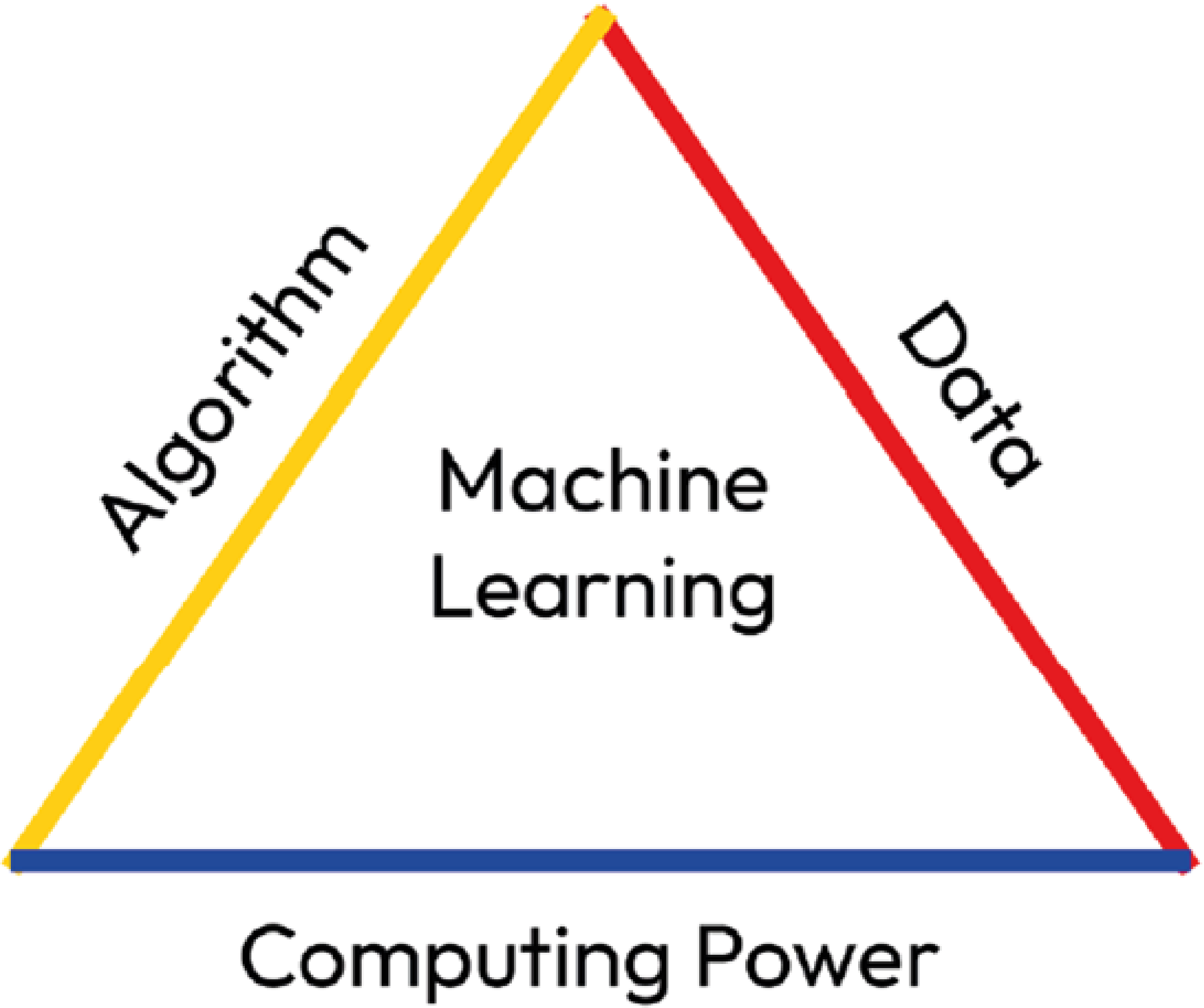
PharmeX Middle East 7th

Next-Generation Drug Discovery: The Role of AI in Global Pharma Innovation

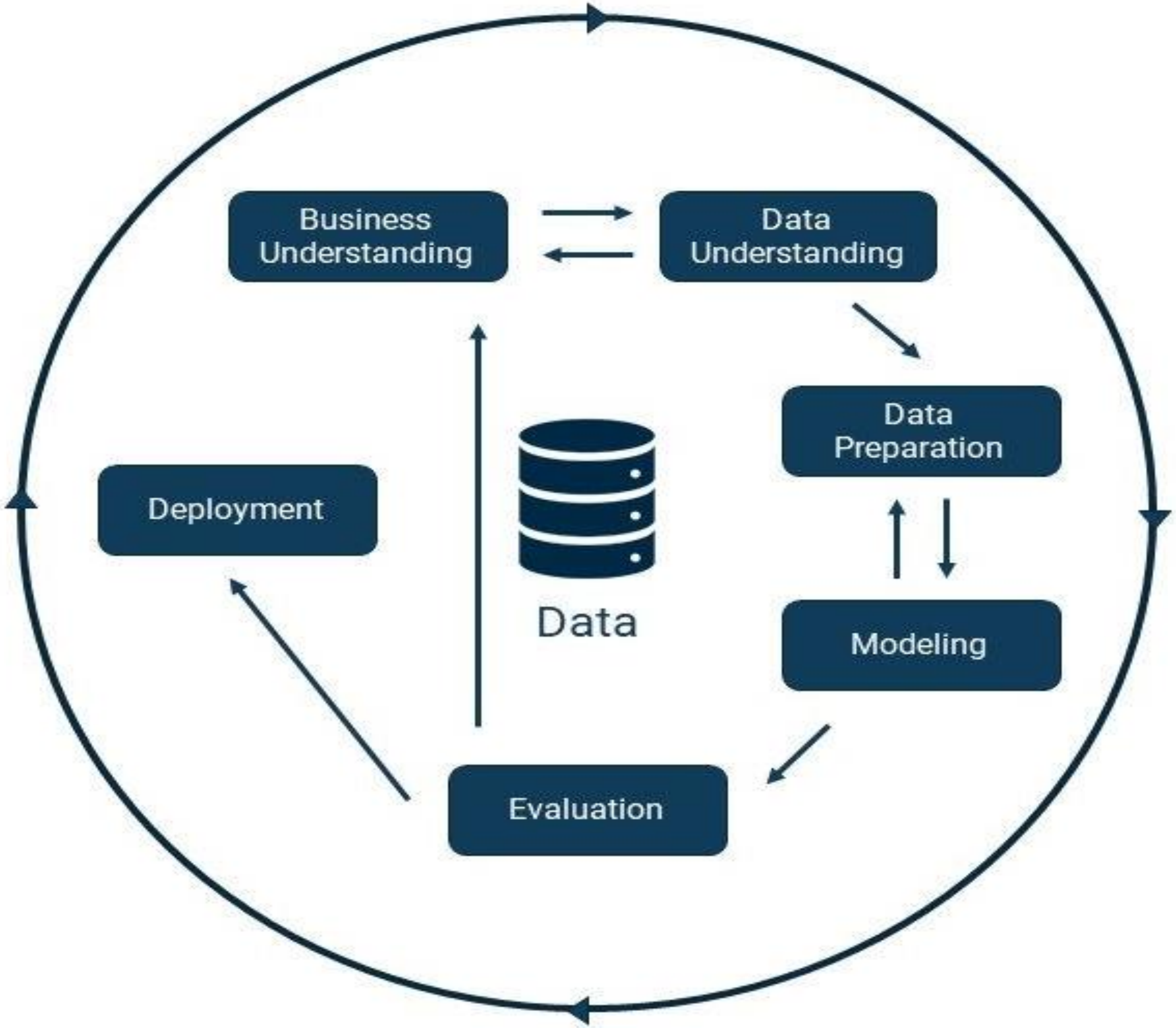
Houman Kazemzadeh, PharmD,

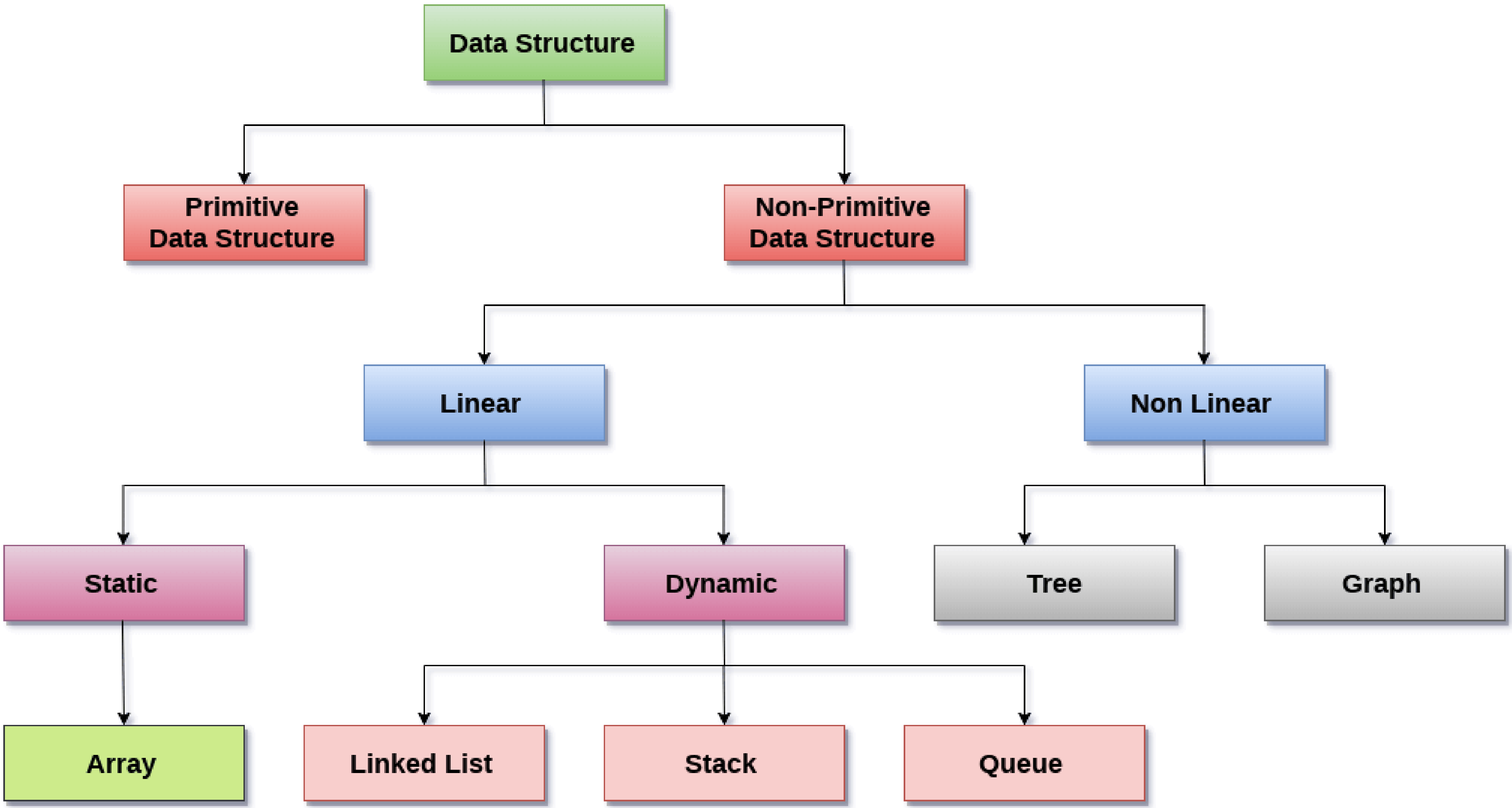
PhD Student in Medicinal Chemistry,

Tehran University of Medical Sciences


Algorithm
Data
Machine Learning
Computing Power

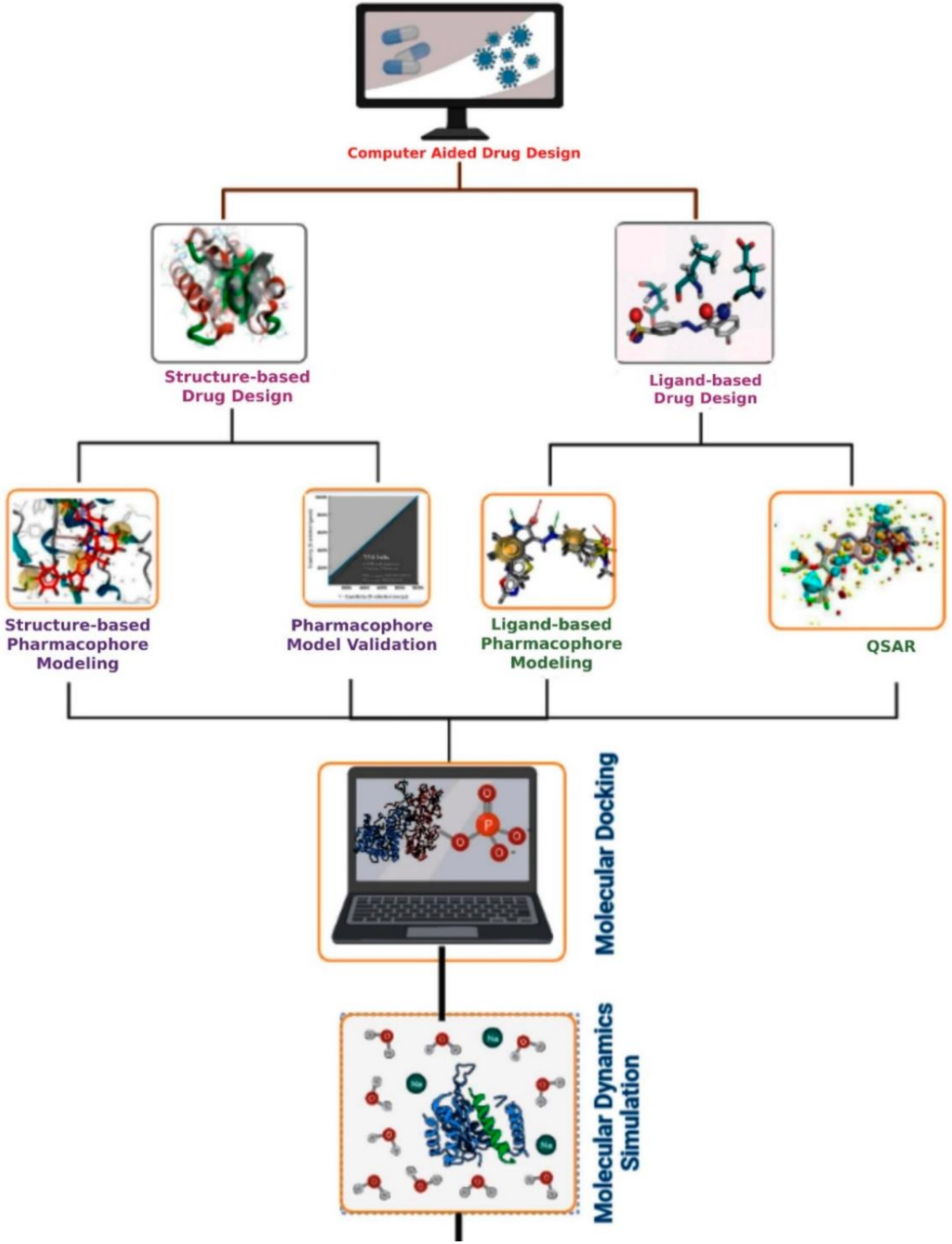
CRISP-DM


Business Understanding
Data Understanding
Data Preparation
Modeling
Evaluation
Deployment
Data


Data Structure
Primitive Data Structure
Non-Primitive Data Structure
Linear
Non Linear
Static
Dynamic
Tree
Graph
Array
Linked List
Stack
Queue

Computer Aided Drug Design (CADD)

- ***Molecular Modeling + Cheminformatics***
- ***Pharmacophore Modeling***
- ***Quantitative Structure-Activity Relationship (QSAR) Analysis***
- ***Docking Studies***
- ***Molecular Dynamics***
- ***ADME-Tox Prediction***



Computer Aided Drug Design

Structure-based Drug Design

Ligand-based Drug Design

Structure-based Pharmacophore Modeling

Pharmacophore Model Validation

Ligand-based Pharmacophore Modeling

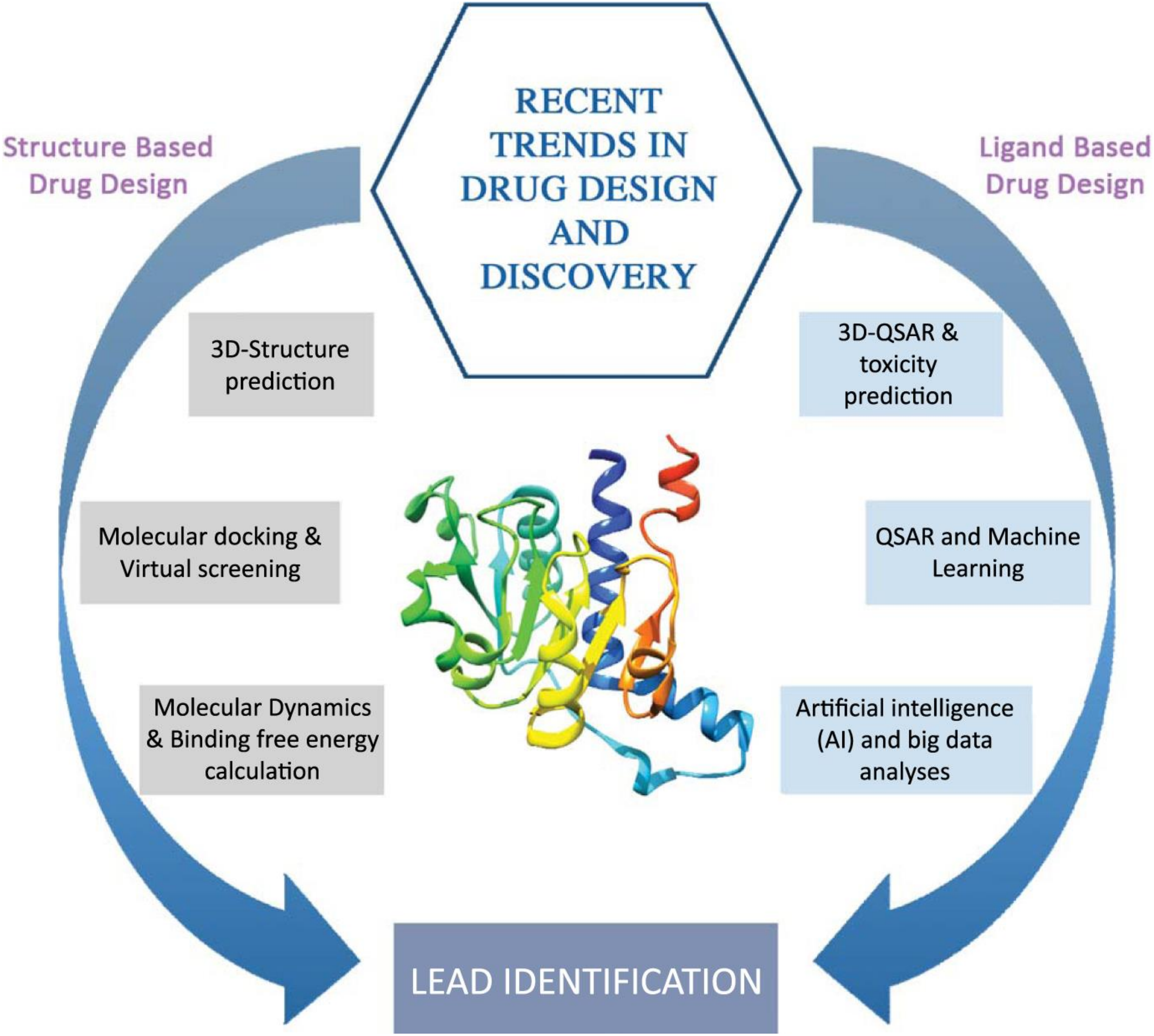
QSAR

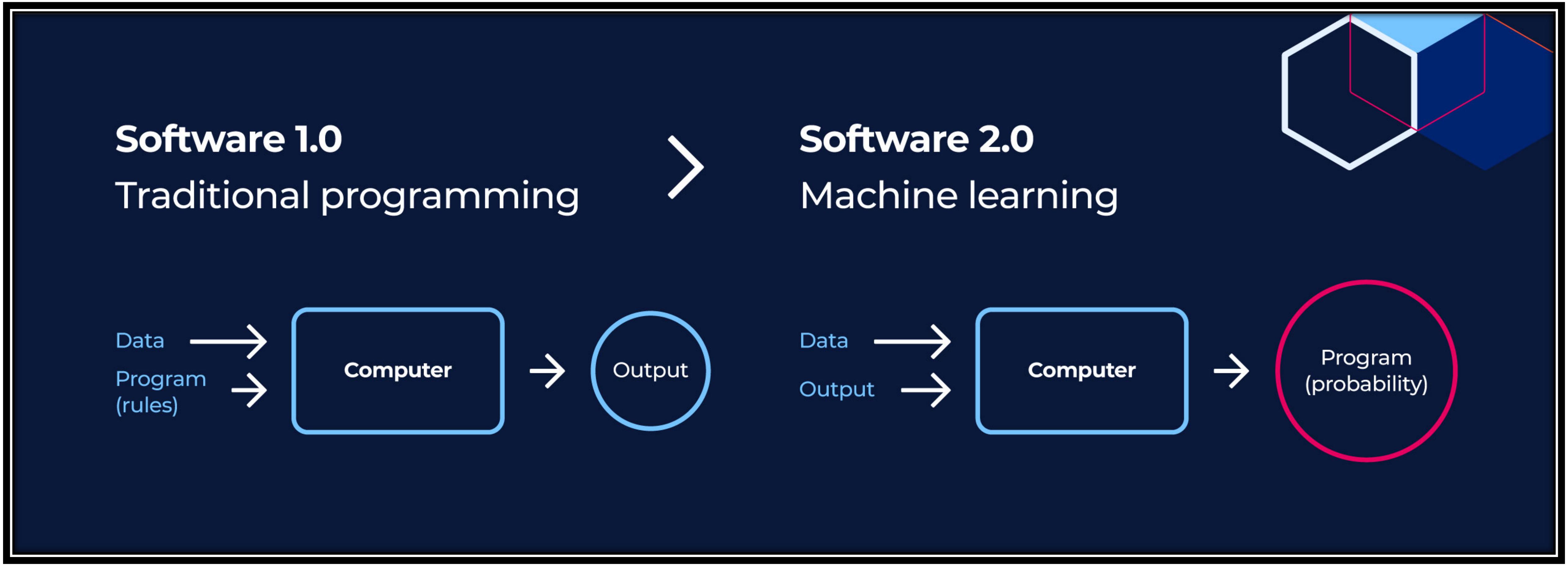
Molecular Docking

Molecular Dynamics Simulation

Method	Operation	Pros	Cons (Limitations)
2D QSAR	Uses molecular descriptors derived from 2D chemical structure (e.g., logP, molecular weight, atom counts) to model biological activity.	Simple, fast, low computational cost; interpretable descriptors.	Ignores 3D conformation, steric and spatial interactions; oversimplified representation.
3D QSAR	Uses 3D molecular alignment and steric/electrostatic fields to relate spatial features to biological activity (e.g., CoMFA, CoMSIA).	Captures spatial orientation; more predictive than 2D QSAR when alignment is optimal.	Highly sensitive to molecular alignment; fails if conformations are inaccurate; still lacks dynamics.
Ligand-Based Pharmacophore	Identifies common 3D features (e.g., hydrogen bond donor/acceptor, hydrophobic regions) from a set of active ligands to define essential pharmacophoric space.	No need for receptor structure; good for hit expansion; interpretable models.	Assumes similar ligands bind similarly; lacks receptor interaction info; static model.
Structure-Based Pharmacophore	Extracts pharmacophore features directly from ligand-receptor complex or protein binding site (e.g., via docking or crystallography).	Utilizes direct structural information from receptor; context-specific modeling.	Requires accurate receptor structure; limited by static snapshot; ignores molecular dynamics.

