Lecture 19 Graph generative models

Learning a distribution of graphs

Autoregressive generation, graph VAEs, and discrete diffusion

One question throughout: how do we sample a new graph?

What does a graph generator learn?

$$G_1, \dots, G_M \sim p_{\text{data}} \quad \Rightarrow \quad p_\theta(G)$$

Training: observe a collection of graphs.
Generation: sample a new graph from the learned distribution.

Examples: molecular structures, network topologies,
and synthetic graphs for testing graph-learning methods.

A plausible sample must preserve dependencies among nodes and edges

A graph has many matrix representations

$$X' = PX, \qquad A' = P\,AP^T$$

X: node features. A: adjacency. P: permutation matrix.
P renames the nodes without changing the underlying graph.

$$p_\theta(PX, PAP^T) = p_\theta(X, A)$$

An equivariant encoder relabels node embeddings consistently.

Permutation invariance is a distribution property, not a particular node order

Why matching edge frequencies is insufficient

TOY DATA: THREE LABELED NODES

Half the graphs are empty; half are triangles.
Each individual edge appears with probability 0.5.

AN INDEPENDENT-EDGE MODEL

Sampling each edge independently gets the triangle frequency wrong.

$$P(A_{12} = 1, A_{13} = 1, A_{23} = 1) = 0.5 \neq 0.5^3 = 0.125$$

The joint distribution determines which edges occur together

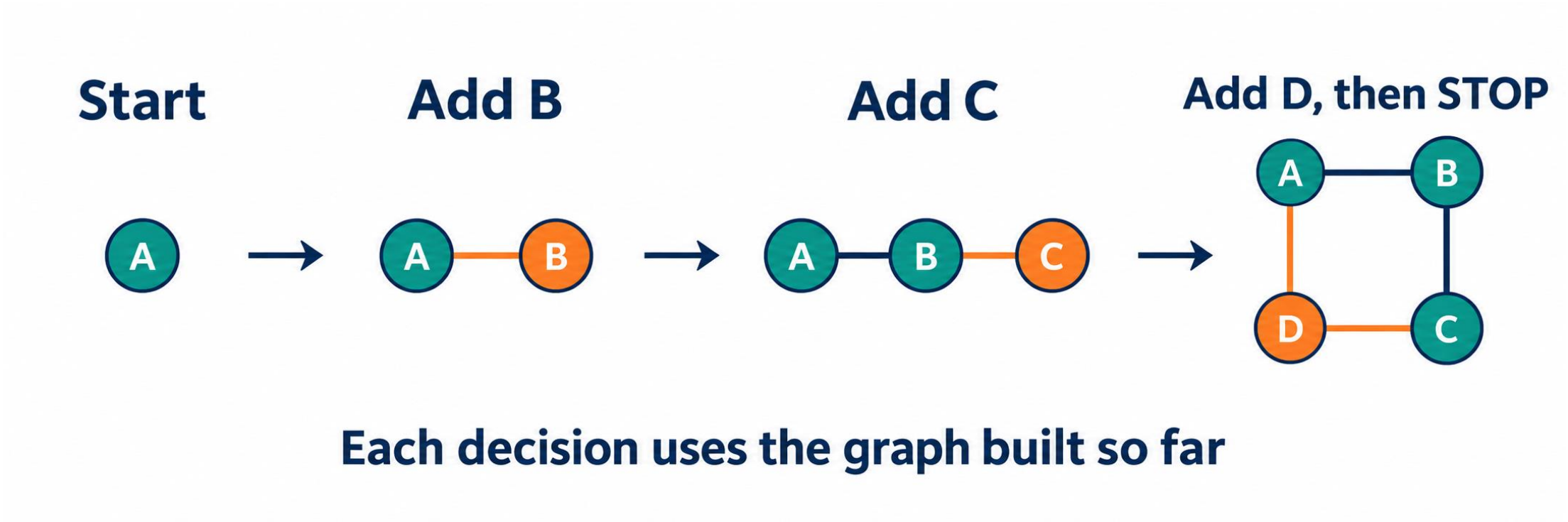
Autoregressive models

Generation by sequential decisions

Chain-rule likelihood and GraphRNN

Each decision conditions on the graph generated so far

Autoregressive generation: one decision at a time

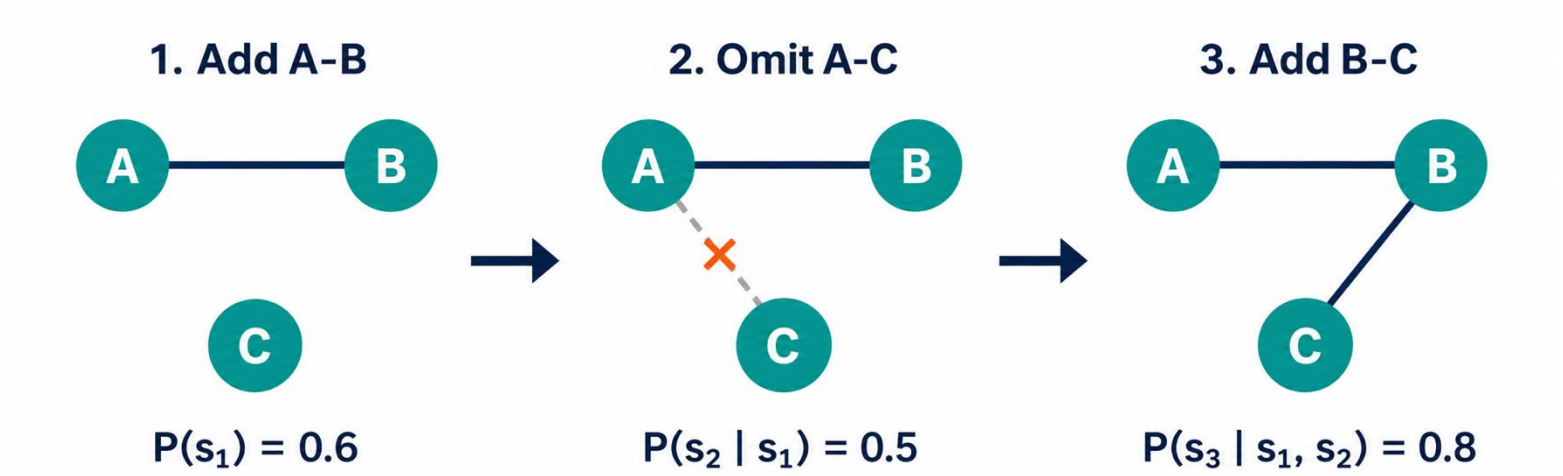


A partial-graph encoder summarizes the history for the next decision.

The autoregressive component is the generative decoder or decision rule

You et al. “GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models.” ICML 2018.

