深層学習に基づくOSS信頼性評価法における ハイパーパラメータの最適化に関する一考察

山口大学大学院・創成科学研究科 田村 慶信（Yoshinobu Tamura）†
山口大学大学院・創成科学研究科 宮本 翔一郎（Shoichiro Miyamoto）†
山口大学大学院・創成科学研究科 周 蕾（Lei Zhou）†
†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

機械学習は，様々な分野において多用されており，近年では深層学習のような人工知能が画像認識などの分野を中心として発展している．特に，深層学習分野は目覚ましい発展を遂げている．しかしながら，深層学習を扱う際においては，いわゆるハイパーパラメータの調整と設定に多くの時間と労力を必要し，職人技的な能力が求められることも知られている．深層学習の成功には，データの前処理と，こうしたハイパーパラメータの調整が大きな鍵を握る [1–3].

従来から，数多くのソフトウェア信頼度成長モデルが精力的に開発されてきた [4,5]．一般的には，確率モデルに基づく動的ソフトウェア信頼性評価法が企業組織などにおいて多用されてきた．他にも，機械学習に基づくソフトウェア信頼性評価法も提案されている．我々の研究グループにおいても，深層学習に基づくオープンソースソフトウェア（Open Source Software，以下 OSS と略す）信頼性評価法に関する研究をいくつか提案してきた．多くの分野において OSS が多用されており，低コスト，短納期，および標準化といった観点から，小規模な OSS コンポーネントも多数組み込まれるようになっている．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，深層学習を適用する際に重要となる，ハイパーパラメータの設定方法について着目する．また，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の基盤 OSS のバグトラッキングシステムから取得されたフォールトビッグデータを適用した数値例を示す．さらに，ハイパーパラメータの設定状況に応じた OSS 信頼性評価結果を示すとともに，ハイパーパラメータの最適化に関して議論する．

2　深層学習に基づく OSS フォールト発見時間間隔の推定

本論文では，ディープラーニングの目的変数に対して，フォールト発見時間を適用する．ソフトウェアフォールトデータが，多くのソフトウェア信頼度成長モデルにおいて利用されてきた．以下のようなフォールト発見時間間隔を考える．

$$F(i) = o_i - o_{i-1}. \tag{1}$$

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，$(o_i - o_i)$ 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，$F(i)$ 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．

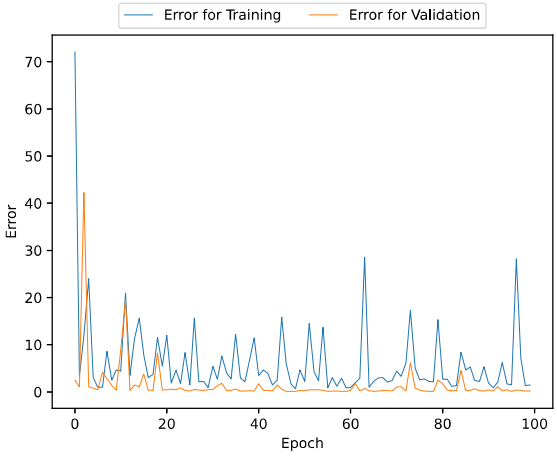

図 1： テストデータが 10%の場合における訓練データと検証データに対する推定された誤差.

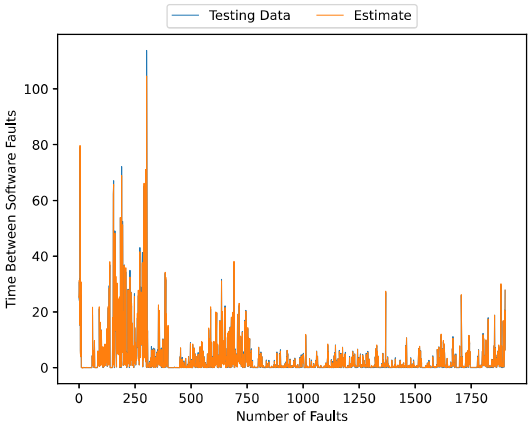

図 2： テストデータが 10%の場合における推定されたフォールト発見時間間隔.

3　深層学習におけるハイパーパラメータ

本論文では，プログラミング言語 Python における TensorFlow を用いた場合におけるハイパーパラメータの設定と結果について考察する．直列結合のディープフィードフォワードネットワークを取り上げる．このとき，各パラメータは以下のように設定した．

Dense: 100,

Activation: relu,

Dropout: 0.001∼0.01,

Layer: 7.

