An Initial Analysis of Repair and Side-effect Prediction for Neural Networks

Yuta Ishimoto*, Ken Matsui*, Masanari Kondo*, Naoyasu Ubayashi*, Yasutaka Kamei*
*Kyushu University, Fukuoka, Japan

***Abstract*—With the prevalence of software systems adopting neural network models, the quality assurance of these systems has become crucial. Hence, various studies have proposed repairing methods for neural network models so far to improve the quality of the models. While these methods are evaluated by researchers, it is difficult to tell whether they succeed in all models and datasets (i.e., all developers' environments). Because these methods require many resources, such as execution times, failing to repair neural networks would cost developers their resources. Hence, if developers can know whether repairing methods succeed before adopting them, they could avoid wasting their resources. This paper proposes prediction models that predict whether repairing methods succeed in repairing neural networks using a small resource. Our prediction models predict *repairs* and *side-effects* of repairing methods, respectively. We evaluated our prediction models on a state-of-the-art repairing method Arachne on three datasets, Fashion-MNIST, CIFAR-10, and GTSRB, and found our prediction models achieved high performance, an average ROC-AUC of 0.931 and an average f1-score of 0.880 for the side-effects and an average ROC-AUC of 0.768 and an average f1-score of 0.725 for the repairs.**

***Index Terms*—software engineering for AI, neural network, repair, side-effect**

I. Introduction

An increasing number of software systems have incorporated neural network models, such as autonomous driving [5], [8], medical image analysis [9], [12], and programming [4], [6]. Because of these various software systems, their misbehavior may incur serious accidents, such as auto-driving cars slamming into a truck [1], [2]. Hence, the software engineering research community has started to study the quality assurance of neural network models as part of *Software Engineering for AI (SE4AI)* [3], [10]. One of the research topics is *repairing neural network models* [7], [13], [17], [20], [21], [23]–[25] to prevent software systems from incurring accidents.

To repair neural network models, various studies proposed repairing methods [7], [13], [17], [20], [21], [23]–[25]. For example, Sohn et al. [17] proposed Arachne, repairing neural networks by adjusting the weights that cause misbehavior using a search-based approach. Eniser et al. [7] proposed DeepFault, which identifies suspicious neurons from neural networks that are responsible for misbehavior and generates new data samples to repair the models by retraining.

While these repairing methods have been evaluated for each study, they are not always performing well for all neural network models and datasets because there are external validities, such as the number of studied datasets being limited. Hence, failing to repair neural network models may occur (e.g., the performance of models does not change; a critical issue, such as fairness problems, occurs). Such failures cost developers their resources. For example, the time, energy consumption, and expense of adopting repairing methods come to nothing.

TABLE I: The concept of repairs and side-effects

	incorr. (aft.)	corr. (aft.)
incorr. (bef.)	non-repaired samples	**repaired samples (repairs)**
corr. (bef.)	**broken samples (side-effects)**	non-broken samples

bef./aft. = a model before/after adopting repairing method
incorr./corr. = incorrectly/correctly predicted

Unfortunately, it is difficult to tell whether repairing methods will succeed in repairing models before applying them to models. In other words, developers cannot know whether they have to apply repairing methods to neural network models in terms of their cost. If they make this decision before adopting repairing methods, they could avoid wasting their resource.

In this paper, we conduct the first study to predict whether repairing methods succeed in repairing neural networks only using a small resource to support the developers' decision. More specifically, we built two types of models; the *repair prediction model* and *side-effect prediction model*. The repair prediction model predicts *repairs* indicating that incorrectly predicted samples become correctly predicted by a repairing method. We call these correctly predicted samples *repaired samples*. The side-effect prediction model, on the other hand, predicts *side-effects* indicating that correctly predicted samples become incorrectly predicted. We call these incorrectly predicted samples *broken samples*.

