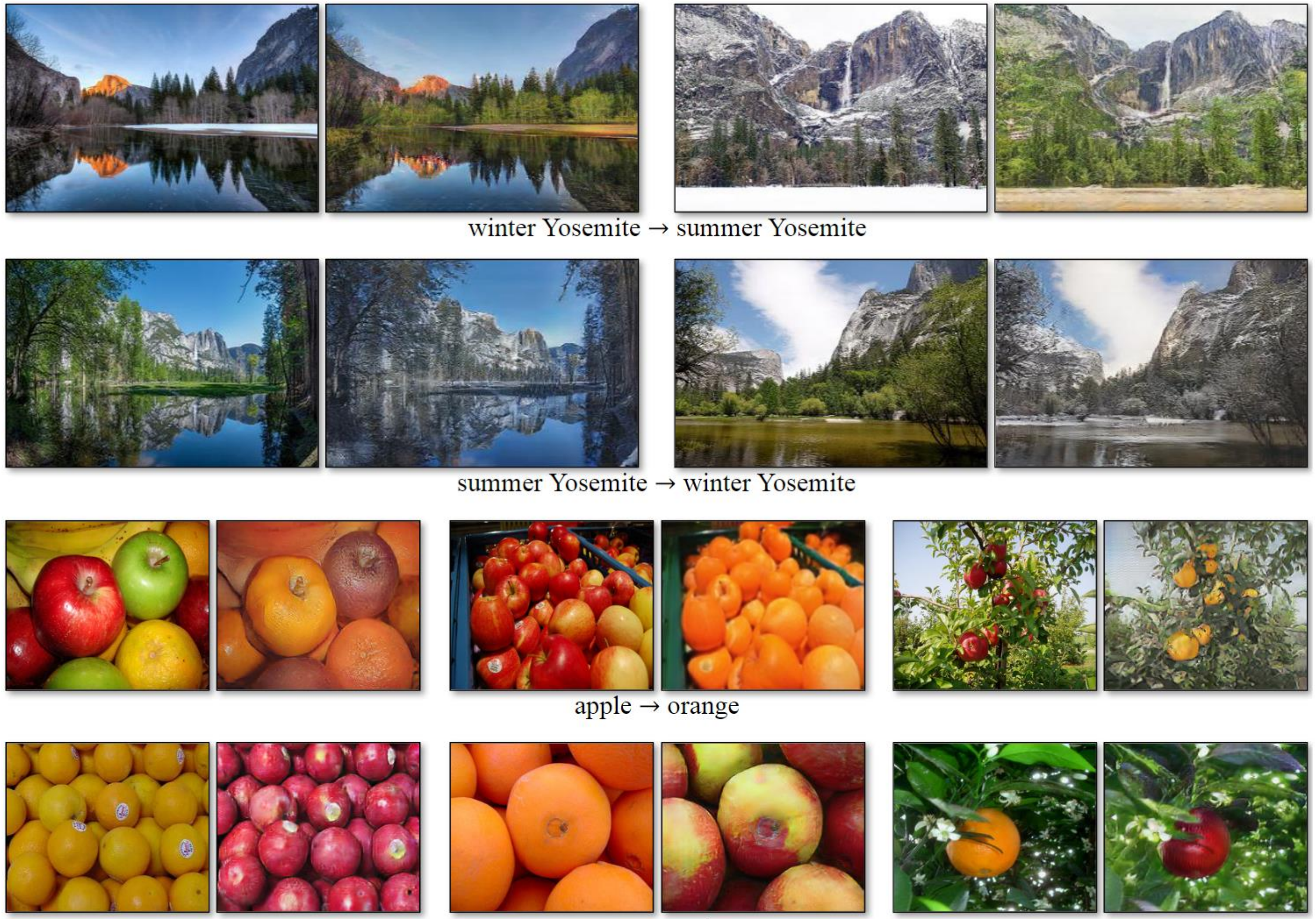
$$G^*, F^* = \arg \min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y)$$



$$x \rightarrow G(x) \rightarrow F(G(x)) \approx x, \qquad y \rightarrow F(y) \rightarrow G(F(y)) \approx y$$

CycleGAN




winter Yosemite → summer Yosemite

summer Yosemite → winter Yosemite

apple → orange

orange → apple

BigGAN

Brock *et al.*, 2019

- Large scale GAN training for high fidelity natural image synthesis
  - Scaling up models
  - Improving class-conditional GANs
  - Trained on ImageNet

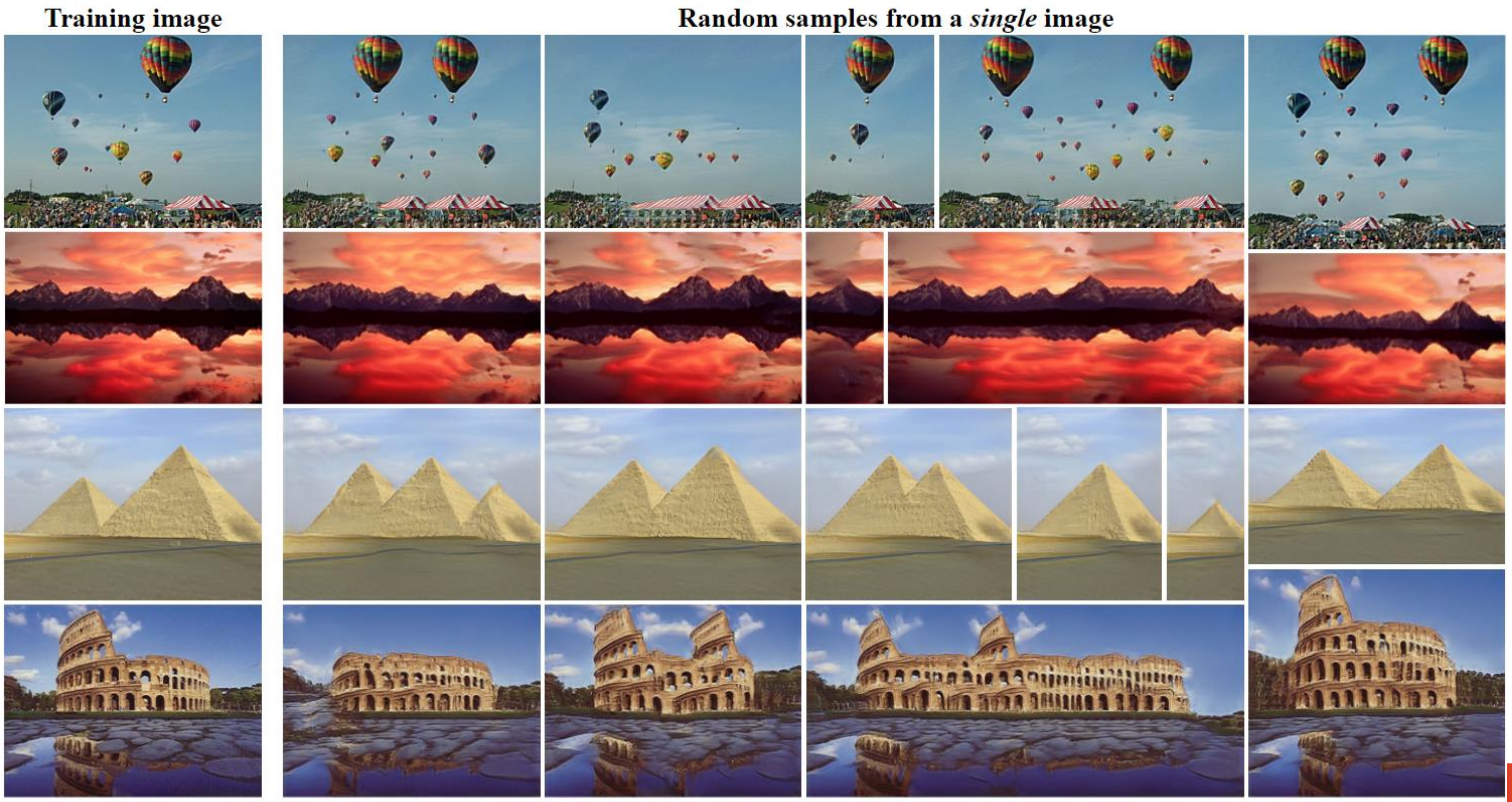
(a) 128×128 (b) 256×256 (c) 512×512

BigGAN

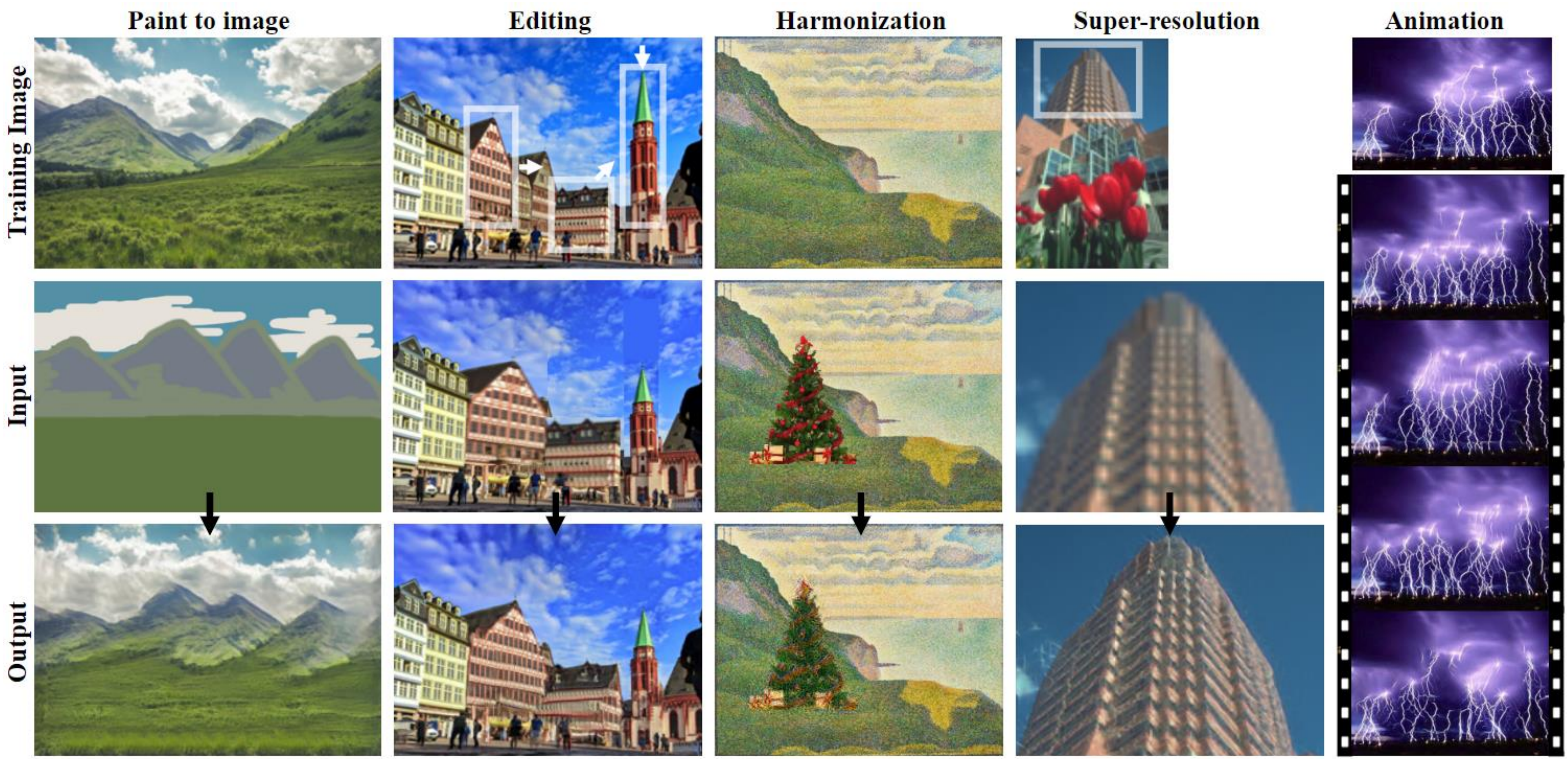
SinGAN

Shaham *et al., 2019*

- Learning a Generative Model from a Single Natural Image



SinGAN



Paint to image
Editing
Harmonization
Super-resolution
Animation
Training Image
Input
Output

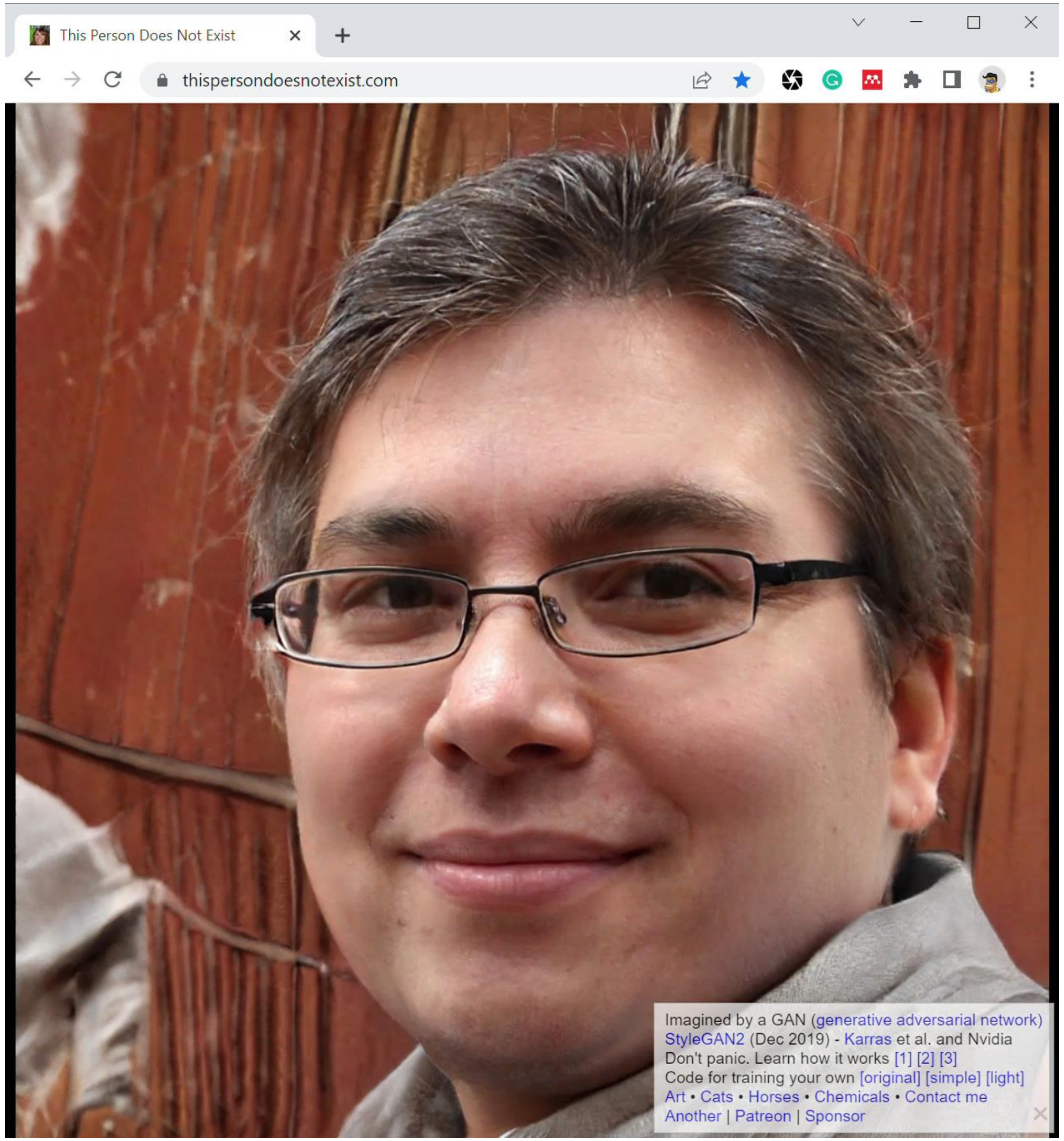
StyleGAN

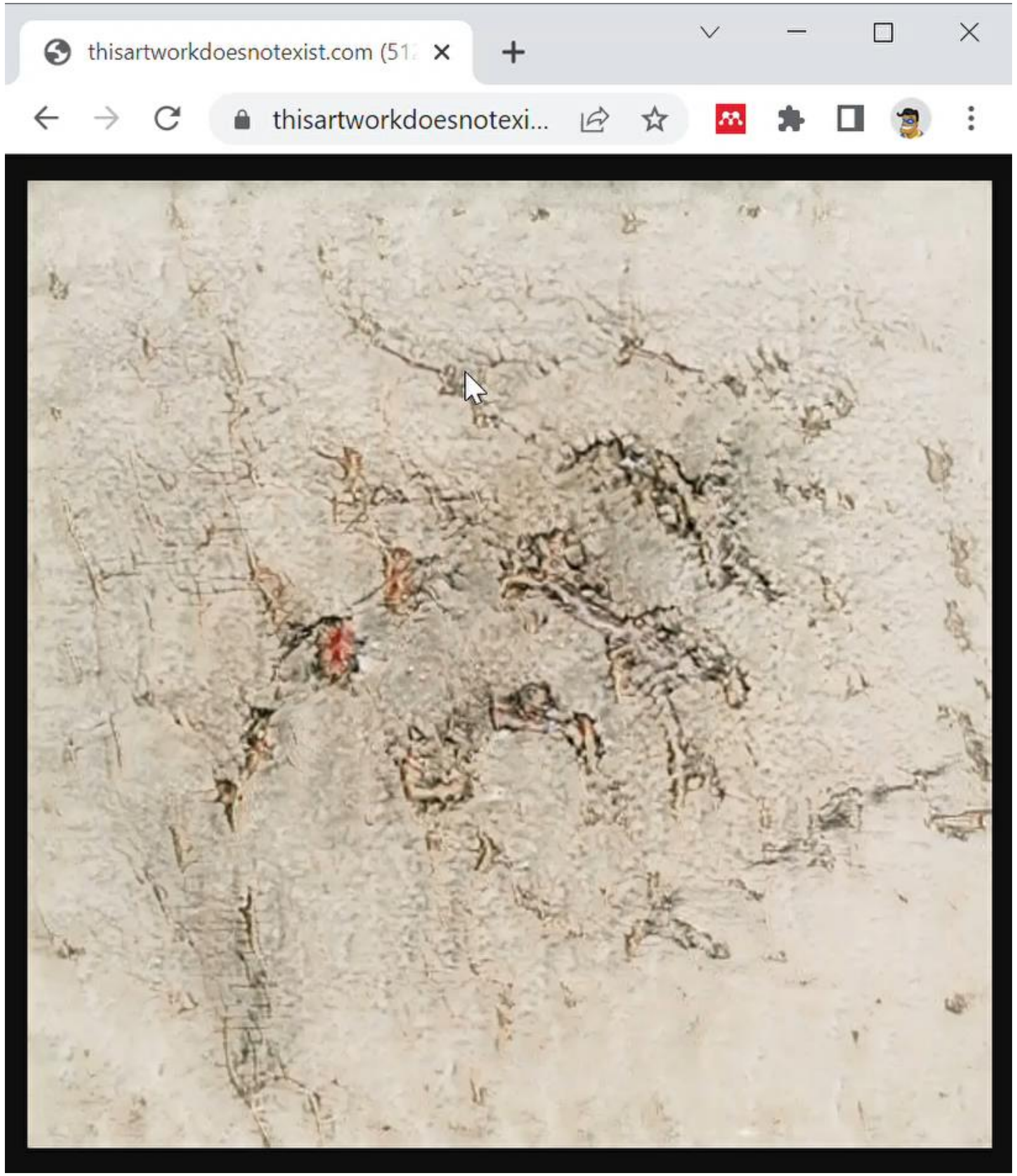
Karras *et al.*, 2018 | Karras *et al.*, 2019 | Karras *et al.*, 2021

- A Style-Based Generator Architecture for Generative Adversarial Networks
- Analyzing and Improving the Image Quality of StyleGAN
- Alias-Free Generative Adversarial Networks

