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, leaved 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].

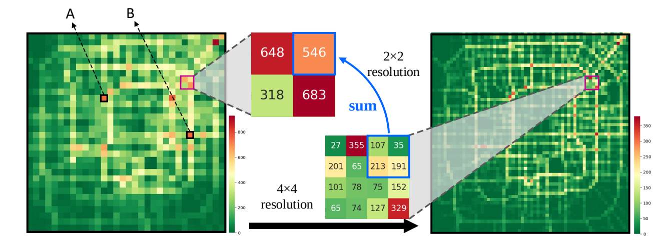
To address such problems, fine-grained urban flow inference (FUFI) is proposed recently, which focuses on estimating fine-grained flows depending solely on observed coarse-grained data [3]–[5]. Figure 1 gives an example of this process. Figure 1 (a) and (b) illustrate the same city area but with two 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

This work was supported in part by the Shandong Excellent Young Scientists Fund (Oversea) under Grant 2022HWYQ-044, in part by the Natural Science Foundation of Shandong Province under Grant ZR2021QF034, in part by the Open Fund of Key Laboratory of Urban Land Resources Monitoring and Simulation, Ministry of Natural Resources, in part by the Shandong Provincial Natural Science Foundation for Distinguished Young Scholars under Grant ZR2021JQ26, in part by the Major Basic Research Project of Natural Science Foundation of Shandong Province under Grant ZR2021ZD15, in part by the Young Scholars Program of Shandong University and in part by the Fundamental Research Funds of Shandong University.
Hao Qu, Yongshun Gong, Meng Chen and Yilong Yin are with the School of Software, Shandong University, Jinan, China. (e-mail: hao.qu@mail.sdu.edu.cn; {ysgong,mchen,ylyin}@sdu.edu.cn).
Junbo Zhang and Yu Zheng are with JD Intelligent Cities Research, China. (e-mail: {msjunbozhang,msyuzheng}@outlook.com).
Hao Qu and Yongshun Gong contributed equally to this work.
**Corresponding authors.*

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3200734



(a) Coarse-grained crowd flows (32×32) (b) Fine-grained crowd flows (64×64)

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 of 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.

• Two kinds of temporal contrast sampling methods, hard sampling and weight sampling, are proposed in this paper. The former method selects the most confident examples as the positive and negative pairs, and the latter leverages an adaptive weight strategy to rebuild positive and negative pairs of the anchor example.

• 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 crowed 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 M^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 equation problems. 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 FUFI 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 technique focuses 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-upervised 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 FUFI problem. We will explore a spatio-temporal contrastive self-supervision method to predict fine-grained urban flows.

TABLE 1: Symbol description.

Symbols	Descriptions
$\mathbf{X} = [\mathbf{X}_1^c, \mathbf{X}_2^c, \ldots, \mathbf{X}_T^c]$	The flow map contains T moments
$I; J$	The granularity of division of latitude and longitude
M	The upscaling factor
$x_{i,j}$	The small region in flow
$H; W$	The length and width of feature maps in $\mathbf{H}^{reg}$ and $\mathbf{H}^{tcs}$
C	The channel number of convolution kernel
$\mathbf{X}^{mc}; \mathbf{X}^c; \mathbf{X}^f$	The down-scaling coarse-grained flows map; The coarse-grained flows map; The fine-grained flows map;
$\mathbf{H}^{reg}; \mathbf{H}^{inf}; \mathbf{H}^{tcs}$	The low-level hidden feature maps for regional contrast, inference net, and temporal contrast
$\mathbf{Z}^{reg}; \mathbf{D}^{tcs}$	The high-level semantic features in regional contrast and temporal contrast
$\mathbf{U}^f$	The flow inference high-level semantic representation
$\mathbf{U}_o^f$	The flow inference distribution map of the hidden state
$\mathbf{W}^f$	The weight matrix of flow inference

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 crowdsourcing [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. Change 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

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3200734

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 sysmbols used in this paper are summarized in Table 1.

Definition 1 (Grid Flow Maps). Given a timestamp t, assume that $\mathbf{X} \in \mathbb{R}_+^{I\times J}$ 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 × 32) and the right one represents the fine-grained flows map (64 × 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 *supperregion* 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 $x^f_{i',j'}$ strictly equals that of the corresponding superregion $x^c_{i,j}$.

$$x^c_{i,j} = \sum_{i',j'} x^f_{i',j'} \quad \text{s.t.} \quad i = \lfloor \frac{i'}{M} \rfloor, j = \lfloor \frac{j'}{M} \rfloor, \tag{1}$$

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

Fine-grained Urban Flow Inference. Given a coarse-grained map $\mathbf{X}^c \in \mathbb{R}_+^{I\times J}$ and the upscaling factor $M \in \mathbb{Z}_+$, the goal of this paper is to infer the fine-grained flow map $\mathbf{X}^f \in \mathbb{R}_+^{MI\times MJ}$ 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 $\mathbf{X}^c$. 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 $\mathbf{x}^q$. The regions with red and blue rectangles can be treated as positive and negative samples respectively via a semantic distance with $\mathbf{x}^q$, as expressed in Equation 2 and 3.

$$dist^s(\mathbf{x}^q, \mathbf{x}_{i,j}) = \sqrt{(\mathbf{x}^q - \mathbf{x}_{i,j})^2}, \tag{2}$$

where $\mathbf{x}_{i,j}$ is a candidate area in the flow map.

$$\mathbf{x}_{i,j} \in \begin{cases} \text{positive}, & dist^s(\mathbf{x}^q, \mathbf{x}_{i,j}) \le \lambda \\ \text{negative}, & dist^s(\mathbf{x}^q, \mathbf{x}_{i,j}) > \lambda \end{cases} \tag{3}$$

in which λ 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 $\mathbf{x}^q$ and positive regional samples $\{\mathbf{x}^+_{k_1}\}_{k_1=1}^{K_1}$ are closer enough, while all negative representations $\{\mathbf{x}^-_{k_2}\}_{k_2=1}^{K_2}$ are moving away from $\mathbf{x}^q$, where K_1 and K_2 are the numbers of selected positive and negative regional samples.

For a coarse-grained flow map $\mathbf{X}^c$, we first project it into a low-level hidden feature map $\mathbf{H}^{reg} \in \mathbb{R}^{H\times W\times C}$ by utilizing a non-linear encoder. This component of our network is named regional level encoder $\mathbf{Enc}_{reg}(\cdot)$ which will be used in the fine-tuning process. Thereafter, $\mathbf{H}^{reg}$ is normalized by a batch normalization method [46] and reshaped to $\mathbf{S}^{reg} \in \mathbb{R}^{HW\times C}$. At last, a fully connected layer with C hidden units produces high-level semantic features $\mathbf{Z}^{reg} \in \mathbb{R}^{HW\times C}$ for the coarse-grained flow map $\mathbf{X}^c$. 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 $\mathbf{Z}^{reg}$ 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 $\mathbf{X}^c$, we can obtain its dense representation $\mathbf{Z}^{reg}$. For each $\mathbf{X}^c$, 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 λ. Then our contrastive loss function is expressed as:

$$\mathcal{L}_{reg} = -\log \frac{\sum_{k_1=1}^{K_1} \exp sim(\mathbf{z}^q, \mathbf{z}^+_{k_1})}{\sum_{k_1=1}^{K_1} \exp sim(\mathbf{z}^q, \mathbf{z}^+_{k_1}) + \sum_{k_2=1}^{K_2} \exp sim(\mathbf{z}^q, \mathbf{z}^-_{k_2})}, \tag{4}$$

where $\mathbf{z}^q, \mathbf{z}^+_{k_1}, \mathbf{z}^-_{k_2} \in \mathbf{Z}^{reg}$ and $sim(\mathbf{u}, \mathbf{v})$ is similarity function between two representations (e.g., inner product).

