Randomly Wired Neural Architecture Search for Cross-Domain Data Fusion

- 作者：pending
- 机构：pending
- 会议：pending（AAAI）
- 地址：pending
- 代码：pending
- 推荐预先阅读：Exploring Randomly Wired Neural Networks for Image Recognition

论文主要内容

摘要

在real world应用中，融合不同domain的数据对提升模型performance很有用。但是设计一个好的神经网络结构来融合不同特征需要专家知识、时间、算力。尽管已经有NAS方法用来确定最优架构，还是有人为干涉（搜索空间的设置、搜索算法等）。文章提出了Randomly Wired Domain Fusion (RWDF)方法，更加灵活，相比已有NAS-based方法，性能和SOTA相当、速度更快。

贡献

1. 相比其他基于NAS的跨域数据融合，RWDF会给出更加灵活的搜索空间
2. 设计了一个用二分图来生成网络的生成器去融合跨域信息
3. 相比手工设计的和NAS-based达到SOTA

研究内容

方法（Randomly Wired Domain-Fusion (RWDF)）

介绍

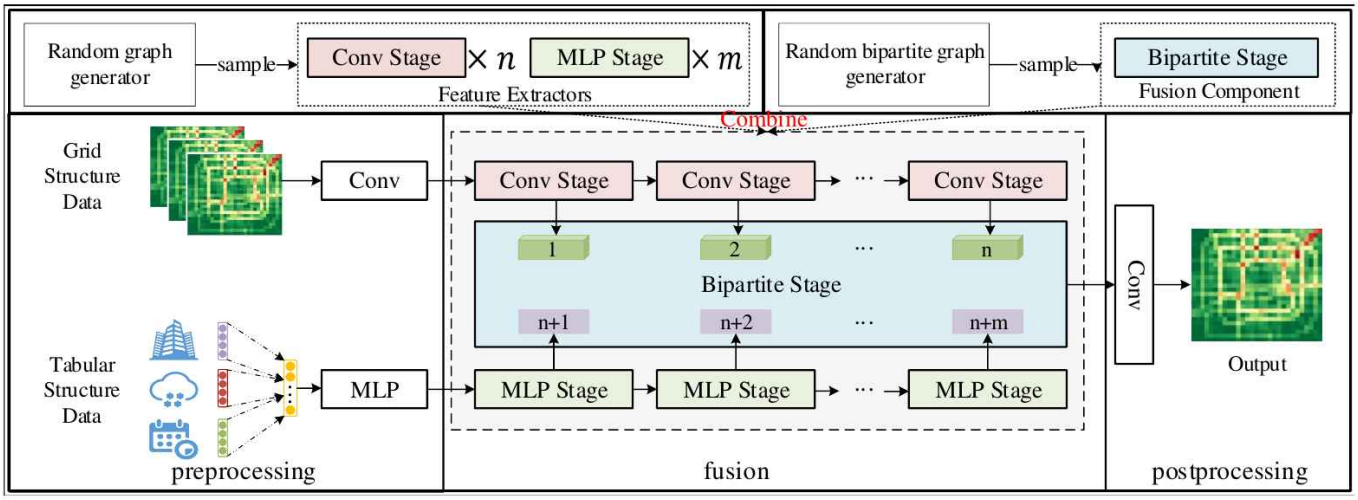


Figure 1: Framework of randomly wired neural networks for cross-domain data fusion. We sample feature extractors from random graph generator and fusion component from random bipartite graph generator separately. Feature extractors (*e.g.* Conv stages and MLP stages) learn the latent representation at different levels from each domain. Here, we present the feature extractors for grid structure data by CNN stages and tabular structure data by MLP stages. The fusion component fuses the features from all domains to serve the downstream applications.