StyleGAN

https://this{person,cat,artwork}doesnotexist.com/

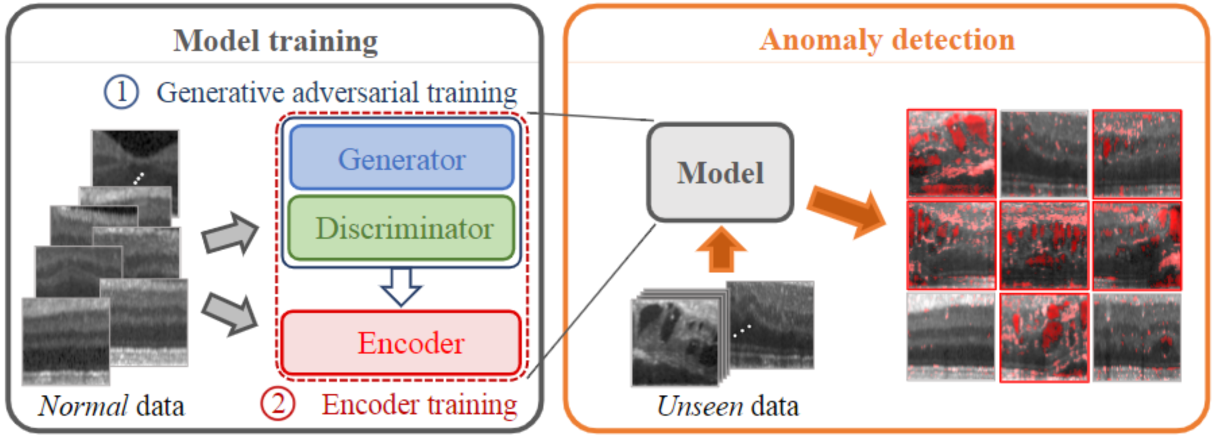


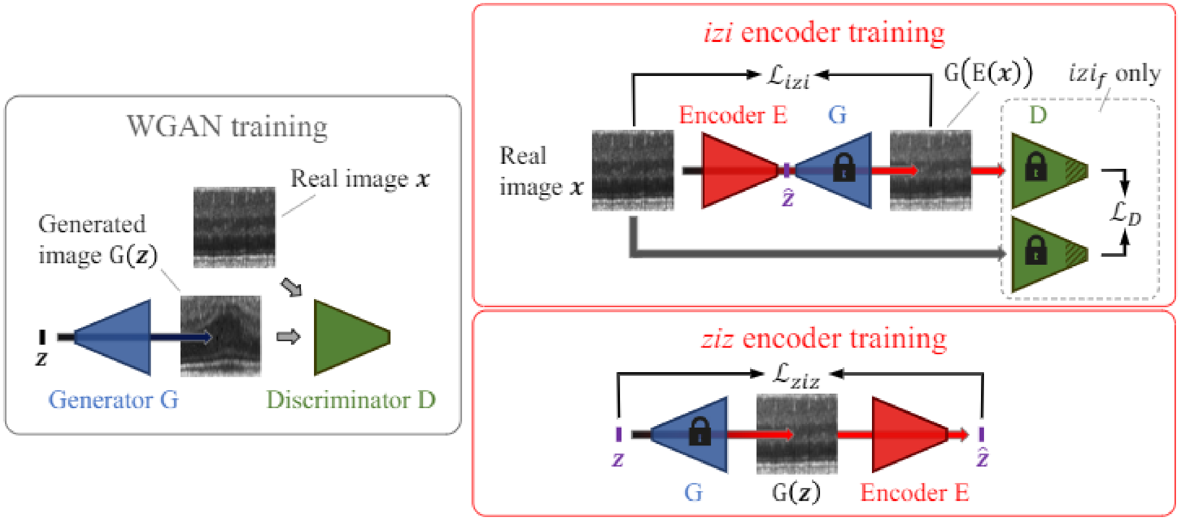


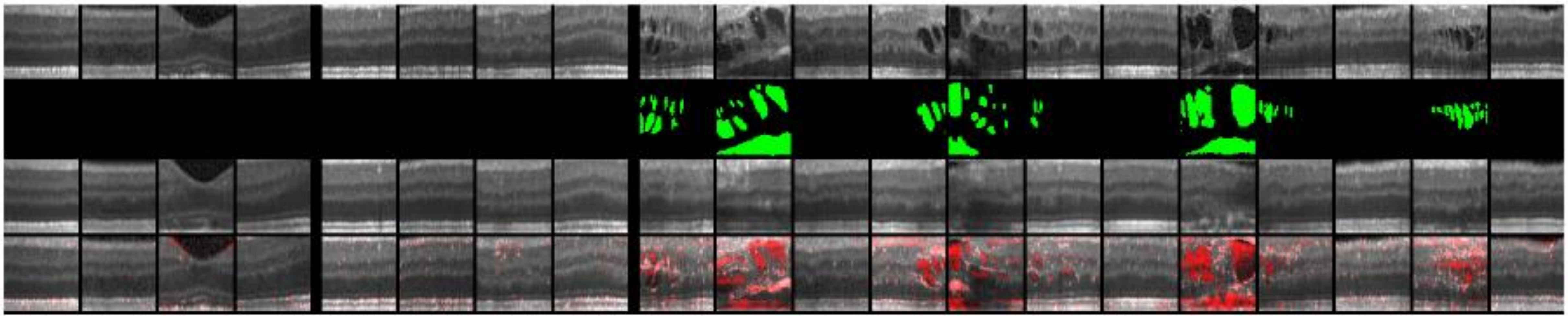
f-AnoGAN

[Schiegl *et al.*, 2019]

- Fast Unsupervised Anomaly Detection with Generative Adversarial Networks



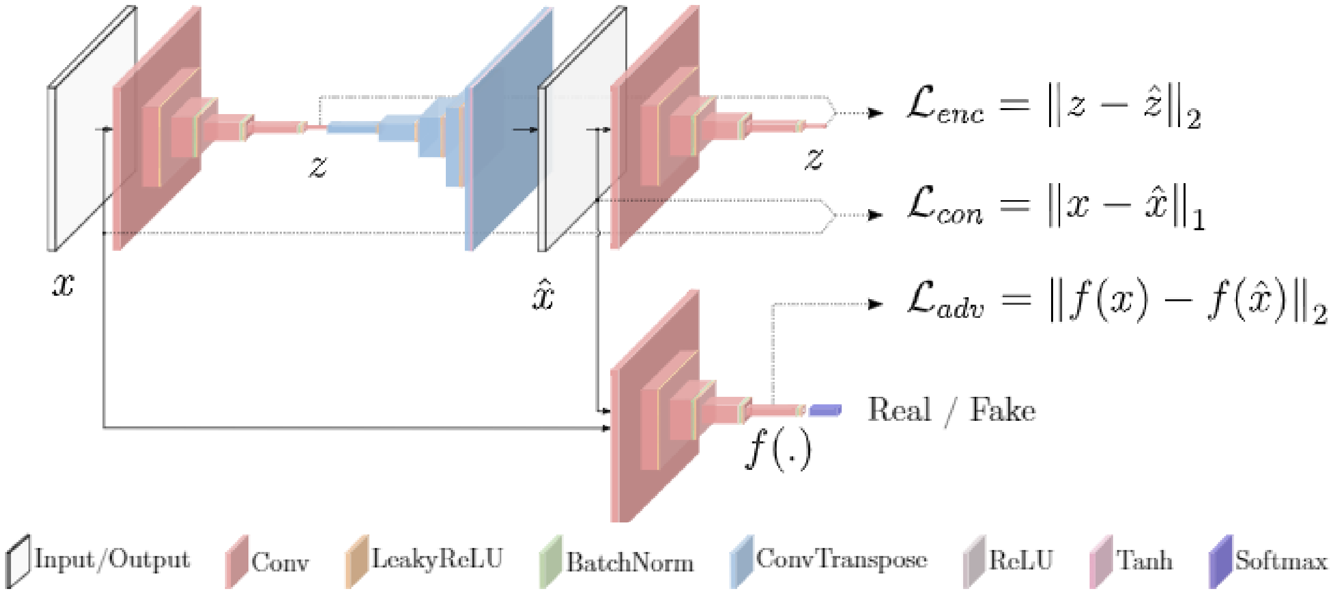




GANomaly

[Akcay et al., 2018]

- Semi-Supervised Anomaly Detection via Adversarial Training

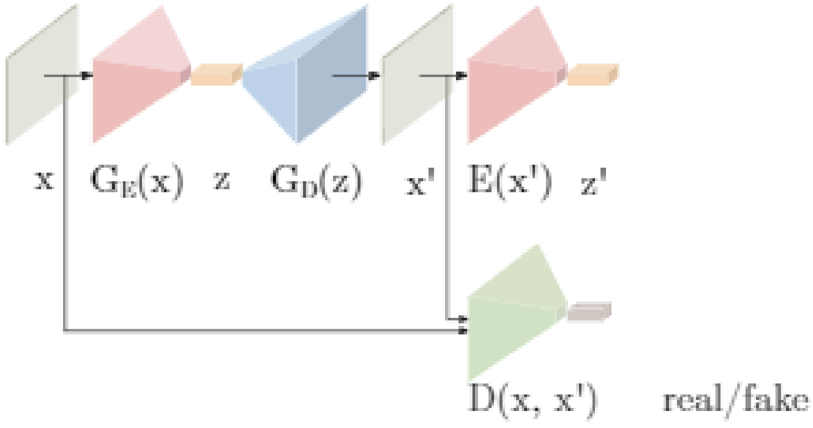


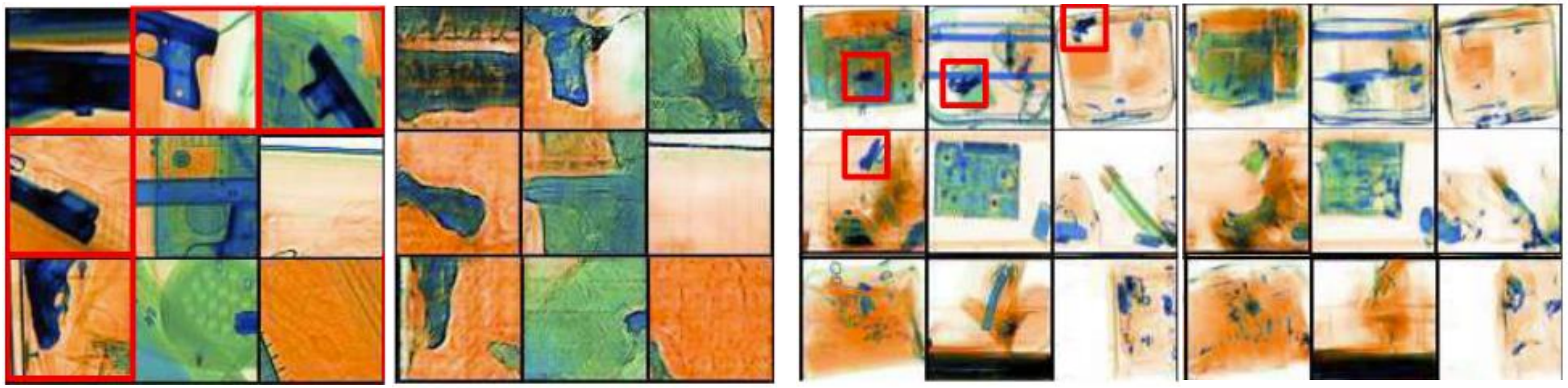
$$\mathcal{L}_{enc} = \mathbb{E}_{x \sim p\mathbf{x}} \|G_E(x) - E(G(x))\|_2$$

$$\mathcal{L}_{con} = \mathbb{E}_{x \sim p\mathbf{x}} \|x - G(x)\|_1$$

$$\mathcal{L}_{adv} = \mathbb{E}_{x \sim p\mathbf{x}} \|f(x) - \mathbb{E}_{x \sim p\mathbf{x}} f(G(x)\|_2$$

$$\mathcal{L} = w_{adv}\mathcal{L}_{adv} + w_{con}\mathcal{L}_{con} + w_{enc}\mathcal{L}_{enc}$$





$$\mathcal{A}(\hat{x}) = \|G_E(\hat{x}) - E(G(\hat{x}))\|_1$$

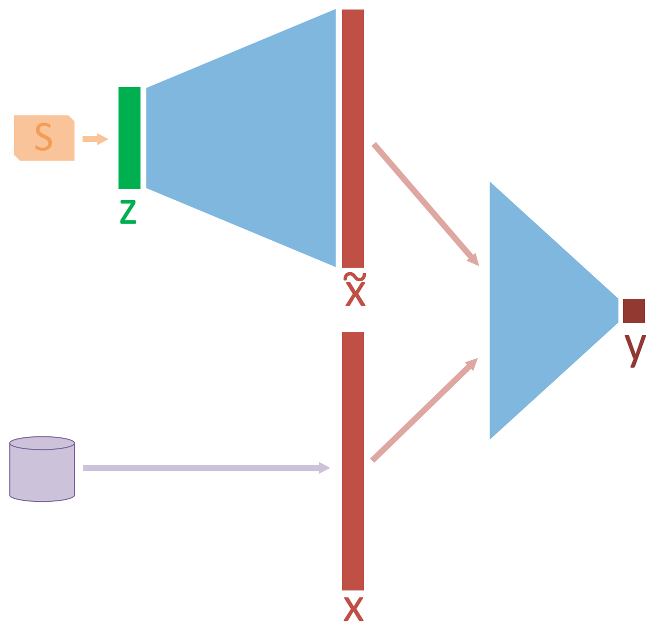
C-VTON

Fele *et al.*, 2022

- Context-Driven Image-Based Virtual Try-On Network
  - geometric matching procedure that aligns the target clothing with the person‵s pose
  - Image generator that utilizes contextual information to synthesize the final try-on result

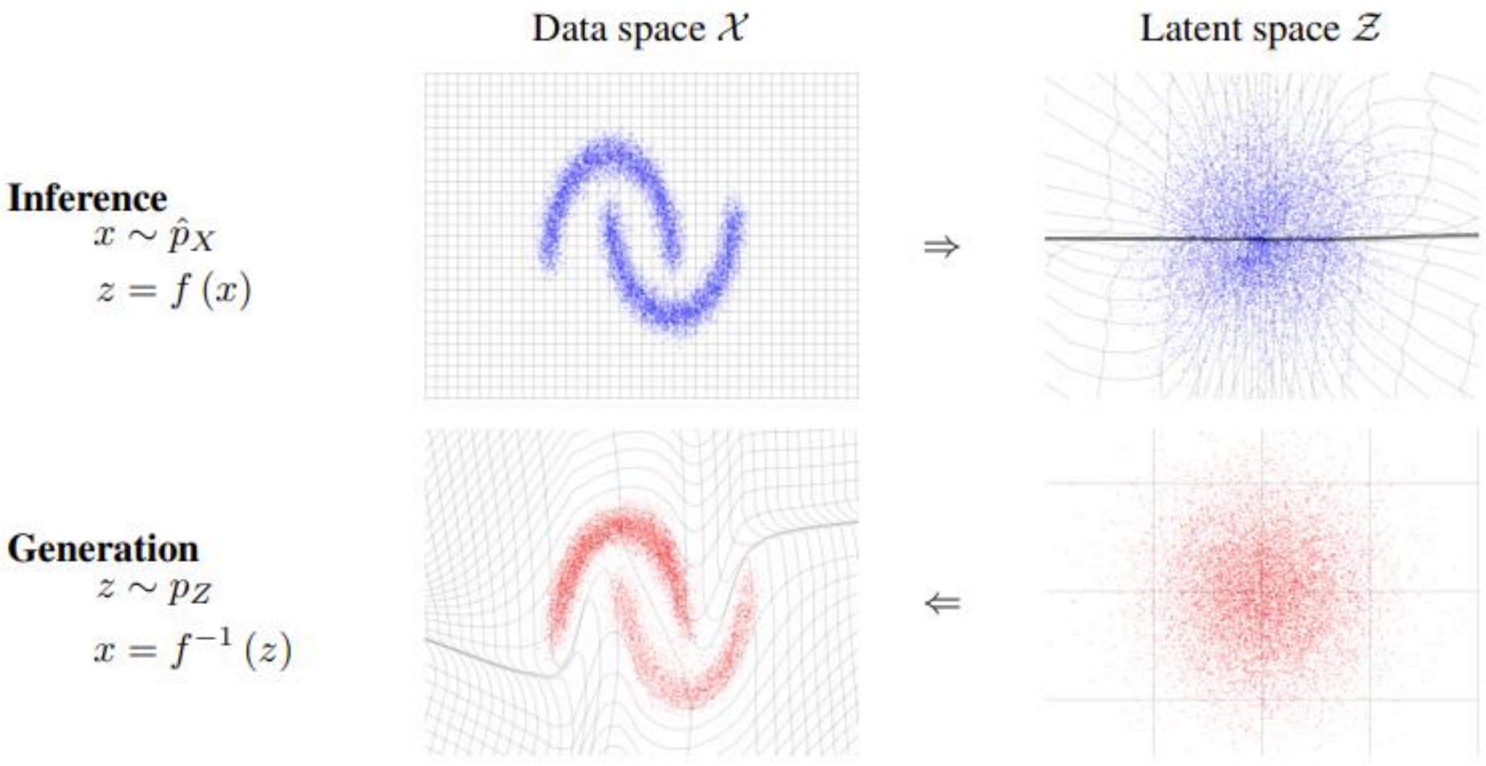
GAN recap

- GANs are a type of generative model in machine learning.
- Consist of two components: a generator and a discriminator.
- Generator generates synthetic data samples from random noise.
- Discriminator learns to distinguish between real and generated data.
- The generator and discriminator are trained simultaneously in a competitive setting.
- The objective is to optimize both networks to improve the quality of generated samples.
- GANs can generate realistic data samples that resemble the training data.
- They are widely used for image synthesis, such as generating photorealistic images.
- GANs have applications in various domains, including computer vision and creative AI.
- GANs have challenges such as training instability and mode collapse.



Normalizing flows

- Main idea: find an invertible function that transforms a complex data distribution to a latent Gaussian distribution
  => sample using the inverse of the obtained function!



Dinh *et al., 2017*

Normalizing flows - math

- Bijective function: $f : X \to Z$
- Change of variable formula:

$$p_X(x) = p_Z(f(x)) \left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|$$

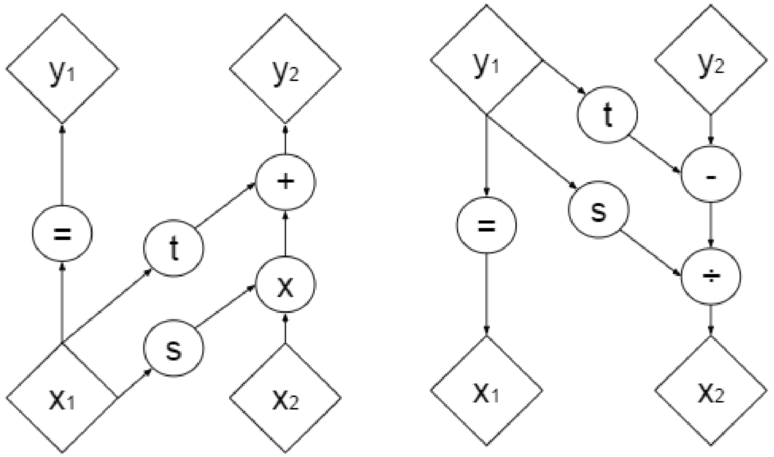
$$\log\left(p_X(x)\right) = \log\Big(p_Z(f(x))\Big) + \log\left(\left|\det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|\right)$$

- Coupling layers:

$$y_{1:d} = x_{1:d}$$

$$y_{d+1:D} = x_{d+1:D} \odot \exp\big(s(x_{1:d})\big) + t(x_{1:d})$$

- Jacobian: $$\frac{\partial y}{\partial x^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x^T_{1:d}} & \operatorname{diag}\left(\exp\left[s\left(x_{1:d}\right)\right]\right) \end{bmatrix}$$
- Determinant: $\left[\sum_j s\left(x_{1:d}\right)_j\right]$
- Inverse: $$\begin{cases} x_{1:d} & = y_{1:d} \\ x_{d+1:D} & = \big(y_{d+1:D} - t(y_{1:d})\big) \odot \exp\big(-s(y_{1:d})\big) \end{cases}$$



Dinh *et al., 2017*

Normalizing flows - math

- Partitioning:

$$y = b \odot x + (1-b) \odot \Big(x \odot \exp\big(s(b \odot x)\big) + t(b \odot x)\Big)$$

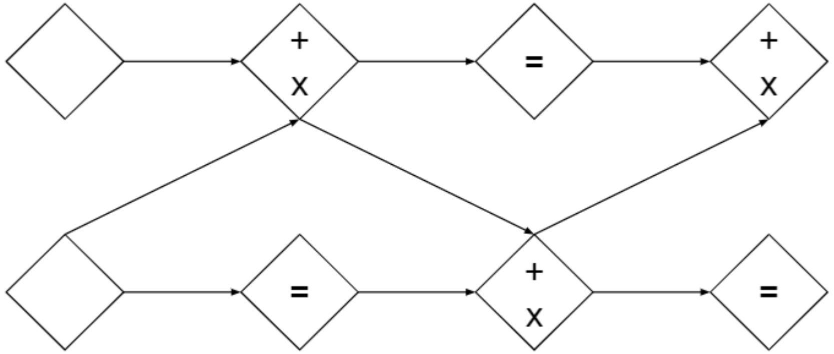
[Dinh *et al.*, 2017]

- Combining coupling layers:

$$\frac{\partial (f_b \circ f_a)}{\partial x_a^T}(x_a) = \frac{\partial f_a}{\partial x_a^T}(x_a) \cdot \frac{\partial f_b}{\partial x_b^T}(x_b = f_a(x_a))$$

$$\det(A \cdot B) = \det(A)\det(B)$$

$$(f_b \circ f_a)^{-1} = f_a^{-1} \circ f_b^{-1}$$



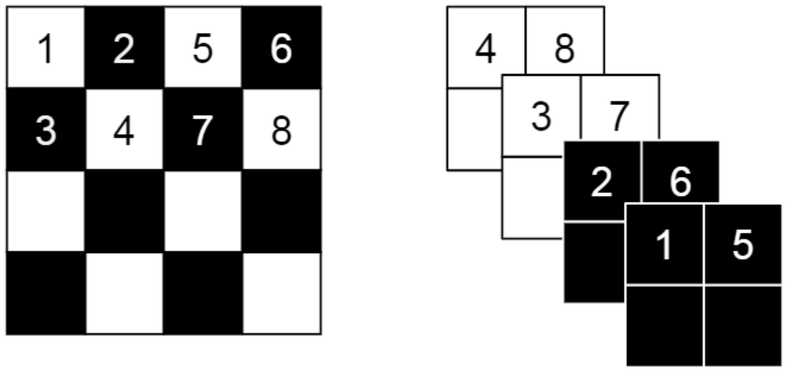
- Multi-scale architecture:

$$h^{(0)} = x$$

$$(z^{(i+1)}, h^{(i+1)}) = f^{(i+1)}(h^{(i)})$$

$$z^{(L)} = f^{(L)}(h^{(L-1)})$$

$$z = (z^{(1)}, \dots, z^{(L)})$$



Real NVP

Dinh *et al.*, 2017

- Density estimation using Real NVP
- Real valued on-volume preserving transformations
- Stably invertible
- Learnable transformations
- Exact log-likelihood computation
- Exact sampling
- Efficient sampling
- Exact inference
- Efficient inference
- Interpretable latent space

