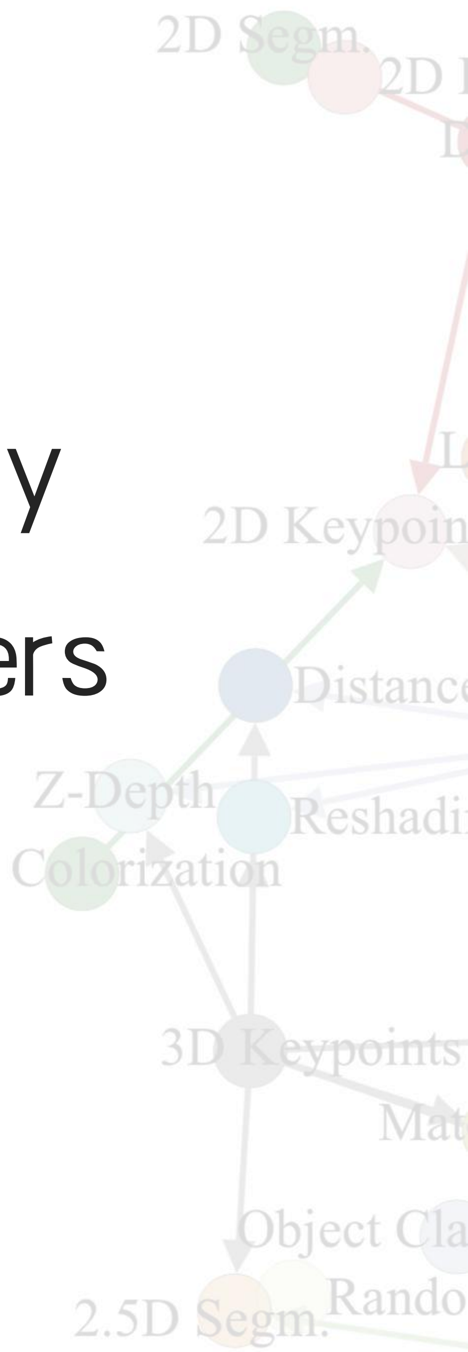
Efficient Transferability Recovery with Directed Graph Autoencoders

Xiangyu Chen

10.11



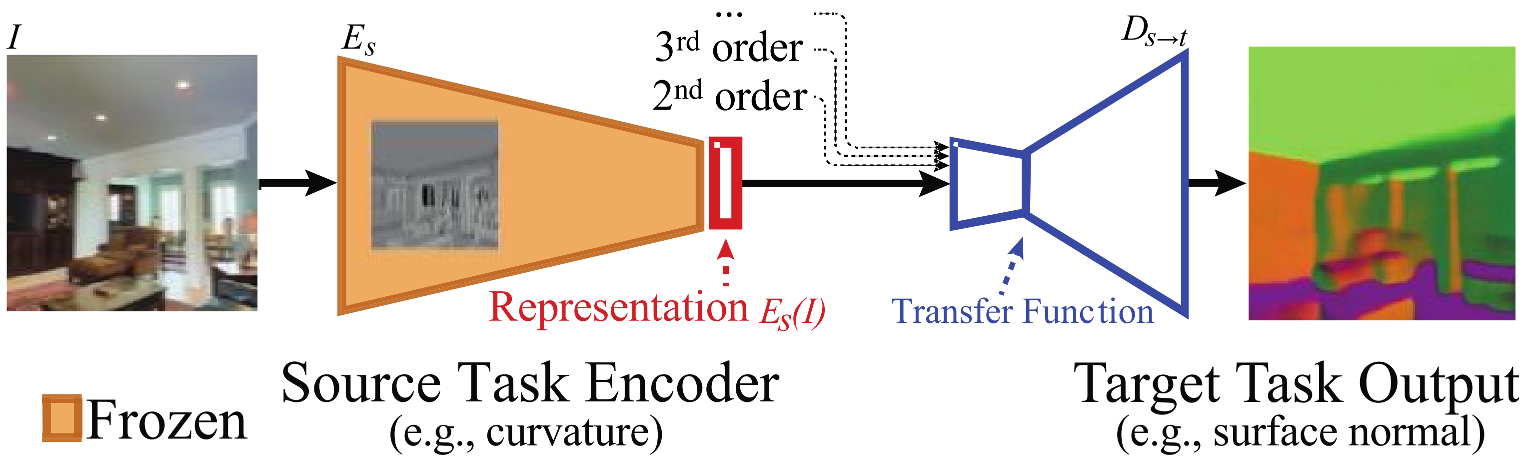
Introduction： transferability

Transfer Learning:
Model developed for task S may be useful for solving task T, if S and T related

Given

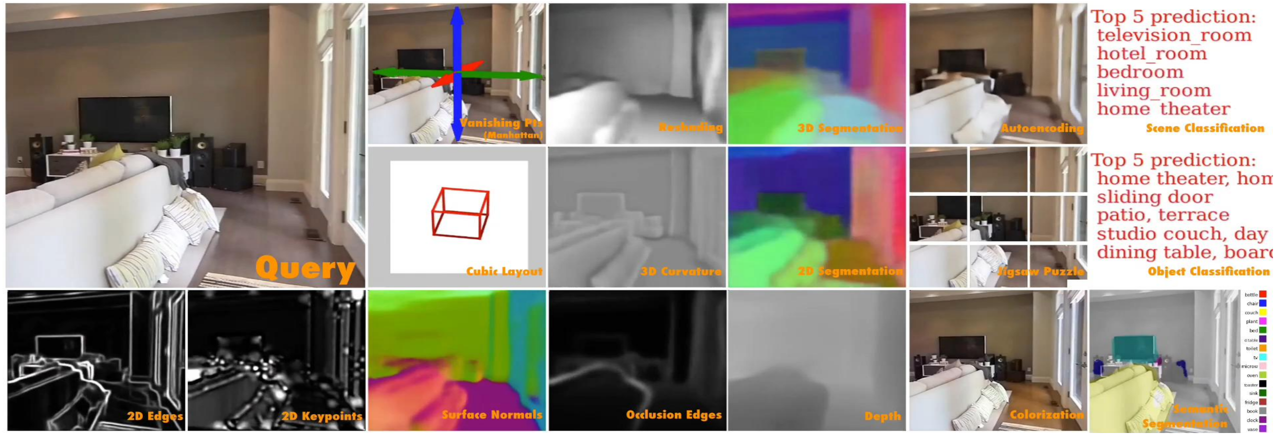
Source task s

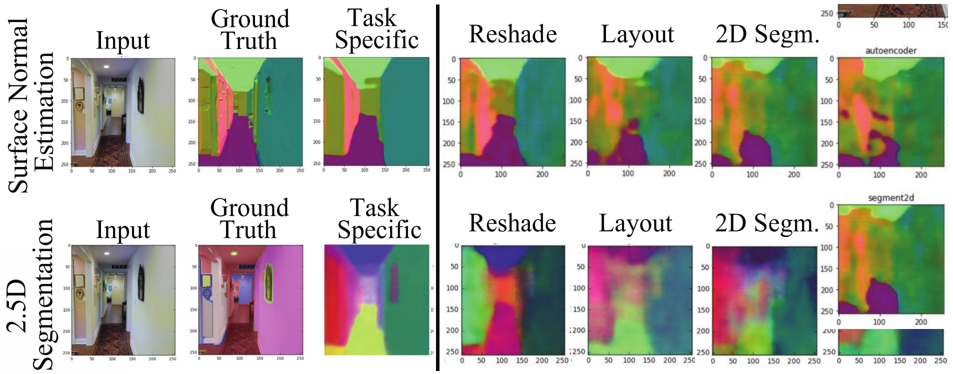
Target task t

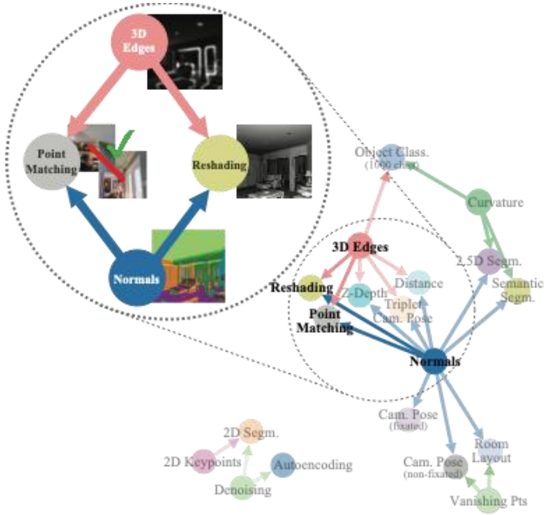


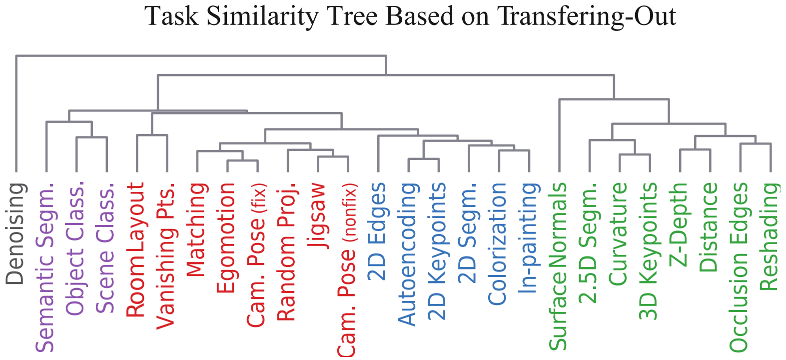
$$D_{s\to t} := \arg\min_{\theta} \mathbb{E}_{I\in\mathcal{D}}[L_t(D_\theta(E_s(I)), f_t(I))]$$