The chain rule gives a trainable likelihood



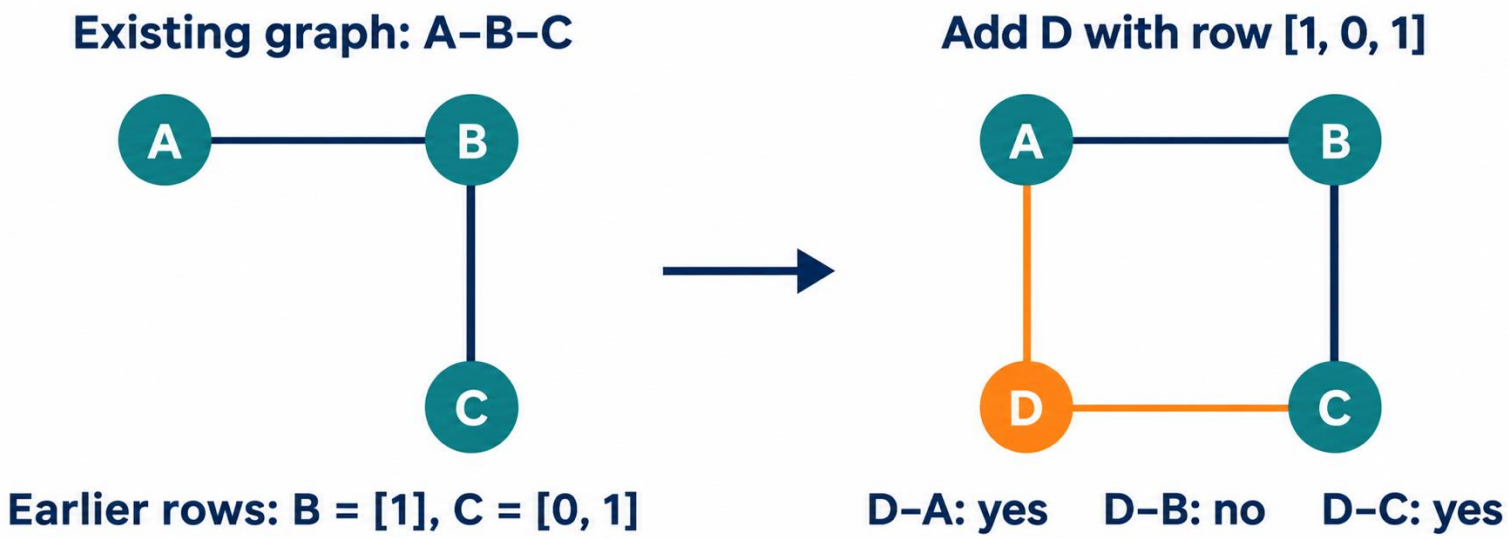
This ordered path has probability 0.6 × 0.5 × 0.8 = 0.24.

$$p_\theta(s_1, \dots, s_K) = \prod_{k=1}^{K} p_\theta(s_k | s_{<k}) \qquad L = -\sum_{k=1}^{K} \log p_\theta(s_k | s_{<k})$$

Each factor uses the decisions already made; the factors need not be independent

You et al. "GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models." ICML 2018.

GraphRNN example: adding a node to close a cycle



GRAPH RNN: MEMORY ACROSS NODES

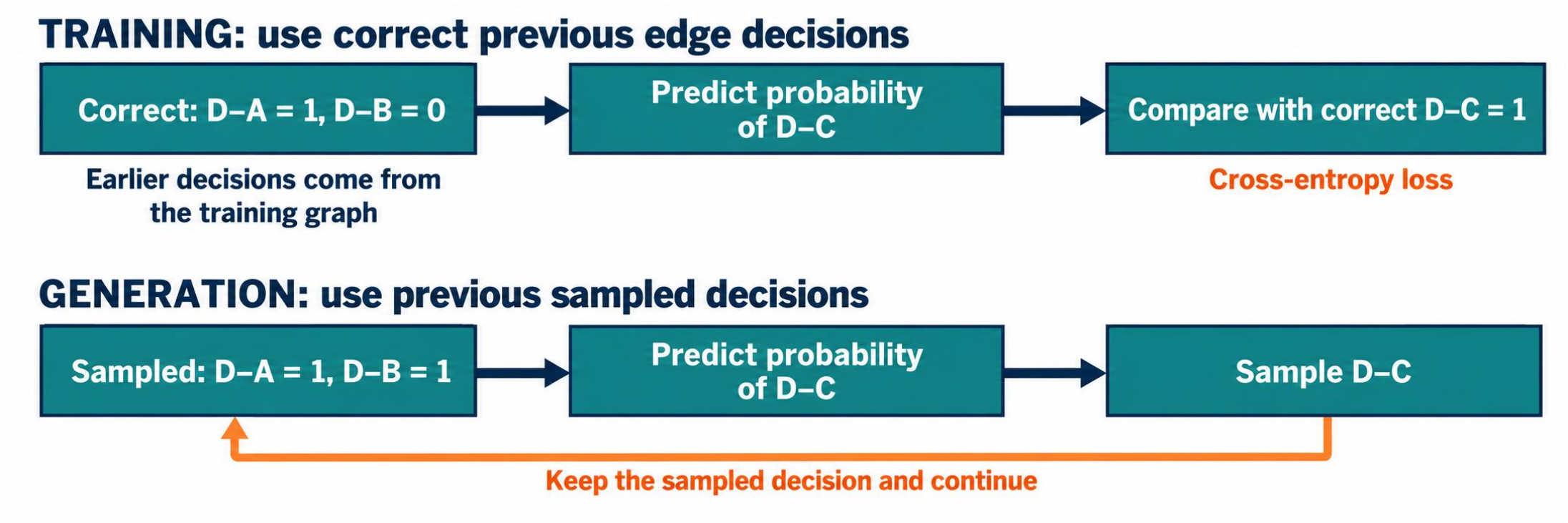
Reads B's [1] and C's [0, 1].
Its hidden state initializes the edge RNN for D.

EDGE RNN: DECISIONS WITHIN D'S ROW

Samples D–A, then D–B, then D–C.
Each decision conditions on earlier bits.

