Deep generative modelling

Concepts and characteristics

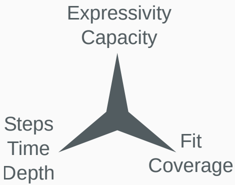
Chris G. Willcocks

Durham University

Modelling data in five equations (modelling approaches)

Today

- Five main modelling approaches $p(\mathbf{x}) \approx ?$
- Their concepts and characteristics
- Future research directions



Recent review

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

Bond-Taylor, Willcocks et al., "Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models" in IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) [1]

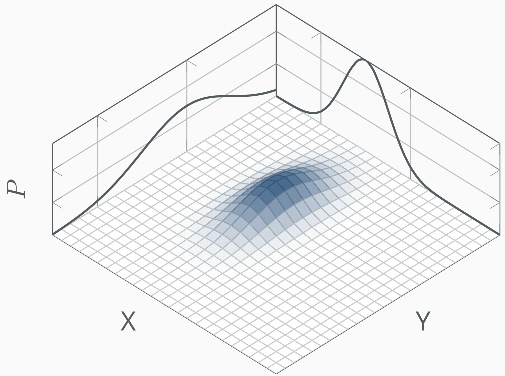
Learning the data distribution

So what is it we want exactly?

- $P(Y|X)$ discriminative model (classification)
- $P(X|Y)$ conditional generative model
- $P(X,Y)$ generative model

We want to learn the probability density function of our data (natures distribution)

The data distribution $P(X,Y)$

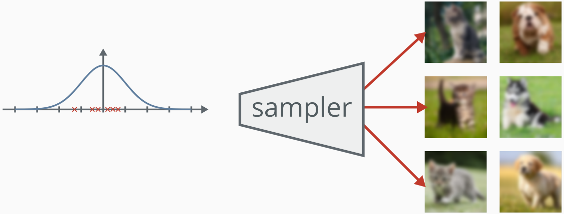


Generative models definition

Definition: Generative models learn a joint distribution over the entire dataset with some target variable(s). They are mostly used for sampling applications or density estimation:

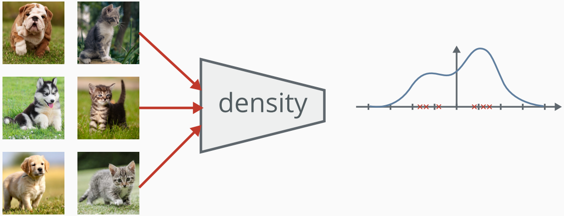
Sampling the model

A generative model learns to fit a model distribution over observations so we can sample novel data from the model distribution, $\mathbf{x}_{\text{new}} \sim p_{\text{model}}(\mathbf{x})$



Density estimation

Density estimation is estimating the probability of observations. Given a datapoint $\mathbf{x}$, what is the probability assigned by the model, $p_{\text{model}}(\mathbf{x})$?



Examples

Linguists

- What is the probability of a sentence? $P(\text{sentence})$
  - $P(\text{`the dog chased after the ball'})$
  - $P(\text{`printers eat avocados when sad'}) \approx 0$

Meteorologists

- What is the probability of whether it will rain? $P(\text{rain})$

Artists

- What is the probability of this image being a face? $P(\text{face})$

Musicians

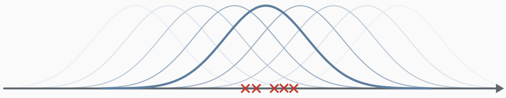
- What is the probability this sounds like Beethoven? $P(\text{Beethoven})$

Definition: maximum likelihood estimation

Maximum likelihood estimation (MLE) is a method for estimating the parameters of a probability distribution by maximizing a likelihood function, so that under the model the observed data is most probable

$$\begin{aligned}\theta^* &= \arg\max_\theta p_{\text{model}}(\mathbb{X};\theta)\\ &= \arg\max_\theta \prod_{i=1}^{n} p_{\text{model}}(\mathbf{x}^i;\theta)\\ &\approx \arg\max_\theta \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\log p_{\text{model}}(\mathbf{x};\theta)\right],\end{aligned}$$

where $\mathbb{X} = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n\}$ are from $p_{\text{data}}(\mathbf{x})$

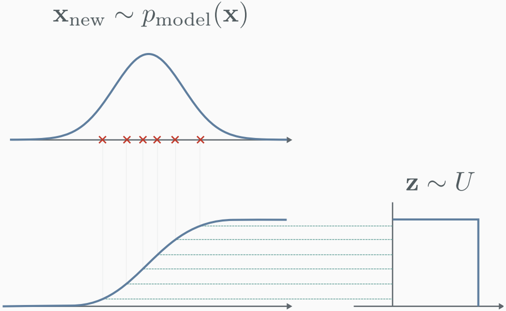


Example: cumulative distribution sampling

Given the CDF $F_X(\mathbf{x})$, the antiderivative of $f_X(\mathbf{x}) = p_{\text{model}}(\mathbf{x})$, e.g. where $F'(\mathbf{x}) = p_{\text{model}}(\mathbf{x})$

$$F_X(\mathbf{x}) = \int_{-\infty}^{\mathbf{x}} f_X(\mathbf{u})\, \mathrm{d}\mathbf{u}$$

we can sample new data by transforming random values $\mathbf{z}$ from the uniform distribution $\mathbf{z} \sim U$ via the inverse of the CDF $F_X^{-1}(\mathbf{z})$.