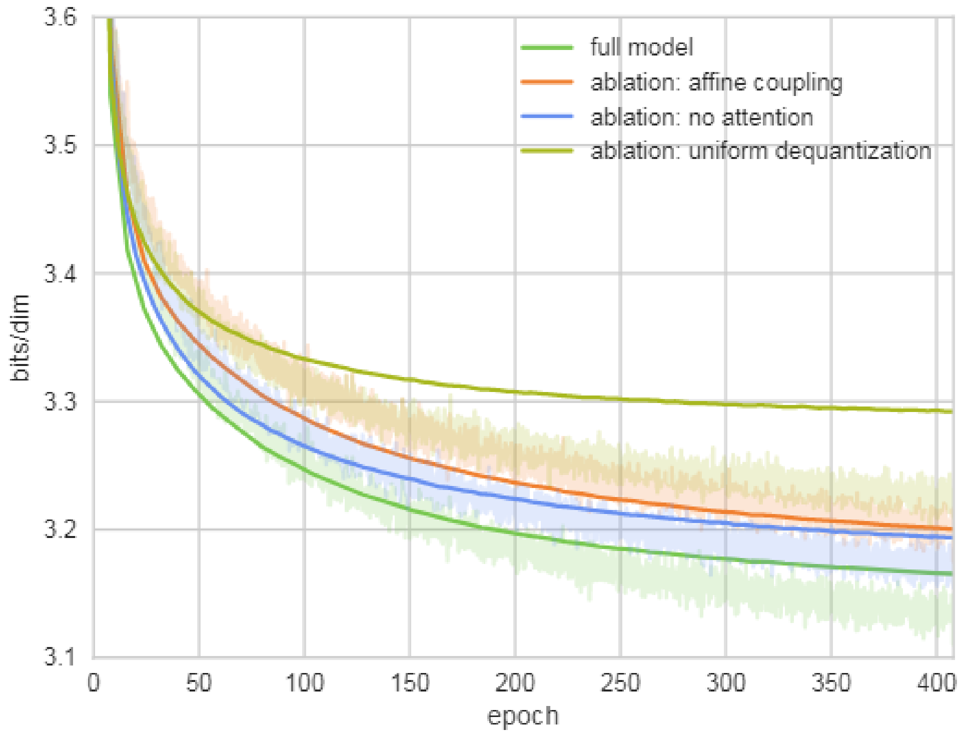
Flow++

Ho et al., 2019

- Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design
  - variational flow-based dequantization instead of uniform dequantization
  - logistic mixture CDF coupling flows

$$\mathrm{MixLogCDF}(x; \boldsymbol{\pi}, \boldsymbol{\mu}, \mathbf{s}) := \sum_{i=1}^{K} \pi_i \sigma\left((x - \mu_i) \cdot \exp(-s_i)\right)$$

$$\mathbf{y}_1 = \mathbf{x}_1,$$
$$\mathbf{y}_2 = \sigma^{-1}\left(\mathrm{MixLogCDF}(\mathbf{x}_2; \boldsymbol{\pi}_\theta(\mathbf{x}_1), \boldsymbol{\mu}_\theta(\mathbf{x}_1), \mathbf{s}_\theta(\mathbf{x}_1))\right) \cdot \exp(\mathbf{a}_\theta(\mathbf{x}_1)) + \mathbf{b}_\theta(\mathbf{x}_1)$$

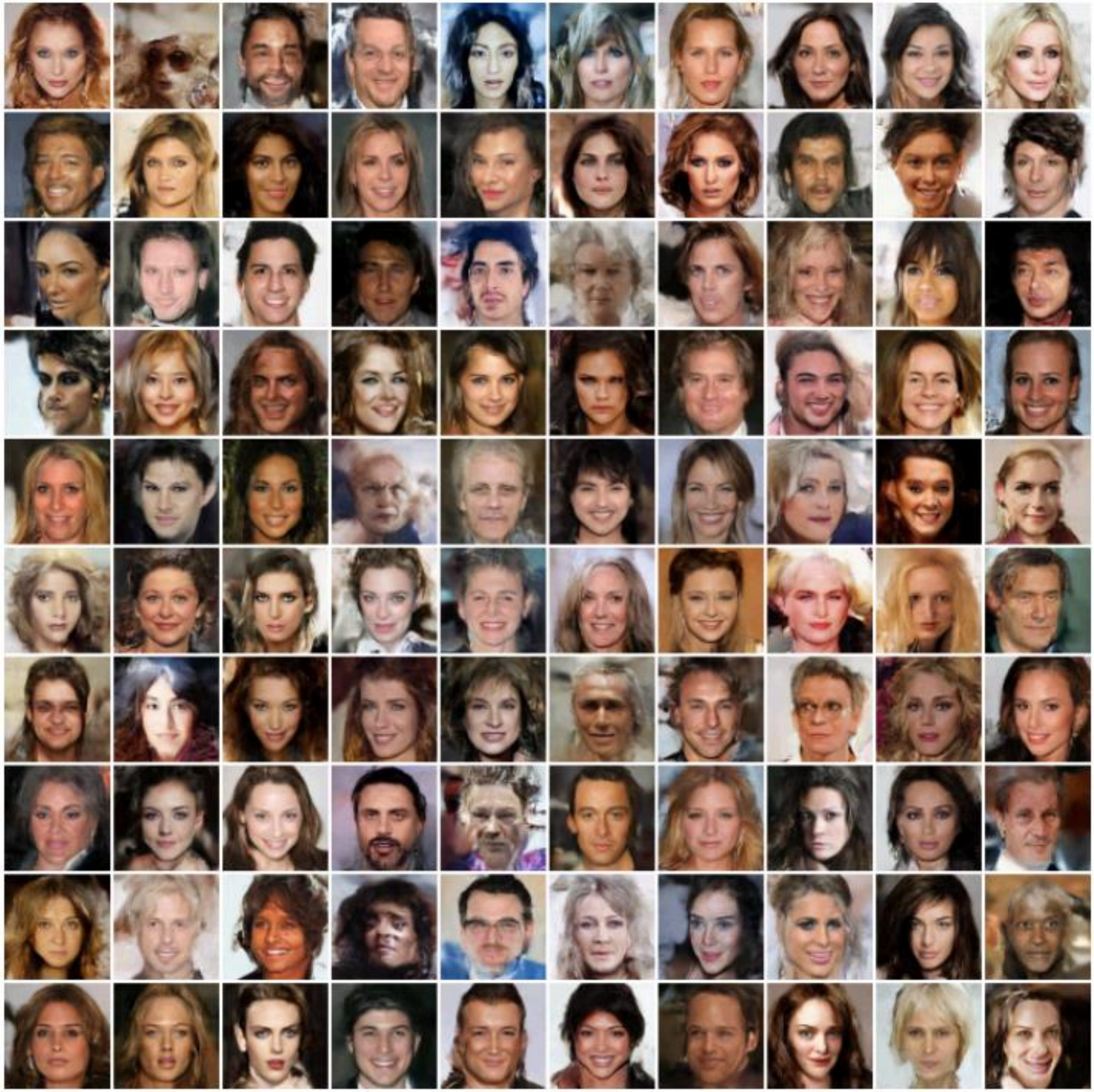


  - self-attention in the conditioning networks of coupling layers

$$\mathrm{Conv} = \mathrm{Input} \to \mathrm{Nonlinearity} \to \mathrm{Conv}_{3\times3} \to \mathrm{Nonlinearity} \to \mathrm{Gate}$$
$$\mathrm{Attn} = \mathrm{Input} \to \mathrm{Conv}_{1\times1} \to \mathrm{MultiHeadSelfAttention} \to \mathrm{Gate}$$

Flow++ results

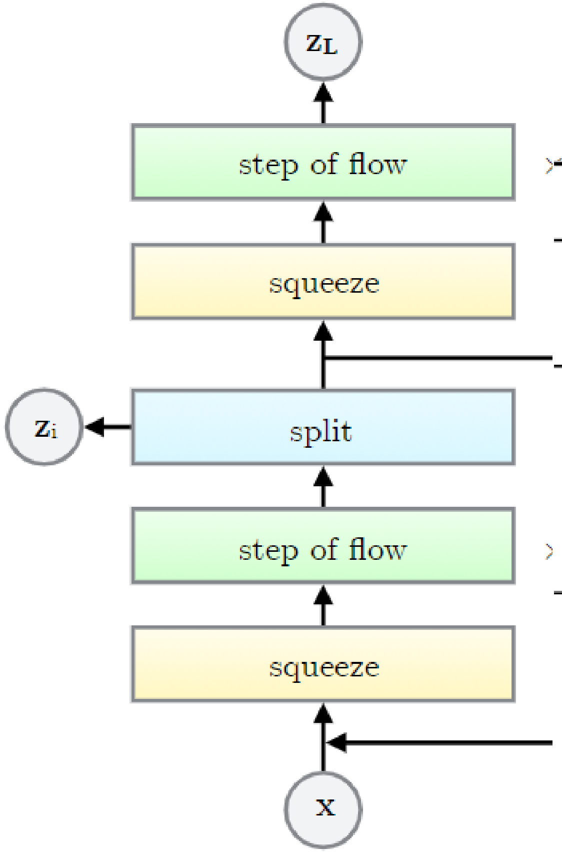
[Ho *et al.*, 2019]



Glow

Kingma & Dhariwal, 2018

- Generative Flow with Invertible 1×1 Convolutions
- Efficient realistic-looking synthesis and manipulation of large images with the plain log-likelihood objective



Description	Function	Reverse Function	Log-determinant
Actnorm.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{s} \odot \mathbf{x}_{i,j} + \mathbf{b}$	$\forall i,j : \mathbf{x}_{i,j} = (\mathbf{y}_{i,j} - \mathbf{b})/\mathbf{s}$	$h \cdot w \cdot \texttt{sum}(\log \lvert\mathbf{s}\rvert)$
Invertible 1×1 convolution. $\mathbf{W} : [c \times c]$.	$\forall i,j : \mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j}$	$\forall i,j : \mathbf{x}_{i,j} = \mathbf{W}^{-1}\mathbf{y}_{i,j}$	$h \cdot w \cdot \log \lvert\det(\mathbf{W})\rvert$ or $h \cdot w \cdot \texttt{sum}(\log \lvert\mathbf{s}\rvert)$
Affine coupling layer.	$\mathbf{x}_a, \mathbf{x}_b = \texttt{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \texttt{NN}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \texttt{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \texttt{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \texttt{NN}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \texttt{concat}(\mathbf{x}_a, \mathbf{x}_b)$	$\texttt{sum}(\log(\lvert\mathbf{s}\rvert))$

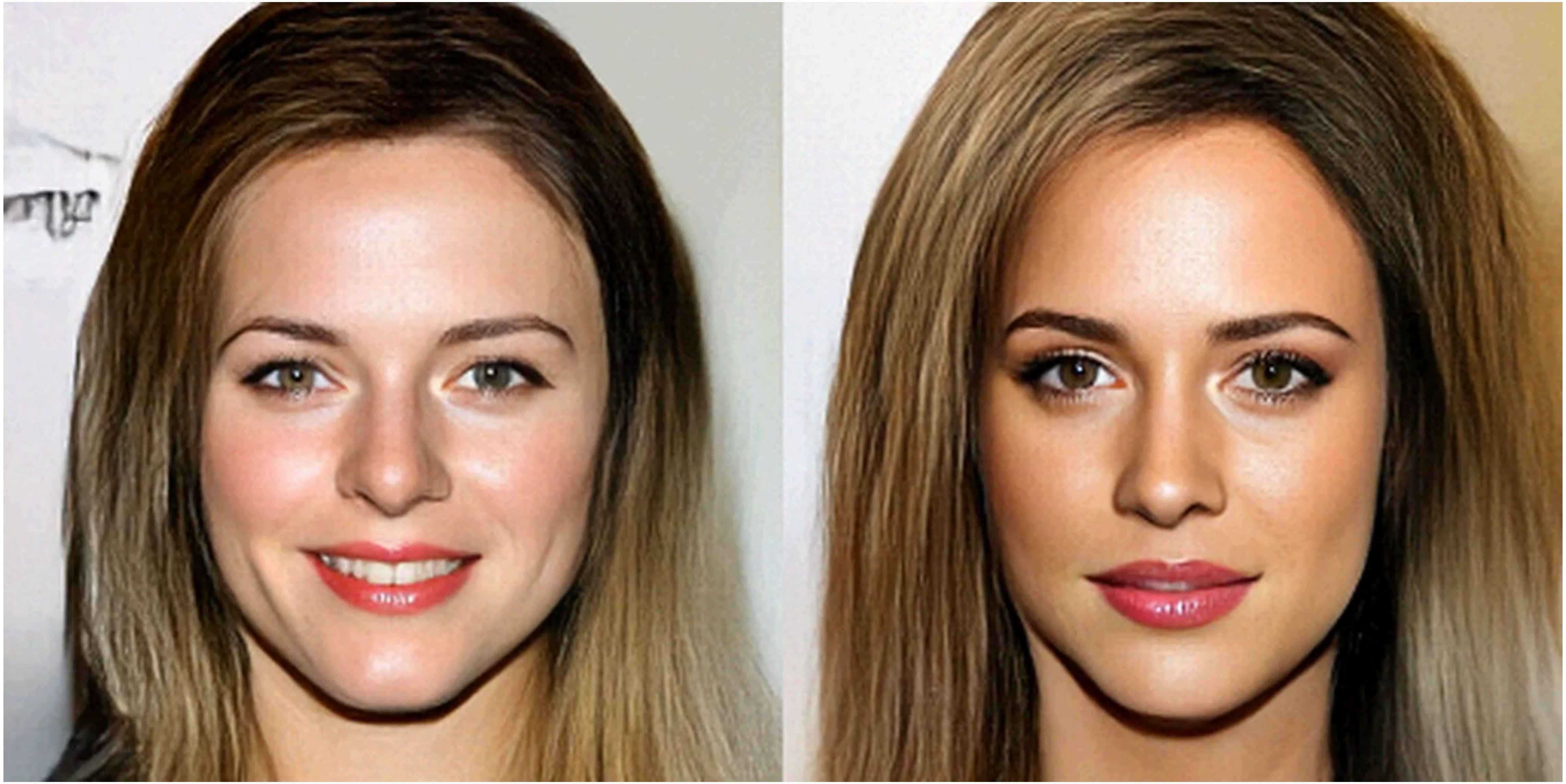
Glow results

(a) Smiling (b) Pale Skin

(c) Blond Hair (d) Narrow Eyes

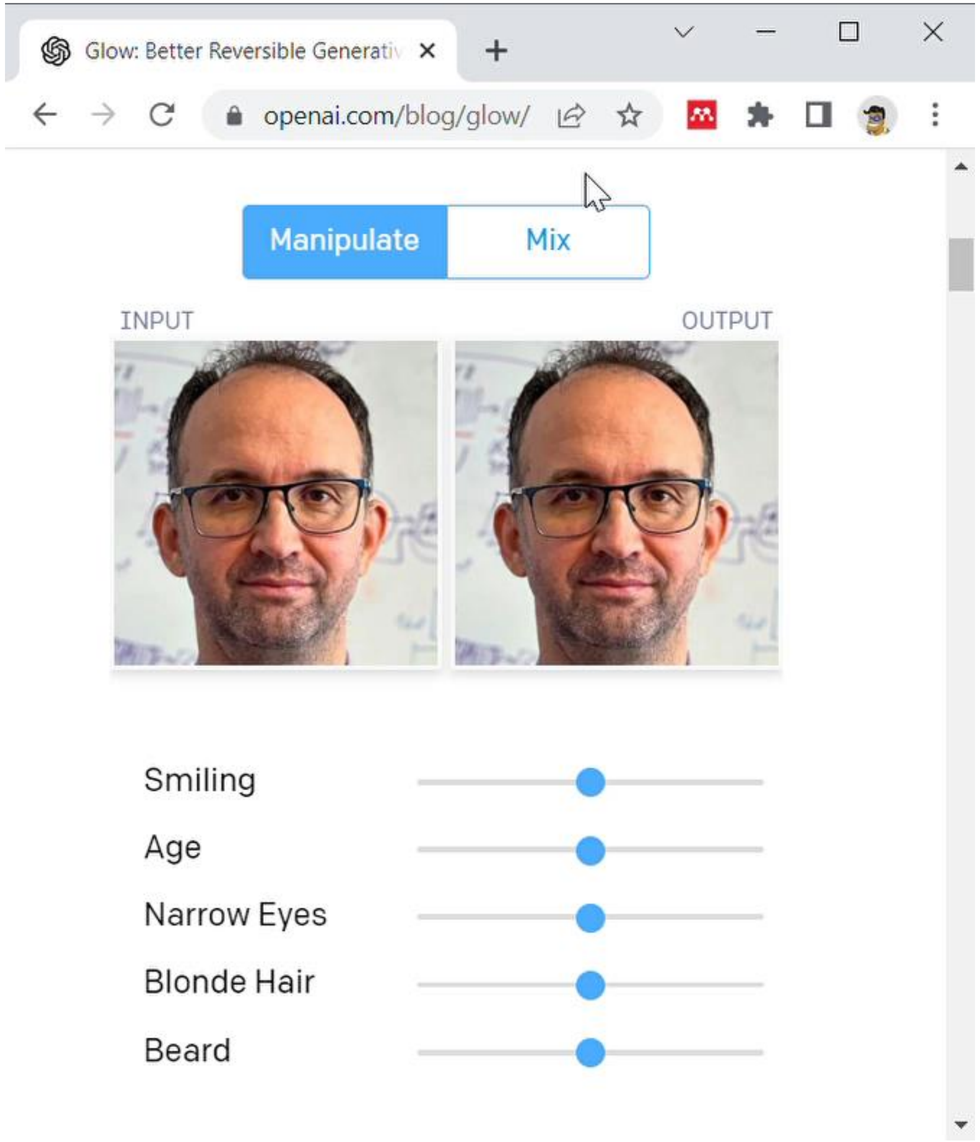
(e) Young (f) Male

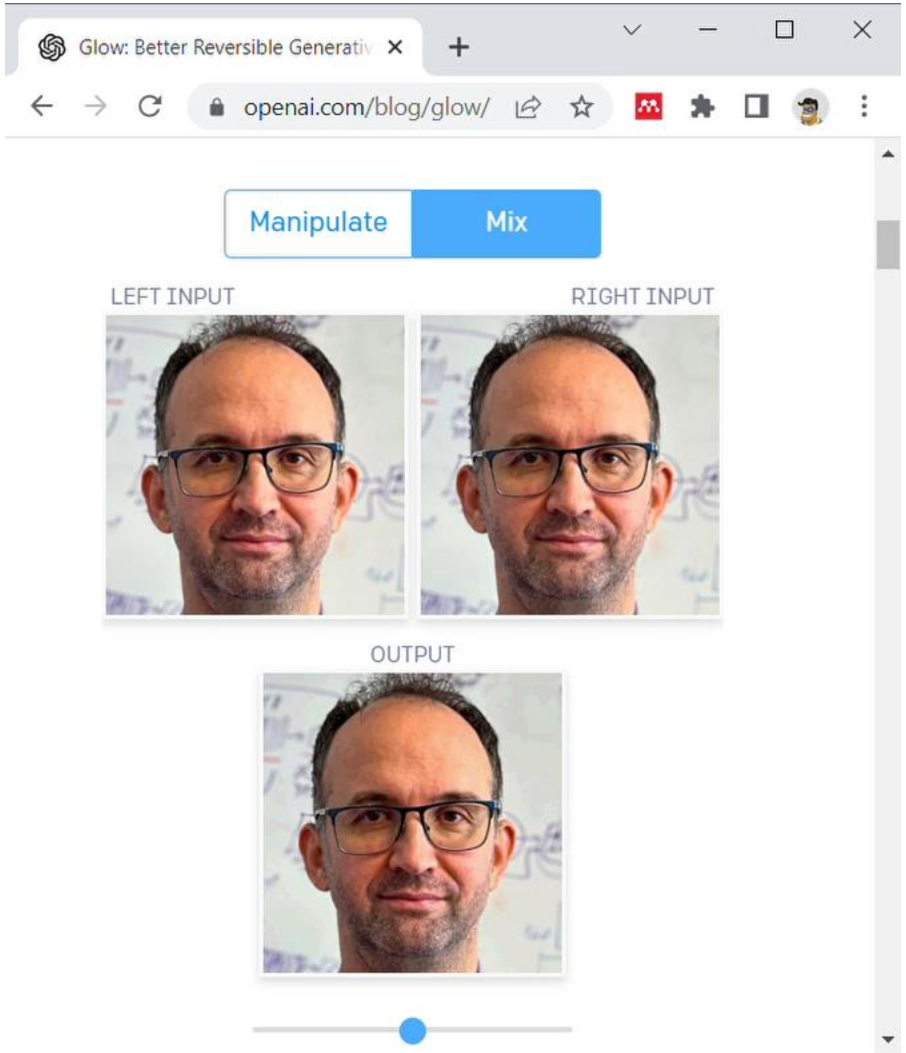
Glow results



https://openai.com/blog/glow/

Glow results





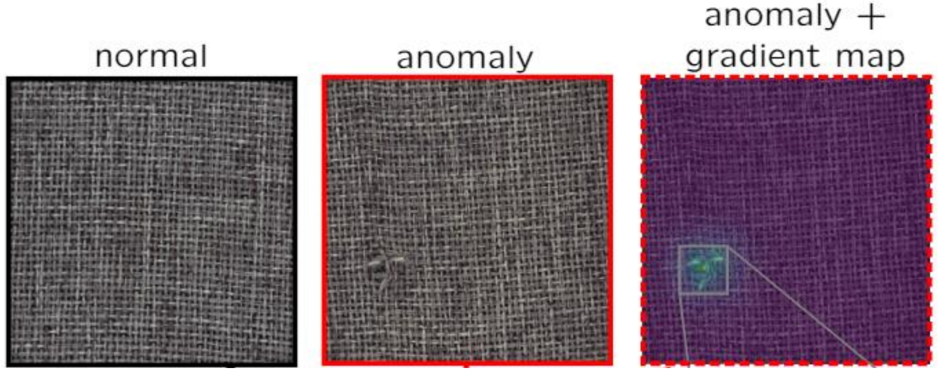
https://openai.com/blog/glow/

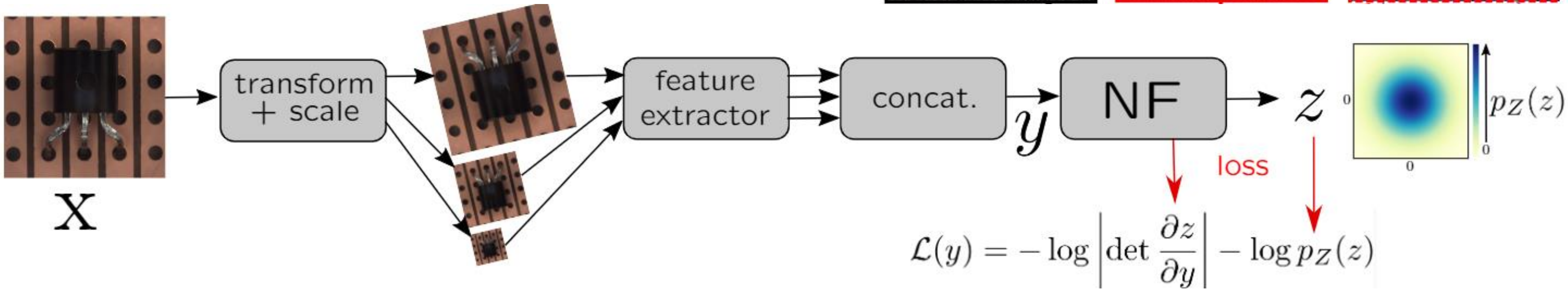
DifferNet

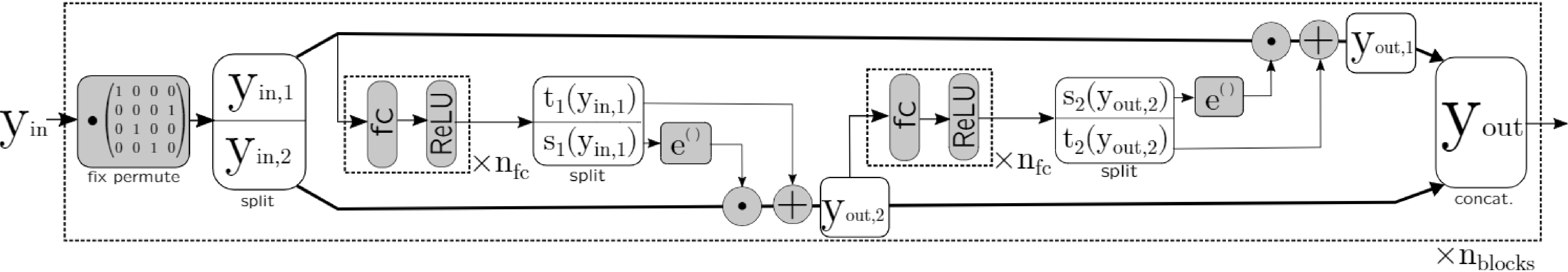
Rudolph et al., 2020

- Same Same But DifferNet: Semi-Supervised Defect Detection with Normalizing Flows

