# 深層マルチタスクとシングルタスク学習に基づく OSS 信頼性評価法に関する比較検討

山口大学大学院・創成科学研究科 田村 慶信（Yoshinobu Tamura）†
†Graduate School of Sciences and Technology for Innovation, Yamaguchi University
鳥取大学・名誉教授 山田 茂（Shigeru Yamada）††
††Emeritus Professor, Tottori University

## 1　はじめに

ソフトウェアの信頼性は，従来から数多くのソフトウェア信頼度成長モデルと呼ばれる確率モデルをソフトウェアの総合テスト工程に適用することにより評価されてきた [1,2]．近年では，多くのオープンソースソフトウェア（Open Source Software，以下 OSS と略す）が，企業組織の下で開発された商用ソフトウェアにも取り込まれている．このような OSS は，フォールト情報をバグトラッキングシステム上で管理されるケースが多く，フォールト発見数だけではなく，多くのフォールト要因に関する情報がデータベース上に記録されている．バグトラッキングシステム上に登録されたフォールトビッグデータを活用することができれば，従来よりも精緻なソフトウェア信頼性評価が可能となる．

ビッグデータから新たな知見を得る手法として，深層学習が知られている．深層学習は，様々な分野で利活用されており，プログラミング領域においても，TensorFlow などの多くのパッケージが提供されている．深層学習の学習過程においては，ハイパーパラメータの設定が非常に重要なプロセスであり，経験則に基づいて決定されることが多い．

我々の研究グループでは，深層学習に基づく OSS 信頼性評価法に関する研究成果をいくつか提案してきた [3,4]．多くの分野において OSS が多用されており，近年では，ネットワーク経由で利用されるクラウドコンピューティングやエッジコンピューティング [5] の基盤ソフトウェアにも OSS が供給されるなど，ネットワークインフラ基盤ソフトウェアとしても OSS が利活用されている．

本論文では，深層マルチタスク学習と深層シングルタスク学習に基づく OSS 信頼性評価法について議論する．特に，深層マルチタスク学習は，深層シングルタスク学習の欠点を補うことが知られている [6–8]．それぞれの手法を OSS 信頼性評価のために適用する．さらに，具体的な数値例を示すことにより，各手法の OSS 信頼性評価に対する適用可能性について議論する．

## 2　深層シングルタスク学習と深層マルチタスク学習

一般的な深層シングルタスク学習の特徴としては，目的変数は 1 種類であり，過学習しやすいケースがある．また，変数選択としては 1 つで良いので目的変数を決めやすい．一方で，特徴量を捕獲し難いという特徴がある．

深層マルチタスク学習においては，目的変数が複数あり，過学習を抑制できる特徴がある．また，変数選択としては，複数あるため効果的な変数を選択することが難しい．さらに，一般的に特徴量を捕獲する確率が高くなることが知られている．

このように，深層シングルタスク学習と深層マルチタスク学習には，双方に利点と欠点があるが，入力データが多い場合には，過学習を抑制でき，特徴量を捕獲しやすいという特徴をもつマルチタスク学

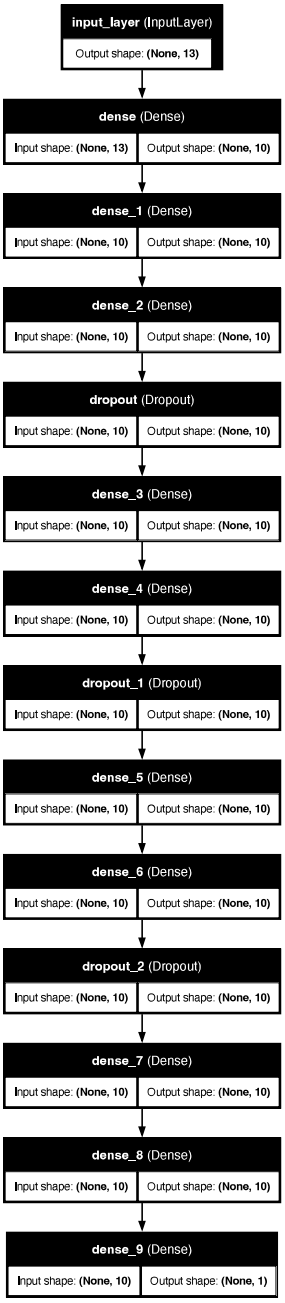


図 1： 深層シングルタスク学習のモデル構造.

