Graph generative models
Part I

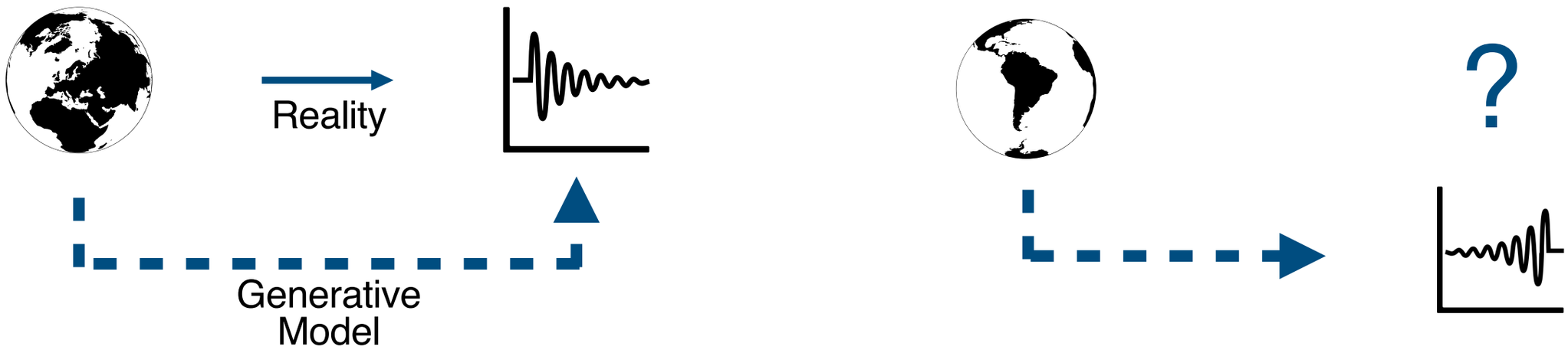
Dr Dorina Thanou

May 13, 2025

Generative models as simulators of real world

A generative model simulates how the data is generated in the real world. "Modelling" is understood in almost every science as unveiling this generating process by hypothesizing theories and testing these theories through observations.

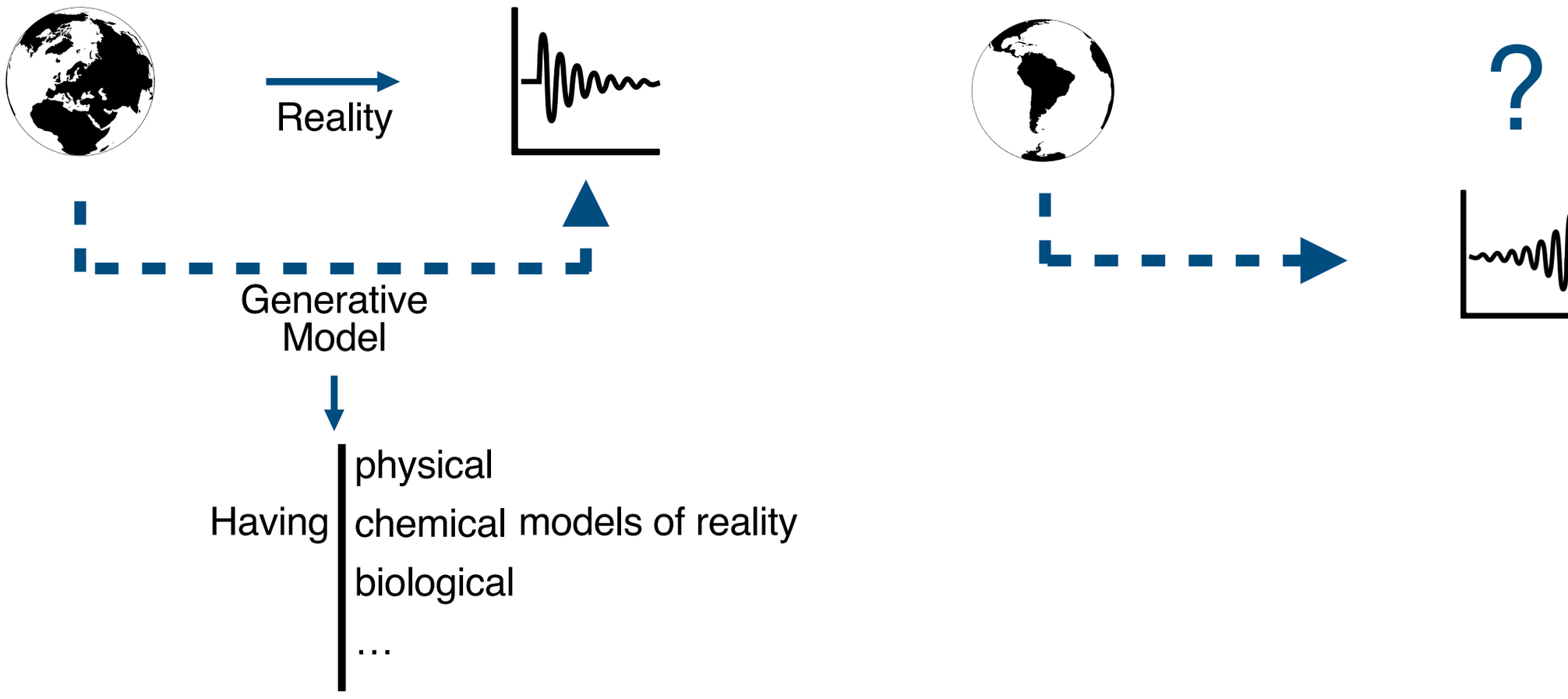
in *An Introduction to Variational Autoencoders*, D. Kingma and M. Welling, 2019



Generative models as simulators of real world

A generative model simulates how the data is generated in the real world. "Modelling" is understood in almost every science as unveiling this generating process by hypothesizing theories and testing these theories through observations.

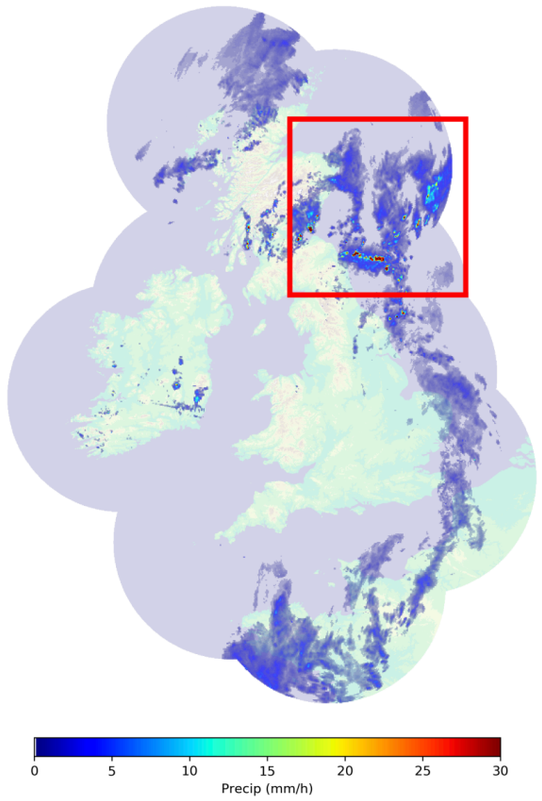
in *An Introduction to Variational Autoencoders*, D. Kingma and M. Welling, 2019



Generative models for science

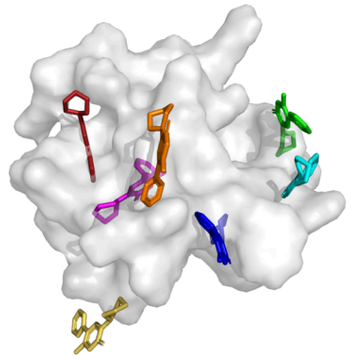
Climate Science

- Precipitation nowcasting



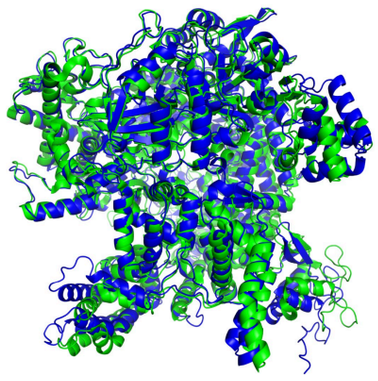
Chemistry/Drug discovery

- Molecular docking



Biology

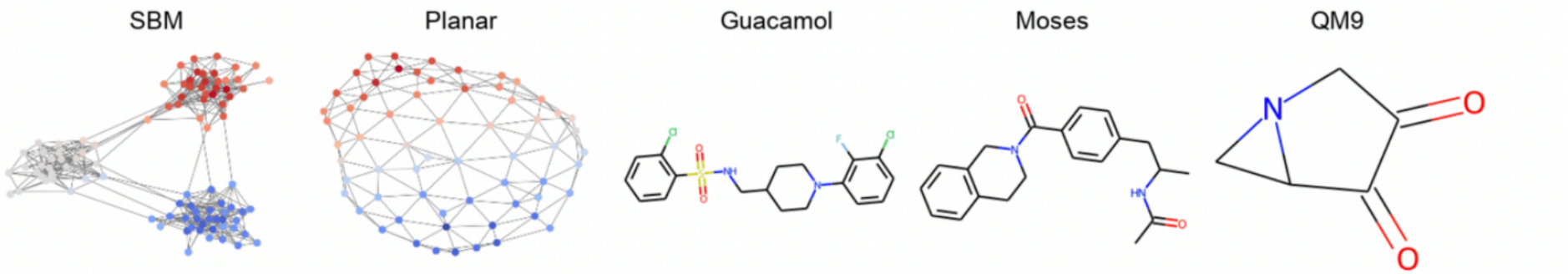
- Protein folding

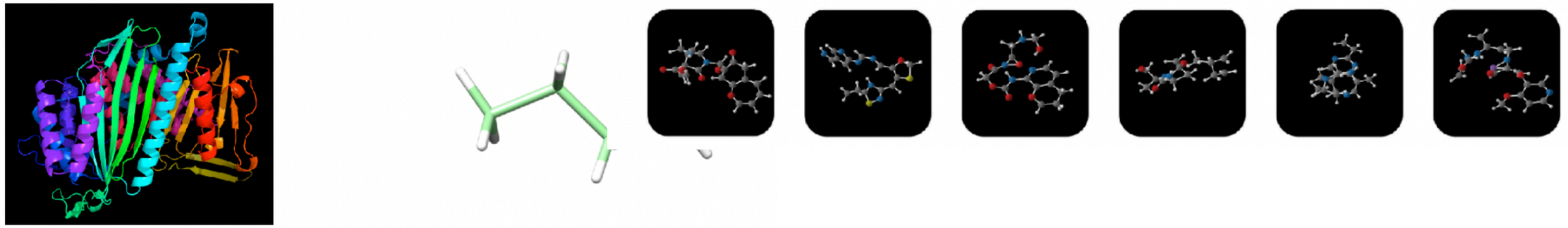


Data Completion and Compressed Representations

[*Skilful precipitation nowcasting using deep generative models of radar*, Ravuri Suman et al., Nature, 2021]
[*DiffDock: Diffusion Steps, Twists, and Turns for Molecular Docking*, Corso et al., ICLR, 2023]
[*Highly accurate protein structure prediction with AlphaFold*, Jumper et al., Nature, 2021]
[*Learning interactive real-world simulators*, Yang et al., ICLR, 2024]

