

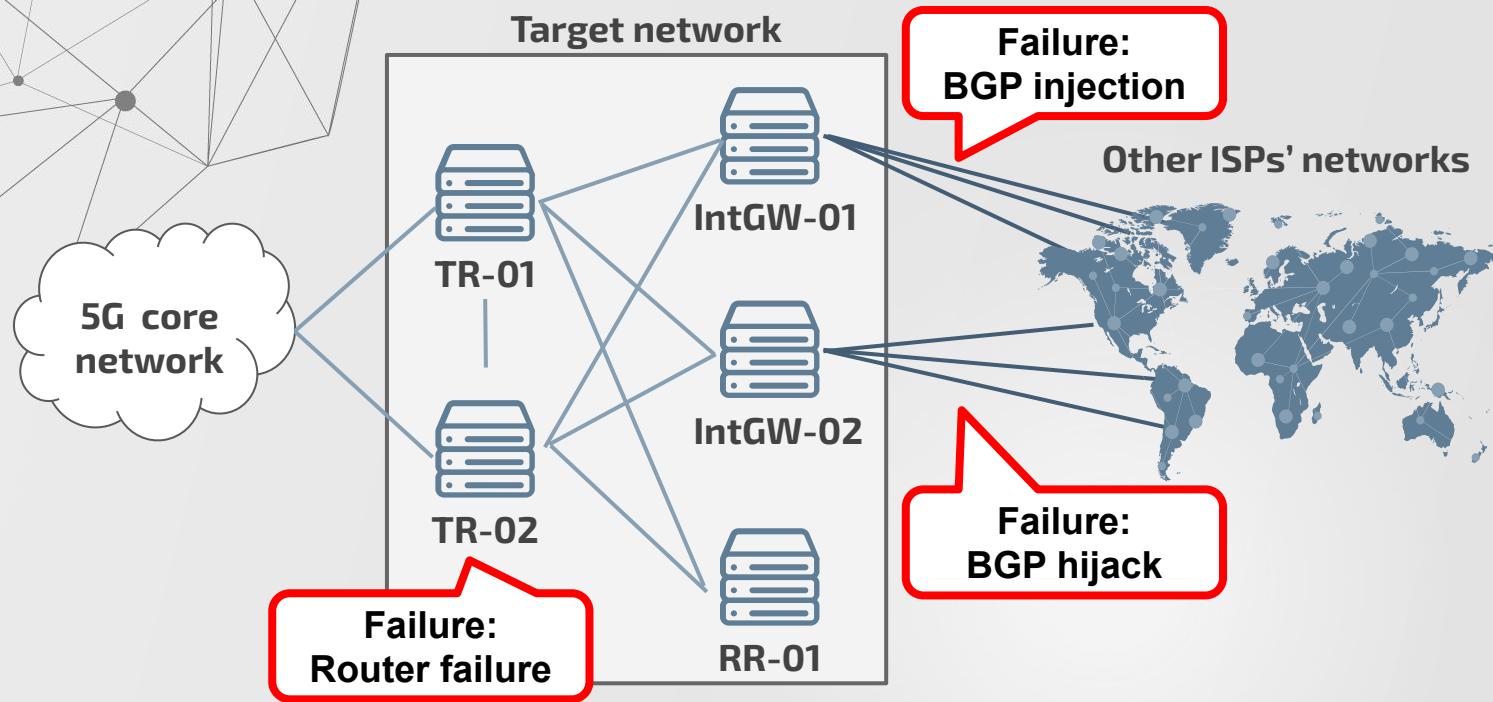
The background features a network diagram with nodes and connections, overlaid on a light gray background with faint geometric shapes and a starburst pattern of dots in the upper right.

