Forecasting Fine-grained Urban Flows via Spatio-temporal Contrastive Self-Supervision

Hao Qu, Yongshun Gong*, Member, IEEE, Meng Chen, Member, IEEE, Junbo Zhang, Member, IEEE, Yu Zheng, Fellow, IEEE, and Yilong Yin*

Abstract—As a critical task of the urban traffic services, fine-grained urban flow inference (FUFI) benefits in many fields including intelligent transportation management, urban planning, public safety. FUFI is a technique that focuses on inferring fine-grained urban flows depending solely on observed coarse-grained data. However, existing methods always require massive learnable parameters and the complex network structures. To reduce these defects, we formulate a contrastive self-supervision method to predict fine-grained urban flows taking into account all correlated spatial and temporal contrastive patterns. Through several well-designed self-supervised tasks, uncomplicated networks have a strong ability to capture high-level representations from flow data. Then, a fine-tuning network combining with three pre-training encoder networks is proposed. We conduct experiments to evaluate our model and compare with other state-of-the-art methods by using two real-world datasets. All the empirical results not only show the superiority of our model against other comparative models, but also demonstrate its effectiveness in the resource-limited environment.

Index Terms—Fine-grained urban flow inference, Contrastive self-supervision, spatio-temporal data

1 INTRODUCTION

With the developing trend of urbanization, intelligent transportation system has become one of the crucial components in the realm of smart cities [1], [2]. A critical requirement from urban planners and administrators is to monitor fine-grained urban flows, along with warnings in case of traffic congestion, public risk, etc [2]–[5]. For example, streamed people caused a chaotic crowd stampede at the Falls Festival in Shanghai, leaving up to 36 people died and 80 people injured in a catastrophic stampede [6]. Urban Managers can locate high-risk regions and prevent people from such real tragedies by utilizing emergency mechanisms based on the fine-grained crowd warning and prediction model. Furthermore, with the telecommunication construction from 4G to 5G, the distance between base stations gradually decreases [7]. Fine-grained inference tasks, such as fine-grained urban flow prediction can provide a more accurate guidance to set 5G base stations from the human mobility aspect.

However, forecasting fine-granularity urban flows signify that large numbers of monitoring equipment (e.g., mobile devices, surveillance cameras and piezoelectric sensors) have to be developed over the city [8]–[10]. Despite thousands of sensing devices bring convenience to the public, they consume huge amounts of power resources. For example, authorities get costly in operating ubiquitous monitoring equipment in terms of the procurement, manpower and maintenance fees, which increases the financial pressure of the government [11], [12].

To address such problems, fine-grained urban flow inference (FUFI) is proposed recently, which focuses on estimating fine-grained flows depending solely on observed coarse-grained data [3]–[5]. Figure 1 gives an example of this process. Figure 1 (a) and (b) illustrate the same city area but with two different division scales, the left sub-figure is the coarse-granularity map (32 × 32) and the right one represents the fine-granularity map (64 × 64). The goal of FUFI is to make an accurate prediction for the fine-grained flow map from the coarse flow data. Intuitively, FUFI is also recognized as a variant of image super-resolution but has its unique structural constraint, i.e., the sum of the flow volumes in fine-grained regions strictly equals that of the corresponding super-region.

Despite achievements in FUFI problem [3]–[5], most of them require a complex neural network architecture, a huge number of parameters, and a long-term training period. To present, contrastive self-supervision is an effective method to handle such issues, which has been well-performed in the field of computer vision [13]–[16] and natural language process [17], [18]. These models have shown a strong representation learning ability in a large amount of unlabeled data or fewer labeled data. To the best our knowledge, existing contrastive self-supervised learning strategies cannot be utilized in the FUFI problem directly. In reality, this work faces several specific challenges when we formulate the problem:

- Spatial Contrastive Self-supervision. Essentially, the flows of a region are mainly affected by the surrounding regions. However, two regions can have similar flows when...
Fig. 1: Traffic flow of two different granularities in Beijing, where each grid denotes a region. Fine-grained urban flow inference aims to infer from (a) Coarse-grained crowd flows to (b) Fine-grained crowd flows.

they fall into the same functional area (e.g., business center, residential area, and tourist area) [19], [20]. As shown in Figure 1 (a), even there is a long distance between regions A and B, they have a similar flow property. Previous FUFI studies focus mainly on neighboring correlations, while ignoring semantic similarities. Moreover, existing contrastive self-supervision usually uses the entire flow maps to set a comparison pair, but neglects the comparisons at the regional level. Therefore, how to devise an effective spatial contrastive learning method is a principal challenge that needs to be resolved.

- **Temporal Contrastive Self-supervision.** Existing FUFI methods aim to predict one fine-grained flow map from a snapshot of the coarse-grained flow map at the current moment. This one-to-one approach does not make effective use of temporal information from the self-supervision perspective. The prediction of fine-grained urban flow is not only inferred from the current timestamp but also affected by previous conditions. Besides, the overall traffic flow changes in an area have strong periodic characteristics, which indicates that both sequential neighborhoods and semantic similarity points contribute to the flow inference. Failure to use this information will lead to a poor performance.

- **External Factors.** External factors also play a crucial role in FUFI [4], [5]. For example, during peak hours of commuting traffic, the traffic flow of arterial roads is greater than other time periods. When severe weather occurs, people tend to be indoors rather than outdoors. Various external factors have different effects on the real-world fine-grained flow inference.

To address all challenges well, we propose a spatio-temporal contrastive self-supervision method for the FUFI, named as UrbanSTC. UrbanSTC contains three self-supervised pretext tasks: regional contrast, spatial super-resolution inference and temporal contrast. Regional contrast focuses on exploring similarities among regional-level flows based on the intrinsic spatial characteristics. Spatial super-resolution inference is an inference network that learns the spatial and upscaling patterns in the super-resolution process. Given the triplet sets of flow maps, the temporal contrast task bridges the distances between all positive pairs, while requires all negative pairs far away from each other. Finally, a fine-tuning network combining with three pre-training encoder networks is devised to make the fine-grained flow prediction. Differing from UrbanFM [3] and FODE [4], the proposed UrbanSTC achieves a significant performance improvement with a light architecture. The main contributions and innovations of this paper are summarized as follows:

- We propose a general framework of spatio-temporal contrastive self-supervision for the FUFI problem. We design two pretext tasks from the spatial aspect, i.e., the regional contrast and the spatial super-resolution inference. These two pretext tasks can identify the spatial underlying relationships among regions in terms of the surrounding property and semantic similarity.

- We incorporate external factors in the fine-tuning UrbanSTC network. Experimental results prove that the external influences benefit the final results because they have drawn useful information from events and weather conditions.

- We perform a collection of experiments on two types of dense and sparse real-world datasets to prove the effectiveness of our method compared with other state-of-the-art models. All evaluation results show that the proposed method UrbanSTC yields the best performance. Specifically, when the training data reduces, our model shows an outperformed prediction performance, which demonstrates that UrbanSTC has its own advantages in the absence of training data resources.

The rest of this paper is organized as follows: Section 2 includes a literature review. Section 3 formally defines our problem. The proposed method is shown in Section 4. All experimental results are shown in Section 5. Finally, conclusions are drawn in Section 6.

## 2 Related Work

In this section, we first review the current studies on the fine-grained urban flow inference (FUFI) and self-supervised learning methods. Since the FUFI problem [3] can be treated as a variant of single image super-resolution (SISR), we then introduce the SISR problem and reveal the difference between them.

