

Neural Networks and Deep Learning

Lecture 8: Generative models: autoencoders, VAE, GAN, diffusion

Prof. dr. Daniel Traian PELE

Bucharest University of Economic Studies (ASE), Department of Statistics and Econometrics
Romanian Academy, Institute for Economic Forecasting (IPE)
IDA Institute Digital Assets ▪ AI4EFin ▪ MSCA Digital Finance

Learning objectives

- Explain the difference between a discriminative model $p(y \mid \mathbf{x})$ and a generative model $p(\mathbf{x})$, and why finance needs both.
- Derive and train four families: the autoencoder (AE), the variational autoencoder (VAE), the generative adversarial network (GAN) and the denoising diffusion probabilistic model (DDPM).
- Use them on real data: card-fraud detection, clothing images in the format of the Modified National Institute of Standards and Technology (MNIST) digits, S&P 500 returns and the euro-area yield curve, each against a classical benchmark.
- Judge synthetic data: tails, volatility clustering, risk measures such as Value-at-Risk (VaR), and what a generator cannot reproduce.

Abbreviations

AE	autoencoder
ASE	Bucharest University of Economic Studies
DDPM	denoising diffusion probabilistic model
ELBO	evidence lower bound
ES	expected shortfall
GAN	generative adversarial network
IPE	Institute for Economic Forecasting, Romanian Academy
JS	Jensen–Shannon divergence
KL	Kullback–Leibler divergence
LLM	large language model
MLP	multilayer perceptron
MNIST	Modified National Institute of Standards and Technology handwritten digits
PCA	principal component analysis
PR-AUC	area under the precision-recall curve
ReLU	rectified linear unit
ROC	receiver operating characteristic
ROC-AUC	area under the ROC curve
TSTR	train on synthetic, test on real
VAE	variational autoencoder
VaR	Value-at-Risk
WGAN	Wasserstein GAN

Outline

- Principles of generative modelling
- Autoencoders: anomaly detection for card fraud
- Variational autoencoders
- Generative adversarial networks
- Diffusion models
- Generative models in practice
- Wrap-up

A generative model learns how the data themselves arise

- **In one sentence:** A classifier learns the boundary between classes; a generative model learns the whole distribution, so it can score, simulate and fill in data.

$$\underbrace{p(y \mid \mathbf{x})}_{\text{discriminative}} \quad \text{vs.} \quad \underbrace{p(\mathbf{x}) \text{ or } p(\mathbf{x}, y) = p(\mathbf{x} \mid y)\, p(y)}_{\text{generative}}$$

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log p_\theta(\mathbf{x}_i)$$

- **where**
  - $\mathbf{x}$ = an observation (transaction, image, day of returns, yield curve)
  - y = a label
  - p_θ = model density, fitted by maximum likelihood.

- **Four uses in finance**
  - Anomaly detection: low $p(\mathbf{x})$ = suspicious (fraud)
  - Scenarios for stress tests and risk measures
  - Synthetic data shared without client records

- **In practice:** a classifier ranks clients, a generative model also shows what could happen.

(Bishop & Bishop, 2024, ch. 16)

Latent-variable models

⊡ **In one sentence:** Eleven yields or 784 pixels are driven by a handful of hidden factors; a latent-variable model writes this down.

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}, \qquad \mathbf{z} \in \mathbb{R}^k,\ k \ll d$$

linear-Gaussian: $\mathbf{x} = W\mathbf{z} + \boldsymbol{\mu} + \varepsilon,\ \mathbf{z} \sim \mathcal{N}(0, I),\ \varepsilon \sim \mathcal{N}(0, \sigma^2 I)$

⊡ **where**

- $\mathbf{z}$ = latent factors, k of them
- d = dimension of the data
- The linear-Gaussian case is probabilistic principal component analysis (PCA), with a closed-form likelihood
- With a neural decoder the integral has no closed form.

⊡ **Yield curve**

- Level, slope, curvature: three factors explain most daily moves

⊡ **The price of non-linearity**

- VAE: a lower bound on $\log p$; GAN: no likelihood at all; diffusion: a bound built step by step

⊡ **In practice:** in a model review, ask which part is the latent story and which part is fitted noise.

(Bishop & Bishop, 2024, sec. 16.2)

Four families of generative models

Family	Training signal	Likelihood	Sampling	In this lecture
AE	reconstruction error	none	not a generator	card fraud as anomaly
VAE	evidence lower bound	lower bound	one pass, blurry	clothing images; yield curves
GAN	game vs a discriminator	none	one pass, sharp	S&P 500 daily returns
DDPM	denoising error	lower bound	many steps, faithful	euro-area yield curves

- **No free lunch**: likelihood, sample quality and speed rarely come together
  - The choice depends on the use.
- **Normalising flows** (exact likelihood, invertible networks) complete the picture
  - See Bishop & Bishop (2024), ch. 18.

(Bishop & Bishop, 2024, ch. 17–20)

Outline

- Principles of generative modelling
- Autoencoders: anomaly detection for card fraud
- Variational autoencoders
- Generative adversarial networks
- Diffusion models
- Generative models in practice
- Wrap-up

An autoencoder compresses the data through a bottleneck and reconstructs them

- **In one sentence:** If the network rebuilds normal transactions from a few numbers, one it cannot rebuild is unusual.

$$\mathbf{z} = f_\phi(\mathbf{x}) \in \mathbb{R}^k, \qquad \hat{\mathbf{x}} = g_\psi(\mathbf{z})$$

$$\min_{\phi,\psi} \frac{1}{n} \sum_{i=1}^{n} \left\| \mathbf{x}_i - g_\psi(f_\phi(\mathbf{x}_i)) \right\|^2, \qquad s(\mathbf{x}) = \tfrac{1}{d} \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

- **where**
  - f_ϕ, g_ψ = encoder and decoder, small multilayer perceptrons (MLP) with rectified linear units (ReLU)
  - k = bottleneck size
  - $s(\mathbf{x})$ = anomaly score
  - Linear f, g span the PCA subspace.