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3200734

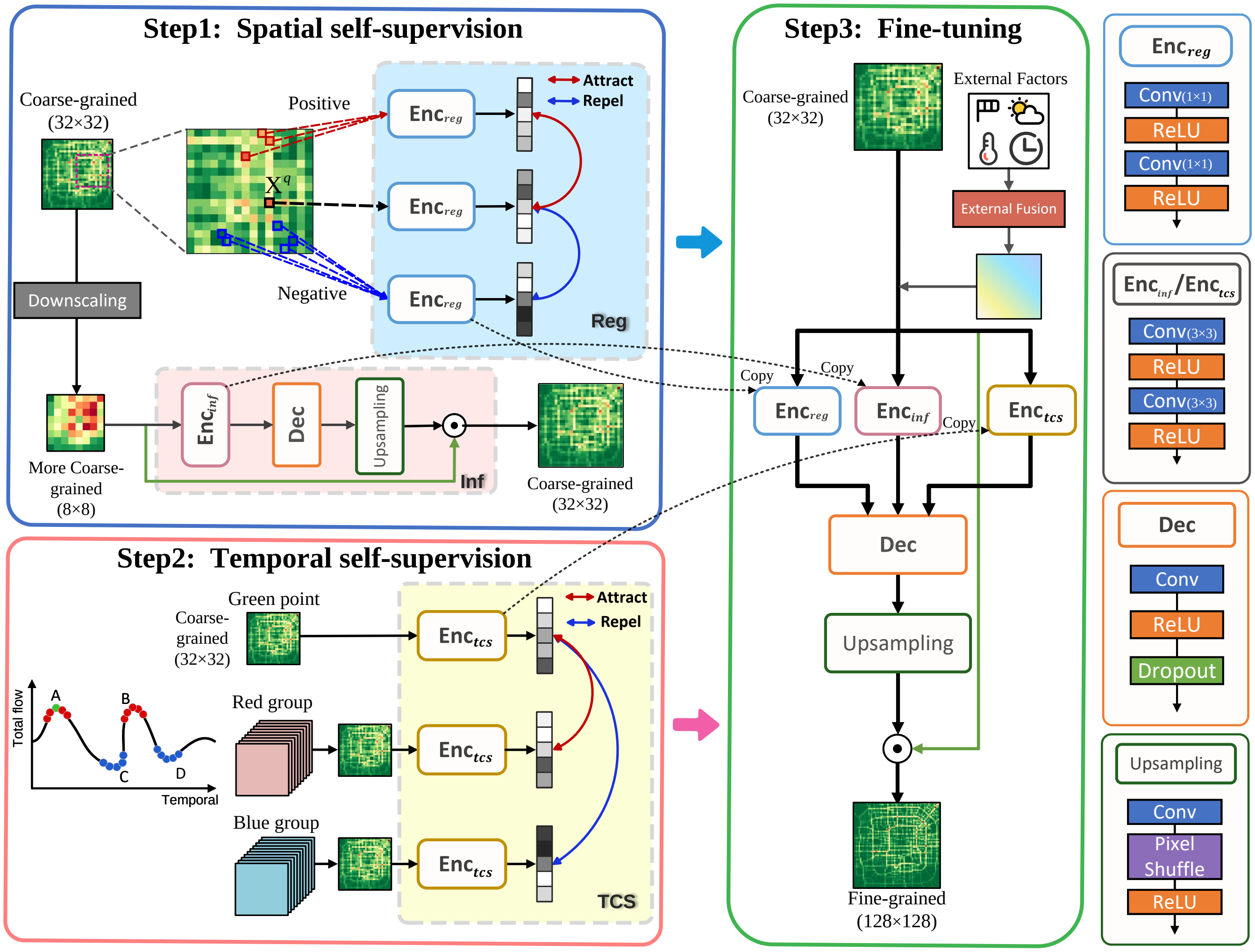


Fig. 2: The framework of UrbanSTC. There are there major parts: spatial self-supervision, temporal self-supervision and fine-tuning stage. Reg (light blue block) represents the Regional-level contrast; Inf (Light pink block) represents the Spatial super-resolution inference; TCS (Light yellow block) represents the Temporal contrast. Dec indicates a decoder that can convert the embedding vectors generated by the spatio-temporal self-supervision into the output fine-grained maps. UrbanSTC includes pre-training and fine-tuning stages. Among, Reg and Inf belong to the spatial self-supervision pre-training. TCS belongs to the temporal self-supervision pre-training. We first learn encoders through a spatio-temporal pre-training, and finally complete the network in the fine-tuning stage.

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 $\mathbf{X}^c \in \mathbb{R}_+^{I\times J}$ and upscaling factor $M \in \mathbb{Z}_+$, FUFI aims to learn a super-resolution model to infer the fine-grained flow map $\mathbf{X}^f \in \mathbb{R}_+^{MI\times MJ}$ under the *structural constraint*. The most important learning mechanism is how to split a coarse region x_{ij}^c to its M^2 fine-grained cells, which can be represented as $\mathbb{I} \in \mathbb{R}_+^{1\times 1} \rightarrow \mathbb{M} \in \mathbb{R}_+^{M\times M}$. 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 $\mathbb{I} \rightarrow \mathbb{M}$ with a pretext-task. In detail, we first get a down-scaling coarser granularity map $\mathbf{X}^{mc} \in \mathbb{R}_+^{\lfloor \frac{I}{M} \rfloor \times \lfloor \frac{J}{M} \rfloor}$ based on the coarse-grained map $\mathbf{X}^c$ and M, where each entry of $\mathbf{X}^{mc}$ equals to the sum of corresponding M^2 flow volumes in $\mathbf{X}^c$. Then we can construct a spatial super-resolution network inferring $\mathbf{X}^c$ from $\mathbf{X}^{mc}$. This pre-text task is able to capture the $\mathbb{I} \rightarrow \mathbb{M}$ pattern in advance, and could be benefit for improving the inference capability of surrounding flows.

For a $\mathbf{X}^{mc}$, we first encode it by two convolutional layers with C channels and 3×3 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 $\mathbf{X}^{mc}$ to the low-level hidden feature maps $\mathbf{H}^{inf} \in \mathbb{R}^{\frac{H}{M} \times \frac{W}{M} \times C}$. This component of our network is named spatial super-resolution encoder $\mathbf{Enc}_{inf}(\cdot)$ which is used later in the fine-tuning process. Then we can

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3200734

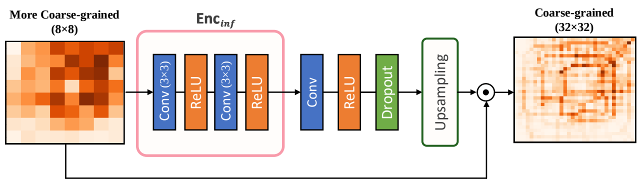


Fig. 3: Spatial Super-resolution Inference Network Pre-training. We get a down-scaling coarser granularity map (more coarse-grained) $\mathbf{X}^{mc}$ based on the coarse-grained map $\mathbf{X}^{c}$ and upscaling factor M. Spatial super-resolution inference network simplifies the difficulty of the task and imitates the process of inferring.

leverage the prior FUFI methods distributional upsampling at the end of their networks [3]–[5]. We also adopt M^2-Normalization[1] to impose the *structural constraint* on the network. The final loss is computed by the pixel-wise Mean Square Error (MSE):

$$\mathcal{L}_{inf} = \frac{1}{T}\sum_{t=1}^{T} \left\| \mathbf{X}_t^c - \mathcal{F}_{inf}\left(\mathbf{X}_t^{mc};\theta\right)\right\|^2, \tag{5}$$

where θ represents all learnable parameters in the inference network.

