Deep Generative Model

original tutorial

VAE

HYL video

Pixel RNN

PixelRNN

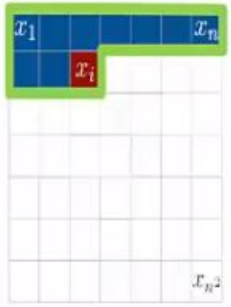
sequence of pixels

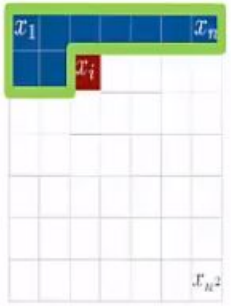
dependent on previous pixels

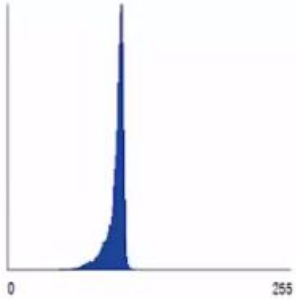
R

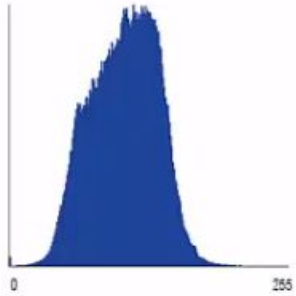
G

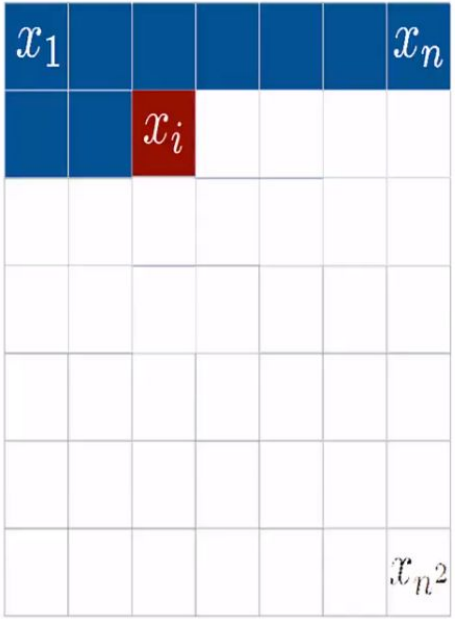
B

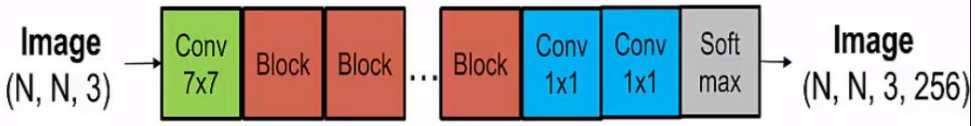








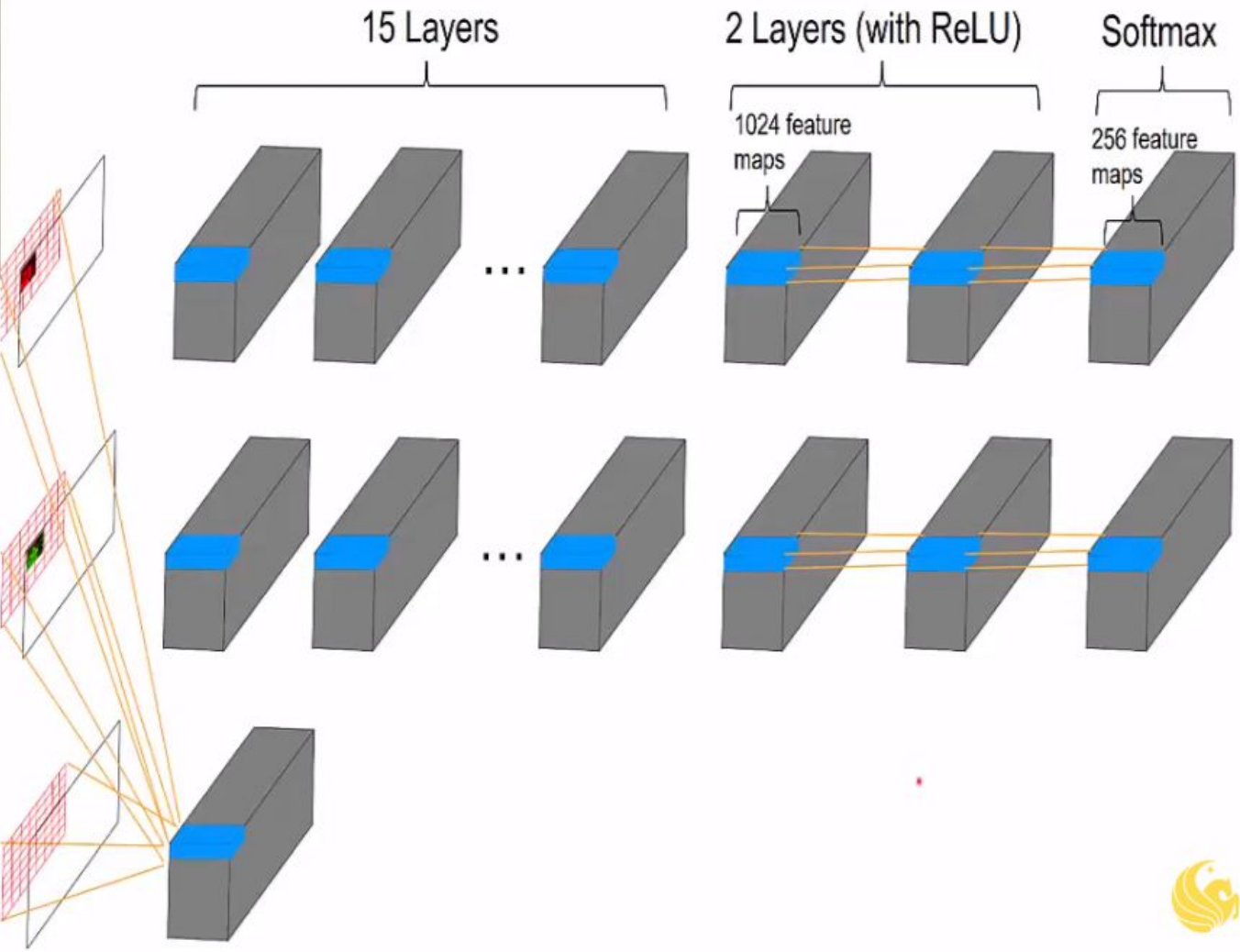




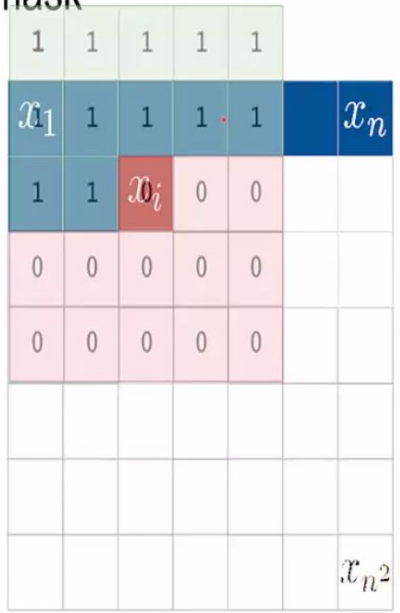
$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, ..., x_{i-1})$$

Pixel CNN

PixelCNN

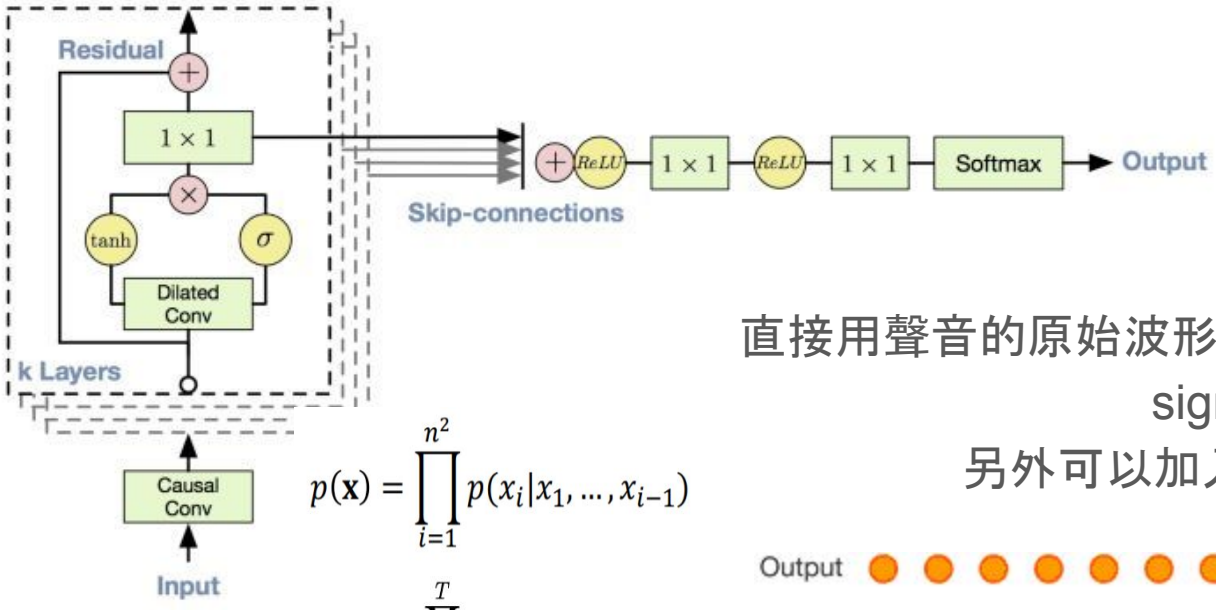


Kernel mask



WaveNet

WaveNet



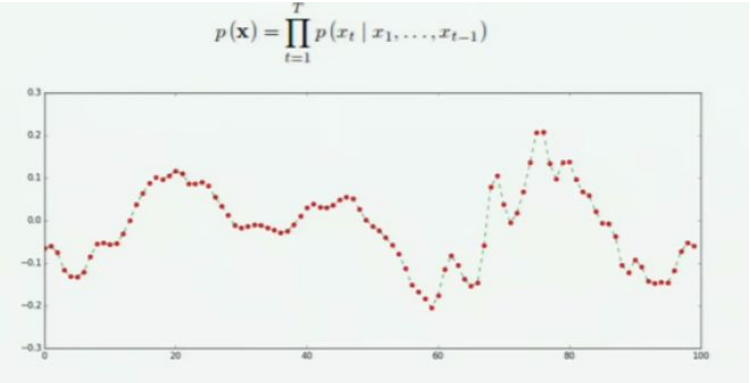
$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, \ldots, x_{i-1})$$