- **Our autoencoder**
  - 29 inputs (28 anonymised components + log amount) $\to 16 \to 8 \to 16 \to 29$
  - 1 253 parameters, trained on legitimate transactions only
- **No labels needed**: useful for new or unconfirmed frauds

- **In practice:** an autoencoder models "normal" and flags what differs, not necessarily what is criminal.

(Bishop & Bishop, 2024, ch. 19)

Experiment yourself: notebook §1

The fraud experiment: two days of card payments, split in time

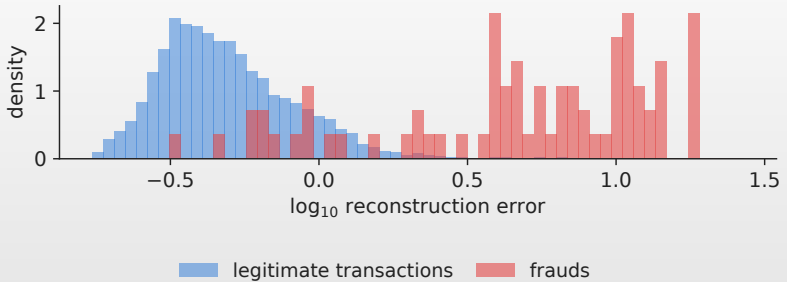
Element	Choice
Data	ULB credit-card transactions (Dal Pozzolo et al., 2015): 28 924 rows = all 492 frauds + 10 % of the legitimate ones, re-weighted to the full data
Split in time	first 60 % of the time span = training (legitimate only, 16 994 rows), next 20 % = validation, last 20 % = test
Test	57 174 transactions of the full data, 75 frauds (0.131 %)
Benchmarks	PCA with 8 components (linear autoencoder); logistic regression trained *with* the fraud labels
Metrics	area under the precision-recall curve (PR-AUC) and the area under the receiver operating characteristic curve (ROC-AUC), weighted to the full data; frauds among the top 0.1 % scores

- **Leakage-free**: means and standard deviations from the legitimate training transactions
  - Early stopping on the validation part
  - Seeds 1–3.

(Dal Pozzolo et al., 2015)

Experiment yourself: notebook §1

Reconstruction error of fraudulent and legitimate payments

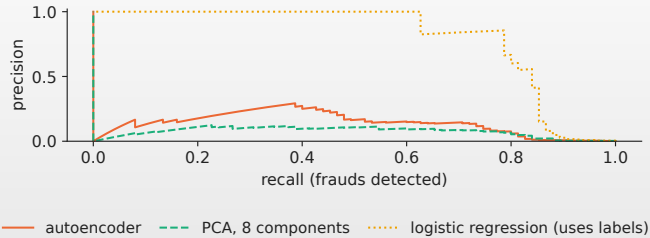


Source: ULB Machine Learning Group and Worldline, credit-card transactions (OpenML 1597). Data: own estimates; two days of September 2013; test = last 20 % of the time span.

- **Separation**: the median error of a fraud is 12.7 times that of a legitimate transaction (test part).
- **Overlap**: many frauds still look normal to the autoencoder
  - They sit in the bulk of the legitimate distribution.

NN_L8_ae_error_dist

Autoencoder without labels compared with a supervised classifier

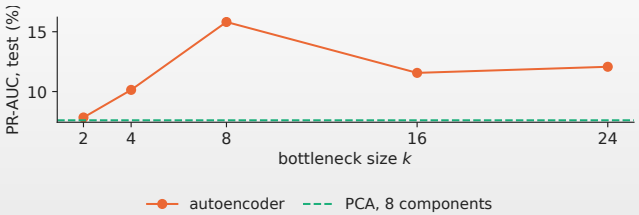


Source: ULB Machine Learning Group and Worldline, credit-card transactions (OpenML 1597). Data: own estimates; two days of September 2013; test = last 20 % of the time span.

- **PR-AUC on the test part**: autoencoder 14.2 % (sd 2.9 over 3 seeds), PCA 7.6 %, logistic regression with labels 80.4 %
  - A random score: 0.131 %.
- **ROC-AUC** is high for all three (94.4, 95.3, 98.2 %): with 1 fraud in 700 it hides the difference that PR-AUC shows.

NN_L8_ae_pr

Bottleneck size and anomaly detection



Source: ULB Machine Learning Group and Worldline, credit-card transactions (OpenML 1597). Data: own estimates; two days of September 2013; test = last 20 % of the time span.

- **PR-AUC by bottleneck size**: 7.8 % with $k = 2$, 15.8 % with $k = 8$, 12.1 % with $k = 24$ (mean of 2 seeds)
  - Best $k = 8$.
- **Too narrow**: even normal payments are rebuilt badly
  - **too wide**: the network learns to copy frauds as well.

- **In practice:** choose k on a validation period, never on the frauds you will report on.

