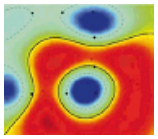
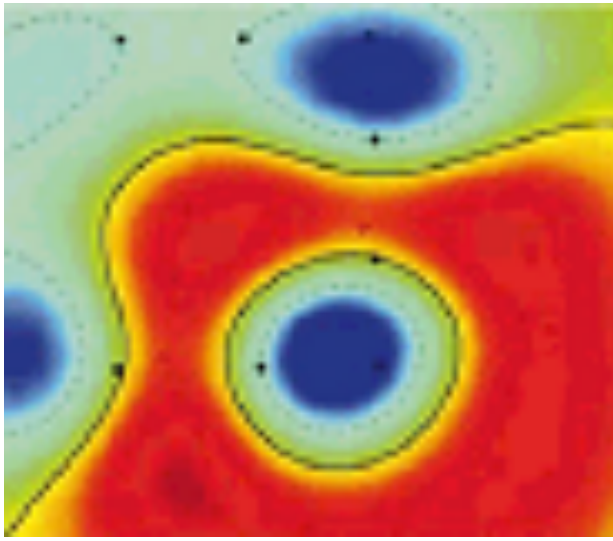


SoSe 2023

Deep Learning 2

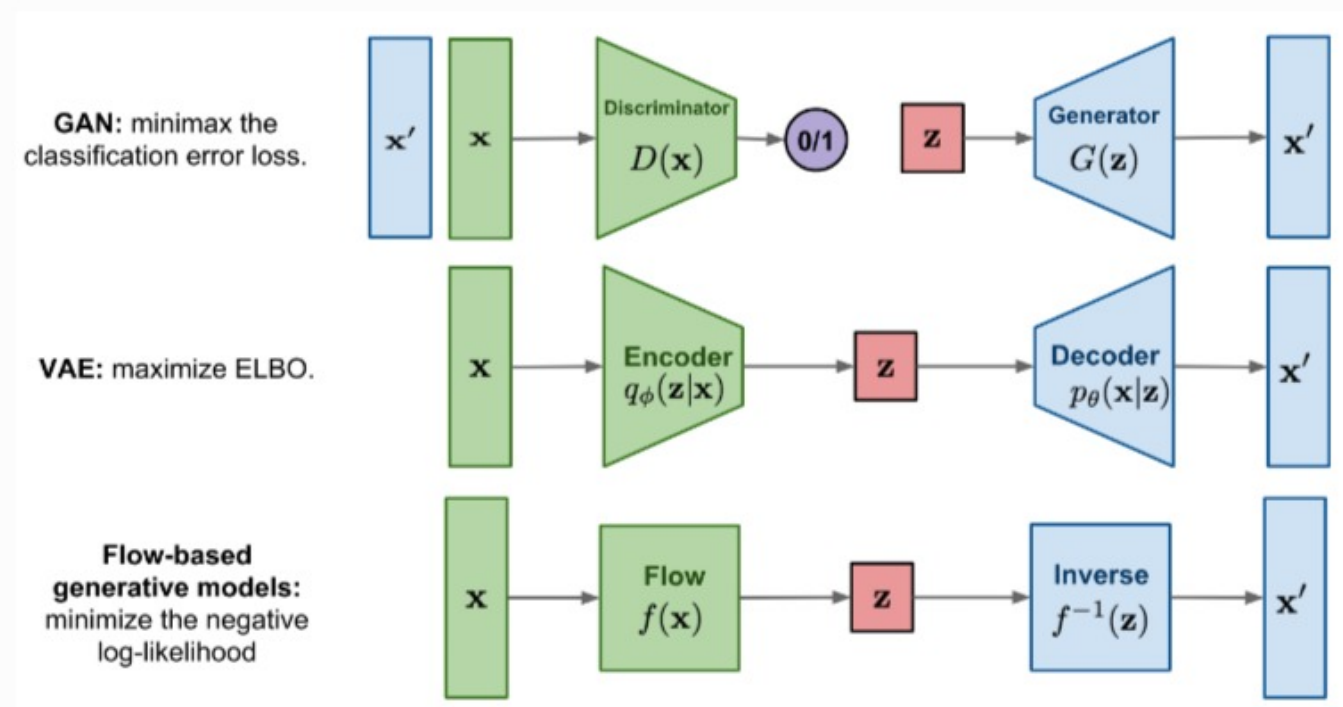
Technische Universität Berlin



Lecture 4

Generative Models - part 2

Generative Adversarial Networks [4]



Another important generative model are Generative Adversarial Networks (GANs) [**?**]. They can sample from the (conditional) distribution of a dataset and assign given data points scores that correlate with its likelihood. This is done by implicit likelihood estimation.

The Minimax Game

Generative Adversarial Networks consist of two subnetworks, the generator G and the discriminator D.

- The generator generates data based on a noise variable z and tries to trick the discriminator into be believing this data is real.
- Meanwhile, the discriminator tries to distinguish the generated data from real samples of the dataset.

This game can be formalized to the following Minimax Game:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{x}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \quad (1)$$

It can be shown that the global maximum of this game is $p_g = p_{data}$ (Exercise!!).