Taskonomy ≈ Task + Taxonomy（分类论）
Disentangling Task Transfer Learning









Task relationships exist

Can be computationally measured

Tasks belonging to a structured space

Data efficiency：Transfers training data 8x−120x less than task−specific

[Zamir et al., CVPR 2018]

Problem: Expensive Task Pairwise Computations (O(n^2))

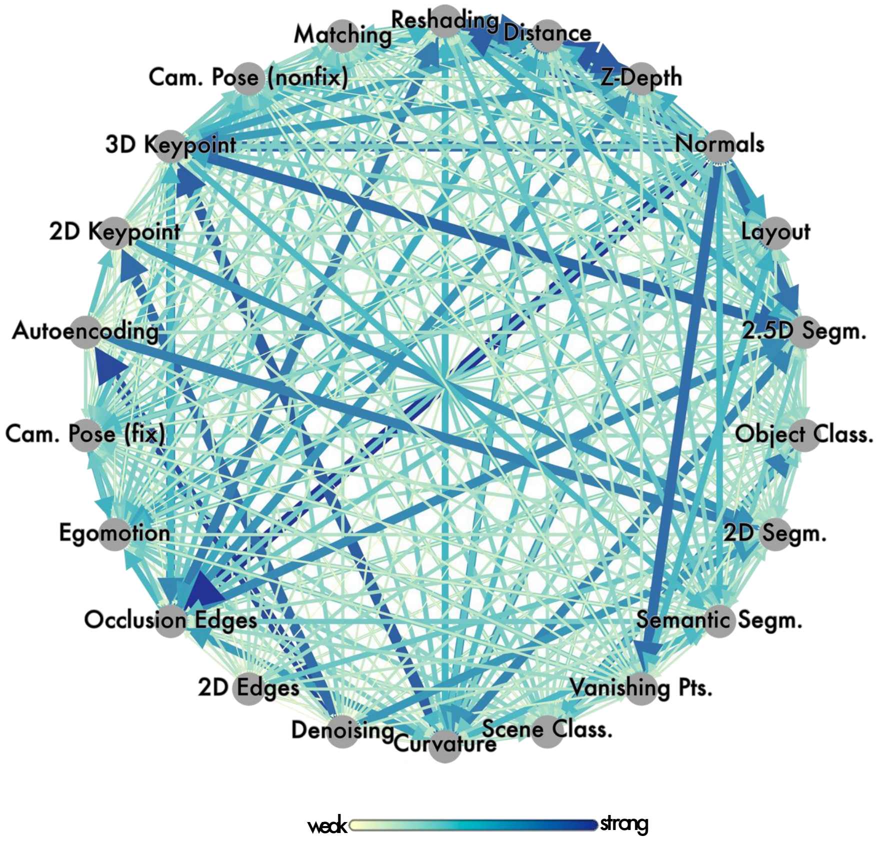
26 Task–Specific Networks

3000 Transfer Networks (include high–order relations)

47,829 GPU hours

Question: How can we efficiently estimate task transferability?

$$\widehat{\mathrm{Tr}}\, f(S \to T) \triangleq \tau(\mathrm{D_S}, D_T, \theta)$$



Problem Formulation

Input: set of tasks $T=\{(X_1,Y_1),(X_2,Y_2),...,(X_N,Y_N)\}$

Goal: Find an “embedding” or “representation” $v_i = f(X_i,Y_i)$ for each task. Utilize the distance between any two embeddings $dist(v_i,v_j)$ approximates the transferability

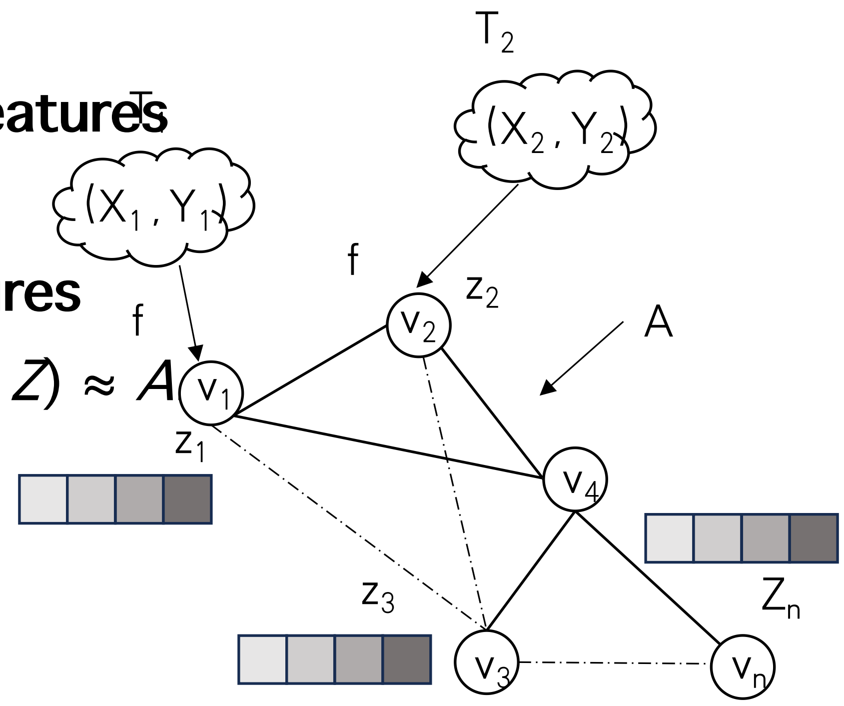
$dist(v_i,v_j) \approx Tr(\{X_i,Y_i\} \rightarrow \{X_j,Y_j\})$

Problem 1: Transferability Recovery without Node Features

Given partial A’ recover A (learn $g(A') \approx A$)

Problem 2: Transferability Recovery with Node Features

Given A' and the node feature Z reconstruct A. $g(A', Z) \approx A$



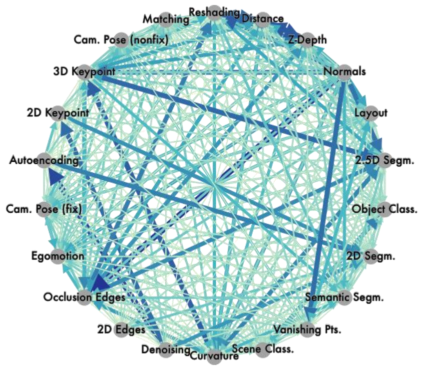
A: True transferability matrix between tasks.

A': Partial or observed transferability matrix between tasks.

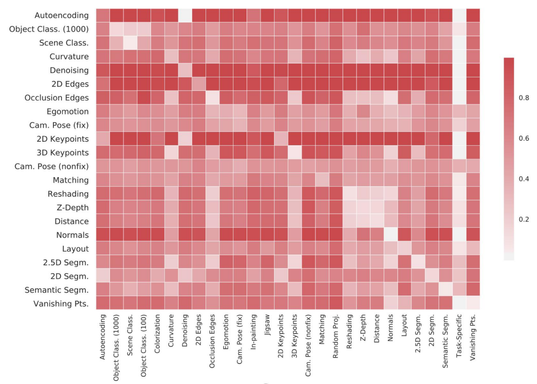
Z: Task specific meta-information / task embeddings.

Framework

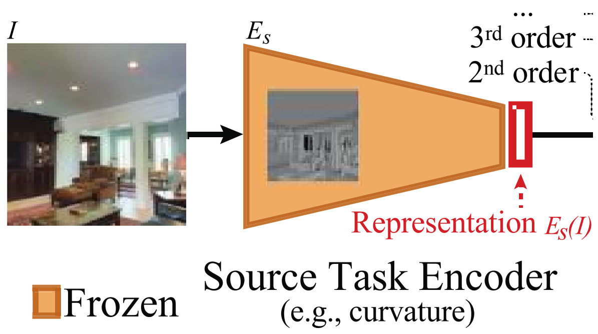
- Given the task graph



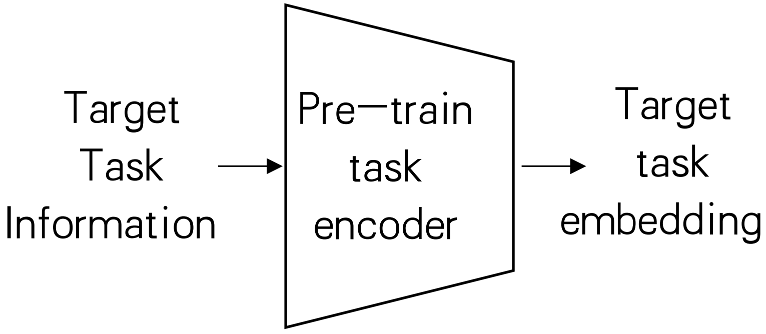
Adjacency Matrix



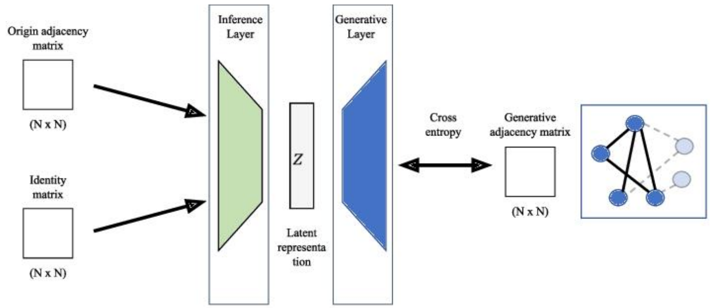
- Task Embedding



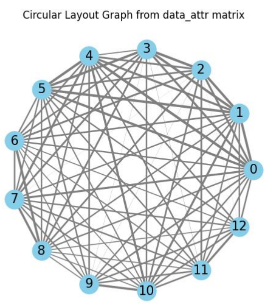
$$D_{s\to t} := \arg\min_{\theta} \mathbb{E}_{I\in\mathcal{D}}[L_t(D_\theta(E_s(I)), f_t(I))]$$

