Neural Processes on Graphs

Huidong (Leon) Liang

Discipline of Business Analytics
The University of Sydney Business School

June 2021

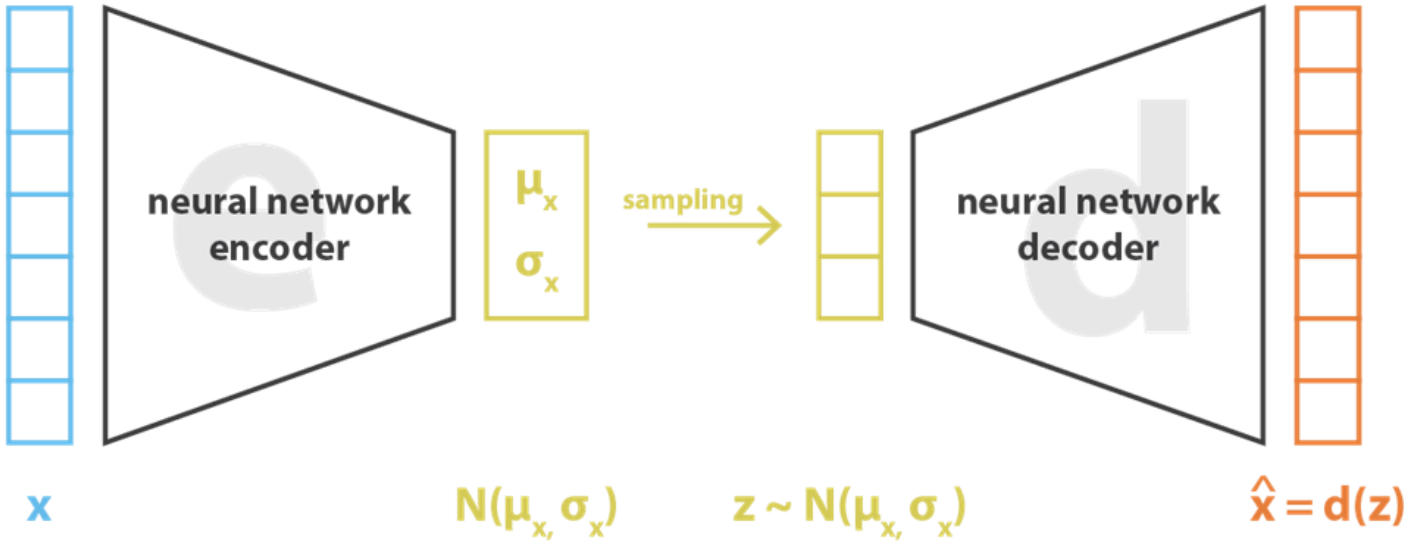
Overview

1. Neural Processes
   - Variational Autoencoders
   - Meta-learning
   - Gaussian Processes
   - Neural Processes
2. Neural Processes on Graphs
   - Definition of Network/Graph
   - Spectral Graph Convolution
   - Variational Graph Autoencoders
   - Graph Neural Processes

Neural Processes

Related Work: Variational Autoencoders

Model Structure



$$\boldsymbol{\mu} = \mathrm{ReLU}(\mathrm{W}_{\boldsymbol{\mu}}\mathrm{ReLU}(\mathrm{W}_1\mathrm{x} + \mathrm{b}_1) + b_\mu) \tag{1}$$

$$\log \boldsymbol{\sigma} = \mathrm{ReLU}(\mathrm{W}_{\boldsymbol{\sigma}}\mathrm{ReLU}(\mathrm{W}_1\mathrm{x} + \mathrm{b}_1) + b_\sigma) \tag{2}$$

$$\log p(\mathrm{x}|\mathrm{z}) = \sum_{i=1}^{d} x_i \log \hat{x}_i + (1 - x_i) \log(1 - \hat{x}_i) \tag{3}$$

$$\textit{where} \quad \hat{\mathrm{x}} = \sigma(\mathrm{W}_3\sigma(\mathrm{W}_2\mathrm{z} + \mathrm{b}_2) + \mathrm{b}_3) \tag{4}$$

Related Work: Variational Autoencoders

Inference

- Bayesian Inference

$$p(\mathrm{z}|\mathrm{x}) = \frac{p(\mathrm{x}|\mathrm{z})p(\mathrm{z})}{p(\mathrm{x})} \propto p(\mathrm{x}|\mathrm{z})p(\mathrm{z}) \tag{5}$$

- Variational Inference
  We want to use a variational distribution $q(\mathrm{z}|\mathrm{x})$ to approximate $p(\mathrm{z}|\mathrm{x})$ by minimizing the Kullback-Leibler (KL) Divergence:

$$\mathrm{KL}\big(q(\mathrm{z}|\mathrm{x})||p(\mathrm{z}|\mathrm{x})\big) = \mathbb{E}_{q(\mathrm{z}|\mathrm{x})}\left[\log \frac{q(\mathrm{z}|\mathrm{x})}{p(\mathrm{z}|\mathrm{x})}\right]$$

Related Work: Variational Autoencoders

Inference

- We can also express KL Divergence as:

$$\log p(\mathrm{x}) = \mathbb{E}_{q(\mathrm{z}|\mathrm{x})}[\log p(\mathrm{x})] \tag{6}$$

$$= \mathbb{E}_{q(\mathrm{z}|\mathrm{x})}\left[\log \frac{p(\mathrm{x},\mathrm{z})}{p(\mathrm{z}|\mathrm{x})}\right] \tag{7}$$

$$= \mathbb{E}_{q(\mathrm{z}|\mathrm{x})}\left[\log \left[\frac{p(\mathrm{x},\mathrm{z})}{q(\mathrm{z}|\mathrm{x})}\frac{q(\mathrm{z}|\mathrm{x})}{p(\mathrm{z}|\mathrm{x})}\right]\right] \tag{8}$$

$$= \mathbb{E}_{q(\mathrm{z}|\mathrm{x})}\left[\log \frac{p(\mathrm{x},\mathrm{z})}{q(\mathrm{z}|\mathrm{x})}\right] + \mathbb{E}_{q(\mathrm{z}|\mathrm{x})}\left[\log \frac{q(\mathrm{z}|\mathrm{x})}{p(\mathrm{z}|\mathrm{x})}\right] \tag{9}$$

- We define $\mathcal{L} = \mathbb{E}_{q(\mathrm{z}|\mathrm{x})}\left[\log \frac{p(\mathrm{x},\mathrm{z})}{q(\mathrm{z}|\mathrm{x})}\right]$ as the Variational Lower Bound.

- Now we rearrange the above equations to have a new form for $\mathcal{L}$:

$$\mathcal{L} = \mathbb{E}_{q(\mathrm{z}|\mathrm{x})}\Big[\log p(\mathrm{x})\Big] - \mathbb{E}_{q(\mathrm{z}|\mathrm{x})}\left[\log \frac{q(\mathrm{z}|\mathrm{x})}{p(\mathrm{z}|\mathrm{x})}\right] \tag{10}$$

$$= \mathbb{E}_{q(\mathrm{z}|\mathrm{x})}\left[\log \frac{p(\mathrm{x}|\mathrm{z})p(\mathrm{z})}{p(\mathrm{z}|\mathrm{x})}\right] - \mathbb{E}_{q(\mathrm{z}|\mathrm{x})}\left[\log \frac{q(\mathrm{z}|\mathrm{x})}{p(\mathrm{z}|\mathrm{x})}\right] \tag{11}$$

$$= \mathbb{E}_{q(\mathrm{z}|\mathrm{x})}\Big[\log p(\mathrm{x}|\mathrm{z})\Big] - \mathbb{E}_{q(\mathrm{z}|\mathrm{x})}\left[\log \frac{q(\mathrm{z}|\mathrm{x})}{p(\mathrm{z})}\right] \tag{12}$$

$$= \mathbb{E}_{q(\mathrm{z}|\mathrm{x})}\Big[\log p(\mathrm{x}|\mathrm{z})\Big] - \mathbf{KL}\Big(q(\mathrm{z}|\mathrm{x})||p(\mathrm{z})\Big) \tag{13}$$

Related Work: Variational Autoencoders

Inference

- To optimize $\mathcal{L}$, we need to calculate the closed-form gradient $\nabla\mathcal{L}$, which requires the *reparameterization trick* when sampling z:

$$\mathbb{E}_{q(\mathrm{z}|\mathrm{x})}[\log p(\mathrm{x}|\mathrm{z})] \simeq \frac{1}{L}\sum_{l=1}^{L} \log p(\mathrm{x}|\mathrm{z}^{(\ell)}) \tag{14}$$

$$\mathrm{z}^{(\ell)} = \boldsymbol{\mu} + \boldsymbol{\sigma}\epsilon^{(\ell)}, \quad \text{with} \quad \epsilon^{(\ell)} \sim \mathcal{N}(0,1) \tag{15}$$

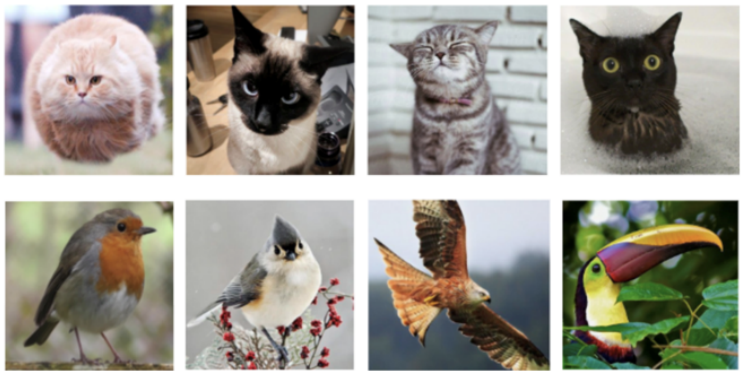
- Here we use an auxiliary noise variable $\epsilon \sim \mathcal{N}(0,1)$ and a function $z^{\ell} = \boldsymbol{\mu} + \boldsymbol{\sigma}\epsilon^{(\ell)}$ to replace $z^{\ell} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma)$.