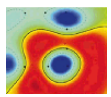
Vanilla GAN results

Patch-GAN [6]

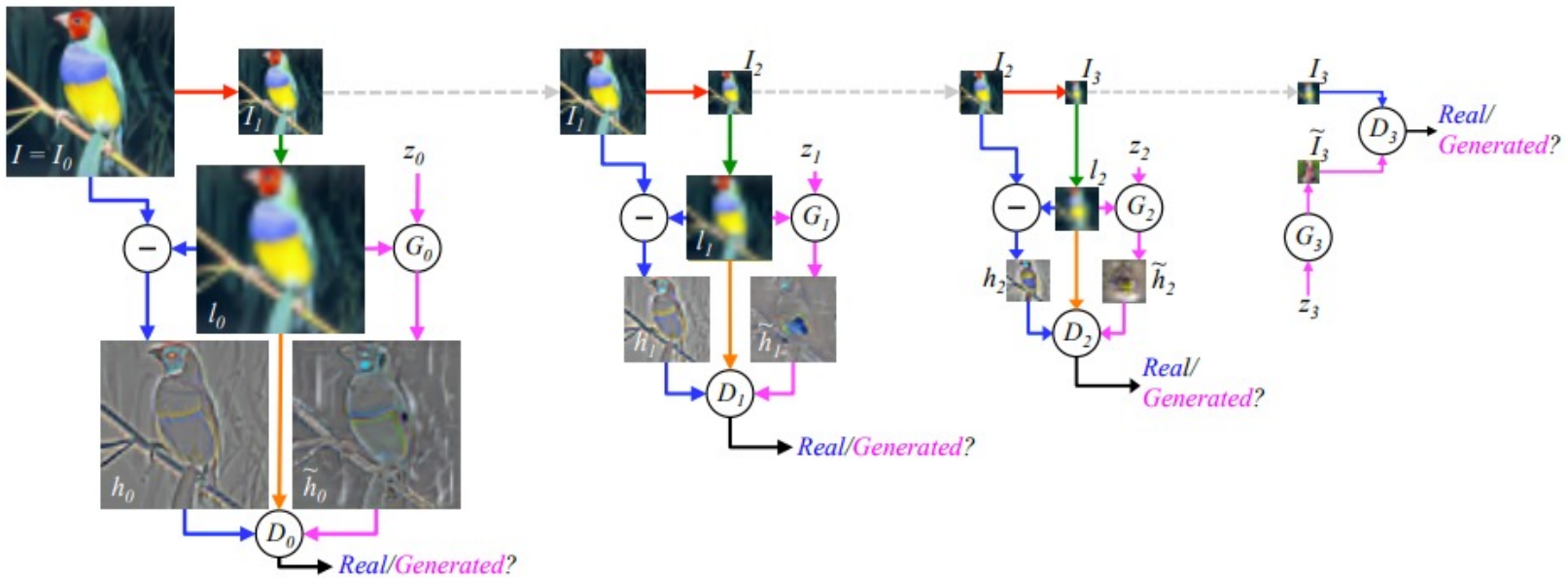
Patch-GAN [6] uses a type of discriminator for generative adversarial networks that penalizes structure at the scale of local image patches.

- The Patch-GAN discriminator tries to classify if each patch in an image is real or fake.
- This discriminator is run convolutionally across the image, averaging all responses to provide the ultimate output of D.
- Such a discriminator effectively models the image as a Markov random field, assuming independence between pixels separated by more than a patch diameter.

Patch-GAN can be understood as a type of style loss.



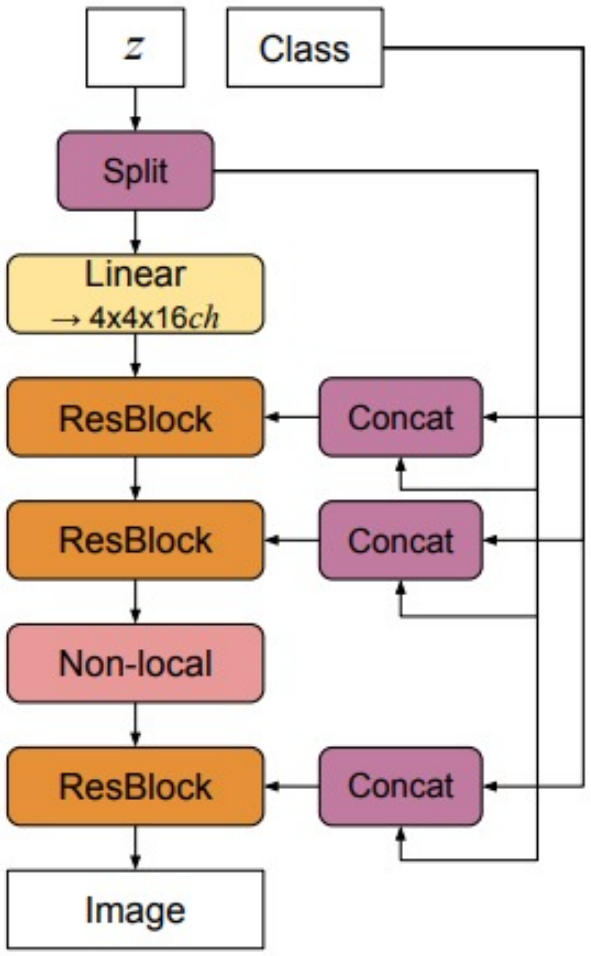
LAPGAN [2]

