Deep Learning for Human Mobility Analytics

-- L6: Learning Spatio-Temporal Trajectory Data

Yuxuan Liang (梁宇轩)

INTR & DSA Thrust

yuxuanliang@hkust-gz.edu.cn

香港科技大学(广州)
THE HONG KONG
UNIVERSITY OF SCIENCE AND
TECHNOLOGY (GUANGZHOU)

Course Reschedule

6	21 Oct	L6: Learning Spatio-Temporal Trajectory Data	✓	
7	28 Oct	L7: Learning Spatio-Temporal Raster Data	✓	Submit Design Report (by 28 Oct @ 23:59)

6:00pm-8:50pm
Nov. 1

Nov. 3!!!

Course Reschedule

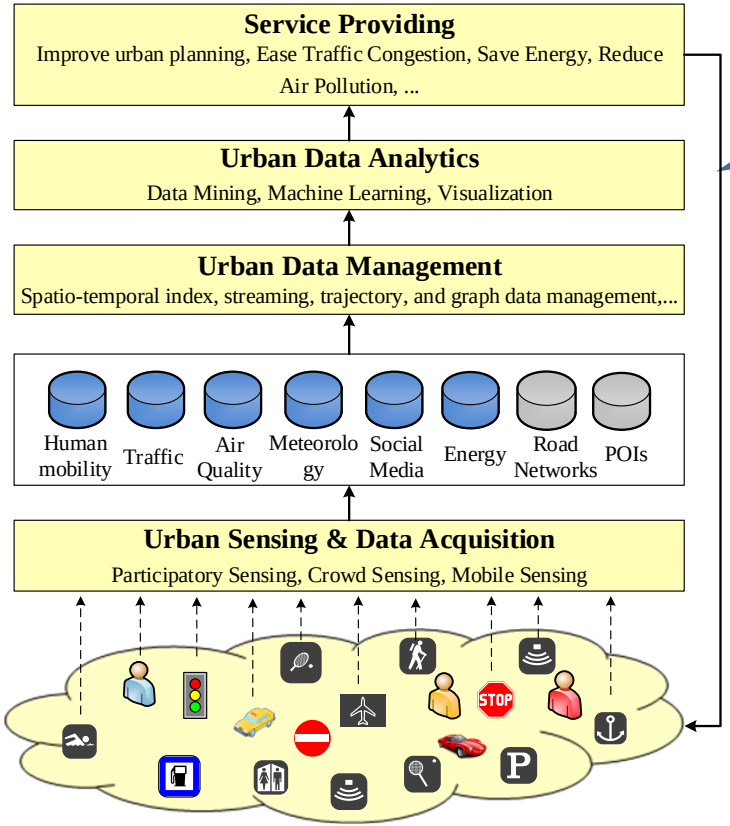
- Week 7: Learning Spatio-Temporal Raster Data
  - **6:00pm-8:50pm, Oct. 28 (today)**
  - The first part is as usual
  - The second part is **project discussion** (no presentation today!)
    - Team 1: Jiahui Liang & Gangyang Zhu
    - Team 2: Jiaxi Hu & Yongzi Yu
    - Team 3: Jingtao He & Pei Liu
    - Team 4: Ruiguo Zhong & Yixuan Wang
    - Team 5: Weilin Ruan & Qiongyan Wang
    - Team 6: Yongkai Gao
    - Team 7: Zhixiong Wang & Tianyu Wei

Course Reschedule

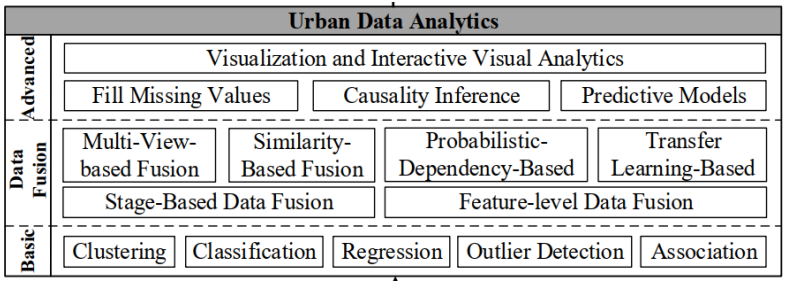
- Week 6: Learning Spatio-Temporal Trajectory Data
  - **6:00pm-8:50pm, Nov. 1**
  - The first part (~70 min) is as usual
  - The second part (~100 min) include presentation from both Week 6 and 7

	1	TrajFormer: Efficient Trajectory Classification with Transformers	Link	CIKM	2022	NUS	Jingtao HE
	2	Stochastic Trajectory Prediction via Motion Indeterminacy Diffusion	Link	CVPR	2022	UCLA	Ruiguo ZHONG
Learning Spatio-Temporal Trajectory Data	3	Modeling Trajectories with Recurrent Neural Networks	Link	IJCAI	2017	Singapore Management University	Zhixiong Wang
	1	Urban regional function guided traffic flow prediction	Link	ıation Sci	2023	Sun Yat-sen University	Jiahui LIANG
Learning Spatio-Temporal Raster Data	2	UrbanFM: Inferring Fine-Grained Urban Flows		KDD	2019	NUS	Gangyong Zhu

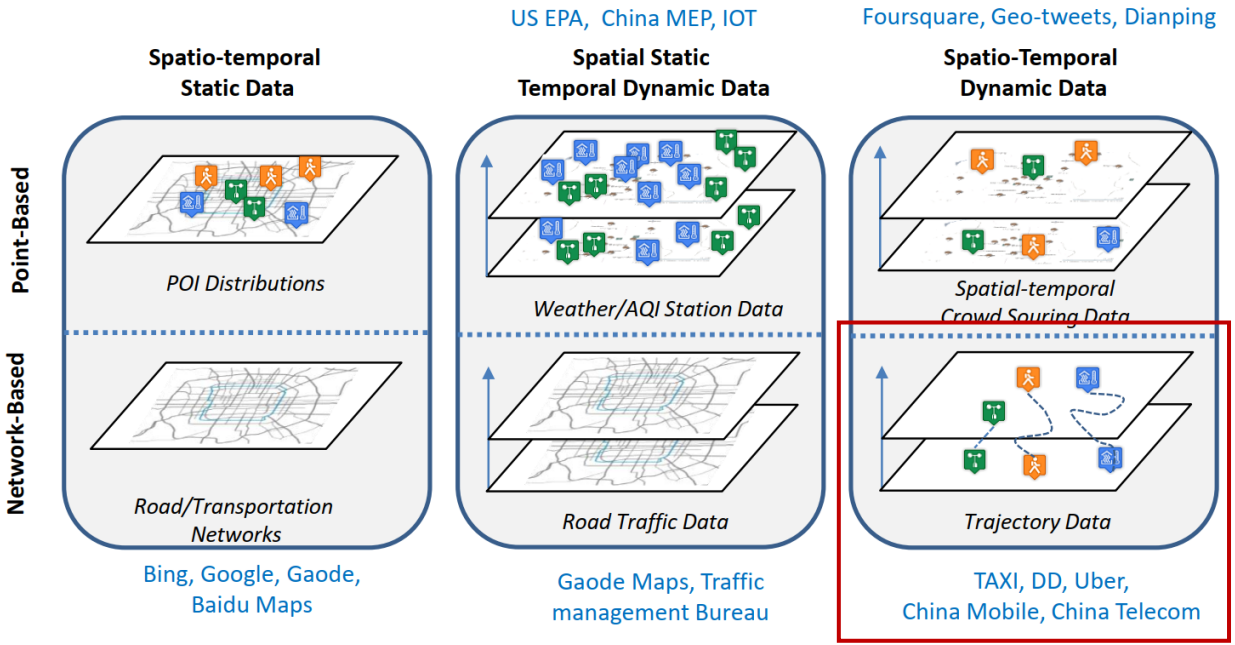
3rd Stage: Urban Data Analytics



- Texts and images → spatio-temporal data
- A single data source → cross-domain data sources
- Separate data mining algorithms → ML + data management
- Visual and interactive data analytics



Learning Spatio-Temporal Trajectories



What is Spatio-Temporal Trajectories?

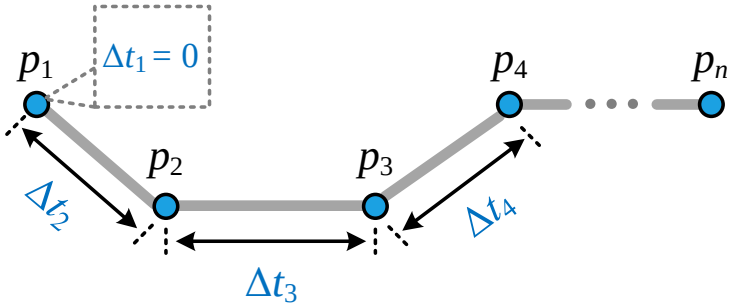
- A ST trajectory is a sequence derived from a moving object in geographical spaces, formulated by a series of chronologically ordered points

$$T = p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n, \quad p_i = (a_i, b_i, t_i)$$

Timestamp

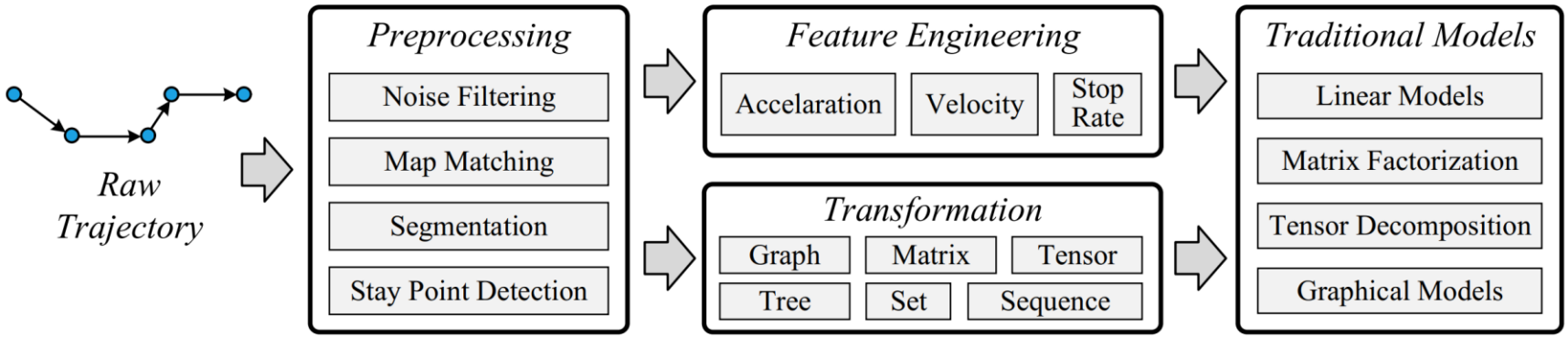
Location (latitude & longitude)

	Latitude,	Longitude,	Time
p_1:	Lat1,	Lng1,	t_1
p_2:	Lat2,	Lng2,	t_2
	……	……	……
p_n:	Latn,	Lngn,	t_n

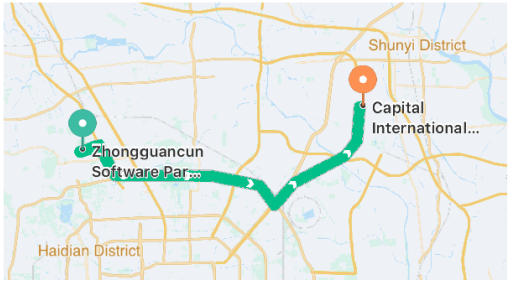


Traditional Approaches

- Including four stages
  - Preprocessing
  - Feature engineering
  - Transformation
  - Machine learning or data mining

