MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

3T-Net: Transformer Encoders for Destination Prediction

Zhang, Jing; Nikovski, Daniel; Kojima, Takuro

TR2023-094 July 25, 2023

Abstract

The need for accurate and timely destination prediction arises in many transportation applications. We formulate destination prediction as a multivariate time series classification problem, and leverage part of the core components of the Transformer network to build a new deep neural network model exclusively for this task. The key building block of our model consists of Two Towers of Transformer encoders, and we call it "3T-Net." Through extensive comparison experiments on a simulated indoor trajectories data set, we show that 3T-Net performs better or close to other investigated state-of-the-art deep learning based models. Our model can also be used for outdoor destination prediction scenarios and more general multivariate time series classification problems.

The Chinese Control Conference 2023

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

3T-Net: Transformer Encoders for Destination Prediction

Jing Zhang[1], Daniel Nikovski[1], Takuro Kojima[2]
1. Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA
E-mail: {jingzhang,nikovsiki}@merl.com
2. Advanced Technology R&D Center, Mitsubishi Electric Corporation, Tokyo 100-8310, Japan
E-mail: kojima.takuro@ah.mitsubishielectric.co.jp

Abstract: The need for accurate and timely destination prediction arises in many transportation applications. We formulate destination prediction as a multivariate time series classification problem, and leverage part of the core components of the Transformer network to build a new deep neural network model exclusively for this task. The key building block of our model consists of Two Towers of Transformer encoders, and we call it "3T-Net." Through extensive comparison experiments on a simulated indoor trajectories data set, we show that 3T-Net performs better or close to other investigated state-of-the-art deep learning based models. Our model can also be used for outdoor destination prediction scenarios and more general multivariate time series classification problems.

Key Words: Destination Prediction, Multivariate Time Series Classification, Transformer, Deep Learning

1 Introduction

The need for accurate and timely destination prediction arises in many transportation applications involving both vehicular and pedestrian traffic. One ubiquitous such application is car navigation, where the successful prediction of the final destination of a car from its initial path could eliminate the need to set the destination manually while driving, thus avoiding the risk of collision and increasing convenience. Similarly, prediction of pedestrians' destinations indoors could open the possibility for innovative services, such as predictive group elevator scheduling [1], where elevator cars are dispatched ahead of time to the correct floor to shorten or eliminate waiting times. At a larger scale, pedestrian destination prediction in shopping centers, airports, and train stations can optimize the control of pedestrian traffic and even the provision of personalized guidance information.

The development of destination prediction technology is aided by advances in technologies for position estimation, both outdoors (for example based on GPS signals), as well as indoors (for example, based on motion detectors, cameras, Wi-Fi signal reflections, and even smart carpets [2–4]). Regardless of the sensing mechanism, the collected data is in the form of trajectories of positions. The objective of destination prediction is to infer the last position in the sequence from an initial recorded sub-sequence, under the usually reasonable assumption that movement patterns are repeatable, and paths that started similarly will also end in the same final position. This is largely true, with the important caveat that paths to different destinations from the same original position might look similar until a point where they diverge. This means that destination prediction is by necessity a probabilistic problem, where a set of likely destinations can be determined only in a probabilistic sense, for example by means of a multinomial probability distribution over them.

It is also clear that the needs for both accurate and timely destination prediction are mutually exclusive. As time progresses and more and more of the traversed path becomes known, prediction can be very accurate, converging to full certainty just before the final destination is reached. However, such very accurate prediction is not timely enough to solve decision and control problems of practical significance, such as rerouting a vehicle around a traffic jam or informing an airline passenger about a gate change. For this reason, predictive methods are needed that can achieve reasonable accuracy based on a relatively small fraction of the path to the destination observed.

The nature of traffic and the information contained in partial paths also presents some challenges as regards the use of machine learning technologies for destination prediction. Basing the prediction only on the latest position in the recorded partial path leading up to the time of prediction is clearly sub-optimal, because it ignores the direction of motion that should be highly indicative of the general area where the final destination is. In other words, the process generating the locations along the path is not first-order Markovian, and more observations should be used for better accuracy. However, if more than one observation should be used, the question arises how many. Ideally, all of the observed path so far should be used for maximum accuracy, but this precludes the use of many machine learning algorithms that use an input vector of a fixed size. That is, what is necessary is a predictive method that can classify entire sequences of variable length, in a probabilistic setting.