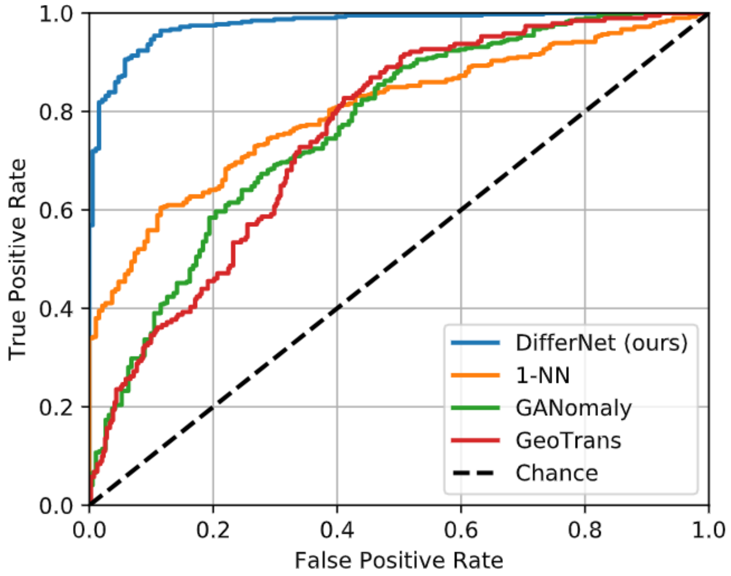
$$\tau(x) = \mathbb{E}_{T_i \in \mathcal{T}} \left[-\log p_Z(f_{\mathrm{NF}}(f_{\mathrm{ex}}(T_i(x)))) \right]$$

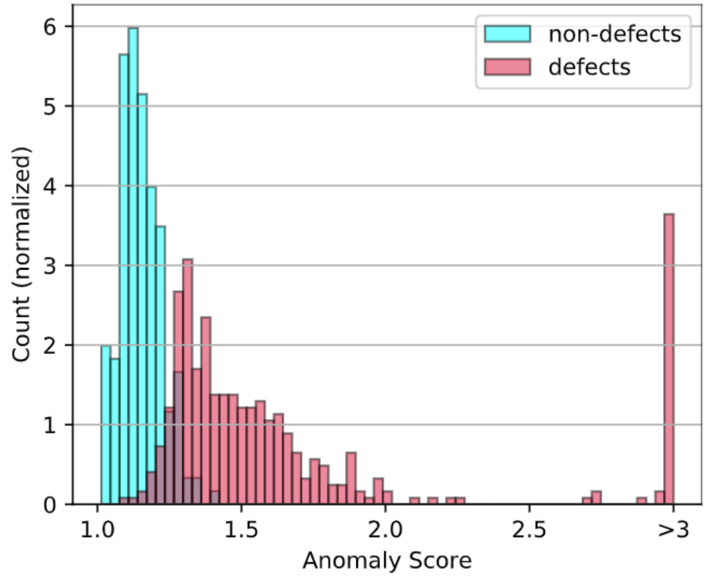






DifferNet results

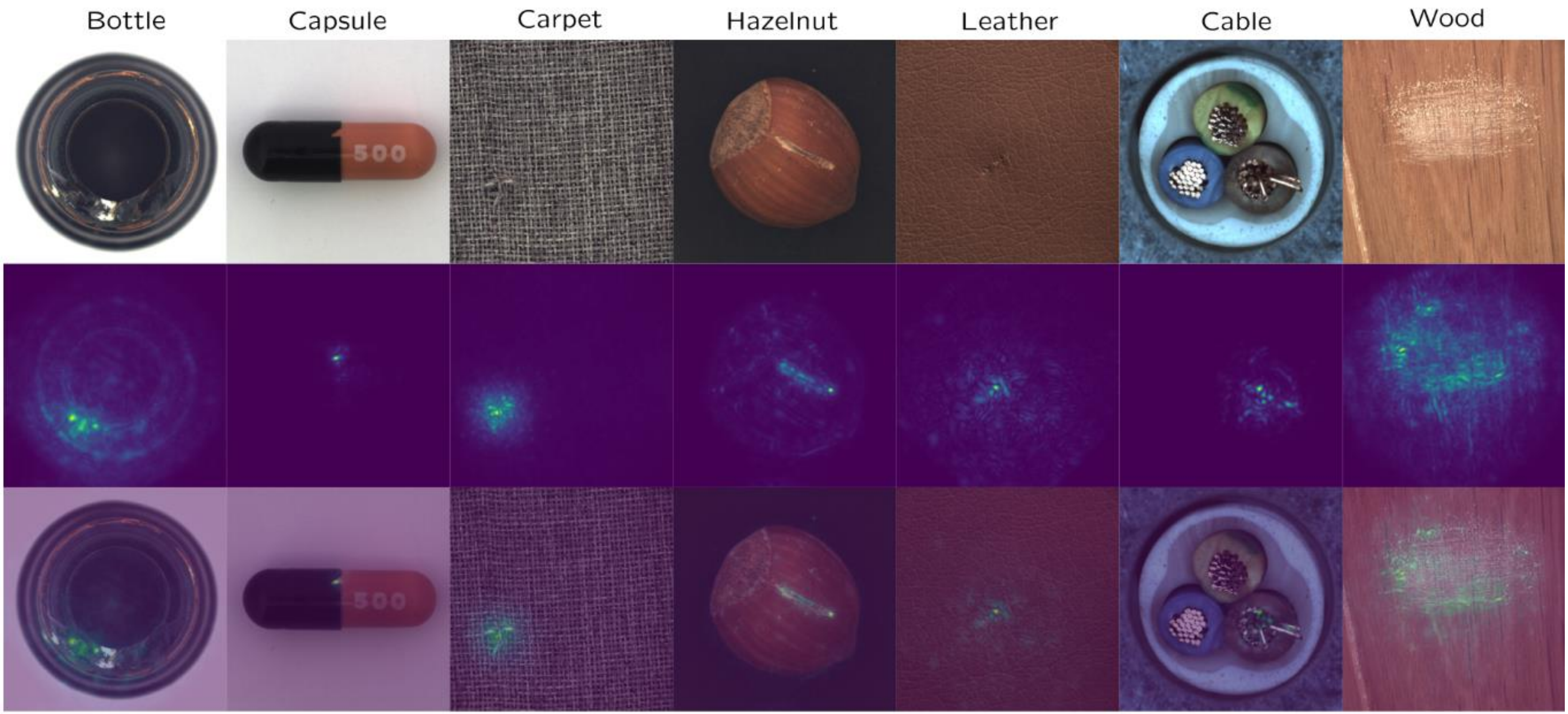




	Category	GeoTrans [10]	GANomaly [1]	DSEBM [30]	OCSVM [2]	1-NN [21]	**DifferNet (ours)**	DifferNet (16 shots)
Textures	Grid	61.9	70.8	71.7	41.0	55.7	**84.0**	65.8
	Leather	84.1	84.2	41.6	88.0	90.3	**97.1**	92.9
	Tile	41.7	79.4	69.0	87.6	96.9	**99.4**	98.9
	Carpet	43.7	69.9	41.3	62.7	81.1	**92.9**	77.0
	Wood	61.1	83.4	95.2	95.3	93.4	**99.8**	99.2
Objects	Bottle	74.4	89.2	81.8	**99.0**	98.7	**99,0**	98.5
	Capsule	67.0	73.2	59.4	54.4	71.1	**86.9**	61.4
	Pill	63.0	74.3	80.6	72.9	83.7	**88.8**	65.1
	Transistor	86.9	79.2	74.1	56.7	75.6	**91.1**	76.6
	Zipper	82.0	74.5	58.4	51.7	88.6	**95.1**	88.3
	Cable	78.3	75.7	68.5	80.3	88.5	**95.9**	86.4
	Hazelnut	35.9	78.5	76.2	91.1	97.9	**99.3**	97.3
	Metal Nut	81.3	70.0	67.9	61.1	76.7	**96.1**	77.7
	Screw	50.0	74.6	**99.9**	74.7	67.0	96.3	75.9
	Toothbrush	97.2	65.3	78.1	61.9	91.9	**98.6**	92.3
	Average	67.2	76.2	70.9	71.9	83.9	**94.9**	87.3

Config.	A	B	C	D	E	F	G
multi-scale	✗	✗	✓	✓	✓	✓	✓
train transf.	✗	✓	✗	✓	✓	✓	✓
# test transf.	1	64	1	1	4	16	64
AUROC [%]	84.4	90.2	86.6	86.5	91.6	94.1	**94.9**

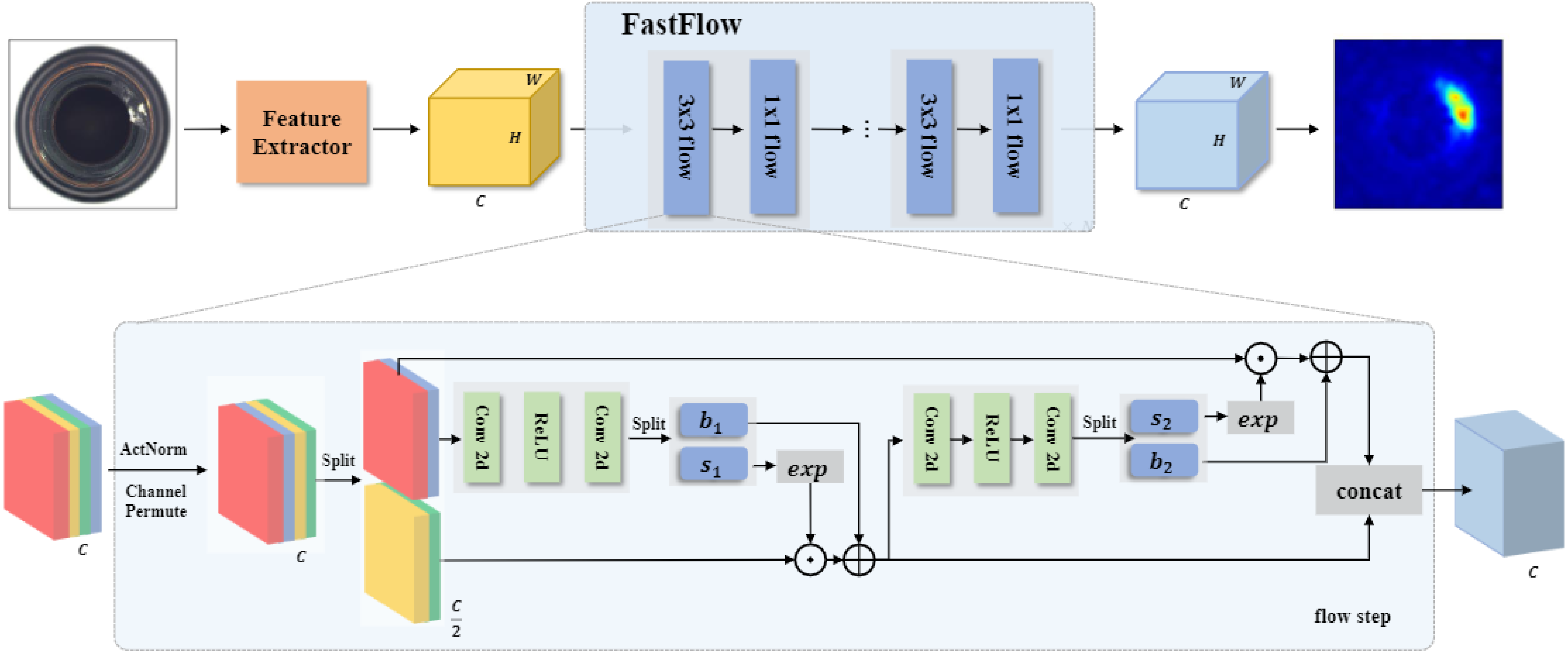
DifferNet results



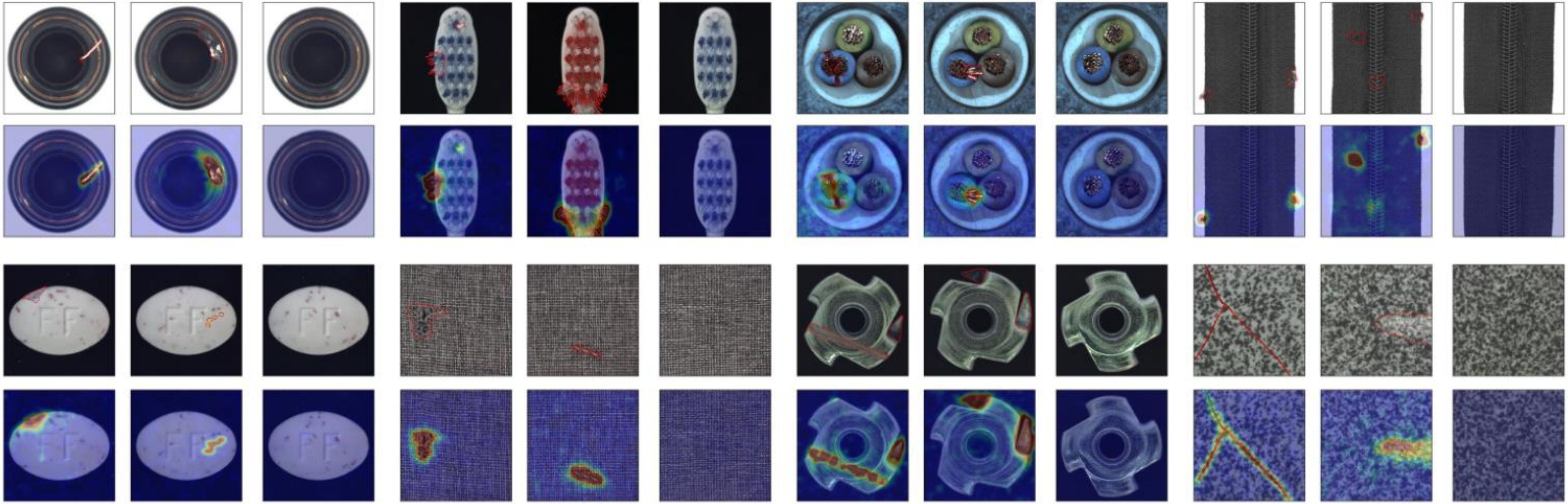
FastFlow

Yu et al., 2021

- Unsupervised Anomaly Detection and Localization via 2D Normalizing Flows



FastFlow results



AUCROC	(92.1,95.7)	(96.2,96.5)	(94.9,-)	(97.9,97.5)	(97.1,96.0)	(99.1,98.1)	(98.3,**98.6**)	(**99.4**,98.5)

AD on MVTech AD benchmark leaderboard

1	**PatchCore**	99.6	98.4	95.0	✓	Towards Total Industrial Ano Detection
2	**Fastflow**	99.4	98.5		✓	FastFlow: Uns Anomaly Dete Localization vi Normalizing F
3	**DRAEM+SSPCAB**	98.9	97.2		×	Self-Supervis Convolutional Block for Ano Detection
4	**CS-Flow**	98.7			✓	Fully Convolu Scale-Flows f based Defect
5	**Reverse Distillation**	98.5	97.8	93.9	✓	Anomaly Dete Reverse Distil One-Class Em
6	**OCR-GAN**	98.3				
7	**CFLOW-AD**	98.26	98.62			
8	**DRAEM**	98.0	97.3			
9	**PaDiM**	97.9	97.5			
10	**FYD**	97.7	98.2			

https://paperswithcode.com/sota/anomaly-detection-on-mvtec-ad
14 May 2022

Normalising flows recap

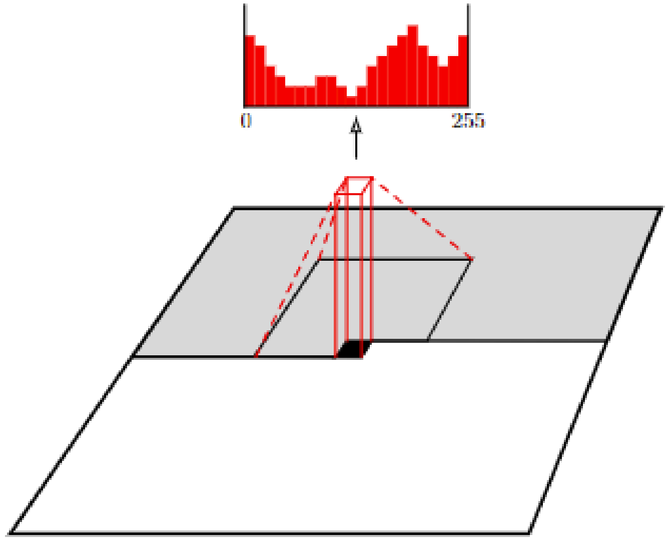
- Normalizing flows transform a simple base distribution into a complex target distribution through invertible transformations.
- Density Estimation: Normalizing flows excel at modeling complex probability distributions and density estimation tasks.
- Change of Variables: They leverage the change of variables formula to compute the probability density function of the target distribution.
- Flexibility and Expressiveness: Normalizing flows can model multimodal distributions and varying correlation structures.
- Sampling and Generation: Efficient sampling from the target distribution is achieved by applying inverse transformations to samples from the base distribution.
- Inference and Latent Space Manipulation: They can perform inference tasks and allow meaningful manipulation of latent variables.
- Training and Optimization: Normalizing flows are trained by maximizing the likelihood of observed data through optimization techniques.

