

定数メモリ使用量を実現する効率的な Transformer アプローチによる ネットワーク侵入検知

埼玉大学 大学院理工学研究科

小林 将道

渡部 康平

研究背景

ネットワーク侵入検知システム(NIDS: Network Intrusion Detection System)が攻撃の対策の一つ。

攻撃とは、システムの機密性や可用性を侵害しようとする悪意ある活動である。

シグネチャ型

- ・ 既知攻撃を高速に照合できる。
- ・ 未登録の攻撃は検知が難しい。



異常検知型

- ・ 正常からの逸脱を学習する。
- ・ 未知攻撃にも対応しうる。

未知攻撃には、深層学習型の異常検知が重要になる。

研究目的

Transformerのアーキテクチャを用いてネットワークトラフィックの特徴量やシーケンスを学習し、正常・異常の分類を行う侵入検知システムであるTransformerベースNIDSに Deep Equilibrium Model(DEQ) を適用し、**検知精度を維持しつつメモリ消費量を削減する。**

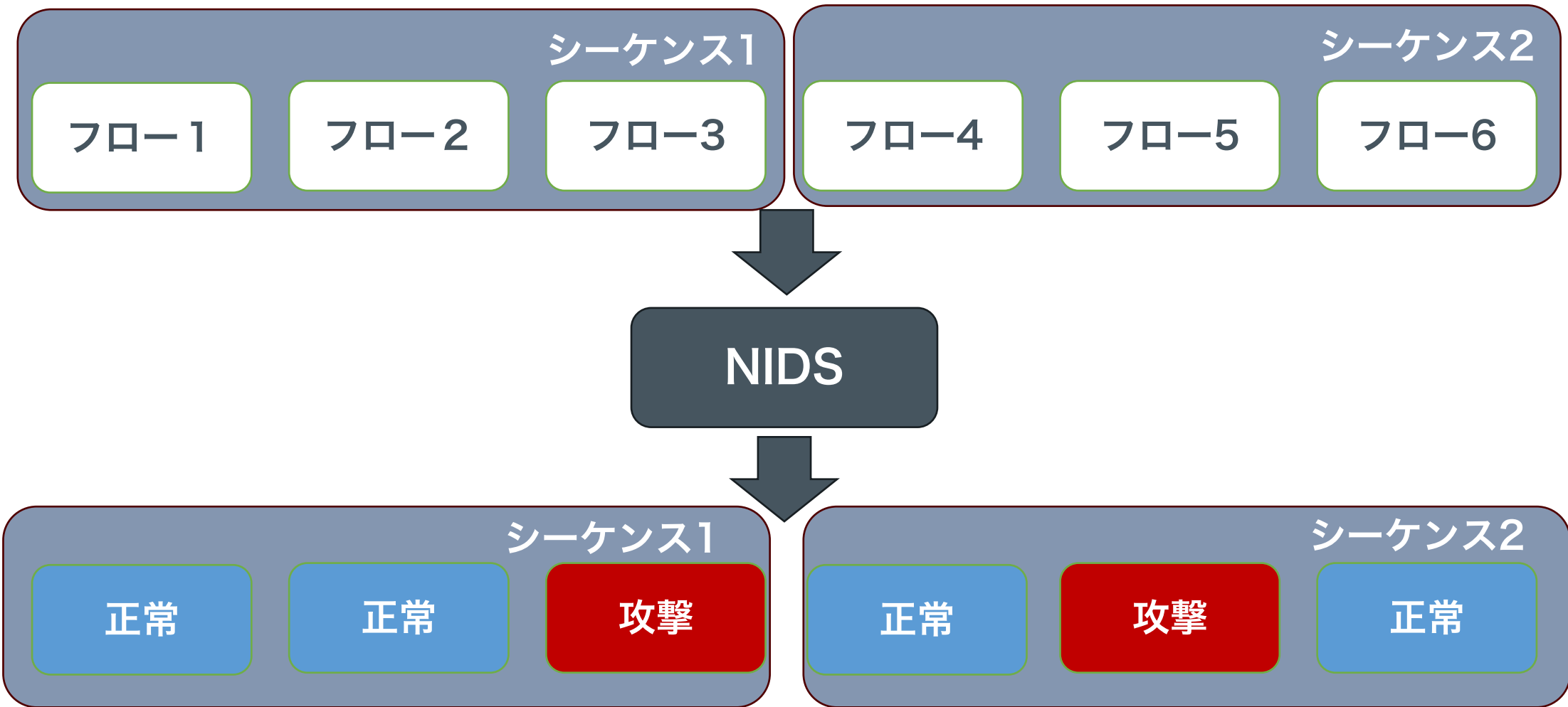