Generative models in non-Euclidean domain

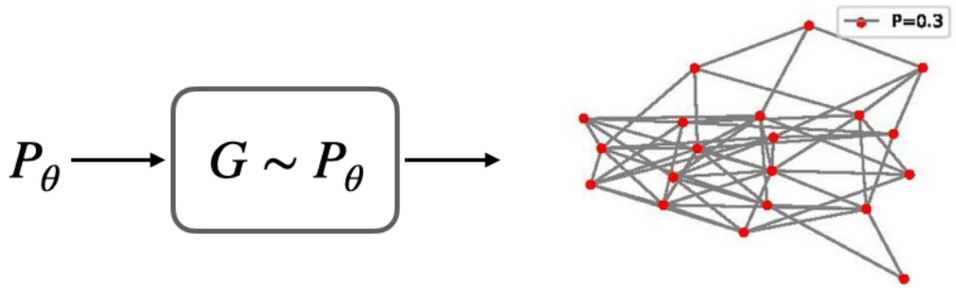




Graph generation includes the process of modelling and generating real graphs

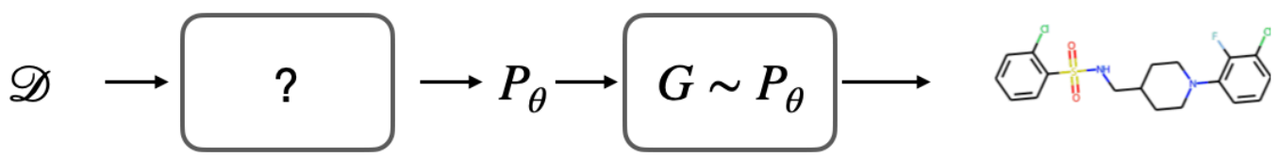
Graph generative models

- Random graph models



- Capture simple graph distribution
- Limited capacity to model complex dependencies
- Only capable of modelling a few statistical properties of graphs

- Deep generative models



- Learn generative models directly from an observed set of graphs
- Can model highly complex structures such as proteins

Today's lecture

- Quick introduction into traditional network models

- Introduction to deep probabilistic generative models

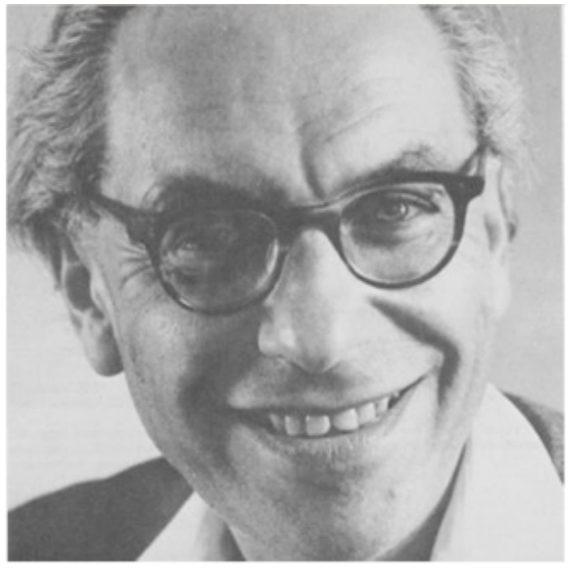
Today's lecture

- **Quick introduction into traditional network models**

- Introduction to deep probabilistic generative models

Erdös-Rényi model (1960)

Pál Erdös
(1913-1996)



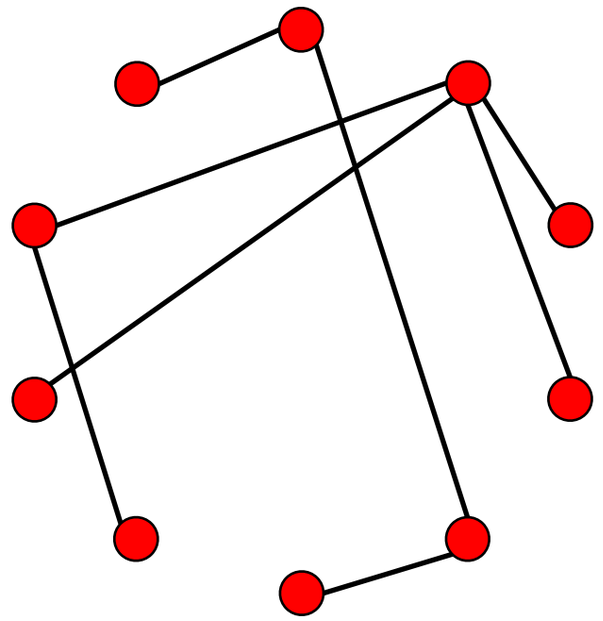
Alfréd Rényi
(1921-1970)

$G(N, L)$ model: N labeled nodes are connected with L randomly placed links.

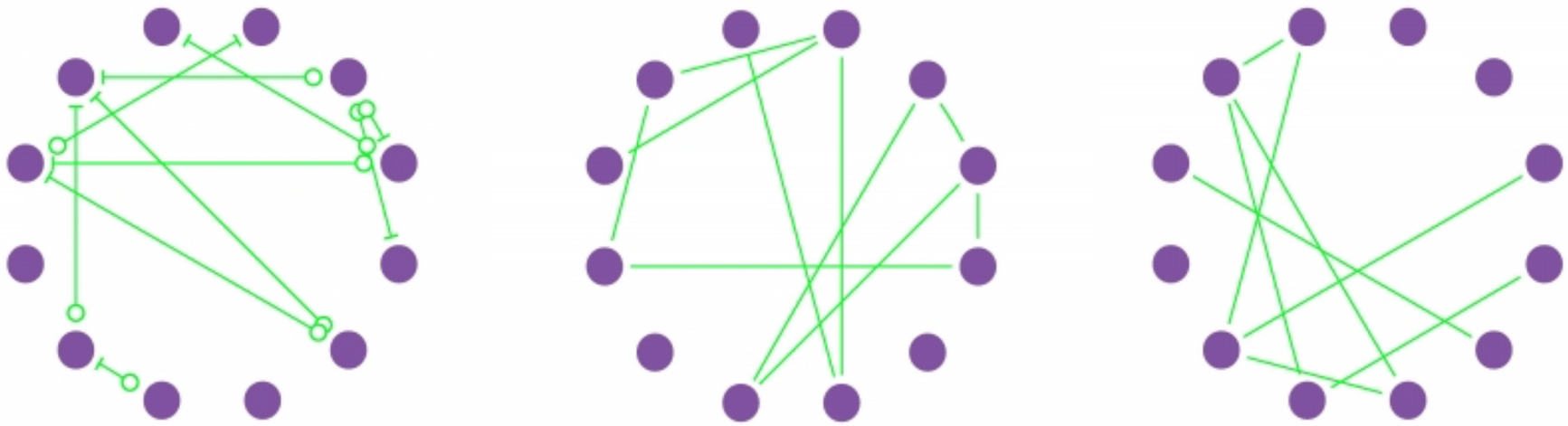
- Gilbert's $G(N, p)$ model: a random network model is a network where each pair of nodes is connected with probability *p*
  - *Example:* $p = 1/6$

    $N = 10$

    $\langle D \rangle \sim 1.5$

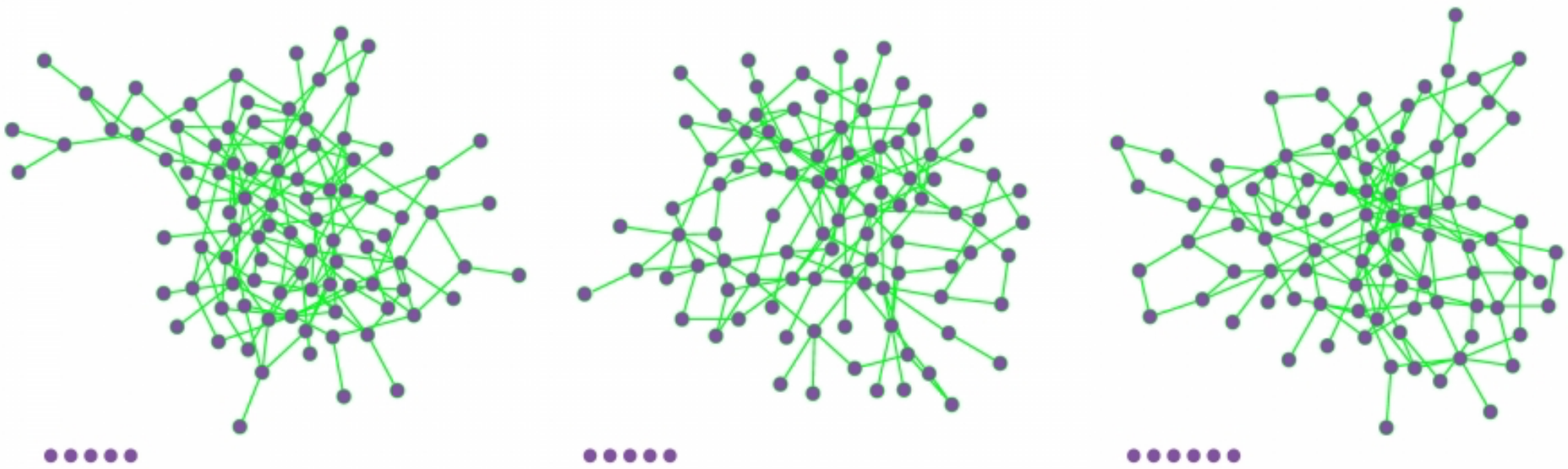


Random network examples



$p = 1/6$

$N = 12$



$p = 0.03$

$N = 100$

From [1]

Number of links in a *G(N,p)* network

- Probability for the N-node network to have *L* links (binomial distribution):

$$p_L = \begin{pmatrix} \frac{N(N-1)}{2} \\ L \end{pmatrix} p^L (1-p)^{\frac{N(N-1)}{2}-L}$$

- Expected number of links in the random graph

$$\langle L \rangle = \sum_{L=0}^{\frac{N(N-1)}{2}} L p_L = p\frac{N(N-1)}{2}$$

- Average degree $\langle k \rangle = \frac{2\langle L \rangle}{N} = p(N-1)$

Number of links in a *G(N,p)* network

- Probability for the N-node network to have *L* links (binomial distribution):

$$p_L = \left(\begin{array}{c} \frac{N(N-1)}{2} \\ L \end{array} \right) p^L (1-p)^{\frac{N(N-1)}{2} - L}$$