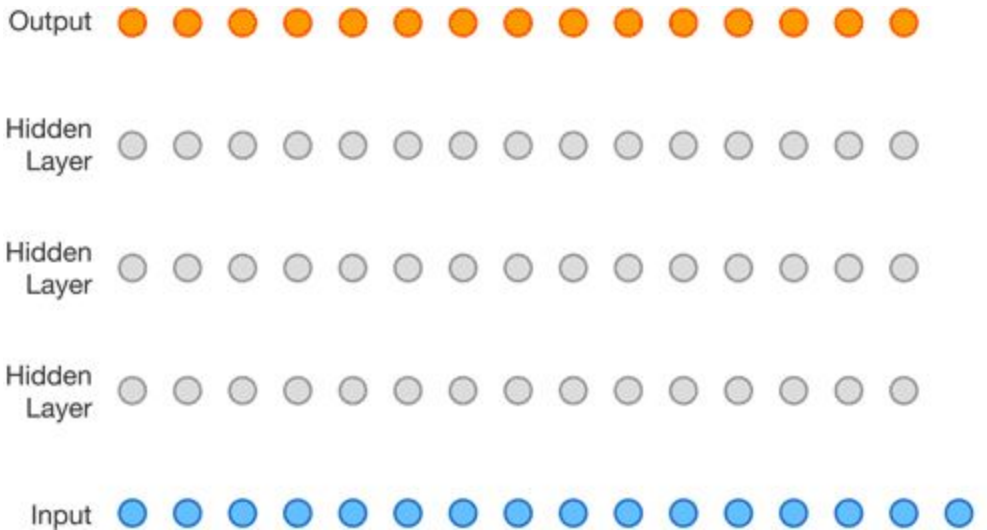
$$p(x|h) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1}, h)$$

直接用聲音的原始波形來進行訓練，生成聲波訊號。

sigmoid 作為activation function

另外可以加入h作為conditional WaveNet





Challenges for Graph Generation

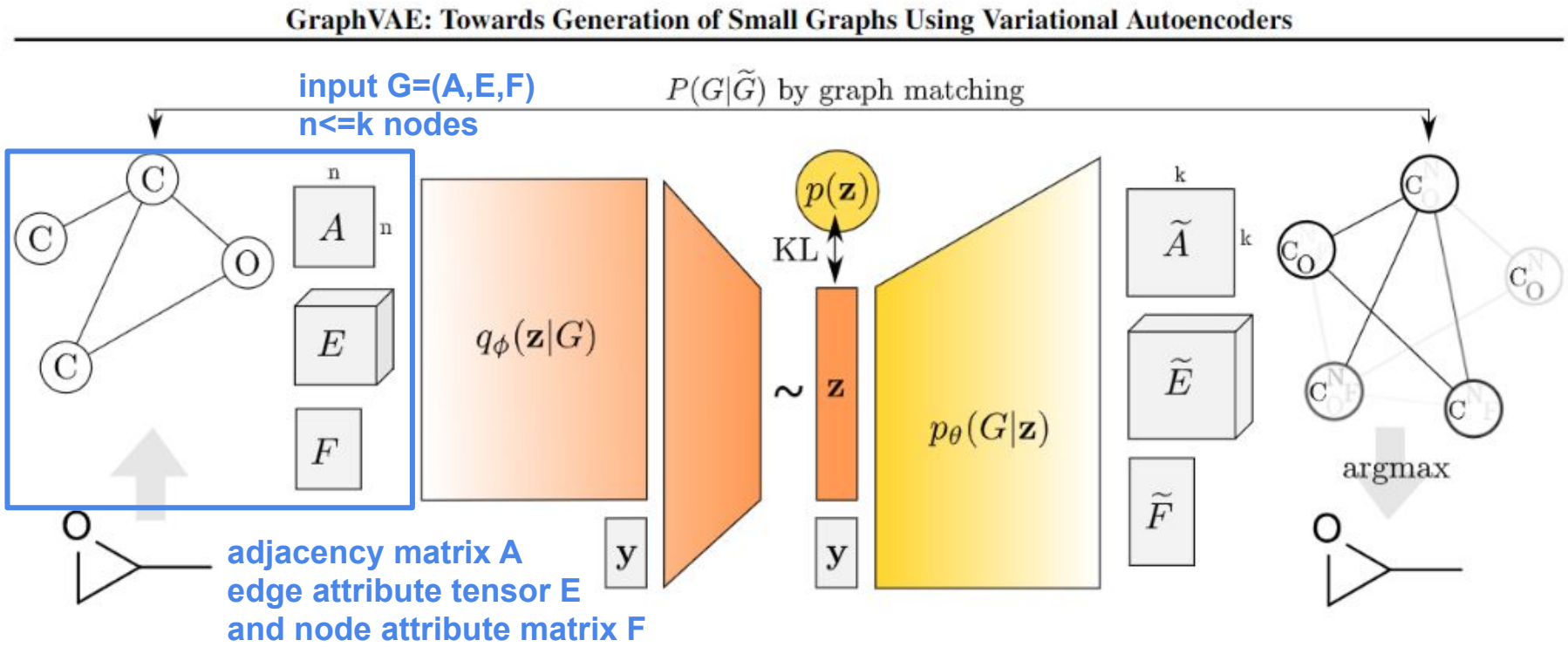
- The structures and sizes of graphs are different
- No orders between the nodes
- Discrete

Graph VAE

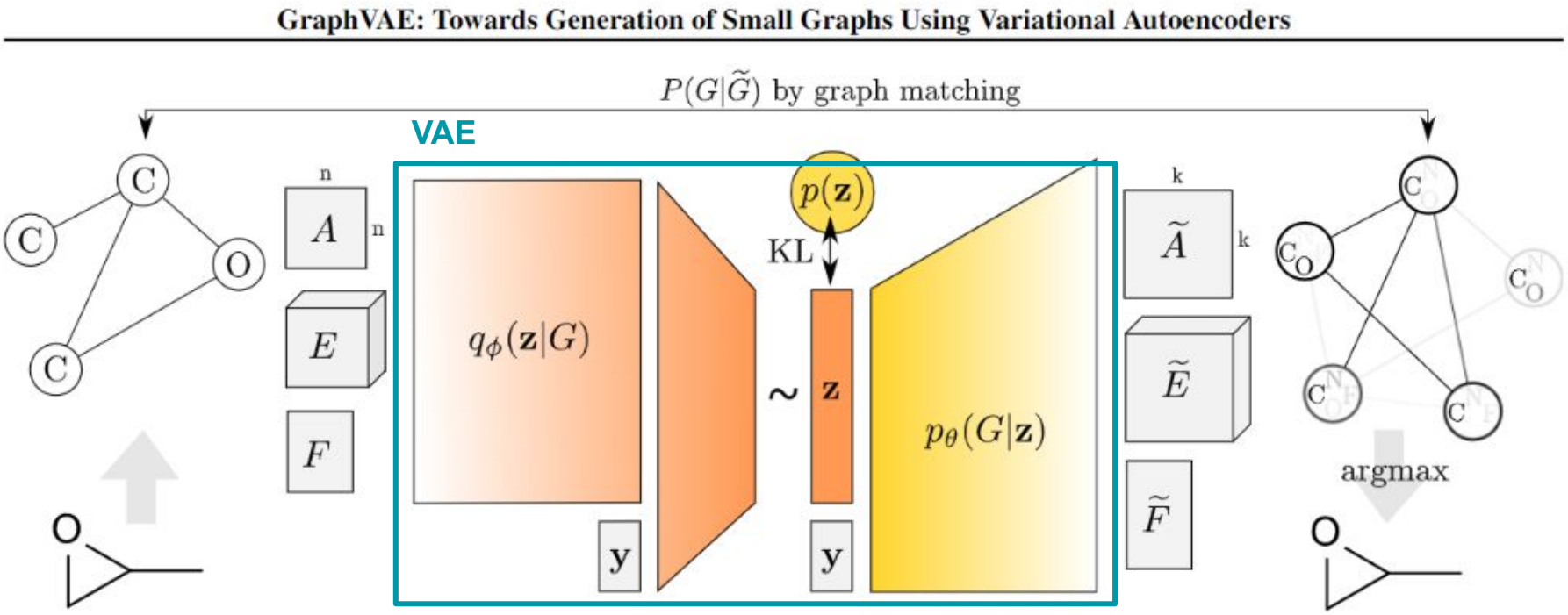
paper: arxiv 1802.03480

9 Feb 2018

Graph VAE


GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders
input G=(A,E,F)
n<=k nodes
$P(G|\tilde{G})$ by graph matching
n
A
n
E
F
$q_\phi(\mathbf{z}|G)$
y
$p(\mathbf{z})$
KL
$\sim$ z
y
$p_\theta(G|\mathbf{z})$
k
$\tilde{A}$
k
$\tilde{E}$
$\tilde{F}$
argmax
adjacency matrix A
edge attribute tensor E
and node attribute matrix F

