AutoST: Efficient Neural Architecture Search for Spatio-Temporal Prediction

Contribution:

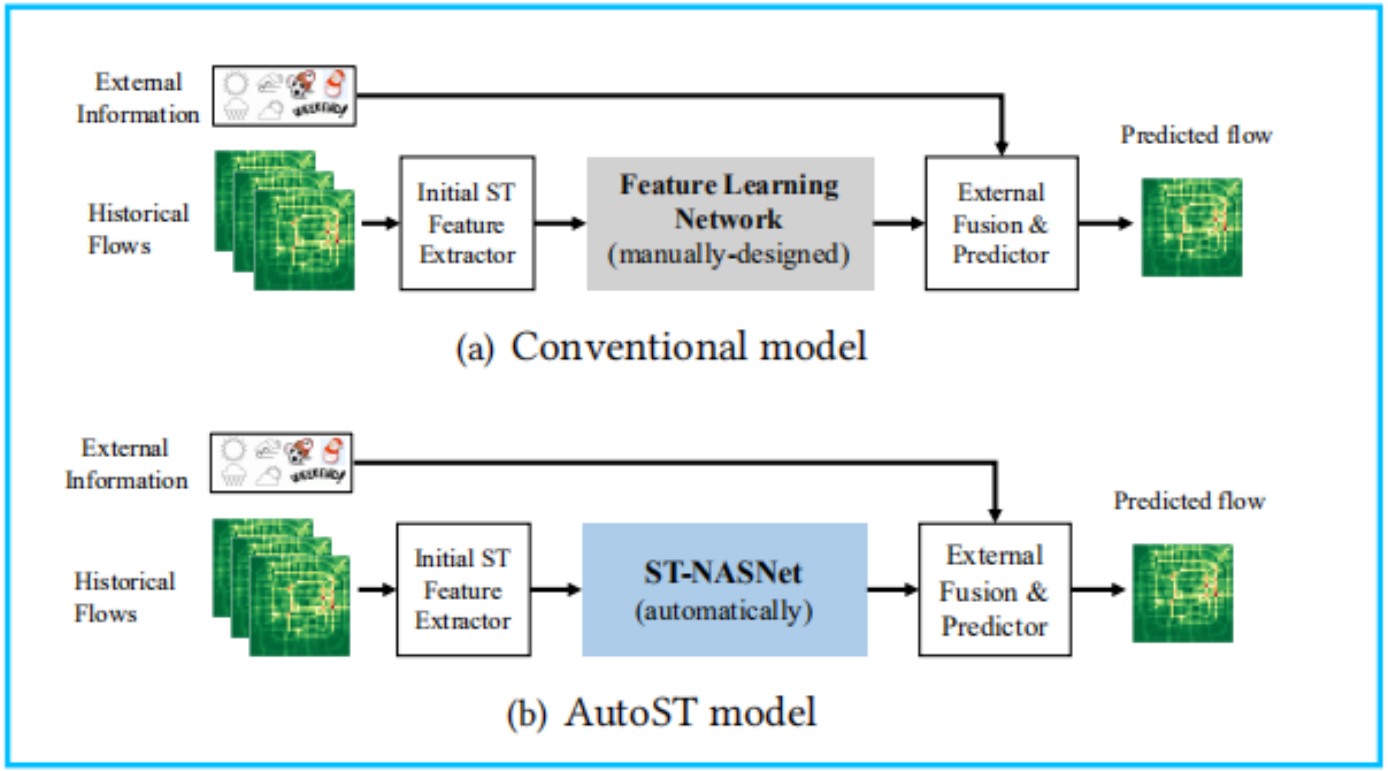


Figure 2: Conventional model vs. our AutoST model

- Proposed a novel model named AutoST for spatio-temporal prediction (the neural architecture search technique to dynamically capture the various-range spatial correlations and to fuse multi-level features).
- Designed a novel search space tailored for ST prediction.(与DARTS相比)