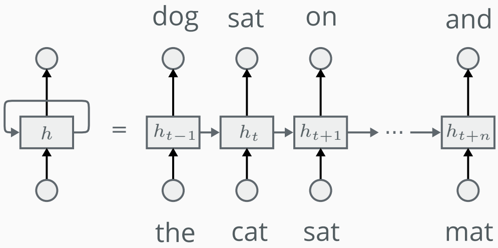
Autoregressive generative models

Definition: autoregressive (AR) generative models

AR models maximise the likelihood of the training data (excellent mode coverage):

$$p_\theta(\mathbf{x}) = p_\theta(x_1, ..., x_N) = \prod_{i=1}^{N} p_\theta(x_i | x_1, ..., x_{i-1})$$

This is slow due to the sequential nature defined by the chain rule of probability.

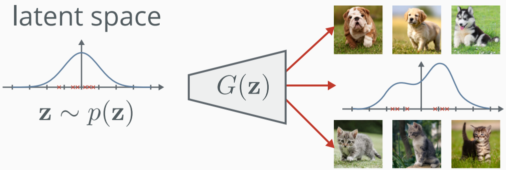


Generative networks definition

Definition: generative networks

The goal of generative networks is to take some simple distribution, like a normal distribution or a uniform distribution, and apply a non-linear transformation (e.g. a deep neural network) to obtain samples from $p_{\text{data}}(\mathbf{x})$

In 1D, we can say $G = F^{-1}_{\text{data}}(\mathbf{x})$ and sample $\mathbf{z} \sim U$, and similarly in ND — but assuming the determinant of the Jacobian and the inverse of G are computable, which is a large restriction. Ideally we want $\mathbf{z}$ in low dimensions

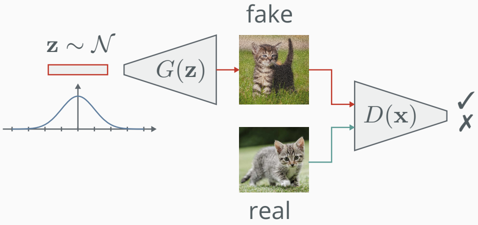


Definition: generative adversarial networks

A generative adversarial network (GAN) is a non-coorporative zero-sum game where two networks compete against each other [2].

One network $G(\mathbf{z})$ generates new samples, whereas D estimates the probability the sample was from the training data rather than G:

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

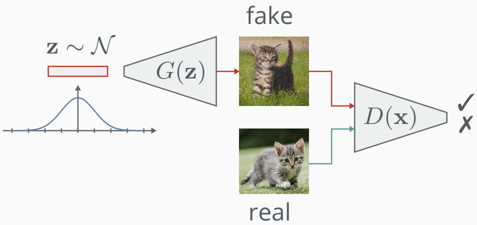


GAN **properties**

GANs benefit from differentiable data augmentation [3] for both reals and fakes, but are otherwise notoriously difficult to train:

- Non-convergence
- Diminishing gradient
- Difficult to balance
- Mode collapse (next slide)

Link to Colab example



Definition: mode collapse

This is where the generator rotates through a small subset of outputs, and the discriminator is unable to get out of the trap. Mode collapse is arguably the main limitation of GANs.

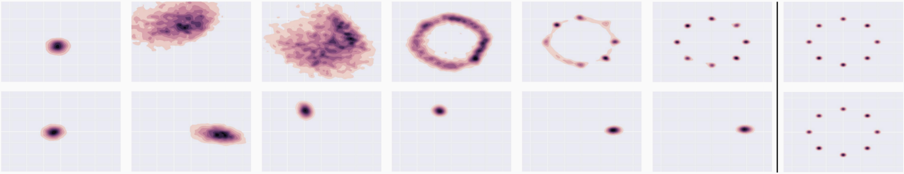


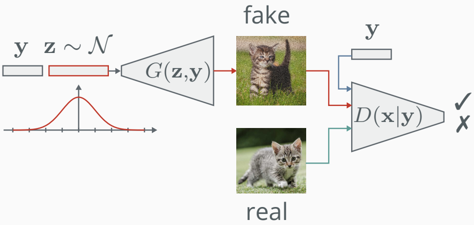
Figure from [4]. The final column shows the target data distribution and the bottom row shows a GAN rotating through the modes.

Definition: conditional GAN

GANs can be conditioned with labels $\mathbf{y}$ if available [5] by feeding the label information into both the generator and the discriminator:

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

Link to Colab example



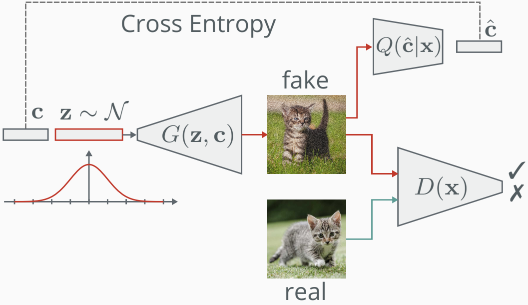
Definition: information maximizing GANs

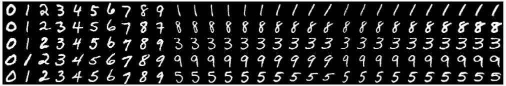
GANs can be trained to learn disentangled latent representations in a completely unsupervised manner. InfoGAN [6] popularised this by maximizing mutual information between the observation and a subset of the latents:

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

where $L_I(G,Q)$ is a variational lower bound of the mutual information.