DEQ: 層を明示的に積み重ねず、収束する値を求めるモデル

無限の深さのネットワークと等価の表現力を獲得する。

学習時のメモリ消費量を $O(L)$ から **$O(1)$** にする。

問題設定

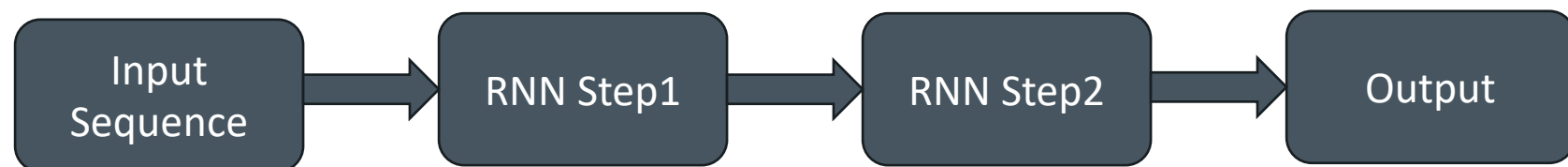
連続する通信フローを識別



関連研究: RNN/LSTMからTransformerへ

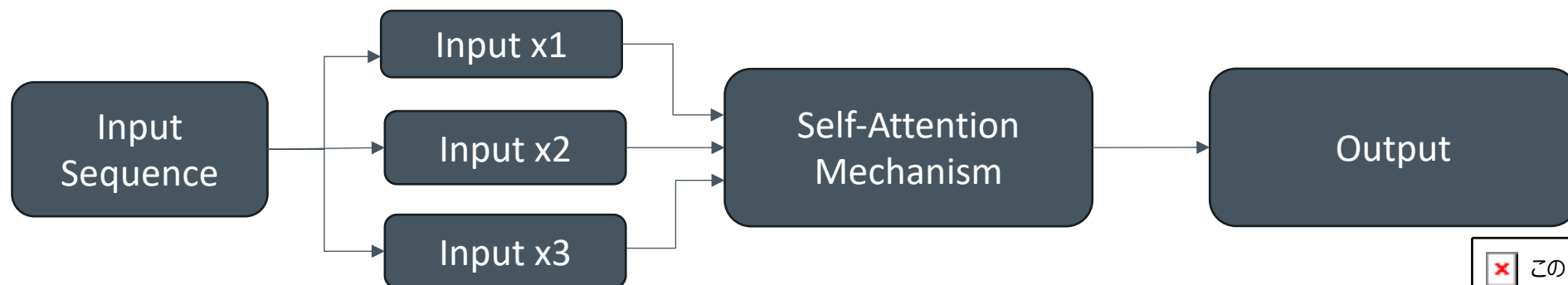
RNN/LSTM系

RNN,LSTMが時系列データを処理するモデルとして利用されてきた。



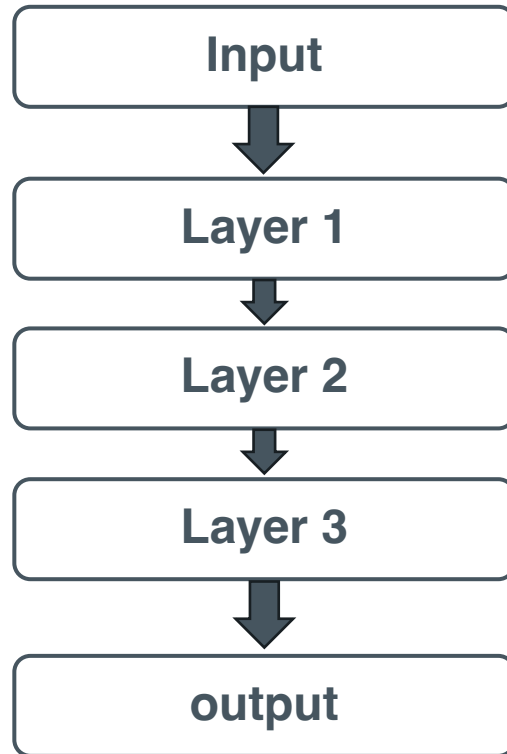
Transformerの登場

Vaswaniら: 「Attention is All You Need」でTransformerを提案した。**並列計算の達成、長期的な依存関係の学習の課題を解決した。**



Transformerの課題

Transformerの深層化



各層の計算結果を
逆伝播まで保存する.

GPUメモリ

Transformerは層が増えるほど、複雑な特徴、より長い依存関係を捉えられると言われている.

