AutoST: Efficient Neural Architecture Search for Spatio-Temporal Prediction

Ting Li[1,2], Junbo Zhang[1,2,3*], Kainan Bao[3], Yuxuan Liang[4], Yexin Li[5], Yu Zheng[1,2,3,6*]
[1] JD Intelligent Cities Business Unit, JD Digits, Beijing, China
[2] JD Intelligent Cities Research, China
[3] Institute of Artificial Intelligence, Southwest Jiaotong University, China
[4] School of Computing, National University of Singapore, Singapore
[5] Hong Kong University of Science and Technology, Hong Kong, China
[6] School of Computer Science and Technology, Xidian University, Xi'an, China
{liting6259,baokainan123}@gmail.com,{msjunbozhang,yuxliang,msyuzheng}@outlook.com,yliby@connect.ust.hk

ABSTRACT

Spatio-temporal (ST) prediction (*e.g.* crowd flow prediction) is of great importance in a wide range of smart city applications from urban planning, intelligent transportation and public safety. Recently, many deep neural network models have been proposed to make accurate prediction. However, manually designing neural networks requires amount of expert efforts and ST domain knowledge. How to automatically construct a general neural network for diverse spatio-temporal predication tasks in cities? In this paper, we study Neural Architecture Search (NAS) for spatio-temporal prediction and propose an efficient spatio-temporal neural architecture search method, entitled AutoST. To our best knowledge, the search space is an important human prior to the success of NAS in different applications while current NAS models concentrated on optimizing search strategy in the fixed search space. Thus, we design a novel search space tailored for ST-domain which consists of two categories of components: (i) optional convolution operations at each layer to automatically extract multi-range spatio-temporal dependencies; (ii) learnable skip connections among layers to dynamically fuse low- and high-level ST-features. We conduct extensive experiments on four real-word spatio-temporal prediction tasks, including taxi flow and crowd flow, showing that the learned network architectures can significantly improve the performance of representative ST neural network models. Furthermore, our proposed efficient NAS approach searches 8-10x faster than state-of-the-art NAS approaches, demonstrating the efficiency and effectiveness of AutoST.

CCS CONCEPTS

• **Information systems → Spatial-temporal systems**; • **Computer methodologies → Automatic machine learning**.

[*]Junbo Zhang and Yu Zheng are the corresponding authors.

KEYWORDS

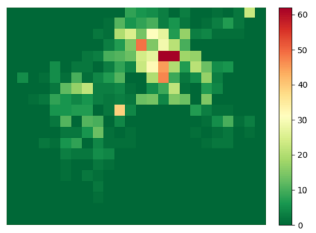
Spatio-temporal Prediction; Neural Architecture Search; AutoML

ACM Reference Format:
Ting Li, Junbo Zhang, Kainan Bao, Yuxuan Liang, Yexin Li and Yu Zheng. 2020. AutoST: Efficient Neural Architecture Search for Spatio-Temporal Prediction. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403122

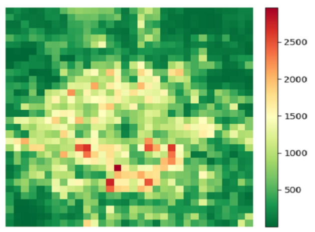
1 INTRODUCTION

Advances in location-acquisition and wireless communication technologies have resulted in massive amounts of spatio-temporal (ST) data, enabling many ST prediction tasks (*e.g.* crowd flow and traffic flow) in cities, which is critically important for smart city applications [34]. With the development of deep learning techniques, many deep spatio-temporal neural networks [6, 12, 30–32] have been proposed to improve the performance for ST prediction. Nevertheless, how to find the optimal neural architecture at at various scenarios in cities is still an unsettled problem, because ST task is usually affected by multiple complex factors: (i) the spatio-temporal correlation is complex including spatial dependency between regions and temporal correlation among timestamps; (ii) spatio-temporal correlation is diverse from location to location, for example, there is a great difference of rush hour between core city and small city; (iii) spatio-temporal correlation is heterogeneous to different tasks, for example, the local spatial correlation is important to crowd flow prediction while the global spatial correlation is significant to taxi flow prediction.

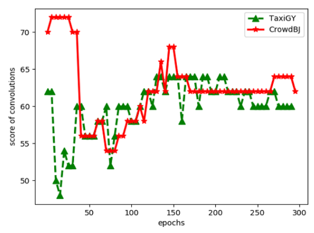
Recently, many studies have focused on designing networks to model the complex ST dependencies. For spatial correlations, authors in [20] argue that the long-distance spatial dependency is increasing important but the stack of multi-layer convolutions [32] can only capture the neighbor correlations. So they propose a *ConvPlus* component to capture the long-range spatial dependency among regions and a multi-scale fusion network to fuse multi-level features. In addition, [8] belief that information in different ranges reveals distinct traffic properties, for example, the neighborhood range indicates local dependency while a long range tends to uncover the overall pattern. Therefore, they proposed a multi-range attention network to model the diverse spatial distance dependency in graph. For temporal correlations, [13] has adopted a series of 3D convolutions network to extract the spatio-temporal features



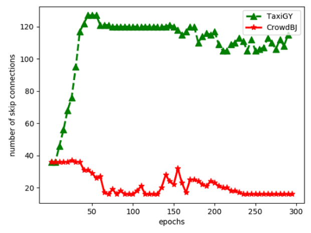
(a) TaxiGY



(b) CrowdBJ



(c) # convolutions



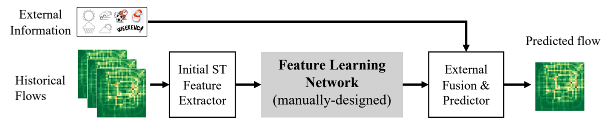
(d) # skip connections

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

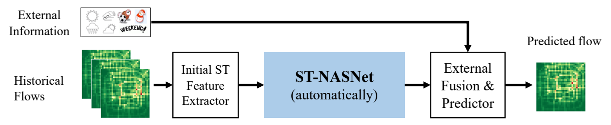
simultaneously. However, these methods mainly focus on modeling the long-range dependency for specific scenarios in Metropolis. Nevertheless, we consider that the optimal neural architectures is distinct among different cities and conclude two important aspects neglected in existing approaches.