Transformer networks [5], as widely acknowledged, are a type of deep neural network that are primarily used for natural language processing tasks such as machine translation, text classification, and question answering. They offer several advantages over traditional recurrent neural networks (RNNs), including faster training and inference due to parallel computation and the ability to process sequences of variable lengths. Transformer networks have also found applications in time series classification, where they can be used both as feature extractors and end-to-end classifiers. When used as feature extractors, the Transformer can extract important representations from raw time series data, which can then be fed into other machine learning models for classification. As end-to-end classifiers, the Transformer can directly predict class labels from raw time series data, without the need for manual feature engineering. The attention mechanism in the Transformer can also help identify key regions in the time series, thereby improving the performance of the

classification model.

In this work, we formulate destination prediction as a multivariate time series classification (MTSC) problem, and leverage part of the core components of the Transformer network to build a new model exclusively for this task. The key building block of our neural network model consists of Two Towers of Transformer encoders, and we call it "3T-Net," for conciseness. Through extensive comparison experiments on a simulated indoor trajectories data set, we show that 3T-Net performs better or close to other state-of-the-art deep learning based models.

The remainder of the paper is organized as follows. We review related work in Section 2, formally formulate the MTSC problem in Section 3, and propose the 3T-Net model in Section 4. In Section 5, we first describe the data sets we use, and then present the experimental results comparing our model with six other deep learning based models. We conclude and propose future work in Section 6. Lists of key hyperparameters of selected state-of-the-art models that we use during training are provided in the Appendix.

2 Related Work

2.1 Multivariate Time Series Classification

Wang et al. [6] provided a baseline to exploit deep neural networks for multivariate time series (MTS) classification without any dedicated feature engineering and data preprocessing. The deep Multi-Layer Perceptrons (MLP) and Fully Convolutional Networks (FCN) are included. Zhao et al. [7] proposed a Convolutional Neural Network (CNN) framework for MTS classification. Zhang et al. [8] proposed a model named time series attentional prototype network (TapNet) for MTS classification, which is capable of extracting low-dimensional features from MTS with little domain knowledge and handling the shortage of labeled data.

Wen et al. [9] reviewed about 60 papers on transformer schemes for time series modeling by highlighting their strengths as well as limitations. The authors summarized the adaptations and modifications made to transformers for time series analysis, and categorized time series transformers according to common tasks including forecasting, anomaly detection, and classification. For MTS representation learning, Zerveas et al. [10] presented a transformer encoder-based framework, which includes an unsupervised pre-training scheme by reusing existing data samples. This framework can deal with both regression and classification tasks. Liu et al. [11] proposed a Gated Transformer Network (GTN) for MTS classification. With a gate that merges two towers of Transformer encoders, the model captures both channel-wise and step-wise correlations. Chowdhury et al. [12] proposed the Task-Aware Reconstruction Network (TARNet), a model using Transformers to learn task-aware data reconstruction that boosts end-task performance; it can also tackle both classification and regression tasks for MTS.

2.2 Destination Prediction

Hidden Markov Models (HMM) have been extensively researched for predicting routes and destinations [13–15]. However, these model-based approaches struggle with long input sequences. Deep learning methods, such as Multi-Layer Perceptrons (MLP) and Recurrent Neural Networks (RNN), can overcome this limitation [16]. RNNs have been commonly used for destination prediction [17–19]. In [20], a hierarchical model combining attention mechanism and Long-Short Term Memory (LSTM) was used for destination prediction. In [21], a Transformer network was used for destination prediction using only positioning information. In [22], another approach was proposed using a spatiotemporal Transformer with geographical information to predict taxi drivers' next destinations. Building on [21], [23] proposed a Transformer Encoder Stack (TES) as a destination prediction model that uses both position and context data to generate a probability distribution for potential destinations in a given area. It is worth pointing out that, as a preprocessing procedure, both [21] and [23] quantized the original continuous spatial data to discrete symbols, which introduced redundant candidate destination (classification) labels that could cause VRAM issues for GPUs. This work, on the other hand, will directly deal with continuous spatial data from the collected trajectories.