### 2.1 Fine-grained Urban Flow Inference

FUFI aims at inferring fine-grained crowded flows in a city based on the coarse-grained observations, which is a variant of SISR in the traffic prediction field [21], [22]. Liang et al. [3] first propose a neural network named UrbanFM to address the FUFI problem, which mainly leverages the SR-ResNet [23] under the *structural constraint*. UrbanFM devises an $A^2$-Normalization layer, which outputs a `distributions` across every patch of M-by-M subregions of an associated superregion. Shen et al. design a weather-affected FUFI Predictor (WFRFP) model based on the super-resolution scheme [24]. WFRFP explores the relationship between the weather conditions and flow distributions, and reduces the scope of the predicting area based on the corresponding...
coarse-grained flow map. However, the proposed architecture heavily relies on empirically stacking deep neural networks. To solve this problem, Chen et al. introduces the deep neural network model from the perspective of the combination of differential equations and neural networks [25]. They regard the training and prediction of neural networks as the ordinary differential equations problem. Since the neural ordinary differential equations (NODE) is proposed, Zhou et al. find that NODE can be used as a core module to solve the SUIF problem, which proposes a more general neural ODE architecture called FODE [4]. FODE can address the numerical instability problem of the previous method without causing additional memory costs. The key idea of FODE is to incorporate an affine coupling layer in each ODE block to avoid the inaccurate gradient issue. The difference between FODE and UrbanFM is that FODE utilizes ODE block instead of the ResNet block. Despite the success of the above models, existing techniques rely on massive parameters and complex neural network architectures.

### 2.2 Self-Supervised Learning

Self-supervised learning has gained popularity because it can avoid the cost of annotating large-scale datasets. It mainly uses auxiliary tasks (pretext) to mine some specific supervised information from the large-scale unsupervised data, and trains the network through this constructed supervised information, in order to learn valuable representations for downstream tasks. According to the manifestation of self-supervision tasks, self-supervision is divided into the following three types: Context-based, Temporal-based and Contrastive-based approaches.

Early Context-based self-supervised techniques focus on common rules to generate labels, such as Jigsaw puzzle [13], Image restoration [14], Color transformation [15] and Image rotations [16]. The above-mentioned methods are applied in the field of computer vision. Besides, in the field of natural language processing, Word2vec [17] is a popular model to use sequence of sentences to construct auxiliary tasks for predicting words. Large-scale pre-training model Bert [18] uses MASK word method to construct auxiliary task. They have achieved remarkable results in many fields. Most of the methods introduced above are based on the samples’ meta information but with specific constraints between samples. One of the Temporal-based methods uses the concept of similar features in the video frame [26], [27]. The assumption is that the features of adjacent frames in the video are similar, while the video frames are far apart that are dissimilar. Self-supervised constraints are performed by constructing such similar (positive) and dissimilar (negative) samples. Another temporal-based method constructs positive and negative example features by tracking different frames of an object [28]. Recently, Contrastive-based has become a dominant component in self-supervised learning, which builds representations by encoding dissimilar or similar properties [29], [30].

While self-supervised learning shines in the field of computer vision, natural language processing, video processing, etc, there is limited study focusing on the urban flow forecasting, especially in the FUIF problem. We will explore a spatio-temporal contrastive self-supervision method to predict fine-grained urban flows.

### 2.3 Image Super-Resolution

Single image super-resolution (SISR) refers to the reconstruction of a high-resolution image with only one low-resolution observation image, combining with some prior knowledge of the target image. It is one of the basic issues related to the image processing, and has a wide range of practical needs and application scenarios, e.g., applied in the digital imaging technology [31], video coding communication technology [32] and fine-grained crowd-sourcing [33]. To date, there are three mainstream algorithms of SISR: interpolation-based, reconstruction-based and learning-based methods. In the interpolation-based method, early techniques focused on bicubic interpolation [34] and Lanczos resampling [35], which is fast but not accurate. Reconstruction-based SR methods [36]–[38] adopt sophisticated prior knowledge to solve Single image super-resolution with flexible and sharp details. However, as the scale factor increases, the performance of many reconstruction-based methods declines rapidly and usually time-consuming. Learning-based SISR methods utilize machine learning algorithms to analyze statistical relationships between the low-resolution (LR) and its corresponding high-resolution (HR) counterpart from a large quantity training dataset. Chang et al. [39] proposed the neighbor embedding method that used the similar local geometry between LR and HR to restore HR image blocks. Meanwhile, many researchers focus on combining the advantages of reconstruction-based with learning-based methods [40]–[42].

With the rapid development of deep learning in recent years, many studies have achieved great success since they do not require many human-engineered features. An end-to-end mapping method represented as CNNs between the LR

---

**TABLE 1: Symbol description.**

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X = [X_1, X_2, \ldots, X_T]$</td>
<td>The flow map contains $T$ moments</td>
</tr>
<tr>
<td>$I, J$</td>
<td>The granularity of division of latitude and longitude</td>
</tr>
<tr>
<td>$M$</td>
<td>The upscaling factor</td>
</tr>
<tr>
<td>$x_{t,s}$</td>
<td>The small region in flow</td>
</tr>
<tr>
<td>$H, W$</td>
<td>The length and width of feature maps in $H^{reg}$ and $H^{tcs}$</td>
</tr>
<tr>
<td>$C$</td>
<td>The channel number of convolution kernel</td>
</tr>
<tr>
<td>$X_{mc}, X_c, X_f$</td>
<td>The down-scaling coarse-grained flows map; The coarse-grained flows map; The fine-grained flows map;</td>
</tr>
<tr>
<td>$H^{reg}, H^{inf}, H^{tcs}$</td>
<td>The low-level hidden feature maps for regional contrast, inference net, and temporal contrast</td>
</tr>
<tr>
<td>$Z^{reg}, D^{tcs}$</td>
<td>The high-level semantic features in regional contrast and temporal contrast</td>
</tr>
<tr>
<td>$U'$</td>
<td>The flow inference high-level semantic representation</td>
</tr>
<tr>
<td>$U'$</td>
<td>The flow inference distribution map of the hidden state</td>
</tr>
<tr>
<td>$W'$</td>
<td>The weight matrix of flow inference</td>
</tr>
</tbody>
</table>
and HR images is first proposed by Dong et al. [43]. Inspired by the superior performance of CNN, various models for CNN began to be applied for SR. Among them, Shi et al. [32] proposed an efficient sub-pixel convolutional layer to recover HR images with little additional computational cost compared with the deconvolutional layer. Due to the great progress of VGG-net in image classification [44], a deep CNN was applied for SISR in [45]. However, the deep network is prone to model degradation in the training phase. Kim et al. [45] proposed a residual structure that makes the training of deeper convolutional neural networks possible, which has greatly promoted the development of SISR.

However, there is a great disparity between the FUFI and image super-resolution task, i.e., the unique structural constraint in FUFI. structural constraint requires mining changes within the data from a coarse-grained view, while single image super-resolution on natural images is more inclined to recover the lost high-frequency information.

3 Problem statement

Before clarifying our method, we firstly introduce some basic notations and then formulate the problem of FUFI. The main symbols used in this paper are summarized in Table 1.

Definition 1 (Grid Flow Maps). Given a timestamp $t$, assume that $X \in \mathbb{R}^{I \times J}$ is an urban flow map partitioned evenly into a $I \times J$ grid map at $t$, where a grid denotes a region as shown in Figure 1. Each entry $x_{i,j} \in \mathbb{R}^+$ denotes the volume of the observed flow.