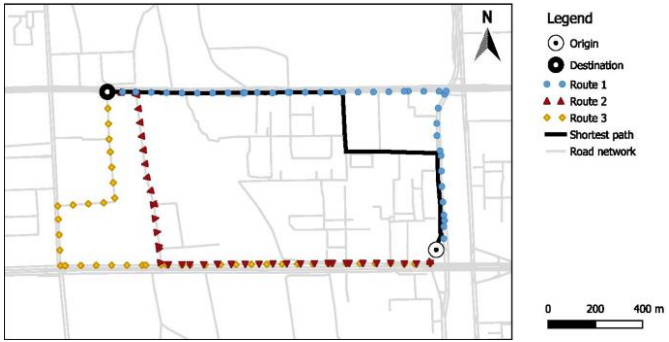


Applications

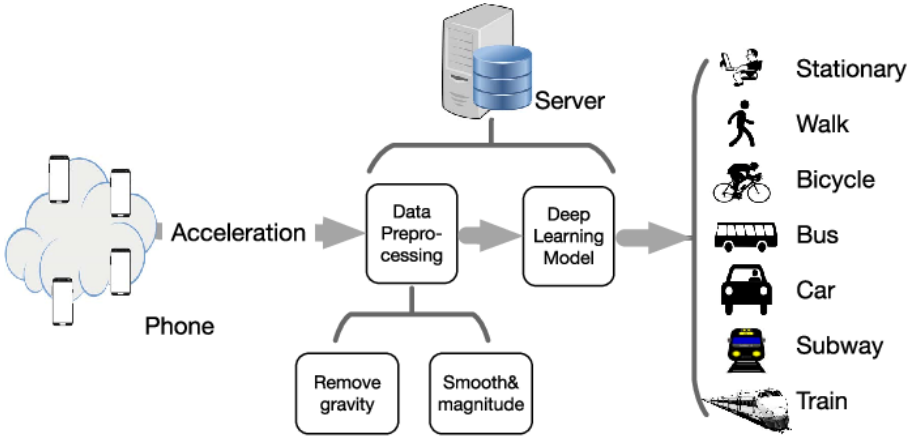


Estimation of travel time

Trajectory similarity computation



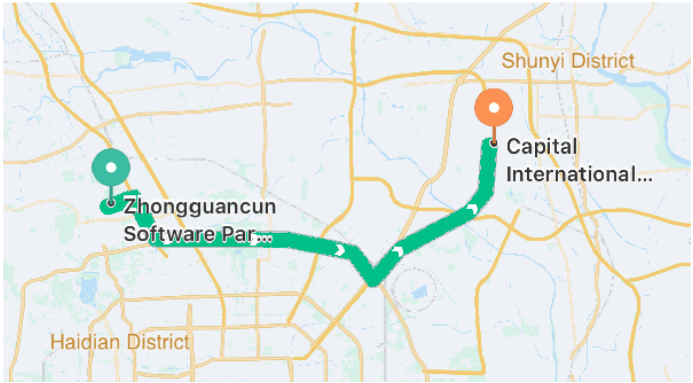
Route recommendation

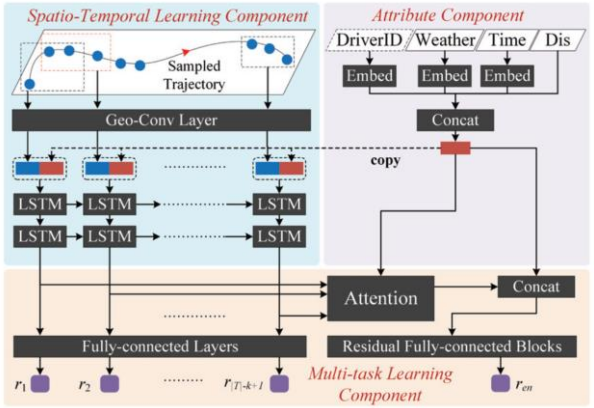


Transportation mode detection

When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks

AAAI 2018

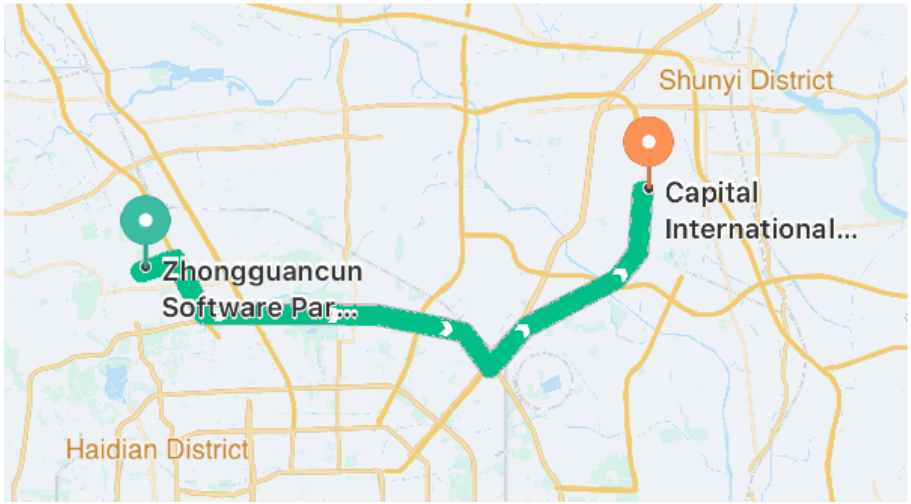




D. Wang et al. When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks. AAAI 2018.

Background

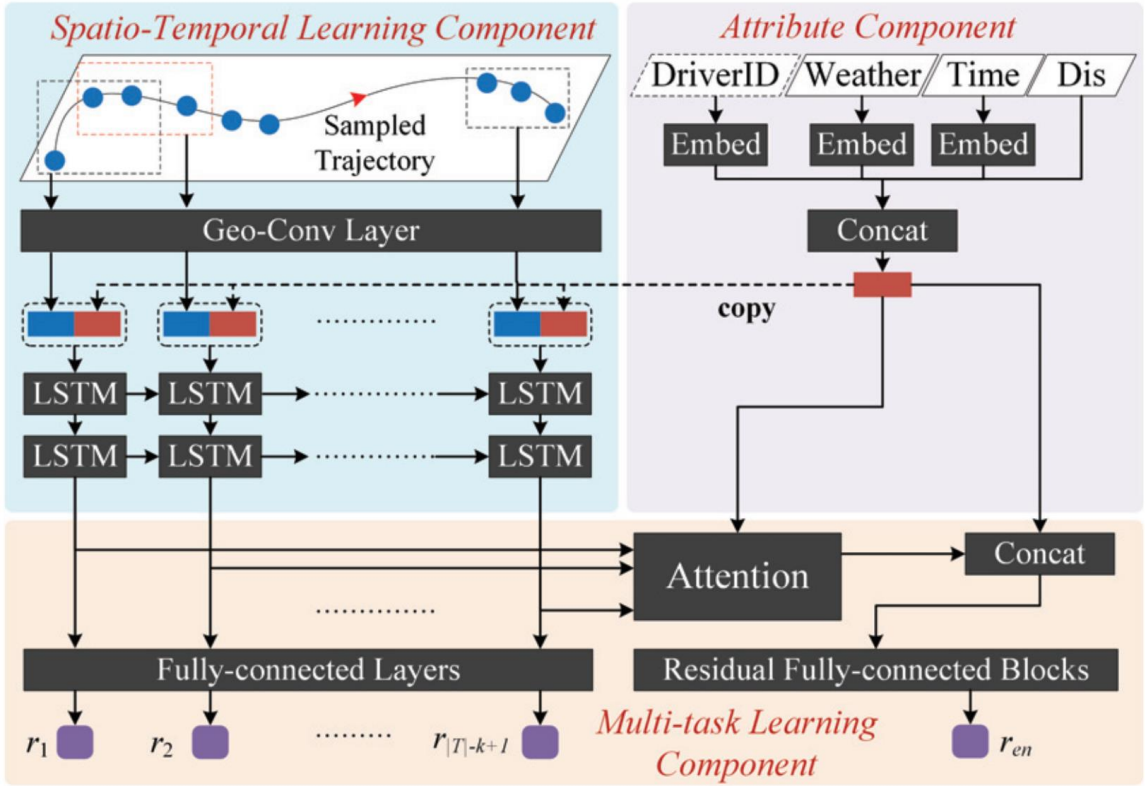
- Estimating the travel time for a given path, which is denoted by a sequence of connected sub-paths, is a fundamental problem in route planning, navigation, and traffic dispatching.



Challenges

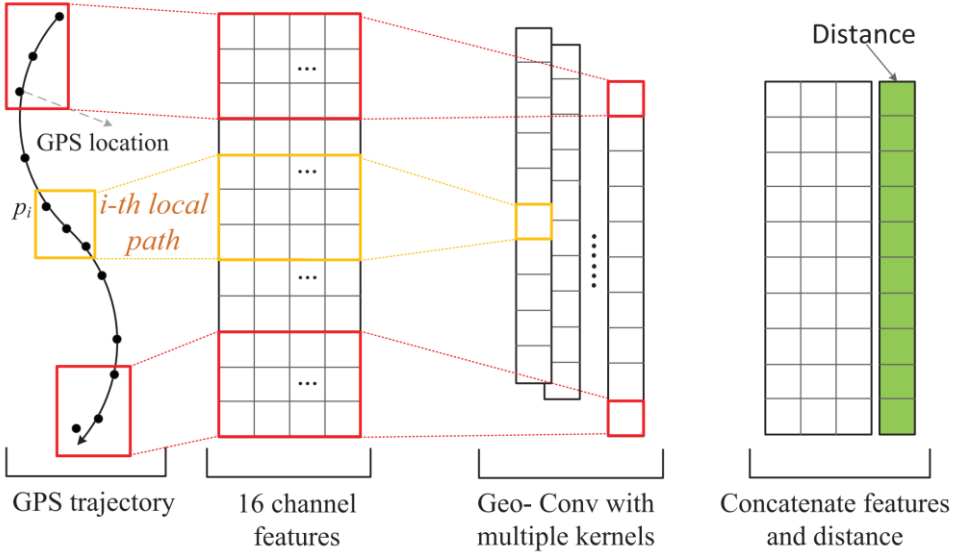
- It is a very challenging problem, affected by diverse complex factors, including
  - spatial correlations
  - temporal dependencies
  - external conditions, e.g. weather, traffic lights.

DeepTTE Framework



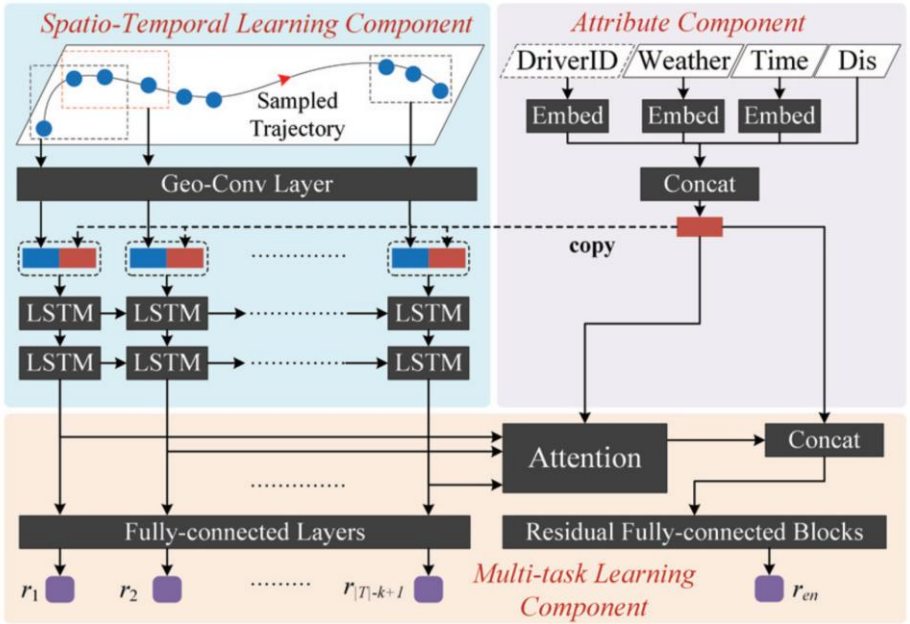
Geo-Conv Layer

- A geo-based convolutional layer that transforms the raw GPS sequence to a series of feature maps
- Capable of capturing the local spatial correlations



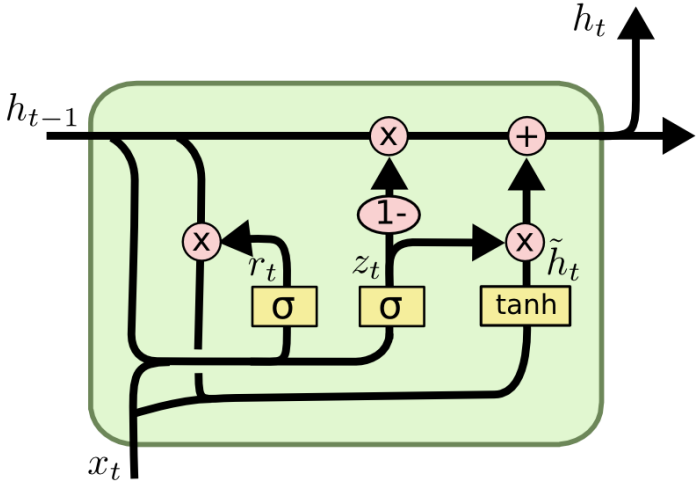
Temporal Encoder

- Learn the temporal dependencies of the obtained feature maps and embeddings from external factors.



LSTM Details

- Learn the temporal dependencies of the obtained feature maps and embeddings from external factors.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

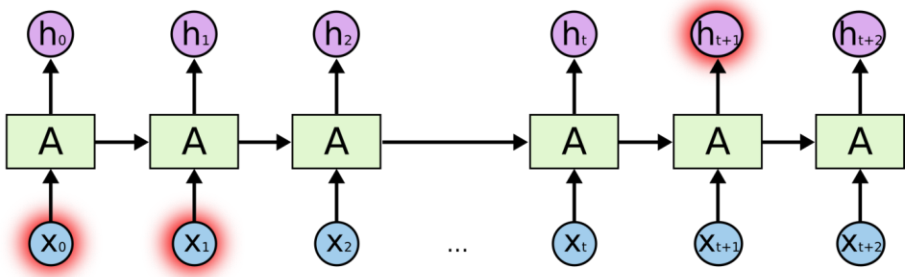
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Vanishing/Exploding Gradient Problem

- Backpropagated errors multiply at each layer, resulting in exponential decay (if derivative is small) or growth (if derivative is large).
- Makes it very difficult train deep networks, or simple recurrent networks over many time steps.