Number of ways to place *L* links

- Expected number of links in the random graph

$$\langle L \rangle = \sum_{L=0}^{\frac{N(N-1)}{2}} L p_L = p \frac{N(N-1)}{2}$$

- Average degree $\langle k \rangle = \frac{2 \langle L \rangle}{N} = p(N-1)$

Number of links in a *G(N,p)* network

- Probability for the N-node network to have *L* links (binomial distribution):

Probability to have *L* links

$$p_L = \left(\frac{N(N-1)}{2} \atop L \right) p^L (1-p)^{\frac{N(N-1)}{2} - L}$$

Number of ways to place *L* links

- Expected number of links in the random graph

$$\langle L \rangle = \sum_{L=0}^{\frac{N(N-1)}{2}} L p_L = p \frac{N(N-1)}{2}$$

- Average degree $\langle k \rangle = \frac{2\langle L \rangle}{N} = p(N-1)$

Number of links in a *G(N,p)* network

- Probability for the N-node network to have *L* links (binomial distribution):

Probability to have *L* links

$$p_L = \left(\begin{array}{c} \frac{N(N-1)}{2} \\ L \end{array} \right) p^L (1-p)^{\frac{N(N-1)}{2}-L}$$

Number of ways to place *L* links

Probability that other attempts did not result in a link

- Expected number of links in the random graph

$$\langle L \rangle = \sum_{L=0}^{\frac{N(N-1)}{2}} L p_L = p \frac{N(N-1)}{2}$$

- Average degree

$$\langle k \rangle = \frac{2\langle L \rangle}{N} = p(N-1)$$

Number of links in a *G(N,p)* network

- Probability for the N-node network to have *L* links (binomial distribution):

Probability to have *L* links

$$p_L = \left(\begin{array}{c} \frac{N(N-1)}{2} \\ L \end{array} \right) p^L (1-p)^{\frac{N(N-1)}{2}-L}$$

Number of ways to place *L* links

Probability that other attempts did not result in a link

- Expected number of links in the random graph

$$\langle L \rangle = \sum_{L=0}^{\frac{N(N-1)}{2}} L p_L = p \frac{N(N-1)}{2}$$

Maximum number of links

- Average degree $\langle k \rangle = \frac{2\langle L \rangle}{N} = p(N-1)$

Number of links in a *G(N,p)* network

- Probability for the N-node network to have *L* links (binomial distribution):

Probability to have *L* links

$$p_L = \left(\begin{array}{c} \frac{N(N-1)}{2} \\ L \end{array} \right) p^L (1-p)^{\frac{N(N-1)}{2} - L}$$

Number of ways to place *L* links

Probability that other attempts did not result in a link

- Expected number of links in the random graph

$$\langle L \rangle = \sum_{L=0}^{\frac{N(N-1)}{2}} L p_L = p \frac{N(N-1)}{2}$$

Maximum number of links

- Average degree

$$\langle k \rangle = \frac{2\langle L \rangle}{N} = p(N-1)$$

Maximum number of links for one node

Degree distribution in *G(N,p)*

- Degree distribution for a random network (binomial distribution)

$$p_k = \left(\begin{array}{c} N-1 \\ k \end{array} \right) p^k (1-p)^{N-1-k}$$

- Degree distribution for $\langle k \rangle \ll N$ (sparse networks) is approximated by the one of the Poisson distribution

$$p_k = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!}$$

Simpler, but does not depend on N and valid only for sparse networks!

From [1]

Degree distribution in *G(N,p)*

- Degree distribution for a random network (binomial distribution)

$$p_k = \binom{N-1}{k} p^k (1-p)^{N-1-k}$$

Probability for a random node to have *k* links

Number of ways to select *k* links

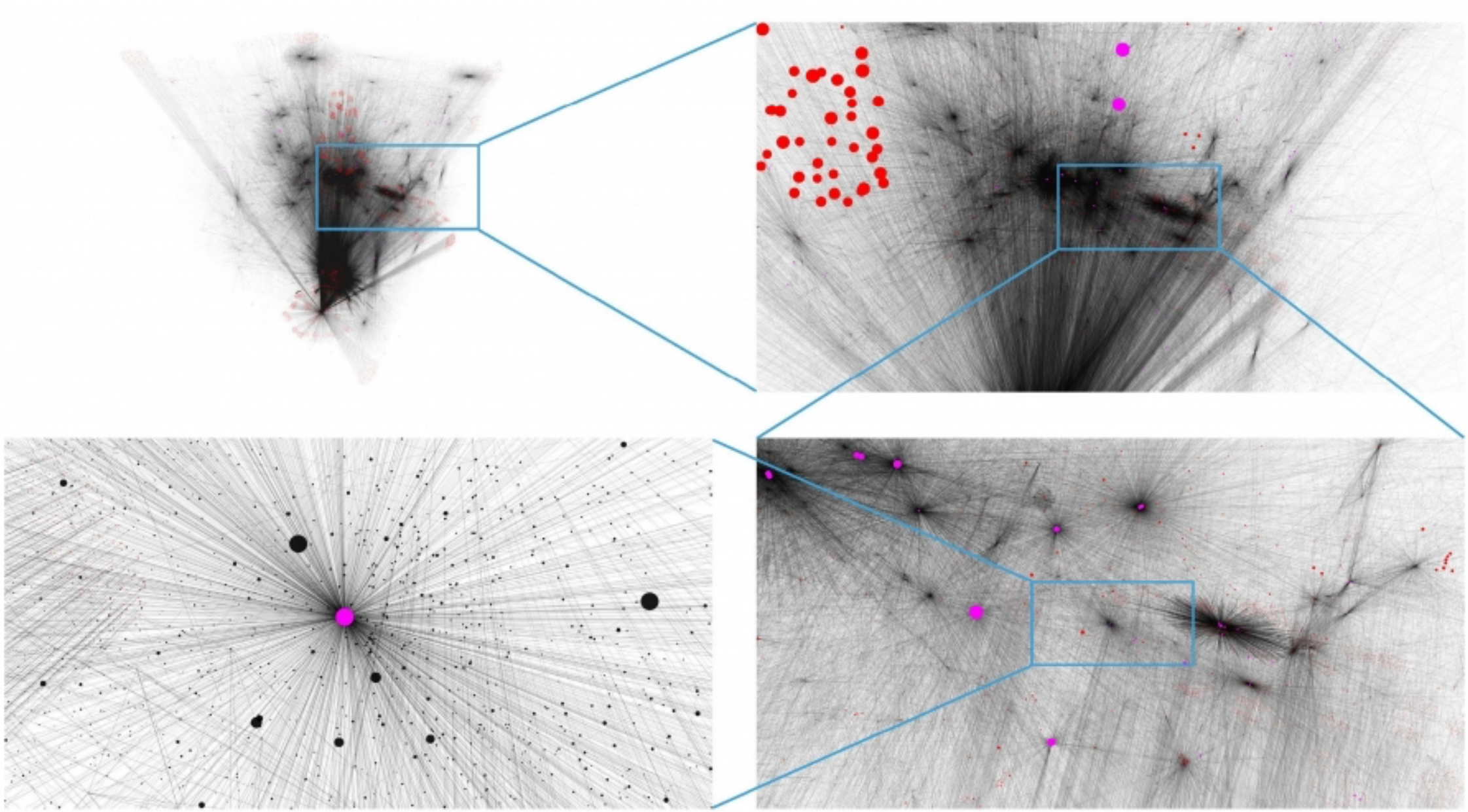
- Degree distribution for $\langle k \rangle \ll N$ (sparse networks) is approximated by the one of the Poisson distribution

$$p_k = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!}$$

Simpler, but does not depend on N and valid only for sparse networks!

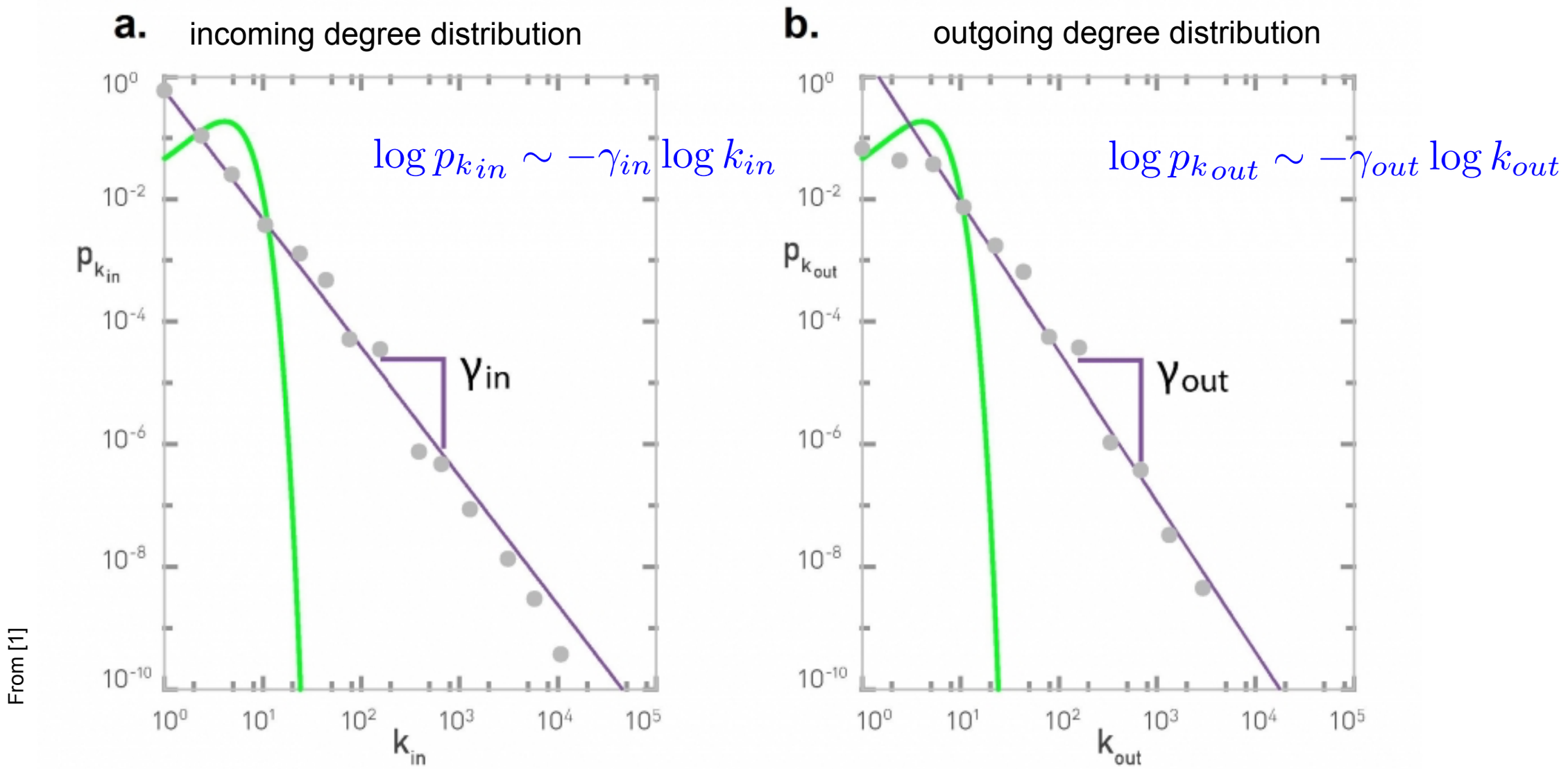
From [1]

The Web is not random



From [1]

WWW: degree distribution



- Degrees do not follow a Poisson distribution (like random networks) but rather a *power law distribution*, of the form $p_k \sim k^{-\gamma}$

Scale-free network

A *scale-free network* is a network whose degree distribution follows a *power law*.

