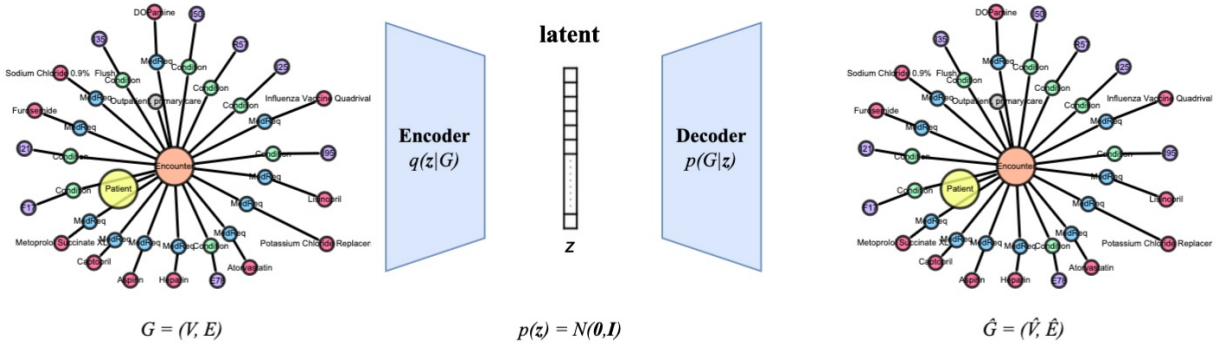
Graph Generative models and multimodality



Michalis Vazirgiannis

Ecole Polytechnique, IPP, France

http://www.lix.polytechnique.fr/dascim

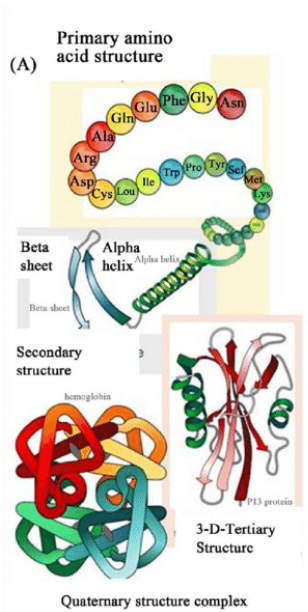
February 2025

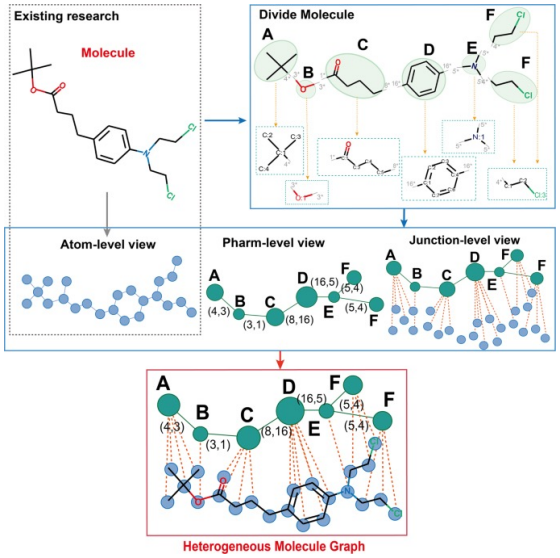
GNNs and Graph Generative models for biomedical applications

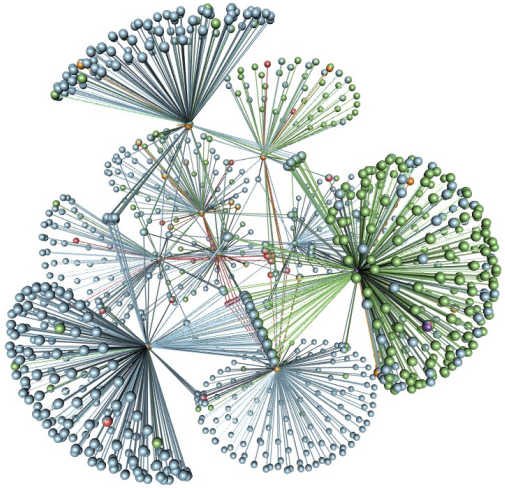
- **Graph Generative models**
- Generative models for Medical Graphs
- Multi modality for Graph generators – protein function text generator, text/mol
- Conclusions

Graphs are ubiquitous

- Chemistry – Bio/Pharma
  - Space of molecules: 10^{60}
  - New proteins, molecules generation







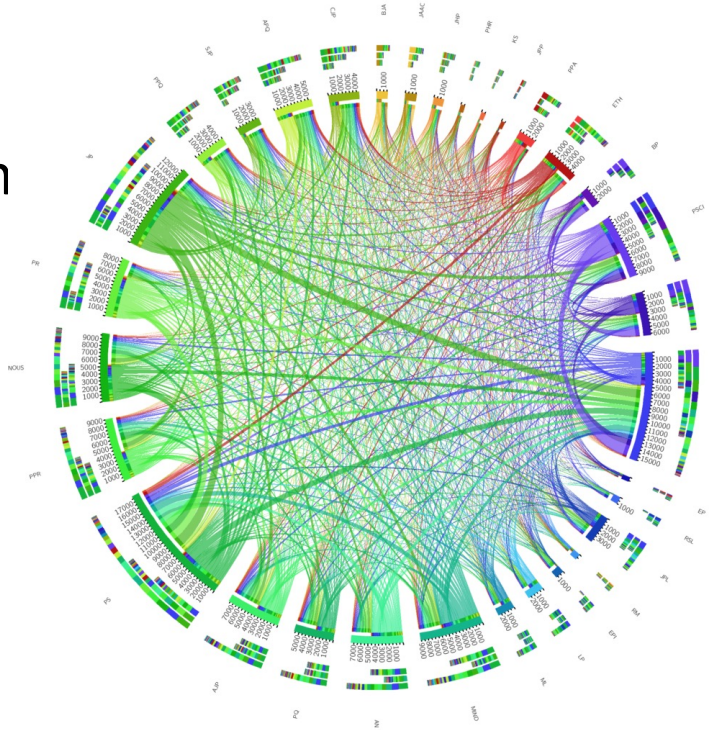
Guided Folding of Life's Proteins in Integrate Cells with Holographic Memory and GM-Biophysical Steering, Dirk K F Meijer, Hans J. H. Geesink, 2018, Open Journal of Biophysics 8(03):117-154 DOI:10.4236/ojbiphy.2018.83010,

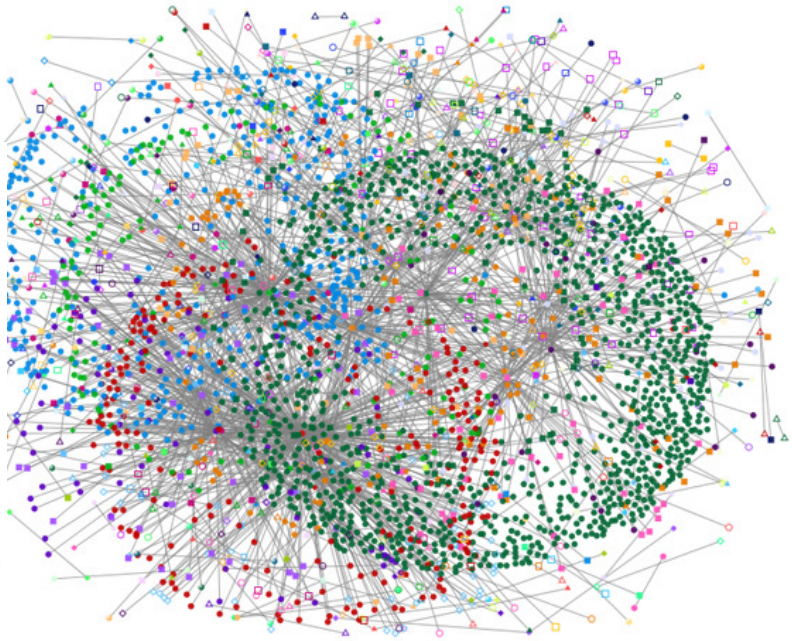
Jiang, Y., Jin, S., Jin, X. *et al.* **Pharmacophoric-constrained heterogeneous graph transformer model for molecular property prediction**. *Commun Chem* **6**, 60 (2023). https://doi.org/10.1038/s42004-023-00857-x

Multiplex Human HIV-1 protein-protein interaction network https://commons.wikimedia.org/wiki/File:Multiplex_Human_HIV-1_protein-protein_interaction_network_%28edge-colored_visualization%29.png

Graphs are ubiquitous

- Social Networks
- Internet/Telecon
- Citation graphs



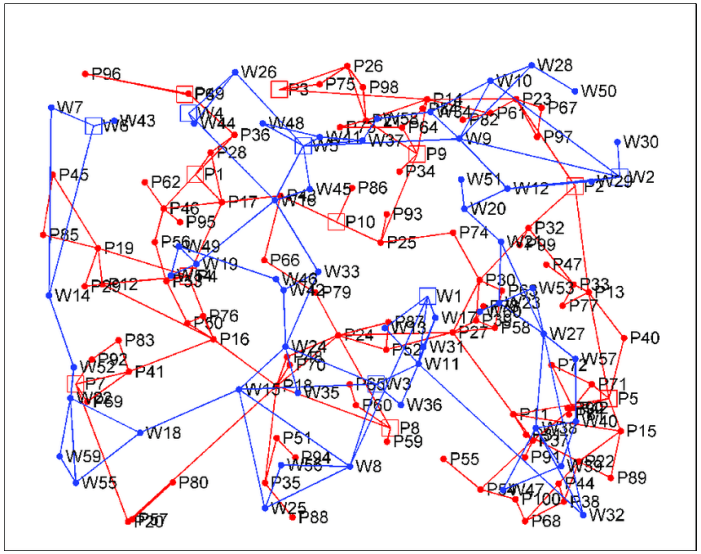


eatpost.com/researchers-graph-social-networks-spot-spammers-
346/

http://tar.weatherson.org/2017/05/04/citation-graphs-and-methodology/

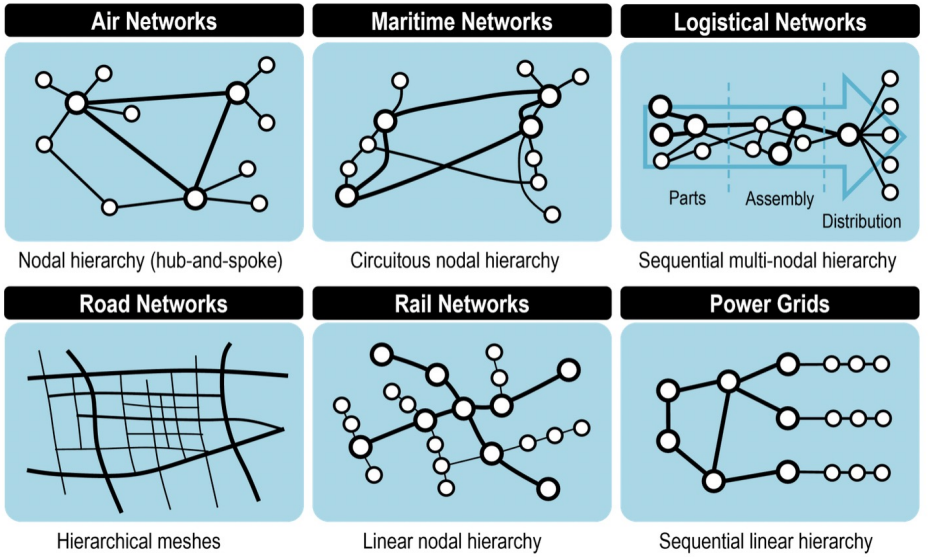
Graphs are ubiquitous

power/water distribution networks



doi.org/10.4324/9780429346323

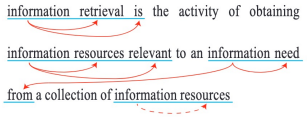
Transport/road networks



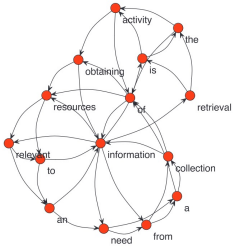
https://doi.org/10.1371/journal.pone.0195727.g005

Graphs in NLP

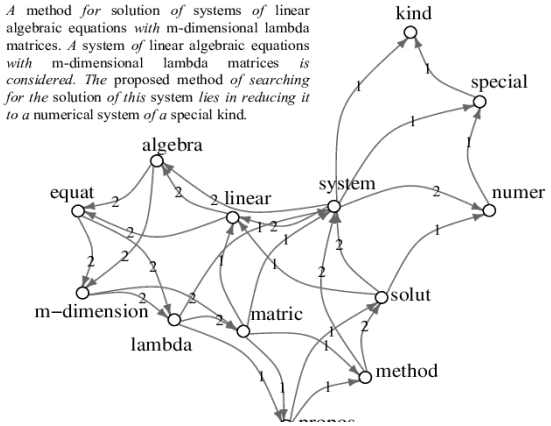
- Graph of Words
  - Information retrieval[1]
  - Keyword extraction [2]
  - Event detection [4]
  - Summarization
  - Document classification [3][1]

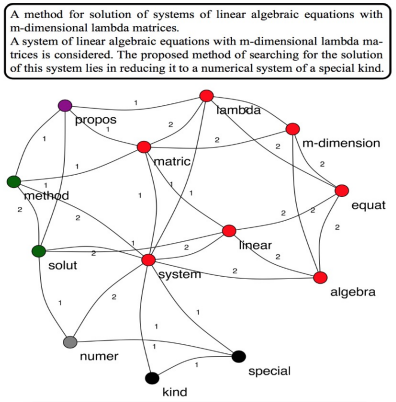


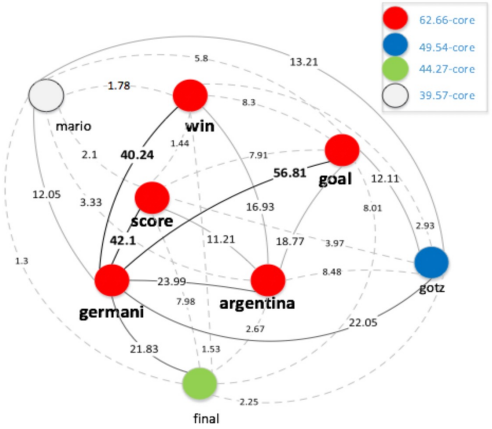
Bag of words: ((activity,1), (collection,1) (information,4), (relevant,1), (resources, 2), (retrieval, 1)..)

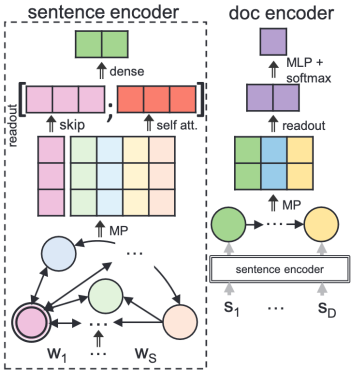


Captures: frequency, order and distance, …









[1] Graph-of-word and TW-IDF: new approach to ad hoc IR, F.Rousseau, Michalis Vazirgiannis - **CIKM '13**: https://doi.org/10.1145/2505515.2505671, Best paper mention award

[2]Main Core Retention on Graph-of-words for Single-Document Keyword Extraction, F. Rousseau,M. Vazirgiannis. **ECIR2015**

[3]Text Categorization as a Graph Classification Problem, F Rousseau, E Kiagias, M Vazirgiannis, **ACL 2015**

[4] Degeneracy-based real-time sub-event detection in twitter stream, P Meladianos, et. al. **AAAI - ICWSM 2015**

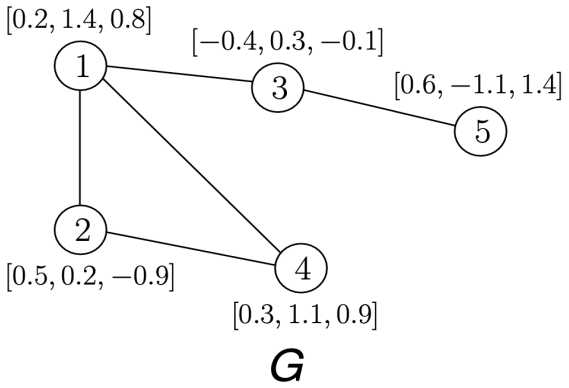
[5]Message Passing Attention Networks for Document Understanding, G. Nikolentzos, A. Tixier, M.Vazirgiannis , **AAAI2020**, https://doi.org/10.1609/aaai.v34i05.6376

Why Graph ML is important & different than sequential ML

- handles complex and rich data structures (graphs, networks, trees, and hypergraphs) not easily represented by vectors or matrices, such as.
- capture relational information and dependencies among nodes, and capitalise on the graph structure and properties to enhance the learning process – capture longer term dependencies
- can leverage the advances GNNs to learn powerful/expressive graph representations useful for downstream tasks: node classification, link prediction, graph generation, and graph matching.
- Graph pretrained models tend to have fewer parameters than traditional DL models, especially those based on transformers.

Graph Machine learning tasks

An attributed graph is a graph with attributes on vertices. Each vertex $v \in V$ is annotated with a feature vector h_v



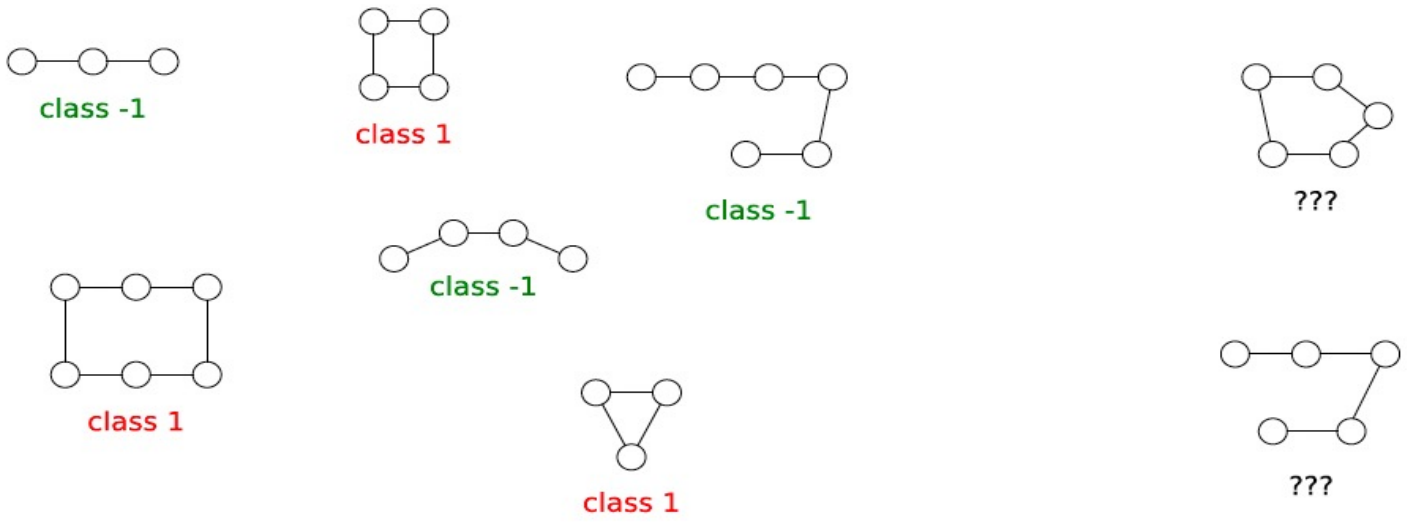
G

$h_1, \ldots, h_5 \in \mathbb{R}^3$

$h_1 = [0.2, 1.4, 0.8]^\top \quad h_3 = [-0.4, 0.3, -0.1]^\top$

- *Node classification*: given a graph with labels on some nodes, provide a high quality labelling for the rest of nodes
- *Graph clustering*: given a graph, group its vertices into clusters in such a way that there are *many edges within each cluster* and relatively few between the clusters (community detection)
- *Link Prediction*: given a pair of vertices, predict if they should be linked with an edge
- ***Graph classification***: given a set of graphs with known class labels for some of them, decide to which class the rest of the graphs belong.
- ***Graph Regression…***

Graph classification

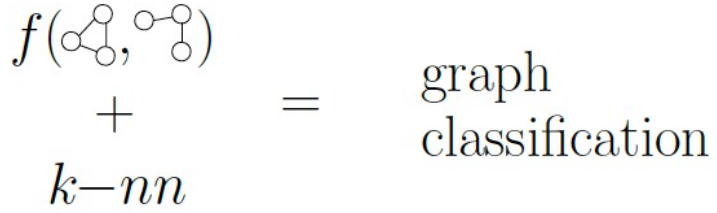


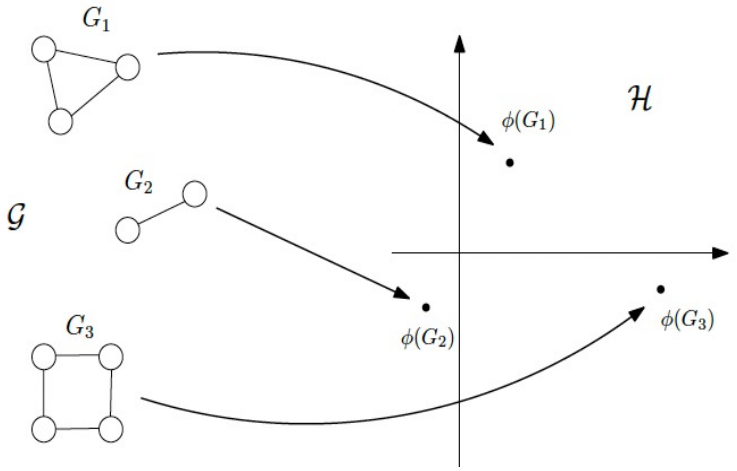
- Input data $G \in \mathcal{X}$
- Output $y \in \{-1, 1\}$
- Training set $\mathcal{D} = \{(G_1, y_1), \ldots, (G_n, y_n)\}$
- Goal: estimate a function $f : \mathcal{X} \to \mathbb{R}$ to predict y from $f(x)$