Link to Colab example

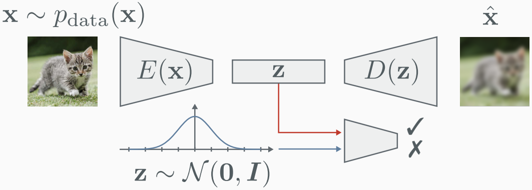




Definition: adversarial autoencoders

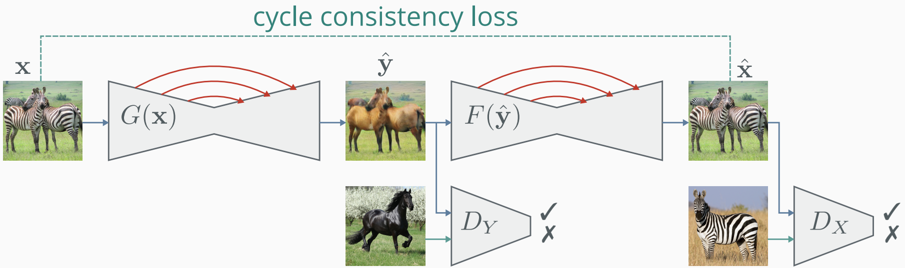
Adversarial autoencoders [7] are generative models that permit sampling.

In addition to the reconstruction loss, such $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$, they use adversarial training to match the aggregated posterior of the hidden code vector $\mathbf{z}$ of the autoencoder with an arbitrary prior distribution, such as $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$.



Definition: unpaired translation (CycleGAN)

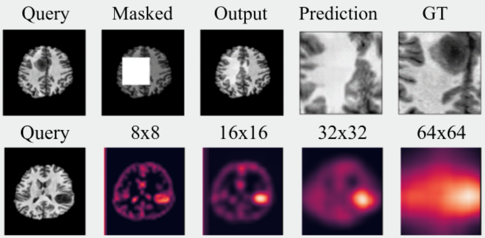
CycleGAN [8] propose an adversarial architecture that enables unpaired image translation. It has twin residual generators and two discriminators, which translate between the domains, alongside a cycle consistency loss (an L1 norm) which ensures the mapping can recover the original image.

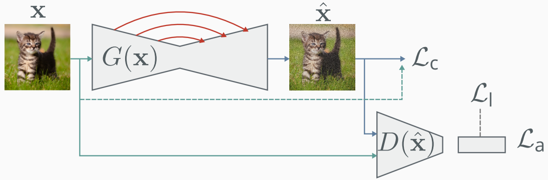


Popular applications adversarial anomaly detection

Definition: anomaly detection

Unsupervised anomaly detectors [9] learn a normal distribution over (healthy) observations. Then, when they observe something not observed in training (unhealthy/dangerous), they fail to reconstruct - detecting it as an anomaly. Region-based anomaly detectors [10] learn a distribution over inpainted (erased) regions.

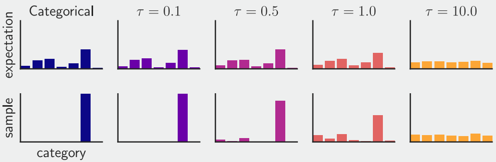




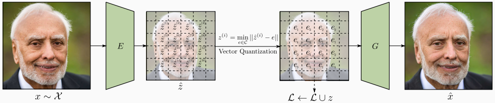
Vector quantization

Vector quanitization

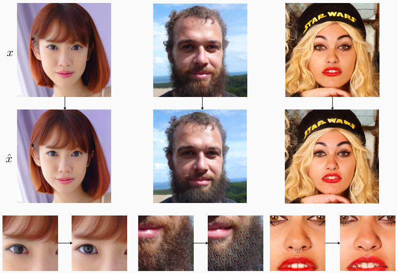
Imposing a discrete prior on the latents can be achieved with either variational or adversarial (non-blurry) approaches.



The Gumbel-Softmax distribution interpolates between discrete one-hot-encoded categorical distributions and continuous categorical densities.



Above: vector quantisation. **Below:** shift mode collapse to perceptually unimportant parts of the signal.



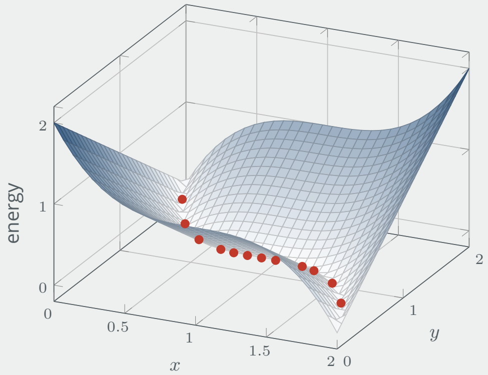
Definition: energy-based models

These are just any function that is happy when you input something that looks like data, and is not happy when you input something that doesn't look like data.

$$E(\mathbf{x}) = 0 \checkmark$$
$$E(\tilde{\mathbf{x}}) > 0 \text{ ✗}$$

This generic definition fits a large majority of machine learning models. For example $\mathcal{L}(E(\mathbf{x}), \mathbf{y})$ (a classifier)

Energy increases off manifold



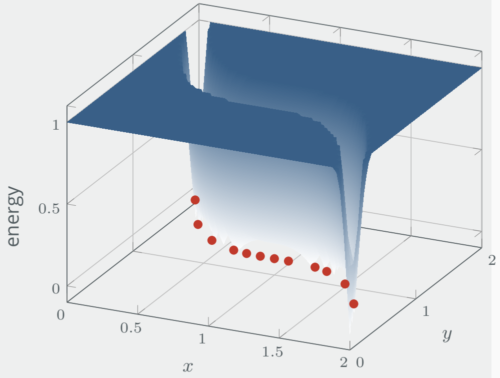
Definition: energy-based models

GANs are also energy models. The generator G generates samples off the manifold, then the descriminator D says these should be one everywhere, whereas it says real samples should be zero everywhere.

