Mitigating Regression Faults Induced by Feature Evolution in Deep Learning Systems

HANMO YOU, ZAN WANG, XUYANG CHEN, and JUNJIE CHEN, College of Intelligence and Computing, Tianjin University, Tianjin, China
JUN SUN, School of Computing and Information Systems, Singapore Management University, Singapore, Singapore
SHUANG LIU, School of Information, Renmin University of China, Beijing, China
ZISHUO DONG, College of Intelligence and Computing, Tianjin University, Tianjin, China

Deep learning (DL) systems have been widely utilized across various domains. However, the evolution of DL systems can result in regression faults. In addition to the evolution of DL systems through the incorporation of new data, feature evolution, such as the addition of new features, is also common and can introduce regression faults. In this work, we first investigate the underlying factors that are correlated with regression faults in feature evolution scenarios, i.e., redundancy and contribution shift. Based on our investigation, we propose a novel mitigation approach called FeaProtect, which aims to minimize the impact of these two factors. To evaluate the performance of FeaProtect, we conducted an extensive study comparing it with state-of-the-art approaches. The results show that FeaProtect outperforms the in-processing baseline approaches, with an average improvement of 50.6%–56.4% in terms of regression fault mitigation. We also show that FeaProtect can further enhance the effectiveness of mitigating regression faults by integrating with state-of-the-art post-processing approaches.

CCS Concepts: • **Software and its engineering → Software evolution**; **Software testing and debugging**;

Additional Key Words and Phrases: Regression Mitigation, Regression Fault, Deep Learning, Feature Evolution, Fault Mitigation

This work was supported by the National Natural Science Foundation of China under Grant Nos. 62472310, 62322208, 62232001, 12411530122, and 62472429, and the Ministry of Education, Singapore under its Academic Research Fund Tier 3 (Award ID: MOET32020-0004).

Authors' Contact Information: Hanmo You, College of Intelligence and Computing, Tianjin University, Tianjin, China; e-mail: youhanmo@tju.edu.cn; Zan Wang, College of Intelligence and Computing, Tianjin University, Tianjin, China; e-mail: wangzan@tju.edu.cn; Xuyang Chen, College of Intelligence and Computing, Tianjin University, Tianjin, China; e-mail: xuyangchen@tju.edu.cn; Junjie Chen (corresponding author), College of Intelligence and Computing, Tianjin University, Tianjin, China; e-mail: junjiechen@tju.edu.cn; Jun Sun, School of Computing and Information Systems, Singapore Management University, Singapore, Singapore; e-mail: junsun@smu.edu.sg; Shuang Liu, School of Information, Renmin University of China, Beijing, China; e-mail: shuang.liu@ruc.edu.cn; Zishuo Dong, College of Intelligence and Computing, Tianjin University, Tianjin, China; e-mail: zishuodong@tju.edu.cn.

ACM 1557-7392/2025/7-ART171
https://doi.org/10.1145/3712199

ACM Reference format:
Hanmo You, Zan Wang, Xuyang Chen, Junjie Chen, Jun Sun, Shuang Liu, and Zishuo Dong. 2025. Mitigating Regression Faults Induced by Feature Evolution in Deep Learning Systems. *ACM Trans. Softw. Eng. Methodol.* 34, 6, Article 171 (July 2025), 33 pages.
https://doi.org/10.1145/3712199

1 Introduction

Deep Learning (DL) systems are extensively applied across various domains, including autonomous driving [136], medical diagnosis [103], and software engineering [47]. Similar to traditional software, DL systems also require continuous evolution to meet new requirements or enhance their performance. During the process of model evolution, apart from continuously collecting new data, introducing new features, i.e., pre-defined attributes, to improve the performance of DL models is also a common and effective strategy [21, 24, 134], and even recommended as an important practice by the Google Team in their guidelines for DL system development [137, Phase II]. For instance, Google's recommendation and ranking system utilized 200 distinct features, such as link quality and page view rate [29], and the number of features kept increasing with the system evolving. They had plans to increase this number by more than 550 features in the same year [56]. Presently, in order to better cater to customer needs, Google's recommendation system continues to expand the number of features it employs. Similar situations can be observed in platforms such as Taobao [67, 134] and YouTube [28], where the effectiveness of their recommendation systems is achieved through the continuous introduction of hundreds, or even thousands, of features.

Like traditional software, during the evolution process, regression faults may be induced in DL systems, i.e., test inputs correctly predicted by the old version of the model are incorrectly predicted by the new version [126]. One example is the Google Flu Trends system, which aims to predict flu outbreaks by analyzing search queries related to flu symptoms. The system worked correctly until certain new features were introduced in the model in 2013, which led to biased estimates [38, 82] and caused the prediction error of the system to increase by over 100% [17]. Consider the context of AI-assisted healthcare systems, where data collection often poses significant challenges, particularly for diseases with very few patients. Given the limited overall patient pool, hospitals have to conduct follow-ups and revisit patients with specific illnesses to collect features throughout their treatment journey. These features span from fundamental clinical data (e.g., heights and weights) to more complex data with potentially invasive examination, such as imaging, biochemical, genetic, and pathological information [20, 73, 123]. While later-stage data collection can be costly, the evolution of integrating these features holds a pivotal role in enhancing the effectiveness of AI-assisted healthcare systems. Suppose that the developers trained a DL model with 95% **Accuracy (ACC)** to aid the doctor in deciding whether a patient requires hospitalization. Initially, he tends to be cautious and carefully verify the recommendations of the DL model, lacking full trust. However, after a year of interacting with this AI system, he gets used to it and learns a solid "mental model" [76, Chapter I], enabling him to confidently rely on the AI-advised actions for the patients. Now, the developers updated the DL model with 98% ACC. Although there is a 3% improvement, the new model makes errors specifically in prediction for certain elderly patients. While the AI has improved by 3%, the doctor is unaware of the new errors, and as a result of this outdated mental model, takes the wrong actions for these elderly patients [13, 62, 104]. These regression faults may lead to severe consequences, such as missing diagnosing diseases [13] or false alarms that cause social panic [17]. Therefore, it is crucial to reduce regression faults induced by the model feature evolution process.

With increasing attention to regression faults in DL systems, several approaches have been proposed to reduce such faults. In the literature, there are two types of in-processing regression fault mitigation approaches. Tokui et al. [102] proposed NeuRecover, which first identifies the weights responsible for regression faults in the DL system and then adjusts the weights to fix them before deployment. While NeuRecover can address regression faults with minimal impact on ACC, it over-relies on a limited number of regression faults from the training set for localizing weights and guiding the repair process, thereby restricting its generalizability. When faced with regression faults deviating from the distribution of the faults in the training set, it has limited effectiveness, as confirmed in our study in Section 5.4.1 (i.e., it fixes only 20% of regression faults from a different distribution). You et al. [126] proposed a finetuning-based approach to fix regression faults by finetuning on regression-fault-triggering test inputs. It can mitigate regression faults from different distributions but may lead to ACC drop or over-fitting due to just focusing on learning from regression-fault-triggering test inputs [42]. In addition, all these approaches are proposed and evaluated in the regression scenario introducing new data (data evolution) and overlooks the regression scenario adding new features (feature evolution). Indeed, as confirmed in our study (to be presented in Section 5), the effectiveness of these approaches becomes unsatisfactory since they do not consider the potential causes of the regression faults in this scenario. Therefore, it is important to design a mitigation approach specifically for the regression scenario introducing new features in DL systems.

To address the aforementioned limitations, in this article, we proposed a novel regression fault mitigation approach, called FeaProtect, for feature evolution scenarios. Specifically, FeaProtect targets two underlying causes of regression faults in feature evolution scenarios, i.e., redundancy and contribution shift, which are inspired by the causes of regression faults in traditional software. First, the developers may focus more on the development of new functionalities and overlook that code implementing new functionality may trigger erroneous behaviors or bring compatibility issues in existing old functionalities [71, 128]. Second, even if the code for new functionality is bug-free, an overemphasis on preserving old functionalities without timely optimization can be problematic. As technology stacks become outdated, the old functionalities based on outdated libraries or techniques may lead to regression faults [11, 80]. That is, regression faults in traditional software are typically induced when developers overemphasize either the development of new functionalities or the preservation of old functionalities, without giving equal consideration to both. Regression faults induced by feature evolution in DL systems may occur in similar ways. First, as the model learns new features, it may neglect the knowledge gained from old features and their important contributions, ultimately leading to regression faults. Second, if the model fails to capture the effective information in new features and instead overly relies on old features, this will adversely affect its generalizability and may trigger regression faults. To mitigate these issues, ideally, the model should strike a balance between learning both new and old features. An excessive focus on either of them could potentially lead to regression faults. To further investigate the potential causes behind the model's reliance on old features and new features, we conducted a motivating study (Section 3.4). The study found that redundancy and contribution shift are correlated with regression faults in feature evolution.

FeaProtect is designed to tackle the challenges of balancing the learning of both new and old features. To address feature redundancy, which causes the model to over-rely on old features, an intuitive approach is to eliminate redundant features as soon as they are identified before they influence the training process. Therefore, FeaProtect employs a pre-processing approach to eliminate them. To address the issue of contribution shift, a suitable approach is to control it throughout the entire training process. Thus, FeaProtect addresses it using an in-processing

approach. For redundancy, FeaProtect employs a pre-processing approach called redundancy-guided feature selection to eliminate redundant information introduced by the new features. This process aims to enable the model to learn more useful new features rather than over-rely on old features. Subsequently, for contribution shift, FeaProtect proposes an in-processing approach, contribution-controlled training. This approach helps protect the knowledge gained from the old features from being severely impacted, thus preventing the model from over-relying on new features. By combining these two complementary approaches, FeaProtect can harmoniously balance learning from both old and new features, thereby effectively mitigating regression faults.

We conducted an extensive study based on 28 popular subjects (a dataset paired with its regression configuration). Our experimental results show that FeaProtect consistently outperforms all the compared approaches with an average improvement of 50.6%–56.4% measured using the reduction of regression faults (to be introduced in Section V-D1). Notably, we integrated FeaProtect with state-of-the-art post-processing regression mitigation approaches. The results show that FeaProtect can reduce more regression faults when integrated with post-processing approaches.

In this work, we make the following major contributions:

- —*New Knowledge*. We conduct an empirical study to investigate the underlying causes of regression faults induced by feature evolution, providing new knowledge and findings for DL regression testing.
- —*New Approach*. We propose a new regression fault mitigation approach that aims to prevent DL systems from over-relying on either new or old features during the feature evolution process. Specifically, our approach comprises a pre-processing component designed to address feature redundancy, which often leads to over-relying on old features. Additionally, we include an in-processing component focused on controlling significant shifts in feature contributions over time to avoid over-relying on new features. These components are modular and can be used independently or in combination. By targeting the potential causes of regression faults, our approach can effectively mitigate these issues while minimizing adverse impact on overall ACC.
- —*Experimental Results*. We conduct extensive experiments with various tasks, models, and datasets, and the results confirm the effectiveness of our approach. In total, we conducted 1,428 experiments (3 times repetition × 28 subjects × 17 approaches, i.e., our approach + 2 ablation variants + 2 in-process baselines +2 natural baselines + 5 post-process baselines + 5 integrated approaches), which is the largest study to our best knowledge.
- —*Benchmark Repository*. We release a high-quality benchmark on regression faults induced by feature evolution on our homepage: https://github.com/youhanmo/FeaProtect and https://zenodo.org/records/14277943, inspiring future research in this field.

Article Organization. The remaining sections of the article are organized as follows: Section 2 briefly introduces the background of regression testing for DL systems. Section 3 elaborates on our motivating study and the findings that motivate the design of our approach. Section 4 describes our regression mitigation approach, FeaProtect. Section 5 presents the details of the experiment setup and the evaluation results of FeaProtect. Sections 6–8 describe the discussion, the related work, and the conclusion, respectively.

2 Background

2.1 Feature Evolution in DL Systems

In the development process of DL systems, feature evolution has become increasingly prevalent. The regression process of feature evolution can be defined as follows: Given a DL model $\mathcal{M}_1$ and

its feature set $\mathbb{F}_1$, after adjusting $\mathbb{F}_1$ by adding, removing, or replacing existing features to obtain $\mathbb{F}_2$, the regression on $\mathcal{M}_1$ is defined as improving $\mathcal{M}_1$ through re-training/finetuning the model with $\mathbb{F}_2$ to obtain $\mathcal{M}_2$. We refer to $\mathcal{M}_1$ as the prior version and $\mathcal{M}_2$ as the regression model of $\mathcal{M}_1$.

In the current DL application development scenarios within the industry, adding new features is a common practice [43, 67, 83]. However, it has not received extensive attention from researchers. Therefore, in this work, we mainly focus on regression by inducing new features to the DL systems and we name it feature-addition regression. Specifically, we define it as follows:

Given a DL model $\mathcal{M}_1$ trained on a set of old features $\mathbb{F}^{\mathcal{O}} = \{f_1^{\mathcal{O}}, f_2^{\mathcal{O}}, \ldots, f_n^{\mathcal{O}}\}$. Then, after collecting a set of new features $\mathbb{F}^{\mathcal{N}} = \{f_1^{\mathcal{N}}, f_2^{\mathcal{N}}, \ldots, f_m^{\mathcal{N}}\}$, the regression process on $\mathcal{M}_1$ is defined as improving $\mathcal{M}_1$ through re-training/finetuning the models based on features in $\mathbb{F}^{\mathcal{O}} \cup \mathbb{F}^{\mathcal{N}}$.

We remark that in DL, features are commonly classified as explicit and implicit. Explicit features are usually designed based on human expertise, business logic, or domain knowledge. They directly correspond to real-world concepts or entities. For example, in the medical imaging field, magnetic resonance imaging, computed tomography, and ultrasound images are explicit features [45]. Conversely, implicit features are not explicitly defined in the raw data but require model learning or data transformation for feature extraction. They are typically high-level abstract representations capturing hidden information within the data. For example, image color distributions and features derived from DL models are implicit features. Both explicit and implicit features evolve in DL, but they face distinct challenges. Explicit features encounter challenges in updating, optimizing, and selecting features to adapt to data shifts or business needs, to improve model performance while mitigating regression faults brought by the evolution. But implicit feature evolution faces the primary challenge of precisely defining and extracting them and accurately quantifying their changes throughout the evolution process. In this work, we mainly tackle challenges related to explicit features, while acknowledging the significance of implicit features for future study.

2.2 Regression Faults in DL Systems and Their Mitigation

Given a k-class classification model $\mathcal{M}$ and an input set $\mathcal{X}$, $\mathcal{M}$ maps an input $x \in \mathcal{X}$ to a k-dimensional vector $\tilde{P}_{\mathcal{M}}(x)$, containing the confidence of each class. The class with the highest confidence in $\tilde{P}_{\mathcal{M}}(x)$ is denoted as $p_{\mathcal{M}}(x)$ and is considered as the prediction result, which is formally defined in Formula (1).

$$p_{\mathcal{M}}(x) = \arg\max \tilde{P}_{\mathcal{M}}(x). \tag{1}$$

Following the definition in previous work [126], given an original version of DL model $\mathcal{M}_1$, its regression model $\mathcal{M}_2$, a test input x, and its ground-truth label y, x triggers a regression fault if the label of x is correctly predicted by $\mathcal{M}_1$, yet wrongly predicted by $\mathcal{M}_2$, i.e., $p_{\mathcal{M}_1}(x) = y \wedge p_{\mathcal{M}_2}(x) \neq y$. In addition, we denote x as the *regression-fault-triggering input* and the *faulty behavior* triggered by x as $(p_{\mathcal{M}_1}(x) \rightarrow p_{\mathcal{M}_2}(x))$.

Given a prior version model $\mathcal{M}_1$, its regression model $\mathcal{M}_2$, a test set $\mathcal{X}$, the problem of *regression fault mitigation* is to train a new model $\mathcal{M}_2'$, that has fewer regression faults than $\mathcal{M}_2$, on the basis of $\mathcal{M}_1$.