深層学習の層数は 7 つのレイヤーを有する．この設定を基本的な構成とした．我々の研究グループでは，過去に深層学習に基づく OSS 信頼性評価法を提案してきた [6–8]．将来的には，アプリケーションソフトウェアとして実装・開発し，様々な環境に対して柔軟かつ実用的に適用できるように，初期値の設定をある程度一般化したい．このためには，最小構成でハイパーパラメータを設定し，あらゆるバグトラッキングシステム上に登録されたデータに対して適用可能なハイパーパラメータの設定状態を模索する必要がある．

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 1 および図 2 は，テストデータが 10%の場合における訓練データと検証データに対する推定された誤差，および推定されたフォールト発見時間間隔を示す．また，図 3 および図 4 は，テストデータが 30%の場合における訓練データと検証データに対する推定された誤差，および推定されたフォールト発見時間間隔を示す．同様に，図 5 および図 6 は，テストデータが 50%の場合における訓練データと検証データに対する推定された誤差，および推定されたフォールト発見時間間隔を示す．

さらに，図 7 および図 8 は，テストデータが 50%においてハイパーパラメータを更新した場合における訓練データと検証データに対する推定された誤差と，推定されたフォールト発見時間間隔を示す．

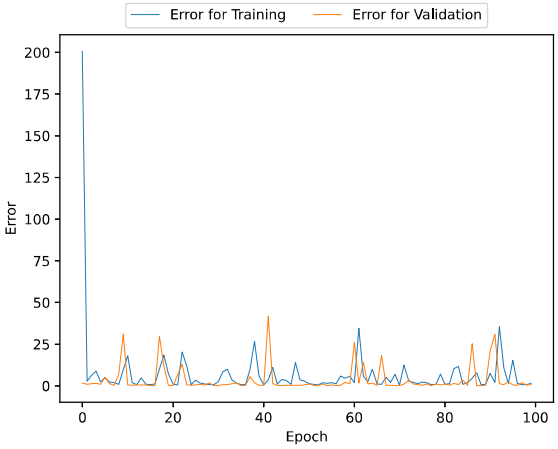

図 3： テストデータが 30%の場合における訓練データと検証データに対する推定された誤差.

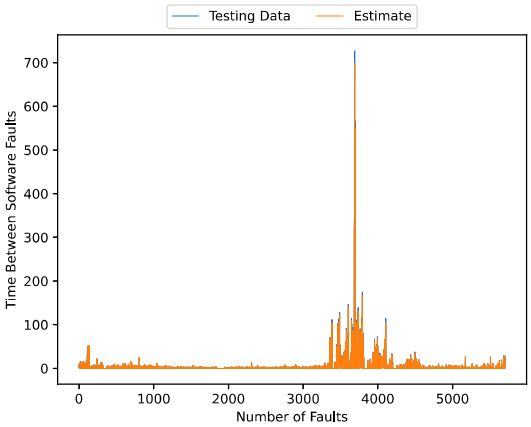

図 4： テストデータが 30%の場合における推定されたフォールト発見時間間隔.

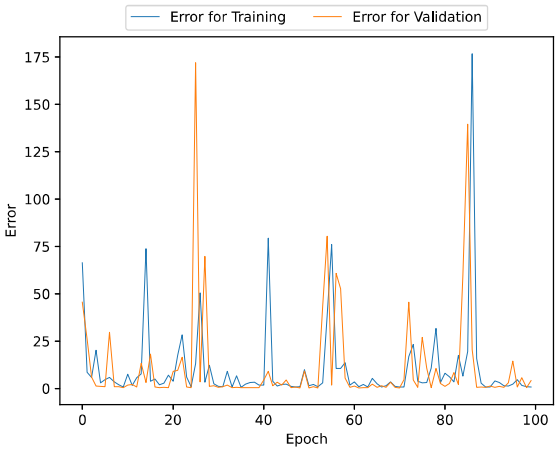

図 5： テストデータが 50%の場合における訓練データと検証データに対する推定された誤差.

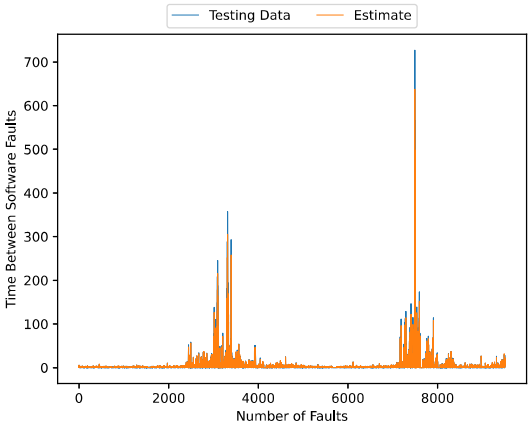

図 6： テストデータが 50%の場合における推定されたフォールト発見時間間隔.

