Data Mining for Network Management in Mobile-aware Large Scale Software-Defined Networking

Kohei Shiomoto

Tokyo City University

November 2019

Outline I

Reducing Inconsistency between Software-Defined Networking Controllers [12](Outline) I

Reducing Inconsistency between Software-Defined Networking Controllers [12]

Logically-centralized controller in SDN networks I

- A centralized controller cannot handle all the incoming request from switches as the network grows in SDN networks.
- One way to alleviate this concern is to distribute the state and/or control computation over multiple controllers while keeping a logically-centralized architecture [11, 7].

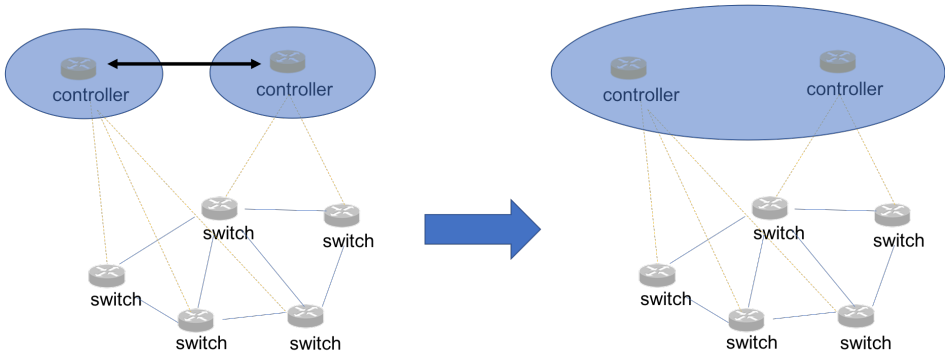


Figure: Logically centralized controller

Logically-centralized controller in SDN networks I

- The physical network is divided into multiple control domains.
- Each controller manages network states (e.g. network topology, switch, port, link, and host information) inside the domain and shares them between controllers as a "global network view".
- All SDN controllers need to maintain consistent view of global network in order to make a correct routing decision.

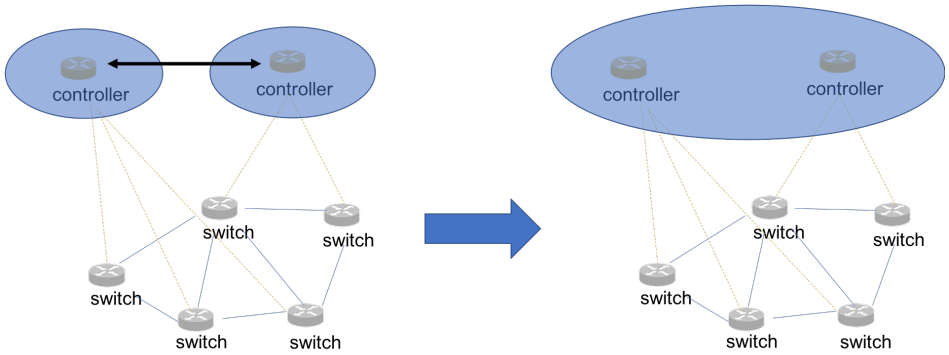


Figure: Logically centralized controller

Consistency models I

There are two types of consistency models.

- Strongly consistent:
  maintains a consistent global network view by synchronizing each of them immediately when a controller receives the changed network states inside same control domain.
- Eventually consistent:
  provides fast responsiveness. Each controller makes the consistent global network view by synchronizing at some later point.

Consistency models: Strongly consistent

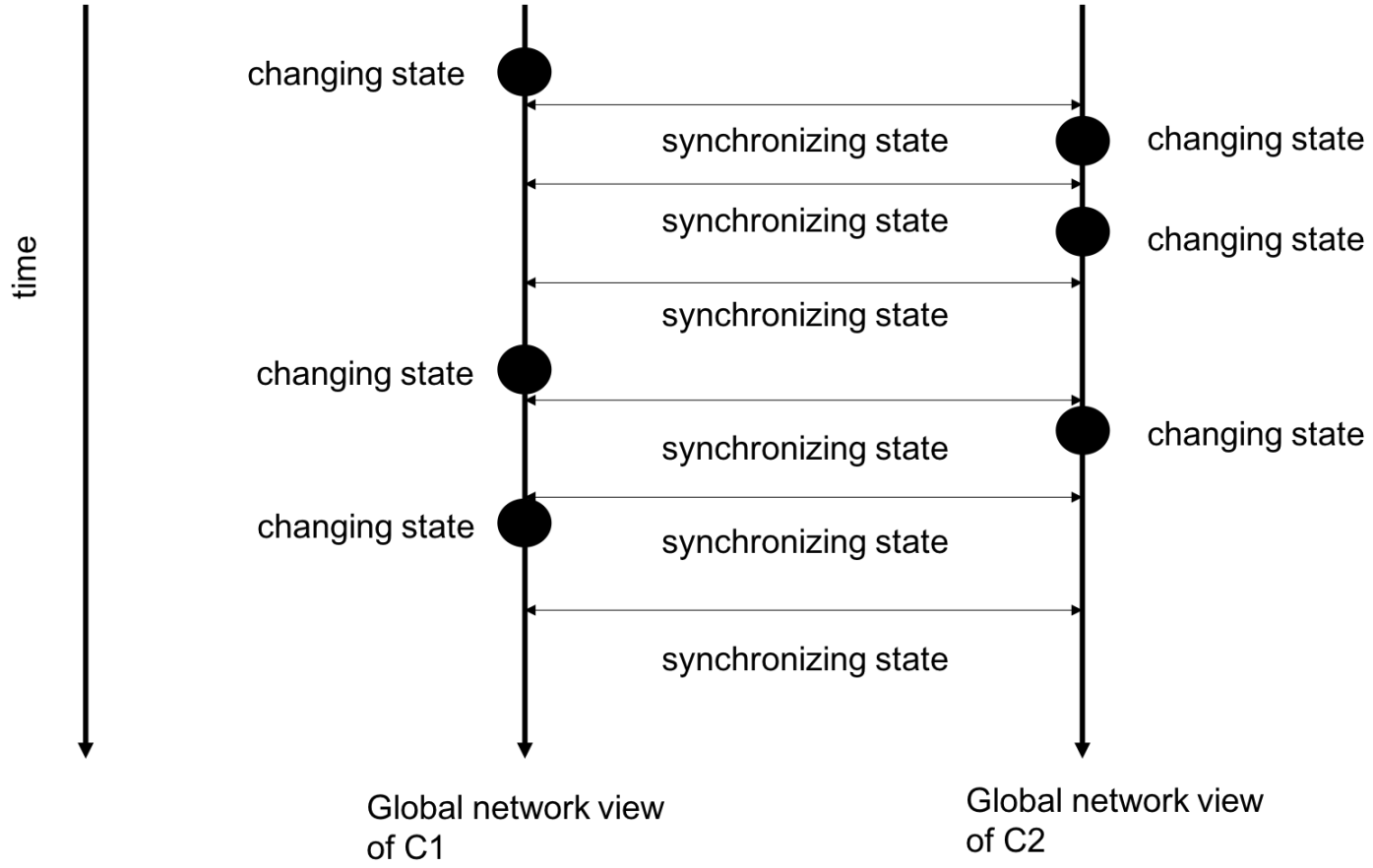


Figure: Strongly consistent

Consistency models: Eventually consistent

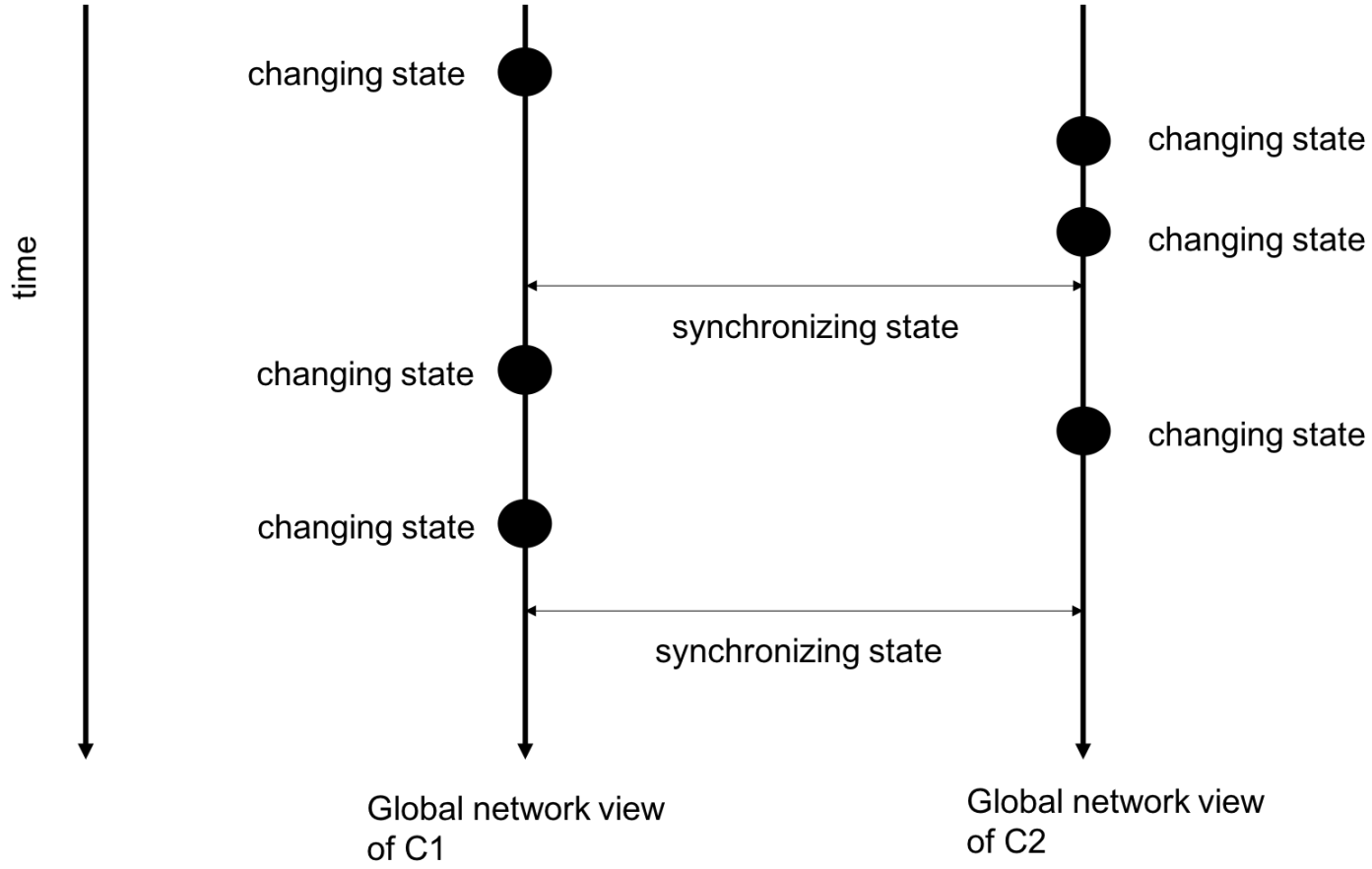


Figure: Eventually consistent

Consistency models I

- Strongly consistent:
  Overhead and delay of the controller are imposed, which the responsiveness is limited.
- Eventually consistent:
  It reacts faster and can cope with higher update rates. However, it is a possibility to use the inconsistent global network view.

Proposed scheme [12]

We propose a controller placement model based on traffic locality [12]. We focus on traffic flow between switches to reduce an inconsistent global network view.

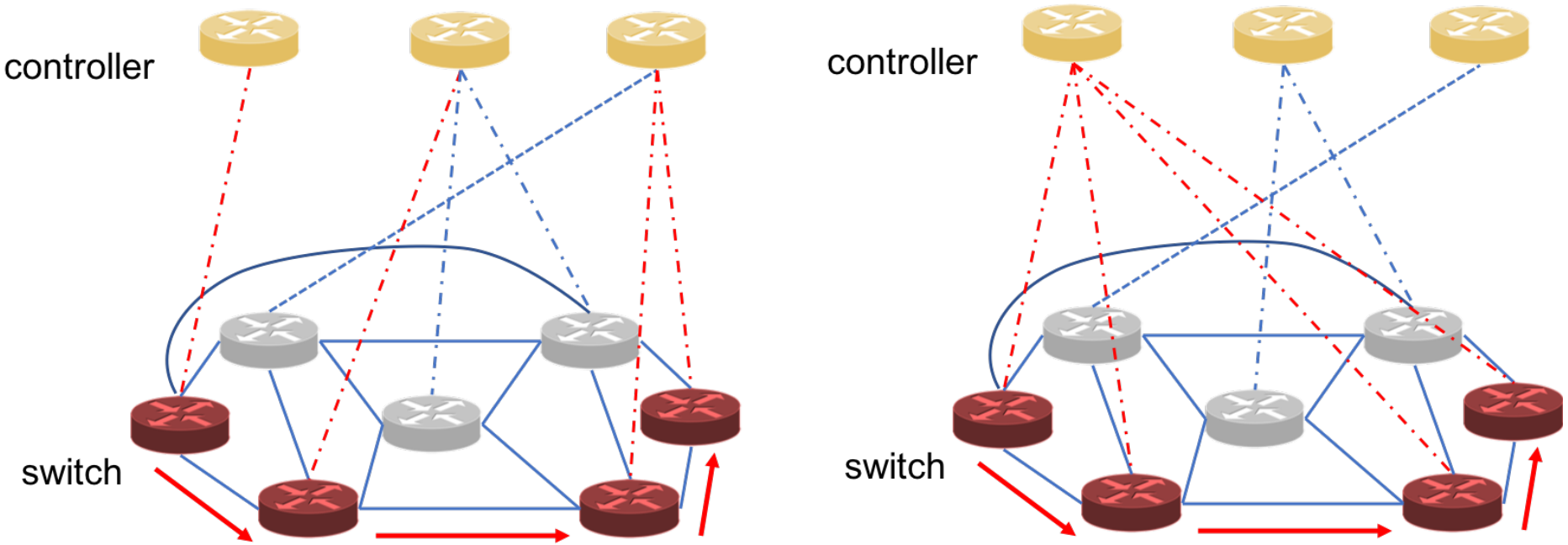


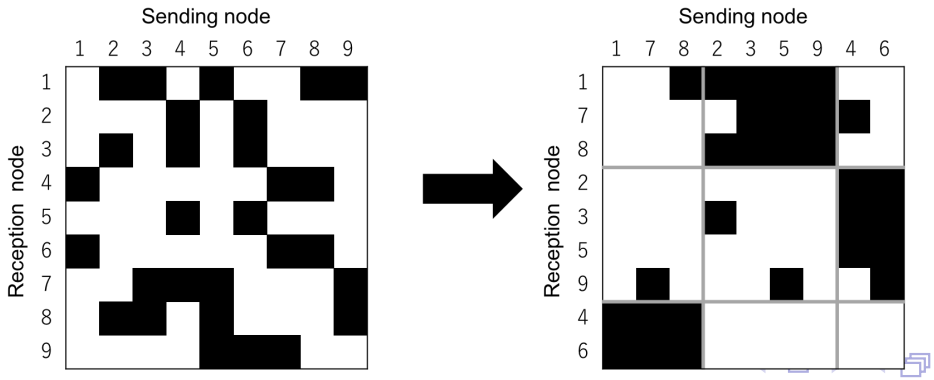
Figure: Controller placement model based on traffic volume

Clustering algorithm

- To minimize an inconsistent probability between controllers, we need to make an assignment of switches to controllers such that routing decision on each flow can be made by as small number of controllers as possible.
- In order to achieve such assignment of switches to controllers, we identify a cluster of switches such that fair amount of traffic flow remains inside the cluster and assign the switches belonging to the cluster to the same controller.