Related Work: Meta-learning

Learning

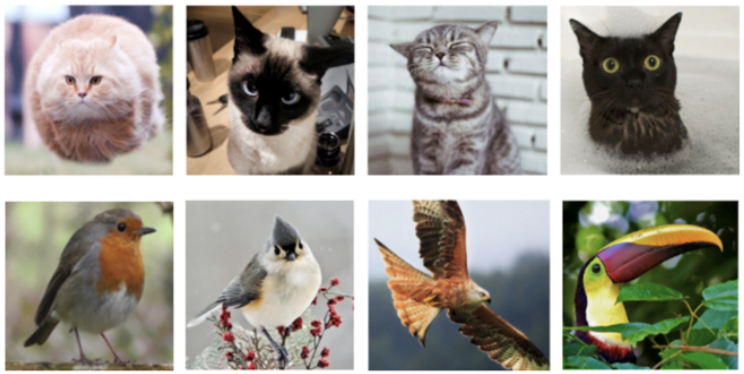
Predicting

Traditional Machine Learning



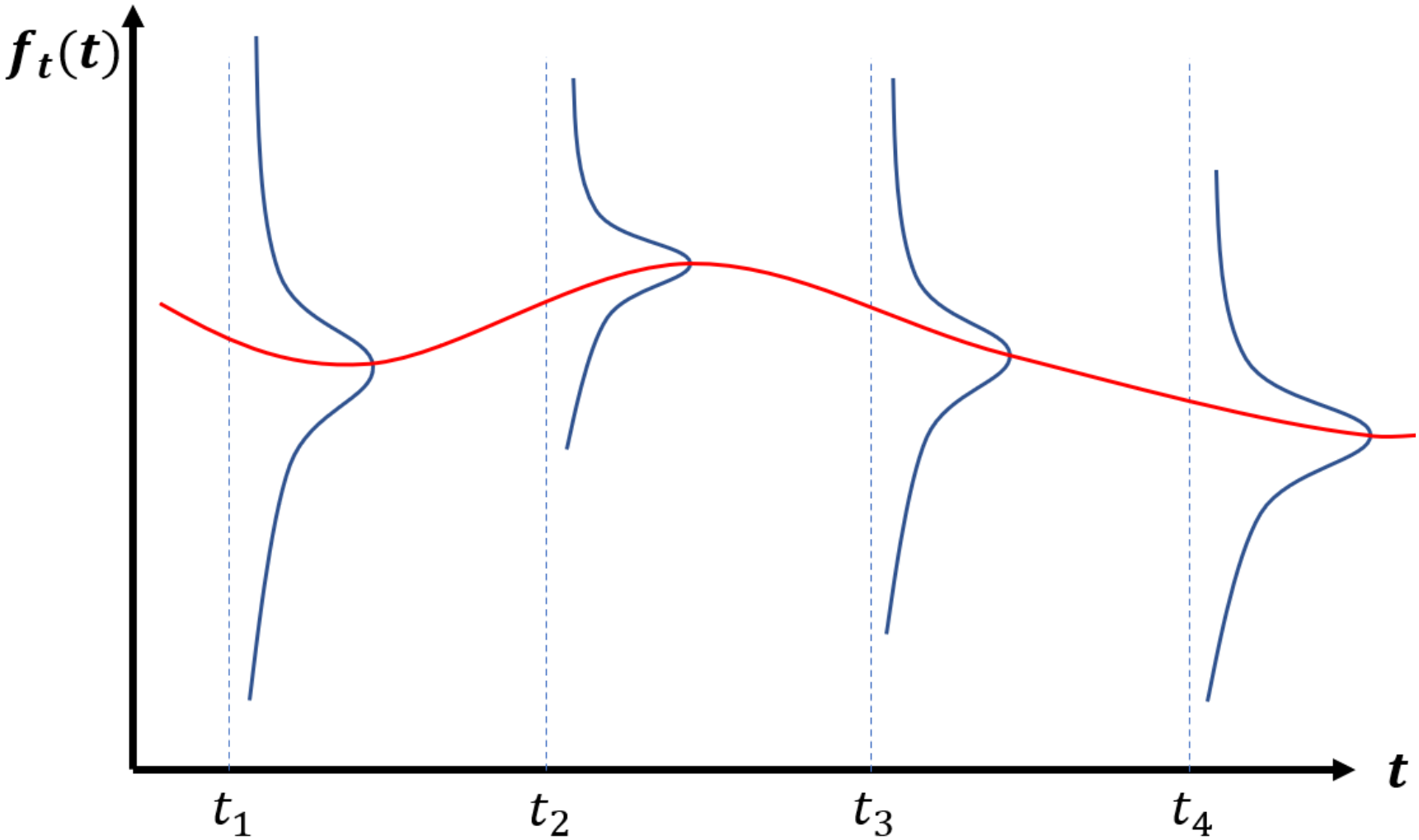
After learning images of cats and birds, traditional machine learning can only predict cats and birds.

Meta-Learning



After learning how to learn cats and birds, meta-learning is able to learn how to predict dogs.