Definition 2 (Superregion & Subregion). Figure 1 (a) and (b) illustrate the same city area but with two different division scales, the left sub-figure is the coarse-grained flows map $(32 \times 32)$ and the right one represents the fine-grained flows map $(64 \times 64)$. $M$ denotes the scaling factor controlling the resolution changes between the coarse- and fine-grained maps. Figure 1 represents an example when $M = 2$. We use superregion and subregions to define the larger grid and its constituent smaller regions respectively [3], [5].

Definition 3 (Structural Constraint). The sum of the flow volumes in fine-grained subregions $x_{i',j'}^c$ strictly equals that of the corresponding superregion $x_{i,j}^c$,

$$x_{i,j}^c = \sum_{i',j'} x_{i',j'}^c \quad \text{s.t. } i = [i' / M], j = [j' / M],$$

where $i = 1, 2, \ldots, I$ and $j = 1, 2, \ldots, J$.

Fine-grained Urban Flow Inference. Given a coarse-grained map $X^c \in \mathbb{R}^{I \times J}$ and the upsampling factor $M \in \mathbb{Z}^+$, the goal of this paper is to infer the fine-grained flow map $X^f \in \mathbb{R}^{IM \times MJ}$ under the structural constraint.

4 The Proposed Method

Figure 2 illustrates the flowchart of UrbanSTC. Our model is pre-trained by spatial self-supervision and temporal self-supervision, and then the pre-trained encoders are copied to the final network for fine-tuning. We propose three kinds of pretext strategies separately for the spatial and temporal self-supervision methods.

4.1 Spatial Self-Supervision

Urban flow data has typical spatial characteristics. Inspired by self-supervised learning, we provide two types of self-supervision tasks on the spatial perspective: regional contrast and spatial super-resolution inference network.

4.1.1 Regional-level Contrast Pre-training

Regional contrast self-supervision is dedicated to mining flow relationships at the regional level. At any timestamp $t$, there are many regions having similar or dissimilar flow conditions in the coarse-grained flow map $X^c$. In Figure 2, the light blue block (Reg) depicts an example for the regional-level contrastive learning. Assume the black rectangle is an anchor region $x^q$. The regions with red and blue rectangles can be treated as positive and negative samples respectively via a semantic distance with $x^q$, as expressed in Equation 2 and 3.

$$dist^\pi(x^q, x_{i,j}) = \sqrt{(x^q_i - x_{i,j})^2},$$

where $x_{i,j} \in \{\text{positive, } dist^\pi(x^q, x_{i,j}) \leq \lambda \}$ \{\text{negative, } dist^\pi(x^q, x_{i,j}) > \lambda \}$

in which $\lambda$ is a threshold for distinguishing between positive and negative samples. Because of the different semantic distances among regions, we hope to remain such properties in their high-level representations, i.e., the representation distances between $x^q$ and positive regional samples $\{x_{k_1}^c \mid k_1 = 1 \}$ are closer enough, while all negative representations $\{x_{k_2}^c \mid k_2 = 1 \}$ are moving away from $x^q$, where $K_1$ and $K_2$ are the numbers of selected positive and negative regional samples.

For a coarse-grained flow map $X^c$, we first project it into a low-level hidden feature map $H^reg \in \mathbb{R}^{HW \times C}$ by utilizing a non-linear encoder. This component of our network is named regional level encoder $Enc_{reg}(\cdot)$ which will be used in the fine-tuning process. Thereafter, $H^reg$ is normalized by a batch normalization method [46] and reshaped to $S^{reg} \in \mathbb{R}^{HW \times C}$. At last, a fully connected layer with $C$ hidden units produces high-level semantic features $Z^{reg} \in \mathbb{R}^{HW \times C}$ for the coarse-grained flow map $X^c$. Unlike some previous contrastive loss functions, such as InfoNCE contrastive loss, only select one example as the positive example strictly [30], [47], our method considers a set of regions from $Z^{reg}$ as positive samples, and put all the rest as negative samples, which is similar as the strategy in [48].

Given a coarse-grained flow map $X^c$, we can obtain its dense representation $Z^{reg}$. For each $X^c$, we will randomly select the regional anchor point, and distinguish positive and negative regional samples by calculating the Euclidean distances based on a pre-defined threshold $\lambda$. Then our contrastive loss function is expressed as:

$$\mathcal{L}_{reg} = - \log \frac{\sum_{k_1=1}^{K_1} \exp \text{sim}(x^q, z_{k_1}^c)}{\sum_{k_1=1}^{K_1} \exp \text{sim}(x^q, z_{k_1}^c) + \sum_{k_2=1}^{K_2} \exp \text{sim}(x^q, z_{k_2}^c)},$$

where $z^q, z_{k_1}^c, z_{k_2}^c \in Z^{reg}$ and $\text{sim}(u, v)$ is similarity function between two representations (e.g., inner product).
Through this method, positive regional samples should make similar representations close to each other rather than negative types of samples.

4.1.2 Spatial Super-resolution Inference Network Pre-training

Given a coarse-grained map $X^c \in \mathbb{R}_+^{I \times J}$ and upsampling factor $M \in \mathbb{Z}_+$, FUFI aims to learn a super-resolution model to infer the fine-grained flow map $X^f \in \mathbb{R}^{M \times M \times J}$ under the structural constraint. The most important learning mechanism is how to split a coarse region $x^c_{ij}$ to its $M^2$ fine-grained cells, which can be represented as $I \in \mathbb{R}_+^{I \times 1}$ to $M \in \mathbb{R}_+^{M \times M}$. To simulate this process, we design a spatial super-resolution inference network in our pre-training.

Our intention is to use a coarser granularity map to infer the pattern $I \rightarrow M$ with a pretext-task. In detail, we first get a down-scaling coarser granularity map $X^{mc} \in \mathbb{R}_+^{I \times J \times M \times M}$ based on the coarse-grained map $X^c$ and $M$, where each entry of $X^{mc}$ equals to the sum of corresponding $M^2$ flow volumes in $X^c$. Then we can construct a spatial super-resolution network inferring $X^c$ from $X^{mc}$. This pre-text task is able to capture the $I \rightarrow M$ pattern in advance, and could be benefit for improving the inference capability of surrounding flows.

For a $X^{mc}$, we first encode it by two convolutional layers with $C$ channels and $3 \times 3$ kernel size, each layer followed by Relu nonlinearity as shown in Figure 3. The two convolutional layers are taken as a feature learning network to map $X^{mc}$ to the low-level hidden feature maps $H^{mf} \in \mathbb{R}_+^{M \times M \times C}$. This component of our network is named spatial super-resolution encoder $Enc_{mf}()$ which is used later in the fine-tuning process. Then we can...
Fig. 3: Spatial Super-resolution Inference Network Pre-training. We get a down-scaling coarser granularity map (more coarse-grained) $X^{mc}$ based on the coarse-grained map $X^c$ and upscaling factor $M$. Spatial super-resolution inference network simplifies the difficulty of the task and imitates the process of inferring.

leverage the prior FUFI methods distributional upsampling to the end of their networks [3]–[5]. We also adopt $M^2$-Normalization to impose the structural constraint on the network. The final loss is computed by the pixel-wise Mean Square Error (MSE):

$$L_{inf} = \frac{1}{T} \sum_{t=1}^{T} \| X_t^c - F_{inf}(X_t^{mc}; \theta) \|^2, \quad (5)$$

where $\theta$ represents all learnable parameters in the inference network.

