SCIENCE CHINA
Information Sciences

• Supplementary File •

Can a CNN perform well on network traffic matrices? A case study on formulating network data to maximize the benefits of deep learning

Ruotian Xie[1,2], Kun Xie[1*], Xin Wang[3], Jigang Wen[4], Gaogang Xie[5] & Guangxing Zhang[6]

[1]*College of Computer Science and Electronics Engineering, Hunan University, Changsha 410082, China*
[2]*School of Software, Nanchang University, Nanchang 330047, China*
[3]*Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook 11794, USA*
[4]*School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China*
[5]*Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China*
[6]*Department of Network Technology Research Center, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China*

Appendix A Interpretability of R&E neural network in matrix rearrangement

With weighted permutation matrices learned in the R&E network, we would like to answer the following two questions:

- Which rows/columns are important in CNN's feature extraction?
- How does an R&E network enhance a matrix's locality?

To investigate the effectiveness of the R&E network in enhancing the matrix's locality for CNN learning, we draw the original TM X and rearranged TM X' of Abilene in Fig. A1, where W_r denotes a row-weighted permutation matrix and W_c denotes a column-weighted permutation matrix. We observe that: • Large values located on different rows and columns in the original TM X are arranged to stay together in the final rearranged X'. Compared with the original matrix X, the locality in the matrix X' is obviously enhanced. This demonstrates that our R&E network is effective in rearranging the matrix data to strengthen the locality so that CNN can more effectively extract the features in matrix data.

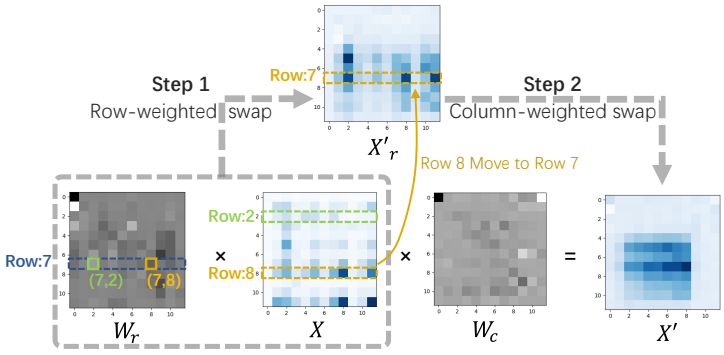

Figure A1 Swap Abilene TM with weighted permutation matrices trained.

To further explore how X becomes X', we decompose the process of rearrangement into row-weighted swap (Step 1) and column-weighted swap (Step 2), as shown at the top of Fig. A1. X'_r denotes the rearranged TM after row swap. The darker the color of the element in the matrix, the greater the value there. We further observe that:

- After the row-weighted swap is performed on X, matrix X'_r is obtained. Compared with the original TM X, the similarity of adjacent rows is enhanced in X'_r.
- After column-weighted swap is further performed on X'_r, the final rearranged X' is obtained. Compared with X'_r, the similarities of adjacent columns in X' obviously increase.

* Corresponding author (email: xiekun@hnu.edu.cn)

To interpret which rows/columns are important for matrix rearrangement to facilitate CNN learning, taking the 7th row of rearranged X_r' as an example, we write the following expression to illustrate how this row is obtained using the original X:

$$row(X_r^{'}, 7) = \sum_{i=0}^{11} \Big(W_r^{(7,i)} \cdot row(X, i) \Big), \tag{A1}$$

where $row(X_r', 7)$ denotes the 7-th row of X_r', $W_r^{(7,i)}$ denotes the $(7, i)$-th element in matrix W_r, and $row(X, i)$ denotes the i-th row of X.

Intuitively, the 7-th row of X_r' is the weighted sum of the rows in the original X, with the weight of the i-th row of X being $W_r^{(7,i)}$. The importance of rows in X for $row(X_r', 7)$ can be reflected according to their weights. In this example, we find that the $(7, 8)$ element (yellow square) is the largest, which means the 8-th row of X (bottom yellow dashed box) is the most important row in the original TM to build the 7-th row of X_r' (top yellow dashed box). In contrast, the 2-th row has the least weight (green square) and thus brings less information to build the 7-th column of X_r'.