NN_L8_ae_bottleneck

Frauds found among the top 0.1 % of scores

Detector	Frauds found	Share of frauds
Autoencoder (no labels)	8	10.7 %
PCA, 8 components (no labels)	0	0.0 %
Logistic regression (labels)	48	64.0 %

- **Alert budget**
  - 0.1 % of the test transactions = 57 alerts to review
- **Reading**
  - Labels are worth a lot: supervised learning wins whenever confirmed frauds exist
  - The autoencoder is the detector of the unknown, a complement, not a replacement

- **In practice:** run both: the classifier for known fraud patterns, the autoencoder to surface new ones for investigators to label.

(own computation, test part, weighted to the full data)

Experiment yourself: notebook §1

Outline

- Principles of generative modelling
- Autoencoders: anomaly detection for card fraud
- Variational autoencoders
- Generative adversarial networks
- Diffusion models
- Generative models in practice
- Wrap-up

The VAE makes the autoencoder a probabilistic generator

- **In one sentence:** Instead of one code per input, the encoder outputs a distribution; the decoder learns to turn random codes into realistic data.

$$\log p_\theta(\mathbf{x}) \;\geq\; \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}\mid\mathbf{z})\big]}_{\text{reconstruction}} - \underbrace{\mathrm{KL}\big(q_\phi(\mathbf{z}\mid\mathbf{x})\,\|\,p(\mathbf{z})\big)}_{\text{regulariser}}$$

- **where**
  - Right-hand side = evidence lower bound (ELBO)
  - $q_\phi(\mathbf{z}\mid\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \mathrm{diag}\,\boldsymbol{\sigma}^2_\phi(\mathbf{x}))$ = encoder
  - $p_\theta(\mathbf{x}\mid\mathbf{z})$ = decoder
  - $p(\mathbf{z}) = \mathcal{N}(0, I)$ = prior
  - The second term is the Kullback–Leibler divergence (KL).

- **Why a bound**
  - $\log p(\mathbf{x}) = \mathrm{ELBO} + \mathrm{KL}\big(q(\mathbf{z}\mid\mathbf{x})\|p(\mathbf{z}\mid\mathbf{x})\big)$
  - The gap is the error of the encoder; derivation in Appendix A
- **Closed-form KL**
  - $\mathrm{KL} = \frac{1}{2}\sum_{j=1}^{k}\big(\mu_j^2 + \sigma_j^2 - \log\sigma_j^2 - 1\big)$

- **In practice:** the KL term is what makes random codes meaningful: it packs all inputs around the origin of the latent space.

(Kingma & Welling, 2014; Bishop & Bishop, 2024, sec. 19.2)

Experiment yourself: notebook §2

The reparameterisation trick

In one sentence: Sampling is not differentiable, but a shifted and scaled standard normal is.

$$\mathbf{z} = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x}) \odot \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, I)$$

$$\nabla_\phi \, \mathbb{E}_{q_\phi}[h(\mathbf{z})] = \mathbb{E}_{\boldsymbol{\varepsilon}}\left[\nabla_\phi \, h(\boldsymbol{\mu}_\phi + \boldsymbol{\sigma}_\phi \odot \varepsilon)\right]$$

where

- $\odot$ = element-wise product
- The randomness sits in ε, which does not depend on ϕ
- One draw of ε per image and step is enough in practice.

Our VAE on Fashion-MNIST

- 28×28 grey images of 10 kinds of clothing, a drop-in for the MNIST digits
- 60 000 training, 10 000 test images; Bernoulli decoder (binary cross-entropy)

Size

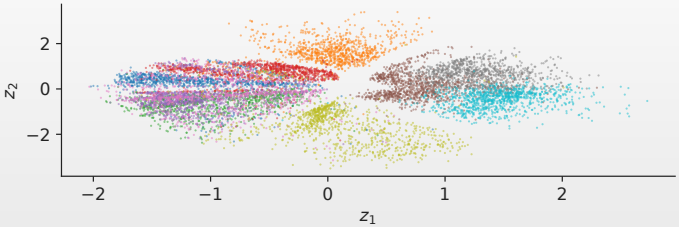
- $784 \to 400 \to 200 \to k$; 798 816 parameters for $k = 16$; 20 epochs

In practice: the same trick drives every model that samples inside the network: VAEs, dropout as inference, stochastic policies.

(Kingma & Welling, 2014; Xiao et al., 2017)

Experiment yourself: notebook §2

A two-dimensional latent space separates the product types



Source: Fashion-MNIST, Zalando Research (Xiao et al., 2017). Data: own estimates; 60,000 training and 10,000 test images.

- **No labels used**: the VAE never saw the classes, yet shoes, bags and garments occupy separate regions.
- **Measured**: a 15-nearest-neighbour classifier on the two latent means reaches 71.7 % accuracy (5-fold, test images).

NN_L8_vae_latent

Quality of VAE reconstructions and generated samples



Source: Fashion-MNIST, Zalando Research (Xiao et al., 2017). Data: own estimates; 60,000 training and 10,000 test images.

- **Rows**: real test images
  - Their reconstructions through a 16-dimensional code
  - New images decoded from random codes $\mathbf{z} \sim \mathcal{N}(0, I)$.
- **Blur** comes from the Gaussian posterior and the pixel-wise loss: the decoder averages over plausible details.

NN_L8_vae_samples

Trade-off between reconstruction and the KL penalty during training



Source: Fashion-MNIST, Zalando Research (Xiao et al., 2017). Data: own estimates; 60,000 training and 10,000 test images.