This inference structure and function $\mathcal{F}_{inf}$ 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 can get an anchor point $\mathbf{X}_t^c$, and then collect its positive ($\{\mathbf{X}_{t,k_3}^{+}\}_{k_3=1}^{K_3}$) and negative samples ($\{\mathbf{X}_{t,k_4}^{-}\}_{k_4=1}^{K_4}$) by identifying the similarities among samples, where K_3 and K_4 are the numbers of selected positive and negative temporal samples.

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 $\mathbf{Enc}_{tcs}(\cdot)$ has a similar structure to the spatial super-resolution inference network. It projects coarse-grained map $\mathbf{X}_t^c$ to the low-level hidden feature map $\mathbf{H}_t^{tcs} \in \mathbb{R}^{H\times W\times C}$. Then we adopt a batch normalization layers and the global average pooling layer. Finally, Multilayer Perceptron (MLP) with Relu activation function is used to make nonlinearity, converting the encoder feature map $\mathbf{H}_t^{tcs}$ to the high-level semantic features $\mathbf{D}_t^{tcs} \in \mathbb{R}^{C}$. 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:

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

where $\mathbf{X}^c$ is the current coarse-grained flow map and $\mathbf{X}^k$ 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 $\{\mathbf{X}_t^c\}_{t=1}^{T}$, 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:

$$\mathbf{X}_t^{+} = \sum_{k=1}^{K}\frac{1/dist_k^t}{\sum_{j=1}^{K}1/dist_j^t}\mathbf{X}_{t,k}^{+}, \tag{7}$$

$$\mathbf{X}_t^{-} = \sum_{k=1}^{K}\frac{dist_k^t}{\sum_{j=1}^{K}dist_j^t}\mathbf{X}_{t,k}^{-}, \tag{8}$$

where $dist_k^t$ 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 $\mathcal{I} =< \mathbf{X}^c, \mathbf{X}^+, \mathbf{X}^- >$. The triplet loss ensures that a pair of co-occuring $\mathbf{X}_t^c$ (*anchor*) and $\mathbf{X}_t^+$ (*positive*) are closer to each other in the embedding space while moving away from $\mathbf{X}_t^-$ (*negative*). We define the score of this triplet as:

$$d_f(\mathcal{I}) = \left\|f(\mathbf{X}_t^c) - f(\mathbf{X}_t^+)\right\|_2^2 - \left\|f(\mathbf{X}_t^c) - f(\mathbf{X}_t^-)\right\|_2^2 + \alpha, \tag{9}$$

$$\mathcal{L}_{TCS} = \frac{1}{T}\sum_{t=1}^{T}(max\left\{d_f(\mathcal{I}),0\right\}), \tag{10}$$

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3200734

Algorithm 1: Weight Sampling

```
Input: original coarse data {X^c}.
Output: complete data {X^c, X^+, X^-}.
for x ∈ {X^c_1, ..., X^c_T} do
    Build Max-heap and Min-heap.
    for y ∈ {X^c_1, ..., X^c_T} do
        if x ≠ y then
            Calculate the Euclidean distance dist^t
              between x and y.
            Adjust Max-head and Min-heap.
    Select Top-k positive and negative samples
      respectively.
    get X^+_t by Equation 7
    get X^-_t by Equation 8
```

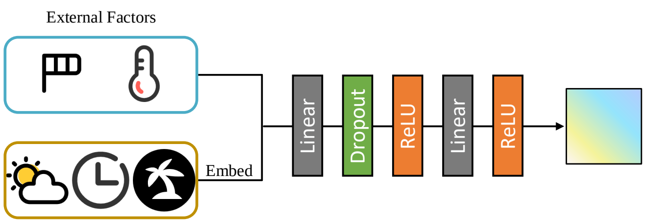


Fig. 4: External Factors Fusion. External Factors are separated into continuous features (blue block) and categorical features (yellow block).

where $f(.)$ is a non-linear mapping function that needs to be learned, and α 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 $\mathbf{e}_{con}$. 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 $\mathbf{e}_{cat}$. Then, we splice the two vectors $\mathbf{e}_{con}$ and $\mathbf{e}_{cat}$ to the final external embedding ($\mathbf{e} = [\mathbf{e}_{con}; \mathbf{e}_{cat}]$).

As shown in Figure 4. we use two layers of multi-layer perception with nonlinear transformation to feed external embedding $\mathbf{e}$. By using nonlinear transformation, different external factors are converged into a hidden state $\mathbf{X}^e \in \mathbb{R}_+^{I\times J}$. 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 $\mathbf{X}^c + \mathbf{X}^e$ 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 constrastive encoder $\mathbf{Enc}_{reg}(\cdot)$, spatial super-resolution inference encoder $\mathbf{Enc}_{inf}(\cdot)$ and TCS encoder $\mathbf{Enc}_{tcs}(\cdot)$. 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:

$$\mathbf{H}^{reg} = \mathbf{Enc}_{reg}(\mathbf{X}^c), \tag{11}$$

$$\mathbf{H}^{inf} = \mathbf{Enc}_{inf}(\mathbf{X}^c), \tag{12}$$

$$\mathbf{H}^{tcs} = \mathbf{Enc}_{tcs}(\mathbf{X}^c), \tag{13}$$

$$\mathbf{H}^{a} = \mathbf{Concat}(\mathbf{H}^{reg}, \mathbf{H}^{inf}, \mathbf{H}^{tcs}), \tag{14}$$

where $\mathbf{Concat}$ is the tensor concatenate operation. Then $\mathbf{Decoder}$ 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 upscaling factor M. At the end of PixelSuffle, we use a ReLU activation function. After the above operations, a feature $\mathbf{U}^f \in \mathbb{R}^{MH\times MW\times C}$ is obtained where the first two dimensions have been increased M times. Next, we use a 3×3 convolution with the 1-size channel to get a fine-grained flow distribution map of the hidden state $\mathbf{U}^f_o \in \mathbb{R}^{MH\times MW\times 1}$. Due to the *structural constraint* of FUFI 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:

$$W^f_{(i,j)} = \frac{U^f_{o(i,j)}}{\sum_{\substack{i'\in(\lfloor \frac{i}{M}\rfloor M, (\lfloor \frac{i}{M}\rfloor+1)M)] \\ j'\in(\lfloor \frac{j}{M}\rfloor M, (\lfloor \frac{j}{M}\rfloor+1)M)]}} U^f_{o(i'j')}}, \tag{15}$$

where $U^f_{o(i,j)}$ is the i-th row and j-th column cell in $\mathbf{U}^f_o$ 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 $\hat{\mathbf{X}}^f = \mathbf{X}^c \odot \mathbf{W}^f$. Mean Square Error (MSE) is used as the loss function:

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

where $\mathcal{F}$ represents the UrbanSTC model and θ represents all learnable parameters used in this model.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3200734

TABLE 2: Statistics of datasets.