Where is AI


RECENT TRENDS IN DRUG DESIGN AND DISCOVERY
Structure Based Drug Design
Ligand Based Drug Design
3D-Structure prediction
Molecular docking & Virtual screening
Molecular Dynamics & Binding free energy calculation
3D-QSAR & toxicity prediction
QSAR and Machine Learning
Artificial intelligence (AI) and big data analyses
LEAD IDENTIFICATION



Software 1.0
Traditional programming

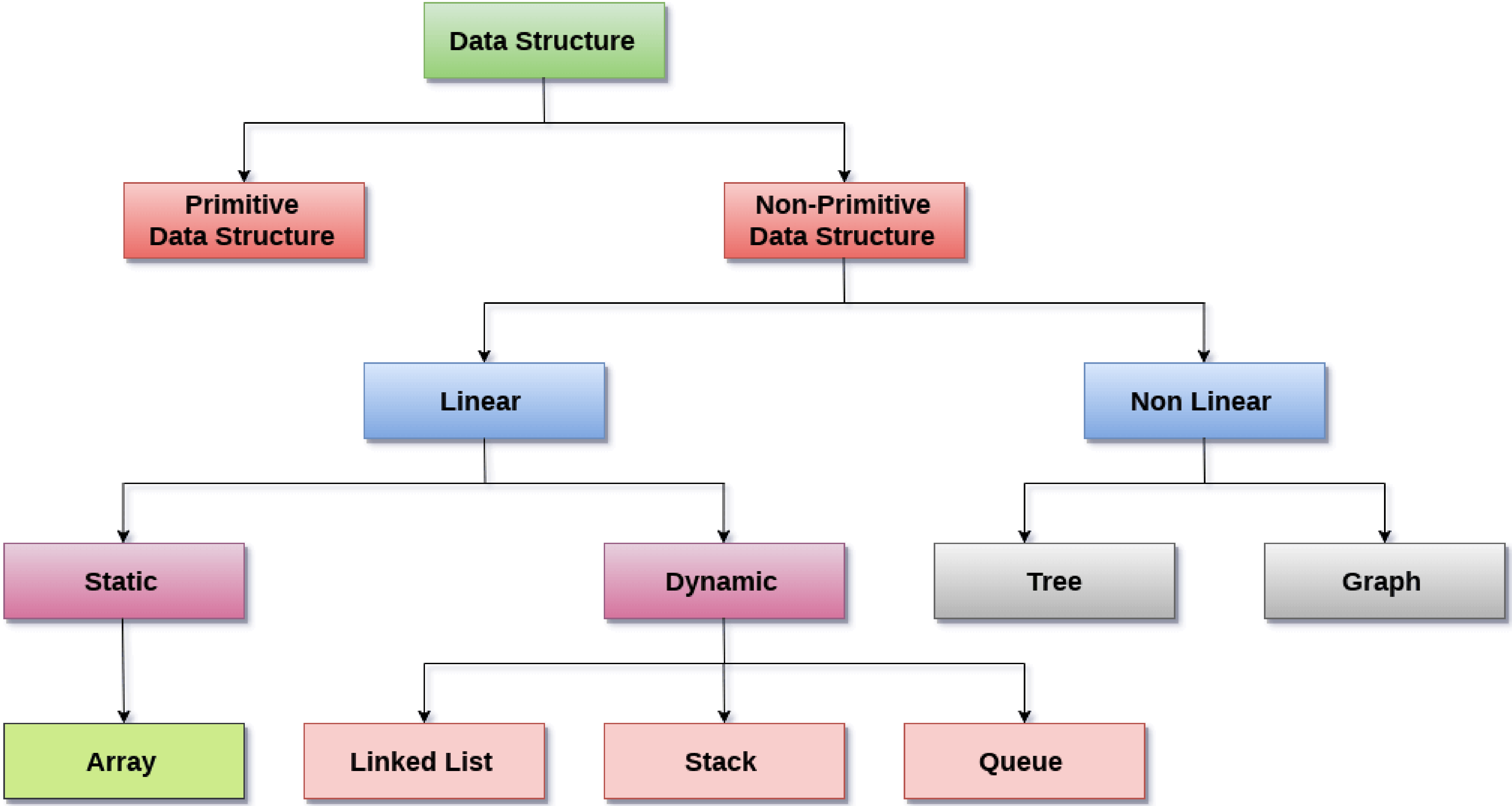
Software 2.0
Machine learning

Data → **Computer** → Output
Program (rules) →

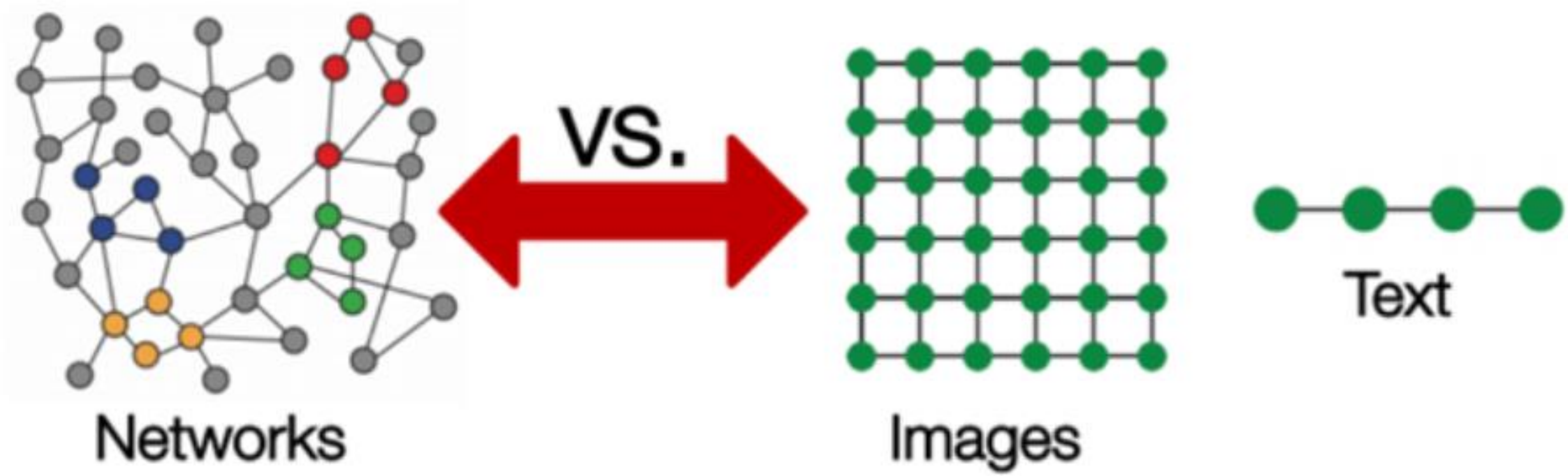
Data → **Computer** → Program (probability)
Output →

The Mathematical Background

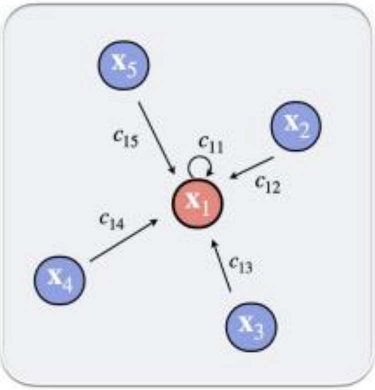
1. Probability
2. Decision Theory
3. Information Theory
4. Linear Algebra
5. Calculus
6. Generalized linear models
7. Optimization
8. Graphs | HyperGraphs
9. Inference algorithms
10. Message Passing Algorithms
11. Markov Chain Monte Carlo (MCMC)
12. Hamiltonian Monte Carlo (HMC)
13. Generative Models
14. State-Space Modeling
15. Latent Space Modeling
16. Reinforcement Learning
17. Deep Neural Networks
18. Causality
19. Beyond the iid assumption
20. …


Data Structure
Primitive Data Structure
Non-Primitive Data Structure
Linear
Non Linear
Static
Dynamic
Tree
Graph
Array
Linked List
Stack
Queue

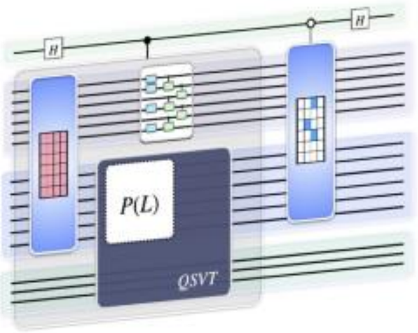
Everything is a Graph



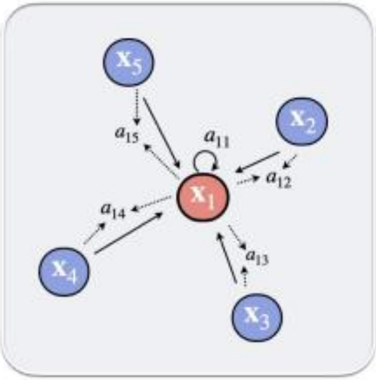
Convolutional



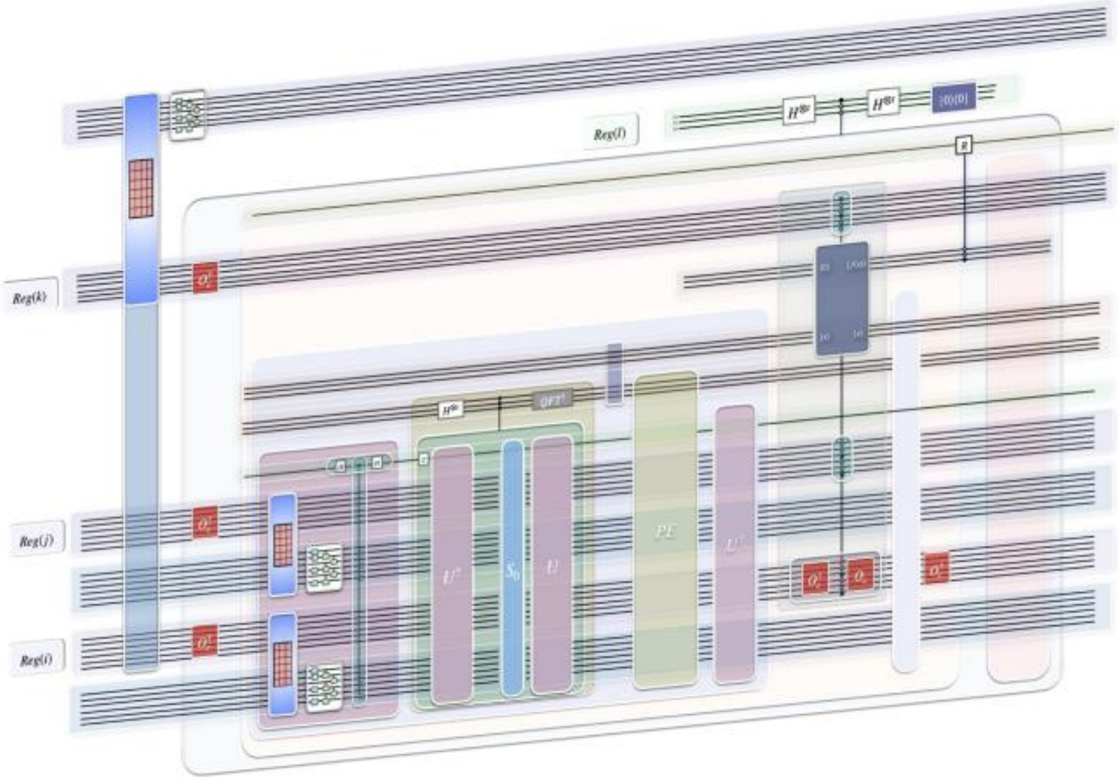
$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j\in\mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right)$$



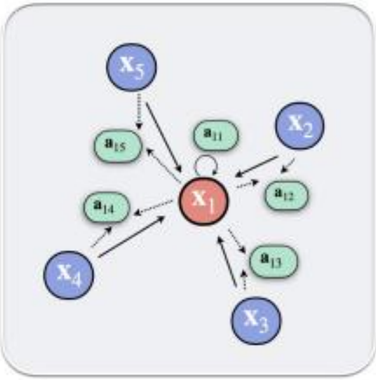
Attentional



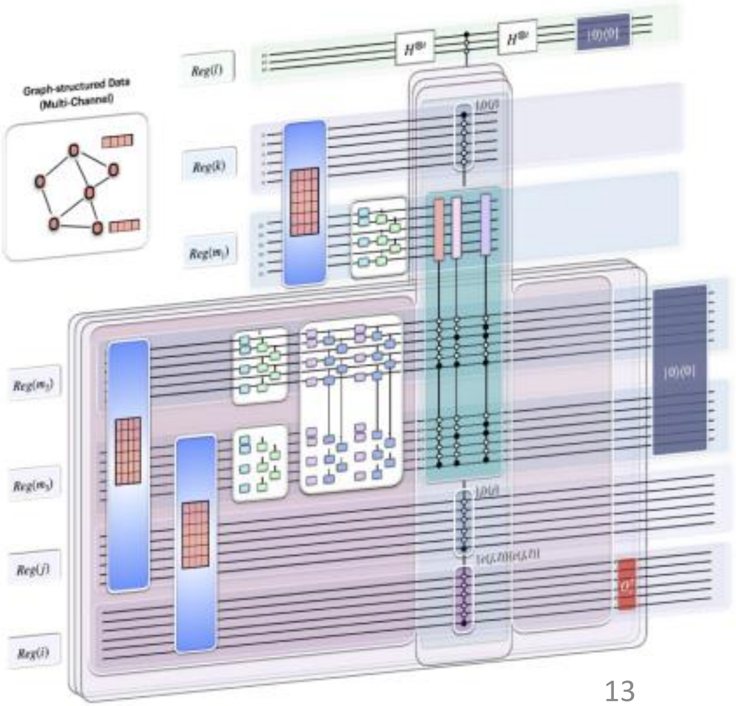
$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j\in\mathcal{N}_i} a(\mathbf{x}_i,\mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$



Message-Passing



$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j\in\mathcal{N}_i} \psi(\mathbf{x}_i,\mathbf{x}_j)\right)$$



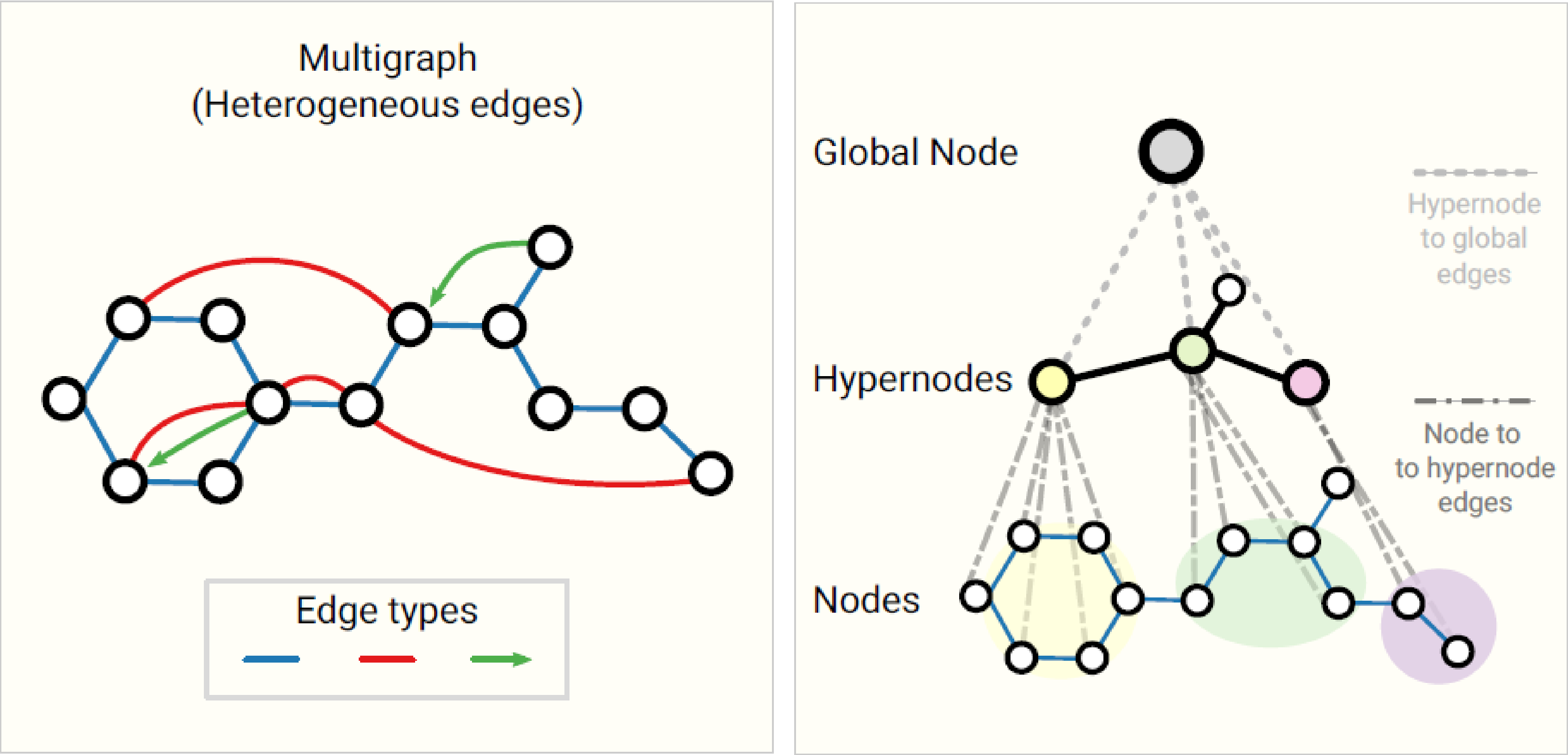
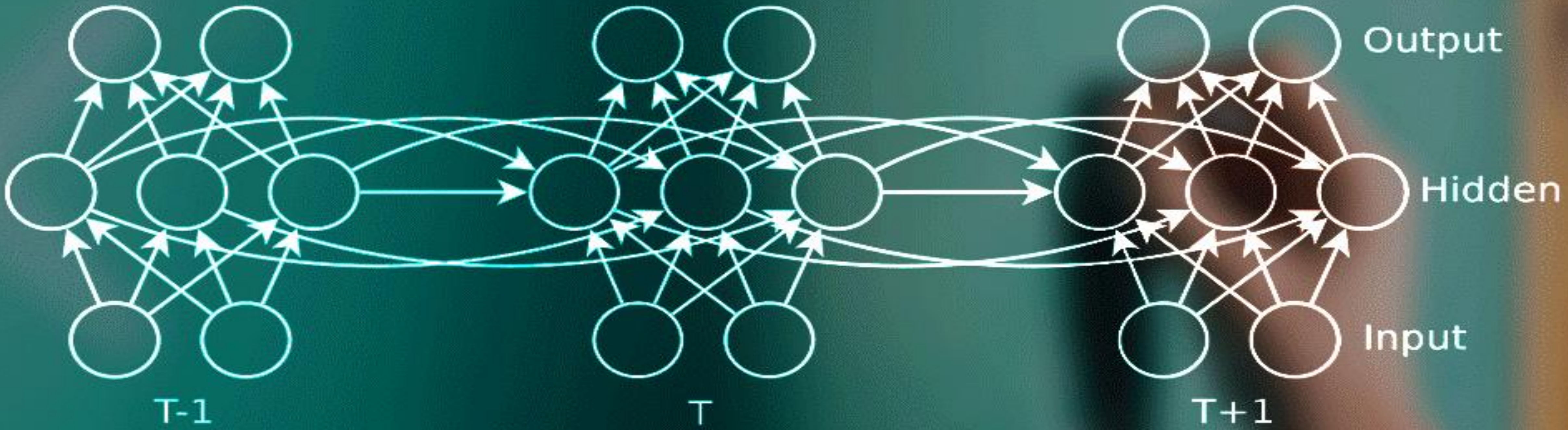


*Figure 16.10: Left: a multigraph can have different edge types. Right: a hypergraph can have edges which connect multiple nodes. From [*__Sanchez-lengeling2021__*]. Used with kind permission of Benjamin Sanchez-Lengeling.*

Recurrent Neural Networks (RNN)



1. **Sequential data**
2. **Order Matters**
3. **Dependency**
4. **Memory Needed:** internal hidden state that serves as a memory of previous inputs.
5. **Time matters:** the same set of weights and biases across all time steps
6. **Dynamic**

- **Sequential data examples:**

  Time Series Data

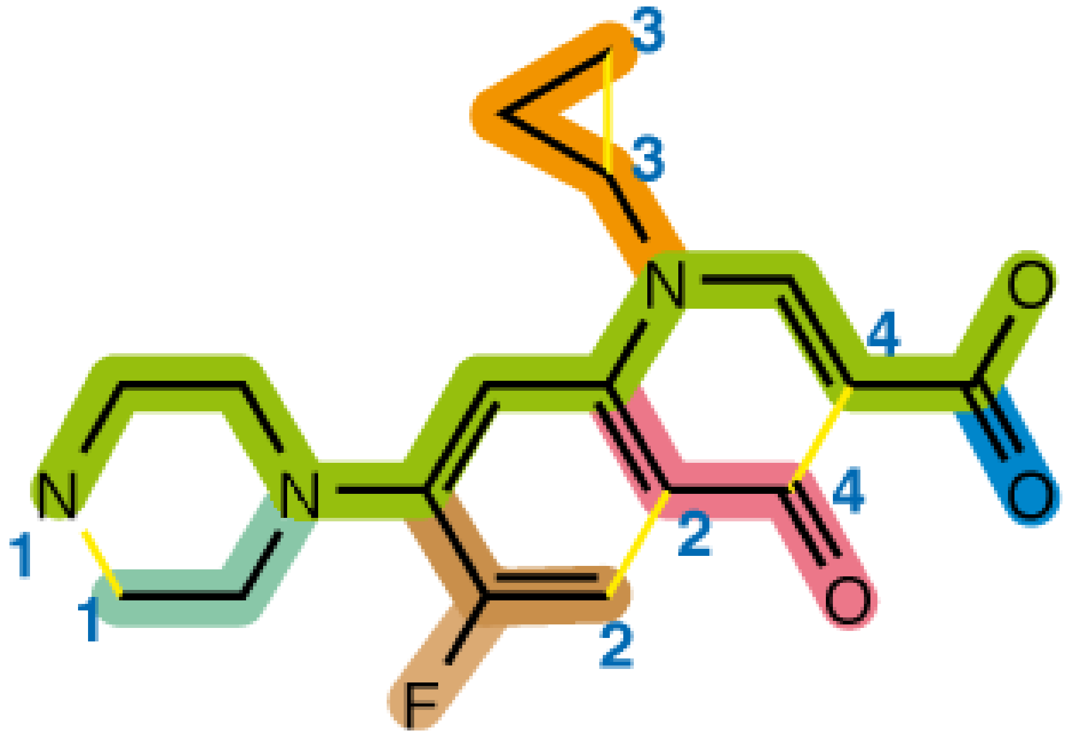
  Text Data

  DNA Sequences

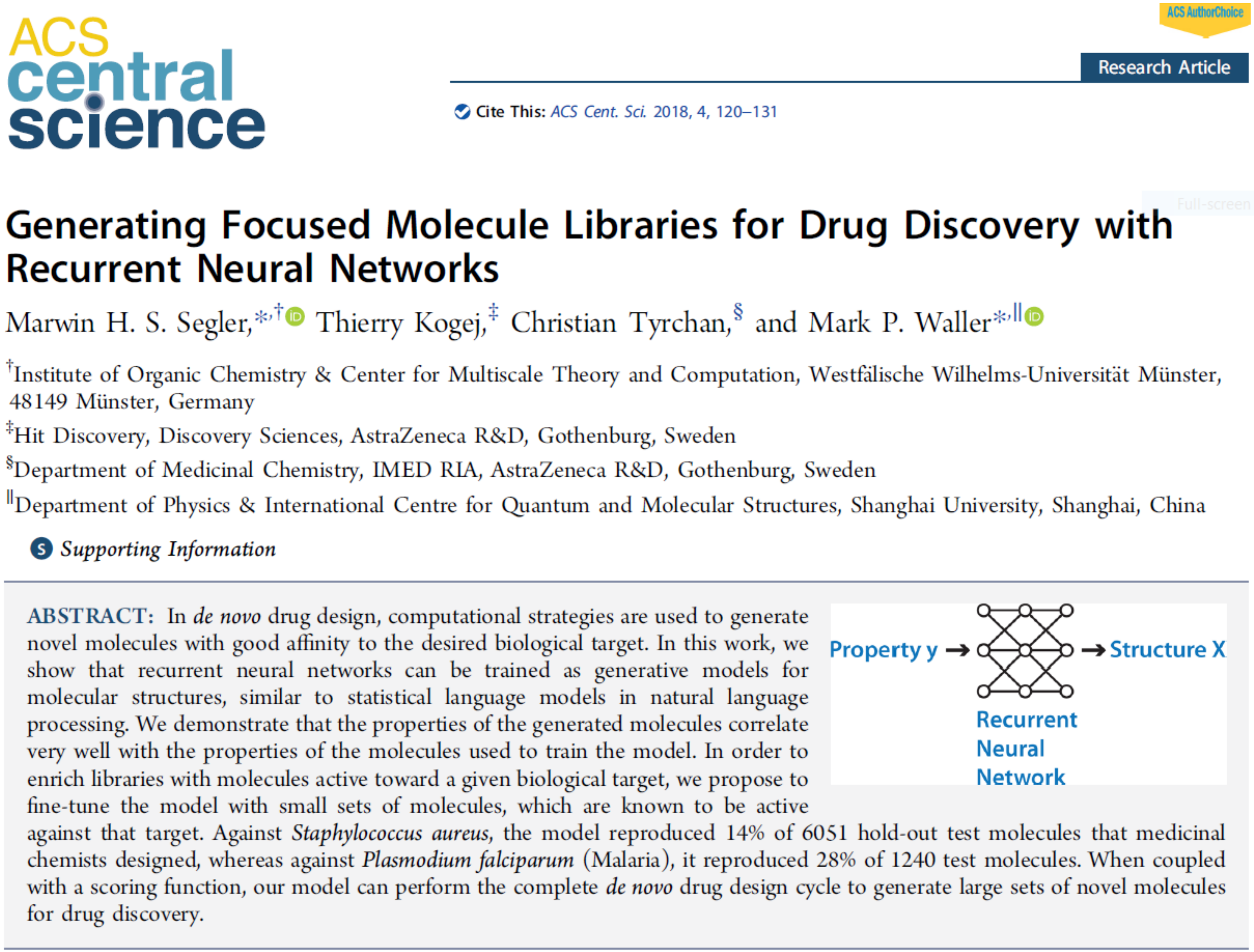
  Speech Signals

  Video Data

  SMILES (Simplified Molecular Input Line Entry System)



N1CCN(CC1)C(C(F)=C2)=CC(=C2C4=O)N(C3CC3)C=C4C(=O)O



ACS central science

ACS AuthorChoice

Research Article

Cite This: ACS Cent. Sci. 2018, 4, 120−131

Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks

Marwin H. S. Segler,*[†] Thierry Kogej,[‡] Christian Tyrchan,[§] and Mark P. Waller*[||]

[†]Institute of Organic Chemistry & Center for Multiscale Theory and Computation, Westfälische Wilhelms-Universität Münster, 48149 Münster, Germany

[‡]Hit Discovery, Discovery Sciences, AstraZeneca R&D, Gothenburg, Sweden

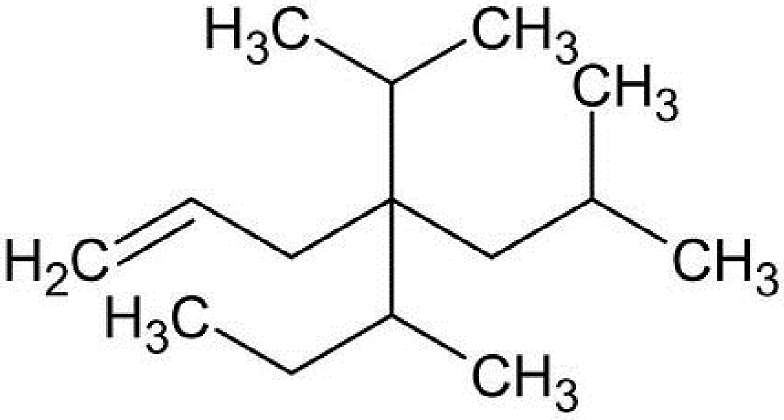
[§]Department of Medicinal Chemistry, IMED RIA, AstraZeneca R&D, Gothenburg, Sweden

[||]Department of Physics & International Centre for Quantum and Molecular Structures, Shanghai University, Shanghai, China

Supporting Information

ABSTRACT: In *de novo* drug design, computational strategies are used to generate novel molecules with good affinity to the desired biological target. In this work, we show that recurrent neural networks can be trained as generative models for molecular structures, similar to statistical language models in natural language processing. We demonstrate that the properties of the generated molecules correlate very well with the properties of the molecules used to train the model. In order to enrich libraries with molecules active toward a given biological target, we propose to fine-tune the model with small sets of molecules, which are known to be active against that target. Against *Staphylococcus aureus*, the model reproduced 14% of 6051 hold-out test molecules that medicinal chemists designed, whereas against *Plasmodium falciparum* (Malaria), it reproduced 28% of 1240 test molecules. When coupled with a scoring function, our model can perform the complete *de novo* drug design cycle to generate large sets of novel molecules for drug discovery.