流程：

- Preprocess
  - 使用卷积(Conv)提取 grid 结构的数据初始特征
  - 使用MLP去提取tabular 结构的数据初始特征
- fusion
  - 准备工作
    - 由**网络生成器**采样 $n+m$ 个random graphs
    - （**随机二分图生成器**）采样一个二分图，作为fusion组件
  - heuristic（某种人定规则）将这些random graphs转换成NN形式： n Conv和 m MLP
  - 使用这些NN（extracors）从不同domain去提取不同粒度、不同level的特征
  - 用fusion组件将Conv-feature、MLP-feature做特征级的融合
- postprocess
  - 卷积去做下游任务应用

Feature extractor （graph generator）

- Graph generator：将参数空间映射到graph空间 $g:\Omega \mapsto \mathcal{G}$

combined as neural architecture with a heuristic method, as we present in section . Then, we train the neural network

- g 有ER、BA、WS（另一篇论文），这篇文章在消融实验结果说了是这三种
- 类比例如：具体的 g 可以是一个多项式函数族， ω 就类比于多项式函数阶数，输出是一个具体多项式**函数实例**
- determinstic：确定g和 ω 就唯一确定一个graph
- stochastic：生成具有某种分布的graph（随机变量），再有一个随机种子才能确定具体生成的graph

- 无向图转化到DAG：（heuristic）
  - 某种策略给每个node标号0, 1，2，3...
  - 无向边变有向边：方向从节点编号小的指向大的
  - 增加S、T俩node
    - S 的输出都是input features的copy，橙色虚线边
    - T将所有input node的特征收集（unweighted average），蓝色虚线边

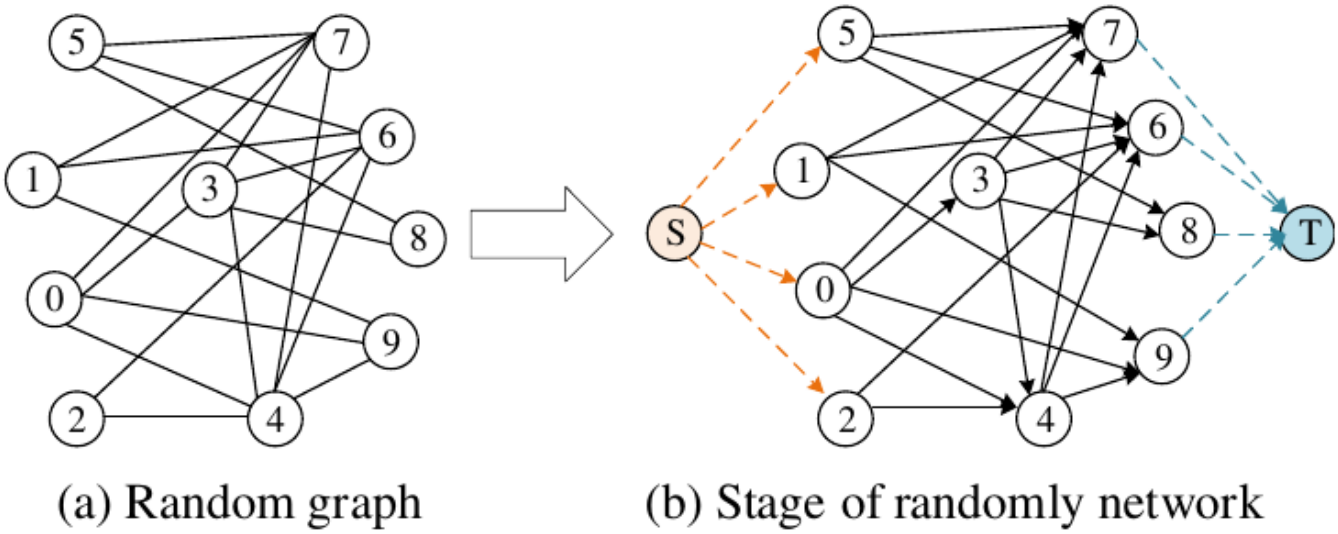


Figure 2: Random graph and randomly wired neural networks.