The full GraphRNN models dependencies within each generated adjacency row

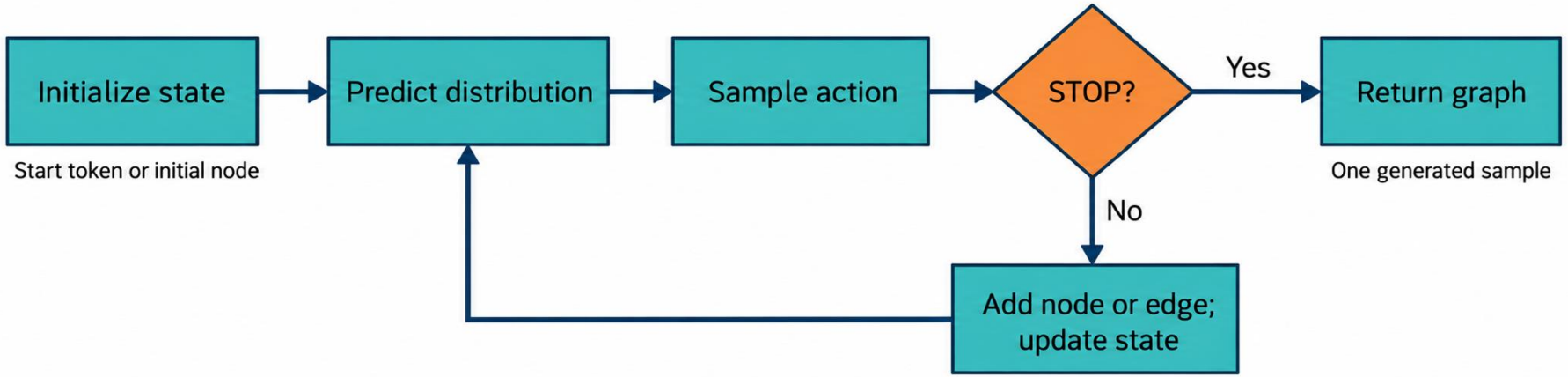
Training and generation use different previous decisions



The correct row for D is [1, 0, 1], for connections to A, B, C.
Before predicting D–C, training supplies D–B = 0 even if the model predicted 1.

Training uses decisions from the observed graph; generation uses its own sampled decisions

Autoregressive generation: sample, update, repeat



Concrete continuation: after A–B–C, sample D's connections [1, 0, 1].
The next outer decision determines whether generation continues.

Generic STOP-action design. Other models specify a node count or an end-of-sequence rule.

A maximum-size cap bounds computation; it is separate from a learned stopping rule

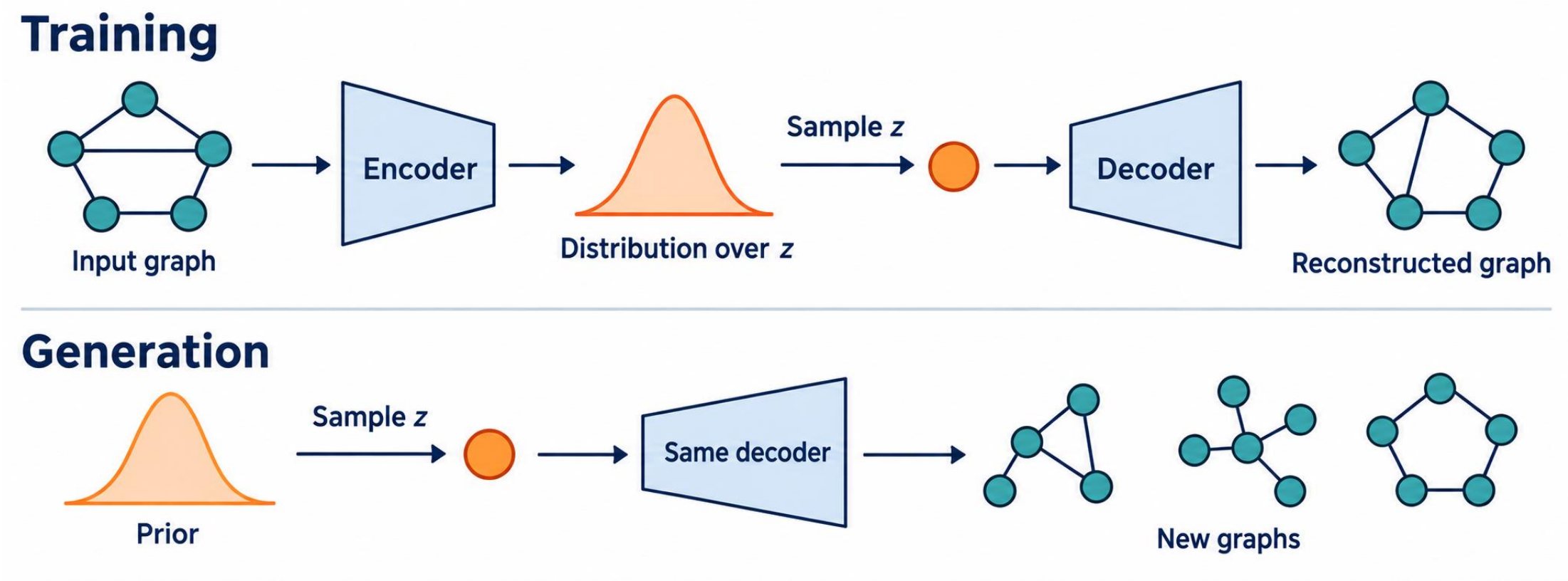
Variational autoencoders

Generation through a latent space

Posterior inference, the ELBO, and graph decoding

A latent sample provides shared context for graph generation

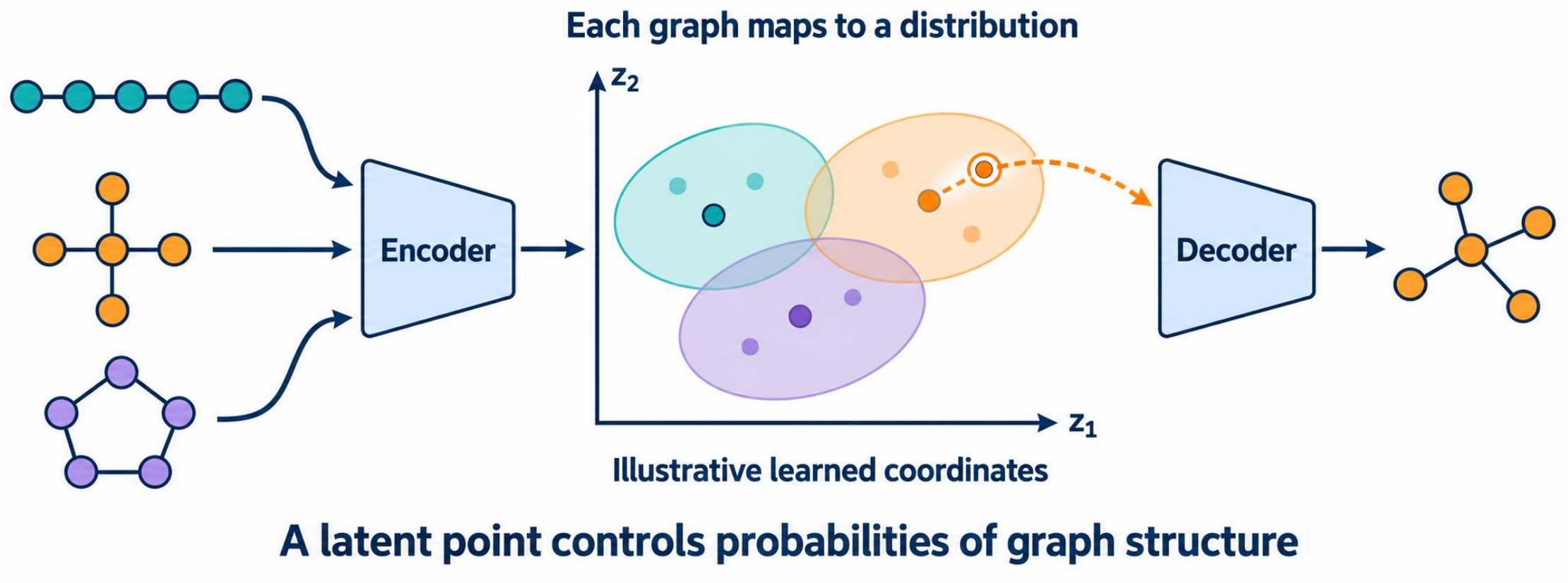
Graph VAE: training and generation



The encoder supports training and inference. Fresh generation starts from the prior.