First, *existing approaches mainly focus on modeling the long-range correlation. However, different cities may have different spatial ranges preference.* Intuitively, compared with areas with undeveloped transportation system, the core cities should take longer distance range as neighborhood information. However, the size of convolution kernel which models the range of neighborhood dependency is usually fixed and empirically set. We select the optimal architecture found in neural architecture search (NAS) and analyze the search process, which is shown in Figure 1. Figure 1(a) represents the taxi flow in Guiyang and 1(b) represents the crowd flow in Beijing where has more convenient transportation than Guiyang. Figure 1(c) shows the scores of convolutions at the search stage. Specifically, we add the weighted kernel size in all layers as the total scores for each architecture at every iteration. We can observe that the evolution tendencies of two cities have some differences and Beijing achieves slightly higher score than Guiyang.

Second, *current approaches usually use the residual network to aggregate the features in adjacent layers, failing to fuse the low- and high-level features.* As we all know, the convolution operator at deep layer tends to capture the high-level feature, and at shallow layer attempts to extract the low-level feature. Furthermore, low- and high-level features don't contribute equally in all cases. Indeed, compared with core cities, the low-level features which indicate the local information contribute more than the overall information for cities with undeveloped transportation system. Figure 1(d) show the tendency of the number of skip connections at the search stage. The more the number of skip connections indicates the more important the low-level features. We can observe that the architecture in Beijing contains obviously less skip connections than Guiyang,


External Information
Historical Flows
Initial ST Feature Extractor
Feature Learning Network (manually-designed)
External Fusion & Predictor
Predicted flow

(a) Conventional model


External Information
Historical Flows
Initial ST Feature Extractor
ST-NASNet (automatically)
External Fusion & Predictor
Predicted flow

(b) AutoST model

Figure 2: Conventional model vs. our AutoST model

verifying our assumption that different cities have distinct architecture preferences and core cities usually pay more attention to the global spatial dependency.

To tackle the aforementioned problems, we propose a general network called AutoST, which is shown in Fig 2. We can observe that the traditional processes of ST prediction consists of three components: (i) initial ST feature extractor: constructing ST features from raw ST data [32]; (ii) feature learning network (such as the residual network in [32] or the multi-scale fusion network in [20]): learning representative multi-level ST features; (iii) external fusion and predictor: fusing the external factors with flow information and then predicting urban flows in the future. The key difference between our AutoST and the conventional method is the ST-NASNet module which can automatically build the neural architecture at different scenarios. We mainly design search space in ST-NASNet to improve the network representation ability while fixed other two components.

To the best of our knowledge, this is the first approach that generalize NAS to the ST prediction problem. Our contributions can be summarized into the following three aspects:

- We propose a novel model named AutoST for spatio-temporal prediction which introduces the neural architecture search technique to dynamically capture the various-range spatial correlations and to fuse multi-level features. In addition, AutoST is oriented to ST data rather than specific application scenarios, which can be easily applied to a series of deep models.
- We design an efficient and effective search space including two basic modules: i) a mix convolution block at each layer to capture various-range spatial correlations; ii) a mix skip connection block among layers to dynamically fuse the multi-level features. Specifically, the mix convolution block consists of multiple convolutions kernels, the larger size of which indicates the longer range correlations. Besides, the mix skip connection block includes a *connection* cell and a *do-not-connection* cell to dynamically learn to fuse the low- and high level features.
- We perform extensive experiments on four real-world spatio-temporal datasets varying from taxi flow to crowd flow, and the experimental results demonstrate that AutoST can significantly improve the spatial-temporal prediction. Furthermore, our proposed spatio-temporal NAS method is more efficient than existing NAS methods.

2 PRELIMINARIES

In this section, we briefly introduce the definitions and the spatial-temporal prediction problem statement. For brevity, the frequently used notations in this paper are presented in Table 1.

Table 1: Notations.

Notations	**Description**
$n_c \in \mathbb{R}$	the number of candidate convolutions
$n_s \in \mathbb{R}$	the number of skip connections
$\theta_l^c \in \mathbb{R}$	the convolution parameters at layer l
$a_l^c \in \mathbb{R}$	the architecture weight of cell c at layer l
$a_l^s \in \mathbb{R}$	the architecture weight of cell s at layer l
$\mathcal{S}_c = \{c_0, ..., c_{n_c}\}$	the set of all candidate convolutions
$\mathcal{S}_s = \{s_0, ...s_{n_s}\}$	the set of all categories of connections

Definition 1. ***Spatial-Temporal prediction [31]:*** *We partition a city into an $I * J$ grids based on longitude and latitude where a grid denotes a region. There are many types of measures in a region for different ST applications, such as the crowd flows, bike recent and return, taxi pick-ups and drop-offs. Then, the urban information at time t can be denoted as $\mathbf{X}_t \in \mathbb{R}^{C*I*J}$ where C is the number of measured values.*

Definition 2. ***External Features:*** *We denote $\mathbf{X}_e$ as the external features including the meteorological and holiday information.*

Problem Statement: Given historical observed citywide urban flows $\{\mathbf{X}_0, \mathbf{X}_1, ..., \mathbf{X}_{t-1}\}$, and external features $\mathbf{X}_e$, predict the traffic flow for all locations in the next timestamps $\mathbf{X}_t$.

3 METHODOLOGY

As shown in Figure 2, the key module in our proposed AutoST is ST-NASNet (Spatio-TemporalNeural Architecture Search Net) that is used to automatically learn spatio-temporal network architecture. In this section, we follow the same 1st-order gradient-based optimization strategy in DARTS [23] which is performed in continuous search space. First, we describe the search space of DARTS [23] which is widely used in image domain, then introduce a novel search space tailored for ST prediction, as shown in Figure 3. Second, we illustrate the exploitation of the proposed NAS network in current ST models. Finally, we elaborate the optimization process.