2.3 Terminology

In this article, we introduce terminology related to the feature evolution process, specifically feature redundancy and contribution shift, which we hypothesize are the underlying causes of regression faults induced by this process. These causes will be explored further in the motivating study presented in Section 3. Prior to that, we offer clarifying definitions of these terms to establish a common understanding.

— *Feature redundancy* refers to the phenomenon that features exhibit a high degree of correlation or similarity. Training a model with redundant features can have detrimental effects, such as reducing the model's generalization ability [78] and hampering prediction precision [60]. Additionally, it increases the risk of over-fitting old features and severely hinders the model's capacity to learn from other important new features [46, 60, 92, 114], thereby increasing the likelihood of inducing regression faults.
— *Contribution shift* refers to the phenomenon that the importance of existing old features to model prediction undergoes a significant change after adding new features. While a certain degree of contribution shift is normal during the feature evolution process, severe changes in the contribution of old features usually indicate that the DL model neglects the knowledge acquired from the old features while learning the new ones [74, 132], and it is likely to cause regression faults.

3 Motivating Study

As aforementioned, the introduction of new features for model training can potentially result in regression faults. To effectively mitigate these regression faults, it is crucial to understand their underlying causes. Therefore, we conduct an empirical study to investigate the reasons behind the negative impact of new features on the old features, which in turn motivates our mitigation approach.

3.1 Hypotheses

Inspired by the causes of regression faults in traditional software in Section 1, the over-reliance on either old or new features can lead to regression faults during the feature evolution process. While according to existing literature [58], there are several reasons that might cause the DL model to over-rely on certain features, such as poor learning rate selection and sub-optimal model structures, these factors are typically not directly tied to the feature evolution process. In addition to the above factors, there are two factors, i.e., *feature redundancy* [127] and *contribution shift* [70], which provide a suitable explanation for the over-reliance on old and new features in the context of feature evolution. Based on existing literature [10, 52, 68, 127, 133], the most common factor that leads to over-relying on old features is feature redundancy, which can be easily introduced during the process of collecting new features [113]. During the evolution process, if the information in new features significantly overlaps with that in old features, the DL model may perceive that it is not necessary to significantly adjust its weights to learn the valuable, non-overlapping information provided by the new features, as the current weights learned from the old features are already effective for inference. Moreover, the collinearity resulting from feature redundancy can exacerbate the tendency of DL models to over-fit these old features [51], thereby negatively impacting their performance and leading to regression faults. In terms of contribution shift, while moderate changes in feature confidence are reasonable, a significant shift in the confidence of old features often indicates a severe change in the model's decision logic [70]. This suggests that the model is likely to abandon the usage of old features, which in turn increases the likelihood of regression faults.

Therefore, we hypothesize that feature redundancy and contribution shift are the underlying causes for inducing regression faults in the context of feature evolution. In the following, we conduct experiments to validate the aforementioned hypotheses.

3.2 Study Design

To validate the hypotheses regarding regression faults in feature evolution, we design our study to investigate *whether old features that are more heavily influenced by redundant and contribution-shift-induced new features are prone to causing more regression faults.* We consider this

Research Question (RQ) as the basis of our motivating study. To answer this question, it is necessary to measure the extent of redundancy between the new and old features, as well as the contribution shift of the old features. We then rank the old features based on the extent to which they are affected by the regression process.

During the evolution process, the knowledge learned from old features that are severely impacted by redundancy and contribution shift are more likely to be influenced. Therefore, when the regression DL models are predicting test inputs crafted by subtly mutating these features, it is more likely to exhibit fault behaviors. That is, test inputs that are generated by mutating these significantly impacted old features are more prone to reveal regression faults of the models compared to mutating other old features. In this study, we specifically select top-k severely impacted old features for fuzzing, following existing work [53], we select k as 3 and 5 instead of one feature for fuzzing for two reasons: (1) For machine learning tasks involving multiple pre-defined features, the prediction outcome is not usually influenced by a single feature in isolation, but rather by the interaction between features [77]. Fuzzing only one feature might limit the number of faults detected and impact the analysis results. In our cases, by simultaneously fuzzing multiple features, we can enhance the effectiveness of the fuzzing process and increase the chances of triggering more regression faults. (2) The pre-defined features vary in their types [96]; for instance, some features are continuous, which allows for more diverse directions of change. However, categorical (or discrete) features have a limited and fixed range of values to choose from. If only one column of features, especially when it is discrete, is targeted for fuzzing, it may be difficult to effectively trigger regression faults due to the constraints of the search space, or the fault detection results may exhibit a high degree of randomness and instability, thus leading to biased results. By simultaneously fuzzing multiple features, including both continuous and discrete types, we can more comprehensively explore the regression faults of the model.

To measure redundancy between new and old features, an effective approach is to analyze their correlation. While mutual information [91] is commonly used for this purpose, it is more suitable for discrete features. In practice, features can be discrete or continuous. Therefore, we opt for a modified version of mutual information, the **Maximum Information Coefficient (MIC)** [85], as our measurement since it is proven to be more effective than other correlation measurements such as Spearman Correlation [30] and it can be applied to both continuous and discrete features, making it more flexible than mutual information. Given a set of old features $\mathbb{F}^{\mathscr{O}} = \{f_1^{\mathscr{O}}, f_2^{\mathscr{O}}, \ldots, f_n^{\mathscr{O}}\}$ and new features $\mathbb{F}^{\mathscr{N}} = \{f_1^{\mathscr{N}}, f_2^{\mathscr{N}}, \ldots, f_m^{\mathscr{N}}\}$, we calculate the extent of redundancy between each old feature $f_i^{\mathscr{O}}$ and the new features $\mathbb{F}^{\mathscr{N}}$ using the Formula (2) following [14, 48, 81, 93]. A higher score indicates a greater level of redundancy between the specific old feature $f_i^{\mathscr{O}}$ and the new features $\mathbb{F}^{\mathscr{N}}$.

$$Redundancy(f_i^{\mathscr{O}}, \mathbb{F}^{\mathscr{N}}) = \frac{1}{|\mathbb{F}^{\mathscr{N}}|} \sum_{f_j^{\mathscr{N}} \in \mathbb{F}^{\mathscr{N}}} MIC(f_i^{\mathscr{O}}, f_j^{\mathscr{N}}). \tag{2}$$

The contribution shift can effectively reflect model prediction behavior change brought by the new data and new features [70]. Therefore, we utilize the concept of **Feature Contribution Change (FCC)** to measure the influence of contribution shift on old features resulting from the introduction of new features. In this work, we employ the Expected Gradient [35] to measure the contribution of each feature. The Expected Gradient calculates the feature contributions by integrating gradients along a direct path from a reference input to the target input, offering a more precise assessment [44, 87, 111] compared to other explanation techniques while requiring less computational resources, making it well-suited for our large-scale study. Additionally, for each test input x and its corresponding old feature sets $x_{\mathbb{F}^{\mathscr{O}}} = \{x_{f_1^{\mathscr{O}}}, x_{f_2^{\mathscr{O}}}, \ldots, x_{f_n^{\mathscr{O}}}\}$, the *FCC* of $x_{f_i^{\mathscr{O}}}$

of test input x can be defined using Formula (3), where $EG_{f_i}(x; \mathcal{M})$ represents the contribution (Expected Gradient) of feature f_i in the test input x when predicted by the model $\mathcal{M}$, and $\mathcal{M}_1$ and $\mathcal{M}_2$ represent the original model and its regression model, respectively. The higher the *FCC* score of the feature, the more significantly it is affected by the contribution shift [70].

$$FCC(x_{f_i^{\mathscr{O}}}) = EG_{f_i^{\mathscr{O}}}(x; \mathcal{M}_2) - EG_{f_i^{\mathscr{O}}}(x; \mathcal{M}_1). \tag{3}$$

To validate our hypothesis, we conduct the following process for our motivating study. Among the multiple pairs of old versions of models and their corresponding regression models that we have obtained, for each regression model, we select, from the features already present in the old version, the top 3/5 old features that are most affected by redundancy (according to Formula(2)) and the same number of new features induced by contribution shift (according to Formula(3)), respectively. Additionally, we randomly select an equal number of features for comparison. We then use a fuzzing tool to slightly mutate these selected features, in order to test whether the regression model exhibits more regression faults when evaluated on the knowledge learned from these selected old features compared to the others. Finally, we report on the number of regression faults detected. The presence of a higher number of regression faults indicates that the knowledge learned from these old features has been severely disrupted by the feature evolution.

In our experiments, we employ DRFuzz [126], a state-of-the-art black-box regression fuzzing tool, to detect regression faults. DRFuzz considers three key features of test inputs: fault-revealing capability, fidelity, and diversity. As for the fault-revealing capability, DRFuzz uses the prediction difference between different versions of the model to guide the fuzzing process and can therefore effectively detect regression faults. In addition to this capability, DRFuzz carefully designs mutation operators based on metamorphic relations and incorporates a fidelity assurance component that prevents the semantics of the test inputs from drastically changing from their seed inputs. It also emphasizes diversity, utilizing a seed maintenance strategy to enhance the diversity of the inputs. The results demonstrate that DRFuzz can effectively uncover more regression faults and can be used to explain potential vulnerabilities introduced during the evolution process.

We use DRFuzz as the fuzzing tool for two reasons: First, this lightweight tool can be easily adapted to various datasets and provides the flexibility to test the model by selectively mutating specific parts of the test inputs. Second, DRFuzz is a state-of-the-art tool specifically designed for detecting regression faults between the current version and the previous version of the DL systems, while other fault detection approaches [31, 116] are usually used for detecting faults in one specific version of the DL model. Thus, DRFuzz is well-suited for our task.

3.3 Experiment Setup

3.3.1 Dataset. As presented in Table 1, in this study, we conducted experiments with 28 subjects, including 24 from manually constructed regression scenarios (ID:1–24) and four from real scenarios (ID:25–28). Table 1 summarizes the key details of studied subjects, including their ID numbers, the corresponding dataset names, and their abbreviations, the model structures employed, the sizes of the training, validation, and test sets used, the number of classes, the tasks and application domains addressed, the accuracies achieved by each model version, and the number of features utilized in each model version. Since data related to feature evolution is commonly used in industrial application development and is not open-sourced, we opted to use open-source datasets to simulate the feature evolution process. Specifically, We collected datasets from Kaggle and Github. To guarantee the datasets are widely used, we filter out datasets with less than 50 upvotes (for Datasets in Kaggles) or 10 stars (for Datasets from Github). To ensure the generalizability of the experimental results, we included datasets with varying task types and application domains. Finally, 10 datasets

Table 1. Basic Information of Studied Subjects

ID	Dataset	Abbr.	Model	Size	#Class	Task	Applied Domain	ACC	Feature
1	Dry Bean Dataset [54]	Bean	6 Layer FFNN [4]	1,575/175/750	7	Bean Recognition	Agriculture	88.67%↗91.20%	14↗16
2								89.60%↗90.27%	12↗16
3								89.47%↗90.27%	11↗16
4	Fetal Health Classification [12]	Fetal	6 Layer FFNN [5]	1,339/149/638	3	Disease Diagnosis	Medicine	91.07%↗91.54%	19↗21
5								89.03%↗91.22%	17↗21
6								89.18%↗90.90%	15↗21
7	Gtzan Music Genre [106]	Music	7 Layer ANN [1]	630/70/300	10	Music Classification	Music	57.67%↗60.67%	24↗26
8								56.00%↗62.67%	21↗26
9								56.67%↗60.00%	20↗26
10	Telco Customer Churn [99]	Churn	7 Layer FFNN [9]	4,437/493/2,113	2	Churn Prediction	Commerce	76.86%↗80.22%	17↗19
11								77.52%↗77.90%	15↗19
12								76.62%↗78.56%	13↗19
13	Classify Gestures by Reading Muscle Activity [124]	Gesture	5 Layer FFNN [6]	7,536/818/3,504	4	Hand Gesture Classification	Biology	89.50%↗89.90%	58↗64
14								87.73%↗89.47%	45↗64
15								87.36%↗89.90%	34↗64
16	Patient Treatment Classification [88]	Patient	6 Layer FFNN [8]	2,084/232/993	2	Patient Status Prediction	Medicine	72.51%↗73.41%	9↗10
17								72.10%↗73.92%	8↗10
18								73.41%↗74.92%	7↗10
19	Mobile Price Classification [34]	Price	5 Layer FFNN [7]	1,260/140/600	4	Price Range Prediction	Commerce	86.67%↗88.50%	18↗20
20								87.50%↗89.17%	16↗20
21								79.17%↗88.83%	14↗20
22	Diabetes Disease Updated Dataset [100]	Diabetes	7 Layer FFNN [3]	483/54/231	2	Disease Diagnosis	Medicine	64.07%↗71.86%	7↗8
23								68.40%↗70.56%	6↗8
24								65.80%↗70.99%	5↗8
25	Climate Data Daily IDN [101]	Climate	8 Layer FFBPNN [89]	8,883/987/4,230	9	Wind Direction Prediction	Meteorology	45.60%↗46.08%	9↗13
26	Predictive Mutation Testing [66, 130] Wire Bitcoincore Vraptor	PMT-W	CNN [2]	3,468/386/1,653	2	Mutants Killability Prediction	Software Engineering	90.32%↗91.77%	14↗95
27		PMT-B	CNN [2]	5,680/632/2,706	2			89.60%↗89.91%	14↗95
28		PMT-V	CNN [2]	6,613/735/3,150	2			83.56%↗83.90%	14↗95

FFNN, Feed Forward Neural Networks; ANN, Artificial Neural Network; FFBPNN, Feed Forward Back Propagation Neural Network; CNN, Convolutional Neural Network.
The Column "Size" refers to the dataset size, which is recorded in the form of (a/b/c), where a, b, and c represent the sizes of the training, validation, and test sets, respectively.

were selected, covering various application domains like medicine and commerce, and varying tasks such as price prediction and disease diagnosis, with class numbers ranging from 2 to 10.

Please note that, in the datasets released online, dataset publishers usually provide well-rounded and carefully collected full-set features, instead of the features collected at prior stages. Therefore, in our experiments, to simulate the feature evolution process, we removed $a\%$ of the overall features and treated these removed features as newly added features. We then used the remaining $(1 - a\%)$ features as the original features for training the previous version of the model. Specifically, we followed these steps for constructing feature evolution projects. First, we randomly sampled $(1-a\%)$ of the total number of features as the original feature set and trained an initial version of model $\mathcal{M}_1$. Next, we combined the remaining $a\%$ features with the original features to train a new model $\mathcal{M}_2$. Following industry practice, the aim of adding new features is to improve model performance. Therefore, We only considered this training process a successful feature evolution for our study if the ACC of $\mathcal{M}_2$ is higher than that of $\mathcal{M}_1$; otherwise, we discarded it and resampled features to repeat the above process.

In real practice, the developers might introduce different numbers of new features according to different projects during the feature evolution. To simulate evolution scenarios involving the introduction of different numbers of new features, we simulated scenarios where $a\%$ is equal to 10%, 20%, and 30%. For example, in the "Dry Bean Dataset," we added two (10% of 16 features, ID:1), four (20% of 16 features, ID:2), five (30% of 16 features, ID:3) new features, respectively, to simulate the scenarios of adding different numbers of new features. Since features are randomly sampled each time a scenario is constructed, there may be some overlap in the features selected across scenarios with different removal percentages within the same dataset. However, our goal is not to compare the differences resulting from removing different percentages of features; rather, it is to simulate a variety of feature evolution scenarios and observe their impact on the DL system. Therefore, the

potential overlap in selected features does not undermine the validity of our results. Furthermore, due to the need to concatenate the remaining features before training $\mathcal{M}_2$, this change in feature order [49, 61] and randomness [90] in the training process may lead to slight differences in ACC when training the $\mathcal{M}_2$ model. Nonetheless, the overall ACC of these models within one dataset remains similar.