Graph classification

- Based on graph similarity – a tough issue (i.e. graph isomorphism – intractable)
- interested in polynomial time algorithms to measure the similarity between two graphs: **graph kernels**
- *GraKel*: A python library for graph kernels

  - https://ysig.github.io/GraKeL/dev/
  - Giannis Siglidis et. al. GraKeL: A Graph Kernel Library in Python, Journal of Machine Learning Research, 2020, 21, 54, pp 1-5

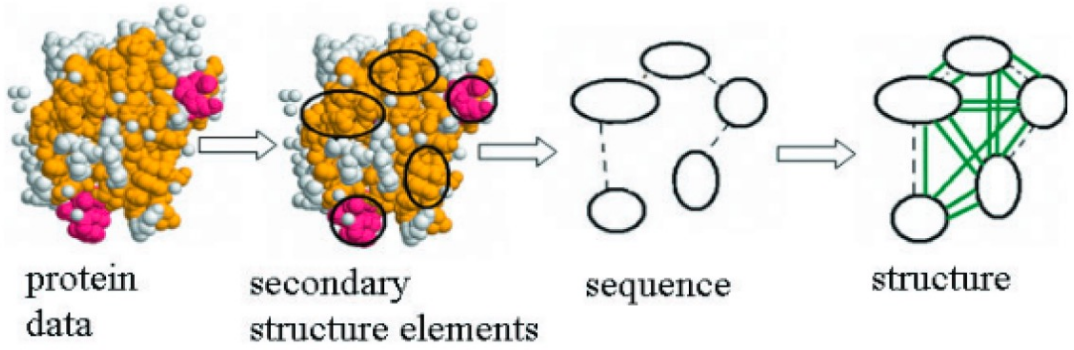




Graph classification - example

For each protein, create a graph that contains information about its

- structure
- sequence
- chemical properties

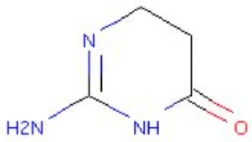


Perform **graph classification** to predict the function of proteins

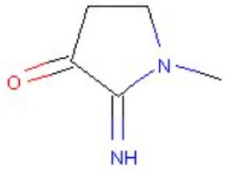
[Borgwardt et al., Bioinformatics 21]

Graph regression - example

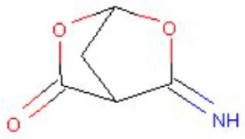
12 targets corresponding to molecular properties: ['mu', 'alpha', 'HOMO', 'LUMO', 'gap', 'R2', 'ZPVE', 'U0', 'U', 'H', 'G', 'Cv']



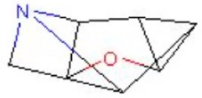
SMILES: NC1=NCCC(=O)N1
Targets: [2.54 64.1 -0.236 -2.79e-03 2.34e-01 900.7 0.12 -396.0 -396.0 -396.0 -396.0 26.9]



SMILES: CN1CCC(=O)C1=N
Targets: [4.218 68.69 -0.224 -0.056 0.168 914.65 0.131 -379.959 -379.951 -379.95 -379.992 27.934]

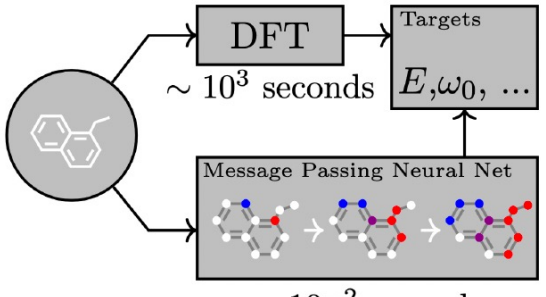


SMILES: N=C1OC2CC1C(=O)O2
Targets: [4.274 61.94 -0.282 -0.026 0.256 887.402 0.104 -473.876 -473.87 -473.869 -473.907 24.823]



SMILES: C1N2C3C4C5OC13C2C5
Targets: [? ? ? ? ? ? ? ? ? ? ? ?]

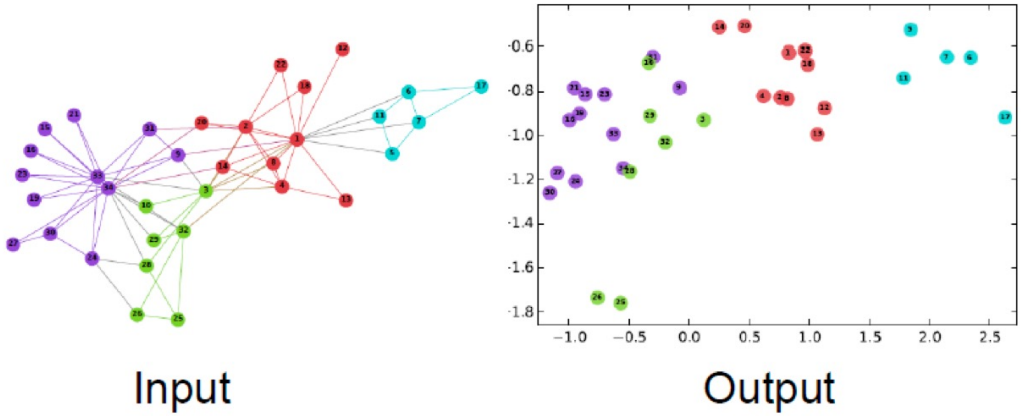
Perform **graph regression** to predict the values of the properties



[Gilmer et al., ICML'17]

Graph Neural Networks

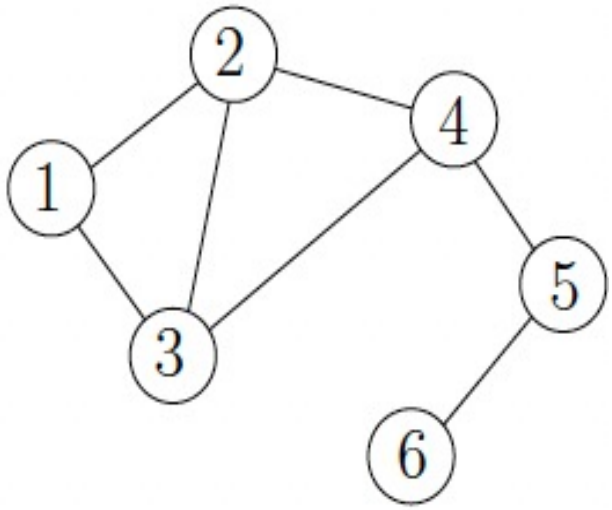
- Node Embeddings



- dimensionality d<<|V|
- similar vertices embedded close to each other in the low-dimensional space

Node embeddings - Example of message passing layer

$$h_1^{(t+1)} = f(W_0^{(t)} h_1^{(t)} + W_1^{(t)} h_2^{(t)} + W_1^{(t)} h_3^{(t)})$$
$$h_2^{(t+1)} = f(W_0^{(t)} h_2^{(t)} + W_1^{(t)} h_1^{(t)} + W_1^{(t)} h_3^{(t)} + W_1^{(t)} h_4^{(t)})$$
$$h_3^{(t+1)} = f(W_0^{(t)} h_3^{(t)} + W_1^{(t)} h_1^{(t)} + W_1^{(t)} h_2^{(t)} + W_1^{(t)} h_4^{(t)})$$
$$h_4^{(t+1)} = f(W_0^{(t)} h_4^{(t)} + W_1^{(t)} h_2^{(t)} + W_1^{(t)} h_3^{(t)} + W_1^{(t)} h_5^{(t)})$$
$$h_5^{(t+1)} = f(W_0^{(t)} h_5^{(t)} + W_1^{(t)} h_4^{(t)} + W_1^{(t)} h_6^{(t)})$$
$$h_6^{(t+1)} = f(W_0^{(t)} h_6^{(t)} + W_1^{(t)} h_5^{(t)})$$



Message Passing Neural Networks for Learning Graph Representations

Consist of a series of message passing layers followed by a readout function

Step 1: The message passing phase runs for T time steps and updates the representation of each vertex h_v^t based on its previous representation and the representations of its neighbors:

$$\mathsf{m}_v^{(t+1)} = \mathrm{AGGREGATE}\left(\left\{\mathsf{h}_u^{(t)} \middle| u \in \mathcal{N}(v)\right\}\right)$$

$$\mathsf{h}_v^{(t+1)} = \mathrm{COMBINE}\left(\mathsf{h}_v^{(t)}, \mathsf{m}_v^{(t+1)}\right)$$

where $\mathcal{N}(v)$ is the set of neighbors of v, and $\mathrm{AGGREGATE}$ and $\mathrm{COMBINE}$ are message functions and vertex update functions respectively

Step 2: The readout step computes a feature vector for the whole graph using some readout function R:

$$\mathsf{h}_G = \mathrm{READOUT}\left(\left\{\mathsf{h}_v^{(T)} \middle| v \in G\right\}\right)$$

Need for Graph Generators

- Graph generator models can produce graphs with given properties or typology for various applications
- Modeling and studying networks in biology, engineering, and social sciences.
  - simulate the evolution of social networks,
  - the structure of protein-protein interactions,
  - topology of power grids.
- Discovering new graph structures and properties.
  - generate novel chemical and molecular structures,
  - design new materials,
  - explore the space of possible graphs with certain characteristics.
- Completing and enhancing existing graphs.
  - fill in missing nodes and edges,
  - add new features and attributes,
  - improve the quality and diversity of graph data.

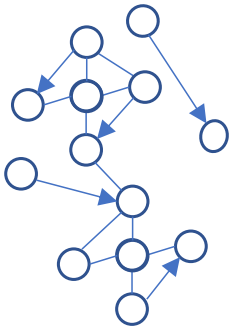
Need for **Deep** Graph Generators

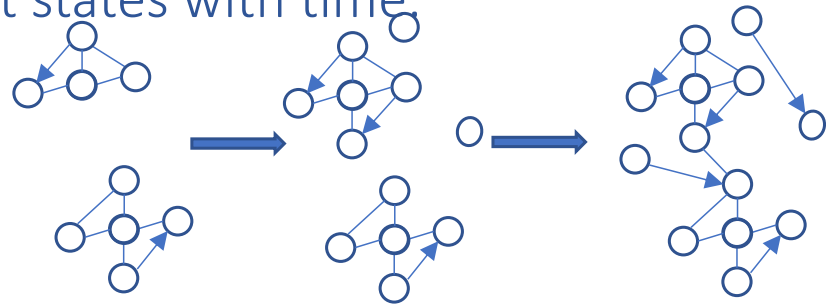
- Traditional graph generation models (i.e. *Erdős-Rényi, Barabási-Albert model, Kronecker graphs, Stochastic block models*) are based on assumptions / heuristics oversimplifying the underlying distributions of graphs.
- Deep models for graph-structured data enable effective complex graph generation
  - can learn the generative model directly from observed data, without relying on hand-engineered processes or pre-defined statistical properties.
  - capture the complex joint probability of all nodes and edges in the graph, and generate realistic graphs that match the structural characteristics of the target distribution.
  - incorporate various advanced methods: (i.e. attention mechanisms, reinforcement learning) to enhance the quality and diversity of the generated graphs.

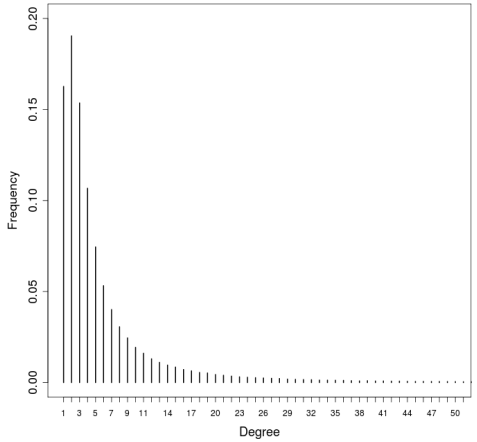
Graph Generative Models

Graph generation challenging task:

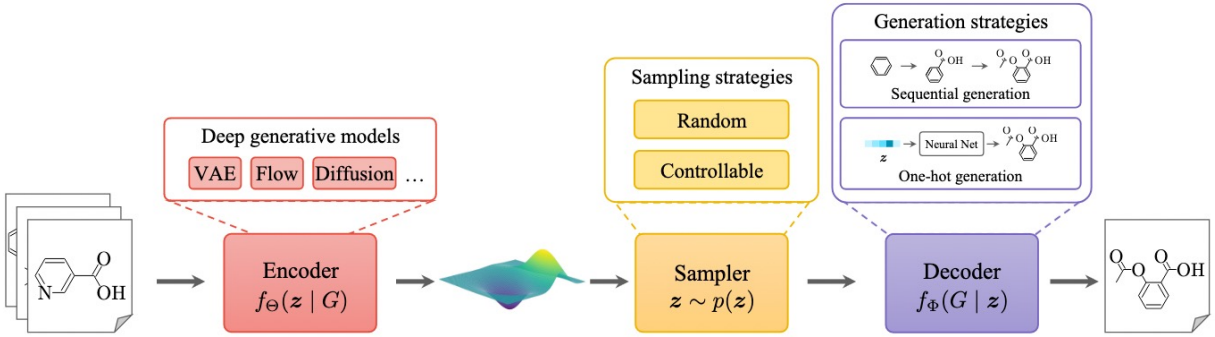
- higher-order (and non symmetric) relationships
- Sparsity and no deterministic order in processing nodes,
- long-tailed distribution of relationships: some are frequent others very rare in real-life graphs.
- dynamic and temporal: change over time, different states with time.







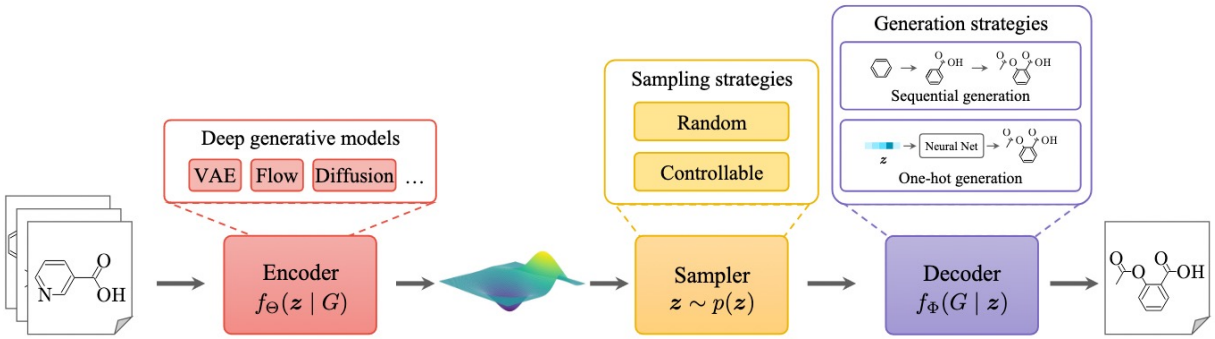
Overview of deep graph generation



A Survey on Deep Graph Generation: Methods and Applications, Yanqiao Zhu et al, LOG 2023

- *encoder* maps observed graphs into a stochastic distribution;
- *sampler* draws latent representations from that distribution;
- *decoder* receives latent codes and produces graphs

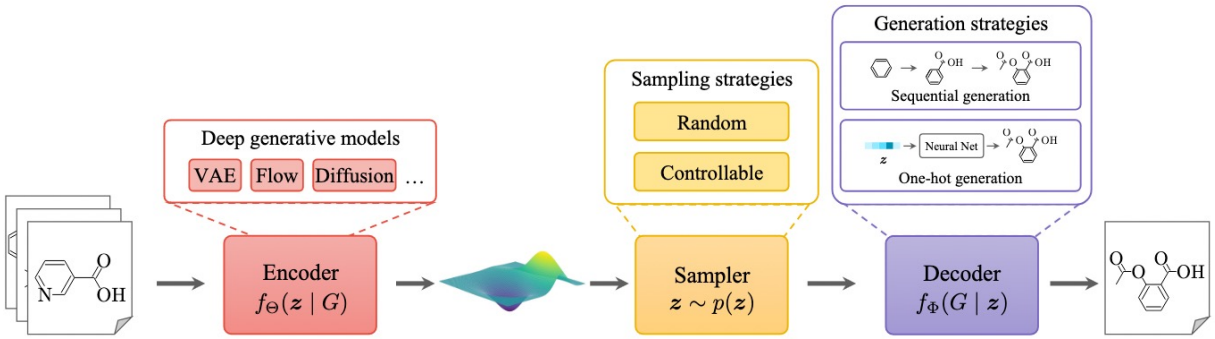
Overview of deep graph generation - Encoder



A Survey on Deep Graph Generation: Methods and Applications, Yanqiao Zhu et al, LOG 2023

- Encoder - The encoding function $f_\Theta(z \mid G)$ represent discrete graph objects as dense, continuous vectors.
- employ probabilistic generative models (e.g., variational graph neural networks) as the encoder.
- encoder function f_Θ outputs the parameters of a stochastic distribution following a prior distribution $p(z)$.

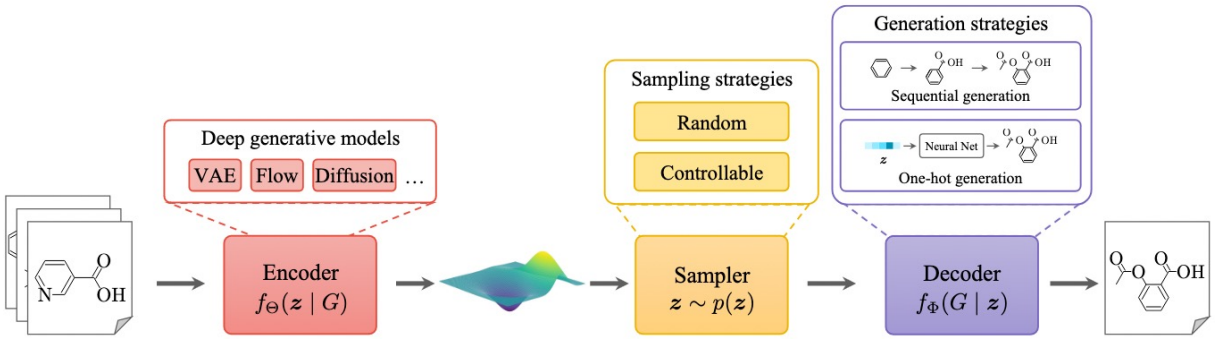
Overview of deep graph generation - Sampler



A Survey on Deep Graph Generation: Methods and Applications, Yanqiao Zhu et al, LOG 2023

- sample latent representations from learned distribution $z \sim p(z)$.
- two sampling strategies: random sampling and controllable sampling.
- Random: randomly sampling latent codes from the learned distribution.
- controllable: sample latent code in an attempt to generate new graphs with desired properties.
  - In practice, controllable sampling usually depends on different types of deep generative models and requires an additional optimization term beyond random generation.

