JANOG52 Lightning talk

eBPFを用いたIPネットワークにおけるAI/ML異常検知手法

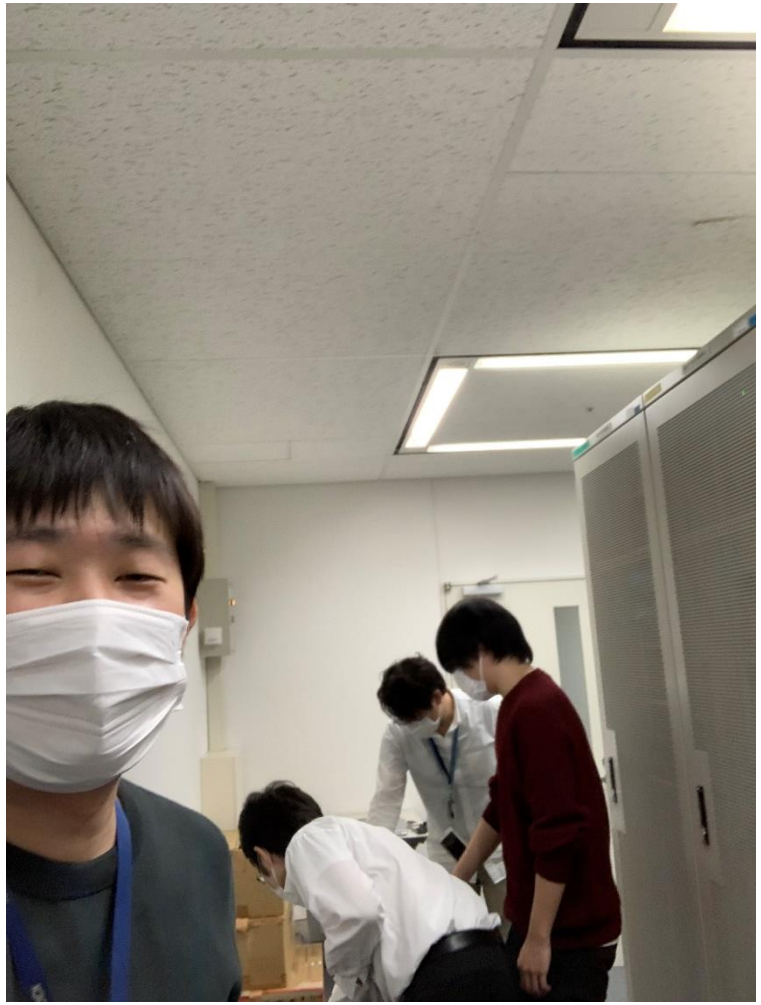
2023/7/5

KDDI総合研究所
桜庭　皆人

発表内容

- **自己紹介**
- **背景**
  - モニタリングにおける課題
- **eBPFを用いたネットワークモニタリングアーキテクチャ**
  - 全体アーキテクチャ
  - SONiC上への実装
- **評価**
  - 評価実験構成およびシナリオ
  - AI/MLを用いた時系列分析による評価実験結果
- **まとめ**

自己紹介



片付けサボって自撮り

- 名前：桜庭　皆人（さくらば　みなと）
- 所属：
  株式会社KDDI総合研究所
  ネットワーク部門　オペレーショングループ
- 経歴
  商用モバイルIP網ルータの開発(2017年～2022年)
  ホワイトボックスルータ（2021年)
  研究所へ出向(2022年～今）
- **JANOG参加歴**
  JANOG#42に参加、登壇は初めて

ネットワークモニタリングにおける課題

ルータにおける正常性を判断する材料として、経路情報はパケットの転送先を決定する情報であり消失や誤りはネットワークの不安定へとつながる。しかしモニタリングにおいて以下の課題が存在する。

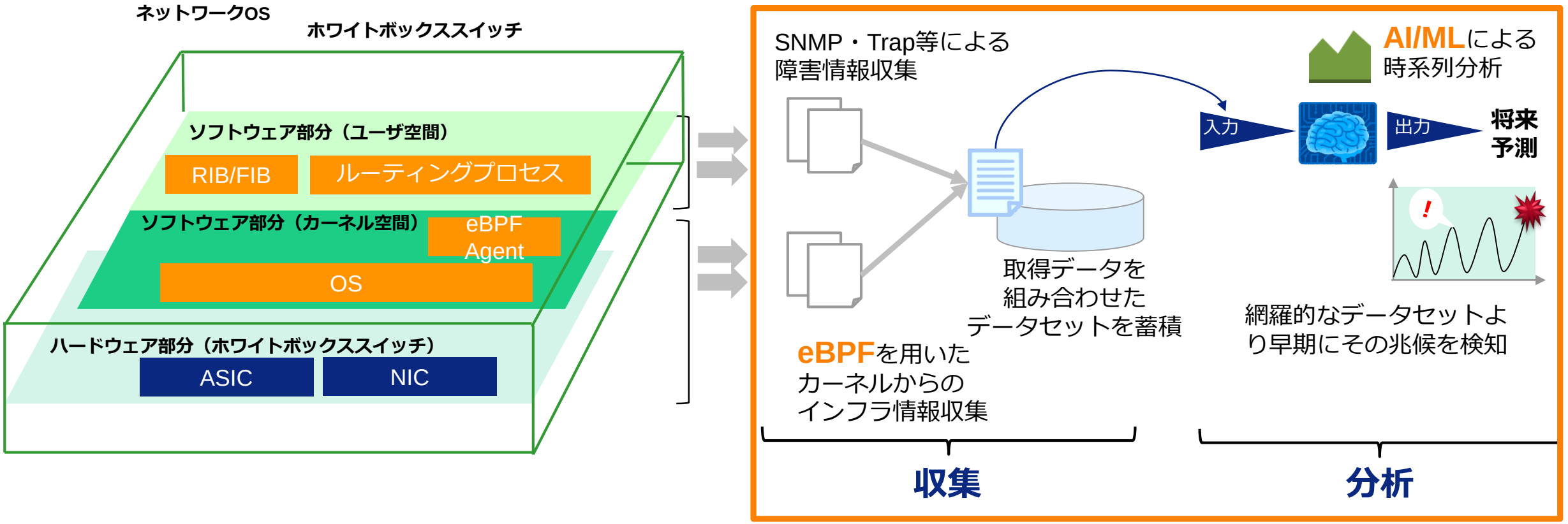
- **ルーティングプロセス**
  - 取得できるデータはルーティングプロセスの計算結果であり、kernelの動作による過程を知ることができない
- **経路情報**
  - フルルートを所持すると90万経路(IPv4)を超えるため、短いサイクルでの経路状態を取得することが困難