Infinite Relational Model (IRM)

- In order to make such clusters of switches, we employ Infinite Relational Model (IRM) [6].
- IRM [6] is a nonparametric Bayesian model that discovers relational structure in data sets as a set of clusters, where the number of clusters is not required to be fixed in advance.
- IRM divides a set of whole entities into a number of clusters automatically choosing an appropriate number of clusters, where a good set of partitions allows relationships between entities to be predicted by their cluster assignments.
- We used IRM in order to cluster switches into groups with high traffic volume.



Simulation model

- We compare an inconsistent probability that controllers have to use the consistent global network view form proposed assignment with random assignment.
- We used a full-mesh topology that connects 13 nodes representing 13 Prefectures in Japan (Tokyo, Kanagawa, Saitama, Ibaraki, Tochigi, Gumma, Chiba, Osaka, Kyoto, Hyougo, Shiga, Nara, and Wakayama).
- We create the traffic matrix by using the gravity model.

$$T_{ij} = G\frac{M_i M_j}{D_{ij}^2} \tag{1}$$

where $G = 1$ is constant of gravitation, M_i and M_j are the population volume of i and j, D_{ij}^2 is the square of the distance between i and j.

Number of requests

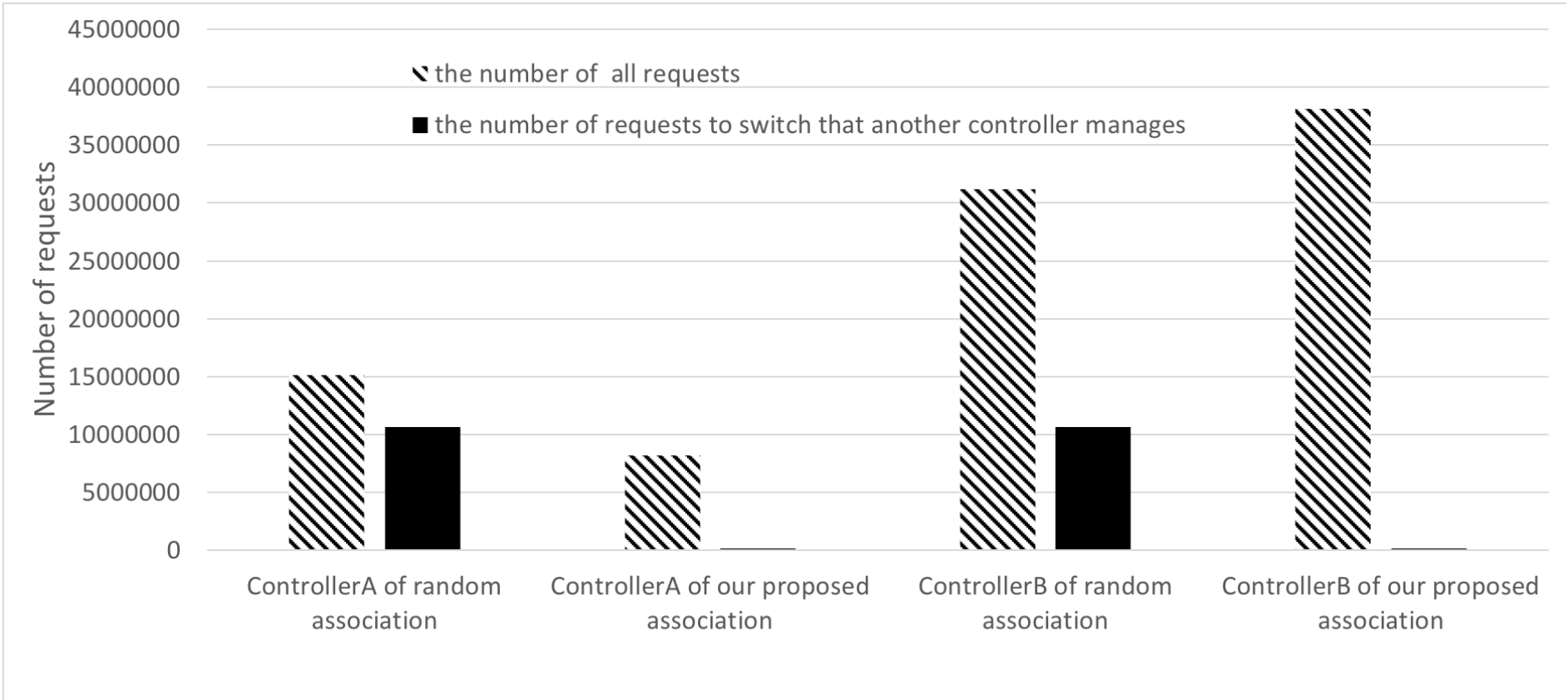


Figure: Number of requests that each controller received

Inconsistent probability

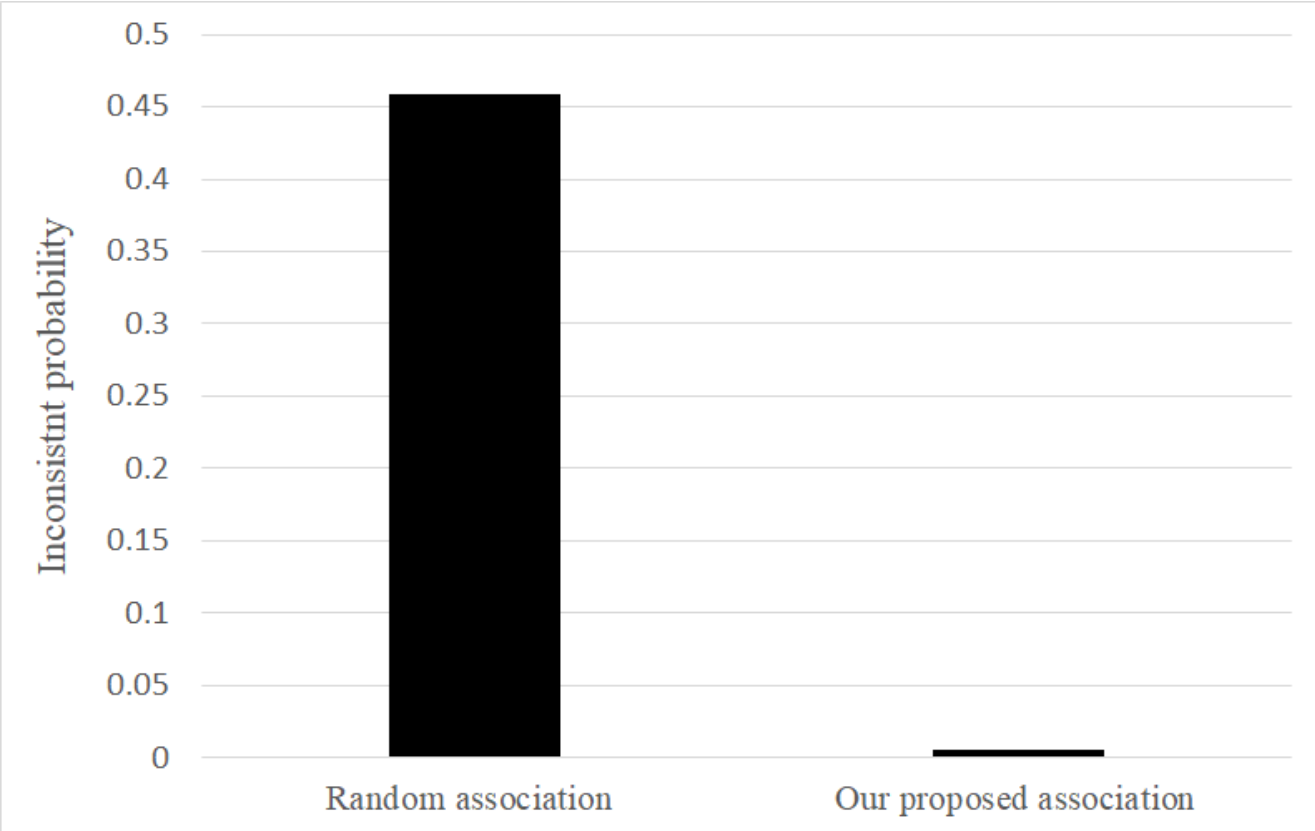


Figure: Inconsistent probability that controllers have to use the consistent global network view

Summary

We propose an assignment that reduce the inconsistent probability from a switch to the switch that assigned a different controller.

- We identify a cluster of switches such that fair amount of traffic flow remains inside the cluster and assign the switches belonging to the cluster to the same controller by using IRM.
- Simulation study shows that the proposed assignment reduces the inconsistent probability.

Future work includes (1) extensive evaluation model (topology, traffic) and (2) dynamic traffic.

Spatio-Temporal Human Mobility Prediction Based on Trajectory Data Mining for Resource Management in Mobile Communication Networks [3] (Outline) I

Spatio-Temporal Human Mobility Prediction Based on Trajectory Data Mining for Resource Management in Mobile Communication Networks [3]

Understanding human mobility

- Understanding the human mobility is crucial for resource management in mobile networks.
- Individual mobility versus Mass mobility.
  - Individual mobility:
    Understand individual mobility from the trajectory data of a user.
    Aggregate individual mobility data to asses the population in an area.
  - Mass mobility:
    Understand mass mobility directly from the population data in an area.

Trajectory

- GPS-equipped portable devices can record geographical position at each moment and transmit their trajectories to a collecting server.
- Increasing availability of large amounts of trajectory data provides useful knowledge.

Trajectory dataset

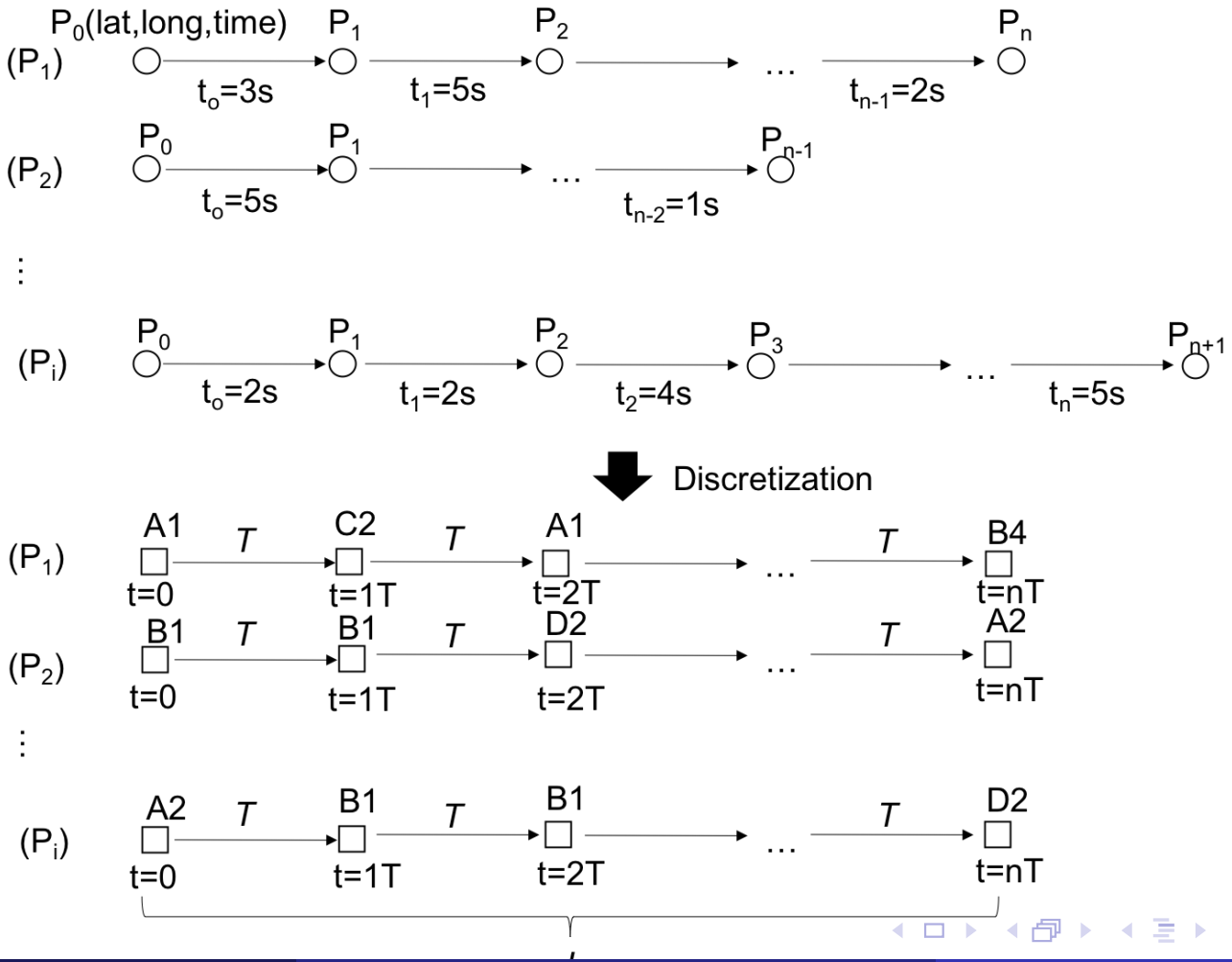
Geolife project [16]

- 182 users in a period of over five years (from April 2007 to August 2012).
- Sequence of time-stamped points, each of which contains the information of latitude, longitude and altitude.
- 17,621 trajectories with a total distance of 1,292,951kilometers and a total duration of 50,176 hours.
- 91.5 percent of the trajectories are logged in a dense representation, e.g. every 1–5 seconds or every 5–10 meters per point.

```
# latitude,longitude,0,altitude,Date,Date(string),Time(String)
40.000031,116.326378,0,492,39902.0362037037,2009-03-30,00:52:08
39.999924,116.326475,0,492,39902.0362615741,2009-03-30,00:52:13
39.99978,116.326453,0,492,39902.0363194444,2009-03-30,00:52:18
...
40.002051,116.324416,0,322,39902.0378819444,2009-03-30,00:54:33
```

Discretization in spatial-temporal domains

Resource management in cellular mobile networks; Granularity of spatial-temporal discretization are the geographical area, cell, and the control cycle of resource management in the cellular mobile networks.