In addition to the manually constructed subjects, we discovered four real feature evolution scenarios. The "Climate Data Daily IDN" [101](ID:25) dataset used weather data as original features and later incorporated geographic information of climate stations to assist prediction. "Predictive Mutation Testing on Wire/Bitcoincore/Vrapto" [66] (ID:26–28) is a widely used software engineering task for mutation testing. Initially, the authors collected 14 features for mutant killability prediction [130], and subsequently introduced an additional 84 features to improve the predictive performance [66]. Since these scenarios are collected from practical feature evolution scenarios, we do not need to manually construct additional subjects of feature evolution for them. To ensure the practicality of our experiments, the DL models used in this work were obtained from Kaggle projects, research papers, and GitHub repositories associated with these datasets. For some models, the model developers have explicitly categorized their models as specifically designed Artificial Neural Networks, Feed Forward Back Propagation Neural Networks, and Convolutional Neural Networks. However, for other models, where the developers do not specify their model types, we categorize these models as **Feed Forward Neural Networks (FFNNs)**. It is important to note that, although these models fall within the broader FFNN category, they vary in their architectures and training processes as they are tailored to some degree for their corresponding datasets. In addition, FeaProtect can be extended to more complex DL models, which we will discuss in Section 6.1.

To evaluate the generalizability of the conclusions of our study, we sampled four datasets (ID: 1–12) for our study and reserved the remaining subjects (ID: 13–28) to evaluate whether our mitigation strategy, which addresses the causes of regression faults in the datasets used in the motivating study, can be effectively applied to mitigate regression faults in more datasets.

3.3.2 Metrics. Like traditional software, the occurrence of regression faults in a specific functionality indicates that this functionality has been affected during the regression process. Since our primary objective is to investigate the potential causes of regression faults, we primarily emphasize metrics related to the quantity and diversity of such faults. Specifically, following existing work [126], we use *the number of regression-fault-triggering test inputs (denoted as #RFI)* to measure the number of regression faults detected in fuzzing. In our study, RFI (Fuzzed) refers to the RFI generated based on fuzzed test sets, while RFI (Test) refers to the RFI in the original test set.

It is important to generate as many fault-triggering test inputs as possible. However, if the generated inputs trigger duplicate faults, this would hinder the effective analysis of the vulnerabilities in DL systems. Therefore, it is crucial to consider the diversity of fault-triggering inputs. Following existing work, we consider two aspects of this diversity: The first aspect considers static information of the fuzzing process, i.e., the initial seeds. The intuition is that if two fault-triggering inputs are obtained from different initial seeds, then they are more likely to trigger different regression faults. Therefore, we adopt *the number of initial seeds for fault-triggering test inputs* (denoted as #Seed) from existing work to measure the seed diversity of test input. The second aspect reflects the dynamic behavior of a regression-fault-triggering test input, which complements the static initial seed information to assist in better diversification of faults. The intuition is that if two fault-triggering inputs show different faulty behaviors, then they are more likely to trigger different regression faults. Based on the static initial seeds and the dynamic faulty behaviors for fault-triggering test inputs, We adopt *the number of regression faults detected by fault-triggering test inputs* (#RF), which

is the count of the [initial seed, faulty behavior] tuples of these inputs (defined in Section 2.2), to measure the behavior diversity.

3.3.3 Implementation and Environment. We conducted the experiments on Tensorflow 2.3.0 and adopted the existing implementation of DRFuzz [126]. Specifically, DRFuzz consists of three fundamental components: mutation rules aimed at enhancing fault-revealing capabilities, GAN-based fidelity assurance to improve the fidelity of generated test cases, and a seed maintenance strategy to increase test case diversity. Since the third component is applicable to any modality of test cases, to adapt DRFuzz to our experiments, we only need to adjust the first two components. Specifically, regarding mutation rules, we designed four mutation rules, inspired by existing works [63, 110], which are listed as follows:

- *Discrete Value Replacement*: Replaces selected feature values with other values randomly selected from the entire set of discrete values.
- *Gaussian Noise Addition*: For continuous values, we randomly add noise following a Gaussian distribution $N(\mu, \sigma)$. μ and σ denote the mean and the SD of the distribution, respectively.
- *Uniform Noise Addition*: For continuous values, we randomly add noise following a uniform distribution within the interval $[-\delta, +\delta]$. δ determines the maximum range within which the noise values can vary, meaning that noise values will be uniformly distributed across the range from $-\delta$ to $+\delta$.
- *Multiplicative Noise Addition*: For continuous values, we randomly multiply the data by a factor drawn from a distribution, i.e., the Gaussian distribution $N(\mu, \sigma)$.

Regarding the fidelity assurance component, the GAN-based fidelity assurance technique is not suitable for tabular data, primarily due to its diverse data types (e.g., integers, decimals, categories), varying distribution shapes (e.g., multi-modal, non-Gaussian), and highly imbalanced categorical columns [27, 119, 120]. Therefore, we followed the approach proposed in existing work [119], modifying our method to utilize TVAE for fidelity assurance, to measure whether the generated test case is semantically similar to the original seed test case.

Furthermore, given the relatively modest sizes of our datasets compared to those employed in previous studies [126], we ran DRFuzz on each subject for a duration of 6 hours. By closely monitoring the trend of the fault detection curve, we observed that the curve flattened out, suggesting that the fuzzing process had reached a saturation point in 6 hours, which is sufficient for our analysis. To mitigate the potential impact of randomness, we repeated each experiment three times.

We conducted our experiments on a machine with Intel Xeon Gold 6338 CPU, 128 cores, 2.00 GHz, 512 GB RAM, running on Ubuntu 18.04.

3.4 Study Result Analysis

Upon analyzing the results, we identify several findings that can provide guidelines for future research as well as motivate our regression mitigation approach.

The overall results of the study are presented in Table 2, where Redundancy and FCC indicate selecting top-3/5 old features based on Formulas (2) and (3) for fuzzing, and random indicates randomly selecting 3 or 5 features for fuzzing. To avoid the influence of randomness, we repeated each approach three times. From Table 2, it can be observed that the RFI (Fuzzed) detected under the scenario of selecting top-3 and top-5 old features for fuzzing based on the Redundancy metric is, on average, 2.34 times and 2.48 times higher than the RFI (Fuzzed) detected under the random selection. Similarly, the RFI (Fuzzed) detected under the scenario of selecting top-3 and top-5 old features for fuzzing based on the FCC metric is, on average, 3.83 times and 2.88 times higher than the RFI generated under the random selection.

Table 2. Regression Fault Generation Results on Different Subjects

Dataset	#RFI (Test)	Measurement	Top-3			Top-5		
			#RFI (Fuzzed)	#Seed	#RF	#RFI (Fuzzed)	#Seed	#RF
Bean	17	Random	21,091	465	843	25,254	463	907
		Redundancy	80,915	513	981	55,489	571	1,072
		FCC	105,626	619	1,856	95,175	631	1,961
Fetal	24	Random	1,849	118	125	3,838	167	202
		Redundancy	7,437	170	216	15,860	290	362
		FCC	17,108	493	705	14,940	510	767
Music	31	Random	7,026	104	148	7,047	110	170
		Redundancy	13,142	122	199	13,470	131	239
		FCC	18,707	150	316	18,033	153	326
Churn	133	Random	1,680	458	458	2,437	645	645
		Redundancy	4,108	522	522	3,710	674	674
		FCC	4,590	917	917	5,363	969	969
Average	52	Random	3,518	227	244	4,441	307	339
		Redundancy	8,229	271	312	11,013	365	425
		FCC	13,468	520	646	12,779	544	687

We also analyzed the differences in distribution between randomly selecting features and based on FCC and redundancy scores. Table 3 presents the results. For each dataset, we had three subjects, and for each of the subjects, we repeated the fuzzing process three times. Therefore, for each metric, we obtained nine data points in total. We then conducted the Wilcoxon Rank-Sum Test under a confidence score of 0.05 to check whether selecting with FCC and redundancy scores for fuzzing are more likely to detect more regression-fault-triggering test inputs than selecting features randomly. Please note that this process poses a risk of a multiple comparison problem, as the chance of observing statistically significant results rises with the number of hypotheses tested. To address this, we followed existing work [18]. We adopted the Benjamini–Hochberg procedure [15] to correct p-values, which allows controlling the false rejection probability while retaining a high power of the tests. The results show that all the corrected p-values are less than 0.05.

To further confirm our observations, we performed the Cliff's Delta effect size measure [26] to investigate the magnitude of difference between random selection and selection based on FCC and redundancy scores for fuzzing. According to existing work [95], the magnitude of the difference is categorized as small, medium, and large when the effect size exceeds 0.15, 0.32, and 0.46, respectively. The results presented in Table 4 show that all effect size values exceed 0.46, indicating that the differences between random selection and selection based on FCC and redundancy scores for fuzzing are significant. *This shows that old features significantly influenced by the redundancy and contribution shift of the new features are more likely to induce regression faults.*

In terms of the diversity of regression fault-triggering test inputs detected, when selecting the top-3 old features based on the Redundancy metric, we found 312 different RFs covering 271 different seeds on average. Similarly, when selecting based on the FCC metric, we found 646 different RFs covering 520 different seeds. In contrast, random selection only detected 244 different RFs covering 227 different seeds. These findings suggest that redundancy and contribution shift not only lead to more RFs but also result in a wider variety of RFs. In addition, we analyzed the diversity overlap of these RFIs. *This observation further supports the finding that feature redundancy and contribution shift have different effects on old features, both contributing to regression faults.*

To further validate the results, we conducted a case study. Specifically, for each subject, we added one new feature along with the old features to train a new model. In this way, we acquired multiple regression models. Next, we calculated the RFI (Test), which represents the number of

Table 3. P-Values Corrected Using the Benjamini–Hochberg Procedure for Wilcoxon Rank-Sum Tests for Feature Selection for Fuzzing: FCC vs. Random and Redundancy vs. Random

Dataset	Measurement	Top-3			Top-5		
		#RFI (Fuzzed)	#RF	#Seed	#RFI (Fuzzed)	#RF	#Seed
Bean	FCC vs. Random	3.09×10^{-4}	3.09×10^{-4}	6.69×10^{-3}	3.72×10^{-3}	3.09×10^{-4}	3.09×10^{-4}
	Redundancy vs. Random	3.90×10^{-2}	4.67×10^{-2}	4.67×10^{-2}	3.90×10^{-2}	3.90×10^{-2}	2.36×10^{-3}
Fetal	FCC vs. Random	2.06×10^{-4}	2.06×10^{-4}	2.06×10^{-4}	2.06×10^{-4}	2.06×10^{-4}	2.06×10^{-4}
	Redundancy vs. Random	3.09×10^{-4}	4.02×10^{-3}	1.02×10^{-3}	3.09×10^{-4}	3.09×10^{-4}	3.09×10^{-4}
Music	FCC vs. Random	2.47×10^{-4}	2.47×10^{-4}	2.47×10^{-4}	2.87×10^{-4}	2.47×10^{-4}	2.47×10^{-4}
	Redundancy vs. Random	1.28×10^{-2}	2.04×10^{-2}	4.62×10^{-2}	1.28×10^{-2}	1.20×10^{-2}	1.20×10^{-2}
Churn	FCC vs. Random	6.18×10^{-4}	2.01×10^{-3}	2.01×10^{-3}	6.18×10^{-4}	2.60×10^{-2}	2.60×10^{-2}
	Redundancy vs. Random	1.24×10^{-3}	4.67×10^{-2}	4.67×10^{-2}	2.22×10^{-3}	4.67×10^{-2}	4.67×10^{-2}

Table 4. Effect Size of Different Feature Selection for Fuzzing: FCC vs. Random and Redundancy vs. Random

Dataset	Measurement	Top-3			Top-5		
		#RFI (Fuzzed)	#RF	#Seed	#RFI (Fuzzed)	#RF	#Seed
Bean	FCC vs. Random	1.00	1.00	0.70	0.78	1.00	1.00
	Redundancy vs. Random	0.56	0.48	0.48	0.58	0.58	0.95
Fetal	FCC vs. Random	1.00	1.00	1.00	1.00	1.00	1.00
	Redundancy vs. Random	1.00	0.75	0.89	1.00	1.00	1.00
Music	FCC vs. Random	1.00	1.00	1.00	0.98	1.00	1.00
	Redundancy vs. Random	0.68	0.60	0.48	0.70	0.78	0.75
Churn	FCC vs. Random	1.00	0.85	0.85	1.00	0.56	0.56
	Redundancy vs. Random	1.00	0.48	0.48	0.90	0.48	0.48

Table 5. Pearson Coefficients: #RFI (Test) Correlation with FCC and Redundancy

Subject	#RFI (Test) and Redundancy Score	#RFI (Test) and FCC Score
3	0.84	0.86
6	0.54	0.85
9	0.69	0.71
12	0.63	0.88

regression faults introduced on the test set, and investigated the correlation between RFI (Test) and redundancy, as well as the correlation between RFI (Test) and contribution shift. To achieve this, we calculated the average redundancy score (Formula (2)) between old features and the added new feature, and the average FCC score (Formula (3)) to measure the contribution shift. To conduct a comprehensive case study, we selected one subject from each of four different datasets (ID: 3, 6, 9, 12) for this study, specifically because these subjects have the most new features to be added among the subjects in their respective datasets, thus increasing the likelihood of obtaining accurate results for correlation analysis. The results are presented in Table 5. For example, in subject ID 12, the analysis reveals that in these regression models, the Pearson coefficients between the number of regression faults in these regression models and redundancy and contribution shift are 0.63 and 0.88, respectively. *This finding aligns with our previous experimental results, further substantiating that redundancy and contribution shift are related to regression faults.*

4 Approach

Based on the findings of the motivating study, regression faults are correlated with redundancy and contribution shift. The former tends to induce the DL model to over-rely on old features, while the latter induces the model to over-rely on new features. To address this issue and prevent the DL model from excessively relying on either new or old features, we propose FeaProtect, an approach for mitigating regression faults induced by feature evolution. FeaProtect consists of two main components: a pre-processing component, named redundancy-guided feature selection (detailed in Section 4.1), and an in-processing component, named contribution-controlled training (detailed in Section 4.2). The former aims to reduce redundancy between new and old features, thereby protecting the DL models from over-fitting to old features. The latter aims to ensure that the model learns new features while mitigating the forgetting of old features caused by contribution shift, thereby preserving the knowledge acquired from the old features. Both of these components are modular, allowing for their use independently or in combination. As our approach comprises pre- and in-processing components, it can seamlessly integrate with various post-processing approaches to further enhance the overall regression fault mitigation performance.

The workflow of FeaProtect is shown in Figure 1. To mitigate regression faults brought about by redundancy, a viable approach is to identify and remove redundant new features before incorporating them for training. For the pre-processing component, FeaProtect targets redundant new features. First, FeaProtect calculates the redundancy score between each new feature and the existing features. Then, it discretizes the new features and selects those that can enhance model performance while minimizing redundancy with old features. To mitigate regression faults caused by contribution shift, an effective strategy is to control the shifting during the training phase. In the in-processing component, following the combination of selected new features with old features and model initialization, FeaProtect utilizes a novel loss function to retain the contribution of old features throughout the training process.

4.1 Redundancy-Guided Feature Selection

To mitigate regression faults in DL models caused by redundant features, it is crucial to eliminate redundancy brought by new features. While some new features may exhibit a certain degree of redundancy with old features, the non-redundant information they contain may contribute to improving model performance. Therefore, it is necessary to balance the potential of introducing regression faults with the ability to improve model performance when deciding whether to retain a feature during the feature evolution. To tackle this challenge, we propose redundancy-guided feature selection, which consists of three steps: feature discretization, redundancy analysis, and feature selection, which are introduced as follows.