また，図 9 および図 10 は，テストデータが 50%においてハイパーパラメータを更新した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と，推定された累積フォールト発見時間間隔を示す．同様に，図 11 および図 12 は，テストデータが 10%における推定されたフォールト発見時間間隔の散布図と，推定された累積フォールト発見時間間隔を示す．さらに，図 13 および図 14 は，テストデータが 30%における推定されたフォールト発見時間間隔の散布図と，推定された累積フォールト発見時間間隔を示す．最後に，図 15 および図 16 は，テストデータが 50%における推定されたフォールト発見時間間隔の散布図と，推定された累積フォールト発見時間間隔を示す．

図 1～ 図 16 から，テストデータが大きくなるにつれ，予測精度が低下している様子が確認できる．特に，テストデータが 50%においてハイパーパラメータを更新した場合と，更新しなかった場合について比較すると，ハイパーパラメータを更新した場合の方が予測精度が高くなっていることが分かった．このことから，データ量に応じたハイパーパラメータの更新は有効であることが分かる．

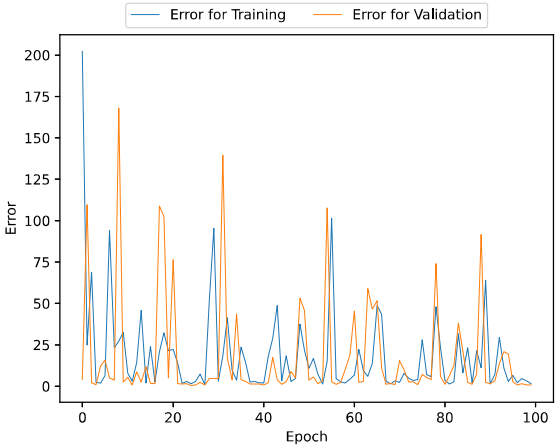

図 7： テストデータが 50%においてハイパーパラメータを更新した場合における訓練データと検証データに対する推定された誤差.

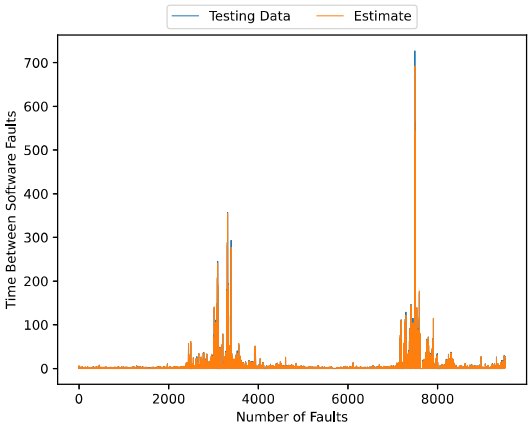

図 8： テストデータが 50%においてハイパーパラメータを更新した場合における推定されたフォールト発見時間間隔.

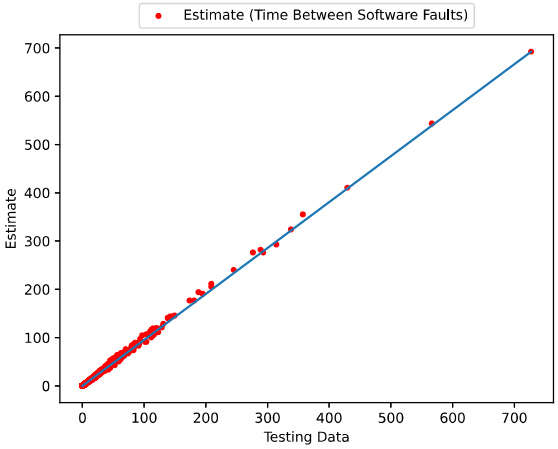

図 9： テストデータが 50%においてハイパーパラメータを更新した場合における推定されたフォールト発見時間間隔の散布図.