- 为每个node定义operation（DAG=>网络block），operation分三阶段
  - Aggregation：weighted sum 输入
  - Transformation：Conv（grid）或MLP（tabular）
  - distribution：把输出copy到每条出边上

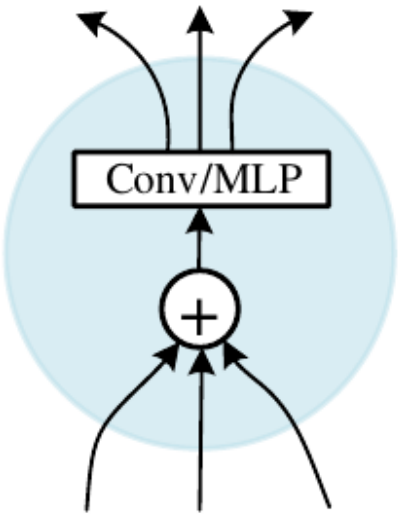


Figure 3: Node operations. Node operations illustrate the method that we transfer a random graph to a neural network. We first aggregate the information from the inputs nodes with a weighted sum. Then, the convoluted features are distributed to the output nodes.

Fusion component

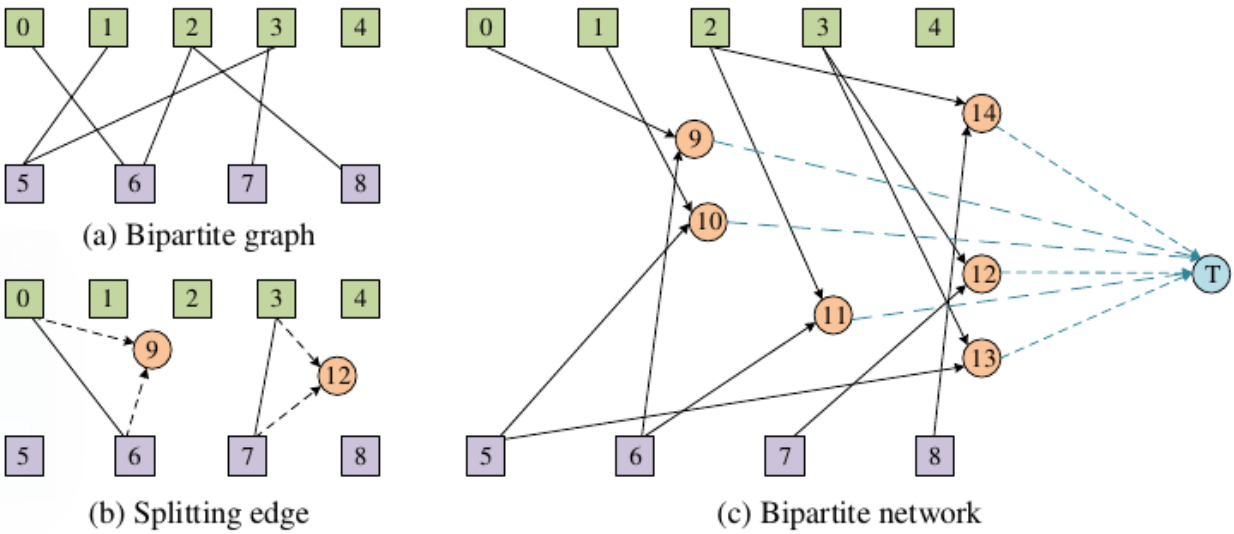


Figure 4: Bipartite network and randomly wired fusion neural network. Green square nodes indicate the features extracted from grid data, while purple square nodes indicate the features extracted from tabular data. The orange circle nodes are the fusion node operations.

- 生成方式和feature extractor类似，只是生成出来的是二分图（换个函数族）
- 二分图 => NN
  - 把二分图的每条边上增加一个node（无向转有向，指向新node）
    - 这个新node表示对输入的两个domain的数据进行融合操作（用fusion 组件）

- 将所有新node拼接成一个输出

Algorithm 2: Algorithm of transformation

Input: The input random bipartite grap, $G(U, V, E)$;
The input features $f_1, f_2, ..., f_{n_1}$,
$f_{n_1+1}, f_{n_1+2}, \cdots, f_{n_1+n_2}$
The number of node-sets, n_1, n_2;
The computation graph $make\_connection$, node operation op, Conv or MLP;
Output: The fusion component FC.