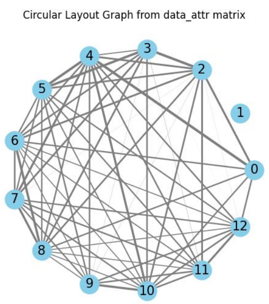


- Directed graph structure learning

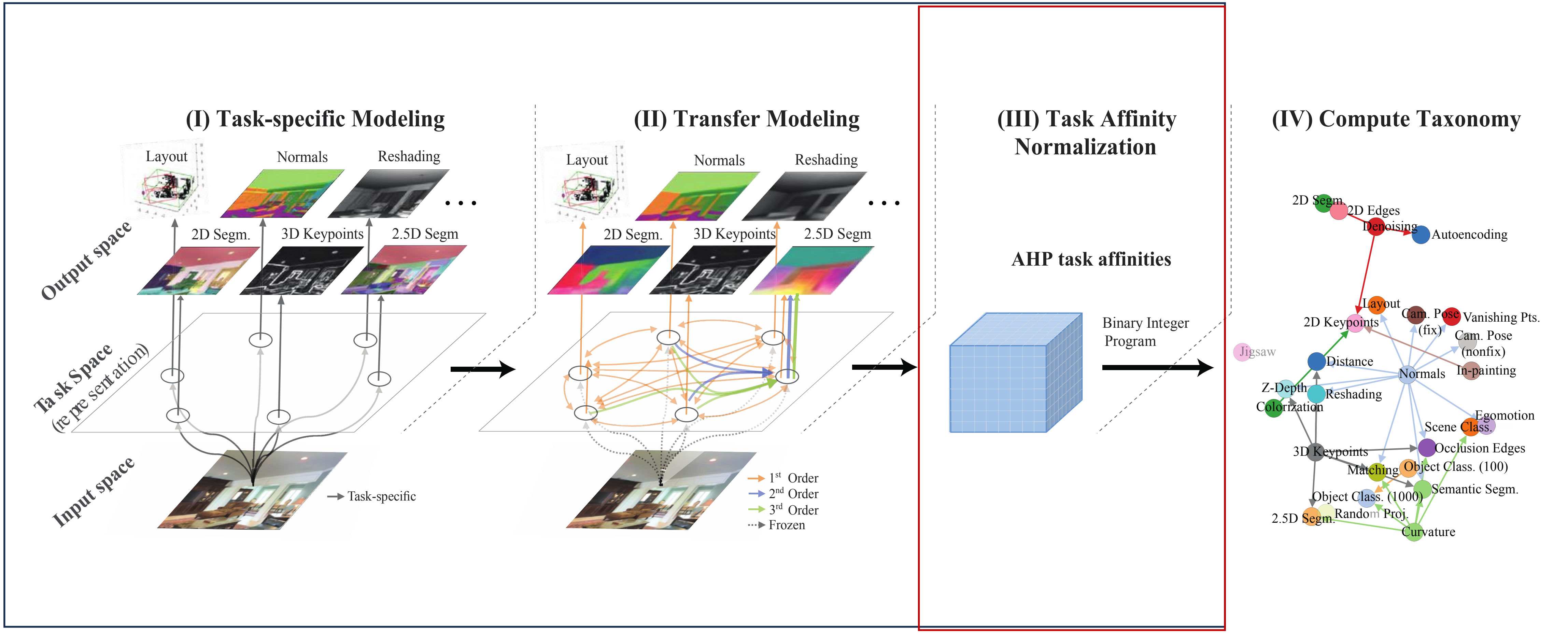


Matrix completion w/ and w/o task embedding
In transductive and inductive setting



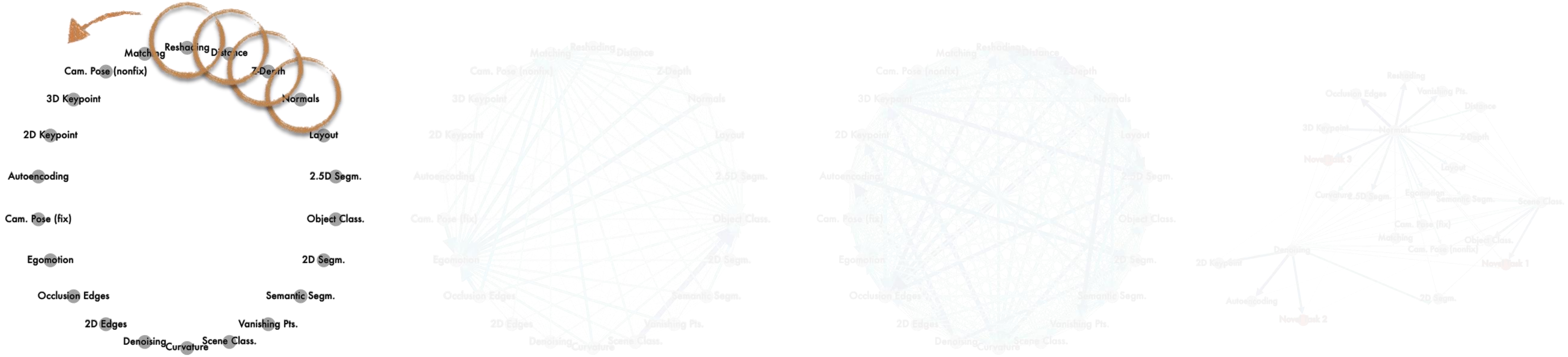


Step1、 How to obtain affinity matrix ?



Zamir et al. Taskonomy 2018

I： Task—Specific Modeling

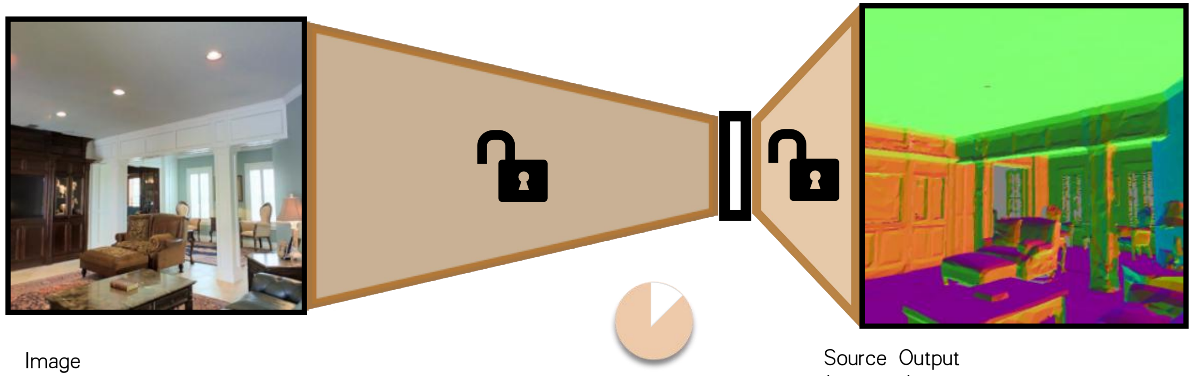


I： Task—Specific Modeling

II： Transfer Modeling

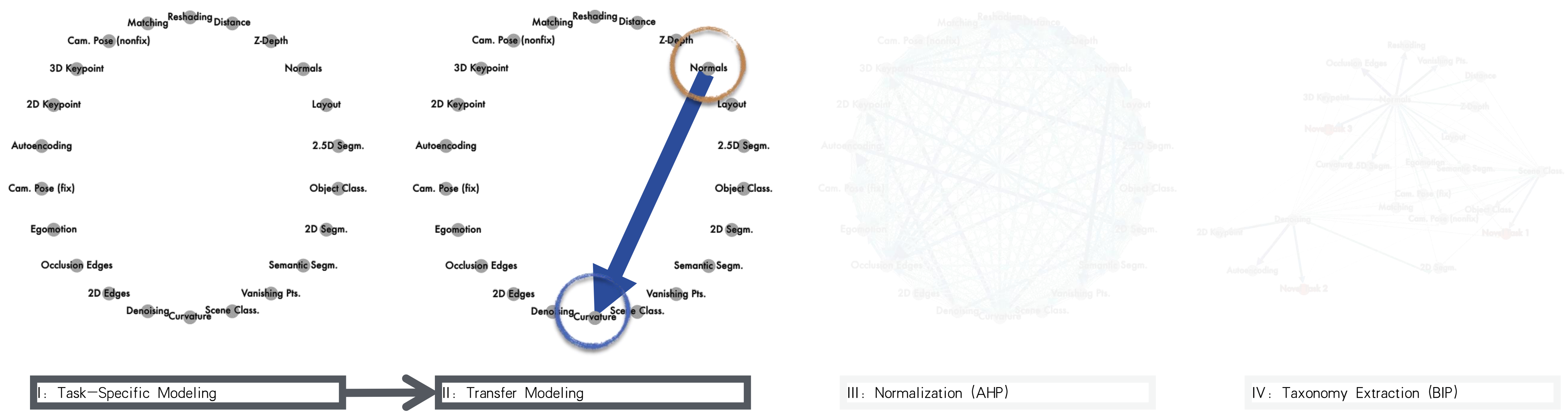
III： Normalization (AHP)