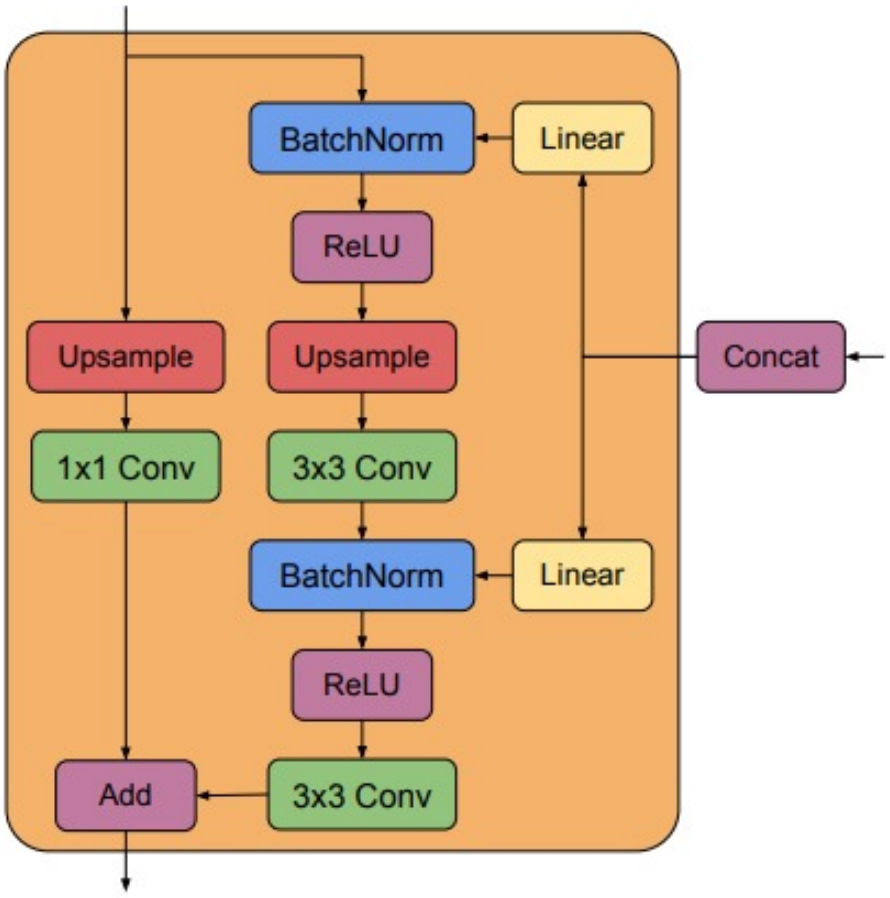


LAPGAN [2] is a type of generative adversarial network that uses a discriminator for different scale of the generated image separately. This is done by using a cascade of convolutional networks within a Laplacian pyramid framework to generate images in a coarse-to-fine fashion. At each level of the pyramid, a separate generative convnet model is trained using the GAN approach. This approach sample quality significantly over a vanilla GAN.

LAPGAN results

Big-GAN [1]



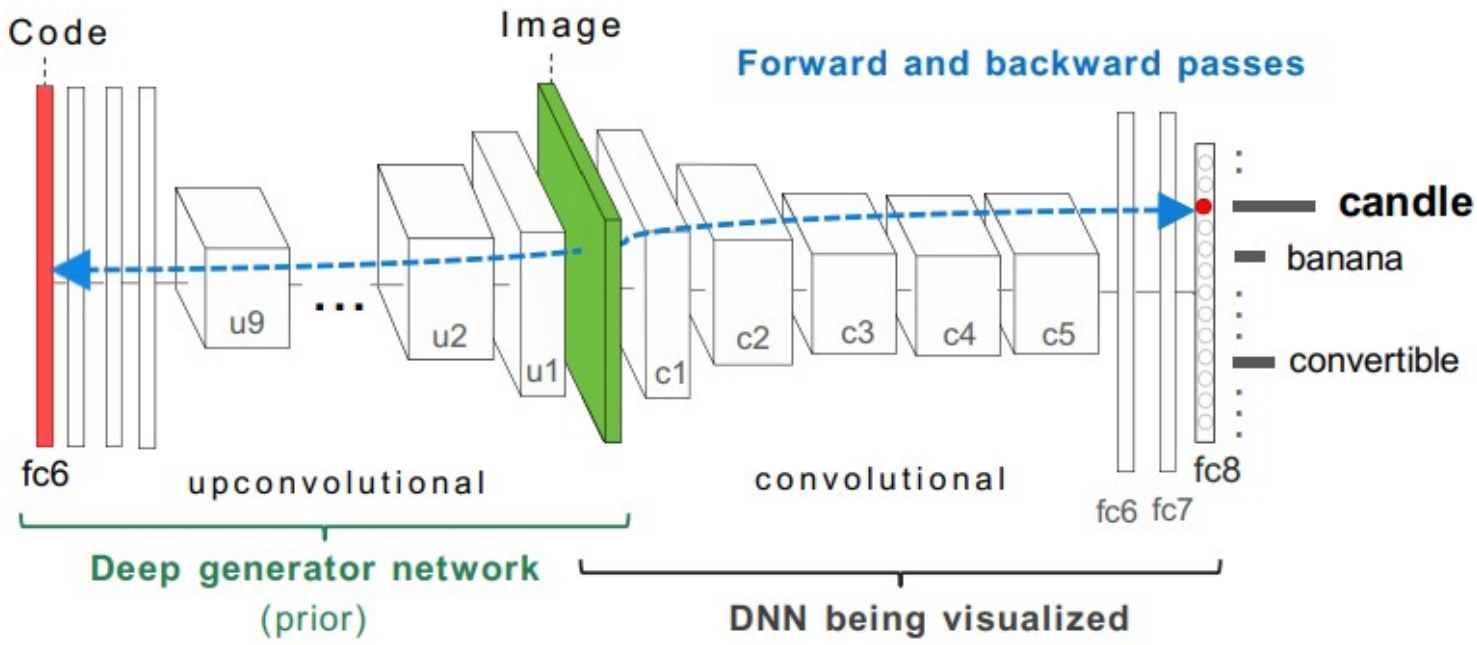


- Architecture of the Big-GAN model (has billions of parameters)
- Non-local usually is a big self-attention layer

Big-GAN [1]

- Network was trained for many weeks on many TPUs by Google
- Manual restarts in the case of mode collapses
- Generated samples are sampled from heavily truncated normal distribution

Application: Deep Generator Networks [8]



- Activation maximization in latent space of the GAN instead in pixel space
- Encourages search to stay on data manifold
- However, sometimes even finding a suitable starting point is not that easy

Application: Deep Generator Networks [8]

- Inversions of certain labels from the logits layer

Cycle-GAN [12]

Zhu et al. [12] introduced Cycle-GAN, which can restyle an image from one domain into an image of another domain without aligned training images.

- The goal of the Cycle-GAN is to learn mappings between two domains X and Y given training samples $\{x_i\}_{i=1}^{N}$ where $x_i \in X$ and $\{y_j\}_{j=1}^{M}$ where $y_j \in Y$[1].
- The data distributions thereby are denoted as $x \sim p_{\text{data}}(x)$ and $y \sim p_{\text{data}}(y)$.