The generator also has to get good at sampling points on the data manifold. So it has to learn to generate points in the valley regions.

Is this smooth? What does a 1-Lipschitz discriminator do to the energy landscape?

GAN energy



Energy-based models clustering as an energy-based model

Definition: clustering algorithm

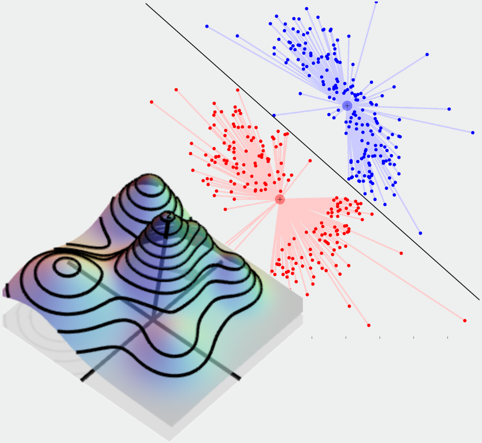
A cluster is a **connected-component** of a **level-set** of the **unknown PDF** over our data observations.

Traditionally:

- We don't know the PDF (the energy landscape)
- We don't necessarily know the level set
    - although 0.5 is appropriate for BCE
- This can be expensive (deep learning)

Click to watch a video that visually explains from the definition ▶

Example: clustering by its definition



Definition: softmax and softmin

Softmax and softmin functions rescale elements to be in the range [0, 1] and such that they sum to 1. So they create a probability mass function, e.g.:

$$\begin{bmatrix} 1.3 \\ 7.2 \\ 2.4 \\ 0.5 \\ 1.1 \end{bmatrix} \rightarrow \frac{e^{\mathbf{z}_i}}{\sum_{j=1}^{K} e^{\mathbf{z}_j}} \rightarrow \begin{bmatrix} 0.0027 \\ 0.9858 \\ 0.0081 \\ 0.0012 \\ 0.0022 \end{bmatrix}$$

Softmax functions are widely used (not just for EBMs) where a distribution is needed, such as the last layer of a classifier.

Energy-based models exact likelihood

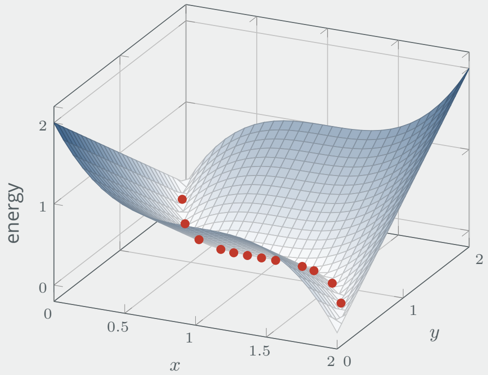
Challenges: energy-based models

EBMs are based on the observation that any probability density function $p(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^n$ can be expressed as:

$$p(\mathbf{x}) = \frac{e^{-E(\mathbf{x})}}{\int_{\tilde{\mathbf{x}} \in \mathcal{X}} e^{-E(\tilde{\mathbf{x}})}},$$

where $E(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ is the energy function. However computation of the integral is intractable [11] for most models.

Energy increases off manifold



Definition: score-based GMs

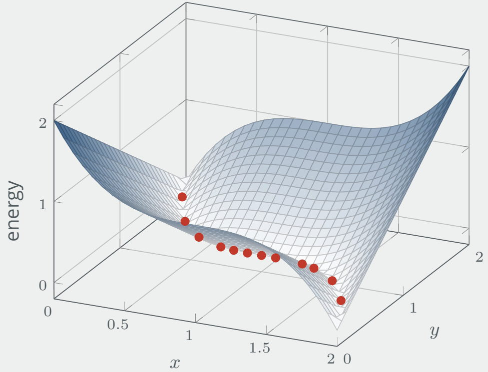
Score-based generative modeling [12] also eliminates the intractable second term (sampling from the model). For the PDF $p(\mathbf{x})$ the score function is:

$$s(\mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x})$$

When the score function is known, we can use Langevin dynamics to sample the model. Given a step size $\alpha > 0$, a total number of iterations T, and an initial sample $x0$ from any prior distribution $\pi(\mathbf{x})$, Langevin dynamics iteratively updates:

$$\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \alpha \nabla_{\mathbf{x}} \log p(\mathbf{x}_{t-1}) + \sqrt{2\alpha}\, \mathbf{z}_t$$

Energy increases off manifold



Diffusion probablistic modeling

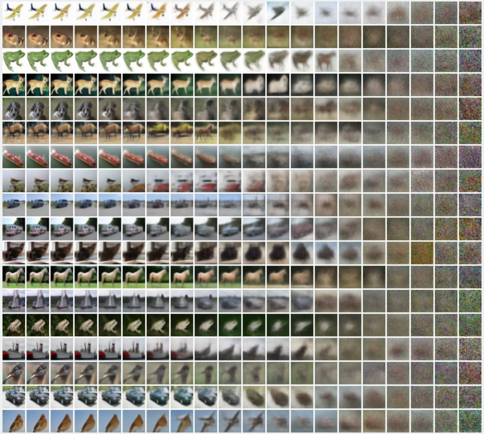
Diffusion Probablistic Modeling approaches (such as DDPMs [13]) typically have a U-Net shaped architecture:

Data is gradually diffused in a forward process for T timesteps until it matches the target distribution.