Overview of deep graph generation - Decoder

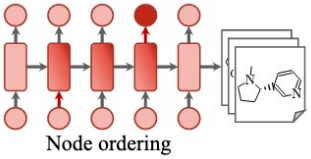


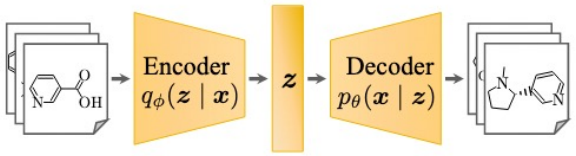
A Survey on Deep Graph Generation: Methods and Applications, Yanqiao Zhu et al, LOG 2023

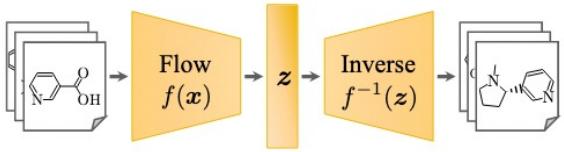
- The decoder receives the latent representations sampled from the learned distribution and generates graph structures.
- decoder is more complicated due to the discrete, non-Euclidean nature of graph objects.
- Decoder types:
  - sequential generation: generating graphs in consecutive steps, one node/edge at a time.
  - (autoregressive) one-shot generation - generating node/edge feature matrices in single step.
- not all methods include all components – i.e. (GANs) do not include a specific encoder

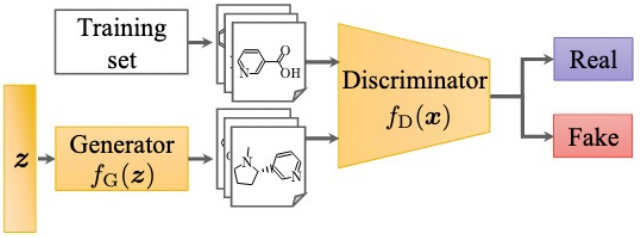
Graph generative models for deep graph generation

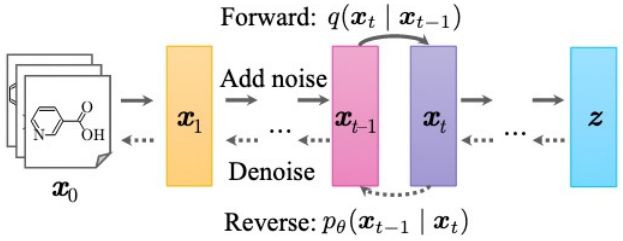
- **auto-regressive models**
- variational autoencoders
- normalizing flows
- generative adversarial networks
- diffusion models











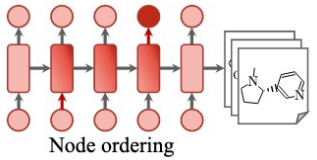
A Survey on Deep Graph Generation: Methods and Applications, Yanqiao Zhu et al, LOG 2023

Deep Graph Generators -Auto-Regressive models

- AR models: Likelihood of a joint distribution over N random variables (nodes/edges) - chain rule of probability.
- generation process: determines the next step action (add node/edge/stop)given the current subgraph. The general formulation of AR models is as follows:

$$p(G^{\pi}) = \prod_{i=1}^{N} p(G_i^{\pi} \mid G_1^{\pi}, G_2^{\pi}, \cdots, G_{i-1}^{\pi}) = \prod_{i=1}^{N} p(G_i^{\pi} \mid G_{<i}^{\pi})$$

- where $G_{<i}^{\pi} = \{G_1^{\pi}, G_2^{\pi}, \cdots, G_{i-1}^{\pi}\}$ random variables in the previous N steps.
- constraint: AR - sequential generation, requires pre-specified ordering of nodes in the graph.
- Many efforts [GraphRNN, 2018], [DeepGMG2018][Bacciu et al.2020], [Goyal et al.2020] , MolecularRNN [2021]…



Deep Generative Models of Graphs - DeepGMG [Li et al., 2018]

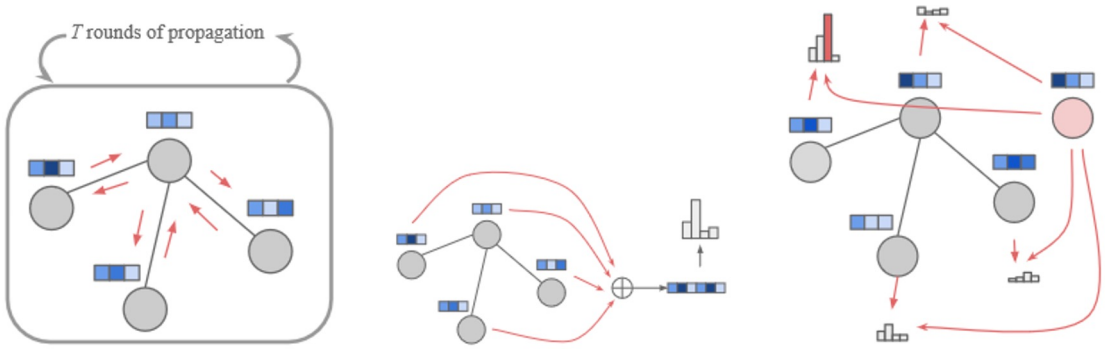
➔ **Autoregressive model** for graph generation: No prior structural assumption

➔ Generation process based on **sequential decisions**

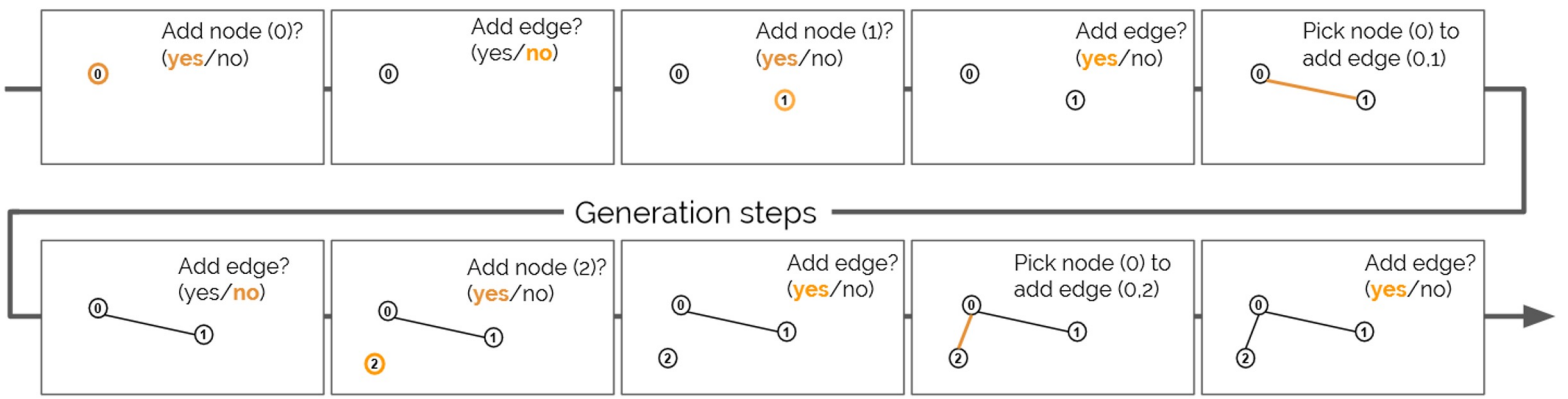
- ◆ Generate **one node at a time**
- ◆ Connect node to graph at current state by creating edges (**one by one**)

➔ Probability of new event depends on **history** of graph derivation

➔ Graphs are modeled by GNN



DeepGMG - Method



1. Check whether to add **a new node of a particular type** or **terminate** ← $f_{\text{add\_node}}$
2. If a node type is chosen, add that type node
3. Check if **any further edges** are needed to connect the new node to the existing graph ← $f_{\text{add\_edge}}$
4. If yes, **select a node** in the graph and **add an edge** connecting the new node to the selected node ← f_{nodes}
5. Return to (3)
6. Repeat **until** the model decides **not to add another edge**
7. Return to (1) to **add subsequent nodes**.

Evaluation for DeepGMG

Synthetic Graphs with Certain Topological Properties

- ➔ Cycles, Trees and Barabasi–Albert graphs (power-law degree distribution)
- ➔ Report portion of valid samples that satisfy the given characteristic
- ➔ Report the KL-divergence between the degree distributions of samples and data for B-A graphs

Molecule Generation

- ➔ **ChEMBL molecule** database (most 20 heavy atoms)
- ➔ RDKit: Convert between SMILES string representations and Graph representation of molecules
- ➔ Node/Edge ordering
  - ◆ Fixed ordering: **canonical from SMILES**
  - ◆ **Uniform random ordering** by permutation
- ➔ Report **Negative Log-Likelihood (NLL)**
- ➔ Report potion of well-formatted (**valid**) samples and unique **novel** samples not seen in training set
- ➔ Report estimated **marginal likelihood** on small molecules (intractable on large molecules)

Dataset	Graph Model	LSTM	E–R Model
Cycles	**84.4%**	48.5%	0.0%
Trees	**96.6%**	30.2%	0.3%
B–A Graphs	**0.0013**	0.0537	0.3715

Table 2. Molecule generation results. N is the number of permutations for each molecule the model is trained on. Typically the number of different SMILES strings for each molecule < 100.

Arch	Grammar	Ordering	N	NLL	%valid	%novel
LSTM	SMILES	Fixed	1	21.48	93.59	81.27
LSTM	SMILES	Random	< 100	**19.99**	93.48	83.95
LSTM	Graph	Fixed	1	22.06	85.16	80.14
LSTM	Graph	Random	$O(n!)$	63.25	91.44	91.26
Graph	Graph	Fixed	1	20.55	**97.52**	90.01
Graph	Graph	Random	$O(n!)$	58.36	95.98	**95.54**

Table 3. Negative log-likelihood evaluation on small molecules with no more than 6 nodes.

Arch	Grammar	Ordering	N	Fixed	Best	Marginal
LSTM	SMILES	Fixed	1	17.28	15.98	15.90
LSTM	SMILES	Random	< 100	**15.95**	15.76	15.67
LSTM	Graph	Fixed	1	16.79	16.35	16.26
LSTM	Graph	Random	$O(n!)$	20.57	18.90	15.96
Graph	Graph	Fixed	1	16.19	**15.75**	15.64
Graph	Graph	Random	$O(n!)$	20.18	18.56	**15.32**

Evaluation for DeepGMG

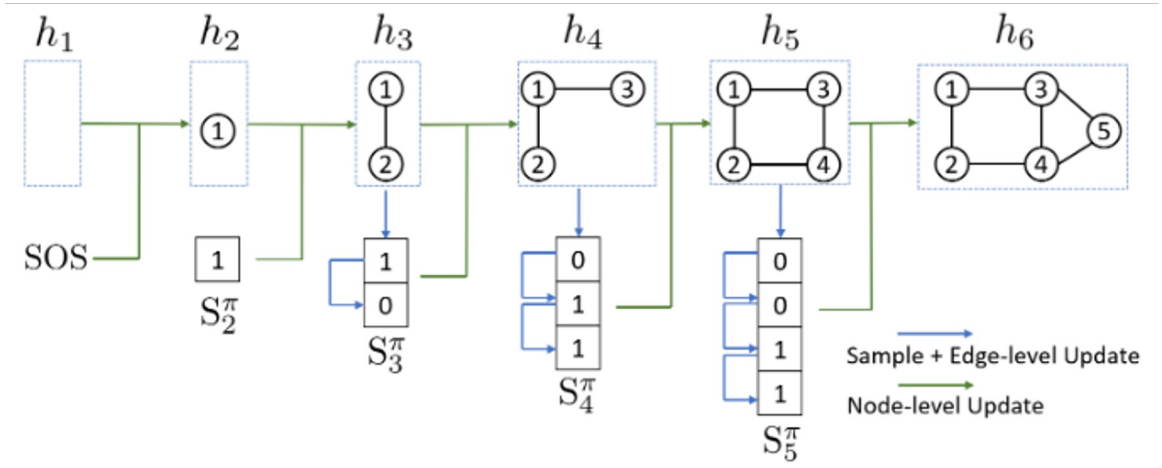
Conditional Graph Generation

- ➔ A subset of previous ChEMBL database with contains molecules of 0, 1 and 3 aromatic rings
- ➔ Report potion of well-formatted (**valid**) samples and unique **novel** samples not seen in training set
- ➔ Report portion of samples that have the number of atoms, bonds, rings, and all three that **match the given condition**

Arch	Grammar	Condition	Valid	Novel	Atom	Bond	Ring	All
LSTM	SMILES	Training	84.3	82.8	71.3	70.9	**82.7**	**69.8**
LSTM	Graph	Training	65.6	64.9	63.3	62.7	50.3	48.2
Graph	Graph	Training	**93.1**	**92.1**	**81.7**	**79.6**	76.4	66.3
LSTM	SMILES	2-rings	64.4	61.2	7.1	4.2	43.8	0.5
LSTM	Graph	2-rings	54.9	54.2	23.5	21.7	23.9	9.8
Graph	Graph	2-rings	**91.5**	**91.3**	**75.8**	**72.4**	**62.1**	**50.2**
LSTM	SMILES	4-rings	71.7	69.4	46.5	3.7	1.3	0.0
LSTM	Graph	4-rings	42.9	42.1	16.4	10.1	3.4	1.8
Graph	Graph	4-rings	**84.8**	**84.0**	**48.7**	**40.9**	**17.0**	**13.3**

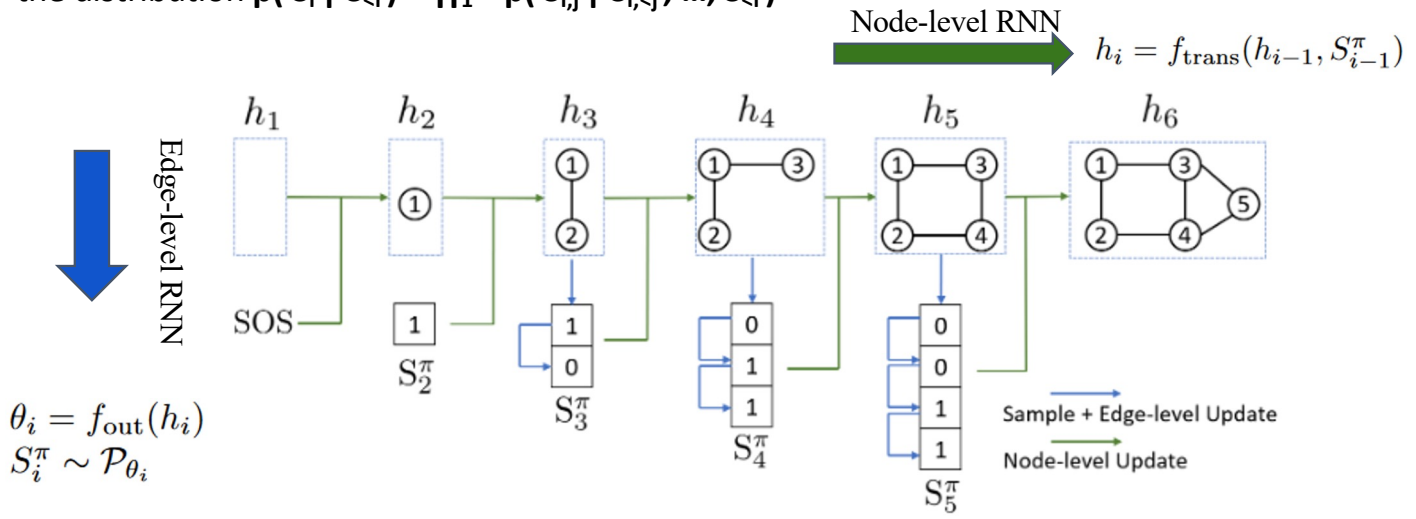
GraphRNN [You et al., 2018]

- **Autoregressive model** for graph generation
  - Key insight: Graph **G** with node permutation $\boldsymbol{\pi}$ can be uniquely mapped into a sequence of node and edge additions $\mathbf{S}^{\boldsymbol{\pi}}$
  - Model the generation process with two RNNs
    - Node-level: generate a state for a new node
    - Edge-level: generate edges for the new node based its hidden state



GraphRNN - Method

- ➔ Omitting symbol of node permutation **π**, Graph **G~p(G)** can be represented as a sequence of adjacency vectors **(S_1 , …, S_n)**
- ➔ **S_i** is a **(i-1)** dimensional *vector represents edges between **i** and previous nodes*: **$\{0, 1\}^{i-1}$**
- ➔ **p(G)** is related to **p(S)** with **$p(S) = \prod_1^{n+1} p(S_i \mid S_1, \ldots, S_{i-1})$**. This product can be modelled by a RNN (node-level) with state transition possible modelled by another RNN (edge-level), which models the distribution **$p(S_i \mid S_{<i}) = \prod_1^{i-1} p(S_{i,j} \mid S_{i,<j}, \ldots, S_{<i})$**



Evaluation for GraphRNN

Visual comparison

➔ First row: Training set
➔ Third row: Kronecker graph, Mixed-Membership Stochastic Block model and Barabasi–Albert graph

Evaluation for GraphRNN

	Community (160,1945)			Ego (399,1071)			Grid (361,684)			Protein (500,1575)		
	Deg.	Clus.	Orbit	Deg.	Clus.	Orbit	Deg.	Clus.	Orbit	Deg.	Clus.	Orbit
E-R	0.021	1.243	0.049	0.508	1.288	0.232	1.011	0.018	0.900	0.145	1.779	1.135
B-A	0.268	0.322	0.047	0.275	0.973	0.095	1.860	0	0.720	1.401	1.706	0.920
Kronecker	0.259	1.685	0.069	0.108	0.975	0.052	1.074	0.008	0.080	0.084	0.441	0.288
MMSB	0.166	1.59	0.054	0.304	0.245	0.048	1.881	0.131	1.239	0.236	0.495	0.775
GraphRNN-S	0.055	0.016	0.041	0.090	**0.006**	0.043	0.029	10^{-5}	0.011	0.057	**0.102**	**0.037**
GraphRNN	**0.014**	**0.002**	**0.039**	**0.077**	0.316	**0.030**	$\mathbf{10^{-5}}$	**0**	$\mathbf{10^{-4}}$	**0.034**	0.935	0.217

$$\mathrm{MMD}^2(p||q) = \mathbb{E}_{x,y\sim p}[k(x,y)] + \mathbb{E}_{x,y\sim q}[k(x,y)] - 2\mathbb{E}_{x\sim p,y\sim q}[k(x,y)].$$

	Community-small (20,83)					Ego-small (18,69)				
	Degree	Clustering	Orbit	Train NLL	Test NLL	Degree	Clustering	Orbit	Train NLL	Test NLL
GraphVAE	0.35	0.98	0.54	13.55	25.48	0.13	0.17	0.05	12.45	14.28
DeepGMG	0.22	0.95	0.40	106.09	112.19	0.04	0.10	0.02	21.17	22.40
GraphRNN-S	**0.02**	0.15	**0.01**	31.24	35.94	0.002	**0.05**	**0.0009**	8.51	9.88
GraphRNN	0.03	**0.03**	**0.01**	28.95	35.10	**0.0003**	**0.05**	**0.0009**	9.05	10.61

Maximum Mean Discrepancy (MMD)

➔ Compare all moments of the empirical distributions using an exponential kernel with Wasserstein distance

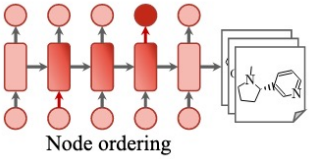
80% decrease of **MMD** over traditional baselines: E-R, B-A, Kronecker, MMSB

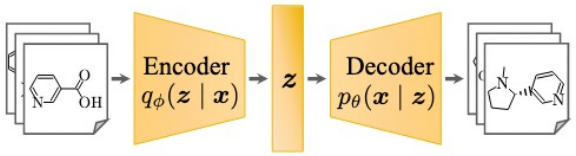
90% decrease of **MMD** over deep learning baselines

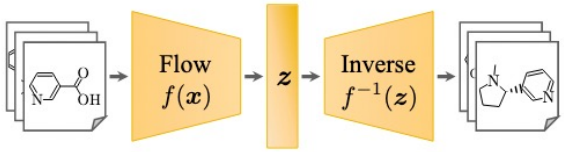
22% smaller average **NLL** gap compared to deep learning baselines

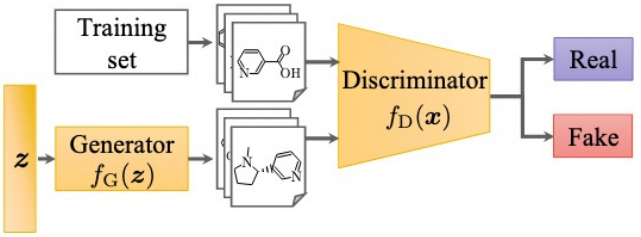
Graph generative models for deep graph generation

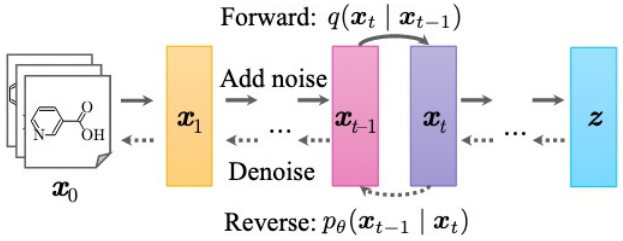
- auto-regressive models
- **variational autoencoders**
- normalizing flows
- generative adversarial networks
- diffusion models





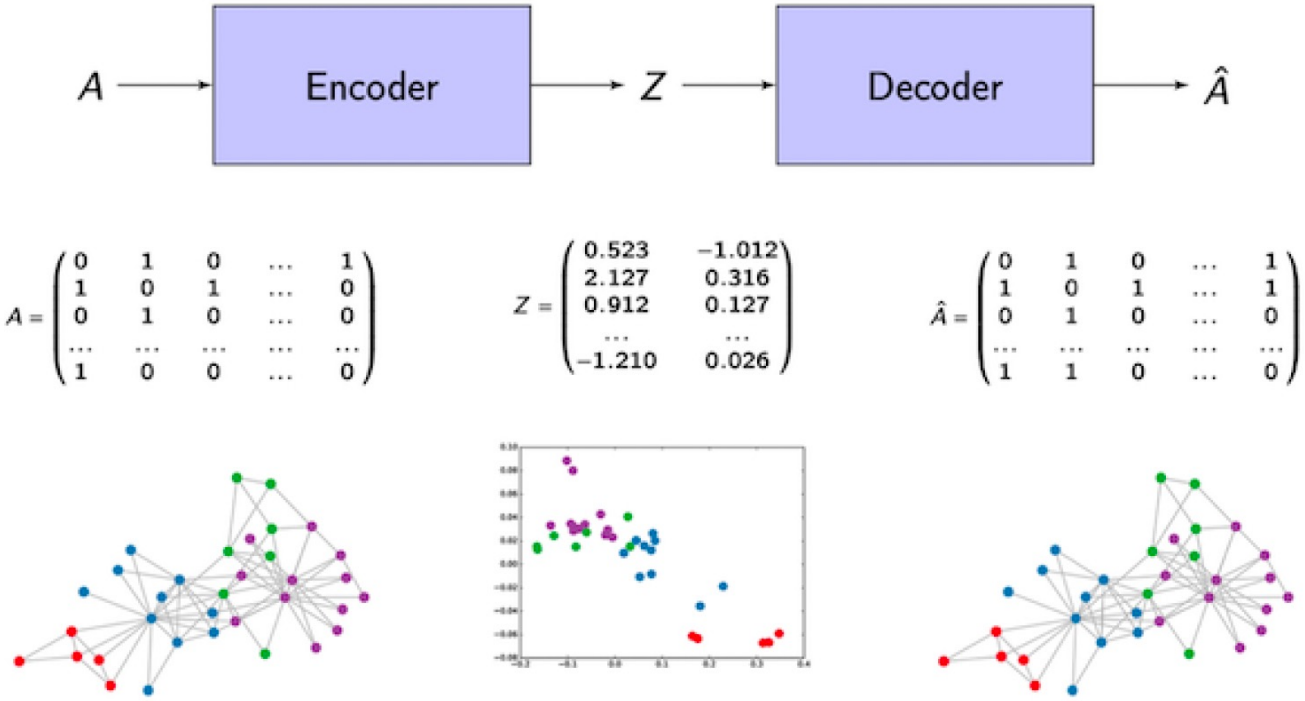




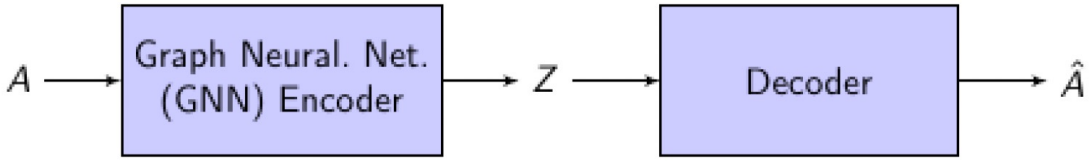


A Survey on Deep Graph Generation: Methods and Applications, Yanqiao Zhu et al, LOG 2023