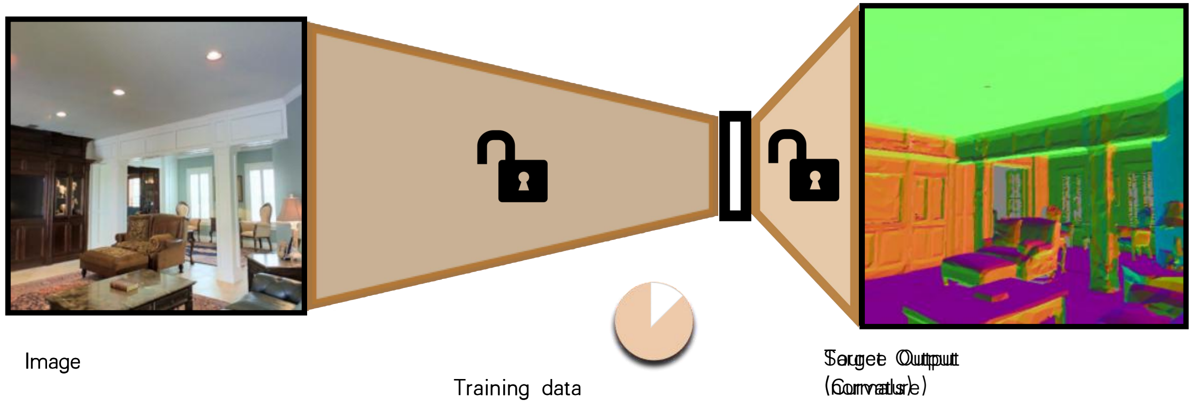
IV： Taxonomy Extraction (BIP)



Zamir et al. Taskonomy 2018

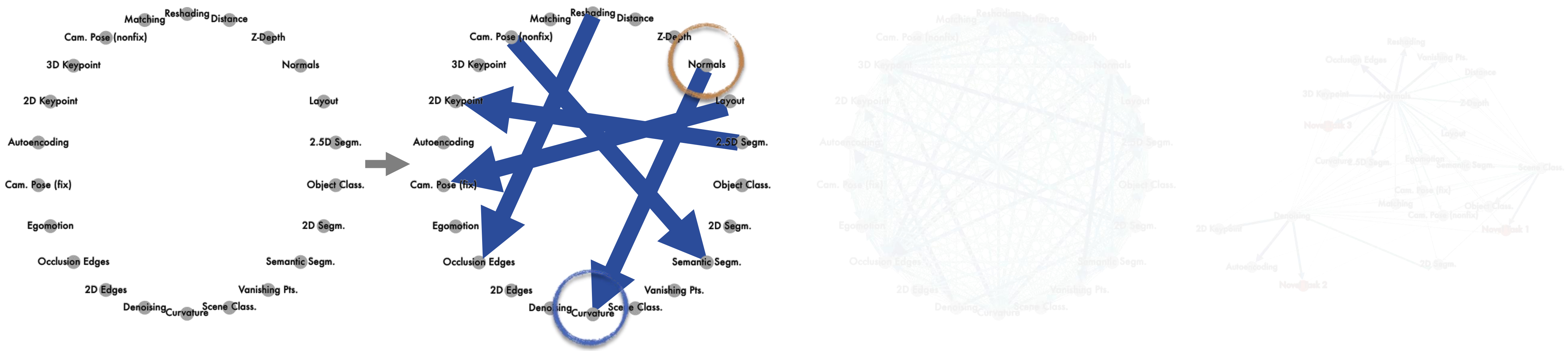
II: Transfer Modeling





Zamir et al. Taskonomy 2018

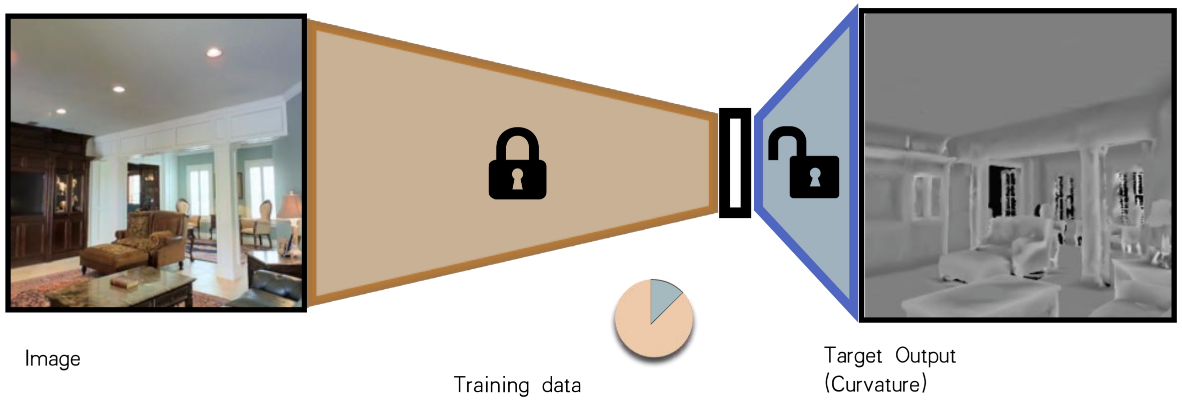
II: Transfer Modeling



I: Task−Specific Modeling → II: Transfer Modeling

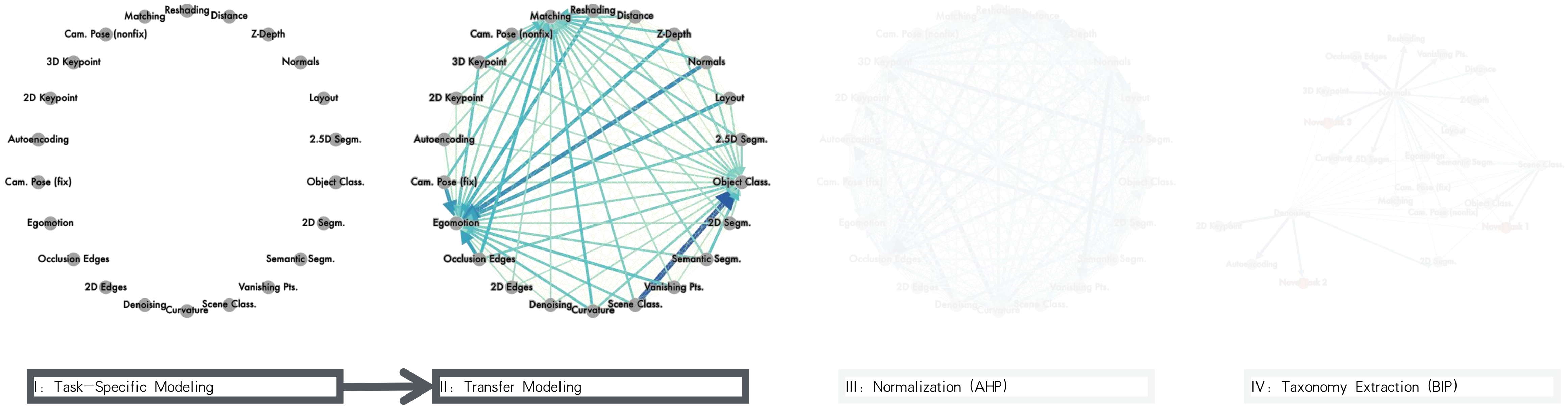
III: Normalization (AHP)

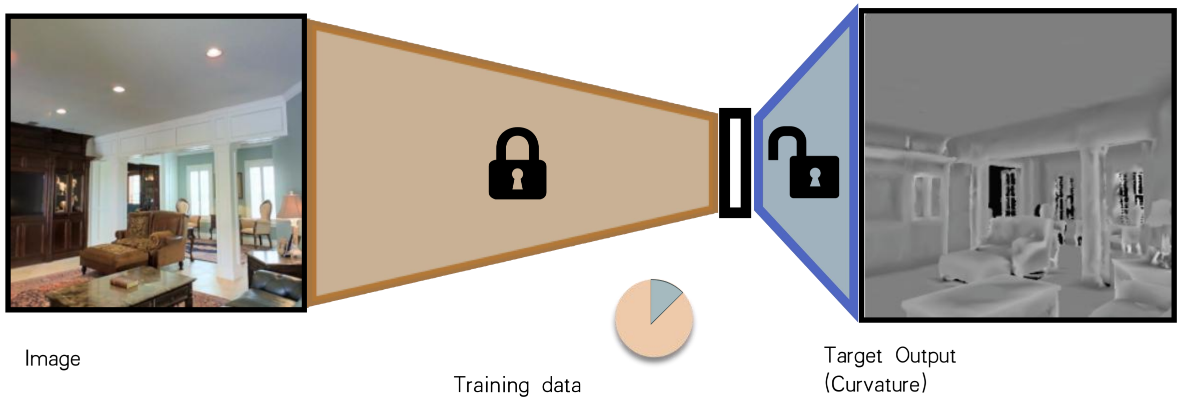
IV: Taxonomy Extraction (BIP)