The reverse process gradually removes noise starting at $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$ for T timesteps.

'Score-Based Generative Modeling through Stochastic Differential Equations' [14] has author code and PyTorch tutorials in the link.

Example: CIFAR10 samples from [13]



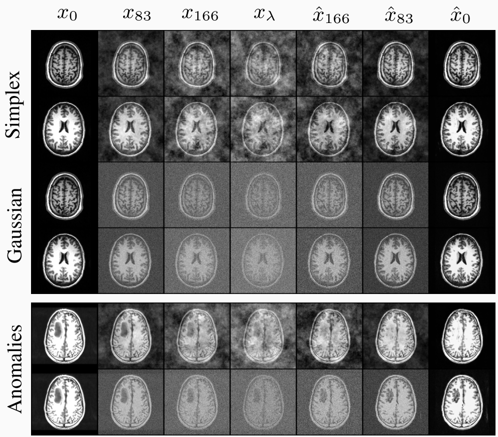
Diffusion-based anomaly detection

Diffusion-based anomaly detection

Like GANs, diffusion-based models work well for anomalies (great for small datasets).

- Do a partial diffusion
- Train only on healthy/normal data
- Abnormal denoising will only know how to make the data look normal
- Any error = surprise = anomalies

Our recent paper, AnoDDPM [15] (CVPR NTIRE), uses simplex noise to capture multi-scale anomalies. See also UNIT-DDPM [16] (unpaired translation).

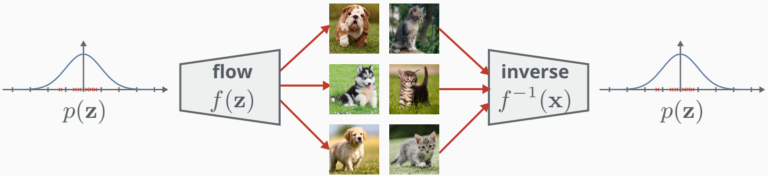


Link to project page

Flow models definition

Definition: flow models

Flow models restrict our function to be a chain of invertible functions, called a flow, therefore the whole function is invertible.



Recap: the determinant

The determinant of an $n \times n$ square matrix M is a scalar value that determines the factor of how much a given region of space increases or decreases by the linear transformation of M:

$$\det M = \det \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} = \sum_{j_1 j_2 \dots j_n} (-1)^{\tau(j_1 j_2 \dots j_n)} a_{1j_1} a_{2j_2} \dots a_{nj_n}$$

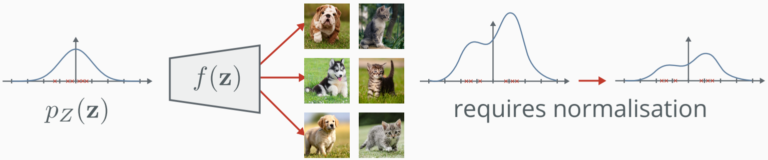
Watch a 3Blue1Brown's video here

PyTorch: `torch.det(M)`, for example: `torch.det(torch.eye(3,3))` returns `1.0` and `torch.det(torch.tensor([[3.,2.],[0.,2.]]))` returns `6.0`

Definition: the change of variables theorem

Given $p_Z(\mathbf{z})$ where $\mathbf{x} = f(\mathbf{z})$ and $\mathbf{z} = f^{-1}(\mathbf{x})$ we ask what is $p_X(\mathbf{x})$?

$$p_X(\mathbf{x}) = p_Z(f^{-1}(\mathbf{x})) \left| \det \left(\frac{\partial f^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$

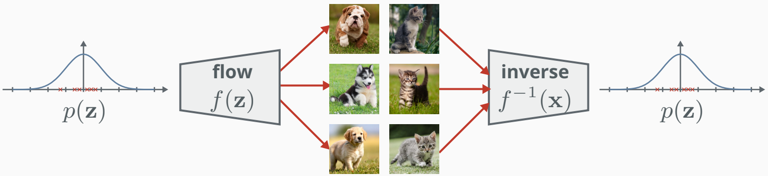


Definition: normalising flows

Normalising flows $f : \mathbb{R}^n \to \mathbb{R}^n$ transform and renormalise a sample $\mathbf{z} \sim p_\theta(\mathbf{z})$ through a chain of bijective transformations f, where:

$$\mathbf{x} = f_\theta(\mathbf{z}) = f_K \circ \cdots \circ f_2 \circ f_1(\mathbf{z})$$

$$\log p_\theta(\mathbf{x}) = \log p_\theta(\mathbf{z}) + \sum_{i=1}^{K} \log \left| \det \left(\frac{\partial f_i^{-1}}{\partial \mathbf{z}_i} \right) \right|$$



Easy to compute determinants

We have a sequence of high-dimensional bijective functions, where we need to compute the Jacobian determinants.