Graph VAE


GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders
$P(G|\tilde{G})$ by graph matching
VAE
n
A
n
E
F
$q_\phi(\mathbf{z}|G)$
y
$p(\mathbf{z})$
KL
$\sim$
z
y
$p_\theta(G|\mathbf{z})$
k
$\tilde{A}$
k
$\tilde{E}$
$\tilde{F}$
argmax

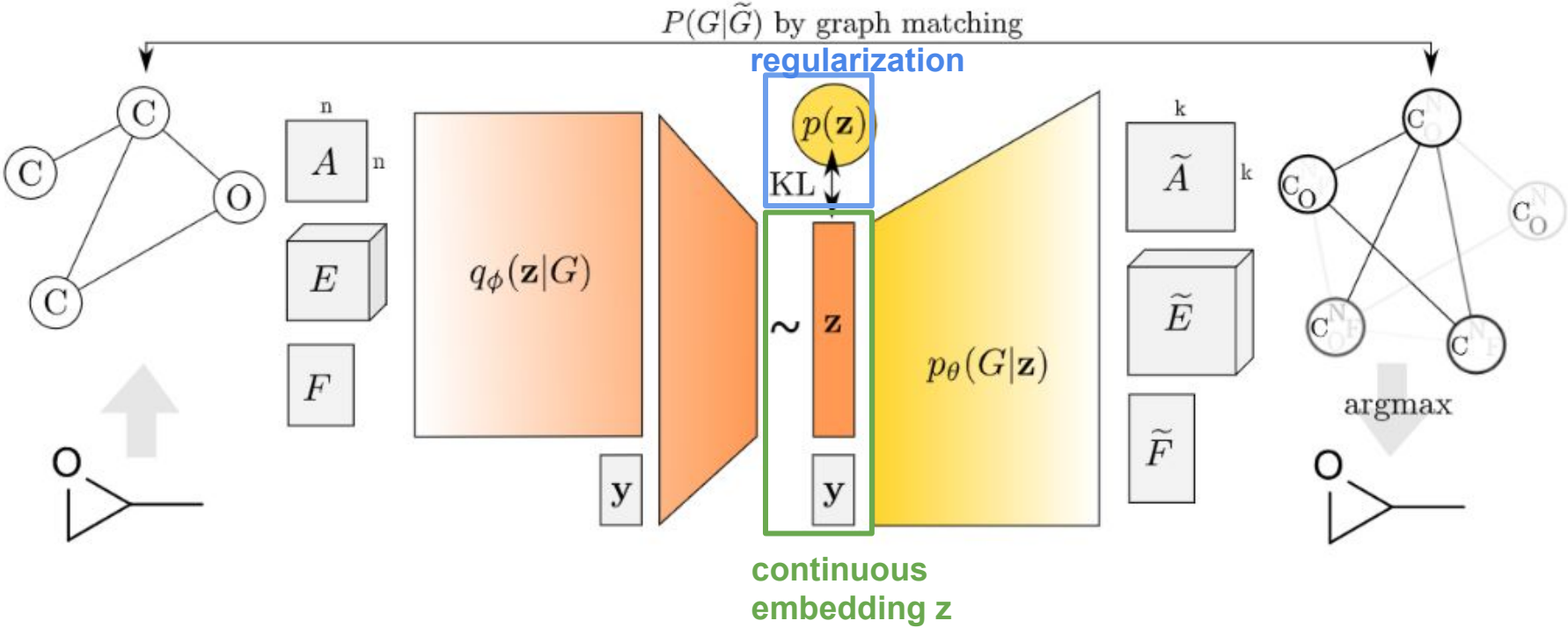
Graph VAE

$$\mathcal{L}(\phi, \theta; G) = \mathbb{E}_{q_\phi(\mathbf{z}|G)}[-\log p_\theta(G|\mathbf{z})] + \mathrm{KL}[q_\phi(\mathbf{z}|G)||p(\mathbf{z})] \quad (1)$$

reconstruction loss

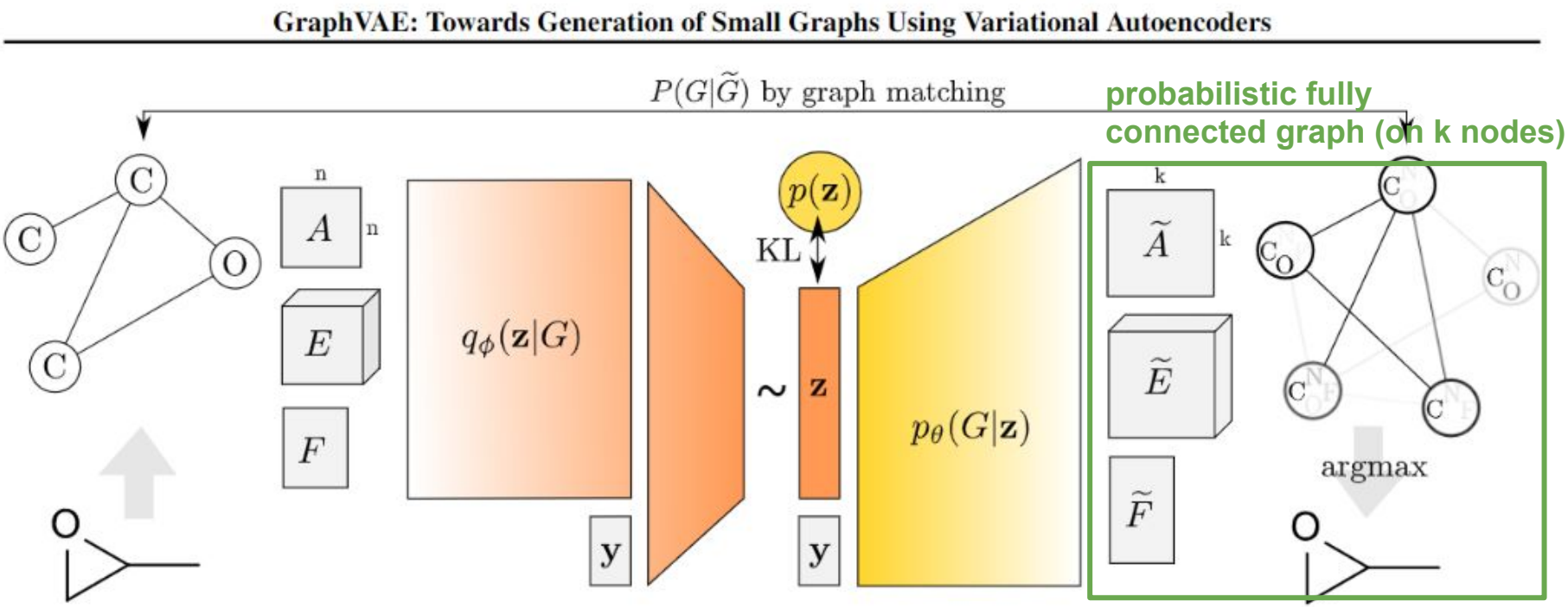
regularization

GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders



Graph VAE

A~: predicted adjacency [node prob & edge prob] sigmoid
E~: edge attribute, (class prob for edges) softmax
F~: node attribute, (class prob for nodes) softmax



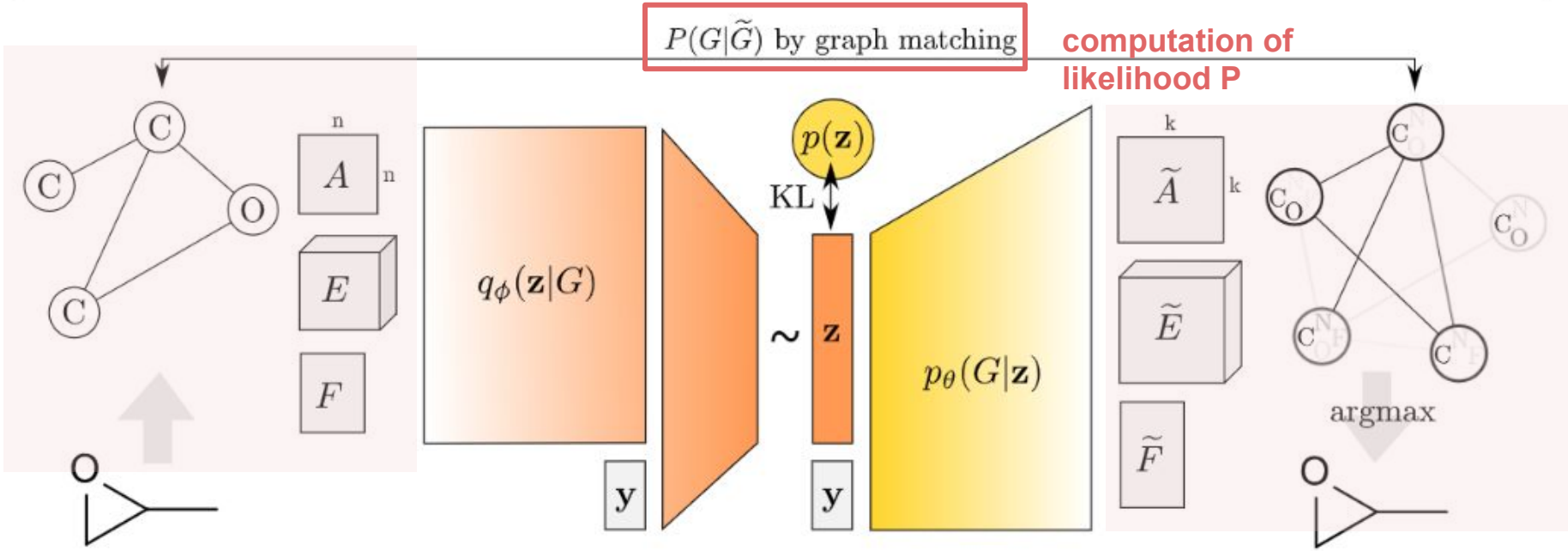
Graph VAE

$$S((i,j),(a,b)) = \\ = (E^T_{i,j,.}\widetilde{E}_{a,b,.})A_{i,j}\widetilde{A}_{a,b}\widetilde{A}_{a,a}\widetilde{A}_{b,b}[i \neq j \wedge a \neq b] + \\ + (F^T_{i,.}\widetilde{F}_{a,.})\widetilde{A}_{a,a}[i = j \wedge a = b] \quad (4)$$

similarity function

$$X \in \{0,1\}^{k \times n}$$

GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders



Graph VAE

$$\mathcal{L}(\phi, \theta; G) = \mathbb{E}_{q_\phi(\mathbf{z}|G)}[-\log p_\theta(G|\mathbf{z})] + \mathrm{KL}[q_\phi(\mathbf{z}|G)||p(\mathbf{z})] \quad (1)$$

reconstruction loss

regularization

$$X \in \{0,1\}^{k \times n}$$

input adjacency matrix is mapped to the predicted graph

$$\bar{A}' = X \bar{A} X^T$$

predicted **node** attribute matrix and slices of **edge** attribute matrix are transferred to the input graph as

$$\tilde{F}' = X^T \tilde{F} \qquad \tilde{E}'_{\cdot,\cdot,l} = X^T \tilde{E}_{\cdot,\cdot,l} X.$$