しかし、層が増えるほどGPUメモリ使用量が $O(L)$ で増えていく.

また、深層化をするほど性能が上がるわけではないということもわかっている.

深層化 ≠ 高性能

メモリ削減法

Standard Transformer

$O(L)$

全中間活性化を保存する.

Gradient checkpointing

$O(\sqrt{L})$

一部の活性化を保存し, 必要時に再計算する.

Reversible network

$O(1)$

後続層の出力から活性化を復元する.

DEQ

$O(1)$

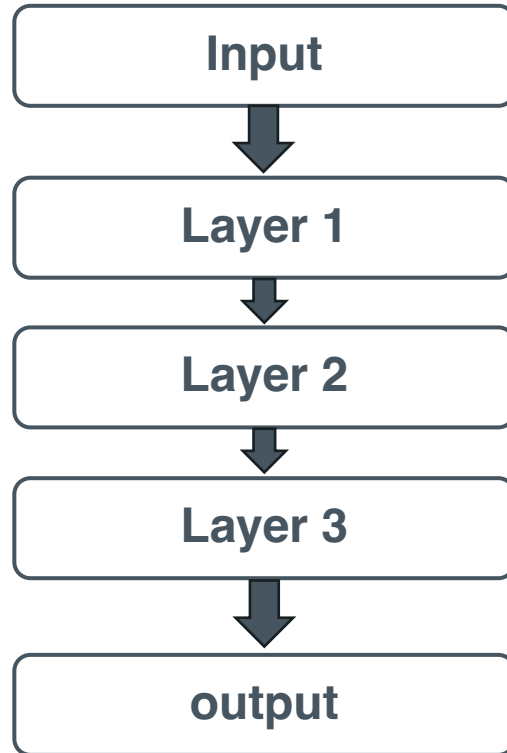
equilibrium pointを直接微分し, 層構造を前提としない.

DEQは「層の保存方法」ではなく, 計算の定義を変える.

Chen et al., 2016; Gomez et al., 2017; Bai et al., NeurIPS 2019.

DEQの基本概念

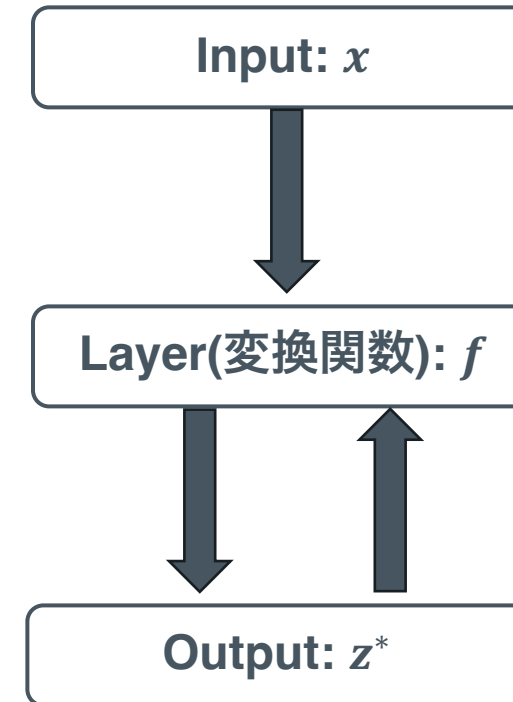
通常の深層モデル



L層を明示的に積み重ねる.