Long Distance Dependencies

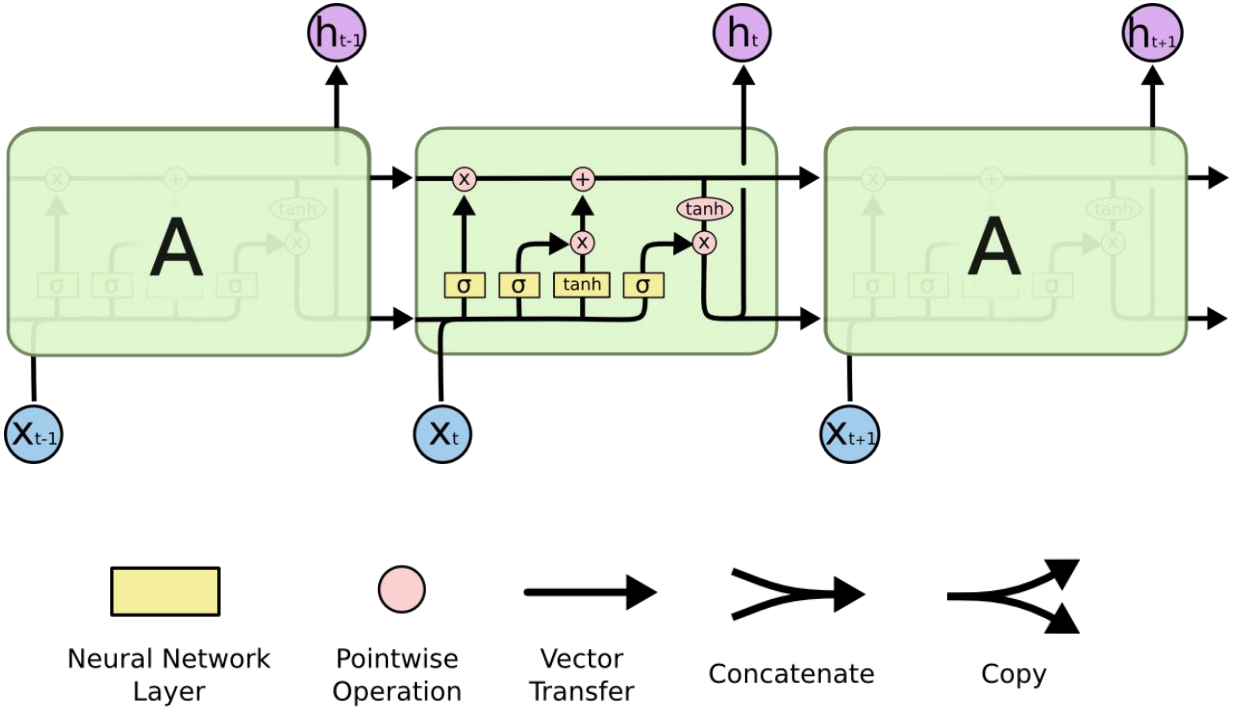
- It is very difficult to train RRNs to retain information over many time steps
- This make is very difficult to learn RRNs that handle long-distance dependencies, such as subject-verb agreement.



Long Short Term Memory

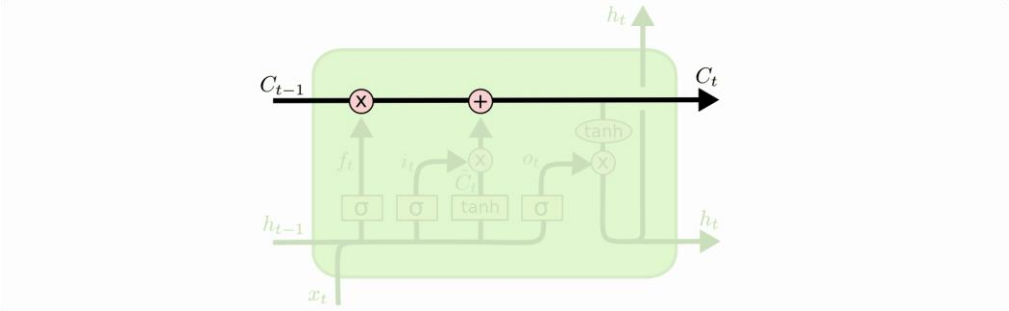
- LSTM networks, add additional gating units in each memory cell.
  - Forget gate
  - Input gate
  - Output gate
- Prevents vanishing/exploding gradient problem and allows network to retain state information over longer periods of time.

LSTM Network Architecture



Cell State

- Maintains a vector C_t that is the same dimensionality as the hidden state, h_t
- Information can be added or deleted from this state vector via the forget and input gates.

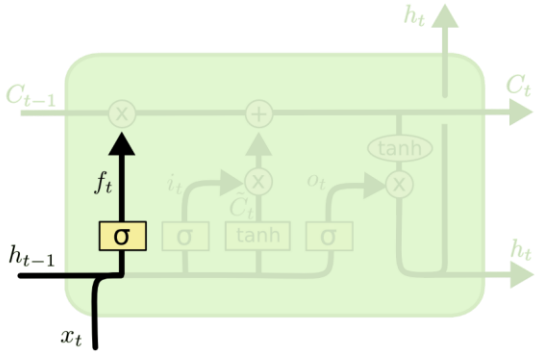


Cell State Example

- Want to remember person & number of a subject noun so that it can be checked to agree with the person & number of verb when it is eventually encountered.
- Forget gate will remove existing information of a prior subject when a new one is encountered.
- Input gate "adds" in the information for the new subject.

Forget Gate

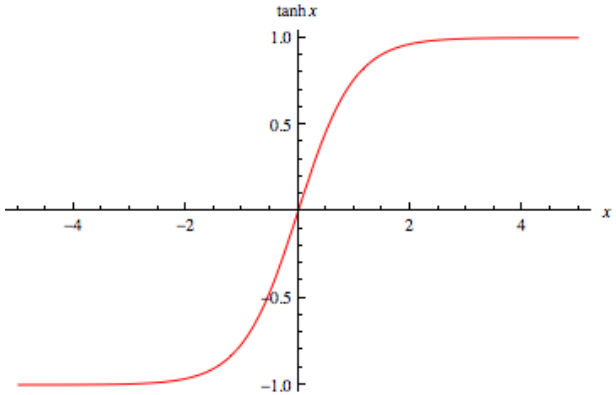
- Forget gate computes a 0-1 value using a logistic sigmoid output function from the input, xt, and the current hidden state, ht:
- Multiplicatively combined with cell state, "forgetting" information where the gate outputs something close to 0.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

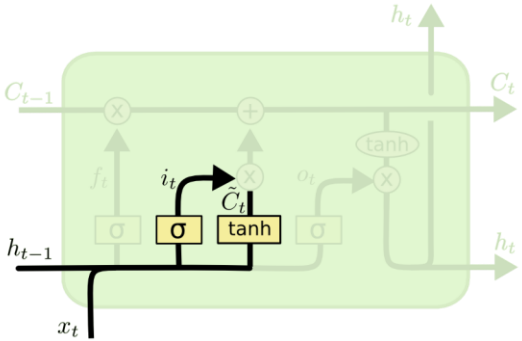
Hyperbolic Tangent Units

- Tanh can be used as an alternative nonlinear function to the sigmoid logistic (0-1) output function.
- Used to produce thresholded output between –1 and 1.



Input Gate

- First, determine which entries in the cell state to update by computing 0-1 sigmoid output.
- Then determine what amount to add/subtract from these entries by computing a tanh output (valued –1 to 1) function of the input and hidden state.

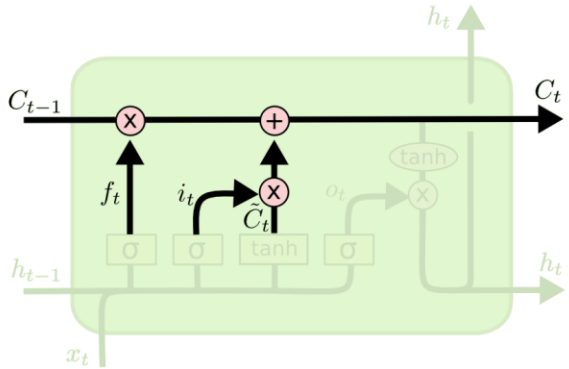


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Updating the Cell State

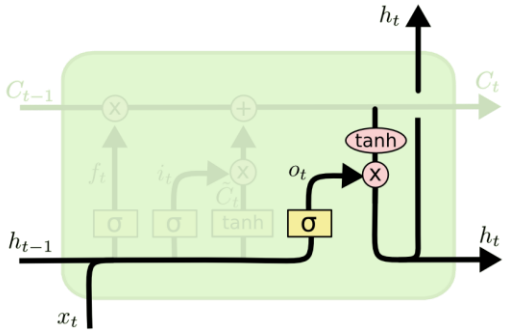
- Cell state is updated by using component-wise vector multiply to "forget" and vector addition to "input" new information.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output Gate

- Hidden state is updated based on a "filtered" version of the cell state, scaled to –1 to 1 using tanh.
- Output gate computes a sigmoid function of the input and current hidden state to determine which elements of the cell state to "output".

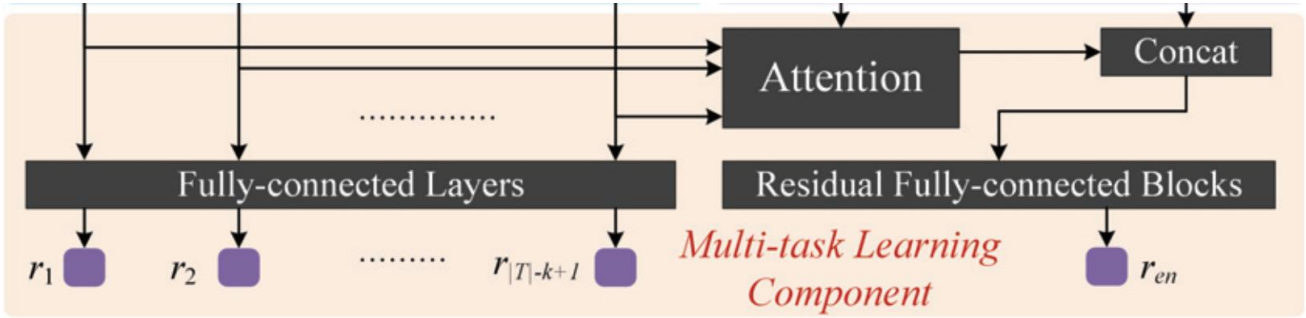


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Multi-Task Learning

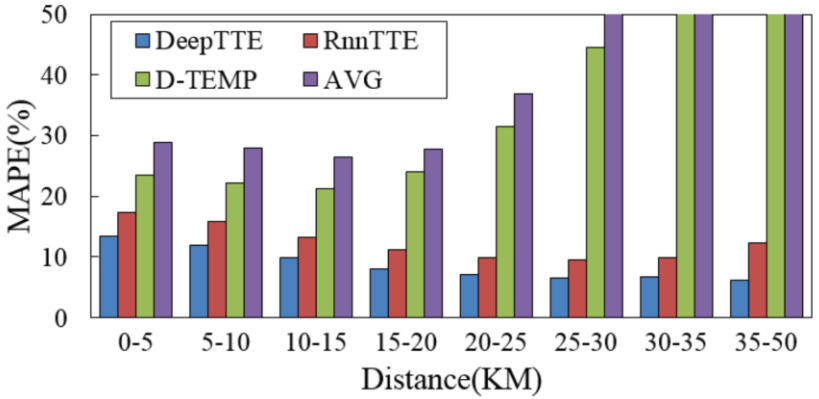
- Two tasks
  - Estimate the local paths
  - Estimate the entire path



Performance Comparison

Data Description We evaluate our model on two large scale datasets:

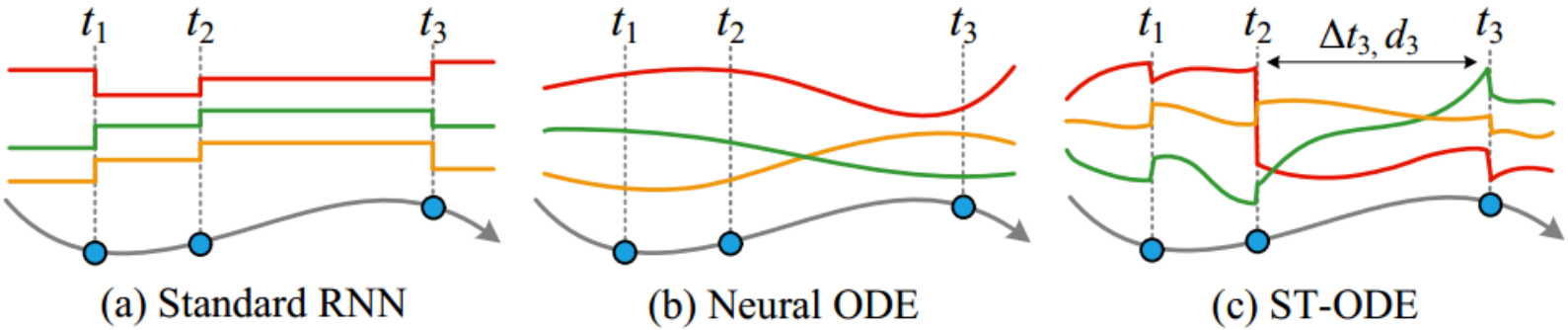
- **Chengdu Dataset**: Chengdu Dataset consists of $9,737,557$ trajectories (1.4 billion GPS records) of 14864 taxis in August 2014 in Chengdu, China. The shortest trajectory contains only 11 GPS records (2km) and the longest trajectory contains 128 GPS records (41km).
- **Beijing Dataset**: Beijing Dataset consists of $3,149,023$ trajectories (0.45 billion GPS records) of 20442 taxis in April 2015 in Beijing, China. The shortest trajectory contains 15 GPS records (3.5km) and the longest trajectory contains 128 GPS records (50km).