Related Work: Gaussian Processes



$$\{f_t(t)\}_{t\in T} \sim \mathrm{GP}\big(\mu(t), k(t,t')\big)$$

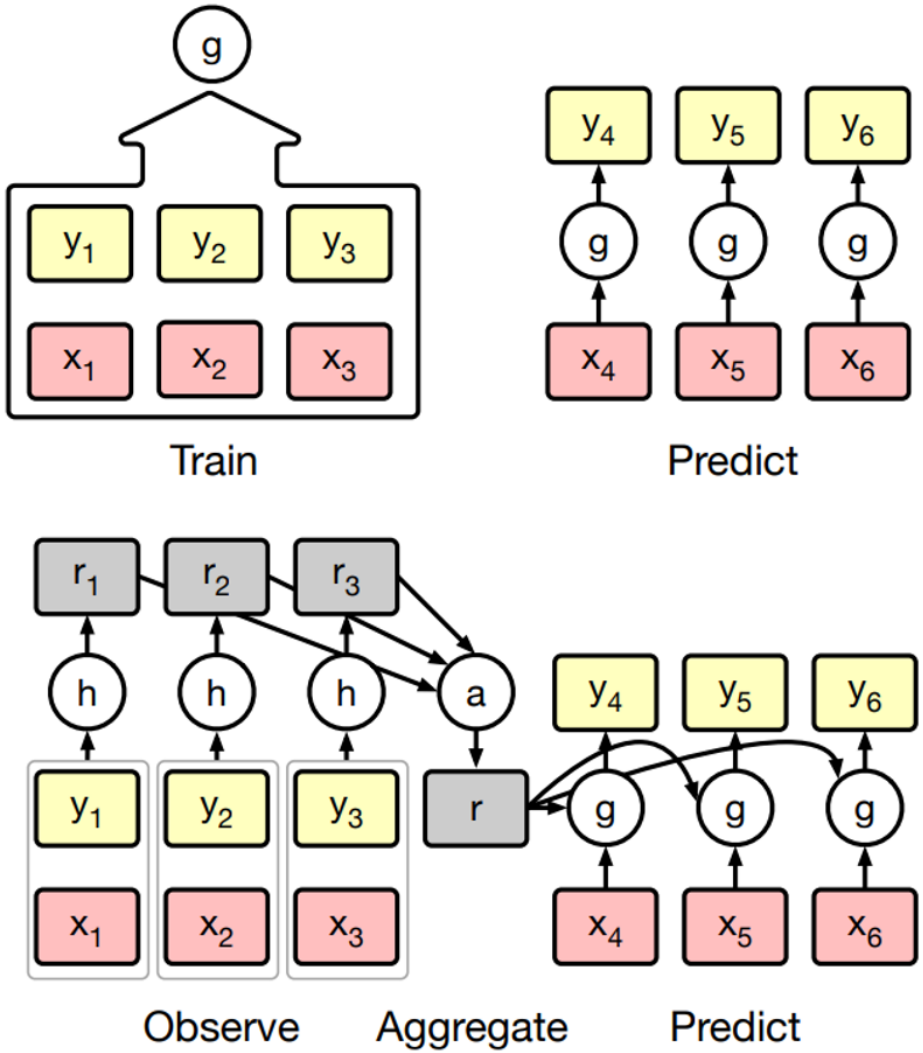
Conditional Neural Processes

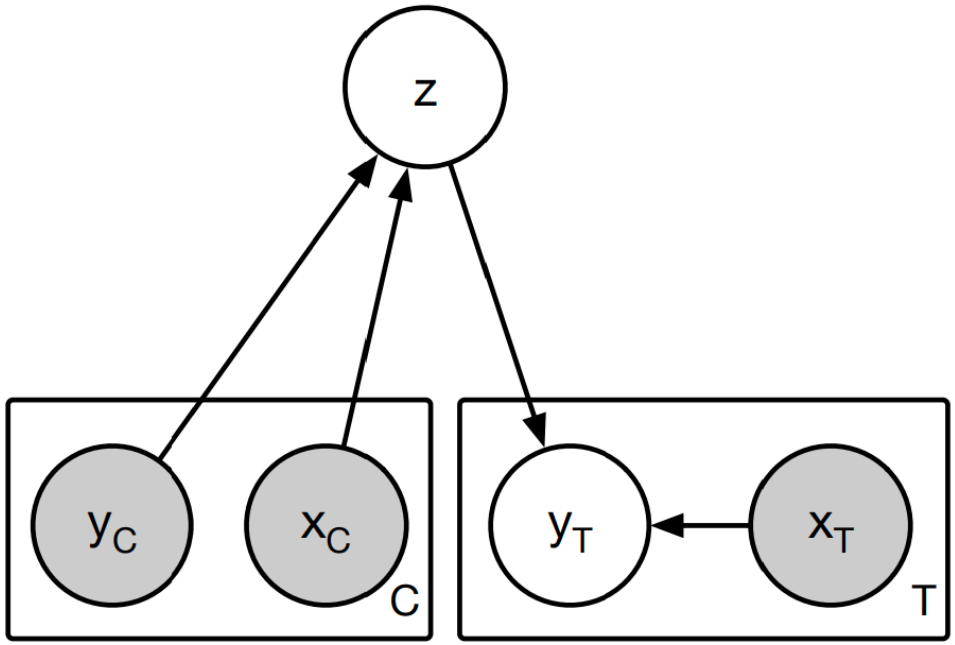
- We have a group of p datasets $\{D_1, \ldots, D_p\}$, and we assume each dataset D_i is generated from a function $y = f_i(x)$.
- $\{f(x)\}_p \sim GP\big(\mu(x), k(x, x')\big)$

  For each $y = f_i(x)$, we sample some x and y pairs.

$$\begin{array}{cccc} x_1 & y_1 & x_4 & y_4 \\ x_2 & y_2 & x_5 & y_5 \\ x_3 & y_3 & x_6 & y_6 \\ f_1(x) & f_2(x) & \cdots & f_p(x) \end{array} \quad \cdots \quad \begin{array}{c} x_{49}\, y_{49} \\ x_{50}\, y_{50} \end{array}$$

- Since we cannot distinguish different datasets all the time, we will then treat them as one large dataset, and we split this complete dataset into two sets:
  - Context Set $C = \{x_C, y_C\}_{i=1}^{n}$
  - Target Set $D = \{x_D, y_D\}_{i=n+1}^{n+m}$

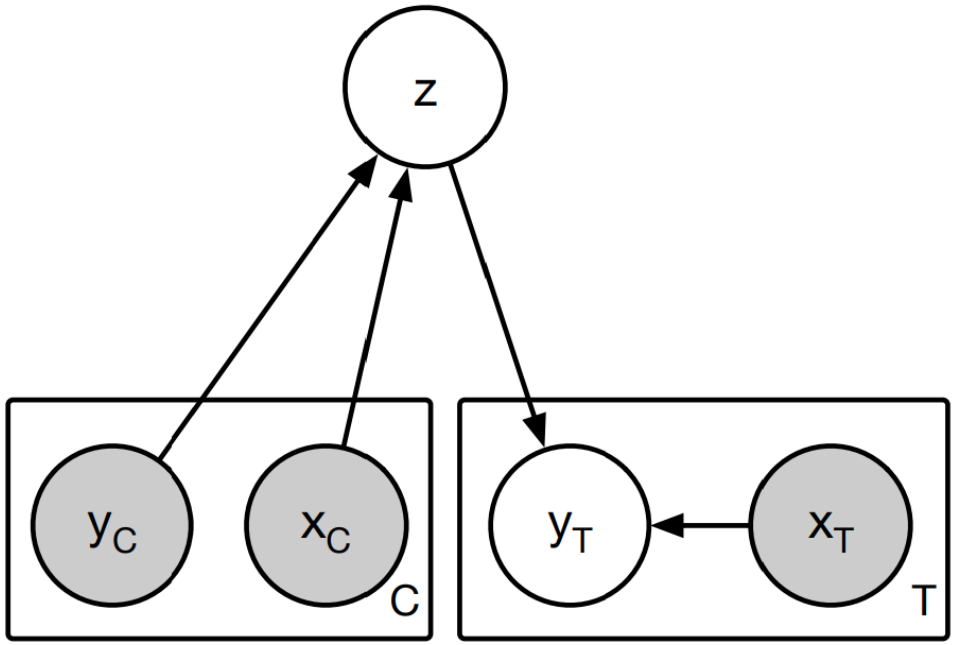




$$\mu_c = \text{ReLU}(W_{\mu_c}\text{ReLU}(W_1[x_C\ y_C] + b_1) + b_\mu) \tag{16}$$

$$\log \sigma_c = \text{ReLU}(W_{\sigma_c}\text{ReLU}(W_1[x_C\ y_C] + b_1) + b_\sigma) \tag{17}$$

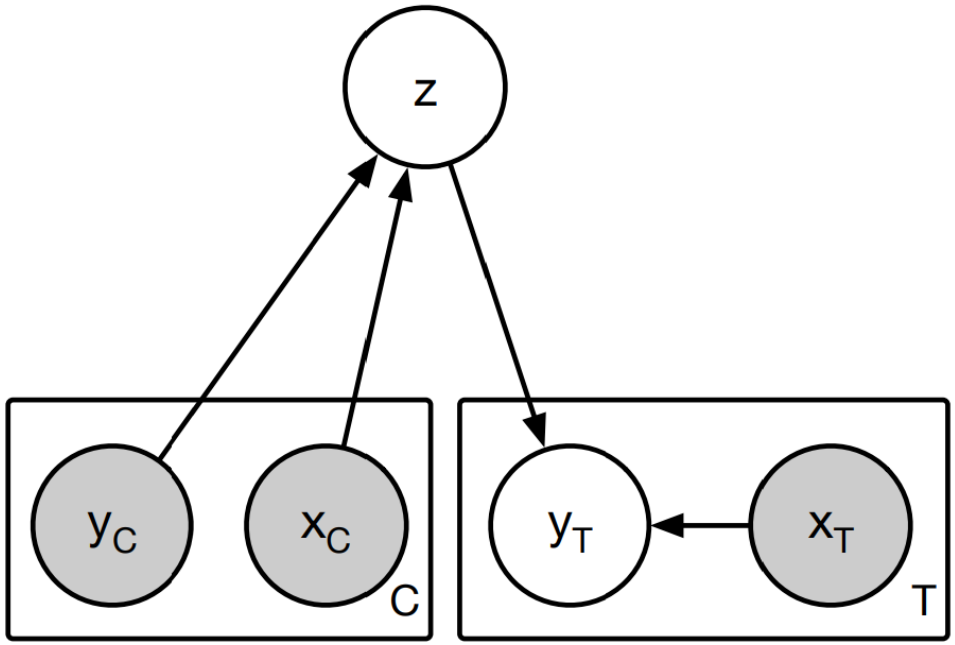
- Here μ_c and σ_c^2 contains means and variances of r_c for each data pair $\{x_C, y_C\}_i$ in the Context Set C.



$$\mu_z(r) = \frac{1}{m}\sum_{i=1}^{m}\mu_{c_i} \qquad \log\sigma_z(r) = \frac{1}{m}\sum_{i=1}^{m}\log\sigma_{c_i} \tag{18}$$

- Use $\mu_z(r)$ and $\sigma_z^2(r)$ to parameterise the distribution for the global latent representation z.

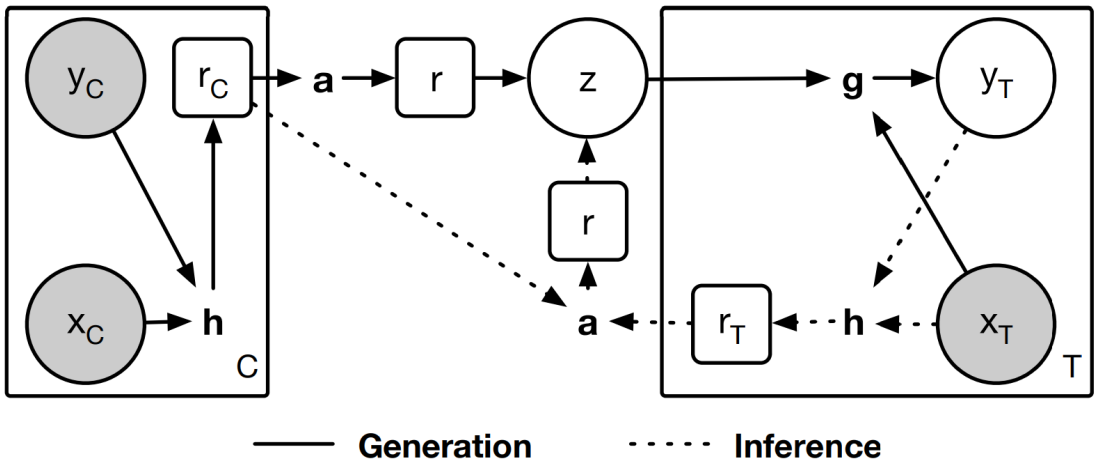
$$q(z|C) = \mathcal{N}(\mu_z(r), \sigma_z^2(r))$$



- Combine a sampled z with every feature $x_{T_i} \in X_T$ as $\tilde{x}_{T_i} \in \tilde{X}_T$, where $\tilde{x}_{T_i} = [x_{T_i}^\top \; z^\top]^\top$.

$$\log p(x|x_T, z) = \sum_{i=1}^{d} y_{T_i} \log \hat{y}_{T_i} + (1 - y_{T_i}) \log(1 - \hat{y}_{T_i}) \tag{19}$$

$$\textit{where} \quad \hat{y}_T = \sigma(W_3 \sigma(W_2 \tilde{x}_{T_i} + b_2) + b_3) \tag{20}$$



