3T-Net: Transformer Encoders for Destination Prediction

Jing Zhang
(Joint work with Daniel Nikovski and Takuro Kojima)

The 42nd Chinese Control Conference

Tianjin, China
July 25, 2023

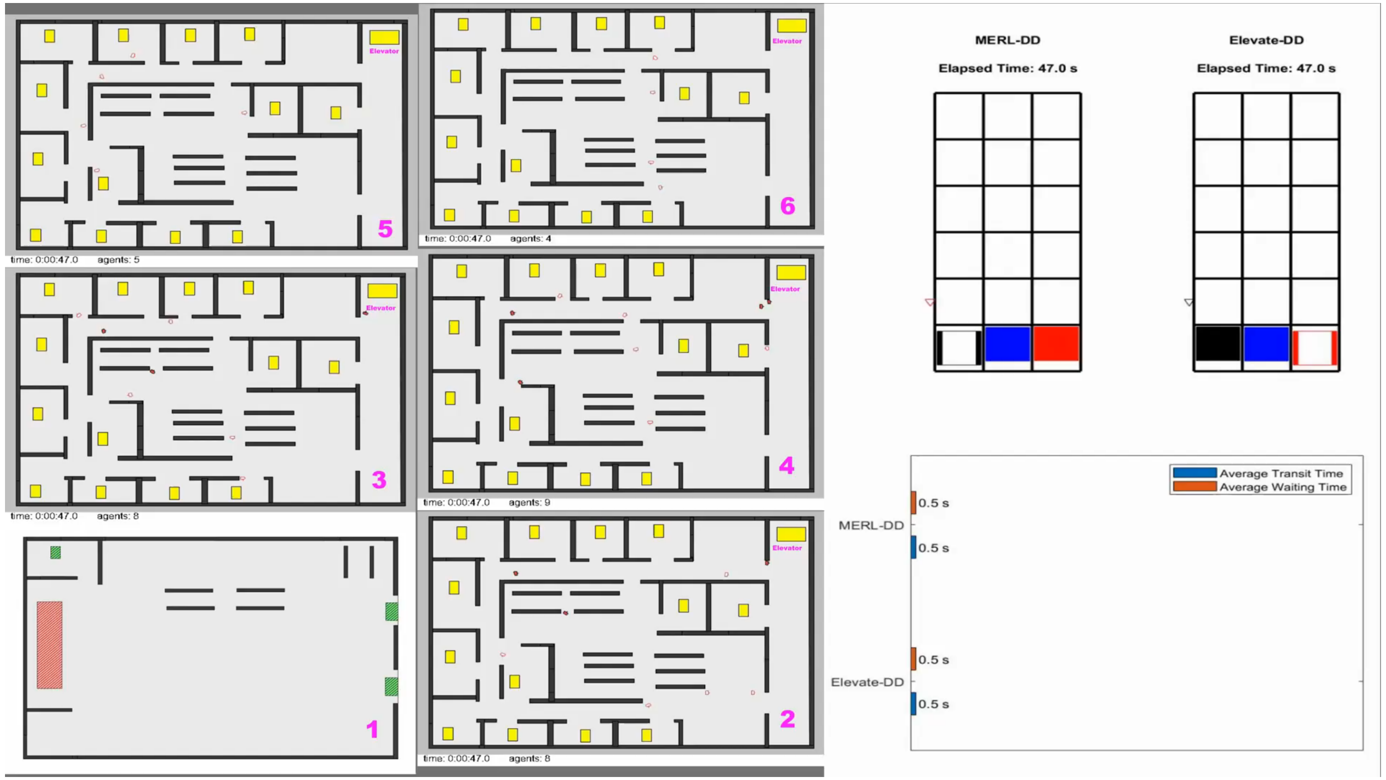
MITSUBISHI ELECTRIC RESEARCH LABORATORIES (MERL)
Cambridge, Massachusetts, USA
http://www.merl.com

Outline

- Motivation
- Problem Formulation
- The Proposed Model
- Numerical Experiments
  - Data Generation
  - Data Preprocessing
- Experimental Results
  - Confusion Matrix
  - Accuracy