3 Problem Formulation

As a convention, throughout the paper, all vectors are column vectors. Since destination prediction is essentially a multivariate time series classification problem, we formulate the problem in a general form. Consider a multivariate time series (MTS) $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_t] \in \mathbb{R}^{m\times t}$, where $\mathbf{x}_j = [x_{1,j}, \dots, x_{m,j}] \in \mathbb{R}^m$, $j = 1, \dots, t$. 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 $\mathcal{X} = [\mathbf{X}_1, \dots, \mathbf{X}_n] \in \mathbb{R}^{n\times m\times t}$ and their corresponding labels are $\mathbf{y} = [y_1, \dots, y_n] \in \Delta^n$. The task of MTSC is to train a classifier $f : \mathbf{X} \mapsto y$ that predicts the class label for a given, previously unseen MTS.

4 The Proposed Model

In this section, we describe the architecture of our Two-Tower Transformer Network (3T-Net). For economy of space, we only briefly present the differences between our model and the existing ones. The Transformer encoders we use are transplanted from [5], and we omit the details of the encoders themselves. It is worth pointing out that, different from GTN [11], we do not use any masking mechanism for multi-head attention computation, because we believe the masking trick is designed exclusively for sequence generation purposes, which would be unnecessary for classification tasks. We will demonstrate this in the next section through extensive numerical experiments.

To capture step-wise dependencies across time stamps of the MTS, the feature vectors $\mathbf{x}_j$ are linearly projected onto a d-dimensional vector space, where d is the *model dimension*:

$$\mathbf{u}_j = \mathbf{W}_s\mathbf{x}_j + \mathbf{b}_s,$$

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

$$\mathbf{v}_i = \mathbf{W}_c\tilde{\mathbf{x}}_i + \mathbf{b}_c,$$

where $\mathbf{W}_c \in \mathbb{R}^{d\times t}$, $\mathbf{b}_c \in \mathbb{R}^d$ 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 (especially in our destination prediction application where MTS data contain ordered positional information; latitudes and longitudes are not interchangeable), we add positional encodings $\mathbf{W}_{poss} \in \mathbb{R}^{d\times t}$ and $\mathbf{W}_{posc} \in \mathbb{R}^{d\times m}$ to the input vectors $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_t] \in \mathbb{R}^{d\times t}$ and $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_m] \in \mathbb{R}^{d\times m}$, respectively:

$$\mathbf{U}' = \mathbf{U} + \mathbf{W}_{poss},$$
$$\mathbf{V}' = \mathbf{V} + \mathbf{W}_{posc},$$

which become the final input vectors to the two-tower Transformer encoders, respectively; refer to Fig. 1. We note that adding an additional positional encoding $\mathbf{W}_{posc}$ to the input vectors $\mathbf{V}$ is another difference between our 3T-Network and the existing GTN. All the positional encodings are computed following the same choice as in [5].

Finally, instead of using a Gate [11] to learn the weights of the outputted vectors from the two towers of Transformer encoders, we simply concatenate them so as to simplify the training process. We believe that the following linear layer would be able to capture the variances of these weights. The last layer of the model is a Softmax function that outputs probabilities of candidate class labels from which we can extract the maximum value and its index to locate a classification label from Δ.

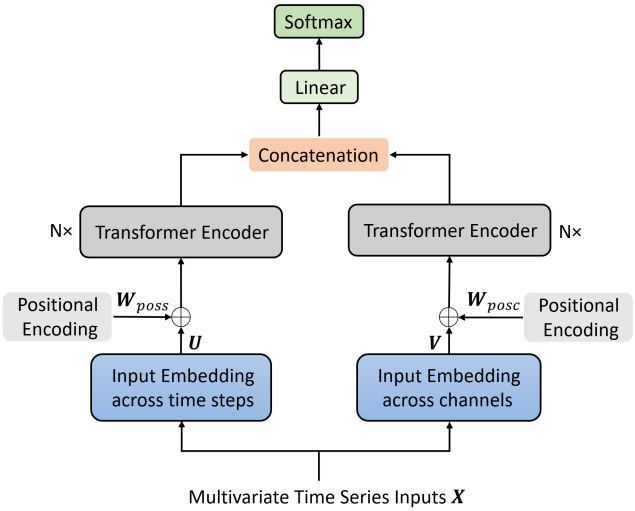


Fig. 1: The architecture of our Two-Tower Transformer Network (3T-Net) for multivariate time series classification.