Bai, Kolter, and Koltun, Deep Equilibrium Models, NeurIPS 2019.

DEQ



一つの層に何度も出力を入力する。
出力は収束し、無限の層と等価な表現になる

$$z^* = f(z^*; x)$$

DEQによる勾配計算

順伝播

$g(z^*; x) = f(z^*; x) - z^* = 0$ となる根を見つける問題として定式化する。
Broyden法, Anderson法を用いて数値的に解く。

逆伝播

Implicit Differentiationを利用する。

$$\frac{\partial \ell}{\partial \theta} = - \frac{\partial \ell}{\partial z^*} (J_g^{-1}) \frac{\partial f(z^*; x)}{\partial \theta}.$$

J_g^{-1} : z^* における逆ヤコビ行列

θ : 任意の変数

ℓ : 損失関数

保存するのは z^* のみ
学習メモリは層数 L に
対して $O(1)$

※ $O(1)$ は層数 L に対する性質であり, 系列長やbatch sizeには依存する。

DEQの安定化

学習の不安定性

初期のDEQは、ヤコビアンの特値半径が、増大し**勾配爆発を引き起こす問題**があった。

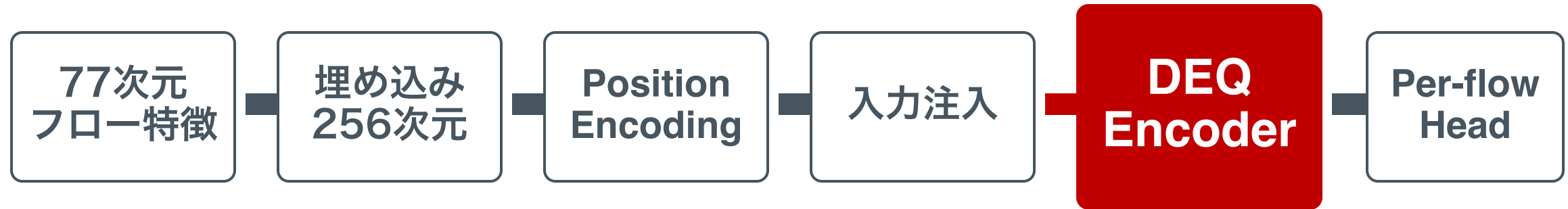
特値半径: 行列の特値の絶対値の最大値

解決策: ヤコビアン正則化

Baiらによって提案された。損失関数に正則化項を追加した。

特値半径が1に近づきすぎることを防ぎ、安定性と高速化を実現した。

提案モデル



重みを共有したSelf-Attention
Transformer変換



root-findingで z^* へ収束

分類タスクのためEncoderのみを用いる.

Architecture follows the weight-tied Transformer formulation of Bai et al., 2019.

クラス条件付きJacobian正則化

通常のJacobian正則化をクラス条件付きに拡張する.