Discretization in spatial-temporal domains

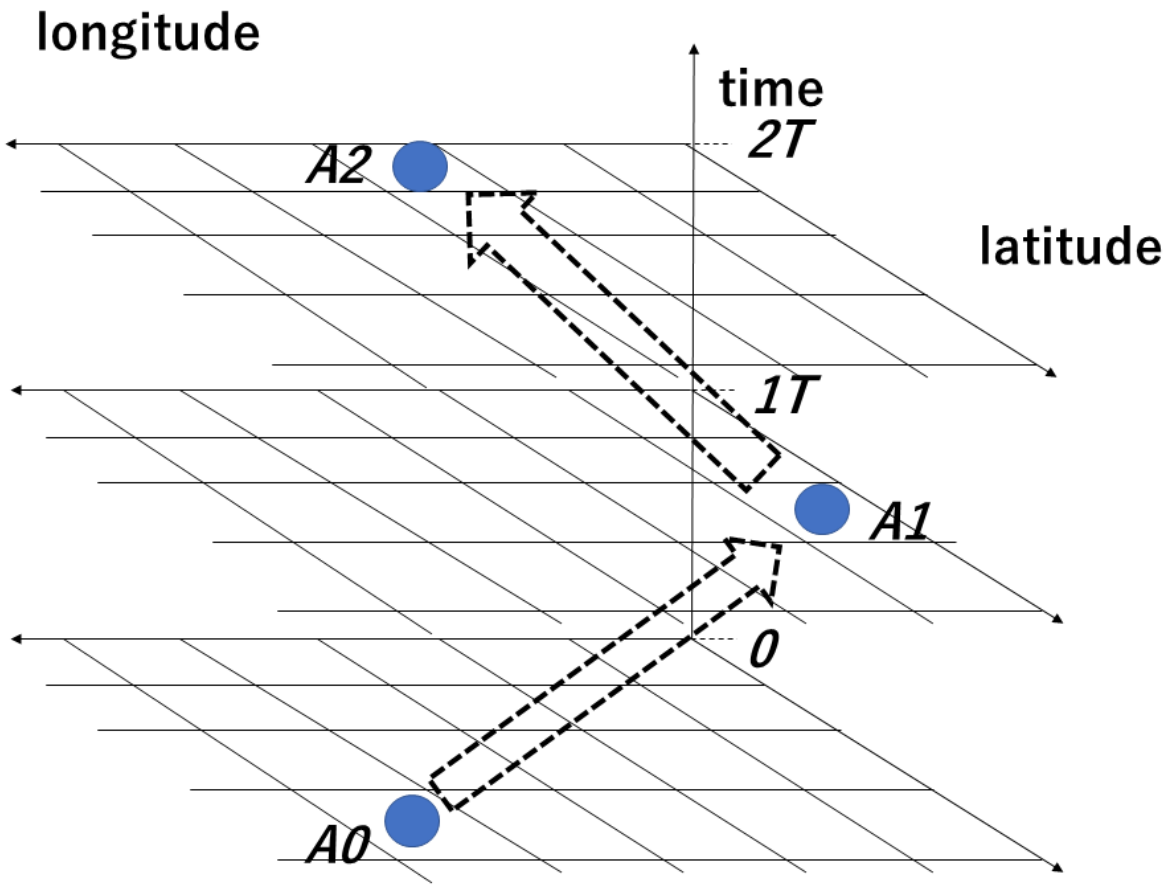


Figure: A sample of spatio-temporal movements of trajectory data

Discretization in spatial-temporal domains

Pa1	BM26	Support:6883
Pa2	BM26 BM26	Support:6574
Pa3	BM26 BM26 BM26	Support:6445
Pa4	BM26 BM26 BM26 BM30	Support:183
Pa5	BM26 BM26 BM26 BM30 BM30	Support:138
Pa6	BM26 BM26 BM26 BM30 BM30 BM26	Support:32
Pa7	BM26 BM26 BM26 BM30 BM30 BM26 BM26	Support:31

Figure: Discretization of trajectory pattern

Sequential pattern mining [1]

- Let $I = \{i_1, i_2, ..., i_n\}$ be a set of all items.
- A sequence S is an ordered list of items, denoted by $\langle e_1, e_2, ..., e_m \rangle$, where e_i is an item, i.e., $e_i \subseteq I$ for $1 \leq i \leq m$.
- A sequence $S_a = \langle a_1 a_2 \ldots a_m \rangle$ is called a subsequence of another sequence $S_b = \langle b_1 b_2 \ldots b_n \rangle$, denoted as $S_a \sqsubseteq S_b$, if S_b includes S_a, i.e., if there exist integers $1 \leq j_1 < j_2 < \cdots < j_m \leq n$ such that $a_1 = b_{j_1}, a_2 = b_{j_2}, \ldots, a_m = b_{j_m}$.

Frequent sequential pattern

- A sequence database SDB is a set of tuples $\langle sid, S \rangle$, where sid is a sequence identifier and S is a sequence.
- A tuple $\langle sid, S \rangle$ is said to contain a sequence S_a, if S_a is a subsequence of S, i.e., $S_a \sqsubseteq S$.
- The support of a sequence S_a in a sequence database SDB is the number of tuples in the database containing S_a, i.e., $support(S_a) = \mid \{\langle sid, S \rangle \mid (\langle sid, S \rangle \in SDB) \bigwedge (S_a \sqsubseteq S)\} \mid$.
- Given a positive integer ξ as the support threshold (we call $minsupp$), a sequence S_a is called a frequent sequential pattern in sequence database SDB if the sequence is contained by at least ξ tuples in the database, i.e., $support_{SDB}(S_a) \geq \xi$.

Sequential pattern mining algorithms

- Many sequential pattern mining algorithms have been proposed such as PrefixSpan [9], SPAM [2], LAPIN-SPAM [15], PRISM [4], and BIDE [14].
- Those algorithms except BIDE produce all frequent sequential patterns with a minimum support threshold *minsupp*.
- BIDE produces frequent *closed* sequential patterns, thereby producing more compact and efficient results.

frequent *closed* sequential pattern

We call S_a a frequent *closed* sequential pattern if a sequence S_a is a frequent sequence pattern and there exist no supersequence of S_a with the same support; there exists no S_b such that $S_a \sqsubseteq S_b$ and $sup(S_b) = sup(S_a)$.

Closed sequential pattern

Table: An example sequence database SDB

Sequence1	<a, b, c >
Sequence2	<a, b, c, d>
Sequence3	<a, b, c, e>

Suppose that $minsupp = 3$.

- PrefixSpan extracts all frequent sequential patterns: $\langle a \rangle$, $\langle b \rangle$, $\langle c \rangle$, $\langle a,b \rangle$, $\langle b,c \rangle$, $\langle a,b,c \rangle$ $\langle a \rangle$, $\langle b \rangle$, $\langle c \rangle$, $\langle a,b \rangle$, $\langle b,c \rangle$.
- BIDE extracts only $\langle a,b,c \rangle$ as a frequent closed sequential pattern.

PrefixSpan

The algorithm of the PrefixSpan with $minsupp = 2$.

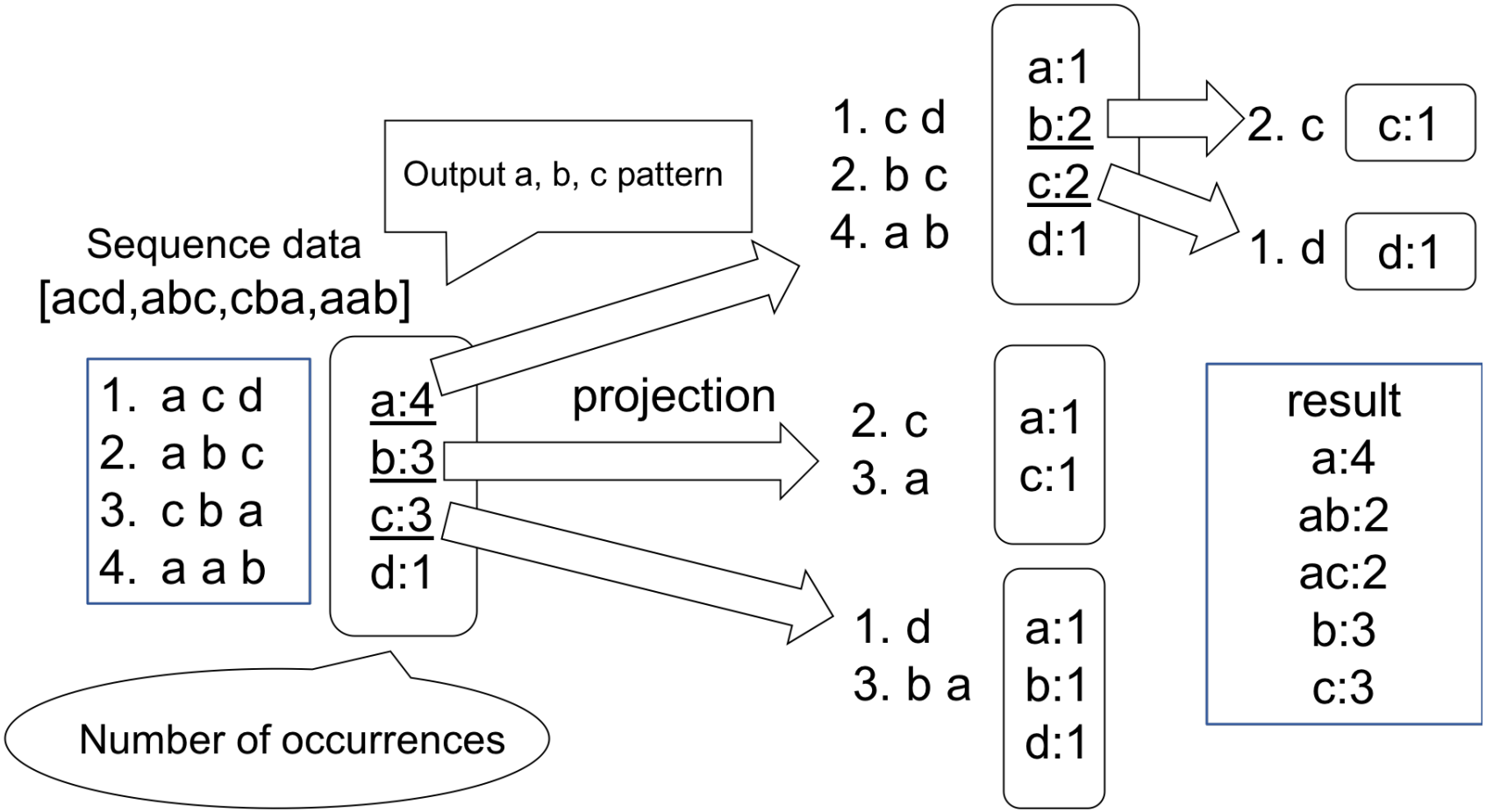


Figure: PrefixSpan

BIDE

BIDE [14] employs a *closure checking* procedure when it grows a frequent prefix sequence in order to assure that it is genuinely closed.

- It performs two kinds of checking for this: forward checking and backward checking.
- If a sequence $S_a = \langle a_1 a_2 \dots a_m \rangle$ is non-closed, it can be extended to S'_a by adding an item a' in three ways:
  1. $S'_a = \langle a_1 \dots a_m a' \rangle$ and $sup(S') = sup(S)$,
  2. $S'_a = \langle a_1 \dots a' a_m \rangle$ and $sup(S') = sup(S)$, and
  3. $S'_a = \langle a' a_1 \dots a_m \rangle$ and $sup(S') = sup(S)$.
- In the first case, an item a' occurs only after the prefix sequence S and we call S' a *forward extension* sequence of S.
- In the latter two cases, an item a' occurs before the end of prefix sequence S and we call S' *backward extension* sequence of S.

Proposed method [3]

The proposed method [3] consists of three steps: (1) discretization of trajectory data, (2) extraction of frequent trajectory patterns, and (3) prediction of future trajectory.

