# Data augmentation of anomaly logs considering emotions

**SCIS 2023** 2023 Symposium on
Cryptography and Information Security
Fukuoka, Japan, Jan. 24 – 27, 2023
The Institute of Electronics,
Information and Communication Engineers

# 感情を考慮した異常ログ生成手法の検討
# Data augmentation of anomaly logs considering emotions

鈴木 伶哉 * 竹内 廉 * 柳生 航平 * 西垣 正勝 * 大木 哲史 *
Reiya Suzuki Ren Takeuchi Kohei Yagyu Masakatsu Nishigaki Tetsushi Ohki

**あらまし** システムが出力するログは，運用やセキュリティの面で重要であり，必要不可欠となっている．システムが故障した際にはログを確認して故障診断を行うが，システムの大規模化により，ログから異常な振る舞いを手動で検出するには多くの時間を要する．そのため，機械学習を利用した異常ログの自動検出が多く研究されている．しかし，異常ログは正常ログに比べると出現頻度が稀なため，学習データに不均衡が生じる．これを解消するために学習データを水増しするデータ拡張が用いられるが，対象とするシステムのログの形式や異常ログなどについての学習が必要である．そこで，本研究では，異常ログに Fail などの Negative な単語が多く含まれることに着目した，システムに依存しない汎用的なデータ拡張手法を提案する．具体的には，正常ログの感情を Negative に変換して異常ログを擬似的に生成する．実験では，感情を考慮した提案手法と感情を考慮しない単純な手法をそれぞれ用いて少ない異常ログを水増しする．これにより学習した異常ログ検知器の検知精度を比較することで，提案手法の有効性について考察する．

**キーワード** ネットワークセキュリティ，異常ログ検知，データ拡張

## 1 はじめに

システムが出力するログは，運用やセキュリティの面で重要であり，必要不可欠となっている．ログはイベントの発生時刻やエラーといった，さまざまな事象を記録する．システムが故障した際には，生成されたログをチェックして故障診断を行い，インシデントが発生した際には，ログから IP アドレスなどを確認してフォレンジック調査を行う．しかし，システムの大規模化により，ログから異常な振る舞いを手動で検出するには多くの時間を要するようになった [1]．そこで，ログから異常な振る舞いを自動検出する技術が研究されている．主な手法として，教師あり学習，半教師あり学習，教師なし学習といった機械学習を用いた手法が挙げられる [2–4]．正常ログは，システムが稼働していれば多く手に入れることができ，学習に用いることが容易である．しかし，異常ログはシステムに異常が発生した際にのみ得られるものであるため，正常ログに比べると非常に数が少ない．また，異常とラベル付けをするためには，専門スキルや多量なログの精査が必要であることから，多量のログからラベル付けにより学習データを作ることが難しい．このような背景から，半教師あり学習や教師なし学習を用いた異常ログ検知の研究が注目されている．半教師あり学習では，多量の正常ログと少量の異常ログを学習し，教師なし学習では，多量の正常ログのみを学習する．いずれの手法においても，正常ログ以外のログを異常として判定することで異常ログ検知を行うため，異常ログを学習に用いる必要がほとんどない．一方，教師あり学習では正常ログと異常ログの双方について大量のログを学習し，正常と異常の 2 クラス分類を行う．識別精度の観点からは，教師あり学習は，正常ログと異常ログの両方について学習を行うため，半教師なし学習や教師なし学習と比べて，精度が最も優れていることがわかっている [5]．しかし，異常ログは正常ログに比べて圧倒的に数が少なく，学習データ量に不均衡が生じるという問題がある．

この問題に対し，データ量が少ないデータを水増しする，データ拡張が用いられる．データ拡張は，画像を用いた機械学習の分野でよく行われており，画像のリサイズや回転などの手法が用いられている．異常ログのデータ拡張では自然言語処理モデルの BERT を用いた研究 [3] や，異常ログを元に検知を回避するようなログを生成する研究 [6] が存在する．しかし，データ拡張の適用には，対象とするシステムのログの形式や異常ログについての情報が必要である．

Zhang ら [2] はテキストデータの感情である Positive, Neutral，Negative を分類する Sentiment Analysis を異常ログの検知に適用した．異常ログはシステムに関わら