# **Pre-training and fine-tuning approach for detecting route information failures in IP core networks**

---

team mlab 2020/12/15

# Background – Difficulty of failure detection in BGP network



**Difficulty**

Conventional rule-based approach is NOT applicable because BGP does not have a mechanism to authenticate each router configuration

# Problem Description – Failure Detection in BGP Network

- **Objective:** Detect the failures in the BGP network from the network status information
- **Input:** Time series data obtained from the border gateway routers

Time series input data

TIME	feature1 (Rx packet)	feature2 (Tx @scket)	feature3 (CPU usage)
2020/07/07 10:21	0.87	0.32	0.23
2020/07/07 10:22	0.67	0.23	0.45
2020/07/07 10:23	0.67	0.49	0.78
	⋮		

Time series prediction result

TIME	predicted isFailure
2020/07/07 10:21	false
2020/07/07 10:22	true
2020/07/07 10:23	true
	⋮

Time series labeled  
training data

TIME	truth isFailure
2020/07/07 10:21	false
2020/07/07 10:22	true
2020/07/07 10:23	false
	⋮

predict

predict

predict

compare

compare

compare



# Related Work – Fault classification using machine learning in an NFV environment [Kawasaki+20]

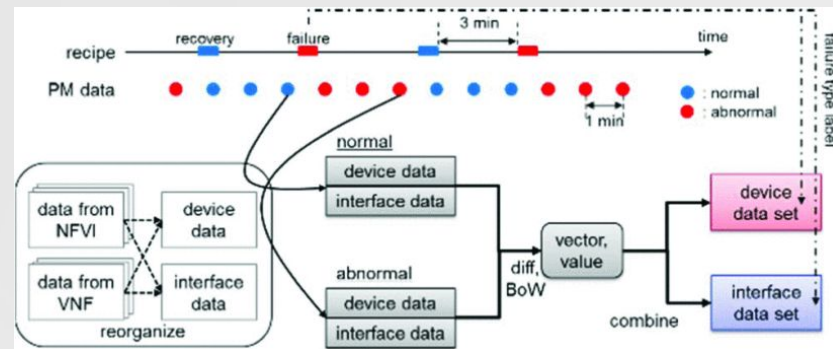
## Failure Scenario:

Node-down, Interface-down, and CPU overload

## Preprocessor:

Bag-of-Words (Bow) using labeled training data

**Fault Classifier:** MLP, RF, and SVM



Pre-processing Steps

- **Failure Scenario does NOT include failures peculiar to BGP network** (injection, hijack, etc...)
- **Only labeled data (normal/abnormal) are used as the training data**

## Problem

In the case of failure detection in the BGP network,

- Overfitting to one domain caused by lack of context should occur
- Additional dataset is required to apply the model to other networks

# 01

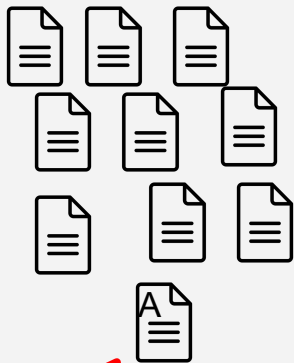
## Methodology

---



# Pre-training and fine-tuning approach

## BGP Path Logs

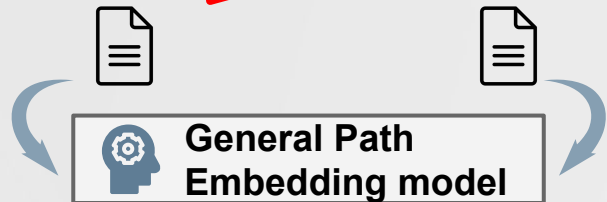


The size of labeled data is small.

## Pre-training phase

We pre-train a general BGP path embedding model to understand BGP context.

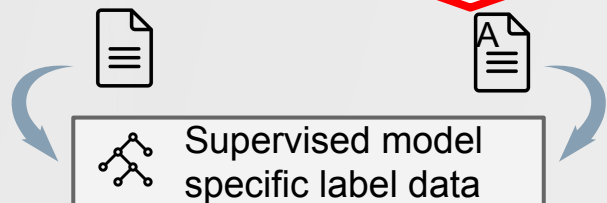
## Unsupervised learning



## Fine-tuning phase

We train a model to predict labels on specific task such as BGP network failure detection.

## Supervised learning



## Advantage

Since the model learning to understand the context does not require specific label data in pre-training, a large amount of BGP path data generated by real network is available.