	Chengdu			Beijing		
	MAPE (%)	RMSE (sec)	MAE (sec)	MAPE (%)	RMSE (sec)	MAE (sec)
AVG	28.1	533.57	403.71	24.78	703.17	501.23
D-TEMP	22.82	441.50	323.37	19.63	606.76	402.50
GBDT	19.32 ± 0.04	357.09 ± 2.44	266.15 ± 2.24	19.98 ± 0.02	512.96 ± 3.96	393.98 ± 2.99
MlpTTE	16.90 ± 0.06	379.39 ± 1.94	265.47 ± 1.53	23.73 ± 0.14	701.61 ± 1.82	489.54 ± 1.61
RnnTTE	15.65 ± 0.06	358.74 ± 2.02	246.52 ± 1.65	13.73 ± 0.05	408.33 ± 1.83	275.07 ± 1.48
DeepTTE	$\mathbf{11.89 \pm 0.04}$	$\mathbf{282.55 \pm 1.32}$	$\mathbf{186.93 \pm 1.01}$	$\mathbf{10.92 \pm 0.06}$	$\mathbf{329.65 \pm 2.17}$	$\mathbf{218.29 \pm 1.63}$

Modeling Trajectories with Neural Ordinary Differential Equations

IJCAI 2021



(a) Standard RNN (b) Neural ODE (c) ST-ODE

Y. Liang et al. Modeling Trajectories with Neural Ordinary Differential Equations. IJCAI 2021.

Problem Statement

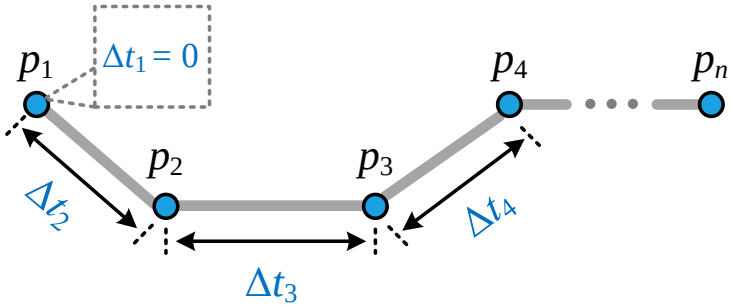
- Given a trajectory, we aim to learn a representation vector of each point
- The representations can be used in downstream tasks, like classification, prediction

Timestamp

$$T = p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n, \quad p_i = (a_i, b_i, t_i)$$

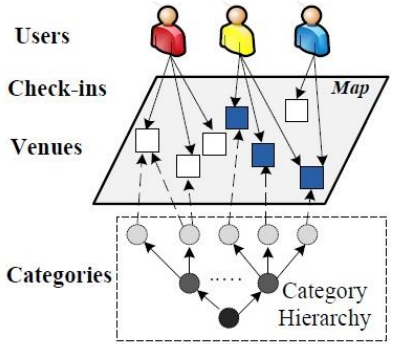
Location (latitude & longitude)

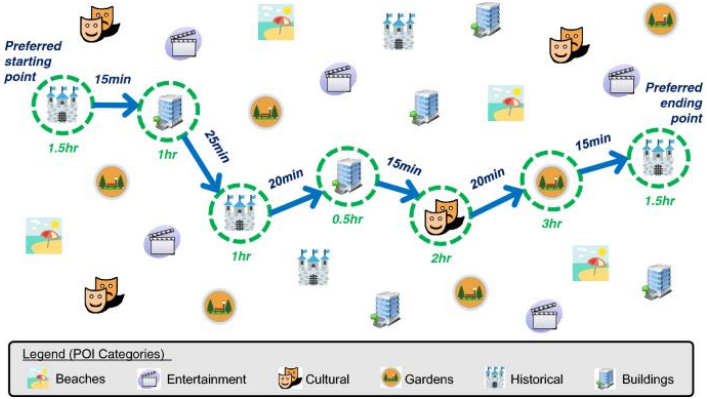
Latitude, Longitude, Time			
p_1:	Lat1,	Lng1,	t_1
p_2:	Lat2,	Lng2,	t_2
	……	……	……
p_n:	Latn,	Lngn,	t_n



Motivation

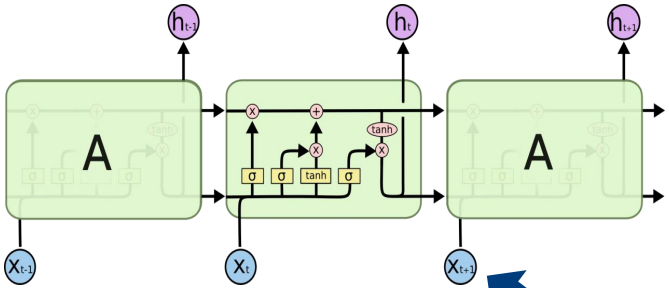
- Modeling spatial trajectories allows us to analytically understand the moving objects and locations, facilitating a broad range of applications
  - Location-based social network
  - Trip recommendation



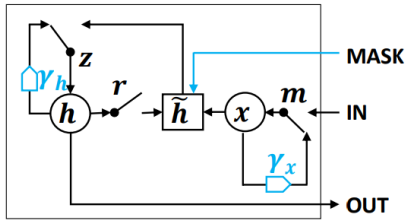


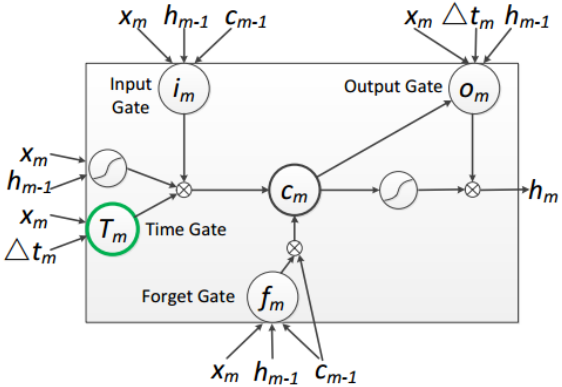
Existing RNN Solutions

- Recurrent Neural Networks (RNNs) are modern tools for modeling trajectories
  - Standard RNNs assume regular time interval. To solve this problem, RNN variants employ
    - A simple trick: concatenating features with the time interval [Liu and Lee, 2017]
    - **Exponential decay on hidden states**: GRU-D [Che et al. 2018]
    - **Temporal gating mechanisms**: TimeLSTM [Zhu *et al.*, 2017], ST-GRU [Liu *et al.*, 2019], etc.



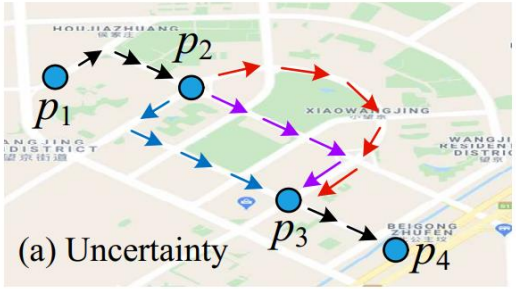
$$p_i = (a_i, b_i, t_i)$$





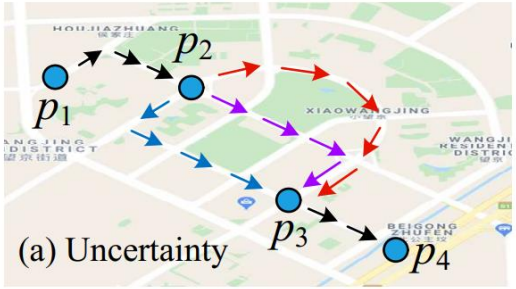
Limitations of RNNs

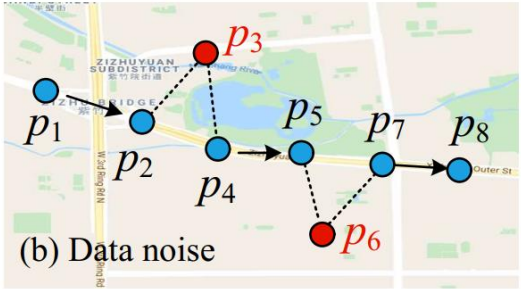
- Uncertainty
  - The time interval between irregular samples in real data can be tens of seconds or even several minutes, e.g., in the GeoLife dataset
  - A larger interval induces larger uncertainty between observations



Limitations of RNNs

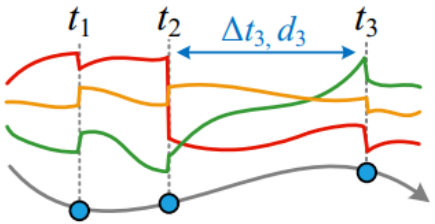
- Uncertainty
  - The time interval between irregular samples in real data can be tens of seconds or even several minutes, e.g., in the GeoLife dataset
  - A larger interval induces larger uncertainty between observations
- Hard to defeat data noise



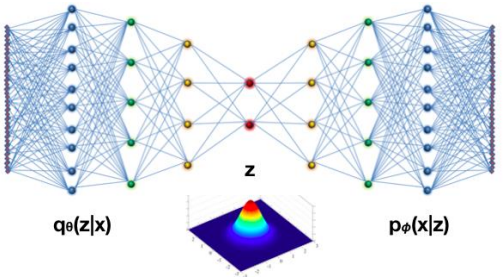


Our Contributions

- We present a novel model based on Neural Ordinary Differential Equations (ODE) for modeling trajectory data
  - **ST-ODE**: capturing the continuous-time dynamics in a principle way compared to RNN counterparts
  - **Posterior approximation**: we enhance ST-ODE with latent variables, strengthening the robustness of our model against data noise by variational inference
- Real-world evaluation on trajectory classification
  - Compared to the state-of-the-art RNN approach, our model can improve the accuracy by 14% ~ 21%

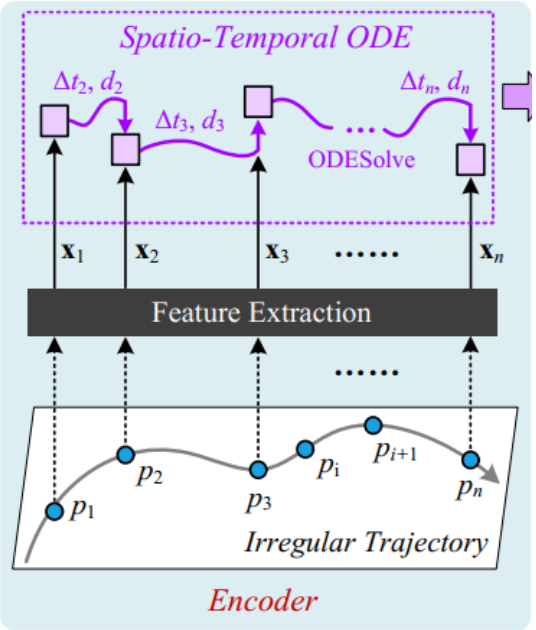


ST-ODE



TrajODE: Continuous Trajectory Modeling

- Considering the domain knowledge of **irregularity**, we present a continuous-time model for modeling human trajectories
- Components of TrajODE
  - ST-ODE encoder

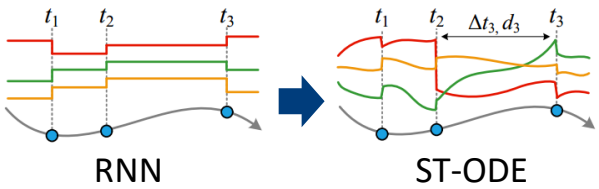


$$\frac{dh(t)}{dt} = f_\theta(h(t), t) \quad \text{where} \quad h(t_0) = h_0$$

Algorithm 1: The ST-ODE model

Input: $\{(\mathbf{x}_i, t_i, \Delta t_i, d_i)\}_{i=1\ldots n}$
Output: The output state $\{\mathbf{o}_i\}_{i=1\ldots n}$

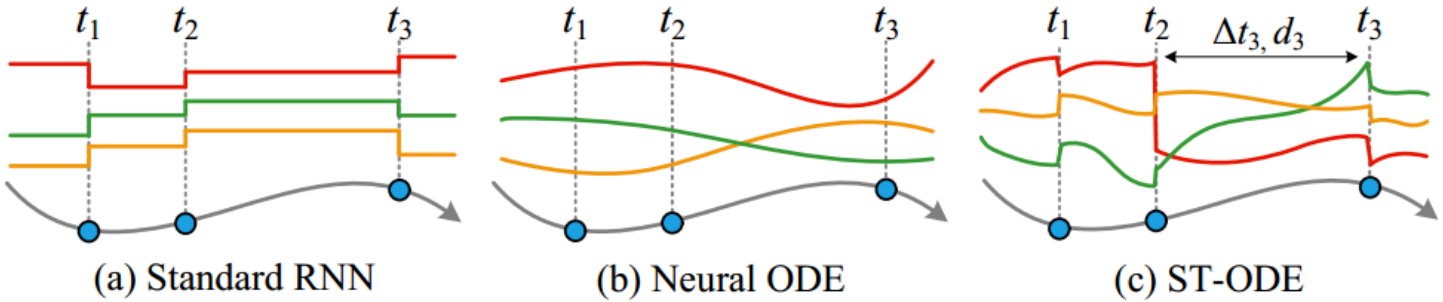
$\mathbf{h}_0 = \mathbf{c}_0 = \mathbf{0}$ ▷ Initialize hidden state and memory cell.
for $i = 1 \ldots n$ **do**
  $(\mathbf{e}_i, \mathbf{c}_i) = \text{LSTMCell}(\mathbf{x}_i, (\mathbf{h}_{i-1}, \mathbf{c}_{i-1}))$
  $\mathbf{h}'_i = \text{ODESolve}\,(f_\psi, \mathbf{h}_{i-1}, (t_{i-1}, t_i))$
  $\mathbf{h}_i = \text{ST-Gate}(\mathbf{e}_i, \mathbf{h}'_i, \Delta t_i, d_i)$
  $\mathbf{o}_i = \mathbf{W}_o\mathbf{h}_i + \mathbf{b}_o$ where $\mathbf{W}_o \in \mathbb{R}^{m\times m}, \mathbf{b}_o \in \mathbb{R}^m$
end



Y. Liang et al. Modeling Trajectories with Neural Ordinary Differential Equations. IJCAI 2021.

Spatio-Temporal ODEs

- Integrating RNNs with Neural ODEs to possess continuous states between observations
- Spatio-temporal gating mechanism: We also consider the time interval Δt or the spatial distance d to renew the hidden states
  - If Δt or d is very large, we believe more in the new observation x_i
- Comparison to RNNs and Neural ODEs



(a) Standard RNN (b) Neural ODE (c) ST-ODE

Spatio-Temporal ODEs

- Algorithm

Algorithm 1: The ST-ODE model

Input: $\{(\mathbf{x}_i, t_i, \Delta t_i, d_i)\}_{i=1\ldots n}$

Output: The output state $\{\mathbf{o}_i\}_{i=1\ldots n}$

$\mathbf{h}_0 = \mathbf{c}_0 = \mathbf{0}$ ▷ Initialize hidden state and memory cell.