Motivation

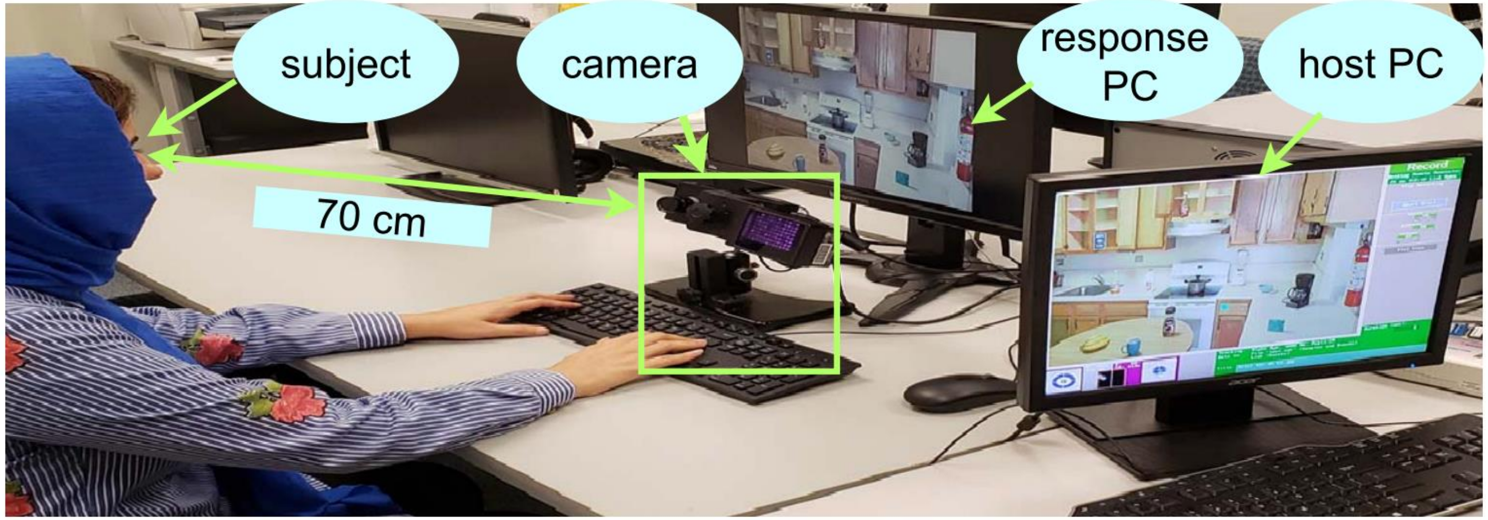
- Application in Elevator Scheduling [ECC’22]
  - Simulated a down-peak scenario for a 6-floor building
  - MERL-DD - a Transformer based scheduler
  - Elevate [ELEVATE] -DD – Elevate’s built-in myopic scheduler



[ECC’22] Jing Zhang, Athanasios Tsiligkaridis, Hiroshi Taguchi, Arvind Raghunathan, and Daniel Nikovski. "Transformer Networks for Predictive Group Elevator Control." In 2022 European Control Conference (ECC), pp. 1429-1435. IEEE, 2022.

[ELEVATE] https://peters-research.com/index.php/elevate/.

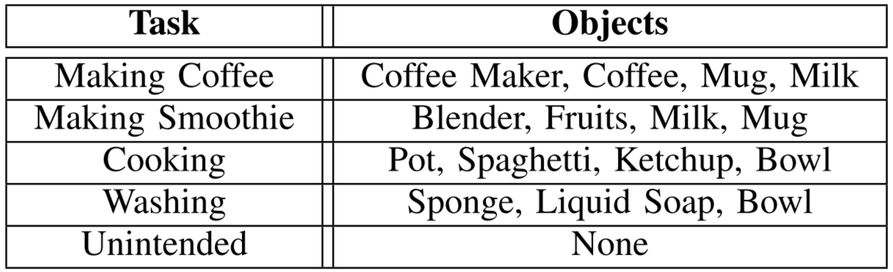
Motivation (Cont'd)



- Let's revisit the destination prediction problem: If we had collected some eye movement data (refer to [TNSRE]), isn't the intention prediction problem a variation of the destination prediction problem?

- Note that, essentially, the eye movement data are 3D trajectories, and the labels (intended tasks) are simply the "destinations".
- "Trajectories" could be very general, depending on what data we are interested in and what sensors we use to collect them.