Kingma & Welling. “Auto-Encoding Variational Bayes.” ICLR 2014.
Simonovsky & Komodakis. “GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders.” ICANN 2018.

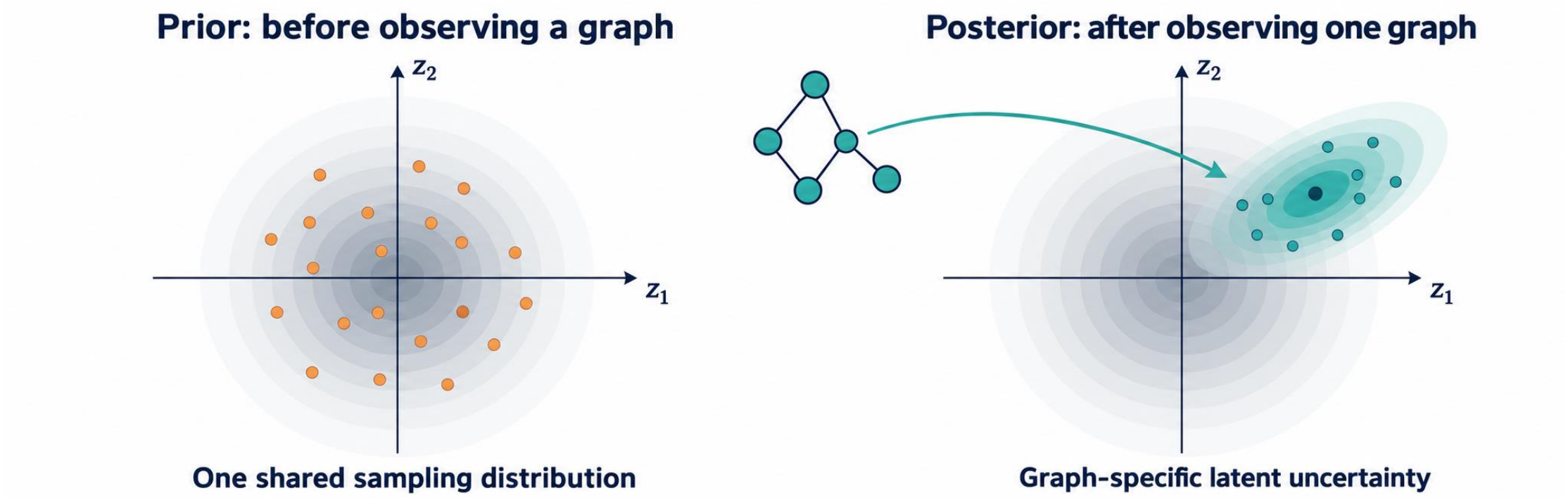
Latent space: a map of possible graph explanations



Each shaded region represents uncertainty for one graph, not a fixed class boundary

Kingma & Welling. “Auto-Encoding Variational Bayes.” ICLR 2014.
Simonovsky & Komodakis. “GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders.” ICANN 2018.

Prior and posterior answer different questions



Training balances reconstruction with the KL cost of departing from the prior

Prior: where to sample before seeing a graph. Posterior: which latents explain this graph.

Kingma & Welling. “Auto-Encoding Variational Bayes.” ICLR 2014.
Simonovsky & Komodakis. “GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders.” ICANN 2018.

Reparameterization makes sampling differentiable

$$\varepsilon \sim N(0, I), \qquad z = \mu_\varphi(G) + \sigma_\varphi(G) \odot \varepsilon$$

ε supplies randomness independent of the encoder parameters.
μ shifts the sample. σ scales each latent coordinate.

$$\mu = 1, \qquad \sigma = 0.5, \qquad \varepsilon = -0.4 \quad \Rightarrow \quad z = 0.8$$

Gradients pass through μ and σ while ε stays fixed in a sampled forward pass

Kingma & Welling. “Auto-Encoding Variational Bayes.” ICLR 2014.

The VAE objective: reconstruction and regularization

$$L = -E_{q_\varphi(z|G)}(\log p_\theta(G|z)) + \mathrm{KL}\big(q_\varphi(z|G) \parallel p(z)\big)$$

RECONSTRUCTION TERM

Can the sampled latent explain this graph?

KL TERM

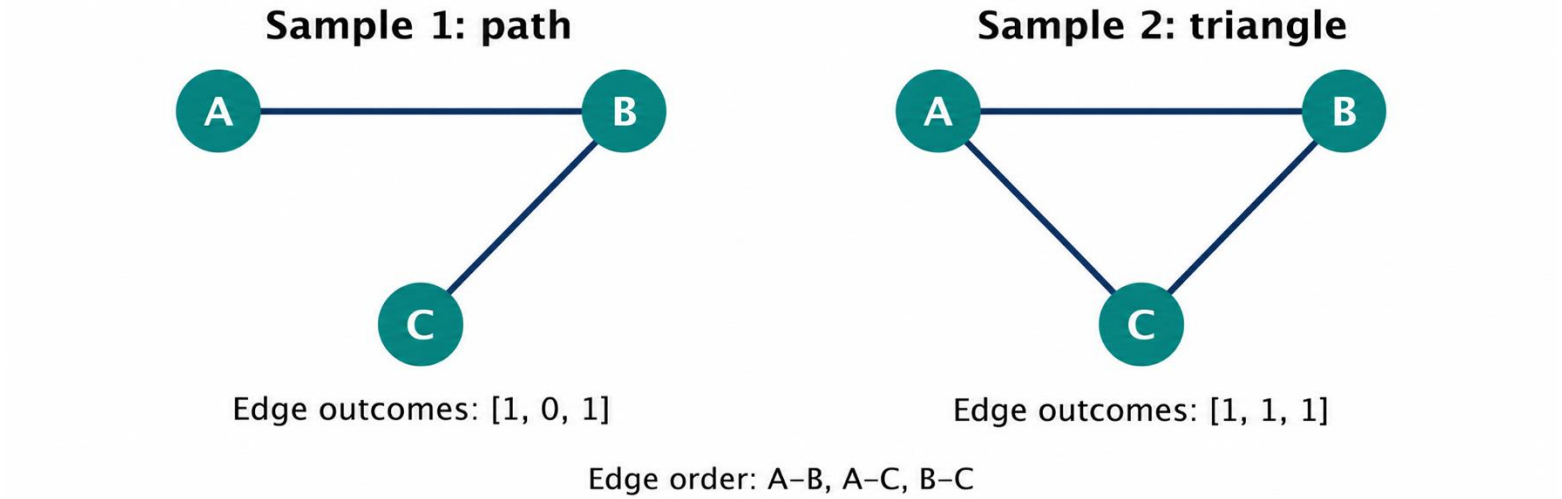
How far does its posterior depart from the prior?