- **Test, $k = 16$, last epoch**: reconstruction 228.6 nats per image, KL 12.0 nats
  - ELBO -240.6 nats.
- **The KL rises** while the error falls: the encoder spends more information on each image as it learns.

(own computation, seed 1)

NN_L8_vae_elbo

β-VAE: reconstruction quality and latent-space structure

$$\mathcal{L}_\beta = -\mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})] + \beta \,\mathrm{KL}\big(q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z})\big)$$

β	Reconstruction	KL	Active dims
0.5	222.5	17.8	9
1.0	228.6	12.0	6
4.0	242.5	6.4	4

- **Larger** β
  - Fewer active latent dimensions (KL per dimension > 0.05): from 9 to 4 of 16
  - Worse reconstruction, smoother and more disentangled codes
- **Smaller** β
  - Closer to a plain autoencoder: good copies, poor samples

- **In practice:** for scenario generation prefer a tidy latent space ($\beta \geq 1$)
  - For compression prefer reconstruction ($\beta < 1$).

(Higgins et al., 2017; own computation, test images, 20 epochs)

Experiment yourself: notebook §2

Outline

- Principles of generative modelling
- Autoencoders: anomaly detection for card fraud
- Variational autoencoders
- Generative adversarial networks
- Diffusion models
- Generative models in practice
- Wrap-up

Generative adversarial networks: generator and discriminator

- **In one sentence:** The generator turns noise into fake data; the discriminator tries to tell fake from real; each improves against the other.

$$\min_G \max_D \; \mathbb{E}_{\mathbf{x}\sim p_{data}}\big[\log D(\mathbf{x})\big] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\big[\log\big(1 - D(G(\mathbf{z}))\big)\big]$$

$$D^*(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_G(\mathbf{x})} \;\Rightarrow\; \text{value} = -\log 4 + 2\,\mathrm{JS}(p_{data}\|p_G)$$

- **where**
  - G = generator network, D = discriminator (probability that $\mathbf{x}$ is real)
  - p_G = distribution of $G(\mathbf{z})$
  - Jensen–Shannon divergence (JS), zero only when $p_G = p_{data}$.

- **At equilibrium**
  - $D^* = 1/2$ everywhere: the detective can only guess; discriminator loss $2\log 2$
- **No likelihood**
  - A GAN never writes down p_G; it only learns to sample from it

- **In practice:** GANs are judged by what their samples get right (tails, dependence), not by a likelihood number.

(Goodfellow et al., 2014; Bishop & Bishop, 2024, ch. 17)

Experiment yourself: notebook §3

Training a GAN: alternate steps and the non-saturating loss

In one sentence: Each step updates the detective on a batch of real and fake data, then the forger on fresh noise.

$$\mathcal{L}_D = -\frac{1}{m}\sum_i \left[\log D(\mathbf{x}_i) + \log(1 - D(G(\mathbf{z}_i)))\right]$$

$$\mathcal{L}_G = -\frac{1}{m}\sum_i \log D(G(\mathbf{z}_i)) \quad \text{(non-saturating)}$$

Our GAN

- S&P 500 daily log returns 4 Jan 2005 to 2019, 3 774 days, standardised
- Generator 4 801 and discriminator 4 353 parameters; 6 000 steps, 3 seeds

where

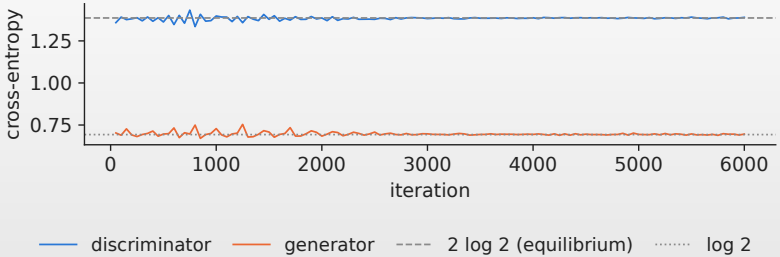
- $m = 256$ days per batch
- $\mathbf{z} \in \mathbb{R}^8$ standard normal
- $\mathcal{L}_G$ replaces $\log(1 - D(G))$, whose gradient vanishes when D wins early
- Adam with learning rate $2 \cdot 10^{-4}$, $\beta_1 = 0.5$.

In practice: watch both losses: a discriminator loss stuck at zero means the generator has stopped learning.

(Goodfellow et al., 2014)

Experiment yourself: notebook §3

The two losses settle near the equilibrium values

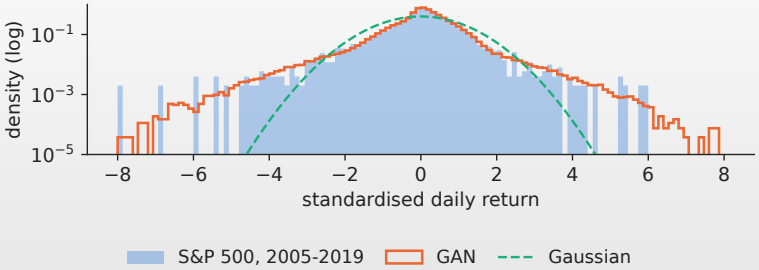


Source: Yahoo Finance, daily quotes. Data: own estimates; S&P 500 4 Jan 2005 to 31 Dec 2019, trading days.

- **Equilibrium**: the discriminator loss hovers around $2\log 2$ and the generator loss around $\log 2$, the values of $D^* = 1/2$.
- **No convergence proof**: gradient play can cycle
  - The loss curves do not measure sample quality.