Graph Auto encoders



Graph Auto encoders



Graph AE:

1. **encoder:** $\boxed{Z = GNN(A, X)}$
2. **decoder:** $\boxed{\hat{A} = \sigma(ZZ^\top)}$
   i.e., for all node pairs (i, j),
   we have $\hat{A}_{ij} = \sigma(z_i^\top z_j)$

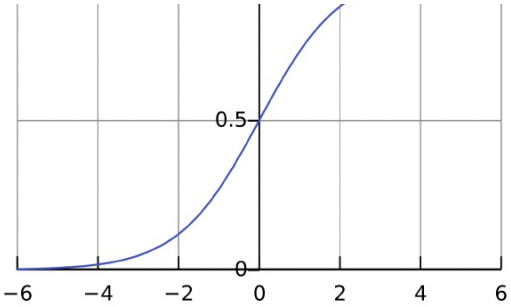
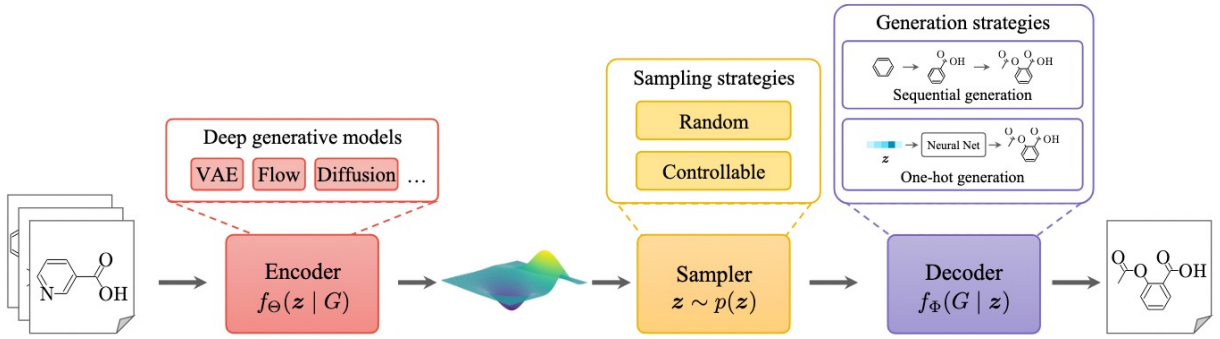


Figure: Sigmoid activation:
$\sigma(x) = \frac{1}{1+e^{-x}}$

Reconstruction Loss[1]**:** capturing the similarity between A and $\hat{A}$

- e.g., **cross-entropy loss**: $-\sum_{i=1}^{n}\sum_{j=1}^{n}(A_{ij}\log(\hat{A}_{ij}) + (1 - A_{ij})\log(1 - \hat{A}_{ij}))$
- or **MSE** loss: $\sum_{i=1}^{n}\sum_{j=1}^{n}(A_{ij} - \hat{A}_{ij})^2$

Graph Variational Auto Encoders



A Survey on Deep Graph Generation: Methods and Applications, Yanqiao Zhu et al, LOG 2023

- VAE[1] estimates the distributions of graphs $p(G)$ by maximizing the Evidence Lower BOund (ELBO):

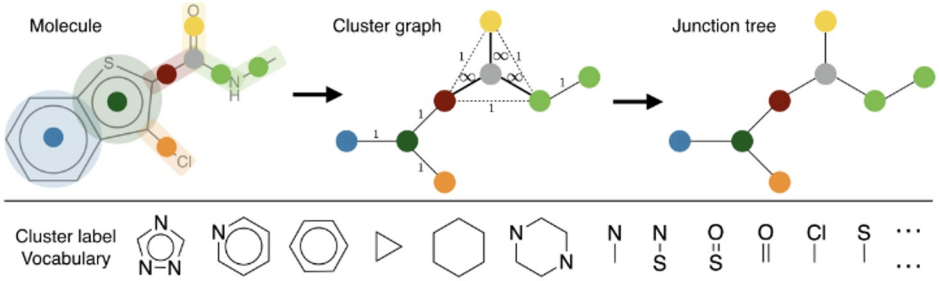
$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{z \sim q_\phi(z|G)} \log(p_\theta(G \mid z)) - D_{\text{KL}}(q_\phi(z \mid G) \parallel p_\theta(z)))$$

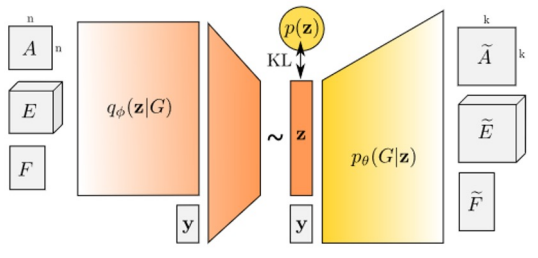
- reconstruction loss between the input G and the reconstructed graph, distance among decoder $q_\phi(z|G)$ and the prior distribution $p_\theta(z)$ – usually Gaussian.
- The encoder $p(z|G)$ and decoder $q(G|z)$ are typically GNNs (i.e. GCN, GAT…).

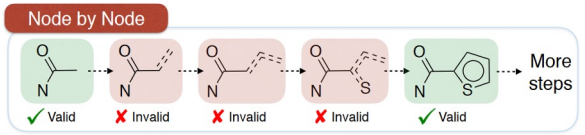
[1] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In ICLR, 2014.

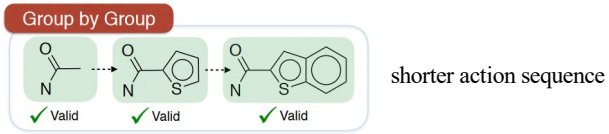
Junction Tree Variational AutoEncoder – JTVAE [Jin et al., 2018]

- ➔ GraphVAE [Simonovsky and Komodakis, 2017]
  - Generate a probabilistic fully-connected graph
    - Model node/edge existence with bernoulli distribution
    - Model node/edge features using multinomial distribution
  - **Loss: Similarity (KL divergence)** between $q_\theta(z|G)$ and prior distribution (normal) $p(z)$ + **similarity (likelihood)** between generated graph $\tilde{G}$ and input graph G .
- ➔ Junction Tree VAE: **molecule generation** leveraging chemical domain knowledge
  - Instead of generating graph **node by node**, generate **group (structure) by group (structure)**: **functional groups** by tree decomposition of molecular graphs
  - Less than 800 groups given 250K molecules





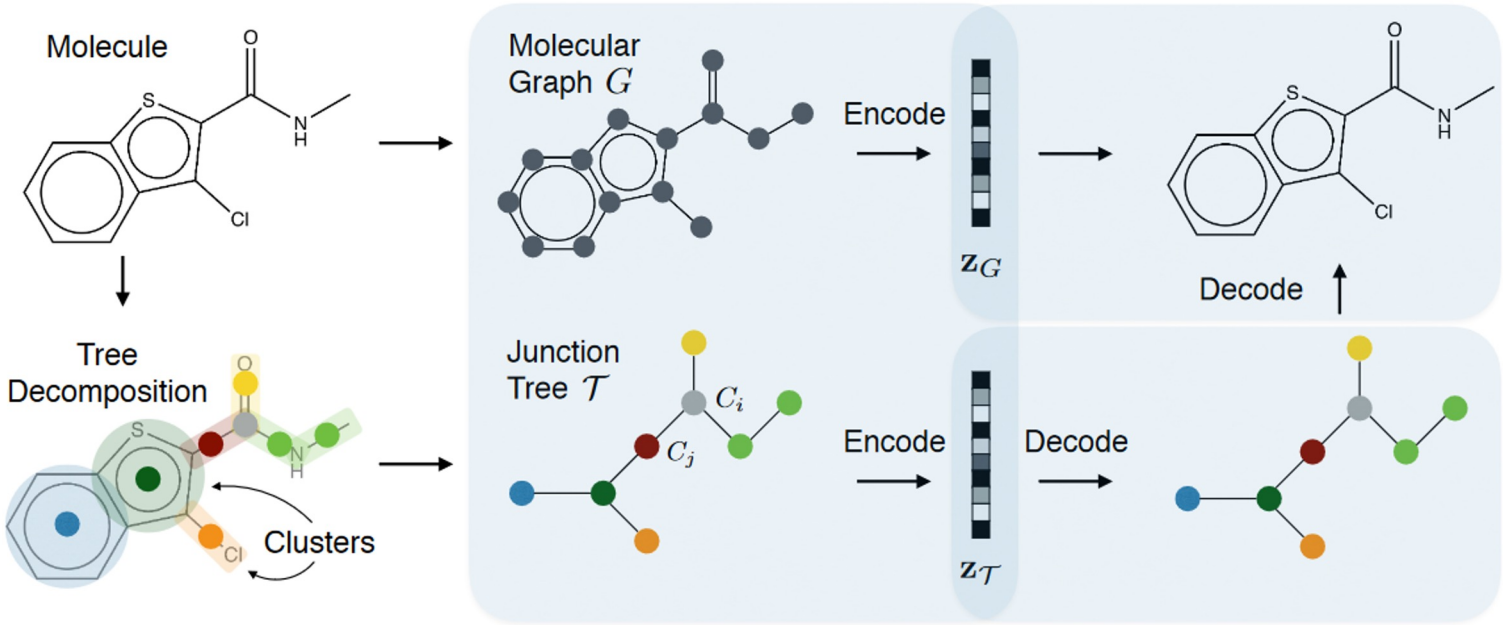




Motivations:

- **Not** every VAE- generated graph is **chemically valid**
- Long action sequence (intermediate states) are **hard to validate** and **difficult to train** (DeepGMG, [Li et al., 2018])

JTVAE - Method



Graph & Tree **encoder**: **graph message passing networks** [Dai et al., 2016]

- Graph encoder output: average pooling
- Tree encoder output: embedding of the root node

Decoder?

Evaluation for JTVAE

➔ **Molecule Reconstruction on ZINC [Sterling and Irwin, 2015]**

- Reconstruct input molecules from latent representations
- 100 monte-carlo trials [Kusner et al., 2017] per molecule
- **Report portion** of decoded molecules **identical to input**

➔ **Molecule Validity**

- Decode valid molecules when sampling from prior distribution
- 100 monte-carlo trials per latent **z** sampled from prior distribution
- **Report portion** of decoded molecules **chemically valid (RDKit)**

Method	Reconstruction	Validity
CVAE	44.6%	0.7%
GVAE	53.7%	7.2%
SD-VAE	76.2%	43.5%
GraphVAE	-	13.5%
JT-VAE (w/o check)	76.4%	93.5%
JT-VAE (full)	**76.7%**	**100.0%**

Neural Graph Generator (NGG)

- a novel graph generative model which leverages latent diffusion for conditional graph generation.
- represents a significant shift from traditional graph generation methods, focusing on *prompting with a vector that includes a set of diverse properties of the graph.*
- introduce a large-scale dataset of synthetic graphs that covers several different types of graphs on which our model was trained. This dataset can be used for pre-training any graph generative model in the future.
- extensively evaluate our model across various graph generation tasks, demonstrating its effectiveness in capturing specific graph properties, generalizing to larger graphs, and generating graphs from subsets of properties.
- release the pre-trained autoencoder, the pre-trained latent diffusion model, and the synthetic dataset of 1M graphs to be useful for both practitioners and the scientific community.

1. Neural Graph Generator: Feature-Conditioned Graph Generation using Latent Diffusion Models, Evdaimon et. al, https://arxiv.org/pdf/2403.01535.pdf

Neural Graph Generator (NGG)

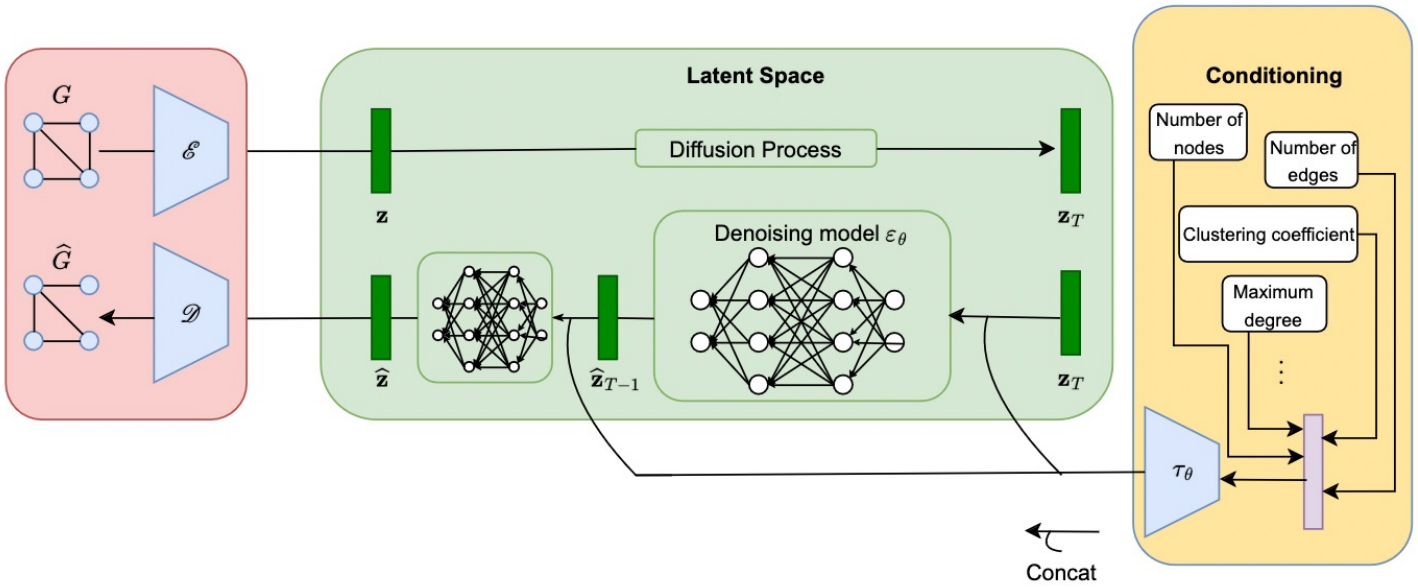
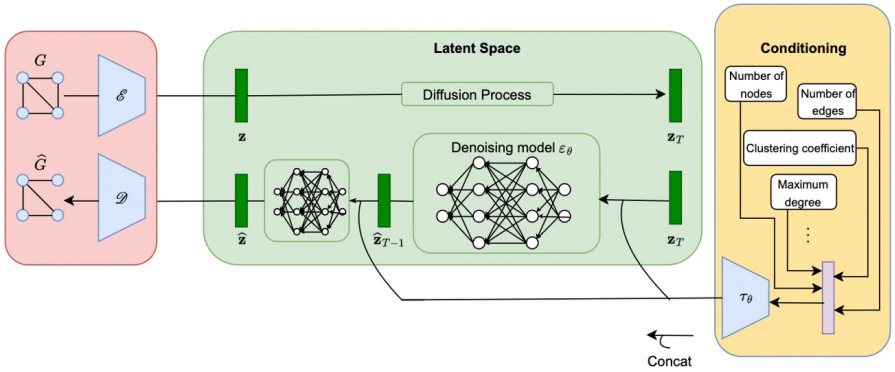


Figure 1: Overview of the proposed architecture. The variational graph autoencoder is responsible for generating a compressed latent representation for each graph. Those representations are fed to the diffusion model which operates in that latent space. The denoising process is conditioned on a vector that contains the graph's properties. The output of the diffusion model is passed on to the decoder which generates a graph.

Neural Graph Generator (NGG) - conditioning

- Dataset: 1M synthetic graphs (<=100 nodes).
- use different types of graph generators.
- 17 families of graphs: (1) paths (2) cycles (3) wheels (4) stars (5) ladders; (6)lollipops (7) Erdos-Renyi random graphs; (8) Newman–Watts–Strogatz small-world graphs; (9) Watts–Strogatz small-world graphs, (10) random d-degree regular graphs (11) Barabasi–Albert graphs; (12) dual Barabasi–Albert graphs (13) extended Barabasi–Albert graphs (14) graphs generated using the Holme and Kim algorithm (15) random lobsters (16) stochastic block model graphs and (17) random partition graphs.
- The generated graphs are devoid of self-loops, isolated nodes, and multigraphs are also excluded.
- NGG model: *3.6M parameters*



Neural Graph Generator (NGG) – exp results

Within distribution performance

Property	VGAE		NGG	
	MAE	SMAPE	MAE	SMAPE
# nodes	24.22	25.18	2.63	3.09
# edges	701.99	66.48	62.33	8.44
Density	0.32	52.13	0.04	7.23
Min. degree	13.95	64.52	11.61	49.46
Max. degree	14.59	22.32	1.59	3.55
Avg. degree	18.99	58.60	1.64	6.68
Assortativity coefficient	0.30	61.32	0.11	39.10
# triangles	10,356.20	99.91	1,026.44	24.38
Avg. # triangles formed by an edge	8.85	66.52	9.44	68.32
Max. # triangles formed by an edge	539.03	83.97	49.26	16.66
Avg. local clustering coefficient	0.29	35.04	0.08	15.42
Global clustering coefficient	0.36	58.12	0.05	14.07
Max. k-core	15.07	54.51	1.66	8.61
# communities	1.74	21.86	0.96	12.34
Diameter	3.73	31.31	2.40	15.96
All	0.80	55.96	0.23	21.05

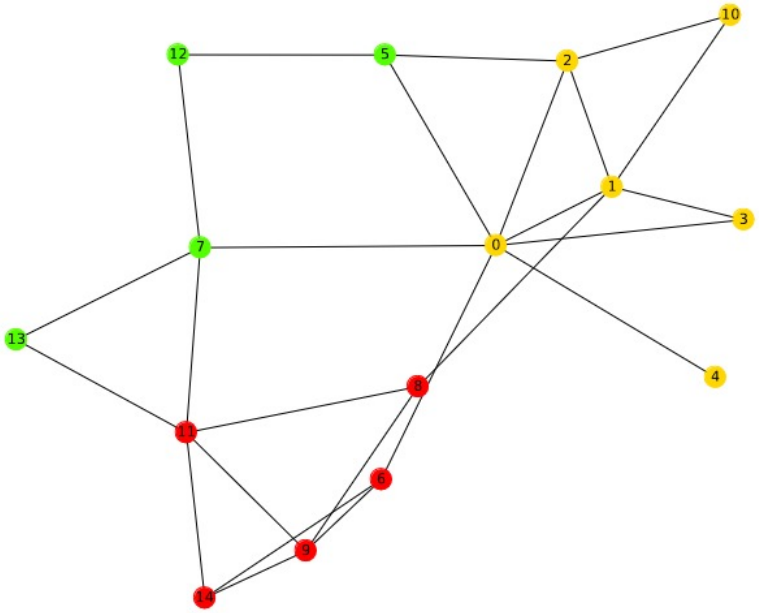
Neural Graph Generator (NGG) – exp results

Performance Comparison of NGG and Baseline Model :

- *Out-of-Distribution Performance* (trained on graphs with up to 50 nodes and evaluated on larger graphs)
- *Within-Distribution Performance with Masking* Applied to some Condition Vector Elements.