The overall **reconstruction loss** is a weighed sum of the previous terms:

$$-\log p(G|\mathbf{z}) = -\lambda_A \log p(A'|\mathbf{z}) - \lambda_F \log p(F|\mathbf{z}) - \\ - \lambda_E \log p(E|\mathbf{z}) \quad (3)$$

Graph VAE

$$\mathcal{L}(\phi, \theta; G) = \\ = \mathbb{E}_{q_\phi(\mathbf{z}|G)}[-\log p_\theta(G|\mathbf{z})] + \mathrm{KL}[q_\phi(\mathbf{z}|G)||p(\mathbf{z})] \quad (1)$$

reconstruction loss

regularization

$X \in \{0,1\}^{k \times n}$ obtain from graph matching, a binary assignment matrix

input adjacency matrix is mapped to the predicted graph $\tilde{A}' = X A X^T$

$\tilde{F}' = X^T \tilde{F}$

predicted **node** attribute matrix and slices of **edge** attribute matrix are transferred to the input graph as $\tilde{E}'_{\cdot,\cdot,l} = X^T \tilde{E}_{\cdot,\cdot,l} X$.

*The **maximum likelihood** estimates, i.e. cross-entropy, of respective variables are as follows:*

$$\log p(A'|\mathbf{z}) = \\ = 1/k \sum_a A'_{a,a} \log \tilde{A}_{a,a} + (1 - A'_{a,a}) \log(1 - \tilde{A}_{a,a}) + \\ + 1/k(k-1) \sum_{a \neq b} A'_{a,b} \log \tilde{A}_{a,b} + (1 - A'_{a,b}) \log(1 - \tilde{A}_{a,b})$$

$$\log p(F|\mathbf{z}) = 1/n \sum_i \log F^T_{i,\cdot} \tilde{F}'_{i,\cdot}$$

$$\log p(E|\mathbf{z}) = 1/(||A||_1 - n) \sum_{i \neq j} \log E^T_{i,j,\cdot} \tilde{E}'_{i,j,\cdot}, \quad (2)$$

The overall **reconstruction loss** is a weighed sum of the previou

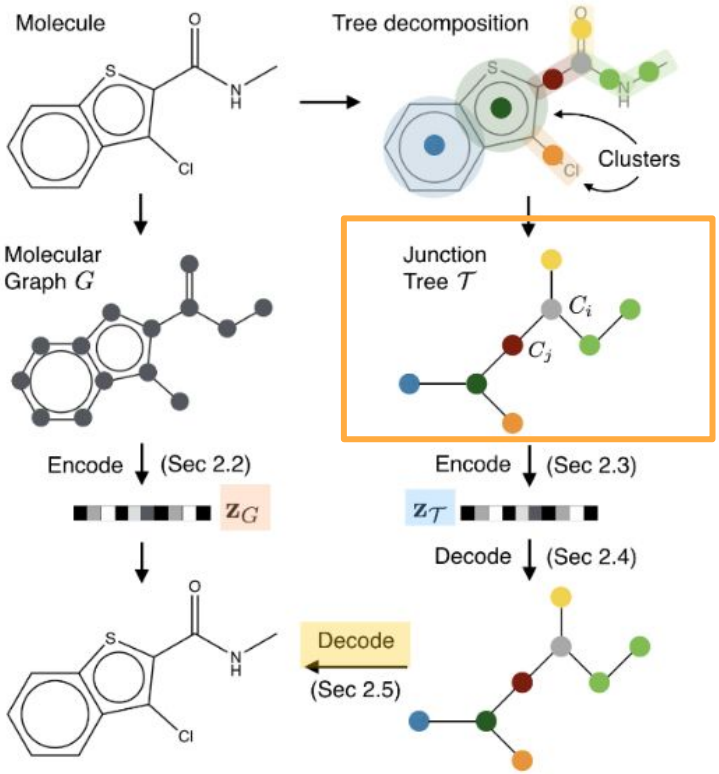
$$-\log p(G|\mathbf{z}) = -\lambda_A \log p(A'|\mathbf{z}) - \lambda_F \log p(F|\mathbf{z}) - \\ - \lambda_E \log p(E|\mathbf{z}) \quad (3)$$

JTVAE

paper: arxiv 1802.04364

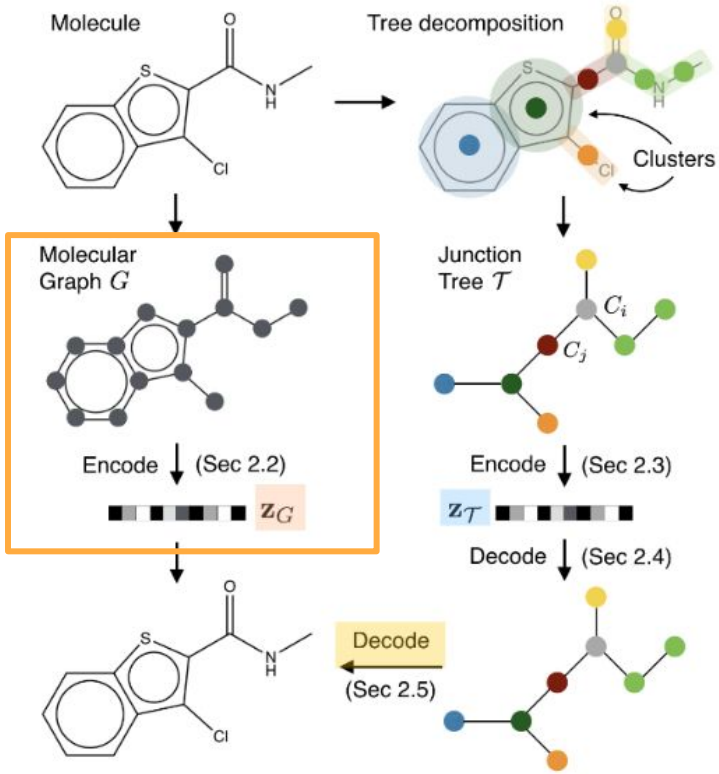
29 Mar 2019

JTVAE - Junction Tree



A tree decomposition maps a graph G into a junction tree by contracting certain vertices into a single node so that G becomes cycle-free.

JTVAE - Graph Encoder



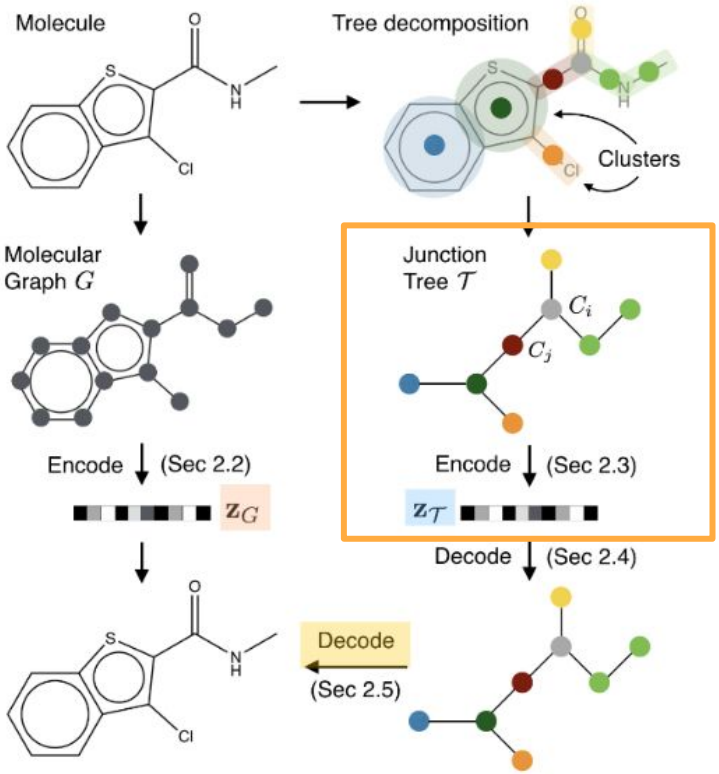
$$\boldsymbol{\nu}_{uv}^{(t)} = \tau(\mathbf{W}_1^g \mathbf{x}_u + \mathbf{W}_2^g \mathbf{x}_{uv} + \mathbf{W}_3^g \sum_{w \in N(u) \setminus v} \boldsymbol{\nu}_{wu}^{(t-1)}) \quad (1)$$

$$\mathbf{h}_u = \tau(\mathbf{U}_1^g \mathbf{x}_u + \sum_{v \in N(u)} \mathbf{U}_2^g \boldsymbol{\nu}_{vu}^{(T)}) \quad (2)$$