1. **Valence Information**
2. **Implicit Hydrogens**
3. **Lack of 3D Geometry Information**
4. **Isomer Representation**



5-methyl-4-(2-methylpropyl)-4-(propan-2-yl)hept-1-ene

CC(C)CC(CC=C)(C(C)C)C(C)CC

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

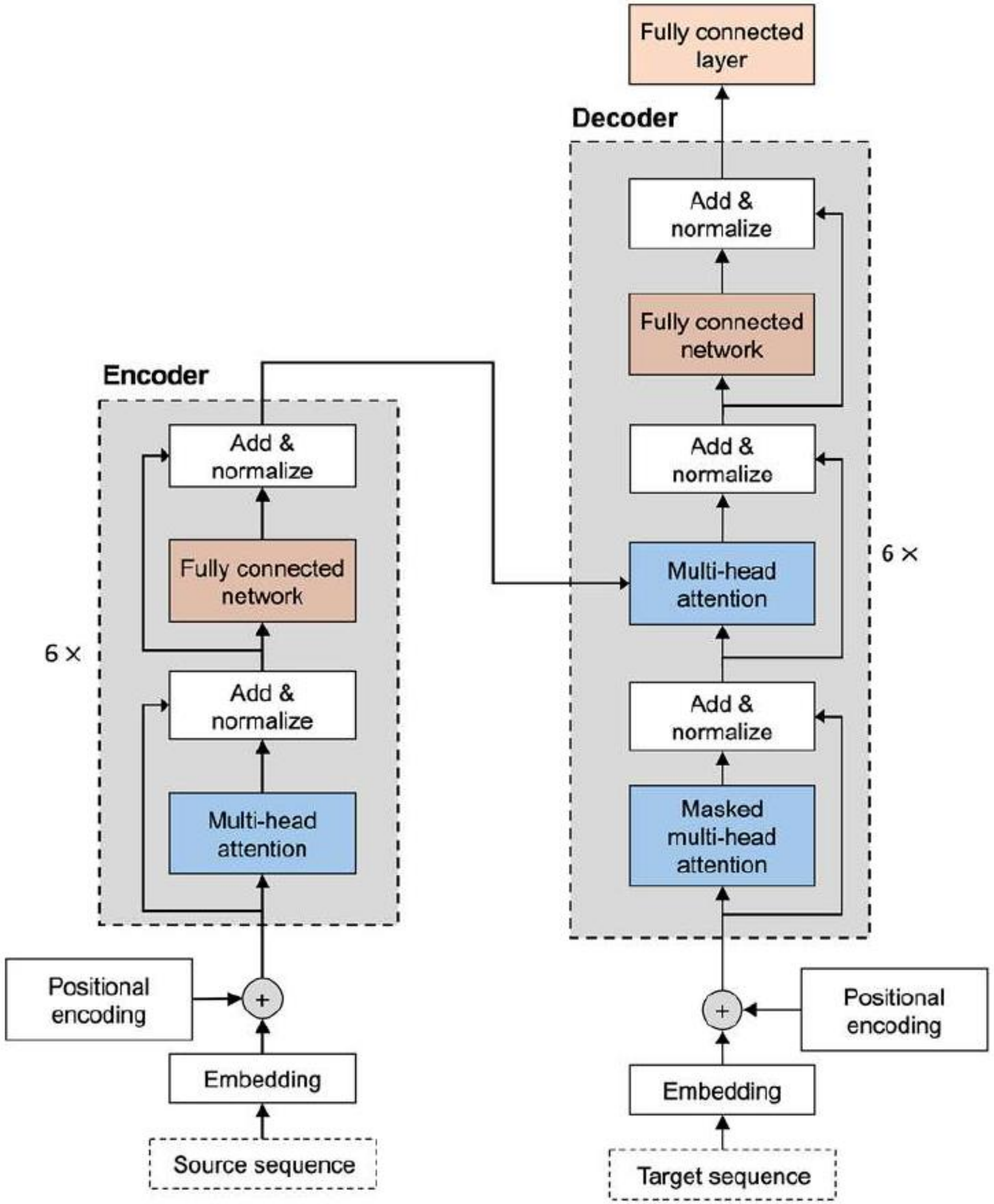


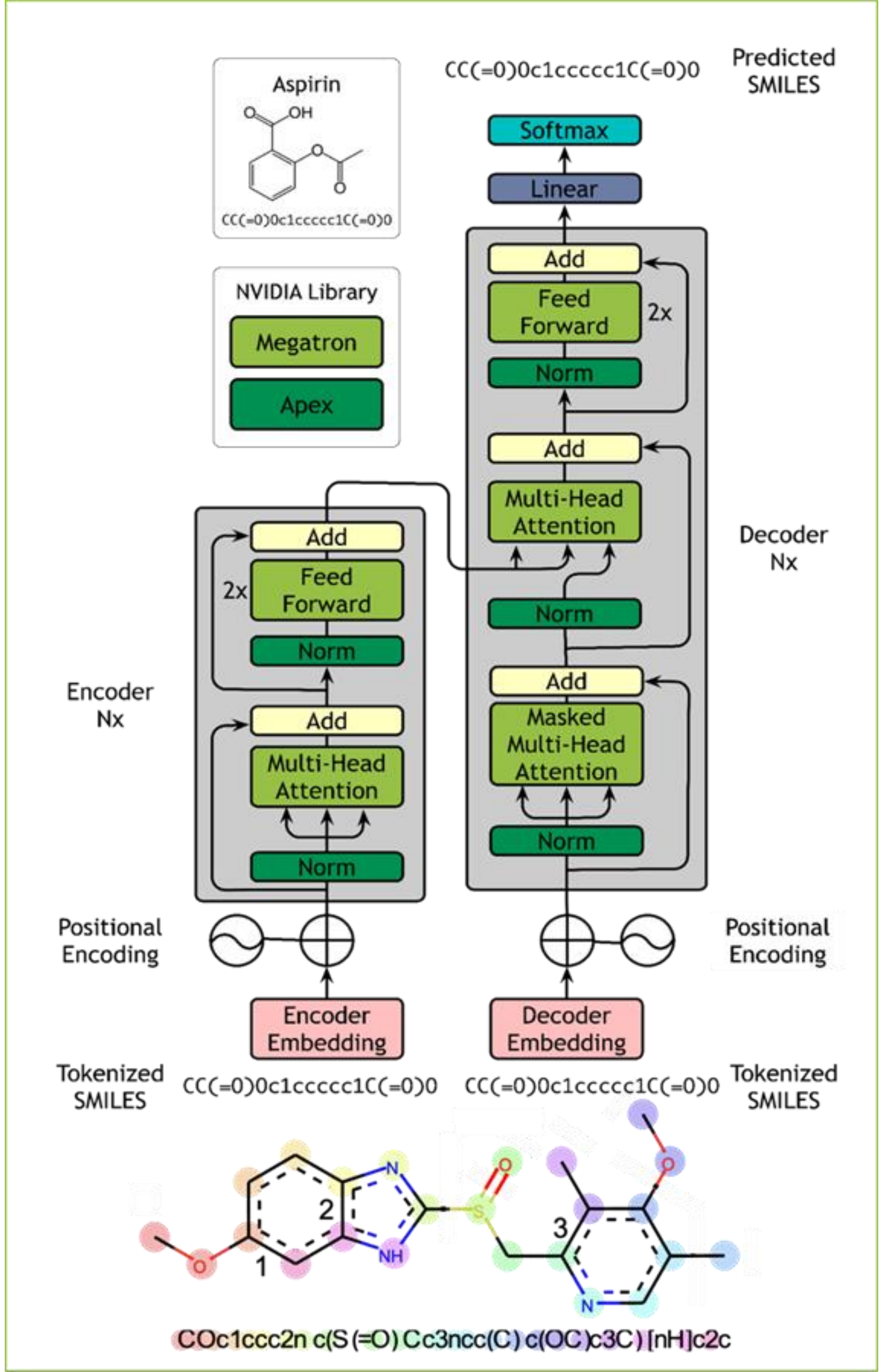
Figure 16.6: The original transformer architecture

1. **Generative Pre-trained Transformer (GPT)**
2. **Bidirectional Encoder Representations from Transformers (BERT)**
3. **Bidirectional and Auto-Regressive Transformer(BART):** The best of both worlds

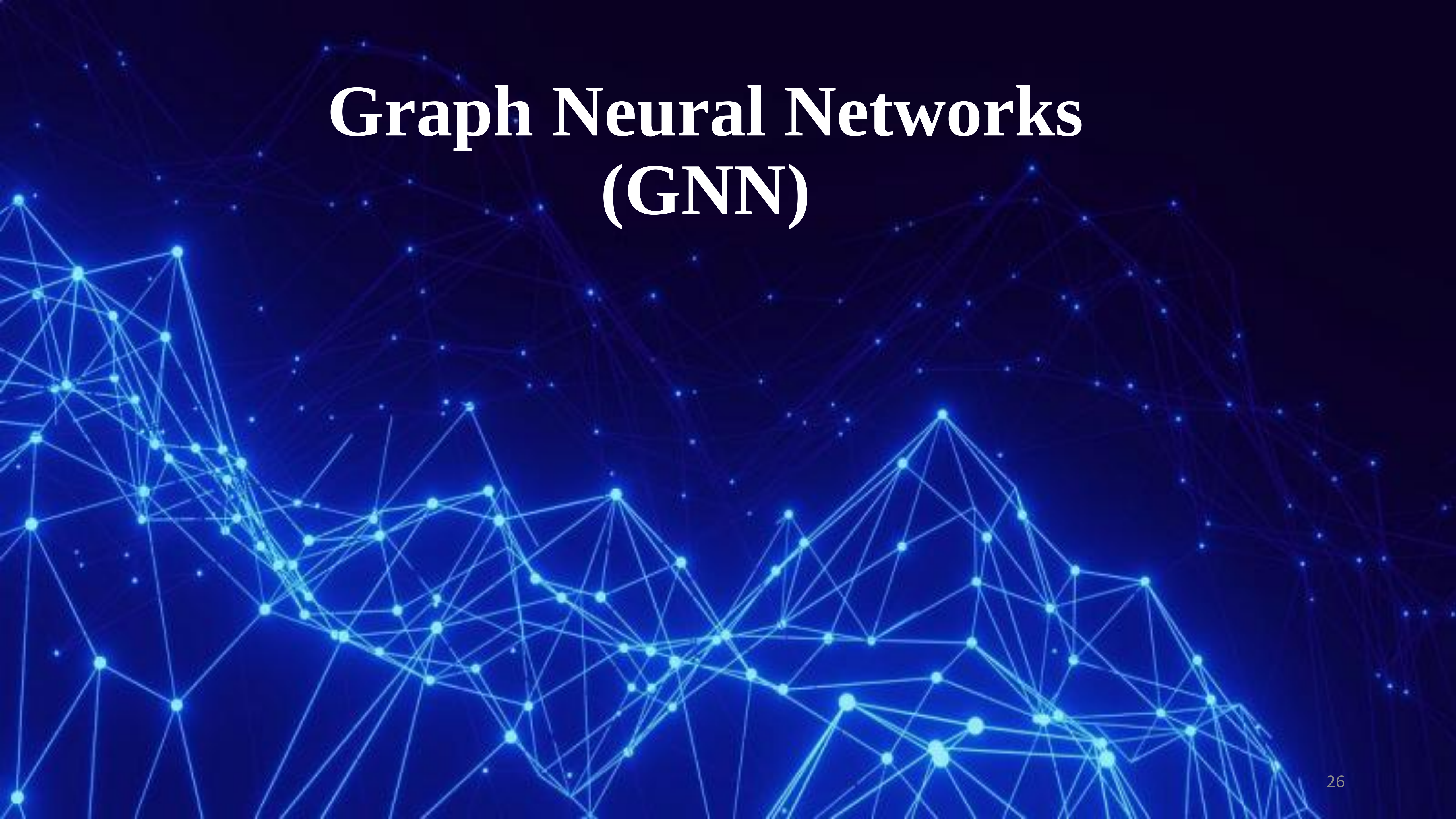
DrugGPT

MegaMolBART

- MegaMolBART molecular sequence, based upon known molecular sequences, is an autoencoder trained on small molecules in the form of SMILES that can be used for molecular representation tasks, molecule generation, and retrosynthesis. It was developed using the BioNeMo framework. MegaMolBART has eight layers, four attention heads, a hidden space dimension of 256, and contains 45M parameters. This model is ready for commercial/non-commercial use.


Aspirin
OH
O
O
O
CC(=O)Oc1ccccc1C(=O)O
NVIDIA Library
Megatron
Apex
CC(=O)Oc1ccccc1C(=O)O
Predicted SMILES
Softmax
Linear
Add
Feed Forward
2x
Norm
Add
Multi-Head Attention
Decoder Nx
Norm
Add
Masked Multi-Head Attention
Norm
Add
Feed Forward
2x
Norm
Encoder Nx
Add
Multi-Head Attention
Norm
Positional Encoding
Positional Encoding
Encoder Embedding
Decoder Embedding
Tokenized SMILES
CC(=O)Oc1ccccc1C(=O)O
CC(=O)Oc1ccccc1C(=O)O
Tokenized SMILES
N
O
O
2
S
3
O
NH
1
N
COc1ccc2n c(S(=O)Cc3ncc(C)c(OC)c3C)[nH]c2c

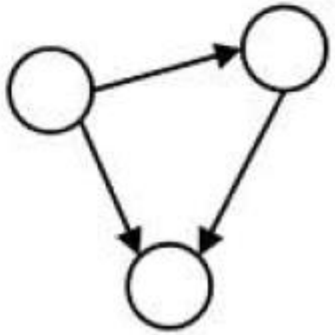
Graph Neural Networks (GNN)



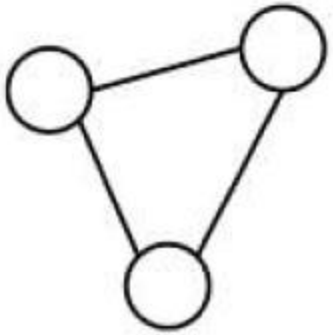
- **GNNs** have been an area of rapid development in recent years. According to the **State of AI report from 2021**, GNNs have evolved **“from niche to the hottest fields of AI research.”**

- **GNNs** have been applied in a variety of areas, including the following:
    - Text classification (https://arxiv.org/abs/1710.10903)
    - Recommender systems (https://arxiv.org/abs/1704.06803)
    - Traffic forecasting (https://arxiv.org/abs/1707.01926)
    - **Drug Discovery (https://arxiv.org/abs/1806.02473**

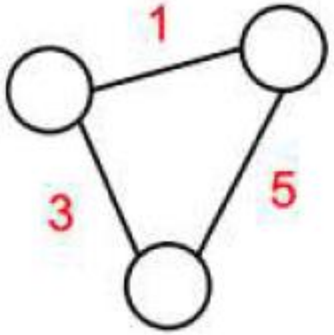
Directed graph



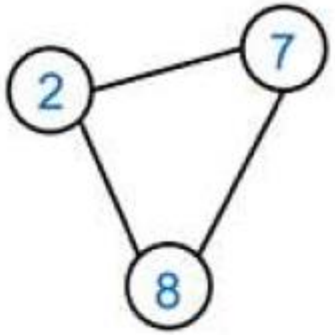
Undirected graph



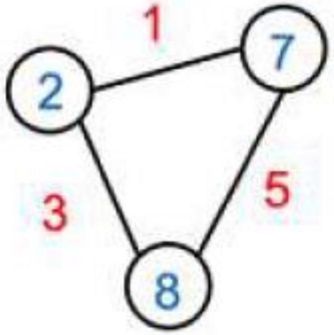
Edge-labeled undirected graph



Node-labeled undirected graph



Node- and edge-labeled undirected graph



Dynamic vs Static Graphs

Temporal Nature:

Static Graphs: Static graphs represent a snapshot of a network or system at a single point in time. They do not capture changes or interactions over time.

Dynamic Graphs: Dynamic graphs explicitly capture changes and interactions over time. They consist of multiple snapshots (timestamps), and edges or nodes can appear, disappear, or change attributes between snapshots.

Dynamic vs Static Graphs

Use Cases:

Static Graphs: Static graphs are suitable for modeling systems or networks that do not change or evolve significantly over time. They are commonly used for social networks, citation networks, and many other applications where the underlying structure remains relatively constant.

Dynamic Graphs: Dynamic graphs are used when modeling systems or networks that exhibit temporal dependencies, where interactions, events, or relationships change over time. Examples include communication networks, transportation systems, and epidemiological models.

Dynamic vs Static Graphs

Representation:

Static Graphs: Static graphs are typically represented using a single adjacency matrix, where each entry represents the presence or absence of an edge between two nodes. Node and edge attributes are constant.

Dynamic Graphs: Dynamic graphs are represented as a sequence of static graphs, each associated with a specific timestamp. Edges and nodes can have associated timestamps and attributes that evolve over time.

Dynamic vs Static Graphs

Analytical Challenges:

Static Graphs: Analyzing static graphs is often simpler, as they do not involve temporal dynamics. Traditional graph algorithms and metrics are commonly applied.

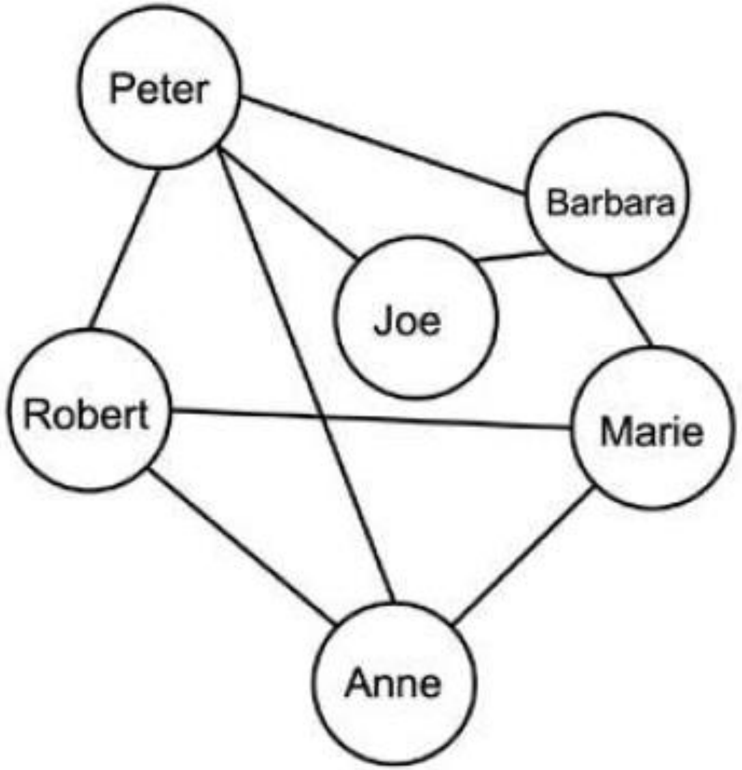
Dynamic Graphs: Analyzing dynamic graphs can be more complex due to the need to consider temporal aspects. Researchers use specialized algorithms for tasks like tracking node or edge changes, detecting patterns over time, and predicting future states.

Dynamic vs Static Graphs

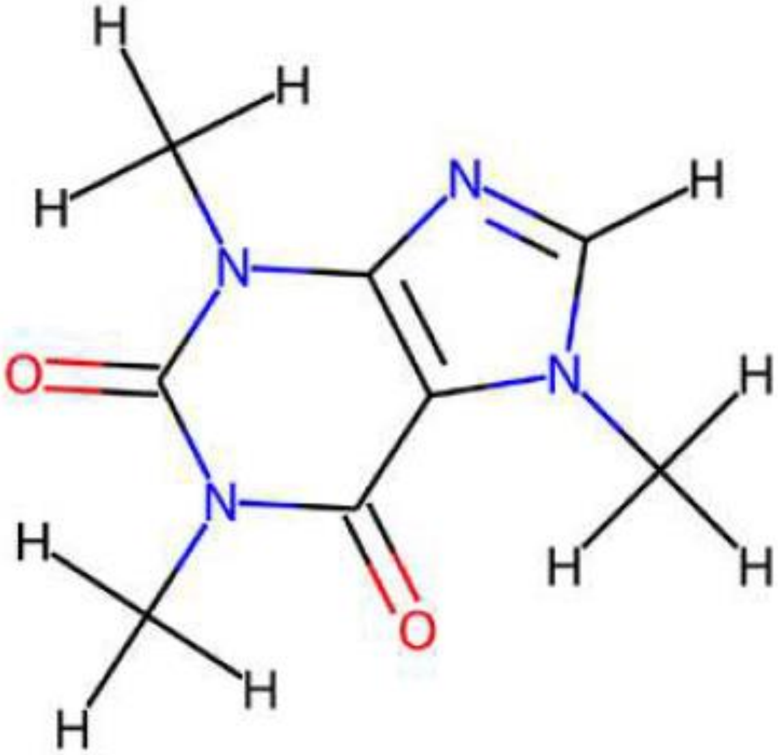
Storage and Processing:

Static Graphs: Storing and processing static graphs are often more straightforward since the graph structure remains constant.

Dynamic Graphs: Handling dynamic graphs requires more advanced data structures and algorithms to efficiently manage changes over time.

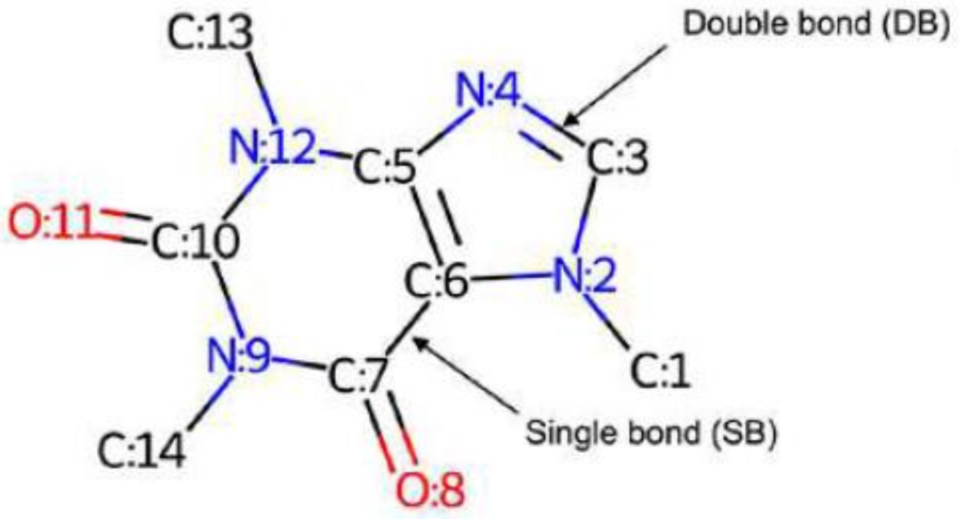


Friend graph



Molecular graph
of caffeine

Caffeine molecule



$$A = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Adjacency matrix

One-hot encoding of the atom type

$$X = \begin{array}{c}
\begin{array}{ccc} \mathbf{C} & \mathbf{N} & \mathbf{O} \end{array} \\
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 0 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{bmatrix}
\end{array}$$

Node label matrix

One-hot encoding of the bond type

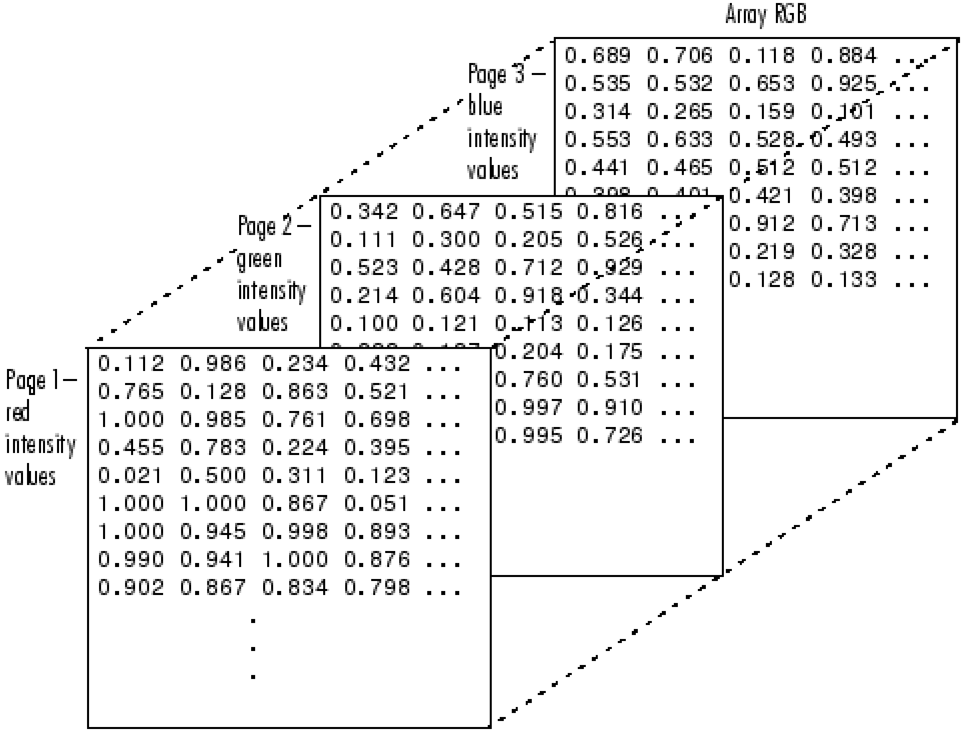
$$X_E = \begin{array}{c}
 \\
(1,2) \\ (2,3) \\ (2,6) \\ (3,4) \\ (4,5) \\ (5,6) \\ (5,12) \\ (6,7) \\ (7,8) \\ (7,9) \\ (9,10) \\ (9,14) \\ (10,11) \\ (10,12) \\ (12,13)
\end{array}
\begin{array}{c}
\begin{array}{cc} \mathbf{SB} & \mathbf{DB} \end{array} \\
\begin{bmatrix}
1 & 0 \\
1 & 0 \\
1 & 0 \\
0 & 1 \\
1 & 0 \\
0 & 1 \\
1 & 0 \\
1 & 0 \\
0 & 1 \\
1 & 0 \\
1 & 0 \\
1 & 0 \\
0 & 1 \\
1 & 0 \\
1 & 0
\end{bmatrix}
\end{array}$$

Edge label matrix

```
Product C:
[[[ 57  21  48   0]
  [ 71  27  60   0]
  [111  51  96   0]]

 [[ 66  16  36  18]
  [ 95  22  58  29]
  [ 83  20  46  23]]

 [[ 44  34  72  44]
  [ 34  32  72  34]
  [ 13  25  64  13]]], shape=(3, 3, 4)
```