- It discretizes trajectory data in spatial-temporal domains.
- It mines them to make a frequent trajectory pattern database by employing sequential pattern mining algorithms: PrefixSpan or BIDE.
- It predicts the future trajectory from the frequent trajectory pattern database. It computes a similarity score between two trajectory patterns to predict the future trajectory.

Trajectory mining

Once we represent trajectory data as sequential pattern, we extract frequent trajectory patterns using sequential pattern mining algorithms such as PrefixSpan or BIDE.

Pa1	BM26	Support:6883
Pa2	BM26 BM26	Support:6574
Pa3	BM26 BM26 BM26	Support:6445
Pa4	BM26 BM26 BM26 BM30	Support:183
Pa5	BM26 BM26 BM26 BM30 BM30	Support:138
Pa6	BM26 BM26 BM26 BM30 BM30 BM26	Support:32
Pa7	BM26 BM26 BM26 BM30 BM30 BM26 BM26	Support:31

Figure: An example frequent trajectory pattern

Trajectory prediction

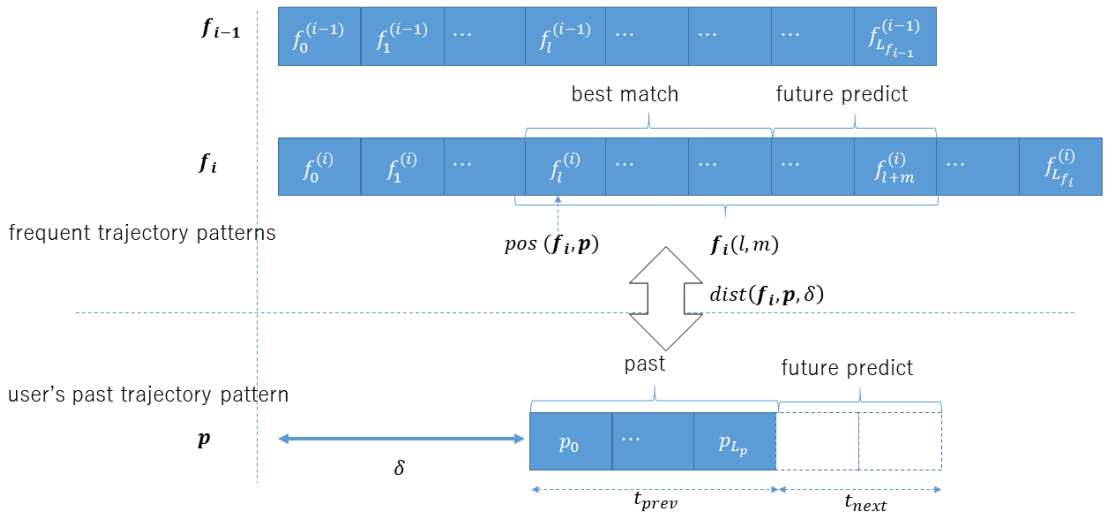
Once we obtain frequent trajectory pattern database, we identify the frequent trajectory pattern most similar to the trajectory of interest.

- The similarity score $score(\boldsymbol{f_i}, \boldsymbol{p})$:

$$score(\boldsymbol{f_i}, \boldsymbol{p}) = \min_{all\delta} dist(\boldsymbol{f_i}, \boldsymbol{p}, \delta).$$

- The index of most similar frequent trajectory pattern $mfp(\boldsymbol{F}, \boldsymbol{p})$:

$$mfp(\boldsymbol{F}, \boldsymbol{p}) = \arg\min_{all\boldsymbol{f_i} \in \boldsymbol{F}} score(\boldsymbol{f_i}, \boldsymbol{p}).$$



Trajectory prediction

The predicting algorithm exploits a set of frequent trajectory patterns.

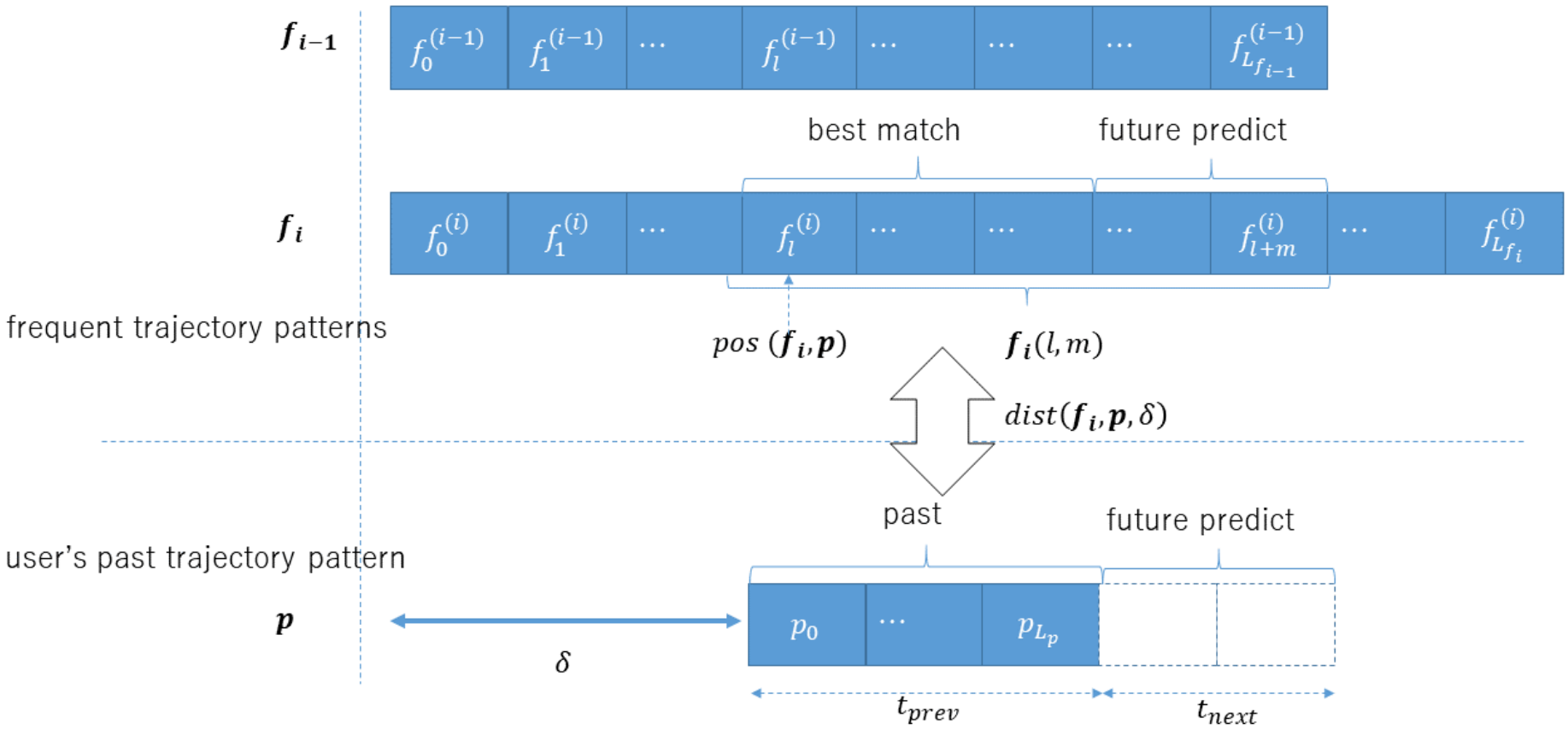


Figure: Computing the partial frequent trajectory pattern most similar to the

Dataset: Geolife project [16]

We use GPS trajectory dataset collected in Geolife project by 178 users [17].

- This dataset contains 17,621 trajectories.
- These trajectories were recorded by different GPS loggers and GPS-phones.
- GPS trajectory of this dataset is represented by a sequence of time-stamped points, each of which contains the information of latitude, longitude and altitude.

Experiments setting

We divide Trajectory dataset into mining data and verification data.

- Frequent trajectory patterns are derived using mining data.
- Verification data is used to evaluate the prediction accuracy of the proposed trajectory prediction method.

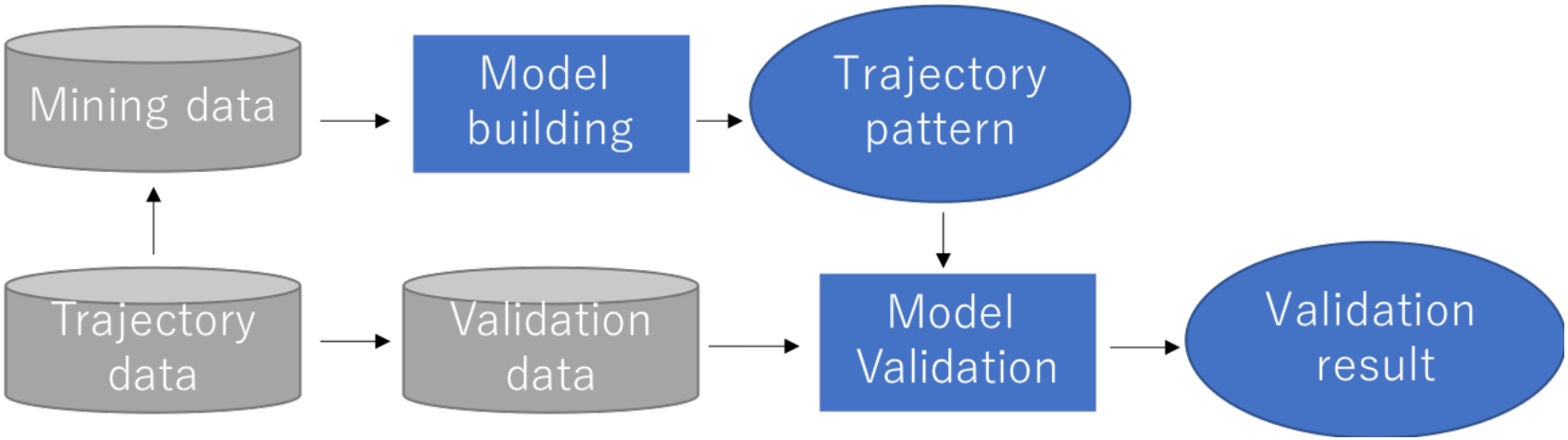


Figure: Experimental setting.

Number of trajectory patterns

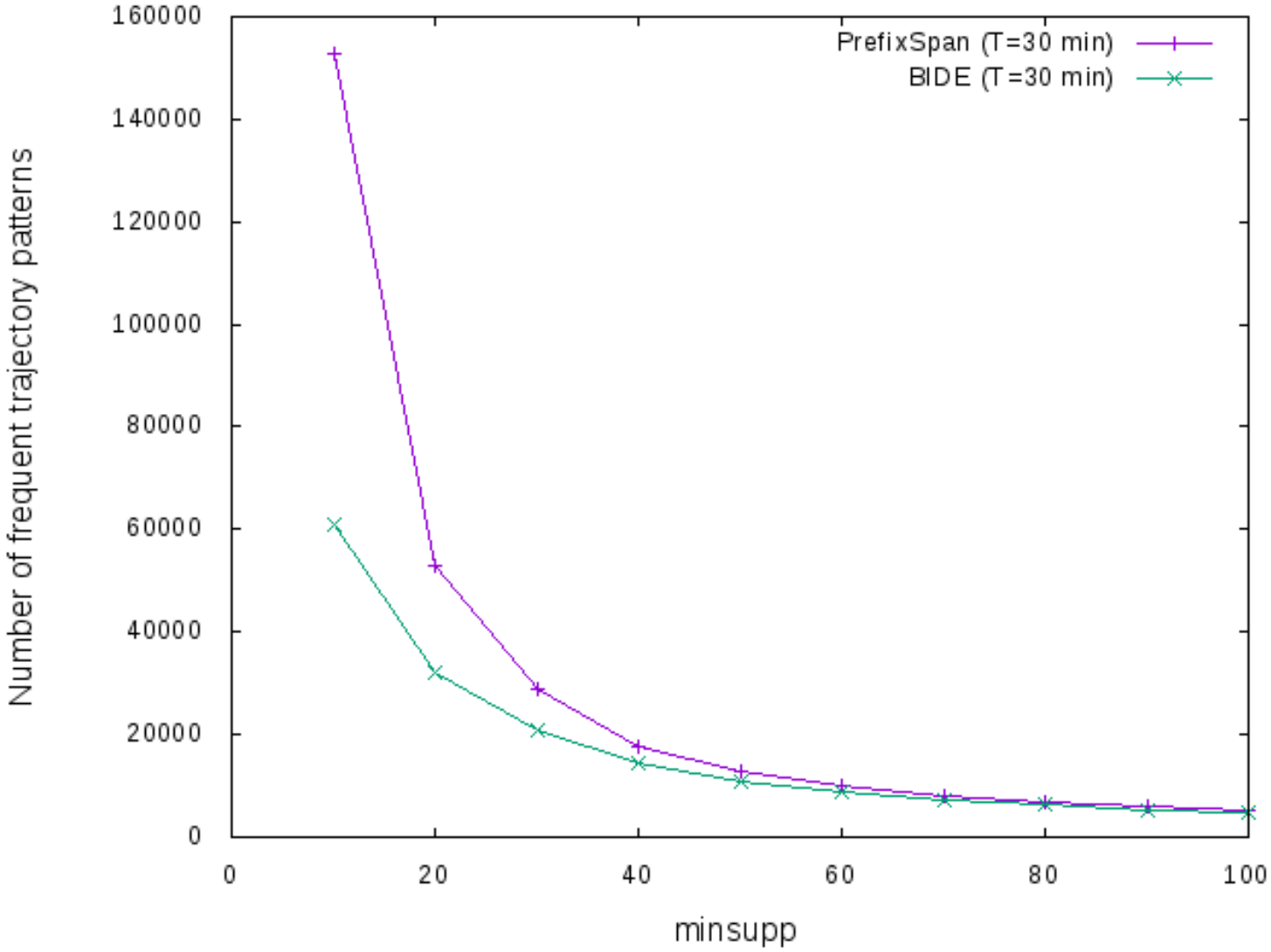


Figure: Number of frequent trajectory patterns: T=30min.