This inference structure and function $F_{inf}$ are similar to our final fine-tuning UrbanSTC, please refer to Section 4.4 for details.

### 4.2 Temporal Self-Supervision

Existing FUFI studies focus on inferring the fine-grained flow map based solely on its coarse-grained one, ignoring that similar flow conditions at different moments will also contribute to the inference. Here we devise a temporal-contrastive self-supervision network (TCS) to extract the similarity information in the temporal dimension. For any timestamp $t$, we get a down-scaling coarser granularity map $X_t^{mc}$ based on the coarse-grained $X_t^c$ and upscaling factor $M$. Spatial super-resolution inference network simplifies the difficulty of the task and imitates the process of inferring.

TCS constructs a self-supervised auxiliary task that narrows encoder features between the anchor example and positive samples, and keeps the negative samples far away. The TCS encoder $Enc_{t,cs} (\cdot)$ has a similar structure to the spatial super-resolution inference network. It projects coarse-grained map $X_t^c$ to the low-level hidden feature map $H_t^{cs} \in \mathbb{R}^{H \times W \times C}$. Then we adopt a batch normalization layers and the global average pooling layer. Finally, Multi-layer Perceptron (MLP) with Relu activation function is used to make nonlinearity, converting the encoder feature map $H_t^{cs}$ to the high-level semantic features $D_t^{cs} \in \mathbb{R}^C$.

As shown in Figure 2 the light yellow block (TCS), there are three kinds of samples: anchor point, positive and negative samples. Next, we will introduce how to select them.

1. $M^2$-Normalization is shown in Section 4.4, Equation 15.

#### 4.2.1 Hard Sampling

We first use a straightforward way to pick the closest and the farthest samples of anchor point as its positive and negative pair. The distances between samples are calculated by the Euclidean distance method:

$$\text{dist}^t(X^c, X^k) = \sqrt{\frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} (X_{i,j}^c - X_{i,j}^k)^2}, \quad (6)$$

where $X^c$ is the current coarse-grained flow map and $X^k$ is the flow map at other times. As shown in Figure 2, the module of Temporal self-supervision, there are three types of samples indicating by green, red and blue points. They represent the anchor point, positive samples and negative samples respectively. Hard sampling method aims to select the closest (positive) sample with the current anchor, and find the farthest one as the negative sample. Note that, time-contrastive approaches are widely used in the video processing, such as [26], [27], which only picks the positive samples within a time window, and put all the rest into the negative pool. It is because the natural analogies between adjacent frames of video data. However, the previous and next traffic snapshots are probably not the closest semantic samples of the anchor point due to the high periodicity in traffic flow prediction problems [49]. Thus we choose to calculate distances between the anchor point and all training samples.

#### 4.2.2 Weight Sampling

Considering that the hard sampling cannot fully use the correlations among all temporal samples $\{X_t^{mc}\}_{t=1}^T$, we further propose a weight sampling method in this section. In detail, we select Top-$K$ positive and negative samples with a weighted combination approach:

$$X_t^+ = \sum_{k=1}^{K} \frac{1}{\text{dist}^t_{i,k}} X_{i,k}^+, \quad \text{(7)}$$

$$X_t^- = \sum_{k=1}^{K} \frac{\text{dist}^t_{i,k}}{\sum_{j=1}^{K} \text{dist}^t_{i,j}} X_{i,k}^-, \quad \text{(8)}$$

where $\text{dist}^t_k$ denotes the Euclidean distance between the anchor point and $k$-th selected sample.

Algorithm 1 shows the detailed procedure of the weight sampling method. The results affected by these two sampling methods have been presented in Section 5.2.3.

TCS uses a triplet loss [50] to optimize the pre-trained model. Given a triplet constraint $\mathcal{I} = \langle X^c, X^+, X^- \rangle$. The triplet loss ensures that a pair of co-occurring $X_t$ (anchor) and $X_t^+$ (positive) are closer to each other in the embedding space while moving away from $X_t^-$ (negative). We define the score of this triplet as:

$$d_f(\mathcal{I}) = \| f(X_t^c) - f(X_t^+) \|^2 - \| f(X_t^c) - f(X_t^-) \|^2 + \alpha, \quad (9)$$

$$L_{TCS} = \frac{1}{T} \sum_{t=1}^{T} \max \{ d_f(\mathcal{I}), 0 \}, \quad (10)$$
Algorithm 1: Weight Sampling

```python
Input: original coarse data \{X^c\}.
Output: complete data \{X^c, X^c, X^-\}.
1 for \(x \in \{X^c_1, \ldots, X^c_T\}\) do
  2 Build Max-heap and Min-heap.
  3 for \(y \in \{X^c_1, \ldots, X^c_T\}\) do
    4 if \(x \neq y\) then
      5 Calculate the Euclidean distance \(d(i, j)\)
      6 Adjust Max-heap and Min-heap.
    7 Select Top-k positive and negative samples respectively.
  8 get \(X^+_t\) by Equation 7
  9 get \(X^-_t\) by Equation 8
```

where \(f(\cdot)\) is a non-linear mapping function that needs to be learned, and \(a\) is a positive margin parameter. Notably, the triplet constraint is more flexible to adapt to different levels of intra-class variances \([51], [52]\), which guarantees the differences between various timestamps.

4.3 External Factor Fusion

External factors (e.g., temperature, wind speed, weather and holidays) affect the flow distribution over the subregions. For example, people are more inclined to walk out of the office area on holidays. And when bad weather comes, people prefer to stay indoors instead of outdoors. Therefore, we should take such external factors into consideration.

We initialize the external factors into continuous features and categorical features. Among them, continuous features including temperature and wind speed are directly concatenated to form a vector \(e_{con}\). Categorical features include timestamps, days, holidays and weather conditions (e.g., windy, rainy). We use the method in UrbanFM \([3]\) to initializes external information. The categorical features are transformed into low-dimensional vectors by feeding into separate embedding layers, and then use concatenate operation to construct the categorical vector \(e_{cat}\). Then, we splice the two vectors \(e_{con}\) and \(e_{cat}\) to the final external embedding \((e = [e_{con}; e_{cat}]^T)\).

As shown in Figure 4, we use two layers of multilayer perception with nonlinear transformation to feed external embedding \(e\). By using nonlinear transformation, different external factors are converted into a hidden state \(X^c \in \mathbb{R}^{T \times J}\). We regard it as a bias of flow graph. In the previous sections, we only used coarse-grained views without external information for pre-training. Finally, we use the tensor addition operation \(X^c + X^c\) as the input of the model in the fine-tuning stage.