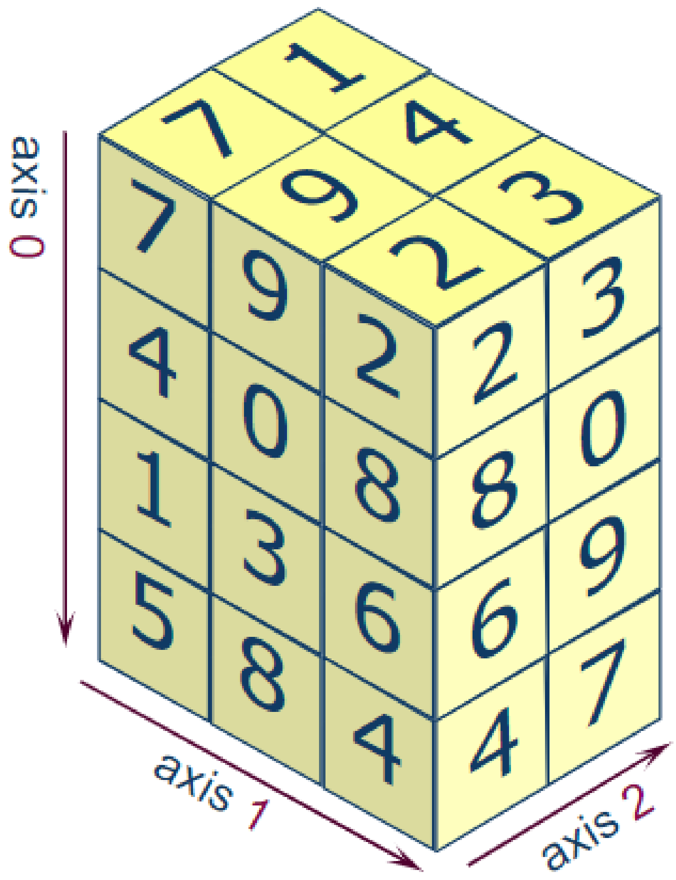

Array RGB
Page 3 – blue intensity values
Page 2 – green intensity values
Page 1 – red intensity values

```
0.689 0.706 0.118 0.884 ...
0.535 0.532 0.653 0.925 ...
0.314 0.265 0.159 0.101 ...
0.553 0.633 0.528 0.493 ...
0.441 0.465 0.512 0.512 ...
            0.421 0.398 ...
            0.912 0.713 ...
            0.219 0.328 ...
            0.128 0.133 ...
```

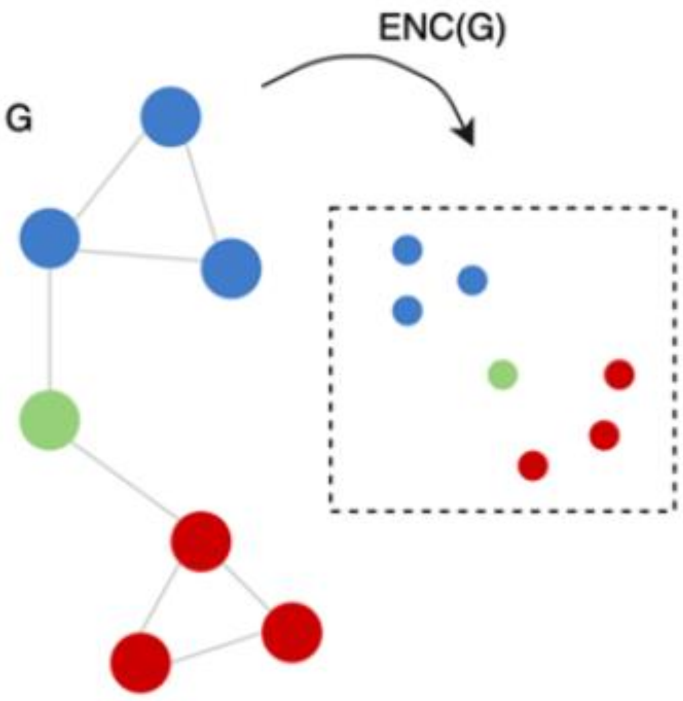
```
0.342 0.647 0.515 0.816 ...
0.111 0.300 0.205 0.526 ...
0.523 0.428 0.712 0.929 ...
0.214 0.604 0.918 0.344 ...
0.100 0.121 0.113 0.126 ...
            0.204 0.175 ...
            0.760 0.531 ...
            0.997 0.910 ...
            0.995 0.726 ...
```

```
0.112 0.986 0.234 0.432 ...
0.765 0.128 0.863 0.521 ...
1.000 0.985 0.761 0.698 ...
0.455 0.783 0.224 0.395 ...
0.021 0.500 0.311 0.123 ...
1.000 1.000 0.867 0.051 ...
1.000 0.945 0.998 0.893 ...
0.990 0.941 1.000 0.876 ...
0.902 0.867 0.834 0.798 ...
.
.
.
```

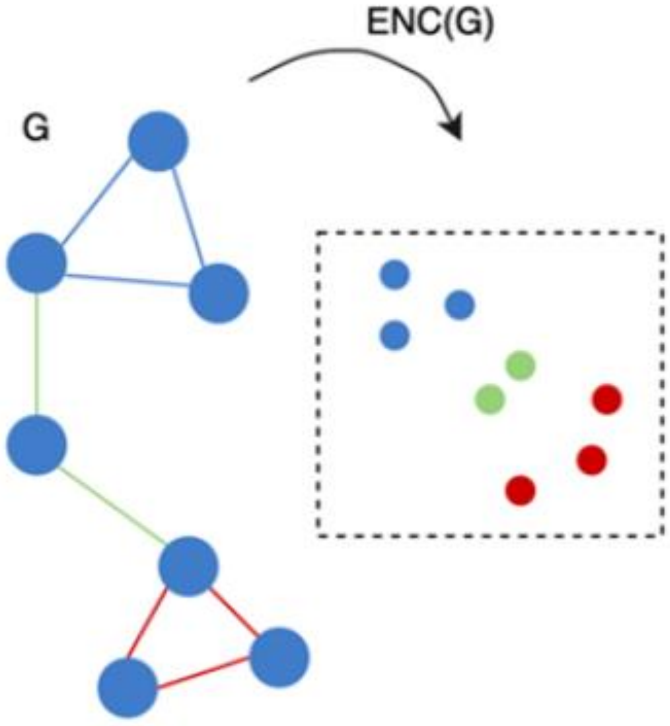
3D Array



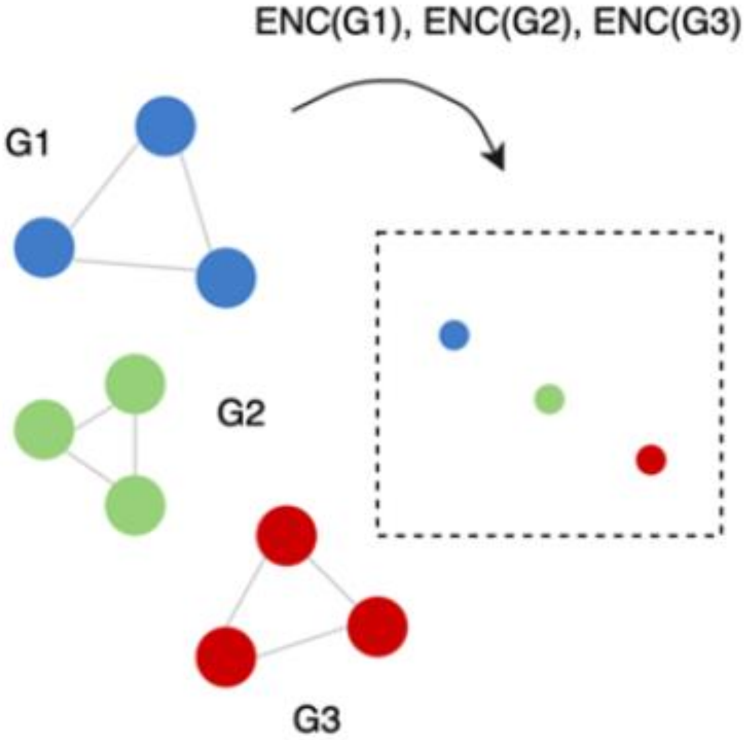
shape : (4, 3, 2)



Node-level embedding

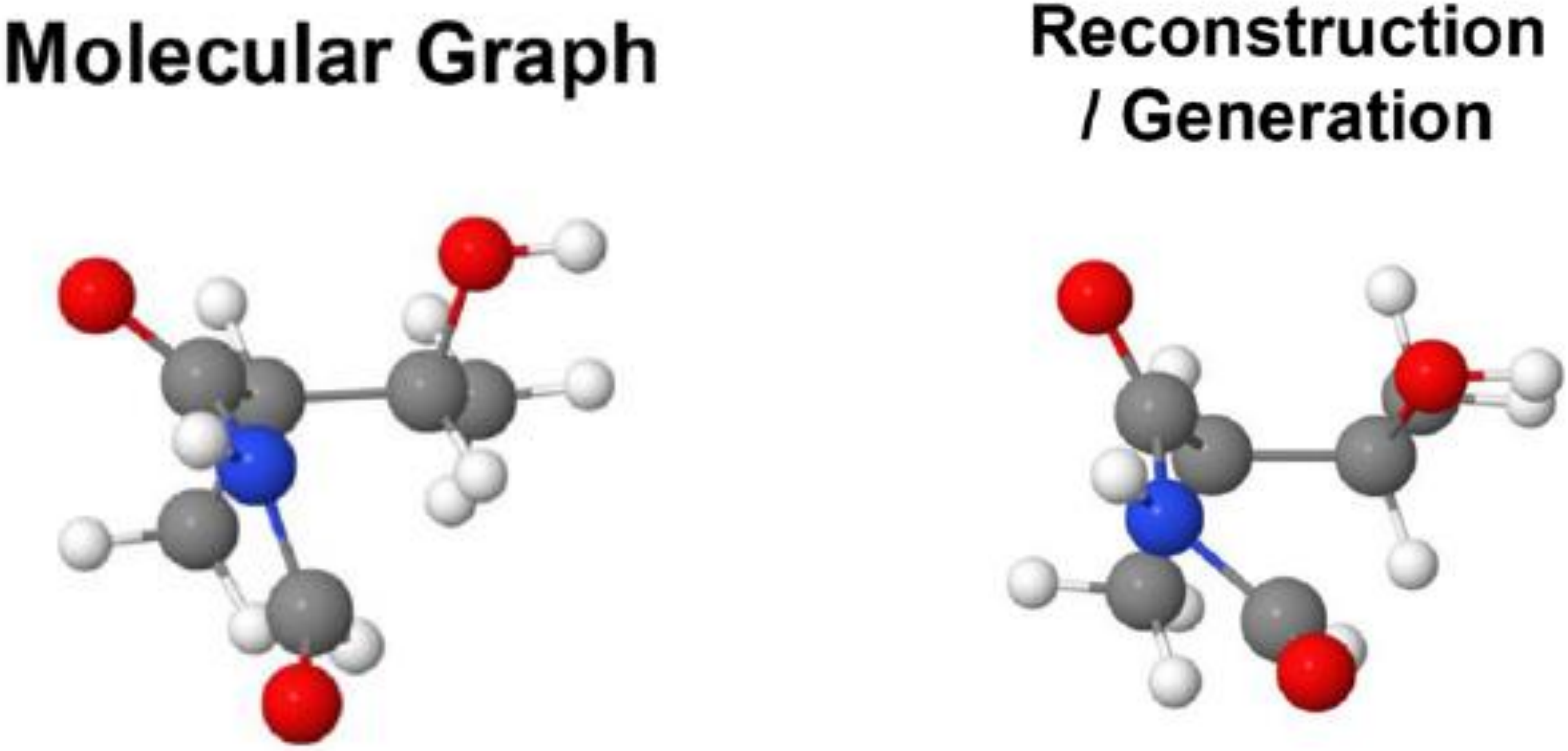


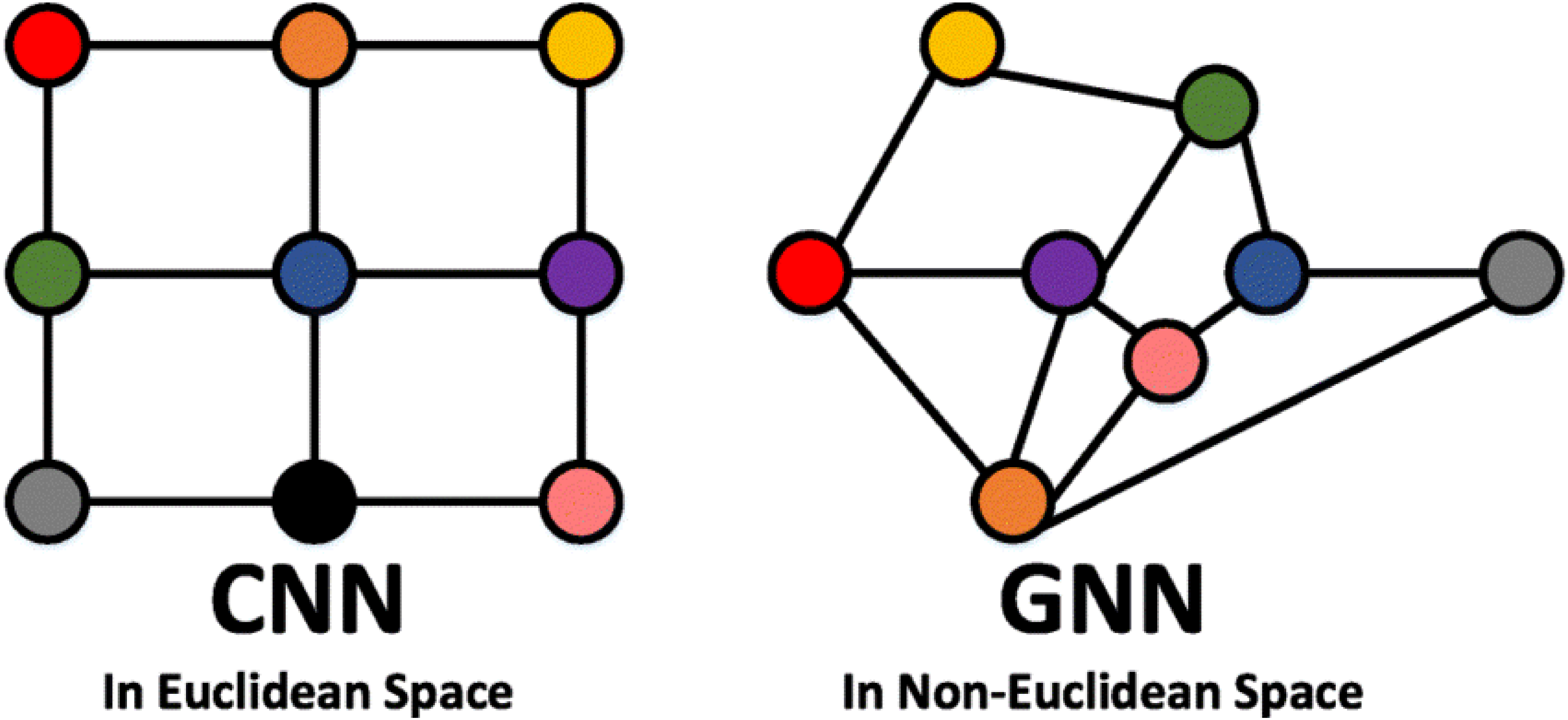
Edge-level embedding



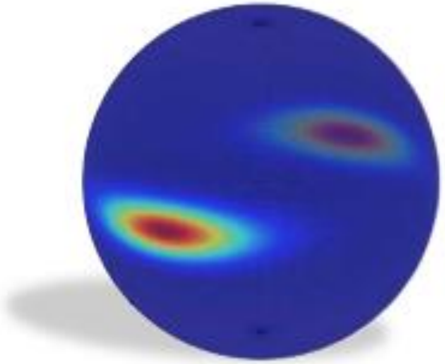
Graph-level embedding

Dynamic Graph




CNN
In Euclidean Space
GNN
In Non-Euclidean Space

Surfaces

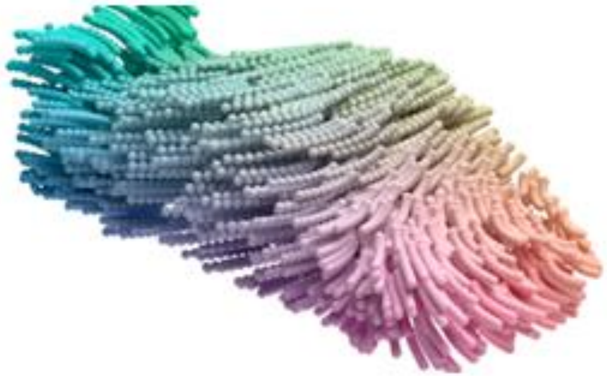


Distributions

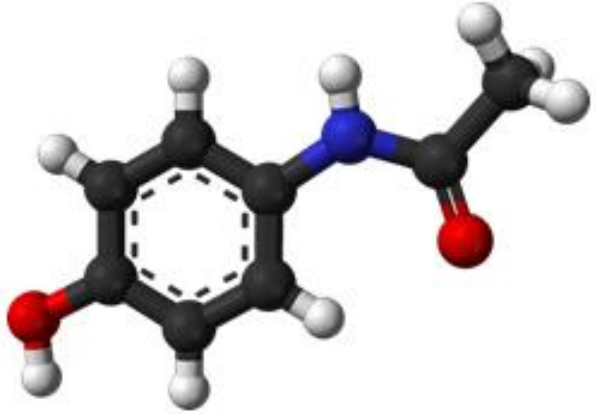
Graphs / Networks

Functions on Manifolds

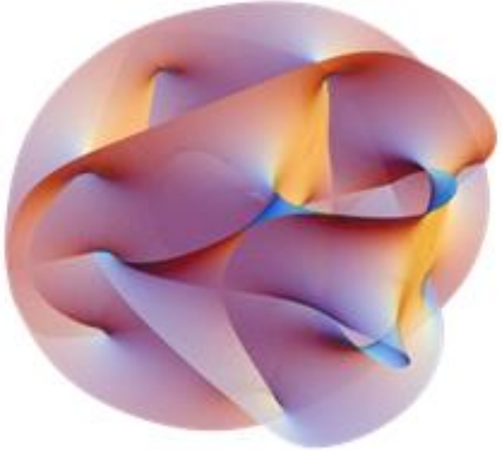
Hyperbolic spaces



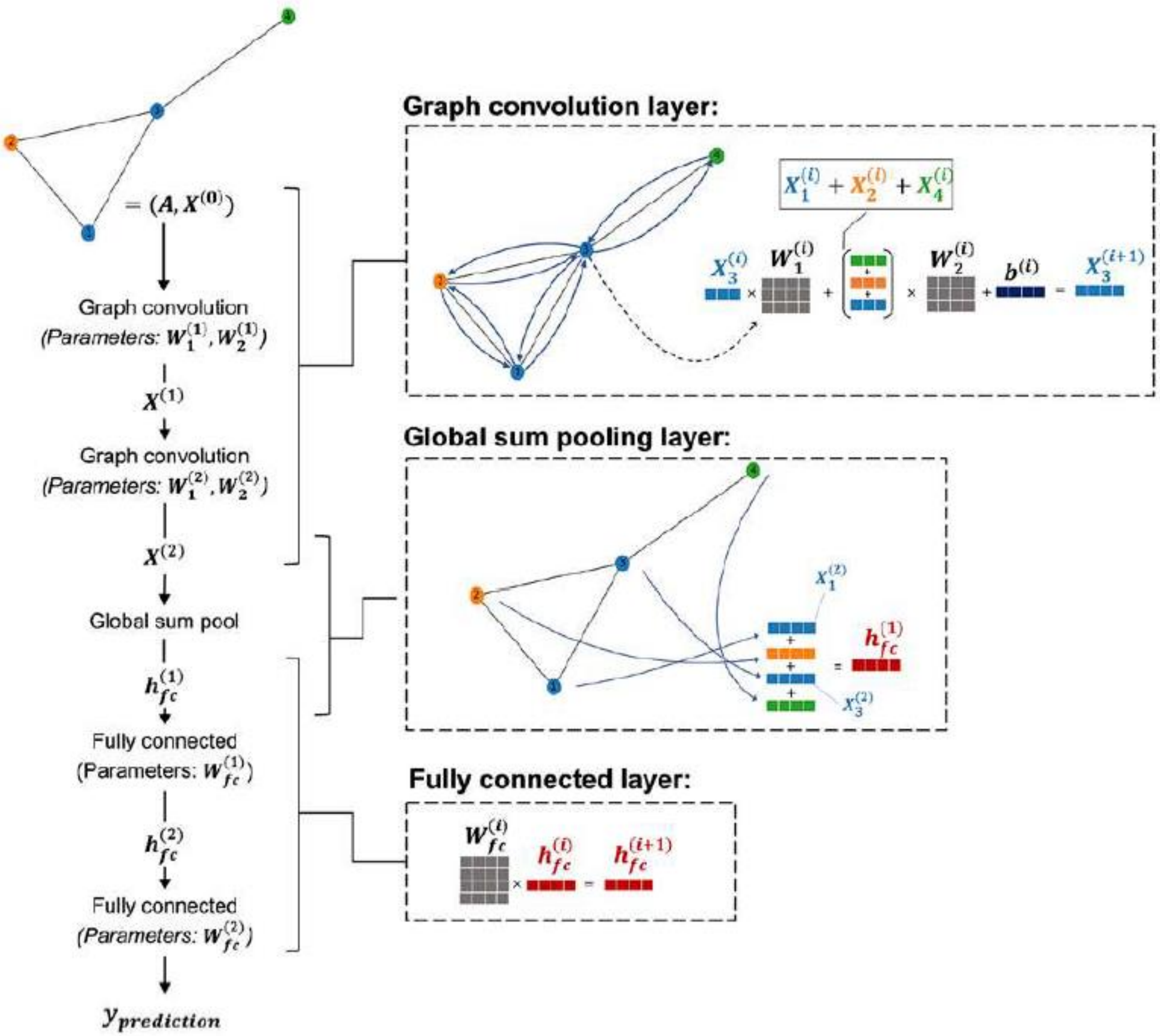
Hyper-surfaces

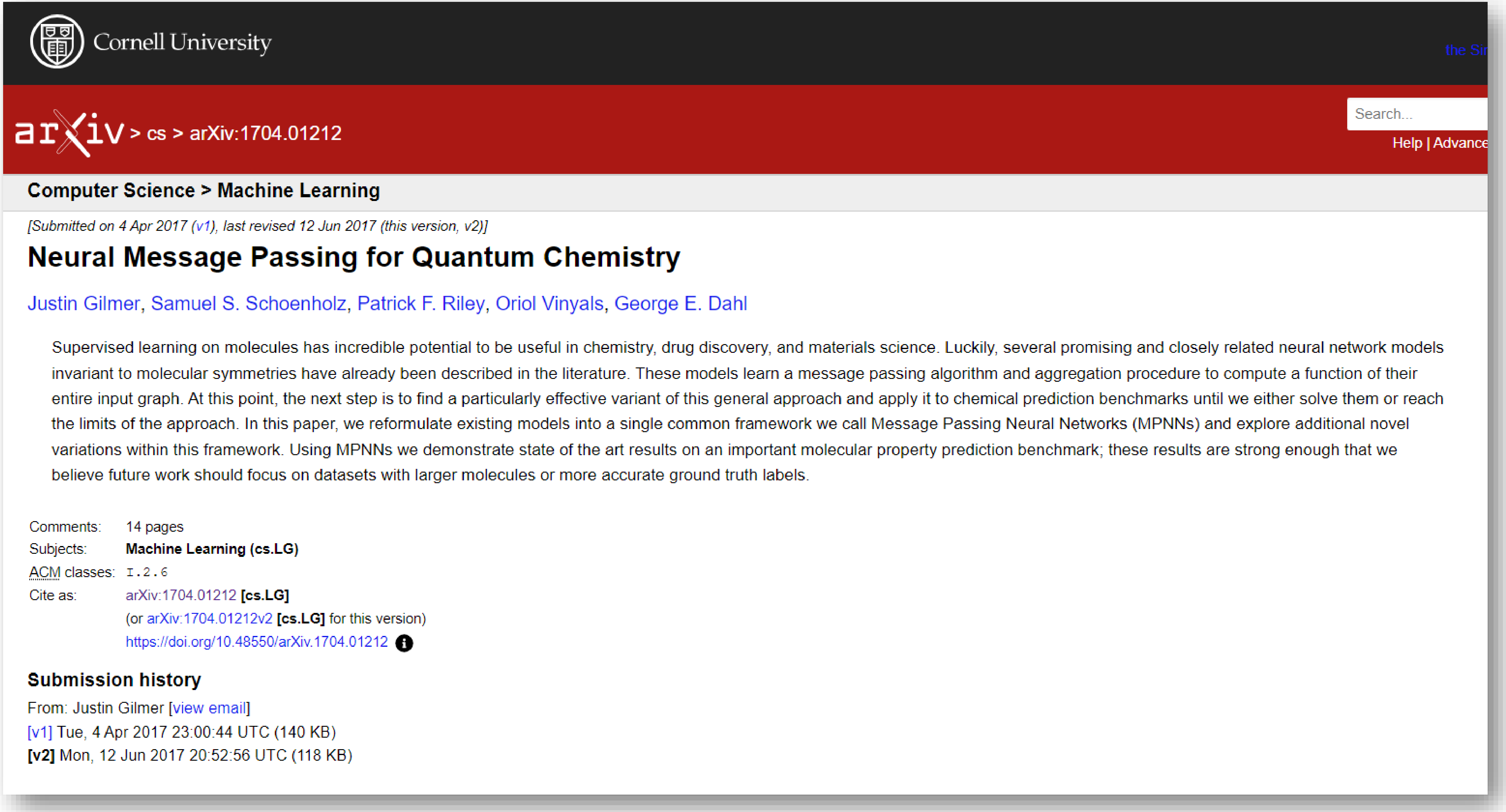


Molecules



General manifolds


$= (A, X^{(0)})$
Graph convolution
(Parameters: $W_1^{(1)}, W_2^{(1)}$)
$X^{(1)}$
Graph convolution
(Parameters: $W_1^{(2)}, W_2^{(2)}$)
$X^{(2)}$
Global sum pool
$h_{fc}^{(1)}$
Fully connected
(Parameters: $W_{fc}^{(1)}$)
$h_{fc}^{(2)}$
Fully connected
(Parameters: $W_{fc}^{(2)}$)
$y_{prediction}$
Graph convolution layer:
$X_1^{(i)} + X_2^{(i)} + X_4^{(i)}$
$X_3^{(i)} \times W_1^{(i)} + (\ldots) \times W_2^{(i)} + b^{(i)} = X_3^{(i+1)}$
Global sum pooling layer:
$X_1^{(2)}$
$X_3^{(2)}$
$= h_{fc}^{(1)}$
Fully connected layer:
$W_{fc}^{(i)} \times h_{fc}^{(i)} = h_{fc}^{(i+1)}$



arXiv > cs > arXiv:1704.01212

Computer Science > Machine Learning

[Submitted on 4 Apr 2017 (v1), last revised 12 Jun 2017 (this version, v2)]

Neural Message Passing for Quantum Chemistry

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, George E. Dahl

Supervised learning on molecules has incredible potential to be useful in chemistry, drug discovery, and materials science. Luckily, several promising and closely related neural network models invariant to molecular symmetries have already been described in the literature. These models learn a message passing algorithm and aggregation procedure to compute a function of their entire input graph. At this point, the next step is to find a particularly effective variant of this general approach and apply it to chemical prediction benchmarks until we either solve them or reach the limits of the approach. In this paper, we reformulate existing models into a single common framework we call Message Passing Neural Networks (MPNNs) and explore additional novel variations within this framework. Using MPNNs we demonstrate state of the art results on an important molecular property prediction benchmark; these results are strong enough that we believe future work should focus on datasets with larger molecules or more accurate ground truth labels.