Add a collection target node T to FC.
$n = n_1 + n_2$;
$ouput_{nodes}$ = [];
for $e \in E$ **do**
    u, v = e.head, e.tail;
    $n = n + 1$;
    make_connection(FC, u, n);
    make_connection(FC, v, n);
    f_n = op(Concat(f_u, f_v));
    $output_{nodes}$.push(n);
for $u \in output_nodes$ **do**
    make_connection(FC, u, T);
$f_T = op(Concat(u|u \in output_nodes))$;

总体算法流程

Algorithm 1: Algorithm of RWDF

Input: The input data, X, E; The number of layers for both domain, n, m; The random seed sets, $\mathcal{S}$; The mapping function $g(\omega)$, $gb(n, m, \omega)$;

Output: The output Y.

$con_1, con_2, ..., con_n \sim g(\omega)$;
$mlp_1, mlp_2, ..., mlp_m \sim g(\omega)$;
$bg \sim gb(n, m, \omega)$;
input = X;
for $i = 1..n$ **do**
    $f_i = conv_1(input)$;
    input = f_i;
input = E;
for $i = 1..m$ **do**
    $f_{n+i-1} = mlp_i(input)$;
    input = f_{n+i-1};
out = bg$(f_1, f_2, ..., f_{n+m})$;
Y = Conv(out); return Y;

RWDF算法实例（Spatio-Temporal Forecasting）

- Task:
  - 输入
    - 交通流history： $X \in R^{L \times f \times H \times W}$ （grid）
      - L 是history长度、 f 是channel， H、W 是map大小
      - 时间 t 上的输入feature： $X_c = [X_{t-l_c \cdot p_c}, X_{t-l_c-1 \cdot p_c}, X_{t-p_c}]$
    - external factors E (tabular) holidays, the weather condition, and POI
  - 输出
    - 预测出的下一时间的spatio-temporal flow $Y = X_{t+1}$ ， $Y_i \in R^{t \times f \times H \times W}$

$$f_0 = X_c \tag{2}$$
$$f_i = conv_i(f_{i-1}), \qquad i = 1..n \tag{3}$$
$$f_{n+1} = mlp_1(E) \tag{4}$$
$$f_{n+i} = mlp_i(f_{n+i-1}), \qquad i = 2..m \tag{5}$$
$$out = bg(f_1, f_2, ..., f_{n+m}) \tag{6}$$
$$Y = Conv(out) \tag{7}$$

- 在特征融合的时候需要考虑align conv和MLP的特征尺寸：

$$E \in R^{b \times d}$$

$$f_i \in R^{b \times f\ t\bar{i}mesH \times W}$$

$$o = \text{Conv}\left([f_i \mid \text{Reshape}(E \cdot W)\right]$$

实验结果

实验设置

- 数据集：MNIST、 FashionMNIST、CIFAR-10
- internal nodes K=5
- 搜索花费100 epochs，前25epoch作为warm-up只训练net weight不训练RNN

Metric

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$
$$SMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{(|\hat{y}_i| + |y_i|)/2} * 100\%$$