Prediction of trajectory

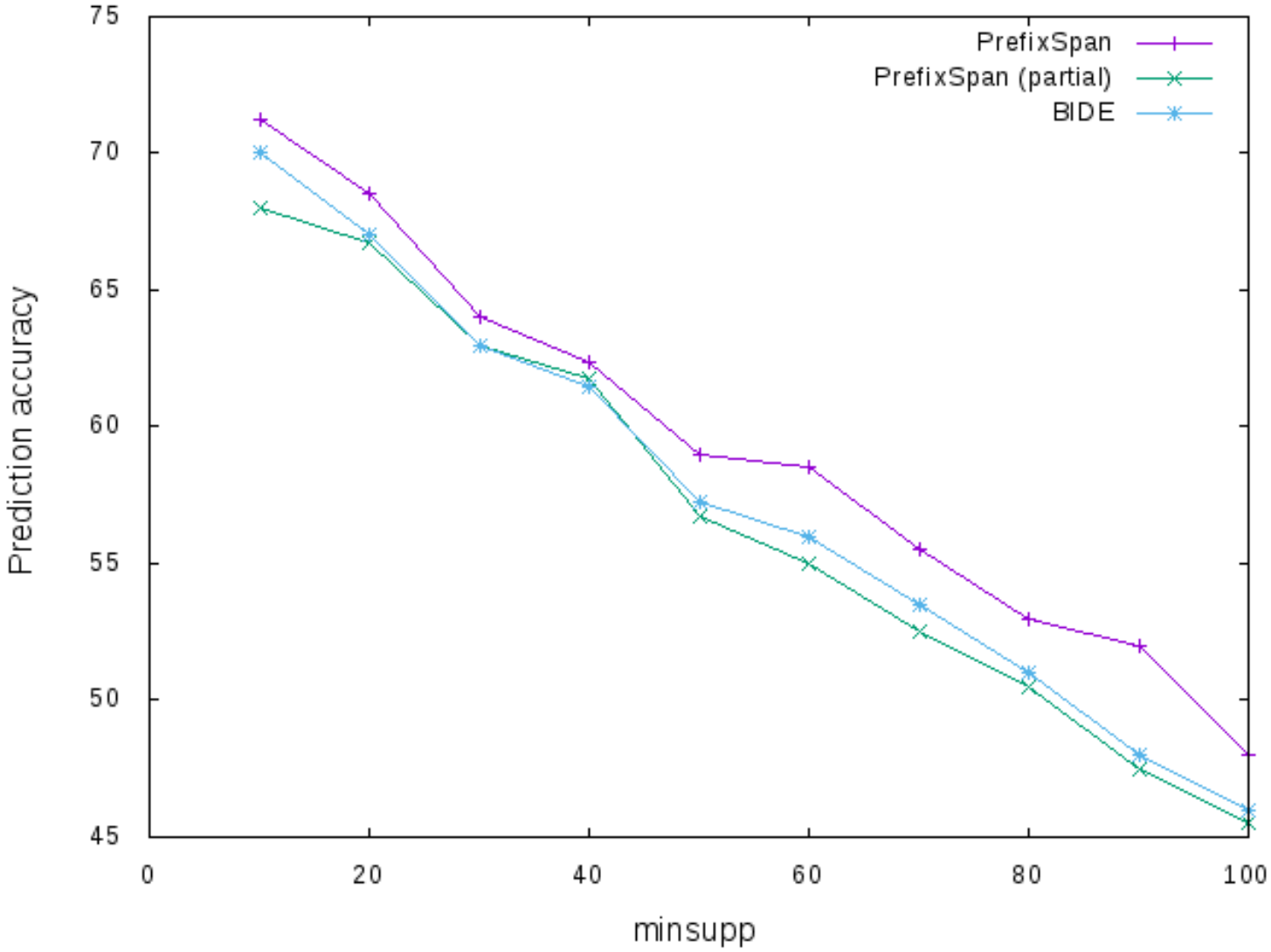


Figure: Prediction accuracy obtained by the methods using the frequent trajectory

Summary

We proposed a human mobility prediction method for resource management in mobile networks.

- We propose a similarity score between two trajectory patterns to predict the future trajectory pattern.
- We evaluate the proposed method using real data set (GeoLife project) [17].
- We confirm that BIDE produces more efficient set of frequent trajectory patterns than PrefixSpan while the proposed method predicts the mobility with the accuracy of 45 % to 70 %.

Future work includes (1) aggregation of human mobility data for resource management in mobile networks, (2) parallel computation to improve scalability.

Few-shot Learning for eNodeB Performance Metric Analysis for Service Level Assurance in LTE Networks [18](Outline) I

Few-shot Learning for eNodeB Performance Metric Analysis for Service Level Assurance in LTE Networks [18]

Service Level Assurance in LTE Networks

- In LTE networks, the radio communication services to the mobile users are provided by eNodeBs.
- A large number of eNodeBs are deployed to cover the entire service areas spanning various kinds of geographical regions.
- The state of the eNodeB can be diverse: radio coverage and channel interference, traffic load in control and data planes, system hardware and software issues, service management issues.
- The state of eNodeB that covers a specific area impacts the service level offered to the customer in that area.
- It is crucial to understand the state of the eNodeB that covers the area.

Analysis of eNodeB's KPI Data for quality management

- By analyzing the performance metric generated by eNodeB, we expect to understand the state of the area that the eNodeB covers.
- Each eNodeB generates a large number of key performance indicators (KPIs).
- Operators need to handle hundreds of millions of KPIs to cover the areas.
- It is impractical to handle manually such a huge amount of KPI data, and automation of data processing is therefore desired.

eNodeB's KPI data

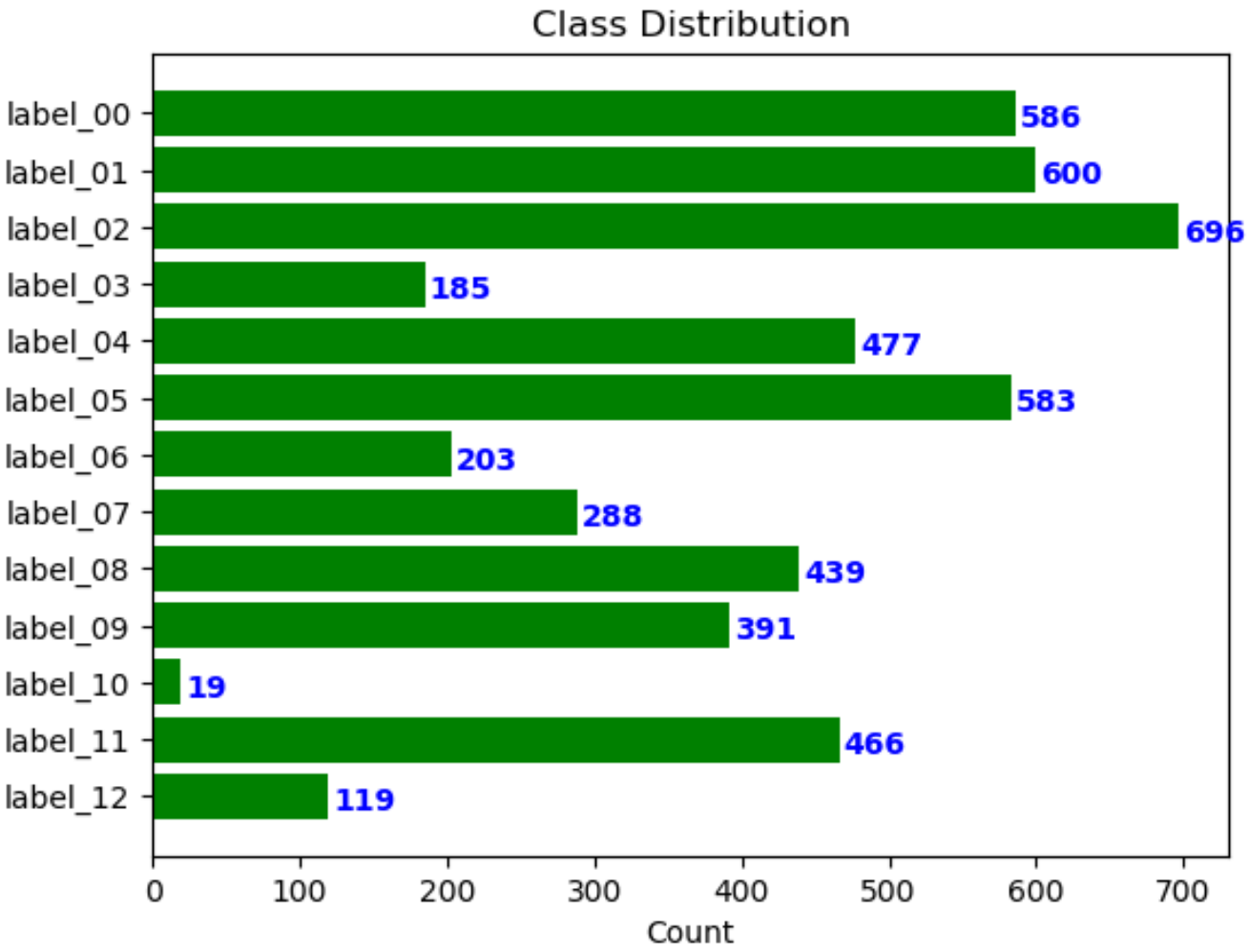


Figure: The number of labeled data for each eNodeB status.

Labeling Data

- A typical workflow of applying machine learning and/or deep learning comprises of
  - (1) the pre-training phase where the classification is done using unsupervised learning,
  - (2) the data annotation phase where the human operators manually examine the data and attache the label to the data, and
  - (3) the supervised learning phase where the classifier is trained using a set of labeled data.
- Many supervised machine learning methods need to be trained with training data annotated with the correct labels.
- In order to build a good supervised machine learning model, we need a large number of training data.

Small Labeled Data

- For the task of classification, the existing supervised learning algorithms require a dataset of high quality and quantity of human annotated data for training.
- To minimize the human labor intensive and time-consuming dataset annotation task, it is thus required to find a data efficient learning algorithm/technique to build a classifier model.
- Anomalies are difficult to occur in practice, so the anomaly classes are usually sparse in the dataset. It is extremely important for the operators to deal with a unbalanced data set where a few class has only handful data instances while others have a lot of data instances.

Few-shot Learning

- Supervised learning requires large amount of labeled training dataset to build a classifier model.
- Few-shot learning is a task to adapt new classes not seen in the training dataset, given only a few labeled example of these classes.
- To build such a model, recent research employs meta-learning concept that is trained using a set of episodes each with a small number of labeled data for training and test.
- Thus, in order to address the above issues in applying machine learning algorithms to the eNodeB performance metric analysis, we propose to use few-shot learning [18].

Prototypical Network I

- Prototypical Network [10] is a few-shot learning that use neural networks (NNs).
- Prototypical Network is based on the idea that the data for each class is distributed near average of themselves.
- It uses the training set to extract a prototype vector from each class as shown in Fig. 20, and then classifies the inputs in the training set based on their distance to the prototype of each class as shown in Fig. 21.

Prototypical Network

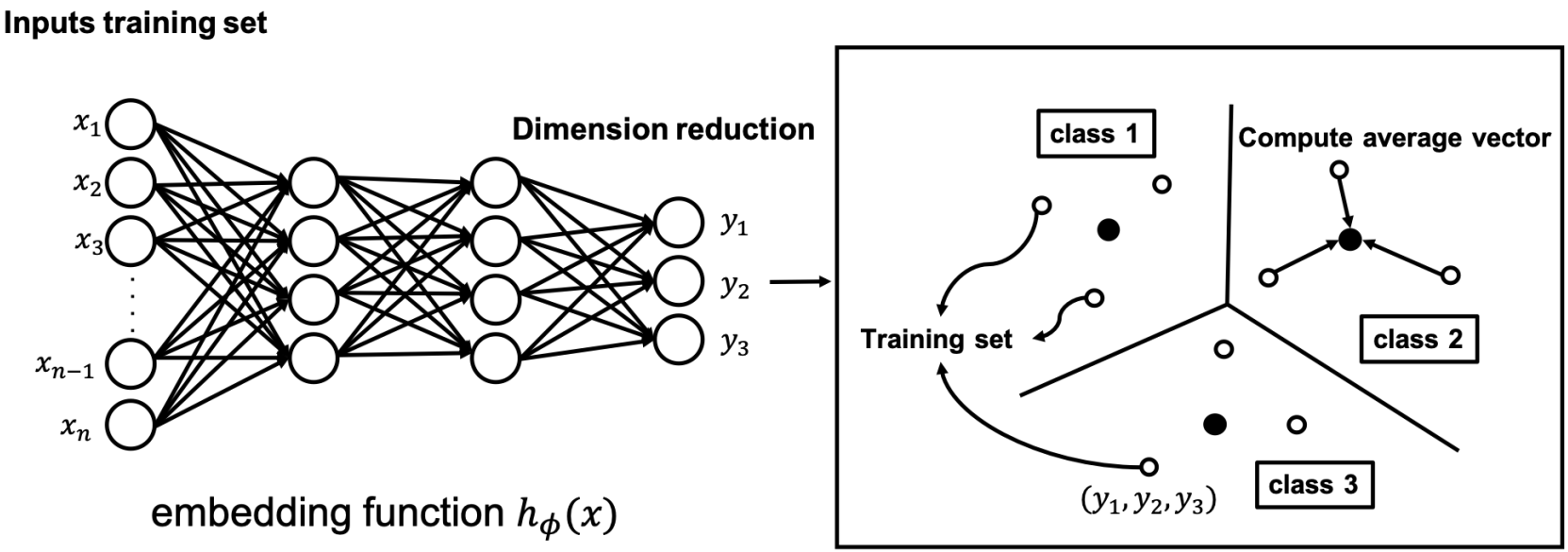


Figure: Left: Embedding function. Right: Prototype vector calculation using training set.