for $i = 1 \ldots n$ **do**

  $(\mathbf{e}_i, \mathbf{c}_i) = \text{LSTMCell}(\mathbf{x}_i, (\mathbf{h}_{i-1}, \mathbf{c}_{i-1}))$

  $\mathbf{h}'_i = \text{ODESolve}\,(f_\psi, \mathbf{h}_{i-1}, (t_{i-1}, t_i))$

  $\mathbf{h}_i = \text{ST-Gate}(\mathbf{e}_i, \mathbf{h}'_i, \Delta t_i, d_i)$

  $\mathbf{o}_i = \mathbf{W}_o \mathbf{h}_i + \mathbf{b}_o$ where $\mathbf{W}_o \in \mathbb{R}^{m \times m}, \mathbf{b}_o \in \mathbb{R}^m$

end

Spatio-Temporal ODEs

- Algorithm

Algorithm 1: The ST-ODE model

Input: $\{(\mathbf{x}_i, t_i, \Delta t_i, d_i)\}_{i=1\ldots n}$
Output: The output state $\{\mathbf{o}_i\}_{i=1\ldots n}$

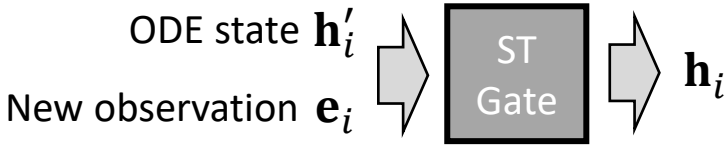
$\mathbf{h}_0 = \mathbf{c}_0 = \mathbf{0}$ ▷ Initialize hidden state and memory cell.
for $i = 1 \ldots n$ **do**
  $(\mathbf{e}_i, \mathbf{c}_i) = \text{LSTMCell}(\mathbf{x}_i, (\mathbf{h}_{i-1}, \mathbf{c}_{i-1}))$
  $\mathbf{h}'_i = \text{ODESolve}\,(f_\psi, \mathbf{h}_{i-1}, (t_{i-1}, t_i))$
  $\mathbf{h}_i = \text{ST-Gate}(\mathbf{e}_i, \mathbf{h}'_i, \Delta t_i, d_i)$
  $\mathbf{o}_i = \mathbf{W}_o \mathbf{h}_i + \mathbf{b}_o$ where $\mathbf{W}_o \in \mathbb{R}^{m \times m}, \mathbf{b}_o \in \mathbb{R}^m$
end

$$\mathbf{h}'_i = \mathbf{h}_{i-1} + \int_{i-1}^{i} f_\psi(\mathbf{h}_t, t)\, dt$$

Adaptive or fixed-sized ODE solver

Spatio-Temporal ODEs

- Algorithm



Algorithm 1: The ST-ODE model

Input: $\{(\mathbf{x}_i, t_i, \Delta t_i, d_i)\}_{i=1\ldots n}$
Output: The output state $\{\mathbf{o}_i\}_{i=1\ldots n}$

$\mathbf{h}_0 = \mathbf{c}_0 = \mathbf{0}$ ▷ Initialize hidden state and memory cell.
for $i = 1 \ldots n$ **do**
  $(\mathbf{e}_i, \mathbf{c}_i) = \text{LSTMCell}(\mathbf{x}_i, (\mathbf{h}_{i-1}, \mathbf{c}_{i-1}))$
  $\mathbf{h}'_i = \text{ODESolve}\,(f_\psi, \mathbf{h}_{i-1}, (t_{i-1}, t_i))$
  $\mathbf{h}_i = \text{ST-Gate}(\mathbf{e}_i, \mathbf{h}'_i, \Delta t_i, d_i)$
  $\mathbf{o}_i = \mathbf{W}_o\mathbf{h}_i + \mathbf{b}_o$ where $\mathbf{W}_o \in \mathbb{R}^{m\times m}, \mathbf{b}_o \in \mathbb{R}^m$
end

Spatio-temporal gating mechanism

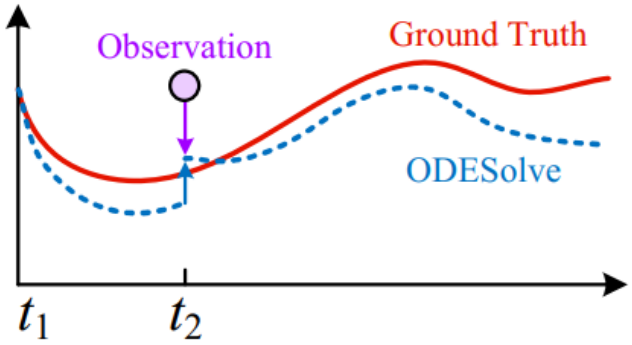
$$\mathbf{h}_i = \mathbf{u}_i \odot \mathbf{h}'_i + (\mathbf{1} - \mathbf{u}_i) \odot \mathbf{e}_i,$$

$$\mathbf{u}_i = \exp\left(-\max\left(\mathbf{0}, \mathbf{W}_u[\Delta t_i, d_i] + \mathbf{b}_u\right)\right)$$

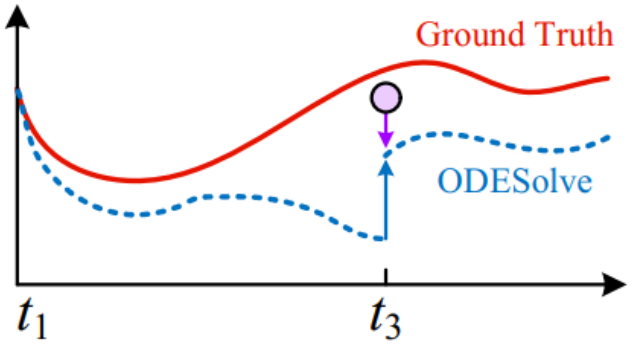
If Δt or d is very large, we believe more in the new observation x_i

Spatio-Temporal ODEs

- ST Gate: a trade-off between ODE state and new observation



(a) Small time interval



(b) Large time interval

Spatio-Temporal ODEs

- Algorithm

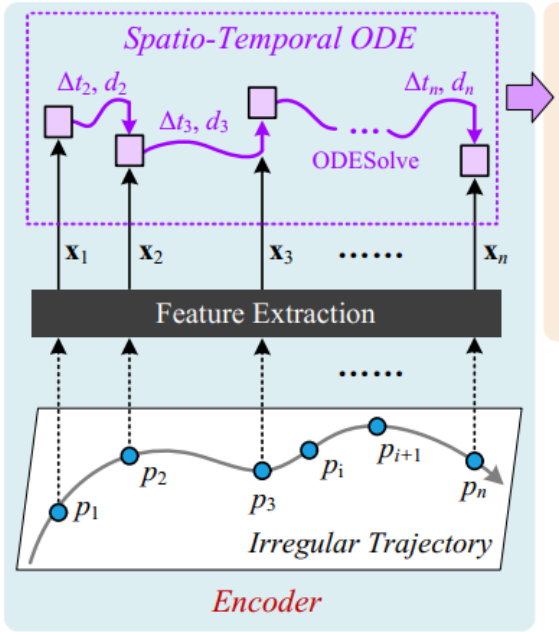
Algorithm 1: The ST-ODE model

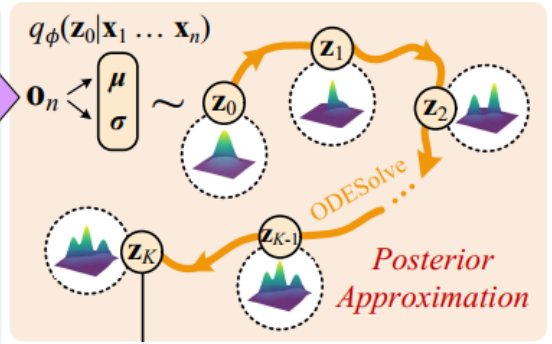
Input: $\{(\mathbf{x}_i, t_i, \Delta t_i, d_i)\}_{i=1\ldots n}$
Output: The output state $\{\mathbf{o}_i\}_{i=1\ldots n}$

$\mathbf{h}_0 = \mathbf{c}_0 = \mathbf{0}$ ▷ Initialize hidden state and memory cell.
for $i = 1 \ldots n$ **do**
  $(\mathbf{e}_i, \mathbf{c}_i) = \text{LSTMCell}(\mathbf{x}_i, (\mathbf{h}_{i-1}, \mathbf{c}_{i-1}))$
  $\mathbf{h}'_i = \text{ODESolve}\,(f_\psi, \mathbf{h}_{i-1}, (t_{i-1}, t_i))$
  $\mathbf{h}_i = \text{ST-Gate}(\mathbf{e}_i, \mathbf{h}'_i, \Delta t_i, d_i)$
  $\mathbf{o}_i = \mathbf{W}_o \mathbf{h}_i + \mathbf{b}_o$ where $\mathbf{W}_o \in \mathbb{R}^{m \times m}, \mathbf{b}_o \in \mathbb{R}^m$ Get outputs
end

TrajODE: Continuous Trajectory Modeling

- Considering the domain knowledge of **irregularity**, we present a continuous-time model for modeling human trajectories
- Components of TrajODE
  - ST-ODE encoder
  - Posterior approximation





Discrete

$$\mathbf{z}_K = g_K \circ g_{K-1} \circ \cdots \circ g_1(\mathbf{z}_0)$$

$$\log P(\mathbf{z}_K) = \log P(\mathbf{z}_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial g_k}{\partial \mathbf{z}_k} \right|$$

Continuous

$$\mathbf{z}_K = \mathbf{z}_0 + \int_0^K g_\omega(\mathbf{z}_k, k)\, dk$$

$$\frac{d \log q(\mathbf{z}_k|\mathbf{X})}{dk} = -\operatorname{tr}\left(\frac{\partial g_\omega}{\partial \mathbf{z}_k}\right)$$

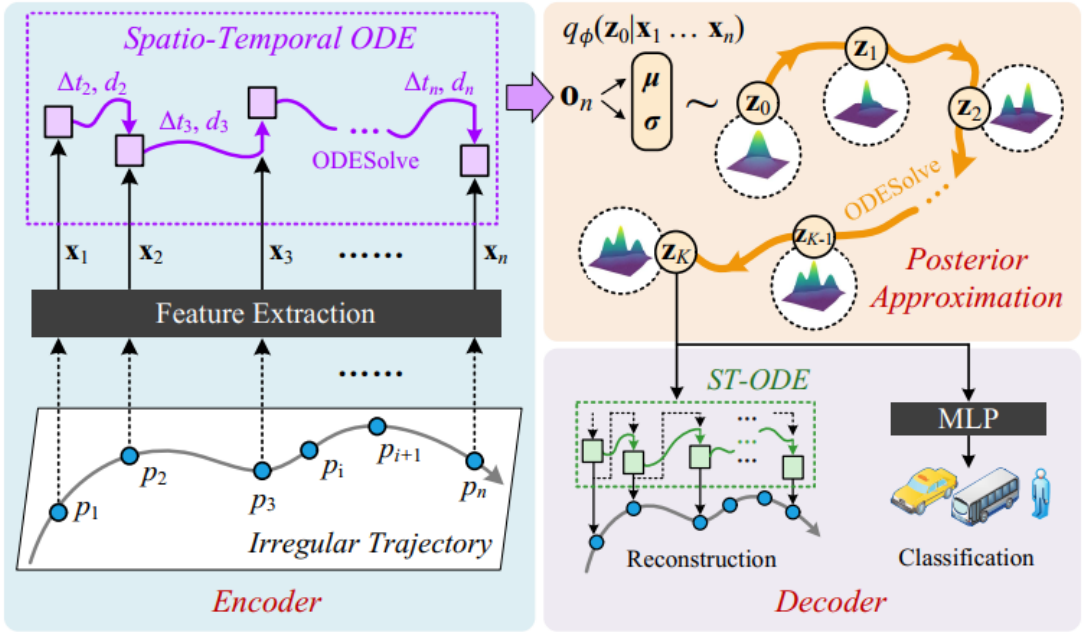
TrajODE: Continuous Trajectory Modeling

- Considering the domain knowledge of **irregularity**, we present a continuous-time model for modeling human trajectories
- Components of TrajODE
  - ST-ODE encoder
  - Posterior approximation
  - ST-ODE decoder

$$\mathcal{L}(\Omega) = \mathcal{L}_{CE}(\hat{\mathbf{y}}, \mathbf{y}) - \gamma \mathcal{L}_{\text{ELBO}}(\theta, \phi),$$

$$\mathcal{L}_{\text{ELBO}}(\theta,\phi) = \underbrace{\mathbb{E}_{q_\phi} \log [p_\theta(\mathbf{X} \mid \mathbf{z}_K)]}_{\text{Reconstruction loss}} + \underbrace{\mathbb{E}_{q_\phi} \log [p_\theta(\mathbf{z}_K)] - \mathbb{E}_{q_\phi} \log [q_\phi(\mathbf{z}_0 \mid \mathbf{X})] + \int_0^K \operatorname{tr}\left(\frac{\partial g_\omega}{\partial \mathbf{z}_k}\right)}_{\text{KL loss}}$$