Table I shows the concept of repairs and side-effects in this study. As shown in the table, we can divide data samples into four categories; non-repaired samples, repaired samples, broken samples, and non-broken samples. In other words, the repair (side-effect) prediction model is a binary classification model that predicts whether an incorrectly (correctly) predicted sample is non-repaired (non-broken) or repaired (broken). The key idea is to predict not only positive outcomes (i.e., repairs) of repairing methods but also negative outcomes (i.e., side-effects). It allows developers to make the decision based not only on the repaired misbehavior of neural networks but also on the injected misbehavior by repairing methods.

As the studied repairing method, we use Arachne [17], a state-of-the-art method. We use fully connected and convolutional neural networks as the studied neural networks

and Fashion-MNIST [22], CIFAR-10 [11], and GTSRB [18] datasets as the studied datasets.

We investigate the following three research questions (RQs) to propose our prediction models.

RQ1: How different are repaired and non-repaired samples?

This RQ identifies the differences between repaired and non-repaired samples for repair prediction. We use four explanatory metrics (i.e., entropy, Prediction Confidence Score (PCS), Label Prediction Score (LPS), and loss) to identify the differences.

RQ2: How different are broken and non-broken samples?

This RQ identifies the differences between broken and non-broken samples for side-effect prediction. We use the same explanatory metrics as RQ1 to identify the differences.

RQ3: To what extent can repairs and side-effects be predicted?

Given the previous RQs, we found that the repair and side-effect are predictable by the four explanatory metrics. Based on these metrics, we build the repair and side-effect prediction models. We evaluate these models in this RQ.

We revealed that (1) the repaired and non-repaired samples and the broken and non-broken samples are distinguishable by the studied four explanatory metrics; (2) our prediction models achieved high performance, an average ROC-AUC of 0.931 and an average f1-score of 0.880 for the side-effects and an average ROC-AUC of 0.768 and an average f1-score of 0.725 for the repairs.

II. Related Work

In recent years, as neural networks have been incorporated into practice, many studies have been conducted to repair the misbehavior of the model [7], [13], [17], [20], [21], [23]–[25].

Sohn et al. [17] proposed Arachne, directly correcting the weights related to the misbehavior of deep neural networks by a search-based approach. Arachne takes as input an *original model*, which is a trained neural network model that misbehaves, and outputs a *patched model* in which the weights that cause the original model to misbehave are corrected, as shown in the upper part of Figure 1. Arachne is the studied repairing method in this paper because it is a state-of-the-art method.

Arachne consists of two phases: *localization* and *patch generation*. The localization phase identifies the suspicious weights to be corrected. The authors propose bidirectional fault localization that identifies the suspicious weights that have a high impact on the output and are responsible for the misbehavior. The patch generation phase uses Differential Evolution (DE) [19] to correct the localized weights. DE is a population-based optimization algorithm that iteratively searches for candidate solutions with a higher value of a fitness function. In the DE implemented in Arachne, the candidate solution is the vector consisting of the weight values identified in the localization phase. The fitness function is computed using the loss value when the candidate solution is set to the identified weights. Arachne corrects the suspicious weights by iteratively generating new candidate solutions and computing the fitness function. The advantage of Arachne is that it can repair models without having to generate new data samples and retrain them.