Comments:	14 pages
Subjects:	**Machine Learning (cs.LG)**
ACM classes:	I.2.6
Cite as:	arXiv:1704.01212 **[cs.LG]**
	(or arXiv:1704.01212v2 **[cs.LG]** for this version)
	https://doi.org/10.48550/arXiv.1704.01212

Submission history

From: Justin Gilmer [view email]
[v1] Tue, 4 Apr 2017 23:00:44 UTC (140 KB)
[v2] Mon, 12 Jun 2017 20:52:56 UTC (118 KB)

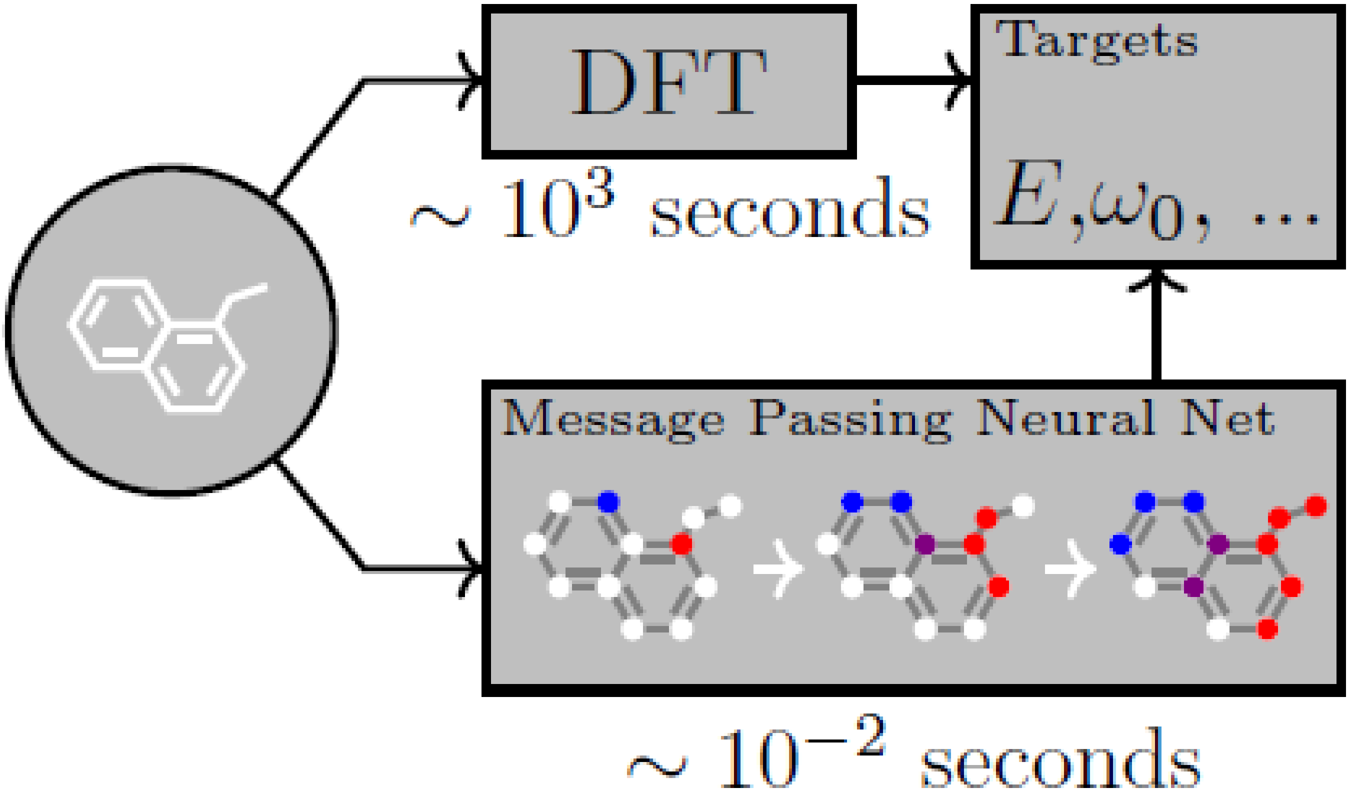


Figure 1. A Message Passing Neural Network predicts quantum properties of an organic molecule by modeling a computationally expensive DFT calculation.

Open-Source Cheminformatics
and Machine Learning

RDKit

1. *Molecule Creation*
2. *Property Calculation*
3. *Chemical Reactions*
4. *Molecular fingerprints*
5. *3D Conformer Generation*
6. *QSAR modeling*
7. *Substructure Searching*
8. *Force Field Optimization*
9. *Pharmacophore Modeling*
10. *Constrained Embedding*
11. *Shape and Volume Calculations*
12. *…*

RDKit Chem Module : Atom-Level Functions

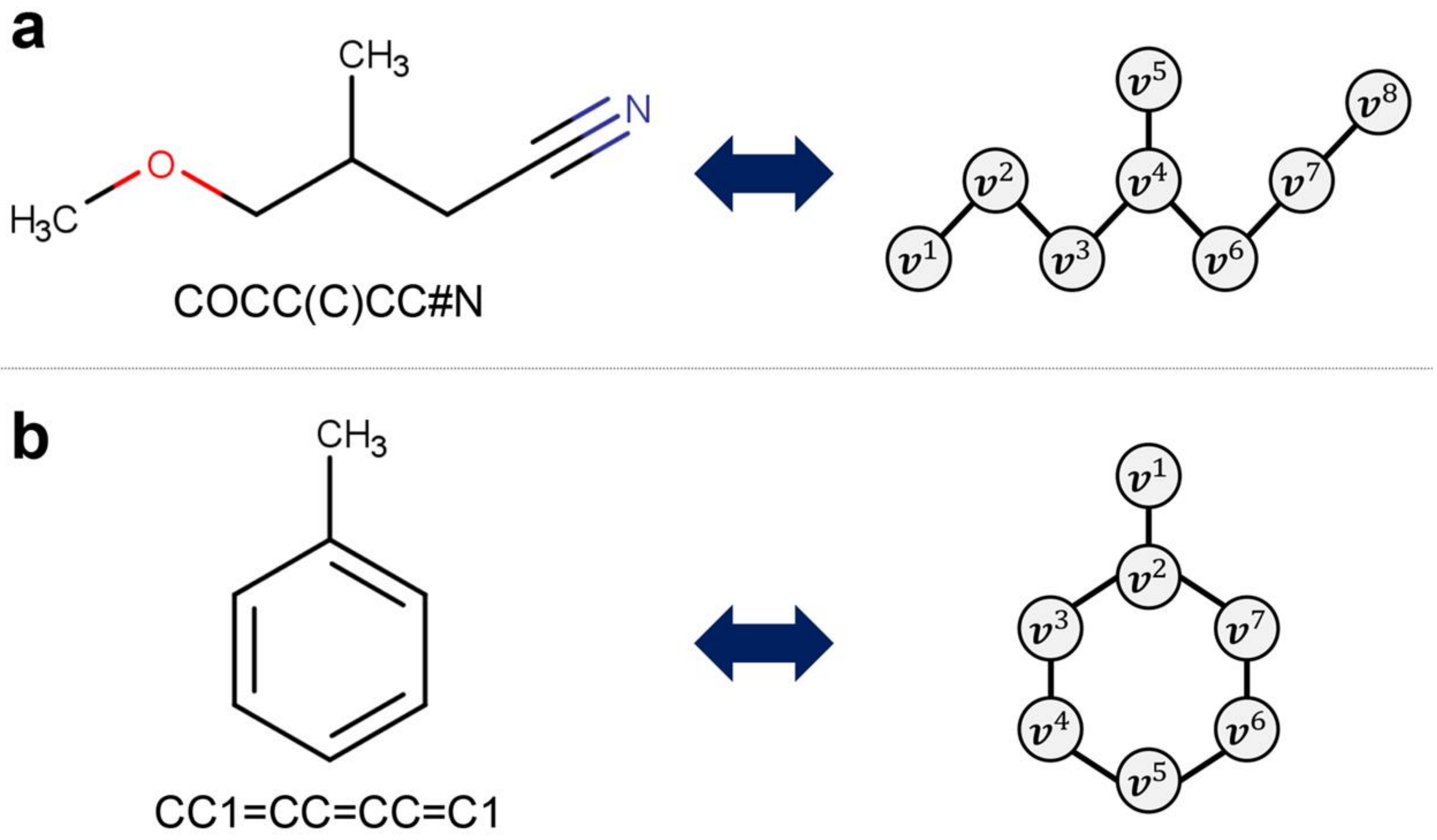
- **atom.GetSymbol()**: Returns the atomic symbol (e.g., "C", "O").
- **atom.GetChiralTag()**: Retrieves the atom's chirality (R/S or achiral).
- **atom.GetTotalDegree()**: Gets the total degree of the atom, including bonds to hydrogen.
- **atom.GetFormalCharge()**: Provides the atom's formal charge.
- **atom.GetTotalNumHs()**: Counts explicit and implicit hydrogens on the atom.
- **atom.GetNumRadicalElectrons()**: Returns the number of radical electrons.
- **atom.GetHybridization()**: Retrieves the atom's hybridization (e.g., sp2, sp3).
- **atom.GetIsAromatic()**: Checks if the atom is aromatic.
- **atom.IsInRing()**: Determines if the atom is part of a ring.
- **atom.GetOwningMol()**: Retrieves the molecule object that owns the atom.

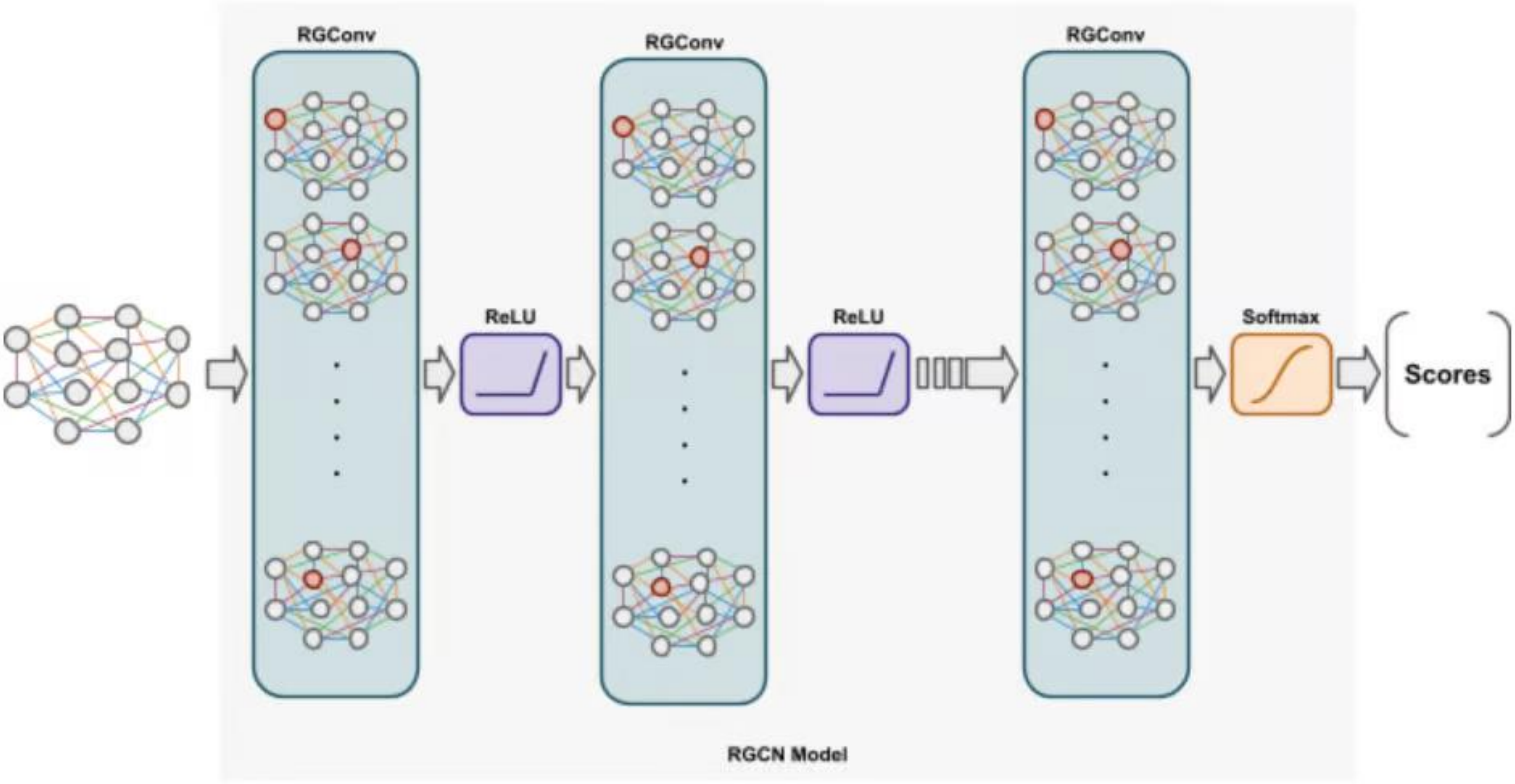
RDKit Chem Module : Bond-Level Functions

- **bond.GetBondType()**: Gets the bond type (single, double, triple, aromatic).
- **bond.GetBondDir()**: Retrieves the bond direction (e.g., BEGINWEDGE).
- **bond.GetStereo()**: Provides stereochemistry information (cis/trans).
- **bond.GetIsConjugated()**: Checks if the bond is conjugated.
- **bond.IsInRing()**: Determines if the bond is part of a ring.
- **bond.GetBeginAtomIdx() and bond.GetEndAtomIdx()**: Fetch the indices of the bonded atoms.
- **bond.GetOwningMol()**: Retrieves the molecule object that owns the bond.

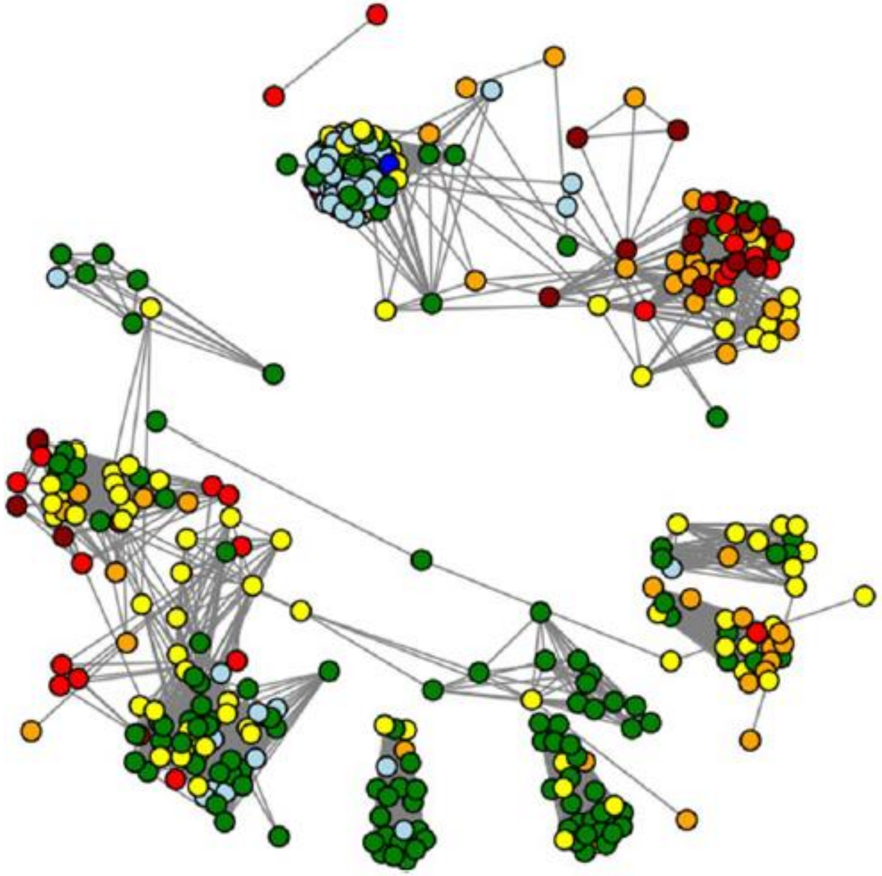
RDKit Chem Module : Molecule-Level Functions

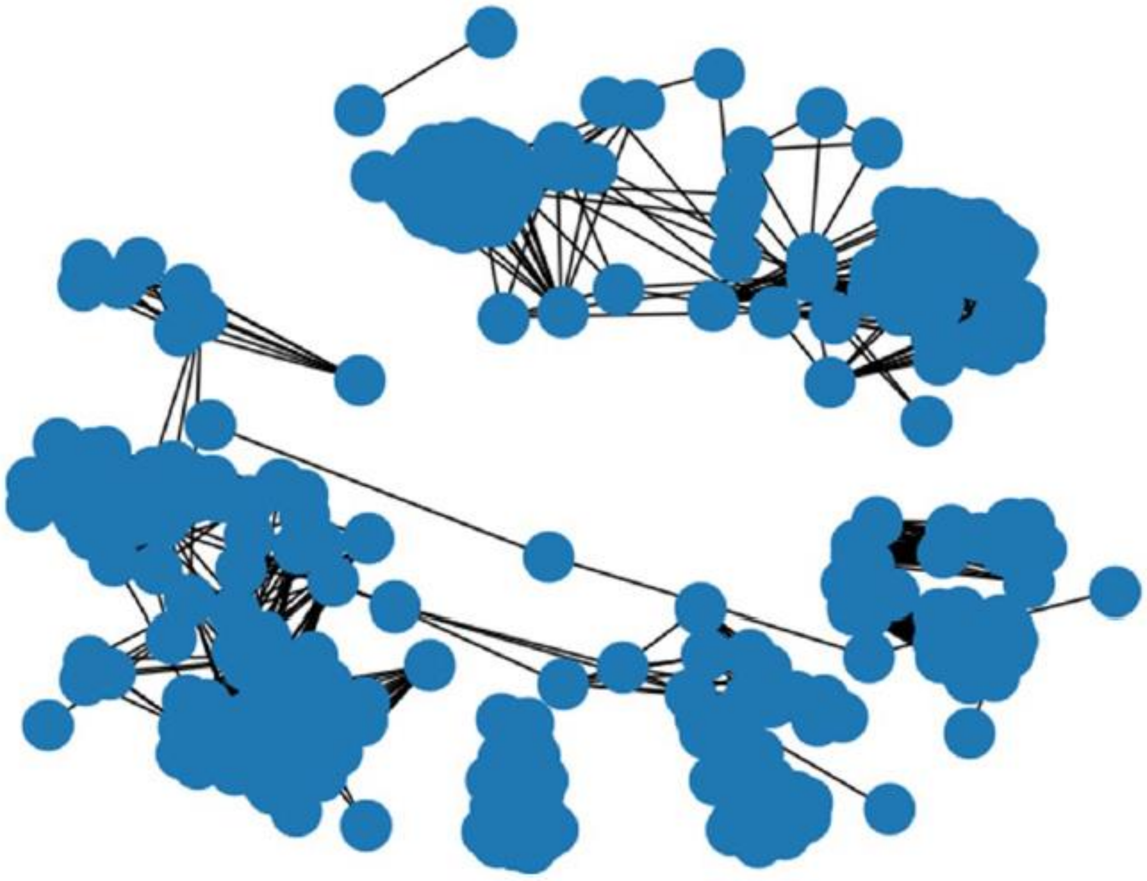
- **mol.GetConformer()**: Retrieves the conformer for accessing 3D positions.
- **mol.GetNumConformers()**: Checks if the molecule has any conformers.
- **mol.Compute2DCoords()**: Generates 2D coordinates for the molecule.


a
CH3
O
N
H3C
COCC(C)CC#N
v^1
v^2
v^3
v^4
v^5
v^6
v^7
v^8
b
CH3
CC1=CC=CC=C1
v^1
v^2
v^3
v^4
v^5
v^6
v^7

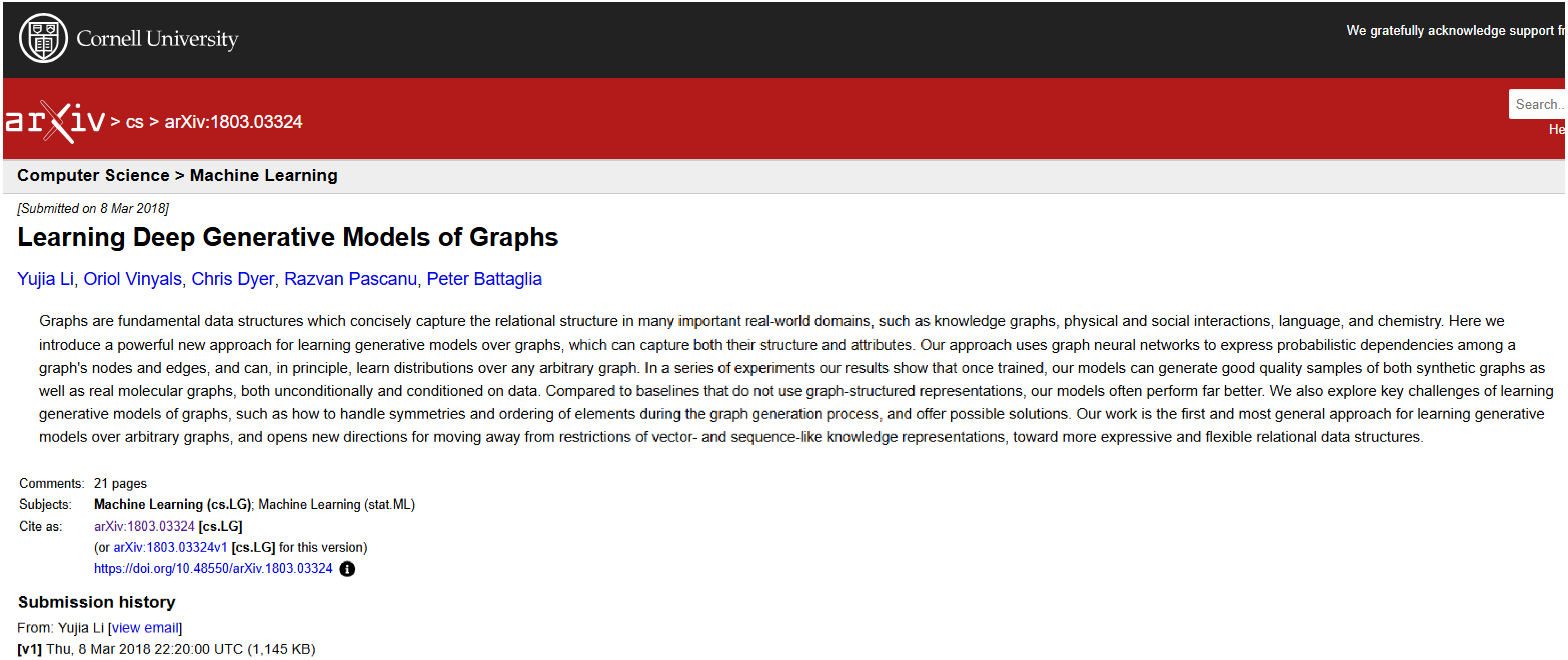

RGConv
RGConv
RGConv
ReLU
ReLU
Softmax
Scores
RGCN Model

Unified Graph: N-Dimensional Non-Linear Probabilistic Graph Model





Model	Pros	Cons / Limitations
GCN (Graph Convolutional Network)	Fast, scalable; good for basic property prediction.	No edge/bond awareness; poor long-range expressiveness; over-smoothing in deep layers.
***MPNN** (Message Passing Neural Network)	Accurate in molecular tasks; good chemical realism.	Higher computational cost; complex architecture tuning.
GAT (Graph Attention Network)	Interpretable; effective on heterogeneous graphs.	Attention overhead increases complexity; limited scalability to large graphs.
***RGCN** (Relational GCN)	Ideal for molecular graphs with heterogeneous bonds.	Model size grows with relation types; overfitting risk in small datasets.
GIN (Graph Isomorphism Network)	Strong representational capacity; good for classification.	Ignores edge types; overfitting possible on small data; less interpretable.
GraphSAGE	Works well on large molecular graphs; generalizable to unseen data.	Less sensitive to chemical structure; weaker granularity in molecular context.
***D-MPNN** (Directed MPNN)	More chemically accurate; great for reaction/property prediction.	More complex implementation; requires directed graph data.
ChebNet	Efficient and deeper local context modeling.	Less intuitive than spatial GCNs; not widely adopted in cheminformatics.



Cornell University

We gratefully acknowledge support fr

arXiv > cs > arXiv:1803.03324

Search..

He

Computer Science > Machine Learning

[Submitted on 8 Mar 2018]

Learning Deep Generative Models of Graphs

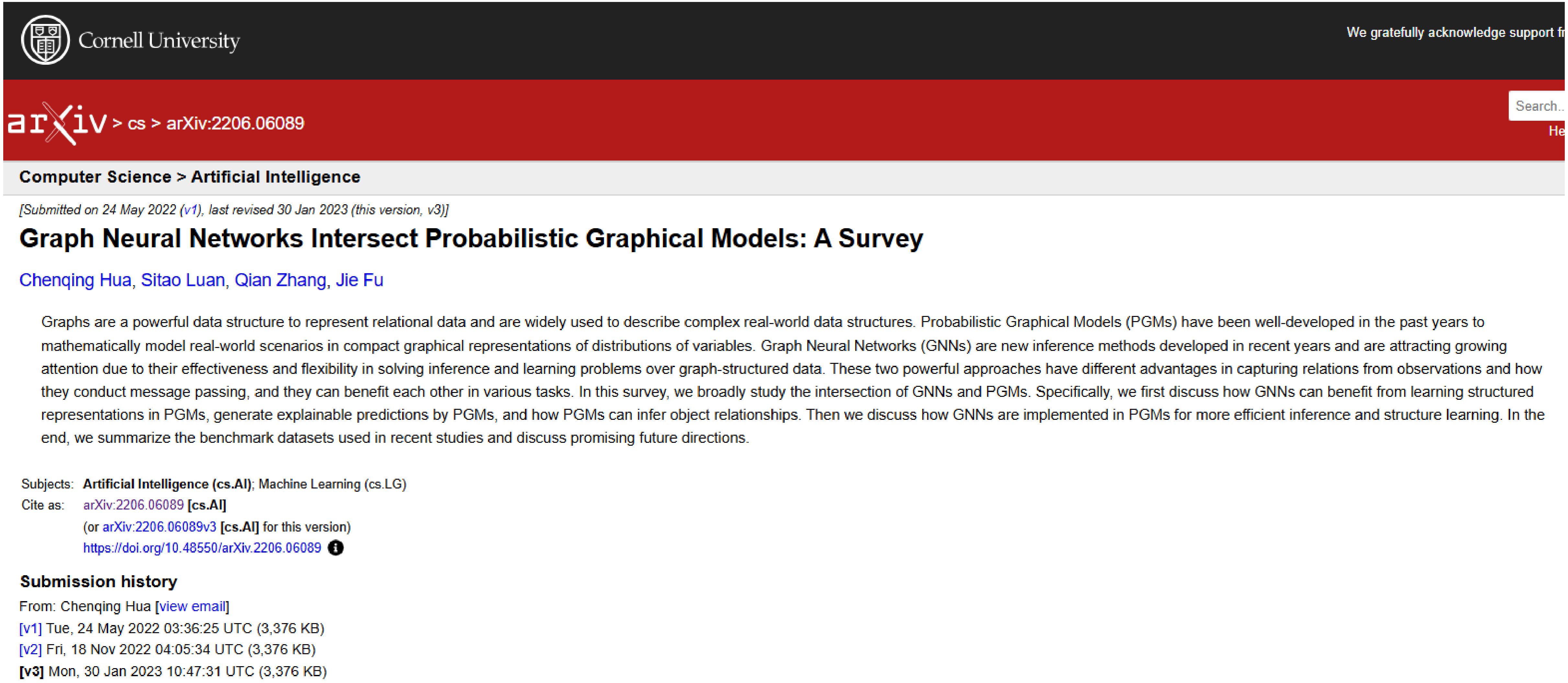
Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, Peter Battaglia

Graphs are fundamental data structures which concisely capture the relational structure in many important real-world domains, such as knowledge graphs, physical and social interactions, language, and chemistry. Here we introduce a powerful new approach for learning generative models over graphs, which can capture both their structure and attributes. Our approach uses graph neural networks to express probabilistic dependencies among a graph's nodes and edges, and can, in principle, learn distributions over any arbitrary graph. In a series of experiments our results show that once trained, our models can generate good quality samples of both synthetic graphs as well as real molecular graphs, both unconditionally and conditioned on data. Compared to baselines that do not use graph-structured representations, our models often perform far better. We also explore key challenges of learning generative models of graphs, such as how to handle symmetries and ordering of elements during the graph generation process, and offer possible solutions. Our work is the first and most general approach for learning generative models over arbitrary graphs, and opens new directions for moving away from restrictions of vector- and sequence-like knowledge representations, toward more expressive and flexible relational data structures.