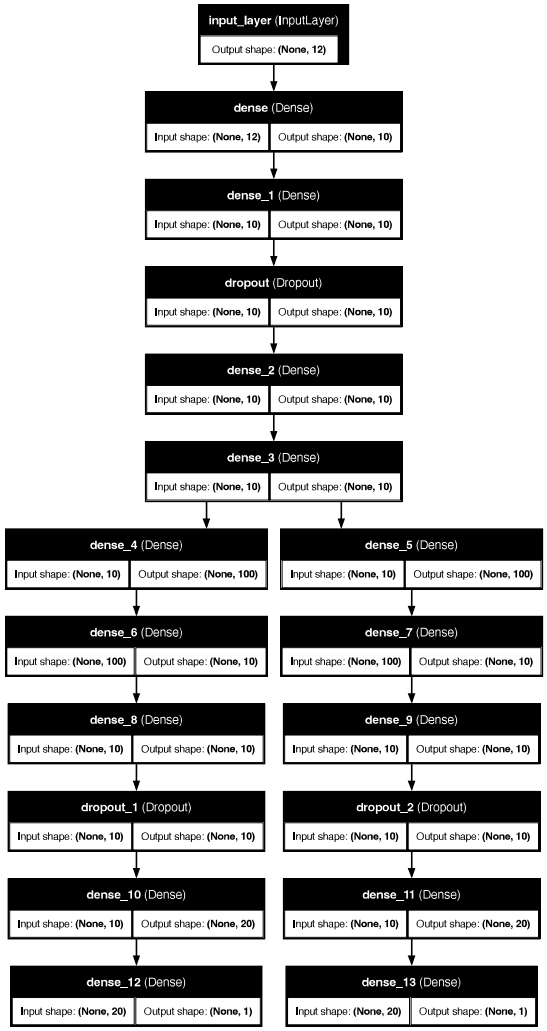


図 2： 深層マルチタスク学習のモデル構造.

習を適用することが可能である．本論文において比較する深層シングルタスク学習と深層マルチタスク学習のモデル構造を図 1 および図 2 に示す.

本論文では，深層学習の適用環境としてプログラミング言語 Python における TensorFlow を用いる．このとき，OSS フォールトビッグデータに対する深層マルチタスク学習の適用可能性について議論する．特に，深層マルチタスク学習による予測値の変化について，実際のフォールトビッグデータを適用した数値例に基づき推定することで，OSS 信頼性評価のために利用できるテストデータの範囲や深層マルチタスク学習の適用可能性について考察する.

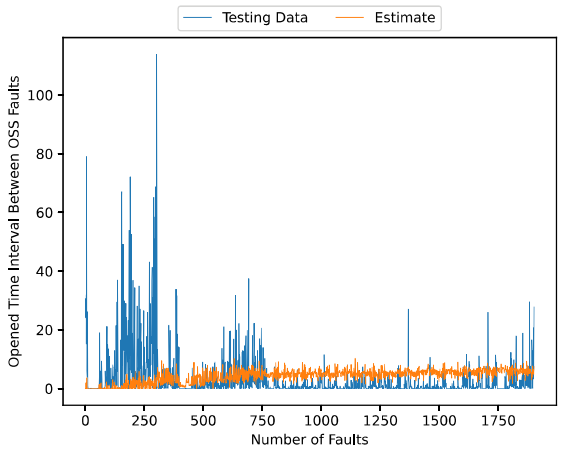


図 3： シングルタスクモデルのテストデータが 10%の場合における推定されたフォールト発見時間間隔.

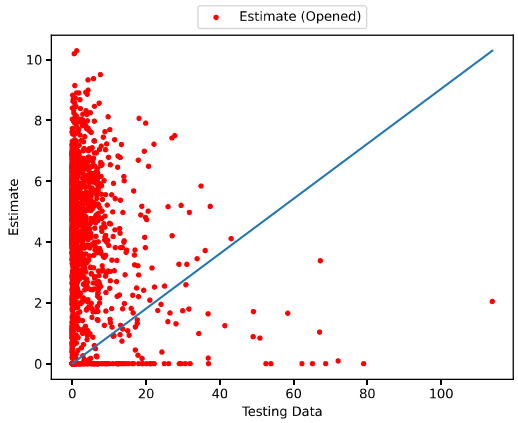


図 4： シングルタスクモデルのテストデータが 10%の場合における推定されたフォールト発見時間間隔の散布図.

## 3 目的変数

本論文では，深層学習の目的変数に対して，以下に示すようなフォールト発見時間間隔およびフォールト修正時間間隔を適用する．

$$\alpha(i) = o_i - o_{i-1}, \tag{1}$$

$$\beta(i) = m_i - m_{i-1}. \tag{2}$$

ここで，$o_i$ は，$i$ 番目のフォールトにおけるフォールト発見時刻を表す．このとき，$o_i - o_{i-1}$ は，$i$ 番目と $i-1$ 番目の間におけるフォールト発見時間間隔を表す．すなわち，$\alpha(i)$ の値が大きくなるにつれて，OSS 信頼度が向上することを意味する．同様に，$m_i$ は，$i$ 番目のフォールトにおけるフォールト修正時刻を表す．このとき，$m_i - m_{i-1}$ は，$i$ 番目と $i-1$ 番目の間におけるフォールト修正時間間隔を表す．すなわち，$\beta(i)$ の値が大きくなるにつれて，フォールト修正難易度が向上することを意味する．

本論文では，深層シングルタスク学習と深層マルチタスク学習における OSS フォールトビッグデータに対する学習と予測特性について議論するために，$\alpha(i)$ を深層シングルタスク学習の目的変数とし，$\alpha(i)$ および $\beta(i)$ を深層マルチタスク学習の目的変数とする．

## 4 数値例

*OpenStack* [9] から得られたバグトラッキングシステムのフォールトビッグデータに基づく数値例を示す．図 3 および図 4 は，シングルタスクモデルのテストデータが 10%の場合における推定されたフォールト発見時間間隔およびその散布図を示す．また，シングルタスクモデルの場合において，テストデータが 20%，30%，40%，および 50%の場合における推定されたフォールト発見時間間隔およびその散布図を図 5~12 にそれぞれ示す．