Table A1 Performance of plug R&E into existing prediction methods (with a historical matrix length $T = 8$)

		AER	ANMAE	ARMSE	Time(s)
Abilene	**ST-ResNet** [1]	0.2406 ±0.0022	0.2003 ±0.0012	0.00147 ±0.00001	**377 ±8**
	ST-ResNet (R&E*)	**0.2339 ±0.0021**	**0.1933 ±0.0019**	**0.00144 ±0.00001**	392 ±16
	ST-GSP [2]	0.2978 ±0.0025	0.2556 ±0.0025	0.00188 ±0.00001	**7331 ±330**
	ST-GSP (R&E*)	**0.2959 ±0.0021**	**0.2552 ±0.0017**	**0.00184 ±0.00001**	7354 ±238
	SaConv [3]	0.1882 ±0.0014	0.132 ±0.0009	0.00113 ±0.00001	**6188 ±297**
	SaConv (R&E*)	**0.1828 ±0.0008**	**0.1313 ±0.0012**	**0.00111 ±0.00001**	6505 ±216
	DeepSTN+ [4]	0.2762 ±0.0013	0.2531 ±0.0011	0.00149 ±0.00001	**1854 ±49**
	DeepSTN+ (R&E*)	**0.2731 ±0.0019**	**0.2518 ±0.0011**	**0.00148 ±0.00001**	1947 ±51
	CRNN [5]	0.2068 ±0.0016	0.1652 ±0.0017	0.00126 ±0.00001	**1101 ±50**
	CRNN (R&E*)	**0.2041 ±0.0016**	**0.1645 ±0.0014**	**0.00125 ±0.00001**	1215 ±61
	M^{2}STL [6] (GNN-based)	0.1935 ±0.0008	0.1723 ±0.0016	0.00118 ±0.00001	407 ±9
	MC-STL [7] (GNN-based)	0.1912 ±0.0008	0.1671 ±0.0013	0.00116 ±0.00001	519 ±13
	ST-SSL [8] (GNN-based)	0.5237 ±0.0028	0.4731 ±0.003	0.00321 ±0.00002	720 ±18
	iTransformer [9] (Trans-based)	0.1897 ±0.001	0.1503 ±0.0015	0.00114 ±0.00001	341 ±9
	PatchTST [10] (Trans-based)	0.1842 ±0.0013	0.1273 ±0.0011	0.0012 ±0.00001	2528 ±86
GÉANT	**ST-ResNet** [1]	0.5455 ±0.0032	0.3634 ±0.0028	32315 ±178	**530 ±19**
	ST-ResNet (R&E*)	**0.4295 ±0.0035**	**0.3381 ±0.0023**	**25484 ±224**	555 ±15
	ST-GSP [2]	0.2215 ±0.0009	0.2206 ±0.0011	13654 ±81	**8044 ±204**
	ST-GSP (R&E*)	**0.2182 ±0.0019**	**0.2195 ±0.0015**	**13630 ±101**	8096 ±364
	SaConv [3]	0.1652 ±0.0015	0.1982 ±0.0018	9696 ±89	**7847 ±377**
	SaConv (R&E*)	**0.1622 ±0.001**	**0.1924 ±0.0013**	**9437 ±64**	8099 ±309
	DeepSTN+ [4]	0.2467 ±0.0017	0.2747 ±0.0016	14990 ±66	**3238 ±84**
	DeepSTN+ (R&E*)	**0.2439 ±0.0016**	**0.2736 ±0.0017**	**14895 ±92**	3244 ±109
	CRNN [5]	0.1991 ±0.0017	0.2095 ±0.001	11935 ±111	**2910 ±77**
	CRNN (R&E*)	**0.1865 ±0.0009**	**0.1981 ±0.0017**	**11324 ±102**	3019 ±134
	M^{2}STL [6] (GNN-based)	0.1935 ±0.0008	0.2135 ±0.0011	11665 ±100	590 ±21
	MC-STL [7] (GNN-based)	0.1731 ±0.0009	0.2231 ±0.001	10276 ±44	2294 ±69
	ST-SSL [8] (GNN-based)	0.2107 ±0.0012	0.1791 ±0.0013	12512 ±121	759 ±17
	iTransformer [9] (Trans-based)	0.1865 ±0.0013	0.2186 ±0.002	11306 ±50	1142 ±34
	PatchTST [10] (Trans-based)	0.1559 ±0.0006	0.1628 ±0.0016	9342 ±45	2140 ±53
TaxiBJ(In&Out)	**ST-ResNet** [1]	0.1638 ±0.0011	0.1616 ±0.001	23.9806 ±0.1176	**1755 ±71**
	ST-ResNet (R&E*)	**0.1336 ±0.0008**	**0.1236 ±0.0006**	**19.5972 ±0.1233**	1835 ±58
	ST-GSP [2]	0.1387 ±0.0007	0.1248 ±0.0006	19.4532 ±0.1933	**2335 ±63**
	ST-GSP (R&E*)	**0.1353 ±0.001**	**0.1231 ±0.0011**	**19.3730 ±0.1086**	2348 ±112
	SaConv [3]	0.2017 ±0.0012	0.1497 ±0.0008	27.7701 ±0.2388	**25662 ±674**
	SaConv (R&E*)	**0.1787 ±0.0014**	**0.1432 ±0.0014**	**25.8549 ±0.2484**	26548 ±974
	DeepSTN+ [4]	0.182 ±0.0015	0.1565 ±0.0009	27.2673 ±0.2228	**4671 ±201**
	DeepSTN+ (R&E*)	**0.1732 ±0.001**	**0.1465 ±0.0012**	**26.5983 ±0.2439**	4760 ±128
	CRNN [5]	0.1316 ±0.0013	0.1234 ±0.0012	19.9244 ±0.0847	**5649 ±263**
	CRNN (R&E*)	**0.126 ±0.0009**	**0.1183 ±0.0008**	**19.0381 ±0.0911**	5847 ±292
	M^{2}STL [6] (GNN-based)	0.1214 ±0.0011	0.1114 ±0.0006	18.4664 ±0.1021	2770 ±104
	MC-STL [7] (GNN-based)	0.1112 ±0.001	0.1027 ±0.0004	17.6051 ±0.0736	5383 ±230
	ST-SSL [8] (GNN-based)	0.1244 ±0.0008	0.1125 ±0.0005	18.8120 ±0.0984	7724 ±243
	iTransformer [9] (Trans-based)	0.9286 ±0.0086	1.0487 ±0.008	29.8069 ±0.1562	7690 ±299
	PatchTST [10] (Trans-based)	0.2707 ±0.0015	0.2412 ±0.0013	40.8342 ±0.3687	4074 ±105