Minimizing this loss maximizes the evidence lower bound, or ELBO

One latent vector, two possible graph samples

FIXED z: EDGE PROBABILITIES FOR A–B, A–C, B–C ARE (0.8, 0.2, 0.7)

For this decoder, sample each edge independently given the same latent vector z.



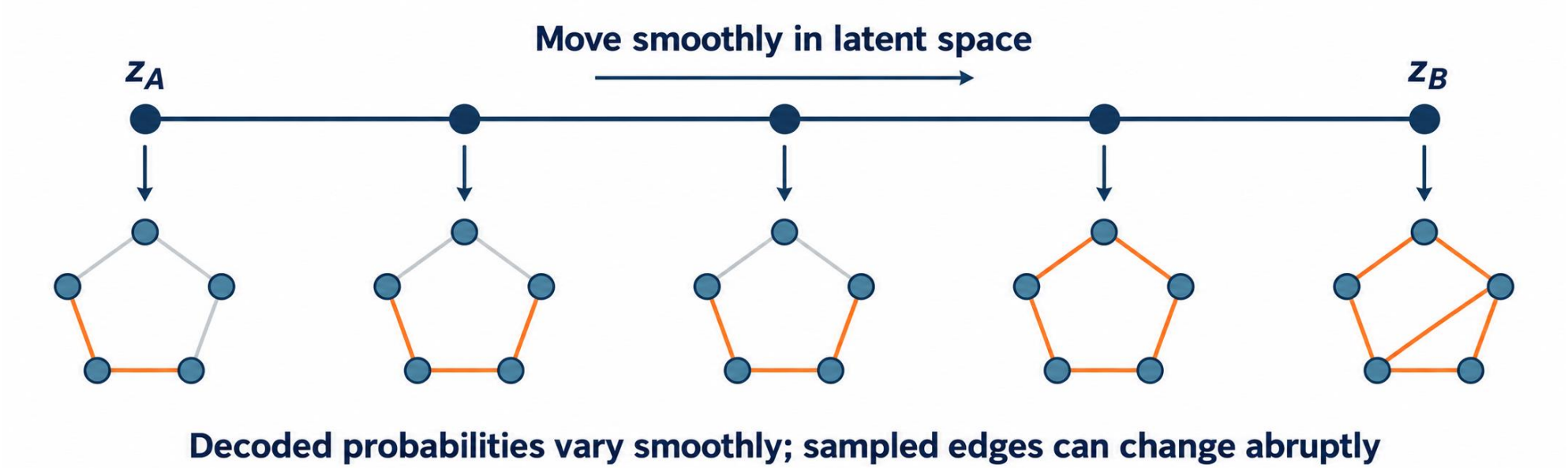
$$P(1,0,1|z) = 0.8\times0.8\times0.7 = 0.448 \qquad P(1,1,1|z) = 0.8\times0.2\times0.7 = 0.112$$

For A–C, an absent edge has probability 0.8; a present edge has probability 0.2.

The latent vector sets probabilities; random edge draws produce individual graphs

Kingma & Welling. “Auto-Encoding Variational Bayes.” ICLR 2014.
Simonovsky & Komodakis. “GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders.” ICANN 2018.

Interpolation changes probabilities continuously



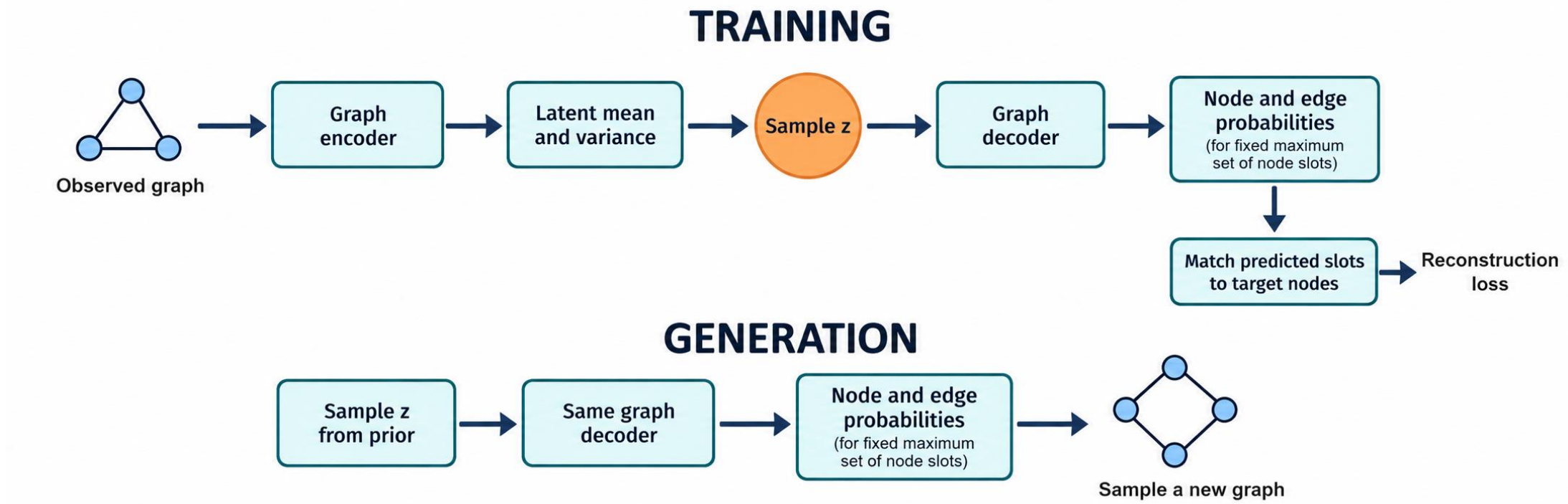
Orange: sampled edges · Gray: candidate connections shown for orientation

$$z(\lambda) = (1-\lambda)z_A + \lambda z_B, \qquad 0 \le \lambda \le 1$$

A continuous path in latent space can still produce abrupt changes in discrete topology

Kingma & Welling. "Auto-Encoding Variational Bayes." ICLR 2014.
Simonovsky & Komodakis. "GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders." ICANN 2018.