同様に，図 13~ 図 16 は，マルチタスクモデルのテストデータが 10%の場合における推定されたフォールト発見時間間隔およびフォールト修正時間間隔と，それらの散布図を示す．さらに，マルチタスクモデルの場合においてテストデータが 20%，30%，40%，および 50%の場合における推定されたフォールト発見時間間隔およびフォールト修正時間間隔と，それらの散布図を図 17~32 にそれぞれ示す．

図 3~32 から，深層シングルタスク学習の場合には，テストデータが 10%の場合が最も予測結果が安定している様子が確認できる．一方で，テストデータ数が大きくなるほど長期的な予測結果は安定してい

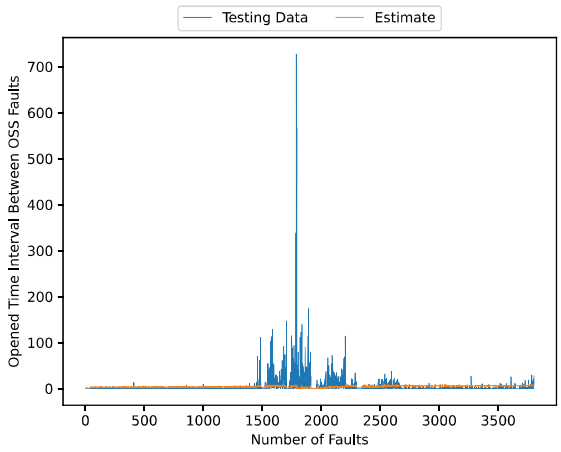


図 5： シングルタスクモデルのテストデータが 20%の場合における推定されたフォールト発見時間間隔.

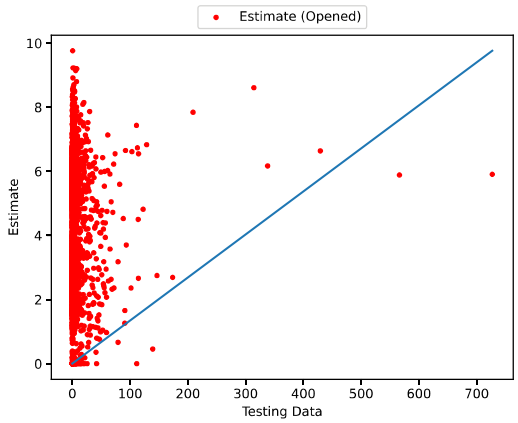


図 6： シングルタスクモデルのテストデータが 20%の場合における推定されたフォールト発見時間間隔の散布図.

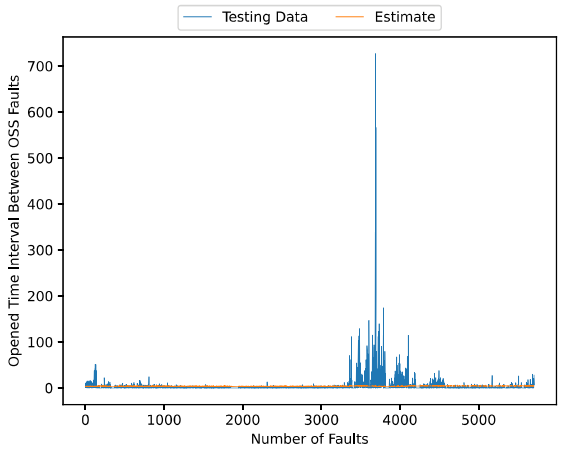


図 7： シングルタスクモデルのテストデータが 30%の場合における推定されたフォールト発見時間間隔.

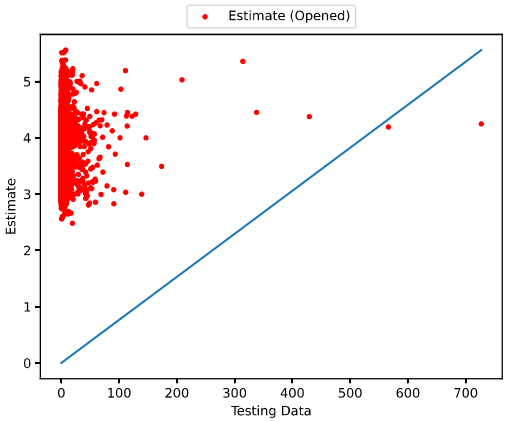


図 8： シングルタスクモデルのテストデータが 30%の場合における推定されたフォールト発見時間間隔の散布図.