* 静岡大学, 静岡県浜松市中区城北 3 丁目 5-1, Shizuoka University, 3-5-1 Jo-hoku, Naka-ku, Hamamatsu City, Shizuoka, Japan

ず，正常ログに比べて error や fail といった Negative な単語が含まれている．そのため，ログレベルごとに感情分析を行い，DEBUG レベルを Neutral，ERROR レベルを Negative として学習している．機械学習を用いた異常ログ検知では，ログの形式やログメッセージが異なるため，対象とするシステムごとに再学習する必要がある．これに対し，ログを感情として捉えることで対象とするシステム以外のログも学習に含めることができ，更に正常ログと比べて異常ログには Negative な単語が多い点に着目することで精度が向上することを示している．しかし，学習するデータセットにおける正常ログ数と異常ログ数の不均衡に対する問題は解決されていない．

本研究では，ログの感情を考慮したデータ拡張の手法を提案する．Zhang ら [2] が検証した，異常ログは正常ログに比べて Negative な単語が含まれている点に着目し，表 1 のように文の感情である Positive, Negative の変換を行う．具体的には正常ログを Positive，異常ログを Negative とし，正常ログの感情を Negative に変換することで異常ログのデータ拡張を行う．この変換により，既存のデータ拡張手法で問題とされていた，対象とするシステムのログの形式や異常ログについての情報が不要となる．したがって，必要な処理は文内の感情を変換するのみとなるため，システムに依存しない汎用的なデータ拡張が可能となる．通常の運用によって得られる異常ログのみからデータ拡張により多量の異常ログを生成する場合，多様性が低く，未知の異常ログが検知できず，精度向上には繋がらないことが推測できる．このような問題に対し，正常ログを用いて擬似的に異常ログを生成し，多様性を高めることで，学習データに含まれていない異常ログを検知する．実験では，データセットに何も手を加えない場合，対象とするシステムではないシステムの単語を用いて感情を考慮せずにデータ拡張を行った場合，提案手法の正常ログを感情を考慮したデータ拡張により Negative な感情を持つ異常ログへと変換した場合を比較することで提案手法の有効性を示す．また，学習に使用するログの量を減らした状態でデータ拡張を行うことで，実際のシステムで異常ログが少ない状態での運用を想定し，運用開始時からどの程度の段階において提案手法による検知が有効となるかについても示す．

本研究の貢献は，以下の通りである．

- 正常ログの感情を Negative に変換することによる感情を考慮したデータ拡張手法の提案
- HDFS データセットに対して，学習データが十分に揃っていない状況を想定した上で，提案手法を適用した場合の精度向上

**表 1:** Positive → Negative 変換の例

| Positive | Negative |
|---|---|
| **good** drinks, and **good** company. | **bad** drinks, and **bad** company. |
| it is consistent and the staff is always **friendly**. | it is consistent and the staff is always **rood**. |
| i will **definitely** go here again ! | i will **never** go here again ! |

## 2 関連研究

### 2.1 異常ログ検知の既存研究

異常ログ検知の既存研究として，He ら [7] による loglizer がある．He らは，異常ログ検知のベースラインとなるソースコードが公開されていないことやそれぞれのアルゴリズムが比較されていないことを問題とし，3 つの教師あり手法と 3 つの教師なし手法を実装し，比較実験を行った．教師あり手法は，ロジスティック回帰 [8]，Decision Tree [9]，SVM [10]．教師なし手法は，クラスタリング [11]，PCA [12]，Invariants Mining [13] である．異常ログ検知の際には，複数のログをいくつかの塊に分割して判断を行う．ログの分割方法として，タイムスタンプを基準に分割する方法とセッション ID ごとに分割する手法がある．それぞれの分割データ中で，どのような事象が発生したかを示すイベント ID の発生回数行列を特徴として学習する．例えば [0,0,2,3,0] の場合，イベント 3 が 2 回，イベント 4 が 3 回発生したことになる．この手法を用いて異常ログ検知の精度比較を行っているが，学習データにおける正常ログと異常ログの不均衡に対する対策は行われておらず，データ拡張による精度向上が見込まれる．

