

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

Shogo Aoki*

* Waseda University

Tokyo, Japan

shogo.aoki@fuji.waseda.jp

Kohei Shiimoto†

† Tokyo City University

Tokyo, Japan

shiimoto@tcu.ac.jp

Chin Lam Eng‡ Sebastian Backstad‡

‡ Ericsson Japan

Kanagawa, Japan

{chin.lam.eng,sebastian.backstad}@ericsson.com

Abstract—With the increasing network topology complexity and continuous evolution of the new wireless technology, it is increasingly challenging to address the network service outage with traditional methods. In order to improve the network operation efficiency, a suitable machine learning technique is used to learn and classify individual base station into different issue based on multiple performance metrics during a specific time window. However an issue classification with supervised learning requires a large amount of labeled dataset, which takes costly human-labor and time to annotate data. To mitigate the cost and time issues, we propose a method based on few-shot learning that uses Prototypical Networks algorithm to complement the eNodeB states analysis. Using dataset from a live LTE network consists of thousand of eNodeB, our experiment results show that the proposed technique provides high performance while using a low number of labeled data.

Index Terms—Cellular Network, eNodeB, KPI, Few-shot Classification

I. INTRODUCTION

In the long-term evolution (LTE) networks, the radio communication services to the mobile users are provided by base stations called eNodeBs. A large number of eNodeBs are deployed to cover the entire service areas spanning various kinds of geographical regions. Radio channel condition, user mobility and traffic load condition varies from area to area. As such, 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 therefore crucial to understand the state of the eNodeB that covers the area.

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). Hundreds of thousands of eNodeBs are typically deployed to cover a nation-wide service area. 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.

Machine learning and deep learning has been used extensively in computer science for voice, face and image recognition. Recently machine learning and deep learning has been applied various network management tasks [11]. The natural question is how we can apply machine learning and deep learning to the task of analyzing eNodeB log-data to assure the service level offered to the mobile customer.

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 attach the label to the data, and (3) the supervised learning phase where the classifier is trained using a set of labeled data. In order to build a good supervised machine learning model, we need a large number of training data. Labeling data is a costly task. Human operator examines data, classifies them, and annotate them with an appropriate label.

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. We should also note that anomalies are difficult to occur in practice, so the anomaly classes are usually sparse in the dataset. As such it is extremely important for the operators to deal with an unbalanced data set where a few class has only handful data instances while others have a lot of data instances.

Current research areas such as few-shot learning show positive results in reducing the amount of annotated data required during training while maintaining the prediction accuracy [3], [4], [6], [8]–[10], [12]. Few-shot learning aims to learn the model using a few samples per each class. 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 in this paper. Taking advantage of few-shot learning algorithm, a classifier model can be trained with

minimum number of labeled data.

In this paper, we investigate the few-shot learning that uses Prototypical Network [10] applied to the task of eNodeB performance metric analysis. By using real trace data from eNodeBs deployed in production networks, we evaluate the proposed few-shot learning method. We confirm that the proposed few-shot learning methods yield better performance than the existing methods that use neural networks including deep learning such as convolutional neural networks (CNN). We also investigate the semi-supervised Prototypical Network to further reduce the number of labeled data and demonstrate that the semi-supervised Prototypical Network yields comparable performance with the supervised Prototypical Network. The remainder of the paper is structured as follows. Section II introduces few-shot learning and Prototypical Network. Section III presents the experiments and results. Section IV concludes the paper.

II. FEW-SHOT CLASSIFICATION

A. Prototypical Network

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.

Prototypical Network [10] is a few-shot learning that use neural networks (NNs). Prototypical networks is based on the idea that there exists an embedding structure in which points cluster around a single prototype representation for each class. It constructs non-linear mapping of the input into an embedding space using a neural network and takes a class's prototype to be the mean of the labeled set of examples (the *support* set) in the embedding space. It uses the training set to extract a prototype vector from each class as shown in Fig. 1, and then classifies the inputs in the training set based on their distance to the prototype of each class as shown in Fig. 2. Prototypical network learns an embedding function $h_\phi(x)$, parameterized ϕ in a NN, that maps examples into a space where examples from the same class are close and those from different classes are far. Classification is then performed for the embedded unlabeled points (the *query* set) by simply finding the nearest class prototype.