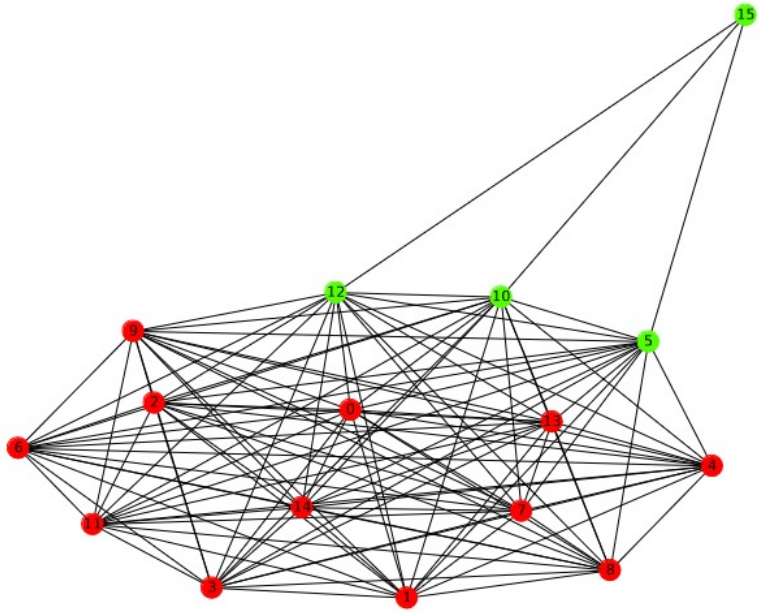
Property	Out of Distribution				Masked			
	VGAE		NGG		VGAE		NGG	
	MAE	SMAPE	MAE	SMAPE	MAE	SMAPE	MAE	SMAPE
# nodes	15.48	18.76	6.68	6.66	25.70	26.05	27.30	27.54
# edges	355.79	60.12	99.09	11.31	785.20	49.83	832.26	52.90
Density	0.28	48.10	0.07	10.74	0.27	34.04	0.29	36.65
Min. degree	12.59	67.12	7.95	41.49	13.96	57.71	13.36	57.50
Max degree	4.46	7.95	2.93	4.64	23.97	35.17	26.04	38.06
Avg. degree	14.34	51.23	3.11	9.27	18.83	40.84	20.17	43.77
Assortativity coefficient	0.29	65.66	0.72	47.57	0.46	60.98	0.43	65.18
# triangles	3,913.93	83.79	1,482.23	27.21	15,101.24	74.92	15,956.96	79.93
Avg . # triangles formed by an edge	7.78	54.99	17.93	74.95	9.11	49.11	17.73	77.18
Max . # triangles formed by an edge	278.36	64.51	83.15	17.71	709.65	66.55	762.29	71.23
Avg. local clustering coefficient	0.26	26.01	0.09	15.91	0.33	40.78	0.36	44.69
Global clustering coefficient	0.27	46.16	0.07	14.44	0.32	41.89	0.34	44.34
Max k-core	11.39	52.05	2.28	9.99	16.93	42.96	17.99	45.76
# communities	2.64	27.47	1.02	12.32	1.86	22.98	1.96	23.40
Diameter	3.52	29.98	2.55	16.61	3.40	23.82	3.31	24.19
All	0.96	56.13	0.54	28.89	0.77	42.14	0.78	42.88

Neural Graph Generator (NGG) – exp results



$$\mathbf{c}_1 = \begin{pmatrix} 15 & 34 & 0.32 & 2 & 8 & 4.5 & -0.046 & 17 & 1.5 & 8 & 0.4 & 0.35 & 4 & 4 & 4 \end{pmatrix}^\top$$

$$\hat{\mathbf{c}}_1 = \begin{pmatrix} 15 & 25 & 0.23 & 1 & 7 & 3.3 & -0.380 & 8 & 0.96 & 3 & 0.43 & 0.32 & 2 & 3 & 4 \end{pmatrix}^\top$$



$$\mathbf{c}_2 = \begin{pmatrix} 15 & 94 & 0.89 & 10 & 14 & 12.5 & -0.149 & 329 & 10.5 & 80 & 0.9 & 0.9 & 10 & 2 & 2 \end{pmatrix}^\top$$

$$\hat{\mathbf{c}}_2 = \begin{pmatrix} 16 & 108 & 0.9 & 3 & 15 & 13.5 & -0.148 & 458 & 12.7 & 93 & 0.97 & 0.97 & 14 & 2 & 2 \end{pmatrix}^\top$$

GraphGPT: Graph Learning with Generative Pre-trained Transformers

- GraphGPT, a novel model for graph learning that leverages self-supervised generative pre-training with transformers.
- **Challenges in Existing Models**:
  - Graph Neural Networks (GNNs) face issues like *over-smoothing* and *over-squashing, limiting scalability.*
  - Current Graph Transformers often require *complex, handcrafted features and may not generalize well across diverse datasets*.

https://arxiv.org/pdf/2401.00529

GraphGPT Key Components

1. Graph-to-Sequence Transformation:

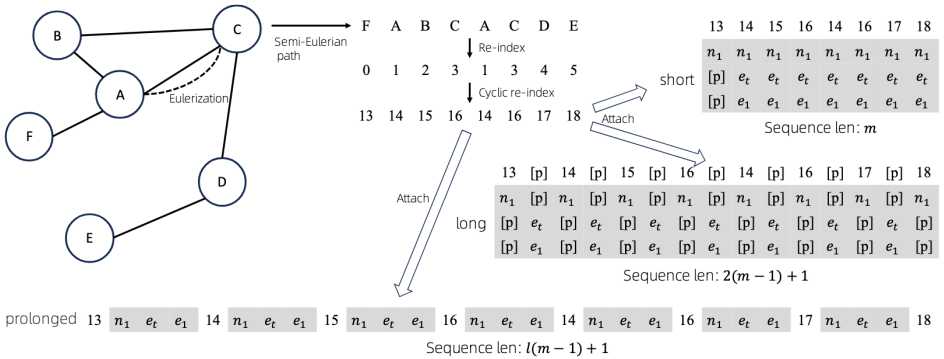
1. Utilizes (semi-)Eulerian paths to convert graphs or subgraphs into reversible sequences of tokens representing nodes, edges, and attributes.
2. An **Eulerian path** is a trail in a graph that traverses every edge exactly once.
   - Complete representation of nodes and edges in the sequence.
   - Bijective mapping between the graph and its serialized form.

2. Generative Pre-Training:

1. Employs a standard transformer decoder trained with the next-token-prediction (NTP) task to learn structural and semantic information without handcrafted features.

3. Supervised Fine-Tuning:

1. After pre-training, the model is fine-tuned for specific graph-related tasks, enhancing its applicability.



Experimental results

Graph regression task for the PCQM4Mv2 dataset.

- > 3.7 million organic molecules
- Nodes represent atoms (9D attributes: atomic number, chirality, etc.), and edges denote chemical bonds (3D attributes: bond type, stereochemistry, conjugation)

Models	MAE ↓ Valid	Test	Params
GCN[1]	0.1379	0.1398	2.0M
GIN[2]	0.1195	0.1218	3.8M
GCN[1]-VN[3]	0.1153	0.1152	4.9M
GIN[2]-VN[3]	0.1083	0.1084	6.7M
TokenGT[4]	0.0910	0.0919	48.5M
Graphformer[5]	0.0864	N/A	48.3M
GPS-Deep[6]	0.0852	0.0862	138.1M
GPS++ (no 3D)[7]	0.0818	N/A	40.0M
GPTrans-L[8]	<u>0.0809</u>	<u>0.0821</u>	86.0M
GraphGPT-B_{12}	<u>0.0807</u>	N/A	113.6M
GraphGPT-B_{24}	**0.0793**	N/A	227.3M
GraphGPT-B_{48}	**0.0792**	**0.0804**	453.4M

Experimental results

Graph classification task results of the graph classification task on the Triangles dataset: counting triangles.
dataset split into:
1). Training/Validation: 30k and 5k small graphs (≤ 25 nodes);
2). Testing: 5k small graphs (Test-small) and 5k large graphs (25–100 nodes, Test-large).
- challenging even for in-distribution (ID) graphs and considerably harder for out-of-distribution (OOD) graphs

Models	Accuracy (%) ↑ T-small	T-large	Params
GIN[1]	$71.53_{\pm 0.94}$	$33.54_{\pm 0.30}$	0.15M
Transformer[2]	$12.08_{\pm 0.31}$	$10.01_{\pm 0.04}$	0.2M
Transformer-LapPE[3]	$78.29_{\pm 0.25}$	$10.64_{\pm 2.94}$	0.2M
Transformer-RWSE[3]	$\mathbf{99.40_{\pm 0.10}}$	$\underline{54.76_{\pm 7.24}}$	0.2M
Graphormer[4]	$\mathbf{99.09_{\pm 0.31}}$	$42.34_{\pm 6.48}$	0.2M
GET-B	$32.60_{\pm 1.86}$	$13.99_{\pm 1.78}$	113.5M
GraphGPT-B[a]	$92.16_{\pm 0.28}$	$26.51_{\pm 1.01}$	113.5M
GraphGPT-B[b]	$81.38_{\pm 0.27}$	$37.68_{\pm 0.99}$	113.5M
GraphGPT-B[c]	$\mathbf{99.08_{\pm 0.14}}$	$38.80_{\pm 3.60}$	113.5M
GraphGPT-B[d]	$90.93_{\pm 0.51}$	$40.79_{\pm 1.40}$	113.5M
GraphGPT-B[e]	$64.28_{\pm 0.33}$	$17.38_{\pm 0.61}$	113.5M
GraphGPT-B[f]	$86.14_{\pm 7.38}$	$26.94_{\pm 4.80}$	113.5M
GraphGPT-B[g]	$86.57_{\pm 2.74}$	$23.45_{\pm 1.44}$	113.5M
GraphGPT-B[a+b]	$84.83_{\pm 0.81}$	$39.62_{\pm 1.84}$	113.5M
GraphGPT-B[a+c]	$\underline{98.68_{\pm 0.18}}$	$50.07_{\pm 3.28}$	113.5M
GraphGPT-B[b+c]	$\underline{98.26_{\pm 0.30}}$	$52.33_{\pm 2.61}$	113.5M
GraphGPT-B[a+b+d]	$89.98_{\pm 0.54}$	$33.45_{\pm 2.51}$	113.5M
GraphGPT-M[a+b+c]	$95.07_{\pm 0.67}$	$51.72_{\pm 1.12}$	33.7M
GraphGPT-B[a+b+c]	$\underline{98.63_{\pm 0.18}}$	$\mathbf{58.96_{\pm 1.90}}$	113.5M

Experimental results

Models	ogbl-ppa HR@100 (%) ↑	ogbl-citation2 MRR (%) ↑
Common Neighbor	$27.65_{\pm 0.00}$	$51.47_{\pm 0.00}$
Adamic Adar	$32.45_{\pm 0.00}$	$51.89_{\pm 0.00}$
Resource Allocation[1]	$49.33_{\pm 0.00}$	$51.98_{\pm 0.00}$
Node2Vec[2]	$22.26_{\pm 0.83}$	$61.41_{\pm 0.11}$
Matrix Factorization[3]	$32.29_{\pm 0.94}$	$51.86_{\pm 4.43}$
GCN[4]	$18.67_{\pm 1.32}$	$84.74_{\pm 0.21}$
GraphSAGE[5]	$16.55_{\pm 2.40}$	$82.60_{\pm 0.36}$
SEAL[6]	$48.80_{\pm 3.16}$	$87.67_{\pm 0.32}$
AGDN[7]	$41.23_{\pm 1.59}$	$85.49_{\pm 0.29}$
SIEG[8]	$63.22_{\pm 1.74}$	$90.18_{\pm 0.15}$
MPLP[9]	$65.24_{\pm 1.50}$	$90.72_{\pm 0.12}$
RefinedGAE[10]	$\underline{73.74_{\pm 0.92}}$	$84.55_{\pm 0.15}$
GraphGPT-M	$65.44_{\pm 0.43}$	$\underline{92.82_{\pm 0.27}}$
GraphGPT-B	$68.76_{\pm 0.67}$	$\mathbf{93.05_{\pm 0.20}}$
GraphGPT-XXL	$\mathbf{76.55_{\pm 0.67}}$	N/A

Link prediction task gbl-ppa and ogbl-citation2 datasets.

Models	ogbn-proteins ROC-AUC (%) ↑	ogbn-arxiv Accuracy (%) ↑
GCN[1,2]	$77.29_{\pm 0.46}$	$\underline{73.53_{\pm 0.12}}$
GraphSAGE[1,3]	$82.21_{\pm 0.32}$	$73.00_{\pm 0.28}$
GAT[1,4]	$\underline{85.01_{\pm 0.46}}$	$\underline{73.30_{\pm 0.18}}$
DRGAT[5]	N/A	$\mathbf{74.16_{\pm 0.07}}$
AGDN[6]	$\mathbf{88.65_{\pm 0.13}}$	$\underline{73.41_{\pm 0.25}}$
DeeperGCN[7]	$\underline{85.80_{\pm 0.17}}$	$71.92_{\pm 0.16}$
GraphGPS[1,8]	$77.15_{\pm 0.64}$	$71.23_{\pm 0.59}$
NAGphormer[1,9]	$72.17_{\pm 0.45}$	$70.88_{\pm 0.24}$
Exphormer[1,10]	$77.62_{\pm 0.33}$	$72.32_{\pm 0.36}$
GOAT[1,11]	$79.31_{\pm 0.42}$	$72.76_{\pm 0.29}$
NodeFormer[1,12]	$77.86_{\pm 0.84}$	$67.78_{\pm 0.28}$
SGFormer[1,13]	$79.92_{\pm 0.48}$	$72.76_{\pm 0.33}$
Polynormer[1,14]	$79.53_{\pm 0.67}$	$\underline{73.40_{\pm 0.22}}$
GraphGPT-S	$83.56_{\pm 0.16}$	$70.83_{\pm 0.33}$
GraphGPT-M	$84.02_{\pm 0.21}$	$71.20_{\pm 0.34}$
GraphGPT-B	$\underline{85.33_{\pm 0.10}}$	$72.10_{\pm 0.30}$

Results of the node classification task on the ogbn-proteins and ogbn-arxiv datasets

Deep Graph Generators - Summary

Graph generation remains a difficult problem!

➔ Challenges
- **Large and variable output space**
  - Quadratic complexity of adjacency matrix; Varying node/edge size of graphs
- **Non-unique representation**
  - Permutation-invariant node ordering
  - Extra matching step may be needed to construct proper objective function
- **Complex and non-local dependencies**, i.e., long-range dependency on edge formation

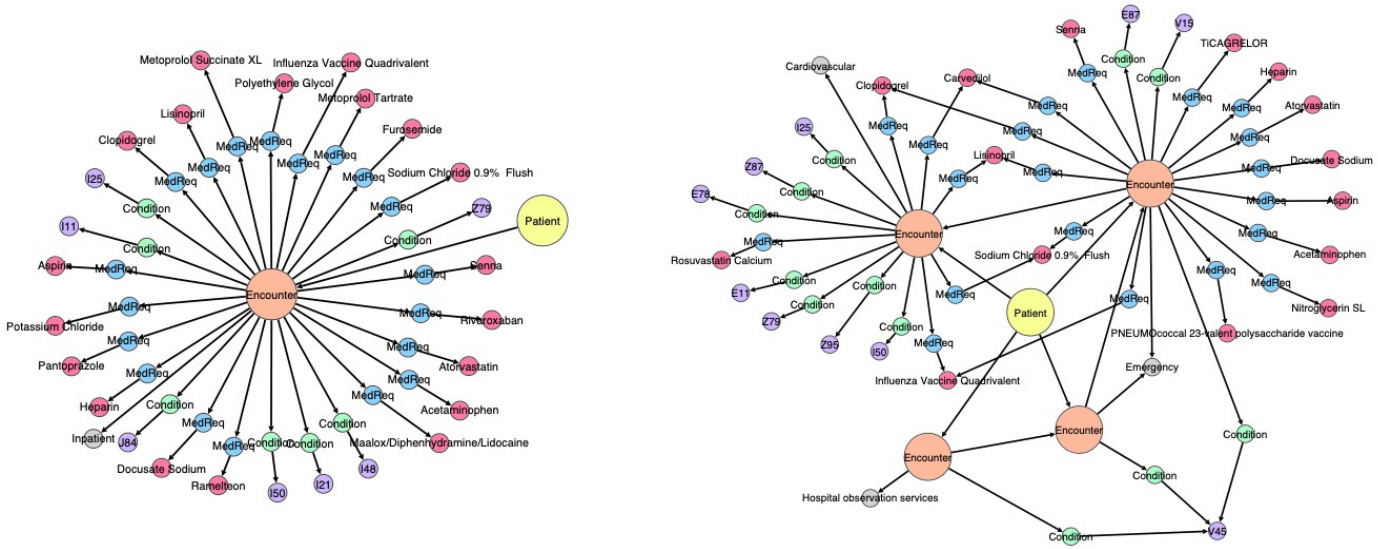
➔ Benefits
- Gain insights for real-world graphs; Predict their evolution; Conduct simulations; Detect anomalies

➔ Real-World Applications
- Drug discovery
- Design electronic circuits (VLSI)
- Social networks simulation
- Energy/Telecom Networks
- Medical Data…

GNNs and Graph Generative models for biomedical applications

- Graph Generative models
- **Generative models for Medical Graphs**
- Multi modality for Graph generators – protein function text generator
- Conclusions

GVAE for Generating Synthetic Patient Trajectories



Synthetic electronic health records generated with variational graph autoencoders,
G. Nikolentzos, M. Vazirgiannis, C. Xypolopoulos, M. Lingman, E. G. Brandt NATURE DIGITAL MEDICINE 2023,
https://www.nature.com/articles/s41746-023-00822-x

Health data - constraints

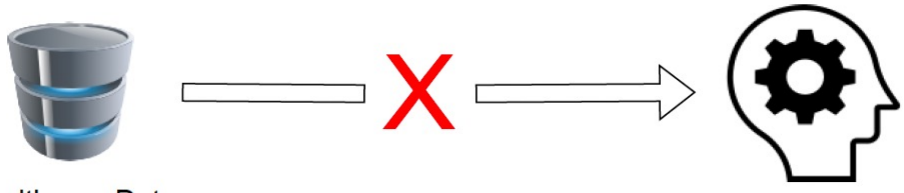
In many fields, open access data sets lead to significant progress, e. g.,

- Computer vision $\rightarrow$ Imagenet
- Natural language processing $\rightarrow$ Wordnet

However, in the case of healthcare data:

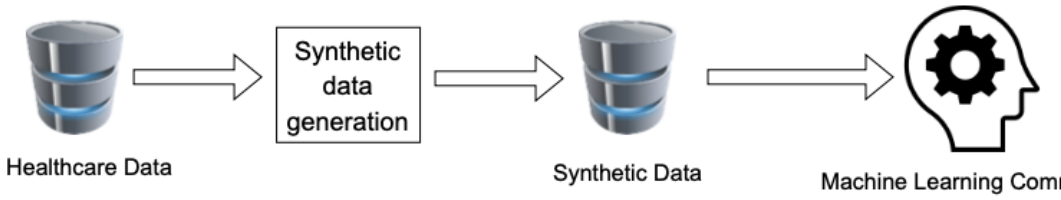
- Lack of high-quality healthcare data
- Strict regulations for data access:
  - The use of patient data leads to privacy concerns
  - Regulations like HIPAA prohibit the unauthorized use and disclosure of protected health information
  - Patient data cannot be freely shared

The above impedes machine learning research in healthcare!

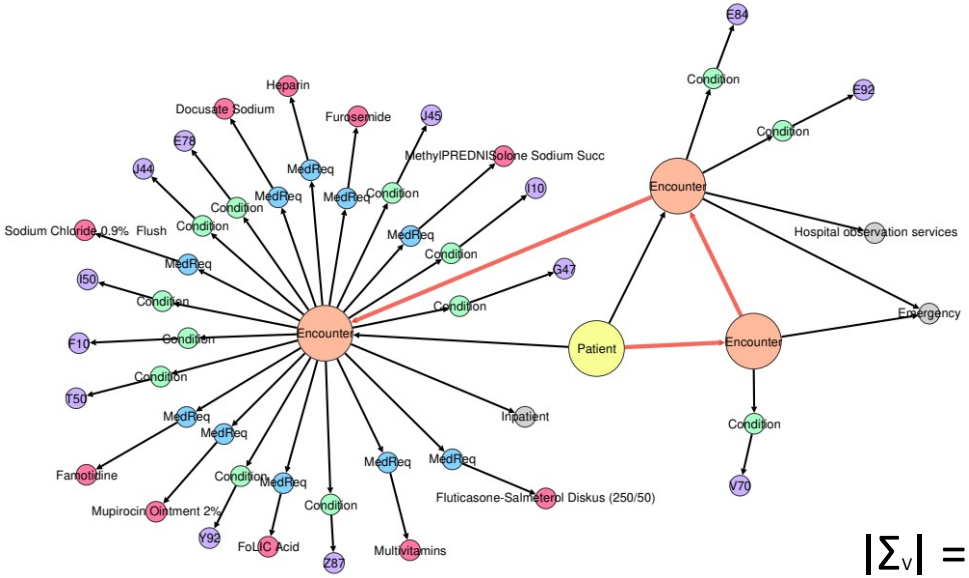


Synthetic Data Generation

- Approaches that generate synthetic data could help the research community deal with those challenges
- Neural networks are recently used to generate synthetic data
  ↪ Real data is fed into the model which learns to produce synthetic data that very closely resembles the real dataset
- The neural network model is designed to produce synthetic data that does not violate data privacy regulations
- Once the synthetic dataset has been created
  - it can be used to train machine learning models (develop analytics, data augmentation, increase robustness of models)
  - it can be shared across research teams and institutions facilitating reproducibility of different models and collaboration

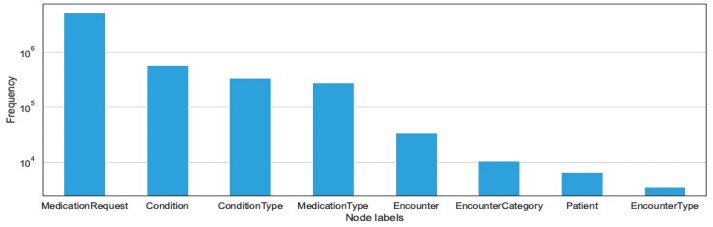


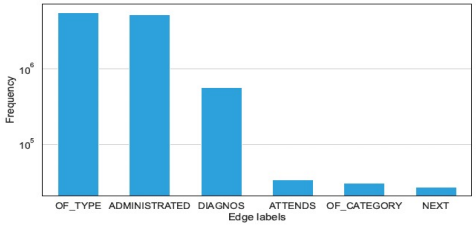
Shaarpeec graph model for patient trajectories



@SHAARPEC graph data model
→ patient trajectories represented as graphs
https://shaarpec.com/

$|\Sigma_V|$ = 13,980 node labels
$|\Sigma_E|$=6 edge labels.