Feature Discretization. Feature selection is an effective method for minimizing redundancy brought by new features. Typically, it involves selecting both continuous and discrete features. However, selecting continuous features poses a great challenge due to their complexity since they usually have an infinite number of possible values, making it difficult to analyze their relationships [40]. While discrete features have a limited number of distinct values, resulting in less noise and making the analysis more straightforward. To address this issue, it is recommended to convert continuous features into discrete features, as this can better facilitate the subsequent feature selection process [16, 93, 105]. In this work, we employ the widely used feature discretization method called **Modified Discretization and Feature Selection (mDSM)** [93].

Redundancy Analysis. To reduce redundancy between new and old features while maintaining model performance, it is important to strike a balance. To achieve this, we conduct feature redundancy analysis to measure the overlap between each new feature and old features, as well

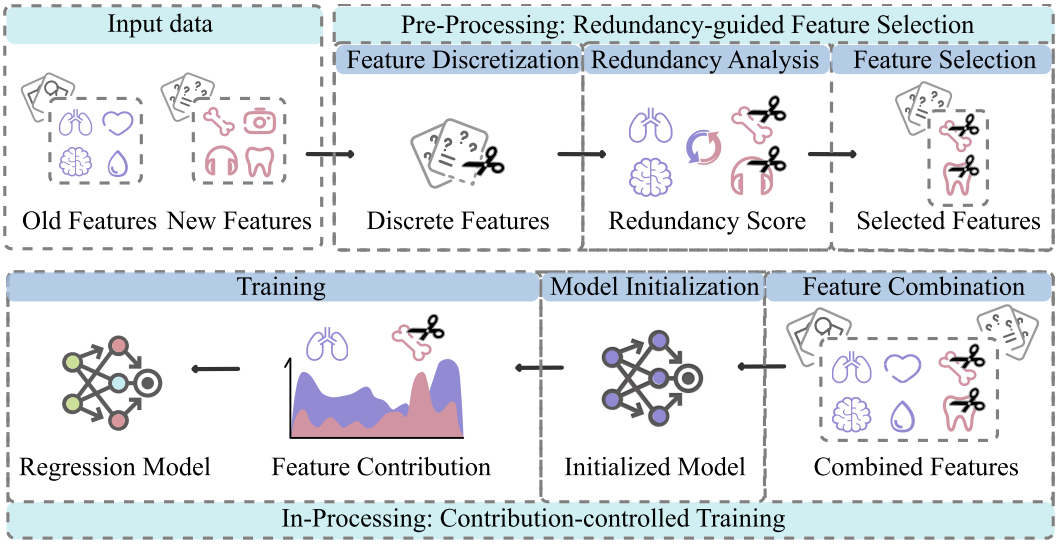


Fig. 1. The framework of FeaProtect.

as the potential improvement in model performance brought by each new feature. We then utilize this information to rank new features, which will be used in the feature selection process to enhance its efficiency. Specifically, we employ the previously mentioned MIC as the metric, which calculates the correlation between two features. When applied to two features, MIC reflects their level of redundancy [57]. When used to assess the correlation between a feature and the ground-truth label, it indicates the importance of that feature for model prediction [118]. To be more specific, we assign a score to each new feature based on Formula (4). Given a set of old features $\mathbb{F}^{\mathscr{O}} = \{f_1^{\mathscr{O}}, f_2^{\mathscr{O}}, \ldots, f_n^{\mathscr{O}}\}$, a set of new discrete features $\mathbb{D}^{\mathscr{N}} = \{d_1^{\mathscr{N}}, d_2^{\mathscr{N}}, \ldots, d_m^{\mathscr{N}}\}$ derived from new feature set $\mathbb{F}^{\mathscr{N}} = \{f_1^{\mathscr{N}}, f_2^{\mathscr{N}}, \ldots, f_m^{\mathscr{N}}\}$ after feature discretization, and the ground-truth label y, for a new feature $d_j^{\mathscr{N}}$ from $\mathbb{D}^{\mathscr{N}}$, the prior part of Formula (4) indicates its importance to model prediction, and the latter part indicates its average redundancy with the old features in $\mathbb{F}^{\mathscr{O}}$. We use the simplest form to integrate these two parts to optimize them simultaneously following existing works [14, 48, 81, 93].

$$Score(d_j^{\mathscr{N}}; \mathbb{F}^{\mathscr{O}}, y) = MIC(d_j^{\mathscr{N}}, y) - \frac{1}{n}\sum_{i=1}^{n} MIC(f_i^{\mathscr{O}}, d_j^{\mathscr{N}}). \tag{4}$$

Feature Selection. Feature selection is an important step in data analysis as it helps reduce feature redundancy, mitigate over-fitting, and improve model performance, which is suitable for our purpose. In this work, we adapt it for feature-addition model regression. Specifically, we use the greedy-based feature selection mDSM [93] since it has been demonstrated effective in many applications [41, 93]. The details are shown in Algorithm 1. Particularly, given the set of old features $\mathbb{F}^{\mathscr{O}}$, the set of discrete new features $\mathbb{D}^{\mathscr{N}}$, and the ground-truth label y. FeaProtect first conducts the initialization step (Line 1). Then, FeaProtect conducts feature analysis to score and rank each new feature for selection and saves it as *FeatureList* (Lines 2–5). After feature analysis, FeaProtect selects the feature with the highest score as the first feature in $\mathbb{D}^{\mathscr{S}}$ (Lines 6–7). As for selecting the subsequent features, FeaProtect tries the other features in the sorted feature ranking list. For each feature δ in the *FeatureList*, we measure whether bringing this new feature δ improves model performance or eliminates redundancy. Note that we also need to consider whether the feature δ is

Algorithm 1: Feature Analysis and Selection

Require: $\mathbb{F}^{\mathscr{O}}$: a set of old features $\mathbb{D}^{\mathscr{N}}$: a set of discrete new features y: the ground truth label
Ensure: $\mathbb{D}^{\mathscr{S}}$: the selected new features from $\mathbb{D}^{\mathscr{N}}$
1: $ScoreList, \mathbb{D}_S^{\mathscr{N}} \leftarrow \emptyset$ ▷ Initialization
2: **for** *each feature* d *in* $\mathbb{D}^{\mathscr{N}}$ **do**
3: $ScoreList.append(Score(d; y, \mathbb{F}^{\mathscr{O}}))$
4: **end for**
5: $FeatureList \leftarrow rank(\mathbb{D}^{\mathscr{N}}, ScoreList)$ ▷ Rank $\mathbb{D}^{\mathscr{N}}$ based on *ScoreList* in decreasing order
6: $\mathbb{D}^{\mathscr{S}} \leftarrow \mathbb{D}^{\mathscr{S}} \cup FeatureList[0]$
7: $FeatureList \leftarrow FeatureList \backslash FeatureList[0]$
8: **for** *each feature* δ *in FeatureList* **do**
9: $Gain \leftarrow GainScore(\delta, \mathbb{F}^{\mathscr{O}}, \mathbb{D}^{\mathscr{S}}, y)$ ▷ Calculating the gain score of δ based on Formula 5
10: **if** $Gain > T$ **then**
11: $T \leftarrow T + \epsilon; \mathbb{D}^{\mathscr{S}} \leftarrow \mathbb{D}^{\mathscr{S}} \cup \delta$
12: **end if**
13: $FeatureList \leftarrow FeatureList \backslash \delta$
14: **end for**
15: **return** $\mathbb{D}^{\mathscr{S}}$

also redundant with selected features in $\mathbb{D}^{\mathscr{S}}$. Therefore, suppose $\mathbb{D}^{\mathscr{S}}$ currently contains z features $\{d_1^{\mathscr{S}}, d_2^{\mathscr{S}}, \ldots, d_z^{\mathscr{S}}\}$, we calculate the gain score of adding each δ as Formula (5) following existing works [14, 48, 81, 93]. If the gain of adding a new feature δ is larger than the current threshold T, it is added in $\mathbb{D}^{\mathscr{S}}$ otherwise discarded (Lines 8–14). Since the gain is usually increasing while including a new feature, to avoid eventually including all features in $\mathbb{D}^{\mathscr{N}}$, we use the incremental gain strategy [93] to adjust the threshold T with a small value ϵ to select a small set of features that are highly informative.

$$Gain(\delta; \mathbb{F}^{\mathscr{O}}, \mathbb{D}^{\mathscr{S}}, y) = Score(\delta; \mathbb{F}^{\mathscr{O}}, y) - \frac{1}{z}\sum_{i=1}^{z} MIC(\delta, d_i^{\mathscr{S}}). \tag{5}$$

4.2 Contribution-Controlled Training

After the addition of new features, the importance of existing old features might change. Without appropriate management, the DL model may increasingly depend on the new features and disregard the contribution of knowledge acquired from the old features, leading to regression faults. Building upon this understanding, we propose a method called contribution-controlled training to tackle the problem.

To effectively control the influence of old features on model predictions and minimize changes during training, it is essential to incorporate a component in the loss function that controls the contribution. This requires the contribution calculation method to meet the following requirements: (1) High computational efficiency of the loss function to enable fast training with large-scale data and complex models [25], and (2) differentiability of the loss function for calculating gradients and updating model parameters [129]. Consequently, contribution calculation methods such as LIME [86] and Shapley Value [64] are excluded, and Expected Gradient [35] is chosen as the metric for contribution calculation. Expected Gradient has been widely applied and has demonstrated its effectiveness [32, 44, 111]. Additionally, our previous motivating study (Section 3.2) proves that this method is highly suitable for our task as it effectively measures contribution.

$$\mathscr{L}_{CC}(x; \mathcal{M}') = -\frac{EG_{\mathbb{F}^{\mathscr{O}}}(x; \mathcal{M}_1) \cdot EG_{\mathbb{F}^{\mathscr{O}}}(x; \mathcal{M}')}{||EG_{\mathbb{F}^{\mathscr{O}}}(x; \mathcal{M}_1)||\ ||EG_{\mathbb{F}^{\mathscr{O}}}(x; \mathcal{M}')||}. \tag{6}$$

We then design the loss function for contribution control (denoted as $\mathscr{L}_{CC}$) in Formula (6). Given the input x from the training set $\mathbb{T}$ and its corresponding ground-truth label y and its one-hot vector $\hat{y}$, the k-class original classification model $\mathcal{M}_1$, the model trained after the current iteration $\mathcal{M}'$, the output logits of the model $\mathcal{M}'$ on input x are denoted as $\hat{p}_{\mathcal{M}'}(x)$, $\hat{p}^{(q)}_{\mathcal{M}'}(x_i)$ and $y_i^{(q)}$ as the qth output logits of $\hat{p}_{\mathcal{M}'}(x)$ and $\hat{y}$, the old feature list $\mathbb{F}^{\mathscr{O}} = \{f_1^{\mathscr{O}}, f_2^{\mathscr{O}}, \ldots, f_n^{\mathscr{O}}\}$, we denote the Expected Gradients vector $EG_{\mathbb{F}^{\mathscr{O}}}(x; \mathcal{M}) = [EG_{f_1}(x; \mathcal{M}), EG_{f_2}(x; \mathcal{M}), \ldots, EG_{f_n}(x; \mathcal{M})]$, where each $EG_{f_1}(x; \mathcal{M})$ indicates the Expected Gradient of each old feature f_i of input x. In particular, we use the cosine similarity to guarantee the contribution of old features to the DL model prediction is maintained during the training process. Additionally, we still need to guarantee that the DL model can learn new knowledge with the constraint of controlling the contribution. We use the original cross-entropy loss function denoted as $\mathscr{L}_{CE}$ to ensure this, which is presented in Formula (7). To better balance the influence between these two components, we use the hyper-parameter λ to control them. Finally, we obtain the overall loss function as Formula (8).

$$\mathscr{L}_{CE}(x, y; \mathcal{M}') = -\sum_{q=1}^{k} \hat{y}^{(q)} \log[\hat{p}^{(q)}_{\mathcal{M}'}(x)] \tag{7}$$

$$\mathscr{L}(x, y, \mathcal{M}') = \lambda \mathscr{L}_{CC}(x; \mathcal{M}') + (1 - \lambda)\mathscr{L}_{CE}(x, y; \mathcal{M}'). \tag{8}$$

5 Evaluation

5.1 RQs

In order to evaluate the effectiveness of our regression mitigation approach, we have conducted an extensive experiment to answer the following RQs.

- *RQ1*: How effective is FeaProtect in mitigating regression faults compared with baseline approaches?
- *RQ2*: What is the contribution of each component in FeaProtect?
- *RQ3*: Can FeaProtect be combined with existing post-processing regression mitigation approaches?

5.2 Experimental Setup

5.2.1 Dataset. As presented in Table 1, in addition to the subjects we used in the motivating study (ID: 1–12), we also used the remaining subjects (ID: 13–28) to evaluate FeaProtect, which includes datasets "Classify Gestures by Reading Muscle Activity," "Patient Treatment Classification," "Mobile Price Classification," "Diabetes Disease Updated Datase," "Climate Data Daily IDN," and "Predictive Mutation Testing."

5.2.2 Metrics. During the evolution process, it is important to strike the balance between the overall ACC and the backward compatibility. Therefore, we select metrics from existing works [62, 117, 122], including one metric to measure ACC improvement and three metrics to measure the regression faults induced, which is related to backward compatibility.

After the regression process from $\mathcal{M}_1$ to $\mathcal{M}_2$, it is crucial to evaluate the performance of $\mathcal{M}_2$. Specifically, following existing work [102, 126], *ACC* [131] is employed as an important metric to quantify the model performance, as shown in Formula (9). Here, N represents the number of test samples, $p_{\mathcal{M}}(x_i)$ represents the prediction result of the model $\mathcal{M}$ on the given test input x_i, y_i represents the ground-truth label, and $\mathcal{I}(\cdot)$ denotes the indicator function, which returns 1 when the condition is true and 0 otherwise. *ACC* intuitively represents the fraction of test samples that are correctly predicted by the model. In this article, we evaluate the ACC of the model $\mathcal{M}_2$ as a

metric and denote it as $ACC_{\mathcal{M}_2}$.

$$ACC_{\mathcal{M}} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{I}(p_{\mathcal{M}}(x_i) = y_i). \tag{9}$$

To measure the effectiveness of regression mitigation, we utilize two widely used metrics from existing works [62, 117, 122]: the **Negative Flip Rate (NFR)** and the **Relative Negative Flip Rate ($\mathbf{NFR_{rel}}$)**. These metrics are employed to measure the regression faults induced through the regression process. *NFR* denotes the proportion of test inputs in the test set that trigger regression faults, which is presented in Formula (10). Since the error rate ($1 - ACC$) serves as an upper limit for *NFR*, comparing regression across models with different error rates becomes challenging [122]. Therefore, we adopt NFR_{rel} [122], with the denominator reflecting the expected error rate (i.e., $1 - ACC_{\mathcal{M}_2}$) on the subset of samples predicted correctly by the $\mathcal{M}_1$ (i.e., $ACC_{\mathcal{M}_1}$), to measure the likelihood of regression faults occurring on the test inputs correctly predicted by $\mathcal{M}_1$, as outlined in Formula (11), where $ACC_{\mathcal{M}_1}$ and $ACC_{\mathcal{M}_2}$ denote the accuracy of $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. A lower value for *NFR* and NFR_{rel} indicates fewer regression faults are induced during the regression process.

$$NFR = \frac{1}{N}\sum_{i=1}^{N} \mathcal{I}(p_{\mathcal{M}_1}(x_i) = y_i, p_{\mathcal{M}_2}(x_i) \neq y_i) \tag{10}$$

$$NFR_{rel} = \frac{NFR}{ACC_{\mathcal{M}_1}(1 - ACC_{\mathcal{M}_2})}. \tag{11}$$

We analyze how regression mitigation approaches handle regression-fault-triggering test inputs with different data distributions, in addition to the regression faults in the test set. To assess this, we use the regression fuzzing tool, DRFuzz, to generate regression-fault-triggering test inputs for the original version, $\mathcal{M}_1$, and its regression model $\mathcal{M}_2$, which is directly trained with all the new features without any other mitigating strategies. We then evaluate whether the models acquired by different mitigation approaches can correctly predict these inputs. A higher ACC in prediction indicates that these models are more robust to these generated inputs, and more regression faults are likely to be fixed. Following existing work [102, 126], the ACC on the generated test inputs is denoted as *Fix*. Since this metric can be seen as the ACC of the regression model on the generated inputs, it follows Formula (9). Specifically, we further analyze the contribution of the test inputs fixed by each mitigation approach. We split the generated test inputs into ten clusters according to their prediction confidence. We partition the interval of prediction confidence [0.5, 1] into 10 equal segments and cluster the test inputs accordingly. Then, we evaluate the fixing ACC on each cluster to assess the mitigation effectiveness across different CIs. This metric can reflect the fixing effectiveness of mitigation approaches on inputs with different confidence levels in a finer-grained way.