4.4 Fine-Tuning UrbanSTC

We derive three encoders when completing the above pre-training tasks, i.e., regional contrastive encoder \(\text{Enc}_{reg}(\cdot)\), spatial super-resolution inference encoder \(\text{Enc}_{srf}(\cdot)\) and TCS encoder \(\text{Enc}_{tcs}(\cdot)\). As illustrated in Figure 2, three encoders are used for fine-tuning the downstream task. First, we combine three low-level hidden feature maps by encoders. This step can be described as:

\[
H^{reg} = \text{Enc}_{reg}(X^c), \quad (11)
\]

\[
H^{srf} = \text{Enc}_{srf}(X^c), \quad (12)
\]

\[
H^{tcs} = \text{Enc}_{tcs}(X^c), \quad (13)
\]

\[
H^a = \text{Concat}(H^{reg}, H^{srf}, H^{tcs}), \quad (14)
\]

where \(\text{Concat}\) is the tensor concatenate operation. Then \(\text{Decoder}\) has a convolutional layer \((3 \times 3, C)\) with ReLU nonlinearity, which is used to decode three low-level hidden features. Besides, we adopt another convolutional layer \((3 \times 3, C \times M^2)\) and PixelShuffle layers, which rearranges features and increases sizes by the upsampling factor \(M\). At the end of PixelShuffle, we use a ReLU activation function. After the above operations, a feature \(U^f \in \mathbb{R}^{M \times M \times C}\) is obtained where the first two dimensions have been increased \(M\) times. Next, we use a \(3 \times 3\) convolution with the 1-size channel to get a fine-grained flow distribution map of the hidden state \(U^f_0 \in \mathbb{R}^{M^2 \times M \times W \times 1}\). Due to the structural constraint of FUF problem, the MSE loss cannot be used directly. Refer to the distributional upsampling in UrbanFM \([3]\) and FODE \([4]\), we choose a \(M^2\)-Normalization that makes the sum of subregions equal to their corresponding superregion, which is described as:

\[
W^f_{(i,j)} = \frac{U^f_{o(i,j)}}{\sum_{j' \in \left(\left\lfloor \frac{i}{M} \right\rfloor, \left\lfloor \frac{i}{M} \right\rfloor + 1\right) \times \left(\left\lfloor \frac{j}{M} \right\rfloor, \left\lfloor \frac{j}{M} \right\rfloor + 1\right)} U^f_{o(i',j')}, \quad (15)
\]

where \(U^f_{o(i,j)}\) is the \(i\)-th row and \(j\)-th column cell in \(U^f_0\) and \(W^f_{(i,j)} \in [0, 1]\) represents probability.

\(M^2\)-Normalization aims to learn the probability mapping from a coarse-grained view to a fine-grained view. Finally, we infer the fine-grained crowds map by \(\hat{X}^f = X^c \odot W^f\). Mean Square Error (MSE) is used as the loss function:

\[
L = \frac{1}{T} \sum_{t=1}^{T} \left\| X^f_t - \mathcal{F}(X^c_t; \theta) \right\|^2, \quad (16)
\]

where \(\mathcal{F}\) represents the UrbanSTC model and \(\theta\) represents all learnable parameters used in this model.
5 EXPERIMENTS

In this chapter, we have conducted comprehensive experiments to demonstrate the effectiveness of our method. The source code has been released at https://github.com/HaoQu59/UrbanSTC.

5.1 Experimental Settings

5.1.1 Datasets

We evaluate the performance of our model as well as baselines on two real-world urban flow datasets. The dataset statistics are shown in Table 2. In the experiments, we partition the data into non-overlapping training, validation, calibration data. It uses a stacked strategy to use multiple SRCNNs for intermediate-level downscaling, and performs further upsampling by simply stacking these SRCNNs. We evenly distribute coarse-grained maps into fine-grained maps according to the scaling factor.

- **TaxiBJ** [3], [5] This dataset is collected from Beijing taxi flows, including four different periods: P1 to P4. The time interval is 30 minutes.
- **BikeNYC** This dataset is collected from an open website that contains bike flow data in New York City from Jan 1 to Mar 31, 2019. We partition the city area into 40 x 20 grids as the coarse-grained map, and define the fine-granularity map with 80 x 40.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TaxiBJ</th>
<th>BikeNYC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time interval</td>
<td>30 minutes</td>
<td>1 hour</td>
</tr>
<tr>
<td>Coarse-grained size</td>
<td>32 x 32</td>
<td>40 x 20</td>
</tr>
<tr>
<td>Fine-grained size</td>
<td>128 x 128</td>
<td>80 x 40</td>
</tr>
<tr>
<td>Upscaling factor(M)</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Latitude range</td>
<td>39.82$\degree$N – 39.99$\degree$N</td>
<td>40.65$\degree$N – 40.81$\degree$N</td>
</tr>
<tr>
<td>Longitude range</td>
<td>116.26$\degree$E – 116.49$\degree$E</td>
<td>73.93$\degree$W – 74.01$\degree$W</td>
</tr>
</tbody>
</table>

5.1.2 Baselines

We compare the proposed method UrbanSTC with the following 13 baselines, including three types of methods, Heuristic, state-of-the-art image super-resolution and FUFI methods. All parameters of the proposed method and baselines adopt $M^2$-Normalization to obey the *structural constraint* of FUFI.

Heuristic methods:

- **Mean Partition (Mean)**: We evenly distribute coarse-grained maps into fine-grained maps according to the scaling factor.
- **Historical Average (HA)**: Predict the fine-grained sub-regions by the historical average of its corresponding super-region, and distribute flows into sub-regions based on historical split proportions.

Image super-resolution methods:


2. https://github.com/HaoQu59/UrbanSTC.

- **SRCNN** [43]: It is the first method to introduce convolutional neural networks (CNNs) into image super-resolution problems. SRCNN first uses bicubic interpolation to enlarge the low-resolution image to the target size, then fits the non-linear mapping through a three-layer convolutional network, and finally outputs the high-resolution image result.
- **ESPCN** [32]: ESPCN proposes a sub-pixel convolution method to extract features directly from low-resolution image size, and calculate an efficient method to obtain high-resolution images.
- **VDSR** [45]: It is different from the three-stage architecture of SRCNN and ESPCN. VDSR is based on the idea of residual structure and uses a resolution method of deep neural networks with a depth of up to 20.
- **SRResNet** [23]: SRResNet uses perceptual loss and adversarial loss to improve the realism of the restored picture. Perceptual loss is the feature extracted by the convolutional neural network.
- **DeepSD** [53]: DeepSD is the state-of-the-art method on statistical upscaling (i.e., super-resolution) for meteorological data. It uses a stacked strategy to use multiple SRCNNs for intermediate-level downscaling, and performs further upsampling by simply stacking these SRCNNs.
- **LapSRN** [54]: LapSRN is divided into two parts: feature extraction and image reconstruction. It uses low-resolution images directly as input to the network, and through step-by-step amplification, while reducing the amount of calculation, it also effectively improves the accuracy. And between the levels of each pyramid and within each level, parameter sharing is carried out through recursive.
- **IMDN** [55]: IMDN is a lightweight network architecture which contains distillation and selective fusion parts to address issues that excessive convolutions will limit the application of super-resolution technology in low computing power devices. They first use the distillation module to extract the hierarchical structure, and then use the contrast-based channel attention to fuse the features.
- **SCN** [56]: It is proved that modeling the scale invariance into the neural network can significantly improve the image restoration performance. Inspired by the spatial convolution of shift-invariance, “scale-wise convolution” is proposed to convolve across multiple scales for scale invariance.

FUFI methods:

- **UrbanFM** [3]: UrbanFM first proposes Fine-grained urban flow super-resolution. Its difficulty is that the sum of the flow of multiple fine-grained areas is equal to the flow of a coarse-grained area and the mutual influence between adjacent areas. UrbanFM designs stacking ResNet-based neural networks and $M^2$-Normalization layer to overcome.
- **UrbanPy** [5]: A progressive method of UrbanFM which uses a cascading model for forecasting fine-grained urban flows by decomposing the original tasks into multiple sub-tasks.
- **FODE** [4]: FODE is the state-of-the-art method in fine-grained Urban Flow Super-Resolution. Inspired by the Neural Ordinary Differential Equations (NODE) [25]. They propose FODE block replaces ResNet as the backbone.
TABLE 3: The average RMSE, MAE and MAPE on TaxiBJ dataset (P1) with different proportions of training data. The best results are bold and the second best are underlined.