Zamir et al. Taskonomy 2018

II: Transfer Modeling

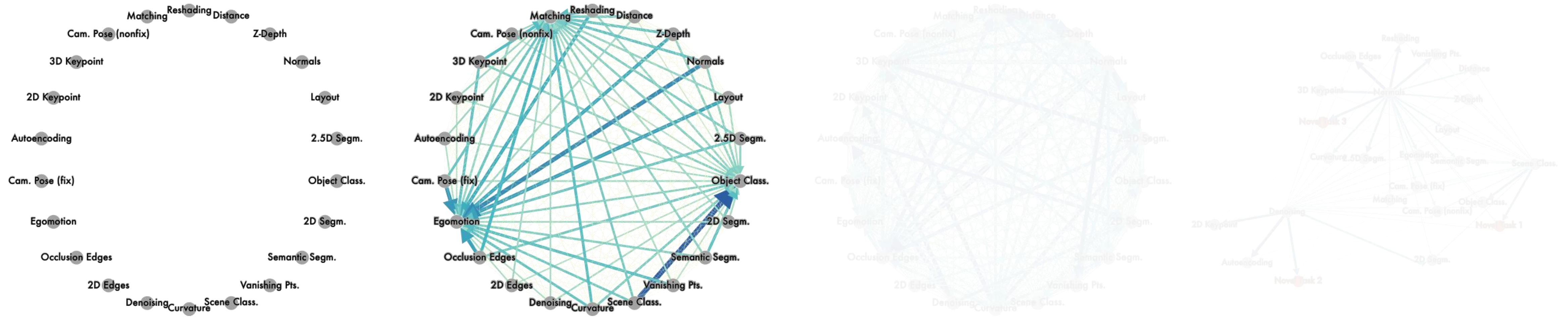




$$D_{s\to t} := \arg\min_{\theta} \mathbb{E}_{I\in\mathcal{D}}\left[L_t\Big(D_\theta\big(E_s(I)\big), f_t(I)\Big)\right]$$

Zamir et al. Taskonomy 2018

III: Normalization



I: Task–Specific Modeling

II: Transfer Modeling

III: Normalization (AHP)

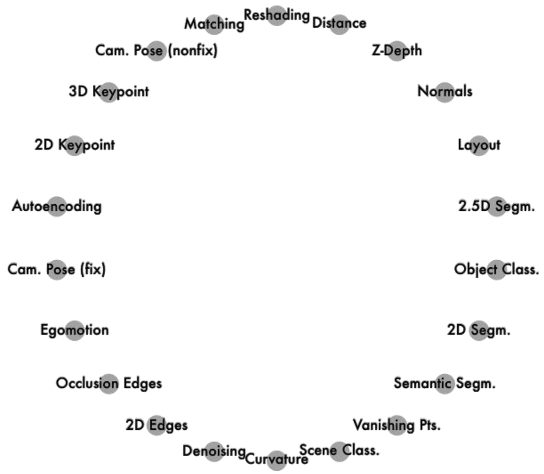
IV: Taxonomy Extraction (BIP)

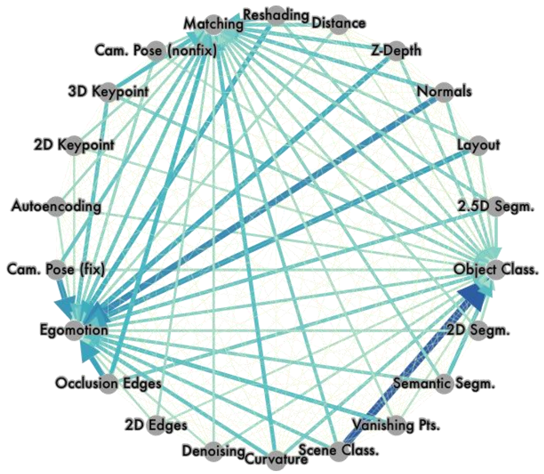
Adjacency Matrix (pre–normalization)

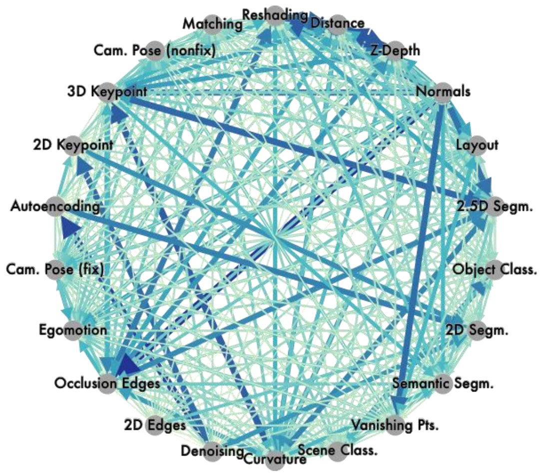
Zamir et al. Taskonomy 2018

weak  strong

III：Normalization







Why we need normalization?

— Different output spaces

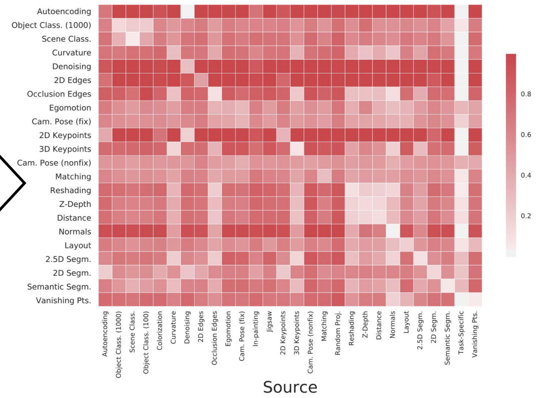
(l2_loss = 0.01) ≠ (CE_loss = 0

I：Task–Specific Modeling → II：Transfer Modeling → III：Normalization (AHP)

Adjacency Matrix (pre–normalization)

?

Adjacency Matrix (post–normalization)



Ordinal Normalization
(基于序数的规范化)
–Analytic Hierarchical Process.
(AHP)

Zamir et al. Taskonomy 2018

Ordinal Normalization — Analytic Hierarchical Process (AHP)

- Quick example: W_t is the pairwise tournament matrix (锦标赛矩阵)

Construct Wt

	1	2	3	4	5
1	500/1000	700/1000	800/1000	900/1000	1000/1000
2	300/1000	500/1000	700/1000	800/1000	900/1000
3	200/1000	300/1000	500/1000	600/1000	800/1000
4	100/1000	200/1000	400/1000	500/1000	700/1000
5	0/1000	100/1000	200/1000	300/1000	500/1000

(1)

Laplace smoothing

	1	2	3	4	5
1	501/1025	701/1025	801/1025	901/1025	1001/1025
2	301/1025	501/1025	701/1025	801/1025	901/1025
3	201/1025	301/1025	501/1025	601/1025	801/1025
4	101/1025	201/1025	401/1025	501/1025	701/1025
5	1/1025	101/1025	201/1025	301/1025	501/1025

(2)

$W_t' = W_t / W_t^T$

	1	2	3	4	5
1	1	2.329	3.985	8.921	1001
2	0.429	1	2.329	3.985	8.921
3	0.251	0.429	1	1.499	3.985
4	0.112	0.251	0.667	1	2.329
5	0.001	0.112	0.251	0.429	1

(3)

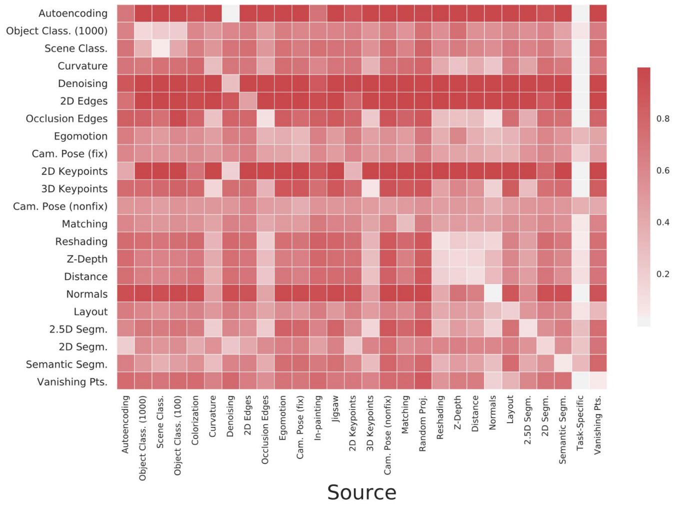
Principal eigenvector

Source1	2	3	4	5
0.9923	0.1081	0.0545	0.0268	0.0058

(4)

$$w'_{i,j} = \frac{\mathbb{E}_{I \in \mathcal{D}_{\text{test}}}\left[D_{s_i \to t}(I) > D_{s_j \to t}(I)\right]}{\mathbb{E}_{I \in \mathcal{D}_{\text{test}}}\left[D_{s_i \to t}(I) < D_{s_j \to t}(I)\right]}$$

Adjacency Matrix (post-normalization)



Principal eigenvector

Source1	2	3	4	5
0.9923	0.1081	0.0545	0.0268	0.0058

⇒

Normalized to sum=1

Source1	2	3	4	5
0.8356	0.0910	0.0459	0.0226	0.0049

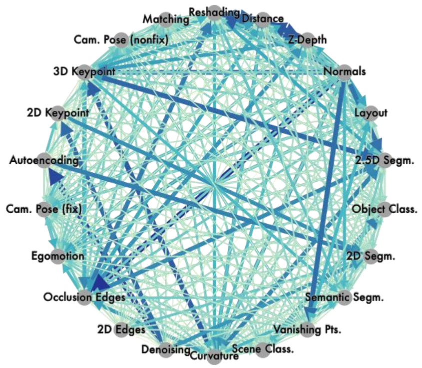
⇒

$dist = e^{-\beta \cdot P}$

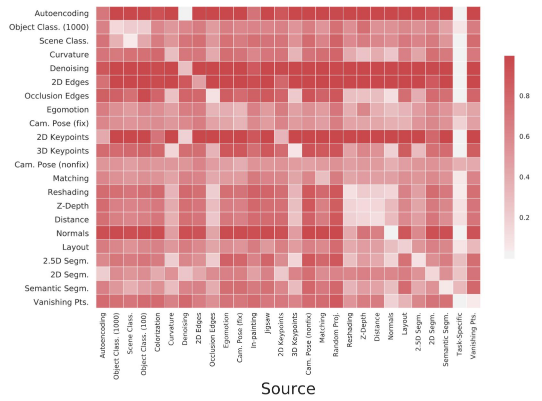
Source1	2	3	4	5
2.4e-9	0.115	0.336	0.585	0.8890

