Advanced Topic A07: *Diffusion Models*

Data Science with Python – BSc Advanced Lectures

Joerg Osterrieder

10 Minutes

Motivation: Beyond GANs and VAEs

Diffusion models surpassed GANs in image quality by 2022

- GANs: adversarial training, mode collapse, training instability
- VAEs: fast but blurry; evidence lower bound limits sharpness
- Diffusion models: no adversarial objective, stable training, log-likelihood tractable
- Ho et al. (2020) DDPM produced photorealistic images; Stable Diffusion scaled to billions
- In finance: diffusion models generate realistic synthetic return scenarios and time-series

Diffusion models: stable training + high fidelity + tractable likelihoods

Core Idea: Destroy and Reconstruct

Gradually add noise, then learn to reverse the process

- **Forward process**: add a small amount of Gaussian noise to data over T steps until the data is indistinguishable from pure noise
- **Reverse process**: train a neural network to predict and remove the noise at each step
- At inference: start from pure Gaussian noise and apply the learned denoising T times
- The model never sees data directly – it only learns to denoise
- Key insight: denoising one small step at a time is much easier than one-shot generation

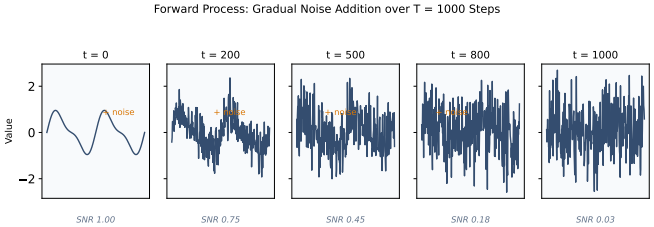
Diffusion = forward corruption + learned reverse; the model solves many easy problems

The Forward Process

A Markov chain that adds noise step by step

$$q(x_t \mid x_{t-1}) = \mathcal{N}\big(x_t;\ \sqrt{1-\beta_t}\, x_{t-1},\ \beta_t I\big)$$

- $\beta_t \in (0, 1)$: noise schedule; controls how quickly the signal is destroyed
- After T steps ($T \approx 1000$): $x_T \approx \mathcal{N}(0, I)$ (approximately pure noise)
- **Closed-form marginal:** $q(x_t \mid x_0) = \mathcal{N}(x_t;\ \sqrt{\bar{\alpha}_t}\, x_0,\ (1-\bar{\alpha}_t) I)$ where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$
- This allows sampling x_t directly from x_0 without iterating t steps



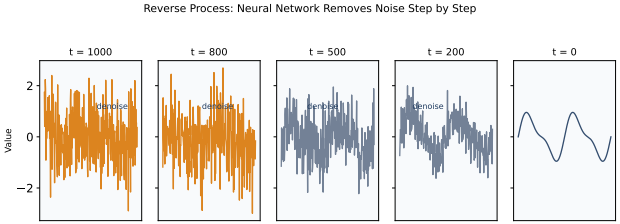
The reparameterisation trick: $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon$, $\varepsilon \sim \mathcal{N}(0, I)$

The Reverse Process

A learned Markov chain that denoises step by step

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\big(x_{t-1};\, \mu_\theta(x_t, t),\, \Sigma_\theta(x_t, t)\big)$$

- μ_θ and Σ_θ: predicted mean and variance from the neural network
- Ho et al. (2020) simplification: fix $\Sigma_\theta = \beta_t I$ and train only μ_θ
- Equivalent reformulation: predict the noise ε added at step t ("ε-prediction")
- Loss: $\mathcal{L}_{\text{simple}} = \mathbb{E}_{t, x_0, \varepsilon}\left[\|\varepsilon - \varepsilon_\theta(x_t, t)\|^2\right]$
- This is just a denoising autoencoder loss, evaluated at random noise levels



DDPM training objective: predict the noise injected at a random timestep

The U-Net Denoiser Architecture

The neural network backbone that predicts noise at each timestep

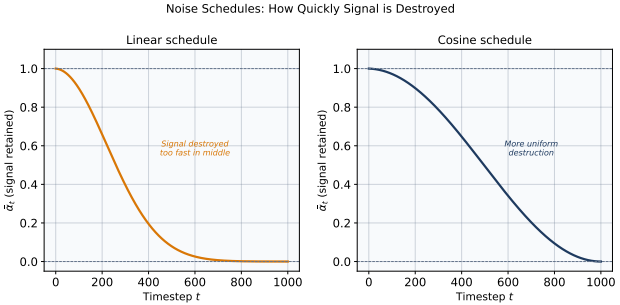
- $\varepsilon_\theta(x_t, t)$ must accept a noisy image and a timestep, and output a noise prediction of the same shape as the input
- **U-Net**: encoder path compresses spatial resolution (global context), decoder path restores it; skip connections preserve fine-grained detail
- **Timestep conditioning**: t is embedded via sinusoidal positional encoding and added to residual blocks throughout the U-Net
- **Attention layers** at low resolutions capture long-range dependencies (critical for coherent structure in images and temporal sequences)
- For time-series: 1D U-Net; for images: 2D U-Net; architecture is otherwise identical

