Enhancing Urban Flow Maps via Neural ODEs

Fan Zhou[1] , **Liang Li**[1] , **Ting Zhong**[1] , **Goce Trajcevski**[2] , **Kunpeng Zhang**[3] and **Jiahao Wang**[1]
[1]School of Information and Software Engineering,
University of Electronic Science and Technology of China
[2]Iowa State University, Ames IA
[3]University of Maryland, College Park MD
{fan.zhou, zhongting, wangjh}@uestc.edu.cn, liliang2333@gmail.com, gocet25@iastate.edu,
kpzhang@umd.edu

Abstract

Flow super-resolution (FSR) enables inferring fine-grained urban flows with coarse-grained observations and plays an important role in traffic monitoring and prediction. The existing FSR solutions rely on deep CNN models (e.g., ResNet) for learning spatial correlation, incurring excessive memory cost and numerous parameter updates. We propose to tackle the urban flows inference using dynamic systems paradigm and present a new method *FODE – F*SR with *O*rdinary *D*ifferential *E*quations (*ODE*s). FODE extends neural ODEs by introducing an affine coupling layer to overcome the problem of numerically unstable gradient computation, which allows more accurate and efficient spatial correlation estimation, without extra memory cost. In addition, FODE provides a flexible balance between flow inference accuracy and computational efficiency. A FODE-based augmented normalization mechanism is further introduced to constrain the flow distribution with the influence of external factors. Experimental evaluations on two real-world datasets demonstrate that FODE significantly outperforms several baseline approaches.

1 Introduction

Urban flow super-resolution (FSR) aims at inferring fine-grained crowd flows in a city based on coarse-grained observations. As a variant of image SR in the traffic domain [Cai *et al.*, 2019; Wang *et al.*, 2019] it has practical significance in urban planning and traffic monitoring. Despite its close relationship to image SR, FSR has certain constraints the most important one being the *structural constraint* – i.e., the sum of the flow volumes in surrounding regions in the inferred fine-grained map strictly equals that of their corresponding superregion in the original map. However, in practice, the distribution of the flows in a given region is affected by many *external factors*, e.g., weather, time-of-day, etc.

A recent work [Liang *et al.*, 2019] addresses FSR problem based on the residual networks [He *et al.*, 2016] and uses a simple normalization scheme to constrain the structural distribution of flows. However, the proposed architecture heavily relies on empirically stacking deep neural networks and, consequently, lacks principles to guide the design of effective and interpretable FSR networks. Furthermore, it pays no attention to the computational overheads as the network goes deeper, which requires significantly more memory cost and network parameters – making it hard to be optimized and prone to be overfitting.

Recently, there has been a growing interest in bridging the gap between neural networks and dynamic systems [E, 2017; Lu *et al.*, 2018]. In particular, a recent study [Chen *et al.*, 2018] has shown that ResNet [He *et al.*, 2016] can be interpreted as discretized Neural Ordinary Differential Equations (NODE), which provides a new perspective of improving stability and trainability of neural networks. For example, an additional state (called adjoint) is used in [Chen *et al.*, 2018] to solve the ODEs. In this way, it does not need to store the intermediate states of the forward pass, resulting in $\mathcal{O}(1)$ memory cost in each ODE block. A few studies have been proposed to improve the performance of NODE [Liu *et al.*, 2019b] and/or to apply NODE in different domains such as graph neural networks [Poli *et al.*, 2019], time series learning [Rubanova *et al.*, 2019; De Brouwer *et al.*, 2019] and generative models [Heinonen and Lähdesmäki, 2019; Grathwohl *et al.*, 2019]. However, the dynamics of either the hidden state or the adjoint are usually unstable, due to the numerical instability of solving the backward ODEs. ANODE [Gholami *et al.*, 2019] and its variant [Zhang *et al.*, 2019] improve the robustness and generalization of the adjoint method by adding a few of the intermediate states from the forward pass however, these cannot fundamentally solve the incorrect gradient problem.

In this work, we introduce a new SR method – FODE (FSR with ODEs) for fine-grained urban flow inference. FODE is a more general neural ODE architecture that addresses the numerical instability problem of previous methods, while not incurring extra memory cost. The main idea of FODE is to incorporate an affine coupling layer in each ODE block to avoid the inaccurate gradient issue. The input to a FODE block can then be accurately reconstructed from its outputs without the need of storing intermediate states. We show theoretically and empirically that the proposed FODE model can be used to learn spatial correlations among urban regions and the influence of external features. The main contributions of this work can be summarized as:

- We present a new neural ODE framework that can accu-

rately compute the gradients in each block.

- Our novel FSR method requires significantly less memory for fine-grained flow inference compared with ResNet-based deep neural network.
- The augmented normalization scheme disentangles the influence of external factors on different regions.
- Experiments conducted on two real-world datasets demonstrate that our FODE not only outperforms strong baselines on FSR task, but also provides a flexible balance between computational cost and inference accuracy.

2 Architecture and Methodology

We now present the problem settings and discuss in detail the main aspects of the proposed FODE approach.

We assume a grid-based segmentation, similar to [Zhang *et al.*, 2017], which divides a city map $\mathcal{M}$ into $H \times W$ spatial grid-cells based on their geographical locations, i.e., $\mathcal{M} = \{r_{ij}\}_{H \times M}$. Each cell r_{ij} corresponds to a spatial region – the i^{th} row and the j^{th} column of $\mathcal{M}$. Let $\mathbf{X} \in \mathbb{R}_+^{H \times W}$ be the flow map at a given time, where each entry $x_{ij} \in \mathbb{R}_+$ denotes the volume of the flow in the region r_{ij}.

Definition 1 (Urban Flow Super-Resolution (FSR))**.** *Given a coarse-grained flow map $\mathbf{X}^c$, an upscaling factor N as well as the external factors E (e.g., weather, events, etc.), the FSR task is to learn a model $\mathcal{F}$ mapping $\mathbf{X}^c$ into a fine-grained flow map $\widehat{\mathbf{X}}^f \in \mathbb{R}_+^{NH \times NW}$:*

$$\widehat{\mathbf{X}}^f = \mathcal{F}\left(\mathbf{X}^c | E, N; \theta\right), \tag{1}$$

where θ represent all learnable parameters.