➡ **eBPFによる異常検知で上記課題の解決を目指す**

➡ **eBPFをSONiCへ組み込んだプロトタイプを作成し評価しました**

提案する全体アーキテクチャ

ソフトウェアルータからeBPFによって得られる情報を含むデータセットを、AI/MLが学習することによって障害検知・障害予測を行う

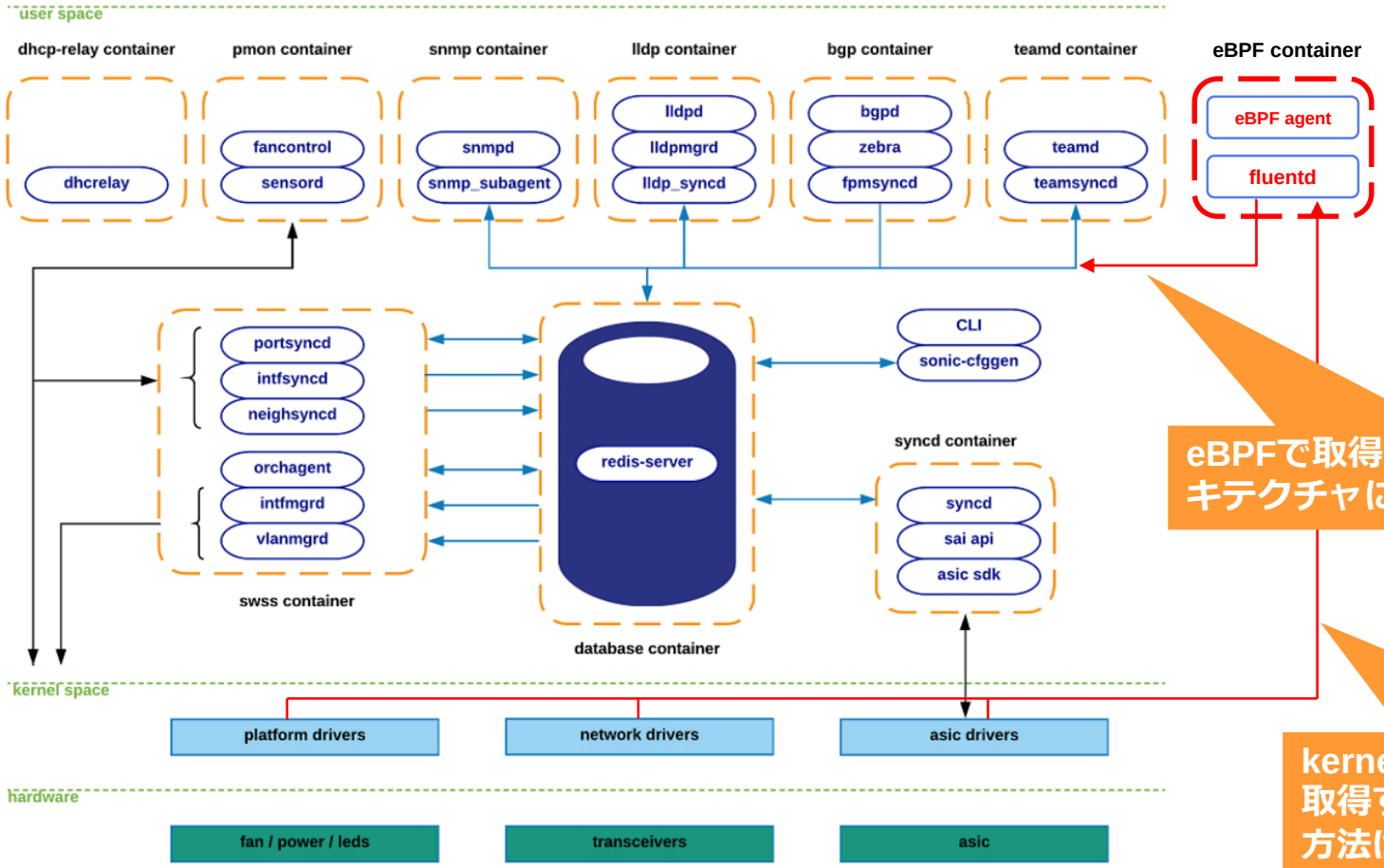


(https://www.irasutoya.com/)