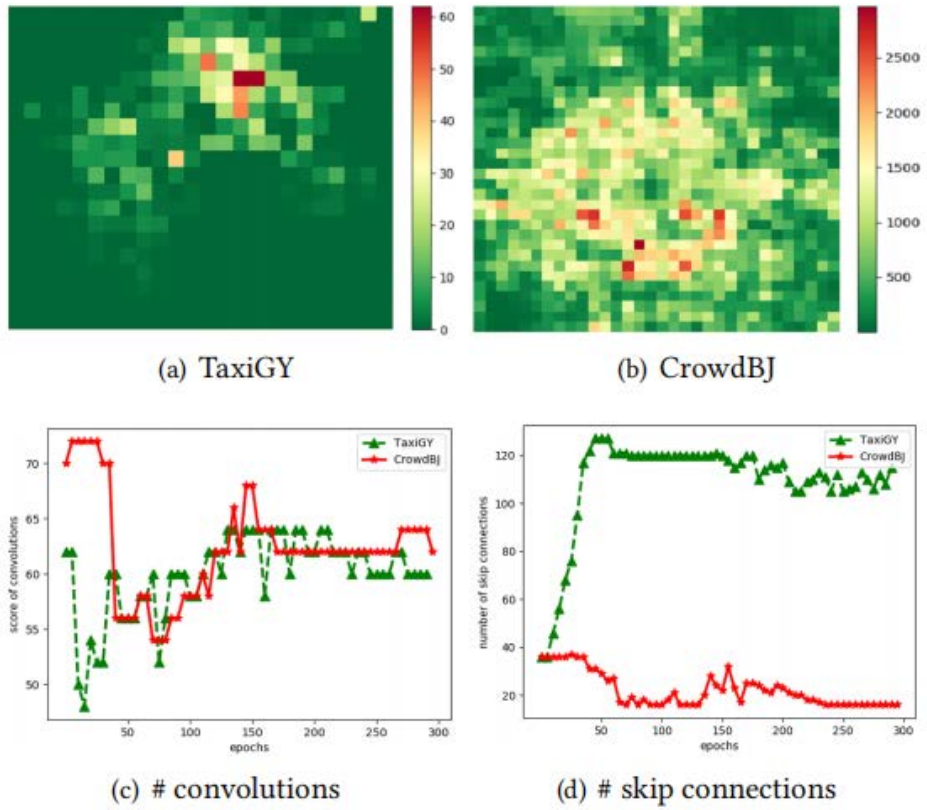
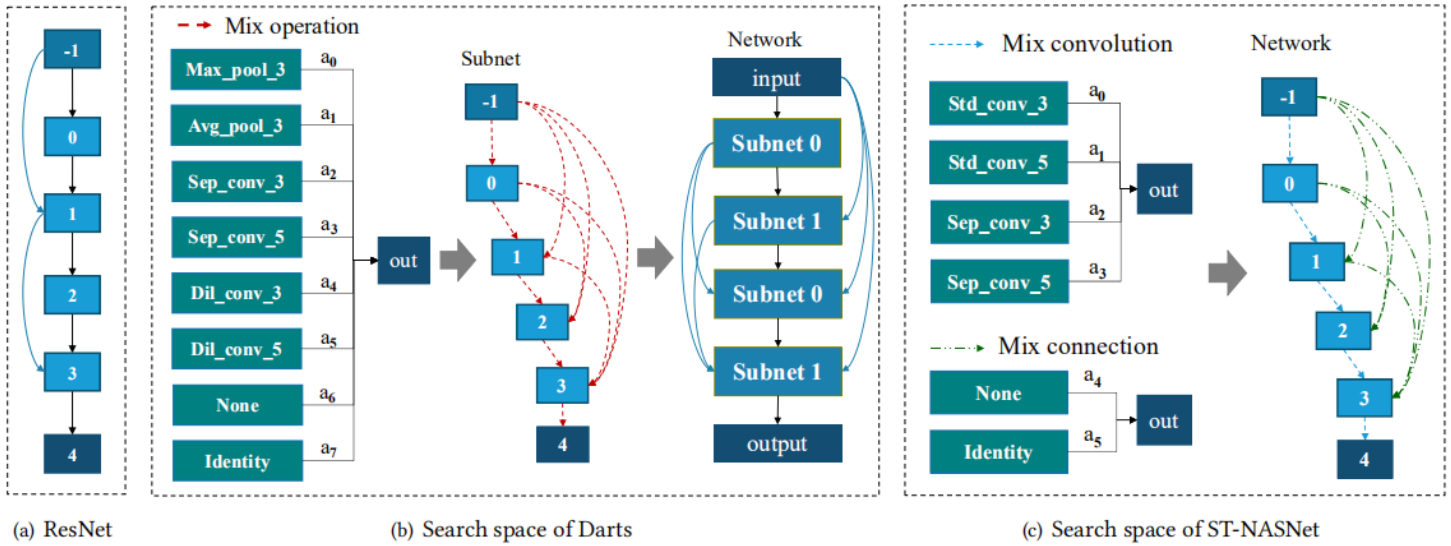