Prototypical Network

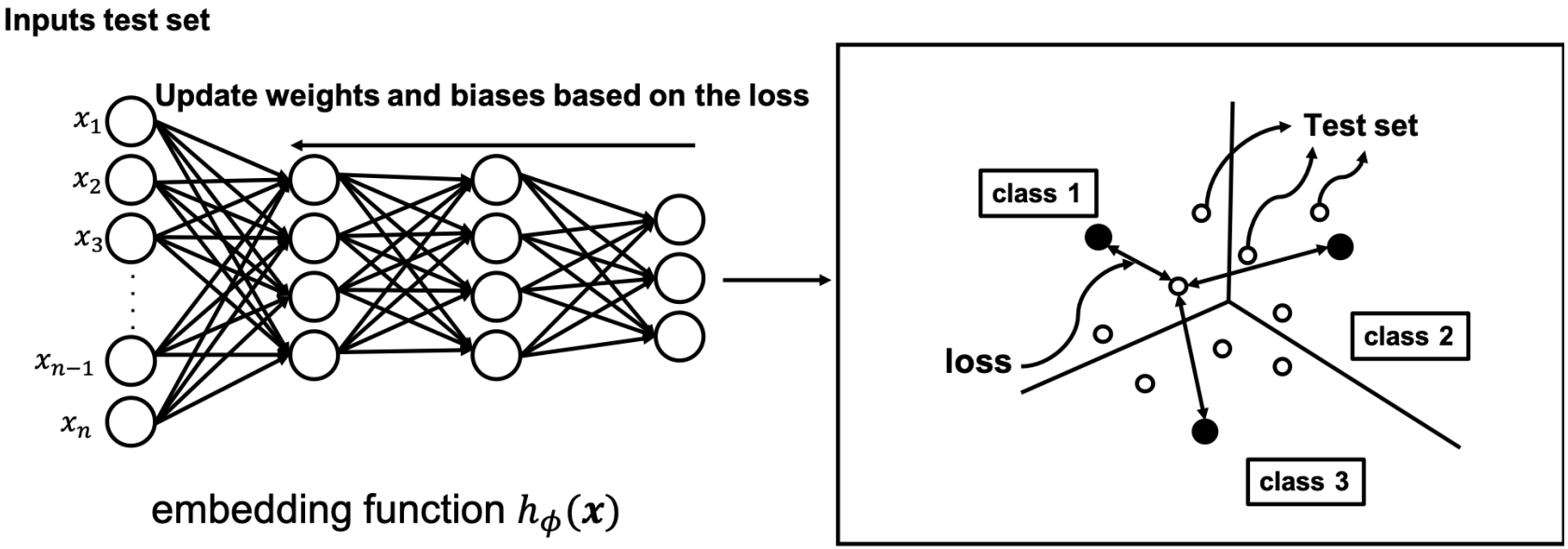


Figure: Left: Embedding function (same as Fig. 20). Right: Loss calculation according to distance using test set.

Embedding Function

- Embedding function maps discrete object eNodeB performance metrics to a vector of real numbers so that similar objects are close.
- We investigate which neural network is suitable for classifying status of eNodeB in Prototypical Network. We focus on three types of neural networks: multilayer perceptron (MLP) [5], two-dimensional convolutional neural networks (2D-CNN) [8] and one-dimensional convolutional neural networks (1D-CNN).

Dataset I

- We collected the 5052 records from eNodeBs deployed in a production networks.
- The state of the eNodeB is categorized into 13 labels as listed in Table 2.
- The number of labeled data for each eNodeB status is shown in Table 22.

Dataset II

Table: Label associated with eNodeB status.

	Status
label_00	High traffic condition
label_01	Weak radio wave coverage
label_02	High transmission of uplink and downlink data
label_03	High transmission of downlink data
label_04	High transmission of uplink data
label_05	Normal operating condition
label_06	Uplink control channel interference
label_07	Uplink traffic channel interference
label_08	Both control and traffic channel interference
label_09	Uplink control channel radio quality
label_10	System hardware processor load
label_11	Service setup issue
label_12	Control channel signaling

Dataset

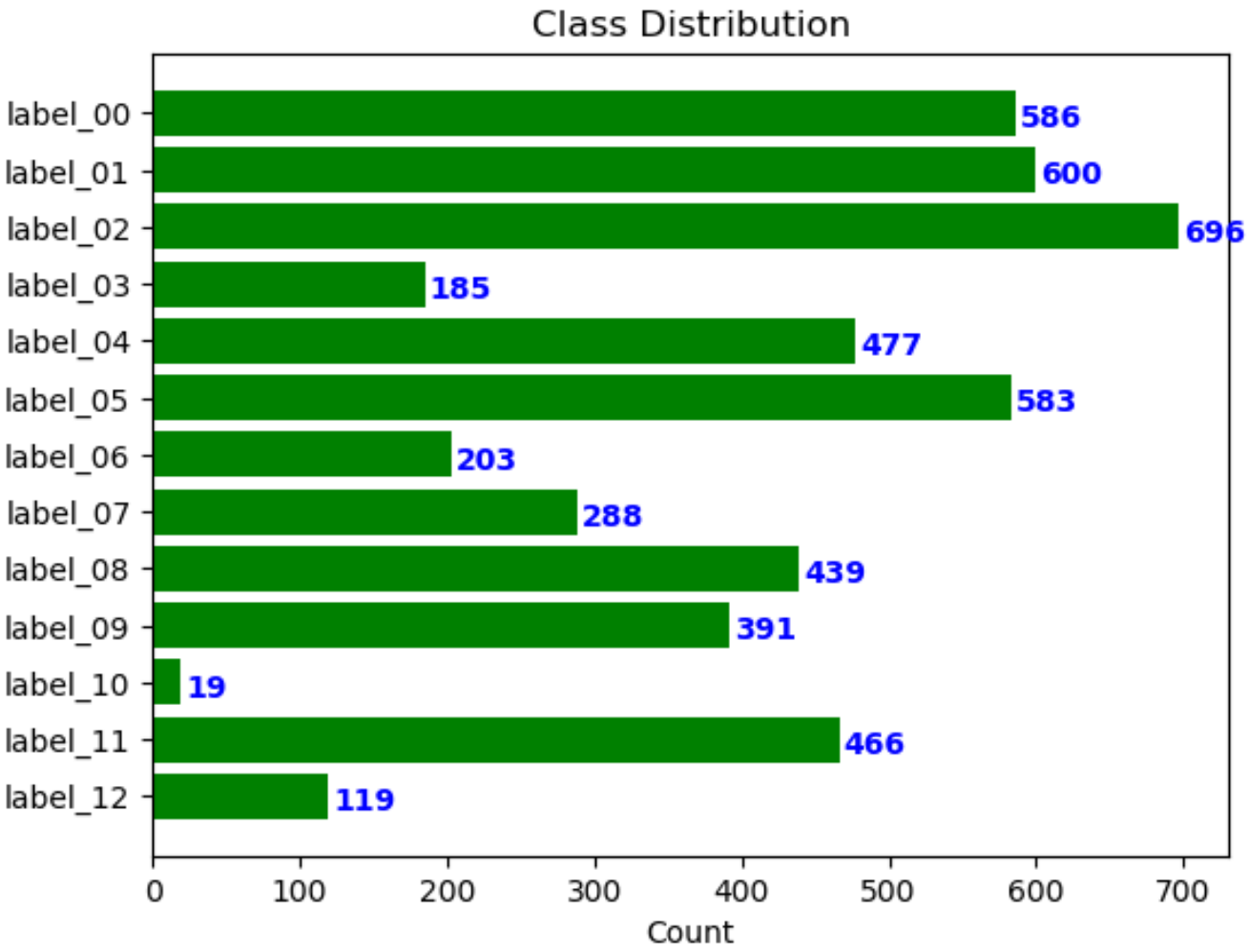


Figure: The number of labeled data for each eNodeB status.