LIST OF TASKS ALONG WITH THE OBJECTS INVOLVED IN EACH TASK

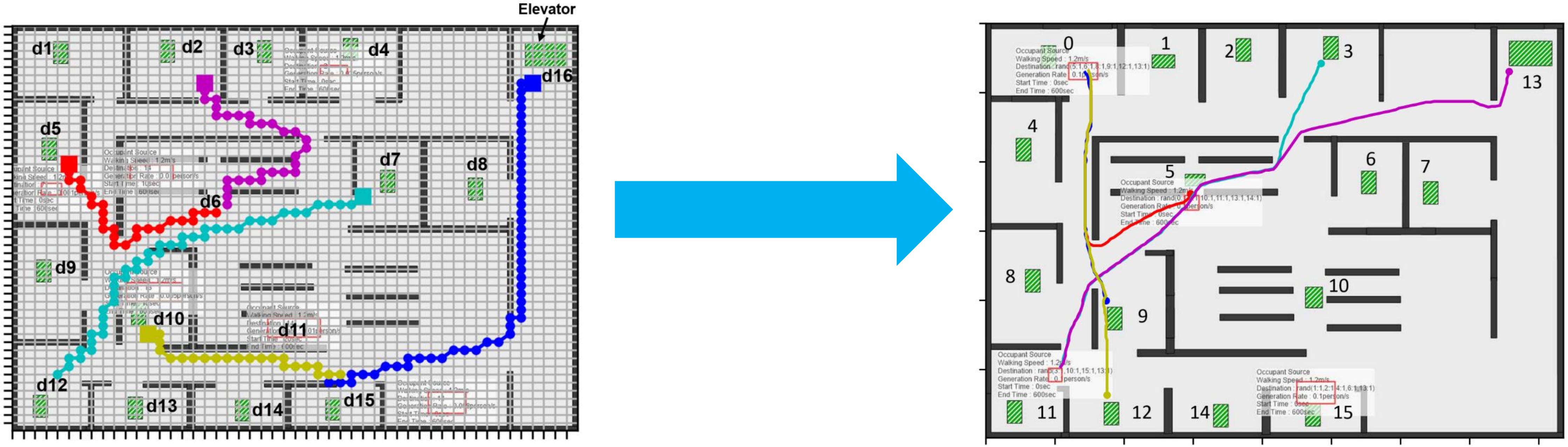


Task	Objects
Making Coffee	Coffee Maker, Coffee, Mug, Milk
Making Smoothie	Blender, Fruits, Milk, Mug
Cooking	Pot, Spaghetti, Ketchup, Bowl
Washing	Sponge, Liquid Soap, Bowl
Unintended	None

[TNSRE] F. Koochaki and L. Najafizadeh, A data-driven framework for intention prediction via eye movement with applications to assistive systems, in IEEE Transactions on Neural Systems and Rehabilitation Engineering, 29, 2021: 974–984.

Motivation (Cont’d)

- Our previous work [ECC’22] can address destination prediction problem, but we introduced a discretization procedure on the spatial data and considered lots of **extra candidate destination labels**, which would cause VRAM issues when #of cells gets large.
- Can we directly deal with continuous trajectories data?



Problem Formulation

- Formulate destination prediction as a multivariate time series classification (MTSC) problem.
- Consider a multivariate time series (MTS) $X = [\mathrm{x}_1, \dots, \mathrm{x}_t] \in \mathbb{R}^{m \times t}$, where $\mathrm{x}_j = [x_{1,j}, \dots, x_{m,j}] \in \mathbb{R}^m, j = 1, \dots, t$.
  - Here, m represents the number of variables and t is the length of the time series.
  - Each MTS is assigned a class label y from the label set Δ.
  - A collection of n such MTS is represented as $\mathcal{X} = [\mathrm{X}_1, \dots, \mathrm{X}_n] \in \mathbb{R}^{n \times m \times t}$ and their corresponding labels are $y = [y_1, \dots, y_n] \in \Delta^n$.
  - The task of MTSC is to train a classifier $f: X \mapsto y$ that predicts the class label for a given, previously unseen MTS.