Figure 1: Illustration of the relationship between flow distributions (a and b) and neural architectures (c and d).

- 不同城市有不同的最优神经网络架构。
- 已有模型忽视的两个问题：
- （1）不同城市有不同的空间范围偏好【与交通较不发达的城市相比，核心城市应该以更长的距离范围作为邻居信息】，但用来建模邻居相关性范围的卷积核的大小通常是固定的，且是根据经验设置。
- （2）不同城市具有不同的架构偏好，核心城市通常更关注全局空间相关性。而当前方法通常使用残差网络来聚合相邻层中的特征，无法融合低级特征和高级特征。【低级特征描述局部信息】

Spatio-Temporal Search Space



(a) ResNet (b) Search space of Darts (c) Search space of ST-NASNet

Figure 3: Architectures comparision (dashed arrow indicates learnable operation, solid arrow indicates fixed operation): (a) Residual network: fixed architecture (b) the search space of Darts: widely used search space in image domain (c) the search space of ST-NASNet: proposed search space in ST domain (best view in color).

- （1）残差网络：无法融合低级特征和高级特征。
- （2）Darts：可能的网络架构总数太多，可能引起巨大的内存消耗。

NAS网络由三个模块组成：

1.候选单元模块（定义搜索单元）
（1）临近区域相互影响，选择卷积操作，且应考虑不同大小卷积核；
（2）全局相关性很重要，因此通常堆叠多个卷积层来捕获大范围城市依赖关系；
（3）去除pooling操作，pooling操作可能引起信息丢失。

2.操作块模块（基于梯度的搜索策略需要对所有可能的操作执行加权和以使搜索空间连续）

$$\bar{\mathbf{c}}_i(\mathbf{x}) = \sum_{c \in \mathcal{S}_c} \sigma(a_i^c) f(\mathbf{x}; \theta_i^c) \quad (2)$$

$$\bar{\mathbf{s}}_{i,j}(\mathbf{x}) = \sum_{s \in \mathcal{S}_s} \sigma(a_{i,j}^s) s_{i,j} \quad (3)$$

3.NAS网络模块

$$\mathbf{o}_l = \bar{\mathbf{c}}_l(\mathbf{o}_{l-1}) + \sum_{i=1}^{l-1} \bar{\mathbf{s}}_i(\mathbf{o}_i) \quad (4)$$