5.2.3 Implementation. We implement FeaProtect on Tensorflow 2.3.0. For the redundancy-guided feature selection, we use the recommended configuration proposed in existing work [93], initializing T as 0 and setting ϵ as 1e-5. For the contribution-controlled training, we set λ as 0.5 to balance the influence of two components in the loss function. Since the training in regression faults mitigating process usually involves randomness, we repeat each approach three times for the average results. The code and extra results can be found on our project homepage.

5.3 Compared Approaches

In this article, we propose a pre- and in-processing approach to mitigate regression faults. We remark that we are the first to target feature-level evolution and thus there is no prior work specific to this problem. To ensure a fair comparison, we select two state-of-the-art in-processing approaches for regression fault mitigation as baselines: finetuning [126] and NeuRecover [102]. Please note that these approaches are originally developed for data evolution scenarios, and can also be applied to feature evolution due to their flexibility.

You et al. [126] proposed a finetuning strategy to mitigate regression faults. They observed that by finetuning DL models using regression fault-triggering test inputs, the robustness of the model can be improved, and regression faults can be fixed to some extent. To prevent data leaks, we use DRFuzz to generate regression-fault-triggering test inputs on the validation set instead of the test set. Furthermore, we randomly select a subset of inputs, equivalent to 20% of the training set's size, for finetuning.

Tokui et al. [102] proposed NeuRecover, a fix-based approach, which is categorized as an in-processing method for repairing regression faults before model deployment. NeuRecover identifies weights responsible for regression faults and utilizes particle swarm optimization to search for suitable weights without compromising overall ACC. Since the specific number of candidate weights for repair is not mentioned in their paper, we considered four values (i.e., 1%, 5%, 10%, 20%) of this parameter for a preliminary evaluation on a sample of our benchmarks. We noticed that setting it as 1% or 5% would decrease the fixing effectiveness due to the limited search space, and setting it as 20% is likely to decrease the overall ACC due to the severe changes. Finally, we set it as 10% since it can better balance both the fixing effectiveness and the ACC drop.

In addition to the aforementioned two approaches, we also consider two natural baselines. One of them is referred to as "Direct Training," which simulates the straightforward approach of directly combining all the new features with the old ones to re-train a new model. Another one is "Feature Selection," which simulates the process of performing feature selection on the features after evolution and then training a new model.

In the literature [59], feature selection approaches in the machine learning field can be categorized into three categories: filter-based, wrapper-based, and embedded-based. Filter-based methods select features before the training process and are generally applicable to any learning algorithm. They are usually based on redundancy and importance analysis, utilizing the Chi2 score [72] or mutual information [33]. Wrapper-based methods involve iteratively re-training the machine learning algorithm on a subset of features to identify the subset that yields the best performance. They assess the utility of feature subsets, often utilizing generic algorithms [121]. However, this approach is not widely used due to the high cost of the re-training process and it is impractical to re-train the model multiple times in practice [36]. Embedded methods incorporate the task of feature selection into the training process, allowing the model to learn which features are most relevant during training [108]. Embedded-based approaches usually require the use of models with specific structures for training, such as XGBoost [22]. To fairly compare with the redundancy-guided feature selection in FeaProtect, which is filter-based, we select the state-of-the-art filter-based approach, CONMI_FS [39], for comparison.

5.4 Result Analysis

5.4.1 RQ1: Effectiveness Comparison. Overall Effectiveness. Given the large number of subjects used in our study and the limited space available, we put the detailed comparison results for each subject on our project homepage. Here, we present the overall results across all subjects in Table 6. In Table 6, Columns 2–5 present the number of subjects where each approach performs

Table 6. Overall Regression Faults Mitigation Results across Different Subjects

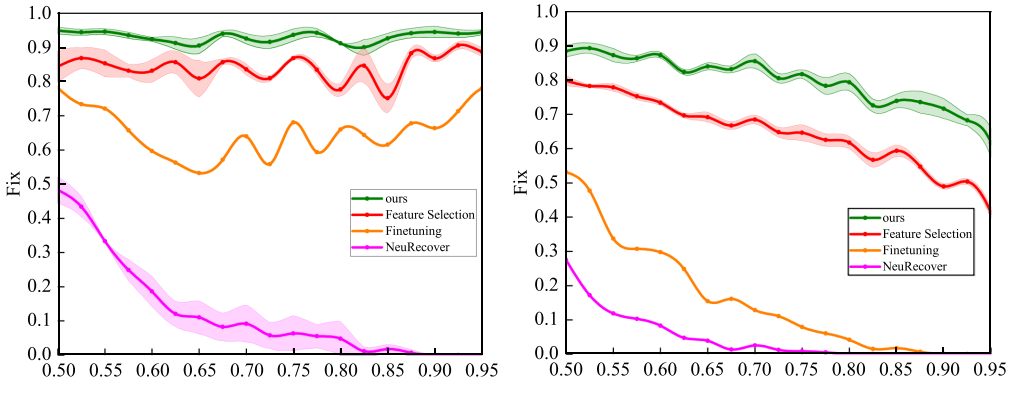
Approach	#Subjects				Average					Improvement			
	$ACC_{\mathcal{M}_2}$	*NFR*	NFR_{rel}	Fix	$ACC_{\mathcal{M}_1}$ (%)	$ACC_{\mathcal{M}_2}$ (%)	*NFR* (%)	NFR_{rel} (%)	Fix (%)	$ACC_{\mathcal{M}_2}$ (%)	*NFR* (%)	NFR_{rel} (%)	Fix (%)
Direct Training	2+2	0	0	-	77.92	80.30	6.19	45.48	-	1.53	47.87	42.59	-
Feature Selection	0+1	0	0	1	77.92	80.65	5.89	46.20	72.62	0.88	45.22	43.48	15.20
NeuRecover	1	0	0	0	77.92	78.71	7.39	50.86	25.60	3.58	56.36	48.66	227.07
Finetuning	2	0	0	0	77.92	79.48	6.53	46.32	52.44	2.58	50.58	43.62	59.65
FeaProtect	20+3	28	28	27	77.92	**81.53**	**3.23**	**26.11**	**83.72**	-	-	-	-

Please note that, in Column 2, on two subjects, the accuracy improvements of FeaProtect and direct training tie, and on one subject, the accuracy improvement of feature selection and FeaProtect ties. Therefore, we present them as 2+2, 0+1, and 20+3. Since we run each experiment multiple times, we calculate the Win/Tie/Loss counts of the Wilcoxon Rank-Sum Test (at a 5% significance level).
Please note that, in two subjects (Subjects 1 and 14), NeuRecover fails to work because it cannot find any regression fault-triggering inputs from the training set so that it cannot localize the neurons responsible for regression faults.
The bold values in Columns 6–9 indicate top performance in the corresponding metric.

the best in terms of each metric, Columns 6–10 present the average results across all the subjects in terms of each metric, and Columns 11–14 present the average improvement of FeaProtect over each compared approach in terms of each metric. Among the 28 subjects, FeaProtect achieves the best performance in terms of ACC improvement in 82.1% (23/28) of the cases. In contrast, feature selection, NeuRecover, and finetuning only perform best in 3.6% (1/28), 3.6% (1/28), and 7.1% (2/28) of the cases, respectively. Furthermore, FeaProtect outperforms the baseline approaches in terms of the *NFR* across all subjects, with improvements of 45.2%, 50.6%, and 56.4% compared to feature selection, finetuning, and NeuRecover, respectively. For NFR_{rel}, it demonstrates improvements of 43.5%, 48.7%, and 43.6% compared to feature selection, finetuning, and NeuRecover. *The results demonstrate the effectiveness of FeaProtect in mitigating regression faults without influencing ACC.*

Effectiveness on Mitigating Regression Faults in Different Distributions. We evaluated the mitigation of regression faults using the Fix metric. FeaProtect outperforms two in-processing approaches, NeuRecover and finetuning, by 227.1% and 59.7%, respectively, in fixing regression faults from different distributions. As shown in Subjects 11 and 21 in Figure 2, where the X axis represents the prediction confidence of the inputs on the regression model trained through direct training, and the Y axis represents the fixing ACC for these inputs, FeaProtect can stably fix regression faults with various ranges of prediction confidence compared to NeuRecover and finetuning. This is mainly because FeaProtect addresses the potential causes of regression faults and enhances model robustness. In contrast, NeuRecover is limited to fixing regression faults near the decision boundary, suggesting its performance may be hindered when encountering regression faults that deviate from the training data. To validate the superiority of FeaProtect, we conducted a statistical analysis. Since we have a total of 28 subjects for our study, for each of which we calculated and derived the average values for each metric through multiple repeated experiments, this means that for each evaluation metric, each approach corresponds to 28 data points, which are paired according to their IDs. Therefore, following existing work [19, 69, 94, 110], we use the Wilcoxon Signed-Rank Test [112] at a significance level of 0.05. The test was performed for each metric, and all the p-values were found to be smaller than 0.05. This indicates that FeaProtect significantly outperforms all the compared approaches in terms of all the metrics, as supported by statistical evidence. *This highlights the effectiveness of FeaProtect in fixing a wide range of regression faults.*

The Effectiveness of FeaProtect on Different Subjects. We then compare the performance of FeaProtect on the subjects used in the motivating study (ID:1–12) and the subjects specifically reserved for evaluating our regression faults mitigation approach (ID:13–28). Given the limited space available, we present the results for one representative subject from each group: one from the motivating study subjects (ID:1) and another from the evaluation subjects (ID:26). The results are presented in Table 7. Specifically, FeaProtect outperforms baseline methods in terms of ACC, achieving improvements ranging from 0.00% to 4.11% on Subject 1 and a comparable range of 3.11% to 7.99%



(a) ID:11 (b) ID:21

Fig. 2. Performance of fixing regression faults from different distributions.

Table 7. Regression Faults Mitigation Results on Motivating Study Subjects and Evaluating Subjects

Subjects	Approach	Average					Improvement			
		$ACC_{\mathcal{M}_1}$ (%)	$ACC_{\mathcal{M}_2}$ (%)	*NFR* (%)	NFR_{rel} (%)	Fix (%)	$ACC_{\mathcal{M}_2}$ (%)	*NFR* (%)	NFR_{rel} (%)	Fix (%)
1	Direct Training	88.67	**91.20**	2.13	27.34	-	0.00	30.99	31.13	-
	Feature Selection	88.67	90.27	2.00	23.17	47.41	1.03	26.50	18.73	81.05
	NeuRecover	88.67	-	-	-	-	-	-	-	-
	Finetuning	88.67	87.60	4.80	43.66	48.32	4.11	69.38	56.87	77.65
	FeaProtect	88.67	**91.20**	**1.47**	**18.83**	**85.84**	-	-	-	-
26	Direct Training	90.32	91.77	2.18	29.31	-	3.11	48.17	20.98	-
	Feature Selection	90.32	90.50	3.75	43.72	79.44	4.55	69.87	47.03	10.26
	NeuRecover	90.32	87.62	6.15	54.94	64.41	7.99	81.63	57.84	35.99
	Finetuning	90.32	90.99	2.12	26.01	46.28	3.99	46.70	10.96	89.26
	FeaProtect	90.32	**94.62**	**1.13**	**23.16**	**87.59**	-	-	-	-

The bold values in Columns 4–7 indicate top performance in corresponding metric.
In Subject 1, NeuRecover fails to work because it cannot find any regression fault-triggering inputs from the training set so that it cannot localize the neurons responsible for regression faults.

on evaluation Subject 26. Similarly, in terms of *NFR*, FeaProtect outperforms baselines by 30.99% to 69.38% on Subject 1 and 48.17% to 81.63% on Subject 26, demonstrating its stable effectiveness. Additionally, we conducted a Wilcoxon Signed-Rank Test at a significance level of 0.05 to evaluate the relative improvement of FeaProtect compared to each baseline in terms of each metric of the motivating study subjects and evaluation subjects. The results indicate that there is no significant difference in the performance of FeaProtect on motivating study subjects and evaluation subjects. The results demonstrate that FeaProtect outperforms baseline approaches across all metrics stably on all subjects.

5.4.2 RQ2: Ablation. FeaProtect comprises two key components: redundancy-guided feature selection and contribution-controlled training. Therefore, we conducted ablation experiments to measure the contribution of each component. Specifically, We compared FeaProtect with two variants of FeaProtect, i.e., FeaProtect$_{NoF}$ and FeaProtect$_{NoC}$, representing FeaProtect without the feature selection component and FeaProtect with the original loss function (as presented in Formula (7)) instead of contribution-controlled training.

Table 8 presents the average results of each variant across all subjects. As shown in Table 8, FeaProtect$_{NoF}$ mitigates 17.3% fewer regression faults compared with FeaProtect and performs

Table 8. Ablation Experimental Results

Approach	$ACC_{\mathcal{M}_2}$ (%)	*NFR* (%)	NFR_{rel} (%)	Fix (%)
Direct Training	80.30	6.19	45.48	-
FeaProtect	**81.53**	**3.23**	**26.11**	**83.72**
FeaProtect_{NoF}	81.29	3.79	30.17	81.61
FeaProtect_{NoC}	80.92	4.87	37.90	74.06

The bold indicates top performance in corresponding metric.

worse than FeaProtect by 15.5% in terms of NFR_{rel}. This highlights the importance of redundancy-guided feature selection as FeaProtect_{NoF} cannot eliminate redundancy and lead to regression faults. Similarly, FeaProtect_{NoC} mitigates 49.8% fewer *NFR* compared to FeaProtect and performs worse than FeaProtect by 50.8% in terms of NFR_{rel}. The reason is that FeaProtect_{NoC} cannot effectively protect the knowledge learned from old features, which causes regression faults. *Overall, the results confirm the contribution of both key components of FeaProtect.*

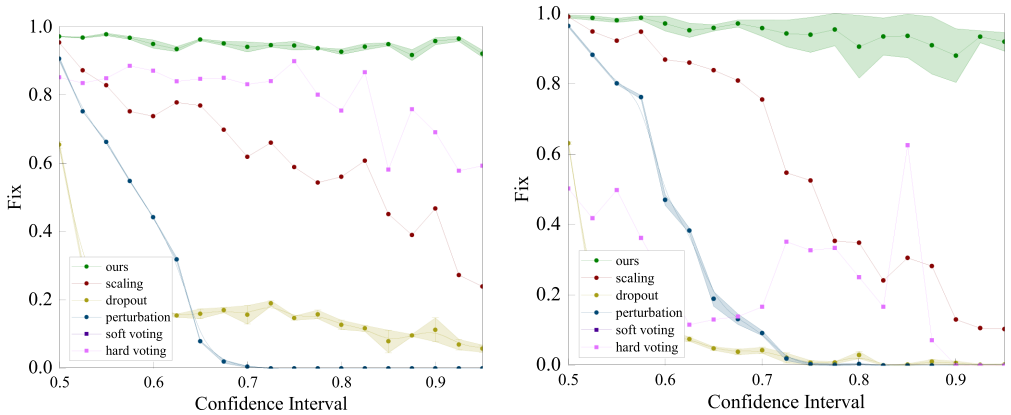
5.4.3 RQ3: Combination with Existing Approaches. We then aim to investigate the impact of integrating FeaProtect with post-processing approaches on enhancing overall mitigation effectiveness. We have selected state-of-the-art ensemble methods: voting-based [84] and uncertainty-based approaches [62]. Please note that, following the usage of existing approaches [62], we use the previous version of the model and its regression model as an ensemble for mitigation. Regarding voting-based approaches, we have applied soft voting and hard voting strategies. Soft voting averages the predicted confidences of both the original and regression models for the final output. Hard voting selects the class labels with the most votes, resorting to the more confident model in case of a tie. Uncertainty-based approaches align the predictions of the original and regression models based on their uncertainty levels before merging them for final predictions. These approaches are specifically designed for scenarios with limited data and unlabeled datasets. In limited data scenarios, Li et al. [62] add noise to the model (i.e., dropout) and inputs (i.e., perturbation) to estimate decision certainty, while for unlabeled datasets, they propose *temperature scaling* (i.e., Scaling) to align model predictions and reduce discrepancies between the two models.