### 2.2 異常ログのデータ拡張

異常ログのデータ拡張として，山本ら [6] の手法がある．山本らの手法は，誤検知が発生しやすいログについて着目し，そのログを元に合成異常ログを生成した．生成手順としては，異常ログの近傍にある正常ログの特徴量を多く含むように異常ログの特徴量を修正し，既存の検知器に正常と判定されるようなログを生成する．この生成方法により，異常ログ検知を回避できる異常ログを擬似的に生成する．この手法で生成したログを学習に含めることで，検知漏れを起こしやすい異常ログについて学習することが可能となり，検知精度が向上した．しかし，異常ログを元に水増しを行うため，一定量の異常ログが必要である点，また多様性が生まれにくい点が問題となる．

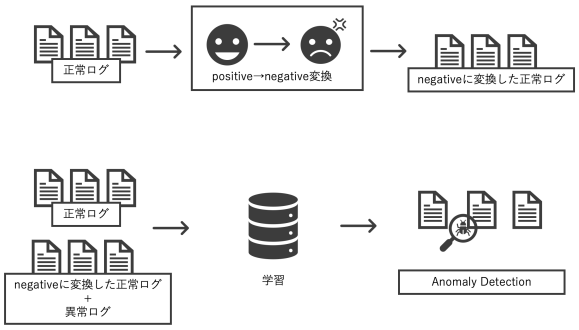



**図 1:** 提案手法の概要図

### 2.3 感情を考慮した異常ログ検知

自然言語処理にて，テキストデータの Positive，Neutral，Negative を分類する Sentiment Analysis と呼ばれる手法が存在し，商品レビューや SNS の分析などに広く適用されている．Zhang ら [2] は Sentiment Analysis をログの異常検知に応用した．ログレベルごとに感情分析を行い，DEBUG レベルであれば Neutral，ERROR レベルを Negative として学習している．手順としては，ログメッセージを収集し，全ての単語を小文字にする，動詞を原形にするといった前処理を行い，感情分析モデルの学習をした後，異常検知を行うという流れである．基本的に機械学習を用いた異常検知では，ログフォーマットやログメッセージが異なるため，対象とするシステムごとに再学習する必要がある．しかし，対象とするシステム以外のログも学習した方が精度が高いという結果が示された．原因として，ログレベルごとの出力が開発者に左右されることが挙げられており，実際に BeeGFS データセットでは，単なる変数の print デバッグが WARNING レベルなどで表示されているログも存在した．他のシステムのログを Negative などの感情として捉えて学習することで，対象とするシステム以外のログも学習に含めることができ，精度が向上するという結果になった．しかし，感情を考慮したデータ拡張を行って，学習データの不均衡を解消した，異常ログ検知の精度向上は行われていない．

## 3 提案手法

### 3.1 概要

図 1 に，提案手法の概要図を示す．まず，正常ログの感情を Positive から Negative に変換する．その後，正常ログを正常，感情を Negative に変換した正常ログとデータセットに含まれている異常ログを異常として学習し，教師あり学習による異常検知を行う．

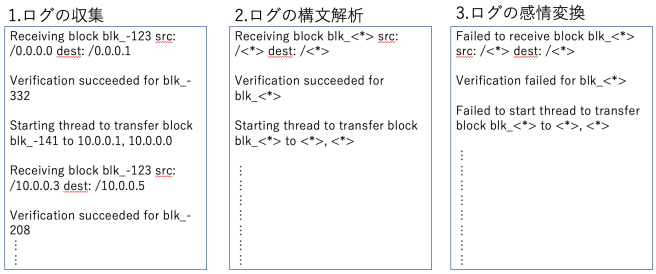



**図 2:** ログの変換手順

### 3.2 ログのデータ拡張

データ拡張を行うための，ログの変換手順の概要図を図 2 に示す．まず，学習に用いるログを収集する．その後，logparser [14,15] を用いてログの構文解析を行う．ここで「Verification succeeded from 10.0.0.0」と「Verification succeeded from 10.0.0.1」というログを例に説明する．この場合「Verification succeeded from」は複数のメッセージに共通する部分であり（以降，これを共通部と呼ぶ）．これに対して「10.0.0.0」といった IP アドレスなどの変化する箇所である（以降，これを変数部と呼ぶ）．ログの構文解析では，ログメッセージの変数部を<*>に置き換え，テンプレートに変換する処理を行う．たとえば，「Verification succeeded from <*>」がテンプレートとなる．その後，テンプレートに変換したログメッセージの全文を対象に，以下のルールに従って正常ログの感情を Negative に変換する．