ソフトウェアルータ（SONiC)上のアーキテクチャ

SONiC上に実装したeBPFネットワークモニタリングシステム

- docker containerとしてeBPF取得ツールを導入
- 取得したeBPFデータはDB(redis)に格納し、他プロセスから参照可能



(https://sonic-net.github.io/SONiC/)

- **eBPF-agent:**
  - eBPFデータの取得およびfluentdへの送信を行う
- **fluentd**
  - eBPF-agentから受け取ったデータをredisDBへ投入する

eBPFで取得したデータをSONiCアーキテクチャに合わせてDBへ格納

kernel空間からeBPFデータを取得
取得するデータ形式や、平均値などの加工方法は起動時に指定する

SONiCへのeBPFの組み込み

eBPF環境構築はbpfcc-toolsの導入手順をベースに実施

■ 苦労した点

- **SONiCのkernel headerが取得できない**
  - SONiC　：linux-headers-5.10.0-12.2-amd64
  - 入手可能な「linux-image-5.10.0-12-amd64」をインストールし、名前をSONiC側に合わせた

- **kernel versionによってアタッチできない関数がある(tcpdropなど)**
  - 1つ目の点の関係で、「動かない⇒kernel versionを落とす」の対処が困難
  - tcpdropは以前はtcpdrop()にアタッチされていたが、5.10.0-12ではkfree_skb()を参照する必要あり

- **L2～L3レイヤをターゲットにしたeBPFプログラムのサンプルが少ない**
  - /sys/kernel/debug/tracing/eventsを見てトレースポイントにあたりをつけて、kernel header ソースコードの宣言箇所を読みながら試行錯誤

```
経路追加だと…
static int inet_rtm_newroute(struct sk_buff *skb, struct nlmsghdr *nlh,struct netlink_ext_ack *extack)
```

※bpfcc-tools: https://github.com/iovisor/bcc

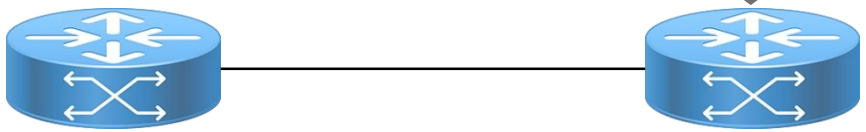
シナリオ1:作業中のオペレーションミス検知(異常判定)

KDDI Research

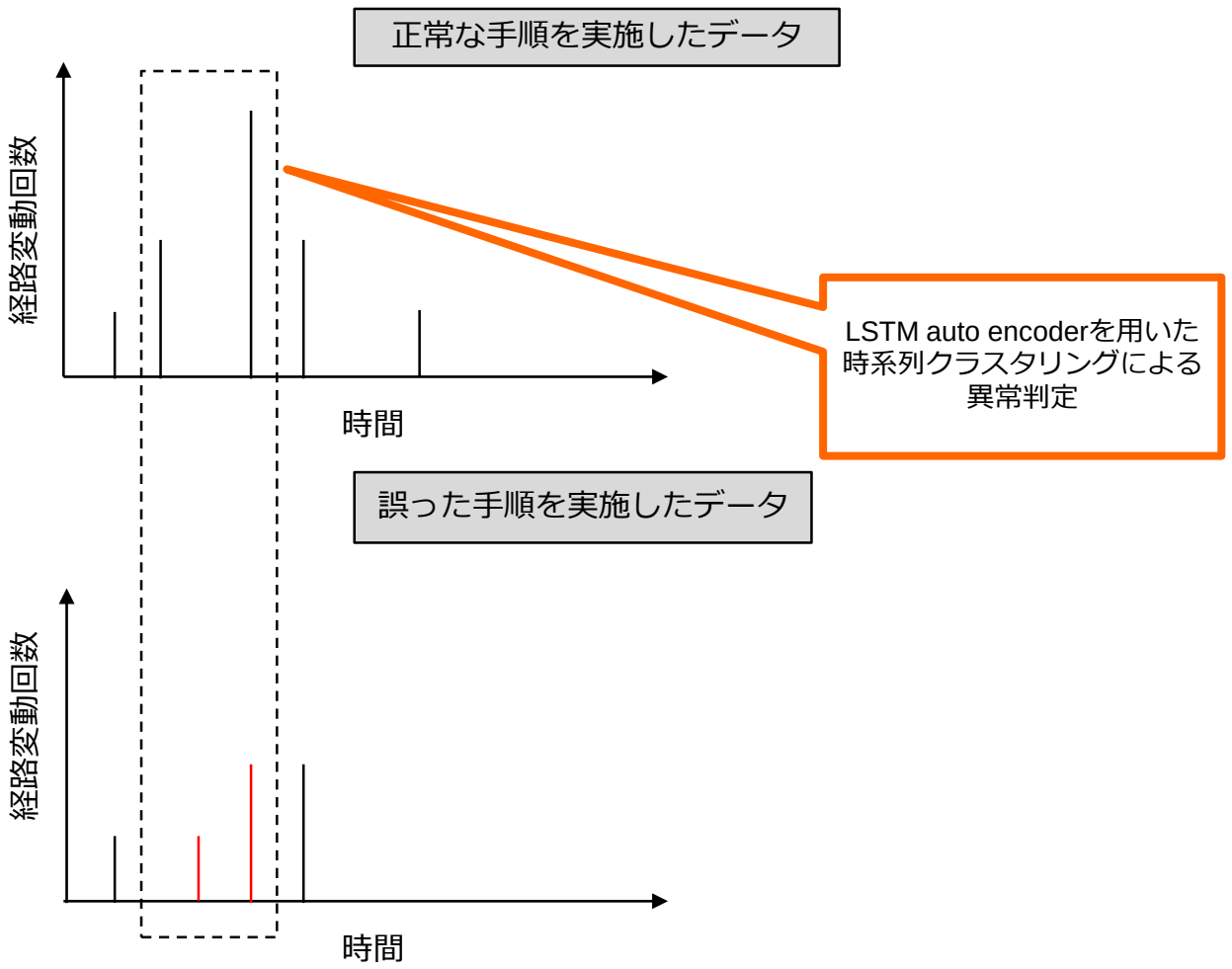
オペレーション手順の誤り

オペレーションミス検知

(https://www.irasutoya.com/)