Comments: 21 pages
Subjects: **Machine Learning (cs.LG)**; Machine Learning (stat.ML)
Cite as: arXiv:1803.03324 **[cs.LG]**
(or arXiv:1803.03324v1 **[cs.LG]** for this version)
https://doi.org/10.48550/arXiv.1803.03324

Submission history

From: Yujia Li [view email]
[v1] Thu, 8 Mar 2018 22:20:00 UTC (1,145 KB)


Cornell University
We gratefully acknowledge support fr
arXiv > cs > arXiv:2206.06089
Search..
He

Computer Science > Artificial Intelligence

[Submitted on 24 May 2022 (v1), last revised 30 Jan 2023 (this version, v3)]

Graph Neural Networks Intersect Probabilistic Graphical Models: A Survey

Chenqing Hua, Sitao Luan, Qian Zhang, Jie Fu

Graphs are a powerful data structure to represent relational data and are widely used to describe complex real-world data structures. Probabilistic Graphical Models (PGMs) have been well-developed in the past years to mathematically model real-world scenarios in compact graphical representations of distributions of variables. Graph Neural Networks (GNNs) are new inference methods developed in recent years and are attracting growing attention due to their effectiveness and flexibility in solving inference and learning problems over graph-structured data. These two powerful approaches have different advantages in capturing relations from observations and how they conduct message passing, and they can benefit each other in various tasks. In this survey, we broadly study the intersection of GNNs and PGMs. Specifically, we first discuss how GNNs can benefit from learning structured representations in PGMs, generate explainable predictions by PGMs, and how PGMs can infer object relationships. Then we discuss how GNNs are implemented in PGMs for more efficient inference and structure learning. In the end, we summarize the benchmark datasets used in recent studies and discuss promising future directions.

Subjects: **Artificial Intelligence (cs.AI)**; Machine Learning (cs.LG)
Cite as: arXiv:2206.06089 **[cs.AI]**
(or arXiv:2206.06089v3 **[cs.AI]** for this version)
https://doi.org/10.48550/arXiv.2206.06089 ℹ

Submission history

From: Chenqing Hua [view email]
[v1] Tue, 24 May 2022 03:36:25 UTC (3,376 KB)
[v2] Fri, 18 Nov 2022 04:05:34 UTC (3,376 KB)
[v3] Mon, 30 Jan 2023 10:47:31 UTC (3,376 KB)

Encoder

Latent Space Optimization

Decoder

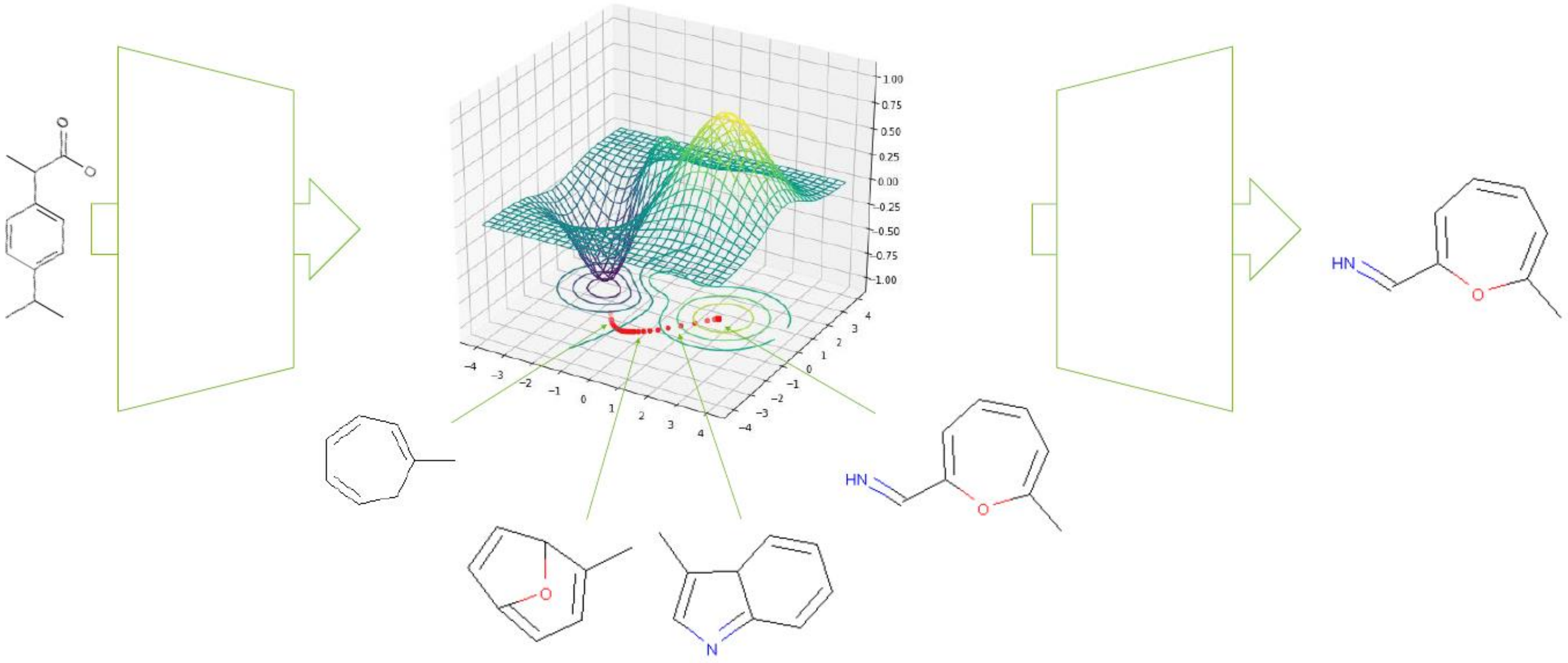


Table 5. Molecular Generation and Reconstruction Performance in ZINC250k

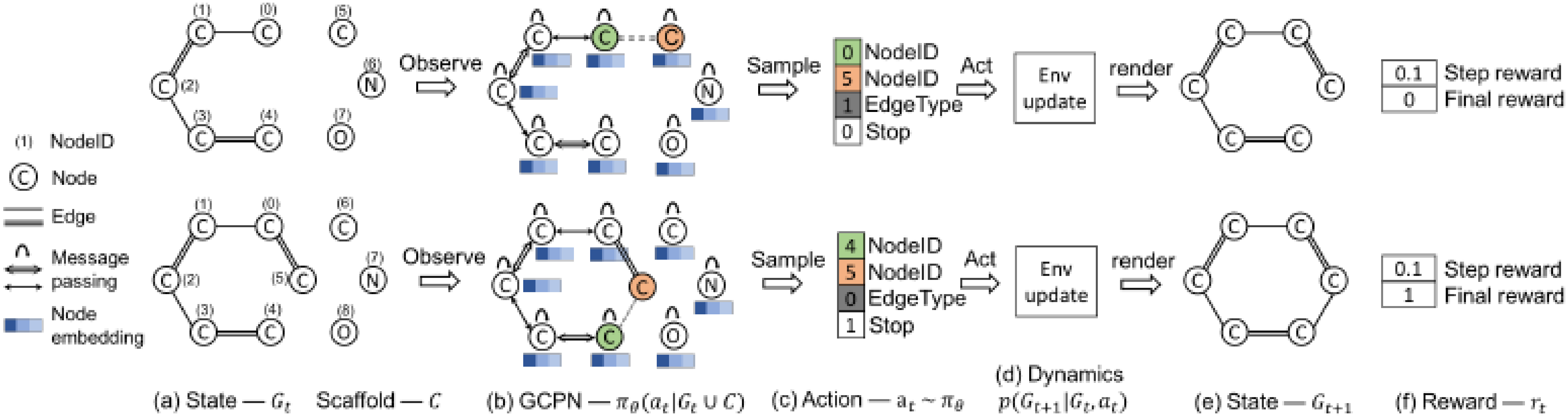
	%Validity	%Validity w/o check	%Novelty	%Uniqueness	%Reconstruction
SDVAE[c]	41.40	41.40	**100**	**100**	76.22
CGVAE[b]	**100**	n/a	**100**	99.82	n/a
JT-VAE[c]	**100**	n/a	**100**	99.96	76.54
GraphNVP[a]	42.60	42.60	**100**	94.80	**100**
GraphAF[c]	**100**	71.40	**100**	99.10	**100**
MoFlow[c]	**100**	45.61	**100**	99.92	**100**
GraphDF[b]	**100**	**89.03**	**100**	99.16	**100**
GCPN[c]	**100**	21.04	**100**	99.93	n/a
MRNN[a]	**100**	65.00	**100**	99.89	n/a

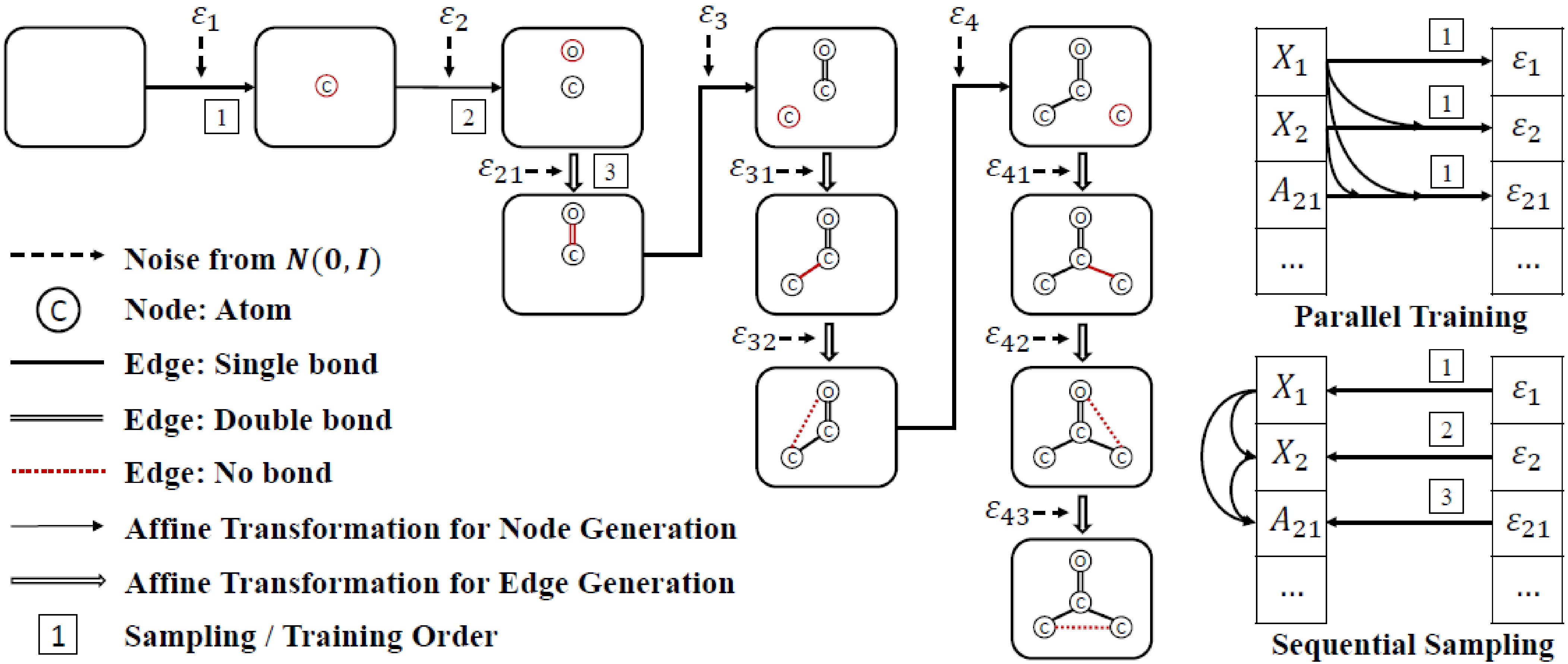
[a] Data is cited from MoFlow.[106] [b] Data is cited from the corresponding papers. [c] Data is obtained by running its official source code.

Table 6. Molecular Generation and Reconstruction Performance in QM9

	%Validity	%Validity w/o check	%Novelty	%Uniqueness	%Reconstruction
CGVAE[a]	**100**	n/a	94.35	98.57	n/a
JT-VAE[b]	99.86	n/a	**100**	96.32	68.53
GraphNVP[b]	50.86	50.86	88.46	97.52	**100**
GraphAF[b]	**100**	46.30	91.54	99.15	**100**
MoFlow[b]	**100**	81.14	97.3	**99.26**	**100**
GraphDF[a]	**100**	**82.67**	98.1	97.62	**100**
GCPN[b]	**100**	18.23	**100**	87.13	n/a

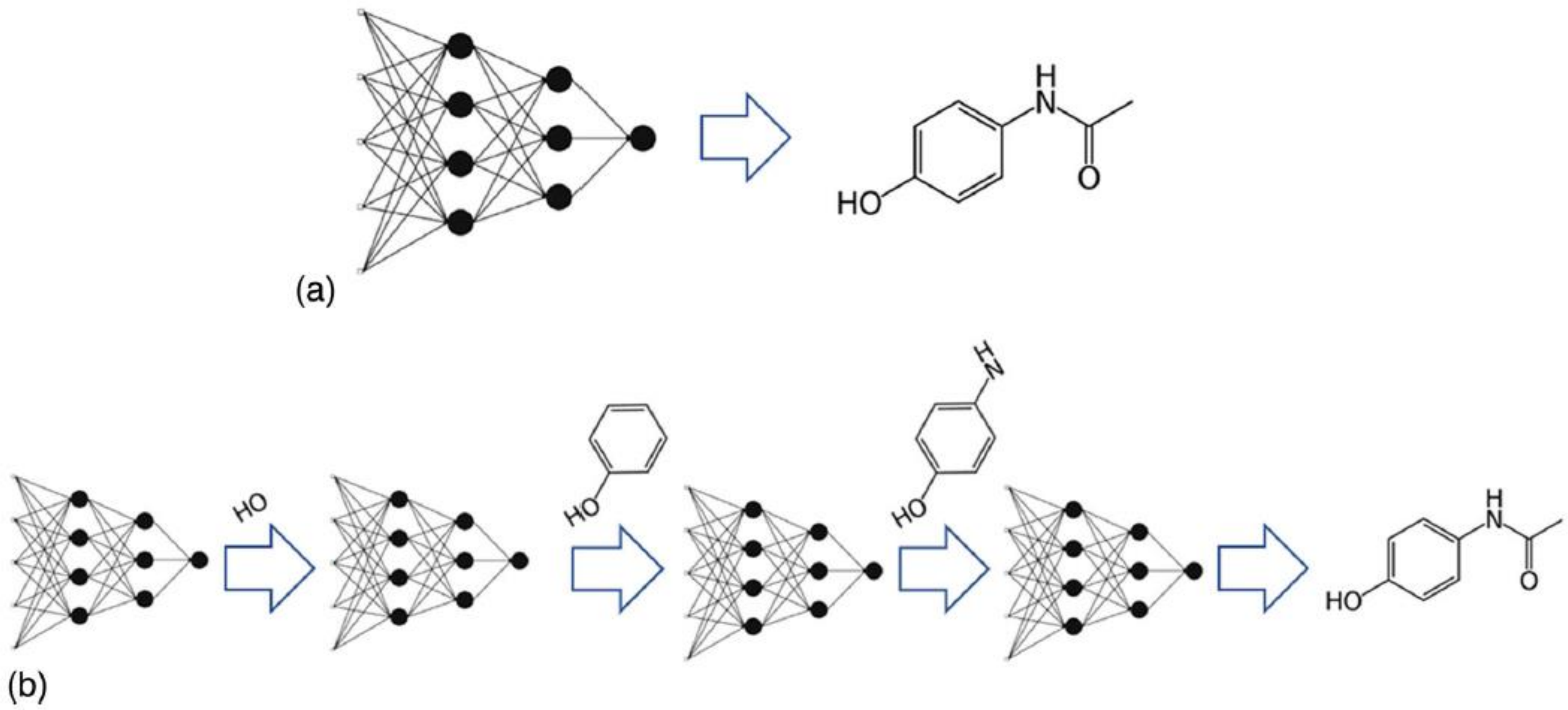
[a] Data is cited from the corresponding papers. [b] Data is obtained by running its official source code.

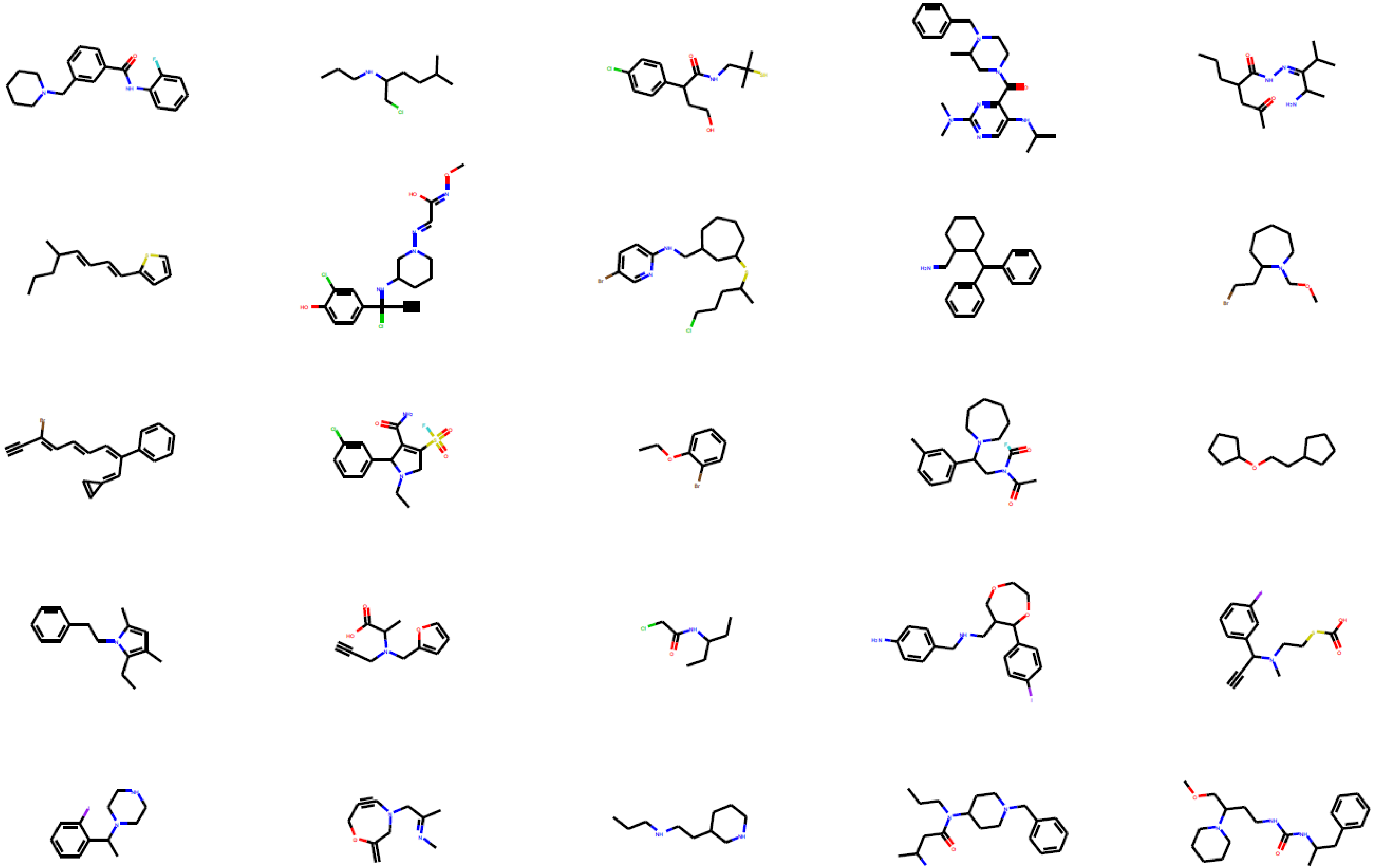

(1) NodeID
Node
Edge
Message passing
Node embedding
(1) (0) (5) (2) (6) (3) (4) (7)
Observe
Sample
0 NodeID
5 NodeID
1 EdgeType
0 Stop
Act
Env update
render
0.1 Step reward
0 Final reward
(1) (0) (6) (2) (5) (7) (3) (4) (8)
Observe
Sample
4 NodeID
5 NodeID
0 EdgeType
1 Stop
Act
Env update
render
0.1 Step reward
1 Final reward
(a) State — G_t
Scaffold — C
(b) GCPN — $\pi_\theta(a_t|G_t \cup C)$
(c) Action — $a_t \sim \pi_\theta$
(d) Dynamics $p(G_{t+1}|G_t, a_t)$
(e) State — G_{t+1}
(f) Reward — r_t

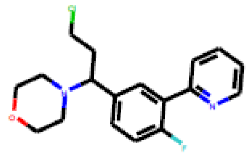


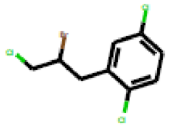
(a) Sampling Phases

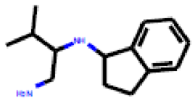
(b) Framework

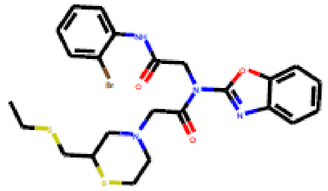

H
N
O
HO
(a)
HO
HO
H
N
HO
H
N
O
HO
(b)