Figure 1 illustrates an example of converting a coarse-grained flow map to a fine-grained flow map in the city of Beijing. The coarse-grained flow map (20km×20km) consists of a total of 32×32 cells, each of which denotes a *superregion*. In the fine-grained flow map, there are 128×128 subregions in total. At the top of Figure 1, a superregion is composed of N^2 subregions ($N = 4$ here). Note that the sum of the flow volume in the N^2 subregions (top right) is equal to the flow volume of the corresponding superregion (top left).

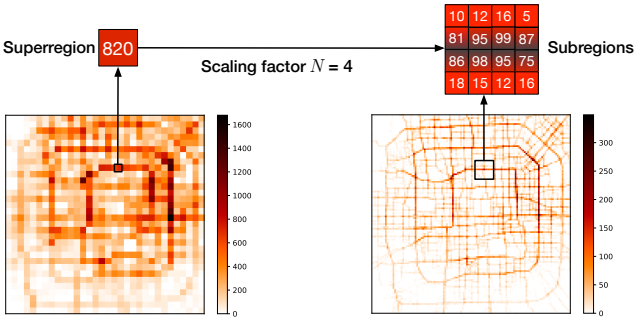


Figure 1: Coarse-grained and fine-grained flow maps in Beijing.

2.1 Architecture

Figure 2 illustrates the overall architecture of FODE, which consists of three main components:

- **FODE block** which extracts spatial correlations among flow regions with the proposed new ODEs.
- **Feature fusion network (FFN)** which combines the FODE blocks and fully connected networks for fusing the influence of external factors.
- **Fine-grained flow inference (FFI)** which leverages an augmented N^2-normalization scheme (AN^2) to estimate the distribution of both flows and external factors for generating the fine-grained flow maps.

2.2 FODE Block

ResNet [He *et al.*, 2016] and its variants have been widely employed for image super-resolution [Ledig *et al.*, 2017; Zhang *et al.*, 2018]. Recently, [Liang *et al.*, 2019] exploited residual block for regional correlation learning and fine-grained urban flow inference, which share the same idea of image SR. Suppose we obtain the high-dimensional hidden state $\mathbf{z}_0$ from the coarse-grained flow $\mathbf{X}^c$ by some convolutional layers. The residual block then transforms the hidden states $\mathbf{z}_n$ according to:

$$\mathbf{z}_{n+1} = \mathbf{z}_n + f\left(\mathbf{z}_n; \theta\right). \tag{2}$$

While achieving promising results on SR tasks, residual networks still confront the problem of intensive computation and require to tune a huge amount of parameters. There is a growing interest in bridging the gap between discrete neural networks and continuous dynamic systems. For example, the iterative updates in the above residual block can be viewed as a discretization of a continuous ODE operator [Lu *et al.*, 2018; Chen *et al.*, 2018], if we take time $t \in \mathcal{T}$ as a continuous variable:

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), t; \theta), \quad \text{where } \mathbf{z}(t_n) = \mathbf{z}_n, \tag{3}$$

$$\mathbf{z}(\mathcal{T}) = \mathbf{z}(0) + \int_0^{\mathcal{T}} f(\mathbf{z}(t), t; \theta) dt. \tag{4}$$

Solving the above ODEs requires computing the gradients through backpropagation, which would be memory prohibitive if the time is reduced into infinite steps. In principle, it requires $\mathcal{O}(\mathcal{N})$ cost to store the all $\mathcal{N}$ intermediate activations. NODE [Chen *et al.*, 2018] proposes to address this problem with the adjoint method. Considering a loss function $\mathbf{L}$, an additional state referred to as the adjoint $\mathbf{a}(t) = \partial \mathbf{L} / \partial \mathbf{z}(t)$ can be used to compute the gradient w.r.t. parameters θ:

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^\intercal \frac{\partial f(\mathbf{z}(t), t; \theta)}{\partial \mathbf{z}(t)} dt, \tag{5}$$

$$\frac{\partial \mathbf{L}}{\partial \theta} = \mathbf{a}(t) \frac{\partial \mathbf{z}(t)}{\partial \theta} = -\int_{\mathcal{T}}^{0} \mathbf{a}(t)^\intercal \frac{\partial f(\mathbf{z}(t), t; \theta)}{\partial \theta} dt, \tag{6}$$

where we only need to store the final state $\mathbf{z}(\mathcal{T})$. Hence this strategy successfully reduces the memory cost. However, it also introduces other problems – the inaccurate gradient and

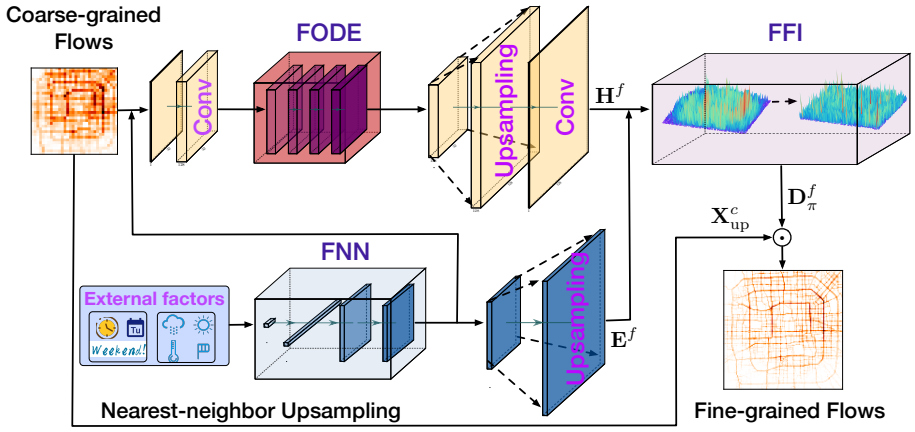


Figure 2: Overview of the proposed FSR architecture.

numerical instability. Taking Eq. (2) for example, we compare the difference between the two calculated gradients: (1) Calculate the gradient w.r.t. $\mathbf{z}_n$ by Eq. (6):

$$\mathbf{a}\left(t_n\right) = \mathbf{a}\left(t_{n+1}\right) + \int_{t_{n+1}}^{t_n} -\mathbf{a}(t)\frac{\partial f(\mathbf{z}(t), t; \theta)}{\partial \mathbf{z}(t)} dt. \tag{7}$$