AutoST for Spatio-Temporal Prediction

- 三个时空预测模型：STResNet，ST-3DNet and DeepSTN

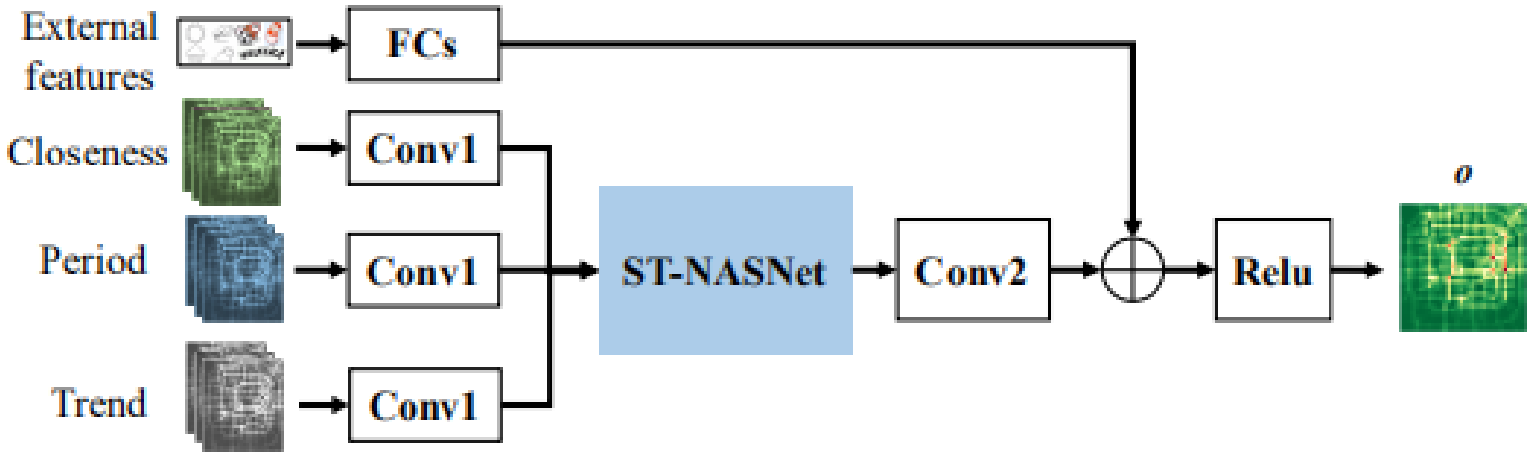


Figure 4: Illustration of AutoST for ST prediction

$$\mathcal{A} = \{a_l^c, a_l^s\}, l = 1, ..., L, c = 1, ..., n_c, s = 1, ..., n_s \quad (5)$$

$$\mathcal{M} = \{\theta_l^c, \theta_{c_1}, \theta_{c_2}, \theta_{fc}\}, l = 1, ..., L, c = 1, ..., n_c \quad (6)$$

$$\mathbf{o} = Relu(f(\mathbf{o}_L; \theta_{c_2}) + f(\mathbf{x}_e; \theta_{fc})) \quad (7)$$

Algorithm and Optimization

Algorithm 1: Search algorithm of AutoST

Input: Historical flows: $\{\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}\}$;
external features: $\{\mathbf{e}_0, \ldots, \mathbf{e}_{n-1}\}$
Output: learned AutoST model

construct training set $\mathcal{D}_{train}$ and validation set $\mathcal{D}_{valid}$
// search the architecture
initialize all trainable parameters
repeat
  // update model parameters
  randomly select a batch from $\mathcal{D}_{train}$
  forward on $\mathcal{D}_{train}$ to get $\mathcal{L}_{train}$
  $\theta' = \theta - \beta \nabla_\theta \mathcal{L}_{train}$
  // update architecture parameters
  randomly select a batch from $\mathcal{D}_{valid}$
  forward on $\mathcal{D}_{valid}$ to get $\mathcal{L}_{valid}$
  $a' = a - \gamma \nabla_a \mathcal{L}_{valid}$;
until *stopping criteria is met*
get the optimal architecture p^* and s^*
// fix the architecture
$\mathcal{D}'_{train} \longleftarrow \mathcal{D}_{train} \bigcup \mathcal{D}_{valid}$
initialize all trainable parameters
repeat
  construct the network architecture
  randomly select a batch from $\mathcal{D}'_{train}$
  forward-backward on $\mathcal{L}'_{train}$ by $\mathcal{D}'_{train}$
until *stopping criteria is met*
output the AutoST model

$$c_l^* = \underset{c \in \mathcal{S}_c}{\operatorname{argmax}}\{a_l^c\} \quad s_l^* = \bigcup_{i=0}^{l-1} \underset{s \in \mathcal{S}_s}{\operatorname{argmax}}\{a_l^s\} \tag{9}$$

