Few-Shot Guided Mix for DNN Repairing

Xuhong Ren[1*], Bing Yu[2*], Hua Qi[2], Felix Juefei-Xu[3], Zhuo Li[2], Wanli Xue[1†], Lei Ma[2†], Jianjun Zhao[2]
[1] School of Computer Science and Engineering, Tianjin University of Technology, China
[2]Kyushu University, Japan [3]Alibaba Group, USA

***Abstract*—Although deep neural networks (DNNs) achieve rather high performance in many cutting-edge applications (*e.g.*, autonomous driving, medical diagnose), their trustworthiness on real-world scenarios still posts concerns, where some specific failure examples are often encountered during the real-world operational environment. With the limited failure examples collected during the practical operation, how to effectively leverage such failure cases to repair and enhance DNN so as to generalize to more potentially suspicious samples is challenging, but of great importance. In this paper, we formulate the failure-data-driven DNN repairing as a data augmentation problem, and design a novel augmentation-based repairing method, which to the best extent leverages limited failure cases. To realize the DNN repairing effects that generalize to specific failure examples, we originally propose *few-shot guided mix (FSGMix)* that *augments training data with the guidance of failure examples*. As a result, our method is able to achieve high generalization to the collected failure examples and other similar suspicious data. The preliminary evaluation on CIFAR-10 dataset demonstrates the potential of our proposed technique, which automatically learns to resolve the potential failure patterns in the DNN operational environment.**

***Keywords*-Deep neural network; repair; data augmentation**

I. Introduction

Deep neural network (DNN) based software systems are becoming more and more pervasive nowadays, which are deployed across many application domains, ranging from perception and decision making in autonomous driving, to identity verification for various access control bottlenecks such as the ones used by the airport customs worldwide, mobile facial authentication for payment verification, and object tracking and control capabilities for heavy-duty camera drones, *etc*. However, sometimes we, whether as DNN software developers or end-users of a DNN-based product, often overestimate the trustworthiness of DNN software. Unfortunately, the fact is that the current DNN software systems are far from being perfect and its erroneous behavior may lead to detrimental outcomes, especially for those deployed in safety- and security-critical scenarios. For example, ever since the inception of assisted and semi-autonomous driving capability, numerous car accidents occurred [21]. Another common victim example of failed DNN software is financial fraud due to identity authentication failure when the attackers try to spoof the DNN-based authentication systems [18], and the list goes on and on.

However, how to effectively repair DNN software after observing failure cases under its operational environment remains an open research question. In traditional software repair pipeline, fault localization is necessary to perform any types of repairs. However, unlike traditional software, the logic of any DNN-based software is not coded line by line, but rather manifested through the weights and connections between the neurons which are evolved from pure randomness to meaningful states before and after the training of such a DNN software, given abundant training data, for a particular task, be it recognition [7], detection [20], or even more complicated policies such as playing the game of Go.

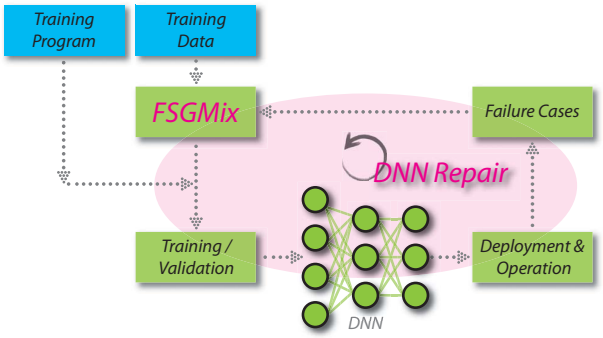

Fig. 1: The workflow of FSGMix for DNN repair.

A promising path of repairing DNN-based software is data-driven, *i.e.*, retrain or fine-tune the DNNs with failure cases so that the DNNs would hopefully make correct decisions when similar data is observed. However, two major challenges still remain. (1) In many applications where DNN software is deployed, the failure cases are quite scarce. The total number of failure cases amounts to only a tiny portion of the training corpus, making the repair through fine-tuning based on failure cases almost futile. (2) Even if the failure cases were sufficient, there exists another big challenge due to the fact that any well-trained DNNs can only deal with data within the bounds of certain high-dimensional distribution, as captured roughly by the training data. Such a distribution captures the amount of variation found in the training corpus. However, the real-world operational environment is usually much more complex, full of unseen data, and data that lie outside of such distribution (*i.e.*, distribution shift). A gap exists between the data variation in the real-world environment and that of the training corpus. Therefore, it still remains unclear how successfully fixing these known failure cases can generalize to prevent the repaired DNNs from failing on similar future unseen cases in the wild.

In this work, we propose a novel DNN repair method that aims at solving the aforementioned two challenges, *i.e.*, the

* Xuhong Ren and Bing Yu contribute equally to this work. † Wanli Xue (xuewanli@tjut.edu.cn) and Lei Ma (malei@ait.kyushu-u.ac.jp) are the corresponding authors.