(2) Calculate the *exact* gradient w.r.t. $\mathbf{z}_n$ by chain rule:

$$\mathbf{a}_n = \mathbf{a}_{n+1} + \mathbf{a}_{n+1}\frac{\partial f(\mathbf{z}_n, \theta)}{\partial \mathbf{z}_n}. \tag{8}$$

As can be seen, the second terms of Eq. (7) and Eq. (8) are different, which, therefore, makes the state $\mathbf{a}\left(t_n\right) \neq \mathbf{a}_n$. In other words, the adjoint method leads to inaccurate and unstable gradient $\partial \mathbf{L}/\partial\theta$ – compared to direct backpropagation that computes correct and stable gradient, but suffers a prohibitive memory cost. ANODE and its variant [Gholami *et al.*, 2019; Zhang *et al.*, 2019] mitigate this problem by splitting a block into time batches and stores in memory a few intermediate states from the forward pass, which, in a sense, is a compromise between time and storage.

In this work, we bridge this gap by introducing an affine coupling layer in the computations of ODEs, to avoid the inaccurate gradient issue while only requiring $\mathcal{O}(1)$ memory in each block. Affine coupling layer referred to a family of neural network whose forward function is a bijective mapping and has been widely used in invertible generative models [Dinh *et al.*, 2015; Dinh *et al.*, 2017; Kingma and Dhariwal, 2018], where the input to a bijective block can be accurately reconstructed from its outputs.

As illustrated in Figure 3, we divide the input $\mathbf{z}_t$ (let $\mathbf{z}_t = \mathbf{z}\left(t\right)$) into two parts of same size $\mathbf{z}_t^a$, $\mathbf{z}_t^b \in \mathbb{R}^{\mathcal{C}/2\times H\times W}$ – where $\mathcal{C}$ is the number of channels. In the forward pass functions in each FODE block with time step size Δt, we have:

$$F: \begin{cases} \mathbf{h}_t^a = \mathbf{z}_t^a \\ \mathbf{h}_t^b = \mathcal{I}\left(\mathbf{z}_t^a\right) \times \mathbf{z}_t^b + \mathcal{J}\left(\mathbf{z}_t^a\right) \end{cases}$$
$$G: \begin{cases} \mathbf{z}_{t+\Delta t}^b = \mathbf{h}_t^b \\ \mathbf{z}_{t+\Delta t}^a = \mathcal{S}\left(\mathbf{h}_t^b\right) \times \mathbf{h}_t^a + \mathcal{K}\left(\mathbf{h}_t^b\right), \end{cases} \tag{9}$$

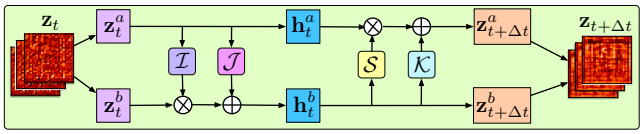


Figure 3: Computational graph for a FODE block.

where $\mathbf{h}_t^a$ and $\mathbf{h}_t^b$ are the intermediate states, $\mathbf{z}_{t+\Delta t}^a$ and $\mathbf{z}_{t+\Delta t}^b$ denote the outputs, $\mathcal{I}$, $\mathcal{J}$, $\mathcal{S}$ and $\mathcal{K}$ are differentiable neural networks. The reverse computations are therefore:

$$\begin{cases} \mathbf{h}_t^b = \mathbf{z}_{t+\Delta t}^b \\ \mathbf{h}_t^a = \left(\mathbf{z}_{t+\Delta t}^a - \mathcal{K}\left(\mathbf{h}_t^b\right)\right) / \mathcal{S}\left(\mathbf{h}_t^b\right) \end{cases}$$
$$\begin{cases} \mathbf{z}_t^a = \mathbf{h}_t^a \\ \mathbf{z}_t^b = \left(\mathbf{h}_t^a - \mathcal{J}\left(\mathbf{z}_t^a\right)\right) / \mathcal{I}\left(\mathbf{z}_t^a\right) \end{cases} \tag{10}$$
$$s.t. \quad 0 < s_i \in \mathcal{S}\left(\mathbf{h}_t^b\right), \quad 0 < i_i \in \mathcal{I}\left(\mathbf{z}_t^a\right).$$

To ensure the reversibility of Eq. (10), all elements in $\mathcal{S}\left(\mathbf{h}_t^b\right)$ and $\mathcal{I}\left(\mathbf{z}_t^a\right)$ need to be greater than zero, which can be met by making a simple transformation for each element e as $\exp(\log e)$, $e \in \mathcal{I}$ or $e \in \mathcal{S}$. Then we have following property which ensures that the affine transformations in the forward pass in Eq. (9) are reversible:

Proposition 1. *The forward pass functions Eq. (9) in the FODE block is reversible as long as each element of $\mathcal{I}(\mathbf{z}_t^a)$ and $\mathcal{S}(\mathbf{h}_t^b)$ is non-zero.*

Proof. Let J_F and J_G be the Jacobians of the transformations in Eq. (9), which can be computed as:

$$J_F\left(\mathbf{z}_t^a, \mathbf{z}_t^b\right) = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \frac{\partial \mathbf{h}_t^b}{\partial \mathbf{z}_t^a} & \text{diag}(\mathcal{I}\left(\mathbf{z}_t^a\right)) \end{bmatrix}, \tag{11}$$

$$J_G\left(\mathbf{h}_t^b, \mathbf{h}_t^a\right) = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \frac{\partial \mathbf{z}_{t+\Delta t}^a}{\partial \mathbf{h}_t^b} & \text{diag}(\mathcal{S}(\mathbf{h}_t^b)) \end{bmatrix}, \tag{12}$$

where $\mathbf{I}$ and $\mathbf{O}$ are identity and zero matrix, respectively, and $\text{diag}(\cdot)$ denotes the diagonal matrix whose diagonal elements correspond to the elements in $\mathcal{I}\left(\mathbf{z}_t^a\right)$ or $\mathcal{S}(\mathbf{h}_t^b)$. Since J_F and J_G are lower triangular matrices, their determinants are

Algorithm 1 Gradient calculation in FODE.