Dataset	TaxiBJ	BikeNYC
Time span	P1: 7/1/2013-10/31/2013 P2: 2/1/2014-6/30/2014 P3: 3/1/2015-6/30/2015 P4: 11/1/2015-3/31/2016	1/1/2019- 31/3/2019
Time interval	30 minutes	1 hour
Coarse-grained size	32×32	40×20
Fine-grained size	128×128	80×40
Upscaling factor(M)	4	2
Latitude range	$39.82°\mathrm{N} - 39.99°\mathrm{N}$	$40.65°\mathrm{N} - 40.81°\mathrm{N}$
Longitude range	$116.26°\mathrm{E} - 116.49°\mathrm{E}$	$73.93°\mathrm{W} - 74.01°\mathrm{W}$
External Factors (meterology, time and event) in TaxiBJ dataset		
Temperature/°C	$[-24.6, 41.0]$	\
Wind speed/mph	$[0, 48.6]$	\
Weather conditions	16 types (e.g., Sunny)	\
Holidays	18	\

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 and test data by a ratio of 2:1:1 respectively.

• **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**[2] 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 40×20 grids as the coarse-grained map, and define the fine-granularity map with 80×40.

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 subregions by the historical average of its corresponding super-region, and distribute flows into sub-regions based on historical split proportions.

Image super-resolution methods:

• ***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 nonlinear 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 subtasks.

• ***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.

2. https://www.citibikenyc.com/system-data

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3200734

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.

Methods	P1(20%)			P1(40%)			P1(60%)			P1(80%)			P1(100%)		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
MEAN	20.918	12.019	4.469	20.918	12.019	4.469	20.918	12.019	4.469	20.918	12.019	4.469	20.918	12.019	4.469
HA	4.794	2.269	0.339	4.802	2.263	0.338	4.793	2.258	0.338	4.785	2.256	0.337	4.772	2.251	0.336
SRCNN	4.737	2.767	0.804	4.498	2.578	0.706	4.506	2.587	0.712	4.290	2.425	0.631	4.275	2.430	0.642
ESPCN	4.552	2.583	0.682	4.493	2.540	0.657	4.264	2.346	0.558	4.216	2.316	0.544	4.208	2.318	0.546
DeepSD	4.532	2.535	0.652	4.346	2.373	0.566	4.883	2.834	0.805	4.287	2.349	0.556	4.128	2.248	0.516
VDSR	4.546	2.556	0.669	4.299	2.354	0.562	4.198	2.279	0.527	4.119	2.229	0.503	4.054	2.186	0.485
SRResNet	4.734	2.800	0.844	4.383	2.520	0.696	4.276	2.437	0.654	4.179	2.366	0.618	4.079	2.291	0.580
LapSRN	4.676	2.738	0.801	4.642	2.715	0.789	4.309	2.432	0.635	4.153	2.305	0.567	4.083	2.255	0.542
IMDN	4.696	2.748	0.794	4.388	2.464	0.635	4.251	2.376	0.601	4.159	2.295	0.554	4.085	2.253	0.538
SCN	4.395	2.491	0.661	4.219	2.351	0.588	4.096	2.250	0.536	4.028	2.203	0.515	3.965	2.162	0.494
UrbanFM	4.560	2.343	0.398	4.321	2.213	0.369	4.195	2.140	0.350	4.108	2.095	0.340	4.042	2.062	0.337
UrbanPy	4.665	2.471	0.547	4.363	2.233	0.415	4.112	2.077	0.349	4.033	2.041	0.343	3.944	1.998	0.333
FODE	4.476	2.304	0.391	4.260	2.170	0.349	4.161	2.116	0.344	4.084	2.078	0.338	4.002	2.044	0.336
UrbanSTC	**4.083**	**2.022**	**0.302**	**3.988**	**1.983**	**0.302**	**3.941**	**1.962**	**0.301**	**3.900**	**1.942**	**0.298**	**3.845**	**1.922**	**0.298**
Δ	+7.10%	+12.24%	+10.91%	+5.48%	+8.62%	+10.65%	+3.78%	+5.54%	+10.95%	+3.18%	+4.85%	+11.57%	+2.51%	+3.80%	+10.51%

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.

Methods	P2(20%)			P2(40%)			P2(60%)			P2(80%)			P2(100%)		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
MEAN	26.729	15.350	5.364	26.729	15.350	5.364	26.729	15.350	5.364	26.729	15.350	5.364	26.729	15.350	5.364
HA	6.568	2.889	0.358	5.875	2.679	0.342	5.669	2.620	0.338	5.544	2.587	0.335	5.512	2.576	0.334
SRCNN	5.613	3.201	0.837	4.994	2.855	0.706	5.172	3.036	0.801	4.924	2.839	0.713	4.978	2.896	0.748
ESPCN	5.461	3.062	0.738	5.186	2.987	0.740	4.934	2.779	0.637	4.554	2.473	0.482	5.072	2.957	0.749
DeepSD	5.412	2.991	0.704	5.608	3.290	0.892	4.716	2.585	0.546	5.018	2.816	0.659	4.909	2.738	0.625
VDSR	5.449	3.024	0.727	4.753	2.608	0.561	4.954	2.795	0.660	4.494	2.444	0.492	4.429	2.402	0.475
SRResNet	5.801	3.420	0.992	4.946	2.878	0.749	4.702	2.760	0.653	4.572	2.600	0.614	4.548	2.573	0.605
LapSRN	5.717	3.343	0.931	4.844	2.751	0.664	4.818	2.753	0.673	4.535	2.525	0.554	4.555	2.556	0.569
IMDN	5.790	3.547	1.123	4.927	2.971	0.853	4.710	2.792	0.755	4.573	2.688	0.703	4.476	2.608	0.661
SCN	5.222	2.932	0.721	4.640	2.567	0.579	4.487	2.475	0.528	4.402	2.422	0.505	4.336	2.381	0.490
UrbanFM	5.546	2.855	0.433	4.805	2.469	0.353	4.588	2.365	0.336	4.489	2.309	0.324	4.414	2.272	0.318
UrbanPy	5.528	2.803	0.485	4.728	2.412	0.370	4.464	2.276	0.334	4.446	2.279	0.341	4.315	2.210	0.323
FODE	5.362	2.734	0.395	4.704	2.416	0.337	4.538	2.331	0.323	4.434	2.285	0.325	4.366	2.248	0.317
UrbanSTC	**4.975**	**2.424**	**0.297**	**4.454**	**2.231**	**0.294**	**4.347**	**2.185**	**0.288**	**4.274**	**2.157**	**0.288**	**4.225**	**2.136**	**0.288**
Δ	+4.73%	+11.34%	+17.04%	+4.01%	+7.50%	+12.76%	+2.62%	+4.00%	+10.84%	+2.91%	+5.35%	+11.11%	+2.09%	+3.35%	+9.15%

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.