The final graph representation is $\mathbf{h}_G = \sum_i \mathbf{h}_i / |V|$.

message passing through edges
compute the messages of latent vector of each node

JTVAE - Tree Encoder



top-down,bottom up
GRU over tree nodes

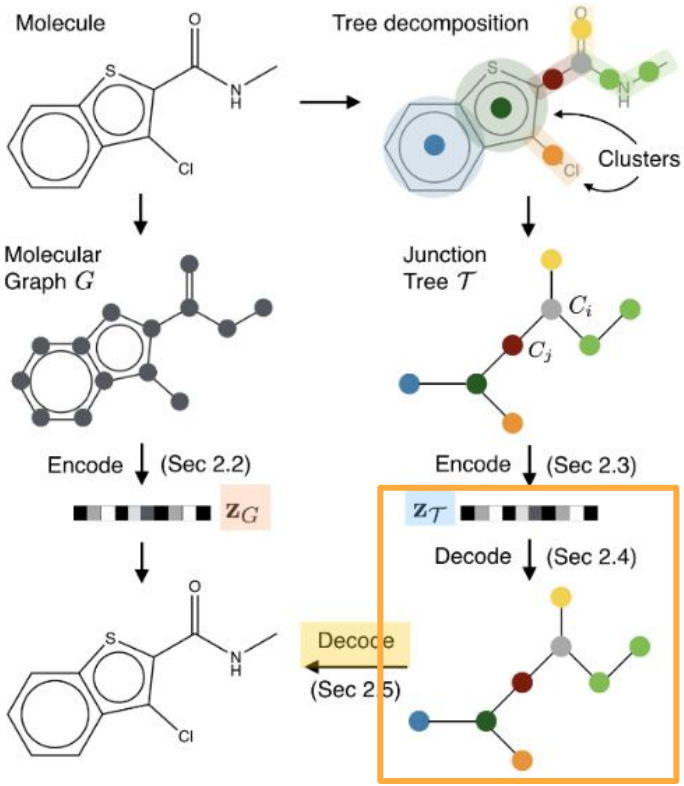
$$\mathbf{m}_{ij} = \text{GRU}(\mathbf{x}_i, \{\mathbf{m}_{ki}\}_{k \in N(i) \setminus j}) \qquad (3)$$

After the message passing, we obtain the latent representation of each node hi by aggregating its inward messages:

$$\mathbf{h}_i = \tau(\mathbf{W}^o \mathbf{x}_i + \sum_{k \in N(i)} \mathbf{U}^o \mathbf{m}_{ki}) \qquad (9)$$

The final tree representation is $\mathbf{h}_{\mathcal{T}_G} = \mathbf{h}_{root}$,

JTVAE - Tree Decoder



topological prediction:

$$p_t = \sigma(\mathbf{u}^d \cdot \tau(\mathbf{W}_1^d \mathbf{x}_{i_t} + \mathbf{W}_2^d \mathbf{z}_\mathcal{T} + \mathbf{W}_3^d \sum_{(k,i_t)\in\tilde{\mathcal{E}}_t} \mathbf{h}_{k,i_t}) \quad (11)$$

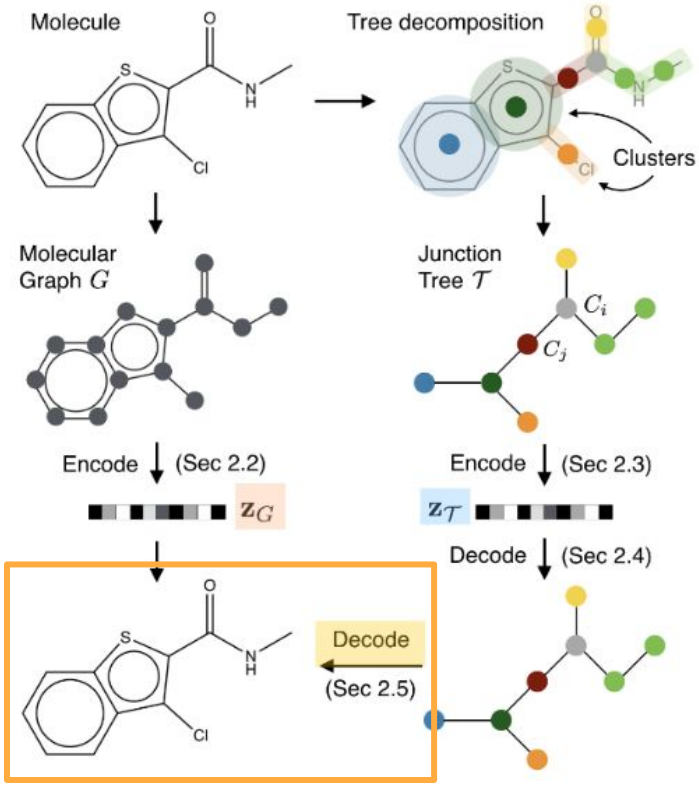
new child's label prediction:

$$\mathbf{q}_j = \text{softmax}(\mathbf{U}^l \tau(\mathbf{W}_1^l \mathbf{z}_\mathcal{T} + \mathbf{W}_2^l \mathbf{h}_{ij})) \quad (12)$$

Let p and q be the ground truth topological and label values, the decoder minimizes the following cross entropy loss:

$$\mathcal{L}_c(\mathcal{T}) = \sum_t \mathcal{L}^d(p_t, \hat{p}_t) + \sum_j \mathcal{L}^l(\mathbf{q}_j, \hat{\mathbf{q}}_j) \quad (13)$$

JTVAE - Graph Decoder



Let $\mathcal{G}(\mathcal{T})$ be the set of graphs whose junction tree is $\mathcal{T}$. Decoding graph $\hat{G}$ from $\hat{\mathcal{T}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$ is a structured prediction:

$$\hat{G} = \arg \max_{G' \in \mathcal{G}(\hat{\mathcal{T}})} f^a(G') \quad (14)$$

scoring function

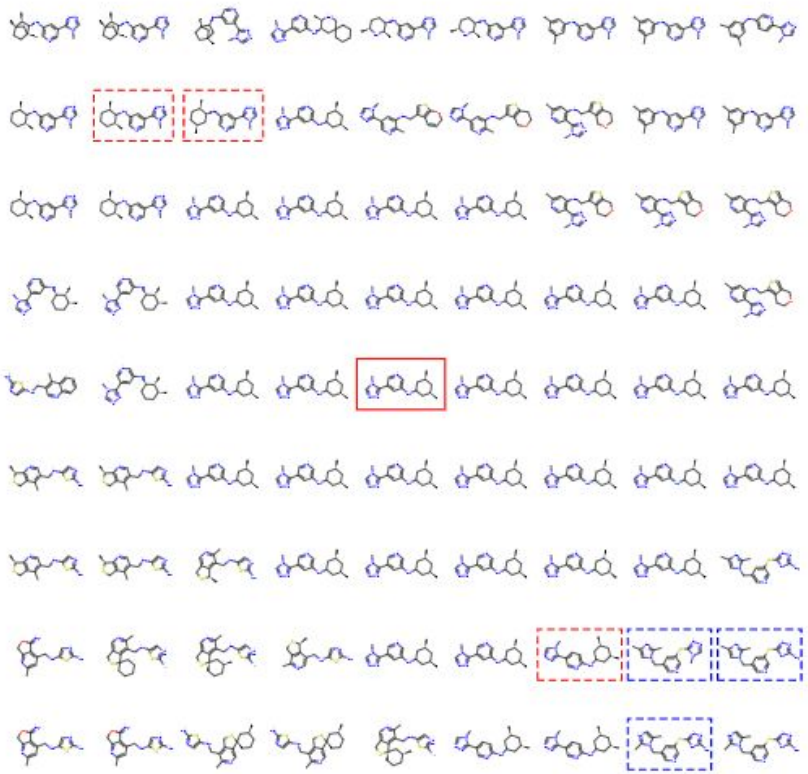
$$f_i^a(G_i) = \mathbf{h}_{G_i} \cdot \mathbf{z}_G$$

$$\mathcal{L}_g(G) = \sum_i \left[f^a(G_i) - \log \sum_{G_i' \in \mathcal{G}_i} \exp(f^a(G_i')) \right] \quad (16)$$

maximize the log-likelihood of predicting correct subgraphs Gi of the ground true graph G at each tree node. We again apply teacher forcing, i.e. we feed the graph decoder with ground truth trees as input.

JTVAE - Graph Decoder

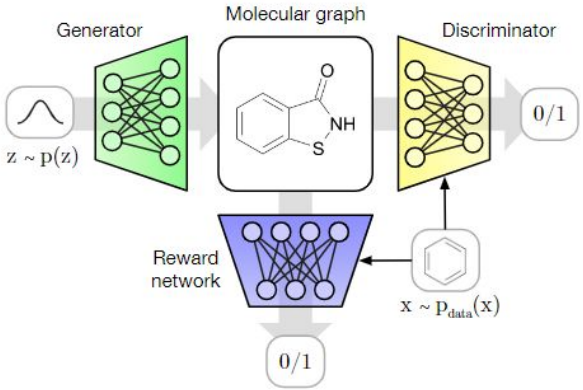
same tree structure
but different moleculars