- Positive なログの感情を Negative に変換する
- 元から感情が Negative なログは変換しない
- 単語だけのログなど感情が変換できないログは変換しない

データ拡張は，データ拡張後の正常ログと異常ログの比率が $1 : (\alpha + x)$ となるようにパラメータ $\alpha$ を定め，実施する．ここで $x$ は，学習データに含まれている少量の異常ログの割合である．

なお，ログの感情変換手法に関しては，Zhang らと同様の Sentiment Analysis を用いる手法などが考えられるが，本稿が実施する実験においては，感情変換をデータ拡張に適用した際の影響をより厳密に調査するために，手動での感情変換を実施した．

### 3.3 感情を考慮した異常ログ検知

異常ログの学習および検知手法は loglizer [7] で提案された手法に対して，感情を考慮可能とした異常検知手法を用いる．

機械学習モデルには，loglizer に実装されている教師あり学習手法の，ロジスティック回帰，Decision Tree，SVM を用いる．

感情を考慮した異常検知を実現するために，入力として与えられた複数のログから特徴を抽出する．まず，特徴を抽出するためにログを分割する．分割には，loglizer で用いられていたセッション ID ごとに分割する手法を用いる．その後，セッション ID ごとに，どのようなイベントが発生したかを示すイベント ID の発生回数行列を算出する．加えて，イベント ID に紐づく，テンプレートに変換したログメッセージの感情度合いを測り，感情度合い行列を算出する．感情度合いの測定には，python のライブラリである textblob を用いる．感情度合いの数値は $[-1.0, 1.0]$ で示され，数値が少ないほど Negative，高いほど Positive である．

学習時には，イベント ID の発生回数行列と，テンプレートに変換したログメッセージの感情度合い行列を特徴として，モデルの学習に用いる．

検知時には，学習時と同じように，イベント ID の発生回数行列と，テンプレートに変換したログメッセージの感情度合い行列をモデルに入力し，異常ログの検知を行う．

## 4 実験・検証

### 4.1 実験方法

本実験では，感情を考慮した提案手法であるデータ拡張の有効性について検証する．まず，**異常ログ検知器評価**として異常ログ検知に用いる loglizer に，感情度合いを学習に含ませることで精度が変化するかを実験で確かめる．次に，**データ拡張評価**として，少量の異常ログを拡張した場合の検知精度の向上について評価を行う．ここでは，比較のために 3 パターンのログを用意する．1 つ目が何も手を加えないログ（*Plain* と呼ぶ）．2 つ目が対象とするシステムのログの単語を，違うシステムのログの単語で置き換えてデータ拡張したログ（*Simple* と呼ぶ）．3 つ目が提案手法の，対象とするシステムの正常ログの感情を Negative に変換してデータ拡張したログである（*Proposed* と呼ぶ）．これら 3 パターンのログをそれぞれ比較する．

また，現実のシステムにおいては，ログは稼働開始を基点として徐々に蓄積されていく．このため，稼働当初は学習に利用可能なログが少ないと考えられる．このことを考慮し，**学習データ数評価**として，学習データに使用するログの量と検知性能との関係についても比較を実施することで，提案手法の有効性を示す．

#### 4.1.1 実験環境

実験は，MacBook Pro 2020 M1 メモリ 16GB を搭載した macOS Monterey(version 12.3.1) のマシン上で行った．また，プログラムの実装や実行にあたって，Python 3.9.13，scikit-learn 1.1.2，numpy 1.23.3，pandas 1.5.0，textblob 0.17.1 を使用した．異常ログ検知には，loglizer の研究において精度が最も良好と報告された Decision tree を用いた．