Results

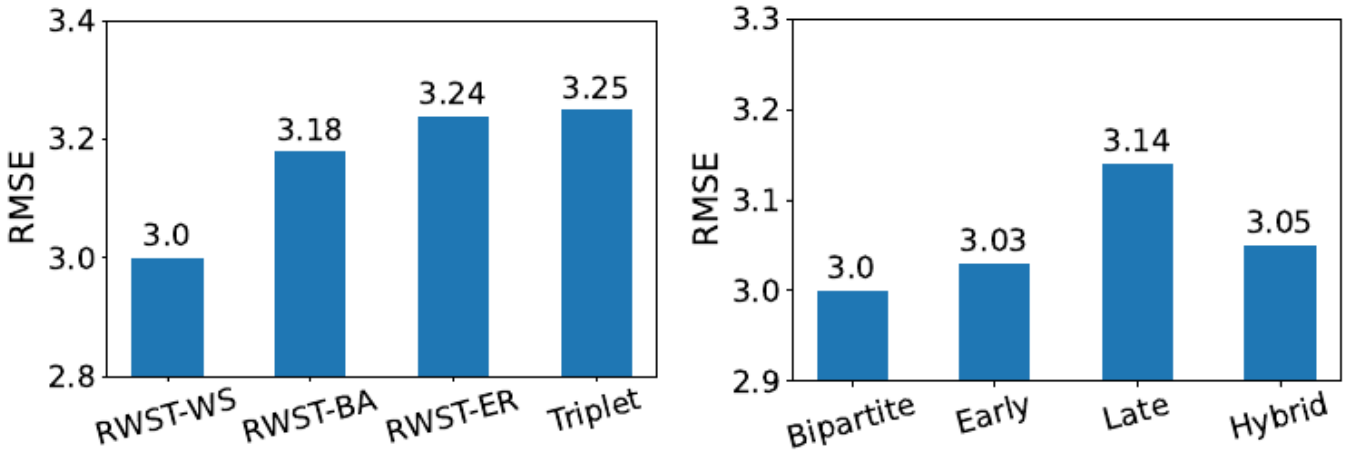
- STResNet：early fusion(low-level)
- DeepSTN：late fusion(high-level)
- CentralNet：hybrid fusion

Table 2: Experimental results on spatio-temporal forecasting tasks

	UnicomBJ		CrowdSQ		TaxiGY	
	RMSE	SMAPE	RMSE	SMAPE	RMSE	SMAPE
STResNet	45.95±1.3810	0.6853±2.98e-4	3.24±0.0242	0.6984±1.48e-4	2.48±0.0857	1.0835±6.80e-4
DeepSTN	44.65±0.8451	0.6655±1.81e-4	3.33±0.0005	0.7795±2.63e-4	2.22±0.0012	1.0985±0.07e-4
CentralNet	43.55±0.2327	0.6510±0.58e-4	3.19±0.0116	0.6649±3.21e-4	2.20±0.0043	0.9498±1.03e-4
MFAS	43.60±0.9161	0.6512±6.36e-4	3.26±0.0003	0.6771±1.12e-4	2.17±0.0011	0.9498±1.03e-4
AutoST	42.97±0.0013	0.6213±1.50e-4	2.95±0.0005	0.6909±4.60e-4	2.16±0.0132	0.9707±3.07e-4
RWDF	43.08±2.9213	0.6390±7.98e-4	2.98±0.0006	0.6516±7.14e-4	2.13±0.0003	0.9772±6.28e-4

- hybrid方法比STResNet、DeepSTN好，说明融合不同level、不同粒度的特征是对task有用的

消融实验



(a) Feature extractors (b) Fusion components

Figure 5: Evaluation of model variants.

- 三种不同（用来特征提取的）随机图生成器
  - Triplet是ReLU-convolutional-BN
  - 说明graph generator很重要
- 不同fusion方式
  - 二分图方式灵活融合不同粒度的特征，表现最好