The model consists of two functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$. Furthermore, two discriminators D_X and D_Y are introduced. D_X aims to distinguish between images $\{x\}$ and translated images $\{F(y)\}$; in the same way, D_Y aims to discriminate between $\{y\}$ and $\{G(x)\}$.

Cycle-GAN [12]

The objective contains two types of terms: adversarial losses, namely:

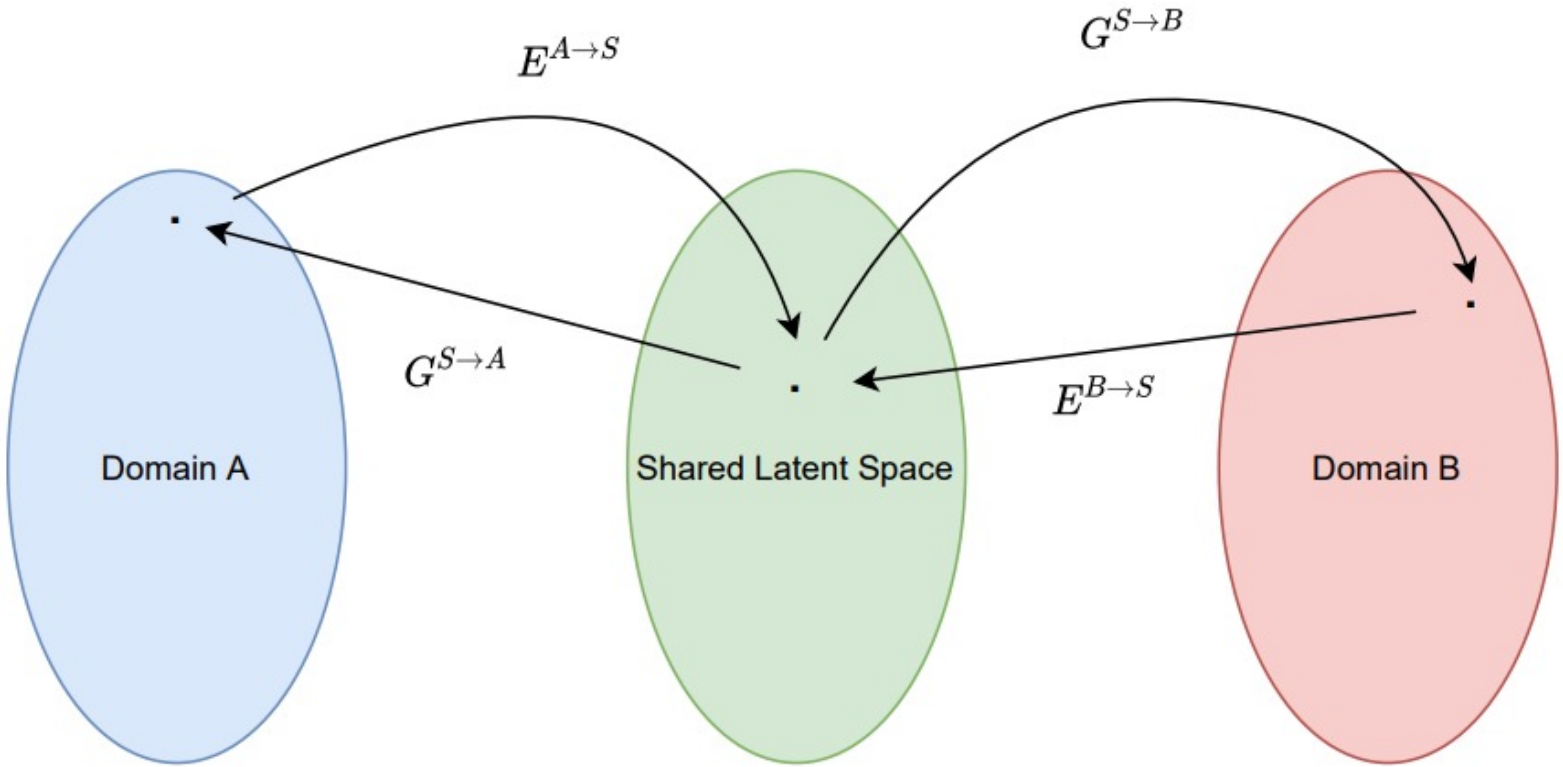
$$\begin{aligned}\mathcal{L}_{\mathrm{GAN}}\left(G, D_Y, X, Y\right) = & \mathbb{E}_{y \sim p_{\Delta\,\mathrm{dat}}(y)}\left[\log D_Y(y)\right] \\ & + \mathbb{E}_{x \sim p_{\mathrm{dat}}(x)}\left[\log\left(1 - D_Y(G(x))\right)\right]\end{aligned} \tag{2}$$

for matching the distribution of generated images to the data distribution in the target domain; and cycle consistency losses, namely:

$$\begin{aligned}\mathcal{L}_{\mathrm{cyc}}\left(G, F\right) = & \mathbb{E}_{x \sim p_{\mathrm{dat}}\,(x)}\left[\|F(G(x)) - x\|_1\right] \\ & + \mathbb{E}_{y \sim p_{\mathrm{da}}\,(y)}\left[\|G(F(y)) - y\|_1\right].\end{aligned} \tag{3}$$