Computing the determinants can be expensive, so most of the literature focuses on restricting the function f^{-1} to those with easy-to-compute Jacobian determinants.

This is done by ensuring the Jacobian matrix of the functions is triangular.

Definition: triangular Jacobian

If the Jacobian is lower triangular:

$$J = \begin{bmatrix} a_{1,1} & & & & 0 \\ a_{2,1} & a_{2,2} & & & \\ a_{3,1} & a_{3,2} & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n-1} & a_{n,n} \end{bmatrix}$$

then the determinant is simply the product of its diagonals.

Normalising flows normalising flow layers

Description	Function	Log-Determinant
Additive Coupling [17]	$\mathbf{y}^{(1:d)} = \mathbf{x}^{(1:d)}$ $\mathbf{y}^{(d+1:D)} = \mathbf{x}^{(d+1:D)} + f(\mathbf{x}^{(1:d)})$	0
Planar [18]	$\mathbf{y} = \mathbf{x} + \mathbf{u}h(\mathbf{w}^T\mathbf{z} + b)$ With $\mathbf{w} \in \mathbb{R}^D$, $\mathbf{u} \in \mathbb{R}^D$, $\mathbf{b} \in \mathbb{R}$	$\ln\|1 + \mathbf{u}^T h'(\mathbf{w}^T\mathbf{z} + b)\mathbf{w}\|$
Affine Coupling [19]	$\mathbf{y}^{(1:d)} = \mathbf{x}^{(1:d)}$ $\mathbf{y}^{(d+1:D)} = \mathbf{x}^{(d+1:D)} \odot f_\sigma(\mathbf{x}^{(1:d)}) + f_\mu(\mathbf{x}^{(1:d)})$	$\sum_1^d \ln\|f_\sigma(x^{(i)})\|$
Batch Normalization [19]	$\mathbf{y} = \frac{\mathbf{x}-\tilde{\mu}}{\sqrt{\tilde{\sigma}^2+\epsilon}}$	$-\frac{1}{2}\sum_i \ln(\tilde{\sigma}_i^2 + \epsilon)$
1x1 Convolution [20]	With $h \times w \times c$ tensor $\mathbf{x}$ & $c \times c$ tensor $\mathbf{W}$ $\forall i, j : \mathbf{y}_{i,j} = \mathbf{W}\mathbf{x}_{i,j}$	$h \cdot w \cdot \ln\|\det \mathbf{W}\|$
i-ResNet [21]	$\mathbf{y} = \mathbf{x} + f(\mathbf{x})$ where $\|f\|_L < 1$	$\mathrm{tr}(\ln(\mathbf{I} + \nabla_\mathbf{x} f)) =$ $\sum_{k=1}^\infty (-1)^{k+1} \frac{\mathrm{tr}((\nabla_\mathbf{x} f)^k)}{k}$
Emerging Convolutions [22]	$\mathbf{k} = \mathbf{w}_1 \odot \mathbf{m}_1, \quad \mathbf{g} = \mathbf{w}_2 \odot \mathbf{m}_2$ $\mathbf{y} = \mathbf{k} \star_l (\mathbf{g} \star_l \mathbf{x})$	$\sum_c \ln\|\mathbf{k}_{c,c,m_y,m_x}\mathbf{g}_{c,c,m_y,m_x}\|$

Generative networks variational autoencoders

Definition: variational autoencoders

Variational autoencoders are generative models, as they impose a prior over the latent space $p(\mathbf{z})$, typically $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ which can be sampled from.

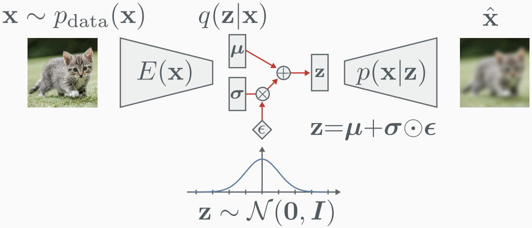
$$\mathbf{z} \sim E(\mathbf{x}) = q(\mathbf{z}|\mathbf{x}), \quad \hat{\mathbf{x}} \sim D(\mathbf{z}) = p(\mathbf{x}|\mathbf{z})$$

The VAE loss is the negated expected log-likelihood (the reconstruction error) and the prior regularization term:

$$\mathcal{L}_{\text{VAE}} = -\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})}\right] = \mathcal{L}_{\text{recon}}^{\text{pixel}} + \mathcal{L}_{\text{prior}}$$

where

$$\mathcal{L}_{\text{recon}}^{\text{pixel}} = -\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})]$$
$$\mathcal{L}_{\text{prior}} = D_{\text{KL}}(q(\mathbf{z}|\mathbf{x}) \,||\, p(\mathbf{z}))$$



Definition: ELBO

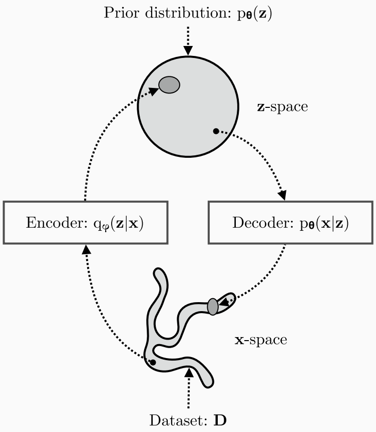
VAEs therefore have three components:

1. the decoder $p_\theta(\mathbf{x}|\mathbf{z})$
2. the *approximate posterior* (encoder) $q_\phi(\mathbf{z}|\mathbf{x})$
3. the prior distribution $p_\theta(\mathbf{z})$

They are trained with the reparameterisation trick to maximise the evidence lower bound (ELBO):

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - D_{\text{KL}}\left[q_\phi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z})\right]$$

Read [23] for detail on the theory (where the figure is from) and [24] for a state-of-the-art method that stacks VAEs hierarchically (Very Deep VAEs).