Step2、how to get the task embedding from meta-info

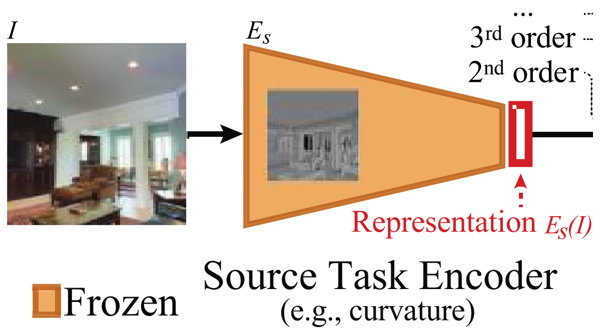
- Given the task graph



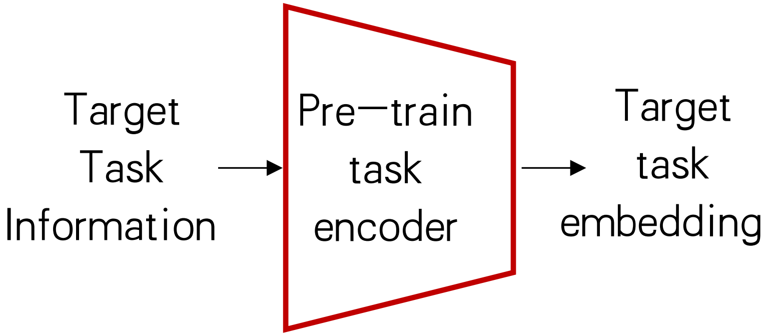
Adjacency Matrix



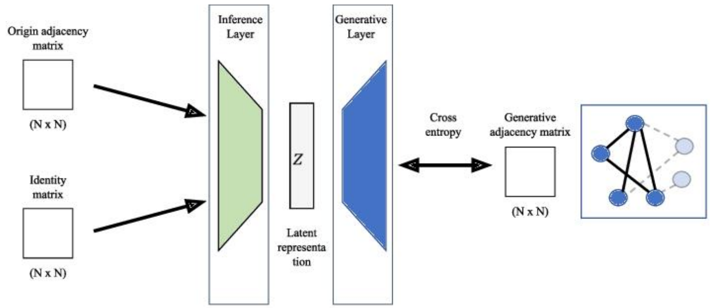
- Task-embedding



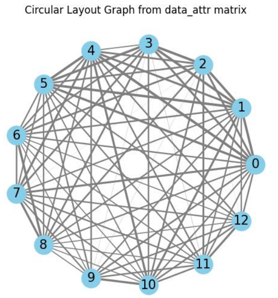
$$D_{s\to t} := \arg\min_{\theta} \mathbb{E}_{I\in\mathcal{D}}[L_t(D_\theta(E_s(I)), f_t(I))]$$

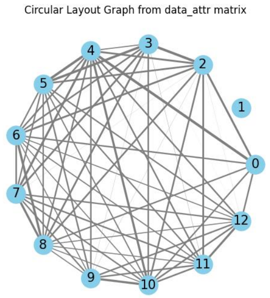


- Directed graph learning



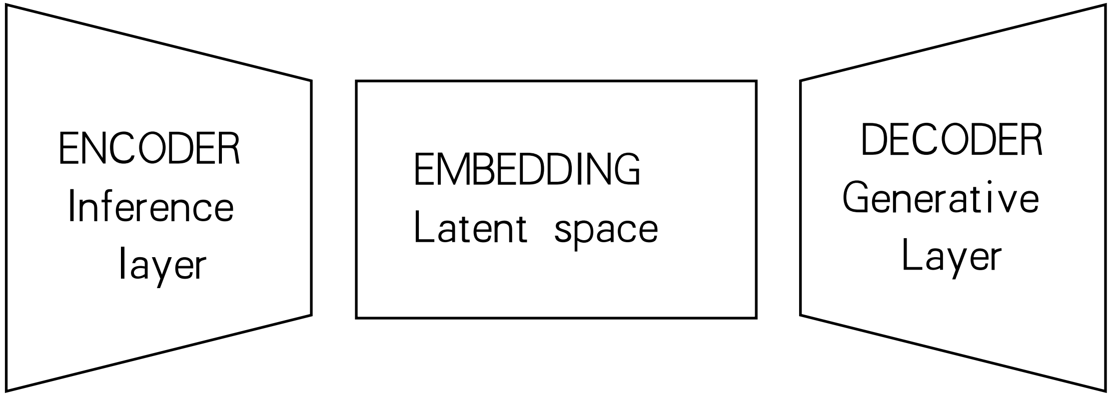
Matrix completion w/ and w/o task embedding
In transductive and inductive setting

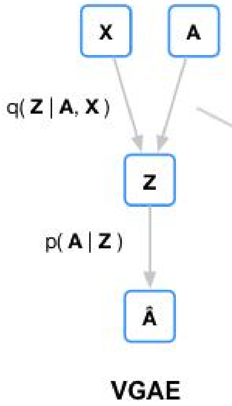


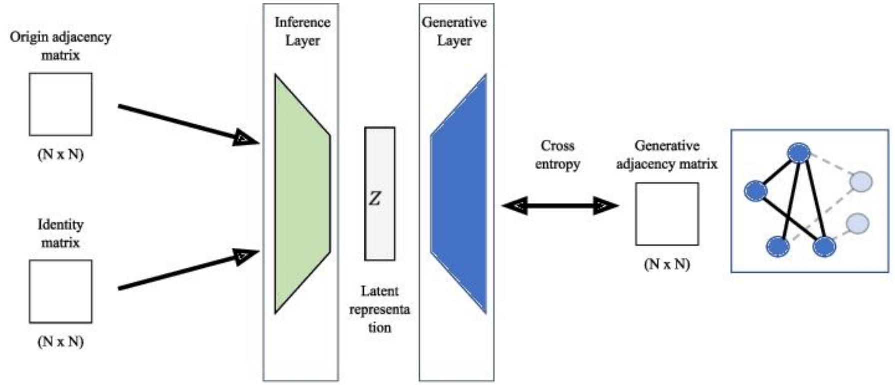


Step3、Graph structure learning

Preliminary： Graph Autoencoder (GAE) & Variational Graph Autoencoder (VGAE)







$$Z = \bar{X}$$

$$\bar{X} = GCN(A, X) = \mathrm{ReLU}\left(\tilde{A}XW_0\right)$$

$$\text{with } \tilde{A} = D^{-1/2}AD^{-1/2}$$

$$\hat{\mathbf{A}} = \sigma\left(\mathbf{Z}\mathbf{Z}^{\top}\right), \text{ with } \quad \mathbf{Z} = \mathrm{GCN}(\mathbf{X}, \mathbf{A})$$

[Kipf & Welling, 2016]

$$GCN(A, X) = \tilde{A}\,\mathrm{Re}\,LU\left(\tilde{A}XW_0\right)W_1$$

$$\text{with } \tilde{A} = D^{-1/2}AD^{-1/2}$$

$$Z = \mu + \sigma \odot \epsilon$$

$$\epsilon \sim \mathrm{Norm}(0, 1)$$

$$\mu = GCN_{\mu}(X, A) = \tilde{A}\bar{X}W_1$$

$$\log \sigma^2 = GCN_{\sigma}(X, A) = \tilde{A}\bar{X}W_1$$

$$\hat{A} = \sigma(zz^T)$$

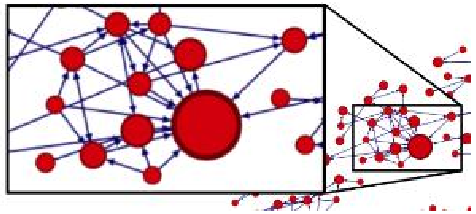
Encoder $q(z_i|X, A) = N(z_i|\mu_i, diag(\sigma_i^2))$