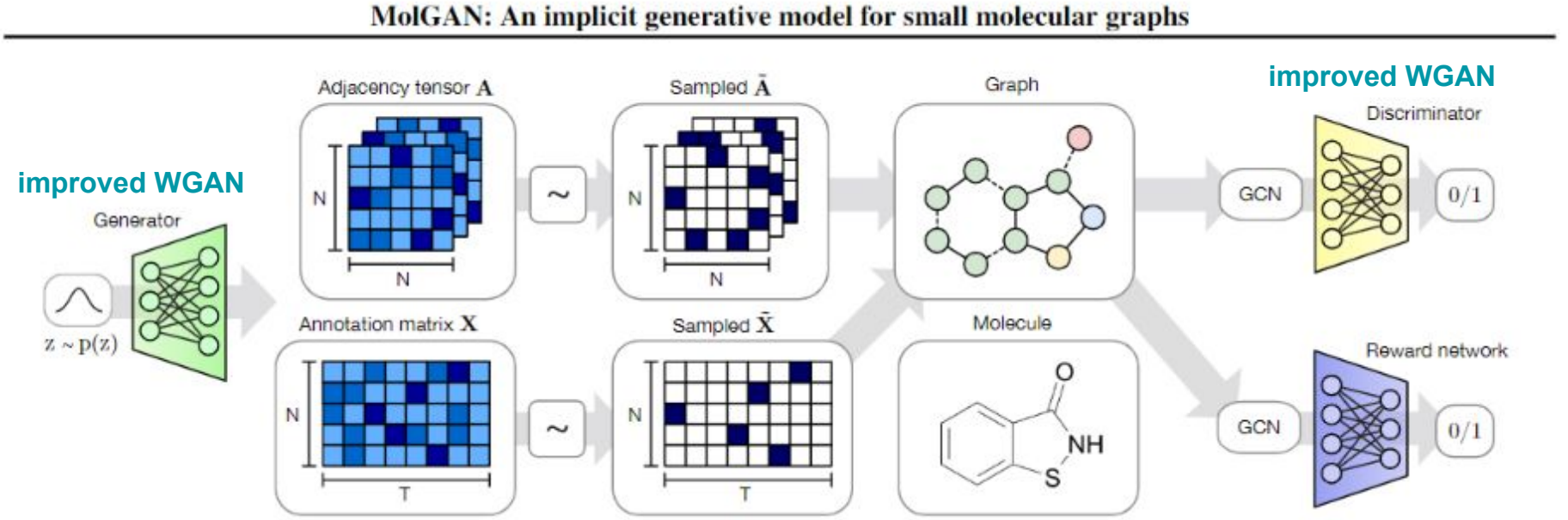
MolGAN

paper: arxiv 1805.11973

30 May 2018



MolGAN

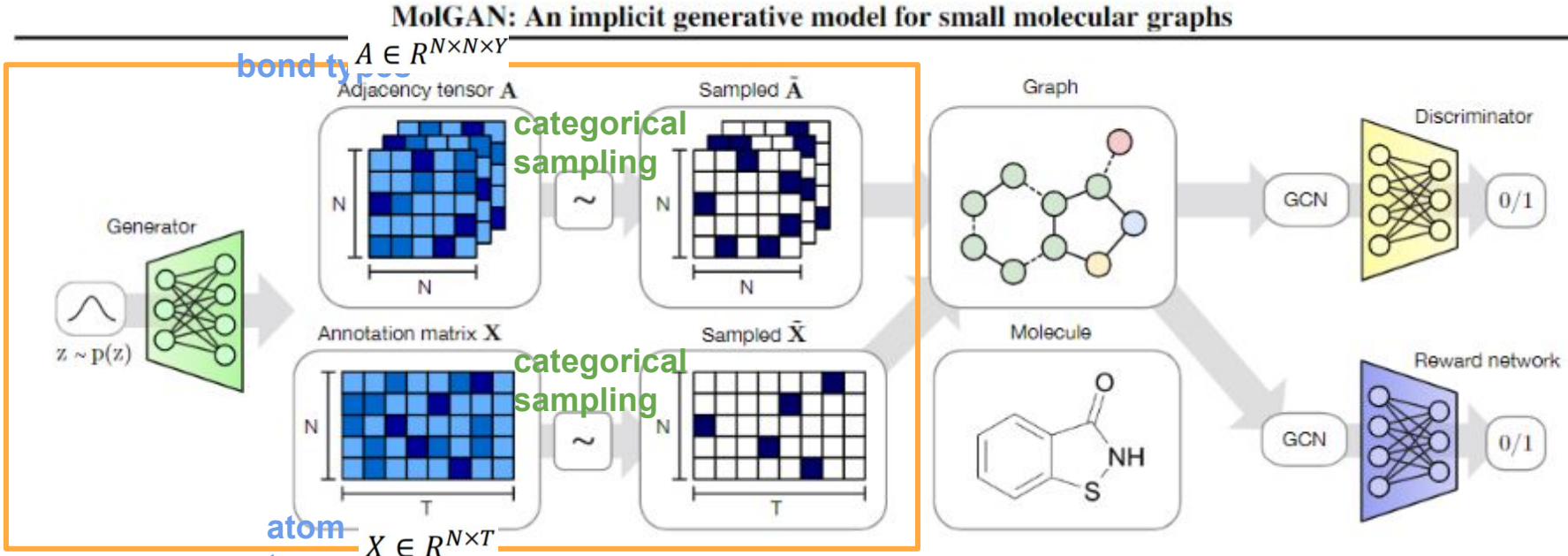


implicit, likelihood-free.
using RL for desired properties.

loss

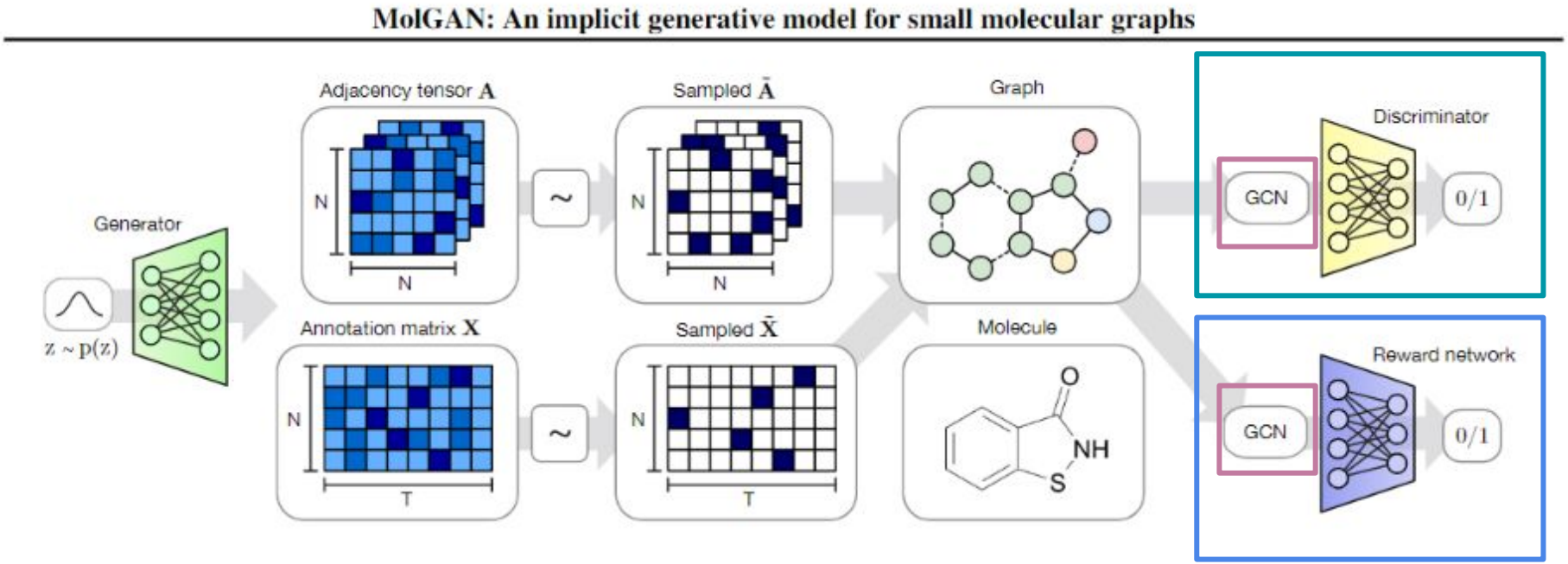
$$L(\theta) = \lambda \cdot L_{WGAN} + (1 - \lambda) \cdot L_{RL}$$

MolGAN - Generator



Generator: predicts the entire graph at once using a simple multi-layer perceptron (MLP). While this limits our study to graphs of a pre-chosen maximum size, we find that it is significantly faster and easier to optimize.