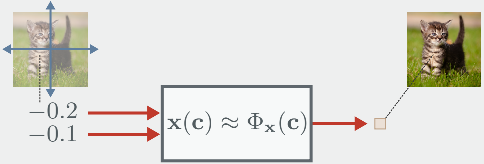
Definition: implicit representations

Consider data $\Phi\colon \mathbb{R}^m \to \mathbb{R}^n$, like a single image, as a function of coordinates $\mathbf{c} \in \mathbb{R}^m$. The aim is to learn a neural approximation of Φ that satisfies an implicit equation:

$$R(\mathbf{c}, \Phi, \nabla_\Phi, \nabla^2_\Phi, \dots) = 0, \quad \Phi\colon \mathbf{c} \mapsto \Phi(\mathbf{c}).$$

Equations with this structure arise in a myriad of fields, namely 3D modelling, image, video, and audio representation.

Example: implicit network



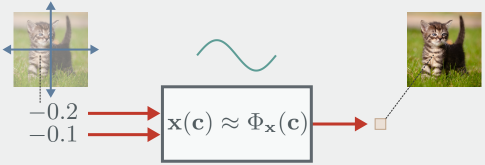
Implicit representation networks SIREN

Definition: SIREN

SInusoidal REpresentation Networks (SIREN) are a simple implicit representation network with fully connected layers, but use `sin` (with clever initialisation to scale it appropriately) as their choice of non-linearity [25].

`sin` is periodic, so it allows to capture patterns over all of the coordinate space (it's translation invariant, like convolutions).

Example: SIREN (implicit network)

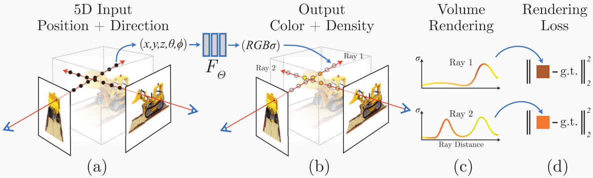


Link to project page

Definition: NeRF

Neural Radiance Fields (NeRF) are similar to SIRENs, but instead of representing an image, they represent a single 3D scene [26].

They map from pixel positions (x, y, z) and a viewing direction (θ, ϕ) to a colour and density value σ integrated via a ray on F_θ.



Link to project page

Definition: gradient origin networks

Gradient origin networks (GON) treat the derivative of the decoder as an encoder [27]. This allows us to compute the latents:

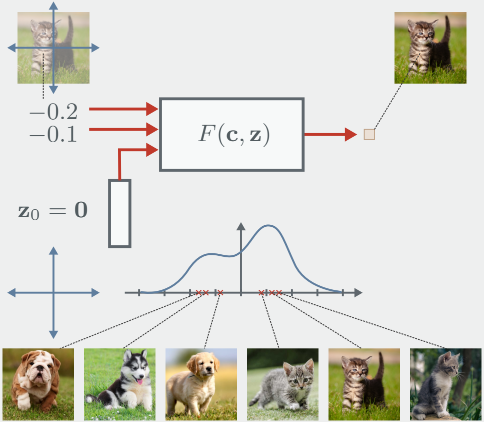
$$\mathbf{z} = -\nabla_{\mathbf{z}_0} \mathcal{L}(\mathbf{x}, F(\mathbf{z}_0))$$

which are then jointly optimised, giving the GON objective:

$$G_{\mathbf{x}} = \mathcal{L}(\mathbf{x}, F(-\nabla_{\mathbf{z}_0} \mathcal{L}(\mathbf{x}, F(\mathbf{z}_0)))).$$

Link to project page

Example: implicit GON



Hybrids combinations of the five modelling equations

Summary

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

$$p(\mathbf{x}) \approx \frac{e^{-E(\mathbf{x})}}{\int_{\tilde{\mathbf{x}} \in \mathcal{X}} e^{-E(\tilde{\mathbf{x}})}} \quad \text{e.g. } \nabla_{\mathbf{x}} \log p(\mathbf{x})$$

$$\log p(\mathbf{x}) \geq \mathcal{L}^{\text{pixel}}_{\text{recon}} - D_{\text{KL}} \left[q_\phi(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}) \right]$$

$$\log p(\mathbf{x}) \neq \log D(\mathbf{x}) \quad \text{(in GAN)}$$

$$p(\mathbf{x}) = p_Z(f^{-1}(\mathbf{x})) \left| \det \left(\frac{\partial f^{-1}(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$

Our hybrids “Unleashing transformers” [28] (ECCV22) or “Megapixel image generation” (new) [29].