Discrete Formalism

Probability that a node has k links:

$$p_k = Ck^{-\gamma}$$

With normalisation constraints

$$\sum_{k=1}^{\infty} p_k = 1 \quad \text{or} \quad C\sum_{k=1}^{\infty} k^{-\gamma} = 1$$

The parameter *C* becomes

$$C = \frac{1}{\sum_{k=1}^{\infty} k^{-\gamma}} = \frac{1}{\zeta(\gamma)}$$

Riemann-zeta function

Finally: $p_k = \frac{k^{-\gamma}}{\zeta(\gamma)}$

p_0 can be specified separately

Continuum Formalism

Probability of node degree between k_1 and k_2

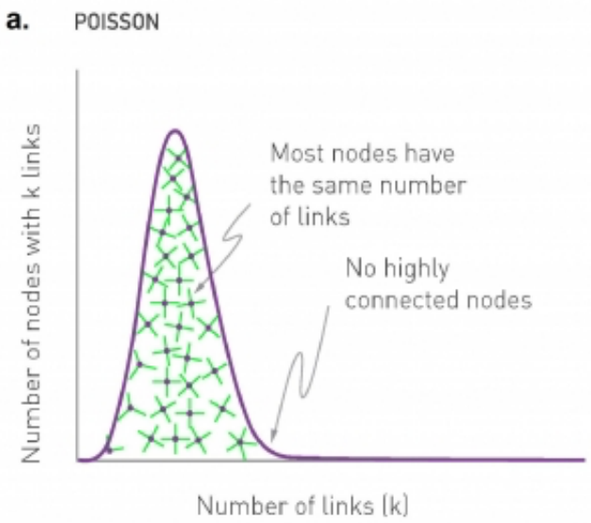
$$\int_{k_1}^{k_2} p(k)dk \qquad \text{with} \qquad p(k) = Ck^{-\gamma}$$

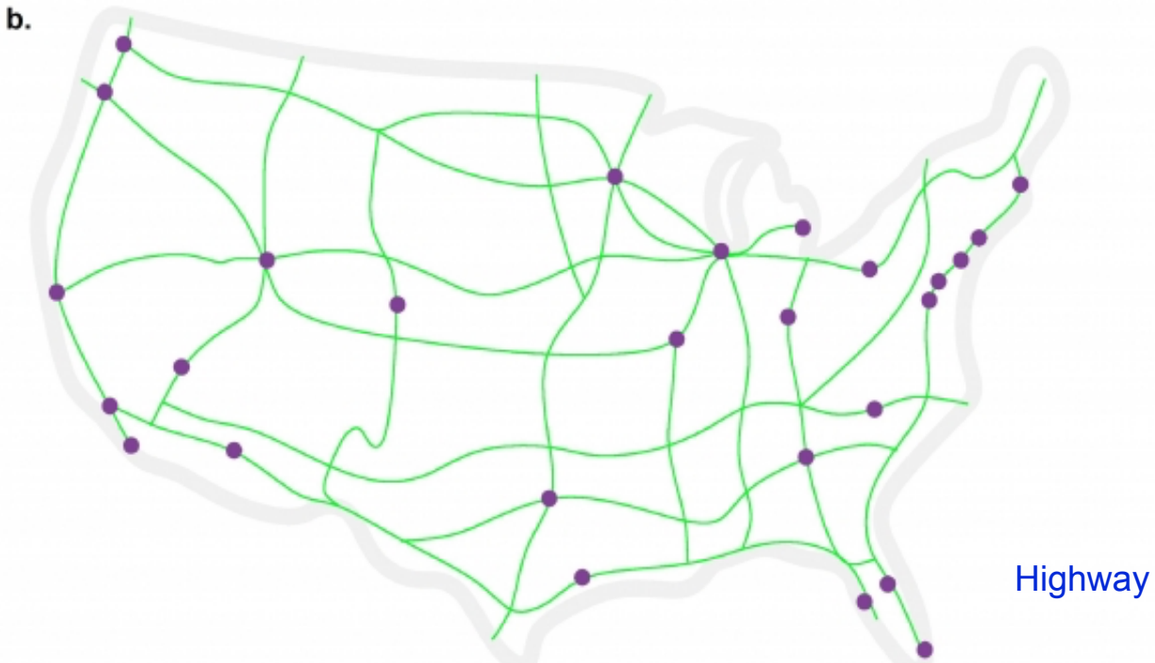
With normalisation constraints $\int_{k_{\min}}^{\infty} p(k)dk = 1$

$$C = \frac{1}{\int_{k_{\min}}^{\infty} k^{-\gamma}dk} = (\gamma-1)k_{\min}^{\gamma-1}$$

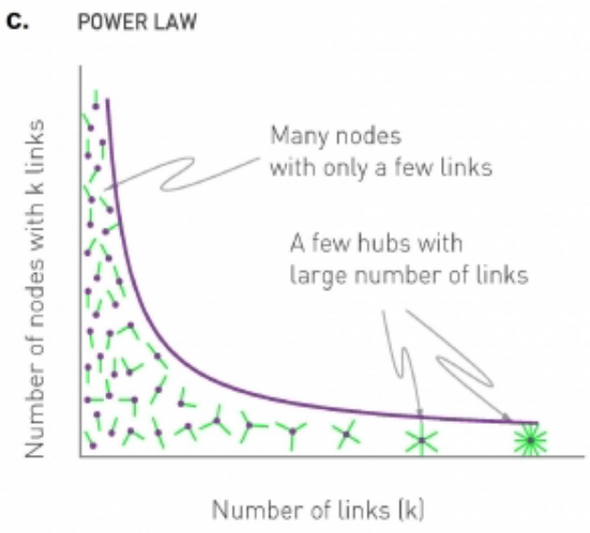
Finally: $p(k) = (\gamma-1)k_{\min}^{\gamma-1}k^{-\gamma}$

Random vs Scale-free networks





Highway network

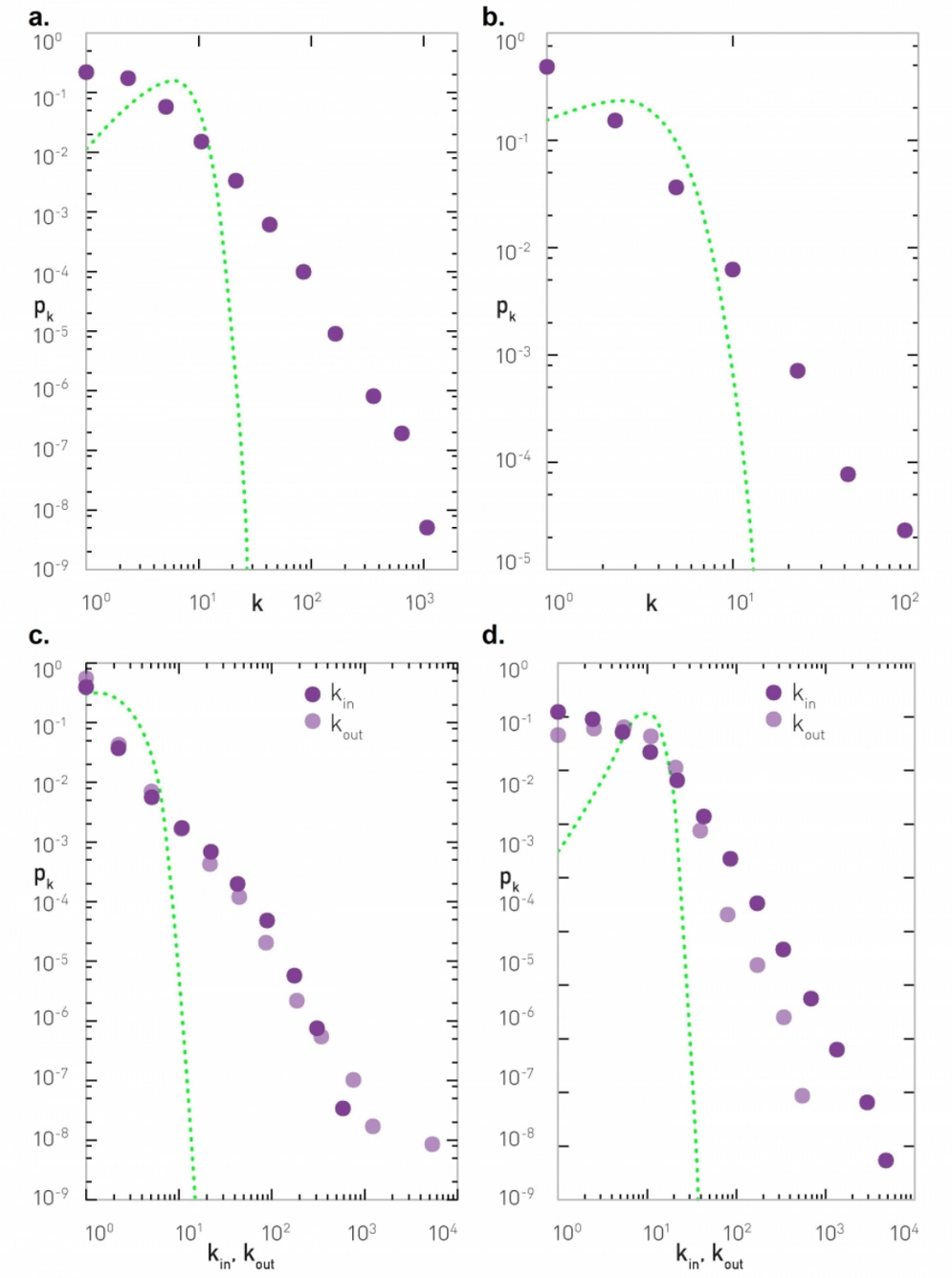


Airline network

From [1]

Universality of scale-free properties

Internet at the router level



Protein–protein interaction network

Email network

Citation network

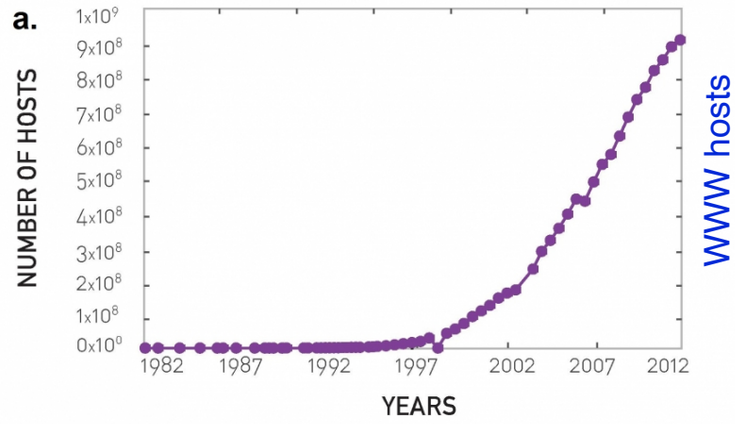
From [1]