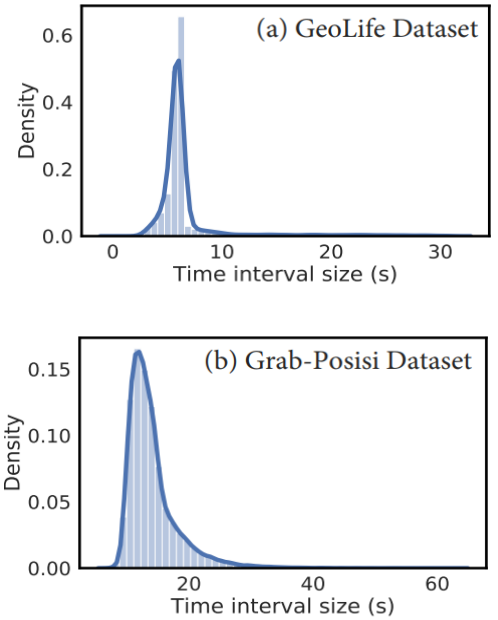
$$-D_{KL}(q_\phi(\mathbf{z}_K \mid \mathbf{X}) \| p_\theta(\mathbf{z}_K))$$



Datasets

- We conduct our experiments over two public datasets

Dataset	GeoLife	Grab-Posisi
# instances	15,659	119,551
Set partition	7:1:2	7:1:2
# classes	4	2
# points	20∼100	20∼100
Time range	5∼10 minutes	5∼30 minutes
Start time	2007-04-01	2019-04-10
End time	2012-08-31	2019-04-13
Label ratio	Walk: 32.64% Bike: 15.31% Bus: 28.96% Car: 23.09%	Car: 50.02% Motorcycle: 49.98% - -



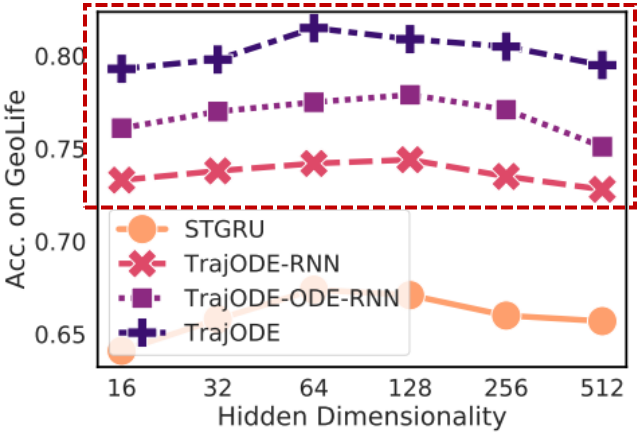
Model Comparison

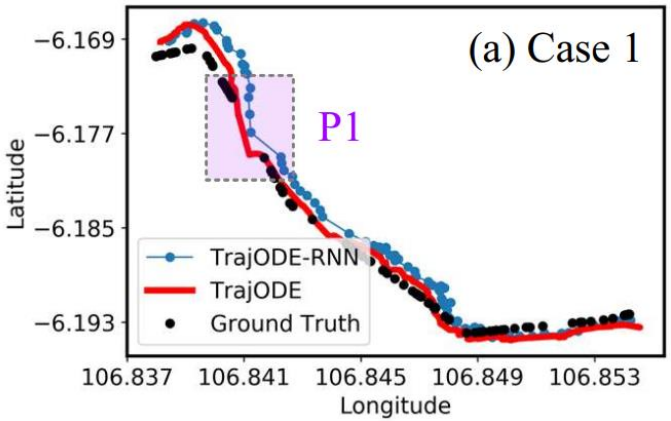
- Evaluated on two real-world datasets

Method	GeoLife		Grab-Posisi	
	Acc	Δ	Acc	Δ
SVM	0.526	-22.0%	0.552	-21.8%
RF	0.592	-12.2%	0.562	-20.4%
RNN	0.614	-8.9%	0.618	-12.5%
BiLSTM	0.629	-6.7%	0.704	-0.3%
TimeLSTM	0.638	-5.3%	0.690	-2.3%
GRU-D	0.625	-7.3%	0.688	-2.6%
STGRU	0.674	-	0.706	-
LatentODE	0.715	+6.1%	0.733	+3.8%
ODE-LSTM	0.645	-4.3%	0.698	-1.1%
TrajODE	**0.815**	**+20.9%**	**0.802**	**+13.6%**

Experiments

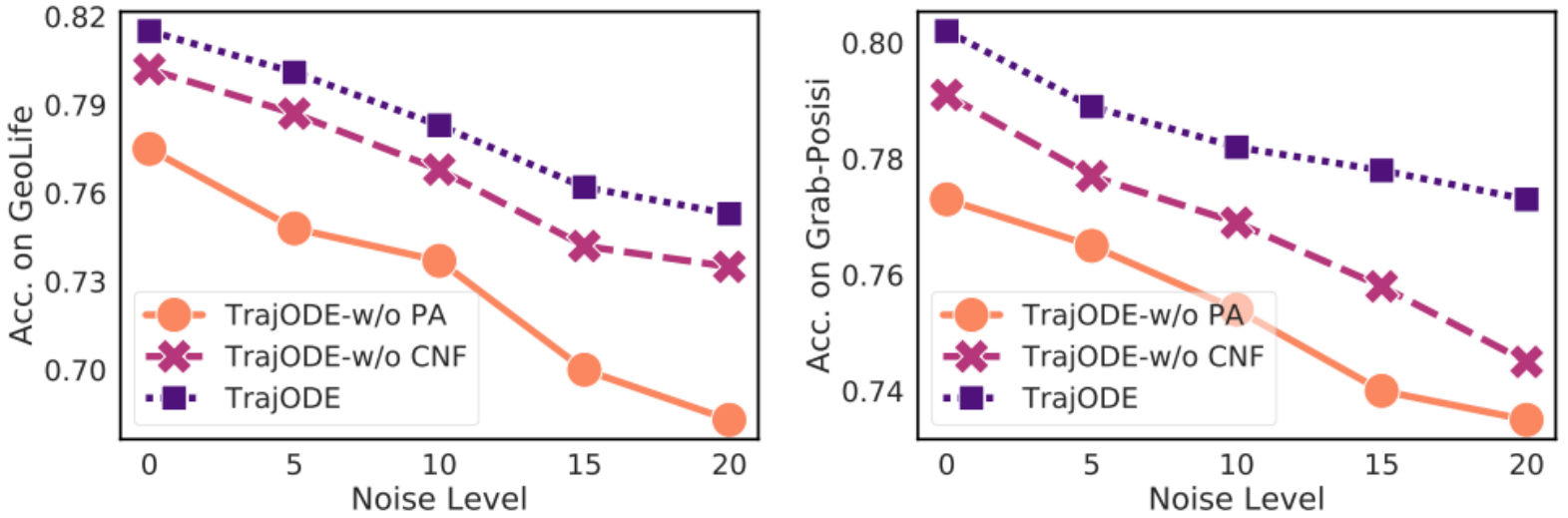
- We evaluate TrajODE on **trajectory classification** over GeoLife dataset
  - Identifying the transportation mode of a trajectory, e.g., walking, driving, taking a bus, etc.
  - Effects of ST-ODE for learning trajectories
    - ST-ODE achieves consistent improvements over RNN and Neural ODEs
    - Visualization on reconstruction





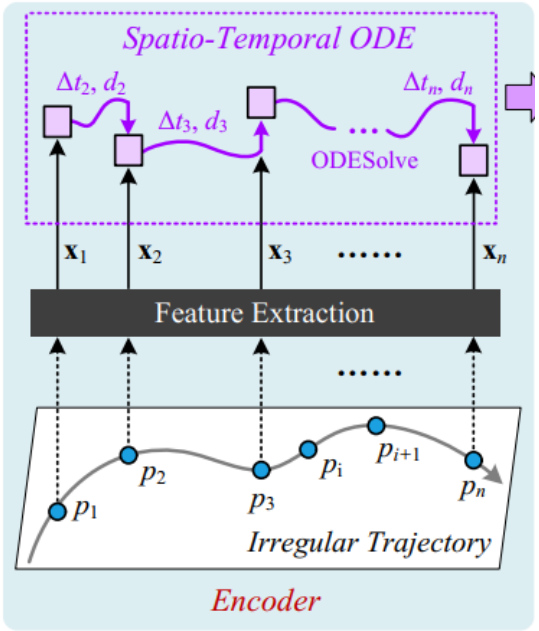
Evaluation on Posterior Approximation

- We use a normal distribution $N(0, \lambda^2)$ to generate an offset for each GPS point, where λ is the standard deviation (in meters) and can be interpreted as the noise level



Limitation of TrajODE

- However, we have noticed one critical issue of TrajODE – **10x slower than RNNs**
  - Bottleneck: the ODE evaluations between two observations

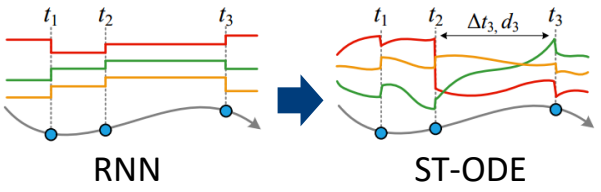


$$\frac{dh(t)}{dt} = f_\theta(h(t), t) \quad \text{where} \quad h(t_0) = h_0$$

Algorithm 1: The ST-ODE model

Input: $\{(\mathbf{x}_i, t_i, \Delta t_i, d_i)\}_{i=1\ldots n}$
Output: The output state $\{\mathbf{o}_i\}_{i=1\ldots n}$

$\mathbf{h}_0 = \mathbf{c}_0 = \mathbf{0}$ ▷ Initialize hidden state and memory cell.
for $i = 1 \ldots n$ **do**
  $(\mathbf{e}_i, \mathbf{c}_i) = \text{LSTMCell}(\mathbf{x}_i, (\mathbf{h}_{i-1}, \mathbf{c}_{i-1}))$
  $\mathbf{h}'_i = \text{ODESolve}\,(f_\psi, \mathbf{h}_{i-1}, (t_{i-1}, t_i))$
  $\mathbf{h}_i = \text{ST-Gate}(\mathbf{e}_i, \mathbf{h}'_i, \Delta t_i, d_i)$
  $\mathbf{o}_i = \mathbf{W}_o \mathbf{h}_i + \mathbf{b}_o$ where $\mathbf{W}_o \in \mathbb{R}^{m \times m}, \mathbf{b}_o \in \mathbb{R}^m$
end



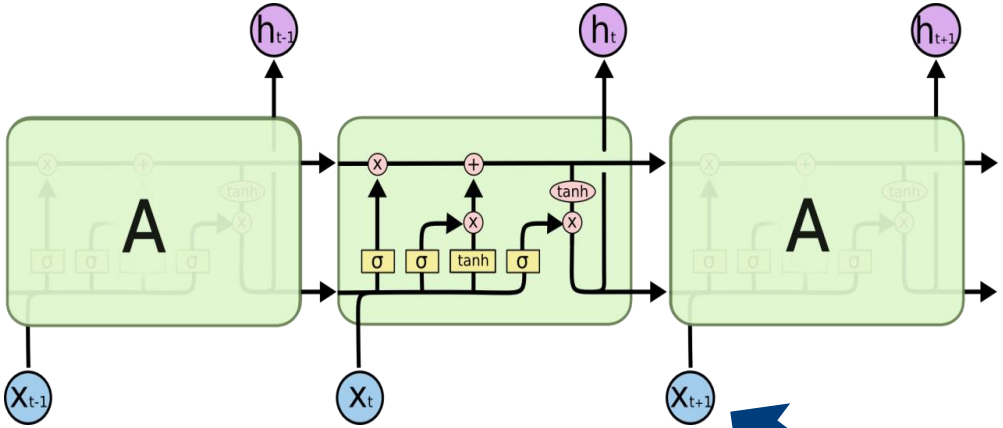
TrajFormer: Efficient Trajectory Classification with Transformers

CIKM 2022

Y. Liang et al. Modeling Trajectories with Neural Ordinary Differential Equations. IJCAI 2021.

Existing RNN Solutions

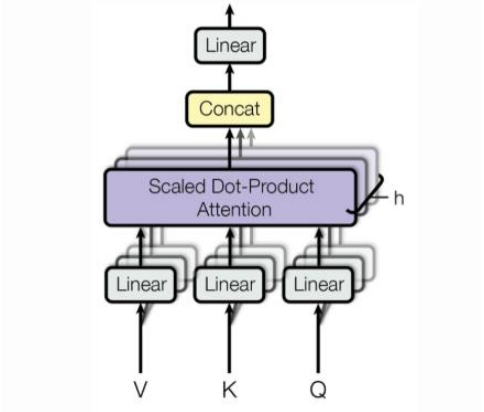
- Recurrent Neural Networks (including Neural ODEs) have long been the dominant models for modeling trajectories, including trajectory classification

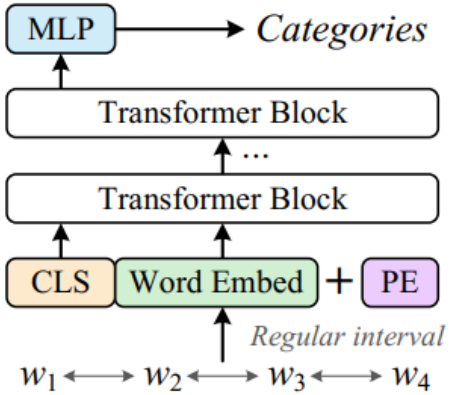


$$p_i = (a_i, b_i, t_i)$$

Transformer Architecture

- Recently, the NLP community has witnessed a shift from RNNs to Transformers
  - Key operation: Multi-head Self-attention mechanism (MSA)
  - Compared to RNNs
    - **Parallel training and inference**
    - Achieving new state-of-the-art results across various NLP tasks