Our hybrid [29] 2 seconds generation, 2 days training, single GTX 1080Ti

Take Away Points

Contributions

- State-of-the-art = hybrids
- Some applications only need partial diffusion (AnoDDPM, UNIT-DDPM)
- Start of non-hybrid generative implicit networks (GONs)—we need new interpolated modelling theory please (more like [30])

Tips, tricks and the future

- Eventually move discriminative modelling tasks to generative modelling
- Measuring progress sucks (not just quality/performance)
- For vision state-of-the-art:
    - Intentionally mode collapse parts of signal you don't care about
    - Model with good coverage + quality (AR,EBM) the remaining signal

References I

[1] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G. Willcocks. “Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (2021), pp. 1–1. DOI: `10.1109/TPAMI.2021.3116668`.

[2] Ian Goodfellow et al. “Generative adversarial nets”. In: Advances in neural information processing systems. 2014, pp. 2672–2680.

[3] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. “Differentiable augmentation for data-efficient gan training”. In: arXiv preprint arXiv:2006.10738 (2020).

[4] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. “Unrolled generative adversarial networks”. In: arXiv preprint arXiv:1611.02163 (2016).

[5] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. In: arXiv preprint arXiv:1411.1784 (2014).

References II

[6] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. “InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets”. In: Advances in neural information processing systems. 2016, pp. 2172–2180.

[7] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. “Adversarial autoencoders”. In: arXiv preprint arXiv:1511.05644 (2015).

[8] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: Proceedings of the IEEE international conference on computer vision. 2017, pp. 2223–2232.

[9] Samet Akçay, Amir Atapour-Abarghouei, and Toby P Breckon. “Skip-ganomaly: Skip connected and adversarially trained encoder-decoder anomaly detection”. In: 2019 International Joint Conference on Neural Networks (IJCNN). IEEE. 2019, pp. 1–8.

References III

[10] Bao Nguyen, Adam Feldman, Sarath Bethapudi, Andrew Jennings, and Chris G Willcocks. "Unsupervised Region-based Anomaly Detection in Brain MRI with Adversarial Image Inpainting". In: arXiv preprint arXiv:2010.01942 (2020).

[11] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. "A tutorial on energy-based learning". In: Predicting structured data 1.0 (2006).

[12] Yang Song and Stefano Ermon. "Improved techniques for training score-based generative models". In: arXiv preprint arXiv:2006.09011 (2020).

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: arXiv preprint arXiv:2006.11239 (2020).

[14] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. "Score-Based Generative Modeling through Stochastic Differential Equations". In: International Conference on Learning Representations. 2021. URL: `https://openreview.net/forum?id=PxTIG12RRHS`.

References IV

[15] Julian Wyatt, Adam Leach, Sebastian M Schmon, and Chris G Willcocks. "AnoDDPM: Anomaly Detection With Denoising Diffusion Probabilistic Models Using Simplex Noise". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022, pp. 650–656.

[16] Hiroshi Sasaki, Chris G Willcocks, and Toby P Breckon. "Unit-ddpm: Unpaired image translation with denoising diffusion probabilistic models". In: arXiv preprint arXiv:2104.05358 (2021).

[17] Laurent Dinh, David Krueger, and Yoshua Bengio. "Nice: Non-linear independent components estimation". In: arXiv preprint arXiv:1410.8516 (2014).

[18] Danilo Jimenez Rezende and Shakir Mohamed. "Variational inference with normalizing flows". In: arXiv preprint arXiv:1505.05770 (2015).

[19] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real nvp". In: arXiv preprint arXiv:1605.08803 (2016).

References V

[20] Durk P Kingma and Prafulla Dhariwal. “Glow: Generative flow with invertible 1x1 convolutions”. In: Advances in neural information processing systems. 2018, pp. 10215–10224.

[21] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. “Invertible residual networks”. In: International Conference on Machine Learning. 2019, pp. 573–582.

[22] Emiel Hoogeboom, Rianne van den Berg, and Max Welling. “Emerging convolutions for generative normalizing flows”. In: arXiv preprint arXiv:1901.11137 (2019).

[23] Diederik P Kingma and Max Welling. “An introduction to variational autoencoders”. In: arXiv preprint arXiv:1906.02691 (2019).

[24] Rewon Child. “Very Deep {VAE}s Generalize Autoregressive Models and Can Outperform Them on Images”. In: International Conference on Learning Representations. 2021. URL: https://openreview.net/forum?id=RLRXCV6DbEJ.

References VI

[25] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. “Implicit neural representations with periodic activation functions”. In: Advances in Neural Information Processing Systems 33 (2020).

[26] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. “Nerf: Representing scenes as neural radiance fields for view synthesis”. In: European conference on computer vision. Springer. 2020, pp. 405–421.

[27] Sam Bond-Taylor and Chris G. Willcocks. “Gradient Origin Networks”. In: International Conference on Learning Representations. 2021. URL: `https://dro.dur.ac.uk/34356/1/34356.pdf`.

[28] Sam Bond-Taylor, Peter Hessey, Hiroshi Sasaki, Toby P Breckon, and Chris G Willcocks. “Unleashing Transformers: Parallel Token Prediction with Discrete Absorbing Diffusion for Fast High-Resolution Image Generation from Vector-Quantized Codes”. In: European Conference on Computer Vision (ECCV) (2022).

References VII

[29] Alex F McKinney and Chris G Willcocks. "Megapixel Image Generation with Step-Unrolled Denoising Autoencoders". In: arXiv preprint arXiv:2206.12351 (2022).

[30] Patrick Kidger. "On neural differential equations". In: arXiv preprint arXiv:2202.02435 (2022).