FeaProtect vs. Post-Processing Approaches. Table 9 presents the results of FeaProtect and post-processing approaches. It shows that FeaProtect outperforms uncertainty-based approaches (i.e., dropout, perturbation, scaling) by 34.3%, 31.7%, and 15.8% in mitigating regression faults in terms of *NFR*. It also surpasses voting-based methods (i.e., soft and hard voting) by 2.2% and 4.6%, respectively. Moreover, FeaProtect shows ACC improvement over post-processing methods. Notably, post-processing approaches may struggle with regression faults from different distributions, particularly when faced with incorrectly predicted test inputs with high confidence (i.e., ≥ 0.90), as depicted in Figure 3, where the X axis represents the prediction confidence of the inputs on the regression model trained through direct training, and the Y axis represents the fixing ACC for these inputs. This is mainly because the ensemble model is susceptible to overconfident decisions from the base models [98]. In contrast, FeaProtect successfully rectifies over 80% of regression faults within this specific confidence range (as shown in subject ID 12 and 23). This is mainly because in-processing approaches such as FeaProtect can directly improve the robustness of the model against regression faults compared to post-processing approaches [50]. *Overall, FeaProtect achieves results comparable to post-processing approaches.*

Effectiveness of Integrated Approaches. As shown in Table 9, integrating FeaProtect with post-processing approaches leads to mitigating 52.6%–57.5% more regression faults in terms of *NFR*,

Table 9. Average Results across All Subjects of Post-Processing and Integrated Approaches

Approach	$ACC_{\mathcal{M}_2}$ (%)	NFR (%)	NFR_{rel} (%)	Fix (%)
Direct Training	80.30	6.19	45.48	-
FeaProtect	81.53	3.23	26.11	83.72
Dropout	80.52	5.13	37.90	39.77
Perturbation	80.64	4.95	37.11	51.02
Scaling	80.86	3.98	30.37	69.80
Soft Voting	80.72	3.30	25.50	75.01
Hard Voting	80.68	3.38	25.88	73.36
FeaProtect+Dropout	81.45	2.43	19.86	86.85
FeaProtect+Perturbation	81.37	2.32	18.69	88.03
FeaProtect+Scaling	80.79	1.69	12.92	91.14
FeaProtect+Soft Voting	80.53	1.45	10.66	92.67
FeaProtect+Hard Voting	80.49	1.47	10.77	92.56



(a) ID:12 (b) ID:23

Fig. 3. Performance of fixing regression faults from different distributions of post-processing approaches.

compared to using post-processing alone. *The results prove that integrating post-processing approaches with FeaProtect can enhance their performance.* This is because FeaProtect effectively reduces regression faults and enhances the prediction consistency between the base models (i.e., the original model and its regression model). With the support of ensemble approaches, the overall robustness of the ensemble model against regression faults can be further enhanced. However, integrating FeaProtect with the voting-based approach results in a slight decrease in ACC, averaging 0.2% lower than using the voting-based approach alone. This is mainly because the increased prediction consistency of the base models impacts the diversity and complementarity of predictions within the ensemble [115]. These findings suggest that *directly using voting-based approaches may not be optimal for mitigating regression faults*, highlighting the need for tailored post-processing approaches to balance ACC and regression fault mitigation.

6 Discussion

6.1 Generality of FeaProtect

Apart from the dataset used in this article, FeaProtect can also be extended to various modalities involving explicit feature evolution. For instance, in **Natural Language Processing (NLP)**, new sememes [65] and external knowledge (e.g., Wiki) [79] are incorporated into original sentences as new explicit features for reasoning. FeaProtect can potentially be applied to the above modality as its two main components are extendable. The redundancy-guided feature selection can be extended to text by measuring the redundancy between them (e.g., using SimCSE [37]). Similarly, contribution-controlled learning can be extended to additional modalities by leveraging Expected Gradients, a general DL explanation framework applicable to various complex models and modalities [35]. Therefore, our approach can be applied to more modalities.

In this article, following existing works [62, 102, 126], we mainly evaluate the performance of FeaProtect on classification tasks. However, FeaProtect can be extended to various tasks, including regression, content generation and recommendation tasks, as long as we define the concept of regression faults in these tasks. For example, for regression tasks,[1] we may define a test input triggering regression faults if it has lower MSE on the previous version and higher MSE in the latest version of the model. Furthermore, in this article, we mainly focus on mitigating regression faults caused by feature regression scenarios of DL systems. Therefore, we do not select regression scenarios such as supplementary training or adversarial training from existing work [126]. Since they are related to data-level evolution instead of feature-level evolution. Nevertheless, we conducted a preliminary experiment to demonstrate that FeaProtect can be extended to more regression scenarios by applying contribution-controlled training. For example, we constructed two regression scenarios, adversarial and supplementary training, based on the settings in DRFuzz [126] in the "Mobile Price Prediction" Dataset. The results demonstrate that our approach can reduce regression faults by 56.5% (i.e., 1.0% *NFR* using our approach while 2.3% *NFR* without mitigation approaches) in the adversarial training scenario and 16.7% (i.e., 2.5% *NFR* using our approach while 3.0% *NFR* without mitigation approaches) in the supplementary training scenario without harming the overall ACC. The results demonstrate the effectiveness of our approach, FeaProtect, in more regression scenarios.

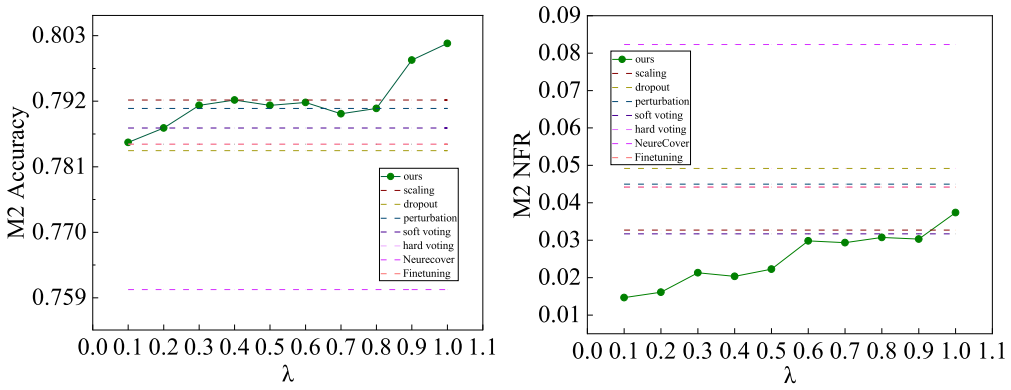
6.2 Efficiency of FeaProtect

We evaluate the efficiency of FeaProtect across all subjects, finding that the average time spent is 916 seconds. In contrast, finetuning requires only 5 seconds, while NeuRecover takes 966 seconds. While the time cost of FeaProtect is indeed longer, primarily due to the contribution-controlled training process, we note that it can be further optimized through parallel training. Despite being less efficient than finetuning, the cost of FeaProtect is acceptable compared to the lengthy debugging process for regression faults.

6.3 Parameter Influence

We investigate the impact of a key parameter, λ, in FeaProtect, which is used to balance the influence of learning new features and preserving the contribution of old features. For this experiment, we present the results in Figure 4 for subject ID 12 as it exhibits similar trends as other subjects. When λ decreases, it may slightly impact the ACC of $\mathcal{M}_2$, but enhance regression fault mitigation. The results suggest that the mitigation effectiveness under our default settings ($\lambda = 0.5$) is already generally satisfactory, yet developers can still optimize the configuration according to specific tasks.

[1] Please note that, here, the "regression" means the regression tasks of DL models, which is related to tasks such as classification and recommendation. In previous sections, the "regression" is related to the evolution process of DL systems.



(a) ID:12 $\mathcal{M}_2$ Acc. (b) ID:12 *NFR*

Fig. 4. Impact of λ in FeaProtect.

6.4 Implications

During the feature evolution process, many developers believe that directly combining old and new features and then re-training on these features to maximize the ACC of the model is already effective enough. However, it is essential to note that the metric of ACC emphasizes the overall performance of the test set, potentially overlooking the performance over specific test cases. However, in certain scenarios, the significance of particular test cases, which may be intricately linked to concerns regarding human safety, cannot be overlooked, as negligence could lead to severe outcomes. For example, In AI-assisted intelligent healthcare systems, any bugs introduced during evolution that lead to incorrect treatment decisions can have severe consequences. This could potentially result in medical incidents and expose the development team to legal or ethical issues. On the other hand, users of these DL systems develop specific usage patterns and trust in the model's judgments over long-term interactions, which is referred to as the mental model. If the model introduces regression faults during evolution, and users are unaware of these faults, it can significantly increase risks in the diagnostic process. Even if users are aware of the regression faults, adapting quickly to the new mental model can be challenging, and the adaptation process itself may introduce additional risks in disease diagnosis. Therefore, maintaining backward compatibility is crucial. *As the developers of DL systems, it is crucial to strike a balance between optimizing overall ACC and maintaining backward compatibility.*

In this article, we aim to introduce a novel development pattern for DL systems. Following this training pattern, we propose an approach to avoid over-relying on either new features or old features during feature evolution. Particularly, it includes eliminating the redundant new features (in a pre-processing component) while protecting the contribution of old features from being severely impacted when learning new features (in an in-processing component). By targeting the potential causes of regression faults, FeaProtect can ensure backward compatibility with minimal adverse impact on ACC. Experimental results indicate that FeaProtect, in comparison to direct training approaches, not only enhances ACC but also significantly mitigates regression faults, further boosting the backward compatibility of the DL systems. Furthermore, our approach can be integrated with post-processing approaches to achieve even greater effectiveness in mitigating regression faults, thereby supporting the high-quality evolution of DL systems.

6.5 Future Work

Based on the results presented in Tables 6 and 9, all approaches [102, 126], including uncertainty-based approaches [62], inherently introduce new regression faults during mitigation. Specifically,

although uncertainty-based approaches are claimed to avoid inducing new regression faults during the fixing process, we observed that the utilization of temperature scaling and variance estimation in these approaches might lead to the occurrence of regression faults to some degree. In this context, FeaProtect has demonstrated its ability to effectively reduce the number of regression faults while largely maintaining ACC, outperforming the baselines. In the future, we will delve deeper into how we can repair existing regression faults without introducing new ones. Our goal is to ensure the reliability and stability of our approach, further minimizing the impact of regression faults,

In our motivating study, we primarily study two potential causes of regression faults in feature evolution. However, upon eliminating these factors, we observed that a small number of regression faults still existed, suggesting a need for a more thorough exploration of potential causes that may lead to such regression faults in our future work.

6.6 Threats to Validity

The *internal* threat to validity mainly lies in the implementation of all the compared approaches, FeaProtect, experimental scripts, and multiple comparison problems in hypotheses tests in the motivating study. To reduce this threat, regarding the compared techniques, for DRFuzz [126] and uncertainty-based approaches [62], we adopted the implementations of the compared approaches released by the existing works. As for NeuRecover, its authors did not release their code, but they mentioned how they conducted their work based on Arachne [97], therefore, we carefully replicate their code according to their paper based on previous work [97]. Regarding the implementation on FeaProtect, to reduce this kind of threat from implementation, two authors have carefully checked all our code. Also, we have adopted some mature libraries to facilitate our implementation as presented in Section 5.2.3. To reduce the threat of multiple comparison problems in hypothesis tests, which we conducted using the Wilcoxon Rank-Sum Test in the motivating study, we adopt the Benjamini–Hochberg procedure to correct the p-values. The reasons for selecting the Benjamini–Hochberg procedure are twofold: (1) it is an alpha-correction method designed to deal with multiple hypothesis testing problems by controlling the false rejection probability while retaining a high power of the tests [15]; it is suitable for use when there are a large number of hypotheses. (2) it has proven its effectiveness in various works [18, 55] within the software engineering field.

The *external* threats to validity mainly lie in the subjects used in our study. To reduce this threat, we collected a large number of subjects with great diversity. In particular, we selected these subjects without any subjective bias and these subjects have diverse functionalities. In the future, we will evaluate FeaProtect on more subjects to further reduce this kind of threat.

The *construct* threats to validity mainly lie in the parameters in FeaProtect and the randomness. To reduce the threat from the parameters in FeaProtect, we presented the parameter settings in Section 5.2.3 and investigated the impact of the main parameters in Section 6.3. To reduce the threat of randomness, we repeated all the approaches involving randomness three times and calculated the average results in our study.

7 Related Work

7.1 Regression Testing of Traditional Software

The purpose of regression testing is to ensure that the software continues to function correctly and that changes made do not have any unintended side effects on the existing functionalities [125]. Current research in this field focuses on three main aspects: (1) enhancing the efficiency and effectiveness of regression testing [23, 109], (2) generating test cases that can detect more regression faults [75, 135], and (3) debugging and fixing regression faults [107, 128]. To improve the efficiency and effectiveness of regression testing, optimization techniques such as test case prioritization,

selection, and reduction are commonly employed [125]. These techniques aim to reduce the cost, effort, and time required for regression testing. For example, Wang et al. [109] proposed a quality-aware test prioritization approach based on the fault proneness of test cases, and Chen et al. [23] used log information to facilitate the test prioritization process. In terms of generating test cases to detect more regression faults, one approach is to employ fuzzing techniques on software system changes, thereby amplifying the impact of these changes and detecting regression bugs induced during the regression process. Zhu and Böhme [135] proposed a grey-box regression fuzzing approach that incorporates code changes, and Noller [75] utilized differential analysis and symbolic execution to further detect regression faults. In terms of debugging and fixing regression faults, it is common to identify code changes for analysis. Yu et al. [128] utilized coverage analysis and delta debugging techniques to automatically isolate failure-inducing changes, while Wang et al. [107] proposed the use of alignment slicing to isolate changes and explain failures. In contrast to the aforementioned works, our research focuses specifically on regression faults in DL systems. These faults are challenging to localize and explain compared to traditional software due to the statistical nature of DL systems.

7.2 Regression Faults and Its Mitigation in DL Systems

In DL systems, regression faults have not received widespread attention. You et al. [126] observed that regression faults occur in various DL regression processes, such as supplementary and adversarial training. To address this issue, they developed a fuzzing tool for detecting regression faults. Given the potential severity of regression faults, several approaches have been proposed to mitigate them. Yan et al. [122] introduced focal distillation to reduce regression faults. Xie et al. [117] found that knowledge distillation is effective in preventing regression faults in NLP models. You et al. [126] proposed finetuning on regression-fault-triggering test inputs to improve model robustness. However, training-based methods often result in decreased ACC. NeuRecover [102] is a fixing-based approach that identifies weights responsible for regression faults, but it may only work well on faults similar to those in the training set. Li et al. [62] proposed a post-processing method that combines results from the original and regression models, but it may produce incorrect predictions for faults with significant output differences. Different from these works, our study focuses on mitigating regression faults in regression scenarios involving feature evolution.