MITSUBISHI ELECTRIC
Changes for the Better

The Proposed Model

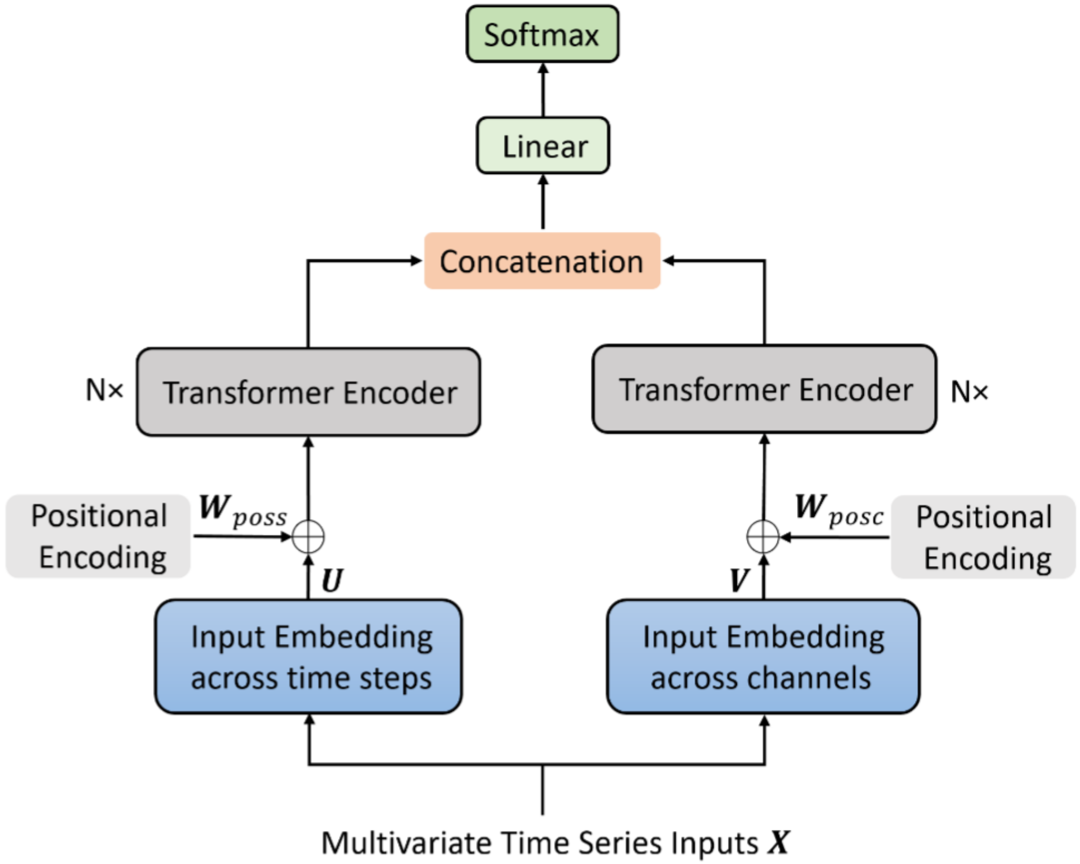


Fig. 1: The architecture of our Two-Tower Transformer Network (3T-Net) for multivariate time series classification.

- To capture **step-wise dependencies** across time stamps of the MTS, the feature vectors x_j are linearly projected onto a d-dimensional vector space, where d is the model dimension: $\mathrm{u}_j = \mathrm{W}_s \mathrm{x}_j + \mathrm{b}_s$, where $\mathrm{W}_s \in \mathbb{R}^{d\times m}$, $\mathrm{b}_s \in \mathbb{R}^d$ are learnable parameters.