BENIGN



$$J_i \rightarrow 0$$

J_i : ヤコビアンノルム

以下の式において $t = 0.8$ とすることで攻撃データのヤコビアンノルムを大きくする

ATTACK



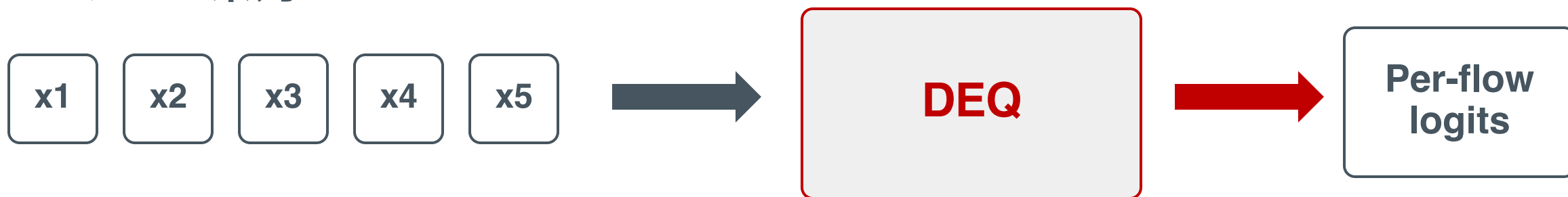
$$J_i \rightarrow 0.8$$

$$\mathcal{L}_{\text{jac}} = \frac{1}{|\mathcal{M}_{\text{benign}}|} \sum_{i \in \mathcal{M}_{\text{benign}}} J_i + \frac{1}{|\mathcal{M}_{\text{attack}}|} \sum_{i \in \mathcal{M}_{\text{attack}}} (J_i - t)^2.$$

M_{benign} : 正常サンプル数 M_{attack} : 攻撃サンプル数

推論方法

フロー系列



各フローを個別に評価する.

推論時に行わない処理

Broyden法による逆伝播



勾配追跡がないため、微分は発生しない.