#### 4.1.2 データセット

実験には公開データセットの HDFS [12] と BGL [16] を用いる．HDFS は Hadoop Distributed File System のログメッセージ 11,175,629 件が含まれているデータセットであり，どのセッション ID が異常であったかラベル付けされている．異常とラベル付けされているセッション ID の数は，575,061 件である．BGL は，BlueGene/L supercomputer system のログメッセージ 4,747,963 件が含まれているデータセットであり，各ログメッセージ一つずつに対して異常かどうかのラベル付けがされている．異常ログの数は 348,460 件である．

提案手法の学習，評価，およびデータ拡張には HDFS を用いる．また，Simple のデータ拡張にあたっては，BGL に含まれる単語を抽出し，これを置換対象の単語セットとして利用する．

#### 4.1.3 実験プロトコル

**異常ログ検知器評価**および**データ拡張評価**においては，データセットの 25%を学習データとし，学習データに含まれている異常ログの何割を学習に用いるかの割合（anomaly_ratio）を，10%，25%，50%，75%，100%と変更して評価を実施する．**データ拡張評価**にあたっては，$\alpha = 0, 0.25, 0.5, 0.75, 1.0$ としてデータ拡張を実施し，それぞれの場合における評価結果を比較する．

また，**学習データ数評価**においては，異常ログに加えて正常ログの量を変更した場合の影響を調査するために，学習データの割合（train_ratio）をデータセットの 1%，2%，3%，4%，5%，10%，20%，30%，40%，50%と変更し．それぞれに対して $\alpha = 0, 0.25, 0.5, 0.75, 1.0$ としてデータ拡張を実施した場合における評価結果を比較する．

#### 4.1.4 ログの変換

本実験では，3.2 節の手順に従ってログの構文解析およびテンプレートへの変換を行った．元のログの数とテンプレートの数を表 2 に記載する．

**データ拡張評価**において，Simple は，対象システムのログからランダムに選択した単語を，置換対象の単語セットからランダムに選択された単語と置き換えることによって作成した．また，Proposed におけるログの感情変換は，3.2 節で述べた通り手作業で実施した．この時，作業する人物による結果への依存性を軽減するためにセキュリティ専門の研究室に所属する 3 名の学生担当者による変換作業を実施した．変換作業にあたっては，それぞれの担当者が 3.2 節で記載したルールに従って変

**表 2:** 各データセットのログの数とテンプレートの数

| データセット | ログの総数 [件] | テンプレート数 |
|---|---|---|
| HDFS | 11,175,629 | 48 |
| BGL | 4,747,963 | 388 |

**表 3:** ログの感情を Positive → Negative 変換した結果

| Positive | Negative |
|---|---|
| **Receiving** block <*>src: <*>dest: <*> | **Failed to receive** block <*>src: <*>dest: <*> |
| Reopen Block <*> | **Can't** reopen block <*> |
| Verification **succeeded** for <*> | Verification **failed** for <*> |

換を行い，その後，変換した結果が全員同じでないものに関して議論し，結果を統合した．

ログの感情変換結果の例を表 3 に示す．Positive の列に変換前のログを，Negative の列に変換後のログをそれぞれ記載した．

#### 4.1.5 評価指標

実験の評価には，Precision，Recall，F1 score を用いる．それぞれは次の式で表される，異常ログは，Positive，正常ログは Negative として測定し，本検証で用いる混同行列を表 4 に示す．