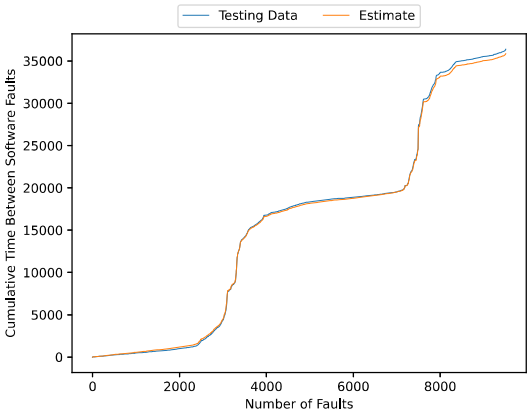

図 10： テストデータが 50%においてハイパーパラメータを更新した場合における推定された累積フォールト発見時間間隔.

5 おわりに

本論文では，深層学習に基づく OSS 信頼性評価法におけるハイパーパラメータの最適化について議論した．特に，バグトラッキングシステム上に登録されているフォールトビッグデータを深層学習で分析する際におけるハイパーパラメータの最適な値について，深層学習のレイヤー数などに基づき調査した．

深層学習を扱う際においては，データ分析者のスキルや習熟度に依存して，データの前処理やハイパーパラメータの設定に要する時間や労力が変化する．こうした熟練技のような工程は，深層学習における処理システムを一般化する妨げとなる．我々の研究グループでは，今後，深層学習に基づく OSS 信頼性評価法をアプリケーションソフトウェアとしてツール化することを計画しており，最適なハイパーパラメータの決定法は，それを実現するために必要となる．

ハイパーパラメータは，データの特性によって変化するものである．OSS の場合は，バグフィックスアップデート，マイナーバージョンアップ，メジャーバージョンアップ，さらには，OSS の種類，アプ

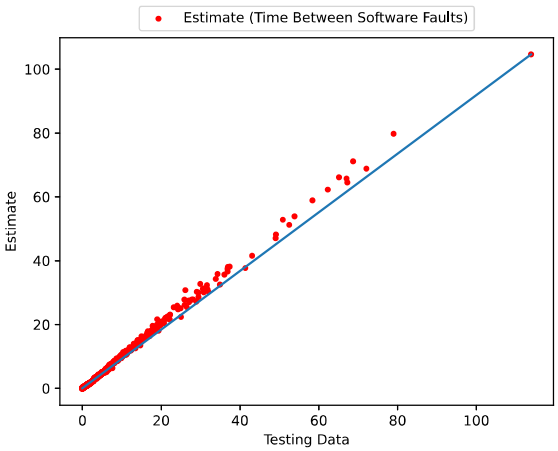

図 11： テストデータが 10%の場合における推定されたフォールト発見時間間隔の散布図.

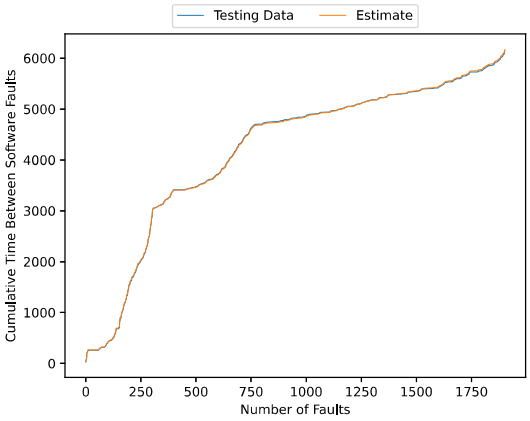

図 12： テストデータが 10%の場合における推定された累積フォールト発見時間間隔.

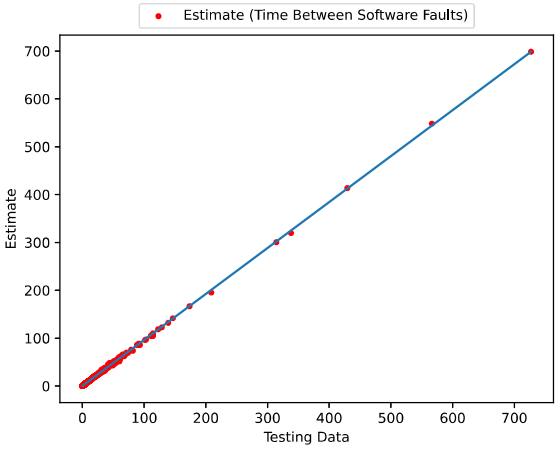

図 13： テストデータが 30%の場合における推定されたフォールト発見時間間隔の散布図.