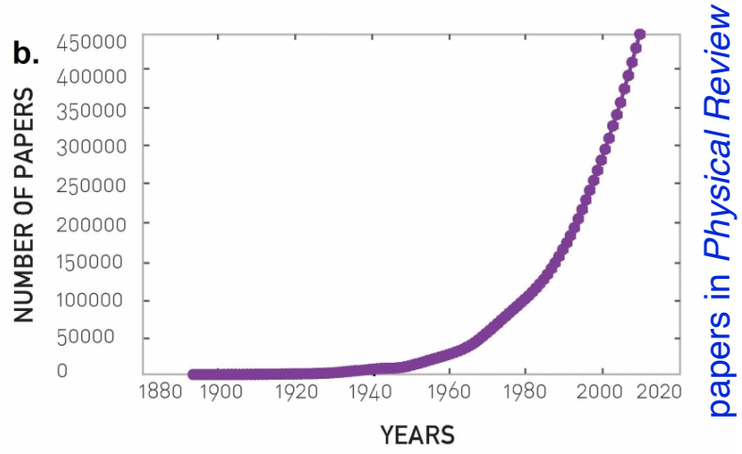
Formation of Scale-Free Networks

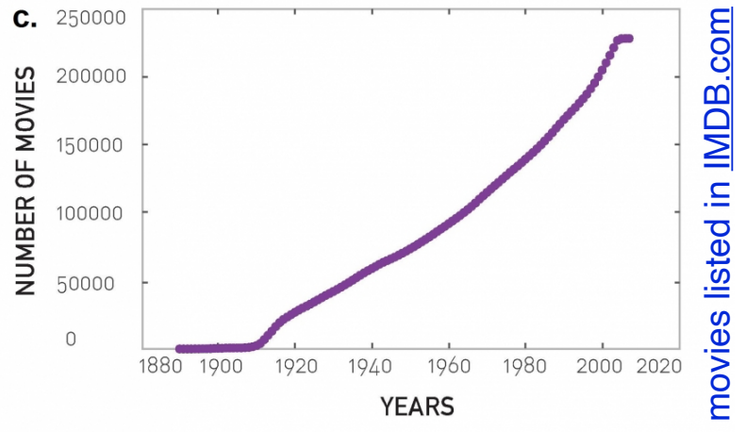
- Many networks seem to have the same properties, but they capture very different data
  - WWW and biological cellular networks are both scale-free, while they are very different

- It is important to understand how networks get formed
  - models explaining the properties of networks

- Growth and preferential attachment lead to properties similar to the ones of real networks
  - the degree distribution of real networks is quite different from the random network one
  - Barabási-Albert model, and other models

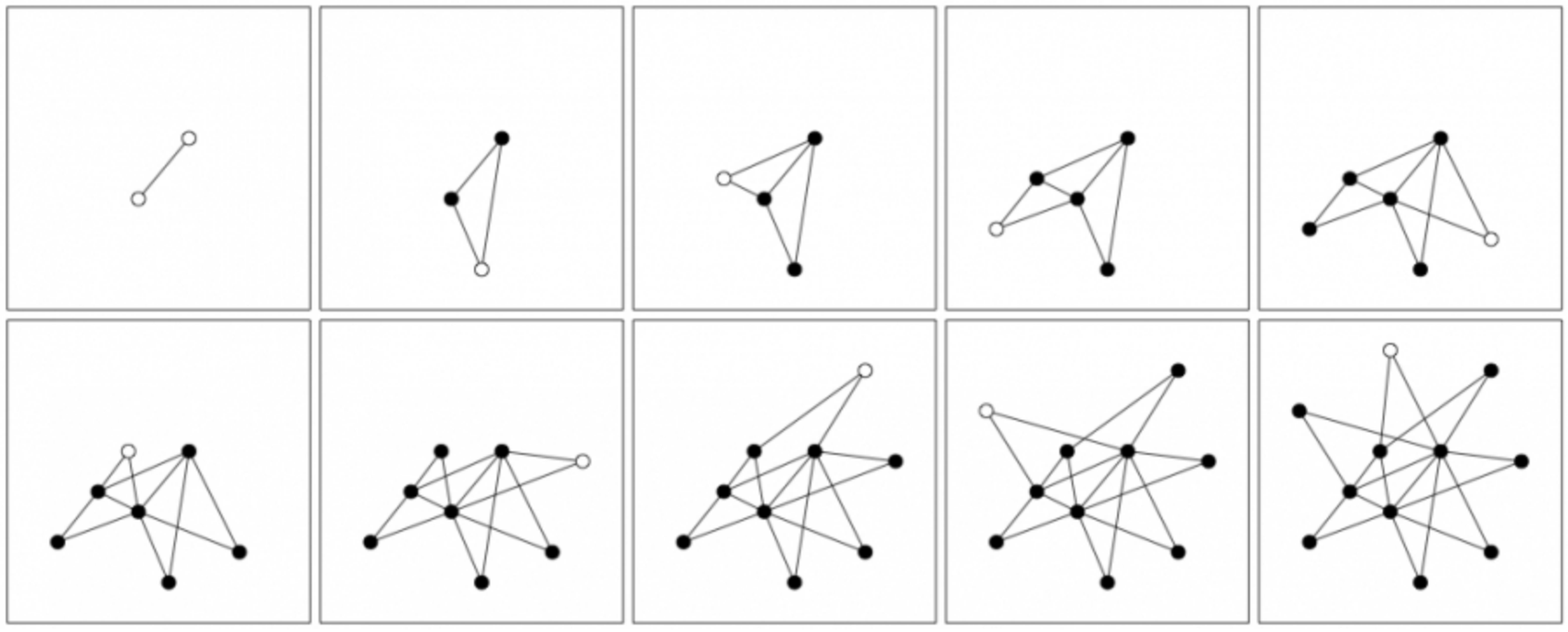




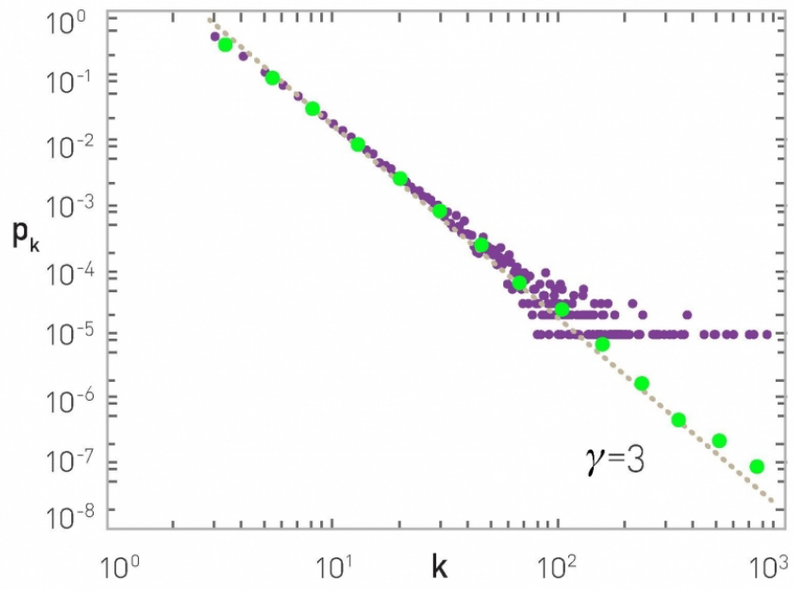


The Barabási-Albert Model

- The BA model generates scale-free networks
  - start with m_0 nodes, and choose links arbitrarily (with at least one link per node)
  - then develop the networks with growth and preferential attachment
    - Growth: add a new node with $m \leq m_0$ links that connects to *m* nodes already in the network
    - Preferential attachment: probability that the new node connects to node *i* depends on $\Pi(k_i) = \frac{k_i}{\sum_j k_j}$
  - after *t* steps, the network has $N = t + m_0$ nodes and $mt + m_0$ links, and a power-law distribution



From [1]



$N = 100000 \quad m = 3$

Summary of network models

- Network models are useful to understand or generate data
  - Simple and tractable models (often do not describe exactly real data)
  - They are based on *assumptions / heuristics* oversimplifying the underlying distributions of graphs

- Network models can help calculate many quantities and properties
  - Those can be compared to the real data
  - They can help develop insights about real data

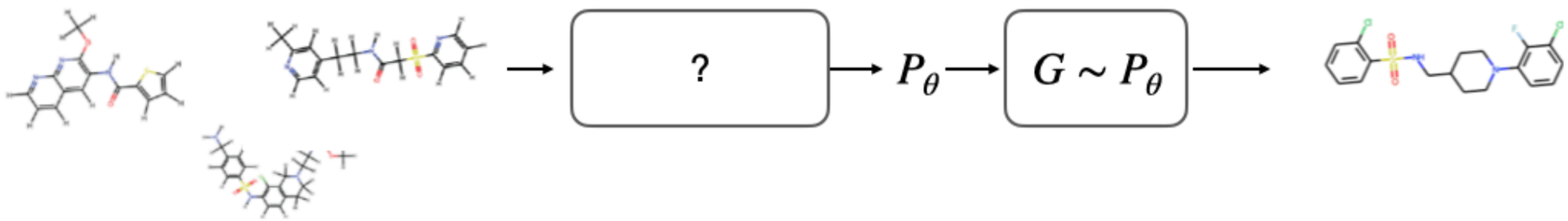
- In order to identify these properties, we need to understand how a network would look like if it is driven entirely by a model

- In particular, the random network model is a sort of benchmark model

Limited capacity to model complex dependencies

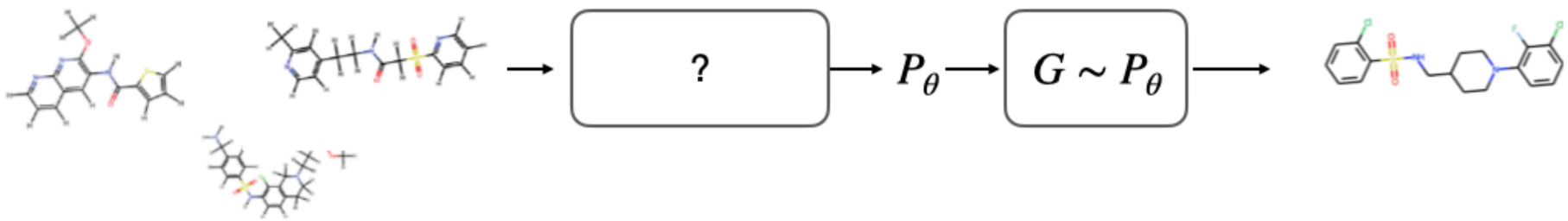
Graph Generative Models

- Given the observation $\mathscr{D} = \{G_i\}_i$, with $G_i \sim P_{data}$, we aim at learning the distribution of the observed set of graphs $P_\theta(G)$ such that sampled graphs looks like the ones in the dataset [unconditional generation]