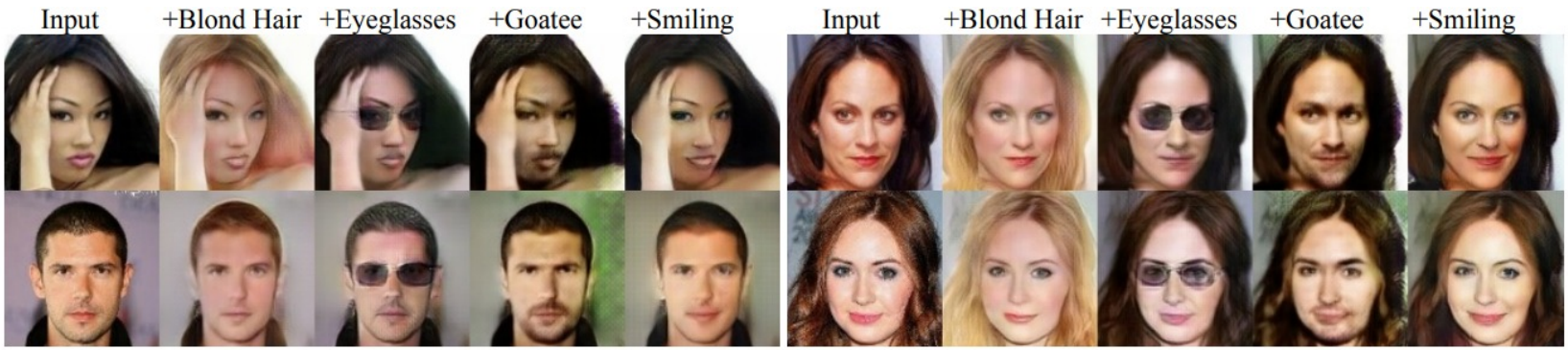
to prevent the learned mappings G and F from contradicting each other.

UNIT [7]



- Very similar to Cycle-GAN, but instead of decoding into the other domain networks are encouraged to use shared domain
- Enables more efficient domain transfer
- Enables encoding as many domains into a shared latent space as wished

UNIT results



- Qualitative Results where domain is switched on CelebA dataset

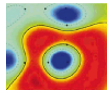
Application: Domain Adaptation

Method	SA [4]	DANN [5]	DTN [26]	CoGAN	UNIT (proposed)
SVHN→ MNIST	0.5932	0.7385	0.8488	-	**0.9053**
MNIST→ USPS	-	-	-	0.9565	**0.9597**
USPS→ MNIST	-	-	-	0.9315	**0.9358**

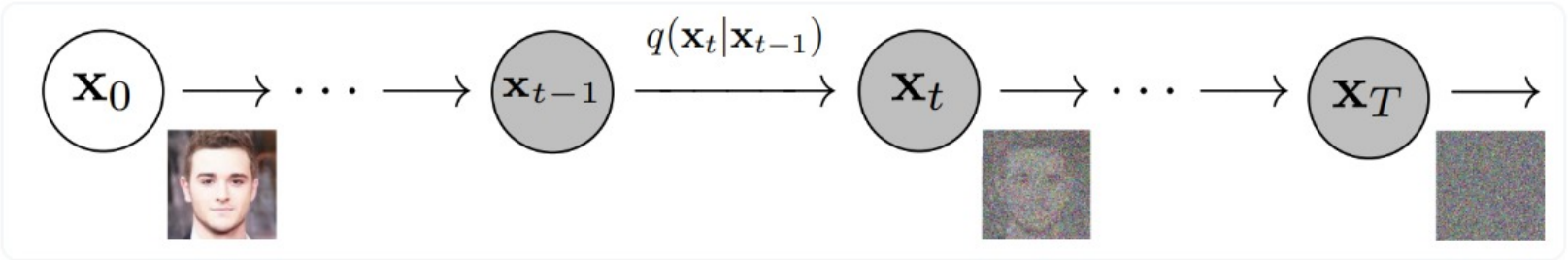
- Quantitative Results that shows how well UNIT is doing compared to other approaches when transferring with zero labels unaligned from one dataset to the other

Energy Based Models

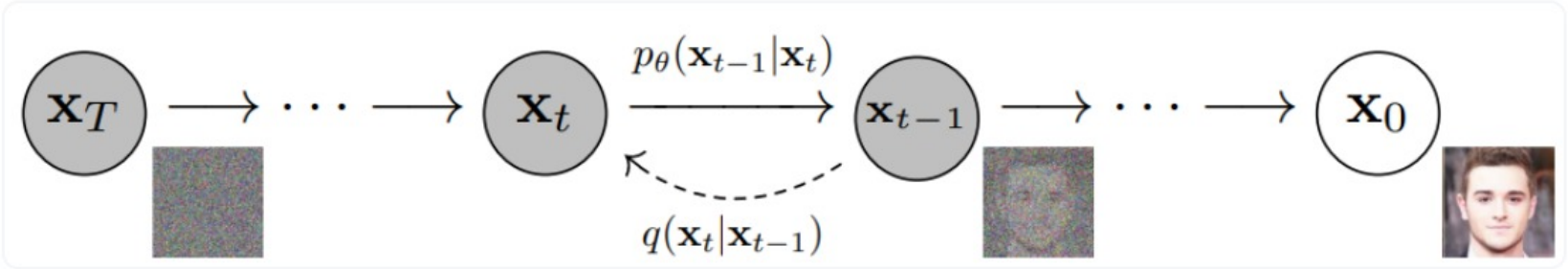
- Have been around for a long time (Restricted Boltzman Machines, Hopfield networks etc)
- Recently gained interest again in the form of denoising diffusion models