3.1 Spatio-Temporal Search Space

The search space of neural network can be depicted by a general DAG. Figure 3(a) shows the architecture of residual network in a DAG view, every node of which represents the output of each layer and arrow indicates the operation. In 3(a), the black arrow which connects the adjacent two layers is the standard convolution with kernel 3×3 and the brown arrow indicates the skip connection. Distinct to neural network with fixed architecture, NAS network is composed of three modules from simple to complex: (i) a candidate cell module which defines the search unit; (ii) a operation block module which perform weighted sum over all possible operations to make the search space continuous; (iii) a NAS network module which is consisted of a series of mix operations. We will illustrate these three modules in detail.

3.1.1 Candidate Cells. For citywide spatio-temporal prediction task, we usually divide a city into a grid map where each grid denotes a region and these grids make up an image. We select the candidate cells based on the following three considerations. First, since nearby regions may affect each other for ST prediction, convolution operation is important to model the local geographical correlation. In addition, convolutions with different kernel size model the spatial dependency in different ranges, so we should take various convolution kernels into consideration. Second, global correlation is also important for better prediction so that current approaches [29, 32] usually stack multiple convolution layers to capture large-scale citywide dependencies. Indeed, local correlation captured by low-level convolution and global correlation encoded with high-level convolution are both important for city-wide flow forecasting. Therefore, skip connection which fuse multi-level correlations is necessary. Finally, distinct to the common cnn networks in image domain, ST prediction task does not need the pooling operation because pooling can cause the information loss probably.

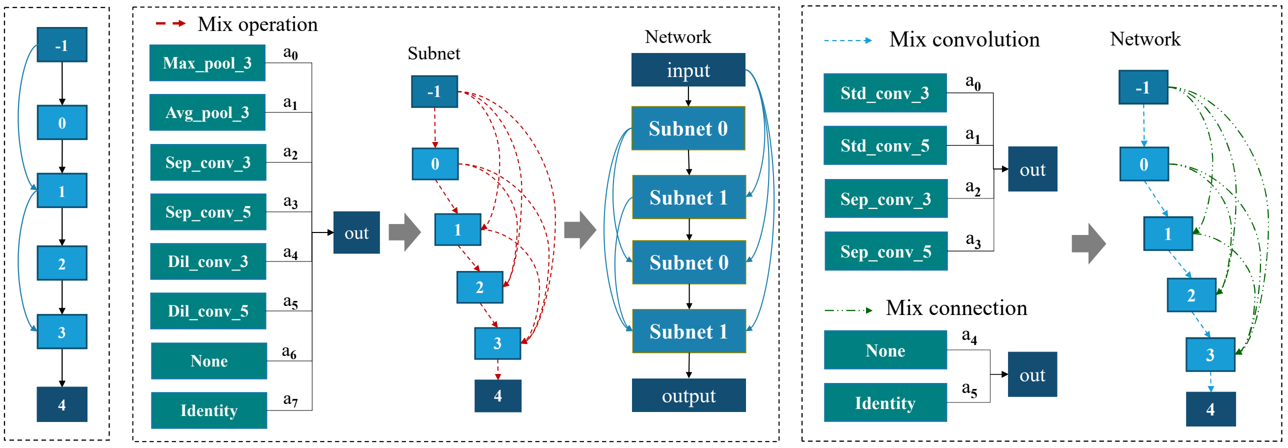
Therefore, we remove the 3×3 max pooling (Max_pool_3) and 3×3 average pooling (Avg_pool_3) operations in DARTS. In addition, we take the standard convolution into consideration. In conclusion, we select the remaining six operations as basic search cells and group them into two classes, they are: (i) convolution operations consisting of 3×3 standard convolution (Std_conv_3), 5×5 standard convolution (Std_conv_5), 3×3 separable convolution (Sep_conv_3), 5×5 separable convolution (Sep_conv_5); (ii) skip connection operation including don't connection (none) operation and connection (identity) operation. To keep the shape of output the same as input, we utilize the convolution of $stride = 1$ with SAME padding and the filter dimensions of output is the same as input. Besides, it should be noted that a convolution cell in this paper refers to a Relu-Conv-BatchNorm unit.

3.1.2 Operation Blocks. As we all known, gradient-based search strategy usually calculate the weighted sums over the outputs of all basic operations to avoid the discrete choice of basic operation unit. The illustration of operation block is shown in 3(b) and 3(c). We can observe that DARTS in 3(b) simply add the outputs of eight basic cells as the the final output of one operation block, which can be formulated as Eq.(1):

$$\bar{\mathbf{p}}_i(\mathbf{x}) = \sum_{p \in \mathcal{S}_p} \frac{exp(a_i^p)}{\sum_{p' \in \mathcal{S}_p} exp(a_i^{p'})} f(\mathbf{x}; \theta_i^p) \quad (1)$$

where $\bar{\mathbf{p}}_i$ is the mix operation at layer i and $\mathcal{S}_p$ is the set of eight candidate search cells. The f is the operation function and θ is the parameters of f. However, we argue that the calculation of all basic convolution operation at each layer may cause large memory consumption. Specifically, suppose the number of layers of inner networks is L, then there are $\frac{L*(L-1)}{2} + L$ possible operations and each operation is selected from eight basic search cells. *That's, there are $8^{L*(L-1)/2+L}$ possible network architectures in total.*

In order to solve the memory inefficient problem, we propose to group the basic cells into two categories and define two types of mix operations shown in 3(c), they are: (i) mix convolution block $\bar{c_i}$ represented as blue arrow which calculates the weighted sum of all convolution outputs; (ii) mix connection block $\bar{s_i}$ represented as



(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).