Graph Generative Models

- Given the observation $\mathscr{D} = \{G_i\}_i$, with $G_i \sim P_{data}$, we aim at learning the distribution of the observed set of graphs $P_\theta(G)$ such that sampled graphs looks like the ones in the dataset [unconditional generation]

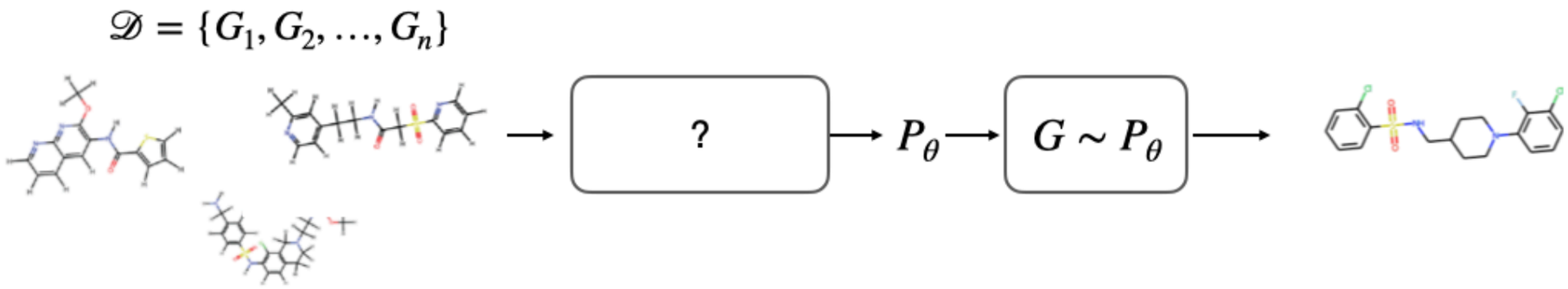


How do we learn the distribution?

What are the main challenges when graphs is the data modality?

What will you learn

- Given the observation $\mathcal{D} = \{G_i\}_i$, with $G_i \sim P_{data}$, we aim at learning the distribution of the observed set of graphs $P_\theta(G)$ such that sampled graphs looks like the ones in the dataset [unconditional generation]



A probabilistic perspective of generative models

Different families of parametrised distributions and how to learn them

Foundations of generative models (today)

Main challenges for generating graphs

Deep understanding of SOTA in diffusion models

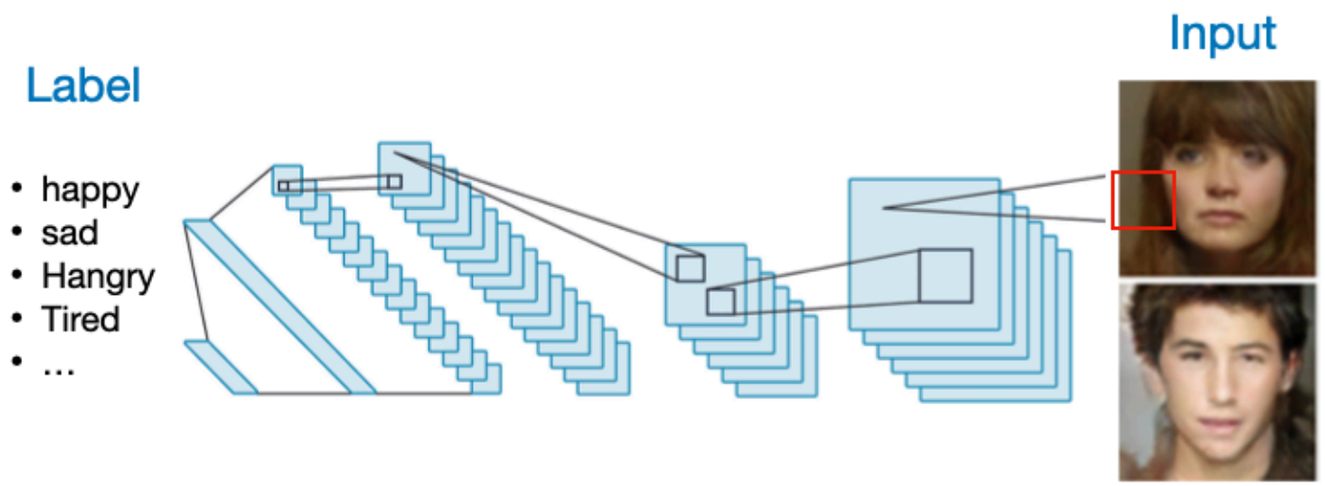
Specific to graphs (next week)

Today's lecture

- Quick introduction into traditional network models

- **Introduction to deep probabilistic generative models**

(Probabilistic Deep) Generative Task

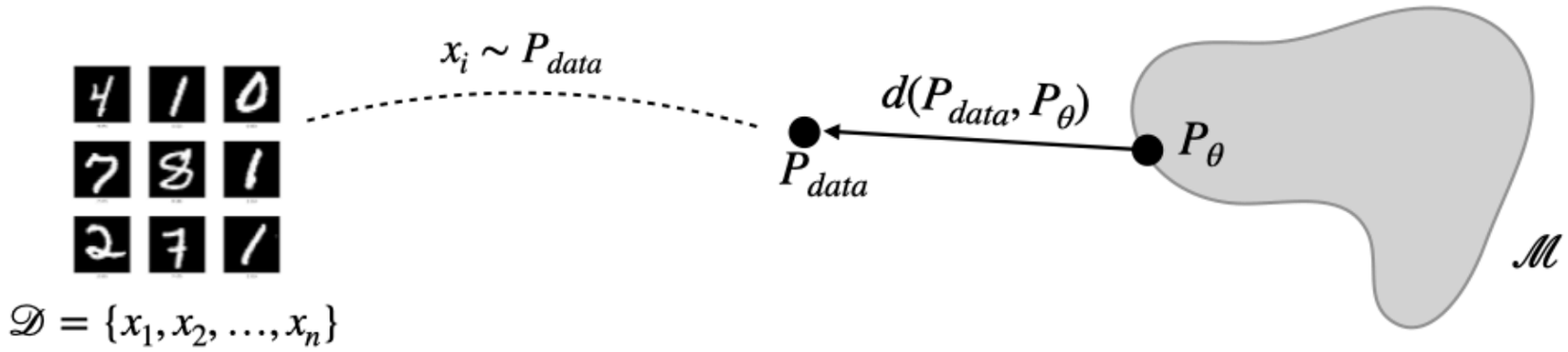


- **Generation:** Focuses on modelling the joint distribution $p(x, y)$
  - Representing $p(x, y)$ is usually intractable
  - We can impose structure on the data (e.g., conditional independence)

- A good generative model P_θ can improve downstream inference

Which is the right hypothesis/dependency to impose?
We learn it from the data!

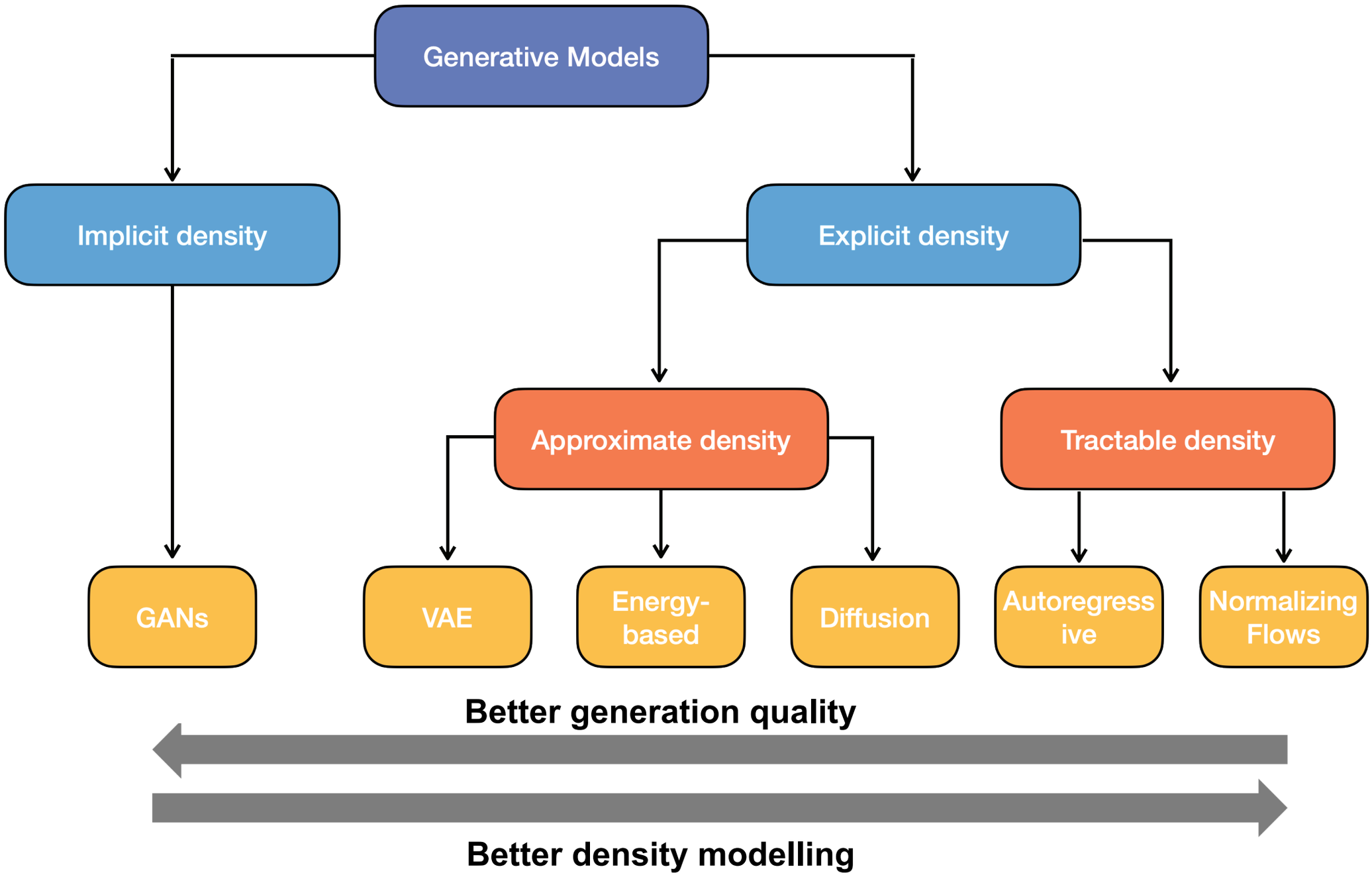
The task of generative modelling



- **Hypothesis:** $\mathscr{D}$ is an observed finite set of samples from an underlying distribution P_{data}
- **Goal:** Approximate this data distribution from $\mathscr{D}$
- **How:** We learn P_θ by maximum likelihood estimation $\theta^\star = \arg\max_{P_\theta} \mathbb{E}_{x \sim P_{data}} \left[\log P_\theta(x)\right]$