Where c_l^* and s_l^* are the optimal convolution and connection operations at layer l respectively. At the training stage, we select the optimal architecture to training the network.

DataSets and Experimental Results

Table 2: Datasets.

Dataset	Time spans	Grid size	# Ins
TaxiBJ	7/1/2013-10/30/2013 3/1/2014-6/30/2014 3/1/2015-6/30/2015 11/1/2015-4/10/2016	(32,32)	15072
CrowdBJ	9/1/2017-11/30/2017	(32, 32)	2016
TaxiJN	9/1/2017-1/31/2018	(32,16)	3323
TaxiGY	10/1/2018-5/26/2019	(20, 24)	5270

Table 3: Performance comparison of different methods on three datasets

Models	CrowdBJ		TaxiJN		TaxiGY	
	RMSE	MAPE	RMSE	MAPE	RMAE	MAPE
ST-ResNet	92.27 ± 4.42	74.24% ± 4.53%	5.876 ± 0.26	62.22% ± 0.80%	2.773 ± 0.10	56.95% ± 1.09%
ST-ResNet+	87.35 ± 4.42	63.17% ± 4.53%	5.624 ± 0.06	72.30% ± 1.92%	2.521 ± 0.07	51.69% ± 0.59%
ST-3DNet	76.13 ± 2.14	55.51% ± 1.18%	5.458 ± 0.19	58.71% ± 2.71%	2.574 ± 0.08	52.71% ± 2.27%
ST-3DNet+	62.28 ± 2.68	36.56% ± 3.49%	5.103 ± 0.04	57.11% ± 1.54%	2.488 ± 0.04	51.32% ± 0.78%
DeepSTN-ne	52.49 ± 0.37	32.17% ± 1.94%	4.664 ± 0.05	45.69% ± 0.97%	2.175 ± 0.02	50.81% ± 0.20%
DeepSTN-ne+	51.38 ± 0.61	**28.43% ± 1.98%**	4.653 ± 0.20	46.58% ± 0.65%	2.169 ± 0.03	**47.61% ± 0.06%**
DeepSTNPlus	49.76 ± 0.57	28.60% ± 2.75%	4.653 ± 0.01	54.52% ± 0.30%	2.172 ± 0.06	50.01% ± 0.71%
DeepSTNPlus+	**49.09 ± 0.61**	29.08% ± 5.80%	**4.602 ± 0.00**	**44.35% ± 0.87%**	**2.157 ± 0.01**	49.55% ± 0.87%

Table 4: Performances on TaxiBJ.

Models	Param	RMSE	MAPE
ST-ResNet	0.92M	17.51 ± 0.05	33.92% ± 0.41%
ST-ResNet+	3.38M	17.47 ± 0.05	33.52% ± 0.40%
ST-3DNet	0.54M	17.82 ± 0.36	31.04% ± 0.02%
ST-3DNet+	1.36M	17.37 ± 0.20	27.77% ± 0.02%
DeepSTN-ne	0.42M	16.09 ± 0.02	27.05% ± 0.15%
DeepSTN-ne+	1.24M	15.97 ± 0.06	27.72% ± 0.14%
DeepSTNPlus	0.44M	15.98 ± 0.05	26.52% ± 0.64%
DeepSTNPlus+	1.26M	15.88 ± 0.19	25.97% ± 0.65%