Prototypical Network uses an episodic training [7] to split data into training set $\mathcal{S} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_{A \times N}, y_{A \times N})\}$ and test set $\mathcal{Q} = \{(\mathbf{x}_1^*, y_1^*), \dots, (\mathbf{x}_{T \times N}^*, y_{T \times N}^*)\}$, where A denotes the number of training set observations of each class, T denotes the number of test set observations of each class, and N denotes the number of classes. They are used for computing prototype vectors and losses as explained below.

To compute prototype $\mathbf{p}_c$ of each class c , a per-class average of the embedded examples is performed:

$$\mathbf{p}_c = \frac{\sum_i h_\phi(\mathbf{x}_i) z_{i,c}}{\sum_i z_{i,c}}, \quad (1)$$

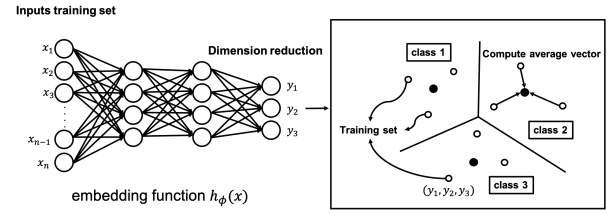


Fig. 1. Left: Embedding function. Right: Prototype vector calculation using training set.

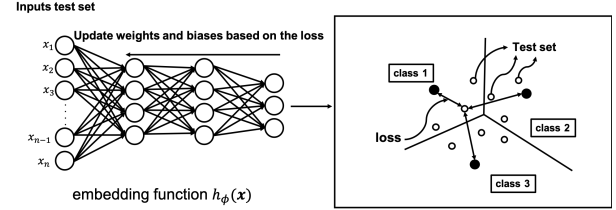


Fig. 2. Left: Embedding function (same as Fig. 1). Right: Loss calculation according to distance using test set.

where z_i is one-hot vector. These prototypes are used as a predictor for the class of any test set example $\mathbf{x}^*$, which assigns a probability over any class c through softmax function based on the distances between $\mathbf{x}^*$ and each prototype, as follows:

$$p(c|\mathbf{x}^*, \{\mathbf{p}_c\}) = \frac{\exp(-\|h_\phi(\mathbf{x}^*) - \mathbf{p}_c\|_2^2)}{\sum_{c'} \exp(-\|h_\phi(\mathbf{x}^*) - \mathbf{p}_{c'}\|_2^2)} \quad (2)$$

The loss function used to update Prototypical Network is then simply the average negative log-probability of the correct class assignments, for all test set examples:

$$-\frac{1}{T} \sum_i \log p(y_i^* | \mathbf{x}_i^*, \{\mathbf{p}_c\}) \quad (3)$$

Training proceeds by minimizing the average loss, iterating over training episodes and performing a gradient descent update for each.

B. Embedding Function

Embedding function maps discrete object eNodeB performance metrics to a vector of real numbers so that similar objects are close. This design is extremely important. 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) [2], two-dimensional convolutional neural networks (2D-CNN) [5] and one-dimensional convolutional neural networks (1D-CNN).