Methods	P3(20%)			P3(40%)			P3(60%)			P3(80%)			P3(100%)		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
MEAN	27.442	16.029	5.612	27.442	16.029	5.612	27.442	16.029	5.612	27.442	16.029	5.612	27.442	16.029	5.612
HA	5.833	2.741	0.337	5.746	2.713	0.333	5.731	2.707	0.331	5.692	2.695	0.330	5.675	2.670	0.328
SRCNN	5.581	3.317	0.906	5.150	2.962	0.728	5.082	2.936	0.718	4.923	2.821	0.666	4.891	2.817	0.673
ESPCN	5.273	3.013	0.717	5.043	2.848	0.638	5.091	2.888	0.656	4.796	2.668	0.556	4.853	2.716	0.579
DeepSD	5.257	2.935	0.666	5.048	2.796	0.606	4.960	2.749	0.583	4.878	2.690	0.559	4.720	2.580	0.510
VDSR	5.285	2.982	0.699	4.963	2.748	0.591	4.786	2.626	0.536	4.695	2.568	0.512	4.616	2.522	0.495
SRResNet	5.578	3.352	0.945	5.120	2.998	0.776	4.934	2.857	0.705	4.773	2.734	0.643	4.658	2.648	0.602
LapSRN	5.832	3.535	1.019	5.135	2.970	0.740	5.041	2.920	0.721	4.923	2.828	0.675	4.641	2.589	0.550
IMDN	5.635	3.493	1.077	5.143	3.107	0.876	4.908	2.930	0.788	4.745	2.794	0.715	4.690	2.765	0.704
SCN	5.090	2.899	0.694	4.826	2.702	0.601	4.670	2.593	0.549	4.575	2.531	0.522	4.514	2.494	0.506
UrbanFM	5.299	2.738	0.379	4.951	2.558	0.350	4.761	2.456	0.336	4.656	2.408	0.330	4.578	2.356	0.314
UrbanPy	5.342	2.827	0.529	4.946	2.532	0.382	4.743	2.443	0.362	4.578	2.346	0.332	4.436	2.272	0.318
FODE	5.165	2.686	0.380	4.875	2.521	0.347	4.712	2.434	0.331	4.616	2.387	0.327	4.536	2.345	0.319
UrbanSTC	**4.781**	**2.383**	**0.287**	**4.607**	**2.309**	**0.292**	**4.512**	**2.271**	**0.288**	**4.439**	**2.240**	**0.288**	**4.382**	**2.215**	**0.285**
Δ	+6.07%	+11.28%	+14.84%	+4.54%	+8.41%	+12.31%	+3.38%	+6.70%	+12.99%	+2.97%	+4.52%	+11.93%	+1.22%	+2.51%	+9.24%

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.

Methods	P4(20%)			P4(40%)			P4(60%)			P4(80%)			P4(100%)		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
MEAN	19.049	11.070	4.192	19.049	11.070	4.192	19.049	11.070	4.192	19.049	11.070	4.192	19.049	11.070	4.192
HA	4.306	2.067	<u>0.319</u>	4.238	2.052	<u>0.319</u>	4.209	2.043	0.319	4.223	2.045	0.320	4.201	2.039	0.320
SRCNN	4.048	2.369	0.668	4.065	2.381	0.660	3.799	2.182	0.569	3.944	2.277	0.613	3.813	2.188	0.571
ESPCN	3.983	2.290	0.600	3.865	2.187	0.542	4.112	2.430	0.684	3.853	2.194	0.552	3.914	2.277	0.607
DeepSD	3.980	2.240	0.562	3.910	2.181	0.527	3.924	2.215	0.552	3.806	2.121	0.511	3.662	2.030	0.472
VDSR	3.952	2.239	0.573	3.741	2.075	0.489	3.655	2.015	0.462	3.644	2.007	0.457	3.555	1.948	0.431
SRResNet	4.118	2.463	0.738	4.053	2.431	0.729	3.761	2.184	0.591	3.710	2.102	0.508	3.630	2.067	0.523
LapSRN	4.467	2.753	0.884	4.150	2.489	0.745	3.705	2.103	0.530	3.673	2.080	0.520	3.679	2.118	0.544
IMDN	4.100	2.530	0.818	3.828	2.301	0.686	3.703	2.203	0.635	3.619	2.119	0.580	3.848	2.340	0.720
SCN	<u>3.798</u>	2.154	0.550	<u>3.660</u>	2.048	0.496	<u>3.573</u>	1.987	0.467	<u>3.524</u>	1.952	0.450	3.486	1.927	0.439
UrbanFM	4.054	2.126	0.373	3.794	1.969	0.330	3.677	1.908	0.323	3.601	1.865	0.315	3.559	1.841	0.305
UrbanPy	3.959	2.088	0.413	3.740	1.936	0.342	3.644	1.889	0.332	3.606	1.868	0.325	<u>3.470</u>	<u>1.801</u>	0.313
FODE	3.912	<u>2.042</u>	0.350	3.725	<u>1.930</u>	0.321	3.627	<u>1.879</u>	<u>0.314</u>	3.565	<u>1.846</u>	<u>0.308</u>	3.529	1.828	<u>0.304</u>
UrbanSTC	**3.640**	**1.837**	**0.278**	**3.542**	**1.796**	**0.282**	**3.474**	**1.769**	**0.282**	**3.454**	**1.759**	**0.283**	**3.416**	**1.742**	**0.278**
Δ	+4.16%	+10.04%	+12.85%	+3.22%	+6.94%	+11.60%	+2.77%	+5.85%	+10.19%	+1.99%	+4.71%	+8.12%	+1.56%	+3.28%	+8.55%

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).

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{X}_i - \hat{\mathbf{X}}_i\right)^2}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|\mathbf{X}_i - \hat{\mathbf{X}}_i\right|$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\mathbf{X}_i - \hat{\mathbf{X}}_i}{\mathbf{X}_i}\right|$$

where $\hat{\mathbf{X}}_i$ is a prediction for fine-grained flow, and $\mathbf{X}_i$ 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.

TABLE 7: Ablation Studies. We report the strategies used in different models on TaxiBJ dataset's average results.

Regional contrast	Spatial super-resolution	Temporal contrast	TaxiBJ RMSE	MAE	MAPE
✓			4.118	2.100	0.311
	✓		4.019	2.040	0.297
		✓	4.008	2.027	0.290
✓	✓		3.970	2.009	0.289
✓		✓	3.983	2.009	0.287
	✓	✓	3.975	2.008	0.288
✓	✓	✓	**3.967**	**2.004**	**0.287**

(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 FUFI performance. Image super-resolution method SCN [56] performers 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 adapt to the FUFI 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

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3200734

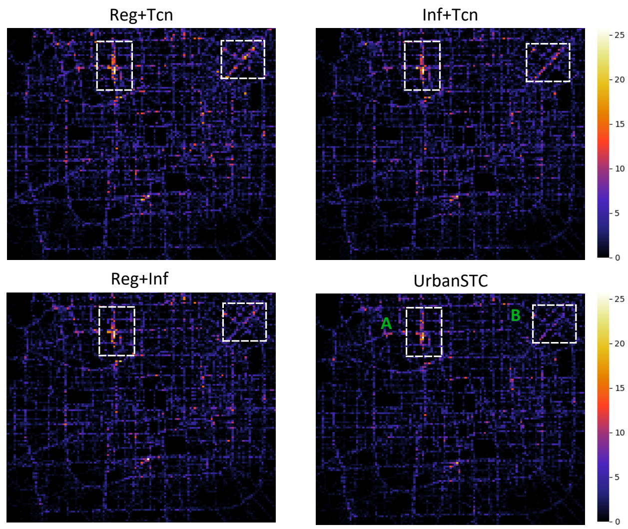


Fig. 5: Visualization of the Ablation Study.

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 1×1, while that of in Inf encoder is 3×3, 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 $\left\|\mathbf{X}^f - \hat{\mathbf{X}}^f\right\|_{1,1}$ generated by UrbanSTC and other ablation parts, where a brighter pixel indicates a large 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].

TABLE 8: End-to-end and two-stage comparison.

Mehtods	TaxiBJ			BikeNYC		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE
End-to-End	3.980	2.053	0.294	1.120	0.245	0.077
Two-stage	**3.958**	**1.998**	**0.284**	**1.093**	**0.236**	**0.073**
Δ	0.55%	2.68%	3.40%	2.41%	3.67%	5.19%

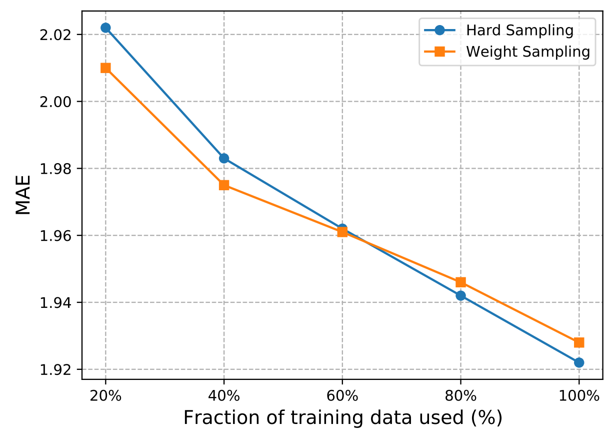


Fig. 6: Performance comparison between Hard Sampling and Weight Sampling on TaxiBJ-P1 dataset.