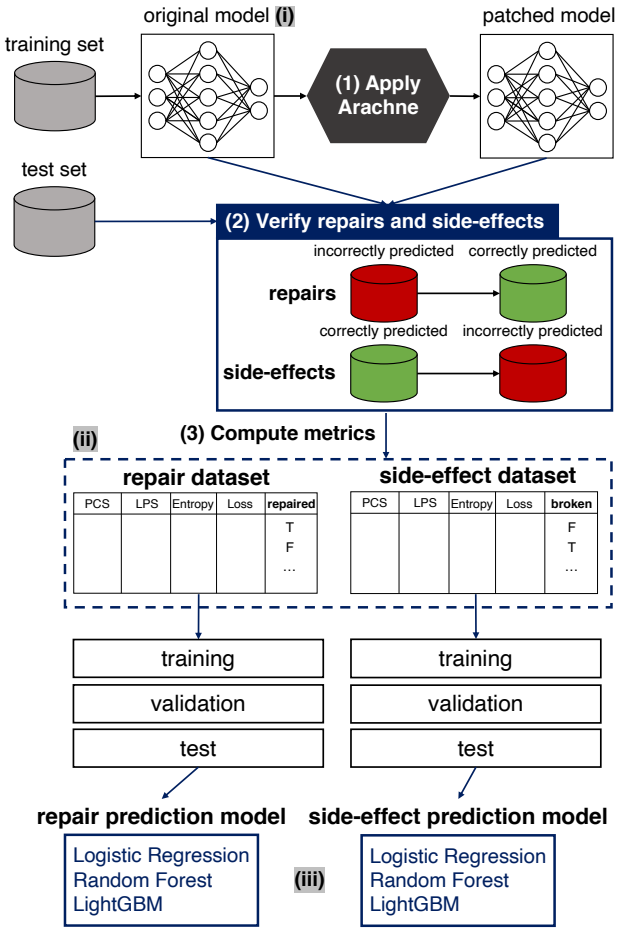

Fig. 1: The overview of our approach

Tokui et al. [20] proposed NeuRecover, a repairing method of deep neural networks using training history. NeuRecover has two phases similar to Arachne (localization and patch generation phases) and uses a search-based approach to generate the patch. However, this method differs from Arachne in that it uses training history to suppress side-effects.

Therefore, it is an important research direction to focus not only on the positive outcomes of repairing methods but also on the negative outcomes. Our study aims to characterize and predict these positive and negative outcomes (i.e., repairs and side-effects) before adopting repairing methods, giving developers a hint of whether they should adopt these methods.

III. Experimental Setup

A. Overview

Figure 1 shows the overview of our approach. To build and evaluate our repair and side-effect prediction models, we conducted the following steps: (i) training original models to be repaired, (ii) making repair and side-effect datasets that have repaired and non-repaired samples and broken and non-broken samples, respectively, based on the prediction results

TABLE II: Top-3 frequent faults (The number in each cell represents the number of samples misclassified, and in parentheses represents the correct label → predicted label.)

dataset	fault1	fault2	fault3
FM	62 (6 → 0)	48 (0 → 6)	27 (2 → 4)
C10	175 (3 → 5)	156 (5 → 3)	89 (1 → 9)
GTSRB	30 (17 → 38)	23 (27 → 2)	22 (26 → 25)

of the original and patched models, and (iii) building repair and side-effect prediction models using repair and side-effect datasets. In Figure 1, the part annotated with Roman numerals is the output of each step. We describe the details of them below.

B. Training original models

Our study uses the same three image classification datasets (*studied datasets*) as those studied in Arachne [17]. For each studied dataset, we train one original model used in Arachne. Each studied dataset has a training and test set, and we use the entire training set to train the original models, and the entire test set to apply Arachne to the original models to repair their weights. The details of these models and our experimental results are all available as our replication package.[1]

Fashion-MNIST [22]. Fashion-MNIST (FM) is a dataset for the classification of fashion images. Each sample is a 28x28 grayscale image and corresponds to a category of fashion images; a label is a number from zero to nine representing a category. This dataset has 60,000 training samples and 10,000 test samples. The corresponding model consists of two convolutional layers followed by two fully connected layers.

CIFAR-10 [11]. CIFAR-10 (C10) is an image dataset for 10-class classification. Each image is 32x32 in size and has three channels (i.e., RGB). This dataset has 50,000 training and 10,000 test samples. The corresponding model consists of four convolutional layers followed by three fully connected layers.

GTSRB [18]. GTSRB is a traffic sign image dataset for the automatic recognition of traffic signs. Each image has three channels (i.e., RGB) and is resized to 48x48 to use the same model used in Arachne. This dataset has 43 labels in total, such as speed limit and stop signs. This dataset has 39,209 training samples and 12,630 test samples. The corresponding model consists of three convolutional layers followed by two fully connected layers.

C. Making repair and side-effect datasets

In our experiments, we make two types of datasets: a *repair dataset* and a *side-effect dataset* as the history of applying Arachne to original models. This section describes the procedure for making these datasets. The Arabic numerals in Figure 1 correspond to the procedure described below.