Input: Initial value: $\mathbf{z}_0^a$, $\mathbf{z}_0^b$; parameters θ; integration time $t \in \mathcal{T}$; time step size: Δt
Output: Gradient $\frac{d\mathbf{L}}{d\theta}$;

Forward Pass:
for $t := 0$ **to** $\mathcal{T}$ **do**
  $\mathbf{z}_{t+\Delta t}^a, \mathbf{z}_{t+\Delta t}^b \leftarrow \text{ODESolve}\left(\text{Eq.}(9), [\mathbf{z}_t^a, \mathbf{z}_t^b], \theta\right)$;
  Delete $\mathbf{z}_t^a, \mathbf{z}_t^b$ and all intermediate activations;
  $t \leftarrow t + \Delta t$;
end for
return $\mathbf{z}_{\mathcal{T}}^a$ and $\mathbf{z}_{\mathcal{T}}^b$.
Backward:
for $t := \mathcal{T}$ **to** 0 **do**
  Restore $\hat{\mathbf{z}}_{t-\Delta t}^a$ and $\hat{\mathbf{z}}_{t-\Delta t}^b$ with $\mathbf{z}_t^a$ and $\mathbf{z}_t^b$ by Eq.(10);
  Let $\mathbf{z}_{t-\Delta t}^a = \hat{\mathbf{z}}_{t-\Delta t}^a, \mathbf{z}_{t-\Delta t}^b = \hat{\mathbf{z}}_{t-\Delta t}^b$;
  Compute $\hat{\mathbf{z}}_t^a$ and $\hat{\mathbf{z}}_t^b$ by Eq.(9) ;
  Compute gradients $\bigtriangledown_t$ by Eq. (13) and Eq. (14);
  Update model gradients $\frac{d\mathbf{L}}{d\theta} \leftarrow \frac{d\mathbf{L}}{d\theta} + \bigtriangledown_t$;
  Delete $\mathbf{z}_t^a, \mathbf{z}_t^b$;
  $t \leftarrow t - \Delta t$;
end for
return $\frac{d\mathbf{L}}{d\theta}$.

computed as $|J_F| = \text{diag}(\mathcal{I}(\mathbf{z}_t^a))$ and $|J_G| = \text{diag}(\mathcal{S}(\mathbf{h}_t^b))$. While each element e in $\mathcal{S}(\mathbf{h}_t^b)$ and $\mathcal{I}(\mathbf{z}_t^a)$ is greater than 0 after transformation $\exp(\log e)$, Jacobians J_F and J_G are therefore invertible. □

Due to the reversibility of FODE, in the forward pass with ODE solver, we only save the output ($\mathbf{z}_{t+\Delta t}^a$,$\mathbf{z}_{t+\Delta t}^b$) without the needs to store other variables and intermediate activations. In the backward stage, we first restore the input ($\hat{\mathbf{z}}_t^a$, $\hat{\mathbf{z}}_t^b$) from the output ($\mathbf{z}_{t+\Delta t}^a$, $\mathbf{z}_{t+\Delta t}^b$) by Eq. (10). Then, we perform one-time step forward pass to obtain the output ($\hat{\mathbf{z}}_{t+\Delta t}^a$,$\hat{\mathbf{z}}_{t+\Delta t}^b$), and then calculate corresponding gradients $\frac{\partial[\hat{\mathbf{z}}_{t+\Delta t}^a, \hat{\mathbf{z}}_{t+\Delta t}^b]}{\partial[\hat{\mathbf{z}}_n^a, \hat{\mathbf{z}}_n^b]}$ and $\frac{\partial[\hat{\mathbf{z}}_{t+\Delta t}^a, \hat{\mathbf{z}}_{t+\Delta t}^b]}{\partial[\theta_{\mathcal{I}}, \theta_{\mathcal{J}}, \theta_{\mathcal{S}}, \theta_{\mathcal{K}}]}$. Subsequently, the gradients w.r.t. $\mathbf{z}_t$ and θ of one-time step are computed as:

$$\frac{\partial \mathbf{L}}{\partial [\mathbf{z}_t^a, \mathbf{z}_t^b]} = \frac{\partial \mathbf{L}}{\partial [\mathbf{z}_{t+\Delta t}^a, \mathbf{z}_{t+\Delta t}^b]} \frac{\partial [\hat{\mathbf{z}}_{t+\Delta t}^a, \hat{\mathbf{z}}_{t+\Delta t}^b]}{\partial [\hat{\mathbf{z}}_t^a, \hat{\mathbf{z}}_t^b]}, \tag{13}$$

$$\frac{\partial \mathbf{L}}{\partial [\theta_{\mathcal{I}}, \theta_{\mathcal{J}}, \theta_{\mathcal{S}}, \theta_{\mathcal{K}}]} = \frac{\partial \mathbf{L}}{\partial [\mathbf{z}_{t+\Delta t}^a, \mathbf{z}_{t+\Delta t}^b]} \frac{\partial [\hat{\mathbf{z}}_{t+\Delta t}^a, \hat{\mathbf{z}}_{t+\Delta t}^b]}{\partial [\theta_{\mathcal{I}}, \theta_{\mathcal{J}}, \theta_{\mathcal{S}}, \theta_{\mathcal{K}}]}. \tag{14}$$

The process of calculating accurate gradients in FODE is summarized in Algorithm 1, where ODESolve can be any numerical solutions, e.g., Euler, Runge-Kutta [Butcher and Wanner, 1996] and Dopris Solver [Ascher *et al.*, 1997]. After FODE block, layer normalization [Ba *et al.*, 2016] is employed to normalize the feature maps in the channel. Then we leverage SubPixel blocks [Liang *et al.*, 2019] to upscale the hidden state from coarse-grained to fine-grained with upscaling factor N, which obtains the fine-grained hidden state output $\mathbf{H}^f$ after the convolutional layer.