green arrow which multiples the outputs of each layers with connection probability. We define the trainable architecture weights of convolution cells $\{a_0, a_1, a_2, a_3\}$ and connection cells $\{a_4, a_5\}$ as architecture parameters $\mathcal{A}$ which controls the possibility of choosing candidate cells. Besides, every convolution cell has parameters including kernels, bias, we define them as model parameters $\mathcal{M}$. For optimization algorithms, authors in [11] observe that the number of skip connections increases when iterating more epochs, and analysis the reason that it is caused by the inherent unfair competition. Therefore, they propose to relax the choice of operations to be dependent and make each operation has an equal opportunity to develop its strength. Specifically, they apply a *sigmoid activation* instead of *softmax* to generate the architecture weights. In addition, we initialize the values in $\mathcal{A}$ with zeros instead of the random numbers. Suppose $\mathcal{S}_c$ and $\mathcal{S}_s$ are the search units for convolution operation and skip connection respectively, then the calculation of mix convolution block and mix connection block is defined as:

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

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

Where f is the convolution operation and θ is the parameters of f. The σ is the *sigmoid* activation funciton. The $\sigma(a_i^c)$ represents the weights of candidate cell c at layer i. The s is the skip connection function, $s_{i,j} = 0$ when s is none and $s_{i,j} = 1$ is $\mathbf{x}$ when s is identity. *For our model, there are L mix convolution block and* $\frac{L*(L-1)}{2}$ *mix connection block, then the number of possible architectures is* $4^L + 2^{\frac{L*(L-1)}{2}}$ *in total, which greatly reduces the search space.*

3.1.3 NAS Network. As mentioned above, taking all possible operations into consideration at each layer may cause the memory inefficient. To solve this problem, DARTS divided the network into two parts, the inner network which learns the architecture by NAS and the outer network which has the fixed architecture shown in 3(b).

However, the fixed architecture of outer network may reduce the performance. In order to make the network more efficient, we constrain the search space according to the characteristics of ST prediction in the following two aspects. First, there are only one convolution in adjacent layers to capture large-range spatial dependency. Second, there are no external feature transform when fusing multi-level features so that we simply add the outputs of previous layers to current layer. The architecture of AutoST is shown in Fig.3(c), the output of l-th layer can be formulated as:

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

where $\mathbf{o}_i$ is the output of i-th layer. The $\bar{\mathbf{c}}_l$ is to generate high-level features at layer l and $\bar{\mathbf{s}}_i$ is to fuse features in layer i and layer l.

In conclusion, the proposed network admits the following three advantages: (i) it is more effective than fixed architecture due that existing networks are a subset of AutoST; (ii) it releases experts from refining networks; (iii) it is more efficient than recent neural architecture search method because the search space is carefully designed and constrained for ST prediction task.

3.2 AutoST for Spatio-Temporal Prediction

We evaluate AutoST on three popular spatio-temporal prediction models (STResNet [32], ST-3DNet [13] and DeepSTN [20]) to verify the efficiency and generality of our algorithms. We follow the same CPT (closeness, period and trend) paradigm as [20, 32] and take DeepSTN as example to explain the AutoST for spatio-temporal prediction, the model is shown in Fig 4. Different to model in [20], the NAS model utilizes AutoST instead of multi-scale fusion network to extract features.

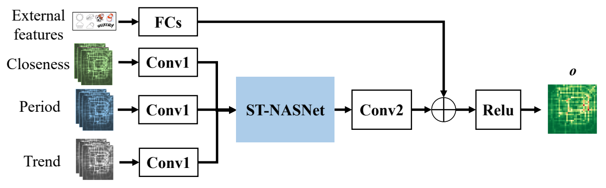


Figure 4: Illustration of AutoST for ST prediction

For ST model with L layers, the architecture parameters *architecture parameter* $\mathcal{A}$ and *model parameter* $\mathcal{M}$ can be defined as:

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

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

where $n_c = 4$ represents the the number of convolution cells and $n_s = 2$ is the number of connections. The θ_l^c is the convolution parameter at layer i. Besides, the θ_{c_1}, θ_{c_2} and θ_{fc} are parameters for Conv1, Conv2 and FCs respectively. Finally, the output of ST architecture search network in Fig.4 is:

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

Where $\mathbf{o}_L$ is the output of AutoST and $\mathbf{x}_e$ representes the external factor.