5 Experimental Results

5.1 Data Sets

We generate people's movement data on a floor in a building. In particular, we use the SimTread software package for this purpose. The layout of a floor in a building that we use in the simulation is shown in Fig. 2, where we have 16 destinations denoted 0 through 15. Also shown in Fig. 2 are five typical trajectories; for example, the yellow line starting from office 0 and ending at office 12 represents a complete trajectory with office 12 as its destination. Note that in our simulations, we assume all destinations 0 through 15 could also be an origin. 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).

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. To that end, 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 enforce a length of 100. Through careful data preprocessing, we examine four combinations of training/testing settings:

- Case 1-1: Partial trajectories always start with their actual origin. For the training data set, partial trajectories with all reasonably possible lengths (e.g., ≥ 5; we note that a too short partial trajectory would hardly provide any meaningful trend information to a predictive model and we have tested lengths less than 5 in our experiments and found that partial trajectories with such limited lengths would confuse classifiers more significantly than longer ones) are included. The testing data set 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., $\theta \in [0.5, 0.6)$).
- Case 1-2: Only entire trajectories excluding their respective final point (destination) are included in the training data. The testing data set is the same as in Case 1-1.
- **Case 2-1**: 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., $\theta \in [0.5, 0.6)$).
- Case 2-2: The training data set is the same as in Case 1-1. The testing data set is the same as in Case 2-1.

Intuitively, Case 2-1 is the most interesting setting, because the training data set contains the richest partial trajectories (MTS) that would help the model learn enough patterns for prediction. Considering the current paper focuses on deep learning (DL)-based models, which are are very hungry for data, we find that the way of extracting partial trajectories for training in Case 2-1 is most favourable, because it can drastically expand the collected raw trajectories. To see this more clearly, considering a complete trajectory with length $t > 5$, then the number of partial trajectories with reasonable lengths (≥ 5) that we can extract would be

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

$\sum_{l=5}^{t-1}(t-l) = \frac{(t-5)(t-4)}{2} = O(t^2)$ for $t \gg 5$. Of course, to tackle such large-sized training data, we need significant computational support (CPU/GPU/memory/VRAM). On the other extreme, Case 1-2 uses the least number of partial trajectories for training and, through extensive experiments, we find that even if we use certain masking tricks (like the one employed by GTN), the performance of all the investigated classifiers would be very poor (see Fig. 5b). The other two cases (1-1 and 2-2) use medium-sized data sets for training, which could be useful when only limited computational resources are available.

In Fig. 4b, we show the class-wise number of MTS (partial trajectories) in the processed training/testing data sets for Case 2-1 (with $\theta \in [0.4, 0.5)$). Clearly, the classes in both data sets are highly imbalanced. In the next section, we will present experimental results showing that the proposed 3T-Net, together with other state-of-the-art DL-based models, can effectively classify the testing MTS without overfitting the training MTS data.

5.2 Results

We compare the classification performance (in terms of accuracy on the testing data set) of our 3T-Net with 6 other DL-based models. Fig. 5 shows the prediction (classification) accuracy vs. the percent of partial trajectory seen so far. It is seen that for Cases 1-1, 2-1, and 2-2, the classification accuracy given by 3T-Net is either the highest or close to the highest. In particular, for the most interesting case (2-1), 3T-Net yields the highest accuracy (above 0.7) even when we only see a very small proportion ($\theta \in [0.2, 0.3)$ or even $\theta \in [0.1, 0.2)$) of the entire trajectories. On the other hand, for Case 1-2, all models give almost unusable prediction (classification) results — the accuracy is way too low. In general, the more partial trajectories we use for training, the higher accuracy the classifiers would produce for the testing partial trajectories.