NN_L8_gan_losses

The GAN reproduces the fat tails missed by the Gaussian model



Source: Yahoo Finance, daily quotes. Data: own estimates; S&P 500 4 Jan 2005 to 31 Dec 2019, trading days.

- **Excess kurtosis**: real 11.86, GAN 6.65 (sd 0.48 over 3 seeds), Gaussian 0.
- **Kolmogorov–Smirnov distance** to the real returns: GAN 0.030, fitted Gaussian 0.115.

(Cont, 2001)

NN_L8_gan_density

A risk check: 1 % Value-at-Risk on the years after training

$$\mathrm{VaR}_{0.01} = -q_{0.01}(r), \qquad \hat{\pi} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{1}\{r_t < q_{0.01}\}$$

Method (fitted 2005–2019)	1 % quantile	Exceedances
Historical simulation	−3.48 %	1.11 %
GAN samples (3 seeds)	−3.61 %	0.95 %
Gaussian	−2.67 %	2.39 %

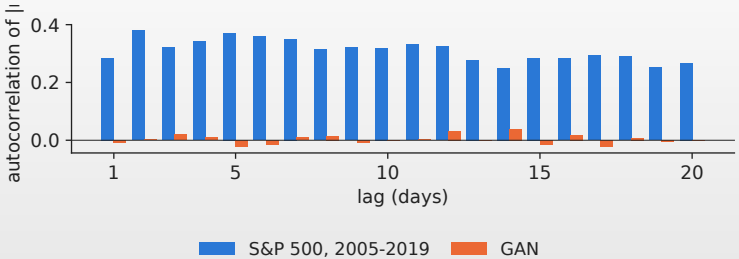
- **Test period**
  - 2 Jan 2020 to 30 Oct 2026, 1 717 days; a correct 1 % VaR is exceeded on about 17 days
- **Reading**
  - The GAN quantile is as good as historical simulation; the Gaussian is exceeded 2.39 % of the days
  - Unconditional quantiles only: no model here reacts to volatility
  - The expected shortfall (ES), the mean loss beyond the VaR, depends on the same tail

- **In practice:** a generator is only as good as the risk number it reproduces: backtest it like any VaR model (Kupiec, 1995).

(own computation)

Experiment yourself: notebook §3

A GAN on single days cannot learn volatility clustering



Source: Yahoo Finance, daily quotes. Data: own estimates; S&P 500 4 Jan 2005 to 31 Dec 2019, trading days.

- **Real returns**: the autocorrelation of $|r_t|$ is 0.28 at lag 1 and still 0.27 at lag 20: calm and turbulent periods cluster.
- **GAN samples** are independent draws (-0.01 at lag 1): the right marginal, the wrong dynamics
  - Sequence models (TimeGAN, Quant GANs) generate whole paths.

(Cont, 2001; Yoon et al., 2019; Wiese et al., 2020)

NN_L8_gan_acf

Mode collapse and the Wasserstein GAN

- **In one sentence:** A generator can win by producing a few convincing samples; the Wasserstein distance gives useful gradients even when the supports do not overlap.

$$W_1(p_{data}, p_G) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{p_{data}}[f(\mathbf{x})] - \mathbb{E}_{p_G}[f(\mathbf{x})]$$

- **where**
  - f = critic network, constrained to be 1-Lipschitz (weight clipping or a gradient penalty)
  - The generator minimises W_1: the Wasserstein GAN (WGAN).

- **Mode collapse**
  - The generator maps all $\mathbf{z}$ to one region: e.g. only calm days, never crashes
- **Checks**
  - Compare tail quantiles, the share of extreme days and the spread of samples with the real data

- **In practice:** a stress-test generator that has collapsed onto calm days is worse than no generator.

(Arjovsky, Chintala & Bottou, 2017)

Outline

- Principles of generative modelling
- Autoencoders: anomaly detection for card fraud
- Variational autoencoders
- Generative adversarial networks
- Diffusion models
- Generative models in practice
- Wrap-up

Diffusion models: forward noising and reverse denoising

- **In one sentence:** Adding a little Gaussian noise many times turns any data into pure noise; a network learns each small reverse step.

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

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\Big(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0,\ (1-\bar{\alpha}_t)I\Big), \quad \bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$$

- **where**
  - $\mathbf{x}_0$ = a real observation (here a standardised yield curve)
  - β_t = noise schedule, linear from 10^{-4} to 0.02 over $T = 200$ steps
  - $\bar{\alpha}_T \approx 0$, so $\mathbf{x}_T$ is almost standard normal.

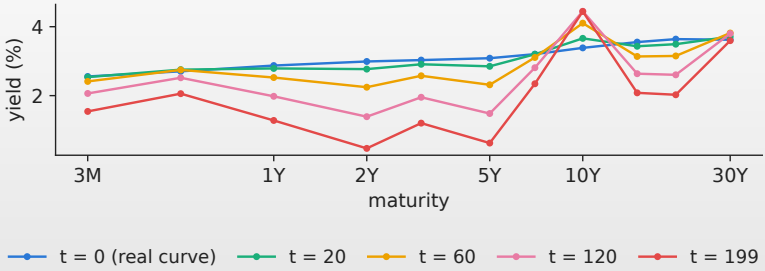
- **Closed form**
  - Any step t is reached in one draw: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon$
- **No parameters**
  - The forward process is fixed; only the reverse process is learnt

- **In practice:** diffusion turns generation into many small regression problems, which is why it trains stably.