Appendix B Enhance CNN-based prediction models using R&E neural networks

To validate that our R&E network can enhance CNN's ability to understand data features even when CNN works together with other deep learning (DL) structures in a complex neural network, we download the source codes of 5 CNN-based prediction network models (ST-ResNet [1], ST-GSP [2], SaConv [3], DeepSTN+ [4], and CRNN [5]). We plug our R&E network into the CNN modules while following all other architectures in these models. For comparison (**RQ3)**, we also evaluate three recent GNN-based traffic prediction models (M^{2}STL [6], MC-STL [7] and ST-SSL [8]), two recent Transformer-based multivariate prediction models iTransformer [9] and PatchTST [10].

In the training process, since deep and complex network architectures require a lot of training to improve the performance of the embedding, updating the R&E parameters at the beginning may affect the model's stability. Therefore, we update the R&E parameters after 40% of the total training iterations. All other settings remain consistent with previous experiments, with mean performance and standard deviation reported as ?mean std? in the tables. Table A1 lists the prediction performance. In Table A1, "R&E*" denotes R&E network plugged models. Three GNN-based and two Transformer-based methods are grayed out as they cannot integrate the R&E network. The Table reveal that:

• Our R&E network is a general technique to enhance the feature extraction ability of CNN, even when CNNs work together with other complex DL modules. Taking TaxiBJ as an example, compared with the original prediction models, the AERs of ST-ResNet, ST-GSP, SaConv, DeepSTN+, and CRNN are reduced by 22.6%, 2.5%, 12.8%, 5.0% and 4.4% in models with our R&E plugged in.

• Although the training time under R&E plugged models is larger than that under the original models, the training time gap between these two kinds of models is small.