るものの，突発的な変化には対応できない様子が確認できる．一方，深層マルチタスク学習の場合には，テストデータの量によって差はあるものの，データの変化の傾向はある程度追従していることが分かる．

さらに，深層シングルタスク学習と深層マルチタスク学習の場合における平均二乗誤差（Mean Square Errors，以下 MSE と略す）の推定結果を表 1 および表 2 に示す．表 1 および表 2 から，深層シングルタスクモデルの場合においては，テストデータが 10%の場合の推定結果が良いことが分かった．一方で，深層マルチタスクモデルの場合においては，両方の目的変数の推定結果が比較的に良いケースはテストデータが 20%の場合であった．

## 5 おわりに

本論文では，深層シングルタスク学習と深層マルチタスク学習に基づく OSS 信頼性評価法における適用可能性とその比較結果について議論した．深層学習においては，ハイパーパラメータの調整に時間と

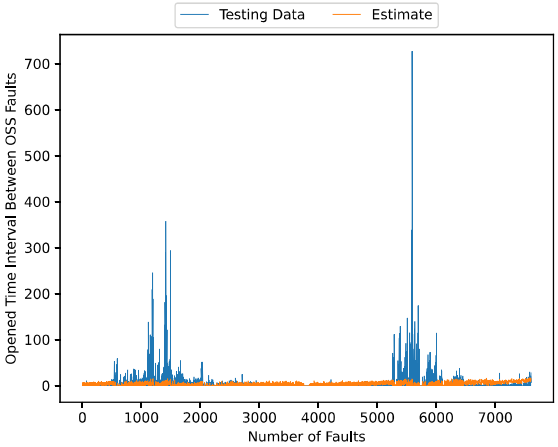


図 9： シングルタスクモデルのテストデータが 40%の場合における推定されたフォールト発見時間間隔.

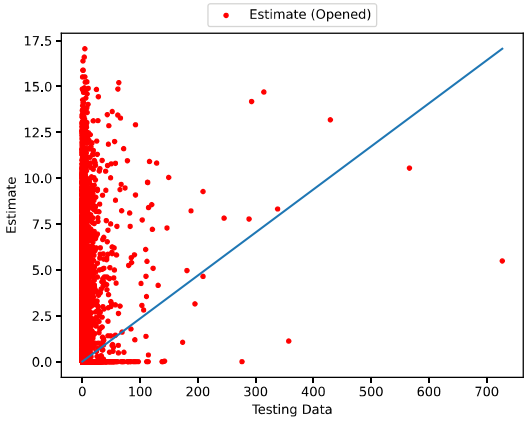


図 10： シングルタスクモデルのテストデータが 40%の場合における推定されたフォールト発見時間間隔の散布図.

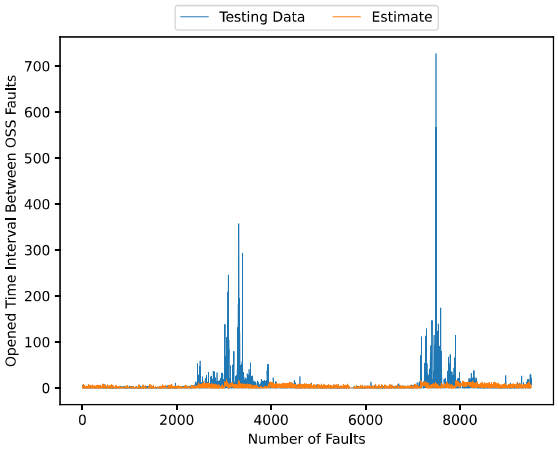


図 11： シングルタスクモデルのテストデータが 50%の場合における推定されたフォールト発見時間間隔.

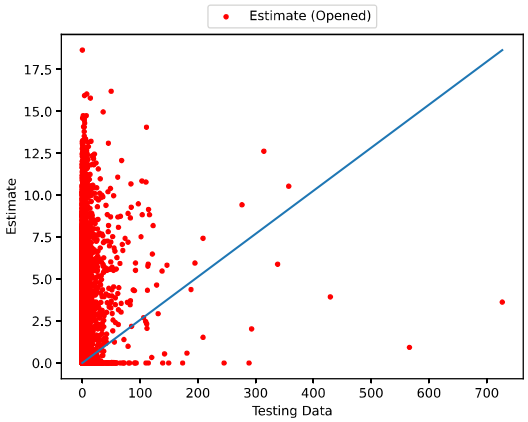


図 12： シングルタスクモデルのテストデータが 50%の場合における推定されたフォールト発見時間間隔の散布図.

労力がかかることが知られている．特に，OSS のフォールトビッグデータを深層シングルタスク学習と深層マルチタスク学習へ適用する際において，その特徴を把握することができれば，OSS 管理者やユーザの信頼性評価が容易になる．

深層シングルタスク学習と深層マルチタスク学習に対して，テストデータを 10%ごとに変化させた場合における 5 つの推定結果について数値例を示すとともに，それらの比較結果について議論した．これらの結果から，OSS のバージョンアップが大きく関係していると思われる．特に，OSS には，バグフィックスバージョン，マイナーバージョンアップ，およびメジャーバージョンアップのように，数種類のバージョンアップが定期的または不規則に実施されている．OSS のフォールト発見事象について考えた場合，こうしたバージョンアップに強く依存するため，今後はバージョンアップに応じた予測期間を設定して評価する必要があると考える．

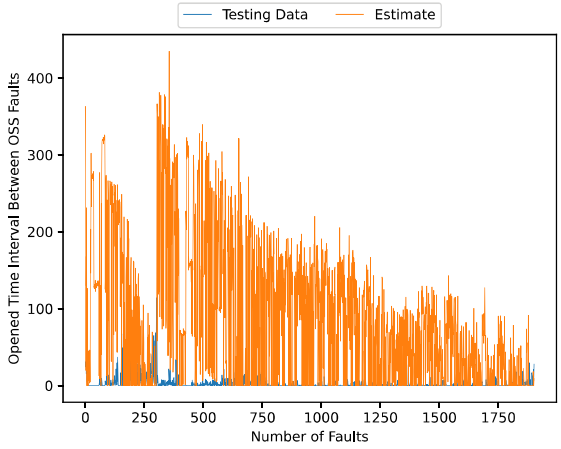


図 13： マルチタスクモデルのテストデータが 10% の場合における推定されたフォールト発見時間間隔.