(1) Apply Arachne to original models. Arachne is applied to original models to repair their *faults* using the test set of studied datasets. Here, *faults* indicate the cases where original models incorrectly predict the labels of samples. Because Arachne in the original paper concentrates on the most frequently appearing top-3 faults, we also used top-3 faults as the studied types of faults. Table II shows the type of top-3 faults and their frequency. For example, in the model in the FM dataset, the most frequent fault is to classify samples of the label of six into zero (62 samples). We get nine patched models (three studied datasets and three types of faults) by Arachne using the algorithm as described in Section II.

(2) Verify repairs and side-effects. The original and patched models predict the samples in the test set of studied datasets. Based on whether the predictions of the original and patched models are correct, the data sample is divided into four categories as shown in Table I. The repair dataset consists of repaired and non-repaired samples (the second row in Table I), and the side-effect dataset consists of broken and non-broken samples (the third row in Table I).

(3) Compute metrics. In the repair and side-effect datasets, we also compute and store four explanatory metrics for each sample. Our repair and side-effect prediction models use these metrics to distinguish repaired samples in the repair dataset and broken samples in the side-effect dataset. We describe the details of these metrics in Section III-D1.

D. Building repair and side-effect prediction models

1) Explanatory metrics: We use four explanatory metrics since we assume that repairs and side-effects can be predicted using confidence and ambiguity of the model's output. These metrics are calculated given an original neural network model M, a sample x, and its ground truth label y.

Entropy. Entropy indicates the ambiguity of the prediction. We use the well-known Shannon entropy [16].

$$\text{Entropy}(x, M) = -\sum_{i \in C} (P_x[i] * \log P_x[i]) \tag{1}$$

where $P_x[i]$ means the prediction probability that the model M predicts that x belongs to label i. C is the set of labels.

PCS (Prediction Confidence Score). PCS [26] is the difference between the highest prediction probability and the second-highest prediction probability. The closer this value is to one, the more confident the model is in its prediction, and the closer it is to zero, the less confident the model is in its prediction. The PCS for a sample x of a model M is defined as follows;

$$\text{PCS}(x, M) = P_x[c_{pred}] - \max_{i \in C \setminus c_{pred}} P_x[i] \tag{2}$$

where c_{pred} is the label predicted by M, which is the label with the highest prediction probability.

LPS (Label Prediction Score). We named LPS the predicted probability for the ground truth.

$$\text{LPS}(x, y, M) = P_x[y] \tag{3}$$

Loss. Loss is the value of the loss function during prediction, and we use *categorical cross entropy* [14] as the loss function since the model is for multi-class image classification.

$$\text{Loss}(x, y, M) = \text{CategoricalCrossEntropy}(x, y, M) \tag{4}$$

[1]https://doi.org/10.5281/zenodo.7329278

2) Procedure: We build the repair and side-effect prediction models using these explanatory metrics as independent variables and repaired or non-repaired (broken or non-broken) as the label in the repair (side-effect) datasets. First, we perform undersampling to balance the numbers of samples since the distribution of the label of the repair and side-effects dataset is imbalanced. As a result, the repair dataset has 402, and the side-effect dataset has 580 samples for each label. Second, for each of the repair and side-effect datasets, 80% of the samples are used to train and validate the model through 5-fold cross-validation to tune the hyperparameters of the repair and side-effect prediction models. The remaining 20% is the test set. We use three modeling techniques: logistic regression (LR), random forest (RF), and LightGBM (LGB). We also use five criteria to evaluate the prediction performance of the repair and side-effect prediction models: accuracy (acc.), precision (pre.), recall (rec.), f1-score (f1.), ROC-AUC, and PR-AUC.

IV. Results

A. RQ1: How different are repaired and non-repaired samples?

Motivation and Approach. We investigate how repaired and non-repaired samples differ in four explanatory metrics. We compare their distribution of metrics in the repair dataset. To depict the distribution of each metric, a box-cox transformation [15] and standardization are applied to each metric.

Results. Figure 2 shows the distribution of four explanatory metrics for repaired and non-repaired samples. The x-axis shows the repaired (True) or non-repaired (False) samples, and the y-axis shows the transformed (not actual) values of each metric.