• CNN with R&E can achieve similar or even better performance than GNN-based models without relying on graph information. Despite MC-STL incorporating graph structure in Abilene, SaConv (R&E) achieves lower AER and ANMAE values of '0.1828' and '0.1313' respectively compared to MC-STL's '0.1912' and '0.1671'. In the case of TaxiBJ, although the performance under CRNN (R&E) is slightly worse than that under MC-STL, it's important to note that MC-STL incorporates graph structure and rich external factors such as temperature, wind speed, and holidays, while CRNN (R&E) only utilizes basic matrix data.

• CNN with R&E can achieve similar or even better performance than Transformer-based models. For instance in Abilene, SaConv (R&E) achieves lower AER and ARMSE values compared to PatchTST. Furthermore, both iTransformer and PatchTST perform poorly in TaxiBJ. This is because these two methods treat TaxiBJ as multivariate not matrix data, whereas TaxiBJ has clear spatial correlations, and the five CNN-based models perform better in capturing these, leading to superior results compared to two Transformer-based models.

These results demonstrate that our R&E network is very effective in enhancing the feature extraction ability of CNN modules while only bringing a very small time burden in the process of model training. By integrating the R&E network, CNN-based methods achieve comparable or superior performance to GNN- and Transformer-based models, even without relying on graph information or additional external factors.

Appendix C Why R&E Is Not Attention or Squeeze-and-Excitation(SE)

The reinforcement in R&E network is radically different from attention-based or SE-style reweighting, both mathematically and structurally:

(1) Reversible structure. Given an input traffic matrix $X \in \mathbb{R}^{N\times N}$, R&E applies a bilinear transformation

$$X' = W_r \cdot X \cdot W_c, \quad \hat{X} = W_r^{-1} \cdot \hat{X}' \cdot W_c^{-1}, \tag{C1}$$

where W_r and W_c are learnable weighted permutation matrices shared by the WS and Restoration layers. This design ensures that rearrangement and enhancement are used only to assist convolution, while the final output strictly preserves the original row–column ordering. In contrast, attention and SE permanently modify feature representations and do not support structural restoration.

(2) Different reinforcement operators. Attention performs element-wise weighting via the Hadamard product ($X' = A \odot X$), and SE rescales the entire feature map using a scalar or channel-wise coefficient ($X' = \alpha X$). R&E instead uses **matrix multiplication**, where reinforcement is applied to rows and columns rather than individual entries or whole matrices.

(3) Different reinforced objects. Left/right multiplication by W_r/W_c enhances correlations among source and destination node subspaces, respectively. Thus, R&E reinforces structured node interactions, whereas attention and SE treat the input as a feature tensor to be reweighted.

References

1 Zhang J, Zheng Y, Qi D. Deep spatio-temporal residual networks for citywide crowd flows prediction. In: Thirty-first AAAI conference on artificial intelligence, 2017

2 Zhao L, Gao M, Wang Z. St-gsp: Spatial-temporal global semantic representation learning for urban flow prediction. In: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, 2022. 1443–1451

3 Lin Z, Li M, Zheng Z, et al. Self-attention convlstm for spatiotemporal prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, volume 34. 11531–11538

4 Lin Z, Feng J, Lu Z, et al. Deepstn+: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis. In: Proceedings of the AAAI conference on artificial intelligence, 2019, volume 33. 1020–1027

5 Gao K, Li D, Chen L, et al. Predicting traffic demand matrix by considering inter-flow correlations. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE2020. 165–170

6 Xie R, Wen J, Chen X, et al. M2stl: Multi-range multi-level spatial-temporal learning model for network traffic prediction. IEEE Transactions on Network Science and Engineering, 2024, 11: 4315–4329

7 Zhang X, Gong Y, Zhang X, et al. Mask-and contrast-enhanced spatio-temporal learning for urban flow prediction. In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, 2023. 3298–3307

8 Ji J, Wang J, Huang C, et al. Spatio-temporal self-supervised learning for traffic flow prediction. In: Proceedings of the AAAI conference on artificial intelligence, 2023, volume 37. 4356–4364

9 Liu Y, Hu T, Zhang H, et al. itransformer: Inverted transformers are effective for time series forecasting. In: The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net2024

10 Nie Y, Nguyen N H, Sinthong P, et al. A time series is worth 64 words: Long-term forecasting with transformers. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net2023