<table>
<thead>
<tr>
<th>Methods</th>
<th>RMSE</th>
<th>MAE</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1(20%)</td>
<td>20.918</td>
<td>12.019</td>
<td>4.469</td>
</tr>
<tr>
<td>P1(40%)</td>
<td>20.918</td>
<td>12.019</td>
<td>4.469</td>
</tr>
<tr>
<td>P1(60%)</td>
<td>20.918</td>
<td>12.019</td>
<td>4.469</td>
</tr>
<tr>
<td>P1(80%)</td>
<td>20.918</td>
<td>12.019</td>
<td>4.469</td>
</tr>
<tr>
<td>P1(100%)</td>
<td>20.918</td>
<td>12.019</td>
<td>4.469</td>
</tr>
</tbody>
</table>

**TABLE 4: The average RMSE, MAE and MAPE on TaxiBJ dataset (P2) with different proportions of training data. The best results are bold and the second best are underlined.**

<table>
<thead>
<tr>
<th>Methods</th>
<th>RMSE</th>
<th>MAE</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2(20%)</td>
<td>26.729</td>
<td>15.350</td>
<td>5.364</td>
</tr>
<tr>
<td>P2(40%)</td>
<td>26.729</td>
<td>15.350</td>
<td>5.364</td>
</tr>
<tr>
<td>P2(60%)</td>
<td>26.729</td>
<td>15.350</td>
<td>5.364</td>
</tr>
<tr>
<td>P2(80%)</td>
<td>26.729</td>
<td>15.350</td>
<td>5.364</td>
</tr>
<tr>
<td>P2(100%)</td>
<td>26.729</td>
<td>15.350</td>
<td>5.364</td>
</tr>
</tbody>
</table>

**TABLE 5: The average RMSE, MAE and MAPE on TaxiBJ dataset (P3) with different proportions of training data. The best results are bold and the second best are underlined.**

<table>
<thead>
<tr>
<th>Methods</th>
<th>RMSE</th>
<th>MAE</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3(20%)</td>
<td>27.442</td>
<td>16.029</td>
<td>5.612</td>
</tr>
<tr>
<td>P3(40%)</td>
<td>27.442</td>
<td>16.029</td>
<td>5.612</td>
</tr>
<tr>
<td>P3(60%)</td>
<td>27.442</td>
<td>16.029</td>
<td>5.612</td>
</tr>
<tr>
<td>P3(80%)</td>
<td>27.442</td>
<td>16.029</td>
<td>5.612</td>
</tr>
<tr>
<td>P3(100%)</td>
<td>27.442</td>
<td>16.029</td>
<td>5.612</td>
</tr>
</tbody>
</table>
TABLE 6: The average RMSE, MAE and MAPE on TaxiBJ dataset (P4) with different proportions of training data. The best results are bold and the second best are underlined.

<table>
<thead>
<tr>
<th>Methods</th>
<th>P4(20%)</th>
<th>P4(40%)</th>
<th>P4(60%)</th>
<th>P4(80%)</th>
<th>P4(100%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>MAE</td>
<td>MAPE</td>
<td>RMSE</td>
<td>MAE</td>
</tr>
<tr>
<td>MEAN HA</td>
<td>4.340</td>
<td>2.077</td>
<td>0.319</td>
<td>4.383</td>
<td>2.052</td>
</tr>
<tr>
<td>SRCNN</td>
<td>4.058</td>
<td>2.369</td>
<td>0.668</td>
<td>4.065</td>
<td>2.381</td>
</tr>
<tr>
<td>ESPCN</td>
<td>3.983</td>
<td>2.290</td>
<td>0.600</td>
<td>3.865</td>
<td>2.187</td>
</tr>
<tr>
<td>DeepSD</td>
<td>3.980</td>
<td>2.240</td>
<td>0.562</td>
<td>3.910</td>
<td>2.181</td>
</tr>
<tr>
<td>VDSR</td>
<td>3.952</td>
<td>2.239</td>
<td>0.573</td>
<td>3.741</td>
<td>2.075</td>
</tr>
<tr>
<td>SRResNet</td>
<td>4.118</td>
<td>2.463</td>
<td>0.738</td>
<td>4.053</td>
<td>2.431</td>
</tr>
<tr>
<td>LapSRN</td>
<td>4.467</td>
<td>2.753</td>
<td>0.894</td>
<td>4.150</td>
<td>2.489</td>
</tr>
<tr>
<td>IMDN</td>
<td>4.100</td>
<td>2.530</td>
<td>0.818</td>
<td>3.828</td>
<td>2.301</td>
</tr>
<tr>
<td>SCNN</td>
<td>3.798</td>
<td>2.154</td>
<td>0.550</td>
<td>3.660</td>
<td>2.048</td>
</tr>
<tr>
<td>UrbanFM</td>
<td>4.054</td>
<td>2.126</td>
<td>0.373</td>
<td>3.794</td>
<td>1.969</td>
</tr>
<tr>
<td>UrbanPy</td>
<td>3.959</td>
<td>2.088</td>
<td>0.413</td>
<td>3.740</td>
<td>1.936</td>
</tr>
<tr>
<td>FODE</td>
<td>3.912</td>
<td>2.042</td>
<td>0.350</td>
<td>3.725</td>
<td>1.930</td>
</tr>
<tr>
<td>UrbanSTC</td>
<td>3.640</td>
<td>1.837</td>
<td>0.278</td>
<td>3.542</td>
<td>1.786</td>
</tr>
<tr>
<td>Δ</td>
<td>+1.16%</td>
<td>+10.44%</td>
<td>+12.85%</td>
<td>+3.22%</td>
<td>+6.94%</td>
</tr>
</tbody>
</table>

5.1.3 Evaluation Metrics

We evaluate different methods with three widely used metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE).

\[
RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (X_i - \hat{X}_i)^2}
\]

\[
MAE = \frac{1}{N} \sum_{i=1}^{N} |X_i - \hat{X}_i|
\]

\[
MAPE = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{X_i - \hat{X}_i}{X_i} \right|
\]

where \(\hat{X}_i\) is a prediction for fine-grained flow, and \(X_i\) is the ground truth; \(N\) is the number of prediction values.

5.1.4 Training Details & Hyperparameters

Our model and baselines are completely implemented by PyTorch 1.60 with a RTX 2080 GPU. The network is trained using Adam with the first and second moment estimates equaling to 0.9 and 0.999, respectively [57]. The initial learning rate is set to be 1e-3, and is divided by 2 after 50 epochs, which allows smoother search near the convergence point. The mini-batch size is 16, and the number of base channels is 128.

5.2 Results on TaxiBJ

We first assess the performances of our model and baselines on TaxiBJ with a varying ratio of training data. Table 3 - 6 report the prediction results. Note that, the variances of the results are almost in the range of 0.000 - 0.002, thus we omit the variances. We summarize the tables with several key observations:

(1) UrbanSTC outperforms all competitive methods across the entire time spans (P1-P4). By comparing to current state-of-the-art methods, UrbanSTC has improved 2.51%, 3.80% and 10.51% for RMSE, MAE and MAPE on average on TaxiBJ-P1 with 100.00% training data.