Parameters

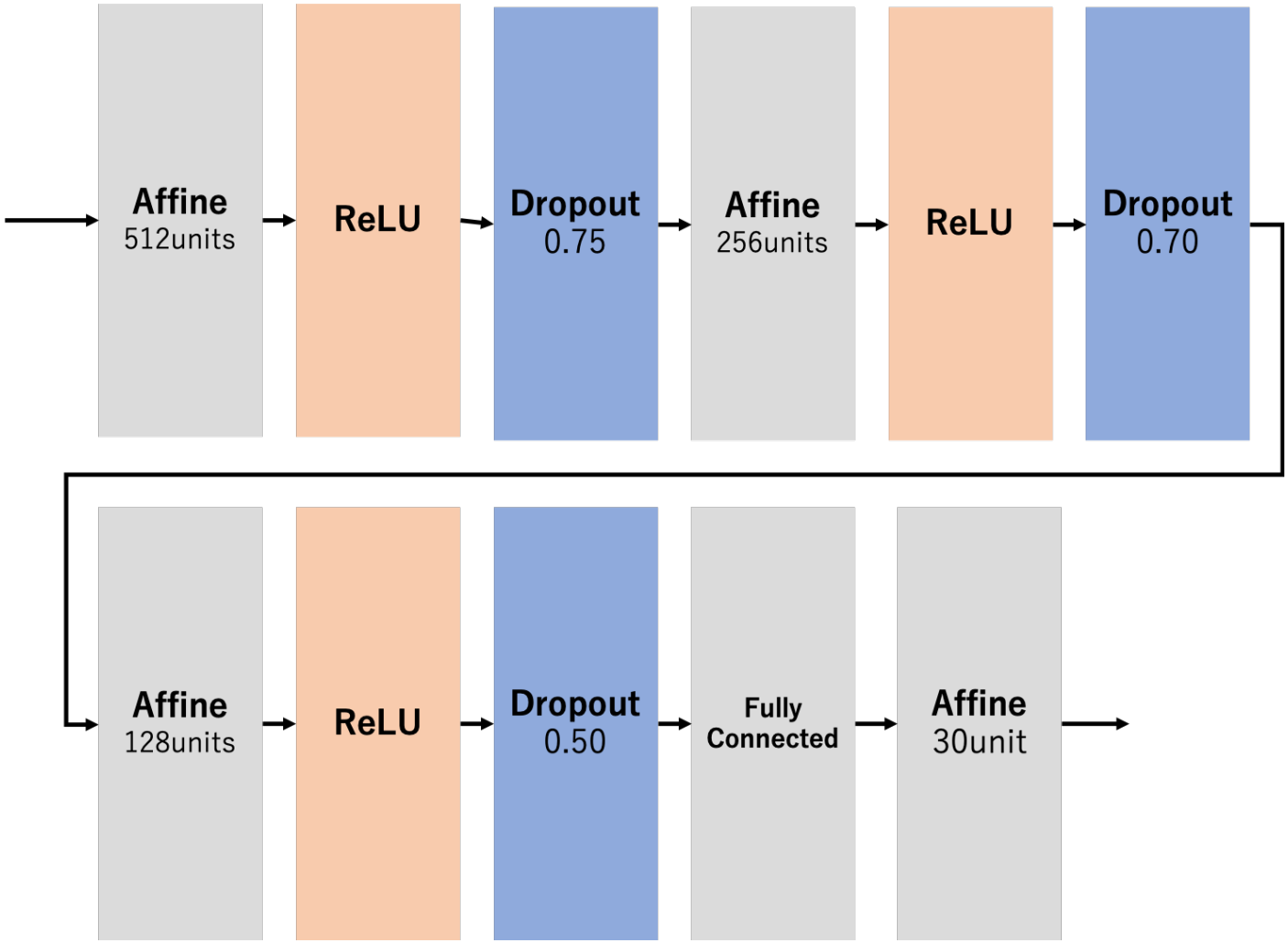


Figure: Parameters of MLP

Parameters

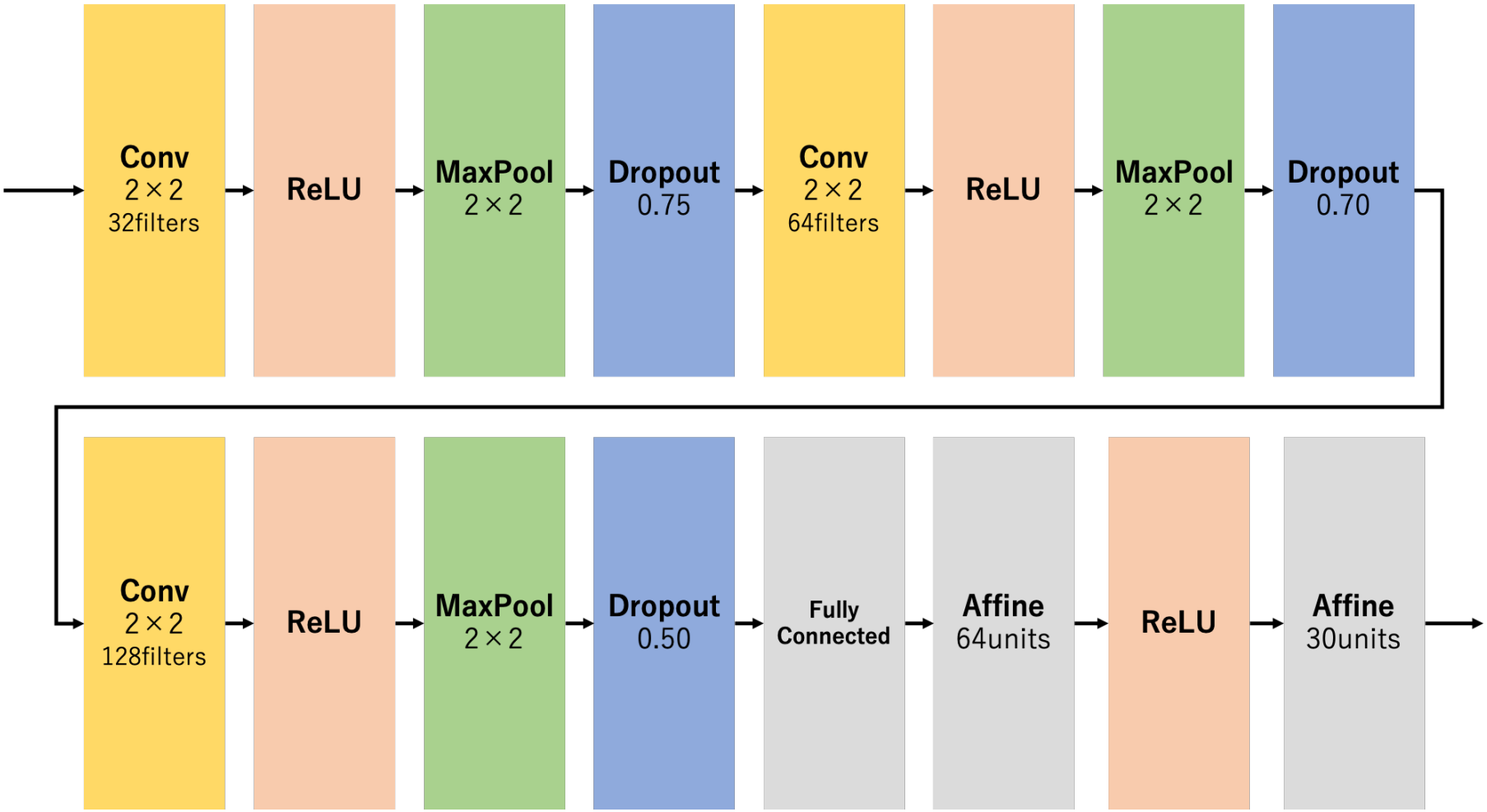


Figure: Parameters of 2D-CNN

Parameters

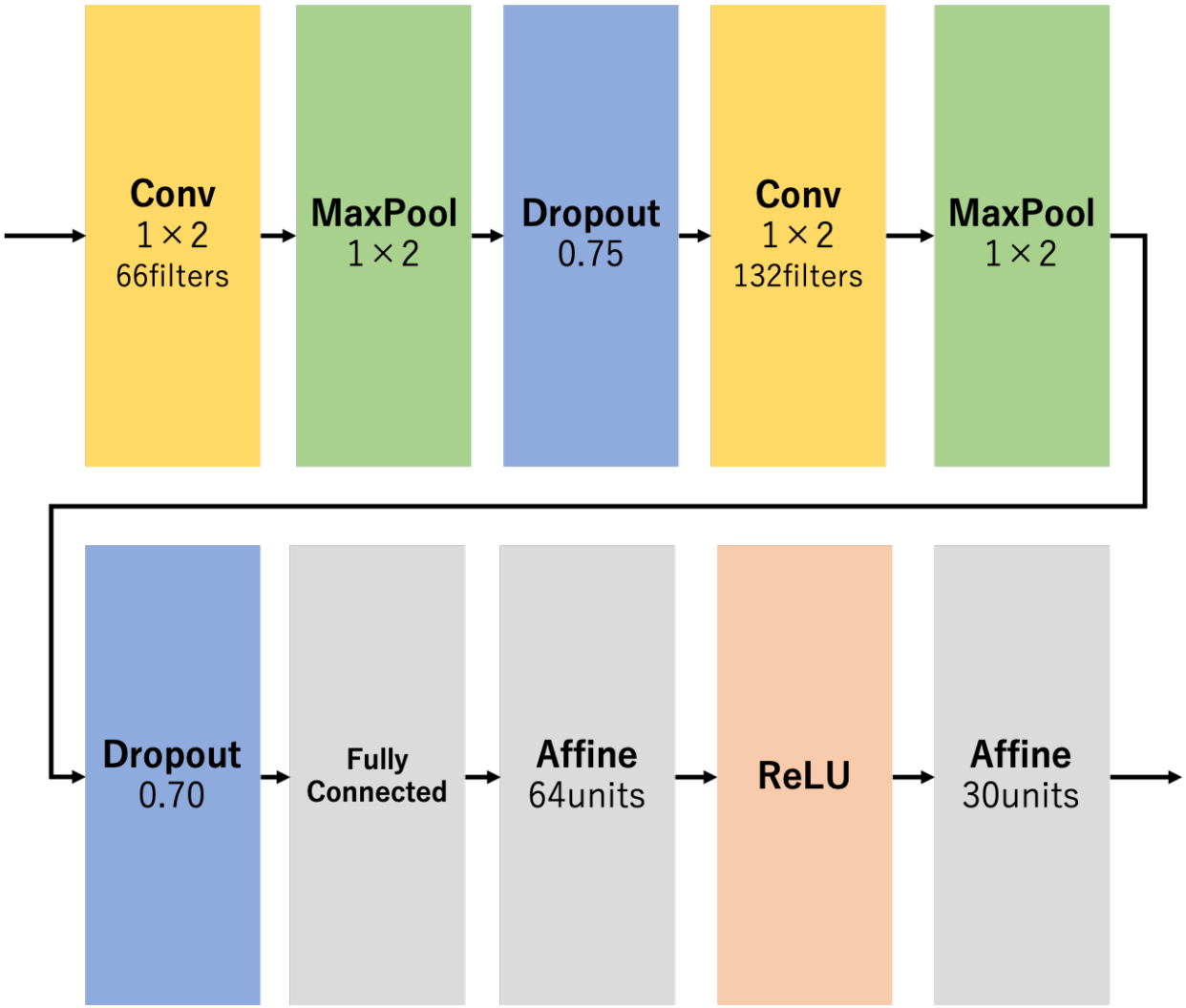


Figure: Parameters of 1D-CNN

Performance Evaluation

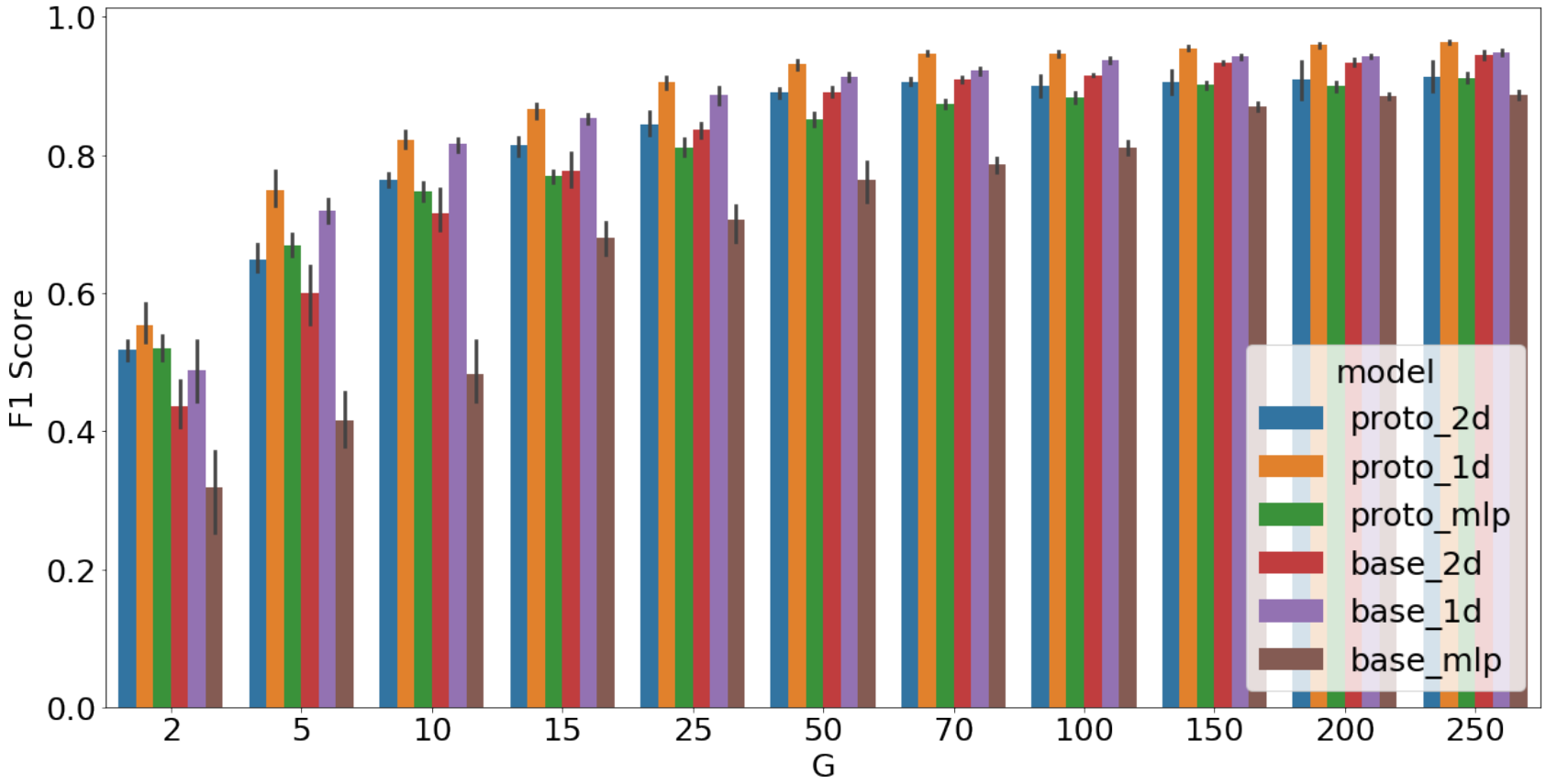


Figure: Comparison between Prototypical Network and baseline.

Performance Evaluation

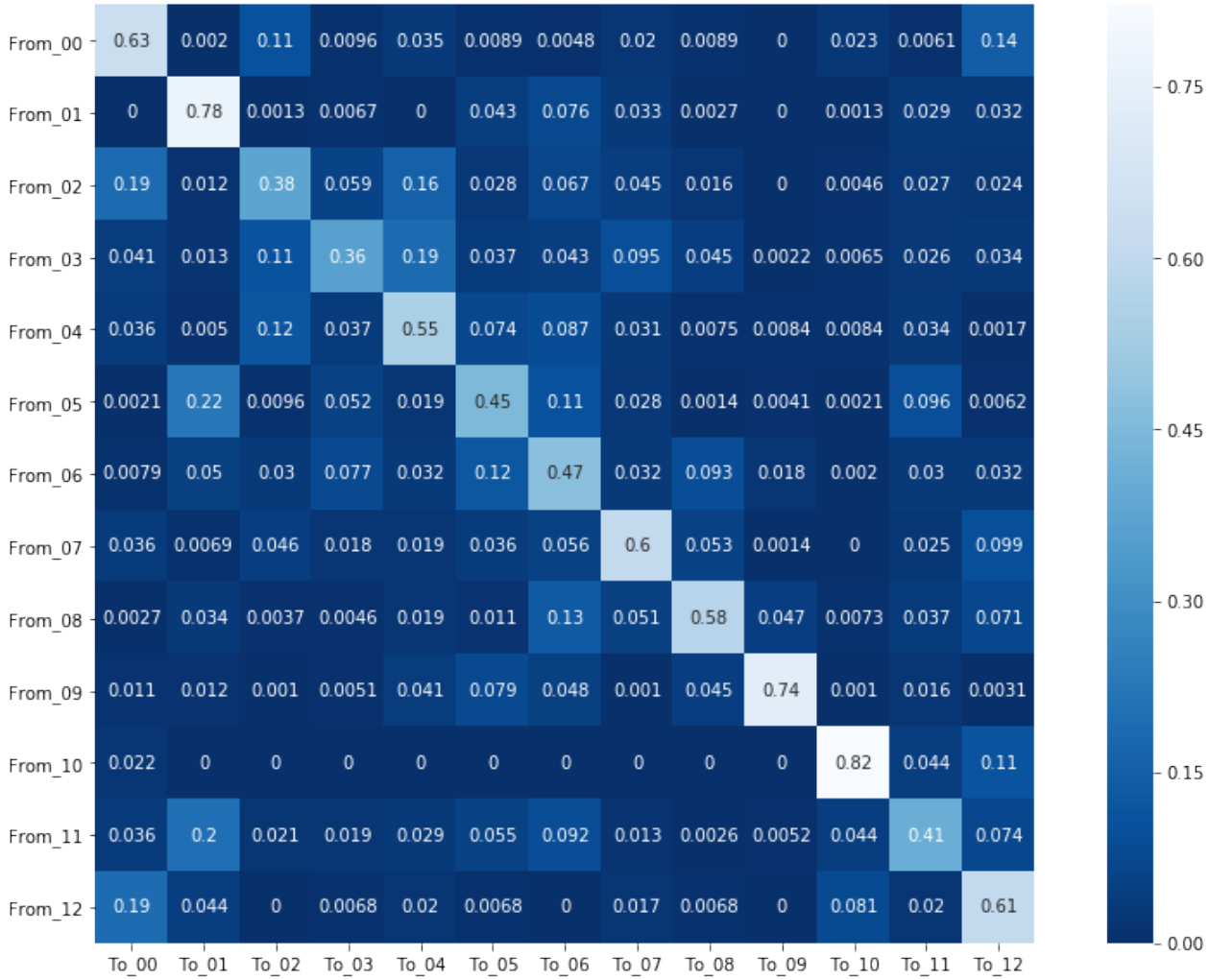


Figure: Confusion matrix $G = 2$.