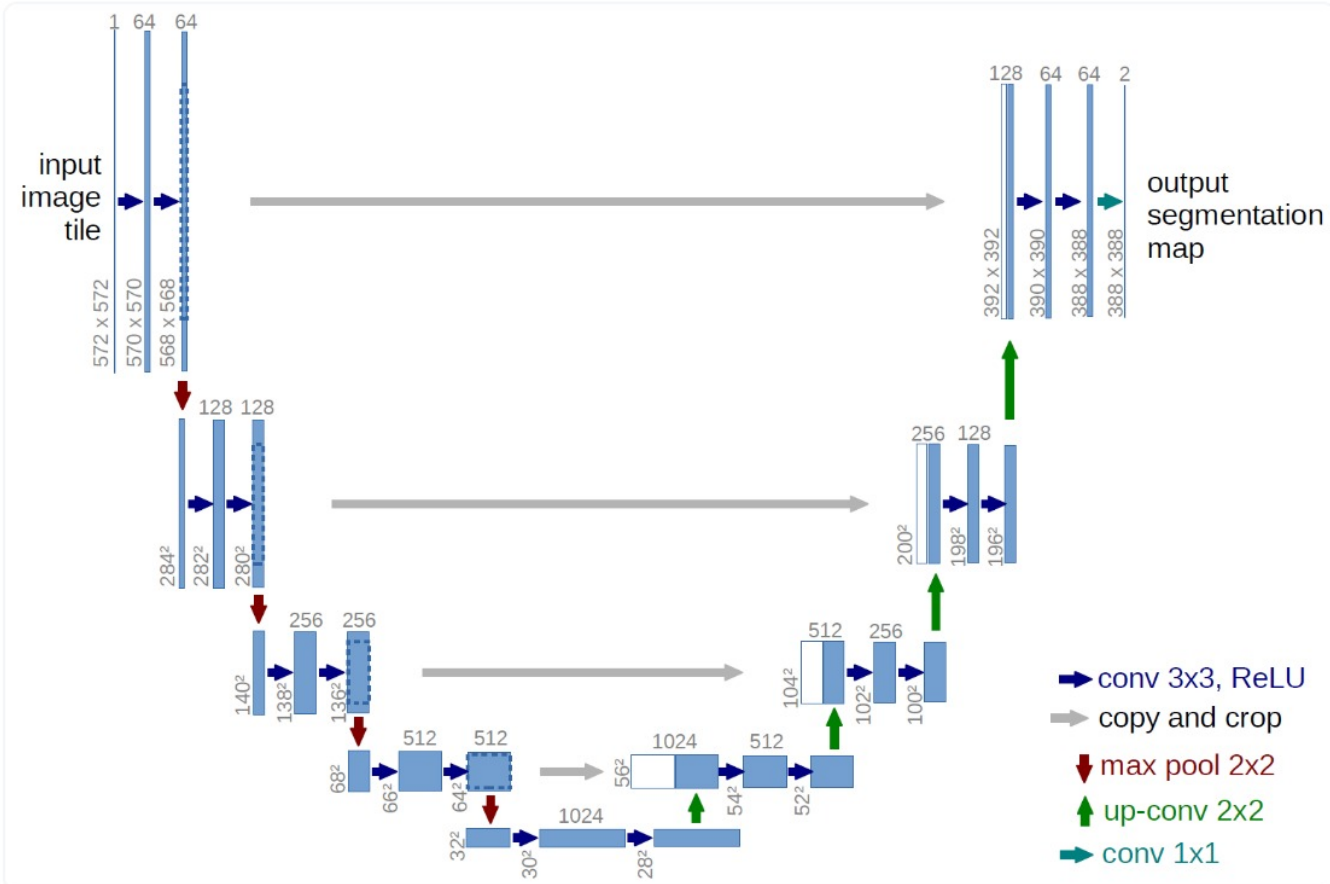
Denoising Diffusion Models [10]



- The idea is to model physical diffusion process by gradually adding gaussian noise to an image
- The generative model learns to reverse this process



Denoising Diffusion Models [10]



- Usually the generative model is a U-Net
- While inference one can start from random noise and apply network so long until it converges to an image

Training Denoising Diffusion Probabilistic Models (DDPM) [5]

A Diffusion Model is trained by finding the reverse Markov transitions that maximize the likelihood of the training data. In practice, training equivalently consists of minimizing the variational upper bound on the negative log likelihood.

$$\mathbb{E}\left[-\log p_\theta\left(\mathbf{x}_0\right)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q\left(\mathbf{x}_{1:T} \mid \mathbf{x}_0\right)}\right] =: L_{vlb}$$

It is possible to rewrite $Lvlb$ almost completely in terms of KL divergences (Exercise!!):

$$L_{vlb} = L_0 + L_1 + \ldots + L_{T-1} + L_T$$

where

$$L_0 = -\log p_\theta\left(x_0 \mid x_1\right)$$
$$L_{t-1} = D_{KL}\left(q\left(x_{t-1} \mid x_t, x_0\right) \| p_\theta\left(x_{t-1} \mid x_t\right)\right)$$
$$L_T = D_{KL}\left(q\left(x_T \mid x_0\right) \| p\left(x_T\right)\right)$$

Rewriting the loss to KL divergences has the benefit that KL divergences between gaussians have a closed form and are thereby tractable.

Algorithms DDPM

Algorithm 1 Training

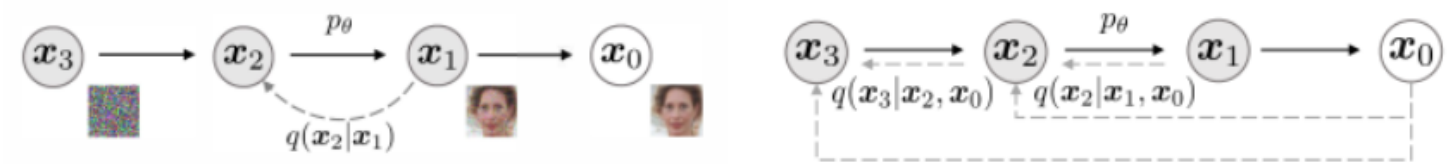
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