(Sohl-Dickstein et al., 2015; Ho, Jain & Abbeel, 2020)

Experiment yourself: notebook §4

The forward process on a real yield curve



Source: European Central Bank, euro area AAA yield curve. Data: curve of 30 Oct 2026; noise schedule of the lecture (mechanism).

- **Curve of 30 Oct 2026**, 11 maturities from 3 months to 30 years
  - By $t = 199$ the shape is gone and only noise remains.
- **The network** must learn to walk back from right to left, one small step at a time.

(mechanism shown on real data; one draw of the noise)

NN_L8_diffusion_forward

Learning to denoise: predict the noise that was added

- **In one sentence:** The network sees a noisy curve and the step t, and guesses the noise; subtracting the guess, step by step, gives a new curve.

$$\mathcal{L}(\theta) = \mathbb{E}_{t,\mathbf{x}_0,\varepsilon}\left\| \varepsilon - \varepsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon,\ t\right)\right\|^2$$

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(\mathbf{x}_t, t)\right) + \sqrt{\beta_t}\,\mathbf{u}, \quad \mathbf{u} \sim \mathcal{N}(0, I)$$

- **where**
  - ε_θ = denoiser network
  - $\alpha_t = 1 - \beta_t$
  - t uniform on $\{1, \ldots, T\}$
  - The loss is a simplified, reweighted ELBO.

- **Our denoiser**
  - MLP with 3 hidden layers of 256 and a sinusoidal embedding of t: 145 675 parameters
  - 12 000 steps of 512 curves; 5,000 curves sampled per seed

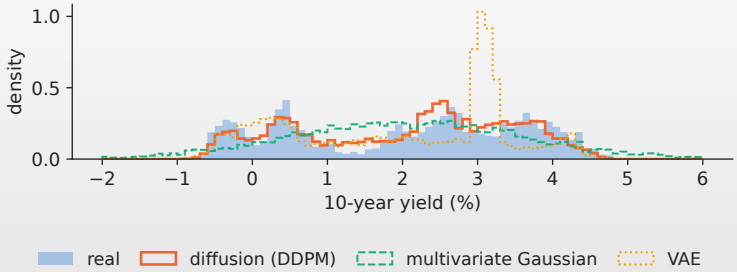
- **Data**
  - 5 664 daily curves of the European Central Bank, 6 Sep 2004 to 30 Oct 2026

- **In practice:** the same loss, with images and a text condition, runs the image generators you know
  - Here it generates interest-rate scenarios.

(Ho, Jain & Abbeel, 2020; Bishop & Bishop, 2024, ch. 20)

Experiment yourself: notebook §4

The two regimes of euro rates in diffusion and Gaussian samples

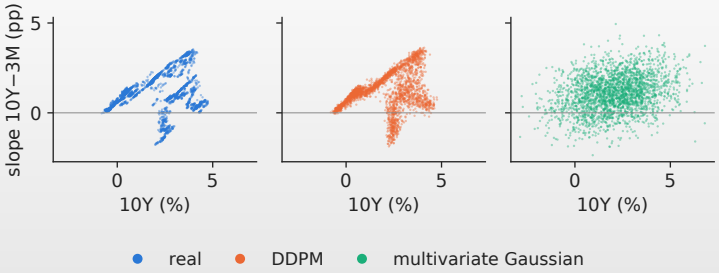


Source: European Central Bank, euro area AAA yield curve. Data: own estimates; 6 Sep 2004 to 30 Oct 2026, business days.

- **Real 10-year yield**: from −0.82 % to 4.78 %, negative on 13.8 % of the days: a low-rate and a normal-rate regime.
- **Wasserstein distance** of the generated 10-year yield to the real one: DDPM 0.149, VAE 0.191, Gaussian 0.325 percentage points.

NN_L8_yc_density

Diffusion preserves the nonlinear co-movement of level and slope

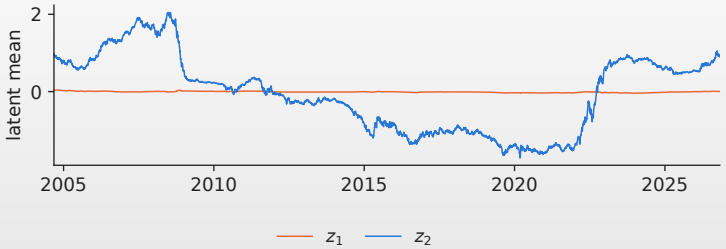


Source: European Central Bank, euro area AAA yield curve. Data: own estimates; 6 Sep 2004 to 30 Oct 2026, business days.

- **Inverted curves** (10 years below 3 months): 8.7 % of real days, 10.2 % of diffusion samples, 14.7 % of Gaussian samples.
- **Slope distance** (Wasserstein): DDPM 0.099, VAE 0.242, Gaussian 0.113.

NN_L8_yc_scatter

A two-factor VAE of the curve rediscovers level and slope



Source: European Central Bank, euro area AAA yield curve. Data: own estimates; 6 Sep 2004 to 30 Oct 2026, business days.

- **Correlations** of the latent means with the level (mean of the 11 yields) and the slope (10 years minus 3 months): up to 0.99 and 0.66 in absolute value.
- **Interpretable compression**: the latent path tracks the 2014–2021 low-rate era and the 2022 tightening without being told.