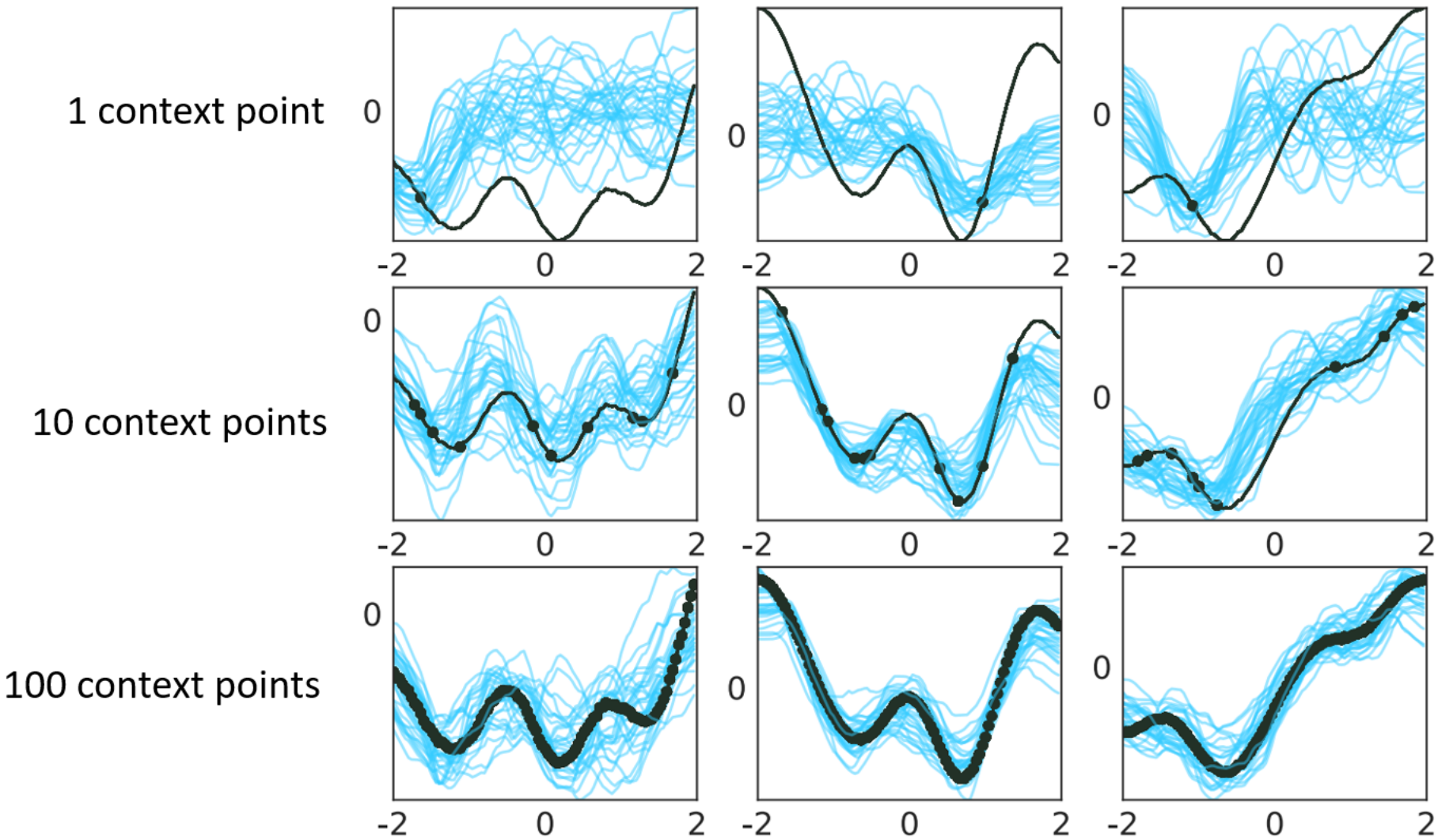
- We want to optimize the variational lower bound $\mathcal{L}$:

$$\mathcal{L} = \mathbb{E}_{q(z|D)}\Big[\log p(y_T|x_T, z)\Big] - \mathbf{KL}\Big(q(z|D)||q(z|C)\Big) \tag{22}$$

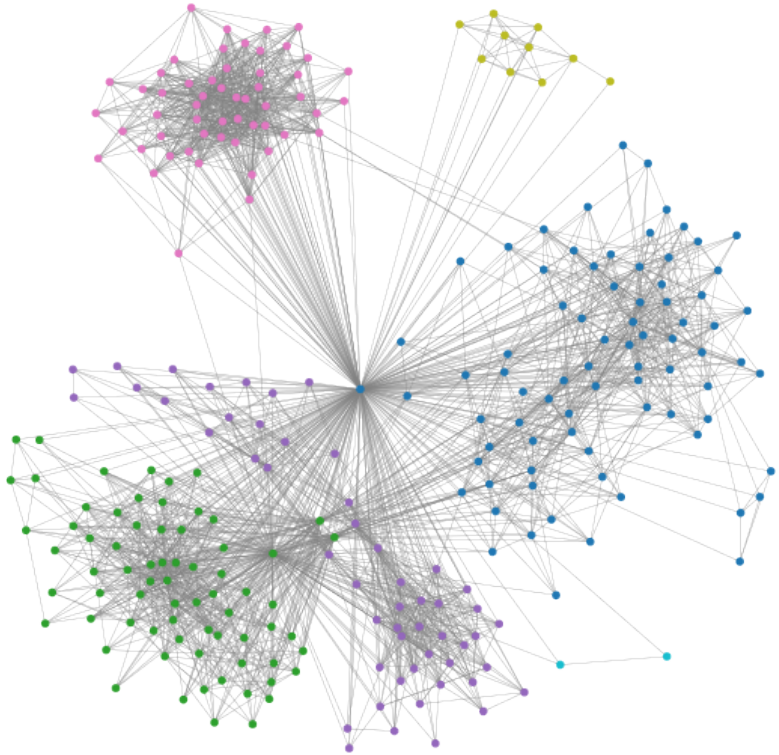
Here $D = C \cup T$, where C is our Context set and T is our Target set.

Neural Processes

An Example on Neural Processes



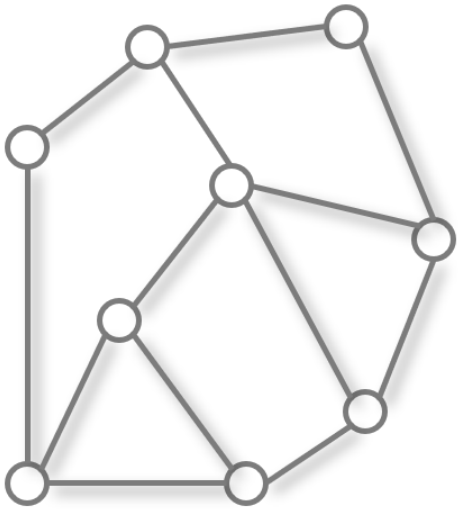
Neural Processes on Graphs



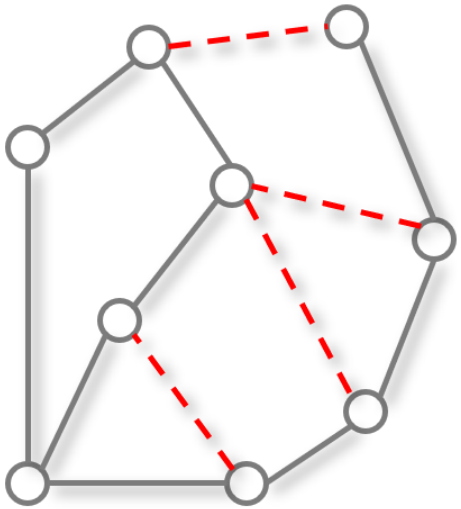
- We define an undirected graph $G = (V, A)$, where V is the set of nodes with features $x_i \in X$ corresponding to each node $v_i \in V$, and the adjacency matrix A with $A_{ij} = 1$ if there is a link between v_i and v_j, and $A_{ij} = 0$ otherwise.