Decoder $p(A_{ij} = 1|z_i, z_j) = \sigma(z_i^T z_j)$

$$L = E_{q(Z|X,A)}[logp(A|Z)] - KL[q(Z|X, A)||p(Z)]$$

Gravity—Inspired GAE

$$a_{1\to 2} = \frac{Gm_2}{r^2}, \quad a_{2\to 1} = \frac{Gm_1}{r^2}$$

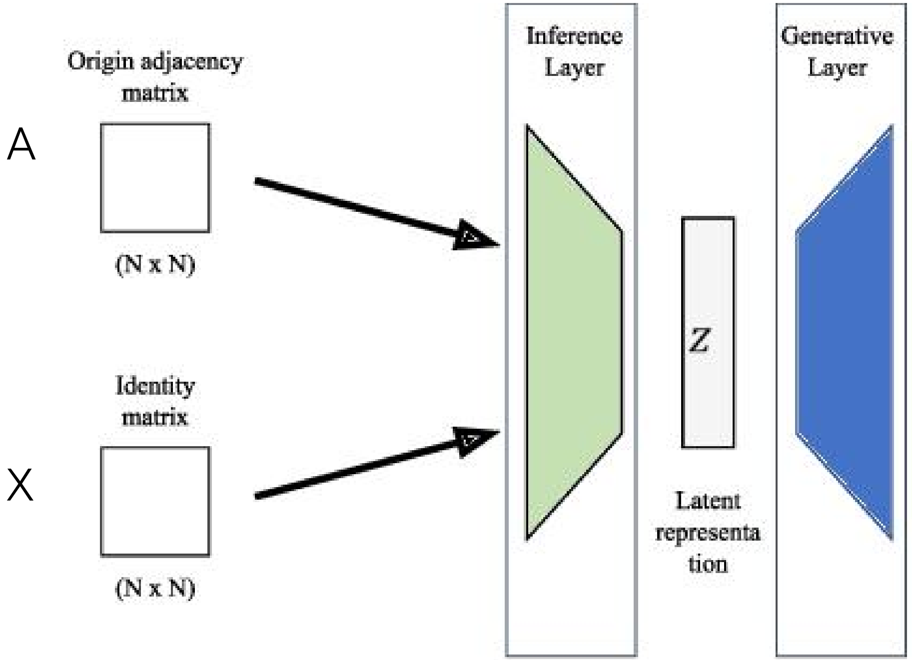


$$P(A_{ij} \mid z_i, z_j) = \sigma\left(\log \frac{\tilde{m}_j}{\|z_i - z_j\|_2^2}\right) \text{ where } \tilde{m}_j = Gm_j$$

$$q((Z, \tilde{M})|A, X) = \prod_{i=1}^{n} q((z_i, \tilde{m}_i)|A, X),$$

$$p(A|Z, \tilde{M}) = \prod_{i=1}^{n}\prod_{j=1}^{n} p(A_{ij}|z_i, z_j, \tilde{m}_j)$$

$$p((Z, \tilde{M})) = \prod_i p(z_i, m_i) = \prod_i \mathcal{N}((z_i, m_i)|0, I)$$



$$\mu = f_\mu(X, A), \log(\sigma^2) = f_\sigma(X, A) \quad z_i \sim N(\mu_i, \sigma_i^2 I)$$

$$\mu = [\mu_1, \ldots, \mu_N, m_1, \ldots, m_N] \ \sigma = [\sigma_1, \ldots, \sigma_N]$$

$$L = \mathrm{E}_{Q(Z|X,A)}[\log P(A \mid Z)] - KL(Q(Z \mid X, A)\|P(Z))$$

[Gravity—Inspired Graph Autoencoders for Directed Link Prediction]

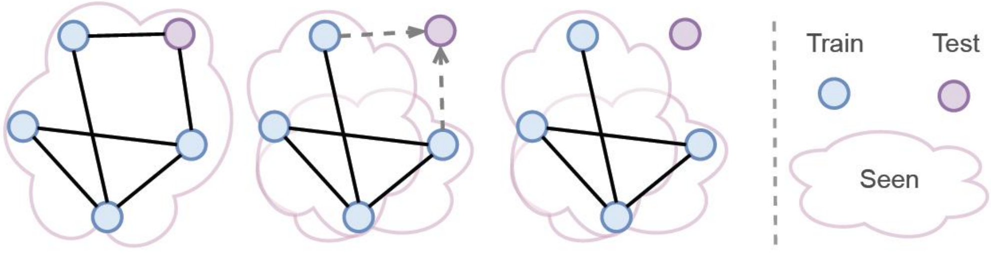
Experiment Setting：Transductive and Inductive

- Transductive Learning
  - In transductive learning, the testing set is known during training, but the labels of the testing set are unknown. The aim is to assign labels to the testing set without learning a generalized model that can be applied to unseen tasks – recover the missing point

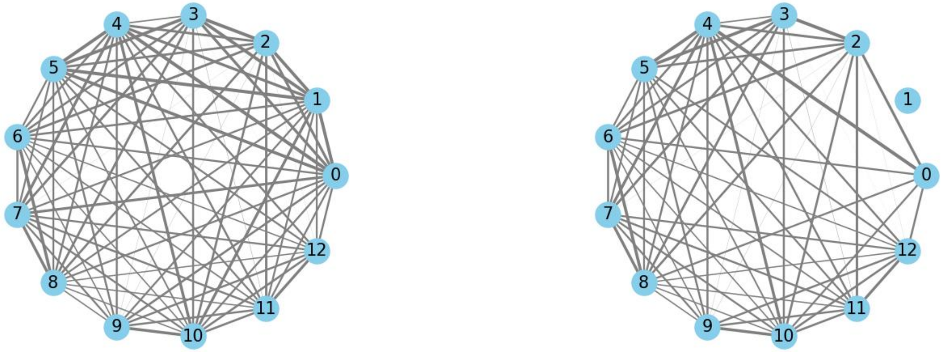
$$L_{\text{recover}} = \sum_{i \in \text{ masked nodes}} \|y_i - \hat{y}_i\|^2$$

- Inductive Learning
  - In inductive learning, the model is trained without knowledge of the testing set. The goal here is to learn a generalized model that can be applied to unseen, new tasks – recover the hole graph structure

$$L_{\text{reconstruct}} = \sum_{j \in \text{ all nodes}} \|z_j - \hat{z}_j\|^2$$



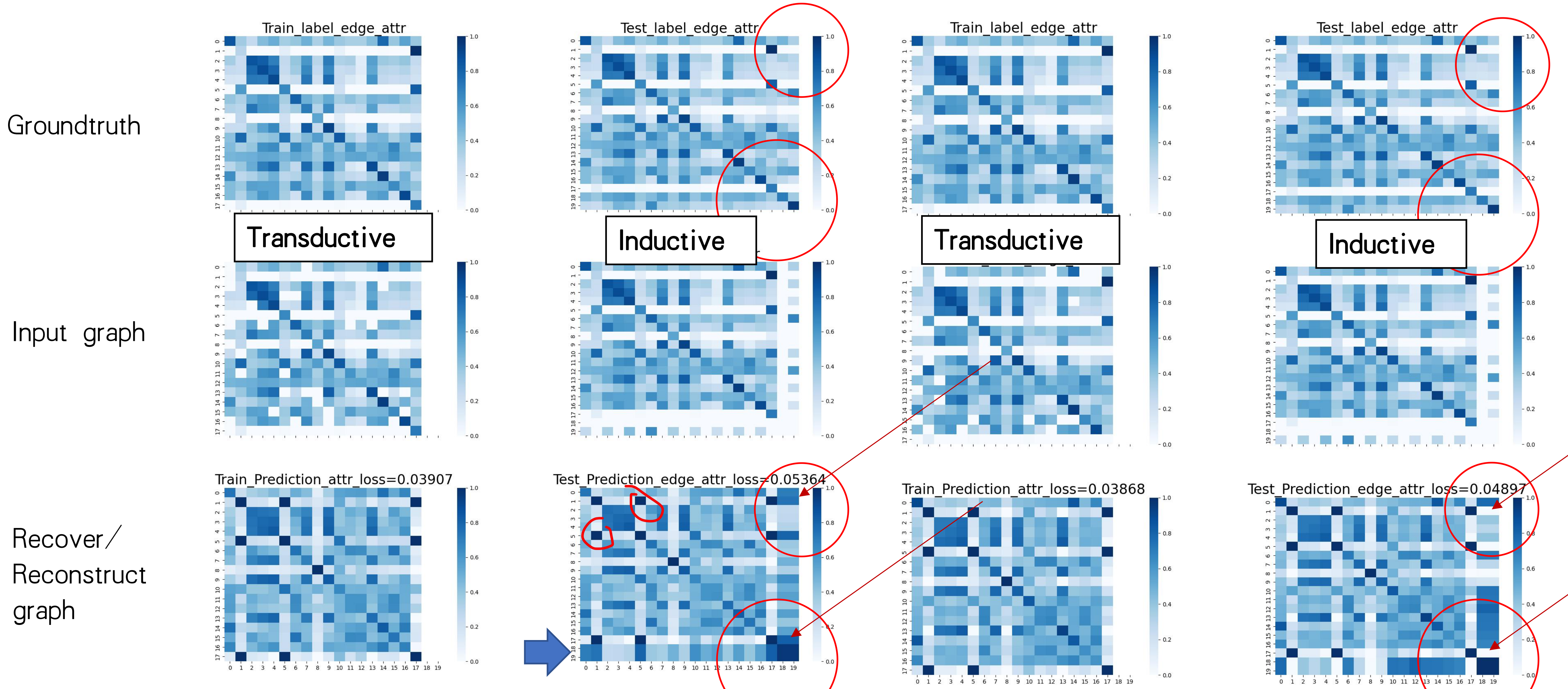
(a) Trans (b) Ind w/ c (c) Ind w/o c



CV_tasks_baseline (GAE)

w/ one-hot embedding **mse=0.0536**

w/ task embedding **mse = 0.0489 (+ 8%)**



DGAE Transdictive results

GAE ：w/ one–hot embedding mse=0.0536

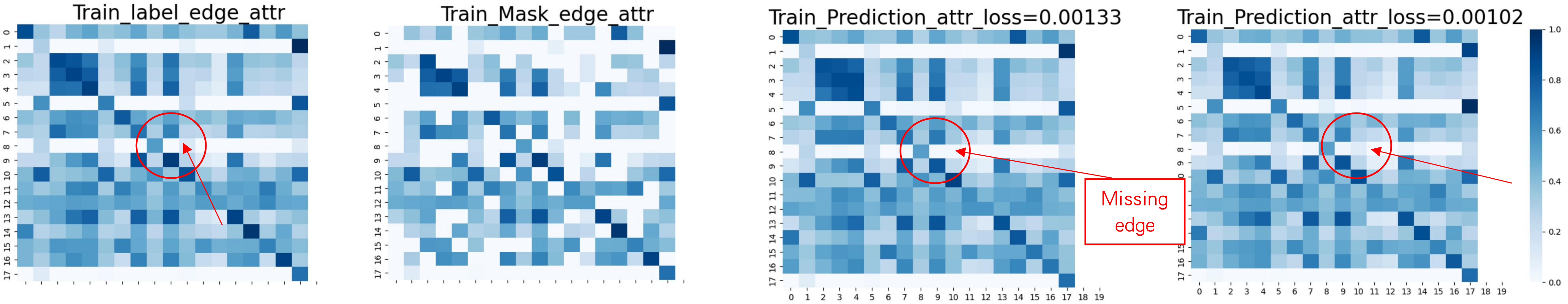
DGAE mse=0.00133

GAE ：w/ task embedding mse = 0.0489 (+ 8%)