(2) It is apparent that UrbanSTC can achieve the best results when training data decreases. Taking TaxiBJ-P1 (20% training data) for example, UrbanSTC yields 7.10%, 12.24% and 10.91% relative improvements in terms of RMSE, MAE and MAPE, respectively.

The above results show that UrbanSTC has its own advantages in the absence of training data resources. This is consistent with our motivation that spatio-temporal contrastive self-supervision can better learn flow feature representations and improve FUIF performance. Image super-resolution method SCN [56] performs better than other baselines with metric RMSE on 20% - 80% TaxiBJ datasets, while shows deteriorate scores on MAE and MAPE. It is mainly because SCN is a state-of-the-art image super-resolution method with the root mean square loss function. However, most image super-resolution methods are not adapted to the FUIF problem since they do not consider the structural constraint when designing models. Compared with UrbanFM, UrbanPy, and FODE, spatio-temporal contrastive learning method UrbanSTC can provide the better latent representations that performs most through all experiments.

5.2.1 Ablation Analysis

To analyses the contribution of each component of UrbanSTC, we analyze the ablation study in this section. We only report the evaluation metrics on TaxiBJ dataset (average result of P1 to P4) because the experimental results are consistent with our motivation that spatio-temporal contrastive self-supervision can better learn flow feature representations and improve FUIF performance. Image super-resolution method SCN [56] performs better than other baselines with metric RMSE on 20% - 80% TaxiBJ datasets, while shows deteriorate scores on MAE and MAPE. It is mainly because SCN is a state-of-the-art image super-resolution method with the root mean square loss function. However, most image super-resolution methods are not adapted to the FUIF problem since they do not consider the structural constraint when designing models. Compared with UrbanFM, UrbanPy, and FODE, spatio-temporal contrastive learning method UrbanSTC can provide the better latent representations that performs most through all experiments.
on BikeNYC can make similar conclusions. All the results are shown in Table 7. The term “Reg” means the regional-level contrast pre-training; “Inf” illustrates the spatial super-resolution inference network; “TCS” indicates the temporal contrast is used or not.

We can clearly see that the combination of any two components is better than the single one, which proves the effectiveness of our proposed components. When only considering one strategy, temporal contrast performs better than regional-level contrast and the spatial super-resolution inference network. Spatial contrast contains two components, “Reg” and “Inf”. We find that the effect of the spatial super-resolution network (Inf) is better than the regional-level contrast (Reg). It is mainly because the kernel of the Reg encoder is $1 \times 1$, while that of in Inf encoder is $3 \times 3$, where the larger convolution kernel size helps to capture more information in the encoder. The results of combination of “Reg” + “TCS” and “Inf” + “TCS” are slightly worse than the final model, indicating that such prior knowledge considered both spatial and temporal information is significant for the fine-grained urban flow inference.

To better present the ablation results, we draw some comparable images in Figure 5. Figure 5 shows the inference errors $\|X - \hat{X}\|_{1,1}$ generated by UrbanSTC and other ablation parts, where a brighter pixel indicates a larger error. A (West TuCheng Road) and B (Sanyuan bridge) are the main traffic arteries in Beijing. It is apparent that UrbanSTC achieves better results than other ablation experiments, which proves that the final structure of the proposed model can better capture the spatio-temporal characteristics of flow data.

### 5.2.2 End-to-end and two-stage Comparison

To verify the effectiveness of the two-stage training process and end-to-end training process, we conduct experiments in TaxiBJ (average result of P1 to P4) and BikeNYC datasets. The end-to-end model integrates three proposed modules, i.e., the coarse-grained flow map is introduced to the spatial self-supervision, temporal self-supervision and external factor learning simultaneously, and optimize these three loss functions integrally. As shown in Table 8, we can clearly find that the two-stage experimental results are better than the end-to-end training process. The end-to-end training method needs to adjust the balance factors between each loss function, while the two-stage training method is not required to adjust the balances among pretexts. The advantage of the self-supervised learning lies in two-stage training. The pretexts help the model in learning the internal characteristics of the data in advance, and the fine-tuning stage then learns the corresponding label information [30], [47], [58], [59].

### 5.2.3 Temporal Contrastive Sampling Analysis

To evaluate the effect of hard sampling and weight sampling methods, we report the experimental results on TaxiBJ-P1. The tests drawn in Figure 6 demonstrate that the weight sampling is better than the hard sampling when the proportion of used training data is lower than 60%. This is because the weight sampling method can comprehensively use top $K$ related samples, while the hard sampling only uses the most similar or dissimilar data. With the amount of training data increases, hard sampling begins to show a better performance than weight sampling. When the training dataset is small, we can hardly to pick up the global most similar sample, but use top-$K$ similar samples instead. Otherwise, if the most similar sample is found with the training data increasing, the hard sampling method can achieve the better result. Therefore, a combination of two methods can be adopted in different training scenarios.

<table>
<thead>
<tr>
<th>Methods</th>
<th>TaxiBJ</th>
<th>BikeNYC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>MAE</td>
</tr>
<tr>
<td>End-to-End</td>
<td>3.980</td>
<td>2.053</td>
</tr>
<tr>
<td>Two-stage</td>
<td>3.958</td>
<td>1.998</td>
</tr>
<tr>
<td>Δ</td>
<td>0.55%</td>
<td>2.68%</td>
</tr>
</tbody>
</table>
In this section, we try to explore the learning abilities of our method in various setting environments. Compared with different channels (32, 64, 128), we can get the results shown in Figure 8. Figure 8(a) illustrates that the larger number of channels, the better performance of UrbanSTC. Besides, Figure 8 (b), (c) and (d) show that a larger number of channels can improve the efficiency of the learning convergence.

We analyze the influence of $\lambda$ in the regional-level contrastive learning. Figure 7a shows the different performance. Note that, even some compared FUFI methods have the well-designed external information fusion module, our proposed method UrbanSTC can leverage external information with a simple network.

5.2.4 Study on External Factor Fusion

In reality, there are complicated external factors in the FUFI problem. In order to verify the effectiveness of the external information in our method, we introduce external factors and conduct experiments on TaxiBJ datasets with different time spans (P1-P4). We only compare our method with available baselines. As test shown in Table 9, we clearly see that UrbanSTC+E performs better than other models across all time spans, which reveals that the combination of our UrbanSTC and external factors can improve the model performance. Note that, even some compared FUFI methods have the well-designed external information fusion module, our proposed method UrbanSTC can leverage external information with a simple network.

5.2.5 Configurations and Parameters Analysis

In this section, we try to explore the learning abilities of our method in various setting environments. Compared with different channels (32, 64, 128), we can get the results shown in Figure 8. Figure 8(a) illustrates that the larger number of channels, the better performance of UrbanSTC. Besides, Figure 8 (b), (c) and (d) show that a larger number of channels can improve the efficiency of the learning convergence.

We analyze the influence of $\lambda$ in the regional-level contrastive learning. Figure 7a shows the different performances with a varying setting of $\lambda$. The regional-level contrast judges which regions are positive and negative...
TABLE 11: The average RMSE, MAE and MAPE on BikeNYC dataset with different proportions of training data. The best results are bold and the second best are underlined.