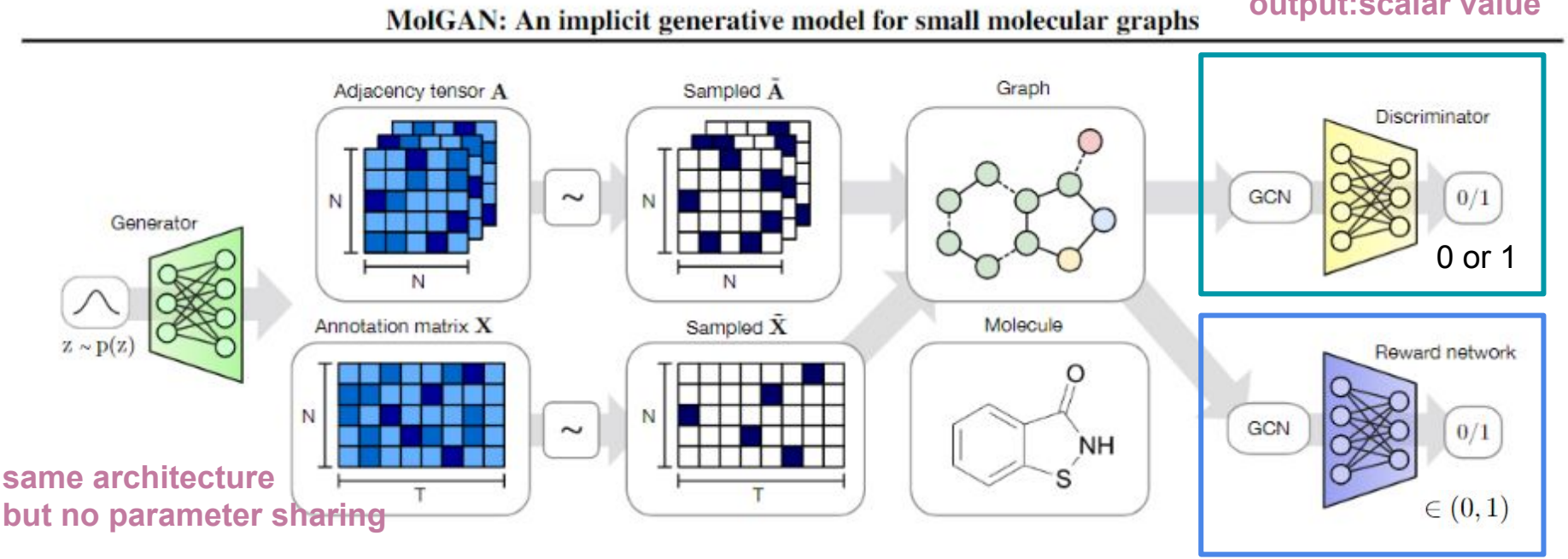
MolGAN - Discriminator & Reward Network



Relational - GCN
support for
multiple edge types

MolGAN - Discriminator & Reward Network

input: graph
output:scalar value



same architecture
but no parameter sharing

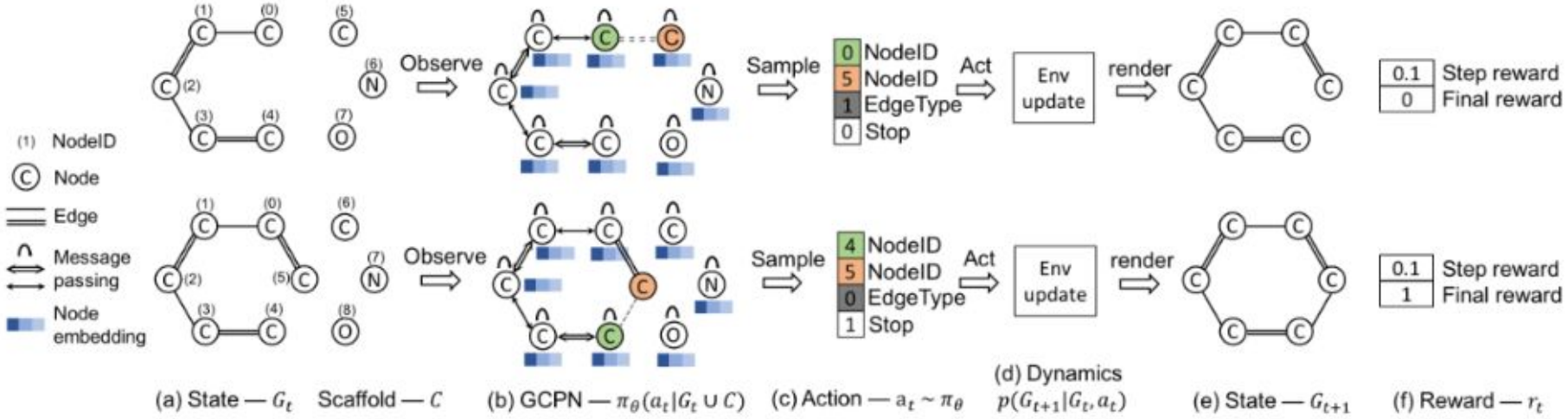
Discriminator: samples from the dataset and the generator -> learns to distinguish them
Reward Network: dataset and generated samples are inputs -> assigns scores to them (using RDKit)

GCPN

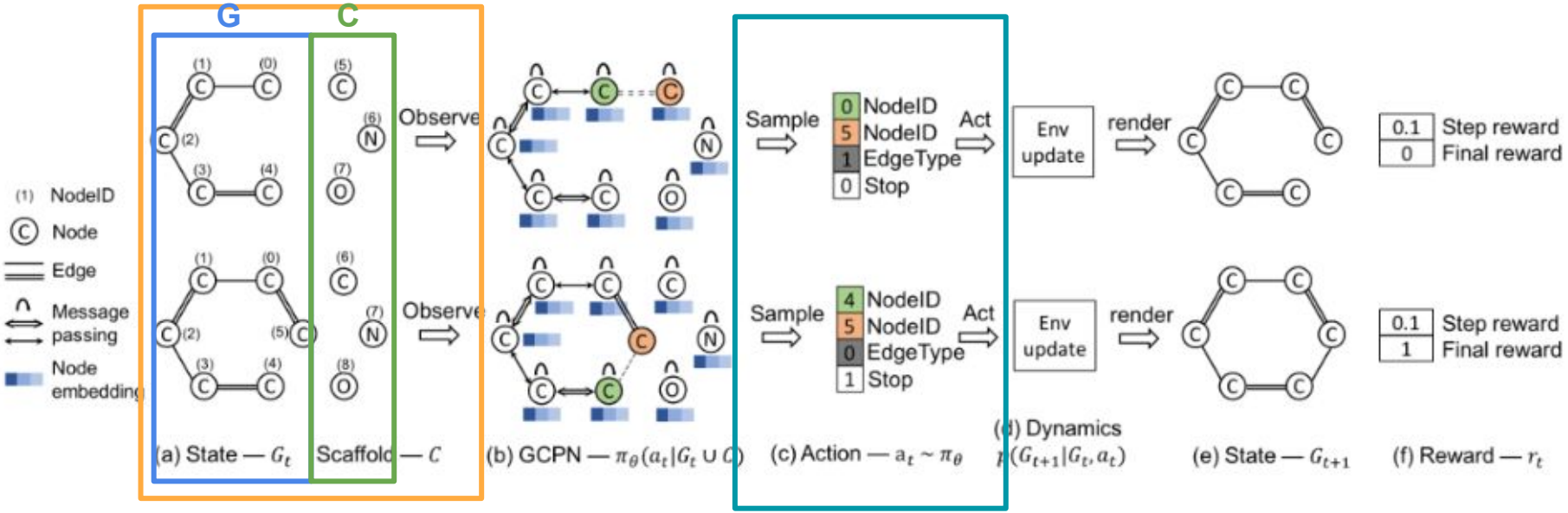
paper

2018

GCPN


(1) NodeID
Node
Edge
Message passing
Node embedding
Observe
Sample
0 NodeID
5 NodeID
1 EdgeType
0 Stop
Act
Env update
render
0.1 Step reward
0 Final reward
4 NodeID
5 NodeID
0 EdgeType
1 Stop
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

GCPN - State Space & Action Space

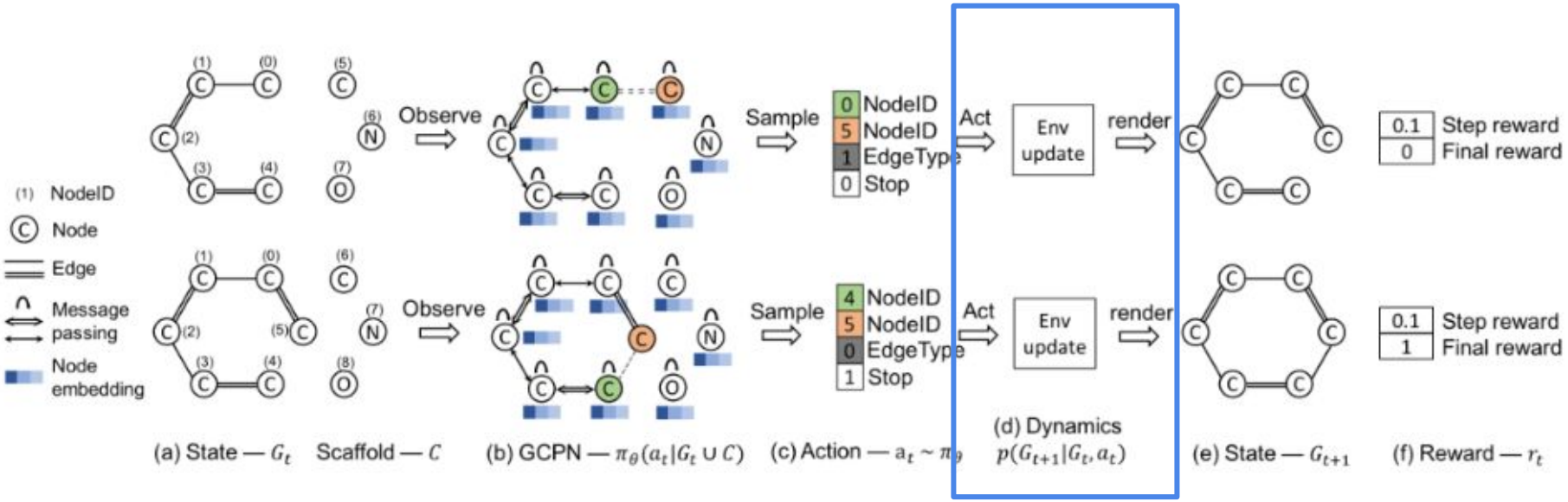


State space: 初始值給定一個碳元素node

Action space: 視作link prediction a set of scaffold subgraphs $\{C_1, \ldots, C_s\}$

- Connect a new subgraph Ci to a node in Gt or
- connecting existing nodes within graph Gt