8 Conclusion

In this article, we conducted a study to explore the potential causes of regression faults brought by feature evolution in DL systems, i.e., redundancy and contribution shift. To mitigate their impact, we propose FeaProtect, which incorporates redundancy-guided feature selection, and contribution-controlled training, to reduce regression faults. To evaluate the effectiveness of FeaProtect, we compare it with two state-of-the-art in-processing regression fault mitigation approaches across 28 subjects. The results demonstrate that FeaProtect outperforms the compared approaches in terms of regression fault mitigation across all scenarios. Furthermore, we integrate FeaProtect with post-processing approaches and observe improved effectiveness in mitigating regression faults.

Acknowledgments

We thank the editors and anonymous reviewers for their constructive suggestions to help improve the quality of this article. We specifically thank Xiaofei Sun, Jin Xiao, and Weile Na for helping to provide crucial insights and information regarding feature evolution in industrial practice.

References

[1] Sparsh Gupta. 2024. Models: ANN for music genre classification. Retrieved from https://www.kaggle.com/code/imsparsh/gtzan-genre-classification-deep-learning-val-92-4

[2] SElab2019. 2024. Models: CNN for predictive mutation testing. Retrieved from https://github.com/SElab2019/ExtPMT

[3] Youssef Elzahar. 2024. Models: FFNN for diabetes disease updated dataset. Retrieved from https://www.kaggle.com/code/youssefelzahar/diabetes-disease

[4] Ali Greo. 2024. Models: FFNN for dry beans classification. Retrieved from https://www.kaggle.com/code/aligreualihassan/dry-beans-classification

[5] Dewan Mahmuda Zaman. 2024. Models: FFNN for fetal health classification. Retrieved from https://www.kaggle.com/code/mahmudazaman/dnn-multiclass-classification-93-accuracy

[6] Oliveira and Henrique Silva. 2024. Models: FFNN for hand gesture prediction. Retrieved from https://www.kaggle.com/code/caldasdeoliveira/hand-gesture-prediction-emg-dense-nn

[7] Ahmed Ali Omar, Haneen Hossam, and Shady Nagy. 2024. Models: FFNN for mobile price prediction. Retrieved from https://www.kaggle.com/code/ahmedaliomar/mobile-price-classification-neural-network-96-acc

[8] Nanzhong-tju. 2024. Models: FFNN for patient treatment prediction. Retrieved from https://github.com/nanzhong-tju/DNN-model-for-patient-treatment-analysis

[9] Pavan Kumar D. 2024. Models: FFNN for telco customer churn. Retrieved from https://www.kaggle.com/code/mragpavank/predicting-customer-churn-for-a-telecom-company

[10] Hadeer Ahmed, Issa Traoré, and Sherif Saad. 2017. Detection of online fake news using n-Gram analysis and machine learning techniques. In *ISDDC*, Lecture Notes in Computer Science, Vol. 10618, Springer, 127–138.

[11] Francesco Altiero, Anna Corazza, Sergio Di Martino, Adriano Peron, and Luigi L. L. Starace. 2022. ReCover: A curated dataset for regression testing research. In *MSR*. ACM, 196–200.

[12] Diogo Ayres-de Campos, Joao Bernardes, Antonio Garrido, Joaquim Marques-de Sa, and Luis Pereira-Leite. 2000. SisPorto 2.0: A program for automated analysis of cardiotocograms. *J. Matern.-Fetal Med.* 9, 5 (2000), 311–318.

[13] Gagan Bansal, Besmira Nushi, Ece Kamar, Daniel S. Weld, Walter S. Lasecki, and Eric Horvitz. 2019. Updates in human-AI teams: Understanding and addressing the performance/compatibility tradeoff. In *AAAI*. AAAI Press, 2429–2437.

[14] Roberto Battiti. 1994. Using mutual information for selecting features in supervised neural Net learning. *IEEE Trans. Neural Netw.* 5, 4 (1994), 537–550.

[15] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J. R. Stat. Soc. Series B (Methodol.)* 57, 1 (1995), 289–300.

[16] Veronica Bolon-Canedo, Noelia Sanchez-Marono, and Amparo Alonso-Betanzos. 2011. Feature selection and classification in multiple class datasets: An application to KDD cup 99 dataset. *Expert Syst. Appl.* 38, 5 (2011), 5947–5957.

[17] Declan Butler. 2013. When Google Got flu wrong: US outbreak foxes a leading web-based method for tracking seasonal flu. *Nature* 494, 7436 (2013), 155–157.

[18] Gabriella Carrozza, Domenico Cotroneo, Roberto Natella, Roberto Pietrantuono, and Stefano Russo. 2013. Analysis and prediction of mandelbugs in an industrial software system. In *ICST*. IEEE Computer Society, 262–271.

[19] Junjie Chen, Zhuo Wu, Zan Wang, Hanmo You, Lingming Zhang, and Ming Yan. 2020. Practical accuracy estimation for efficient deep neural network testing. *ACM Trans. Softw. Eng. Methodol.* 29, 4 (2020), 30:1–30:35.

[20] Richard J. Chen, Ming Y. Lu, Jingwen Wang, Drew F. K. Williamson, Scott J. Rodig, Neal I. Lindeman, and Faisal Mahmood. 2022. Pathomic fusion: An integrated framework for fusing histopathology and genomic features for cancer diagnosis and prognosis. *IEEE Trans. Medical Imaging* 41, 4 (2022), 757–770.

[21] Suming J. Chen, Zhen Qin, Zac Wilson, Brian Calaci, Michael Rose, Ryan Evans, Sean Abraham, Donald Metzler, Sandeep Tata, and Mike Colagrosso. 2020. Improving recommendation quality in Google drive. In *KDD*. ACM, 2900–2908.

[22] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *KDD*. ACM, 785–794.

[23] Zhichao Chen, Junjie Chen, Weijing Wang, Jianyi Zhou, Meng Wang, Xiang Chen, Shan Zhou, and Jianmin Wang. 2023. Exploring better black-box test case prioritization via log analysis. *ACM Trans. Softw. Eng. Methodol.* 32, 3 (2023), 72:1–72:32.

[24] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *DLRS@RecSys*. ACM, 7–10.

[25] Lorenzo Ciampiconi, Adam Elwood, Marco Leonardi, Ashraf Mohamed, and Alessandro Rozza. 2023. A survey and taxonomy of loss functions in machine learning. arXiv:2301.05579. DOI: https://doi.org/10.48550/arXiv.2301.05579

[26] Norman Cliff. 1993. Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychol. Bull.* 114, 3 (1993), 494.

[27] Edouard Couplet, John Aldo Lee, and Michel Verleysen. 2021. *Tabular Data Synthesis Using Generative Adversarial Networks: An Application to Table Augmentation*. Master's thesis. UCLouvain.
[28] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for YouTube recommendations. In *RecSys*. ACM, 191–198.
[29] Brian Dean. Accessed: 2024. Google's 200 ranking factors: The complete list. Retrieved from https://backlinko.com/google-ranking-factors
[30] Wejdan Deebani and Nezamoddin Nezamoddini-Kachouie. 2022. Monte Carlo ensemble correlation coefficient for association detection. *Commun. Stat. Simul. Comput.* 51, 12 (2022), 7095–7109.
[31] Samet Demir, Hasan Ferit Eniser, and Alper Sen. 2020. DeepSmartFuzzer: Reward guided Test generation for deep learning. In *AISafety@IJCAI*, CEUR Workshop Proceedings, Vol. 2640, CEUR-WS.org, 134–140.
[32] Huiqi Deng, Na Zou, Mengnan Du, Weifu Chen, Guocan Feng, Ziwei Yang, Zheyang Li, and Quanshi Zhang. 2023. Understanding and unifying fourteen attribution methods with Taylor interactions. arXiv:2303.01506. DOI: https://doi.org/10.48550/arXiv.2303.01506
[33] Samrat Kumar Dey, Khandaker Mohammad Mohi Uddin, Hafiz Md. Hasan Babu, Md. Mahbubur Rahman, Arpita Howlader, and K. M. Aslam Uddin. 2022. Chi^2-MI: A hybrid feature selection based machine learning approach in diagnosis of chronic kidney disease. *Intell. Syst. Appl.* 16 (2022), 200144.
[34] Seda İpek Aksoy Ercan and Murat ŞİMŞEK. 2023. Mobile phone price classification using machine learning. *Int. J. Adv. Nat. Sci. Eng. Res.* 7, 4 (2023), 458–462.
[35] Gabriel G. Erion, Joseph D. Janizek, Pascal Sturmfels, Scott M. Lundberg, and Su-In Lee. 2021. Improving performance of deep learning models with axiomatic attribution priors and expected gradients. *Nat. Mach. Intell.* 3, 7 (2021), 620–631.
[36] Raquel Espinosa, Fernando Jiménez, and José Palma. 2024. Surrogate-assisted and filter-based multiobjective evolutionary feature selection for deep learning. *IEEE Trans. Neural Networks Learn. Syst.* 35, 7 (2024), 9591–9605.
[37] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *EMNLP (1)*. Association for Computational Linguistics, 6894–6910.
[38] Lester Darryl Geneviève, Andrea Martani, Tenzin Wangmo, Daniela Paolotti, Carl Koppeschaar, Charlotte Kjelsø, Caroline Guerrisi, Marco Hirsch, Olivia Woolley-Meza, Paul Lukowicz, et al. 2019. Participatory disease surveillance systems: Ethical framework. *J. Medi. Internet Res.* 21, 5 (2019), e12273.
[39] Huanhuan Gong, Yanying Li, Jiaoni Zhang, Baoshuang Zhang, and Xialin Wang. 2024. A new filter feature selection algorithm for classification task by ensembling Pearson correlation coefficient and mutual information. *Eng. Appl. Artif. Intell.* 131 (2024), 107865.
[40] Yonghao Gu, Kaiyue Li, Zhenyang Guo, and Yongfei Wang. 2019. Semi-supervised k-means DDoS detection method using hybrid feature selection algorithm. *IEEE Access* 7 (2019), 64351–64365.
[41] Md Nazmul Haque, Sadia Sharmin, Amin Ahsan Ali, Abu Ashfaqur Sajib, and Mohammad Shoyaib. 2021. Use of relevancy and complementary information for discriminatory gene selection from high-dimensional gene expression data. *PLoS One* 16, 10 (2021), e0230164.
[42] Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. In *ACL/IJCNLP (1)*. Association for Computational Linguistics, 2208–2222.
[43] Jeff Heaton. 2016. An empirical analysis of feature engineering for predictive modeling. In *SoutheastCon2016*. IEEE, 1–6.
[44] Robin Hesse, Simone Schaub-Meyer, and Stefan Roth. 2021. Fast axiomatic attribution for neural networks. In *NeurIPS*, 19513–19524.
[45] Bing Huang, Feng Yang, Meng-Xiao Yin, Xiaoying Mo, and Cheng Zhong. 2020. A review of multimodal medical image fusion techniques. *Comput. Math. Methods Med.* 2020 (2020), 8279342:1–8279342:16.
[46] Lan Huang, Xuemei Hu, Yan Wang, and Yuan Fu. 2022. EGFAFS: A novel feature selection algorithm based on explosion gravitation field algorithm. *Entropy* 24, 7 (2022), 873.
[47] Zhenfei Huang, Junjie Chen, Jiajun Jiang, Yihua Liang, Hanmo You, and Fengjie Li. 2024. Mapping APIs in dynamic-typed programs by leveraging transfer learning. *ACM Trans. Softw. Eng. Methodol.* 33, 4 (2024), 102:1–102:29.
[48] Insik Jo, Sangbum Lee, and Sejong Oh. 2019. Improved measures of redundancy and relevance for mRMR feature selection. *Computers* 8, 2 (2019), 42.
[49] Leonid Joffe. 2021. Transfer learning for tabular data. TechRxiv (2021). DOI: https://doi.org/10.36227/techrxiv.16974124.v
[50] Archit Karandikar, Nicholas Cain, Dustin Tran, Balaji Lakshminarayanan, Jonathon Shlens, Michael C. Mozer, and Becca Roelofs. 2021. Soft calibration objectives for neural networks. In *NeurIPS*, 29768–29779.
[51] Alexandr Katrutsa and Vadim V. Strijov. 2017. Comprehensive study of feature selection methods to solve multicollinearity problem according to evaluation criteria. *Expert Syst. Appl.* 76 (2017), 1–11.

[52] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. 2014. A survey of feature selection and feature extraction techniques in machine learning. In *SAI*. IEEE, 372–378.
[53] Klim Kireev, Bogdan Kulynych, and Carmela Troncoso. 2023. Adversarial robustness for tabular data through cost and utility awareness. In *NDSS*. The Internet Society, 1–18.
[54] Murat Koklu and Ilker Ali Ozkan. 2020. Multiclass classification of dry beans using computer vision and machine learning techniques. *Comput. Electron. Agric.* 174 (2020), 105507.
[55] Tien-Duy B. Le and David Lo. 2015. Beyond support and confidence: Exploring interestingness measures for rule-based specification mining. In *SANER*. IEEE Computer Society, 331–340.
[56] Steven Levy. 2024. Exclusive: How Google's algorithm rules the web. Retrieved from https://www.wired.com/2010/02/ff-google-algorithm/
[57] Gai-Ling Li, Zhi-Jie Zhou, Changhua Hu, Lei-Lei Chang, Hongtao Zhang, and Chuanqiang Yu. 2019. An optimal safety assessment model for complex systems considering correlation and redundancy. *Int. J. Approx. Reason.* 104 (2019), 38–56.
[58] Haidong Li, Jiongcheng Li, Xiaoming Guan, Binghao Liang, Yuting Lai, and Xinglong Luo. 2019. Research on overfitting of deep learning. In *CIS*. IEEE, 78–81.
[59] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. 2018. Feature selection: A data perspective. *ACM Comput. Surv.* 50, 6 (2018), 94:1–94:45.
[60] Lei Li, Yabin Wu, Yihang Ou, Qi Li, Yanquan Zhou, and Daoxin Chen. 2017. Research on machine learning algorithms and feature extraction for time series. In *PIMRC*. IEEE, 1–5.
[61] Mengxuan Li, Peng Peng, Haiyue Sun, Min Wang, and Hongwei Wang. 2024. An order-invariant and interpretable dilated convolution neural network for chemical process fault detection and diagnosis. *IEEE Trans Autom. Sci. Eng.* 21, 3 (2024), 3933–3943.
[62] Zenan Li, Maorun Zhang, Jingwei Xu, Yuan Yao, Chun Cao, Taolue Chen, Xiaoxing Ma, and Jian Lu. 2023. Lightweight approaches to DNN regression error reduction: An uncertainty alignment perspective. In *ICSE*. IEEE, 1187–1199.
[63] Zifan Liu, Zhechun Zhou, and Theodoros Rekatsinas. 2022. Picket: Guarding against corrupted data in tabular data during learning and inference. *VLDB J.* 31, 5 (2022), 927–955.
[64] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *NIPS*, 4765–4774.
[65] Ling Luo, Xiang Ao, Yan Song, Jinyao Li, Xiaopeng Yang, Qing He, and Dong Yu. 2019. Unsupervised neural aspect extraction with sememes. In *IJCAI*. ijcai.org, 5123–5129.
[66] Dongyu Mao, Lingchao Chen, and Lingming Zhang. 2019. An extensive study on cross-project predictive mutation testing. In *ICST*. IEEE, 160–171.
[67] Alibaba Cloud MaxCompute. 2024. The power of AI: Why taobao knows online shoppers better Than they know themselves. Retrieved from https://tinyurl.com/ailibaba
[68] Dipti Mishra, Satish Kumar Singh, and Rajat Kumar Singh. 2022. Deep CNN based image compression with redundancy minimization via attention guidance. *Neurocomputing* 507 (2022), 397–411.
[69] Rebecca Moussa and Federica Sarro. 2022. On the use of evaluation measures for defect prediction studies. In *ISSTA*. ACM, 101–113.
[70] Maximilian Muschalik, Fabian Fumagalli, Barbara Hammer, and Eyke Hüllermeier. 2022. Agnostic explanation of model change based on feature importance. *Künstliche Intell.* 36, 3 (2022), 211–224.
[71] Agastya Nanda, Senthil Mani, Saurabh Sinha, Mary Jean Harrold, and Alessandro Orso. 2011. Regression testing in the presence of non-code changes. In *ICST*. IEEE Computer Society, 21–30.
[72] Mohamed Nassar, Haïdar Safa, Alaa Al Mutawa, Ahmed Helal, and Iskander Gaba. 2019. Chi squared feature selection over Apache spark. In *IDEAS*. ACM, 41:1–41:5.
[73] Zhenyuan Ning, Denghui Du, Chao Tu, Qianjin Feng, and Yu Zhang. 2022. Relation-aware shared representation learning for cancer prognosis analysis with auxiliary clinical variables and incomplete multi-modality data. *IEEE Trans. Med. Imaging* 41, 1 (2022), 186–198.
[74] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. 2022. Efficient test-time model adaptation without forgetting. In *ICML*, Proceedings of Machine Learning Research, Vol. 162, PMLR, 16888–16905.
[75] Yannic Noller. 2018. Differential program analysis with fuzzing and symbolic execution. In *ASE*. ACM, 944–947.
[76] Don Norman. 2013. *The Design of Everyday Things: Revised and Expanded Edition*. Basic books.
[77] Sejong Oh. 2019. Feature interaction in terms of prediction performance. *Appl. Sci.* 9, 23 (2019), 5191.
[78] Guillermo Ortiz Jimenez, Apostolos Modas, Seyed Mohsen Moosavi Dezfooli, and Pascal Frossard. 2020. Redundant features can hurt robustness to distributions shift. In *Uncertainty & Robustness in Deep Learning Workshop*. ICML, 1–8.
[79] Prasanna Parthasarathi and Joelle Pineau. 2018. Extending neural generative conversational model using external knowledge sources. In *EMNLP*. Association for Computational Linguistics, 690–695.