U-Net + residual blocks + self-attention + timestep embedding: the standard DDPM backbone (Ho et al. 2020)

Noise Schedule

How fast noise accumulates determines training and generation quality

- **Linear schedule** (Ho et al. 2020): β_t increases linearly from 10^{-4} to 0.02
- **Cosine schedule** (Nichol & Dhariwal 2021): $\bar{\alpha}_t = \cos^2\left(\frac{t/T+s}{1+s} \cdot \frac{\pi}{2}\right)$; preserves more signal at early steps, destroys signal more uniformly
- **Key property:** the schedule must reach $\bar{\alpha}_T \approx 0$ so x_T is near-Gaussian
- Cosine schedule improved image fidelity measurably on CIFAR-10



Noise schedule choice is a hyperparameter with significant impact on generation quality

DDIM: Faster Sampling

Deterministic sampling in far fewer steps than DDPM

- DDPM requires $T = 1000$ reverse steps for high quality: expensive at inference
- Song et al. (2020) DDIM: reformulate reverse process as a non-Markovian ODE
- Deterministic reverse: same model, but follow an ODE trajectory instead of stochastic steps
- Result: acceptable quality with 50–100 steps (10–20x speedup)
- Interpolation in latent space: DDIM trajectories are smooth and invertible

DDIM is the default sampler for most production diffusion systems

Consistency Models: One-Step Generation

Distilling diffusion trajectories into a direct mapping

- DDIM reduces steps to 50–100, but each step still requires one forward pass through the U-Net
- **Consistency models** (Song et al. 2023): train $f_\theta(x_t, t)$ to map any point on a diffusion trajectory directly to x_0 in one step
- Key property (self-consistency): $f_\theta(x_t, t) = f_\theta(x_{t'}, t')$ for any two points on the same trajectory
- **Consistency distillation**: supervise f_θ to match a pre-trained DDIM teacher; achieves near-DDIM quality with 1–4 neural function evaluations
- **Latent Consistency Model (LCM)**: applies consistency distillation to Stable Diffusion's latent space; SDXL-Turbo reaches single-step image generation

Consistency models: the path from 1000-step DDPM to real-time 1-step generation (Song et al. 2023)

Classifier-Free Guidance

Conditional generation without a separate classifier

- Goal: generate samples conditioned on a class label or text prompt
- **Classifier guidance** (Dhariwal 2021): use a classifier's gradient to steer denoising; requires training a separate noise-robust classifier
- **Classifier-free guidance** (Ho & Salimans 2022): train a single model with both conditional (c) and unconditional (null label) objectives
- At inference: $\tilde{\varepsilon}_\theta = \varepsilon_\theta(\cdot \mid \varnothing) + w\,[\varepsilon_\theta(\cdot \mid c) - \varepsilon_\theta(\cdot \mid \varnothing)]$
- Guidance scale w: higher w increases fidelity to the condition at the cost of diversity

Classifier-free guidance powers text-to-image models: DALL-E 2, Stable Diffusion, Imagen

Latent Diffusion Models

Diffusion in a compressed latent space for scalability

- Running diffusion in pixel space is expensive: 512×512 images have 786k dimensions
- **Latent diffusion** (Rombach et al. 2022, Stable Diffusion): train a VAE encoder-decoder first, then run diffusion in the $64 \times 64 \times 4$ latent
- VAE compresses by $8\times$ spatially: DDPM becomes 64x cheaper
- The cross-attention layer conditions the U-Net on text embeddings (CLIP or T5)
- Enables 1B+ parameter models on consumer hardware

Stable Diffusion = VAE + U-Net DDPM + CLIP conditioning, all trained sequentially

Evaluation: FID, Precision, and Recall

Measuring the quality and diversity of generated samples

- **FID (Frechet Inception Distance):** distance between feature distributions of real and generated images; lower is better; standard benchmark
- **Precision:** fraction of generated samples close to the real distribution (fidelity)
- **Recall:** fraction of real distribution covered by generated samples (diversity)
- Diffusion models outperform GANs on FID while also having higher recall (less mode collapse)
- Finance evaluation: KS test on marginal distributions, autocorrelation matching, tail-risk statistics (CVaR) for synthetic return scenarios