GraphVAE: whole-graph training and generation



Decoder output: node existence, node types, and edge existence/types in a fixed set of slots.
Training loss combines reconstruction after matching with KL regularization of the latent distribution.

One graph-level latent vector controls the decoder's predictions for the whole graph

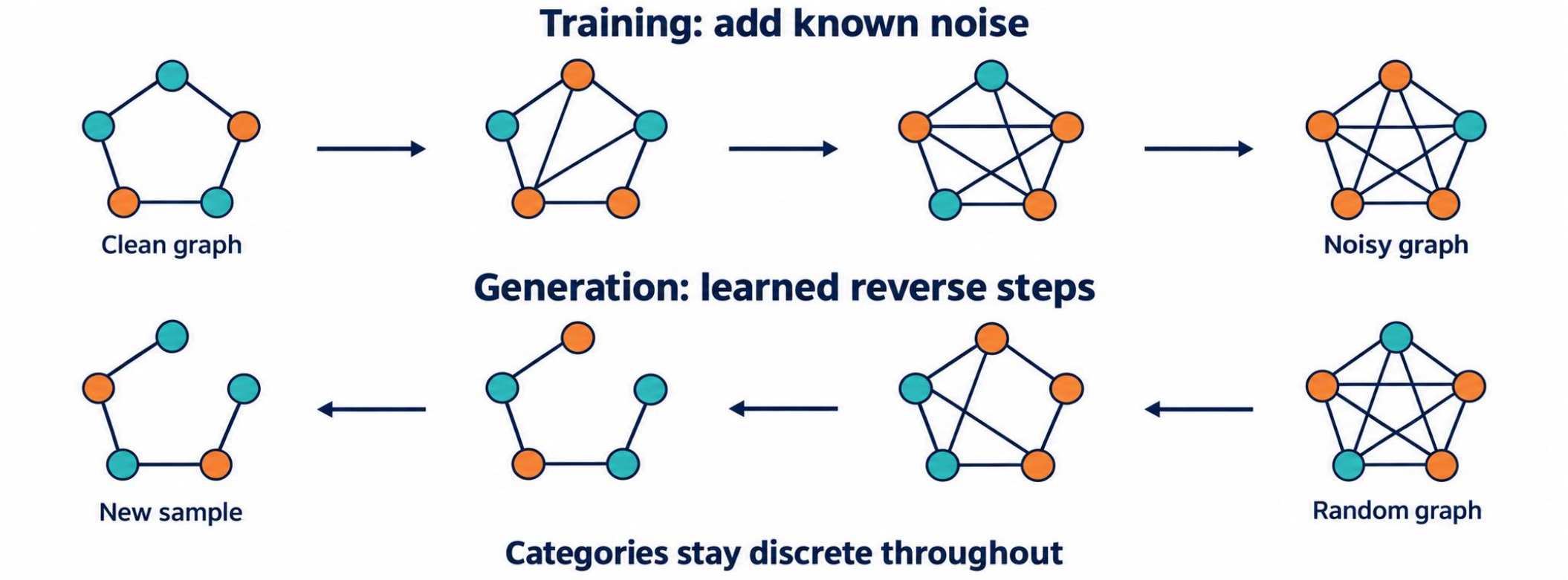
Discrete graph diffusion

Generation by repeated denoising

Corruption, reverse probabilities, and DiGress

The network predicts clean categories from a noisy graph

Diffusion: learn to reverse corruption



Training corrupts observed graphs. Sampling denoises a fresh random graph.

Diffusion time indexes noise level

Ho et al. “Denoising Diffusion Probabilistic Models.” NeurIPS 2020.
Vignac et al. “DiGress: Discrete Denoising diffusion for graph generation.” ICLR 2023.

Two Markov chains with opposite directions

FORWARD: known corruption

$$q(G^{1:T}|G^0) = \prod_{t=1}^{T} q(G^t|G^{t-1})$$

REVERSE: learned generation

$$p_\theta(G^{0:T}) = p(G^T) \prod_{t=1}^{T} p_\theta(G^{t-1}|G^t)$$

The denoiser is shared across time and receives the current noise level

Ho et al. “Denoising Diffusion Probabilistic Models.” NeurIPS 2020.
Song et al. “Score-Based Generative Modeling through Stochastic Differential Equations.” ICLR 2021.

Continuous and categorical diffusion

CONTINUOUS STATES

Gaussian noise produces real-valued features or relaxed adjacency entries.

CATEGORICAL STATES

Each node or pair remains a discrete type, including a no-edge category.

MODELING CONSEQUENCE

The state space determines the corruption law and the reverse predictor.

Categorical corruption can add, remove, or change an edge type

Ho et al. “Denoising Diffusion Probabilistic Models.” NeurIPS 2020.
Austin et al. “Structured Denoising Diffusion Models in Discrete State-Spaces.” NeurIPS 2021.

Categorical forward transitions

$$q(e^t = b | e^{t-1} = a) = [Q_t]_{ab}$$

$$Q_t = (1 - \beta_t) I + \beta_t 1 m^T$$

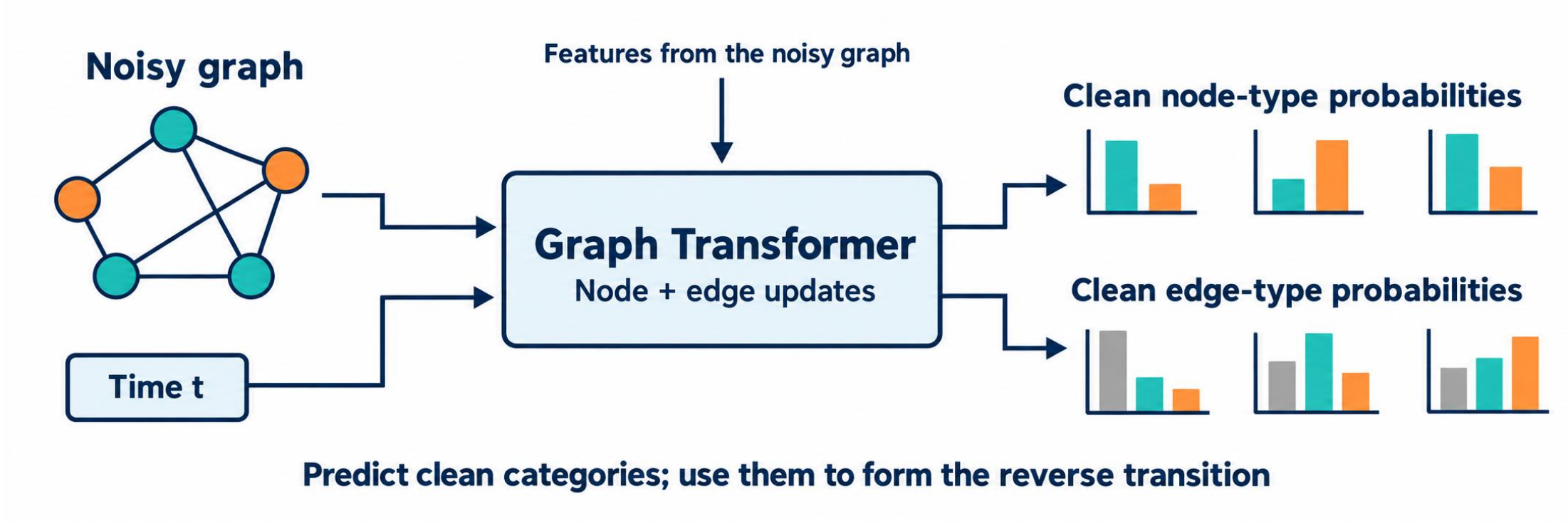
$$q(e^t | e^0) = \mathrm{Cat}(e^0 \bar{Q}_t), \qquad \bar{Q}_t = Q_1 \cdots Q_t$$