III. EXPERIMENTS

A. Dataset

By using real trace data from eNodeBs deployed in production networks (we are not able to disclose the details of dataset due to confidentiality), we evaluate the proposed few-shot learning method. Each eNodeB reports 33 KPIs including the number of users, connection attempts, data/signaling

packet rate, resource occupancy and radio/channel quality and normalized zero to one for each. Each eNodeB generates 33 KPIs every hour, a record of each eNodeB per day consists of 792 KPIs. 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 I. The list of the eNodeB status includes normal operating condition (label_05), traffic related conditions (labels_00,02,03,04), radio related conditions (labels_01,06,07,08,09), service, system, control related conditions (label_10,11,12). The number of labeled data for each eNodeB status is shown in Table 3. We notice that label_10 (System hardware processor load) has only 19 instances whereas other labels have from 119 to 696 instances. Normal operating condition (label_05) has 583 instances whereas high transmission of uplink and downlink data (label_02), weak radio wave coverage (label_01), high traffic condition (label_00), high transmission of uplink data (label_04), and service setup issue (label_11) are the top-5 anomaly instances in terms of the number of instances in the dataset.

TABLE I
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

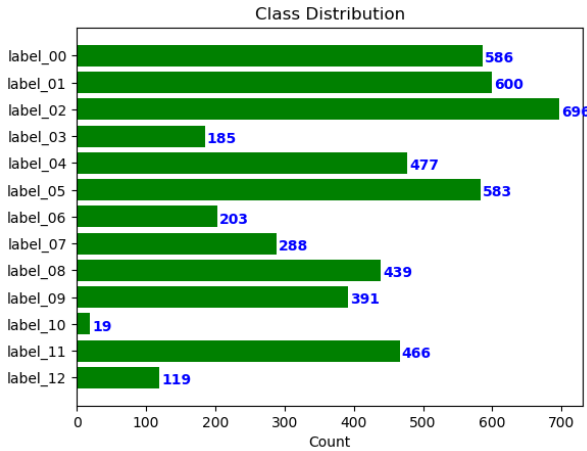


Fig. 3. The number of labeled data for each eNodeB status.

B. Prototypical Network

The aim of the experiment is to investigate the minimum number of labeled data required to build a classifier for eNodeB performance metric analysis. Let G denote the number of labeled data used in each class G . We investigate the impact of the number of labeled data G on the performance of Prototypical Network. We use $G = 2, 5, 10, 15, 25, 50, 70, 100, 150, 200, 250$. As previously mentioned, Prototypical Network uses an episodic training [7] that selects A samples in training set $\mathcal{S}$ and T samples in test set $\mathcal{Q}$ for each class from the training dataset. We use $(A, T) = (3, 3)$ in the following evaluation. Prototypical Network iterates m episodes by randomly picking those samples to yield the result.

Prototypical Network uses embedding function to map discrete object to a vector of real numbers so that similar objects are close. This design is extremely important. We investigate which neural network is suitable for embedding function in Prototypical Network. We developed a Python program that implements the Prototype Network using Keras/Tensorflow [1]. We evaluate the performance of few-shot learning methods that use Prototype Network that use MLP, 2D-CNN, and 1D-CNN as embedding function. We use $m = 20,000$ for MLP and 1D-CNN, and $m = 3,500$ for 2D-CNN in the following evaluation. We compute average of results of five experiments. We also evaluate the performance of existing classifiers that use MLP, 2D-CNN, and 1D-CNN as baseline methods.

F1-Scores that both few-shot learning methods and baseline methods yield are depicted in Fig. 4. Compared with the baseline methods, the proposed few-shot learning methods yield better performance when the number of labeled data for each class G is less than 50. The proposed few-shot learning method that uses 1D-CNN outperforms all other methods under all conditions evaluated in the experiments. The reason why the Prototypical Network outperforms the baseline method is explained as follows. When the number of labeled data is small, the baseline method that computes probabilities associated with individual classes poorly classifies the data whereas the Prototypical Network that computes distances to the prototype of individual classes correctly classifies the data.

We observe that the performance of the baseline method that uses MLP quickly degrades as the number of labeled data decreases. Even though the baseline methods that use 2D-CNN and 1D-CNN outperform the baseline method that uses MLP, their performance degrade as the number of labeled data decrease when the number of labeled data for each class G is less than 50.