# BGP path embedding

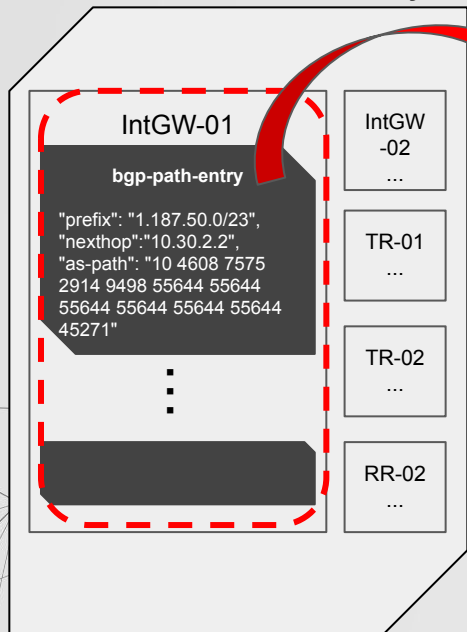
Unsupervised learning

BGP path has an ordered structure.

We address the ordered structure as a language model.

We developed a general path model which transforms objects on BGP to vectors.

ある時刻のnetwork-deviceのjson



IntGW-01のpath-entries document

```
1.0.0.0/2410 4608 13335 10 4608 13335 p10 p30 p2 p2
1.0.4.0/2410 4608 4826 38803 56203 10 4608 4826 38803 56203 p10 p30 p2 p2
1.0.4.0/2210 4608 4826 38803 56203 10 4608 4826 38803 56203 p10 p30 p2 p2
1.0.5.0/2410 4608 4826 38803 56203 10 4608 4826 38803 56203 p10 p30 p2 p2
```

IntGW-02のpath-entries document

...

TR-01のpath-entries document

...

TR-02のpath-entries document

...

RR-01のpath-entries document

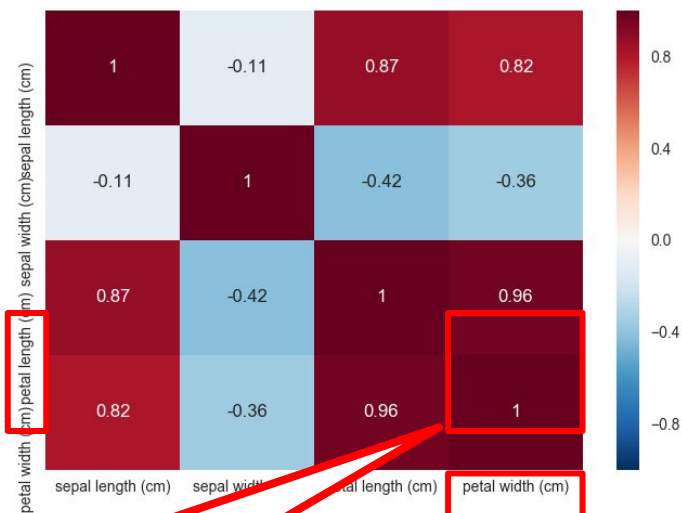
...

	v0	v1	v2
0	4.364887	2.084560	-0.502491
1	4.176826	1.739415	-0.662214
2	4.167618	1.965276	-0.431082
3	4.102726	2.043609	-0.724891
4	4.189735	1.810896	-0.756527
...	...	...	...

# Dimensionality Reduction

## Example Image of Correlation Matrix

<https://blog.amedama.jp/entry/2017/04/18/230431>



**Delete the column  
with correlation coef. == 1**

## To cope with the Curse of Dimensionality

Because there are too many columns in the dataframe we need to delete redundant columns.

ex. 238 columns ( physical infrastructure )

## Method

1. Delete repetitive columns in each row
2. Delete the column with correlation coef. == 1
  - Generated correlation coef. matrix (Left)
  - Pearson product-moment correlation coefficient

$$\rho(a, b) = \frac{\sum_{i=1}^n (X_{a,i} - \bar{X}_a)(Y_{b,i} - \bar{Y}_b)}{\left\{ \sum_{i=1}^n (X_{a,i} - \bar{X}_a)^2 \sum_{j=1}^n (Y_{b,j} - \bar{Y}_b)^2 \right\}^{1/2}}$$