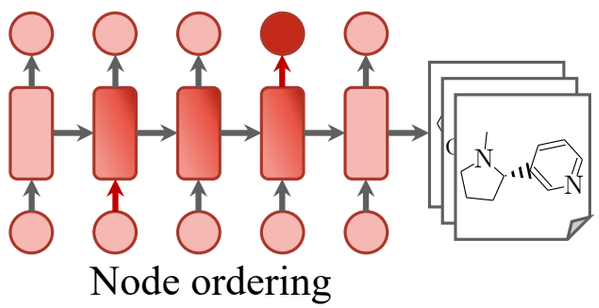
$$\begin{aligned} \mathbf{D}(P_{\text{data}} || P_\theta) &= \mathbf{E}_{\mathbf{x} \sim P_{\text{data}}} \left[\log \left(\frac{P_{\text{data}}(\mathbf{x})}{P_\theta(\mathbf{x})}\right)\right] \\ &= \mathbf{E}_{\mathbf{x} \sim P_{\text{data}}} \left[\log P_{\text{data}}(\mathbf{x})\right] - \mathbf{E}_{\mathbf{x} \sim P_{\text{data}}} \left[\log P_\theta(\mathbf{x})\right] \end{aligned}$$

Types of generative models



Autoregressive models

- The generative process is factorized as a sequential step which determines the next step activation conditioned on the proceeding one



- The joint distribution is factorized by the chain rule $p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i \mid \mathbf{x}_{<i})$
- The factorized distribution is learned via MLE

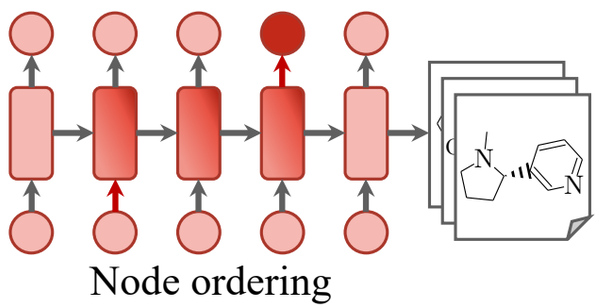
$$\arg\min_{P_\theta} \mathbf{D}(P_{\text{data}} || P_\theta) = \arg\min_{P_\theta} -\mathbf{E}_{\mathbf{x}\sim P_{\text{data}}} [\log P_\theta(\mathbf{x})] = \arg\max_{P_\theta} \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}} [\log P_\theta(\mathbf{x})]$$

- The expected log-likelihood is approximated as

$$\mathbf{E}_{\mathcal{D}} [\log P_\theta(\mathbf{x})] = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}\in\mathcal{D}} \log P_\theta(\mathbf{x})$$

Autoregressive models

- The generative process is factorized as a sequential step which determines the next step activation conditioned on the proceeding one



- The joint distribution is factorized by the chain rule $p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i \,|\, \mathbf{x}_{<i})$
- The factorized distribution is learned via MLE

$$\arg\min_{P_\theta} \mathbf{D}(P_{\text{data}} || P_\theta) = \arg\min_{P_\theta} -\mathbf{E}_{\mathbf{x}\sim P_{\text{data}}} [\log P_\theta(\mathbf{x})] = \arg\max_{P_\theta} \mathbf{E}_{\mathbf{x}\sim P_{\text{data}}} [\log P_\theta(\mathbf{x})]$$

- The expected log-likelihood is approximated as

$$\mathbf{E}_{\mathcal{D}} [\log P_\theta(\mathbf{x})] \approx \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}\in\mathcal{D}} \log P_\theta(\mathbf{x})$$

Explicit estimate of the joint distribution

Latent variable models

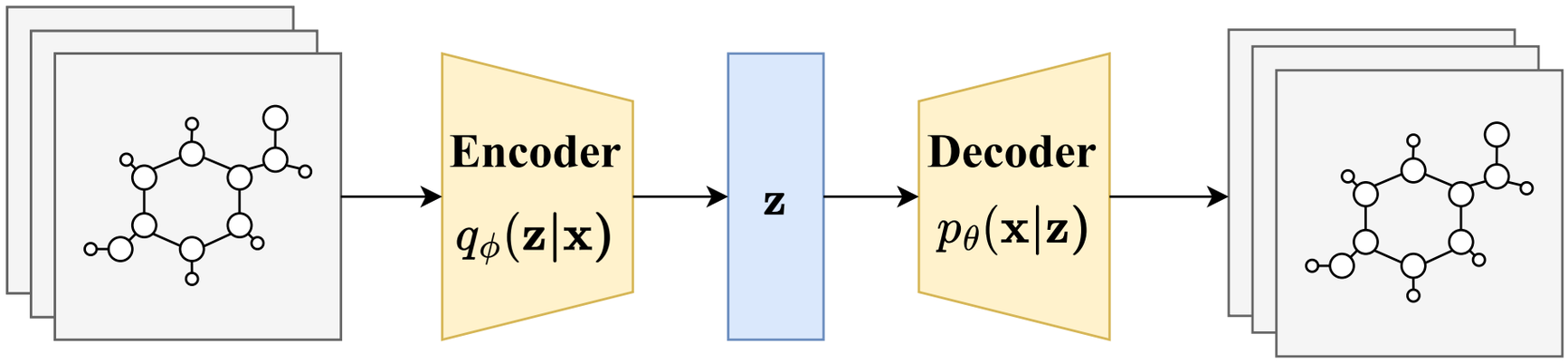
- High-dimensional variables $\mathbf{x}$ often arise from **unknown low-dimensional latent features $\mathbf{z}$**
  - These latent variables are unobserved
  - Learning them is usually intractable
  - Deep learning is used to model and infer $p(\mathbf{x} \mid \mathbf{z})$

- The joint distribution can be obtained by marginalizing over the latent variables

$$p(\mathbf{x}) = \sum_z p(x, z) = \sum_z p(x \mid z) p(z)$$

prior on the latent variable

Variational autoencoders (VAEs)



- Given x_i, the **encoder** (stochastically) compresses it using $\hat{z} \sim q_\theta(z \mid x_i)$

- Given $\hat{z}$, the decoder reconstructs the input using $p_\theta(x \mid \hat{z})$

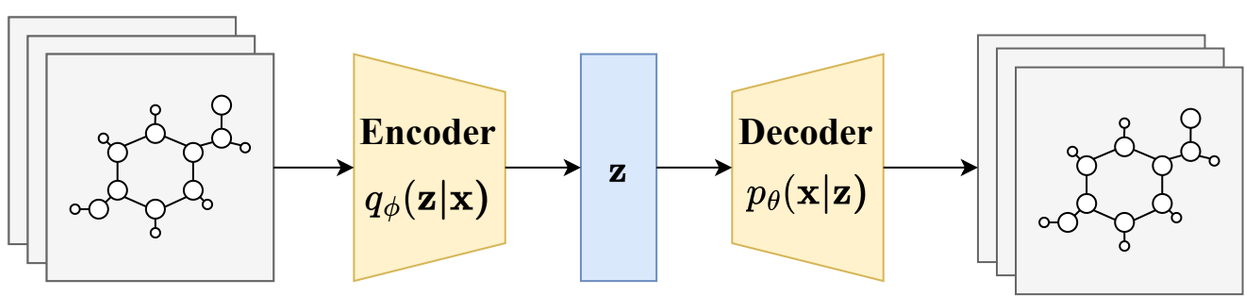
- The network is trained with the following loss

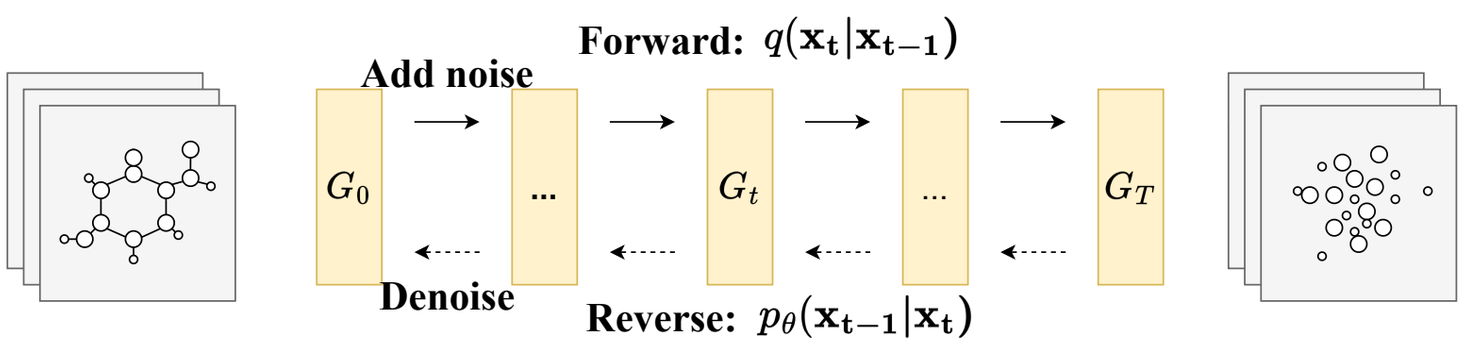
$$\mathcal{L}(\mathbf{x}; \theta, \phi) = E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z}; \theta)] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z}))$$

good reconstruction — **enforce specific shape**

Diffusion models

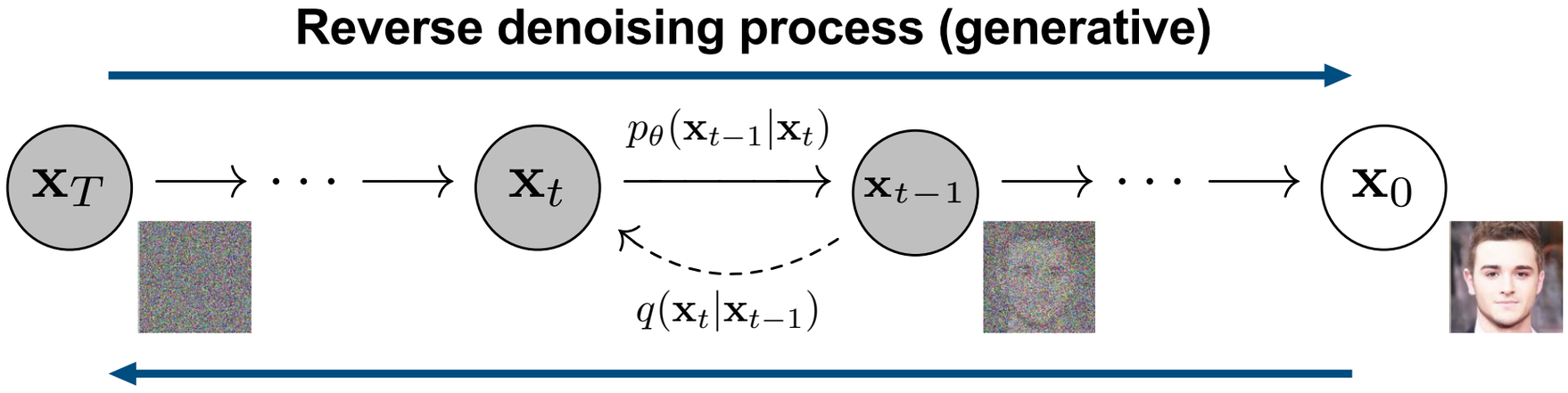
- **Encoder:** Sequential (predefined) steps with increasing level of Gaussian noise
- **Decoder:** It learns to reverse that process (denoiser)