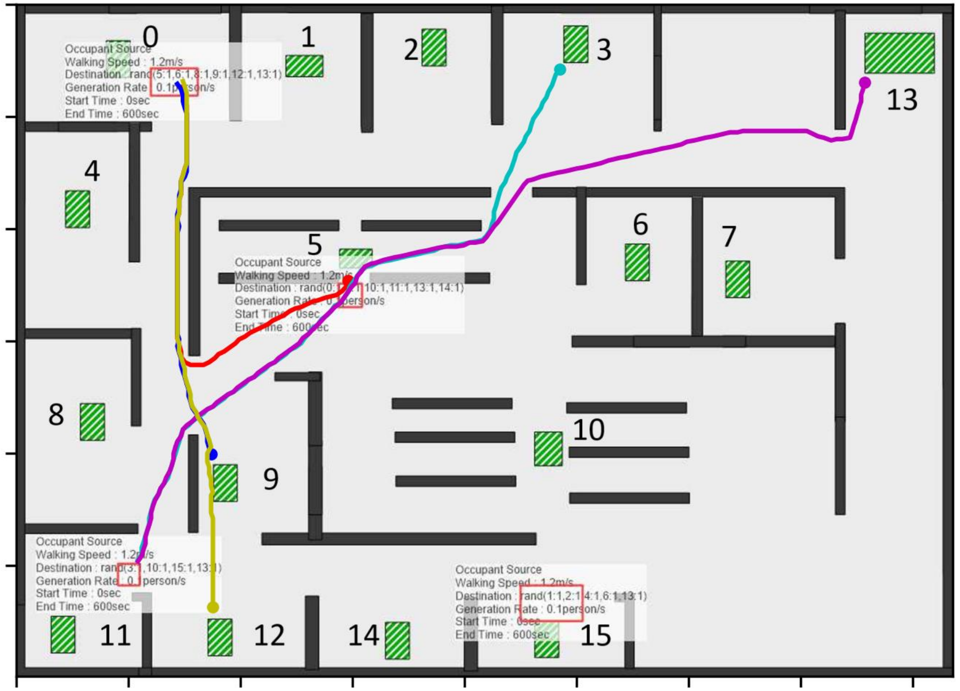
- To capture **channel-wise dependencies** across variables (dimensions) of the MTS, we transpose X and write it as $\mathrm{X}^\tau = [\tilde{\mathrm{x}}_1, \dots, \tilde{\mathrm{x}}_m] \in \mathbb{R}^{t\times m}$, where $\tilde{\mathrm{x}}_i = [x_{i,1}, \dots, x_{i,t}] \in \mathbb{R}^t$, $i = 1, \dots, m$. Similarly, the univariate time series $\tilde{\mathrm{x}}_i$ are linearly projected onto the same d-dimensional vector space: $\mathrm{v}_i = \mathrm{W}_c \tilde{\mathrm{x}}_i + \mathrm{b}_c$, where $\mathrm{W}_c \in \mathbb{R}^{d\times t}$, $\mathrm{b}_c \in \mathbb{R}^d$ are learnable parameters.

- Next, since the transformer is a feed-forward architecture that is insensitive to the ordering of input, in order to make it aware of the **sequential nature** of the time series and also the variables, we add positional encodings $\mathrm{W}_{poss} \in \mathbb{R}^{d\times t}$ and $\mathrm{W}_{posc} \in \mathbb{R}^{d\times m}$ to the input vectors $U = [\mathrm{u}_1, \dots, \mathrm{u}_t] \in \mathbb{R}^{d\times t}$ and $V = [\mathrm{v}_1, \dots, \mathrm{v}_m] \in \mathbb{R}^{d\times m}$, respectively: $U' = U + \mathrm{W}_{poss}, V' = V + \mathrm{W}_{posc}$, which become the final input vectors to the two-tower Transformer encoders, respectively.

Numerical Experiments – Data Generation

Table 1: Prior probabilities for destinations.

Destination	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Probability	0.05	0.03	0.05	0.06	0.05	0.04	0.11	0.04	0.05	0.04	0.1	0.03	0.03	0.2	0.04	0.08



- The layout of a floor in a simulated building [ECC'22].
- The bold black bars depict the walls or office desks, the green rectangles with slashes depict origins/destinations, and the light gray space represents corridors and other walkable areas.
- 5, 10: open offices
- 6: a lab
- 7: a restroom
- 9: a kitchen
- 13: an elevator
- others: offices with a door

Numerical Experiments – Data Generation (Cont'd)

- To generate raw trajectories data sets for training/testing, we use the prior probabilities specified in Table 1, and end up with 236 complete trajectories for each data set.

- The number of variables in each trajectory (represented as a multivariate time series) is 2, corresponding to horizontal and vertical coordinates, respectively.

- The maximum length of the trajectories in the raw training (resp., testing) data set is 83 (resp., 82), and the minimum length of the trajectories in the raw training (resp., testing) data set is 20 (resp., 18).

Numerical Experiments – Data Preprocessing

- For MTSC experiments, we extract partial trajectories from the raw data sets.
- Note that our actual goal is predicting a destination for a given partial trajectory. For each and every partial trajectory, its ground truth label (0 through 15) is the converted final point of the corresponding entire trajectory.
- We also pad zeros to the end of each partial trajectory to force its length be 100.