データセット

CIC-IDS-2017 / 2018

8,973,342

総フロー数

77

連続値特徴量

28

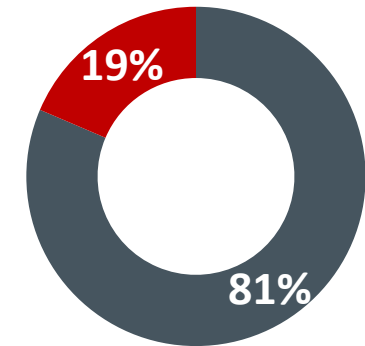
BENIGN+27攻撃

BENIGN 81.42%

ATTACK 18.58%

下位10クラスの合計は約0.035%

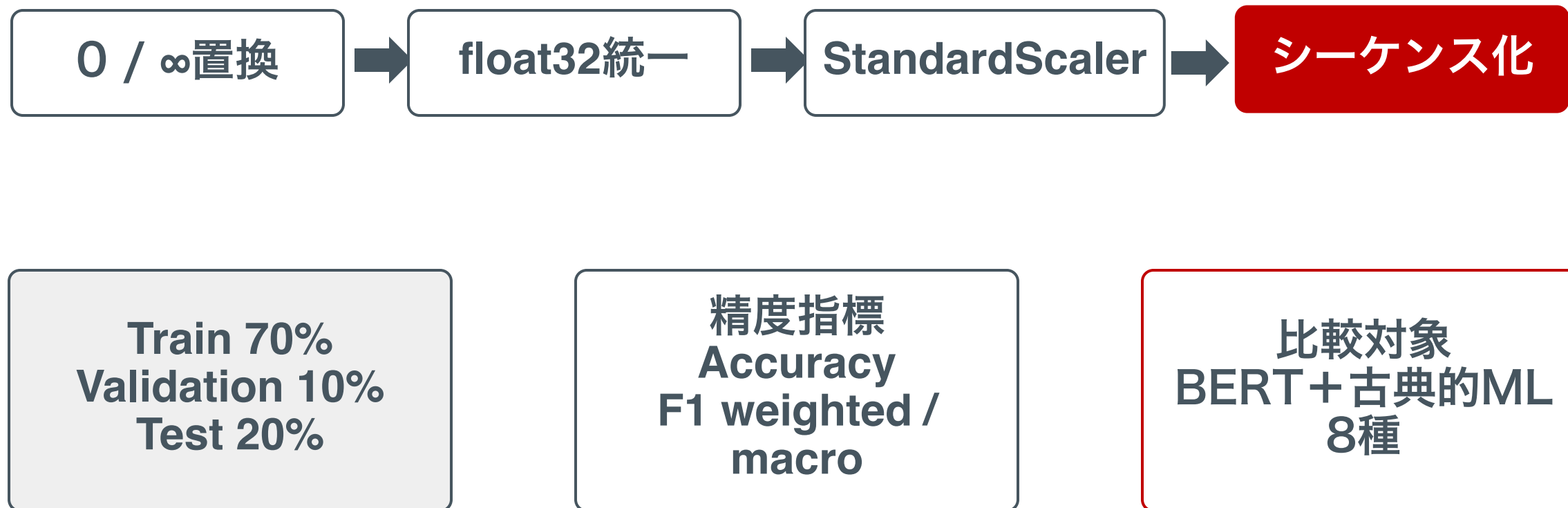
強いロングテール分布



■ BENIGN ■ ATTACK

前処理と実験設定

EXPERIMENTAL SETUP



test: 35,892 sequences / RTX 3080 Ti / PyTorch 2.5.1

※ 古典的MLは308次元の集約特徴, 深層モデルはフロー系列を入力とする.

検知性能

モデル	F1 weighted	F1 macro	Accuracy
Proposal	0.9802	0.7396	0.9839
BERT	0.9648	0.5842	0.9688
MLP	0.9617	0.5409	0.9673
Random Forest	0.9613	0.5607	0.9679
LinearSVC	0.9564	0.5417	0.9650
KNN	0.9562	0.5548	0.9626
Decision Tree	0.9511	0.5254	0.9497
SVM	0.9488	0.4776	0.9609
AdaBoost	0.8354	0.1336	0.8849
GaussianNB	0.2790	0.3929	0.2499

モデル効率

モデル	パラメータ数	スループット	学習時GPUメモリ	推論時GPUメモリ
proposal	1.17M	5740 (seq/s)	614MB	241MB
BERT	19.79M	4211 (seq/s)	668MB	394MB

提案手法のパラメータ数は、BERTの約6%にとどまり、小さなモデルとなった。
また推論時のスループットにおいてもBERTを上回ることができた。
学習時GPUメモリは、BERTに比べて大きくは変わらなかった。
推論時GPUメモリは、BERTの約61%になった。

正則化効果

損失関数	入力ベクトルの次元数	パラメータ数	F1 weighted	F1 macro	Throughput
DEQ(d=512)	512	4.99M	0.9719	0.5314	1701 (seq/s)
DEQ(d=256)	256	1.17M	0.9728	0.5466	4330 (seq/s)
proposal	256	1.17M	0.9780	0.5991	6101 (seq/s)

入力ベクトルの次元数を減らしたことで、F1 score, Throughputともに向上した。
損失関数の正則化項を拡張したことで、F1 score, Throughputともに向上した。
導入した正則化項が、期待通りに機能していることがわかった。

ウィンドウ長の影響

seq_len	Accuracy	F1 weighted	F1 macro	Throughput	推論時GPU	学習時GPU
50	0.9865	0.9841	0.6518	259,450	257	547
128	0.9842	0.9813	0.5868	251,648	313	740
256	0.9857	0.9822	0.5091	218,624	313	1029
512	0.9830	0.9760	0.4928	173,056	313	1733
1024	0.9799	0.9720	0.4980	122,880	313	3141

Throughputの単位はflow/s = (seq/s * seq_len)

seq_len = 50が精度・速度・メモリのすべてで最良

結論と今後の課題

01 学習メモリを層数 L に対して $O(1)$ 化した.

02 精度3指標で9ベースラインを上回った.

03 BERTの約6%のパラメータ数で実現した.

限界

- ・ ロングテール分布.
- ・ 未知攻撃は未評価.
- ・ オフラインデータのみ.

今後：希少クラス・未知攻撃・実トラフィックで検証する.

GPUメモリ使用量がそこまで減っていない

$O(1)$ は層数に対する計算量の話であり、GPUメモリ使用量が一定になるわけではない。実装では、根探索の履歴、Attentionの一時領域などが残るため、そこまで減ることがなかったと考えている。

攻撃サンプルのJacobian目標値を0.8にした理由

ヤコビアンノルムが1未満であると収束安定性を保つことができる。その上で、バランスをとって0.8にした。実験を行い、0.8が結果として良かったため採用した。