e^0 is a one-hot row vector. m is a probability vector over categories.
With enough corruption, the distribution approaches m.

Using empirical edge-type frequencies can preserve sparse category marginals

Austin et al. “Structured Denoising Diffusion Models in Discrete State-Spaces.” NeurIPS 2021.
Vignac et al. “DiGress: Discrete Denoising diffusion for graph generation.” ICLR 2023.

DiGress: a graph Transformer as denoiser

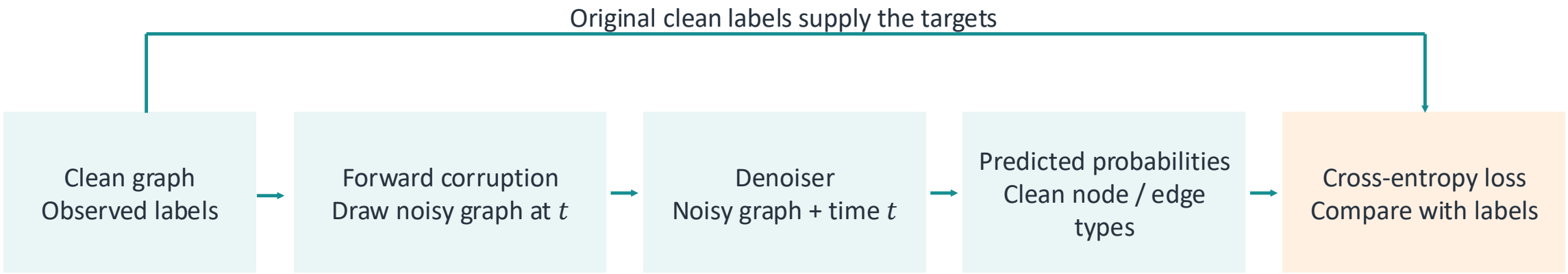


Node and edge outputs are categorical distributions over clean types.

The network must be permutation equivariant and preserve undirected edge symmetry

Vignac et al. "DiGress: Discrete Denoising diffusion for graph generation." ICLR 2023.

Training a discrete graph diffusion model



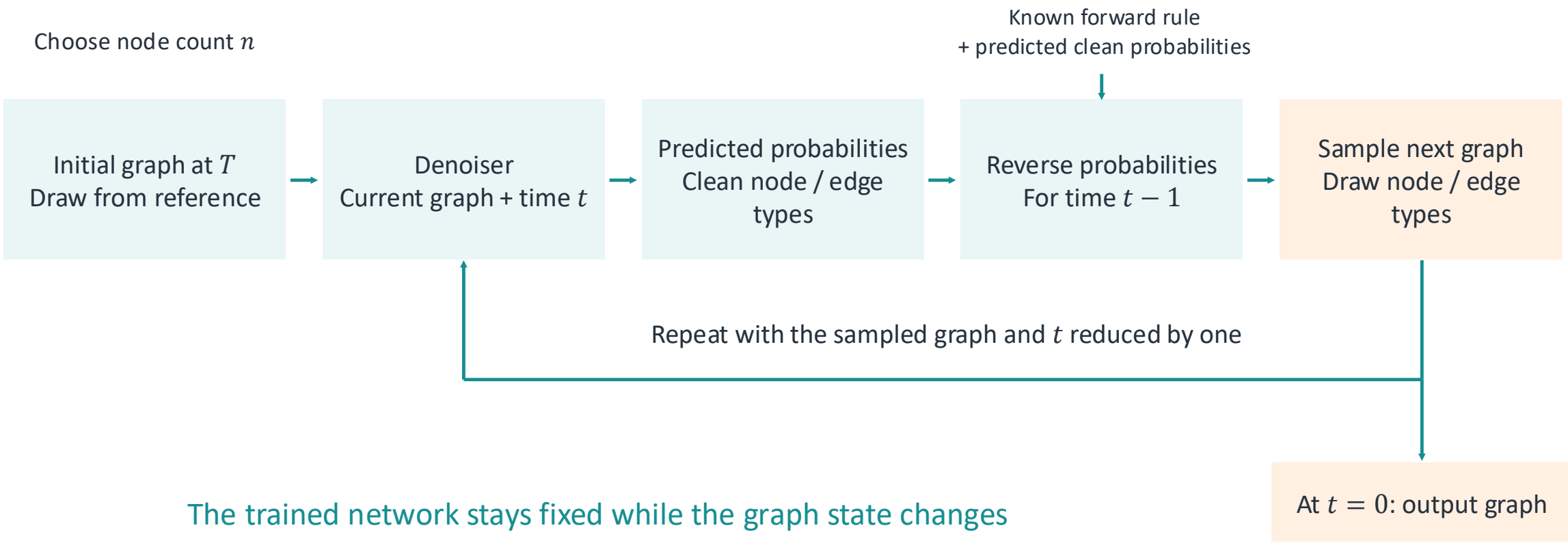
Sample t for each update

$$L = \sum_i \mathrm{CE}(x_i{}^0, \hat{p}_i) + \lambda \sum_{i<j} \mathrm{CE}\left(e_{ij}{}^0, \hat{p}_{ij}\right)$$