DGAE mse=0.00102

Ground_truth

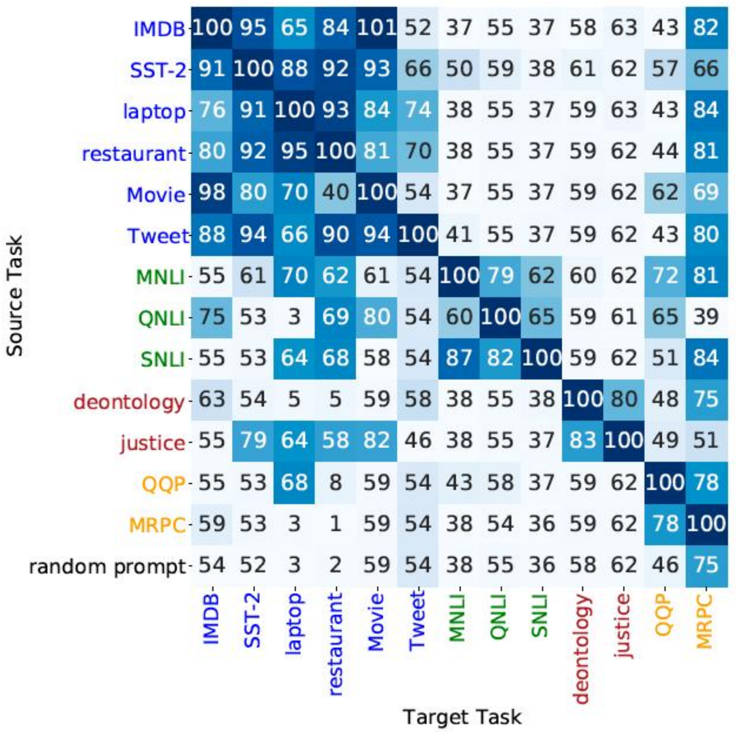
Random_mask



Recover the missing edges
w/o task embedding

Recover the missing edges
w/ task embedding mse： 20%+

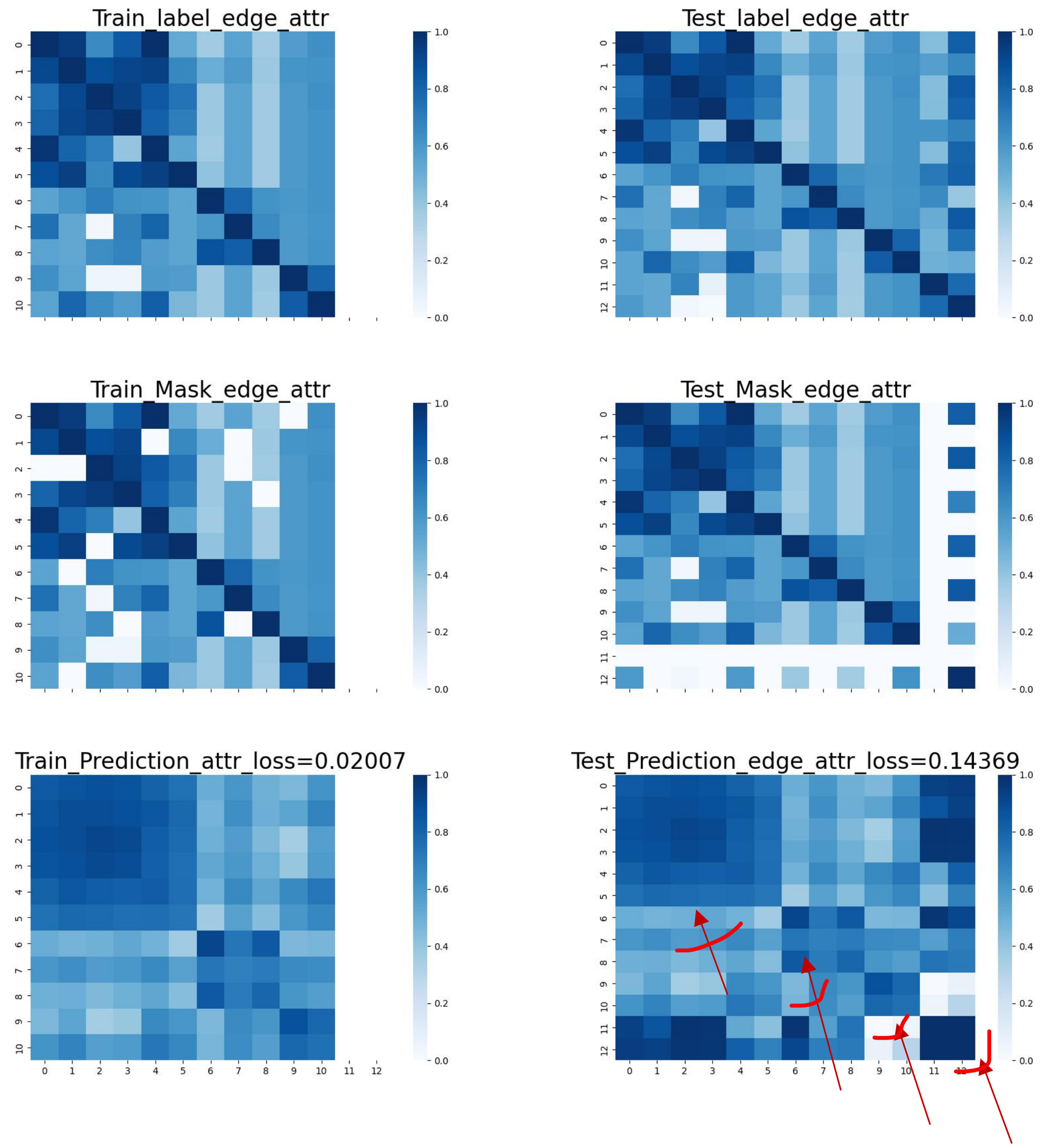
NLP_Result (Prompt Transferability)



Groundtruth

Transductive

Inductive



Colors of the task names indicate task types. Blue: SA. Green: NLI. Brown: EJ. Orange: PI.

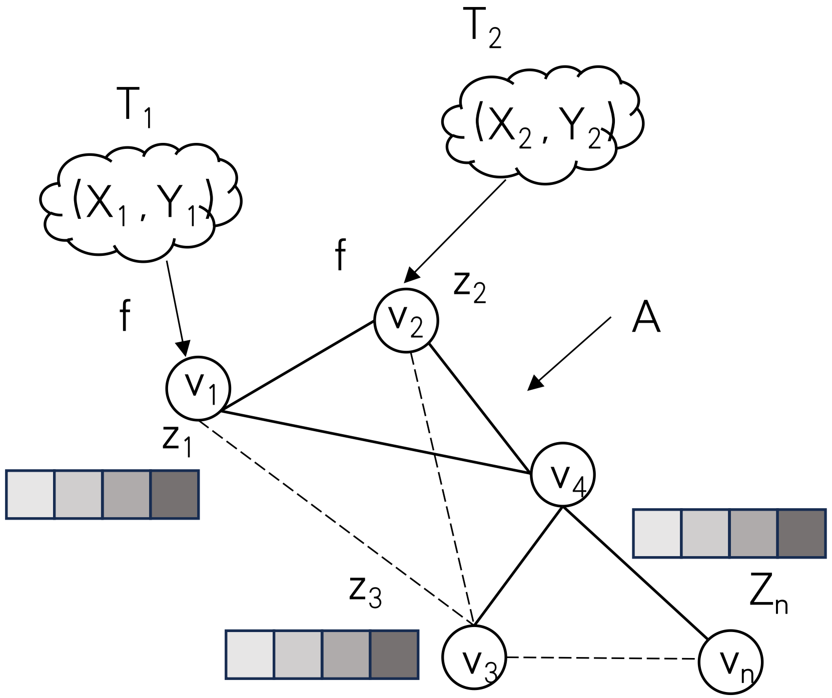
[On transferability of prompt tuning for natural language processing]

Summary

1. Verified the feasibility of using **directed graph learning** to **recover task transferability.**
2. Enhanced the restoration of graph structure using **task-specific representations.**
3. Compared the recovery and reconstruction capabilities of **directed graph models** with undirected graph models.
4. Validated the efficacy and accuracy of our approach in **both transductive and inductive** settings across real-world CV and NLP tasks.

Next step

1. Find task meta-information in real-world **medical** and **autonomous driving** scenarios.
2. Try to incorporate the edge representation in directed graph learning.



Possible extensions

Multi-source transferability: Address the multi-source ensemble selection challenge by utilizing **hyper-graph learning** combined with task embedding.

Questions?