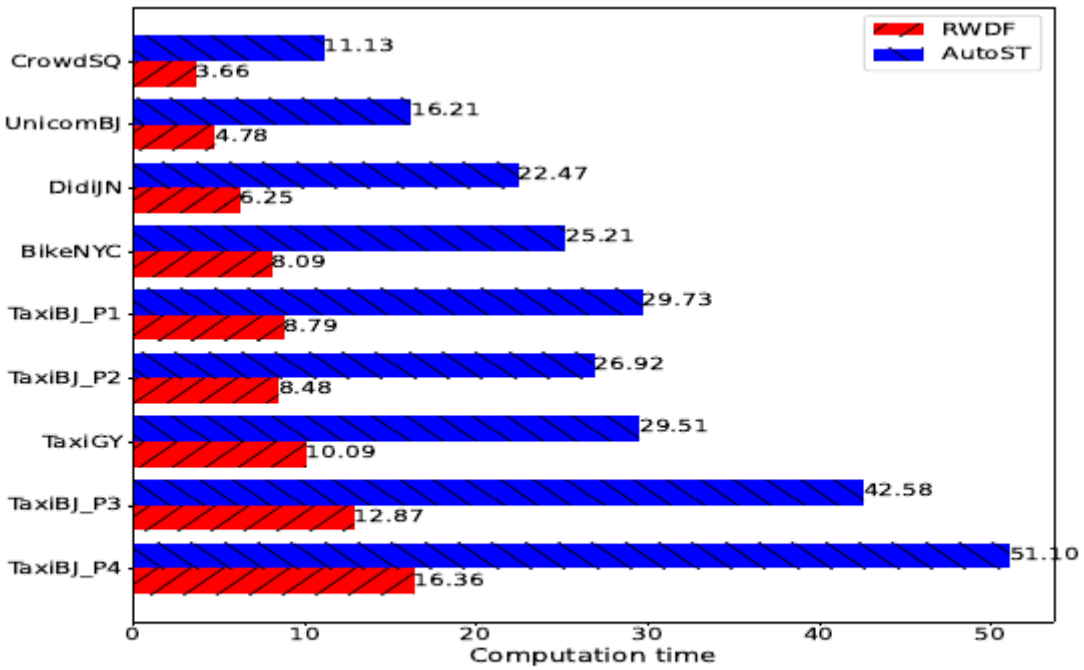


Figure 6: Computation consuming of RWDF and AutoST.

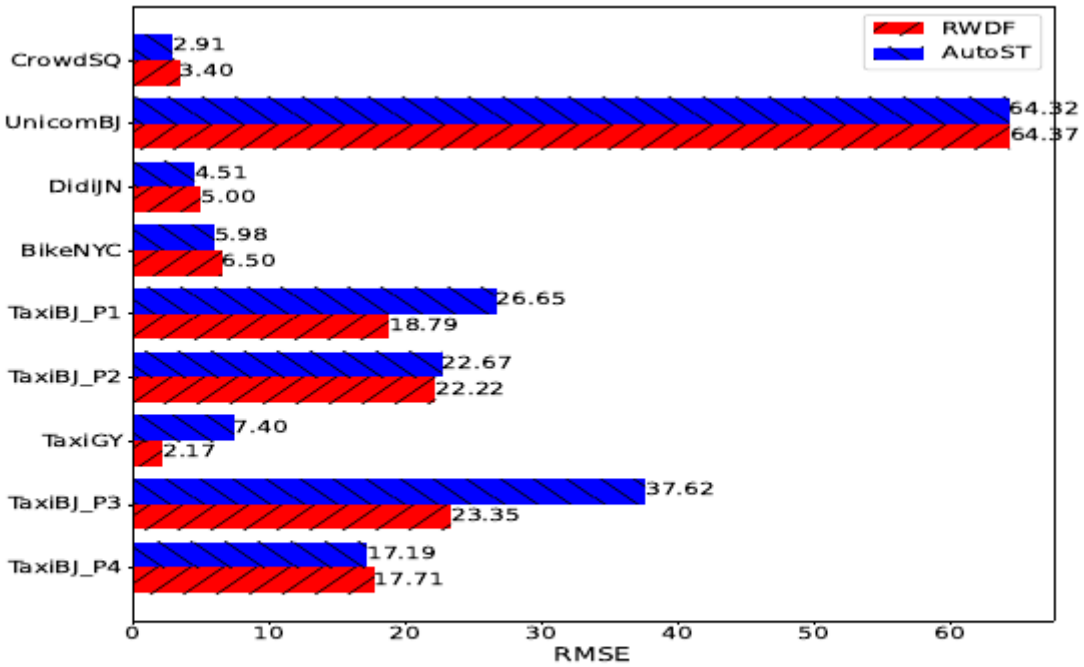
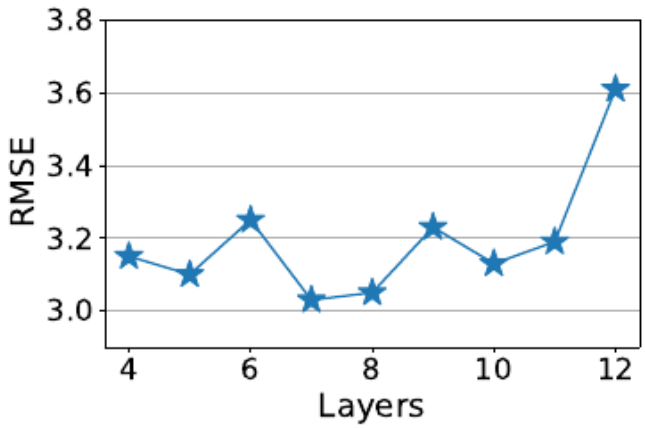
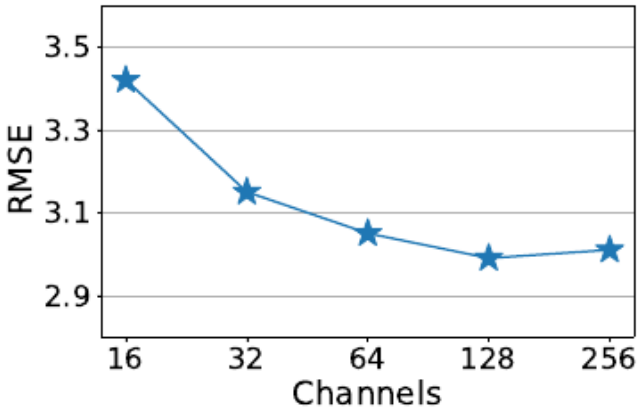