(Svensson, 1994)

NN_L8_yc_latent_time

Outline

- Principles of generative modelling
- Autoencoders: anomaly detection for card fraud
- Variational autoencoders
- Generative adversarial networks
- Diffusion models
- Generative models in practice
- Wrap-up

Evaluation criteria for synthetic financial data

- **Fidelity**
  - Marginals and tails (kurtosis, 1 % quantile), dependence across assets and over time (autocorrelation of $|r_t|$)
- **Diversity**
  - No mode collapse: the share of extreme days and regimes as in the data
- **Utility**
  - Train on synthetic, test on real (TSTR): a model fitted on synthetic data must work on real data
- **Privacy**
  - Generators can memorise records: check the distance of each sample to its nearest real record
- **Our scorecard**
  - GAN returns: tails right, clustering wrong
  - DDPM curves: regimes and inversions right, Gaussian wrong

In practice: never use synthetic data for a decision before it passes the same backtests as the model it feeds.

(Cont, 2001; Yoon et al., 2019)

Choosing a generative model

Need	AE	VAE	GAN	DDPM
Anomaly score	yes (error)	yes (ELBO)	indirect	yes (bound)
Sample quality	–	blurry	sharp	best
Sampling speed	–	one pass	one pass	200 passes
Training stability	high	high	low	high
Latent factors	yes	yes, smooth	no inverse	no

- **Rule of thumb**: autoencoders to monitor, VAEs to compress and explore, diffusion to simulate faithfully, GAN models when sampling must be fast.

(Bishop & Bishop, 2024, ch. 16–20)

Outline

- Principles of generative modelling
- Autoencoders: anomaly detection for card fraud
- Variational autoencoders
- Generative adversarial networks
- Diffusion models
- Generative models in practice
- Wrap-up

Key takeaways

- **Autoencoders** model "normal": on card fraud PR-AUC 14.2 % without labels, against 80.4 % for a classifier that uses labels.
- **VAEs** maximise the ELBO = reconstruction − KL
  - The reparameterisation trick makes them trainable
  - β trades detail for a tidy latent space.
- **GANs** reproduce fat tails (kurtosis 6.65 vs 11.86 real) and a usable 1 % VaR, but not volatility clustering when trained on single days.
- **Diffusion** learns to undo noise
  - On euro yield curves it keeps both rate regimes and the inversions that a Gaussian misses.

Research from our group: generative models for tail risk

- **Large language models (LLM) as risk forecasters: LLM-VaR and LLM-ES** (Pele, Bolovăneanu, Lin, Ren, Ginavar, Spilak, Andrei, Toma, Lessmann & Härdle, 2026): uses generative language models, prompted with return histories, to forecast Value-at-Risk and expected shortfall, evaluated with standard risk backtests.
- **Foundation models for tail risk** (Jheng, Tak, Găman, Mazurencu-Marinescu-Pele & Pele, 2026): uses pretrained probabilistic forecasting models, which generate sample paths, without retraining and with conformal calibration, to forecast VaR and ES in Central and Eastern European markets.
- **Link with this lecture**: both treat a generator of scenarios as a risk model, and judge it by the backtests used on this lecture's GAN.

Findings as stated in the papers; the numbers on the other slides come from our own computations.

(Bucharest University of Economic Studies; Romanian Academy, Institute for Economic Forecasting)

Further reading and resources

- **Bishop & Bishop (2024)**, chapters 16–20: latent variables, GANs, flows, autoencoders, diffusion.
- **Lilian Weng's notes** on diffusion models: lilianweng.github.io.
- **Ho et al. (2020)**, the DDPM paper, short and readable.
- **This course**: code, data, notebooks — github.com/danpele/NEURAL_BIZ.
- **Experiment yourself**: the notebook reproduces every chart and number of this lecture.

Quiz 1: autoencoder

- Why is the fraud autoencoder trained on legitimate transactions only?

Answer

- It must learn what "normal" looks like
  - Frauds are then rebuilt badly and get a high score (median error 12.7 times larger).

Quiz 2: ELBO

- Which two terms make up the ELBO, and what does each do?

Answer

- Expected log-likelihood of the data given the code (reconstruction) minus the KL divergence of the encoder from the prior (keeps codes near $\mathcal{N}(0, I)$, so random codes decode to realistic data).

Quiz 3: GAN equilibrium

- What does the optimal discriminator output once the generator matches the data?

Answer

- $D^*(\mathbf{x}) = 1/2$ everywhere, because $p_{data} = p_G$
  - Its loss is then $2 \log 2$.

Quiz 4: synthetic returns

- Our GAN gets the tails of S&P 500 returns right.
  - Why is it still unfit for a volatility model?

Answer

- It draws days independently: the autocorrelation of $|r_t|$ is -0.01 instead of 0.28
  - Volatility clustering needs a sequence generator.

Quiz 5: diffusion

- What does the diffusion network predict during training?

Answer

- The Gaussian noise ε that was added to $\mathbf{x}_0$ to obtain $\mathbf{x}_t$, given $\mathbf{x}_t$ and t
  - Generation subtracts the predicted noise step by step.