Neural Processes on Graphs

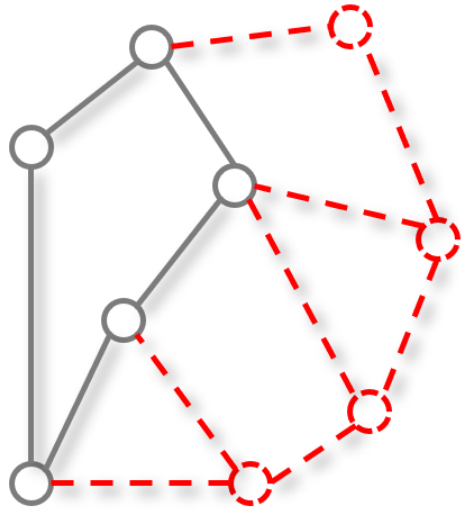
Link Prediction on Network/Graph



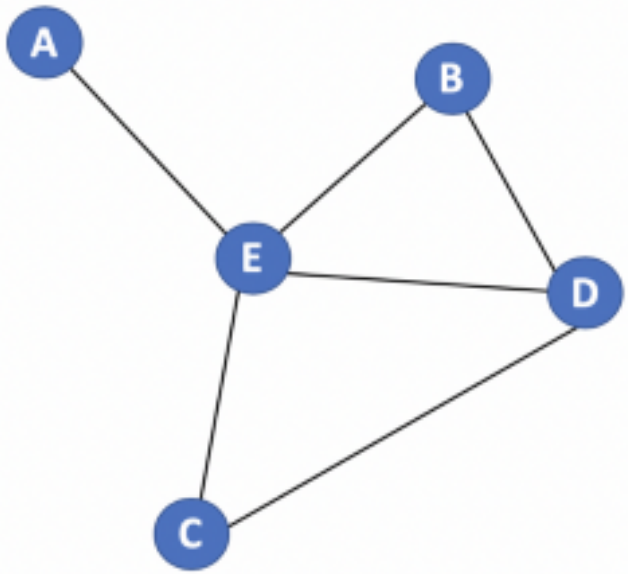
Original Graph



Transductive
Link Prediction



Inductive
Link Prediction



Graph G

	A	B	C	D	E
A	0	0	0	0	1
B	0	0	0	1	1
C	0	0	0	1	1
D	0	1	1	0	1
E	1	1	1	1	0

Adjacency matrix A

	A	B	C	D	E
A	1	0	0	0	0
B	0	2	0	0	0
C	0	0	2	0	0
D	0	0	0	3	0
E	0	0	0	0	4

Degree matrix D

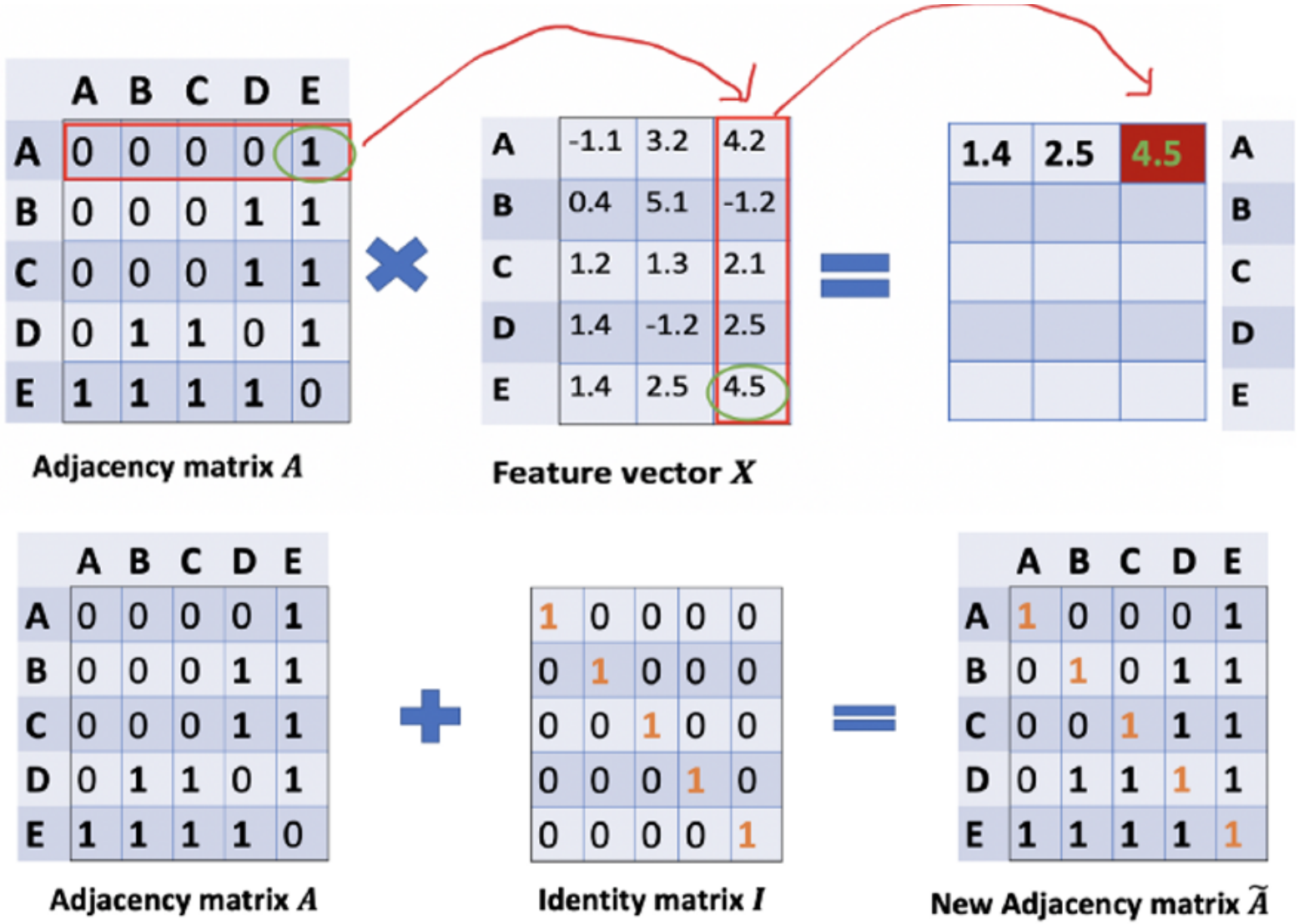
A	-1.1	3.2	4.2
B	0.4	5.1	-1.2
C	1.2	1.3	2.1
D	1.4	-1.2	2.5
E	1.4	2.5	4.5

Feature vector X

https://www.topbots.com/graph-convolutional-networks/

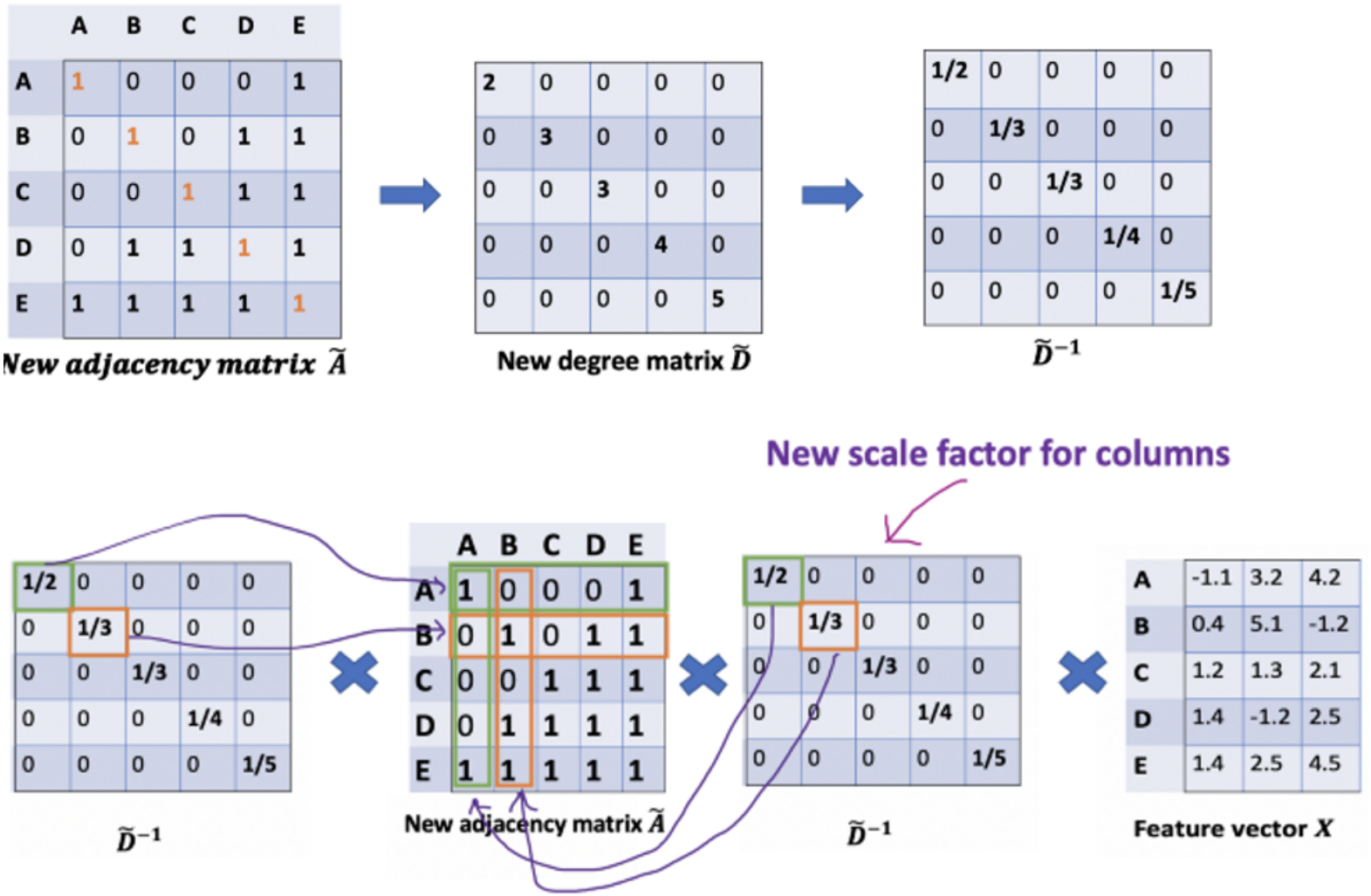
Neural Processes on Graphs

Related Work: Spectral Graph Convolutional Networks (GCN)