Confidently incorrectly predicted samples are difficult to be repaired. The distribution of PCS for repaired and non-repaired samples are the opposite; non-repaired samples tend to have a higher PCS, but repaired samples tend to have a lower PCS. The more confident the model is in predicting the sample, the more difficult it is to repair the sample. In addition, non-repaired samples tend to have a lower LPS. In other words, the prediction probability for the correct label is low. From this, we can see that non-repaired samples are in which the model is confident in its predictions despite the model's mistake. The loss tends to be lower for the repaired samples than for the non-repaired samples. On the other hand, it is difficult to find a clear trend in the entropy.

Summary of RQ1

Non-repaired samples are the case where the model confidently incorrectly predicts, with high PCS and low LPS. On the other hand, repaired samples are the case where the model less confidently incorrectly predicts but has lower loss and higher LPS.

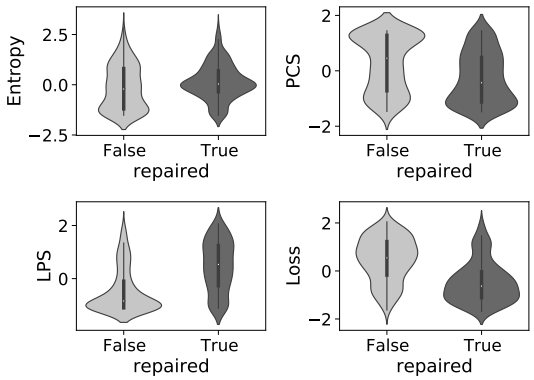

Fig. 2: Distribution of metrics for repaired/non-repaired samples

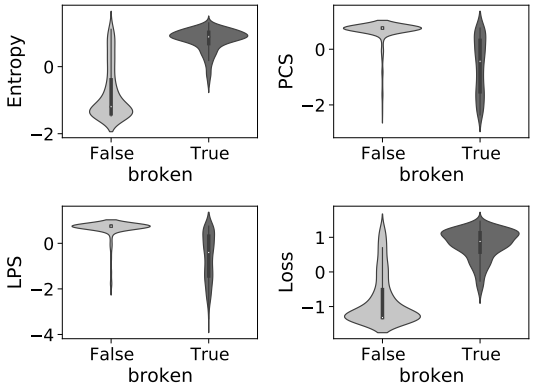

Fig. 3: Distribution of metrics for broken/non-broken samples

B. RQ2: How different are broken and non-broken samples?

Motivation and Approach. In this RQ, we conduct the same approach for broken and non-broken samples instead of repaired and non-repaired ones. We compare their distribution of metrics in the side-effect dataset.

Results. Samples in which the model's prediction is ambiguous are likely to be broken when adopting Arachne. Figure 3 shows the distribution of four explanatory metrics for broken and non-broken samples. Broken samples tend to have higher entropy and loss compared to non-broken samples. Entropy represents the ambiguity of prediction, and our loss function includes a type of entropy. Thus, samples in which the model's prediction is ambiguous are more likely to be broken when adopting Arachne. PCS and LPS are high for both broken and non-broken samples because samples in the side-effect dataset are originally correctly predicted; in other words, the models confidently predict them.

Summary of RQ2

Broken samples are likely to be ambiguously predicted by the model and have high entropy and loss. PCS and LPS are ineffective for distinguishing between broken and non-broken.

TABLE III: Prediction results of the repair prediction models

model	acc.	pre.	rec.	f1.	ROC-AUC	PR-AUC
LR	**0.727**	**0.720**	0.720	0.720	**0.779**	**0.746**
RF	0.723	0.695	**0.771**	**0.731**	0.774	0.724
LGB	0.715	0.684	**0.771**	0.725	0.752	0.696
Avg.	0.722	0.700	0.754	0.725	0.768	0.722

The bolded numbers indicate the model with the best value for each metric.