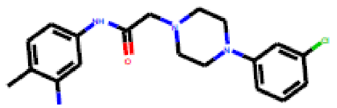


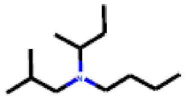

Cl
O
N
N
F

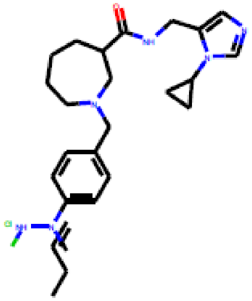

Cl
Br
Cl
Cl

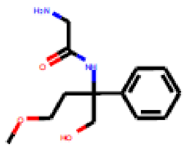

NH
H2N

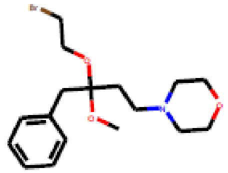

NH
Br
O
N
O
N
S
S
O
N

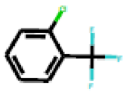

NH
O
N
N
Cl
NH2

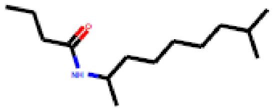

N

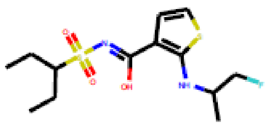

O
NH
N
N
N
Cl
NH
N

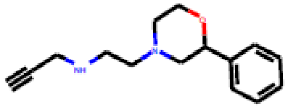

H2N
O
NH
O
HO

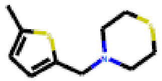

Br
O
O
N
O

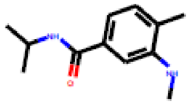

Cl
F
F
F

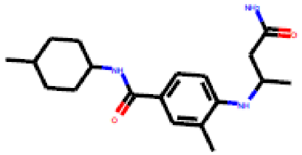

O
NH

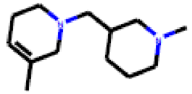

O
S
O
N
OH
S
NH
F

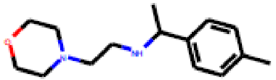

NH
N
O

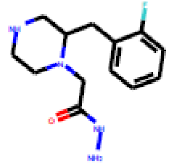

S
N
S

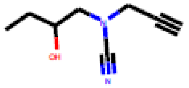

NH
O
NH

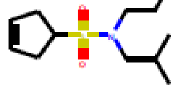

NH2
O
NH
O
NH

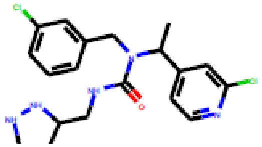

N
N

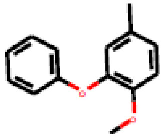

O
N
NH

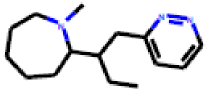

HN
N
F
O
NH
NH2

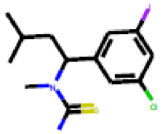

OH
N
N

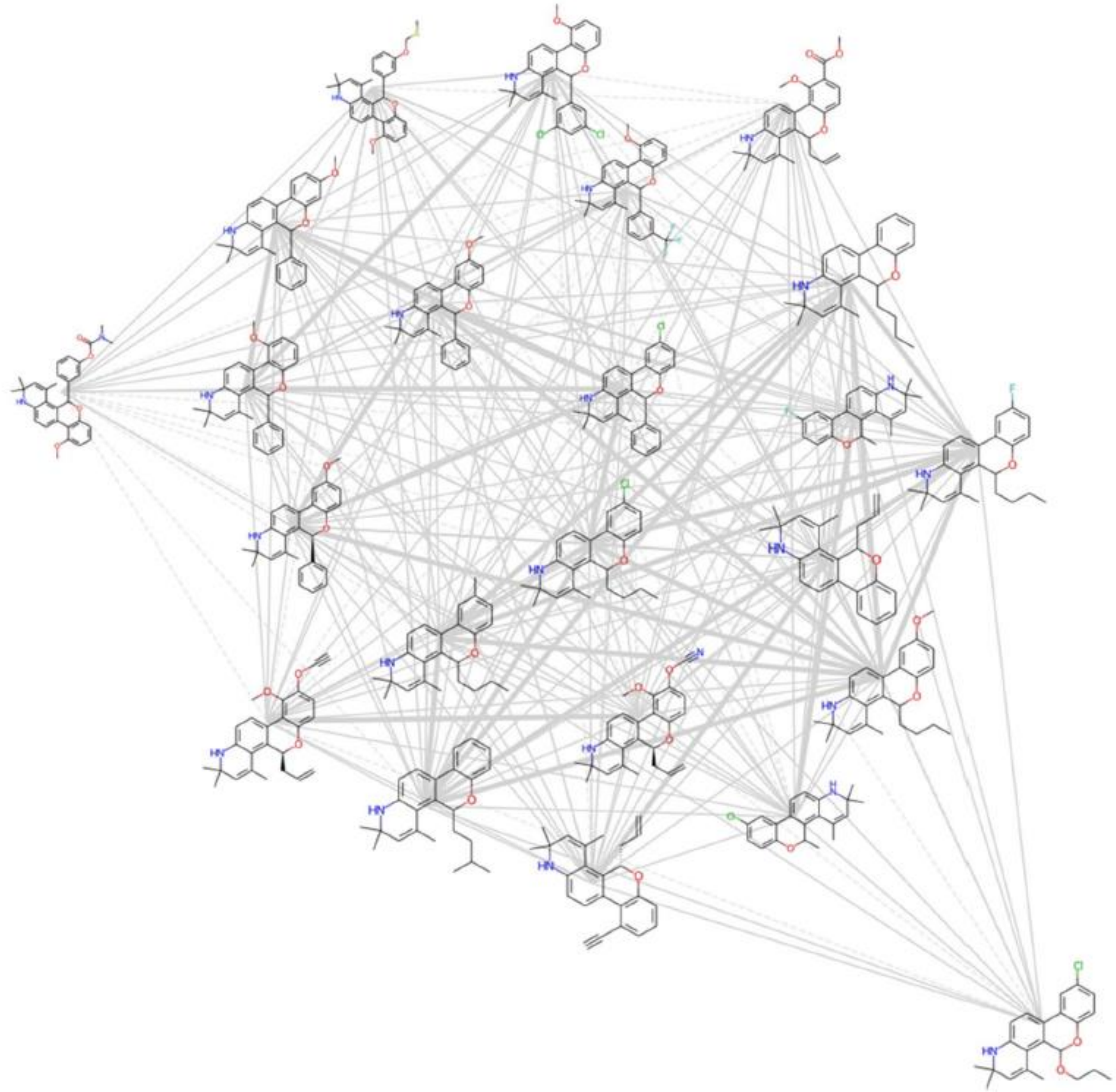

O
S
O
N

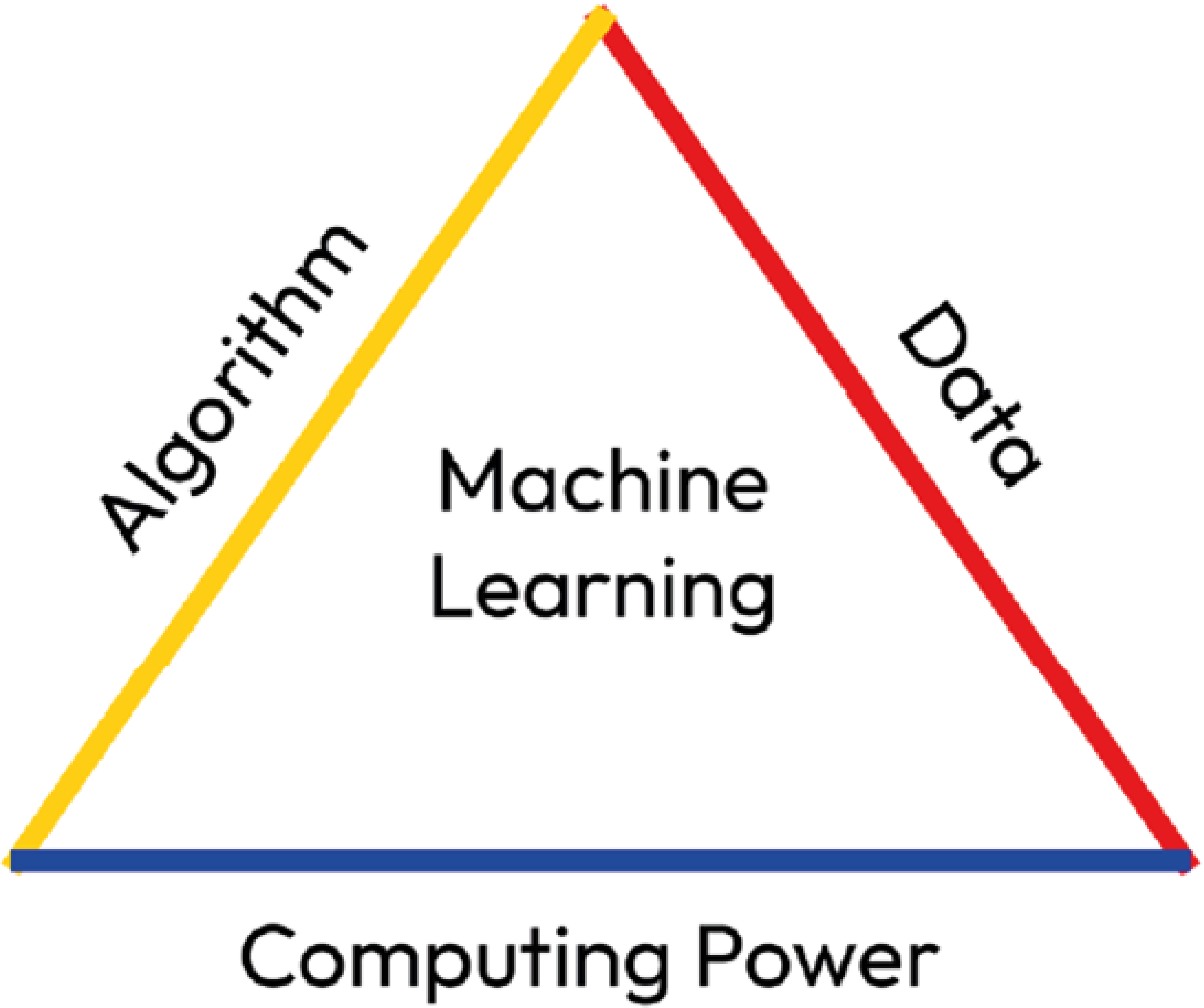

Cl
N
NH
O
N
Cl
HN
NH


O
O


N
N
N


I
N
Cl
S





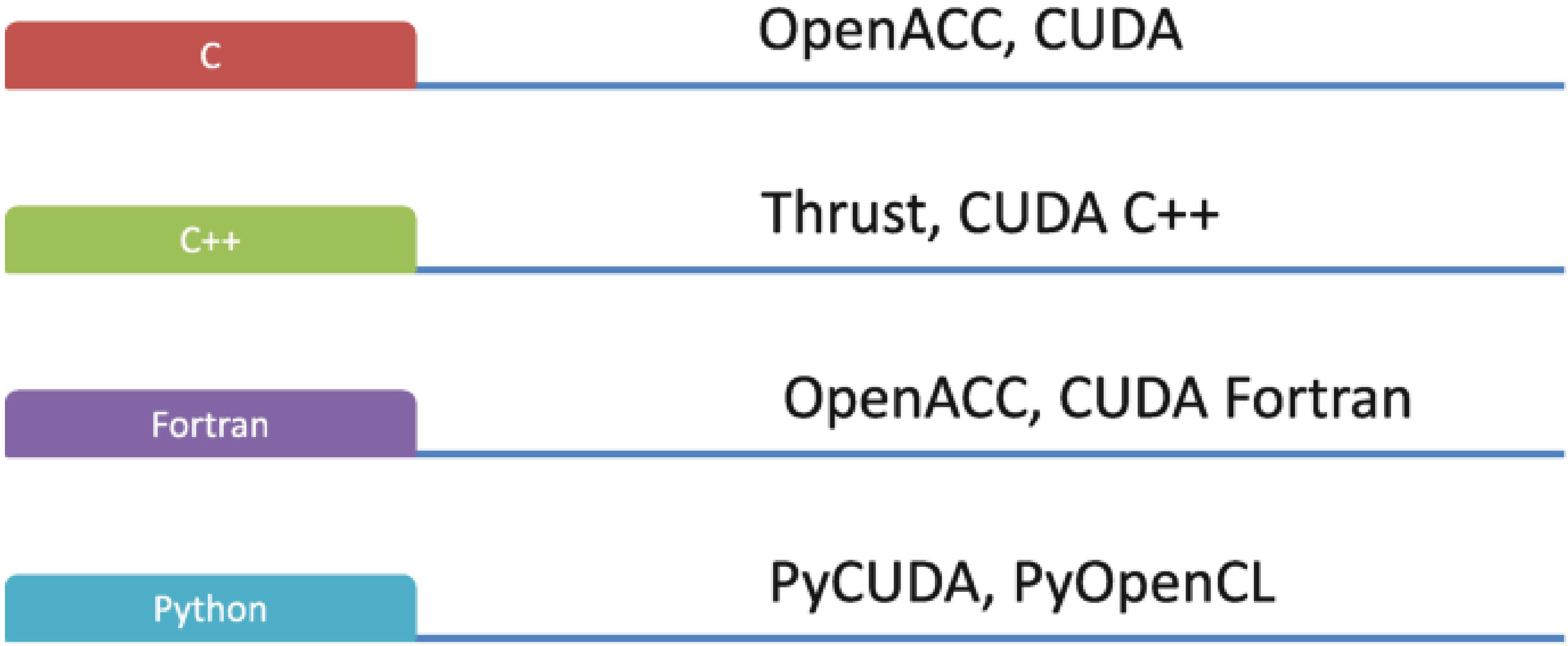
Algorithm

Data

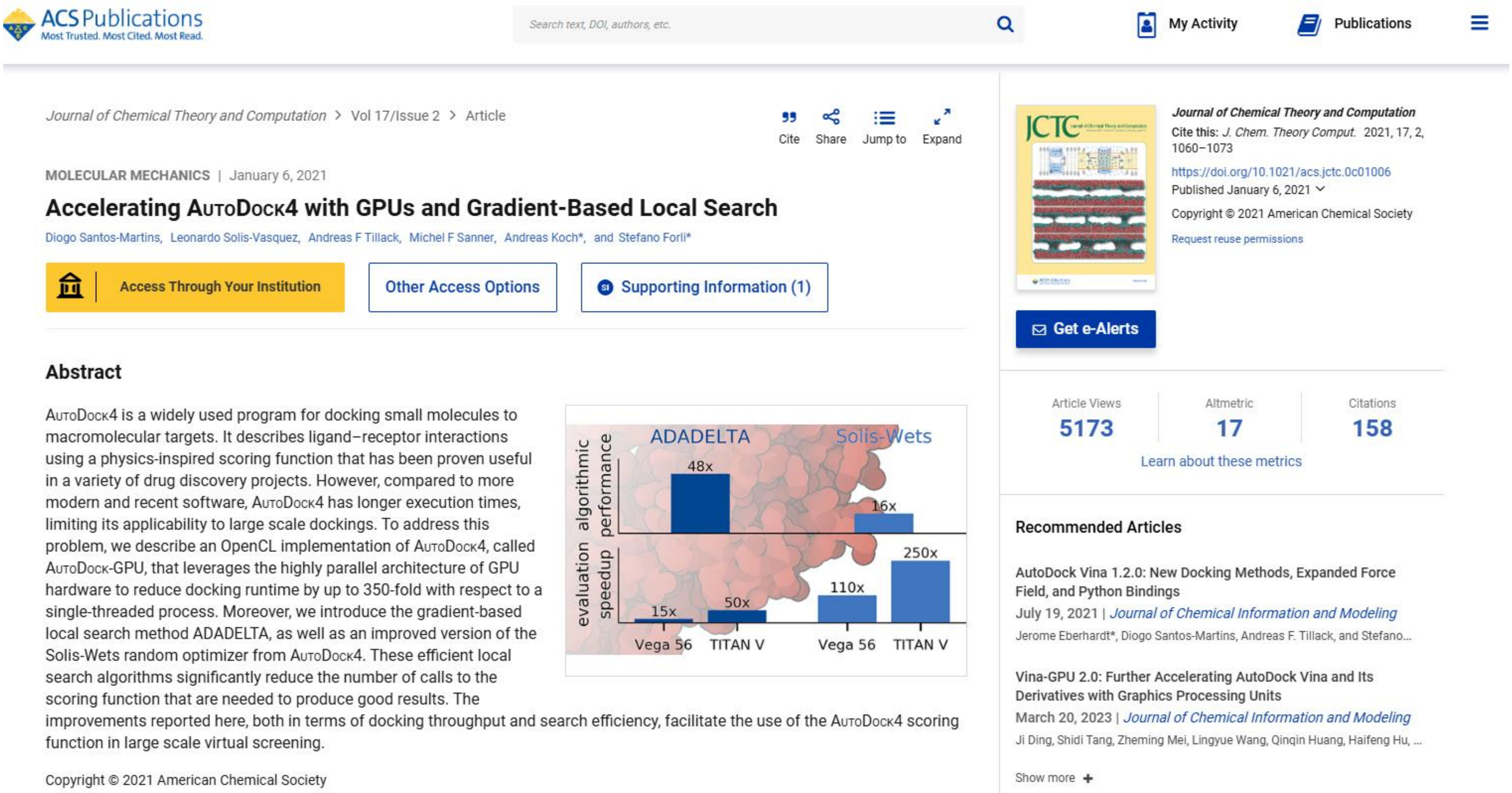
Machine Learning

Computing Power

Compute Unified Device Architecture (CUDA)



C	OpenACC, CUDA
C++	Thrust, CUDA C++
Fortran	OpenACC, CUDA Fortran
Python	PyCUDA, PyOpenCL



ACS Publications
Most Trusted. Most Cited. Most Read.
Search text, DOI, authors, etc.
My Activity
Publications
Cite
Share
Jump to
Expand

Journal of Chemical Theory and Computation > Vol 17/Issue 2 > Article

MOLECULAR MECHANICS | January 6, 2021

Accelerating AutoDock4 with GPUs and Gradient-Based Local Search

Diogo Santos-Martins, Leonardo Solis-Vasquez, Andreas F Tillack, Michel F Sanner, Andreas Koch*, and Stefano Forli*

Access Through Your Institution
Other Access Options
Supporting Information (1)

Abstract

AutoDock4 is a widely used program for docking small molecules to macromolecular targets. It describes ligand–receptor interactions using a physics-inspired scoring function that has been proven useful in a variety of drug discovery projects. However, compared to more modern and recent software, AutoDock4 has longer execution times, limiting its applicability to large scale dockings. To address this problem, we describe an OpenCL implementation of AutoDock4, called AutoDock-GPU, that leverages the highly parallel architecture of GPU hardware to reduce docking runtime by up to 350-fold with respect to a single-threaded process. Moreover, we introduce the gradient-based local search method ADADELTA, as well as an improved version of the Solis-Wets random optimizer from AutoDock4. These efficient local search algorithms significantly reduce the number of calls to the scoring function that are needed to produce good results. The improvements reported here, both in terms of docking throughput and search efficiency, facilitate the use of the AutoDock4 scoring function in large scale virtual screening.

Journal of Chemical Theory and Computation
Cite this: *J. Chem. Theory Comput.* 2021, 17, 2, 1060–1073
https://doi.org/10.1021/acs.jctc.0c01006
Published January 6, 2021

Article Views	Altmetric	Citations
5173	17	158

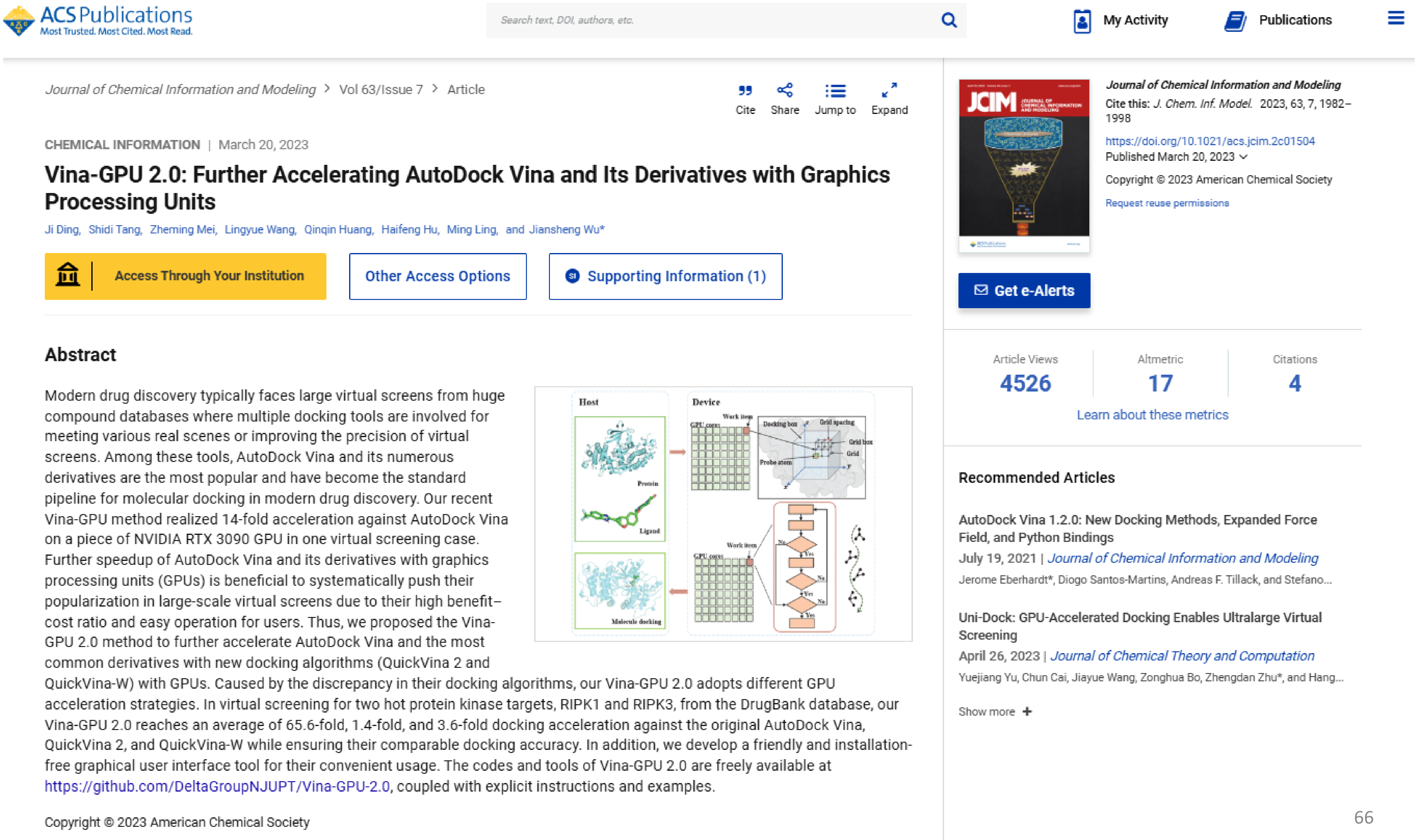
Learn about these metrics

Recommended Articles

AutoDock Vina 1.2.0: New Docking Methods, Expanded Force Field, and Python Bindings
July 19, 2021 | *Journal of Chemical Information and Modeling*
Jerome Eberhardt*, Diogo Santos-Martins, Andreas F. Tillack, and Stefano...

Vina-GPU 2.0: Further Accelerating AutoDock Vina and Its Derivatives with Graphics Processing Units
March 20, 2023 | *Journal of Chemical Information and Modeling*
Ji Ding, Shidi Tang, Zheming Mei, Lingyue Wang, Qinqin Huang, Haifeng Hu, ...

Show more +



Journal of Chemical Information and Modeling > Vol 63/Issue 7 > Article

CHEMICAL INFORMATION | March 20, 2023

Vina-GPU 2.0: Further Accelerating AutoDock Vina and Its Derivatives with Graphics Processing Units

Ji Ding, Shidi Tang, Zheming Mei, Lingyue Wang, Qinqin Huang, Haifeng Hu, Ming Ling, and Jiansheng Wu*

Abstract

Modern drug discovery typically faces large virtual screens from huge compound databases where multiple docking tools are involved for meeting various real scenes or improving the precision of virtual screens. Among these tools, AutoDock Vina and its numerous derivatives are the most popular and have become the standard pipeline for molecular docking in modern drug discovery. Our recent Vina-GPU method realized 14-fold acceleration against AutoDock Vina on a piece of NVIDIA RTX 3090 GPU in one virtual screening case. Further speedup of AutoDock Vina and its derivatives with graphics processing units (GPUs) is beneficial to systematically push their popularization in large-scale virtual screens due to their high benefit–cost ratio and easy operation for users. Thus, we proposed the Vina-GPU 2.0 method to further accelerate AutoDock Vina and the most common derivatives with new docking algorithms (QuickVina 2 and QuickVina-W) with GPUs. Caused by the discrepancy in their docking algorithms, our Vina-GPU 2.0 adopts different GPU acceleration strategies. In virtual screening for two hot protein kinase targets, RIPK1 and RIPK3, from the DrugBank database, our Vina-GPU 2.0 reaches an average of 65.6-fold, 1.4-fold, and 3.6-fold docking acceleration against the original AutoDock Vina, QuickVina 2, and QuickVina-W while ensuring their comparable docking accuracy. In addition, we develop a friendly and installation-free graphical user interface tool for their convenient usage. The codes and tools of Vina-GPU 2.0 are freely available at https://github.com/DeltaGroupNJUPT/Vina-GPU-2.0, coupled with explicit instructions and examples.

Journal of Chemical Information and Modeling
Cite this: *J. Chem. Inf. Model.* 2023, 63, 7, 1982–1998
https://doi.org/10.1021/acs.jcim.2c01504
Published March 20, 2023
Copyright © 2023 American Chemical Society
Request reuse permissions

Article Views	Altmetric	Citations
4526	17	4

Learn about these metrics

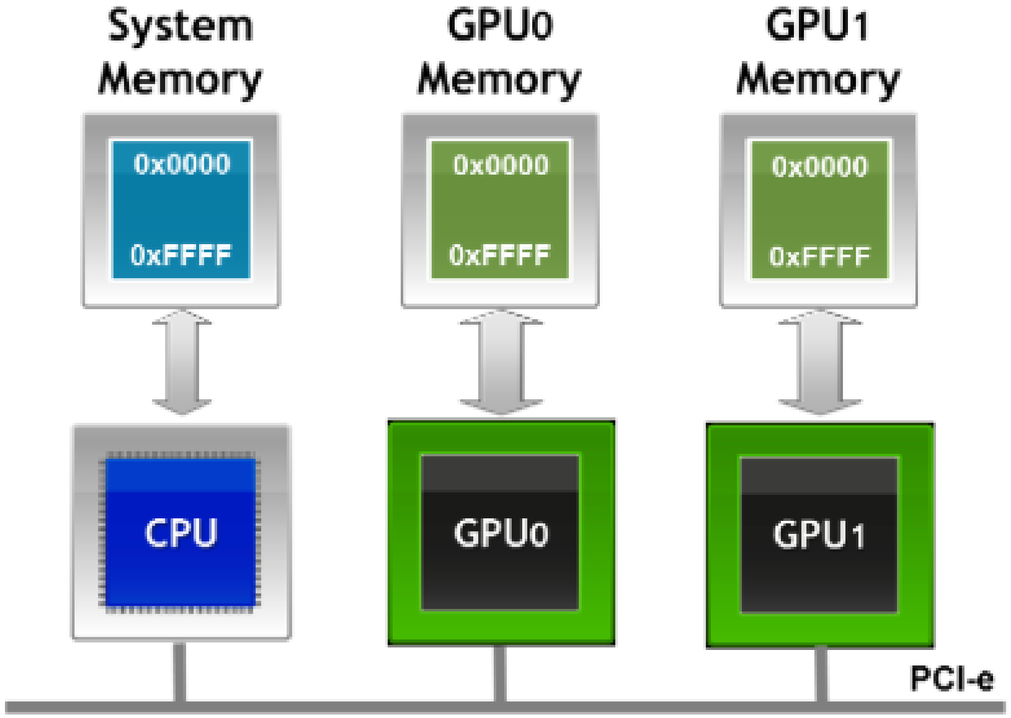
Recommended Articles

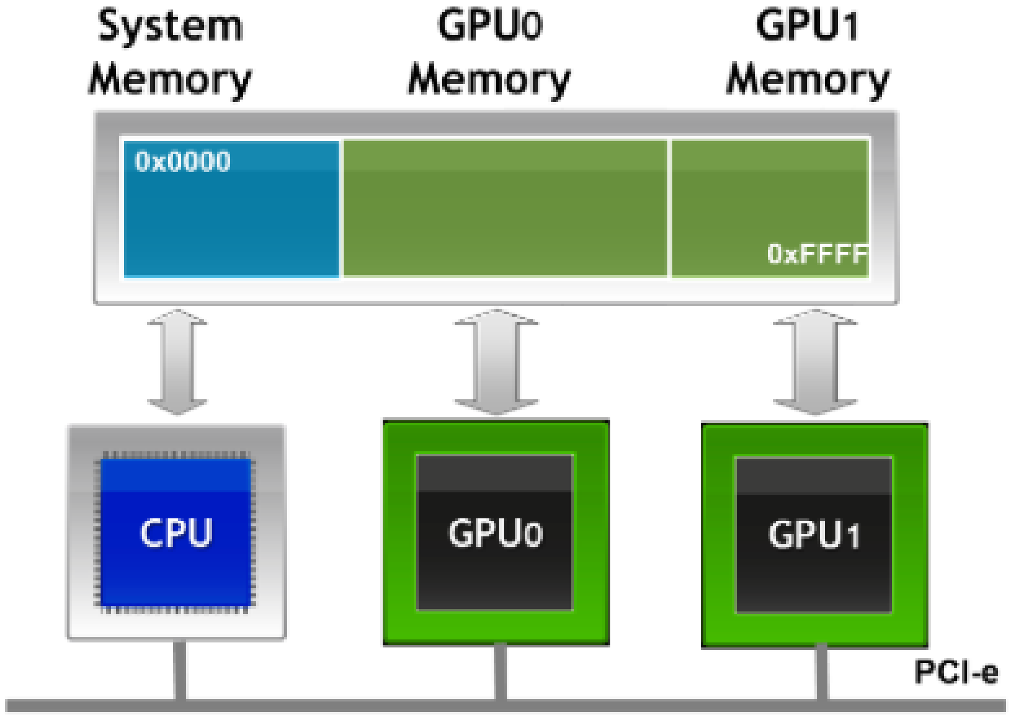
AutoDock Vina 1.2.0: New Docking Methods, Expanded Force Field, and Python Bindings
July 19, 2021 | *Journal of Chemical Information and Modeling*
Jerome Eberhardt*, Diogo Santos-Martins, Andreas F. Tillack, and Stefano...