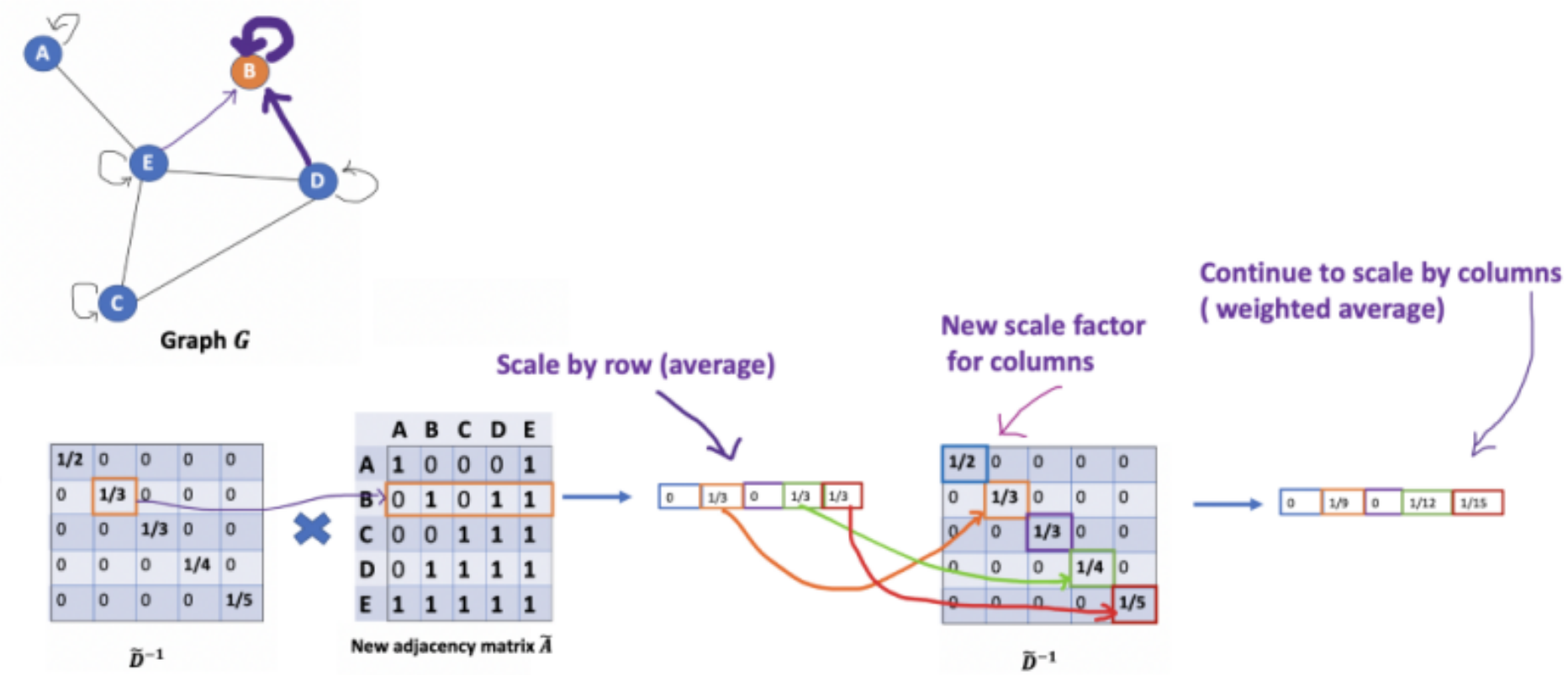
Neural Processes on Graphs

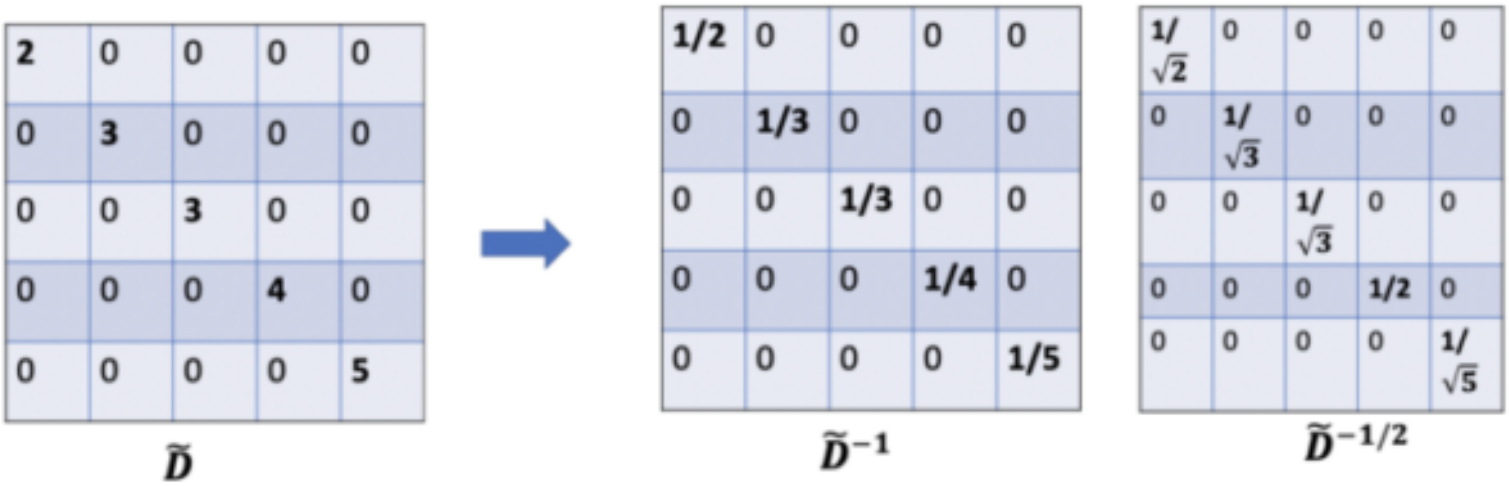
Related Work: Spectral Graph Convolutional Networks (GCN)



Neural Processes on Graphs

Related Work: Spectral Graph Convolutional Networks (GCN)



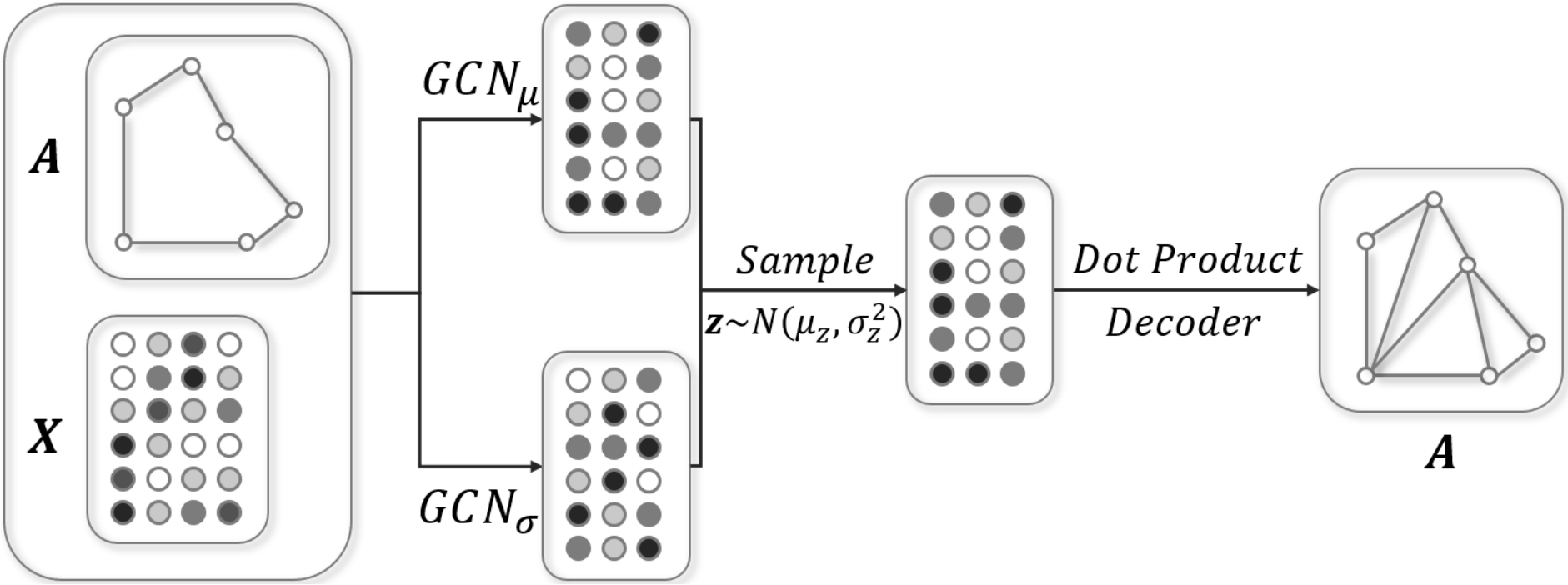


$$\boldsymbol{Z} = \mathrm{ReLU}(\bar{\mathrm{A}}\,\mathrm{ReLU}(\bar{\mathrm{A}}\mathrm{X}\mathrm{W}_1)\mathrm{W}_2) \tag{23}$$

- Here $\bar{\mathrm{A}} = \tilde{\mathrm{D}}^{-\frac{1}{2}}\tilde{\mathrm{A}}\tilde{\mathrm{D}}^{-\frac{1}{2}}$, where $\tilde{\mathrm{D}}_{ii} = \sum_j \tilde{\mathrm{A}}_{ij}$ is the degree matrix of $\tilde{\mathrm{A}}$, and $\tilde{\mathrm{A}} = \mathrm{A} + \mathrm{I}$.