<table>
<thead>
<tr>
<th>Methods</th>
<th>BikeNYC(20%)</th>
<th>BikeNYC(40%)</th>
<th>BikeNYC(60%)</th>
<th>BikeNYC(80%)</th>
<th>BikeNYC(100%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>MAE</td>
<td>RMSE</td>
<td>MAE</td>
<td>RMSE</td>
</tr>
<tr>
<td>HA</td>
<td>1.498</td>
<td>0.399</td>
<td>1.476</td>
<td>0.355</td>
<td>1.511</td>
</tr>
<tr>
<td>SRCNN</td>
<td>1.419</td>
<td>0.452</td>
<td>1.306</td>
<td>0.421</td>
<td>1.228</td>
</tr>
<tr>
<td>ESPCN</td>
<td>1.458</td>
<td>0.489</td>
<td>1.432</td>
<td>0.495</td>
<td>1.302</td>
</tr>
<tr>
<td>VDSR</td>
<td>1.888</td>
<td>0.838</td>
<td>1.740</td>
<td>0.758</td>
<td>1.616</td>
</tr>
<tr>
<td>SRResNet</td>
<td>1.843</td>
<td>0.891</td>
<td>1.713</td>
<td>0.781</td>
<td>1.607</td>
</tr>
<tr>
<td>LapSRN</td>
<td>1.382</td>
<td>0.635</td>
<td>1.448</td>
<td>0.550</td>
<td>1.392</td>
</tr>
<tr>
<td>IMDN</td>
<td>1.407</td>
<td>0.521</td>
<td>1.345</td>
<td>0.456</td>
<td>1.292</td>
</tr>
<tr>
<td>SCN</td>
<td>1.331</td>
<td>0.424</td>
<td>1.276</td>
<td>0.404</td>
<td>1.191</td>
</tr>
<tr>
<td>UrbanFM</td>
<td>1.405</td>
<td>0.316</td>
<td>1.302</td>
<td>0.309</td>
<td>1.215</td>
</tr>
<tr>
<td>UrbanPy</td>
<td>1.381</td>
<td>0.315</td>
<td>1.310</td>
<td>0.301</td>
<td>1.271</td>
</tr>
<tr>
<td>FODE</td>
<td>1.293</td>
<td>0.302</td>
<td>1.214</td>
<td>0.279</td>
<td>1.167</td>
</tr>
<tr>
<td>UrbanSTC</td>
<td>1.267</td>
<td>0.276</td>
<td>1.191</td>
<td>0.257</td>
<td>1.146</td>
</tr>
<tr>
<td>∆</td>
<td>+2.01%</td>
<td>+8.61%</td>
<td>+1.89%</td>
<td>+7.89%</td>
<td>+1.80%</td>
</tr>
</tbody>
</table>

Fig. 10: Convergence rate of different methods.
provements on average in terms of RMSE and MAE, respectively. Note that due to the extremely sparsity of BikeNYC dataset, the metric MAPE is not available. It is apparent that the experimental results lead to similar conclusions to the test on TaxiBJ. The proposed model outperforms other baseline methods on both sparse and dense datasets, which has a good robustness.

5.4 Visualization of Fine-grained Flow Prediction

Figure 11 gives an intuitive presentation of the fine-grained urban flow prediction in BikeNYC data. Figure 11 (a) represents the coarse-grained crowd flows and (b) is the ground-truth of the fine-grained flow map from (a), and (c) is our prediction result. This visualization illustrates the effectiveness of our model.

Figure 12 shows the inference errors $\|X_f - \hat{X}_f\|_{1,1}$ generated by UrbanSTC and the other three baselines for a sample at the 4x task, where a brighter pixel indicates a large error. Overall, UrbanSTC has obtained more detailed inference effects and less global error. To better visualize the quality of inference, we select four busy subregions (A, B, C and D) where the UrbanSTC performs better than other methods obviously. Area A is the Sanyuan bridge (the main entrance to downtown); area B is the Beijing zoo (a large number of tourists); areas C and D cover the Beijing west railway stations. Compared with existing FUFI methods, we observe that UrbanSTC has made great improvements in the above areas. Besides, UrbanSTC shows a darker tone than other methods from the heat map, which corresponds to the quantitative results from Table 3.

6 CONCLUSION

In this paper, we propose a spatio-temporal contrastive self-supervision method named UrbanSTC for the fine-grained urban flow inference problem. Our model can extract rich spatial and temporal characteristics from urban flows. In detail, we establish self-supervision pretext tasks from two aspects, that are spatial and temporal correlations. For the spatial correlation, regional contrast and spatial super-resolution inference network make great contributions to capture similarities among regional-level flows and upscaling patterns. Moreover, we devise two sampling strategies based on temporal attributes. The overall architecture of our model obeys the self-supervised training mode: pre-training & fine-tuning. Through well-designed self-supervised tasks, uncomplicated networks have a strong ability to learn high-level representations from urban flows. We conduct intensive experiments on two real-world datasets to compare the performances between UrbanSTC and other state-of-the-art approaches. The results not only show that our approach outperforms all other methods, but also represent a high performance when the training data decrease.

REFERENCES


Yongshun Gong (S’19-M’21) is an Associate Professor at the School of Software, Shandong University, China. He received his Ph.D. degree from University of Technology Sydney in 2021. His principal research interest covers the data science and machine learning, in particular, the following areas: spatio-temporal data mining and traffic prediction. He has published above 30 papers in top journals and refereed conference proceedings, including the IEEE T-KDE, IEEE T-NNSL, IEEE T-CYB, NeurIPS, KDD, CIKM, AAAI, and UCAI.

Dr. Junbo Zhang is a Senior Researcher of JD Intelligent Cities Research. He is leading the Urban AI Product Department of JD iCity at JD Technology, as well as AI Lab of JD Intelligent Cities Research. His research interests include Spatio-Temporal Data Mining and AI, Urban Computing, Deep Learning, Federated Learning. He has published over 50 research papers (e.g., AI Journal, IEEE TKDE, KDD, AAAI, IJCAI, WWW, ACL, UbiComp) in refereed journals and conferences. He received the ACM Chengdu Doctoral Dissertation Award in 2016, the Chinese Association for Artificial Intelligence (CAAI) Excellent Doctoral Dissertation Nomination Award in 2016, the Si Shi Yang Hua Medal of SWJTU in 2012, and the Outstanding Ph.D. Graduate of Sichuan Province in 2013.

Dr. Yu Zheng is the Vice President of JD.COM and the Chief Data Scientist at JD Digits, passionate about using big data and AI technology to tackle urban challenges. Before Joining JD.COM, he was a senior research manager at Microsoft Research. Zheng is also a Chair Professor at Shanghai Jiao Tong University. He currently serves as the Editor-in-Chief of ACM Transactions on Intelligent Systems and Technology and has served as chair on over 10 prestigious international conferences. He is also a keynote speaker of AAAI 2019, KDD 2019 Plenary Keynote Panel and IJCAI 2019 Industrial Days. His monograph, entitled Urban Computing, has been used as the first textbook in this field. In 2013, he was named one of the Top Innovators under 35 by MIT Technology Review (TR35) and featured by Time Magazine for his research on urban computing. In 2017, Zheng is honored as an ACM Distinguished Scientist.

Hao Qu is now a postgraduate student in the School of Software, Shandong University, China. His principal research interest covers the data science and machine learning, in particular, the following areas: traffic analysis; spatio-temporal data mining; natural language processing. He has published 2 papers in Computer Science journal.
Yilong Yin received the Ph.D. degree from Jilin University, Changchun, China, in 2000. From 2000 to 2002, he was a Postdoctoral Fellow with the Department of Electronic Science and Engineering, Nanjing University, Nanjing, China. He is the Director of the Machine Learning and Applications Group and a Professor with Shandong University, Jinan, China. His research interests include machine learning, data mining, computational medicine, and biometrics.