To see the classification results more clearly, in Fig. 3 we show the confusion matrix obtained from 3T-Net for Case 2-1 with $\theta \in [0.5, 0.6)$. It is seen that, when only 50%-60% of an entire testing trajectory is available, if its true label is 9, the 3T-Net classifier would predict its destination as 9 or 5 or 12, with probabilities of $245/(162+245+93) = 0.49$, $162/(162+245+93) = 0.32$, and $93/(162+245+93) = 0.19$, respectively. This is not surprising, because we see from Fig. 2 that a trajectory with origin 0 and destination 9 partially overlaps a trajectory with origin 0 and destination 12, and also partially overlaps another trajectory with origin 0 and destination 5. Such overlapping would confuse the classifier when training; in this case, if only seeing a small proportion of an entire trajectory, it is hard to judge how the remaining trajectory would evolve and where the final destination would be. Similar observations could be made for other entries in the confusion matrix (e.g., the true label 3 is predicted to be 3 or 13, with probabilities around 0.50 and 0.50, respectively).

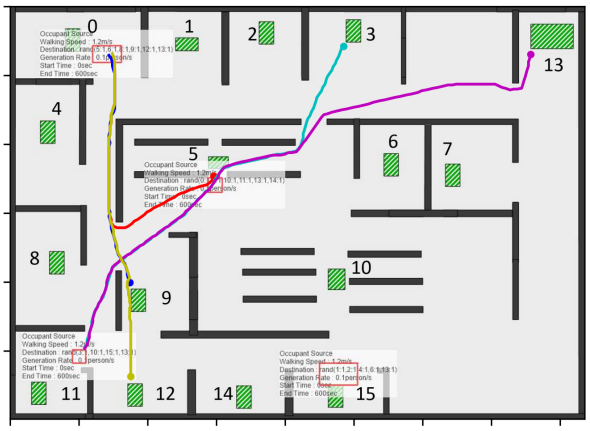


Fig. 2: The layout of a floor in a simulated building [1]. The bold black bars depict the walls or office desks, the green rectangles with slashes depict origins/destinations, and the light gray space represents corridors and other walkable areas. 5, 10: open offices; 6: a lab, 7: a restroom; 9: a kitchen; 13: an elevator; others: offices with a door.

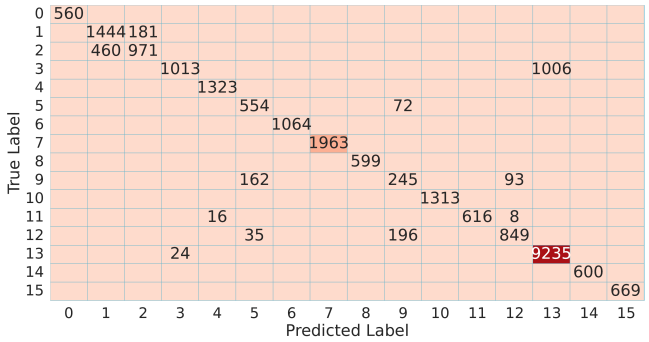


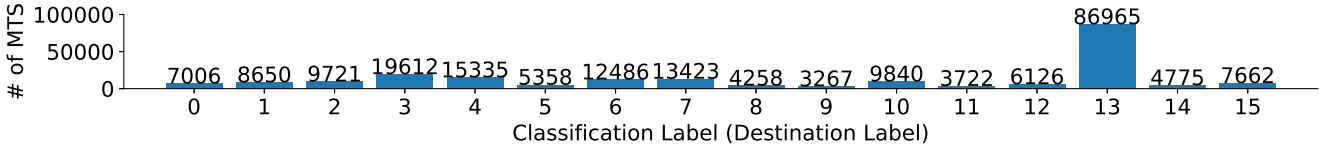
Fig. 3: The confusion matrix obtained from 3T-Net for Case 2-1 with $\theta \in [0.5, 0.6)$; the overall accuracy is 0.91. For economy of space, we omitted the zero (0) values which indicate a true label would never be predicted as a certain label; e.g., the true label 15 would never be predicted as 0.