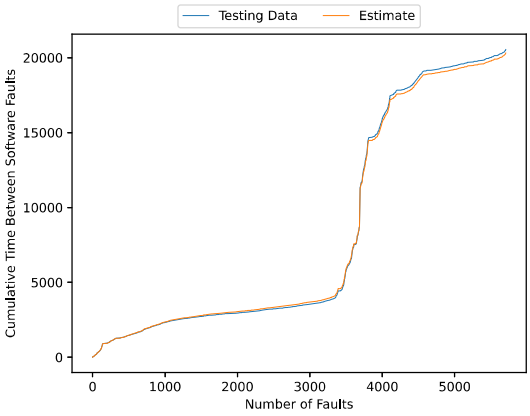

図 14： テストデータが 30%の場合における推定された累積フォールト発見時間間隔.

リケーションソフトウェア，サーバソフトウェア，クラウドソフトウェアなどの OSS の利用形態など，多くの特性を有しているため，これらの特性に応じて最適なハイパーパラメータが存在することも考えられる．今後は，このような OSS の特性に応じた最適なハイパーパラメータについて模索したい．

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] N. Karunanithi, Y. K. Malaiya, and D. Whitley, “Prediction of software reliability using neural networks,” Proc. 2nd Int. Symp. Soft. Reli. Eng., pp. 124-130. IEEE Computer Society Press, 1991.

[2] N. Karunanithi, D. Whitley, and Y. K. Malaiya, “Using neural networks in reliability prediction,” IEEE Software, vol. 9, pp. 53-59, 1992.

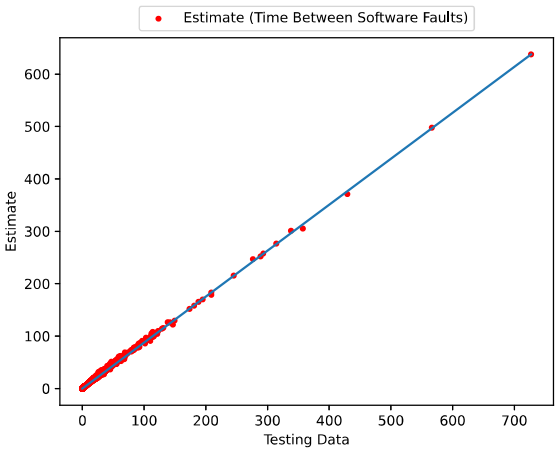

図 15： テストデータが 50%の場合における推定されたフォールト発見時間間隔の散布図.

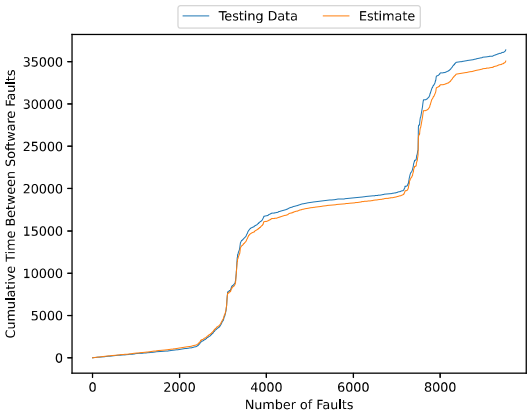

図 16： テストデータが 50%の場合における推定された累積フォールト発見時間間隔.

[3] N. Karunanithi and Y. K. Malaiya, “Neural networks for software reliability engineering,” in Handbook of Software Reliability Engineering, ed. M. R. Lyu, pp. 699-728, McGraw-Hill, New York, 1996.

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

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

[6] Y. Tamura, S. Miyamoto, L. Zhou, A. Anand, P.K. Kapur, and S. Yamada, “OSS reliability assessment method based on deep learning and independent Wiener data preprocessing,” Journal International Journal of System Assurance Engineering and Management, Vol. ahead-of-print No. ahead-of-print, Springer, https://doi.org/10.1007/s13198-024-02288-w, pp. 1-9, February 2024.

[7] Y. Tamura, S. Miyamoto, L. Zhou, and S. Yamada, “OSS sustainability assessment based on the deep learning considering effort Wiener process data,” International Journal of Reliability, Quality and Safety Engineering, Vol. 31, No. 1, World Scientific, https:///dx.doi.org/10.1142/S0218539323500328, pp. 2350032-1–2350032-15, 2024.

[8] Y. Tamura and S. Yamada, “A method of reliability assessment based on fine tuning deep learning model for open source software in edge computing,” International Journal of Reliability, Quality and Safety Engineering, Vol. 30, No. 4, World Scientific, pp. 2350010-1–2350010-13, 2023.

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