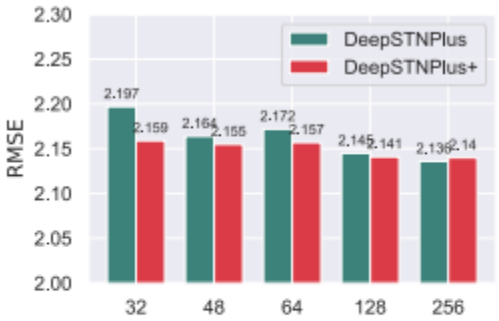
(a) Performance on TaxiBJ

(b) Performance on TaxiGY

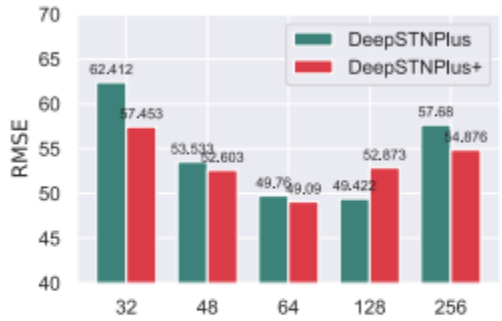
(c) Time on TaxiBJ

(d) Time on TaxiGY

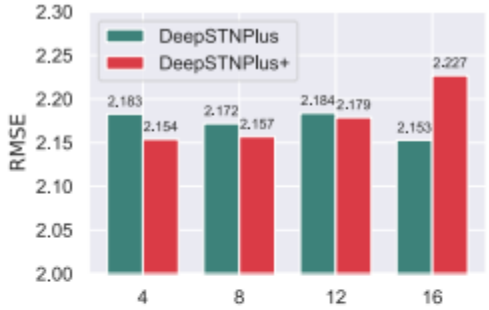
Figure 5: Computation time and performance comparison



(a) # channels on TaxiGY



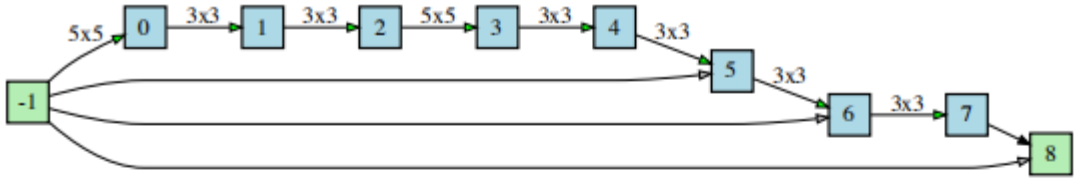
(b) # channels on CrowdBJ



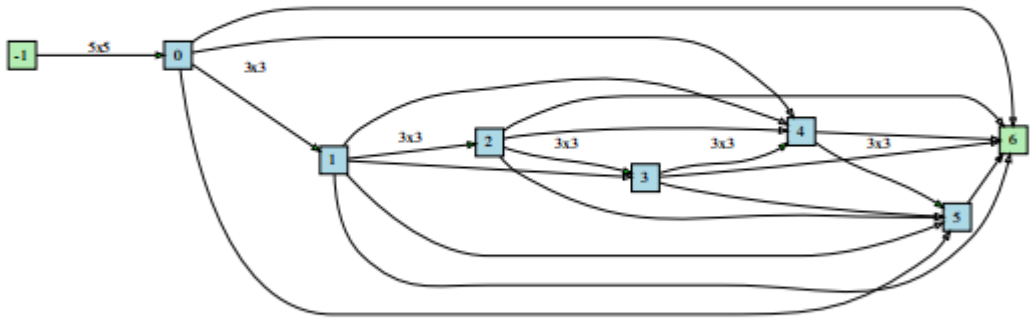
(c) # layers on TaxiGY

(d) # layers on CrowdBJ

Figure 6: Evaluation on the number of channels



(a) CrowdBJ



(b) TaxiGY

Figure 7: Architectures learned by AutoST