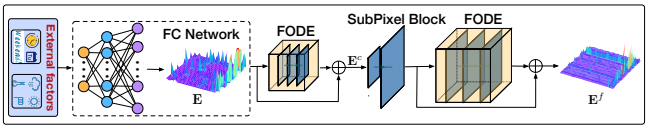


Figure 4: Feature fusion network.

2.3 Feature Fusion Network

It has been demonstrated that external features (e.g., wind speed, temperature, weather and holidays) affect the traffic distribution of the flows [Liang *et al.*, 2019]. Here we also take these factors into consideration for improving performance. In addition to a simple fully connected network (FCN) for feature fusion, we use the proposed FODE blocks to improve the ability of estimating the feature influence. As shown in Figure 4, we obtain the coarse-grained feature map $\mathbf{E}^c \in \mathbb{R}_+^{H \times W}$ and fine-grained feature map $\mathbf{E}^f \in \mathbb{R}_+^{NH \times NW}$ by appending two FODE blocks:

$$\mathbf{E}^c = \text{FODE}(\mathbf{E}) \oplus \mathbf{E}, \tag{15}$$

$$\mathbf{E}^f = \text{FODE}(\mathcal{SP}(\mathbf{E}^c)) \oplus \mathcal{SP}(\mathbf{E}^c), \tag{16}$$

where $\mathcal{SP}$ indicates a SubPixel block used for upsampling.

2.4 Augmented Flow Normalization

One of the main differences between FSR and image SR tasks is that there is a structural constraint in FSR, i.e., the amount of flows in the subregions should equal the flows in the respective superregion:

$$x_{ij}^c = \sum_{i'j'} x_{i'j'}^f \quad s.t. \left\lfloor \frac{i'}{N} \right\rfloor = i, \left\lfloor \frac{j'}{N} \right\rfloor = j, \tag{17}$$

where x_{ij}^c (resp. $x_{i'j'}^f$) denotes the flow volume in a superregion (resp. subregion) of the coarse-grained (resp. fine-grained) grid map. This is a straightforward method that normalizes the flow in the N^2 subregions to meet the constraint, a.k.a. N^2-Normalization [Liang *et al.*, 2019]. However, it ignores the influence of external factors on the subregions, and to address this problem, we propose an augmented N^2 normalization method which takes the distribution of external factors into account when constraining the flow, as illustrated in Figure 5.

More specifically, we replace the $x_{i'j'}^f$ in a subregion with probability value $\alpha_{i',j'}$, and we modify Eq. (17) as follows:

$$x_{i,j}^c = \sum_{i',j'} \alpha_{i',j'} x_{i,j}^c, e_{i,j}^c = \sum_{i',j'} \beta_{i',j'} e_{i,j}^c. \ \alpha, \beta \in \mathbb{R}_+,$$

$$s.t. \sum \alpha_{i',j'} = 1, \sum \beta_{i',j'} = 1, \left\lfloor \frac{i'}{N} \right\rfloor = i, \left\lfloor \frac{j'}{N} \right\rfloor = j, \tag{18}$$

where $e_{i,j}^c$ indicates the degree of influence of complex factors on the region $r_{i,j}$. $\alpha_{i',j'}$ and $\beta_{i',j'}$ denote the proportion of flow and factor influence assigned to the subregions from corresponding superregion, respectively.

Now, we learn the joint distribution of factors' influence $\mathbf{D}_e^f$ and flow $\mathbf{D}_h^f$ to obtain the final flow distribution $\mathbf{D}_\pi^f$. Here we present a **D**istribution **G**ating **M**echanism (**DGM**)

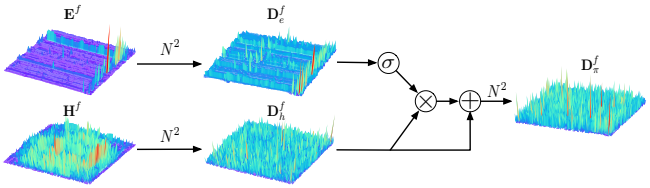


Figure 5: Illustration of AN^2-Normalization.

to explicitly capture the dynamic spatial dependence between the two distributions:

$$\mathbf{D}_\pi^f = N^2\left(\mathbf{D}_h^f \oplus \left(\mathbf{D}_h^f \otimes \sigma\left(\mathbf{D}_e^f\right)\right)\right), \tag{19}$$

where σ denotes distribution gate, $\mathbf{D}_e^f$ and $\mathbf{D}_h^f$ are generated by N^2-Normalization with input $\mathbf{E}^f$ and $\mathbf{H}^f$. AN^2-Normalization not only inherits all the advantages of N^2-Normalization (e.g., parallelizable computation and no extra parameters), but also bridges gaps between external factors and urban flows when normalizing the values of subregions.

Fine-grained flow inference. At this point, we are ready to generate the fine-grained flow $\widehat{\mathbf{X}}^f$ from the learned joint distribution as:

$$\widehat{\mathbf{X}}^f = \mathbf{X}_{\text{up}}^c \odot \mathbf{D}_\pi^f, \tag{20}$$

where $\mathbf{X}_{\text{up}}^c \in \mathbb{R}_+^{NH \times NW}$ is produced by nearest-neighbor upsampling with scaling factor N.

Optimization. Finally, we minimize the mean squared error between the ground truth fine-grained flow map $\mathbf{X}^f$ and the model inferred output $\widehat{\mathbf{X}}^f$:

$$\mathcal{L}(\theta) = \left\|\mathbf{X}^f - \widehat{\mathbf{X}}^f\right\|_2^2 = \left\|\mathbf{X}^f - \mathcal{F}\left(\mathbf{X}^c, |E, N; \theta\right)\right\|_2^2,$$

where θ represents all learnable parameters in our model.

3 Experiments

We now present the details of our experimental evaluations.

3.1 Experimental Settings

Datasets. We evaluate all the methods using two real-world urban flow datasets: (1) **TaxiBJ** [Liang *et al.*, 2019] – a taxi GPS data including taxi flows from July 1, 2014 to October 31, 2014; and (2) **BikeNYC** – collected from an open website[1] which contains data from January 1, 2019 to June 30, 2019. Each dataset contains two sub-datasets: coarse-grained and fine-grained flows. A detailed description of the datasets is shown in Table 1. Note that the scaling factors are different for the two datasets, i.e., $N = 4$ and $N = 2$ for TaxiBJ and BikeNYC, respectively.