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall}$$

### 4.2 実験結果

#### 4.2.1 異常ログ検知器評価

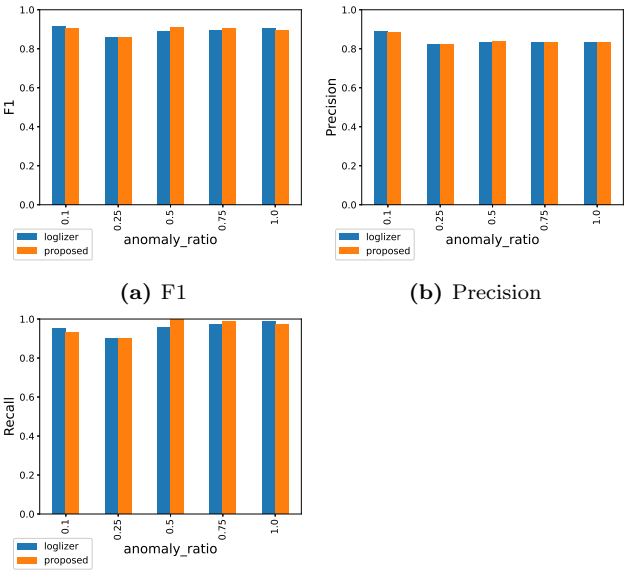
loglizer および提案手法による異常検知精度の結果を図 3(a)~(c) に示す．縦軸がそれぞれの評価指標の値，横軸が学習データの異常ログの数の割合であり，loglizer が感情度合いを学習に含めない実装，proposed が感情度合いを学習に含めた実装である．図 3(a)~(c) から，データ拡張を行わない場合には，感情を考慮した検知器を使用したとしても，異常検知精度の改善が見られないことがわかる．

#### 4.2.2 データ拡張評価

異常ログの数を変更した場合の検知精度の変化を図 4，5，6 に示す．縦軸がそれぞれの評価指標の値，横軸が学習データに含まれる異常ログの数の割合になっている．各折れ線がそれぞれデータ拡張の割合ごとの結果になっている．結果としては，F1 の値を見ると，異常ログの割合が 25%以上で，提案手法のデータ拡張の割合が 50%の時に精度が最もよくなるという結果だった．

**表 4:** 混同行列

| | | 実測 | |
|---|---|---|---|
| | | Positive | Negative |
| 予測 | Positive | TP (True Positive) | FP (False Positive) |
| | Negative | FN (False Negative) | TN (True Negative) |




**(a)** F1 **(b)** Precision

**(c)** Recall

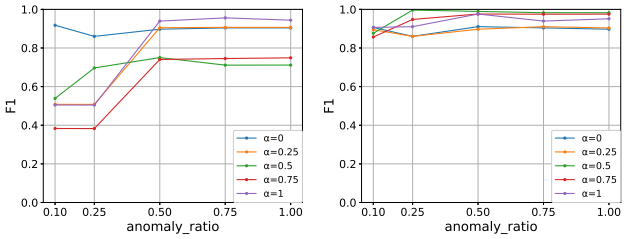
**図 3:** 感情度合いが精度に影響するかの比較

#### 4.2.3 学習データ数評価

学習データ数を変動させた場合の検知精度の変化を図 7，8，9 に示す．縦軸がそれぞれの評価指標の値，横軸がデータセットに対する学習データの割合であり，各折れ線がそれぞれデータ拡張の割合ごとの結果になっている．結果としては，F1 の値を見ると，学習データが少なすぎると精度が向上せず，学習データの数がデータセットの 20%から 30%付近で特に提案手法の有効性が見られる．しかし，それ以上学習データを増やすと，データ拡張の効果が見られない結果となった．