3.3 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
1  construct training set $\mathcal{D}_{train}$ and validation set $\mathcal{D}_{valid}$
   // search the architecture
2  initialize all trainable parameters
3  repeat
       // update model parameters
4      randomly select a batch from $\mathcal{D}_{train}$
5      forward on $\mathcal{D}_{train}$ to get $\mathcal{L}_{train}$
6      $\theta' = \theta - \beta\nabla_\theta\mathcal{L}_{train}$
       // update architecture parameters
7      randomly select a batch from $\mathcal{D}_{valid}$
8      forward on $\mathcal{D}_{valid}$ to get $\mathcal{L}_{valid}$
9      $a' = a - \gamma\nabla_a\mathcal{L}_{valid}$;
10 until stopping criteria is met
11 get the optimal architecture $p^*$ and $s^*$
   // fix the architecture
12 $\mathcal{D}'_{train} \longleftarrow \mathcal{D}_{train} \bigcup \mathcal{D}_{valid}$
13 initialize all trainable parameters
14 repeat
15     construct the network architecture
16     randomly select a batch from $\mathcal{D}'_{train}$
17     forward-backward on $\mathcal{L}'_{train}$ by $\mathcal{D}'_{train}$
18 until stopping criteria is met
19 output the AutoST model
```

The training of NAS contains two stages: search stage and train stage. Algorithm 1 outlines the searching process of AutoST. At the search stage, we first split the data into training set and validation set, then use the training loss $\mathcal{L}_{train}$ to optimize the θ which denotes the trainable parameters in common neural networks and use the validation loss $\mathcal{L}_{valid}$ to optimize the *architecture parameter* a. The update processes of θ and a are as following:

$$\theta' = \theta - \beta\nabla_\theta\mathcal{L}_{train}, \quad a' = a - \gamma\nabla_a\mathcal{L}_{valid} \tag{8}$$

Where β and γ are the learning rates. The optimal convolution and connection operations at each layer is calculated as following:

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

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.

4 EXPERIMENTAL RESULTS

In this section, we conduct experiments on four real-word city-wide traffic flow datasets to evaluate the network performance. Particularly, we answer the following questions:

Q1. Can AutoST be applied to a wide range of spatial-temporal prediction tasks and steadily improve performance compared with the state-of-the-art network?

Q2. Does the proposed search space be more efficient than which in image domain?

Q3. How do the settings of AutoST, i.e., the number of layers and the number of channels impact the prediction result?

4.1 Experimental Setting

4.1.1 Data Description. The brief introduction of used datasets as shown in Table 2.

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

We divide each dataset into training, validation and test sets. At the search stage, we use the validation set to learn neural architectures. At the training stage, we utilize the training set to train model and validation set to perform the early-stopping strategy. The details are as follows:

- **TaxiBJ**: This dataset was published by [32], indicating the taxi flow traveling throughout Beijing. We aim to predict the future inflows and outflows according to the historical observations. We choose data from the last month as the test set, the next last one month as validation set and all other data as training set.
- **CrowdBJ**: This dataset, which is extracted from the mobile base station, indicates the crowd flow in Beijing. We first partition Beijing into 32×32 grids and then calculate the inflows and outflow in each grid. We select the last ten days as the test data, the next last one month as the validation data and the others as the training set.

- **TaxiJN**: This dataset denotes the taxi flow in Jinan. We first partition Jinan into 32×16 grids. Then for each grid, we calculate the hourly numbers of pick-ups and drop-offs. We select the last 10 days as the testing data, the next last 2 months as validation data and others as training data.
- **TaxiGY**: Extracted from the taxicab GPS trajectory, TaxiGY represents the taxi flow in Guiyang. The studied area is split into 20×24 grids and we count the number of pick-ups and drop-offs in each grid. We choose the first five months as the training set, the next two months as the validation set, and the last month as the test set.

4.1.2 Evaluation Metrics. We measure the accuracy of our methods and baselines by Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \quad \mathrm{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \qquad (10)$$

where n is the number of values, y_i is the ground truth, and $\hat{y}_i$ is the prediction value. When calculating the MAPE loss, we remove the samples with $y_i = 0$.

4.1.3 Baseline Algorithms. We first compare AutoST with the state-of-the-art methods for urban prediction as follows:

- **ST-ResNet [32]**: It follows the CPT paradigm and adopts the late-fusion strategy to build the network. Specifically, it first employs three residual network to model the spatial dependency in closeness, period and trend respectively and then fuses the outputs of these three parts as the final output.
- **ST-3DNet [13]**: It first exploits 3D convolution to capture the correlation of traffic data in both spatial and temporal dimensions. Different from the CPT paradigm, it only considers two properties including closeness and trend. In addition, it utilizes a residual network to capture the spatial dependency in closeness and one 3D convolution layer to model the that in trend.
- **DeepSTNPlus [20]**: It reduces the architecture redundancy by exploiting the early-fusion at the beginning of whole model to integrate the closeness, period and trend features, and then use a multi-scale fusion network at the end of model to fuse multi-level features, which shows state-of-the-art performance on citywide flow prediction.

Second, we compare our proposed AutoST with the widely used NAS algorithms including:

- **ENAS [15]**: It takes the reinforcement learning as search strategy and accelerates search procedure by sharing parameters, showing state-of-the-art performance on image domain.
- **DARTS [23]**: It is the first method to transforms discrete and non-differentiable search space to continuous search space so that more efficient search is allowed with gradient-based optimization strategy.

In addition, DeepSTN-ne is a variant of DeepSTNPlus with no external subnet, and ST-ResNet+, ST-3DNet+, DeepSTN-ne+, DeepSTNPlus+ represent the AutoST enhanced algorithms for ST-ResNet, ST-3DNet, DeepSTN-ne and DeepSTNPlus respectively.