Baselines. We compare FODE with the following 10 baselines:

- Mean Partition (**Mean**) evenly distributes the flow volume in each subregion.

Dataset	TaxiBJ	BikeNYC
Time range	7/1/2013-10/31/2013	1/1/2019-3/31/2019
Time interval	30 minutes	1 hour
Coarse-grained size	32×32	40×16
Fine-grained size	128×128	80×32
Upscaling factor (N)	4	2
Latitude range	39.82°N - 39.99°N	40.65°N-40.81°N
Longitude range	116.26°E-116.49°E	74.00°E-74.07°E
External Factors (meteorology, time (e.g., hourofday, dayofweek))		
Temperature / °C	[-24.6,41.0]	\
Wind speed / mph	[0,48.6]	\
Weather conditions	16 types (e.g., Rainy,Sunny)	\
Holidays	18	10

Table 1: Statistics of datasets.

- Historical Average (**HA**) models historical average data to predict the flow in the subregion.
- **SRCNN** [Dong *et al.*, 2015] is a classic model for image SR based on CNNs.
- **ESPCN** [Shi *et al.*, 2016] introduces a sub-pixel convolutional layer for image SR.
- **VDSR** [Kim *et al.*, 2016] employs residual networks to solve the slow convergence and limited representation problems in SR.
- **SRResNet** [Ledig *et al.*, 2017] is a ResNet-based variant of VDSR model, which allows stacking more network layers.
- **OISR** [He *et al.*, 2019] is an ODE-inspired SR model using Runge-Kutta (RK3) method [Butcher and Wanner, 1996] as ODE solver.
- **NODE** [Chen *et al.*, 2018] is a neural ODE method that uses adjoint method for discretization and optimal control of ODEs.
- **ANODE** [Gholami *et al.*, 2019] is an improved version of NODE by introducing checking points for alleviating the incorrect gradient issue.
- **UrbanFM** [Liang *et al.*, 2019] infers fine-grained urban flow with external factors by stacking ResNet-based neural networks.

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).

Implementation details. Adam [Kingma and Ba, 2014] is adopted to train FODE with batch size 16 and learning rate e^{-4}. We leverage Dopri5 numerical method, which can adaptively choose the step size, as ODESovle in FODE. FODE consists of 128 channels and 1 ODE block. We also present a simplified version S-FODE which contains 64 channels while other components are the same as FODE. During training, we halve the learning rate and perform a test on the validation set every 20 epochs. For all image SR models numerical methods (NODE and ANODE), we use N^2-normalization to constrain the inferred flow distribution. We note that the details of other network settings are described in the source-implementation[2].

[1] https://www.citibikenyc.com/system-data

[2] https://github.com/Anewnoob/FODE

Datasets	TaxiBJ			BikeNYC		
Method	**RMSE**	**MAE**	**MAPE**	**RMSE**	**MAE**	**MAPE**
Mean	20.918	12.019	4.469	4.554	1.379	0.678
HA	4.741	2.214	0.332	2.414	0.676	0.216
SRCNN	4.297	2.491	0.714	2.385	0.821	0.433
ESPCN	4.206	2.497	0.732	2.356	0.825	0.441
VDSR	4.159	2.213	0.467	2.344	0.734	0.285
SRResNet	4.164	2.457	0.713	2.355	0.741	0.430
OISR	4.126	2.134	0.421	2.318	0.683	0.246
NODE	4.058	2.125	0.408	2.309	0.671	0.243
ANODE	3.967	2.043	0.351	2.246	0.635	0.217
UrbanFM	3.951	2.011	0.327	2.234	0.627	0.209
S-FODE	**3.941**	**2.001**	**0.322**	**1.951**	**0.517**	**0.129**
FODE	**3.860**	**1.963**	**0.313**	**1.916**	**0.512**	**0.115**

Table 2: Performance comparisons on TaxiBJ and BikeNYC.

Method	$\mathcal{C}$	**#Params (M)**	**memory**	**time**
SRResNet	128	5.5	$\mathcal{O}(\mathcal{L}\mathcal{H})$	$\mathcal{O}(\mathcal{L}\mathcal{H})$
OISR	128	15.7	$\mathcal{O}(\mathcal{L}\mathcal{H})$	$\mathcal{O}(\mathcal{L}\mathcal{H})$
NODE	128	2.1	$\mathcal{O}(\widetilde{\mathcal{L}})$	$\mathcal{O}(\mathcal{N})$
ANODE	128	2.4	$\mathcal{O}(\widetilde{\mathcal{L}})$+$\mathcal{O}(\mathcal{N})$	$\mathcal{O}(\mathcal{N})$
UrbanFM	128	6.2	$\mathcal{O}(\mathcal{L}\mathcal{H})$	$\mathcal{O}(\mathcal{L}\mathcal{H})$
S-FODE	64	**0.7**	$\mathcal{O}(\widetilde{\mathcal{L}})$	$\mathcal{O}(\mathcal{N})$
FODE	128	**2.1**	$\mathcal{O}(\widetilde{\mathcal{L}})$	$\mathcal{O}(\mathcal{N})$

Table 3: Comparisons of parameters and memory cost. $\mathcal{C}$: the number of channels; $\mathcal{L}(\widetilde{\mathcal{L}})$: the number of ResNet (ODE blocks); $\mathcal{H}$: the number of layers in each ResNet; $\mathcal{N}$: the number of function evaluations.

3.2 Results

Overall performance. Table 2 reports the FSR results of all the methods, demonstrating that FODE and its variant outperform all baselines on all metrics across both datasets. Taking TaxiBJ for example, FODE yields 14.9%, 19.1%, and 41.6% improvement on average in terms of RMSE, MAE, and MAPE, respectively. In a relatively smaller area of NYC, FODE achieves larger improvement for inferring the bicycle flow. The performance gain of FODE over baselines demonstrates the effectiveness of continuous-time ODEs, which provides an alternative view of improving fine-grained flow inference.