We notice that the baseline method that uses 1D-CNN outperforms the baseline method that uses 2D-CNN. 2D CNN captures spatial characteristics of feature vectors in two dimension while 1D CNN does in one dimension. In our situation, 2D CNN tries to capture correlation in both vertical axis (i.e., correlation between different KPIs) and horizontal axis (i.e., time correlation in time) while 1D CNN does correlation in

only horizontal axis (time correlation in time for each KPI).

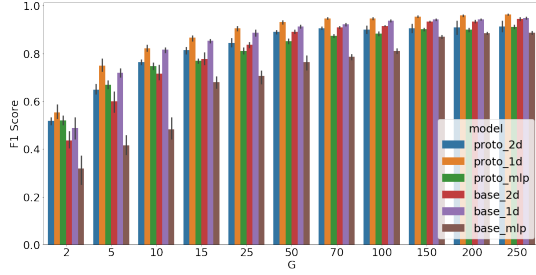


Fig. 4. Comparison between Prototypical Network and baseline.

Figs. 5 and 6 show the confusion matrix with $G = 2$ and $G = 50$. When the number of labeled data is small ($G = 2$), the label_00 and the label_02 are confused, the label_00 and the label_12 are also confused. When the number of labeled data is large ($G = 50$), each label is accurately classified. As the number of labeled data increases, the boundaries between classes become more conspicuous even with the class with small number of labeled data such as label_10 that has only 19 instances.



Fig. 5. Confusion matrix $G = 2$.

IV. CLOSING REMARKS

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, we proposed a few-shot learning method that uses Prototypical Network. We demonstrate that the proposed few-shot learning method yields 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. Extensive exploration of parameters is a direction for our future works. Another direction of our future work includes how to select data to be labeled to further reduce the required number of labeled data per class.

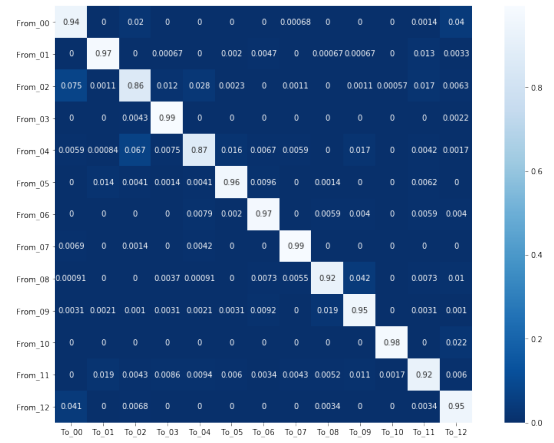


Fig. 6. Confusion matrix $G = 50$.

REFERENCES

- [1] François Chollet et al. Keras. <https://keras.io>, 2015.
- [2] B. Hussain, Q. Du, and P. Ren. Deep learning-based big data-assisted anomaly detection in cellular networks. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2018.
- [3] Lukasz Kaiser, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. *CoRR*, abs/1706.05137, 2017.
- [4] Diederik P. Kingma, Danilo Jimenez Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. *CoRR*, abs/1406.5298, 2014.
- [5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS'12*, pages 1097–1105, USA, 2012. Curran Associates Inc.
- [6] Avital Oliver, Augustus Odena, Colin Raffel, Ekin D. Cubuk, and Ian J. Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. *CoRR*, abs/1804.09170, 2018.
- [7] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2017.
- [8] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In *Proceedings of 6th International Conference on Learning Representations ICLR*, 2018.
- [9] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *CoRR*, abs/1606.04586, 2016.
- [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] Ion Stoica, Dawn Song, Raluca Ada Popa, David A. Patterson, Michael W. Mahoney, Randy H. Katz, Anthony D. Joseph, Michael Jordan, Joseph M. Hellerstein, Joseph Gonzalez, Ken Goldberg, Ali Ghodsi, David E. Culler, and Pieter Abbeel. A berkeley view of systems challenges for ai. Technical Report UCB/EECS-2017-159, EECS Department, University of California, Berkeley, Oct 2017.
- [12] Oriol Vinyals, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *CoRR*, abs/1606.04080, 2016.