6 Conclusion

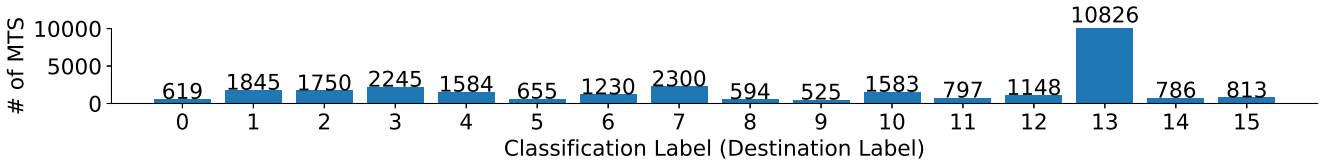
We have proposed "3T-Net," a deep neural network model that utilizes two towers of Transformer encoders for destination prediction tasks, whose effectiveness comparing to other state-of-the-art deep learning based models was shown through extensive experiments on a simulated indoor trajectories data set. It can also be used for outdoor destination prediction scenarios and more general multivariate time series classification problems. For future work, it is of interest to incorporate contextual information of pedestrians in a building or drivers in an outdoor road network.

References

[1] J. Zhang, A. Tsiligkaridis, H. Taguchi, A. Raghunathan, and D. Nikovski, Transformer networks for predictive group ele-

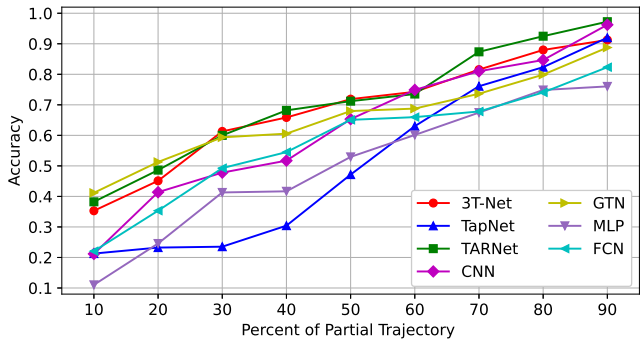


(a) For a full training data set; Case 2-1.

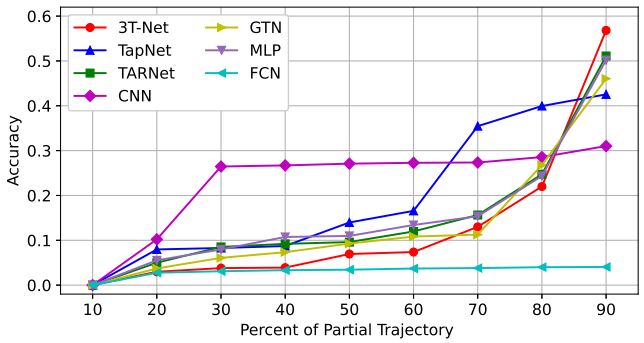


(b) For a selected partial testing data set; Case 2-1 with $\theta \in [0.4, 0.5)$.

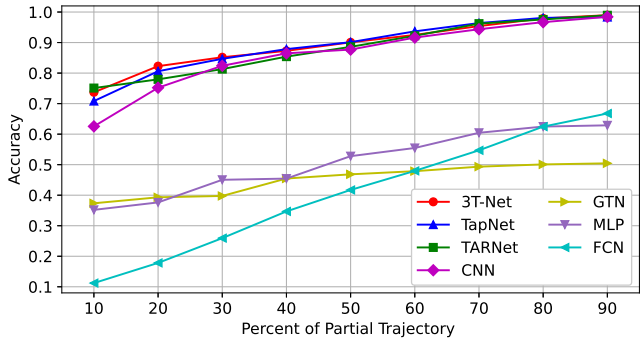
Fig. 4: Number of MTS (partial trajectories) vs. classification label (destinational label).