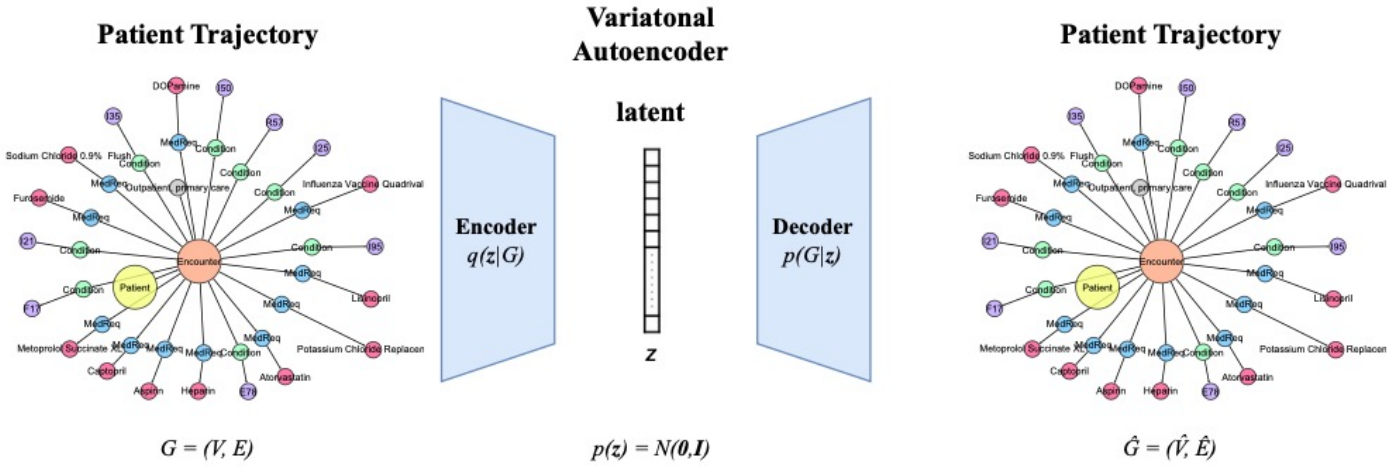


GVAE architecture

We use a variational autoencoder:

- The encoder maps input DAGs into d-dimensional gaussian distributions

- The decoder reconstructs the input DAGS given vectors sampled from the gaussian distributions

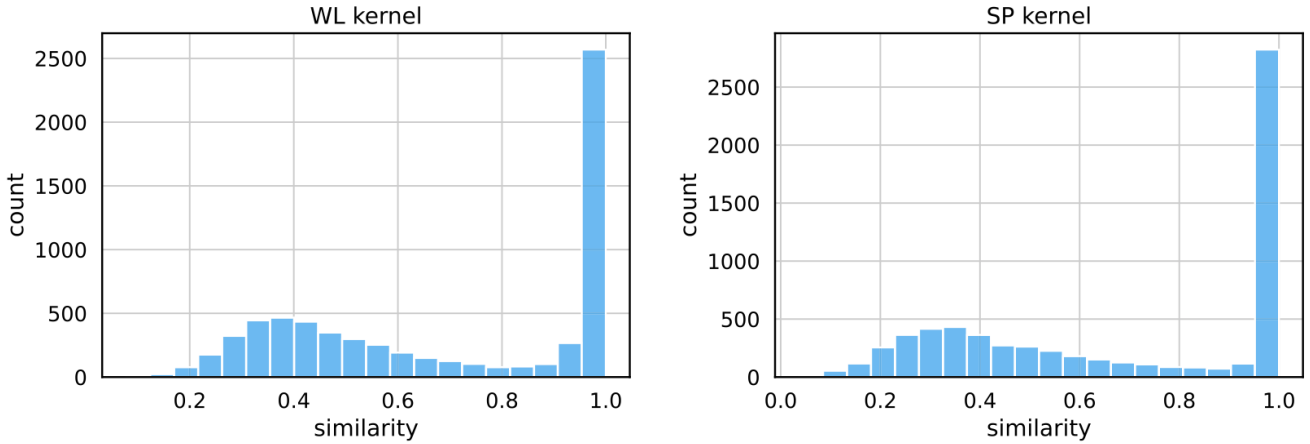


MIMIC dataset

Statistics on the patient trajectories calculated from the *atrial fibrillation* cohort from the MIMIC-IV database.

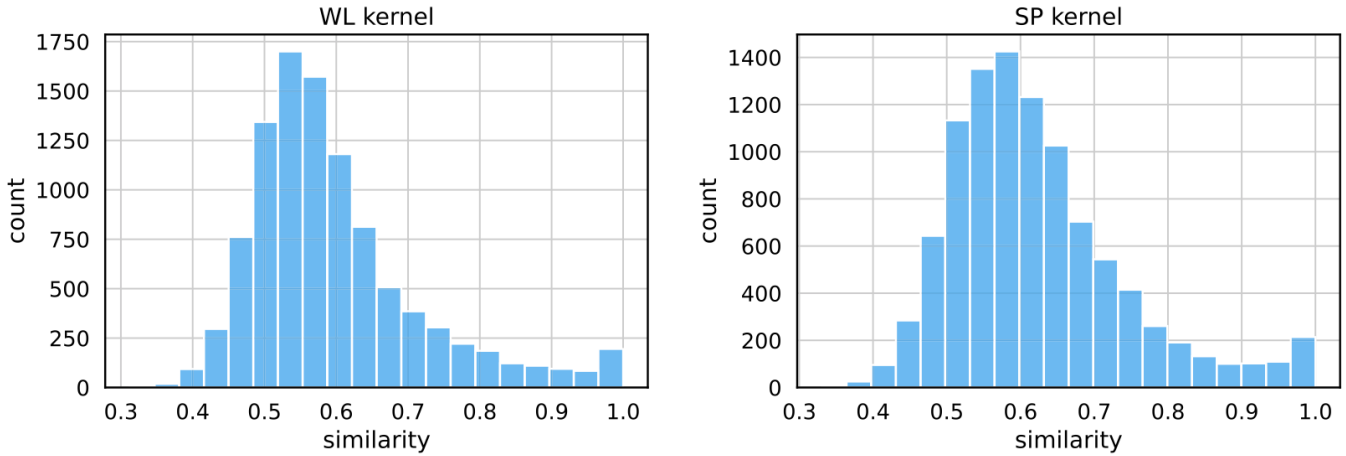
	Raw	**After Pre-processing**
Max # nodes	18,947	2,772
Min # nodes	10	10
Average # nodes	1,044.1	221.2
Max # edges	36,811	5,162
Min # edges	9	9
Average # edges	1,867.3	294.7
# node labels ($\|\Sigma_V\|$)	13,980	944
# edge labels ($\|\Sigma_E\|$)	6	6
# graphs	6,535	6,535

Quality of graph reconstruction



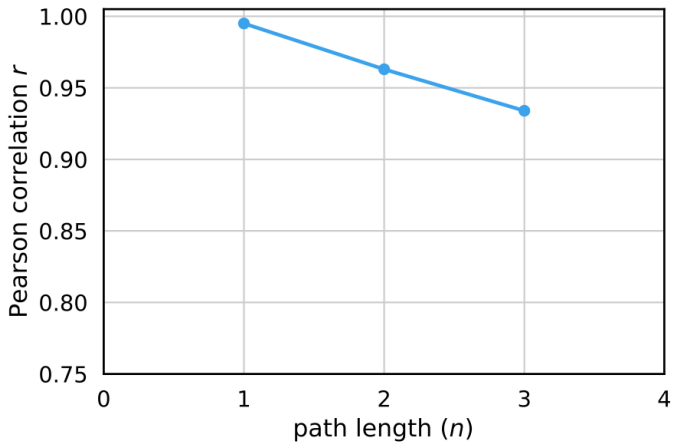
- Real graph input to encoder - decoder produces a reconstructed version
- Histogram of similarities between input graphs and reconstructed graphs using the Weisfeiler Lehman subtree (WL) kernel and the shortest path (SP) kernel.

Graph generation – similarity to real graphs

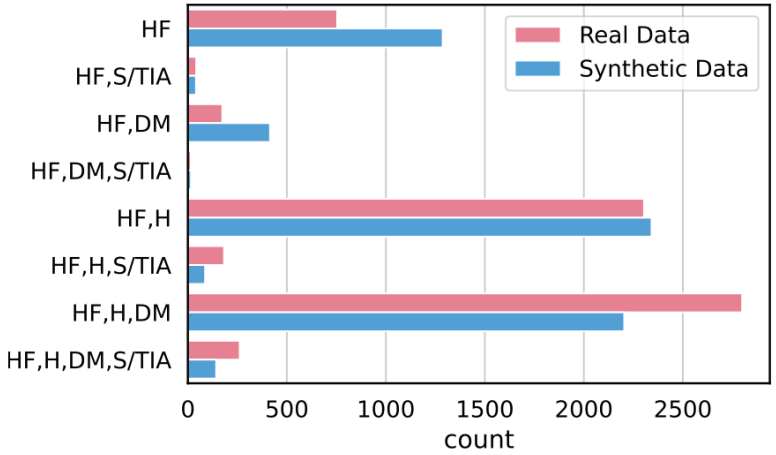


- Generated graph: Sample from the learned distribution and feed the decoder
- Similarity histogram between input and generated graphs using the Weisfeiler-Lehman subtree (WL) kernel and the shortest path (SP) kernel.
- lower similarity => privacy

Graph generation – similarity to real graphs

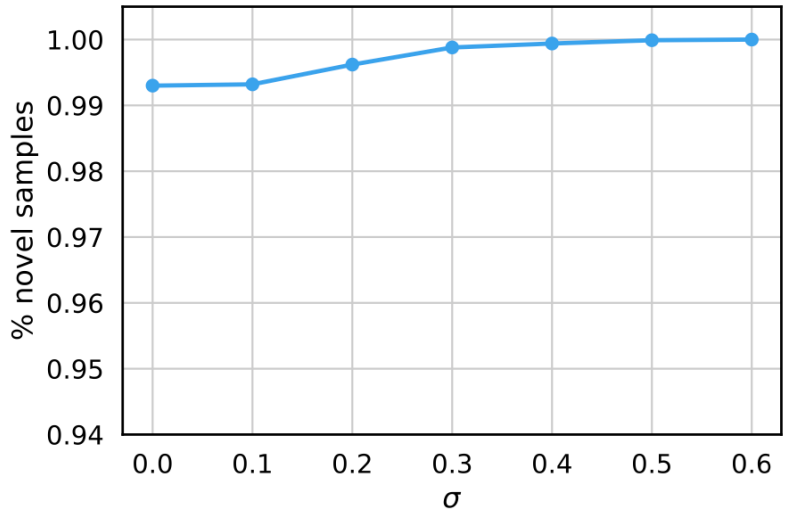


- Pearson's r between frequency of different structures in training trajectories and generated trajectories.
- 1-paths (node labels), 2-paths, and 3-paths,



- Comorbidities for real and synthetic atrial fibrillation cohorts.
- comorbidities: HF heart failure, S/TIA stroke/TIA, DM diabetes mellitus, H hypertension.

Privacy concerns



- % novel trajectories vs standard deviation of the Gaussian noise.
- mean Gaussian = **0** vector.
- noise added to Encountes.
- σ ∈ {0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6}, 20000 trajectories generated, compared to real

Table 3. Classification accuracy of the two experimental scenarios in two separate downstream analytics tasks.

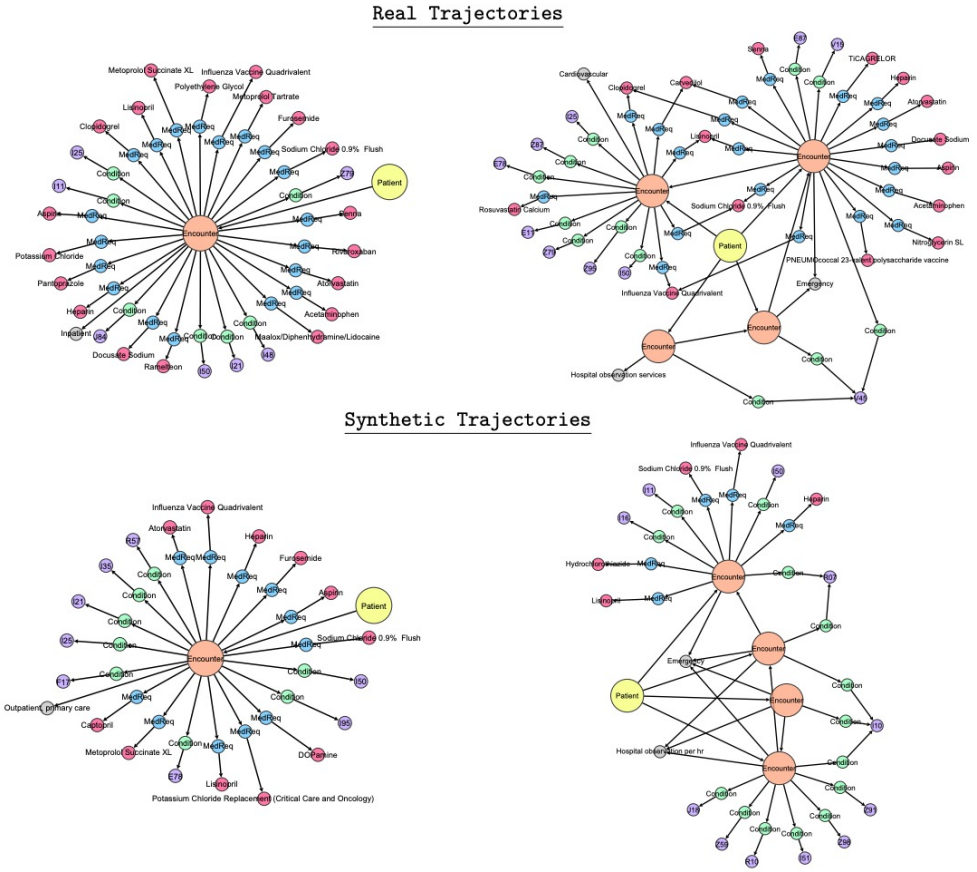
Scenario	Task 1	Task 2
Train on real, test on real	64.02% ± 2.93	73.04% ± 1.35
Train on synthetic, test on real	63.85% ± 2.57	73.32% ± 2.90

Both tasks are variants of predicting the onset of heart failure in patients. In the first scenario, the classifier is trained on real data and is evaluated on real data, while in the second scenario, the classifier is trained on synthetic data and is evaluated on real data.

- Real data are barely distinguishable by the synthetic ones.

GVAEs for medical graph generation - Conclusions

- the model generates novel synthetic patient trajectories,
- sufficiently different to preserve patient privacy,
- yet retains the characteristics of the real-world data. Thus data exchange and ML is feasible
- most significant feature: capacity to learn long-range correlations between trajectory nodes.



GNNs and Graph Generative models for biomedical applications

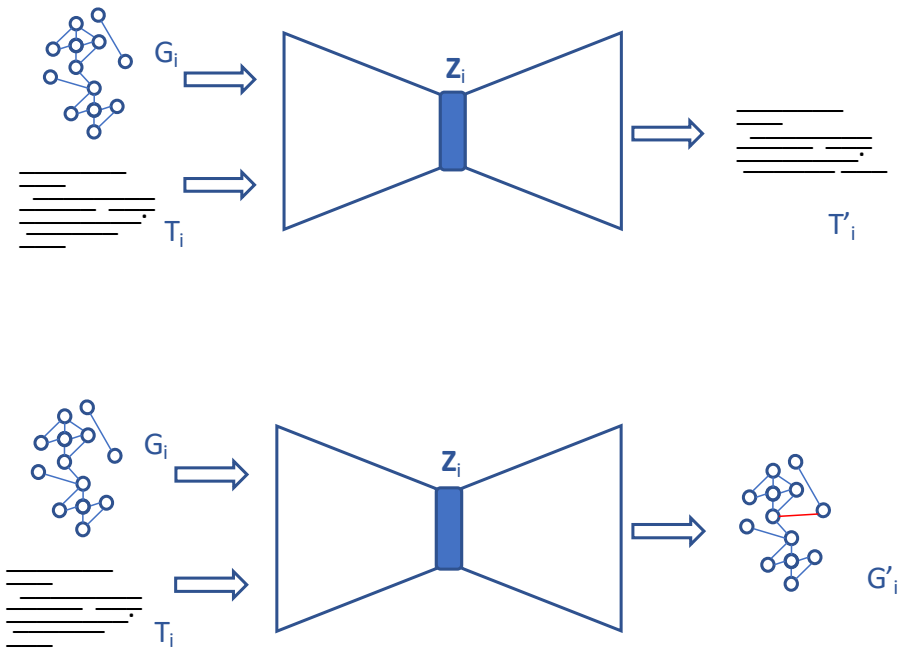
- Graph Generative models
- Generative models for Medical Graphs
- **Multi modality for Graph generators -** protein function text generator
- Conclusions

Multimodal graph pretrained models

- Modality m_i in {text, image, sound, …}
- Pretrain {m_i}+graph => graph/ m_i

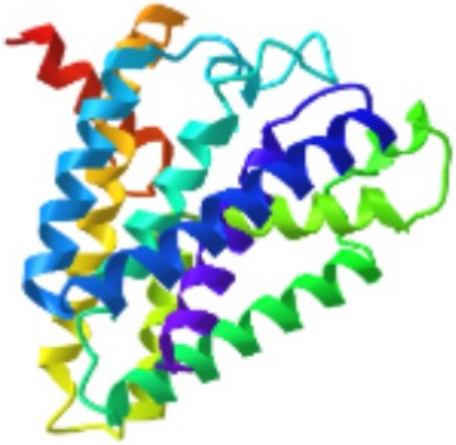
Recent efforts:

- Multimodal learning with graphs [Ektefaie et al, 2023] – *guidelines and applications*
- Investigating Pretrained Language Models for Graph-to-Text Generation[Ribeiro et al, 2021]
- Structural Information Preserving for Graph-to-Text Generation[Song et al, 2021] – *linearised graph; transformer encoder decoder – no graph encoding/generation*
- ProtNLM: Model-based Natural Language Protein [Gane et al, 2022] – *does merely classification*



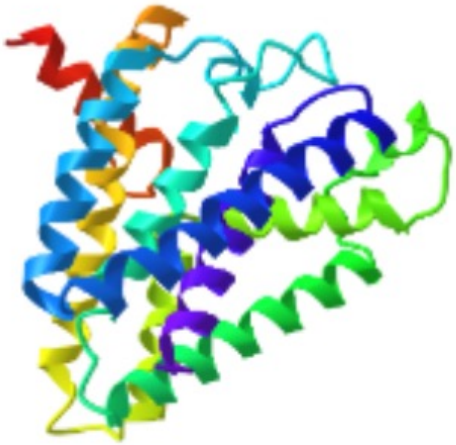
Prot2Text: Multimodal Protein's Function Generation with GNNs and Transformer

- Understanding proteins' function is a central challenge in the field of biological sciences
- essential for various applications: drug discovery, enabling identify and target specific proteins that play critical roles in disease pathways.
- Traditionally, proteins' functions prediction is assigning predefined labels based on their characteristics [Kulmanov and Hoehndorf 2019].
- However, this oversimplifies the complexity of proteins' functionality, limiting the depth of our understanding.
- To address this limitation, *we propose a novel method that given an unknown protein produces a free text predicting its function*



Prot2Text: Multimodal Protein's Function Generation with GNNs and Transformers, H. Abdine, M. Chatzianastasis, C. Bouyioukos, M. Vazirgiannis, AAAI 2024

Prot2Text: Multimodal Protein's Function Generation with GNNs and Transformer

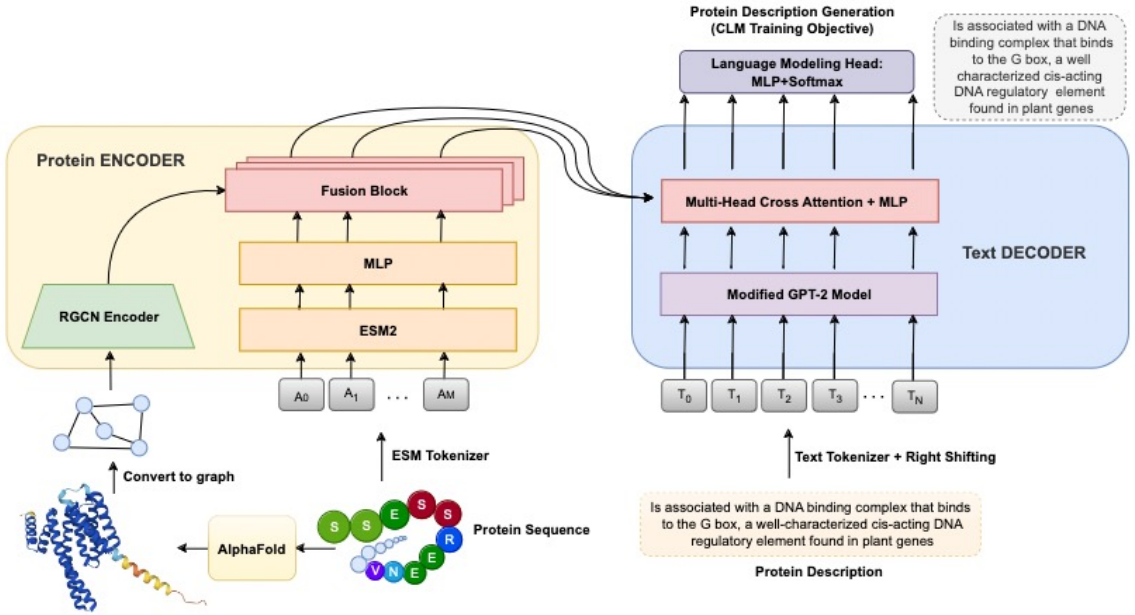


Contributions

- develop a novel *multimodal framework*, that generate detailed and accurate descriptions of proteins' functions in free text
- first time - integrate GNNs and Large Language Models (LLMs), to encompass both structural (protein's 3D structure) and sequential (amino acid's sequence) information
- propose various baselines for protein text generation
- demonstrate *integration of both graph and sequence protein information leads to better generation capabilities*.
- release a comprehensive multimodal protein dataset - 256, 690 protein structures, sequences, and textual function descriptions.