Figure 7: RMSE of RWDF and AutoST.

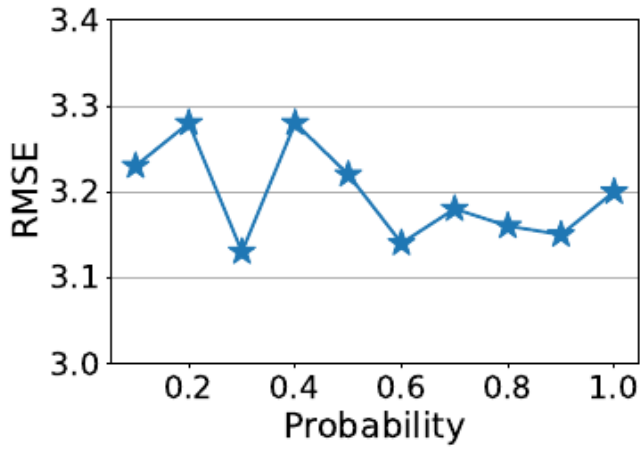
- 计算时间
  - 性能和SOTA相当，速度更快
  - 对比NAS方法（AutoST），速度快4倍
  - 因为在随机图生成中加入了先验知识（搜索空间小了）



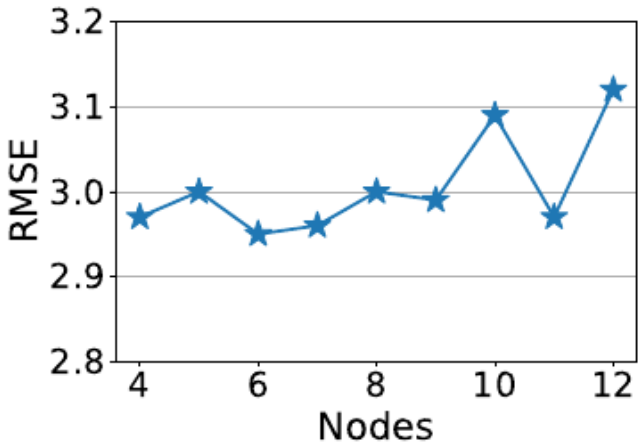
(a) The number of layers.



(b) The number of channels.



(c) The probability of generator.



(d) The number of nodes.

Figure 8: Evaluation of the hyperparameters in CrowdSQ.

- 整个网络的层数
  - 7层最优（图a）
  - 层数增加，也就带来二分图的node增加，也就增加了收敛难度
- channel数量
  - 数量太小会导致model capacity不足以学习flow pattern
  - 128层最优（图b）
- 二分图里采样的概率
  - 不敏感，说明对这个超参鲁棒（图c）
  - 高的概率，0.6-0.9稳定
- 随机图中的node数
  - 数量增加会导致metric变差（图d）

- 最优是6