TABLE IV: Prediction results of the side-effect prediction models

model	acc.	pre.	rec.	f1.	ROC-AUC	PR-AUC
LR	**0.888**	**0.858**	**0.924**	**0.890**	**0.936**	**0.901**
RF	0.874	0.854	0.894	0.874	0.931	0.874
LGB	0.874	0.839	0.918	0.876	0.925	0.870
Avg.	0.879	0.850	0.912	0.880	0.931	0.882

The bolded numbers indicate the model with the best value for each metric.

C. RQ3: To what extent can repairs and side-effects be predicted?

Motivation and Approach. Given RQ1 and RQ2, we found that the studied metrics distinguish between repaired/broken samples and non-repaired/non-broken samples. In this RQ, we build and evaluate our repair and side-effect prediction models as described in Section III-D.

Results. Table III and IV show the prediction results for the repair and side-effect prediction models in the test set.

All evaluation criteria of the side-effect prediction model are above 0.850 on average. We believe this is good performance for an initial model. This high performance is because the four explanatory metrics can distinguish broken/non-broken samples, as confirmed in RQ2. For all evaluation criteria, the LR model showed the best values.

The repair prediction model achieves an average of 0.700 to 0.768, less than the side-effect prediction model. From Table III and IV, we can see that distinguishing repaired/non-repaired samples is more difficult than broken/non-broken samples. This result is due to the difference in the distribution of the four explanatory metrics shown in Figure 2 and 3. These metrics for repaired/non-repaired samples overlapped more than broken/non-broken samples resulting in difficulty distinguishing repaired/non-repaired samples.

Summary of RQ3

The side-effect prediction model can tell broken/non-broken samples with high performance, especially an average ROC-AUC of 0.931 and an average f1-score of 0.880. The repair prediction model, on the other hand, can tell repaired/non-repaired samples with an average ROC-AUC of 0.768 and an average f1-score of 0.725. Hence, predicting repaired samples is more difficult than predicting broken samples.

V. Threats to Validity

Internal Validity. We used only four metrics, entropy, PCS, LPS, and loss, as explanatory variables for the repair and side-effect prediction models; however, this may not be sufficient. Other explanatory variables, such as the variance of the predicted probability and the quantities of information in the data itself, may be effective for repair and side-effect prediction. The number of samples of the repair and side-effect datasets is limited to 804 and 1,160, respectively. It is due to the undersampling described in Section III-D2. Our results can be more reliable by increasing the number of samples (e.g., considering faults after top-3) in the repair and side-effect datasets. The number of folds for cross-validation when training repair and side-effect prediction models are set to five, but there is no strong evidence for this. It is worth considering increasing the number of folds or using alternatives such as bootstrap sampling for training.

External Validity. We used Arachne as a state-of-the-art repairing method, but many other methods exist. Further research is needed to investigate its applicability to other repairing methods. We used three image datasets in the experiments. For future work, we need to apply our method to classification tasks for datasets other than image datasets (e.g., natural language datasets). When training the original model described in Section III-B, the hyperparameters of the original model were manually adjusted without having the validation set. Therefore, it is necessary to investigate whether similar results can be obtained for a well-tuned neural network model.

VI. Conclusion and Future Work

In this study, we investigated the differences between repaired and non-repaired samples and broken and non-broken samples when adopting a repairing method, Arachne, in terms of four explanatory metrics (i.e., entropy, PCS, LPS, and loss). Because of the cost of adopting Arachne, we built lightweight models that predict whether repairs and side-effects occur, called the repair and side-effect prediction models. The experimental results showed that the side-effect prediction model achieves an average ROC-AUC of 0.931 and an average f1-score of 0.880; the repair prediction model achieves an average ROC-AUC of 0.768 and an average f1-score of 0.725.

Our goal is to reduce costs for deep learning model developers by building repair and side-effect prediction models that can handle multiple repairing methods. Hence, we plan to extend the experiment as follows: (1) Compare multiple repairing methods. (2) Use not only traditional models but also state-of-the-art models as repair and side-effect prediction models. (3) Evaluate the outcome of the repair and side-effect with more evaluation criteria, such as the execution cost, the robustness, and the fairness of the repaired/broken samples.