FID is the de-facto metric but has known biases; always report precision and recall alongside

Limitations of Diffusion Models

Powerful but not without drawbacks

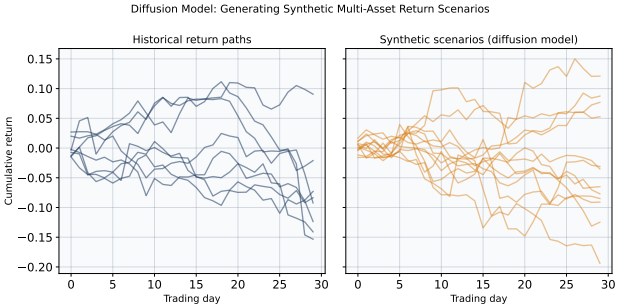
- **Slow inference:** 50–1000 NFEs (neural function evaluations) vs 1 for GAN
- **Memory:** U-Net backbone with attention is large; latent diffusion mitigates but not eliminates
- **Memorisation:** large models can memorise training examples (privacy risk in financial data)
- **Evaluation:** FID measures image-space features; no universal metric for time-series quality
- **Conditionality:** conditioning on structured financial data (covariance matrices) remains research

Inference speed remains the main deployment bottleneck; consistency models reduce it to 1–4 steps

Finance Application: Synthetic Return Scenarios

Generating realistic multi-asset return paths for stress testing

- Train a diffusion model on historical multi-day return windows (rolling 30-day blocks)
- Condition on a market state embedding (VIX level, yield curve slope)
- Generate 10,000 synthetic scenarios: each is a realistic return trajectory
- Use synthetic scenarios for: VaR / CVaR estimation, portfolio optimisation under stress, regulatory stress tests that require distributional diversity



Synthetic returns from diffusion models preserve fat tails and cross-asset correlations

Code Pointer

Diffusion models in Python

- **Hugging Face Diffusers**: `pip install diffusers transformers`; DDPM, DDIM, latent diffusion, and Stable Diffusion with two lines of code
- **Training from scratch**: implement forward process + ε-prediction loss; U-Net backbone from `diffusers.models.UNet2DModel`
- **For time-series**: `TSDiff` (IBM) or `conditional-ts-diffusion`; adapt the 1D U-Net to temporal sequences
- **Evaluation**: `torch-fidelity` for FID; `scipy.stats.ks_2samp` for distributional tests on returns

Diffusers library: 200+ models, DDPM to SDXL, uniform pipeline API

Score-Based Perspective

Diffusion models as score matching in continuous time

- Song et al. (2021): unify diffusion with score-based models via stochastic differential equations
- The score function $\nabla_x \log p_t(x)$ is the optimal denoiser direction
- Training a denoiser ε_θ is equivalent to training a score network
- Continuous-time SDE formulation enables flexible discretisation schedules (ODE or SDE)
- Probability flow ODE allows exact likelihood computation via the instantaneous change-of-variables

Score-SDE perspective: diffusion = score matching + SDE/ODE solvers (Song et al. 2021)

Flow Matching: A Modern Alternative

Straight-line optimal transport paths replace curved diffusion trajectories

- **Flow matching** (Lipman et al. 2022): train a vector field $v_\theta(x, t)$ that transports $\mathcal{N}(0, I)$ to the data distribution along straight-line interpolants
- Straight paths matter: straight ODE trajectories integrate in 4–8 neural function evaluations vs 50–1000 for diffusion
- **Conditional flow matching**: define per-sample straight paths from noise to data; train on per-sample vector field targets; simple regression objective
- Stable Diffusion 3, Meta Voicebox, and Llama 3 image models all use flow matching; SD 3 outperforms SDXL on FID while using fewer steps
- Finance: straight paths reduce inference cost for real-time scenario generation; 8-step ODE integration is fast enough for live portfolio risk updates

Flow matching is the successor to diffusion: straight paths, fewer steps, competitive quality (Lipman et al. 2022)

Summary and Open Questions

Three things to remember

- Diffusion models: gradually add noise (forward), then train a U-Net to reverse it step by step; loss is simply mean squared error on the predicted noise at a random timestep
- Classifier-free guidance controls the fidelity-diversity trade-off; latent diffusion (Stable Diffusion) makes the method practical at scale
- Finance application: generate synthetic multi-asset return scenarios conditioned on market regimes for stress testing and data augmentation

Open questions:

1. Can consistency models reduce inference to 1 step without quality loss?
2. How do we condition diffusion on covariance structure for portfolio simulation?
3. What metrics best capture distributional quality for financial time-series?

Further reading: Ho et al. (2020) DDPM; Song et al. (2021) Score SDE; Rombach et al. (2022) LDM