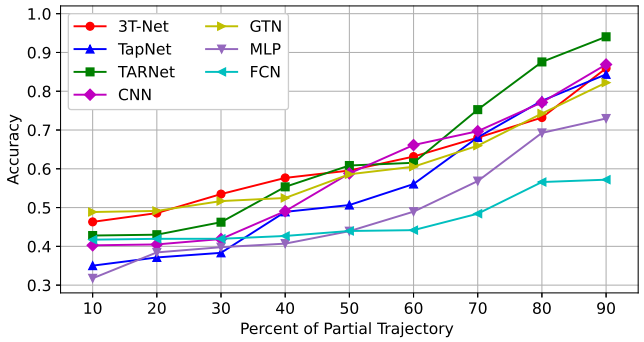
(a) For Case 1-1.



(b) For Case 1-2.



(c) For **Case 2-1**.



(d) For Case 2-2.

Fig. 5: The prediction accuracy vs. the percent of partial trajectory seen so far, with $\theta \in [0.0, 0.1), [0.1, 0.2), \ldots, [0.8, 0.9)$, respectively. The x-axis values represent the corresponding intervals; e.g., 60 percent represents interval $[0.5, 0.6)$, and the overall prediction accuracy given by 3T-Net for Case 2-1 with $\theta \in [0.5, 0.6)$ is 0.91.

vator control, in *2022 European Control Conference (ECC)*, 2022: 1429–1435.

[2] Y. Ivanov, A. Sorokin, C. Wren, and I. Kaur, Tracking people in mixed modality systems, in *Visual Communications and Image Processing 2007*, Vol. 6508, 2007: 209–219.

[3] F. Adib, Z. Kabelac, D. Katabi, and R. C. Miller, 3D tracking via body radio reflections, in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, 2014: 317–329.

[4] R. J. Orr and G. D. Abowd, The smart floor: A mechanism for natural user identification and tracking, in *Conference on Human Factors in Computing Systems - Proceedings*, 2000: 275–276.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, Attention is all you need, in *Advances in Neural Information Processing Systems*, 30, 2017.

[6] Z. Wang, W. Yan, and T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, in *2017 International joint conference on neural networks (IJCNN)*, 2017: 1578–1585.

[7] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, Convolutional neural networks for time series classification, in *Journal of Systems Engineering and Electronics*, 28(1), 2017: 162–169.

[8] X. Zhang, Y. Gao, J. Lin, and C.-T. Lu, Tapnet: Multivariate time series classification with attentional prototypical network, in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020: 6845–6852.

[9] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, Transformers in time series: A survey, in *arXiv preprint arXiv:2202.07125*, 2022.

[10] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, A transformer-based framework for multivariate time series representation learning, in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021: 2114–2124.

[11] M. Liu, S. Ren, S. Ma, J. Jiao, Y. Chen, Z. Wang, and W. Song, Gated transformer networks for multivariate time series classification, in *arXiv preprint arXiv:2103.14438*, 2021.

[12] R. R. Chowdhury, X. Zhang, J. Shang, R. K. Gupta, and D. Hong, Tarnet: Task-aware reconstruction for time-series transformer, in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022: 212–220.

[13] N. Ye, Z. Wang, R. Malekian, Q. Lin, and R. Wang, A method for driving route predictions based on hidden markov model, in *Mathematical Problems in Engineering*, 2015.

[14] Y. Lassoued, J. Monteil, Y. Gu, G. Russo, R. Shorten, and M. Mevissen, A hidden markov model for route and destination prediction, in *The 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017: 1–6.

[15] J. P. Epperlein, J. Monteil, M. Liu, Y. Gu, S. Zhuk, and R. Shorten, Bayesian classifier for route prediction with markov chains, in *The 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018: 677–682.

[16] J. Xu, J. Zhao, R. Zhou, C. Liu, P. Zhao, and L. Zhao, Predicting destinations by a deep learning based approach, in *IEEE Transactions on Knowledge and Data Engineering*, 33(2), 2019: 651–666.

[17] L. Zhang, G. Zhang, Z. Liang, and E. F. Ozioko, Multi-features taxi destination prediction with frequency domain processing, in *PloS one*, 13(3), 2018.