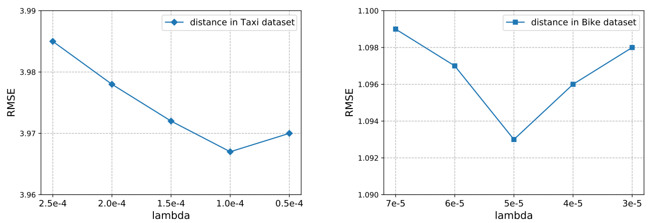
5.2.3 Temporal Contrastive Sampling Analysis

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.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3200734

TABLE 9: The average RMSE, MAE and MAPE on TaxiBJ dataset with external factors. Note that "+E" represents a model that incorporates external factors. The best results are bold.

Methods	P1			P2			P3			P4		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
UrbanFM+E	3.970	2.023	0.334	4.355	2.239	0.317	4.530	2.335	0.321	3.528	1.824	0.303
UrbanPy+E	3.909	1.981	0.330	4.353	2.230	0.327	4.466	2.294	0.323	3.498	1.817	0.317
FODE+E	3.915	1.996	0.332	4.348	2.235	0.316	4.505	2.329	0.314	3.505	1.821	0.311
UrbanSTC	3.845	1.922	0.298	4.225	2.136	0.288	4.382	2.215	0.285	3.416	1.742	0.278
UrbanSTC+E	**3.841**	**1.917**	**0.292**	**4.209**	**2.125**	**0.284**	**4.376**	**2.210**	**0.283**	**3.404**	**1.738**	**0.275**

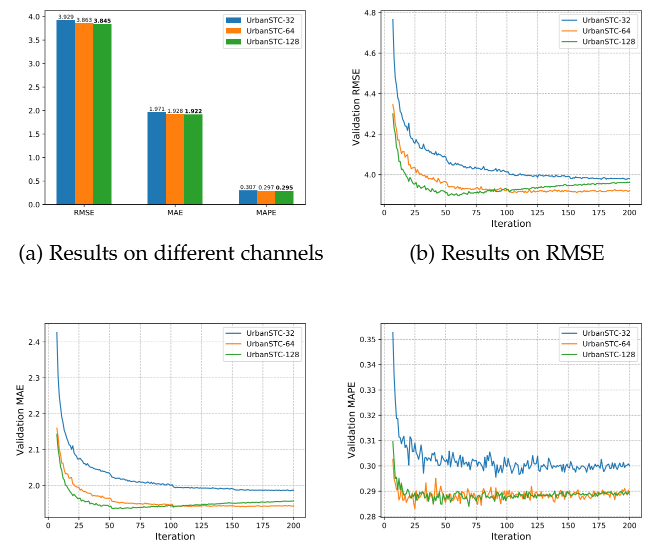


(a) λ in Taxi dataset (b) λ in Bike dataset

Fig. 7: Effect of λ. We explore the influence of λ in the spatial contrastive learning.

TABLE 10: Efficiency Evaluated on the P1 dataset. The entire model UrbanSTC and its four components have been tested separately.

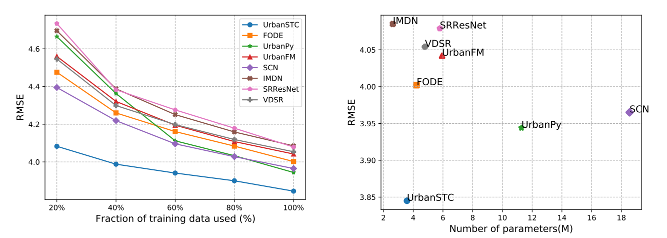
Method	Params	Training Time	Inference Time	Total Time	RMSE
VDSR	4.79M	4.37s	0.76s	9.10mins	4.054
SRResNet	5.79M	11.4s	1.57s	33.25mins	4.079
IMDN	2.63M	4.21s	0.69s	13.33mins	4.085
SCN	18.55M	23.86s	5.21s	59.65mins	3.965
UrbanFM	5.94M	12.28s	1.80s	10.23mins	4.042
UrbanPy	11.28M	27.39s	11.89s	79.89mins	3.944
FODE	4.23M	14.05s	1.91s	17.56mins	4.002
Reg	0.03M	3.89s	-	6.48mins	-
Inf	1.41M	0.80s	-	1.60mins	-
TCS	0.16M	0.99s	-	1.65mins	-
Fine-tuning	1.98M	3.34s	0.66s	2.78mins	3.845
UrbanSTC	3.58M	9.02s	0.66s	12.51mins	3.845



(a) Results on different channels (b) Results on RMSE

(c) Results on MAE (d) Result on MAPE

Fig. 8: Study on Configurations. The convergence rate loss error of the self-supervised module under different channel dimensions



(a) Results on RMSE (b) Model parameters comparison

Fig. 9: Study on Parameters. Experiments on the P1 dataset with different training data fractions and the comparison of parameter cost.

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 λ in the regional-level contrastive learning. Figure 7a shows the different performances with a varying setting of λ. The regional-level contrast judges which regions are positive and negative

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3200734

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.

Methods	BikeNYC(20%)		BikeNYC(40%)		BikeNYC(60%)		BikeNYC(80%)		BikeNYC(100%)	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
MEAN	3.776	1.281	3.776	1.281	3.776	1.281	3.776	1.281	3.776	1.281
HA	1.498	0.359	1.476	0.355	1.511	0.365	1.506	0.364	1.502	0.364
SRCNN	1.419	0.452	1.306	0.421	1.228	0.373	1.201	0.364	1.262	0.413
ESPCN	1.458	0.489	1.432	0.495	1.302	0.402	1.322	0.451	1.295	0.411
VDSR	1.888	0.838	1.740	0.758	1.616	0.700	1.531	0.665	1.476	0.626
SRResNet	1.843	0.891	1.713	0.781	1.607	0.712	1.488	0.643	1.443	0.600
LapSRN	1.582	0.635	1.448	0.550	1.392	0.516	1.339	0.492	1.320	0.464
IMDN	1.407	0.521	1.345	0.456	1.292	0.447	1.241	0.422	1.220	0.402
SCN	1.331	0.424	1.276	0.404	1.191	0.356	1.200	0.362	1.162	0.332
UrbanFM	1.405	0.316	1.302	0.309	1.215	0.283	1.215	0.265	1.172	0.263
UrbanPy	1.381	0.315	1.310	0.301	1.271	0.286	1.200	0.273	<u>1.126</u>	<u>0.250</u>
FODE	<u>1.293</u>	<u>0.302</u>	<u>1.214</u>	<u>0.279</u>	<u>1.167</u>	<u>0.265</u>	<u>1.146</u>	<u>0.258</u>	1.134	0.253
UrbanSTC	**1.267**	**0.276**	**1.191**	**0.257**	**1.146**	**0.246**	**1.107**	**0.239**	**1.093**	**0.236**
Δ	+2.01%	+8.61%	+1.89%	+7.89%	+1.80%	+7.17%	+3.40%	+7.36%	+2.93%	+5.60%