Time	経路情報	Next-hop	Status	eBPF metric
2022-12-08 00:00:00	10.0.0.0/24	10.0.1.254	Add	{“0:00:00”,”ADD”}
2022-12-08 00:01:00	172.16.0.0/24	172.16.1.254	Delete	{“0:01:00”,”DEL”}
2022-12-08 00:02:00	192.168.0.0/24	192.168.1.254	Delete	{“0:02:00”,”DEL”}
2022-12-08 00:02:01	192.168.0.0/24	192.168.1.253	Add	{“0:02:01”,”ADD”}
2022-12-08 00:03:00	100.0.0.0/24	100.0.1.254	Add	{“0:03:00”,”ADD”}
2022-12-08 00:03:01	100.0.0.0/25	100.0.1.253	Add	{“0:03:01”,”ADD”}

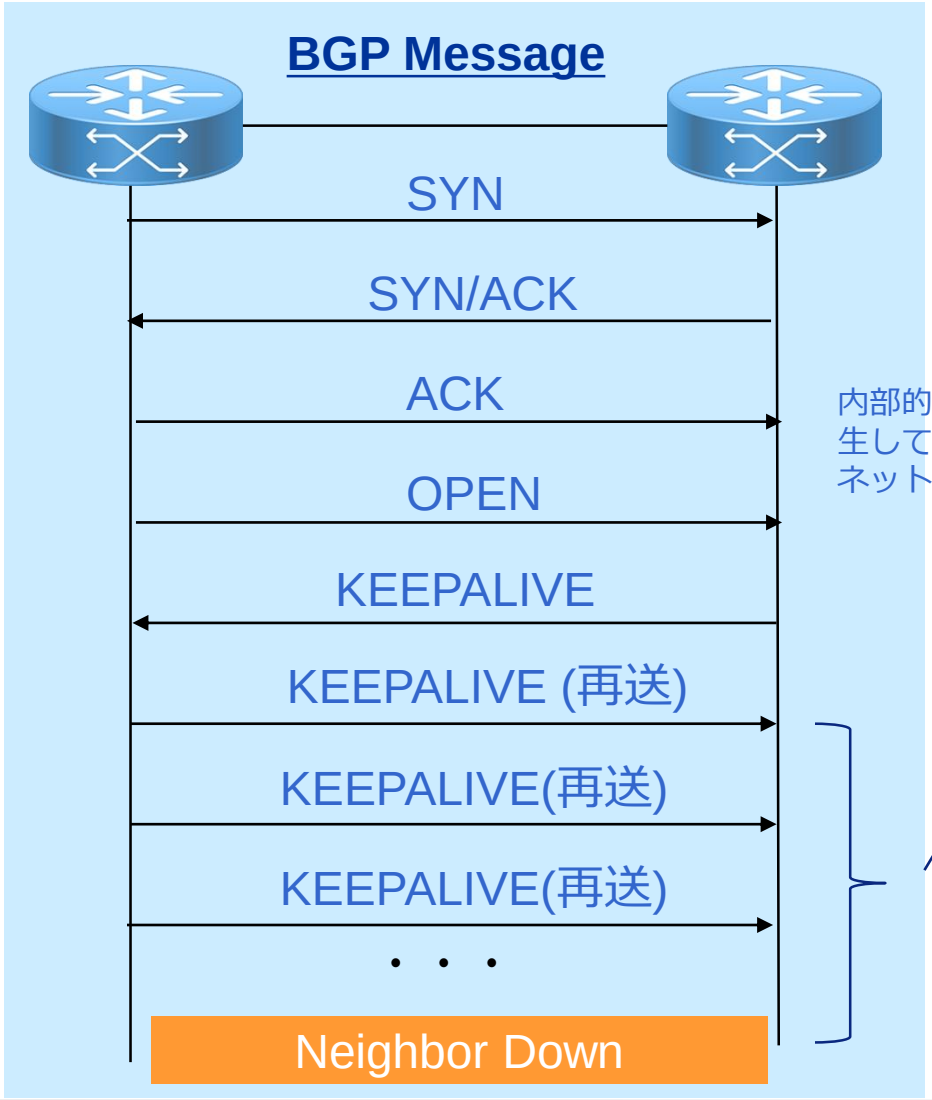


(https://www.irasutoya.com/)

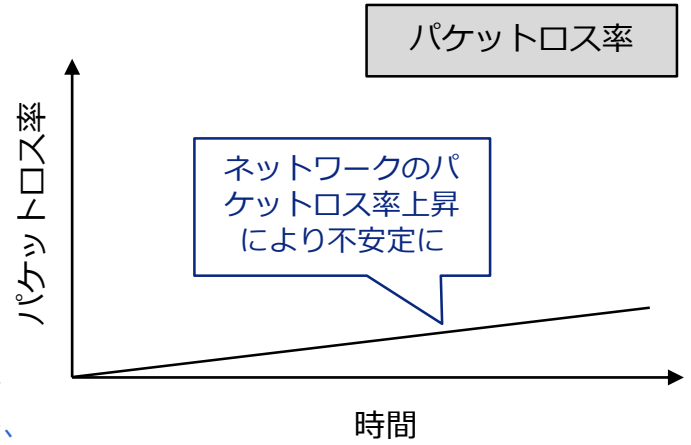
シナリオ2:パケロス率上昇によるBGPセッション不安定(異常検知)