[18] L. Zhang, G. Zhang, Z. Liang, Q. Fan, and Y. Li, Predicting taxi destination by regularized RNN with SDZ, in *IEICE Transactions on Information and Systems*, 101(8), 2018: 2141–2144.

[19] Y. Endo, K. Nishida, H. Toda, and H. Sawada, Predicting destinations from partial trajectories using recurrent neural network, in *Advances in Knowledge Discovery and Data Mining: 21st Pacific-Asia Conference*, Springer, 2017: 160–172.

[20] J. Xu, J. Zhao, R. Zhou, C. Liu, P. Zhao, and L. Zhao, Predicting destinations by a deep learning based approach, in *IEEE Transactions on Knowledge and Data Engineering*, 33(2), 2019: 651–666.

[21] A. Tsiligkaridis, J. Zhang, H. Taguchi, and D. Nikovski, Personalized destination prediction using transformers in a contextless data setting, in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020: 1–7.

[22] Z. U. Abideen, H. Sun, Z. Yang, R. Z. Ahmad, A. Iftekhar, and A. Ali, Deep wide spatial-temporal based transformer networks modeling for the next destination according to the taxi driver behavior prediction, in *Applied Sciences*, 11(1), 2020.

[23] A. Tsiligkaridis, J. Zhang, I. C. Paschalidis, H. Taguchi, S. Sakajo, and D. Nikovski, Context-aware destination and time-to-destination prediction using machine learning, in *2022 IEEE International Smart Cities Conference (ISC2)*, 2022: 1–7.

[24] M. Liu, S. Ren, S. Ma, J. Jiao, Y. Chen, Z. Wang, and W. Song, Gated transformer networks for multivariate time series classification, in *https://github.com/ZZUFaceBookDL/GTN*, 2021.

[25] SKTIME, in *https://www.sktime.org/en/stable/index.html*, 2022.

7 Appendix

In this section, we list the key hyperparameters of selected state-of-the-art models that we use during training for the experiments.

- 3T-Net and GTN [24]:
  - $epoch = 100$.
  - $batch\_size = 512$.
  - $learning\_rate = 10^{-4}$.
  - $d = 512$. Since the model processes time series instead of natural language, the encoding of words in NLP is omitted and only a linear layer is used to map to a dense vector of dimension d. Additionally, d ensures that the dimensions are the same at each connecting point of the modules.
  - The dimension of the hidden layer in Position-wise Feed Forward: $d\_hidden = 1024$.
  - The linear mapping dimensions in Multi-Head Attention: $q = 8$, $v = 8$.
  - The number of heads in Multi-Head Attention: $h = 8$.
  - The number of encoders in each tower: $N = 8$.
  - $dropout = 0.2$.
- TapNet [25]:
  - $epoch = 100$.
  - $batch\_size = 512$.
  - $dropout = 0.25$.
  - $filter\_sizes = (256, 256, 128)$. This sets the kernel size argument for each convolutional block and controls number of convolutional filters and number of neurons in attention dense layers.
  - $kernel\_sizes = (8, 5, 3)$.
  - $layers = (500, 300)$. This is the size of dense layers.
  - $reduction = 16$. This is for dividing the number of dense neurons in the first layer of the attention block.
- TARNet[1]:
  - $epoch = 100$.
  - $batch\_size = 128$.
  - $dropout = 0.1$.
  - $learning\_rate = 10^{-3}$.
  - $n\_layers = 4$.
  - $n\_head = 8$.
- CNN [25]:
  - $epoch = 100$.
  - $batch\_size = 512$.
  - $kernel\_size = 7$.
  - $avg\_pool\_size = 3$. This is the size of the average pooling windows.
  - $n\_conv\_layers = 2$. This is the number of convolutional plus average pooling layers.

Acknowledgement

The first author would like to thank Prof. Ioannis (Yannis) Paschalidis and Dr. Athanasios Tsiligkaridis for helpful discussions on part of the motivations behind this work. The authors would also like to thank anonymous reviewers for their comments on our earlier papers related to this work.

[1] https://github.com/ranakroychowdhury/TARNet