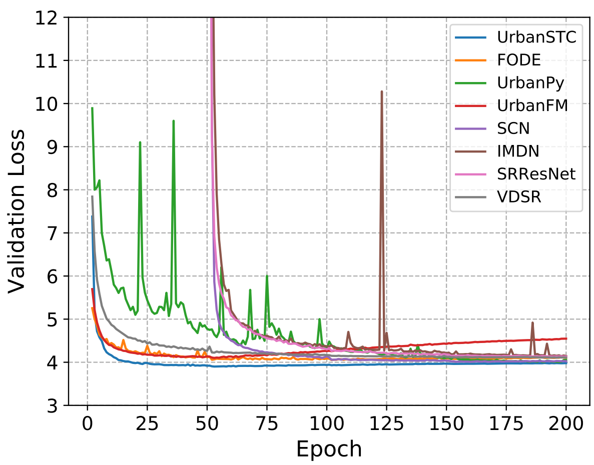


Fig. 10: Convergence rate of different methods.

samples based on the threshold λ. The experimental result shows that the best result is achieved when the threshold is 1e-4 on the taxi dataset. Figures 7b represents that λ = 5e-5 yields the best performance.

For the parameter analysis, Figure 9a represents that UrbanSTC can get better results than other models with different training data fractions. Figure 9b indicates the traditional image super-resolution methods, e.g.,, IMDN, VDSR and SRResNet are not suitable for the FUFI problem due to the inherent difference. Although SRResNet and UrbanFM have similar structures, the M^2-Normalization layer in UrbanFM contributes to the FUFI problem. UrbanPy uses a cascading model for forecasting fine-grained urban flows by decomposing the original task into multiple subtasks, which leads to the increase of computing complexity. FODE utilizes ODE module to replace the ResNet strucure in the UrbanFM. Because the above modules can be viewed as a discretization of a continuous ODE operator, which greatly improves the convergence speed and reduces the number of parameters. For our model UrbanSTC, we design several self-supervised pretext tasks to make encoders rich in spatio-temporal information. As shown in Table 10, "Reg" indicates the regional-level contrast pre-training; "Inf" illustrates the spatial super-resolution inference network; "TCS" denotes the temporal contrast. UrbanSTC consists of three self-supervised modules and a fine-tuning stage. The parameter cost of UrbanSTC is slightly higher than IMDN because the latter is a lightweight image super-resolution method designed in mobile devices. Based on our well-designed self-supervised tasks, UrbanSTC can capture spatio-temporal knowledge in advance and perform better than other baselines with a relatively small amount of parameters.

We further conduct a comparison between UrbanSTC and baselines in terms of the training time and inference time. We reported the training time of each epoch and the total training time until model convergence in P1 dataset of TaxiBJ, which contains 1530 training snapshots and 765 test snapshots respectively. Even our model contains two stages, the training time of each epoch is less than all previous FUFI models (UrbanFM, UrbanPy and FODE) as shown in Table 10. It is mainly because the structure of proposed encoders is simple while well-designed that can capture rich spatio-temporal characteristics in advance. Two image super-resolution methods, VDSR and IMDN are efficient in the training process, yet their performances are far more worse than our method.

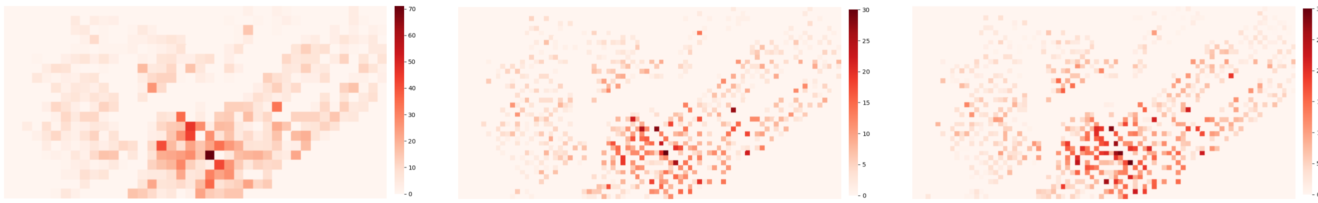
As shown in Fig.10, UrbanSTC can efficiently converge with a small number of epochs. Even UrbanSTC spent slightly more total training time than UrbanFM and VDSR, it is efficient with the best results achieved. In summary, extensive experiments demonstrate that UrbanSTC can achieve the best results efficiently by using a small amount of parameters.

5.3 Results on BikeNYC

Table 11 presents the comparison results on the BikeNYC dataset. Since we cannot get the external factors of this dataset, we will do not add such information in the experiments. In this experiment, the baseline DeepSD will be the same as SRCNN when M is $2\times$, therefore we remove the DeepSD.

BikNYC dataset is more sparse than TaxiBJ dataset. Nonetheless, UrbanSTC still yields 2.93% and 5.60% im-

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3200734



(a) Coarse-grained Crowd Flows (b) Fine-grained Crowd Flows (c) UrbanSTC Fine-grained Inference

Fig. 11: Visualization of crowd flows in BikeNYC.

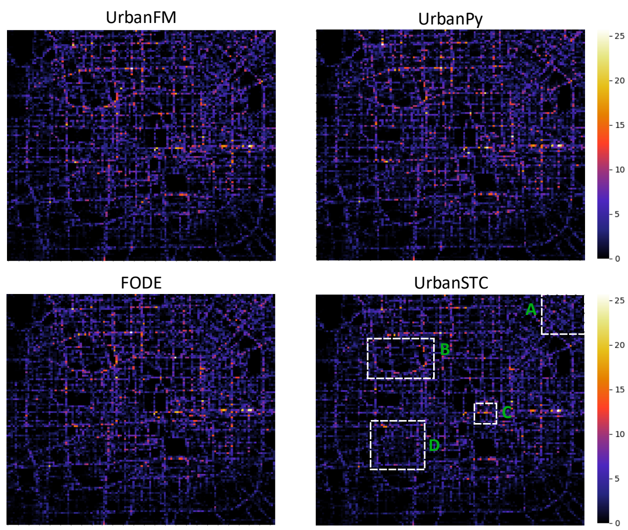


Fig. 12: Visualization for inference errors among different methods on P1 dataset. Best view in color.

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 $\left\|\mathbf{X}^f - \hat{\mathbf{X}}^f\right\|_{1,1}$ generated by UrbanSTC and the other three baselines for a sample at the $4\times$ 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 and 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 quantitive 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

[1] Z. Li, J. Zhang, Q. Wu, Y. Gong, J. Yi, and C. Kirsch, "Sample adaptive multiple kernel learning for failure prediction of railway points," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2848–2856.

[2] Y. Gong, Z. Li, J. Zhang, W. Liu, and Y. Zheng, "Online spatio-temporal crowd flow distribution prediction for complex metro system," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[3] 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 & Data Mining*, 2019, pp. 3132–3142.

[4] F. Zhou, L. Li, T. Zhong, G. Trajcevski, K. Zhang, and J. Wang, "Enhancing urban flow maps via neural odes," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, 2020, pp. 1295–1302.

[5] K. Ouyang, Y. Liang, Y. Liu, Z. Tong, S. Ruan, D. Rosenblum, and Y. Zheng, "Fine-grained urban flow inference," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2022.3200734

[6] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, and T. Li, "Predicting city-wide crowd flows using deep spatio-temporal residual networks," *Artificial Intelligence*, vol. 259, pp. 147–166, 2018.

[7] W. S. H. M. W. Ahmad, N. A. M. Radzi, F. Samidi, A. Ismail, F. Abdullah, M. Z. Jamaludin, and M. Zakaria, "5g technology: Towards dynamic spectrum sharing using cognitive radio networks," *IEEE Access*, vol. 8, pp. 14 460–14 488, 2020.