Performance Evaluation

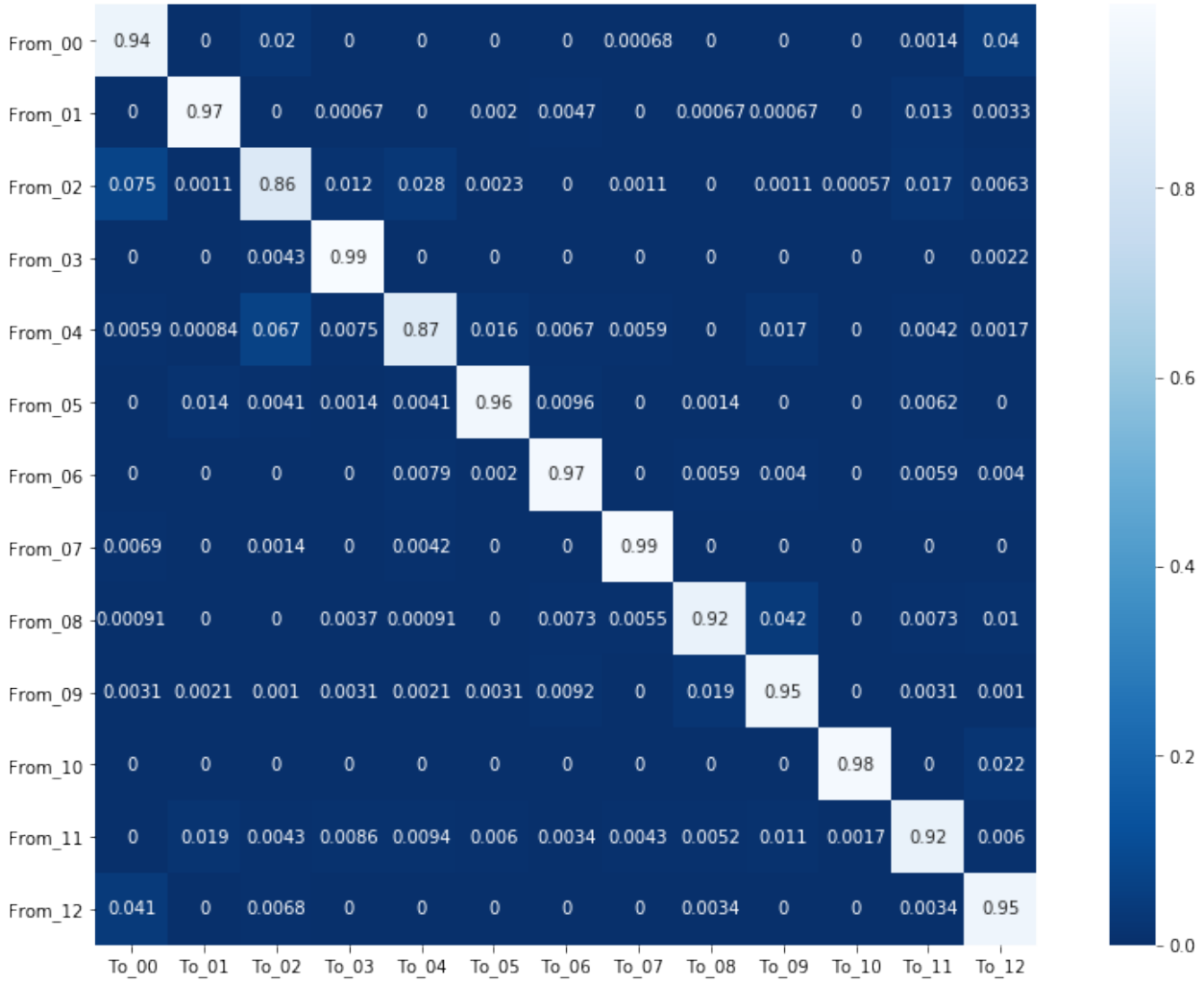


Figure: Confusion matrix $G = 50$.

Performance Evaluation

- For visualization of multidimensional embedding points, we employed T-distributed Stochastic Neighbor Embedding (t-SNE) [13] that can dimensional reduction for multidimensional data.
- t-SNE is a nonlinear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions.

Performance Evaluation

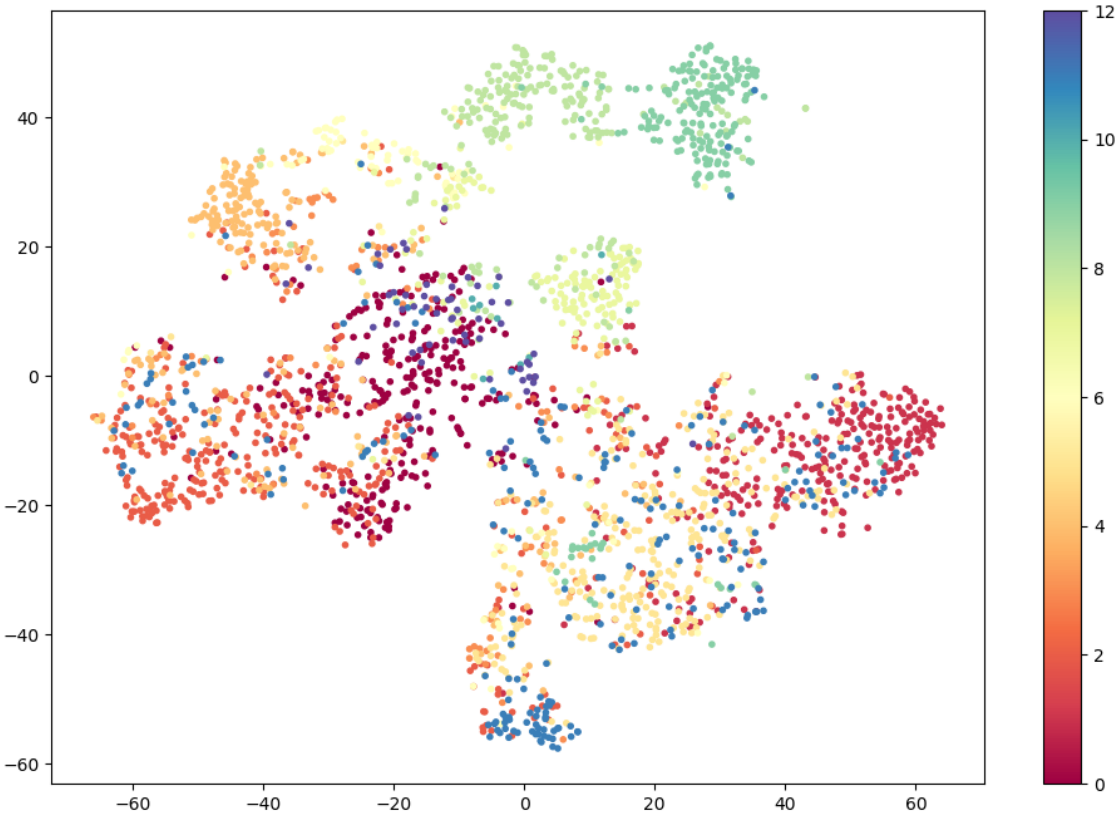


Figure: Visualization by t-SNE with $G = 2$

Performance Evaluation

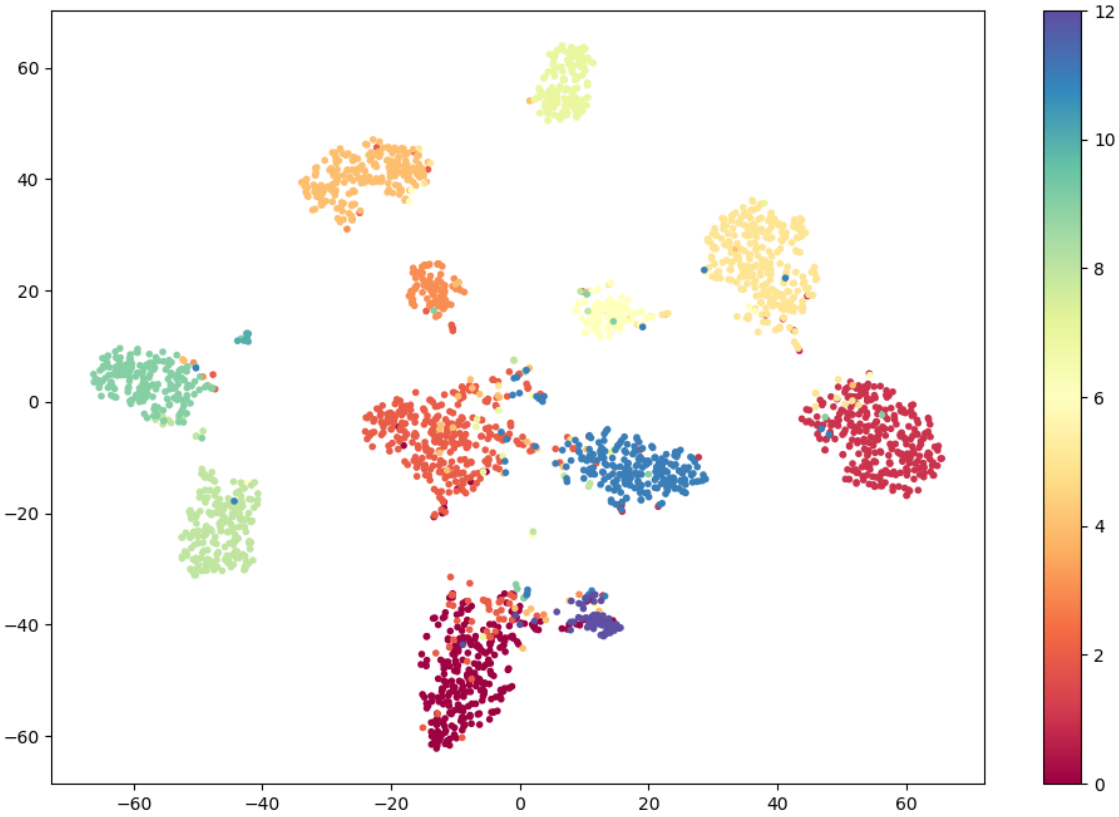


Figure: Visualization by t-SNE with $G = 50$

Summary I

- We investigate machine learning algorithms to analyze eNodeB performance metric for service level assurance in LTE networks.
- To minimize the human labor intensive and time-consuming dataset annotation task, it is thus required to find a data efficient learning algorithm/technique to build a classifier model.
- We proposed a method that uses Prototypical Network as few-shot learning.
- We demonstrate that the proposed few-shot learning methods yield better performance than the existing methods.
- In particular the proposed few-shot learning method that uses 1D-CNN outperforms all other methods under all conditions evaluated in the experiments.

Acknowledgment

- This work is partially supported by Bosco Technologies inc. I thank Tsunemasa Hayashi and Tatsuya Morita for their support and technical discussion.
- This work is partially supported by the Grant-in-Aid for Scientific Research(Grant No. J17H07156) from the Japan Society for the Promotion of Science.
- This work is partially supported by Ericsson Japan K.K. I thank Chin Lam Eng, Sebastian Backstad, Drazen Jarnjak, and Masanobu Fujioka for their support and technical discussion.

References I

[1] Rakesh Agrawal and Ramakrishnan Srikant.
Mining sequential patterns.
In *Proceedings of the Eleventh International Conference on Data Engineering*, ICDE '95, pages 3–14, Washington, DC, USA, 1995. IEEE Computer Society.

[2] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu.
Sequential pattern mining using a bitmap representation.
In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 429–435, New York, NY, USA, 2002. ACM.

References II

[3] S. Enami and K. Shiomoto.
Spatio-temporal human mobility prediction based on trajectory data mining for resource management in mobile communication networks.
In *2019 IEEE 20th International Conference on High Performance Switching and Routing (HPSR)*, pages 1–6, May 2019.

[4] K. Gouda, M. Hassaan, and M. J. Zaki.
Prism: A primal-encoding approach for frequent sequence mining.
In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 487–492, Oct 2007.

[5] Bilal Hussain.
Deep learning-based big data-assisted anomaly detection in cellular networks.
12 2018.

References III

[6] Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda.
Learning systems of concepts with an infinite relational model.
In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 381–388. AAAI Press, 2006.

[7] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, and Scott Shenker.
Onix: A distributed control platform for large-scale production networks.
In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pages 351–364, Berkeley, CA, USA, 2010. USENIX Association.

References IV

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton.
Imagenet classification with deep convolutional neural networks.
In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[9] Jian Pei, Jiawei Han, B. Mortazavi-Asl, H. Pinto, Qiming Chen, U. Dayal, and Mei-Chun Hsu.
Prefixspan,: mining sequential patterns efficiently by prefix-projected pattern growth.
In *Proceedings 17th International Conference on Data Engineering*, pages 215–224, April 2001.

References V

[10] Jake Snell, Kevin Swersky, and Richard Zemel.
Prototypical networks for few-shot learning.
In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4077–4087. Curran Associates, Inc., 2017.

[11] Amin Tootoonchian and Yashar Ganjali.
Hyperflow: A distributed control plane for openflow.
In *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, INM/WREN'10, pages 3–3, Berkeley, CA, USA, 2010. USENIX Association.

References VI

[12] Y. Tsukuda, M. Kosugi, K. Shiomoto, T. Morita, and T. Hayashi.
Reducing inconsistency between software-defined networking controllers.
In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 301–305, June 2019.

[13] Laurens van der Maaten and Geoffrey Hinton.
Visualizing data using t-sne.
Journal of Machine Learning Research, 9:2579–2605, 2008.

[14] J. Wang and J. Han.
Bide: efficient mining of frequent closed sequences.
In *Proceedings. 20th International Conference on Data Engineering*, pages 79–90, April 2004.

References VII

[15] Zhenglu Yang and M. Kitsuregawa.
Lapin-spam: An improved algorithm for mining sequential pattern.
In *21st International Conference on Data Engineering Workshops (ICDEW'05)*, pages 1222–1222, April 2005.

[16] Y. Zheng, L. Wang, R. Zhang, X. Xie, and W. Ma.
Geolife: Managing and understanding your past life over maps.
In *The Ninth International Conference on Mobile Data Management (mdm 2008)*, pages 211–212, April 2008.

[17] Yu Zheng, Hao Fu, Xing Xie, Wei-Ying Ma, and Quannan Li.
Geolife GPS trajectory dataset - User Guide, July 2011.

References VIII

[18] 青木 章悟, 塩本 公平, エン チンラム, and バックスタッド セバスチャン.
Prototypical network を用いた few-shot 学習による lte 網のエリア毎のサービスレベルの把握.
In **信学技報**, volume 119, pages 177–182, July 2019.
NS2019-78.