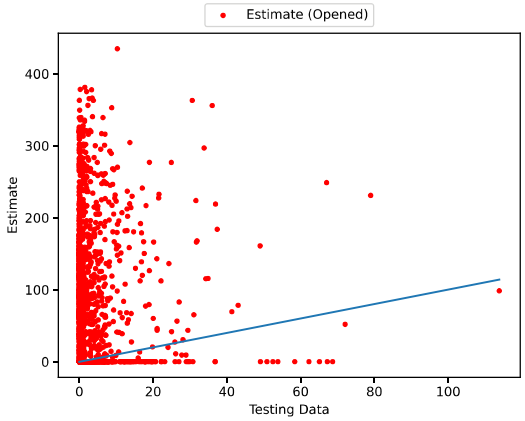


図 14： マルチタスクモデルのテストデータが 10% の場合における推定されたフォールト発見時間間隔の散布図.

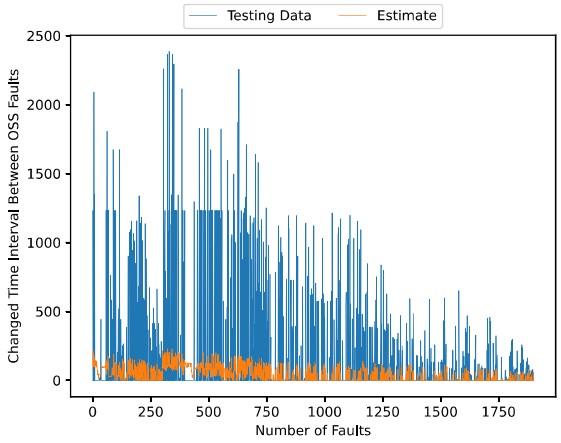


図 15： マルチタスクモデルのテストデータが 10% の場合における推定されたフォールト修正時間間隔.

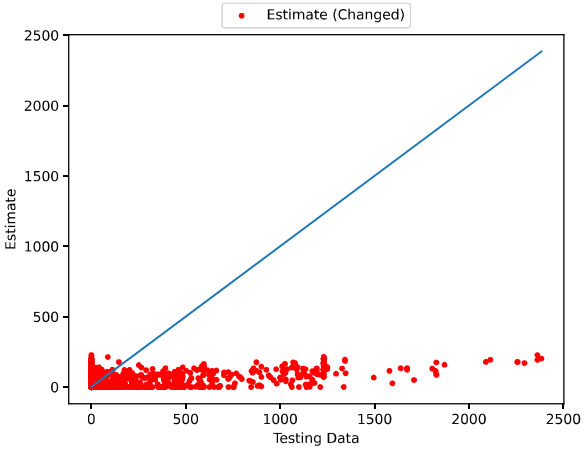


図 16： マルチタスクモデルのテストデータが 10% の場合における推定されたフォールト修正時間間隔の散布図.

## 謝辞

本研究の一部は，JSPS 科研費基盤研究（C）（課題番号 23K11066）の援助を受けたことを付記する．

## 参考文献

[1] P.K. Kapur, H. Pham, A. Gupta, and P.C. Jha, *Software Reliability Assessment with OR Applications*, Springer–Verlag, London, 2011.

[2] S. Yamada, *Software Reliability Modeling: Fundamentals and Applications*, Springer–Verlag, Tokyo/Heidelberg, 2014.

[3] S. Yamada and Y. Tamura, “OSS reliability measurement and assessment,” Springer Series in Reliability Engineering, Springer, 2016.

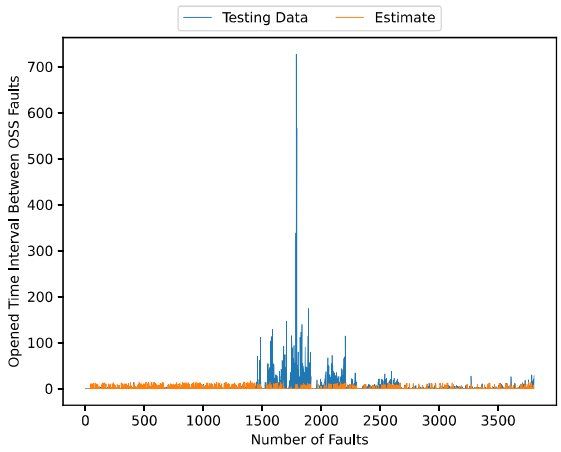


図 17： マルチタスクモデルのテストデータが 20% の場合における推定されたフォールト発見時間間隔.