Neural Processes on Graphs

Related Work: Variational Graph Autoencoders Model



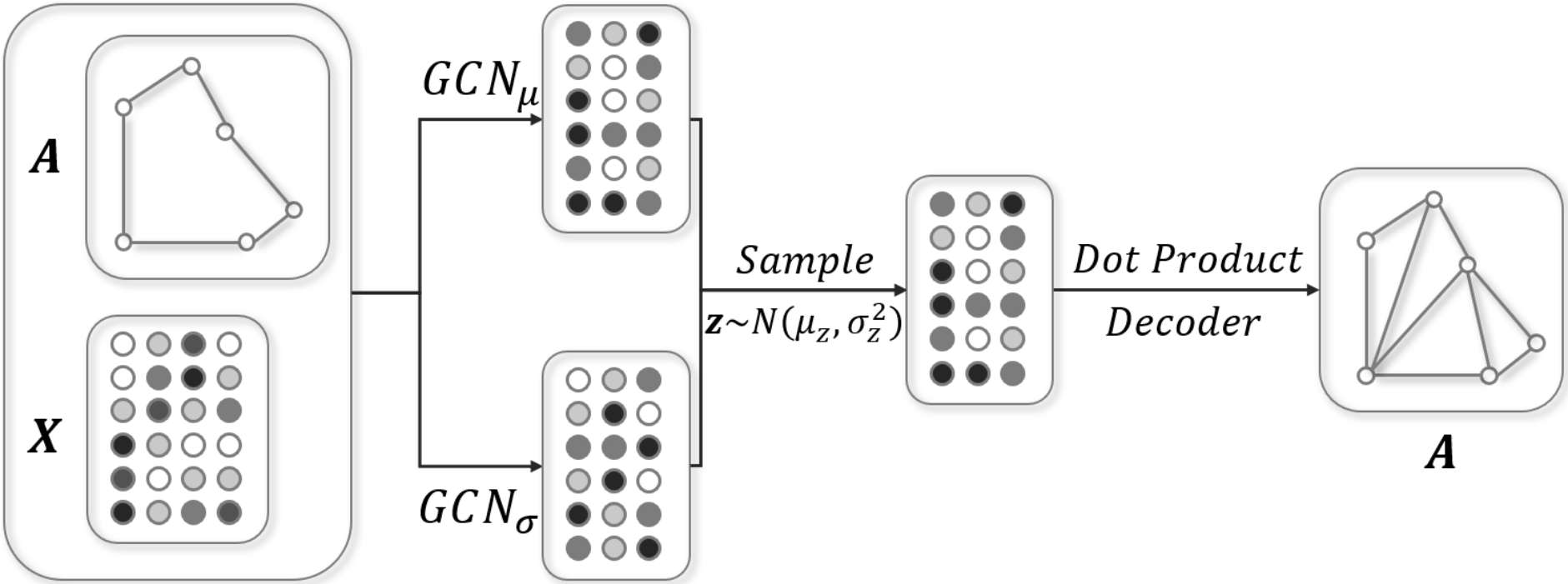
$$\boldsymbol{\mu} = \mathrm{ReLU}(\bar{\mathrm{A}}\,\mathrm{ReLU}(\bar{\mathrm{A}}\mathrm{X}\mathrm{W}_1)\mathrm{W}_{\boldsymbol{\mu}}) \tag{24}$$

$$\log \boldsymbol{\sigma} = \mathrm{ReLU}(\bar{\mathrm{A}}\,\mathrm{ReLU}(\bar{\mathrm{A}}\mathrm{X}\mathrm{W}_1)\mathrm{W}_{\boldsymbol{\sigma}}) \tag{25}$$

$$p(\mathrm{A}|\mathrm{X},\mathrm{z}) = \prod_{i=1}^{n}\prod_{j=1}^{n} p(\mathrm{A}_{ij}|\mathrm{z}_i,\mathrm{z}_j), \quad \text{with} \quad p(\mathrm{A}_{ij}=1|\mathrm{z}_i,\mathrm{z}_j) = \sigma(\mathrm{z}_i^\top \mathrm{z}_j) \tag{26}$$

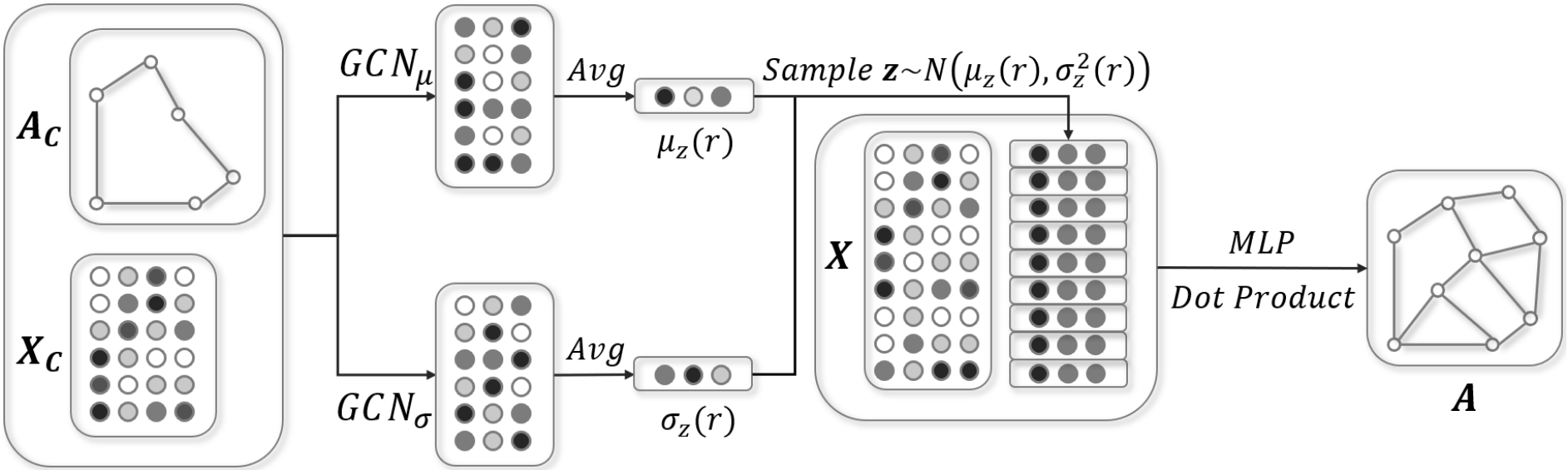
Neural Processes on Graphs

Related Work: Variational Graph Autoencoders Inference



$$\mathcal{L} = \mathbb{E}_{q(\mathrm{Z}|\mathrm{A},\mathrm{X})}[\log p(\mathrm{A}|\mathrm{Z})] - \mathbf{KL}[q(\mathrm{Z}|\mathrm{A},\mathrm{X})||p(\mathrm{Z})] \tag{27}$$

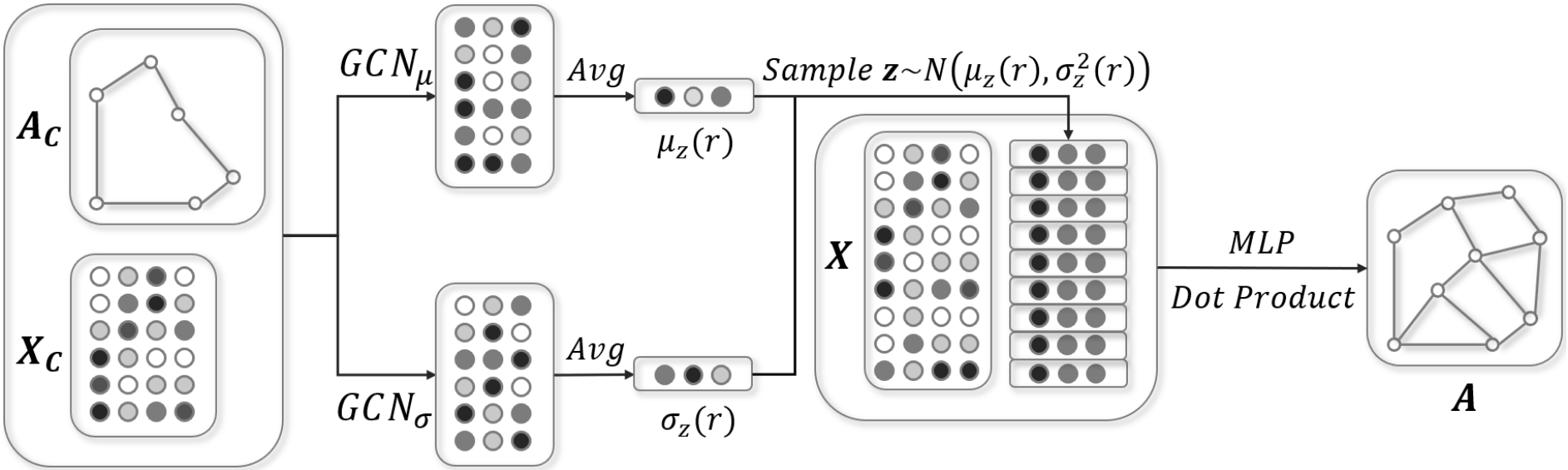
- First we define an undirected graph $G = (V, A)$, where V is the set of nodes with features $x_i \in X$ corresponding to each node $v_i \in V$, and the adjacency matrix A with $A_{ij} = 1$ if there is a link between v_i and v_j, and $A_{ij} = 0$ otherwise.
- Then we randomly select a subset of nodes $V_C \in V$ with its related features $X_C \in X$ and adjacency matrix A_C to construct a context subgraph G_C. We assume there are n nodes in the complete graph G, and the first m nodes are the context nodes V_C.
- Our goal is to model the adjacency matrix A for the complete graph G conditional on the context subgraph G_C.