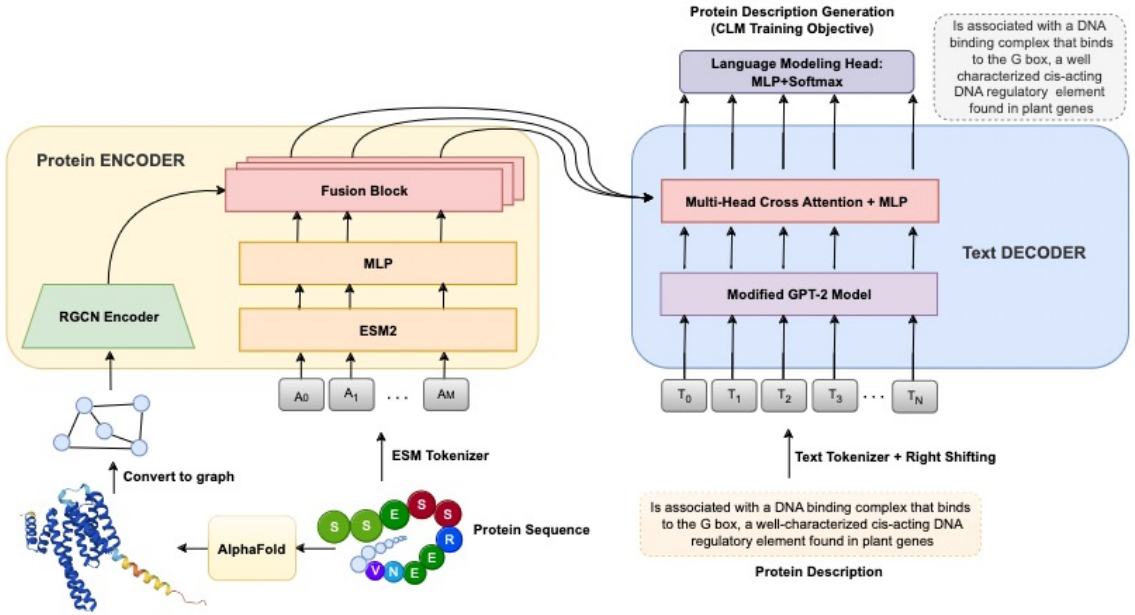
Prot2Text: Multimodal Protein's Function Generation with GNNs and Transformer

- Encoder- Decoder framework forms the backbone of the model
- encoder component
  - Relational graph convolution network (RGCN) [Schlichtkrullet al. 2018] to process the protei graphs,
  - ESM protein language model (Lin et al. 2023a) to encode the protein's sequence.
- cross-attention mechanism facilitates exchange of relevant information between the graph-encoded a the sequence-encoded vectors => a fused representation synthesizing structural and textual aspects.
- Decoder component: pre-trained GPT-2 model generates detailed and accurate protein descriptions from the fused protein representation.



Prot2Text: Multimodal Protein's Function Generation with GNNs and Transformer

- **Graph Construction** - obtaining the 3D proteins' from AlphaFold
- Protein graph $G = (V, E, R)$, where
  - $V = [N] := \{1, ..., N\}$ is the set of nodes/amino-acids of the proteins,
  - $E \subseteq V \times V$ is the set of edges/interactions between the nodes
  - R: set of different edge interactions.
  - Each node u is associated with a feature vector $x_u \in R^d$, with attributes: local structural features, physico-chemical properties of amino-acids.



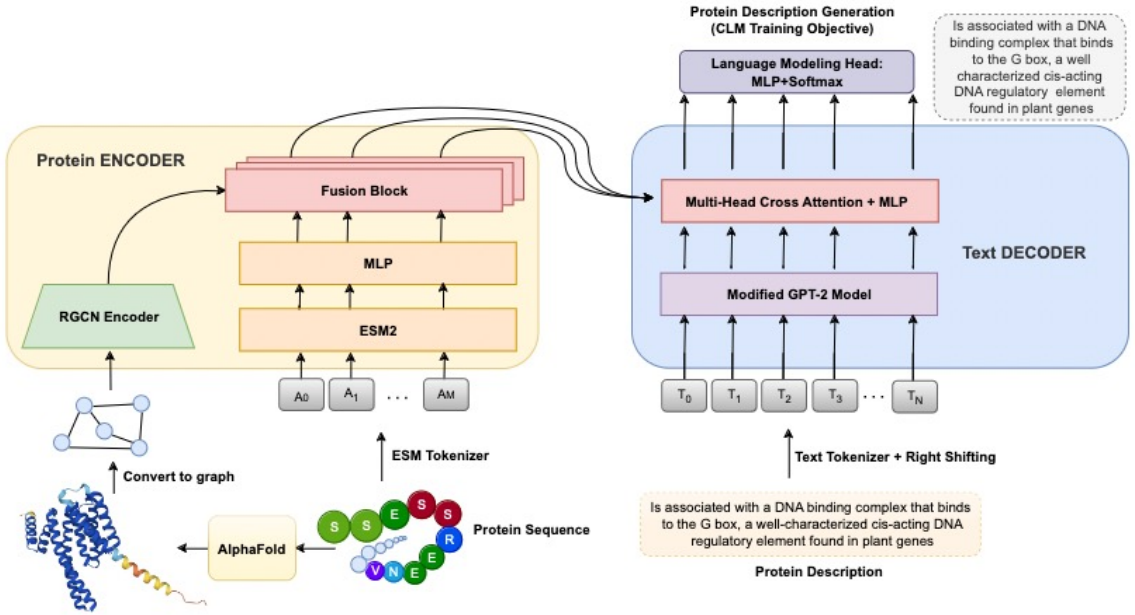
Prot2Text: Multimodal Protein's Function Generation with GNNs and Transformer

- **Graph Construction** – 3D graphs from AlphaFold
- Protein graph $G = (V, E, R)$, where
  - $V = [N] := \{1, ..., N\}$ is the set of nodes/amino-acids of the proteins,
  - $E \subseteq V \times V$ is the set of edges/interactions between the nodes
  - R: set of different edge interactions.
  - Each node u is associated with a feature vector $x_u \in R^d$, with attributes: local structural features, physico-chemical properties of amino-acids.
- **Graph Encoding**. employ a RGCN: effectively treats edge types in the message-passing mechanism.
- In layer k of the GNN, update node representations

$$\boldsymbol{x}_i^k = \sigma\left(\boldsymbol{W}_{\text{root}}^k \cdot \boldsymbol{x}_i^{k-1} + \sum_{r\in\mathcal{R}} \sum_{j\in\mathcal{N}_r(i)} \frac{1}{|\mathcal{N}_r(i)|} \boldsymbol{W}_r^k \cdot \boldsymbol{x}_j^{k-1}\right)$$

- Final graph representation:

$$\boldsymbol{h}_G = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i^K$$

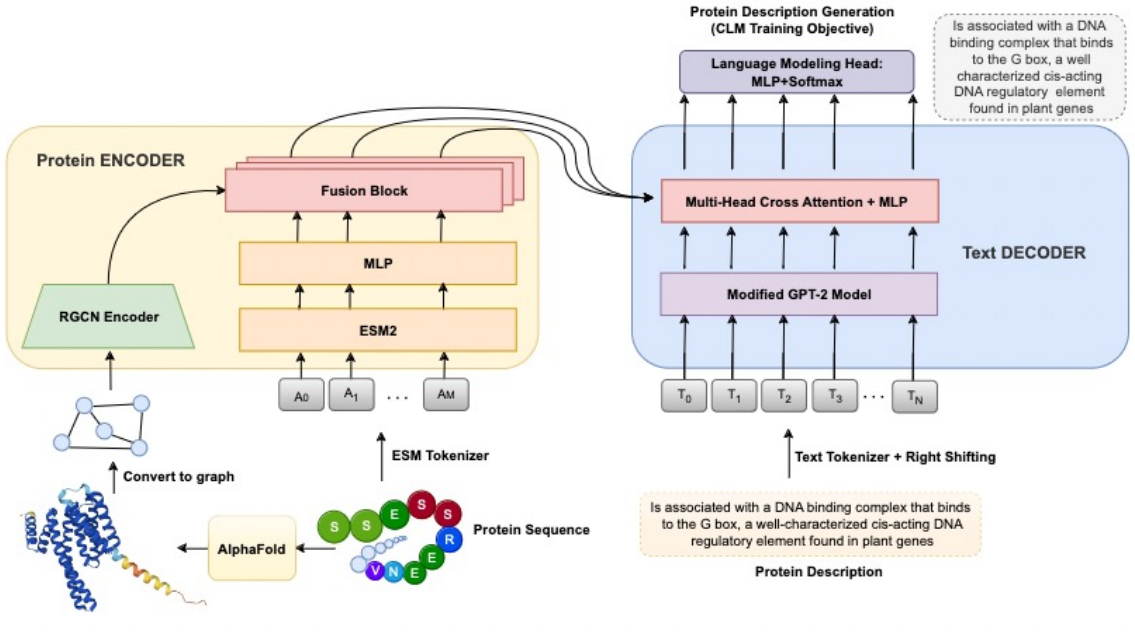


Prot2Text: Multimodal Protein's Function Generation with GNNs and Transformer

- **Sequence Encoding**. used ESM2-35M (Lin et al. 2023a) as our base model. $\boldsymbol{H}_S^0 = ESM(P_S)\boldsymbol{W}_p$
- transforms the individual amino-acid representations, derived from the ESM embedding dimension, into the graph embedding dimension d_{out}.
- **Multimodal Fusion**.
- obtain the final protein encoding, with a fusion block combining the two representations:

$$\boldsymbol{H}_S^{k+1} = \left(\boldsymbol{H}_S^k + \mathbf{1}_n \boldsymbol{h}_G \boldsymbol{W}_V^k\right) \boldsymbol{W}_O^k$$

- **Text Generation** transformer decoder architecture for generating protein descriptions.
- initialize decoder components (text embedding matrix, self-attention, and language modelling head), with the pre-trained weights of GPT-2.
- forward the protein representation obtained from the protein encoder as input to the multihead *cross-attention module* within the transformer decoder.
- enabled to *effectively incorporate context from the protein representation*, to generate coherent and meaningful protein descriptions.



Prot2Text: Multimodal Protein's Function Generation with GNNs and Transformer

Dataset.

- build a *multimodal dataset with 256, 690 proteins.*
- For each protein: sequence, the AlphaFold accession ID and the textual description.
- To build this dataset, we used the SwissProt dababase (Bairoch and Apweiler 1996)
- Apply CD-HIT clustering algorithm (Li and Godzik 2006) to create a train/validation/test scheme (248.215/ 4. 172/4, 023 proteins respectively).
- maximum similarity threshold between the (train, validation test)
  sets used in the CD-HIT algorithm is 40%.

Metrics

- evaluate the performance of the model in the text generation task.
- BLEU Score (Papineni et al.2002): similarity between generated and reference text based onn-grams.
- Rouge-* (Lin 2004): uni/bi/longest common subsequence between generated and reference text.
- BERT Score (Zhanget al. 2020): measures the similarity between the generated text and the reference text using contextualized word embeddings from a transformer-based model.

Prot2Text: Experimental Resutls (ongoing)

Model	# Params	BLEU Score	Rouge-1	Rouge-2	Rouge-L	BERT Score
vanilla-Transformer	225M	15.75	27.80	19.44	26.07	75.58
ESM2-35M	225M	32.11	47.46	39.18	45.31	83.21
RGCN	220M	21.63	36.20	28.01	34.40	78.91
RGCN + ESM2-35M	255M	30.39	45.75	37.38	43.63	82.51
RGCN × vanilla-Transformer	283M	27.97	42.43	34.91	40.72	81.12
Prot2Text$_{\text{BASE}}$	283M	**35.11**	**50.59**	**42.71**	**48.49**	**84.30**

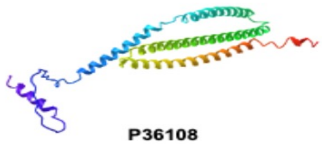
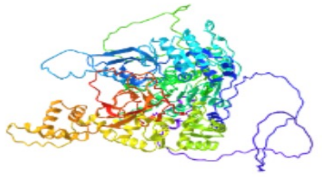
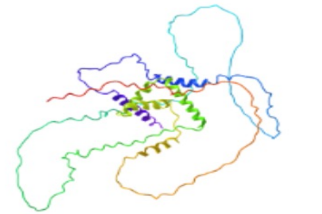
- Test set results encoder models,
- unimodal encoders: vanilla-Transformer, ESM2-35M,and RGCN,
- multimodal encoders: RGCN × vanilla-Transformer, RGCN + ESM2-35.
- All models share the same GPT-2 decoder.
- Structure increases performance (cross attention)
- Prot2TextBASE achieves the highest performance across all evaluation metrics

Prot2Text: Experimental Resutls (ongoing)

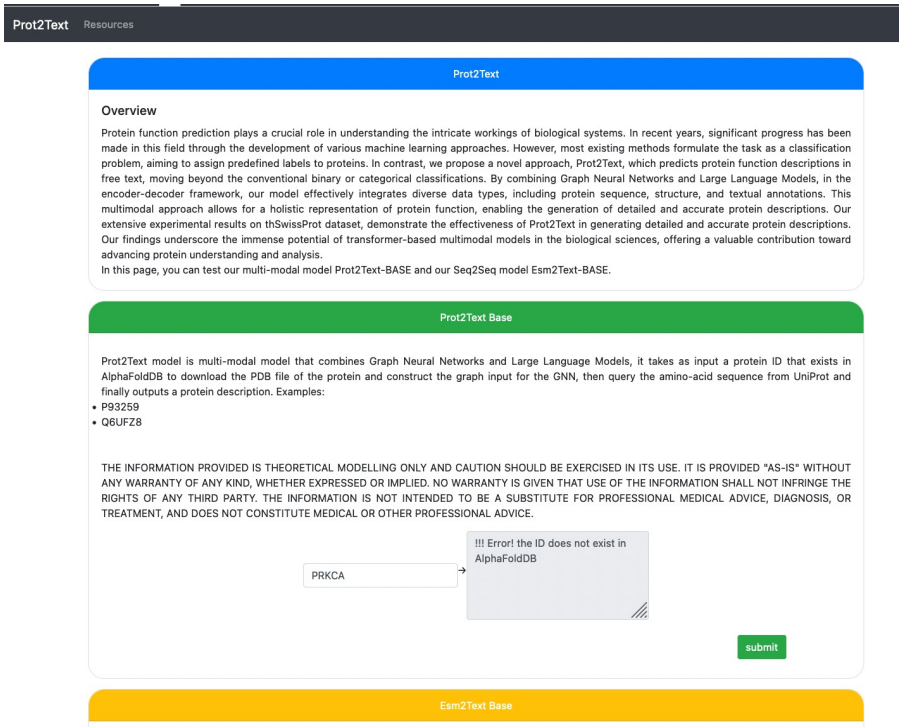
Model	# Params	BLEU Score	Rouge-1	Rouge-2	Rouge-L	BERT Score	Inference Time
Prot2Text$_{SMALL}$	256M	30.01	45.78	38.08	43.97	82.60	1,225
Prot2Text$_{BASE}$	283M	35.11	50.59	42.71	48.49	84.30	1,379
Prot2Text$_{MEDIUM}$	398M	**36.51**	52.13	44.17	50.04	84.83	1,334
Prot2Text$_{LARGE}$	898M	36.29	**53.68**	**45.60**	**51.40**	**85.20**	1,667

- Test set results for different size variations of Prot2Text.
- Larger models outperform their smaller counterparts across most evaluation metrics, indicating the benefits of employing larger language models in the Prot2Text framework.
- The Prot2TextBASE model, strikes an optimal balance between performance and computational efficiency. This configuration demonstrates improved performance compared to the smaller model while still maintaining reasonable computational costs.

Prot2Text: Experimental Resutls (ongoing)

Protein	Labeled Text	Generated Text
 P36108	Required for the sorting and concentration of proteins resulting in the entry of these proteins into the invaginating vesicles of the multivesicular body (MVB). Acts a component of the ESCRT-III complex, which appears to be critical for late steps in MVB sorting, such as membrane invagination and final cargo sorting and recruitment of late-acting components of the sorting machinery. The MVB pathway requires the sequential function of ESCRT-O, -I,-II and -III complex assemblies. Can directly stimulate VPS4 ATPase activity. The DID4/VPS2-VPS24 subcomplex is required for the VPS4-dependent dissociation of ESCRT-III.	Probable peripherally associated component of the endosomal sorting required for transport complex III (ESCRT-III) which is involved in multivesicular bodies (MVBs) formation and sorting of endosome-derived cargo proteins into MVBs. MVBs contain intraluminal vesicles (ILVs) that are generated by invagination and scission from the limiting membrane of the cell and mostly are delivered to lysosomes enabling degradation of membrane proteins, such as stimulated growth factor receptors, lysogenic enzymes and lipids. The MVB pathway appears to require the sequential function of ESCRT-O, -I,-II and -III complexes. ESCRT proteins mostly dissociate from the invaginating membrane before the ILV is released. The ESCRT machinery also functions in topologically equivalent membrane fission events, such the terminal stages of cytokinesis and the budding of enveloped viruses (HIV-1 and other lentiviruses). ESCRT I, -II and III proteins mostly mediate the sorting of proteins into lysosphingolipids, primarily cytoplasmic membrane vesicle-derived proteins. ESCR-III proteins are believed to mediate a specialized role in the final stages
 Q8NG08	5'-3' DNA helicase involved in DNA damage response by acting as an inhibitor of DNA end resection. Recruitment to single-stranded DNA (ssDNA) following DNA damage leads to inhibit the nucleases catalyzing resection, such as EXO1, BLM and DNA2, possibly via the 5'-3' ssDNA translocase activity of HELB. As cells approach S phase, DNA end resection is promoted by the nuclear export of HELB following phosphorylation. Acts independently of TP53BP1. Unwinds duplex DNA with 5'-3' polarity. Has single-strand DNA-dependent ATPase and DNA helicase activities. Prefers ATP and dATP as substrates. During S phase, may facilitate cellular recovery from replication stress.	DNA-dependent ATPase and 5'-3' DNA helicase required for the maintenance of genome stability. Involved in various processes such as transcription-coupled nucleotide excision repair, mitotic spindle assembly, DNA damage response and DNA repair. During DNA double-strand break (DSB) repair, involved in limiting the number of single-stranded DSBs through DNA end resection by promoting the recruitment of RAD51 to sites of DNA damage. In addition to DNA end reannealing, also promotes the resumption of DNA synthesis from the new 3'-tail end of the existing DNA ends. Acts as a regulator of telomerase by inhibiting telomeric silencing: binds to telomeromyosin, and is required for telomeronucleolytic silencing of endogenous small interfering RNAs (endo-siRNAs) and their targeting to the telomere. Involves also DNA damage signaling via the 5'-5' helicase and DNA helicases activities, preventing disintegration of the DNA duplex and subsequent resection of the damaged DNA strand. In vitro able to unwind 5'-overhanging flap DNA and catalyzes ATP-dependent unwinding of 5'-DNA ends. Also plays a role in
 P35713	Transcriptional activator that binds to the consensus sequence 5'-AACAAAG-3' in the promoter of target genes and plays an essential role in embryonic cardiovascular development and lymphangiogenesis. Activates transcription of PROX1 and other genes coding for lymphatic endothelial markers. Plays an essential role in triggering the differentiation of lymph vessels, but is not required for the maintenance of differentiated lymphatic endothelial cells. Plays an important role in postnatal angiogenesis, where it is functionally redundant with SOX17. Interaction with MEF2C enhances transcriptional activation. Besides, required for normal hair development.	Transcriptional activator that binds to the consensus sequence 5'-AACAAAG-3'. Plays an essential role in cellular differentiation, proliferation and survival. Plays a critical role in macrophage differentiation, migration and invasion, particularly in the gut. Required for normal gene expression in the macrophages, which are activated by biglycan-producing bacteria and fungi. Also required for normal chemotaxis. Plays important roles in the development of the central nervous system, where it is required for proper proliferation and migration of progenitor cells.