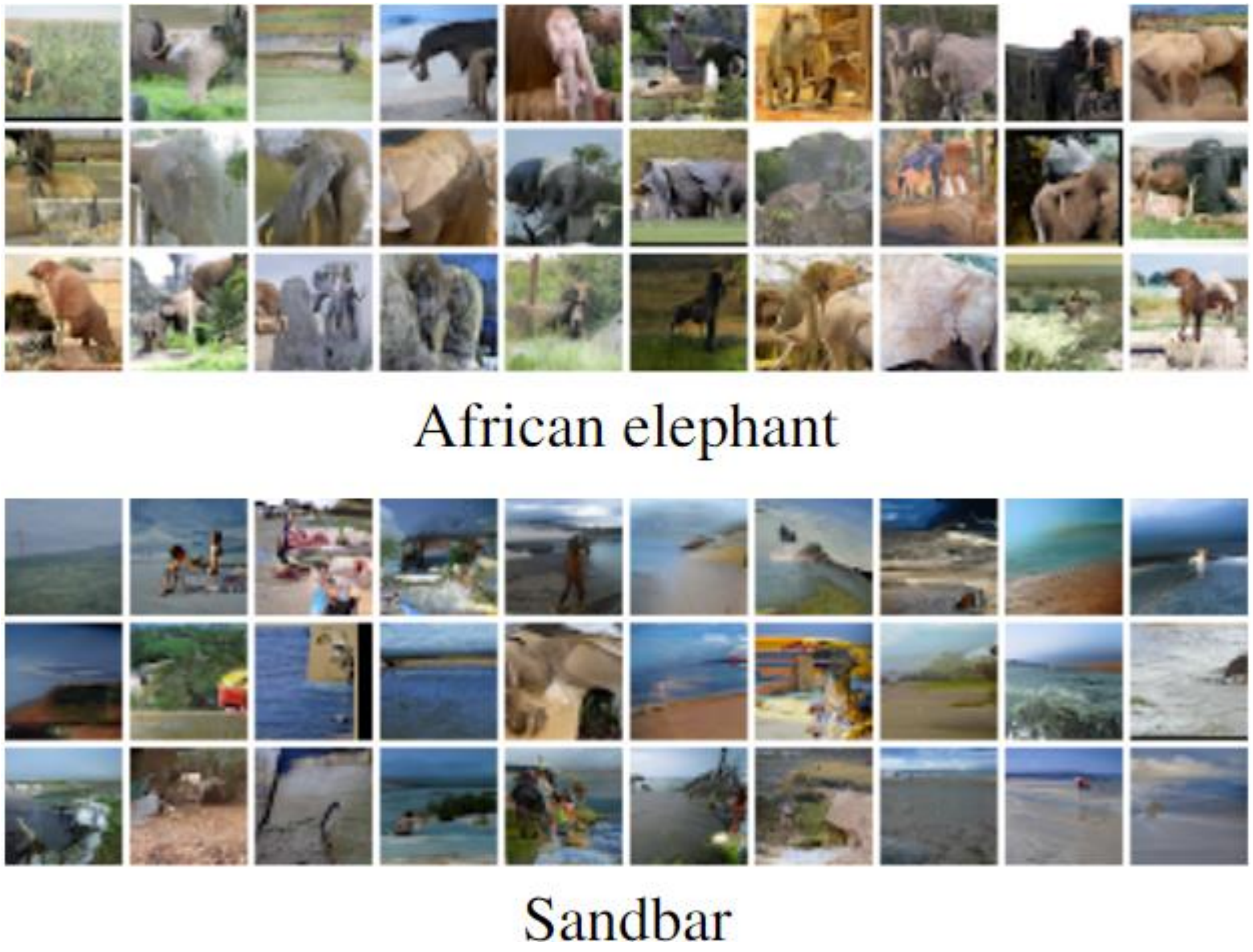
PixelCNN

Van der Oord et al., 2016

- Conditional Image Generation with PixelCNN Decoders
  - Autoregressive model
  - Sequential pixel generation

$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, ..., x_{i-1})$$





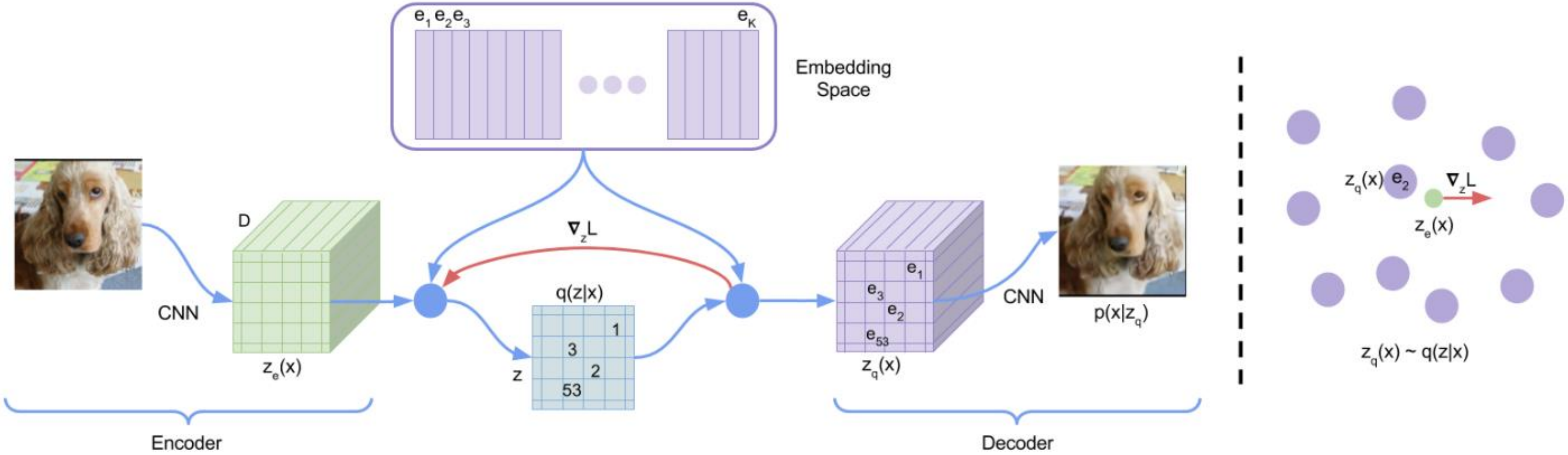
VQ-VAE

Van der Oord et al., 2017

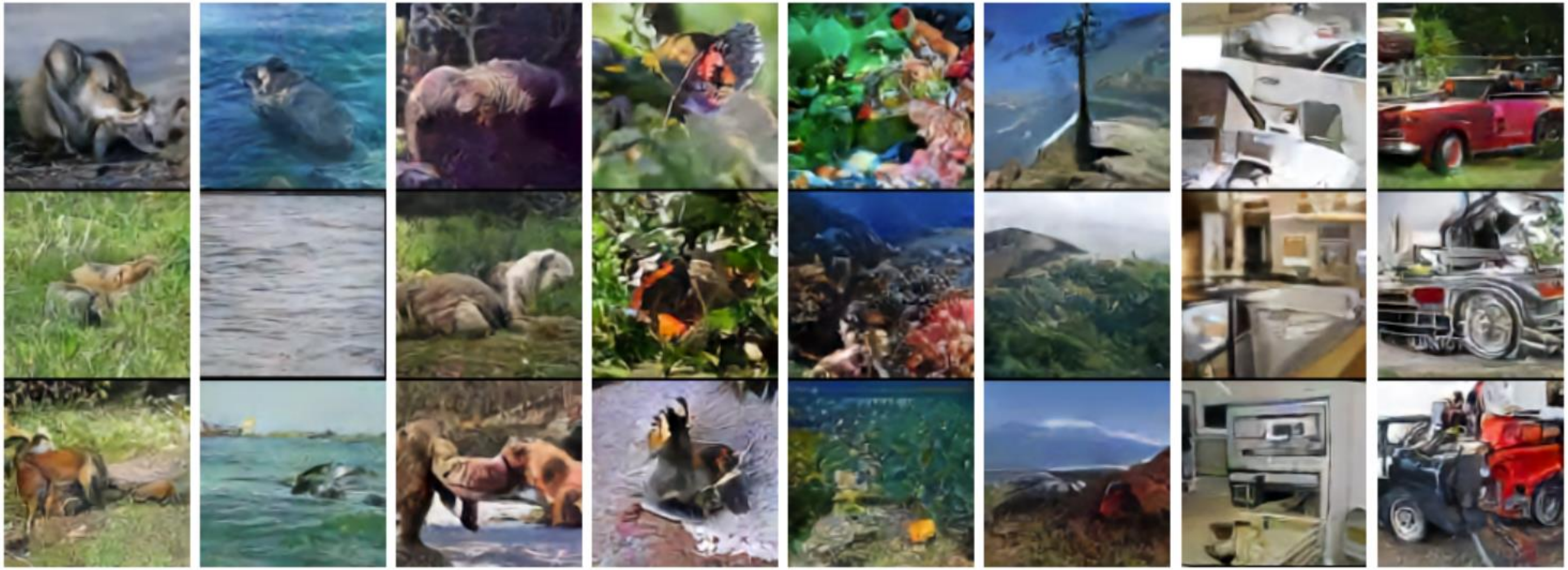
- Neural Discrete Representation Learning
- Discrete latent space – embeddings
  - Vector quantisation

$$q(z = k|x) = \begin{cases} 1 & \text{for } k = \text{argmin}_j \|z_e(x) - e_j\|_2 \\ 0 & \text{otherwise} \end{cases}$$

$$z_q(x) = e_k, \quad \text{where} \quad k = \text{argmin}_j \|z_e(x) - e_j\|_2$$

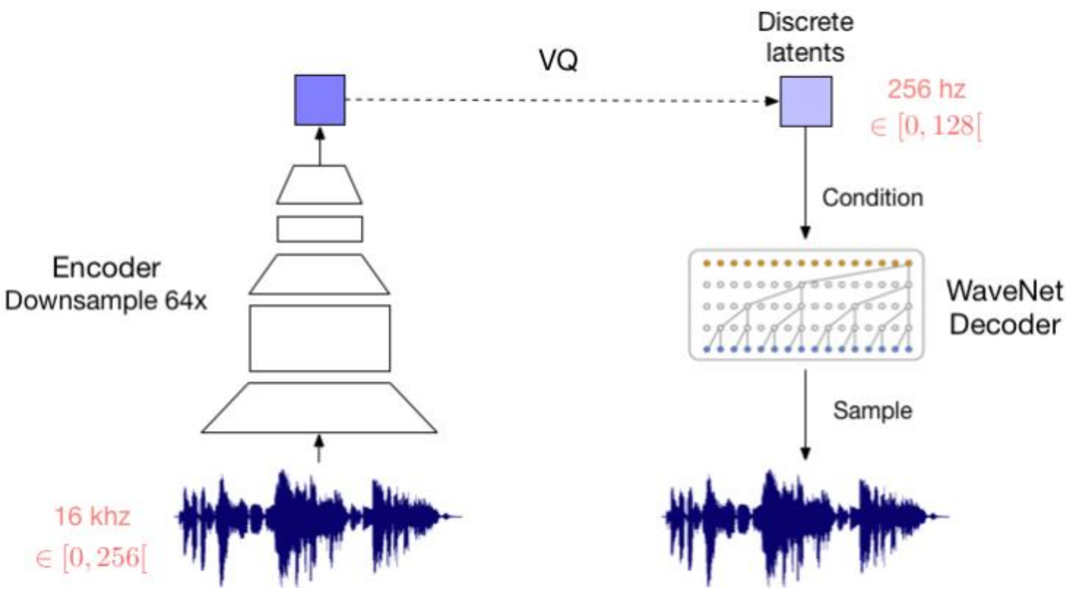


VQ-VAE results

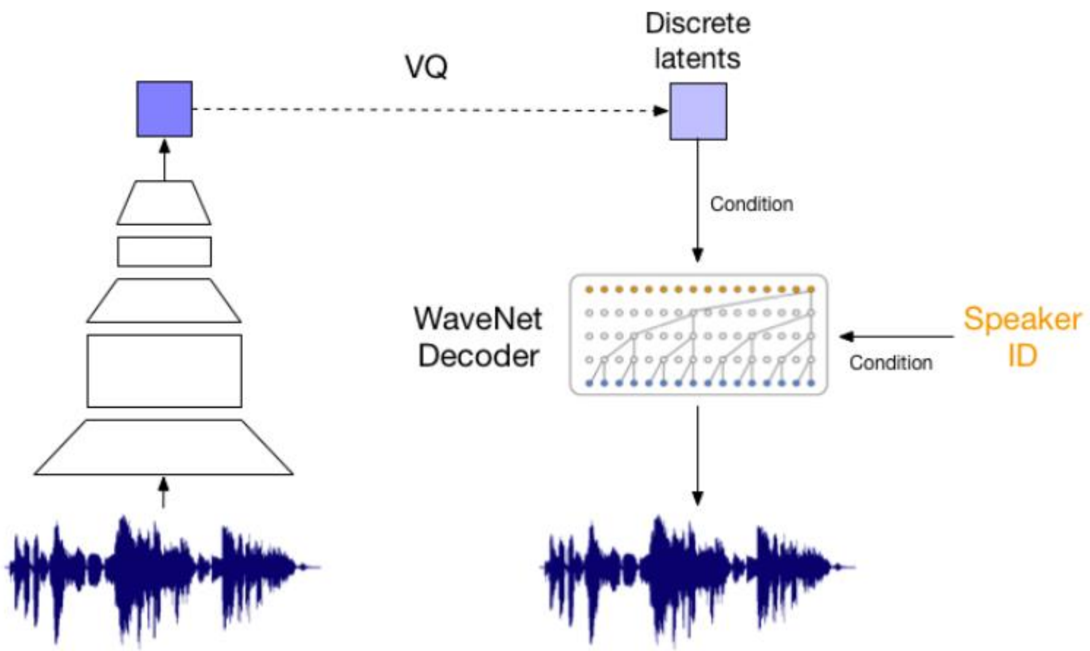


VQ-VAE audio

Voice Reconstruction



Voice Style-Transfer

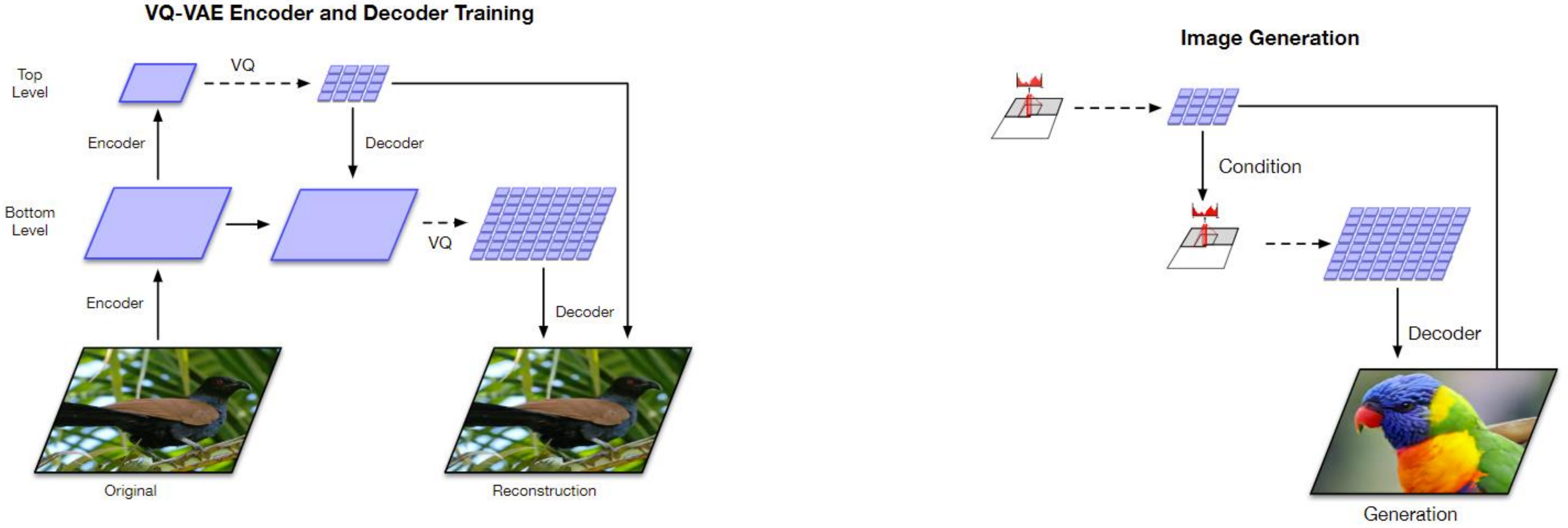


Samples from prior:

https://avdnoord.github.io/homepage/vqvae/

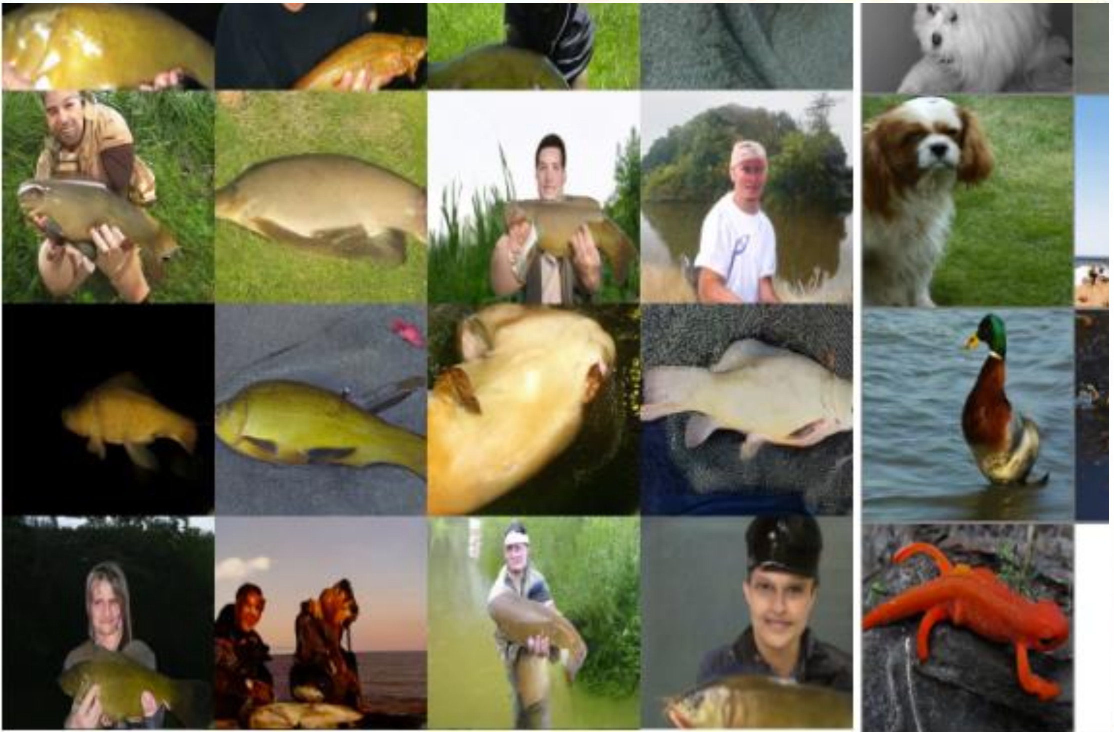
VQ-VAE2

- Generating Diverse High-Fidelity Images with VQ-VAE-2



h_{top} | $h_{\text{top}}, h_{\text{middle}}$ | $h_{\text{top}}, h_{\text{middle}}, h_{\text{bottom}}$ | Original

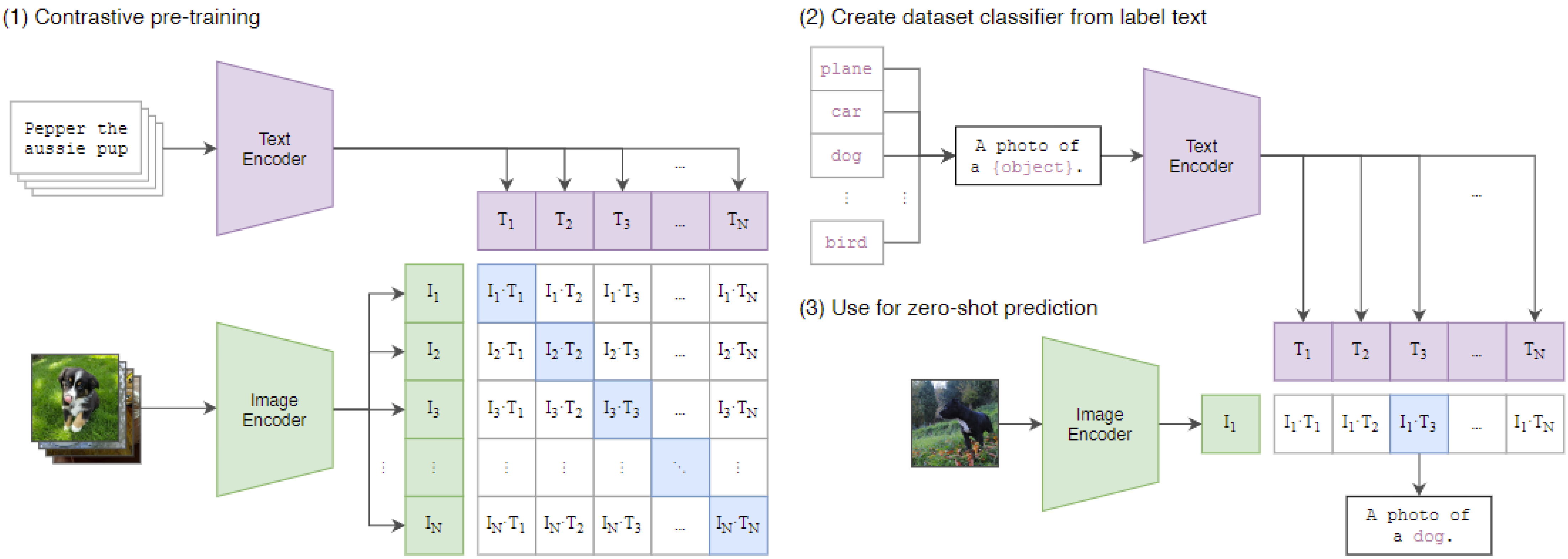
VQ-VAE2 results



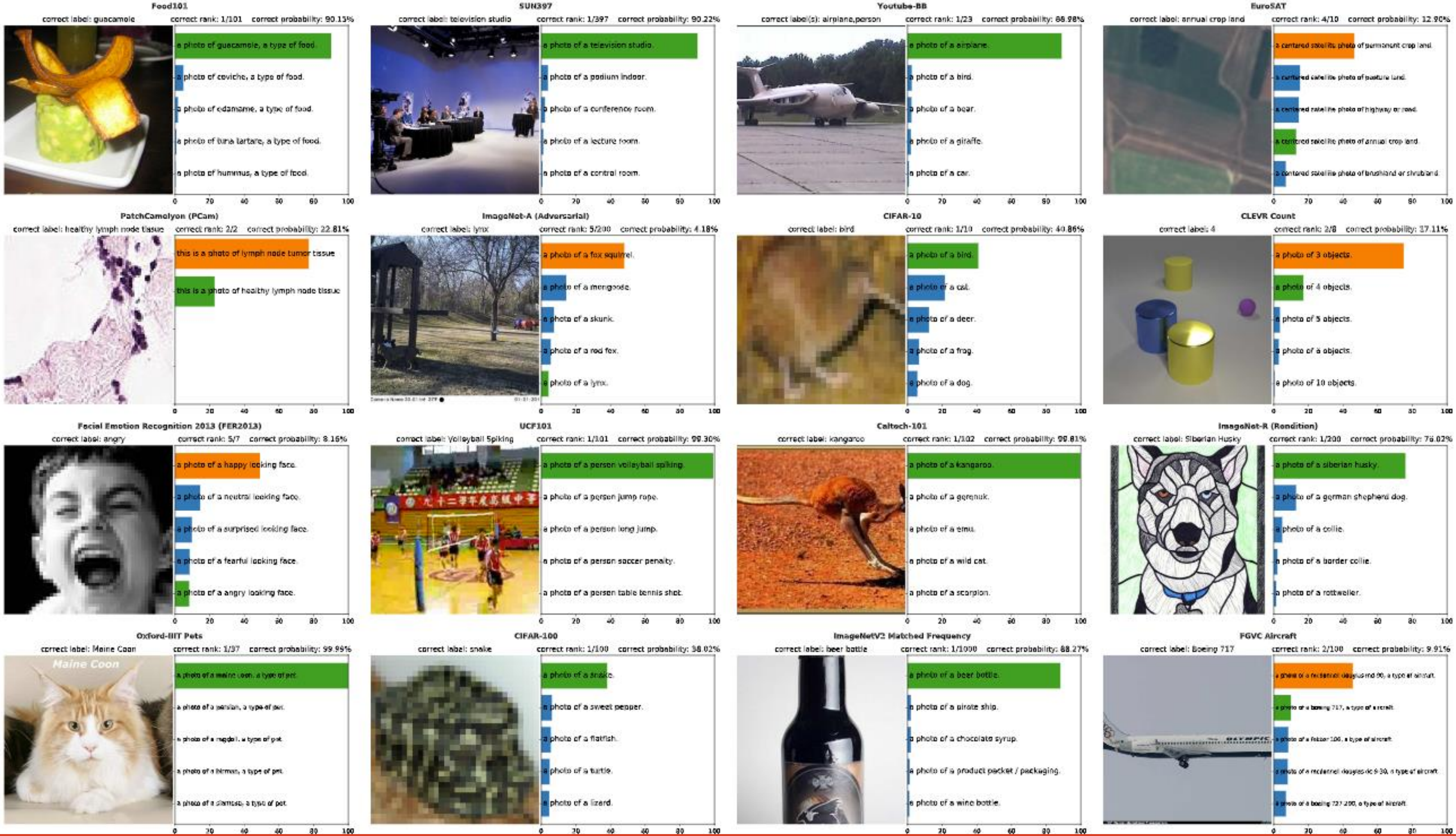
CLIP

[Radford et al., 2021]