Numerical Experiments – Data Preprocessing (Cont’d)

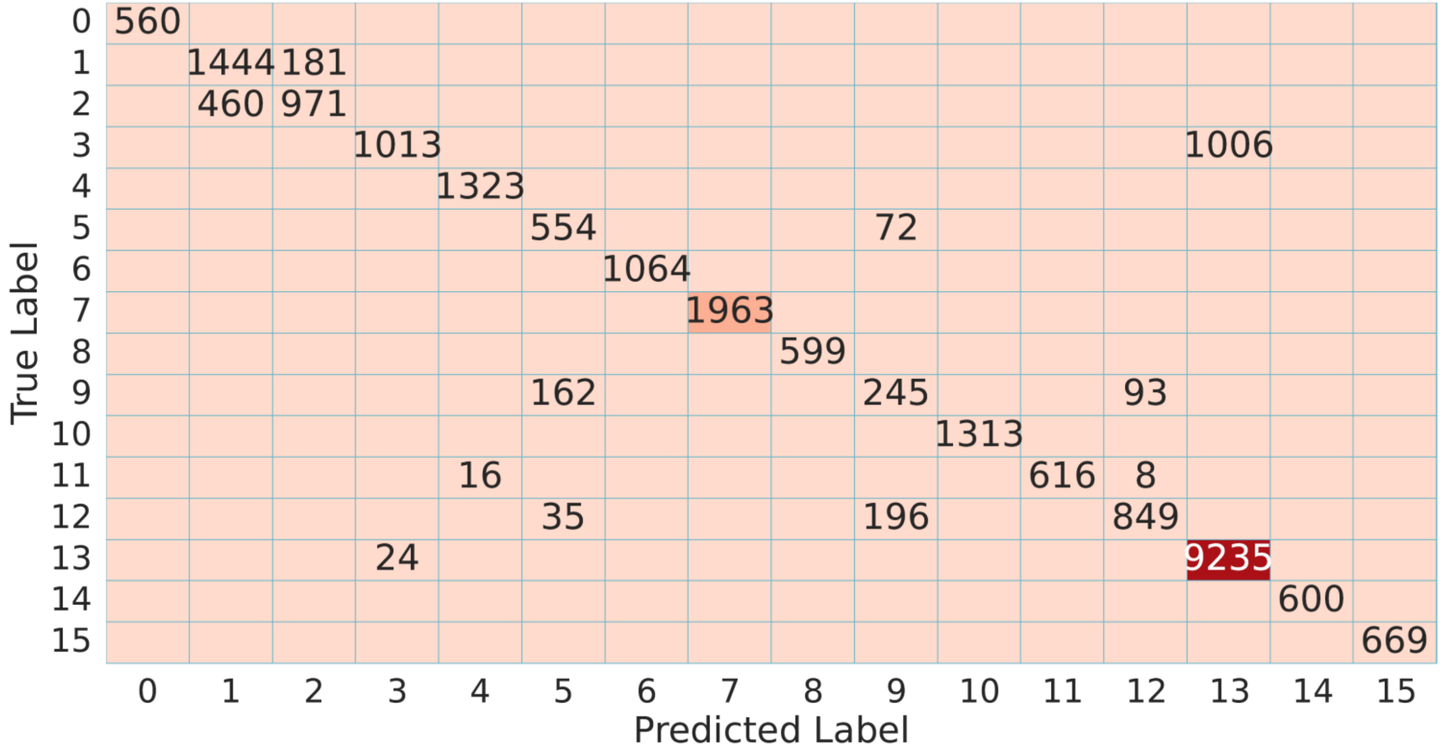
- Partial trajectories start with their actual origin or any intermediate point along the route to their respective destination.

- For the training data set, partial trajectories with all reasonably possible lengths (e.g., ≥ 5) are included.

- The testing data contains only a selected proportion of the partial trajectories whose length is reasonable (e.g., ≥ 5) and equals the length of their respective entire trajectory multiplied by a factor θ on some interval (e.g., $\theta \in [0.5, 0.6)$).