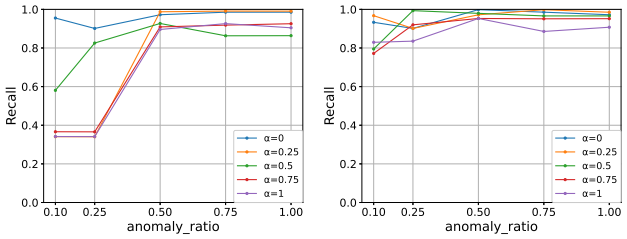
## 5 議論

実験結果の図 3，4，5，6 より，提案手法の感情を考慮したデータ拡張手法が，感情を考慮しないデータ拡張手法と比べて，高い検知精度となった．この結果から，正



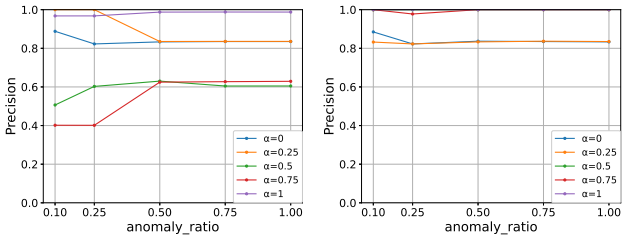

**(a)** Simple **(b)** Proposed

**図 4:** 異常ログの割合を変更した場合の F1 の比較



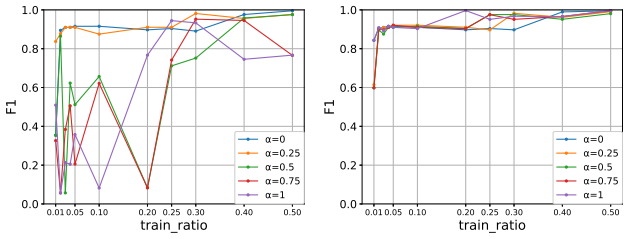

**(a)** Simple **(b)** Proposed

**図 6:** 異常ログの割合を変更した場合の Recall の比較




**(a)** Simple **(b)** Proposed

**図 5:** 異常ログの割合を変更した場合の Precision の比較




**(a)** Simple **(b)** Proposed

**図 7:** 学習データの割合を変更した場合の F1 の比較

常ログの感情を Negative に変換することによるデータ拡張が，異常ログ検知の精度向上に寄与すると言える．

図 4 より，学習データの異常ログに対する割合が 25% 以上で，正常ログに対するデータ拡張の割合が 50%の時に精度が最も高くなることがわかる．データ拡張を行う動機としては，学習データの不均衡を解消し，精度を向上させることであった．しかし，正常ログに対するデータ拡張の割合が 50%，つまり正常ログと異常ログの不均衡が存在する場合に最も高い精度が得られた．これは，感情を考慮したデータ拡張では，正常ログで発生したイベント発生回数行列に手を加えず，そのまま異常ログとして学習を行なったため，過剰に拡張を行った場合に，適切でないイベント発生回数行列を学習し，正常データを異常として誤検知することが要因であると考えられる．

また，図 7 より，学習データが少ない場合と充分にある場合では，データ拡張の有効性に差は見られなかった．しかし，データセットに対して学習データが 20%から 30%の場合，提案手法のデータ拡張による精度の向上が確認できた．

これらの議論をふまえれば，提案手法は，システムが稼働し始めて，機械学習による異常ログ検知の学習に用いるログは多少あるが，充分な量ではない場合に，検知システムの精度を向上させるために特に有効となる可能性があると言える．

## 6 結論

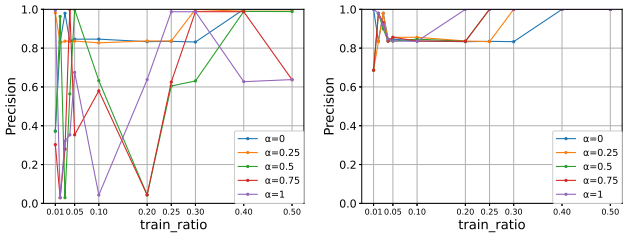
本研究では，正常ログの感情を Negative に変換することで異常ログのデータ拡張を行う手法の提案及び評価を行った．実験ではまず，ベースとなる異常ログ検知アルゴリズム，loglizer を用いて，感情を考慮しないデータ拡張手法と感情を考慮した提案手法の比較実験を行った．その後，異常ログのみの割合を変更する場合と，異常ログと正常ログ両方の割合を変更する場合で，データ拡張の割合を変更する比較実験を行った．これらの比較実験により，提案手法のデータ拡張によって精度が向上することを示した．提案手法を用いることで，異常ログが満足に無い状態であれば，異常ログ検知精度が向上することが分かった．しかし，正常ログと異常ログが共に充分に取得できている場合，データ拡張を行っても精度の向上があまり見られなかった．そのため，提案手法は，サービスの稼働当初で，異常ログが満足に取得できていない状態の時に有効であると考えられる．

今後の課題としては，感情変換を含めた自動的な学習手法に関する検討，サービス開始当初だけでなく，継続的な利用を可能とするための共通部分の過学習抑制に関する検討，より多様なログ形式や異常ログ検知手法を対象とした場合の調査などが挙げられる．これらの調査を進め，汎用的に利用可能な異常ログのデータ拡張手法を実現したい．