- Learning Transferable Visual Models From Natural Language Supervision
- CLIP - Contrastive Language-Image Pre-Training
  - Pre-training task of predicting which caption goes with which image
  - 400 M (image, text) pairs

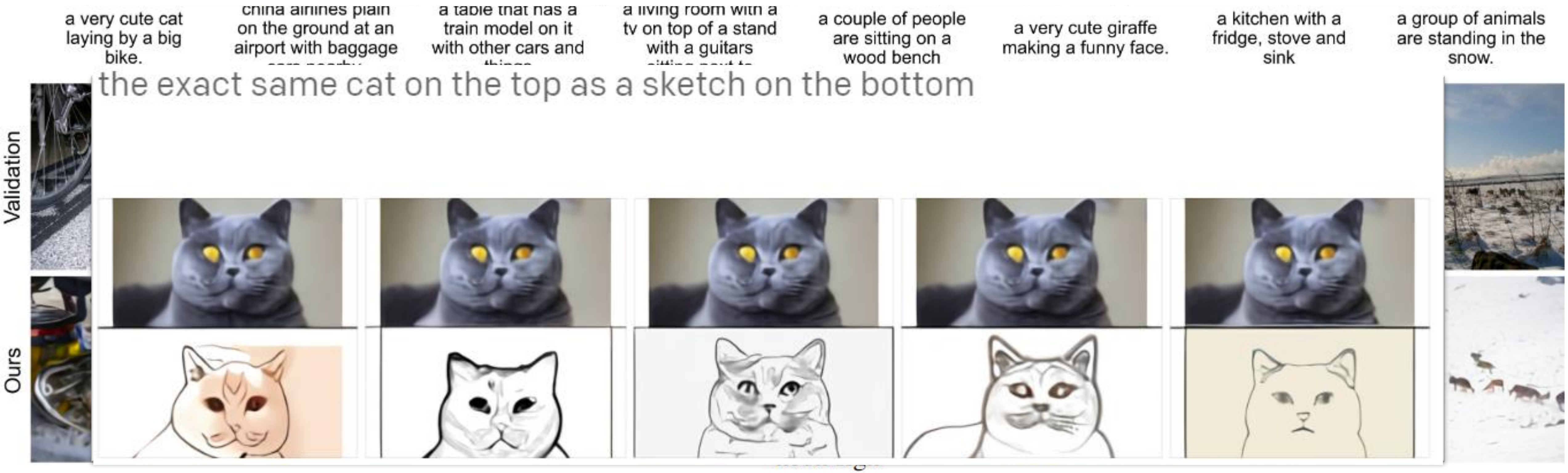


CLIP results



Ramesh et al., 2021

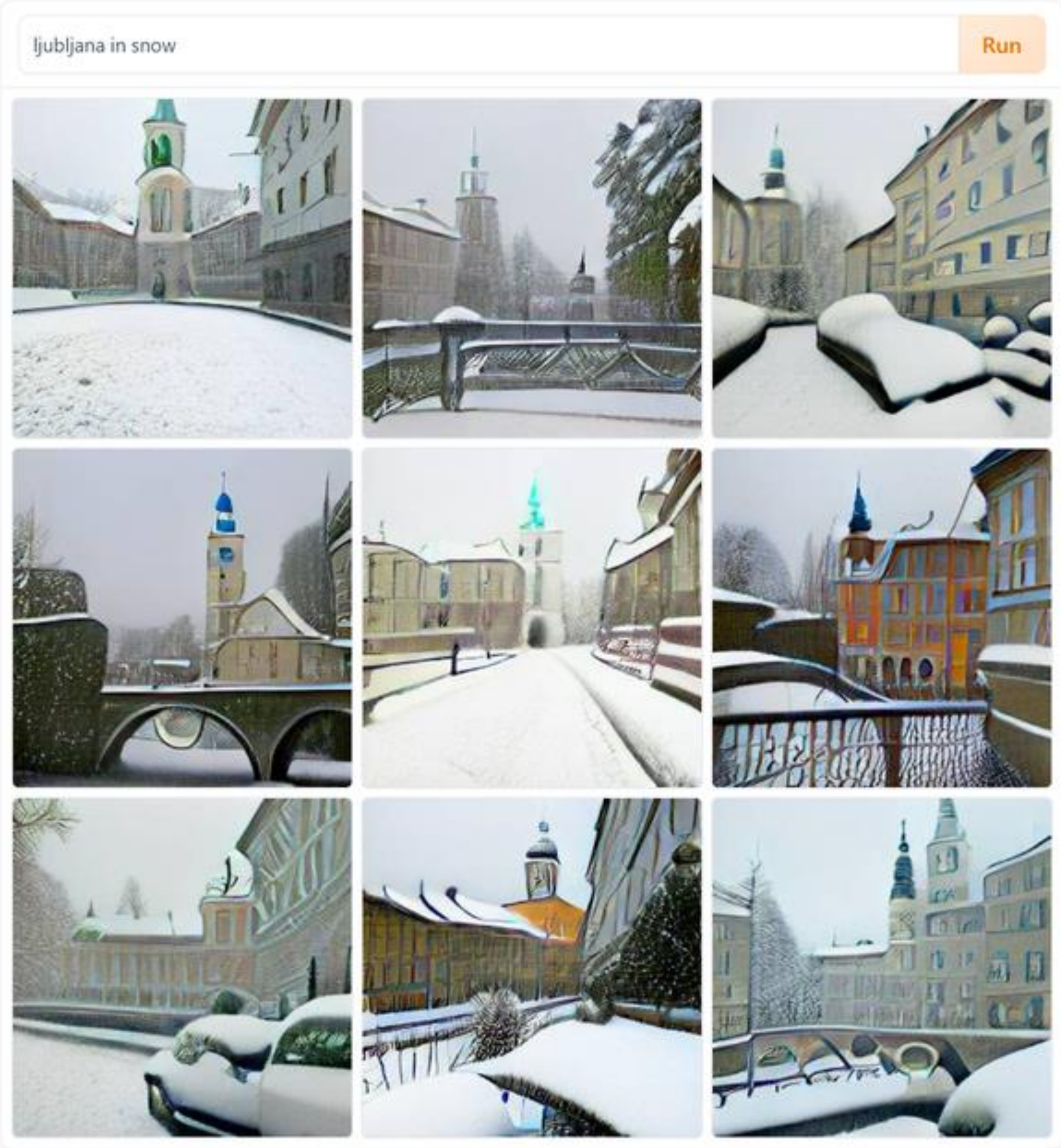
- Zero-Shot Text-to-Image Generation
- Learning joint distribution over images and captions $p_{\theta,\psi}(x, y, z) = p_\theta(x \mid y, z)p_\psi(y, z)$

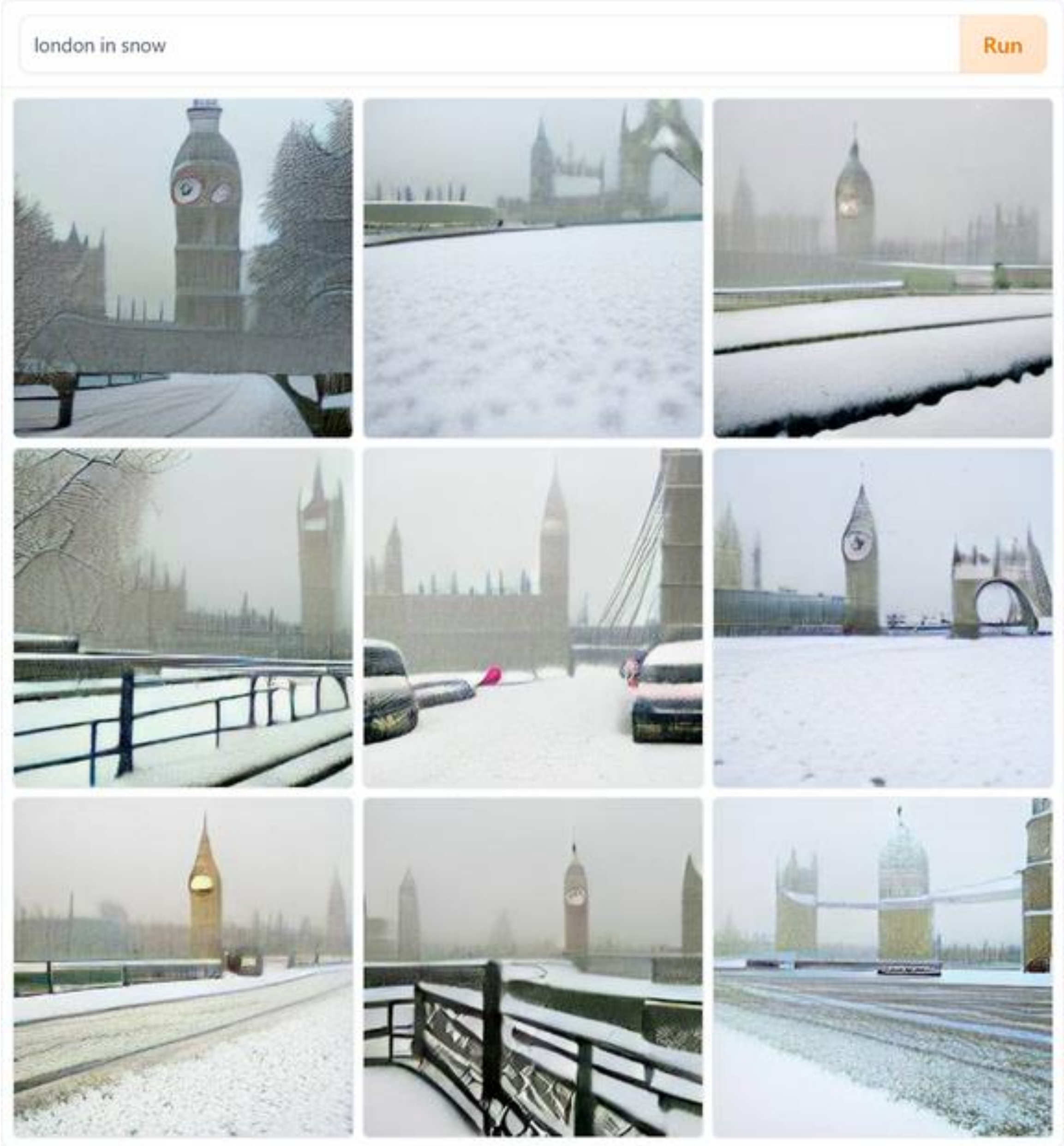


https://openai.com/blog/dall-e/

DALL·E mini results

https://huggingface.co/spaces/dalle-mini/dalle-mini

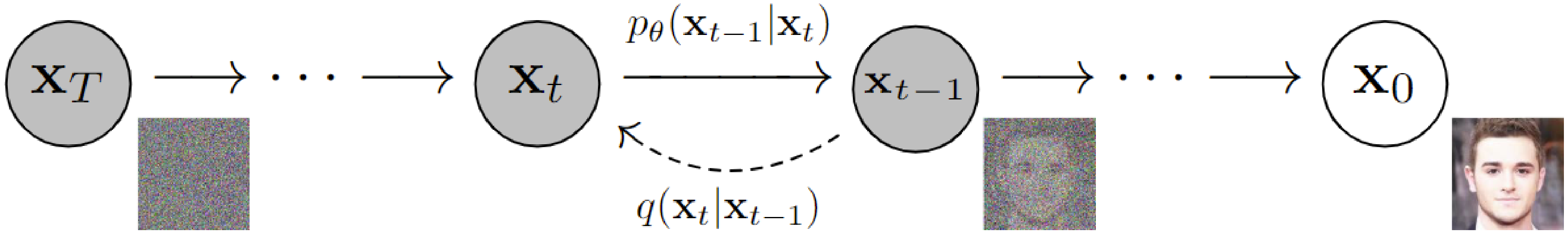




Diffusion models

- Denoising Diffusion Probabilistic Models

Ho et al., 2020



Denoising Diffusion Probabilistic Models

- Forward process
  - =diffusion process
  - from image to noise
  - gradually add Gausian noise
  - create x_t from x_{t-1} (or from x_0)
- Reverse process
  - from noise to image
  - train network to estimate noise
  - inference reconstructs x_{t-1} from x_t

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar\alpha_t}\mathbf{x}_0, (1-\bar\alpha_t)\mathbf{I}) \qquad \alpha_i = 1-\beta_i$$

$$\mathbf{x}_t = \sqrt{\bar\alpha_t}\mathbf{x}_0 + \epsilon\sqrt{1-\bar\alpha_t},\ \epsilon \sim \mathcal{N}(0,1) \qquad \bar\alpha_t = \prod_t^{i=1}\alpha_i$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t z \qquad \sigma_t^2 = \beta_t$$

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

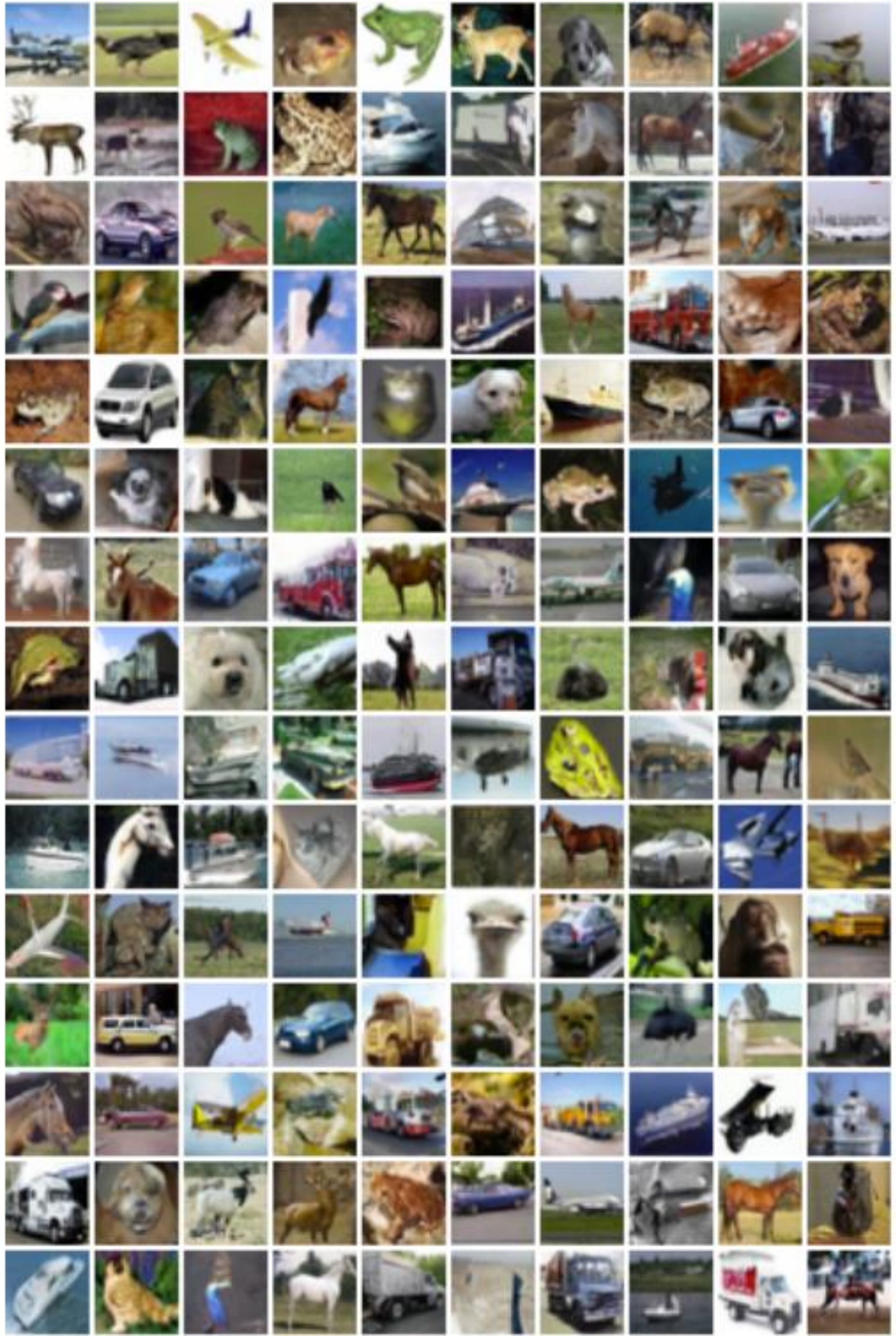
$$x_t \rightarrow x_{t-1} \rightarrow \dots \rightarrow x_0$$

Algorithm 1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1,\dots,T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
  Take gradient descent step on
  $$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar\alpha_t}\mathbf{x}_0 + \sqrt{1-\bar\alpha_t}\boldsymbol{\epsilon}, t)\right\|^2$$
until converged

Algorithm 2 Sampling

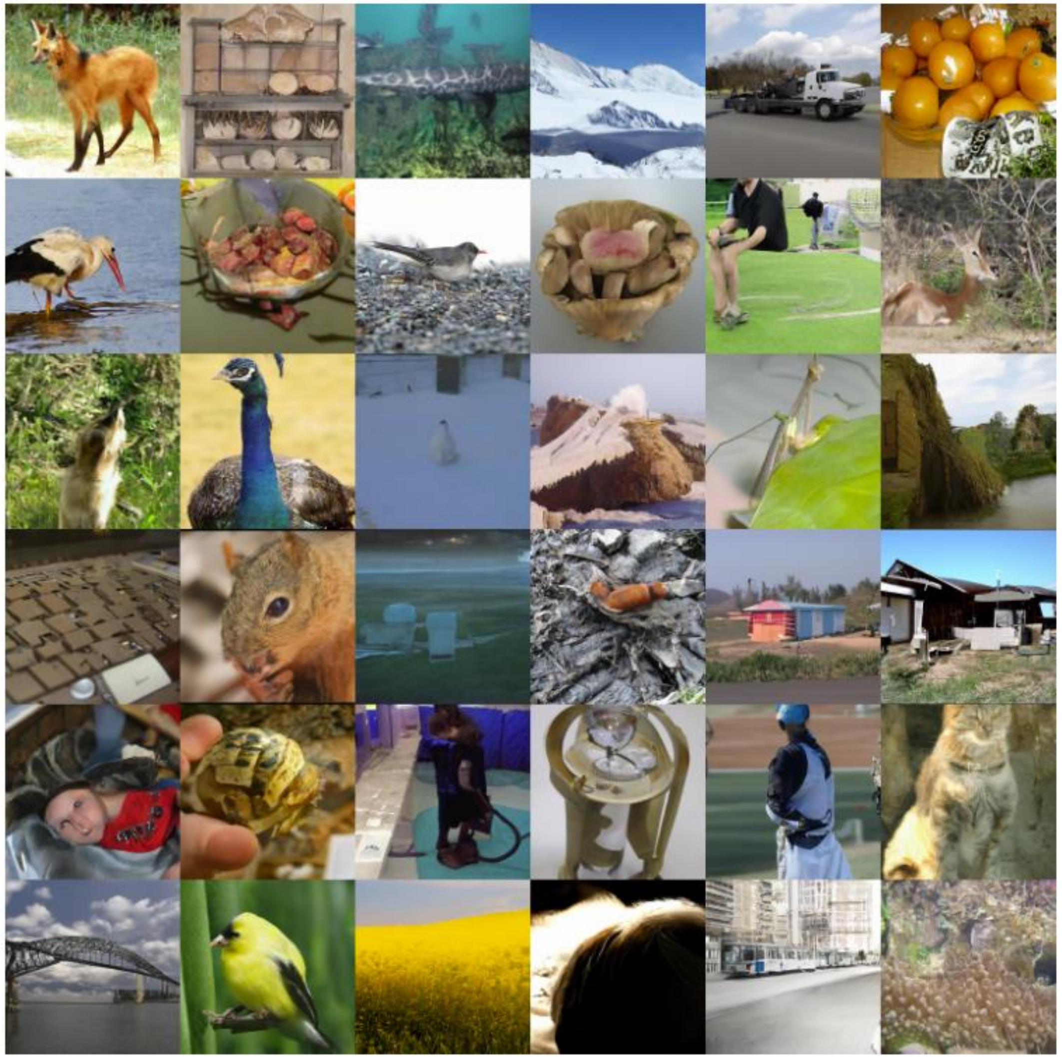
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
for $t = T,\dots,1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z}$
end for
return $\mathbf{x}_0$