References (1/2)

- Arjovsky, M., Chintala, S. & Bottou, L. (2017). Wasserstein generative adversarial networks. *Proceedings of ICML*, PMLR 70, 214–223. arXiv:1701.07875
- Bishop, C. M. & Bishop, H. (2024). *Deep Learning: Foundations and Concepts*, ch. 16–20. Springer. doi:10.1007/978-3-031-45468-4
- Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, 1(2), 223–236. doi:10.1080/713665670
- Dal Pozzolo, A., Caelen, O., Johnson, R. A. & Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. *IEEE Symposium Series on Computational Intelligence*, 159–166. doi:10.1109/SSCI.2015.33
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27. arXiv:1406.2661
- Higgins, I., Matthey, L., Pal, A. et al. (2017). beta-VAE: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*. openreview.net
- Ho, J., Jain, A. & Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33. arXiv:2006.11239
- Jheng, S.-L., Tak, R., Găman, Ș., Mazurencu-Marinescu-Pele, M. & Pele, D. T. (2026). Can foundation models manage risk? Zero-shot VaR and ES forecasting with conformal calibration in CEE markets. *Economic Computation and Economic Cybernetics Studies and Research*, 60(2), 267–290. doi:10.24818/18423264/60.2.26.14
- Kingma, D. P. & Welling, M. (2014). Auto-encoding variational Bayes. *International Conference on Learning Representations*. arXiv:1312.6114

References (2/2)

- Kupiec, P. H. (1995). Techniques for verifying the accuracy of risk measurement models. *Journal of Derivatives*, 3(2), 73–84. doi:10.3905/jod.1995.407942
- Pele, D. T., Bolovăneanu, V., Lin, M.-B., Ren, R., Ginavar, A. T., Spilak, B., Andrei, A.-V., Toma, F.-M., Lessmann, S. & Härdle, W. K. (2026). In the beginning was the word: LLM-VaR and LLM-ES. SSRN Working Paper 5104383. doi:10.2139/ssrn.5104383
- Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N. & Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *Proceedings of ICML*, PMLR 37, 2256–2265. arXiv:1503.03585
- Svensson, L. E. O. (1994). Estimating and interpreting forward interest rates: Sweden 1992–1994. NBER Working Paper 4871. doi:10.3386/w4871
- Wiese, M., Knobloch, R., Korn, R. & Kretschmer, P. (2020). Quant GANs: Deep generation of financial time series. *Quantitative Finance*, 20(9), 1419–1440. doi:10.1080/14697688.2020.1730426
- Xiao, H., Rasul, K. & Vollgraf, R. (2017). Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747
- Yoon, J., Jarrett, D. & van der Schaar, M. (2019). Time-series generative adversarial networks. *Advances in Neural Information Processing Systems*, 32. papers.nips.cc/paper/8789

Appendix A: derivation of the ELBO

$$\log p(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log \frac{p(\mathbf{x},\mathbf{z})}{q(\mathbf{z}\mid\mathbf{x})}\right] + \mathbb{E}_q\left[\log \frac{q(\mathbf{z}\mid\mathbf{x})}{p(\mathbf{z}\mid\mathbf{x})}\right]$$

$$= \underbrace{\mathbb{E}_q[\log p(\mathbf{x}\mid\mathbf{z})] - \mathrm{KL}(q\|p(\mathbf{z}))}_{\text{ELBO}} + \underbrace{\mathrm{KL}(q\|p(\mathbf{z}\mid\mathbf{x}))}_{\geq 0}$$

- Multiply and divide by $q(\mathbf{z}\mid\mathbf{x})$ inside $\log p(\mathbf{x}) = \log p(\mathbf{x},\mathbf{z}) - \log p(\mathbf{z}\mid\mathbf{x})$ and take the expectation under q.
- The second KL is non-negative, so the ELBO is a lower bound
  - It is tight when q equals the true posterior.
- For Gaussians $\mathcal{N}(\boldsymbol{\mu}, \operatorname{diag}\boldsymbol{\sigma}^2)$ and $\mathcal{N}(0, I)$ the KL is $\frac{1}{2}\sum_j(\mu_j^2 + \sigma_j^2 - \log\sigma_j^2 - 1)$.

(Kingma & Welling, 2014)

Appendix B: the optimal discriminator

$$V(D) = \int \left[p_{data}(\mathbf{x}) \log D(\mathbf{x}) + p_G(\mathbf{x}) \log(1 - D(\mathbf{x})) \right] d\mathbf{x}$$

$$\frac{\partial}{\partial D} : \frac{p_{data}}{D} - \frac{p_G}{1 - D} = 0 \;\Rightarrow\; D^* = \frac{p_{data}}{p_{data} + p_G}$$

- Maximise the integrand pointwise: $a \log y + b \log(1 - y)$ peaks at $y = a/(a + b)$.
- Plugging D^* back gives $V = -\log 4 + \mathrm{KL}(p_{data} \| m) + \mathrm{KL}(p_G \| m)$, $m = (p_{data} + p_G)/2$, i.e. $-\log 4 + 2\,\mathrm{JS}$.

(Goodfellow et al., 2014)

Appendix C: settings of the four experiments

Model	Settings
AE	29–16–8–16–29, ReLU, Adam 10^{-3}, batch 256, at most 60 epochs, patience 6 on validation error; seeds 1–3
VAE	784–400–200–k, $k \in \{2, 16\}$, Adam 10^{-3}, batch 128, 20 epochs, Bernoulli decoder
GAN	noise 8; two hidden layers of 64, LeakyReLU 0.2; Adam $2 \cdot 10^{-4}$, $\beta_1 = 0.5$; 6,000 steps of 256 days
DDPM	$T = 200$, linear β_t; MLP 3×256 with time embedding; Adam 10^{-3}, cosine decay; 12,000 steps of 512
Hardware	Apple-silicon laptop; total run 0.3 minutes