Uni-Dock: GPU-Accelerated Docking Enables Ultralarge Virtual Screening
April 26, 2023 | *Journal of Chemical Theory and Computation*
Yuejiang Yu, Chun Cai, Jiayue Wang, Zonghua Bo, Zhengdan Zhu*, and Hang...

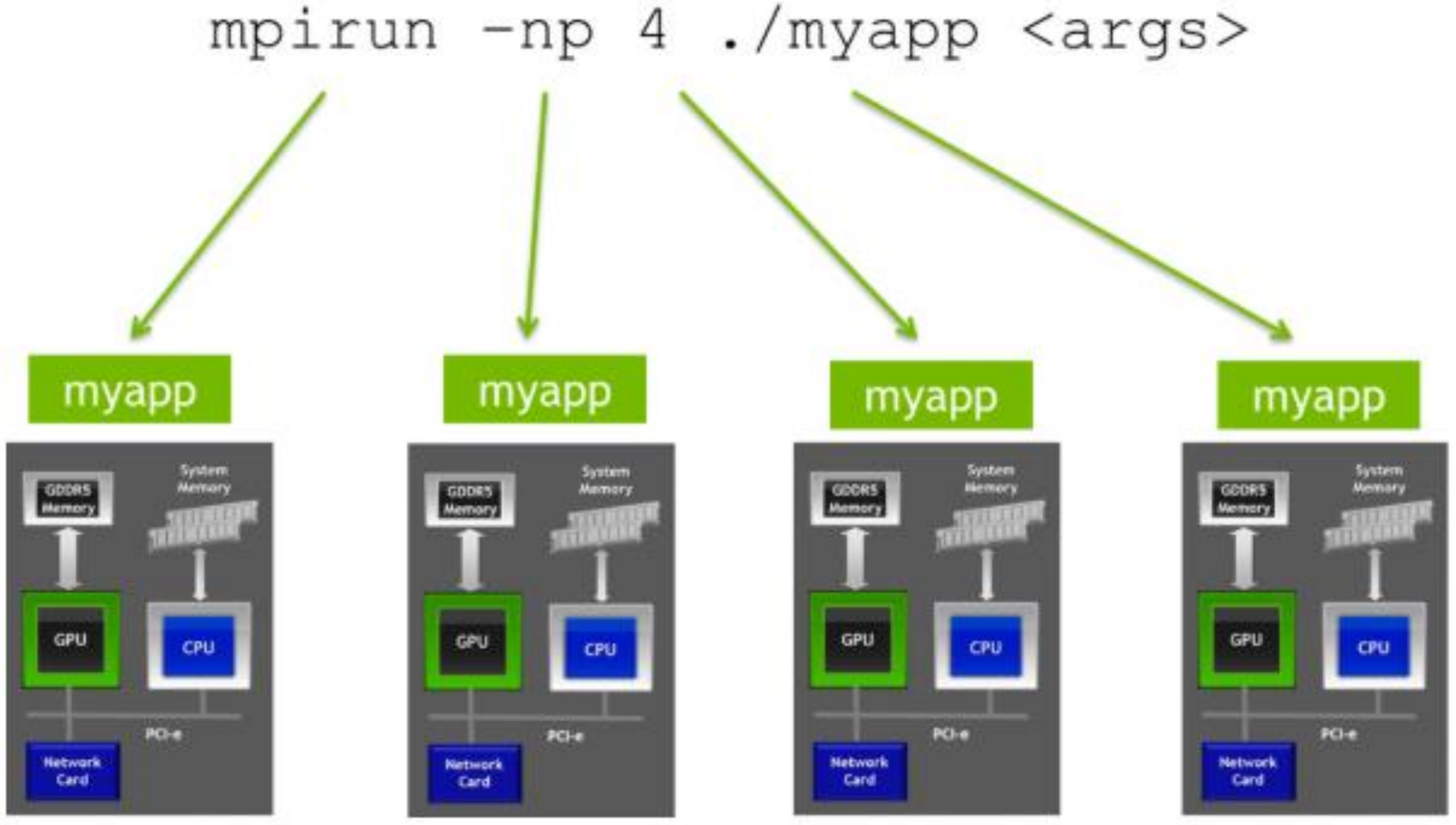
Show more +

Message Passing Interface (MPI)



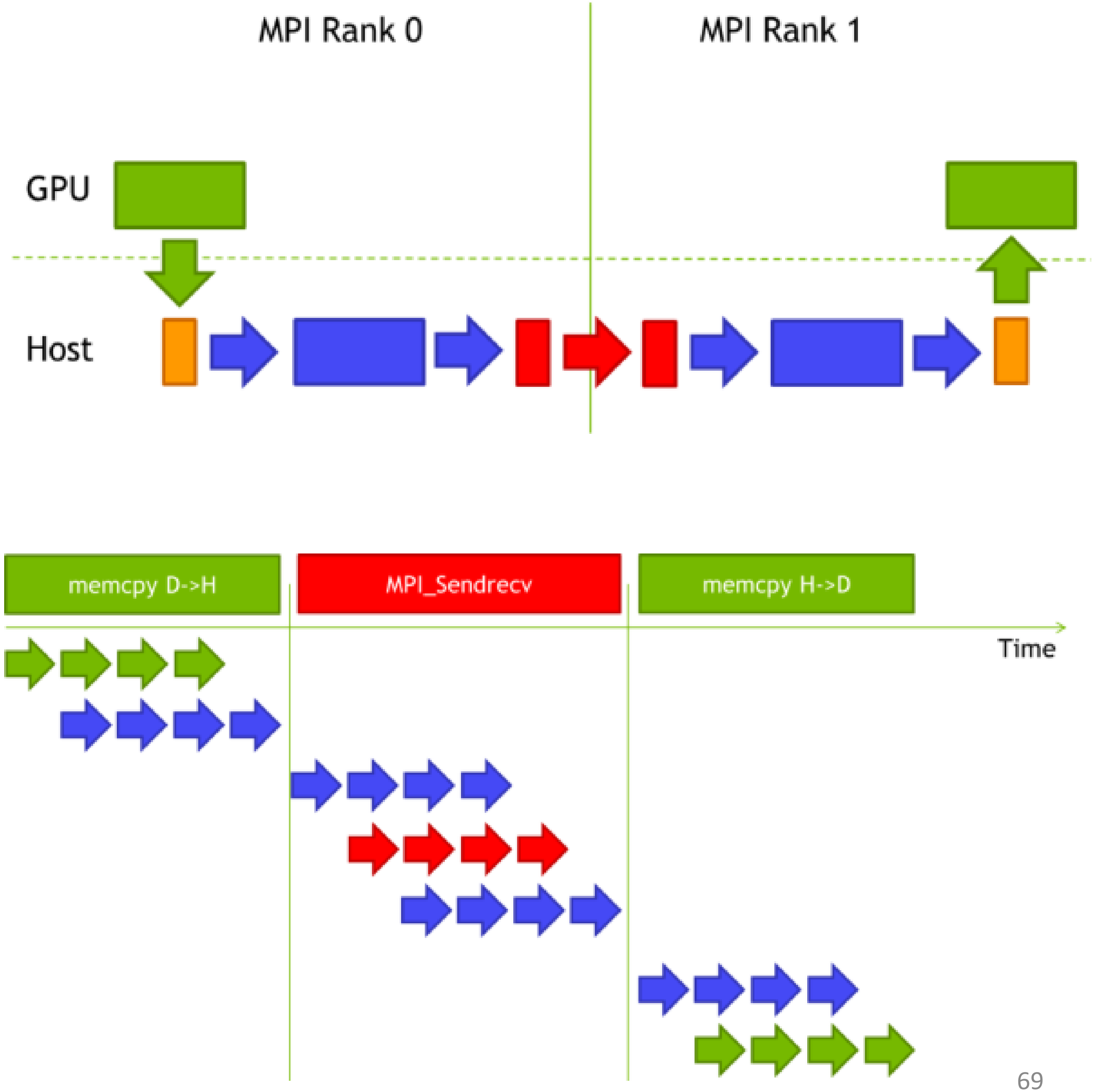


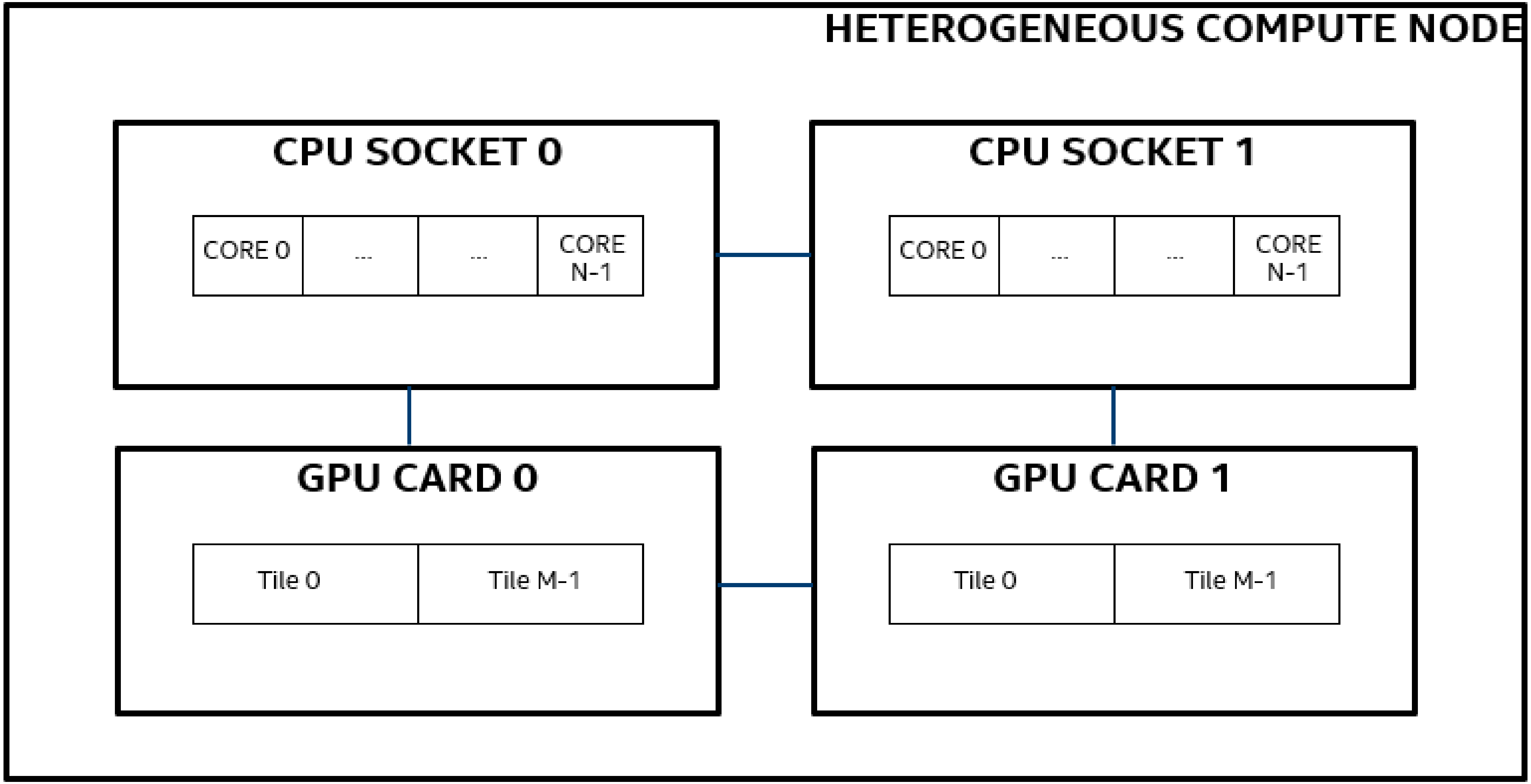
Message Passing Interface (MPI)



```
mpirun -np 4 ./myapp <args>
```

Message Passing Interface (MPI)





HETEROGENEOUS COMPUTE NODE

CPU SOCKET 0

CORE 0 | ... | ... | CORE N-1

CPU SOCKET 1

CORE 0 | ... | ... | CORE N-1

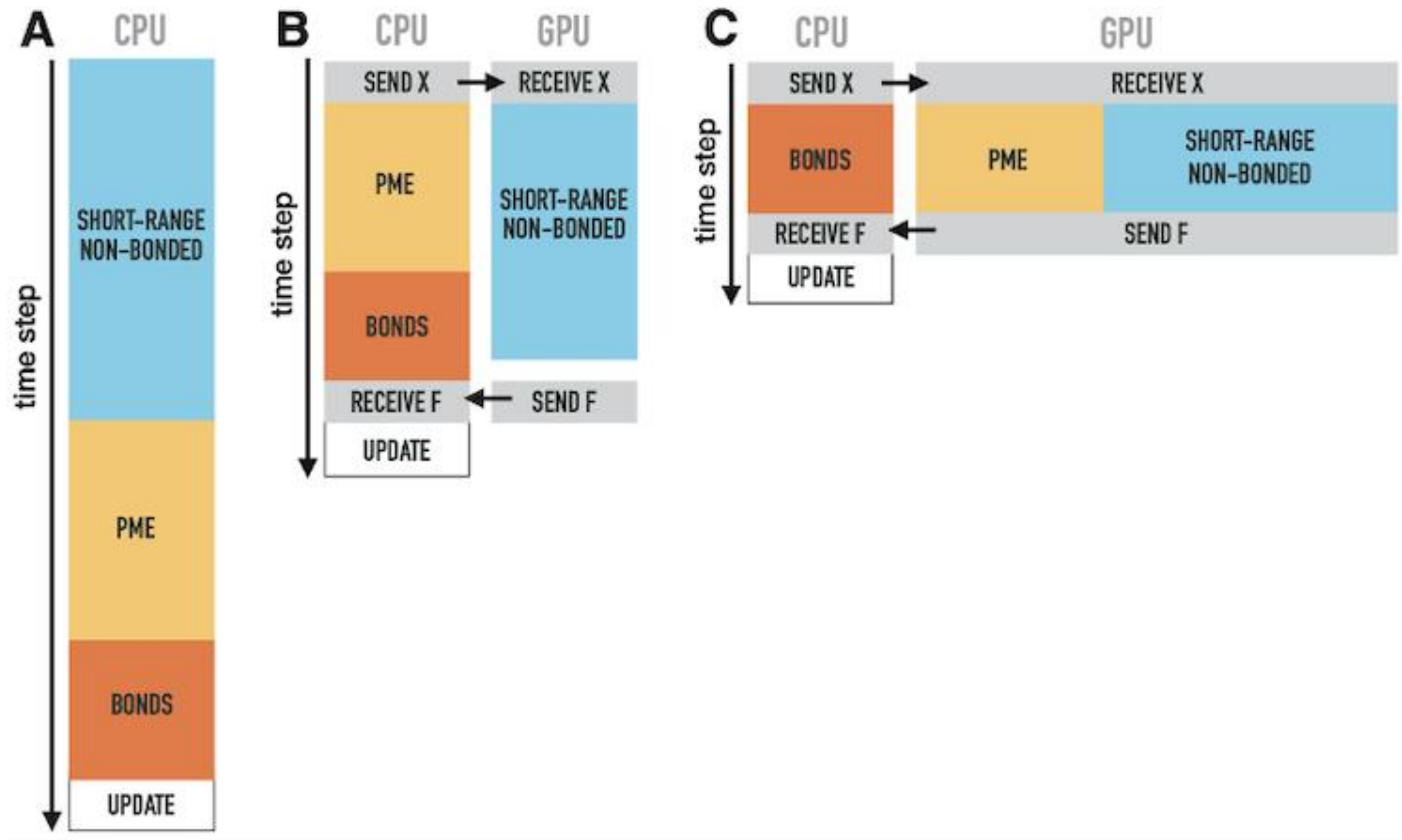
GPU CARD 0

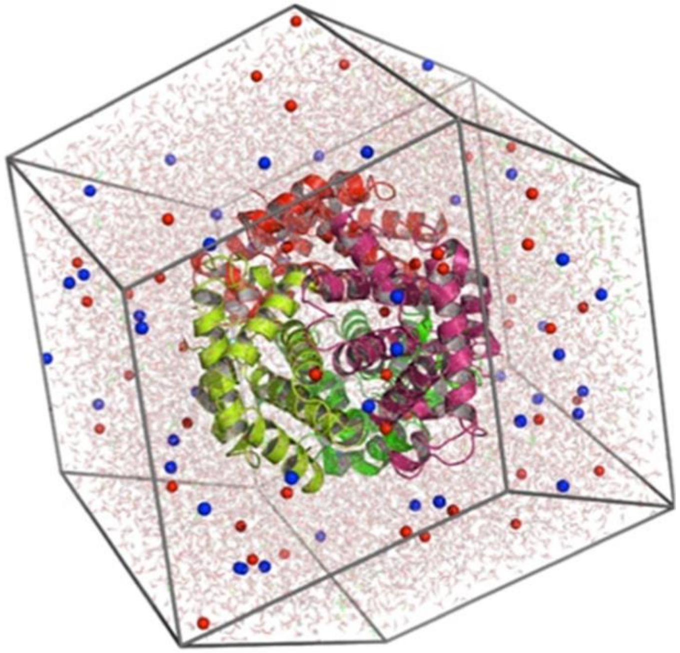
Tile 0 | Tile M-1

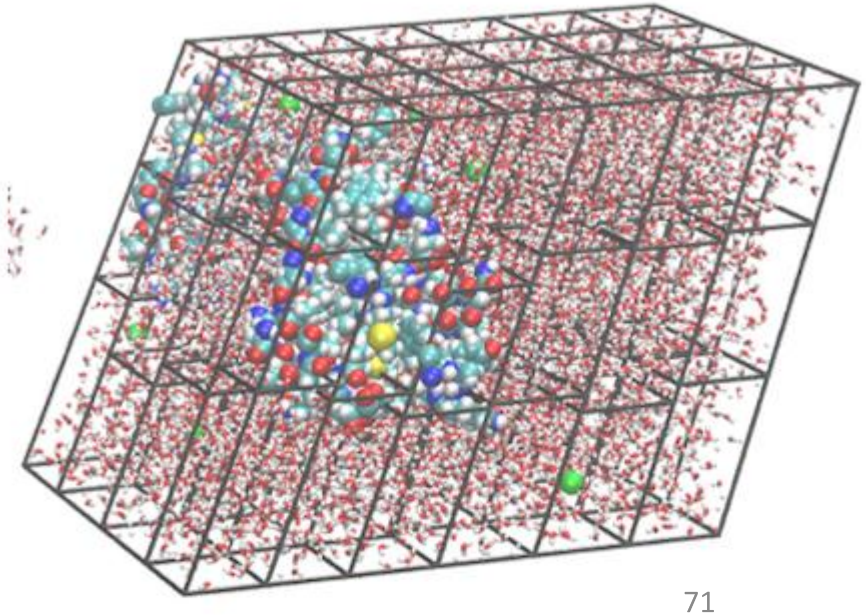
GPU CARD 1

Tile 0 | Tile M-1

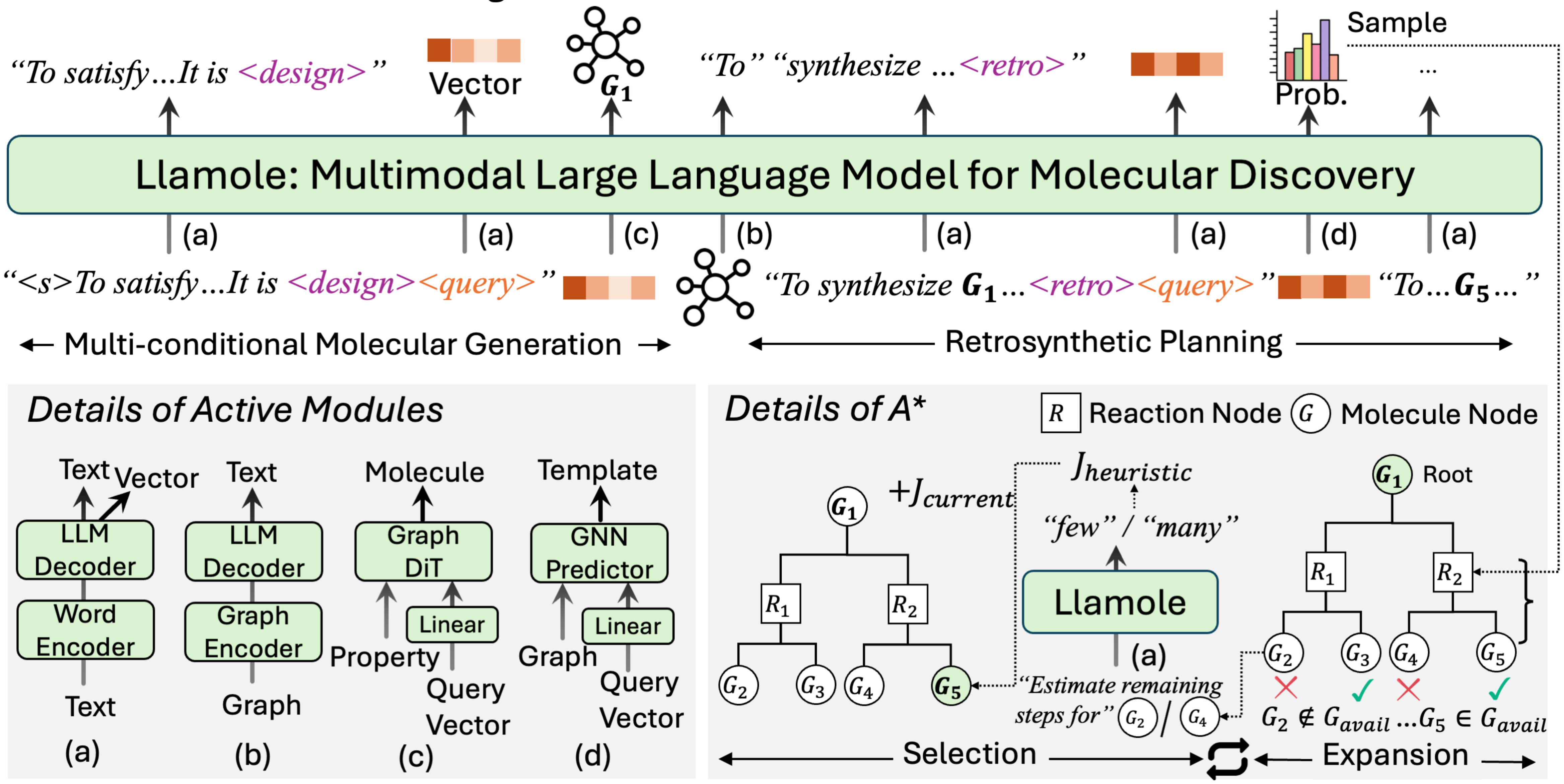
GROMACS_MPI (GMX_MPI)







Llamole's Multimodal Autoregressive Framework



"To satisfy…It is <design>"

Vector

G_1

"To" "synthesize …<retro>"

Prob.

Sample

…

Llamole: Multimodal Large Language Model for Molecular Discovery

(a) (a) (c) (b) (a) (a) (d) (a)

"<s>To satisfy…It is <design><query>"

"To synthesize G_1…<retro><query>"

"To…G_5…"

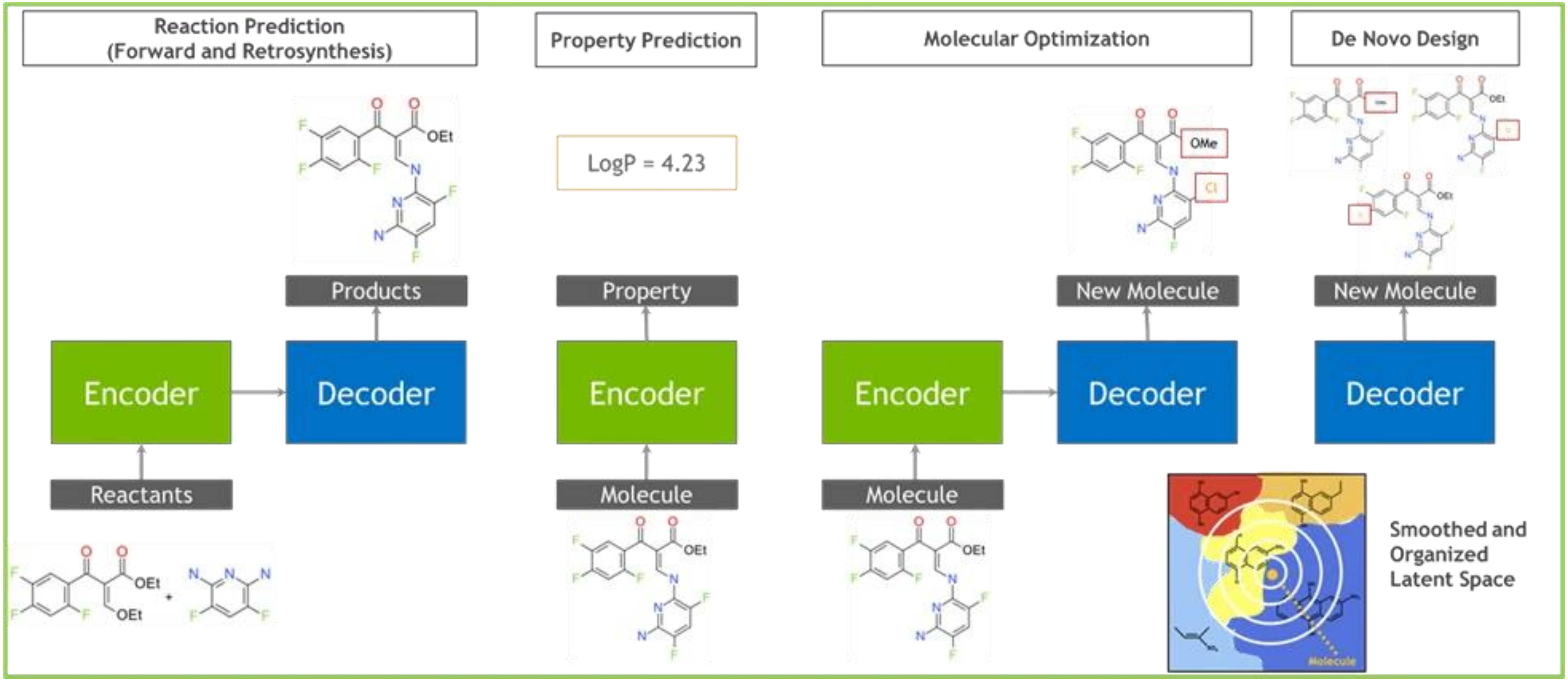
← Multi-conditional Molecular Generation →

← Retrosynthetic Planning →

Details of Active Modules

*Details of A**

R Reaction Node *G* Molecule Node



Reaction Prediction (Forward and Retrosynthesis)

Reactants → Encoder → Decoder → Products

Property Prediction

Molecule → Encoder → Property

LogP = 4.23

Molecular Optimization

Molecule → Encoder → Decoder → New Molecule

De Novo Design

Decoder → New Molecule

Smoothed and Organized Latent Space

THANK
YOU!