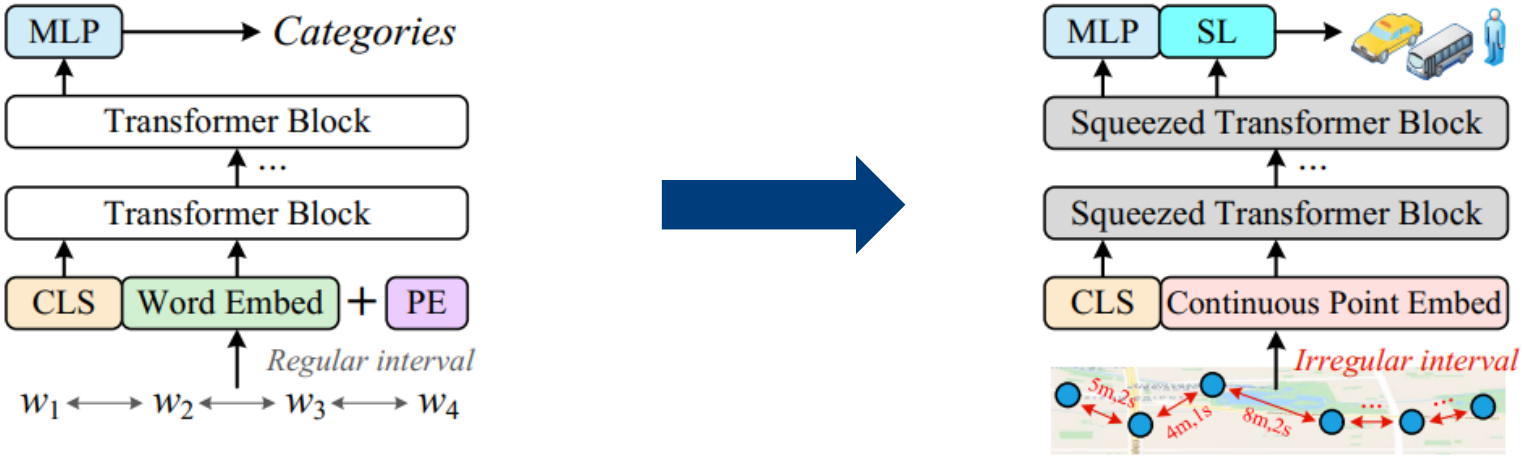




Limitations of Transformers

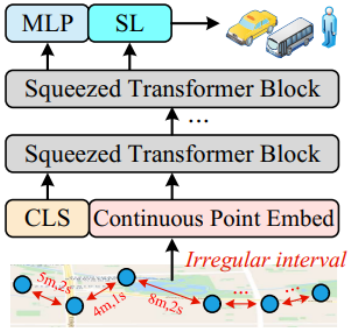
- Two Key factors that limit the adaption of Transformers for trajectory modeling
  - Irregularity: existing position encoding can only address regular time intervals
  - Scalability of multi-head self-attention (MSA)

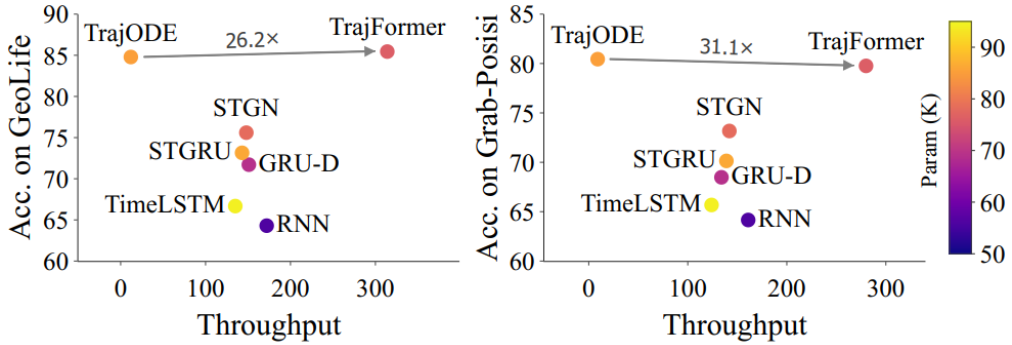
$$\mathbf{X}_h = \text{Softmax}\left(\alpha \mathbf{Q}_h \mathbf{K}_h^\top\right)\mathbf{V}_h,$$



Contributions

- We present a Transformer-based model (TrajFormer) for trajectory modeling
  - Continuous Point Embedding (CPE): we unify the tasks of learning point embeddings and capturing positional information into a single layer.
  - Squeezed Transformer: we present a Squeezed Transformer which can significantly reduce the number of keys and values in attention computations to boost efficiency.
- Experiments validate the superiority of TrajFormer in terms of accuracy & throughput.

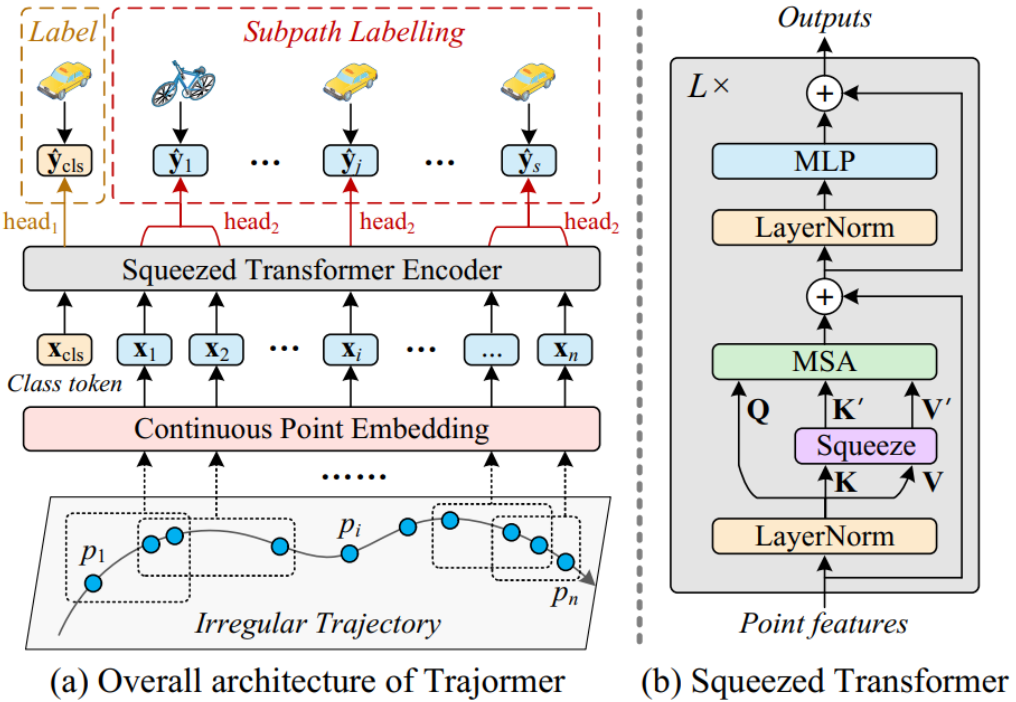




Framework of TrajFormer

- Three components
  - Continuous point embedding
  - Squeezed transformer
  - Prediction & Subpath labeling

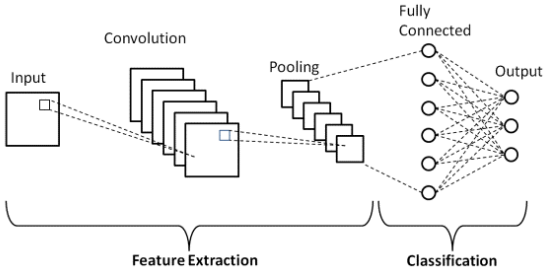
$$\mathcal{L} = \mathcal{H}(\hat{\mathbf{y}}_{cls}, \mathbf{y}_{cls}) + \mathcal{L}_{SL},$$

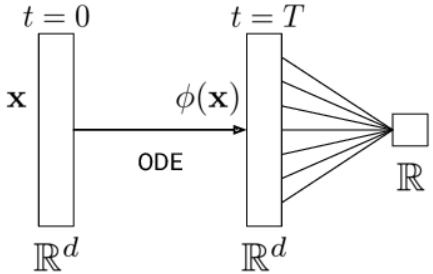


(a) Overall architecture of Trajormer (b) Squeezed Transformer

Continuous Point Embedding

- We argue that a powerful and efficient point embedding mechanism should meet the following four requirements:
  1. Being trained in a **parallel** manner (i.e., no recurrence).
  2. Being able to model **arbitrarily long** trajectories.
  3. Providing **position information** (either absolute or relative) to a certain degree
  4. Being aware of the **spatio-temporal intervals** between a token and its nearby tokens.
- Examples
  - CNNs satisfy the first three demands. ODEs support the last three requirements





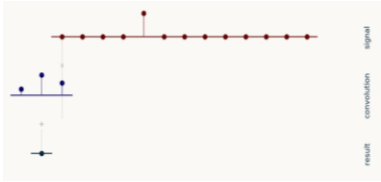
Continuous Point Embedding

- We develop CPE by extending convolutions to be continuous in both temporal and spatial domains

p: input sequence
f: discrete kernel

$$\mathrm{Conv}(x) = (f * p)(x) = \sum_{y=-k}^{k} f(y)p(x+y),$$

1D Convolution



p: input sequence
f: continuous kernel

$$\mathrm{CPE}(x) = (f * p)(x) = \int_{-k}^{k} f(y)p(x+y)dy,$$

Integrating continuity into 1D Convolution

Monte-Carlo integration

Smoothing the kernels

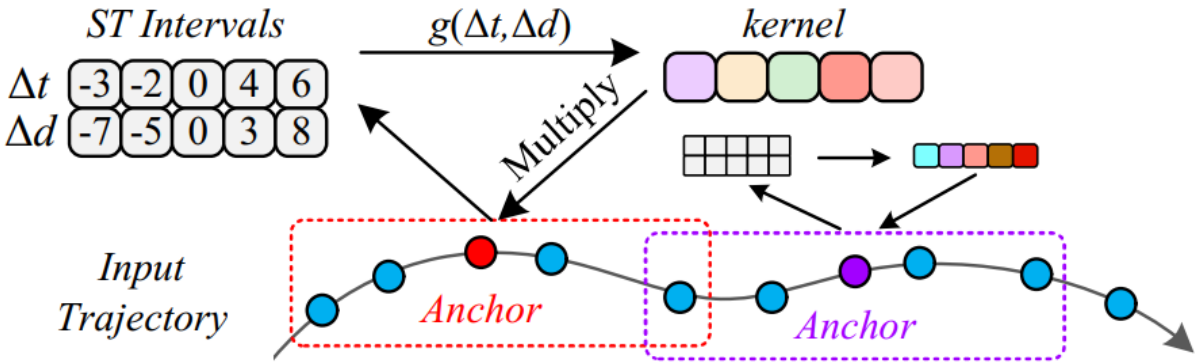
p: input sequence
g: cont. kernel w.r.t intervals
$\mathcal{N}(x)$: the neighbors of x

$$\mathrm{CPE}(x) \approx \sum_{u \in \mathcal{N}(x)} \frac{1}{|\mathcal{N}(x)|} g\left(\underbrace{\Delta t_{u,x}, \Delta d_{u,x}}_{\text{ST intervals}}\right) p(u)$$

Our CPE

Continuous Point Embedding

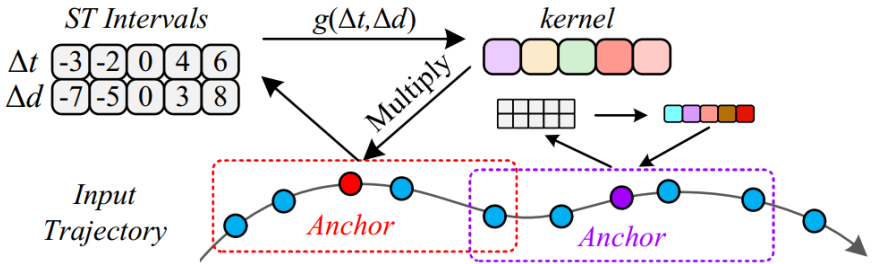
- To help better understand how CPE works, the following figure shows a simple example of the pipeline



Continuous Point Embedding

- How does CPE address the four requirements?
  1. Being trained in a **parallel** manner (i.e., no recurrence).
  2. Being able to model **arbitrarily long** trajectories.
  3. Providing **position information** (either absolute or relative) to a certain degree
  4. Being aware of the **spatio-temporal intervals** between a token and its nearby tokens.