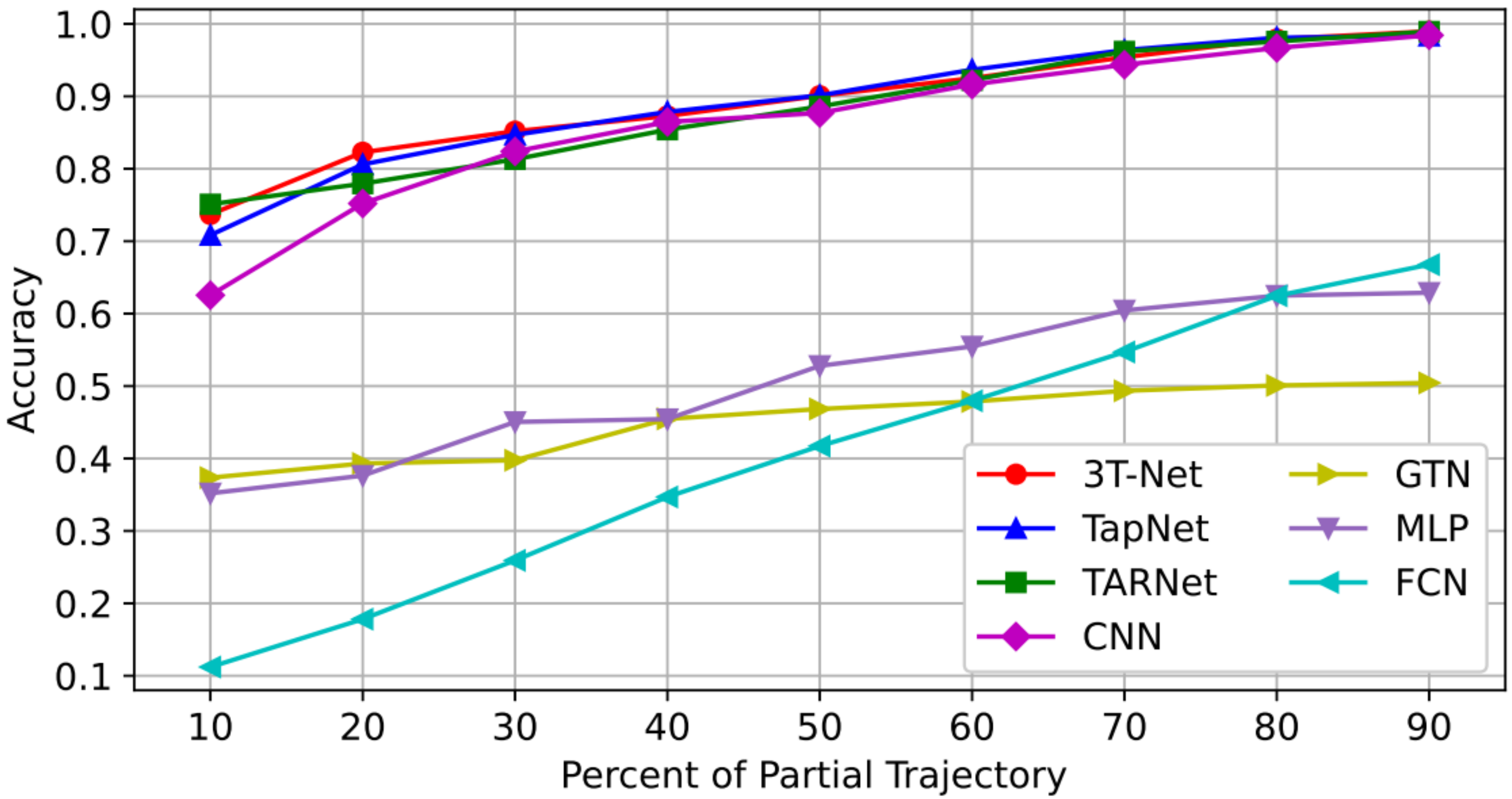
Experimental Results – Confusion Matrix

- The confusion matrix obtained from 3T-Net with $\theta \in [0.5, 0.6)$.
- Overall accuracy: 0.91; capable of dealing with unbalanced class labels
- Omitted the zero (0) values which indicate a true label would never be predicted as a certain label; e.g., the true label 15 would never be predicted as 0.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	560															
1		1444	181													
2		460	971													
3				1013										1006		
4					1323											
5						554				72						
6							1064									
7								1963								
8									599							
9						162				245			93			
10											1313					
11					16							616	8			
12						35				196			849			
13				24										9235		
14															600	
15																669

Experimental Results – Accuracy


Accuracy
Percent of Partial Trajectory
3T-Net
TapNet
TARNet
CNN
GTN
MLP
FCN