4.1.4 Hyperparameters. There are three types of hyperparameters in STResNet: (i) data features including the channel of closeness, period and trend; (ii) model hyperparameters consist of the number of filters d and layers l; (iii) trainer including learning rate, weight decay, number of training epochs. For a fair comparison, we first run a grid search on d ranging in [16, 32, 64, 128, 256] and l ranges in [4, 8, 12, 16] on feature extraction network, then choose the best parameter settings. For data features, we set the channel of closeness to 6, trend to 2 for ST-3DNet and the channel of closeness to 3, period to 1, trend to 1 for other methods. In addition, we set the number of epochs as 300, both the model optimizer and architecture optimizer to Adam. Besides, we run a grid search on the learning rate ranging from $[2e^{-3}, 1e^{-3}, 2e^{-4}, 1e^{-4}]$ with the weight decay fixed to $3e^{-4}$, learning rate of architecture learner to $2e^{-4}$.

4.2 Overall Performances

4.2.1 Performance Comparison (Q1). We first give the performance comparisons with three baseline models with fixed architecture on three datasets, as shown in Table 3. For all deep models, we train and test all methods ten times, and show the results as the format "mean ± standard deviation". We have the following three observations. First, DeepSTNPlus performs best among all three fixed networks. Compared with ST-ResNet, it exploits effective early-fusion strategy which not only significantly reduces the structure redundancy but also greatly improves the feature extraction ability. Besides, ST-3DNet performs better than ST-ResNet by 17.5%, 7.1%, 7.2% on three datasets. The reason is that ST-3DNet uses 3D convolutions to model ST dependencies jointly which can capture longer temporal correlations. Second, the proposed network performances better than experts designed architectures. Specifically, AutoST has enhanced DeepSTNPlus by 1.3%, 1.1%, 0.6% due to that the proposed network can decide whether to fuse the outputs of previous layers automatically which is more effective the fixed multi-level fusion mechanism. Besides, AutoST improves the performance of ST-3DNet and ST-ResNet more obviously than DeepSTNPlus, because they both utilize the residual network to capture the high-level spatial correlation ignoring the fusion of different level features. Finally, AutoST has enhanced backbone network by 2.1%, 0.24%, 0.28% and external factors contribute the performance improvement slightly.

In addition, we show the occupied memory and performances on TaxiBJ dataset in Table 4. We can observe that AutoST enhances existing architectures consistently on TaxiBJ dataset. From the memory consumption perspective, it is obviously that DeepSTNPlus occupied the least memory among all baselines. The reason is that DeepSTNPlus constructs only one multi-level fusion network for closeness, period and trend attributes while ST-ResNet exploits late-fuison mechanism which requires three feature extraction networks. Besides, the NAS enhanced models consume about three times as much as fixed architectures due to that the gradient-based search strategy needs to perform convolution operation on all candidate cells at every layer which largely increases the memory consumption.

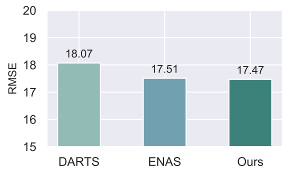
4.2.2 Complexity Comparison (Q2). Figure 5 shows the computation time and performance comparisons between ST-ResNet+ and existing NAS models. We select ST-ResNet as backbone to study the performance of three categories of NSA (including DARTS, ENAS and AutoST) enhance models and only take the results on TaxiBJ

Table 3: Performance comparison of different methods on three datasets

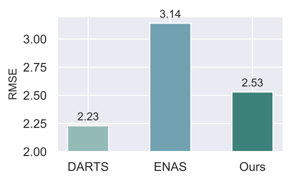
Models	CrowdBJ		TaxiJN		TaxiGY	
	RMSE	MAPE	RMSE	MAPE	RMAE	MAPE
ST-ResNet	92.27 ± 4.42	74.24% ± 4.53%	5.876 ± 0.26	62.22% ± 0.80%	2.773 ± 0.10	56.95% ± 1.09%
ST-ResNet+	87.35 ± 4.42	63.17% ± 4.53%	5.624 ± 0.06	72.30% ± 1.92%	2.521 ± 0.07	51.69% ± 0.59%
ST-3DNet	76.13 ± 2.14	55.51% ± 1.18%	5.458 ± 0.19	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.

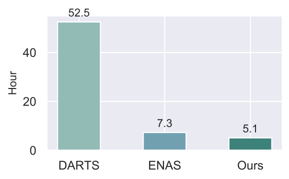
Models	Param	RMSE	MAPE
ST-ResNet	0.92M	17.51 ± 0.05	33.92% ± 0.41%
ST-ResNet+	3.38M	17.47 ± 0.05	33.52% ± 0.40%
ST-3DNet	0.54M	17.82 ± 0.36	31.04% ± 0.02%
ST-3DNet+	1.36M	17.37 ± 0.20	27.77% ± 0.02%
DeepSTN-ne	0.42M	16.09 ± 0.02	27.05% ± 0.15%
DeepSTN-ne+	1.24M	15.97 ± 0.06	27.72% ± 0.14%
DeepSTNPlus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



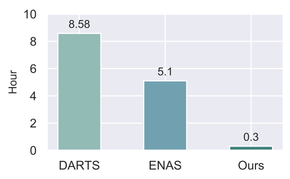
(a) Performance on TaxiBJ



(b) Performance on TaxiGY



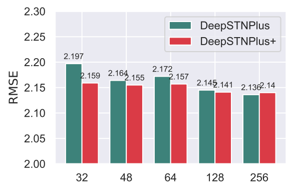
(c) Time on TaxiBJ



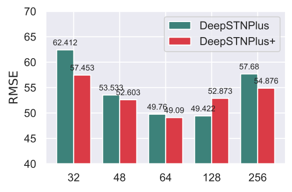
(d) Time on TaxiGY

Figure 5: Computation time and performance comparison

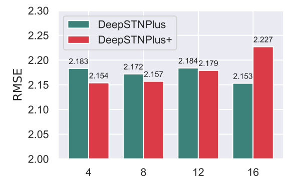
and TaxiGY dataset as two examples to illustrate the superiority of AutoST because the time consumption of DARTS is too large to be evaluated at multiple scenes. We can observe that our method costs less time to find the optimal architecture without loss accuracy. More specifically, from the efficiency perspective, DARTS consumes almost 10.29 longer times than AutoST on TaxiBJ and 28.6 longer times on TaxiGY. Besides, from the effectiveness view, AutoST slightly worse than DARTS on TaxiBJ. We conclude the reasons from the following two aspects. First, DARTS constrain the inner network at every layer to have two inputs and every operations should consider all candidate units, the multi-path ensemble mechanism of which improves the performance yet greatly increases the computation complexity of architecture search. In addition, the fixed architecture of outer network and the parameters-sharing among all sub networks at search stage in DARTS leads to a large performance gap between search and training stages.



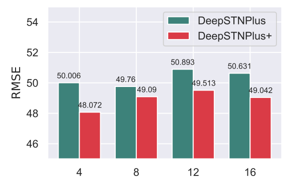
(a) # channels on TaxiGY



(b) # channels on CrowdBJ



(c) # layers on TaxiGY



(d) # layers on CrowdBJ

Figure 6: Evaluation on the number of channels

4.2.3 Parameter Sensitivity Analysis (Q3). AutoST has many settings, including the number of channels d, the number of layers l and so on. To investigate the robustness of AutoST, we compare the AutoST enhanced networks with DeepSTNPlus on d and l on CrowdBJ and TaxiGY datasets.