[80] Fabrizio Pastore and Leonardo Mariani. 2017. VART: A tool for the automatic detection of regression faults. In *ESEC/SIGSOFT FSE*. ACM, 964–968.
[81] Hanchuan Peng, Fuhui Long, and Chris H. Q. Ding. 2005. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.* 27, 8 (2005), 1226–1238.
[82] Calton Pu, Abhijit Suprem, Rodrigo Alves Lima, Aibek Musaev, De Wang, Danesh Irani, Steve Webb, and Joao Eduardo Ferreira. 2020. Beyond artificial reality: Finding and monitoring live events from social sensors. *ACM Trans. Internet Tech.* 20, 1 (2020), 1–21.
[83] Masoumeh Rahimi, Alireza Alghassi, Mominul Ahsan, and Julfikar Haider. 2020. Deep learning model for industrial leakage detection using acoustic emission signal. *Informatics* 7, 4 (2020), 49.
[84] Md Raihan-Al-Masud and Hossen Asiful Mustafa. 2019. Network intrusion detection system using voting ensemble machine learning. In *ICTP*. IEEE, 1–4.
[85] David N. Reshef, Yakir A. Reshef, Hilary K. Finucane, Sharon R. Grossman, Gilean McVean, Peter J. Turnbaugh, Eric S. Lander, Michael Mitzenmacher, and Pardis C. Sabeti. 2011. Detecting novel associations in large data sets. *Science* 334, 6062 (2011), 1518–1524.
[86] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?": Explaining the predictions of any classifier. In *KDD*. ACM, 1135–1144.
[87] Raquel Rodríguez-Pérez and Jürgen Bajorath. 2020. Interpretation of machine learning models using Shapley values: Application to compound potency and multi-target activity predictions. *J. Comput. Aided Mol. Des.* 34, 10 (2020), 1013–1026.
[88] Mujiono Sadikin, Ida Nurhaida, and Ria Puspita Sari. 2021. Exploratory study of some machine learning techniques to classify the patient treatment. *Int. J. Adv. Comput. Sci. Appl.* 12, 2 (2021), 380–387.
[89] Anggraini Puspita Sari, Hiroshi Suzuki, Takahiro Kitajima, Takashi Yasuno, Dwi Arman Prasetya, and Nachrowie Nachrowie. 2020. Prediction model of wind Speed and direction using deep neural network. *J. Electr. Eng. Mechatron. Comput. Sci.* 3, 1 (2020), 1–10.
[90] Simone Scardapane and Dianhui Wang. 2017. Randomness in neural networks: An overview. *WIREs Data Mining Knowl. Discov.* 7, 2 (2017).
[91] Claude E. Shannon. 1948. A mathematical theory of communication. *Bell Syst. Tech. J.* 27, 3 (1948), 379–423.
[92] Amanda J. C. Sharkey. 1999. Treating harmful collinearity in neural network ensembles. In *Combining artificial neural nets: Ensemble and modular multi-net systems*. Springer, 101–125.
[93] Sadia Sharmin, Mohammad Shoyaib, Amin Ahsan Ali, Muhammad Asif Hossain Khan, and Oksam Chae. 2019. Simultaneous feature selection and discretization based on mutual information. *Pattern Recognit.* 91 (2019), 162–174.
[94] August Shi, Peiyuan Zhao, and Darko Marinov. 2019. Understanding and improving regression test selection in continuous integration. In *ISSRE*. IEEE, 228–238.
[95] Xiu-Yu Shi, Jun Ju, Qian Lu, Lin-Yan Hu, Ya-Ping Tian, Guang-Hong Guo, Zhi-Sheng Liu, Ge-Fei Wu, Hong-Min Zhu, Yu-Qin Zhang, et al. 2023. Both epilepsy and anti-seizure medications affect bone metabolism in children with self-limited epilepsy with centrotemporal spikes. *Epilepsia* 64, 10 (2023), 2667–2678.
[96] Ravid Shwartz-Ziv and Amitai Armon. 2022. Tabular data: Deep learning is not all you need. *Inf. Fusion* 81 (2022), 84–90.
[97] Jeongju Sohn, Sungmin Kang, and Shin Yoo. 2023. Arachne: Search-based repair of deep neural networks. *ACM Trans. Softw. Eng. Methodol.* 32, 4 (2023), 85:1–85:26.
[98] Cedrique Rovile Njieutcheu Tassi, Jakob Gawlikowski, Auliya Unnisa Fitri, and Rudolph Triebel. 2022. The impact of averaging logits over probabilities on ensembles of neural networks. In *AISafety@IJCAI*, CEUR Workshop Proceedings, Vol. 3215, CEUR-WS.org, 132–141.
[99] IBM Samples Team. 2024. Telco customer churn (11.1.3+). Retrieved from https://community.ibm.com/community/user/businessanalytics/blogs/steven-macko/2019/07/11/telco-customer-churn-1113
[100] Jillani Soft Tech. 2024. Diabetes disease updated dataset. Retrieved from https://www.kaggle.com/datasets/jillanisofttech/diabetes-disease-updated-dataset
[101] Greeg Titan and Elmajo Adriel. 2024. Climate data daily IDN. Retrieved from https://www.kaggle.com/datasets/greegtitan/indonesia-climate
[102] Shogo Tokui, Susumu Tokumoto, Akihito Yoshii, Fuyuki Ishikawa, Takao Nakagawa, Kazuki Munakata, and Shinji Kikuchi. 2022. NeuRecover: Regression-controlled repair of deep neural networks with training history. In *SANER*. IEEE, 1111–1121.
[103] Ryota Tozuka, Noriyuki Kadoya, Seiji Tomori, Yuto Kimura, Tomohiro Kajikawa, Yuto Sugai, Yushan Xiao, and Keiichi Jingu. 2023. Improvement of deep learning prediction model in patient-specific QA for VMAT with MLC Leaf position map and patient's dose distribution. *J. Appl. Clin. Med. Phys.* 24, 10 (2023), e14055.
[104] Frederik Träuble, Julius von Kügelgen, Matthäus Kleindessner, Francesco Locatello, Bernhard Schölkopf, and Peter V. Gehler. 2021. Backward-compatible prediction updates: A probabilistic approach. In *NeurIPS*, 116–128.

[105] Chih-Fong Tsai and Yu-Chi Chen. 2019. The optimal combination of feature selection and data discretization: An empirical study. *Inf. Sci.* 505 (2019), 282–293.
[106] George Tzanetakis and Perry Cook. 2002. Musical genre classification of audio signals. *IEEE Trans. Speech Audio Process.* 10, 5 (2002), 293–302.
[107] Haijun Wang, Yun Lin, Zijiang Yang, Jun Sun, Yang Liu, Jin Song Dong, Qinghua Zheng, and Ting Liu. 2021. Explaining regressions via alignment slicing and mending. *IEEE Trans. Softw. Eng.* 47, 11 (2021), 2421–2437.
[108] Qingjie Wang, Chunfang Yue, Xiaoqing Li, Pan Liao, and Xiaoyao Li. 2023. Enhancing robustness of monthly streamflow forecasting model using embedded-feature selection algorithm based on improved gray wolf optimizer. *J. Hydrol.* 617 (2023), 128995.
[109] Song Wang, Jaechang Nam, and Lin Tan. 2017. QTEP: Quality-aware test case prioritization. In *ESEC/SIGSOFT FSE*. ACM, 523–534.
[110] Zan Wang, Hanmo You, Junjie Chen, Yingyi Zhang, Xuyuan Dong, and Wenbin Zhang. 2021. Prioritizing test inputs for deep neural networks via mutation analysis. In *ICSE*. IEEE, 397–409.
[111] Valentine Wargnier-Dauchelle, Thomas Grenier, Françoise Durand-Dubief, François Cotton, and Michaël Sdika. 2023. A weakly supervised gradient attribution constraint for interpretable classification and anomaly detection. *IEEE Trans. Med. Imaging* 42, 11 (2023), 3336–3347.
[112] Frank Wilcoxon, S. K. Katti, and Roberta A. Wilcox. 1970. Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test. In *Selected Tables in Mathematical Statistics*. Vol. 1, 171–259.
[113] Tom Nuno Wolf, Sebastian Pölsterl, Christian Wachinger, Alzheimer's disease neuroimaging initiative, and Australian imaging biomarkers lifestyle flagship study of ageing. 2022. DAFT: A universal module to interweave tabular data and 3D images in CNNs. *NeuroImage* 260 (2022), 119505.
[114] Siwei Wu, Zhenxing Pan, Xiaojing Li, Yang Wang, Jiacheng Tang, Haishan Li, Guibo Lu, Jianzhong Li, Zhenzhen Feng, Yan He, et al. 2023. Machine learning assisted photothermal conversion efficiency prediction of anticancer photothermal agents. *Chem. Eng. Sci.* 273 (2023), 118619.
[115] Yanzhao Wu, Ling Liu, Zhongwei Xie, Ka Ho Chow, and Wenqi Wei. 2021. Boosting ensemble accuracy by revisiting ensemble diversity metrics. In *CVPR*. Computer Vision Foundation/IEEE, 16469–16477.
[116] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. 2019. DeepHunter: A coverage-guided fuzz testing framework for deep neural networks. In *ISSTA*. ACM, 146–157.
[117] Yuqing Xie, Yi-An Lai, Yuanjun Xiong, Yi Zhang, and Stefano Soatto. 2021. Regression bugs are in your model! Measuring, reducing and analyzing regressions in NLP model updates. In *ACL/IJCNLP (1)*. Association for Computational Linguistics, 6589–6602.
[118] Yang Xing, Chen Lv, Zhaozhong Zhang, Huaji Wang, Xiaoxiang Na, Dongpu Cao, Efstathios Velenis, and Fei-Yue Wang. 2018. Identification and analysis of driver postures for in-vehicle driving activities and secondary tasks recognition. *IEEE Trans. Comput. Soc. Syst.* 5, 1 (2018), 95–108.
[119] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional GAN. In *NeurIPS*, 7333–7343.
[120] Lei Xu and Kalyan Veeramachaneni. 2018. Synthesizing tabular data using generative adversarial networks. arXiv:1811.11264. DOI: https://doi.org/10.48550/arXiv.1811.11264
[121] Xiaowei Xue, Min Yao, and Zhaohui Wu. 2018. A novel ensemble-based wrapper method for feature selection using extreme learning machine and genetic algorithm. *Knowl. Inf. Syst.* 57, 2 (2018), 389–412.
[122] Sijie Yan, Yuanjun Xiong, Kaustav Kundu, Shuo Yang, Siqi Deng, Meng Wang, Wei Xia, and Stefano Soatto. 2021. Positive-congruent training: Towards regression-free model updates. In *CVPR*. Computer Vision Foundation/IEEE, 14299–14308.
[123] Jialiang Yang, Jie Ju, Lei Guo, Binbin Ji, Shufang Shi, Zixuan Yang, Songlin Gao, Xu Yuan, Geng Tian, Yuebin Liang, et al. 2022. Prediction of HER2-positive breast cancer recurrence and metastasis risk from histopathological images and clinical information via multimodal deep learning. *Comput. Struct. Biotechnol. J.* 20 (2022), 333–342.
[124] Kirill Yashuk. 2024. Classify gestures by reading muscle activity. Retrieved from https://www.kaggle.com/datasets/kyr7plus/emg-4
[125] Shin Yoo and Mark Harman. 2012. Regression testing minimization, selection and prioritization: A survey. *Softw. Test. Verifi. Reliab.* 22, 2 (2012), 67–120.
[126] Hanmo You, Zan Wang, Junjie Chen, Shuang Liu, and Shuochuan Li. 2023. Regression fuzzing for deep learning systems. In *ICSE*. IEEE, 82–94.
[127] Jun Yu, Benjamin Zalatan, Yong Chen, Li Shen, and Lifang He. 2022. Tensor-based multi-modal multi-target regression for Alzheimer's disease prediction. In *BIBM*. IEEE, 639–646.
[128] Kai Yu, Mengxiang Lin, Jin Chen, and Xiangyu Zhang. 2012. Practical isolation of failure-inducing changes for debugging regression faults. In *ASE*. ACM, 20–29.

[129] Yuan Yuan, Chengze Wang, and Zhiyu Jiang. 2022. Proxy-based deep learning framework for spectral-spatial hyperspectral image classification: Efficient and robust. *IEEE Trans. Geosci. Remote. Sens.* 60 (2022), 1–15.
[130] Jie Zhang, Lingming Zhang, Mark Harman, Dan Hao, Yue Jia, and Lu Zhang. 2019. Predictive mutation testing. *IEEE Trans. Softw. Eng.* 45, 9 (2019), 898–918.
[131] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2022. Machine learning testing: Survey, landscapes and horizons. *IEEE Trans. Softw. Eng.* 48, 2 (2022), 1–36.
[132] Yuqing Zhao, Divya Saxena, and Jiannong Cao. 2023. AdaptCL: Adaptive Continual Learning for Tackling Heterogeneity in Sequential Datasets. *IEEE Trans. Neural Netw. Learn. Syst.* (2023), 1–14.
[133] Zheng Zhao, Lei Wang, and Huan Liu. 2010. Efficient spectral feature selection with minimum redundancy. In *AAAI*. AAAI Press, 673–678.
[134] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning tree-based deep model for recommender systems. In *KDD*. ACM, 1079–1088.
[135] Xiaogang Zhu and Marcel Böhme. 2021. Regression greybox fuzzing. In *CCS*. ACM, 2169–2182.
[136] Xianglei Zhu, Haichi Wang, Hanmo You, Weiheng Zhang, Yingyi Zhang, Shuang Liu, Junjie Chen, Zan Wang, and Keqiu Li. 2021. Survey on testing of intelligent systems in autonomous vehicles. *J. Softw.* 32, 7 (2021), 2056–2077.
[137] Martin Zinkevich. 2017. Rules of machine learning: Best practices for ML engineering. Retrieved from https://developers.google.com/machine-learning/guides/rules-of-ml

Received 26 June 2024; revised 5 December 2024; accepted 22 December 2024