$$\boldsymbol{\mu}_{\mathrm{c}} = \mathrm{ReLU}(\bar{\mathrm{A}}_{\mathrm{C}}\,\mathrm{ReLU}(\bar{\mathrm{A}}_{\mathrm{C}}\mathrm{X}_{\mathrm{C}}\mathrm{W}_1)\mathrm{W}_{\boldsymbol{\mu}}) \tag{28}$$

$$\log \boldsymbol{\sigma}_{\mathrm{c}} = \mathrm{ReLU}(\bar{\mathrm{A}}_{\mathrm{C}}\,\mathrm{ReLU}(\bar{\mathrm{A}}_{\mathrm{C}}\mathrm{X}_{\mathrm{C}}\mathrm{W}_1)\mathrm{W}_{\boldsymbol{\sigma}}) \tag{29}$$

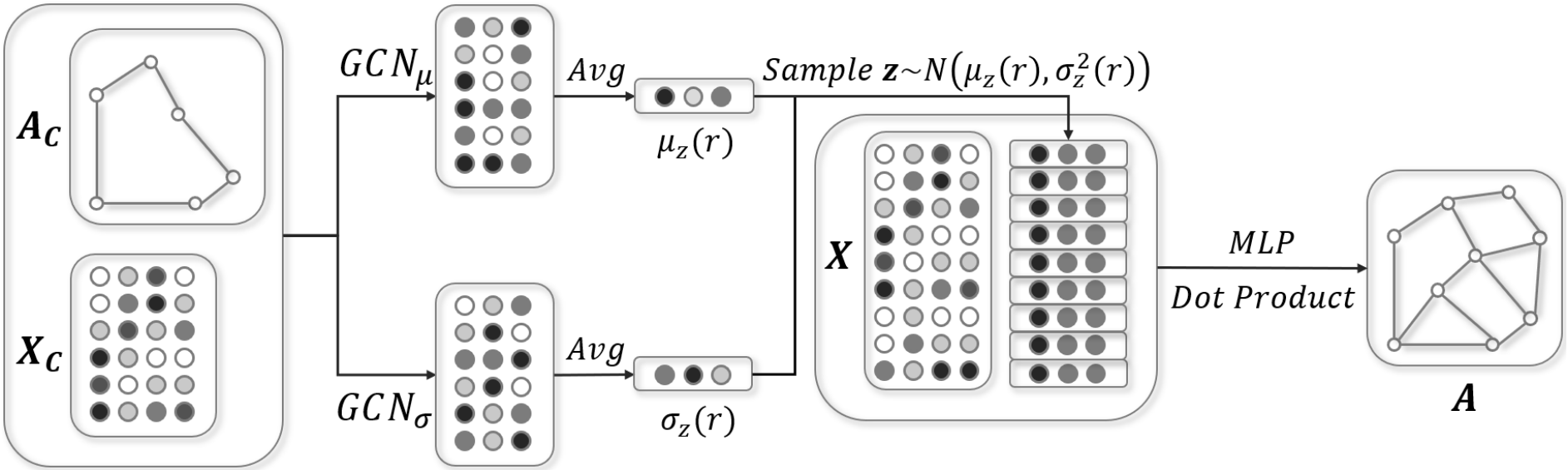
- Here $\bar{\mathrm{A}}_{\mathrm{C}} = \tilde{\mathrm{D}}_{\mathrm{C}}^{-\frac{1}{2}}\tilde{\mathrm{A}}_{\mathrm{C}}\tilde{\mathrm{D}}_{\mathrm{C}}^{-\frac{1}{2}}$, where $\tilde{\mathrm{D}}_{\mathrm{C}_{ii}} = \sum_j \tilde{\mathrm{A}}_{\mathrm{C}_{ij}}$ is the degree matrix of $\tilde{\mathrm{A}}_{\mathrm{C}}$, and $\tilde{\mathrm{A}}_{\mathrm{C}} = \mathrm{A}_{\mathrm{C}} + \mathrm{I}$.



$$\boldsymbol{\mu}_{\mathrm{z}}(\mathrm{r}) = \frac{1}{m}\sum_{i=1}^{m}\boldsymbol{\mu}_{\mathrm{c}_i} \qquad \log\boldsymbol{\sigma}_{\mathrm{z}}(\mathrm{r}) = \frac{1}{m}\sum_{i=1}^{m}\log\boldsymbol{\sigma}_{\mathrm{c}_i} \tag{30}$$

- Use $\boldsymbol{\mu}_{\mathrm{z}}(\mathrm{r})$ and $\boldsymbol{\sigma}_{\mathrm{z}}^2(\mathrm{r})$ to parameterise the distribution for the global latent representation.

$$q(\mathrm{z}|\mathrm{A_C}, \mathrm{X_C}) = \mathcal{N}(\boldsymbol{\mu}_{\mathrm{z}}(\mathrm{r}), \boldsymbol{\sigma}_{\mathrm{z}}^2(\mathrm{r}))$$



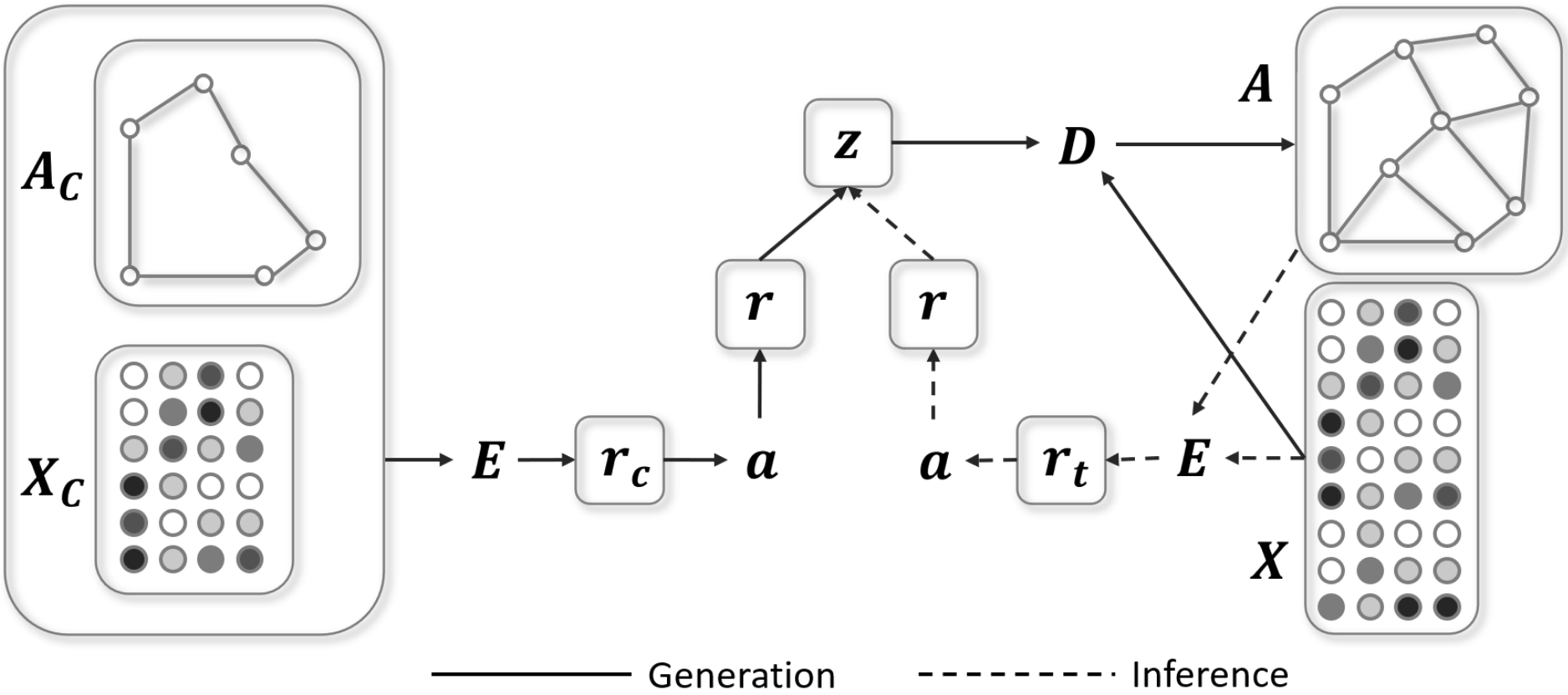
- Combine a sampled z with every feature $x_i \in X$ as $\tilde{x}_i \in \tilde{X}$, where $\tilde{x}_i = [x_i^\top \; z^\top]^\top$.

$$U = \sigma(W_3\sigma(W_2\tilde{X} + b_1) + b_2) \tag{31}$$

$$p(A|X,z) = \prod_{i=1}^{n}\prod_{j=1}^{n} p(A_{ij}|u_i,u_j), \quad \text{with} \quad p(A_{ij}=1|u_i,u_j) = \sigma(u_i^\top u_j) \tag{32}$$

Neural Processes on Graphs

Inference and Learning



$$\mathcal{L} = \mathbb{E}_{q(\mathrm{z}|\mathrm{A},\mathrm{X})}[\log p(\mathrm{A}|\mathrm{X},\mathrm{z})] - \mathbf{KL}[q(\mathrm{z}|\mathrm{A},\mathrm{X})||q(\mathrm{z}|\mathrm{A_C},\mathrm{X_C})] \tag{33}$$

The End