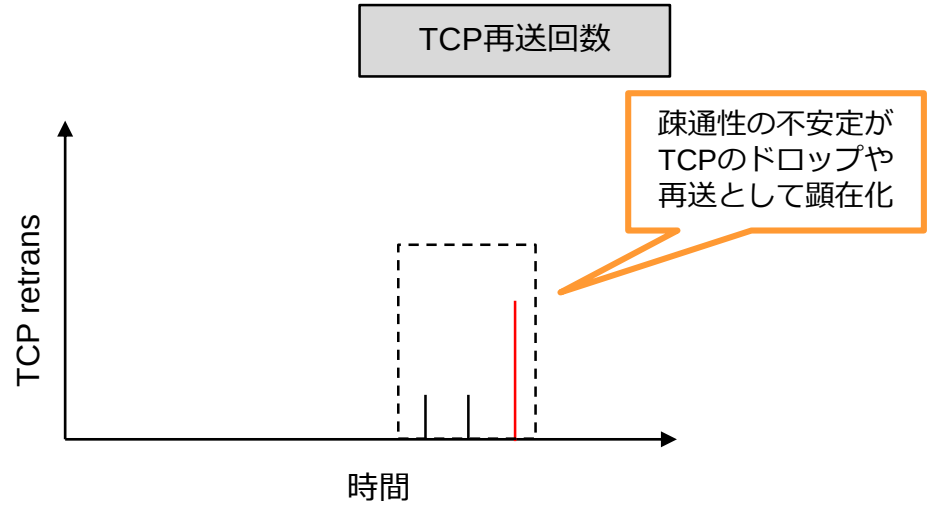
KDDI Research

BGPセッションの異常検知



内部的にKEEPALIVEの再送が発生していることをeBPFで検知し、ネットワークの不良を予兆

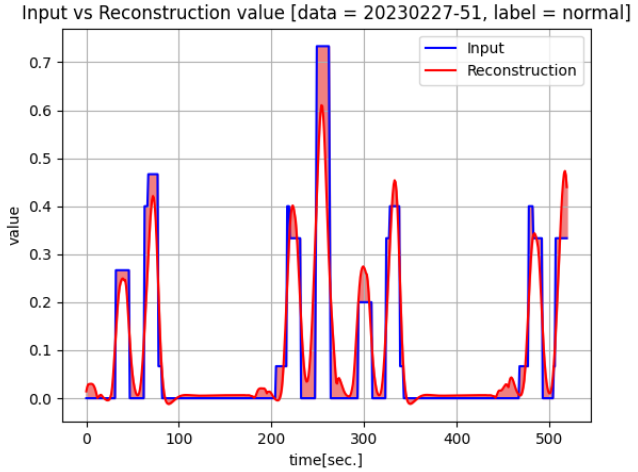




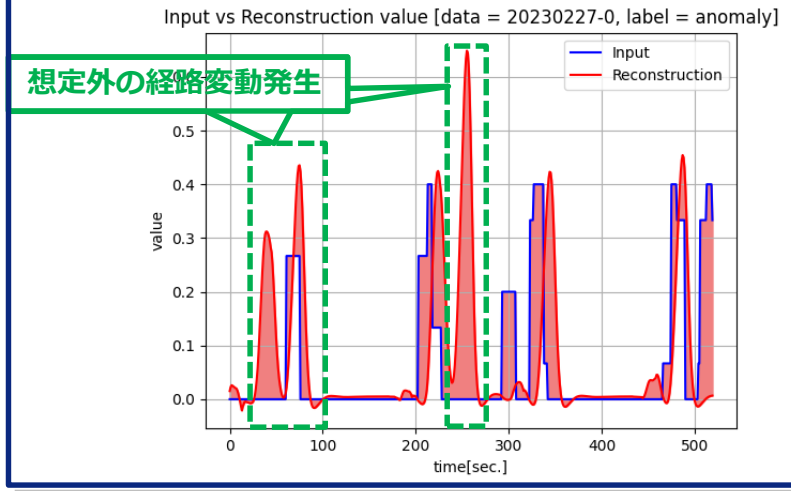
実験結果サマリー

オペレーションミス検知

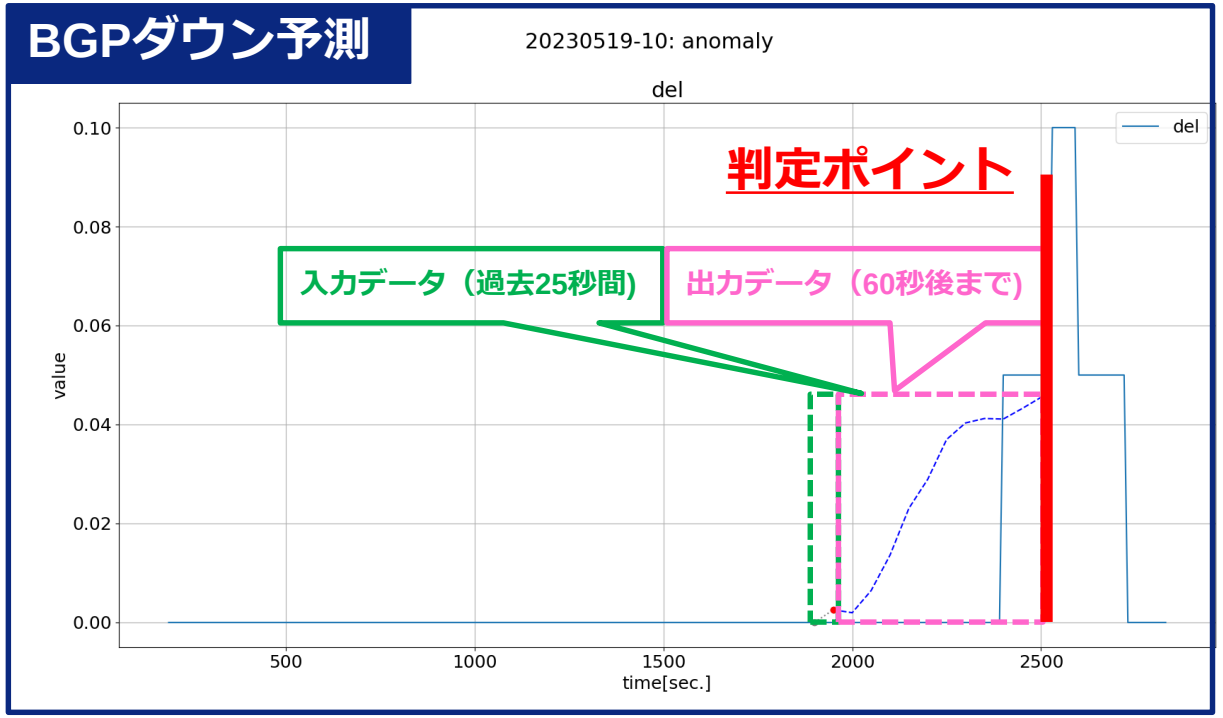
- 正常な手順で実施した場合



- 想定外の手順で実施した場合



BGPダウン予測



評価スコア

- シナリオ1
  - 手法:LSTM autoencoder
  - F1score: 0.82
- シナリオ2
  - 手法:LSTM
  - F1score: 0.81