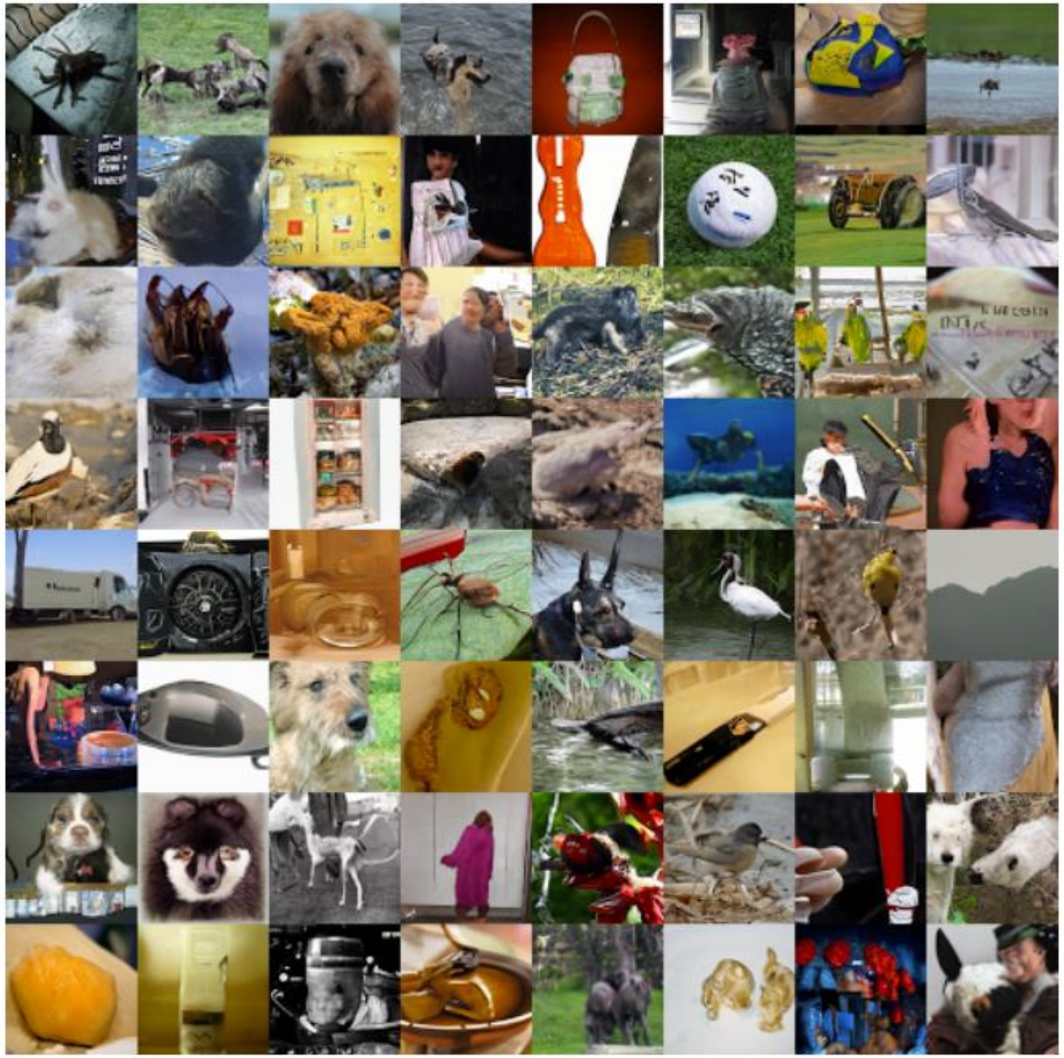



DDPM

- Improved Denoising Diffusion Probabilistic Models

Nichol et al., 2021





DDPM

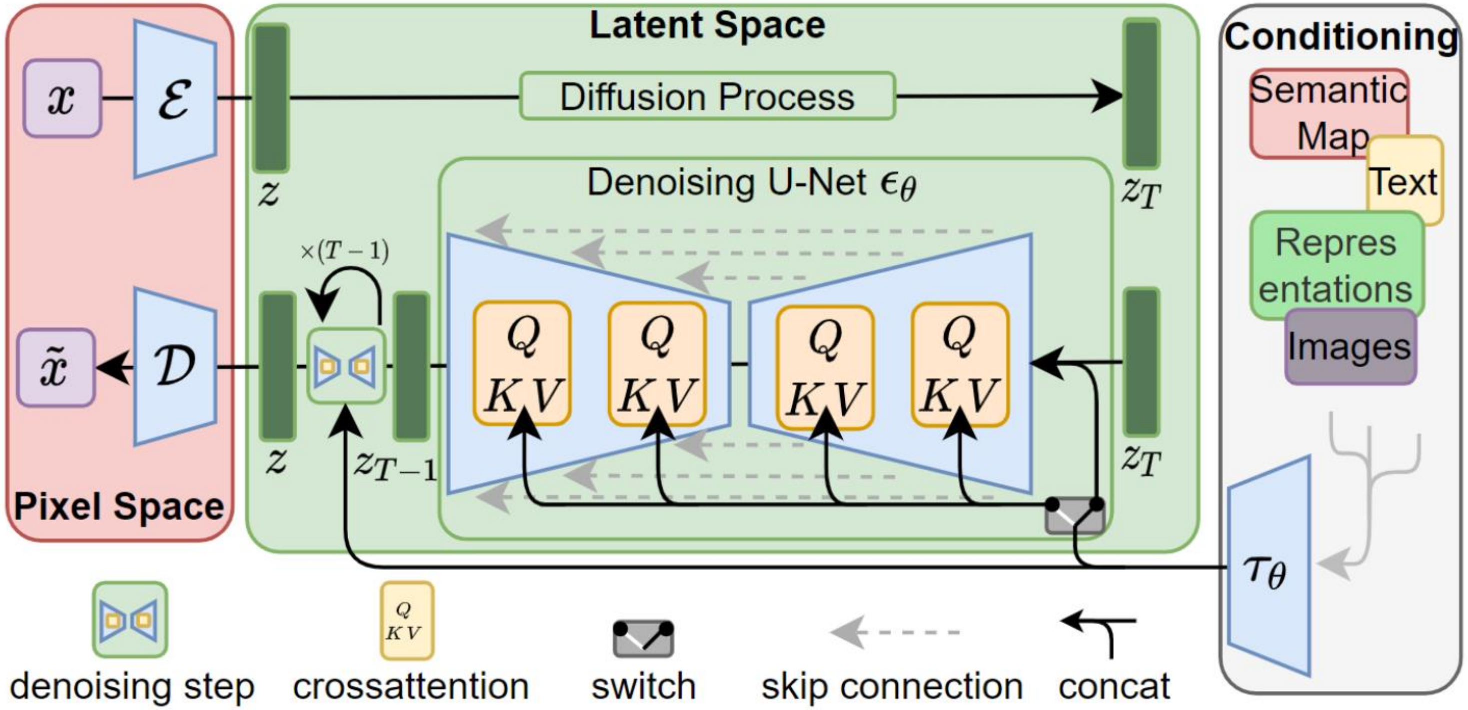
- Diffusion Models Beat GANs on Image Synthesis

[Dhariwal et al., 2021]

Stable Diffusion

- **Latent Diffusion:** Sequential application of denoising autoencoders in the latent space
  - Cross-attention layers for conditioning inputs

Rombach et al., 2022



- **Stable Diffusion** = (modified) Latent Diffusion

  \+ https://stability.ai/ – Cluster ~4000 A100 GPUs, well financed startup doing open source

  \+ https://laion.ai/blog/laion-5b/ – Open data scraping project

Stable Diffusion

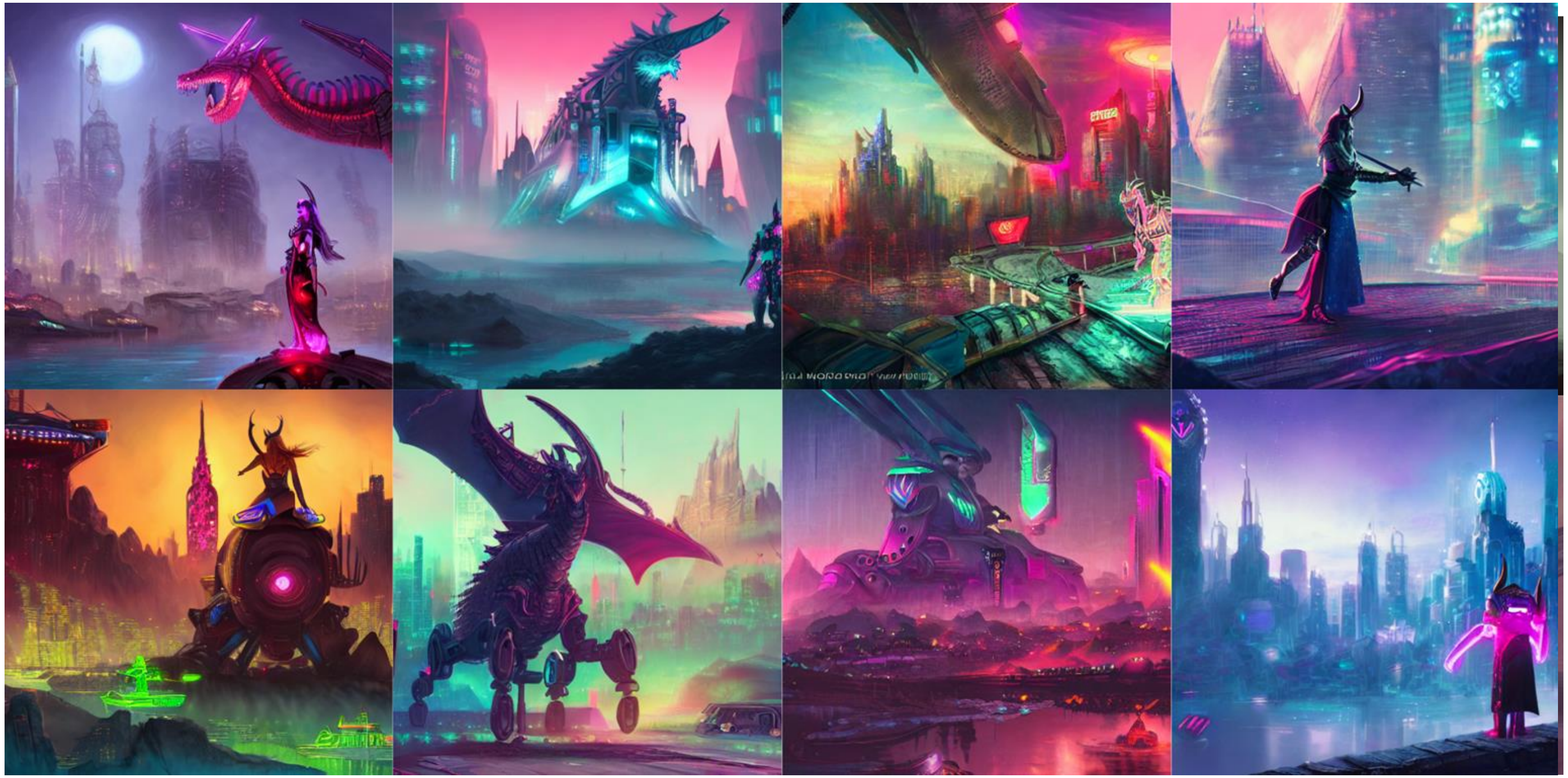
bear wearing sunglasses riding a rocket into outer space, digital art, trending on artstation, 4k, hd

[https://github.com/huggingface/diffusers]

Teddy bears swimming at the olympics 400m butterfly event

Stable Diffusion

A matte painting of a viking in a princess dress riding a robot dragon, robotics, futuristic, city in background, neon lights, vivid color scheme, cyberpunk, digital art, hd, 4k, trending on artstation



[https://github.com/huggingface/diffusers]

Stable Diffusion

A renaissance painting of a man screaming at a computer, digital art, renaissance, hd, 4k, trending on artstation

[https://github.com/huggingface/diffusers]

Stable Diffusion inpainting

Prompt: Cat wearing a funny hat

Fine-tunning Stable Diffusion

- Fine tune Diffusion models with only a small number of additional examples
- Finetune with “photo of vitjan” as a prompt.

Ruiz et al., 2022

Text-to-video

Make-a-video

A golden retriever eating ice cream on a beautiful tropical beach at sunset, high resolution

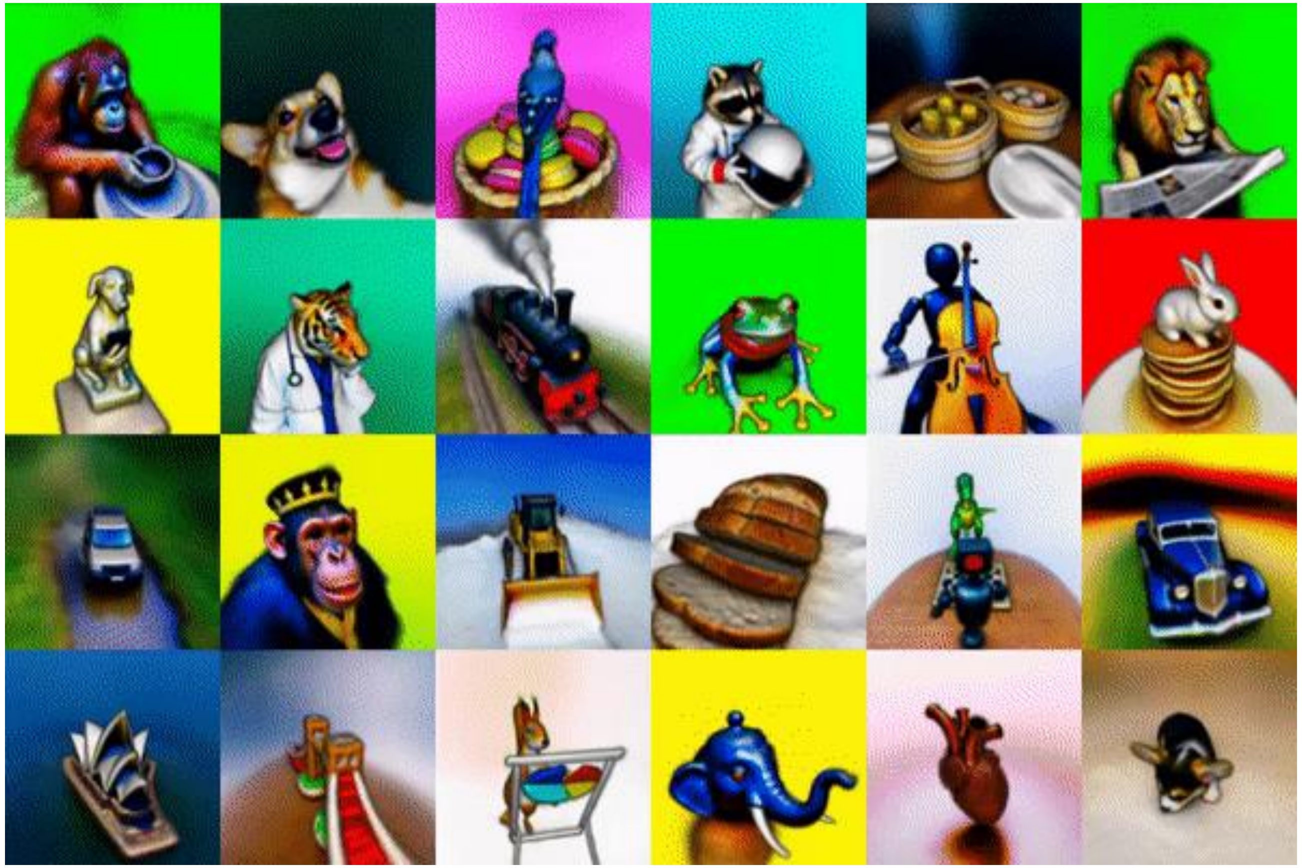
A teddy bear painting a portrait

Cat watching TV with a remote in hand

Singer et al., 2022

Text-to-3D

DreamFusion: Text-to-3D using 2D Diffusion



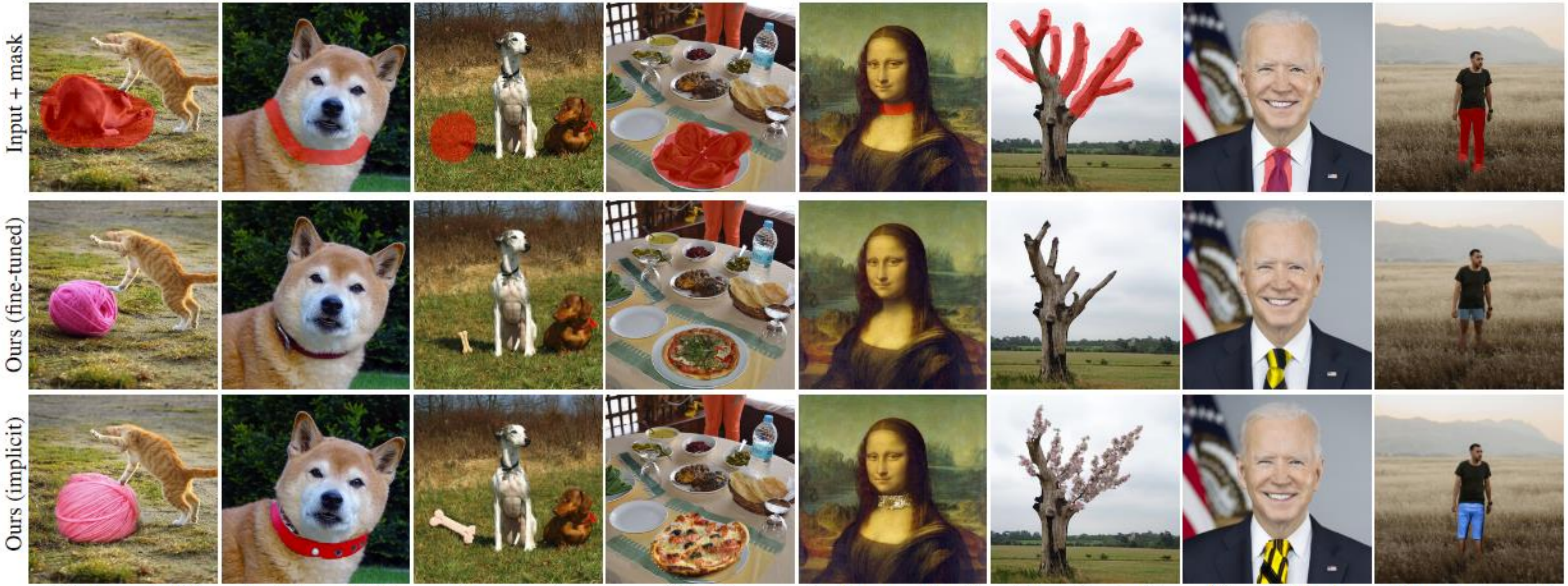
https://dreamfusion3d.github.io/

Poole et al., 2022

GLIDE

Nichol et al., 2022

- Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models



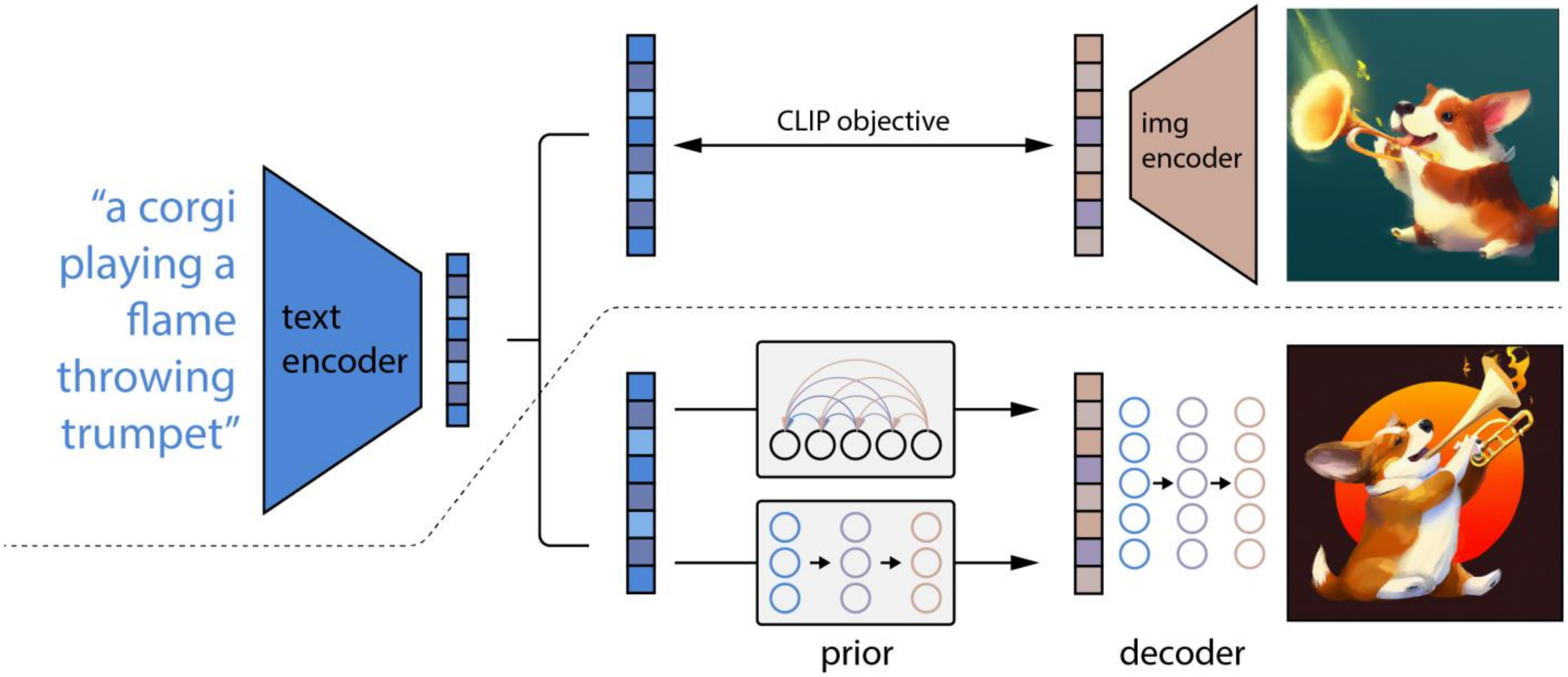
"pink yarn ball" "red dog collar" "dog bone" "pizza" "golden necklace" "blooming tree" "tie with black and yellow stripes" "blue short pants"

"a green train is coming down the tracks" "a group of skiers are preparing to ski down a mountain." "a small kitchen with a low ceiling" "a group of elephants walking in muddy water." "a living area with a television and a table"

DALL-E 2

Ramesh et al., 2022

- Hierarchical Text-Conditional Image Generation with CLIP Latents
  - prior generates a CLIP image embedding given a text caption
  - decoder generates an image conditioned on the image embedding



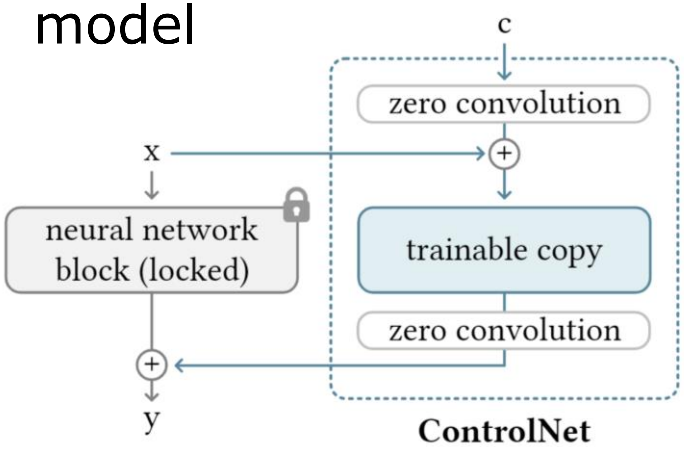
“A teddybear on a skateboard in Times Square.”