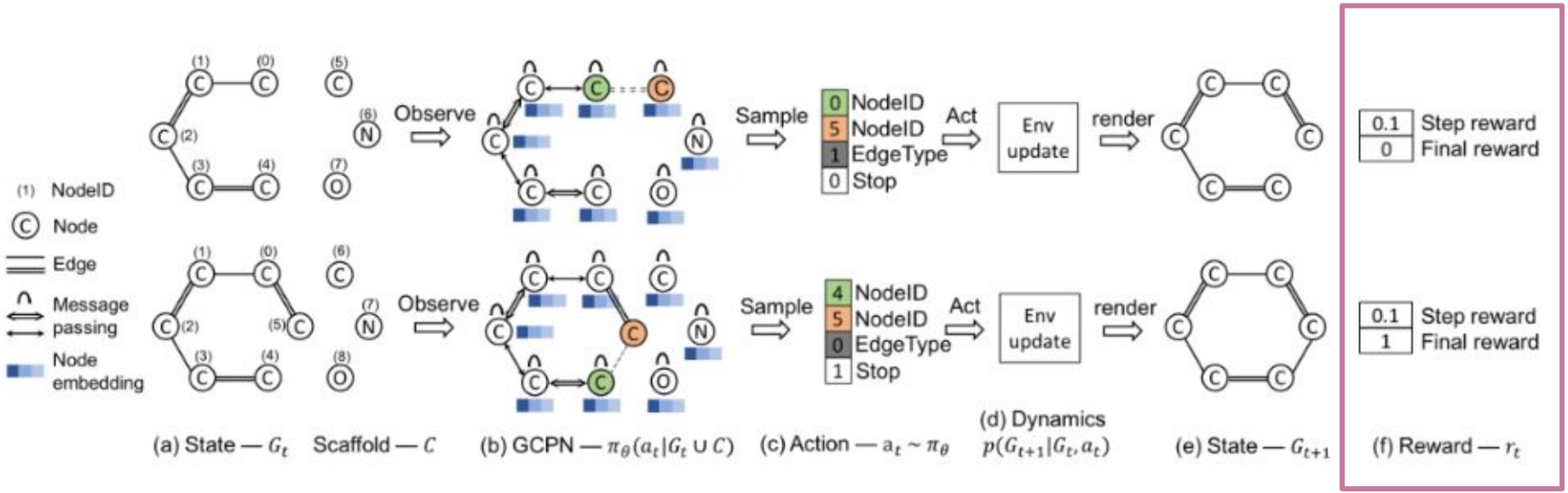
GCPN - State Transition Dynamics



State Transition Dynamics: 考量Domain-specific rules, step-wise valency check in incompleted graph

- obey the given rules or
- be rejected and the state remains unchanged

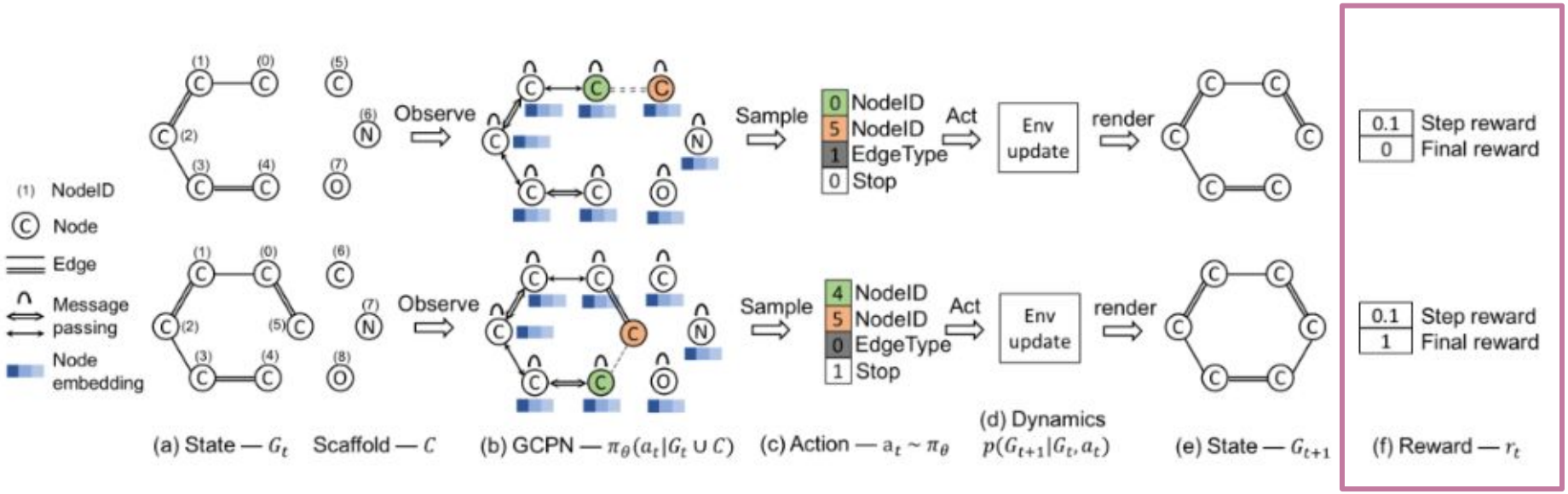
GCPN - Reward Design



Reward Design:

- intermediate rewards:
  - step-wise validity rewards and adversarial rewards
- final rewards
  - a sum over domain-specific rewards and adversarial rewards

GCPN - Reward Design



Reward Design: the adversarial rewards

$$\min_\theta \max_\phi V(\pi_\theta, D_\phi) = \mathbb{E}_{x\sim p_{data}}[\log D_\phi(x)] + \mathbb{E}_{x\sim \pi_\theta}[\log D_\phi(1-x)]$$

GCPN - Graph Convolutional Policy Network

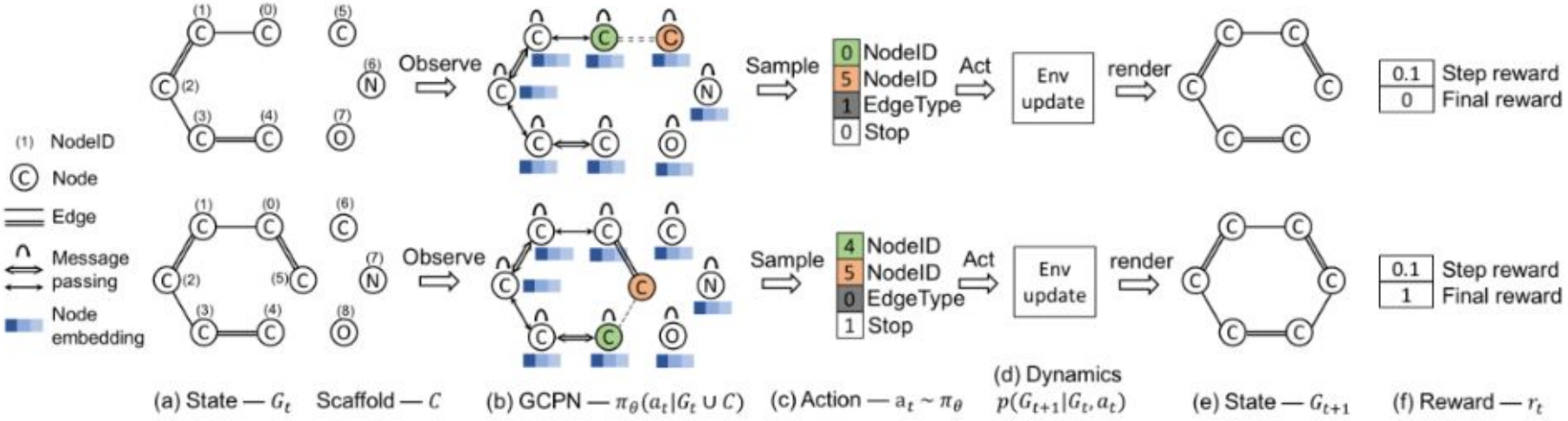
computing node embeddings using GCN

then make an action prediction

$$a_t = \text{CONCAT}(a_{\text{first}}, a_{\text{second}}, a_{\text{edge}}, a_{\text{stop}}) \tag{3}$$

$$\begin{aligned}
f_{\text{first}}(s_t) &= \text{SOFTMAX}(m_f(X)), & a_{\text{first}} &\sim f_{\text{first}}(s_t) \in \{0,1\}^n \\
f_{\text{second}}(s_t) &= \text{SOFTMAX}(m_s(X_{a_{\text{first}}}, X)), & a_{\text{second}} &\sim f_{\text{second}}(s_t) \in \{0,1\}^{n+c} \\
f_{\text{edge}}(s_t) &= \text{SOFTMAX}(m_e(X_{a_{\text{first}}}, X_{a_{\text{second}}})), & a_{\text{edge}} &\sim f_{\text{edge}}(s_t) \in \{0,1\}^b \\
f_{\text{stop}}(s_t) &= \text{SOFTMAX}(m_t(\text{AGG}(X))), & a_{\text{stop}} &\sim f_{\text{stop}}(s_t) \in \{0,1\}
\end{aligned} \tag{4}$$

GCPN - Policy Gradient Training



Proximal Policy Optimization (PPO) & objective function during pretraining (supervised)

$$\max L^{\text{CLIP}}(\theta) = \mathbb{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)], r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \quad (5)$$

$$\min L^{\text{EXPERT}}(\theta) = -\log(\pi_\theta(a_t|s_t))$$ where (st;at) pairs are obtained from ground truth molecules.

GCPN - Experiments

specified properties are optimized

Property Optimization. The task is to generate novel molecules whose specified molecular properties are optimized. This can be useful in many applications such as drug discovery and materials science, where the goal is to identify molecules with highly optimized properties of interest.

properties are as close to the target score

Property Targeting. The task is to generate novel molecules whose specified molecular properties are as close to the target scores as possible. This is crucial in generating virtual libraries of molecules with properties that are generally suitable for a desired application. For example, a virtual molecule library for drug discovery should have high drug-likeness and synthesizability.

Constrained Property Optimization. The task is to generate novel molecules whose specified molecular properties are optimized, while also containing a specified molecular substructure. This can be useful in lead optimization problems in drug discovery and materials science, where we want to make modifications to a promising lead molecule and improve its properties [2].

specified properties are optimized,
and also containing specified substructure

GCPN[98]利用RL生成目標導向的分子圖，以處理不可導目標和約束。實驗結果證明了GCPN在各種圖生成問題中的有效性。

MolGAN[99]也採用了類似的思想，即使用RL生成分子圖。MolGAN建議直接生成完整的圖，而不是通過一系列的動作來生成圖，這對小分子很有效。