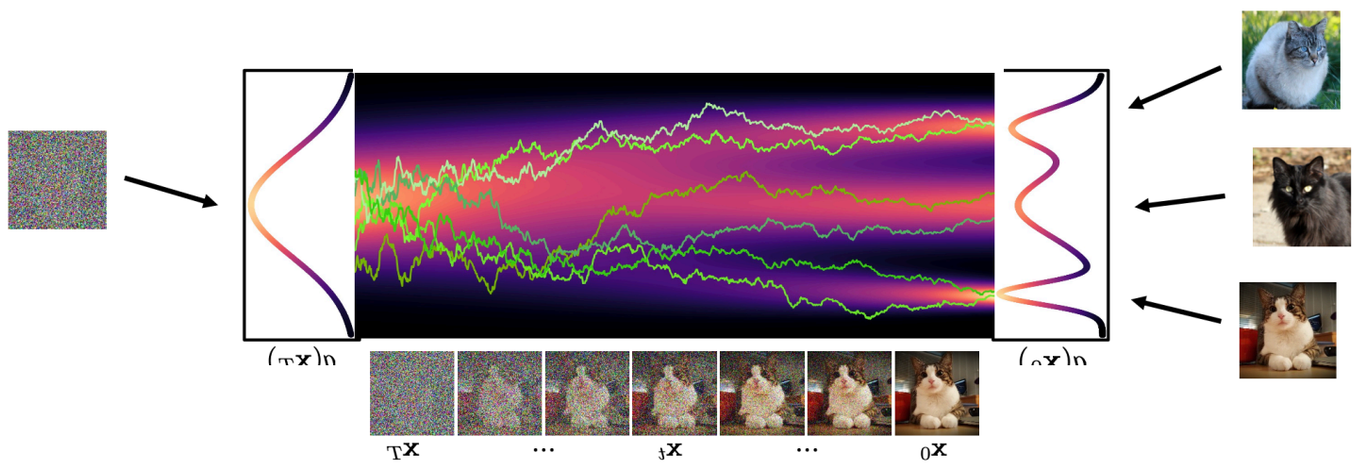
Diffusion Models: Key components

- Two main components: noise model and denoise network

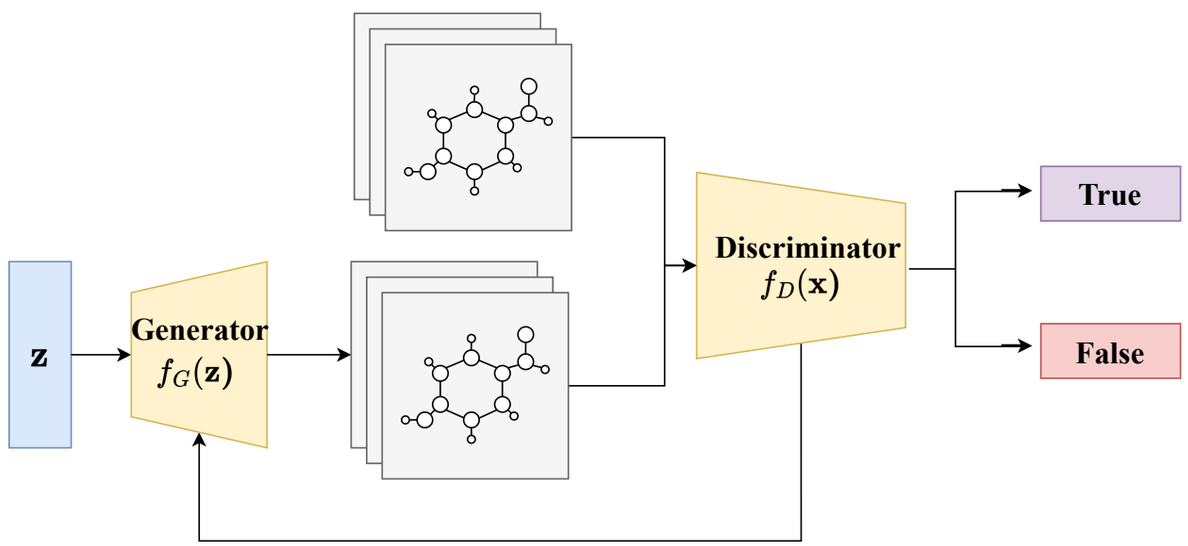


Forward diffusion process (fixed)

$q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is not tractable! → Learn $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \approx q(\mathbf{x}_{t-1}|\mathbf{x}_t)$



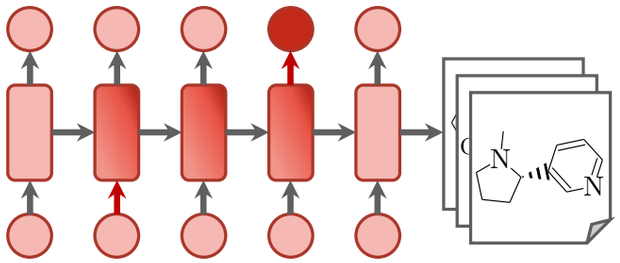
Generative adversarial networks (GANs)



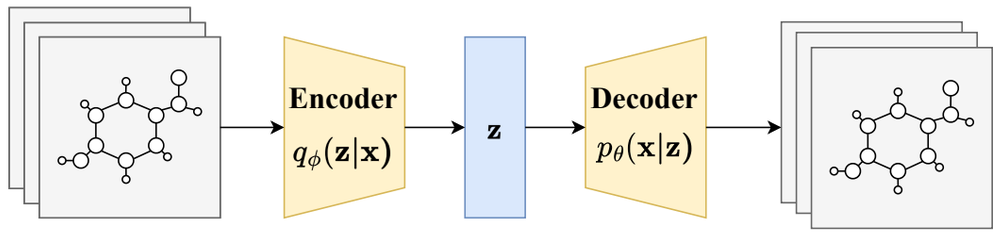
- Train two neural networks in a min-max game:
  - Generator: Produces fake data to fool the discriminators
  - Discriminator: Learns to distinguish real data from fake
- Loss function:

$$\mathscr{L} = \min \max \mathbb{E}_{\mathbf{x} \sim p_{data}} \left[\log(f_D(\mathbf{x}))\right] + \mathbb{E}_{\mathbf{z} \sim p(z)} \left[\log(1 - f_D(f_G(\mathbf{z})))\right]$$

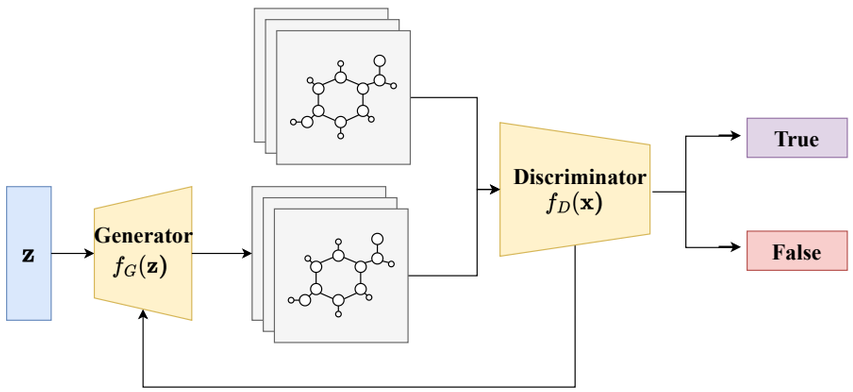
Summary



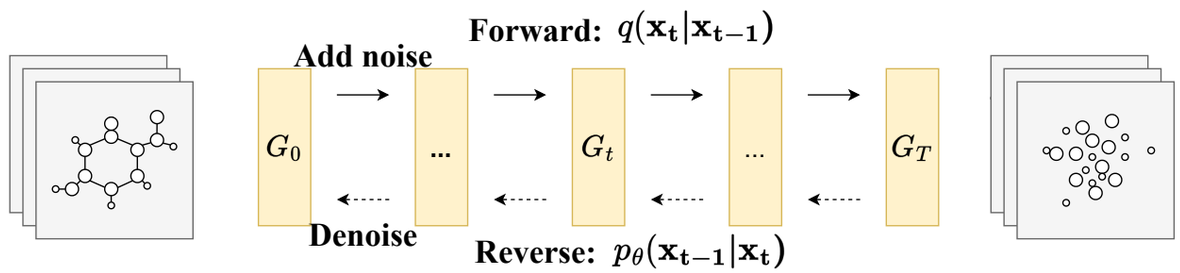
Auto-regressive models



Variational AutoEncoders



Generative Adversarial Networks (GAN)



Diffusion models

Take home messages

- **Implicit models** (GANs) generate high-quality, realistic samples but are hard to train and prone to mode collapse

- **Tractable models** (Autoregressive) offer exact likelihoods and interpretability but suffer from slow sampling due to their sequential nature; Errors are accumulated over iterations

- **Approximate models** (VAEs) enable stable training via a likelihood lower bound but produce blurrier samples

- **Diffusion models** combine benefits of others by transforming noise into data through a reverse process, achieving high fidelity but at the cost of slower sampling than VAEs and GANs; faster than autoregressive models

References

1. Network Science, by Albert-László Barabási, 2016 - Chapters 4-5

2. CS236 Deep Generative Models - Course notes - https://deepgenerativemodels.github.io/notes/index.html

2. CS 6785 - Deep Generative Models - https://kuleshov-group.github.io/dgm-website/

3. Denoising Diffusion Models: A Generative Learning Big Bang https://cvpr.thecvf.com/virtual/2023/tutorial/18546

4. Cao, Hanqun, et al. "A survey on generative diffusion models." IEEE Transactions on Knowledge and Data Engineering (2024)

References: tutorials & surveys

1. Tutorial on Deep Generative Models. A. Grover and S. Ermon. International Joint Conference on Artificial Intelligence, 2018

2. Tutorial on Generative Adversarial Networks. Computer Vision and Pattern Recognition, 2018

3. Tutorial on Deep Generative Models. S. Mohamed and D. Rezende. Uncertainty in Artificial Intelligence, 2017

4. Tutorial on Generative Adversarial Networks. I. Goodfellow. Neural Information Processing Systems, 2016

5. Learning deep generative models. R. Salakhutdinov. Annual Review of Statistics and Its Application, 2015