Loss gradients update the denoiser parameters

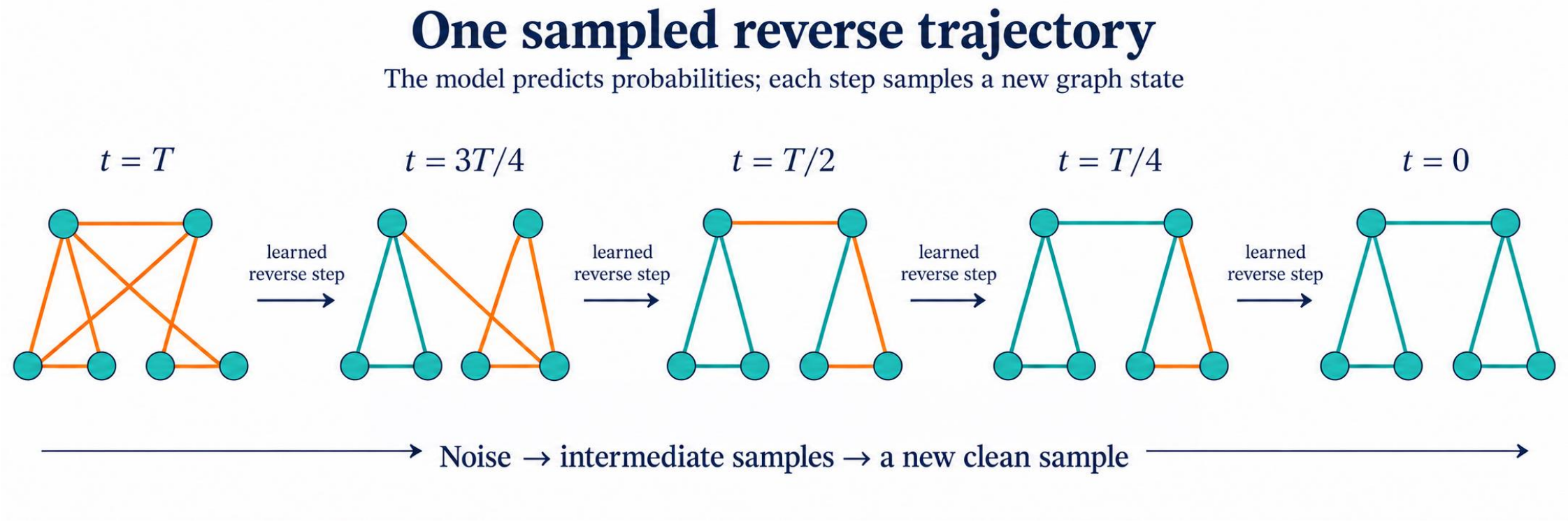
Vignac et al. “DiGress: Discrete Denoising diffusion for graph generation.” ICLR 2023.

Sampling a discrete graph diffusion model



The trained network stays fixed while the graph state changes

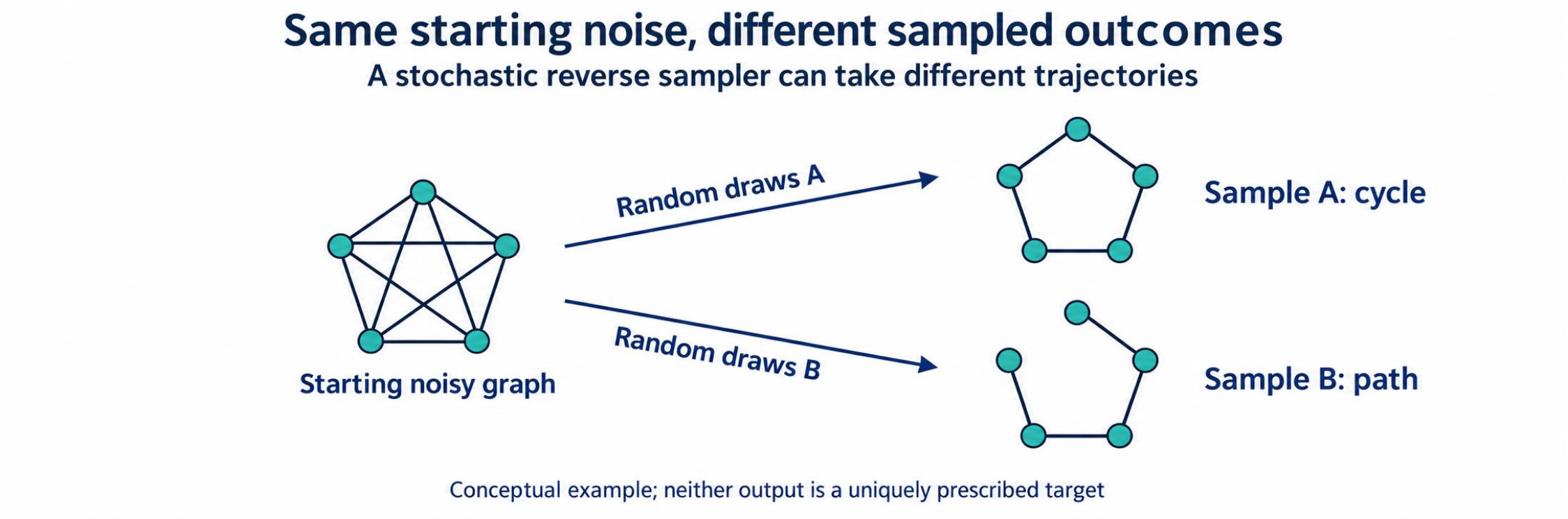
Watch a noisy graph become a structured sample



A reverse trajectory can add, delete, or change edges; progress is not necessarily monotone

Austin et al. "Structured Denoising Diffusion Models in Discrete State-Spaces." NeurIPS 2021.
Vignac et al. "DiGress: Discrete Denoising diffusion for graph generation." ICLR 2023.

Random reverse steps allow different clean outcomes



The same initial noisy state can lead to different samples under different random draws

Austin et al. “Structured Denoising Diffusion Models in Discrete State-Spaces.” NeurIPS 2021.
Vignac et al. “DiGress: Discrete Denoising diffusion for graph generation.” ICLR 2023.

Factorized steps can create dependent final edges

A common reverse kernel samples edge categories independently conditional on the full current graph.

$$p_\theta(E^{t-1}|G^t) = \prod_{i<j} p_\theta(e_{ij}^{t-1}|G^t)$$

The probabilities share graph context.
Marginalizing over random intermediate graphs couples the final edges.

Conditional independence at one step does not imply independent edges in the final model

Austin et al. “Structured Denoising Diffusion Models in Discrete State-Spaces.” NeurIPS 2021.
Vignac et al. “DiGress: Discrete Denoising diffusion for graph generation.” ICLR 2023.

Evaluation and synthesis

Assessing generated graph distributions

Validity, diversity, generalization, and computational cost

A plausible individual graph is only one part of the evaluation

Evaluation of graph generation

VALIDITY AND DISTRIBUTION

Check admissibility, diversity, novelty, and relevant graph statistics.
Compare generated samples with held-out graph distributions.

CONDITIONING AND COMPUTATION

Report condition satisfaction together with validity and diversity.
Measure runtime, memory, sampling steps, and any repair or rejection.

Attractive examples alone do not establish the quality of a distribution

Three ways to generate a graph

AUTOREGRESSIVE

Sample the next decision conditioned on earlier decisions.

VARIATIONAL AUTOENCODER

Sample a latent representation, then decode a graph.

DIFFUSION

Sample noisy graph states and repeatedly predict cleaner states.

Next: infer a graph from observed signals, then recover missing signals