まとめ

■ ネットワークにおけるモニタリング

- モニタリングを行う情報の精査や取得の高速化が必要
- ルータにおいて経路情報の取得に課題あり

■ eBPFを用いたネットワークモニタリング手法

- オペレーションミス検知、BGPセッションダウン予測ともに異常検知を行うことができた
- F1scoreは0.8

■ 今後の課題

- ルータ上のシステムにおける全体的な評価（eBPF観測のオーバーヘッドなど）
- eBPFの他メトリックの使用検討
- HWのホワイトボックスでの動作実証

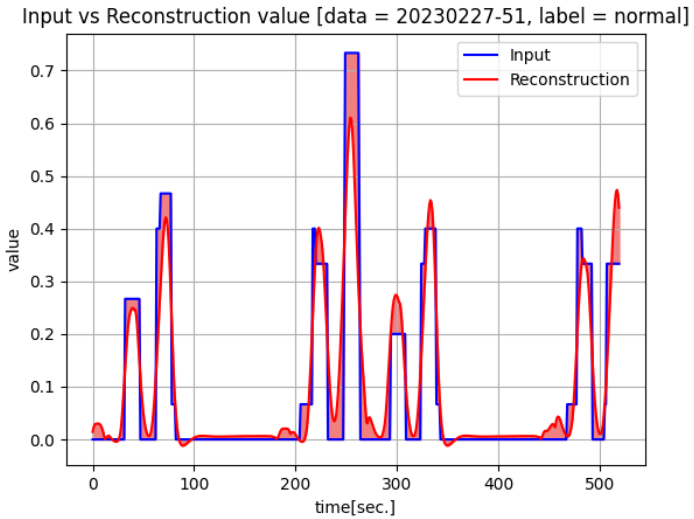
KDDI Research

参考資料

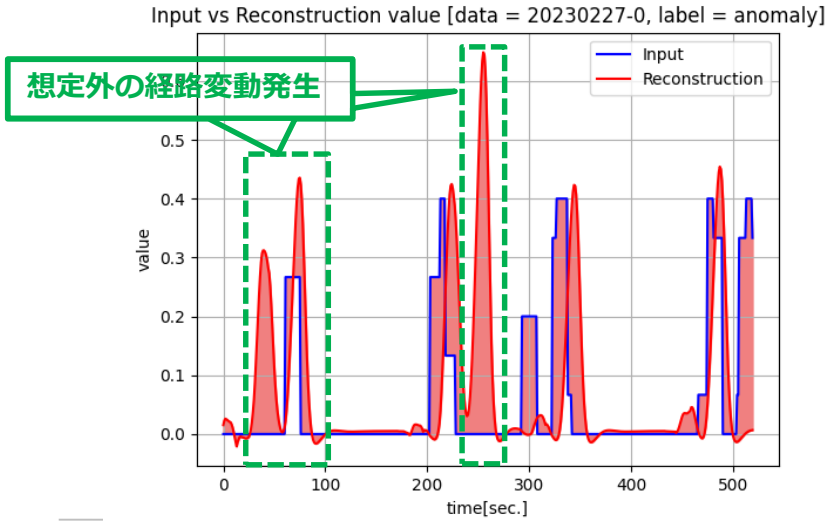
（参考資料）シナリオ1：オペレーションミス検知の実験結果詳細