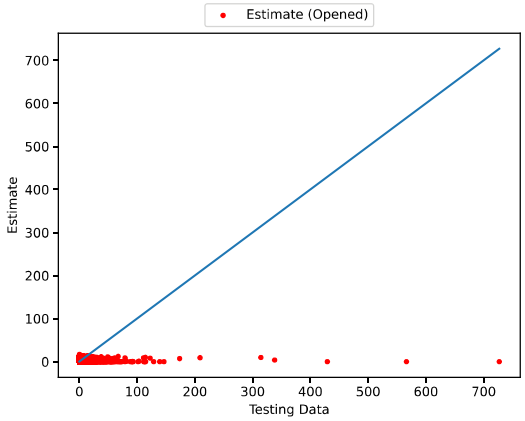


図 18： マルチタスクモデルのテストデータが 20% の場合における推定されたフォールト発見時間間隔の散布図.

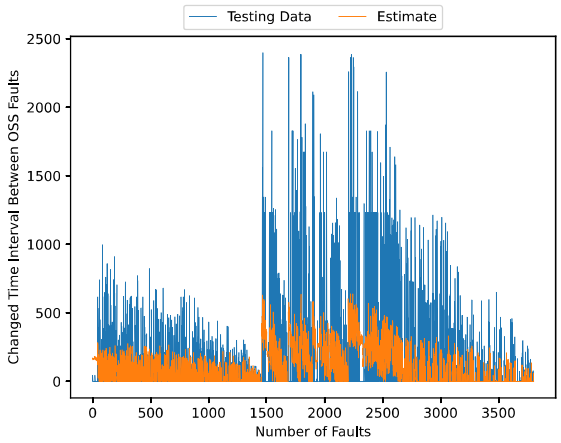


図 19： マルチタスクモデルのテストデータが 20% の場合における推定されたフォールト修正時間間隔.

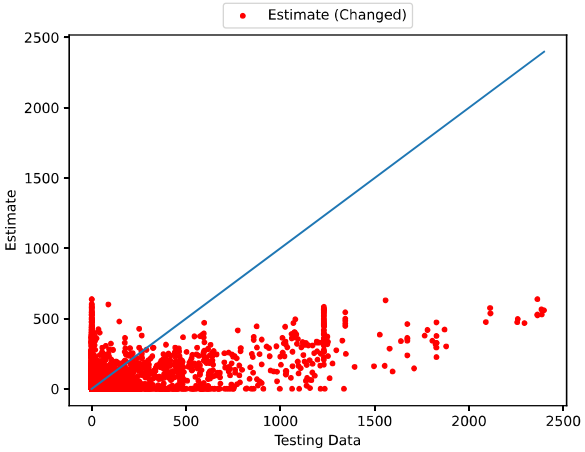


図 20： マルチタスクモデルのテストデータが 20% の場合における推定されたフォールト修正時間間隔の散布図.

[4] Y. Tamura and S. Yamada, “Applied OSS Reliability Assessment Modeling, AI and Tools,” Springer Series in Reliability Engineering, Springer, 2024.

[5] R. Pandey, et al., *Artificial Intelligence and Machine Learning for EDGE Computing*, Elsevier, 2022.

[6] X. Gibert, V.M. Patel and R. Chellappa, “Deep multitask learning for railway track inspection,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 1, pp. 153-164, Jan. 2017.

[7] R. Ranjan, V.M. Patel and R. Chellappa, “Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition,“ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 1, pp. 121-135, 1 Jan. 2019.

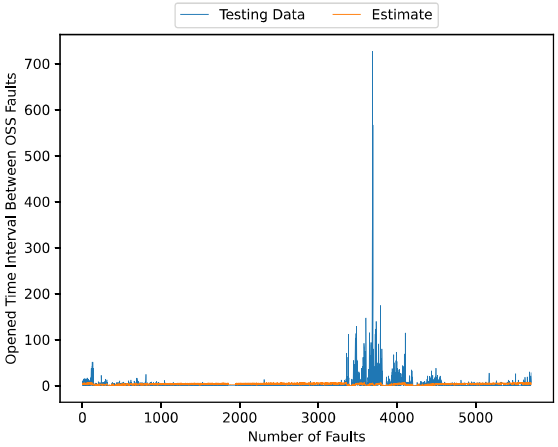


図 21： マルチタスクモデルのテストデータが 30% の場合における推定されたフォールト発見時間間隔.

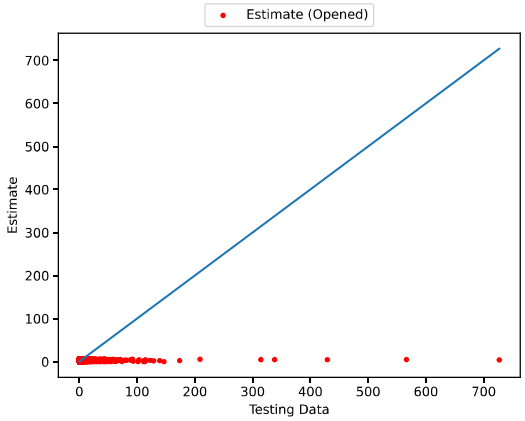


図 22： マルチタスクモデルのテストデータが 30% の場合における推定されたフォールト発見時間間隔の散布図.