DALL-E 2

https://openai.com/product/dall-e-2

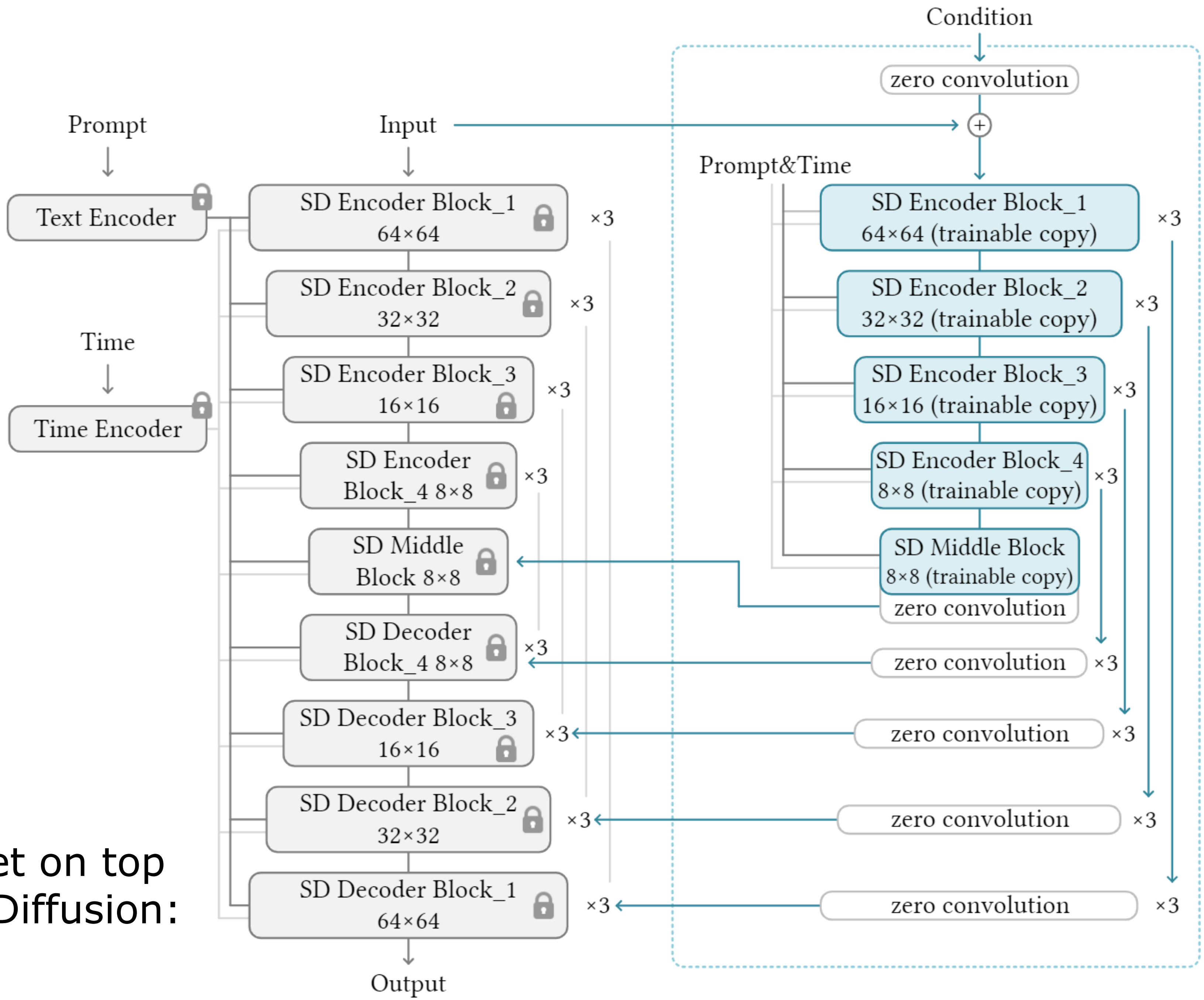
ControlNet

- Controlling large pretrained diff. models
- Learns task-specific conditions en-to-end
- Augments a pretrained model



[Zhang et al., 2023]

- ControlNet on top of StableDiffusion:

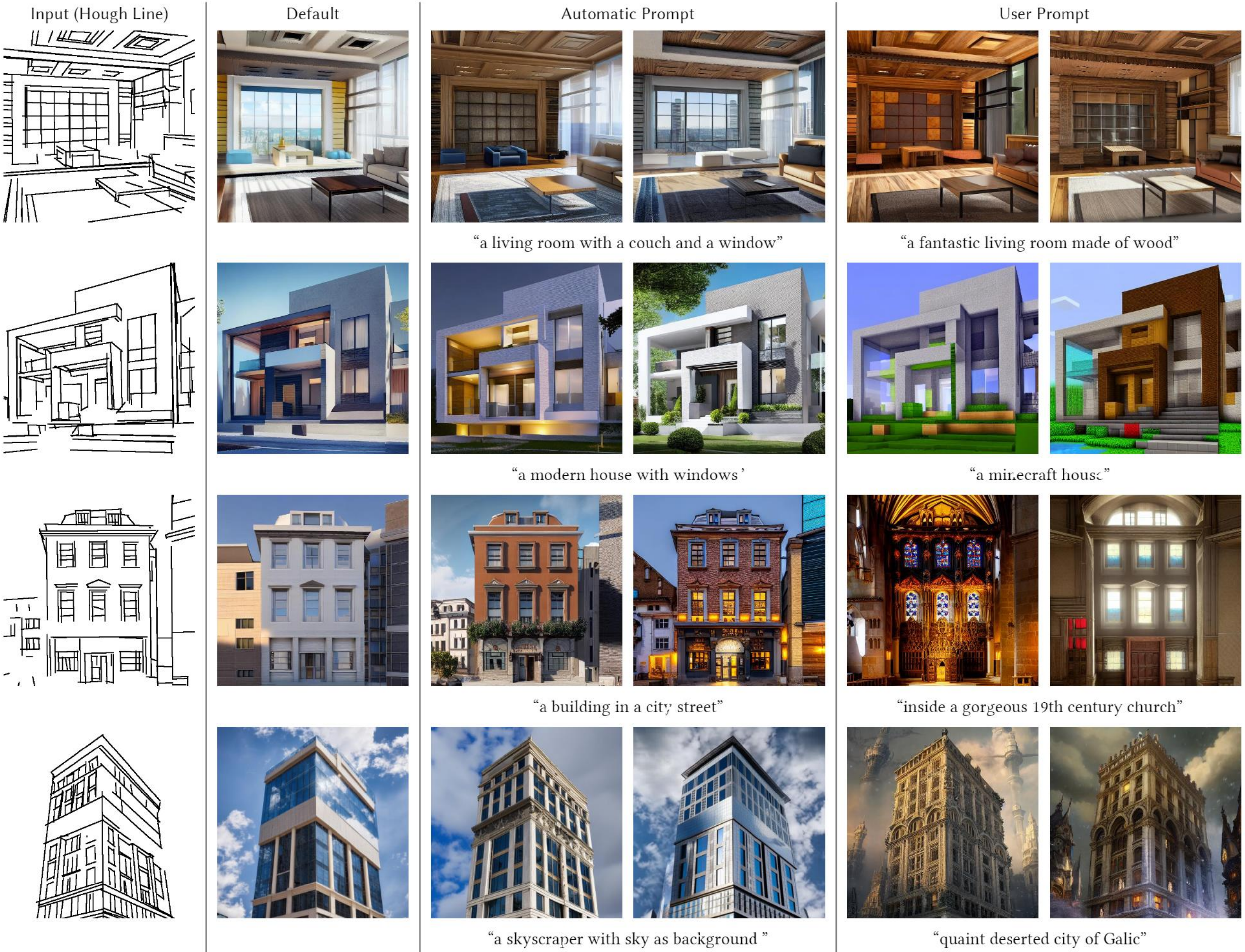


ControlNet

- Image generation controlled by edges

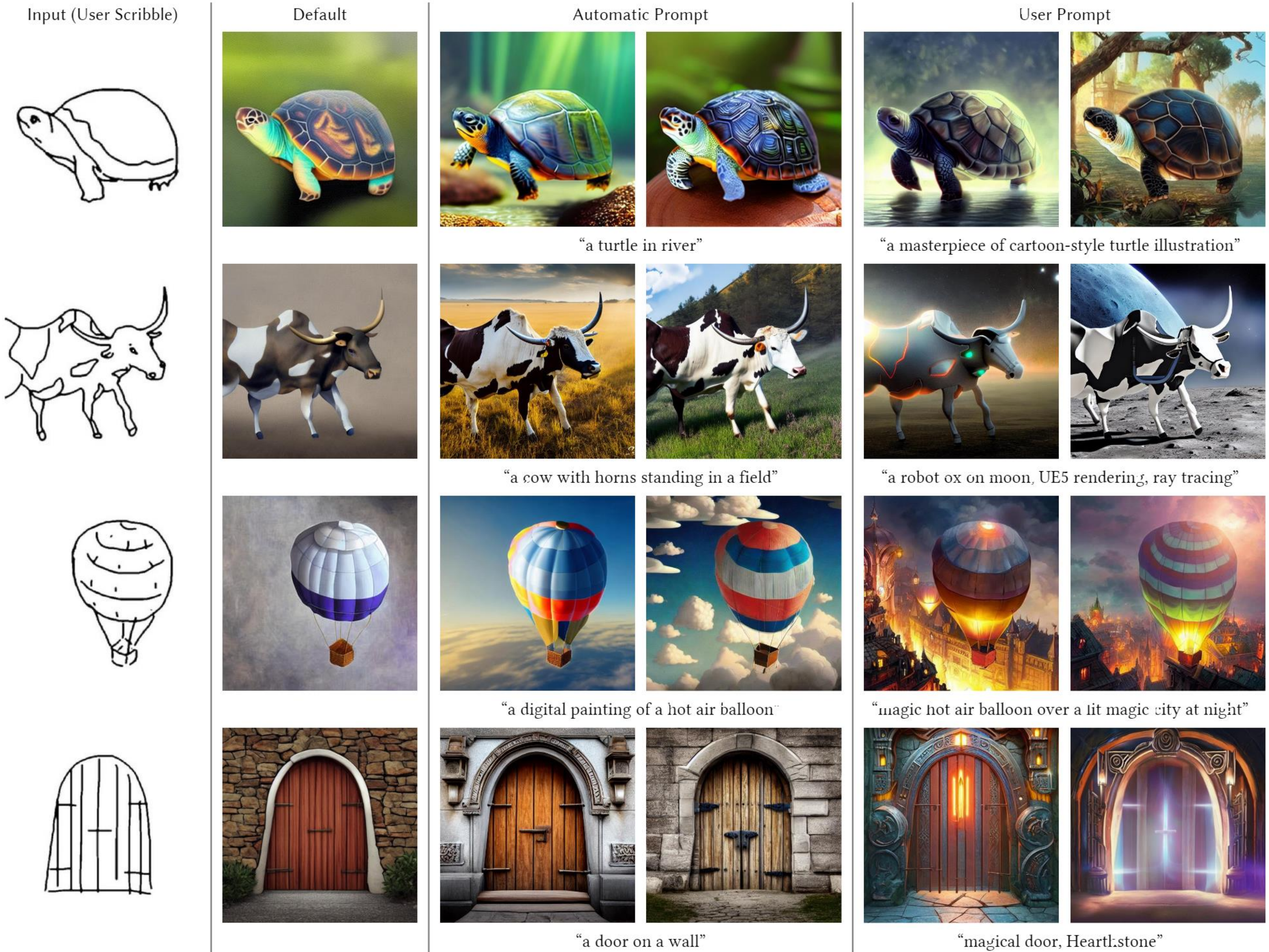
ControlNet

- Image generation controlled by lines



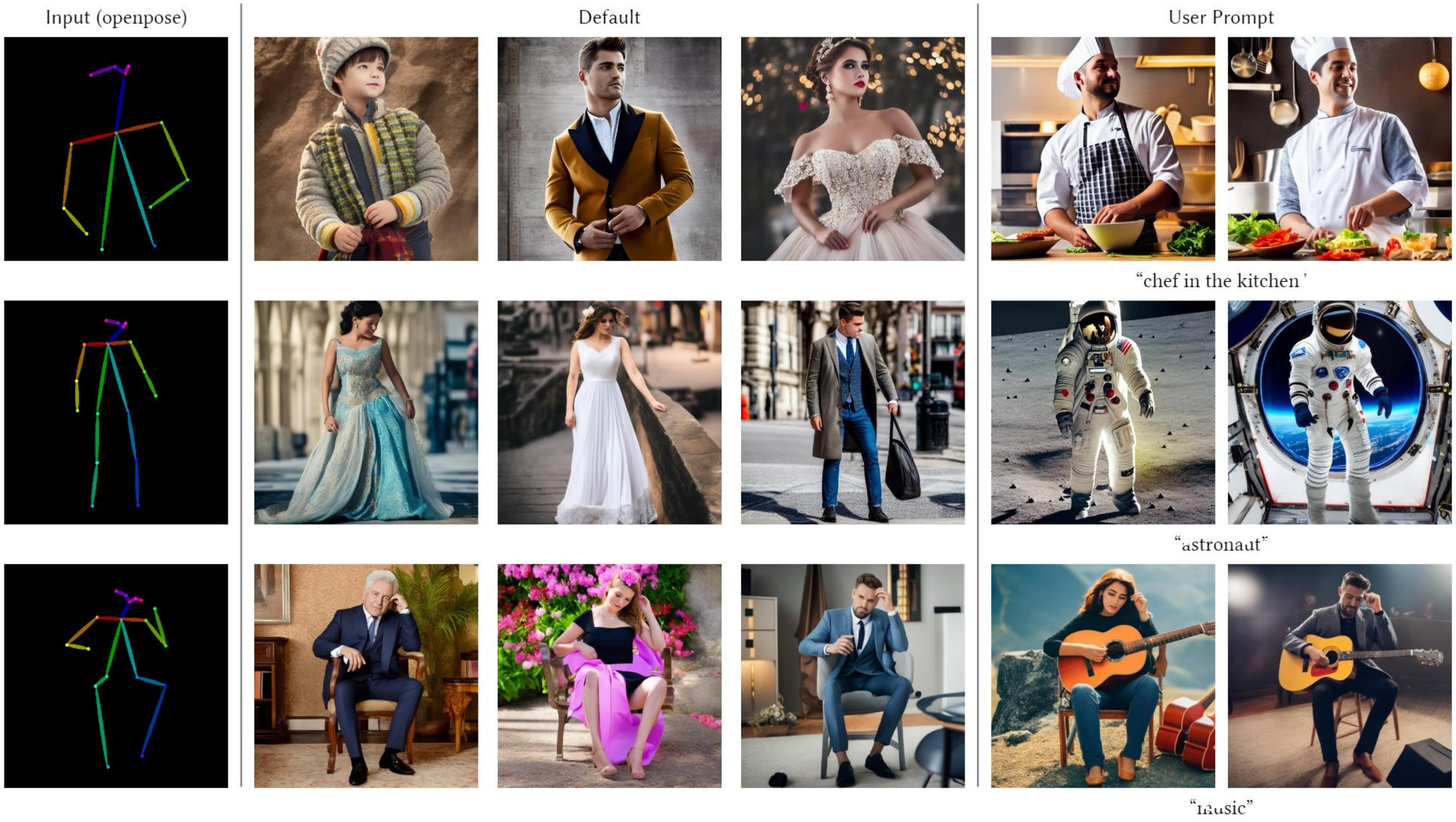
ControlNet

- Image generation controlled by human scribbles

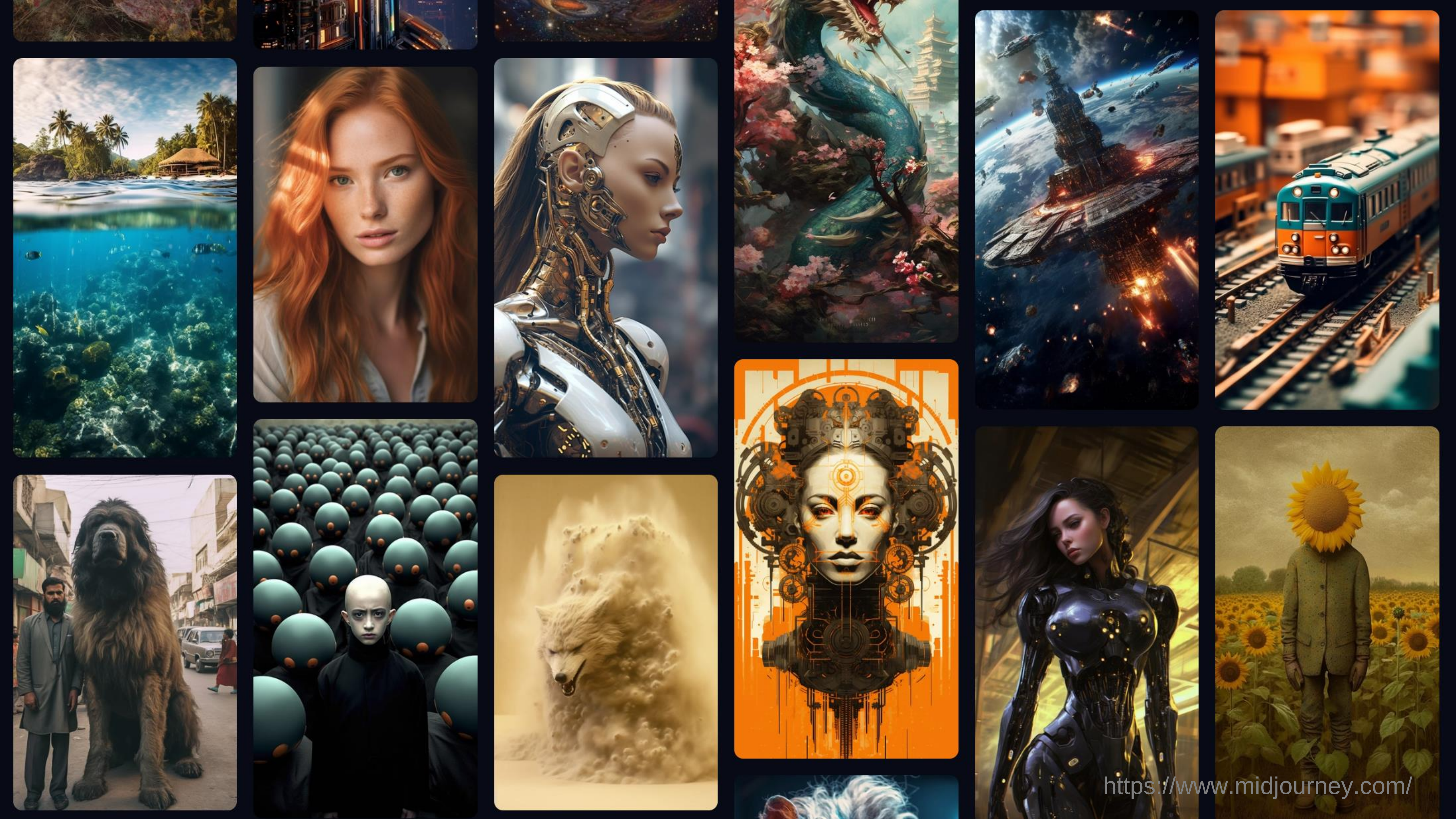


ControlNet

- Image generation controlled by human keypoints



Midjourney



https://www.midjourney.com/


BECOME CHATGPT
PROMPT ENGINEER
TODAY
Be the future!
Midjourney Prompt
Generator
Outputs four extremely detailed midjourney prompts
for your keyword.
SEO Optimized
Keyword Strategy
Create a keyword strategy and SEO content plan
from 1 [KEYWORD]
https://www.midjourney.com/

Diffusion models recap

- Diffusion models are generative models that capture the process of diffusion, or how probability distributions evolve over time.
- They provide a framework for modeling complex distributions and generating high-quality samples.
- Key concept: Diffusion process, where a sample is gradually transformed by adding noise in a controlled manner.
- Samples are iteratively updated over multiple steps to approximate the target distribution.
- Diffusion models can be used for tasks such as image generation, inpainting, and denoising.
- They offer advantages like capturing long-range dependencies and handling multimodal distributions.
- Training diffusion models typically involves maximizing the likelihood of observed data through gradient-based optimization.
- Inference with diffusion models involves reversing the diffusion process to obtain the initial sample or perform tasks like inpainting.

Generative models recap

- GMM
  - Simple, work well on low-dimensional data
  - Problems on high-dimensional data, difficult to increase the model capacity
- PCA
  - Simple, fast, robust, enables reconstruction
  - Linear, limited capacity
- AE
  - Simple setup, enables reconstruction, self-supervised learning for model pretraining
  - Latent space not nice, not smooth, does not enable useful sampling
- VAE
  - Principled approach, allows inference of $q(z|x)$, nice latent space, useful representations
  - Maximising lower bound, not exact, samples tend to be blurrier and lower quality vs. GAN
- GAN
  - Game theoretic approach, great quality
  - No explicit probability modelling, no inference queries, more unstable to train
- NF
  - Exact log-likelihood computation, exact and efficient sampling and inference
  - Samples tend to be lower quality than the results of GAN
- Diffusion models
  - Modelling complex distributions, capturing long-range dependencies, SOTA performance
  - Computational complexity