# 02

## Evaluation

---

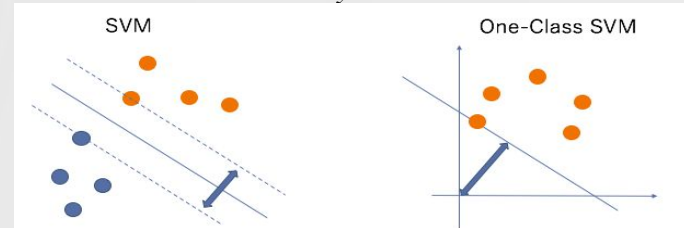
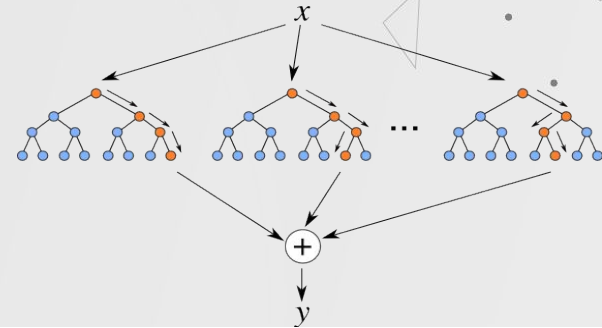
# For the Evaluation

## Failure Prediction Baseline :

- **Failure detection by tree-based classifiers**  
AdaBoost, Bagging, ExtraTrees, GradientBoosting, Random Forest
  - Regression analysis for the failure indicating column of each failure types
  - Less failure examples problem : oversampling by SMOTE
- **One Class SVM** : Unsupervised Learning
  - Learn normal state to detect anomaly status as outlier
  - Useful for unknown datasets
  - Support Vector Machine : Supervised Learning

## Example Image of Random Forest

<http://kazoo04.hatenablog.com/entry/2013/12/04/175402>



## Support Vector Machine and One Class SVM

<https://www.smartbowwow.com/2018/12/anomaly-detection-using-one-class.html>

# Evaluation – Tap Loss & Tap Delay

## Tap Loss

Method	AdaBoost	Bagging	ExtraTrees	Gradient Boosting	Random Forest	OneClass SVM
f1 score(%)	62.90	63.25	63.27	63.25	63.25	18.61

---

## Tap Delay

Method	AdaBoost	Bagging	ExtraTrees	Gradient Boosting	Random Forest	OneClass SVM
f1 score(%)	53.26	54.83	54.84	54.86	54.84	17.38

# Evaluation – Node Down & Interface Down

## Node Down

Method	AdaBoost	Bagging	ExtraTrees	Gradient Boosting	Random Forest	OneClass SVM
f1 score(%)	76.31	71.60	73.62	70.09	73.07	22.93

## Interface Down

Method	AdaBoost	Bagging	ExtraTrees	Gradient Boosting	Random Forest	OneClass SVM
f1 score(%)	60.92	65.11	65.11	65.12	65.11	10.22

# Evaluation – BGP injection & BGP hijack

## BGP injection

Method	AdaBoost	Bagging	ExtraTrees	Gradient Boosting	Random Forest	OneClass SVM
f1 score(%)	74.34	64.22	61.81	63.63	63.70	1.16

## BGP hijack

Method	AdaBoost	Bagging	ExtraTrees	Gradient Boosting	Random Forest	OneClass SVM
f1 score(%)	64.04	54.96	54.56	56.47	56.31	2.07

# 03

## Summary



# Summary

**BGP path data from routers**



**Pre-training: BGP path embedding  
(unsupervised training)**



**Fine-tuning: existing methods with labeled data  
(supervised learning)**



**Evaluation with labeled data**

Anomaly type	Anomaly name	f1 score
Hardware	Tap Loss	63.27
Hardware	Tap Delay	54.86
Software	Node Down	76.31
Software	Interface	65.11
Software	BGP Injection	74.34
Software	BGP Hijack	64.04

# Thanks

- member

- Ryoma Kondo [kondo@mlab.t.u-tokyo.ac.jp](mailto:kondo@mlab.t.u-tokyo.ac.jp)
- Takashi Ubukata [t\\_ubukata@mlab.t.u-tokyo.ac.jp](mailto:t_ubukata@mlab.t.u-tokyo.ac.jp)
- Kentaro Matsuura [matsuura@mlab.t.u-tokyo.ac.jp](mailto:matsuura@mlab.t.u-tokyo.ac.jp)