KDDI Research

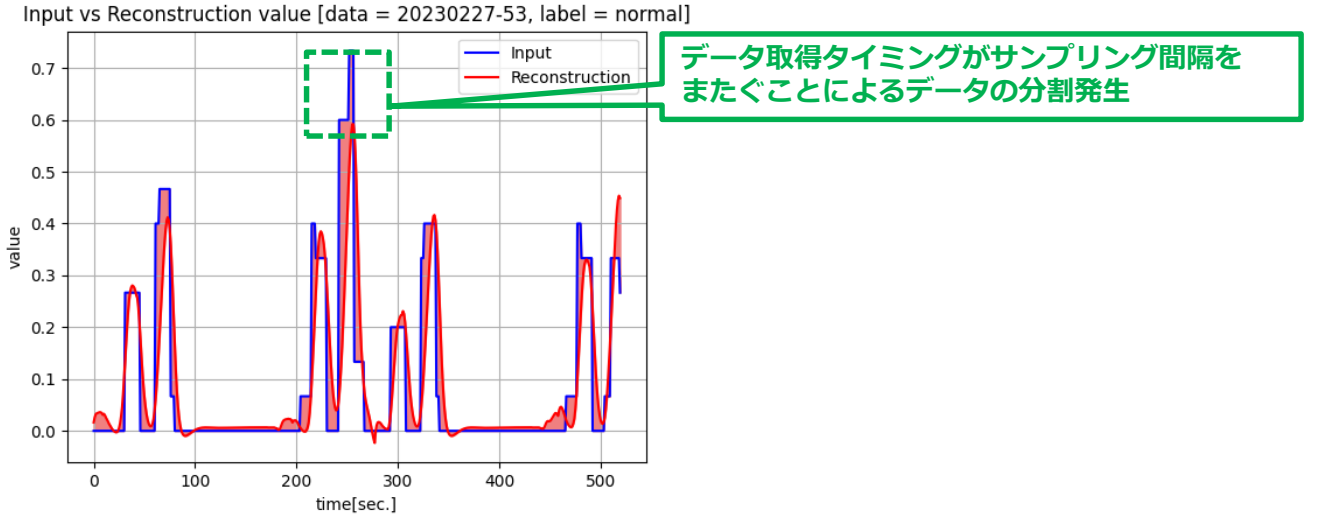
● 正常な手順で実施した場合



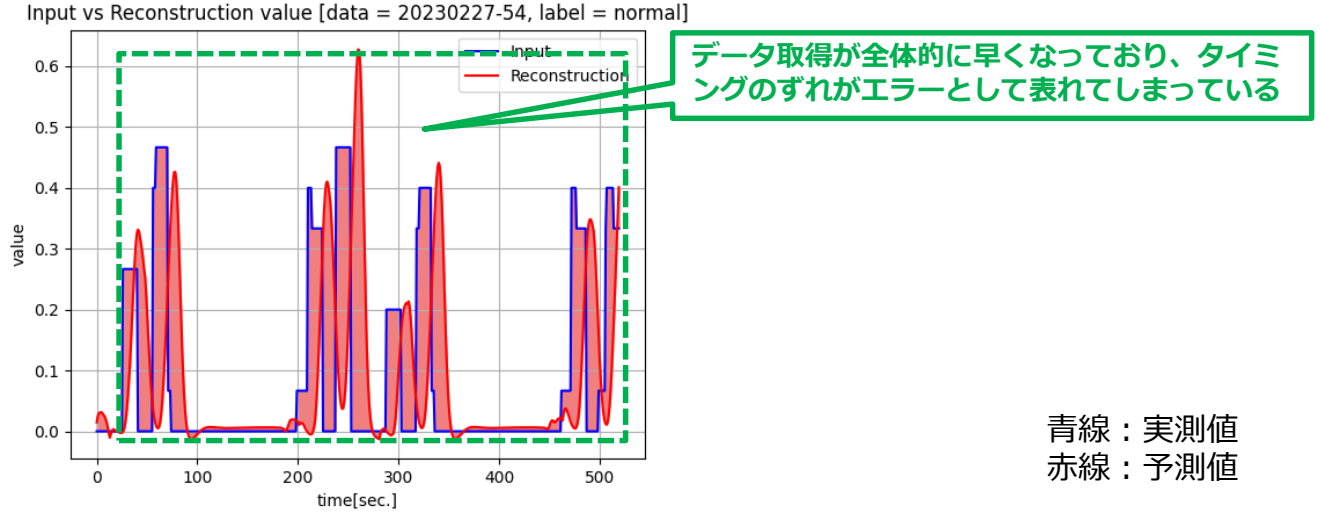
● 想定外の手順で実施した場合



● 正常な手順であるが、取得データの形状に変動がある場合



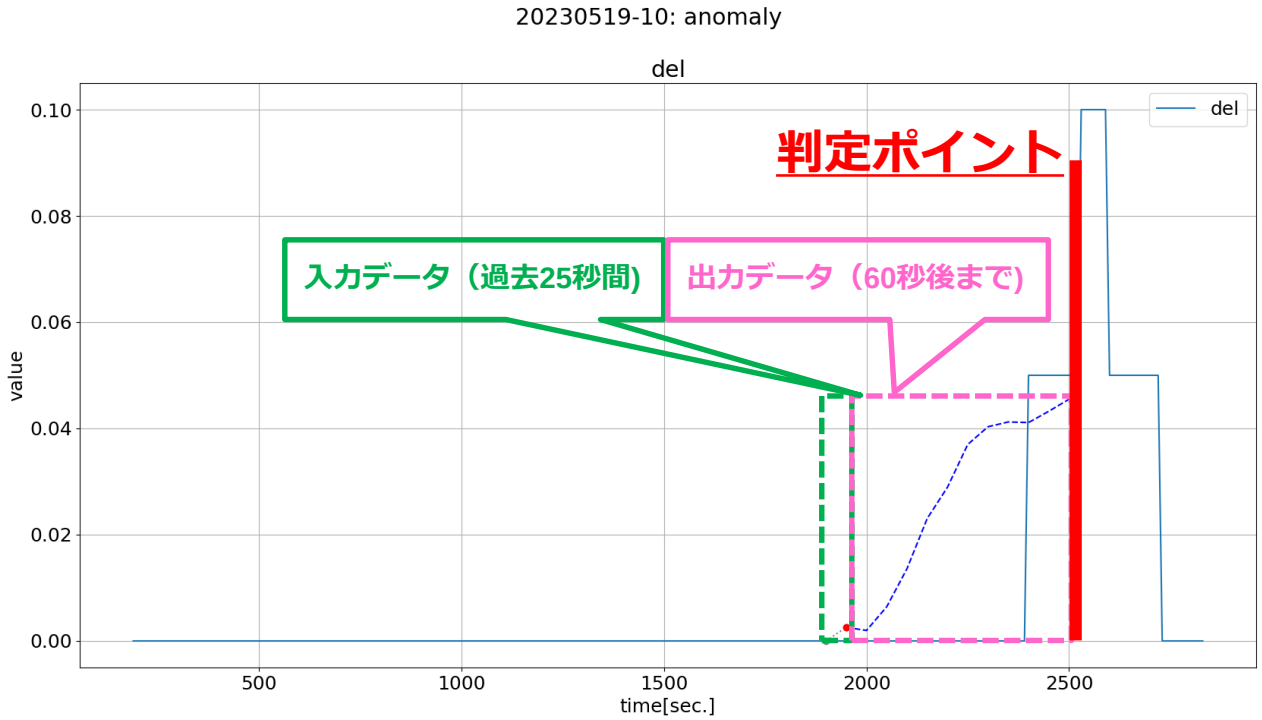
● 正常な手順であるが、取得データの時間軸がずれた場合

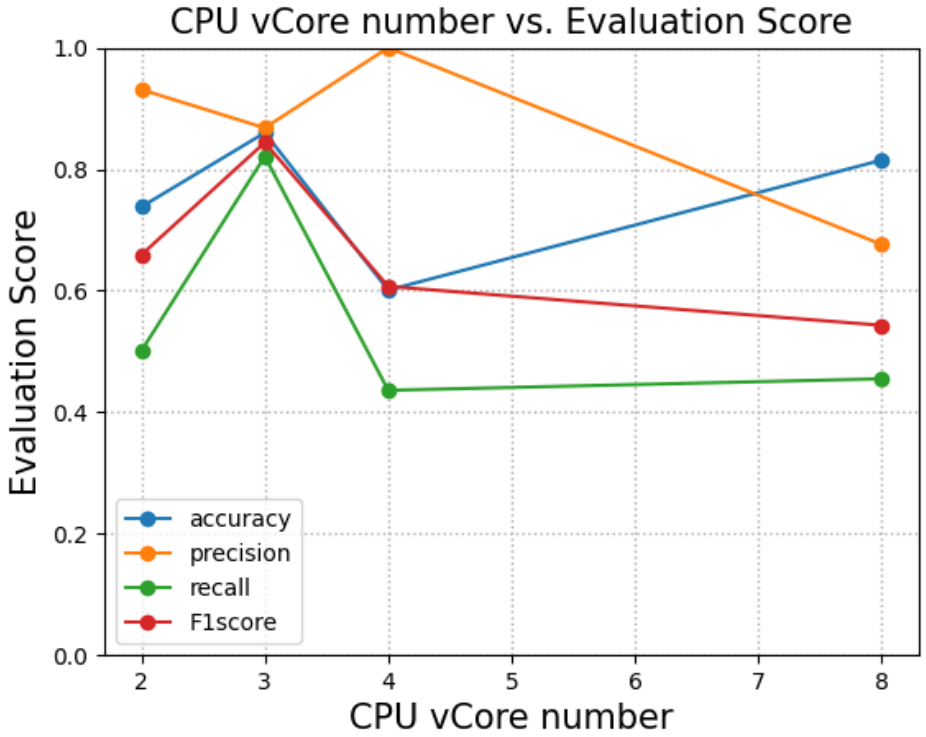


青線：実測値
赤線：予測値

（参考資料）シナリオ2：パケロス率上昇によるBGPセッション不安定の結果詳細

CPUデータによる異常検出の精度がよい理由として、 BGPネイバーの切断直前にCPU使用率が高騰することが挙げられる。SONiCに割り当てするvCoreの数を増減させた場合、3コア割り当て時が最もF1scoreが高く、コアを減らした場合や増やした場合にF1scoreの劣化が見られた





（参考資料）実験結果サマリー

eBPFデータを観測することで、シナリオ1においてはF1scoreの改善が見られた。しかしシナリオ2においてはCPUデータを用いたほうがF1scoreが高い結果となった。

- 共通
  - データセット
    - 訓練データ　200set
    - テストデータ 100set
    - 異常データ割合:50%
  - 閾値
    - 平均値+2σ
- シナリオ1
  - 手法:LSTM autoencoder
  - F1score: 0.82
- シナリオ2
  - 手法:LSTM
  - F1score: 0.81