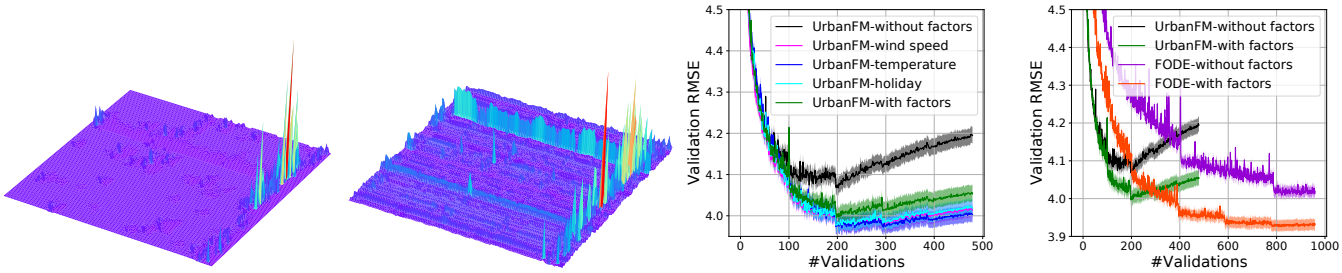
Comparison analysis. Image SR methods are usually not comparable even with N^2 normalization, due to the structural constraints and the influence of external factors in FSR application. This implies that FSR requires specific model design that seamlessly taking constraints and factors into account. As a specifically tailored FSR model, UrbanFM achieves best performance among baselines. However, as a ResNet-based model, it models the urban flow in a discrete manner by stacking deep neural networks, which could be problematic since urban flow inherently can be viewed as a continuous dynamic system. OISR, NODE and ANODE are numerical methods tailored for FSR. OISR uses RK-block as the network structure, which suffers from a huge number of parameters and numerical errors that significantly affect the performance. Additionally, it requires a significant amount of memory for storing intermediate quantities during back-propagation (cf. Table 3). NODE, in contrast, uses adjoint method for solving ODEs, which only needs $\mathcal{O}(\widetilde{\mathcal{L}})$ memory as FODE (note that $\widetilde{\mathcal{L}} = 1$ in FODE), while ANODE requires $\mathcal{O}(\widetilde{\mathcal{L}})$+$\mathcal{O}(\mathcal{N})$ memory. Nevertheless, the dynamics of either the hidden state or the adjoint might be unstable, which incurs inaccurate gradient computation, as we analyzed in previous section. The performance gain of FODE over these numerical methods indicates that our method estimates the gradients more accurately, due to the introduced affine coupling layer while incurring no memory cost.

Memory efficiency. In addition to FSR performance, FODE has a significant memory and parameter efficiency, compared to ResNet-based models. Table 3 outlines the memory cost and parameters required for different methods (we omit other image SR methods due to the similar architectures as SRResNet). In particular, FODE requires only $1/3\times$ parameters of UrbanFM and reduces the memory cost to $\mathcal{O}(1)$. It is worthwhile to note that a simplified version S-FODE contains only 0.7M parameters while still outperforming all the baseline methods on FSR task.

Ability of factor fusion. External factors play important roles and should be carefully considered in FSR models. Figure 6(a) and 6(b) compare the influence of factors learned by UrbanFM and FODE, respectively. UrbanFM uses a FCN to fuse external factors, which is too weak to correlate complex factors with flow distributions. For example, the impact of factors concentrates in a smaller area for UrbanFM – i.e., two main roads are more affected by external factors than other regions. In contrast, FODE estimates the influence of factors by evenly distributing the external influences and is therefore more robust. This can be further verified by the results shown in Figure 6(c) and Figure 6(d), where we observe that FODE consistently converges while UrbanFM, surprisingly, achieves best performance using temperature only, rather than all the factors.

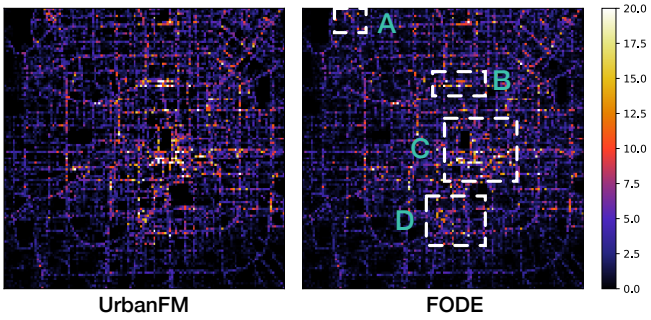
Error analysis. Figure 7(a) shows the inference errors of UrbanFM and FODE on a data sample, where a brighter pixel indicates a larger error. To better visualize the quality of inference, we select four subregions (A, B, C, and D), from which we clearly see that the flow inference by FODE performs better than UrbanFM in crowded areas. Similarly, Figure 7(b) depicts the overall inference error using two different normalization schemes. We observe that in most areas, especially in E, F, G subregions, the distribution generated by AN^2-normalization is closer to the ground-truth, which demonstrates that the proposed AN^2 method is a more effective way of constraining the flow distributions. This improvement can be attributed to jointly modeling the distribution of external factors and urban flows in AN^2, compared to simply constraining the flows in subregions in N^2-normalization.

Accuracy vs. efficiency. Another merit of FODE is that it allows to balance the trade off between the flow inference accuracy and the computational overhead, by varying the number of function evaluations $\mathcal{N}$. As shown in Figure 8, the more function evaluated in the forward pass, the lower the MSE loss. Accordingly, the time required for training the model increases linearly with the number of function evaluations. This result is important since downstream applications

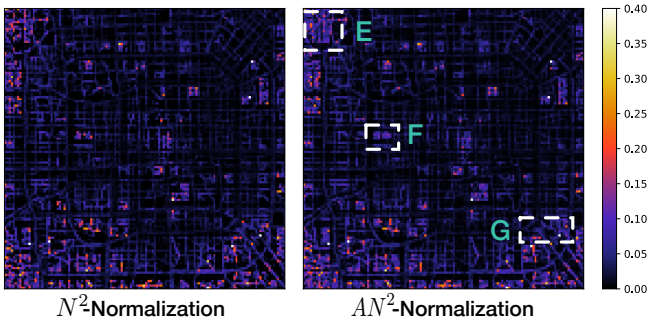


(a) UrbanFM. (b) FODE. (c) UrbanFM fusion capability. (d) FODE fusion capability.