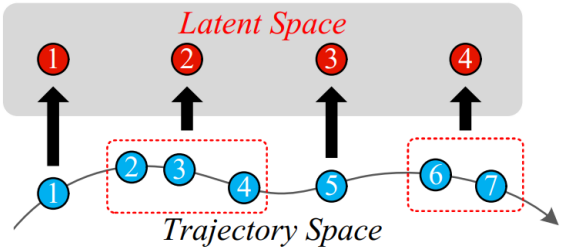
$$\mathrm{CPE}(x) \approx \sum_{u \in \mathcal{N}(x)} \frac{1}{|\mathcal{N}(x)|} g\left(\Delta t_{u,x}, \Delta d_{u,x}\right) p\left(u\right)$$

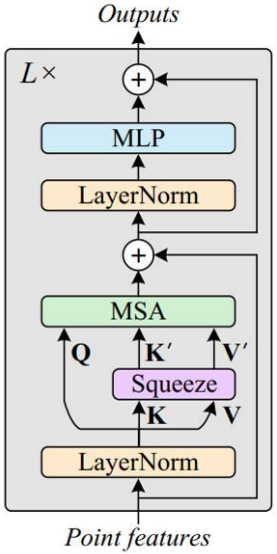


Squeezed Transformer

- Standard transformer
  - Given the embeddings of all points $\mathbf{X} \in \mathbb{R}^{n \times d}$, we produce queries, keys, values for multiple heads
  - The complexity of MSA is $\mathcal{O}(n^2 d)$, making it inefficient to accommodate long trajectories
- We propose a Squeezed Transformer to alleviate this issue
  - Squeezing the trajectory points to the latent space
  - The number of key/values after squeeze is m
  - New complexity $\mathcal{O}(mnd)$

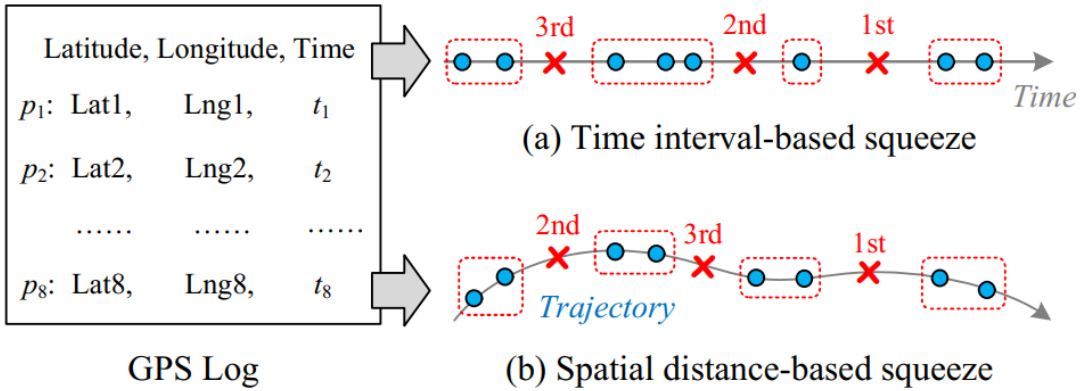
$$\mathbf{X}_h = \text{Softmax}\left(\alpha \mathbf{Q}_h {\mathbf{K}'_h}^\top\right)\mathbf{V}'_h,$$





Squeeze Functions

- The assignment matrix can be either fixed or learnable, which is flexible to a variety of trajectory-based applications
  - Pooling-based squeeze (fixed)
  - Interval-based squeeze (fixed, domain Knowledge-based)
  - Projection-based squeeze (learnable)

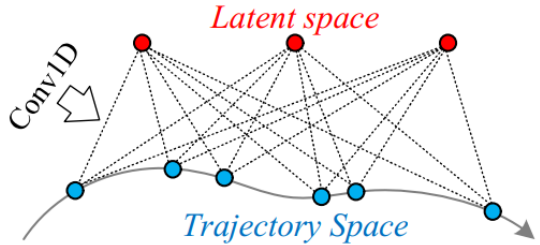


GPS Log

(a) Time interval-based squeeze

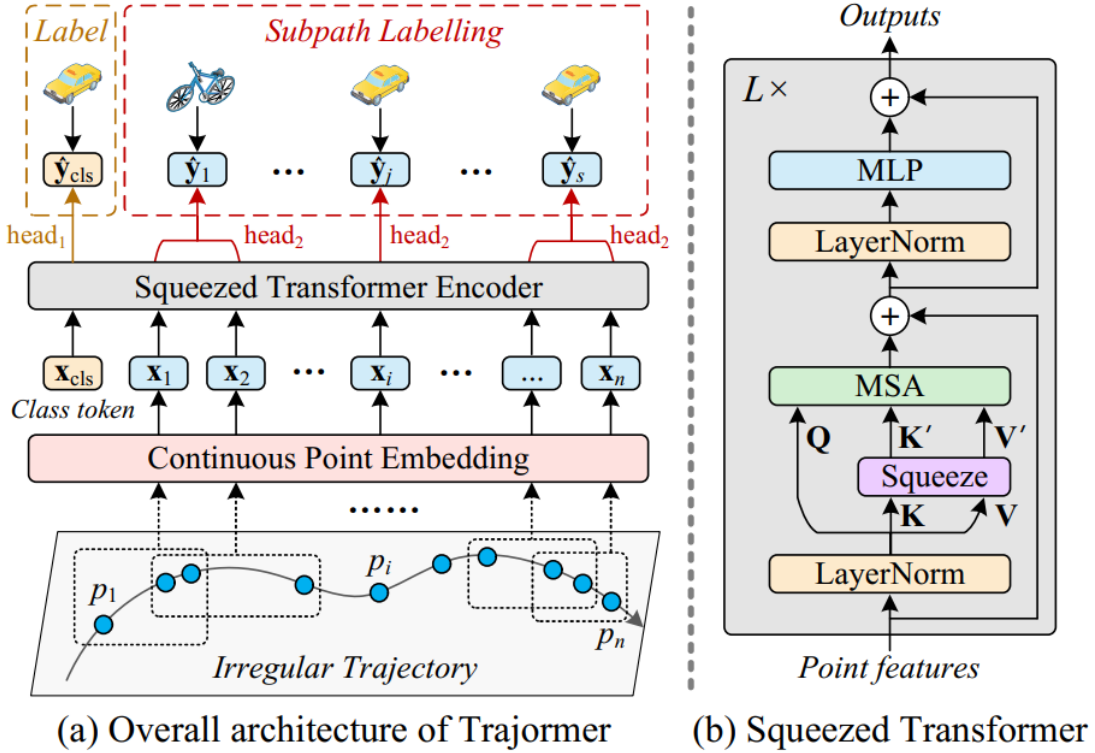
(b) Spatial distance-based squeeze

$$\mathbf{X}_m = \text{Softmax}(\text{Conv1D}(\mathbf{X})), \quad \mathbf{B} = \mathbf{X}_m^\top$$



(c) Projection-based squeeze

Subpath Labeling



(a) Overall architecture of Trajormer (b) Squeezed Transformer

Datasets

- We conduct our experiments over two public datasets
  - GeoLife
  - Grab-Posisi

Dataset	GeoLife	Grab-Posisi
# instances	12,684	319,587
# points	20~100; Avg: 94	20~178; Avg: 84
Time range	5~10 minutes	5~30 minutes
Date	2007-04-01	2019-04-10
End date	2012-08-31	2019-04-16
Class ratio	Walk: 31.77% Bike: 16.39% Bus: 31.82% Car: 20.02%	Car: 50.01% Motorcycle: 49.99% - -
MBR	39.68°N~40.13°N 116.09°E~116.60°E	5.93°S~6.84°S 105.88°E~107.41°E

Model Comparison

- Evaluation metrics
  - Accuracy
  - Throughput per second (TPS)
- Three versions
  - TrajFormer
  - TrajFormer-SL
  - TrajFormer*

Table 2: Model comparison. Param denotes the number of learnable parameters and FLOPs means floating point operations per second per trajectory with 100 points; The magnitudes are Kilo (10^3), Mega (10^3) and % for #Param, FLOPs and Acc. The bold and underlined font mean the best and the second best metrics, respectively

	Method	Param	FLOPs	GeoLife		Grab-Posisi	
				TPS↑	Acc↑	TPS↑	Acc↑
	SVM	-	-	-	49.88	-	51.17
	RF	-	-	-	56.16	-	55.68
RNN/ODE -based	RNN	56	0.32	172	64.30	161	64.16
	TimeLSTM	94	0.78	135	66.68	124	65.68
	GRU-D	69	0.54	151	71.71	134	68.48
	STGN	78	0.62	148	75.60	142	73.17
	STGRU	86	0.70	143	73.15	139	70.14
	TrajODE	85	2.37	12	<u>85.25</u>	9	<u>80.44</u>
Ours	TrajFormer	76	2.18	**314**	**85.45**	**280**	79.76
	TrajFormer-SL	76	2.18	314	85.92	280	80.29
	TrajFormer*	105	2.50	<u>271</u>	84.90	<u>248</u>	**81.08**

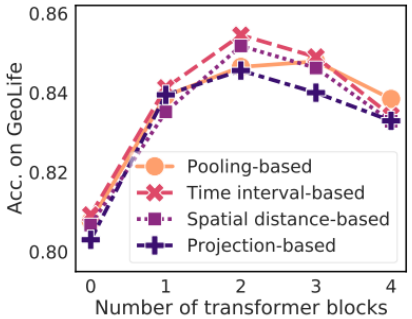
Effects of Continuous Position Embedding

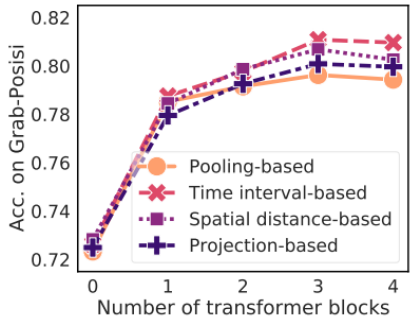
- Variants with different position encodings
  - Base: no position encoding (PE)
  - Absolute PE (APE): encoding 1st, 2nd, 3rd
  - Continuous APE (CAPE): concatenating time intervals in channels
  - Relative PDE (RPE): adding a learnable bias term to the attention weights

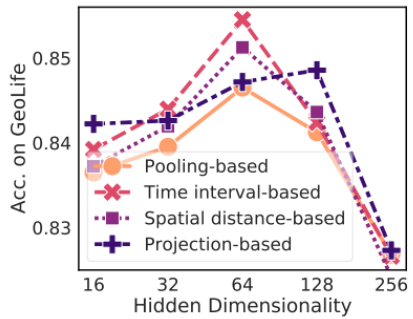
Variant	Param	GeoLife		Grab-Posisi	
		Acc↑	Δ	Acc↑	Δ
Base	59	83.42	0.00	78.59	0.00
Base+APE	65	82.38	-1.04	77.02	-1.57
Base+CAPE	59	83.51	+0.09	78.71	+0.12
Base+RPE	61	83.72	+0.30	78.65	+0.06
Base+Conv	61	83.05	-0.37	79.20	+0.61
TrajFormer	76	**85.45**	**+2.03**	**79.76**	**+1.17**

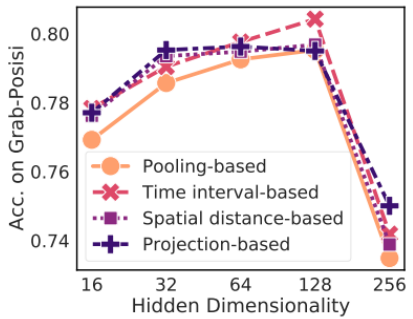
Effects of Squeezed Transformer Block

- Hyperparameter study
  - Number of transformer blocks
  - Number of hidden units
  - Different squeeze functions









Effects of Squeeze Rate

- Varying the squeeze rate r
  - Using time interval-based squeeze
  - A trade-off between throughput and accuracy

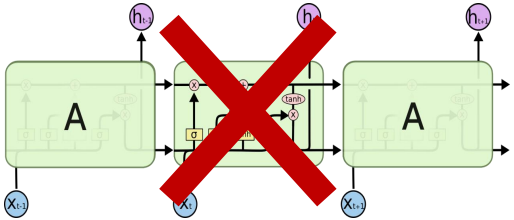
r	FLOPs	GeoLife		Grab-Posisi	
		TPS↑	Acc↑	TPS ↑	Acc↑
1	2.39	267	**85.59**	214	**80.02**
2	2.18	314	85.45	280	79.76
4	2.08	340	85.20	296	79.35
8	2.02	**356**	84.97	**304**	79.07

Conclusions

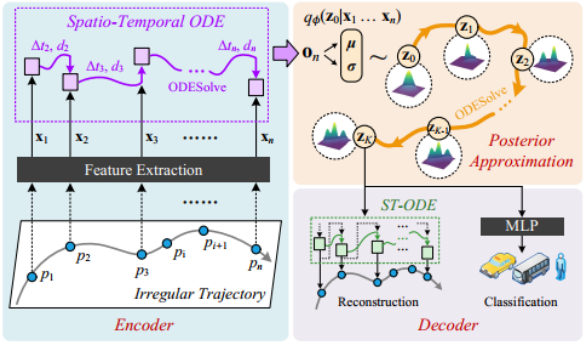
- CPE has the potential to be a basic building block for modeling trajectories
  - Promising performance
  - Can be plugged in any trajectory models
- Compared to SoTA RNN models
  - TrajFormer achieves comparable performance
  - Running at least 26× faster than TrajODE
- **Future work**
  - How to couple the transformer model with latent variable to enhance robustness?
  - Extend our model to broader applications, such as trajectory prediction

Conclusions on ST Trajectories

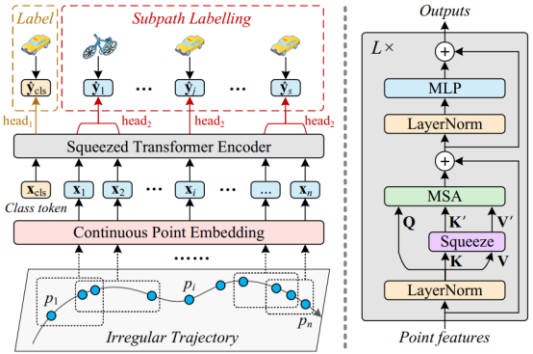
- Existing AI approaches for CV/NLP are NOT always good choices for modeling trajectories
- Capturing the irregularity of trajectories is of great importance to trajectory modeling
- We demonstrate how to encode the domain knowledge (i.e., irregularity) into existing AI methods, including RNNs and Transformers



Classic RNNs



Continuous trajectory modeling



Efficient trajectory modeling

Presented Papers

1	TrajFormer: Efficient Trajectory Classification with Transformers	Link	CIKM	2022	NUS	Jingtao HE
2	Stochastic Trajectory Prediction via Motion Indeterminacy Diffusion	Link	CVPR	2022	UCLA	Ruiguo ZHONG
3	Modeling Trajectories with Recurrent Neural Networks	Link	IJCAI	2017	Singapore Management University	Zhixiong Wang
1	Urban regional function guided traffic flow prediction	Link	nation Sci	2023	Sun Yat-sen University	Jiahui LIANG
2	UrbanFM: Inferring Fine-Grained Urban Flows		KDD	2019	NUS	Gangyong Zhu

Q&A

Thanks!

CityMind Lab

amazon Microsoft Tencent Alibaba.com 京东 JD.COM CAINIAO 菜鸟