## 参考文献

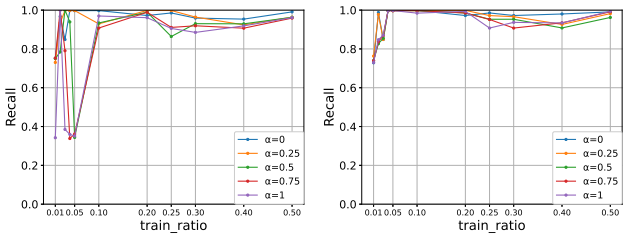
[1] Shilin He, Jieming Zhu, Pinjia He, and Michael R Lyu. Loghub: a large collection of system log datasets towards automated log analytics. *arXiv preprint arXiv:2008.06448*, 2020.

[2] Di Zhang, Dong Dai, Runzhou Han, and Mai Zheng. Sentilog: Anomaly detecting on parallel file systems via log-based sentiment analysis. In *Proceedings of the 13th ACM Workshop on Hot*




**(a)** Simple **(b)** Proposed

**図 8:** 学習データの割合を変更した場合の Precision の比較




**(a)** Simple **(b)** Proposed

**図 9:** 学習データの割合を変更した場合の Recall の比較

*Topics in Storage and File Systems*, pp. 86–93, 2021.

[3] Thorsten Wittkopp, Alexander Acker, Sasho Nedelkoski, Jasmin Bogatinovski, Dominik Scheinert, Wu Fan, and Odej Kao. A2Log: Attentive augmented log anomaly detection. September 2021.

[4] Haixuan Guo, Shuhan Yuan, and Xintao Wu. Logbert: Log anomaly detection via bert. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

[5] Zhuangbin Chen, Jinyang Liu, Wenwei Gu, Yuxin Su, and Michael R Lyu. Experience report: deep learning-based system log analysis for anomaly detection. *arXiv preprint arXiv:2107.05908*, 2021.

[6] Takumi Yamamoto, Aiko Iwasaki, Hajime Kobayashi, Kiyoto Kawauchi, and Ayako Yoshimura. A study on enhancing anomaly detection technology with synthetic-log generation. In *International Conference on Advanced Information Networking and Applications*, pp. 528–538. Springer, 2022.

[7] Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu. Experience report: System log analysis for anomaly detection. In *27th IEEE International Symposium on Software Reliability Engineering, ISSRE 2016, Ottawa, ON, Canada, October 23-27, 2016*, pp. 207–218. IEEE Computer Society, 2016.

[8] Peter Bodik, Moises Goldszmidt, Armando Fox, Dawn B Woodard, and Hans Andersen. Fingerprinting the datacenter: automated classification of performance crises. In *Proceedings of the 5th European conference on Computer systems*, pp. 111–124, 2010.

[9] Mike Chen, Alice X Zheng, Jim Lloyd, Michael I Jordan, and Eric Brewer. Failure diagnosis using decision trees. In *International Conference on Autonomic Computing, 2004. Proceedings.*, pp. 36–43. IEEE, 2004.

[10] Yinglung Liang, Yanyong Zhang, Hui Xiong, and Ramendra Sahoo. Failure prediction in ibm bluegene/l event logs. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pp. 583–588. IEEE, 2007.

[11] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. Log clustering based problem identification for online service systems. In *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, pp. 102–111. IEEE, 2016.

[12] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pp. 117–132, 2009.

[13] Jian-Guang Lou, Qiang Fu, Shenqi Yang, Ye Xu, and Jiang Li. Mining invariants from console logs for system problem detection. In *2010 USENIX Annual Technical Conference (USENIX ATC 10)*, 2010.

[14] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R Lyu. Tools and benchmarks for automated log parsing. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 121–130. IEEE, 2019.

[15] Pinjia He, Jieming Zhu, Shilin He, Jian Li, and Michael R Lyu. An evaluation study on log parsing and its use in log mining. In *2016 46th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, pp. 654–661. IEEE, 2016.

[16] Adam Oliner and Jon Stearley. What supercomputers say: A study of five system logs. In *37th annual IEEE/IFIP international conference on dependable systems and networks (DSN'07)*, pp. 575–584. IEEE, 2007.