- To evaluate the impact of the d, we fix $l = 8$ by default and vary d in [32, 48, 64, 128, 256], the results are reported in Fig.6(a)-6(b). We can observe that TaxiGY and CrowdBJ have different optimal settings. Specifically, the prediction loss slightly decreases as d increases on TaxiGY dataset and achieve the best results when $d = 256$. Due to that the number of regions in Guiyang is less than that in Beijing as well as the spatial dependency in Guiyang is more simple. Therefore, $d = 32$ is effective enough for TaxiBJ and adding more channels doesn't improves the performance too much. For the results in Beijing, we can observe that the performances of both DeepSTNPlus and our proposed models first improved significantly and achieve the best when $d = 32$, then slightly reduced with more channels. The reason is that the spatial correlations of Beijing is complex, so the model is underfitting when $d = 32$ and increasing more channels improves the representation ability of both models significantly. In addition, we can observe the AutoST enhanced model outperforms DeepSTNPlus consistently on TaxiGY dataset yet our model performs slightly worse than DeepSTNPlus when d is larger than 128 due to that more channels increases the model parameters exponentially which made the model hard to converge.

- To evaluate the impact of l, we fix the $d = 64$ by default and change l in ranges [4, 8, 12, 16], the comparison results are shown in Fig.6(c)-6(d). We can observe that the prediction accuracy of DeepSTNPlus increases slightly with more layers and achieve the best when $l = 16$ on TaxiGY dataset. However, the AutoST enhanced model performs best when $l = 4$ and perform worse when stacking more convolution layers. The reason is that stacking too much layers to model simple ST correlations with complex NAS algorithms leads the model hard to converge. In addition, we can also observe that our proposed model outperforms DeepSTNPlus with all settings on CrowdBJ dataset due to that the proposed AutoST is more suitable when modeling complex ST dependencies.

4.3 Case Study

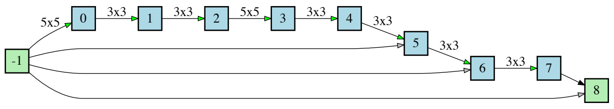
We also evaluate the architecture learned by AutoST and the results are shown in Fig.7. We can observe that the architecture searched on CrowdBJ dataset has fewer skip connections. Specifically, there is no skip connection operation in first four layers and on connections to the input in last layers. The reason is that Beijing has large-range spatial correlation which can be captured by stacking multiple convolutions. Besides, the neighborhood information is also important for accurate predictions so AutoST fuses the initial neighborhood features with high-level features. In addition, we also shows the finding architecture with $l = 6$ on TaxiGY daraset which includes a large amount of skip connections among layers as shown in 7(b). The reason is that the traffic conditions in Guiyang is not as developed as that in Beijing and the short-range neighborhood dependency contributes more than global features.

5 RELATED WORK

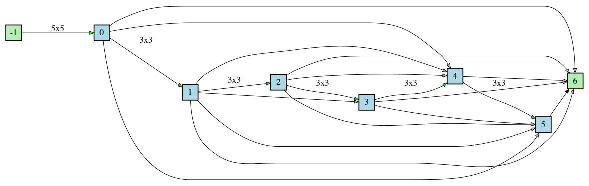
5.1 Spatial-Temporal Prediction

Spatio-temporal data are ubiquitous in the physical world, such as the traffic flow and the regional rainfall. Accurately predicting the future dynamic of them from previous observations is very essential to a wide range of real-world applications like traffic management and weather forecasts [26].

Recently, deep learning has been successfully applied to various scenarios in the ST area. For example, the architectures of (CNNs) were widely used in grid-based data modeling. Typically, [18, 19, 29, 31, 32] aim to design specific neural network structures



(a) CrowdBJ



(b) TaxiGY

Figure 7: Architectures learned by AutoST

for modeling or predicting crowd flow as well as taxi demands. However, most of them are predicting the citywide traffic flow based on multi-view [29] or multi-task [33] which need to incorporate large number of expert knowledge. In addition, some researchers tries to formulate the ST prediction problem on graphs and build the model with graph convolution network [6, 12]. However, the data quality of cities has large difference that some cities release multiple yeas of data while others only release a few days of data. To tackle this problem, transfer learning and meta learning [25, 28] are often utilized for more accurate prediction.

Moreover, Recurrent Neural Nets (RNNs) became popular due to their success in sequence learning. However, existing RNN models (such as LSTM [16] and GRU [9]) have only considered the sequence information but discard the spatial correlation. To tackle this issue, there were several studies that were also motivated by RNNs like video prediction [27]. Very recent studies [17] have indicated that the attention mechanism enables RNNs to capture dynamic spatio-temporal correlation in geo-sensory data.

Different from previous studies which design complex network based on domain knowledge, we aim to automatically learn neural architecture for different data to improve the generalization ability of model and release human out of designing networks.

5.2 Network Architecture Search

Current employed architectures in ST prediction tasks have been designed manually by human experts, which is time-consuming. Because of this, there is increasing interest in automated neural architecture search method [36]. In previous years, NAS usually adopts the bayesian optimization [2] which first achieves the competitive performance against human experts on CIFAR-10 and Penn Treebank. However, bayesian optimization costs vast computional resources (800 GPU for three or four weeks) [2]. To solve these problem, several approaches for accelerating NAS has been proposed [3–5, 14, 15, 22, 24]. In addition, [15] formulates NAS as a reinforcement learning problem and accelerate the search by sharing parameters which speeds up NAS by more than 1000x. The authors in [23] observe that either evolution or reinforcement learning treats the architecture search as a black-optimization problem over a discrete search space, so they propose to relax the discrete search space to continuous space which allows more efficient search of the architecture using gradient descent.

However, researchers in [11] argue that DARTS suffers from the unfair competition so that the number of skip connections increases when iterating more epochs. To solve the unfair competition, they propose to utilize sigmoid function instead of softmax to make the candidate cells dependent to each other. Moreover, the gradient-based search strategy has ineffective problem that there is a large performance gap between search and train stages. To solve this problem, many studies like [7, 14] propose to binary the operations so as to compress the model. The low source and time consumption inspire researchers to apply NAS [1, 10, 21, 35] into more domains.

Different from most of existing methods which optimizes the search strategy under fixed search space in image domain, we make the first try to design an effective search space tailored for ST prediction task.