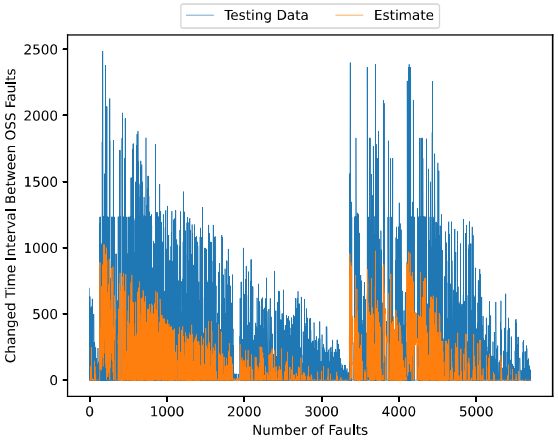


図 23： マルチタスクモデルのテストデータが 30% の場合における推定されたフォールト修正時間間隔.

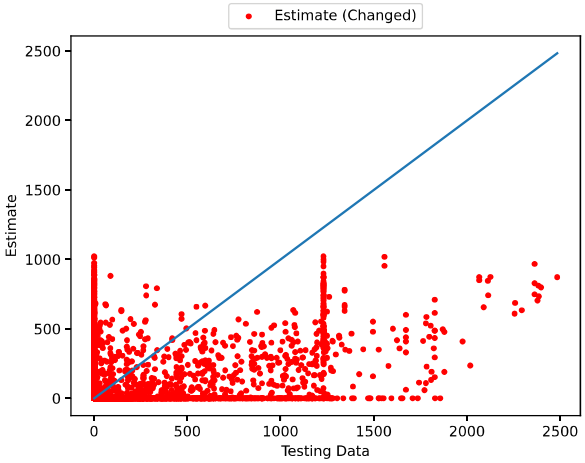


図 24： マルチタスクモデルのテストデータが 30% の場合における推定されたフォールト修正時間間隔の散布図.

[8] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai and L. Van Gool, “Multi-task learning for dense prediction tasks: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3614-3633, 1 July 2022.

[9] The OpenStack project, OpenStack, http://www.openstack.org/

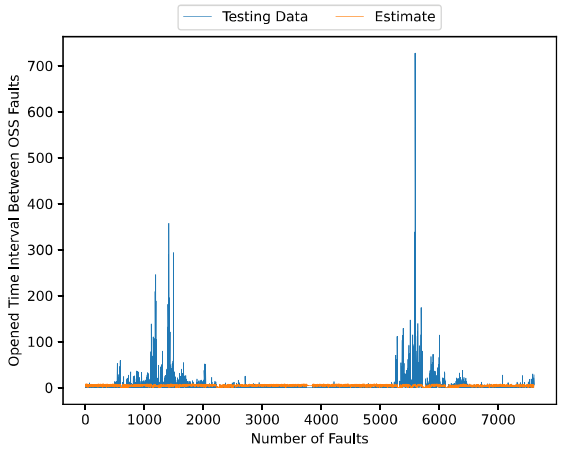


図 25： マルチタスクモデルのテストデータが 40% の場合における推定されたフォールト発見時間間隔.

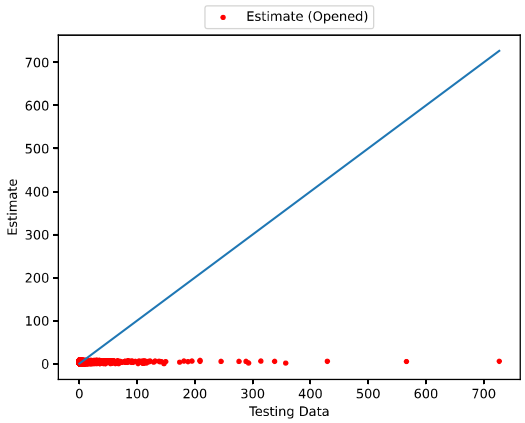


図 26： マルチタスクモデルのテストデータが 40% の場合における推定されたフォールト発見時間間隔の散布図.

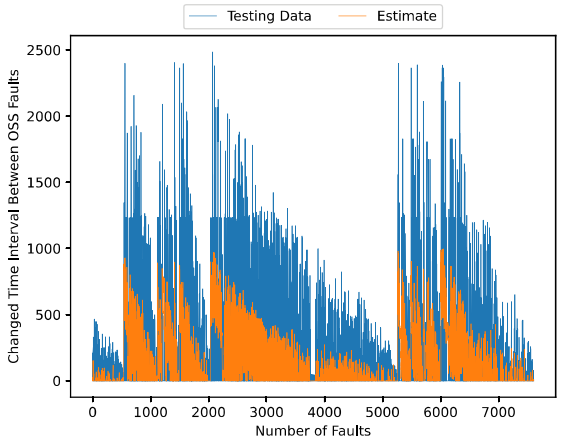


図 27： マルチタスクモデルのテストデータが 40% の場合における推定されたフォールト修正時間間隔.