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

Results DDPM

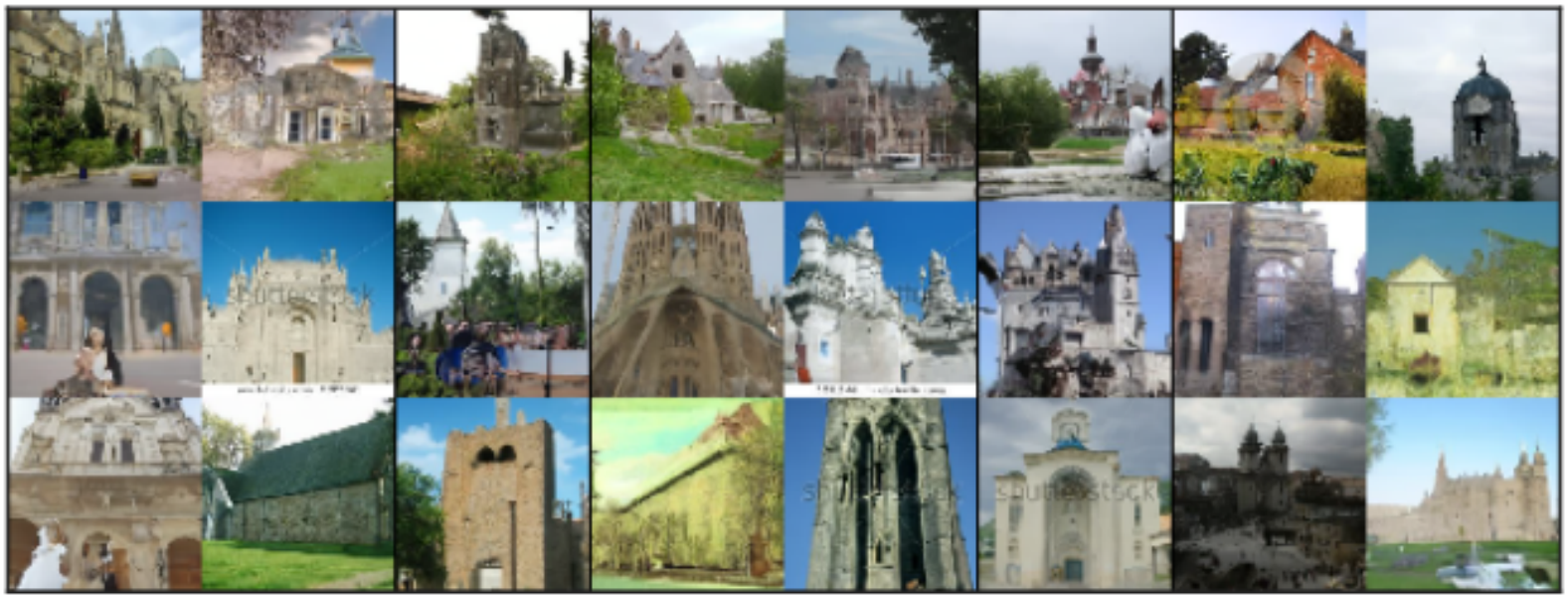
Denoising Diffusion Implicit Models (DDIM) [11]



DDPM DDIM

$$x_{t-1} = \sqrt{\alpha_{t-1}} \underbrace{\left(\frac{x_t - \sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(x_t)}{\sqrt{\alpha_t}} \right)}_{\text{“ predicted } x_0\text{”}} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_\theta^{(t)}(x_t)}_{\text{“direction pointing to } x_t\text{”}} + \underbrace{\sigma_t \epsilon_t}_{\text{random noise}}$$

Results DDPM vs DDIM



Diffusion Models Beat GANs on Image Synthesis [3]

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
  $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$
end for
return x_0

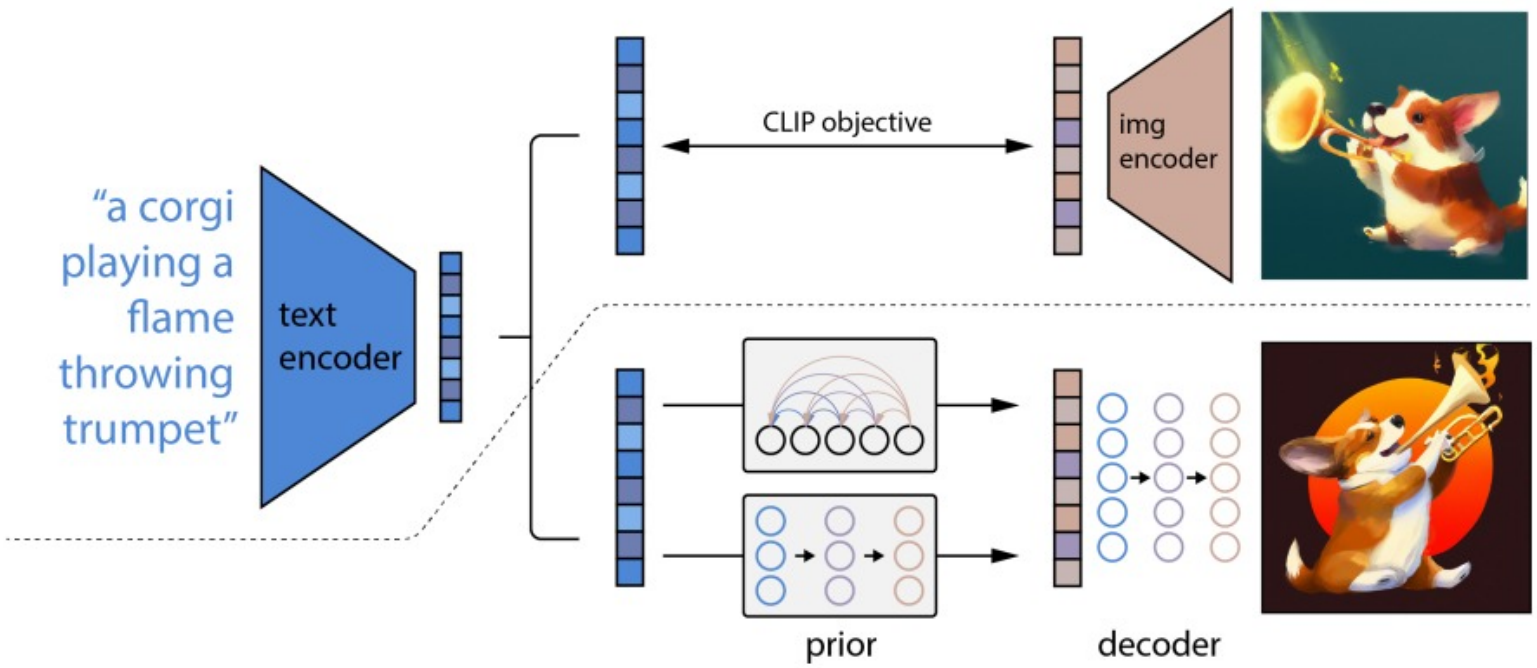
Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log p_\phi(y|x_t)$
  $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$
end for
return x_0