Figure 6: Analysis of external factor fusion.



(a) Comparison of inference errors.



(b) N^2 vs. AN^2.

Figure 7: Inference error visualization.

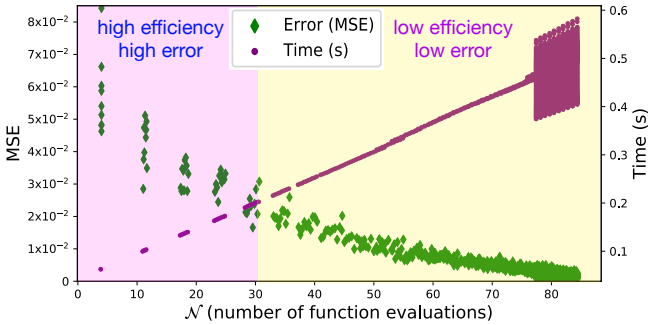


Figure 8: Tradeoff between training time and inference accuracy.

may make flexible solutions by conciliating inference accuracy with computational cost.

4 Conclusion

We proposed a novel method FODE for inferring fine-grained urban flow. FODE learns urban flow distribution through a new ODEs parameterized by affine coupling neural networks alleviating the numerical instability gradient computation issue, which allows for both memory and model parameter savings. In addition, it is capable of explicitly providing more flexible prediction performance by adaptively balancing prediction accuracy and computation overheads. Furthermore, we believe that FODE is a more general ODE-based architecture and can be better exploited for time series prediction or other fine-grained inference tasks such as single image SR [Cai *et al.*, 2019; Wang *et al.*, 2019] and air quality inference [Liu *et al.*, 2019a].

Acknowledgments

This work was supported by National Natural Science Foundation of China (Grant No.61602097) and NSF grant CNS 1646107.

References

[Ascher *et al.*, 1997] Uri M Ascher, Steven J Ruuth, and Raymond J Spiteri. Implicit-explicit runge-kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25(2-3):151–167, 1997.

[Ba *et al.*, 2016] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv: 1607.06450*, 2016.

[Butcher and Wanner, 1996] John Charles Butcher and Gerhard Wanner. Runge-kutta methods: some historical notes. *Applied Numerical Mathematics*, 22(1-3):113–151, 1996.

[Cai *et al.*, 2019] Jianrui Cai, Shuhang Gu, Radu Timofte, Lei Zhang, and et al. Ntire 2019 challenge on real image super-resolution - methods and results. In *CVPR Challenge*, pages 2211–2223, 2019.

[Chen *et al.*, 2018] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *NeuIPS*, pages 6572–6583, 2018.

[De Brouwer *et al.*, 2019] Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes - continuous modeling of sporadically-observed time series. In *NeurIPS*, pages 7377–7388, 2019.

[Dinh *et al.*, 2015] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In *ICLR*, 2015.

[Dinh *et al.*, 2017] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *ICLR*, 2017.

[Dong *et al.*, 2015] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *TPAMI*, 38(2):295–307, 2015.

[E, 2017] Weinan E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

[Gholami *et al.*, 2019] Amir Gholami, Kurt Keutzer, and George Biros. Anode: Unconditionally accurate memory-efficient gradients for neural odes. In *IJCAI*, pages 730–736, 2019.

[Grathwohl *et al.*, 2019] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: free-form continuous dynamics for scalable reversible generative models. In *ICLR*, 2019.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[He *et al.*, 2019] Xiangyu He, Zitao Mo, Peisong Wang, Yang Liu, Mingyuan Yang, and Jian Cheng. Ode-inspired network design for single image super-resolution. In *CVPR*, pages 1732–1741, 2019.

[Heinonen and Lähdesmäki, 2019] Markus Heinonen and Harri Lähdesmäki. ODE^{2}VAE: Deep Generative Second Order ODEs with Bayesian Neural Networks. In *NeurIPS*, pages 13412–13421, 2019.

[Kim *et al.*, 2016] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, pages 1646–1654, 2016.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv: 1412.6980*, 2014.

[Kingma and Dhariwal, 2018] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *NIPS*, pages 10236–10245, 2018.

[Ledig *et al.*, 2017] Christian Ledig, Lucas Theis, Ferenc Huszár, Caballero, and et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, pages 105–114, 2017.

[Liang *et al.*, 2019] Yuxuan Liang, Kun Ouyang, Lin Jing, Sijie Ruan, Ye Liu, Junbo Zhang, David S Rosenblum, and Yu Zheng. Urbanfm: Inferring fine-grained urban flows. In *KDD*, pages 3132–3142, 2019.

[Liu *et al.*, 2019a] Ning Liu, Rui Ma, Yue Wang, and Lin Zhang. Inferring fine-grained air pollution map via a spatiotemporal super-resolution scheme. In *UbiqComp*, pages 498–504, 2019.

[Liu *et al.*, 2019b] Xuanqing Liu, Tesi Xiao, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. Neural SDE: Stabilizing neural ode networks with stochastic noise. *arXiv: 1906.02355v1*, 2019.

[Lu *et al.*, 2018] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *ICML*, pages 3282–3291, 2018.

[Poli *et al.*, 2019] Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. *arXiv: 1911.07532v1*, 2019.

[Rubanova *et al.*, 2019] Yulia Rubanova, Ricky T.Q. Chen, and David Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *NeurIPS*, pages 5321–5331, 2019.

[Shi *et al.*, 2016] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, pages 1874–1883, 2016.

[Wang *et al.*, 2019] Zhihao Wang, Jian Chen, and Steven CH Hoi. Deep learning for image super-resolution: A survey. *arXiv:1902.06068*, 2019.

[Zhang *et al.*, 2017] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *AAAI*, pages 1655–1661, 2017.

[Zhang *et al.*, 2018] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, pages 2472–2481, 2018.

[Zhang *et al.*, 2019] Tianjun Zhang, Zhewei Yao, Amir Gholami, Kurt Keutzer, Joseph Gonzalez, George Biros, and Michael Mahoney. Anodev2: A coupled neural ode evolution framework. In *NeurIPS*, 2019.