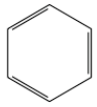
Demonstration



Multimodal Gen AI for molecules

- Task: Generate or modify the structure of molecules based on text descriptions.
- Motivation:
  - Discovery in medicine and science is expensive.
  - huge amount of time and money for experts to design new molecules with the desired functionality every year.
  - deep learning tools are essential for facilitating molecule discovery.
- `“Water is an oxygen hydride consisting of an oxygen atom that is covalently bonded to two hydrogen atoms”` => H - O - H

- Molecule representations
  - SMILES(Simplified Molecular Input Line Entry System): string of compact encoding of molecular structures, making it easy to share and search.
  - Graph: each atom in the molecule corresponds to a node in the graph, bonds between atoms are represented as edges, where some attributes can be applied to nodes and edges.
  - else: SELFIES, 3-D structure, fingerprints..

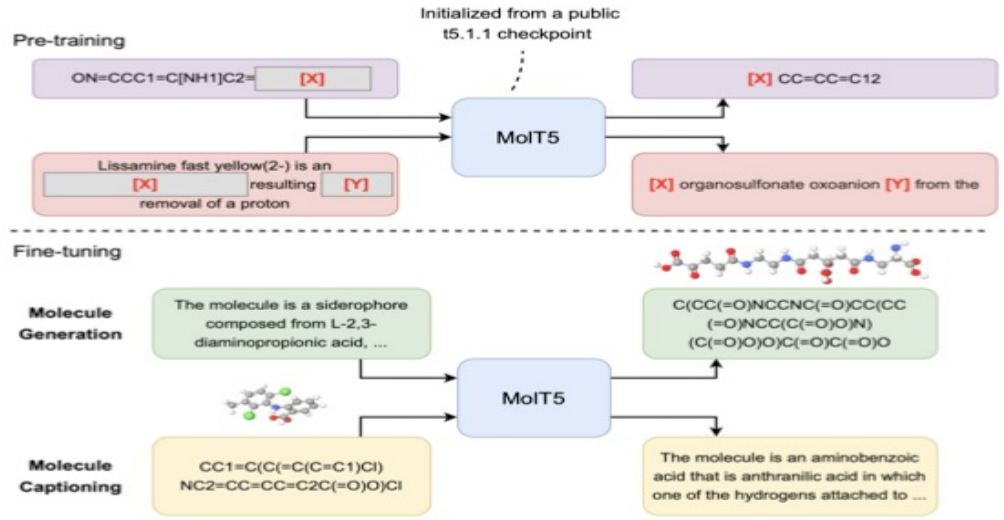
Benzene:  (C6H6)

- SMILES: c1ccccc1
- Graph: as adjacency matrix:

$$\begin{matrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{matrix}$$

MolT5[1]

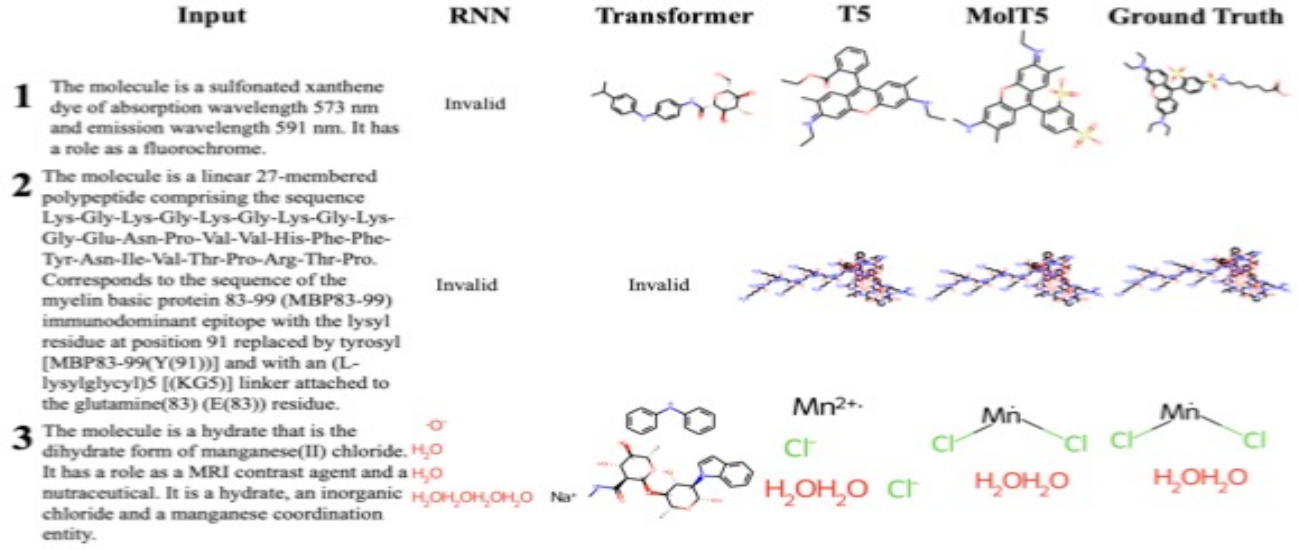
- A self-supervised learning framework for pre-training models on unlabeled natural language text and molecule strings.
- First pre-train MolT5 on both SMILES string and natural language using the “replace corrupted spans” objective. Fine-tuned for task of molecule captioning or generation.



[1] Translation between Molecules and Natural Language, Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, Heng Ji, https://arxiv.org/abs/2204.11817

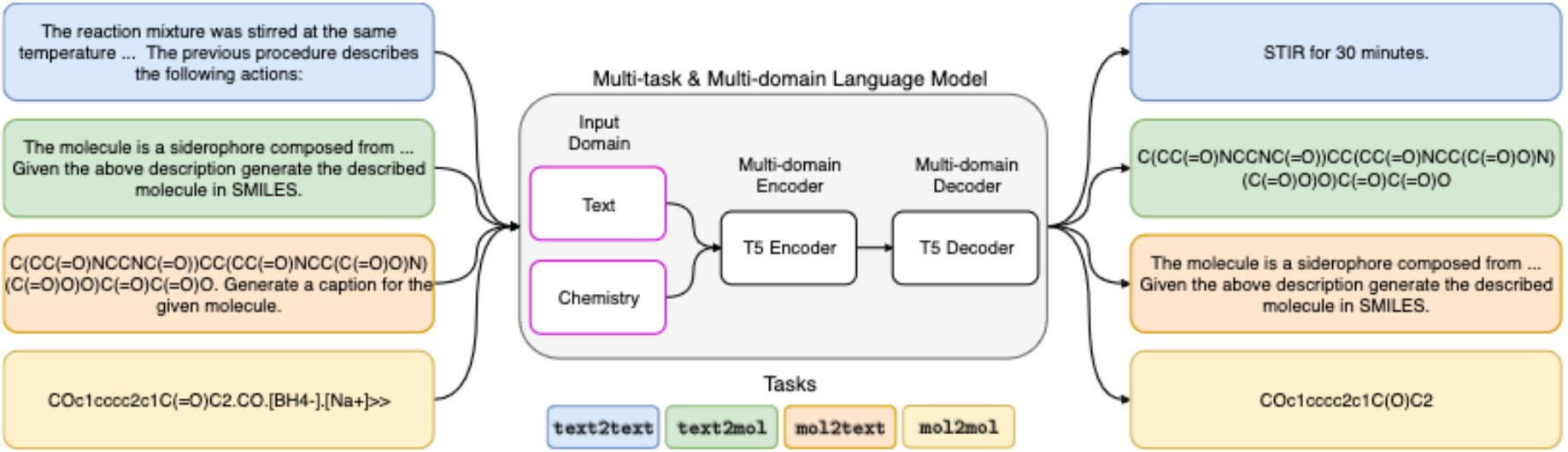
MolT5[1]

Model	BLEU↑	Exact↑	Levenshtein↓	MACCS FTS↑	RDK FTS↑	Morgan FTS↑	FCD↓	Text2Mol↑	Validity↑
Ground Truth	1.000	1.000	0.0	1.000	1.000	1.000	0.0	0.609	1.0
RNN	0.652	0.005	38.09	0.591	0.400	0.362	4.55	0.409	0.542
Transformer	0.499	0.000	57.66	0.480	0.320	0.217	11.32	0.277	**0.906**
T5-Small	0.741	0.064	27.703	0.704	0.578	0.525	2.89	0.479	0.608
MolT5-Small	0.755	0.079	25.988	0.703	0.568	0.517	2.49	0.482	0.721
T5-Base	0.762	0.069	24.950	0.731	0.605	0.545	2.48	0.499	0.660
MolT5-Base	0.769	0.081	24.458	0.721	0.588	0.529	2.18	0.496	0.772
T5-Large	0.854	0.279	16.721	0.823	0.731	0.670	1.22	0.552	0.902
MolT5-Large	**0.854**	**0.311**	**16.071**	**0.834**	**0.746**	**0.684**	**1.20**	**0.554**	0.905



Text+Chem T5[1]

- first multi-domain, multi-task LM for chemical and natural language domains.1
- Key Ideas:
  - Multi-tasking across multiple domains.
    Weight sharing and information sharing across domains(encoders) .
  - efficient training strategy without the need for costly pre-training on large dataset and task-specific fine-tuning.



Unifying Molecular and Textual Representations via Multi-task Language Modelling, https://arxiv.org/abs/2301.12586
Dimitrios Christofidellis, Giorgio Giannone, Jannis Born, Ole Winther, Teodoro Laino, Matteo Manica
Demo: https://huggingface.co/spaces/GT4SD/multitask-text-and-chemistry-t5

Text+Chem T5

Results(all):

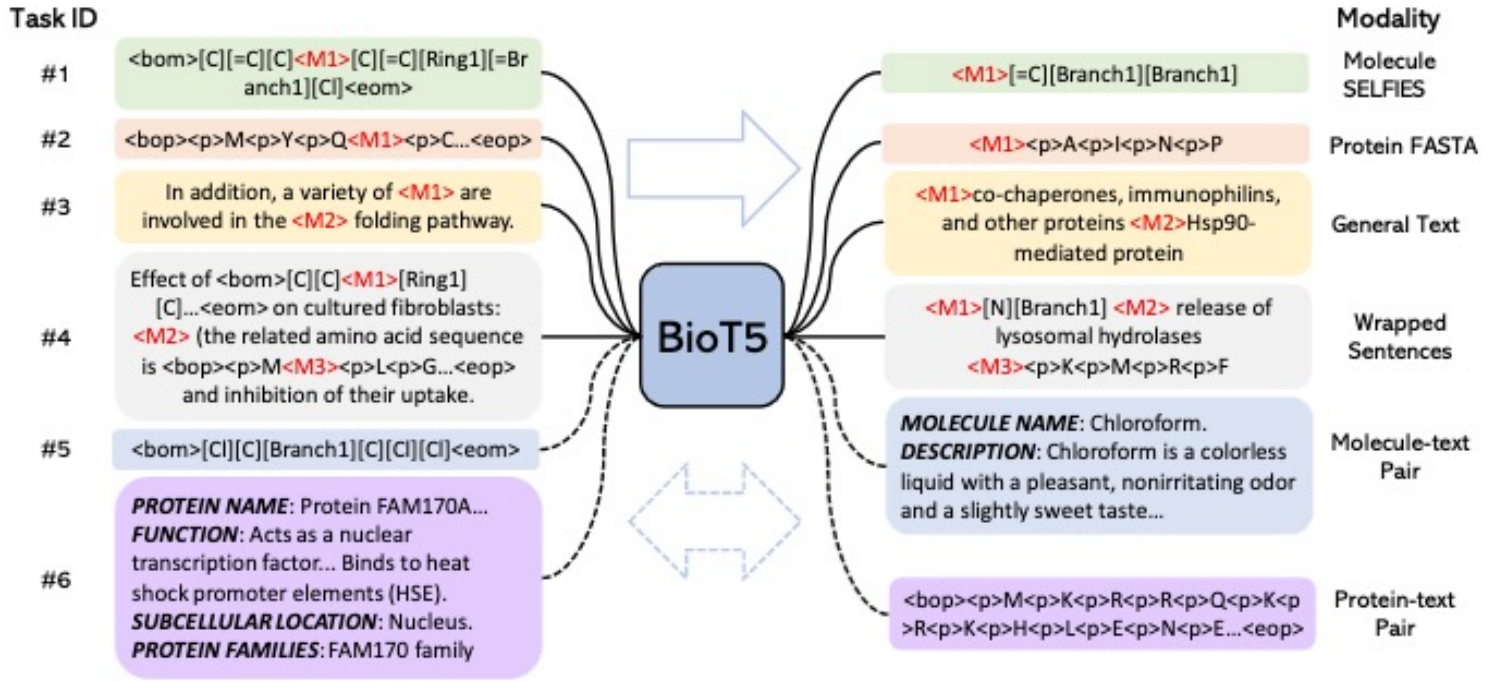
Domain		mol2mol		cross-domain		text2text
Task	Size	forward	retrosynthesis	text2mol	mol2text	paragraph-actions
T5 (fine-tuned) (Raffel et al., 2020)	small	0.603	0.245	0.499	0.501	**0.953**
T5 (fine-tuned) (Raffel et al., 2020)	base	0.629	-	0.762	0.511	-
RXN-forward (Toniato et al., 2021)	-	**0.685**	-	-	-	-
RXN-retrosynthesis (Toniato et al., 2021)	-	-	**0.733**	-	-	-
RXN-paragraph2actions (Vaucher et al., 2020)	-	-	-	-	-	0.850
MolT5 (Edwards et al., 2022)	small	-	-	0.755	0.519	-
MolT5 (Edwards et al., 2022)	base	-	-	0.769	0.540	-
***Text+Chem T5* (ours)**	small	0.412	0.249	0.815	0.553	0.929
***Text+Chem T5* (ours)**	base	0.459	0.478	0.750	0.580	0.935
***Text+Chem T5-augm* (ours)**	small	0.413	0.405	0.815	0.560	0.926
***Text+Chem T5-augm* (ours)**	base	0.594	0.372	**0.853**	**0.625**	0.943

Results(text2mol):

	Size	BLEU score ↑	Accuracy ↑	Levenshtein ↓	MACCS FTS↑	RDK FTS↑	Morgan FTS↑	FCD↓	Validity↑
Transformer (Edwards et al., 2022)	-	0.499	0	57.66	0.480	0.320	0.217	11.32	0.906
T5 (fine-tuned) (Raffel et al., 2020)	small	0.741	0.064	27.7	0.704	0.578	0.525	2.89	0.608
MolT5 (Edwards et al., 2022)	small	0.755	0.079	25.99	0.703	0.568	0.517	2.49	0.721
***Text+Chem T5* (ours)**	small	0.739	0.157	28.54	0.859	0.736	0.660	0.066	0.776
***Text+Chem T5-augm* (ours)**	small	**0.815**	**0.191**	**21.78**	**0.864**	**0.744**	**0.672**	**0.060**	**0.951**
T5 (fine-tuned) (Raffel et al., 2020)	base	0.762	0.069	24.95	0.731	0.605	0.545	2.48	0.660
MolT5 (Edwards et al., 2022)	base	0.769	0.081	24.49	0.721	0.588	0.529	0.218	0.772
***Text+Chem T5* (ours)**	base	0.750	0.212	27.39	0.874	0.767	0.697	0.061	0.792
***Text+Chem T5-augm* (ours)**	base	**0.853**	**0.322**	**16.87**	**0.901**	**0.816**	**0.757**	**0.050**	**0.943**

Bio T5 - https://arxiv.org/abs/2310.07276

- BioT5 uses T5 model to incorporate modalities, in pre-training it uses:
- modality including molecule SELFIES, and general text independently.
- wrapped text from scientific corpus.
- Bidirectional translation for the molecule SELFIES-text pairs.



Bio T5

- Text2mol task

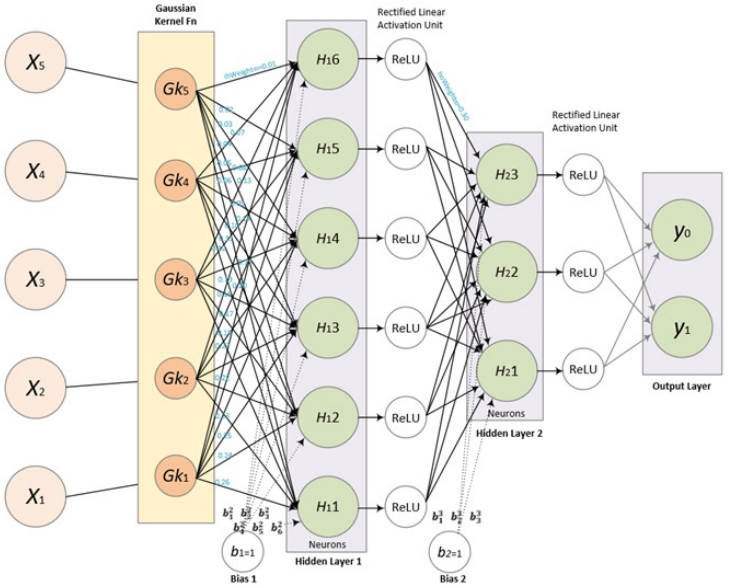
Model	#Params.	BLEU↑	Exact↑	Levenshtein↓	MACCS FTS↑	RDK FTS↑	Morgan FTS↑	FCD↓	Text2Mol↑	Validity↑
RNN	56M	0.652	0.005	38.09	0.591	0.400	0.362	4.55	0.409	0.542
Transformer	76M	0.499	0.000	57.66	0.480	0.320	0.217	11.32	0.277	0.906
T5-small	77M	0.741	0.064	27.703	0.704	0.578	0.525	2.89	0.479	0.608
T5-base	248M	0.762	0.069	24.950	0.731	0.605	0.545	2.48	0.499	0.660
T5-large	783M	0.854	0.279	16.721	0.823	0.731	0.670	1.22	0.552	0.902
MolT5-small	77M	0.755	0.079	25.988	0.703	0.568	0.517	2.49	0.482	0.721
MolT5-base	248M	0.769	0.081	24.458	0.721	0.588	0.529	2.18	0.496	0.772
MolT5-large	783M	0.854	0.311	16.071	0.834	0.746	0.684	1.20	0.554	0.905
GPT-3.5-turbo (zero-shot)	>175B	0.489	0.019	52.13	0.705	0.462	0.367	2.05	0.479	0.802
GPT-3.5-turbo (10-shot MolReGPT)	>175B	0.790	0.139	24.91	0.847	0.708	0.624	0.57	0.571	0.887
MolXPT	350M	-	0.215	-	0.859	0.757	0.667	0.45	**0.578**	0.983
BioT5	252M	**0.867**	**0.413**	**15.097**	**0.886**	**0.801**	**0.734**	**0.43**	0.576	**1.000**

Conclusions

- Graph Generators
  - High potential topic with crucial applications: Power grid/telecoms, VLSI, social networks, proteins/pharma
  - Different challenges
    - pretraining tasks are diverse, masking may not be enough
    - Decoder architecture – permutation invariance
    - Graph data loaders scaling
    - Prompting…
    - Multimodality with graphs – architectural challenges
- Tasks for evaluation – potential
  - for genenerated graphs similarity is non trivial (graph kernels/embeddings)

Some thoughts on AI for interpretability

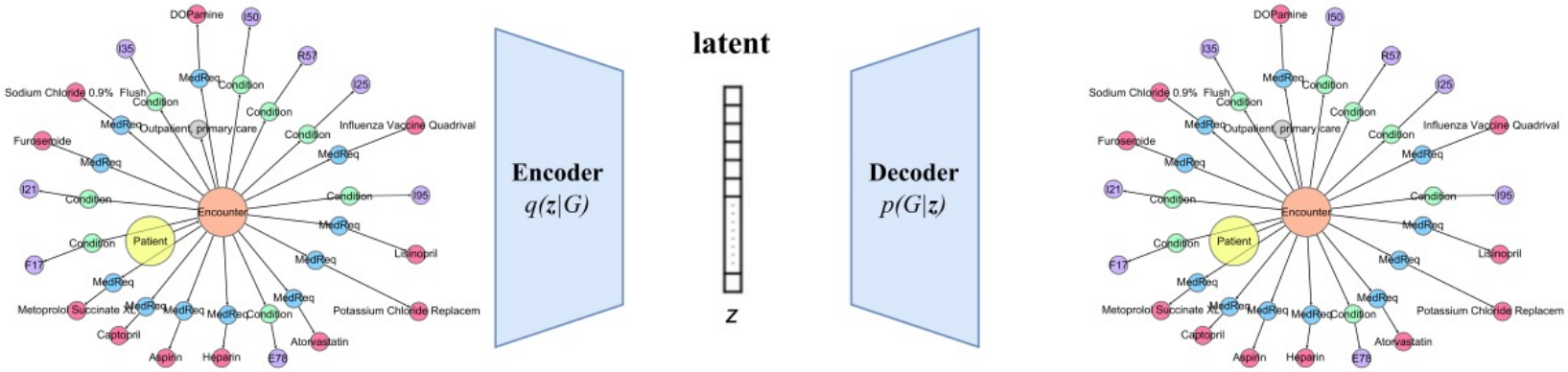
- Deep learning learn during training a high dimensional function between the input X and output Y: $f(X)=>Y$
- Current AI models are very large (i.e. in the order of 10^9 of parameters)
- Human congnition insufficient to interpret such functions (limit to 4-7 dimensions/items[1])
- Challenges:
  - try to interpret these huge models
  - extract novel patterns/operators
  (i.e. equivariant operators for seq. data or graphs[2])



https://authoritypartners.com/deep-neural-networks-with-r-tensorflow-and-keras/

[1] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5822395/
[2] https://arxiv.org/abs/2101.10050, Dasoulas, Lutzyer, Vazirgiannis, **(ICLR), 2021.**

THANK YOU!



Acknowledgements

Dr. G. Nikolentzos, H.Abdine, M. Chatzianastassis, Yang Zhang