- Reverse diffusion process is conditioned on class label by using gradients extracted via a classifier
- Uses SOTA components mainly taken from Big-GAN

Diffusion Models Beat GANs on Image Synthesis [3]

Application: Text-to-Image with DALL-E2 [9]



- Decoder diffusion model conditions diffusion process on CLIP embedding of the encoded text
- Stable Diffusion, Googles ImageGen and Midjourney work very similar

Application: DALL-E2 [9]

a dolphin in an astronaut suit on saturn, artstation

a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese

a teddy bear on a skateboard in times square

Tradeoffs between different generative models

Method	Train Speed	Sample Speed	Num. Params.	Resolution Scaling	Free-form Jacobian	Exact Density	FID	NLL (in BPD)
Generative Adversarial Networks								
DCGAN [182]	★★★★★	★★★★★	★★★★★	★★★☆☆	✓	✗	37.11	-
ProGAN [114]	★★★★☆	★★★★★	★★★★☆	★★★★★	✓	✗	15.52	-
BigGAN [19]	★★★★☆	★★★★★	★★★★★	★★★★☆	✓	✗	14.73	-
StyleGAN2 + ADA [115]	★☆☆☆☆	★★★★★	★★★☆☆	★★★★★	✓	✗	2.42	-
Energy Based Models								
IGEBM [46]	★★★☆☆	★★☆☆☆	★★★★★	★★★☆☆	✓	✗	37.9	-
Denoising Diffusion [87]	★★★★★	★★☆☆☆	★★★☆☆	★★★★☆	✓	(✓)	3.17	≤ 3.75
DDPM++ Continuous [206]	★★★★★	★★☆☆☆	★★★☆☆	★★★★☆	✓	(✓)	2.20	-
Flow Contrastive (EBM) [55]	★☆☆☆☆	★★☆☆☆	★★☆☆☆	★★☆☆☆	✓	✗	37.30	≈ 3.27
VAEBM [247]	★★☆☆☆	★★★★☆	★☆☆☆☆	★★★★☆	✓	✗	12.19	-
Variational Autoencoders								
Convolutional VAE [123]	★★★★★	★★★★★	★★★★★	★★☆☆☆	✓	(✓)	106.37	≤ 4.54
Variational Lossy AE [29]	★★★☆☆	★☆☆☆☆	★☆☆☆☆	★★☆☆☆	✗	(✓)	-	≤ 2.95
VQ-VAE [184], [235]	★★☆☆☆	★★★☆☆	★☆☆☆☆	★★★★★	✗	(✓)	-	≤ 4.67
VD-VAE [31]	★☆☆☆☆	★★★★★	★★★☆☆	★★★★★	✓	(✓)	-	≤ 2.87
Autoregressive Models								
PixelRNN [234]	★★☆☆☆	★☆☆☆☆	★☆☆☆☆	★★★☆☆	✗	✓	-	3.00
Gated PixelCNN [233]	★★★☆☆	★☆☆☆☆	★☆☆☆☆	★★★☆☆	✗	✓	65.93	3.03
PixelIQN [173]	★★★☆☆	★☆☆☆☆	★☆☆☆☆	★★★☆☆	✗	✓	49.46	-
Sparse Trans. + DistAug [32], [110]	★★★★☆	★☆☆☆☆	★★★☆☆	★★★☆☆	✗	✓	14.74	2.66
Normalizing Flows								
RealNVP [43]	★★☆☆☆	★★★★★	★★★★★	★★★☆☆	✗	✓	-	3.49
GLOW [124]	★☆☆☆☆	★★★★★	★★☆☆☆	★★★★☆	✗	✓	45.99	3.35
FFJORD [62]	★☆☆☆☆	★★★☆☆	★★★★★	★★☆☆☆	✓	(✓)	-	3.40
Residual Flow [26]	★★☆☆☆	★★★★☆	★★★★☆	★★★★☆	✓	(✓)	46.37	3.28

References I

A. Brock, J. Donahue, and K. Simonyan.

Large scale gan training for high fidelity natural image synthesis.
arXiv preprint arXiv:1809.11096, 2018.

E. L. Denton, S. Chintala, R. Fergus, et al.

Deep generative image models using a laplacian pyramid of adversarial networks.
Advances in neural information processing systems, 28, 2015.

P. Dhariwal and A. Nichol.

Diffusion models beat gans on image synthesis.
Advances in Neural Information Processing Systems, 34:8780–8794, 2021.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio.

Generative adversarial networks.
Communications of the ACM, 63(11):139–144, 2020.

J. Ho, A. Jain, and P. Abbeel.

Denoising diffusion probabilistic models.
Advances in Neural Information Processing Systems, 33:6840–6851, 2020.

P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros.

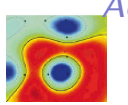
Image-to-image translation with conditional adversarial networks.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

M.-Y. Liu, T. Breuel, and J. Kautz.

Unsupervised image-to-image translation networks.
Advances in neural information processing systems, 30, 2017.

A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune.

Synthesizing the preferred inputs for neurons in neural networks via deep generator networks.
Advances in neural information processing systems, 29, 2016.



References II

A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen.

Hierarchical text-conditional image generation with clip latents.
arXiv preprint arXiv:2204.06125, 2022.

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli.

Deep unsupervised learning using nonequilibrium thermodynamics.
In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

J. Song, C. Meng, and S. Ermon.

Denoising diffusion implicit models.
arXiv preprint arXiv:2010.02502, 2020.

J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros.

Unpaired image-to-image translation using cycle-consistent adversarial networks.
In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.