6 CONCLUSIONS

In this paper, we study the problem of spaito-temporal prediction using neural architecture search method. We propose a novel NAS network named AutoST with an efficient search space tailored for spatio-temporal prediction task which can be generalized to multiple different scenes. Specifically, AutoST includes a optional convolution block composed of multi-scale kernels to capture different ranges of features at variable scales and a trainable connection block to dynamically fuse multi-scale spatial features. The proposed AutoST can automatically search the architecture which can handle the multi-range and multi-scale problems in prediction. In addition, AutoST is efficient and insensitive to different scenes. We evaluate our AutoST on four real-word datasets varying from crowd to taxi flow prediction, the performances of which are better than fixed architectures and more efficient than other search methods. The results demonstrate the efficiency and effectiveness of AutoST.

ACKNOWLEDGEMENT

This work was supported by the National Key R&D Program of China (2019YFB2101805) and Beijing Academy of Artificial Intelligence (BAAI).

REFERENCES

[1] J. An, H. Xiong, J. Huan, and J. Luo. Ultrafast photorealistic style transfer via neural architecture search. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*, 2020.
[2] Z. Barret and V. L. Quoc. Neural architecture search with reinforcement learning. In *In International Conference on Learning Representations (ICLR-17)*, 2017.
[3] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le. Understanding and simplifying one-shot architecture search. In *Thirty-fifth International Conference on Machine Learning (ICML-2018)*, 2018.
[4] A. Brock, T. Lim, . J. Ritchie, and N. Weston. Smash: One-shot model architecture search through hypernetworks. In *arXiv:1711.00536*, 2017.
[5] H. Cai, L. Zhu, and S. Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *Proceedings of the Seventh International Conference on Learning Representations (ICLR-2019)*, 2019.
[6] C. Chen, K. Li, S. G. Teo, X. Zou, k. Wang, j. Wang, and Z. Zeng. Gated residual recurrent graph neural networks for traffic prediction. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, 2019.
[7] H. Chen, L. Zhuo, B. Zhang, X. Zheng, J. Liu, D. Doermann, and R. Ji. Binarized neural architecture search. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*, 2020.
[8] W. Chen, L. Chen, Y. Xie, W. Cao, Y. Gao, and X. Feng. Multi-range attentive bicomponent graph convolutional network for traffic forecasting. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*, 2020.
[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
[10] X. Chu, B. Zhang, H. Ma, R. Xu, J. Li, and Q. Li. Fast, accurate and lightweight super-resolution with neural architecture search. In *arXiv:1901.07261*, 2019.
[11] X. Chu, T. Zhou, B. Zhang, and J. Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *arXiv preprint arXiv:1911.12126*, 2019.
[12] x. Geng, Y. Li, L. Wang, L. Zhang, q. Yang, j. Ye, and Y. Liu. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, 2019.
[13] S. Guo, Y. Lin, S. Li, Z. Chen, and H. Wan. Deep spatial–temporal 3d convolutional neural networks for traffic data forecasting. *IEEE Transaction on intelligent transportation system (TITS-2019)*, 2019.
[14] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*, 2019.
[15] P. Hieu, Y. Melody, Z. Barret, V. L. Quoc, and D. Jeff. Efficient neural architecture search via parameter sharing. In *In International Conference on Learning Representations (ICLR-18)*, 2018.
[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
[17] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng. Geoman: Multi-level attention networks for geo-sensory time series prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3428–3434. IJCAI, 7 2018.
[18] Y. Liang, K. Ouyang, L. Jing, S. Ruan, Y. Liu, J. Zhang, D. S. Rosenblum, and Y. Zheng. Urbanfm: Inferring fine-grained urban flows. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-19)*, 2019.
[19] Y. Liang, K. Ouyang, J. Zhang, Y. Zheng, and D. S. Rosenblum. Revisiting convolutional neural networks for urban flow analytics. *arXiv:2003.00895*, 2020.
[20] Z. Lin, J. Feng, Z. Lu, Y. Li, and D. Jin. Deepstn+: Context-aware spatial temporal neural network for crowd flow prediction in metropolis. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, 2019.
[21] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. Yuille, and F. Li. Hierarchical neural architecture search for semantic image segmentation. In *arXiv:1901.02985*, 2019.
[22] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. In *Proceedings of the Sixth International Conference on Learning Representations (ICLR-2018)*, 2018.
[23] H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.
[24] H. Lu, J. Langford, R. Caruana, S. Mukherjee, E. Horvitz, and D. Dey. Efficient forward architecture search. In *arXiv:1905.13360*, 2019.
[25] z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-19)*, 2019.
[26] X. Shi and D.-Y. Yeung. Machine learning for spatiotemporal sequence forecasting: A survey. *arXiv preprint arXiv:1808.06865*, 2018.
[27] Y. Wang, M. Long, J. Wang, Z. Gao, and S. Y. Philip. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In *Advances in Neural Information Processing Systems*, pages 879–888, 2017.
[28] H. Yao, Y. Liu, Y. Wei, X. Tang, and Z. Li. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *Proceedings of the Web Conference (WWW-2019)*, 2019.
[29] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and L. Zhenhui. Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.
[30] B. Yu, H. Yin, and Z. Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the Twenty-sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017.
[31] J. Zhang, Y. Zheng, Q. Dekang, R. Li, and X. Yi. Dnn-based prediction model for spatial-temporal data. In *SIGSPATIAL*, 2016.
[32] J. Zhang, Y. Zheng, and D. Qi. Deep spatio-temporal residual networks for city-wide crowd flows prediction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 1655–1661, 2017.
[33] J. Zhang, Y. Zheng, J. Sun, and D. Qi. Flow prediction in spatio-temporal networks based on multitask deep learning. *IEEE Transactions on Knowledge and Data Engineering (TKDE-2019)*, 09 2019.
[34] Y. Zheng, L. Capra, O. Wolfson, and H. Yang. Urban computing: Concepts, methodologies, and applications. *ACM Transaction on Intelligent Systems and Technology*, October 2014.
[35] Z. Zhu, C. Liu, D. Yang, A. Yuille, and D. Xu. V-nas: Neural architecture search for volumetric medical image segmentation. In *3DV*, 2019.
[36] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *arXiv:1707.07012*, 2017.