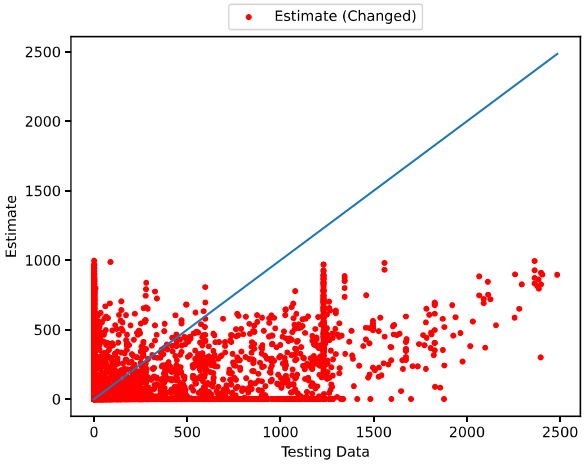


図 28： マルチタスクモデルのテストデータが 40% の場合における推定されたフォールト修正時間間隔の散布図.

表 1： シングルタスクモデルの MSE に基づく比較結果.

| テストデータ | MSE |
|---|---|
| 10% | 72.92490 |
| 20% | 468.87881 |
| 30% | 317.47913 |
| 40% | 377.00720 |
| 50% | 300.96091 |

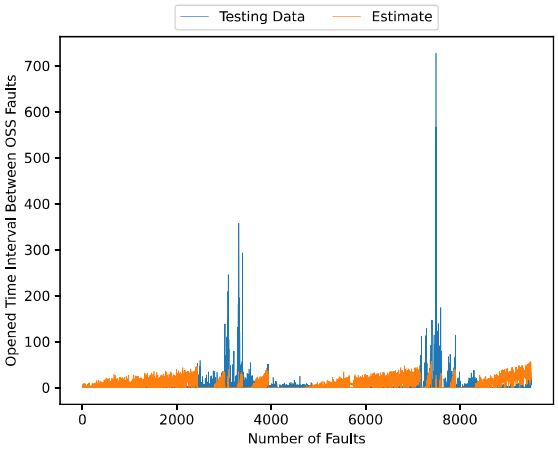


図 29： マルチタスクモデルのテストデータが 50% の場合における推定されたフォールト発見時間間隔.

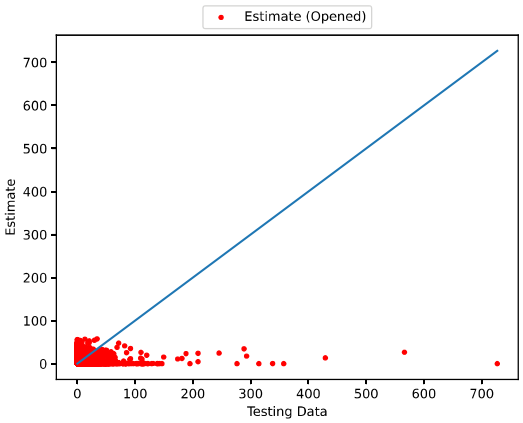


図 30： マルチタスクモデルのテストデータが 50% の場合における推定されたフォールト発見時間間隔の散布図.

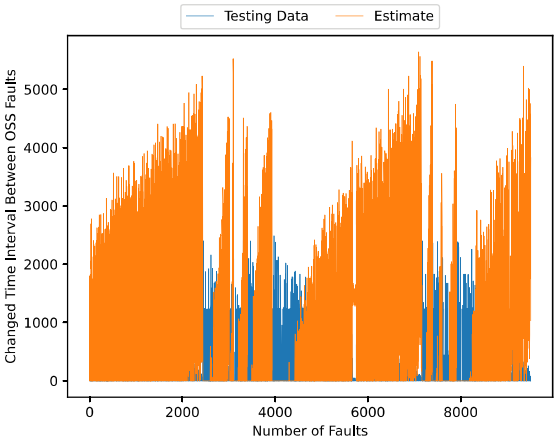


図 31： マルチタスクモデルのテストデータが 50% の場合における推定されたフォールト修正時間間隔.

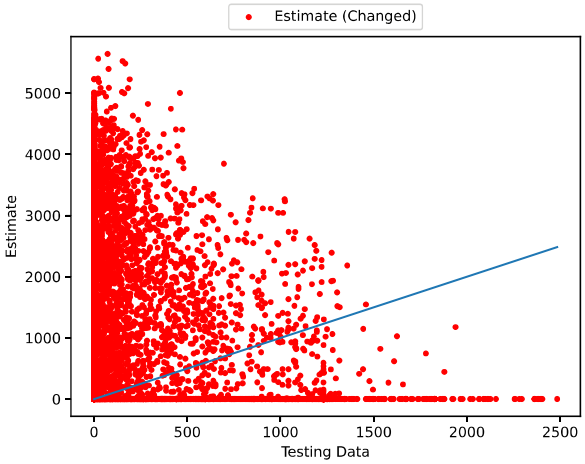


図 32： マルチタスクモデルのテストデータが 50% の場合における推定されたフォールト修正時間間隔の散布図.

表 2： マルチタスクモデルにおける MSE に基づく比較結果.

| テストデータ | 1st MSE | 2nd MSE |
|---|---|---|
| 10% | 15115.284 | 123817.008 |
| 20% | 484.835 | 88175.109 |
| 30% | 321.620 | 122998.297 |
| 40% | 371.631 | 129761.25 |
| 50% | 457.777 | 2585779.75 |