Acknowledgement

This research was partially supported by JSPS KAKENHI Japan (Grant Numbers: JP18H04097, JP21H04877, JP20H04167).

REFERENCES

[1] "Video shows tesla on autopilot slam into truck on taiwan highway," https://www.taiwannews.com.tw/en/news/3943199, 2020.

[2] "2 dead in tesla crash after car 'no one was driving' hits tree, authorities say," https://www.nbcnews.com/news/us-news/2-dead-tesla-crash-after-car-no-one-was-driving-n1264470, 2021.

[3] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software engineering for machine learning: A case study," in *Proc. of ICSE*, 2019, pp. 291–300.

[4] M. Balog, A. L. Gaunt, M. Brockschmidt, S. Nowozin, and D. Tarlow, "Deepcoder: Learning to write programs," in *Proc. of ICLR*, 2017.

[5] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proc. of ICCV*, 2015, pp. 2722–2730.

[6] M. Chen et al., "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[7] H. F. Eniser, S. Gerasimou, and A. Sen, "DeepFault: Fault localization for deep neural networks," in *Proc. of FASE*, 2019, pp. 171–191.

[8] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1341–1360, 2020.

[9] M. H. Hesamian, W. Jia, X. He, and P. Kennedy, "Deep learning techniques for medical image segmentation: achievements and challenges," *J. of digital imaging*, vol. 32, no. 4, pp. 582–596, 2019.

[10] F. Khomh, B. Adams, J. Cheng, M. Fokaefs, and G. Antoniol, "Software engineering for machine-learning applications: The road ahead," *IEEE Software*, vol. 35, no. 5, pp. 81–84, 2018.

[11] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[12] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Med. Image Anal.*, vol. 42, pp. 60–88, 2017.

[13] S. Ma, Y. Liu, W. C. Lee, X. Zhang, and A. Grama, "Mode: Automated neural network model debugging via state differential analysis and input selection," in *Proc. of ESEC/FSE*, 2018, pp. 175–186.

[14] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[15] R. M. Sakia, "The box-cox transformation technique: a review," *J. R. Stat. Soc. Ser. D*, vol. 41, no. 2, pp. 169–178, 1992.

[16] C. E. Shannon and W. Weaver, *The mathematical theory of communication*. Univ. of Illinois Press, 1949.

[17] J. Sohn, S. Kang, and S. Yoo, "Arachne: Search based repair of deep neural networks," *ACM Trans. Softw. Eng. Methodol.*, sep 2022, just Accepted.

[18] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The german traffic sign recognition benchmark: a multi-class classification competition," in *Proc. of IJCNN*, 2011, pp. 1453–1460.

[19] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *J. of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[20] S. Tokui, S. Tokumoto, A. Yoshii, F. Ishikawa, T. Nakagawa, K. Munakata, and S. Kikuchi, "Neurecover: Regression-controlled repair of deep neural networks with training history," in *Proc. of SANER*, 2022, pp. 1111–1121.

[21] M. Wardat, W. Le, and H. Rajan, "Deeplocalize: fault localization for deep neural networks," in *Proc. of ICSE*, 2021, pp. 251–262.

[22] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[23] B. Yu, H. Qi, Q. Guo, F. Juefei-Xu, X. Xie, L. Ma, and J. Zhao, "Deeprepair: Style-guided repairing for deep neural networks in the real-world operational environment," *IEEE Trans. Reliab.*, pp. 1–16, 2021.

[24] H. Zhang and W. Chan, "Apricot: a weight-adaptation approach to fixing deep learning models," in *Proc.of ASE*, 2019, pp. 376–387.

[25] X. Zhang, J. Zhai, S. Ma, and C. Shen, "Autotrainer: An automatic dnn training problem detection and repair system," in *Proc. of ICSE*, 2021, pp. 359–371.

[26] X. Zhang, X. Xie, L. Ma, X. Du, Q. Hu, Y. Liu, J. Zhao, and M. Sun, "Towards characterizing adversarial defects of deep learning software from the lens of uncertainty," in *Proc. of ICSE*, 2020, pp. 739–751.