[8] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 3, pp. 1–55, 2014.

[9] Y. Gong, Z. Li, J. Zhang, W. Liu, and J. Yi, "Potential passenger flow prediction: A novel study for urban transportation development," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4020–4027.

[10] S. Sarkar, S. Chawla, S. Ahmad, J. Srivastava, H. Hammady, F. Filali, W. Znaidi, and J. Borge-Holthoefer, "Effective urban structure inference from traffic flow dynamics," *IEEE Transactions on Big Data*, vol. 3, no. 2, pp. 181–193, 2017.

[11] J. Gutiérrez Bayo, "International case studies of smart cities: Santander, spain," *Inter-American Development Bank*, 2016.

[12] C. Schreiner, "International case studies of smart cities: Rio de janeiro, brazil," *Inter-American Development Bank*, 2016.

[13] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1422–1430.

[14] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[15] R. Y. Zhang, J. Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros, "Real-time user-guided image colorization with learned deep priors," *ACM Transactions on Graphics*, vol. 36, no. 4, p. 119, 2017.

[16] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *International Conference on Learning Representations*, 2018.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[19] Y. Gong, Z. Li, J. Zhang, W. Liu, Y. Yin, and Y. Zheng, "Missing value imputation for multi-view urban statistical data via spatial correlation learning," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[20] Y. Gong, Z. Li, J. Zhang, W. Liu, B. Chen, and X. Dong, "A spatial missing value imputation method for multi-view urban statistical data," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 1310–1316.

[21] J. Cai, S. Gu, R. Timofte, and L. Zhang, "Ntire 2019 challenge on real image super-resolution: Methods and results," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[22] Z. Wang, J. Chen, and S. C. Hoi, "Deep learning for image super-resolution: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[23] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[24] R. Shen, J. Xu, Q. Bao, W. Li, H. Yuan, and M. Xu, "Fine-grained urban flow prediction via a spatio-temporal super-resolution scheme," in *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*. Springer, 2020, pp. 360–375.

[25] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 6572–6583.

[26] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1134–1141.

[27] P. Sermanet, C. Lynch, J. Hsu, and S. Levine, "Time-contrastive networks: Self-supervised learning from multi-view observation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2017, pp. 486–487.

[28] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2794–2802.

[29] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *International Conference on Learning Representations*, 2018.

[30] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[31] K. Nazeri, H. Thasarathan, and M. Ebrahimi, "Edge-informed single image super-resolution," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[32] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1874–1883.

[33] M. W. Thornton, P. M. Atkinson, and D. Holland, "Sub-pixel mapping of rural land cover objects from fine spatial resolution satellite sensor imagery using super-resolution pixel-swapping," *International Journal of Remote Sensing*, vol. 27, no. 3, pp. 473–491, 2006.

[34] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE transactions on acoustics, speech, and signal processing*, vol. 29, no. 6, pp. 1153–1160, 1981.

[35] C. E. Duchon, "Lanczos filtering in one and two dimensions," *Journal of Applied Meteorology and Climatology*, vol. 18, no. 8, pp. 1016–1022, 1979.

[36] S. Dai, M. Han, W. Xu, Y. Wu, Y. Gong, and A. K. Katsaggelos, "Softcuts: a soft edge smoothness prior for color image super-resolution," *IEEE Transactions on Image Processing*, vol. 18, no. 5, pp. 969–981, 2009.

[37] J. Sun, Z. Xu, and H.-Y. Shum, "Image super-resolution using gradient profile prior," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[38] Q. Yan, Y. Xu, X. Yang, and T. Q. Nguyen, "Single image superresolution based on gradient profile sharpness," *IEEE Transactions on Image Processing*, vol. 24, no. 10, pp. 3187–3202, 2015.

[39] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1. IEEE, 2004, pp. I–I.

[40] K. Zhang, D. Tao, X. Gao, X. Li, and J. Li, "Coarse-to-fine learning for single-image super-resolution," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 5, pp. 1109–1122, 2016.

[41] C. Deng, J. Xu, K. Zhang, D. Tao, X. Gao, and X. Li, "Similarity constraints-based structured output regression machine: An approach to image super-resolution," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 12, pp. 2472–2485, 2015.

[42] W. Yang, Y. Tian, F. Zhou, Q. Liao, H. Chen, and C. Zheng, "Consistent coding scheme for single-image super-resolution via independent dictionaries," *IEEE Transactions on Multimedia*, vol. 18, no. 3, pp. 313–325, 2016.

[43] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[45] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1646–1654.

[46] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[47] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in

International conference on machine learning. PMLR, 2020, pp. 1597–1607.

[48] A. Miech, J.-B. Alayrac, L. Smaira, I. Laptev, J. Sivic, and A. Zisserman, "End-to-end learning of visual representations from uncurated instructional videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9879–9889.

[49] Y. Gong, Z. Li, J. Zhang, W. Liu, Y. Zheng, and C. Kirsch, "Network-wide crowd flow prediction of sydney trains via customized online non-negative matrix factorization," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1243–1252.

[50] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[51] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2840–2848.

[52] Y. Gong, J. Yi, D.-D. Chen, J. Zhang, J. Zhou, and Z. Zhou, "Inferring the importance of product appearance with semi-supervised multi-modal enhancement: A step towards the screenless retailing," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 1120–1128.

[53] T. Vandal, E. Kodra, S. Ganguly, A. Michaelis, R. Nemani, and A. R. Ganguly, "Deepsd: Generating high resolution climate change projections through single image super-resolution," in *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining*, 2017, pp. 1663–1672.

[54] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep laplacian pyramid networks for fast and accurate super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 624–632.

[55] Z. Hui, X. Gao, Y. Yang, and X. Wang, "Lightweight image super-resolution with information multi-distillation network," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 2024–2032.

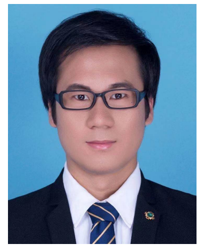
[56] Y. Fan, J. Yu, D. Liu, and T. S. Huang, "Scale-wise convolution for image restoration," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 10 770–10 777.

[57] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[58] O. Sumer, T. Dencker, and B. Ommer, "Self-supervised learning of pose embeddings from spatiotemporal relations in videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4298–4307.

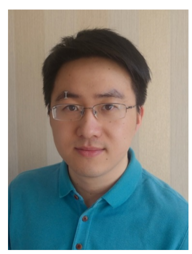
[59] J. Yang, J. M. Alvarez, and M. Liu, "Self-supervised learning of depth inference for multi-view stereo," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7526–7534.

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.



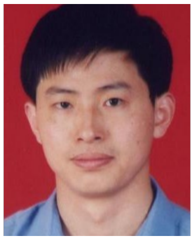
Yongshun Gong (S'19-M'21) is an Associate Professor at the School of Software, Shandong University, China. He received his Ph.D. degree from University of Technology Sydney in 2021. His principal research interest covers the data science and machine learning, in particular, the following areas: spatio-temporal 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-NNLS, IEEE T-CYB, NeurIPS, KDD, CIKM, AAAI and IJCAI.

Meng Chen received his Ph.D. degree in computer science and technology in 2016 from Shandong University, China. He worked as a Postdoctoral fellow from 2016 to 2018 in the School of Information Technology, York University, Canada. He is currently an assistant professor in the School of Software, Shandong University, China. His research interest is in the area of trajectory data mining and traffic management.



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 text book 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.



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.