2576-3148/20/$31.00 ©2020 IEEE
DOI 10.1109/ICSME46990.2020.00079

scarcity issue and the generalization issue. Specifically, we formulate the failure-data-driven DNN repairing as a data augmentation problem, and design a novel augmentation-based repairing method while achieving high data efficiency, *i.e.*, to the best extent leveraging limited failure cases. Data augmentation not only solves the scarcity problem by allowing the limited amount of meaningful retraining data to be perturbed into many folds of usable ones, but also leads to better generalization on these failure cases. To further make the failure case augmentation more data-efficient, we devise a novel learning mechanism for the augmenter itself guided by the failure cases. In this way, how the retraining data are augmented for repairing the DNNs is directly supervised by the error signals provided by the failure cases. We call this method *few-shot guided mix (FSGMix)* that *augments training data with the guidance of failure examples*. As a result, our method is able to achieve high generalization to the collected failure examples and other similar suspicious data.

The DNN development and enhancement are an iterative and incremental process as depicted in Figure 1. After collecting the failed examples, how to, to the best extent, leverage the failure case information to guide the DNN repairing process efficiently and effectively by means of learnable data augmentation on the training data for the next development iteration, would a be a promising direction that this paper tries to explore. We conduct preliminary evaluations on CIFAR-10 dataset and the results demonstrate the potential of our proposed technique, which automatically learns to resolve the potential failure patterns in the DNN operational environment. To the best of our knowledge, this work is the first that learns to repair of DNN guided by failure case driven augmentation.

II. Few-Shot Guided Mix

A. Data Augmentation for DNN Repairing

Given a labeled dataset $\mathcal{D}_{\mathrm{o}}$ for classification, we can train a DNN that is denoted as $\mathrm{f}_\theta(\cdot)$ with θ representing its parameters. A well-trained DNN usually achieves high classification accuracy on the testing dataset that has similar distribution with $\mathcal{D}_{\mathrm{o}}$. However, some failure examples would often occur when deploying the DNN in the operational environment, since the training dataset $\mathcal{D}_{\mathrm{o}}$ could hardly cover all possible and diverse data, as well as the potential noise patterns that occur in the real world. Even though, it is not difficult to collect a set of failure examples and construct a small dataset denoted as $\mathcal{D}_{\mathrm{f}}$, after some operation time. To better adapt to the real-world operational environment, we should properly repair the DNN on the collected failure examples so far, while trying to generalize to future similar data. The continuous DNN repairing and enhancement on the given limited data while achieving high generalization capability is rather challenging but of great importance for real-world applications. A DNN is often iteratively improved through continuous enhancement.

A straightforward repairing solution is to fine-tune the DNN with $\mathcal{D}_{\mathrm{f}}$. However, such a fine-tuned model easily overfits the failure examples and hardly generalizes to similar data. Although recent works demonstrate the effectiveness of random augmentation to improve DNNs generalization on degradation data, existing augmentation solutions are unable to repair a DNN on a target set of failure examples. Inspired by the recent AugMix method [9], we introduce *FSGMix*, a failure-example guided data augmentation technique for repairing of a DNN on a given failure set while trying to achieve certain generalization capability. Next, we first formulate the DNN repairing as a data augmentation problem. In general, the data augmentation problem consists of two key parts: *augmentation operation set* (*i.e.*, $\mathcal{O} = \{\mathrm{O}_i\}_1^N$ including N augmentation operations, *e.g.*, rotation and colorization, and an *operation weighting strategy*, *i.e.*, $\mathbf{w} = [w_0, \ldots, w_i, \ldots, w_N] = \mathrm{S}(\phi)$. ϕ denotes the random sampling distributions and $w_i \in [0, 1]$ represents the weight for each operation. When training a DNN (*e.g.*, $\mathrm{f}_\theta(\cdot)$), the data augmentation often follows the steps below:

(i) Randomly sampling an example $\mathbf{I}$ from training set $\mathcal{D}_{\mathrm{o}}$.
(ii) Sampling the operation through $\mathrm{S}(\phi)$.
(iii) Augmenting data $\mathbf{I}$ with all operations $\{\mathrm{O}_i | w_i > 0, \mathrm{O}_i \in \mathcal{O}\}$, obtaining the augmented examples $\{\hat{\mathbf{I}}_i = \mathrm{O}_i(\mathbf{I}) | \mathrm{O}_i \in \mathcal{O}\}$.
(iv) Mixing all augmented examples with their weights into a final augmented example, *i.e.*, $\hat{\mathbf{I}} = \sum_{i=1}^N w_i \hat{\mathbf{I}}_i$.

The augmented examples are packed with the original example $\mathbf{I}$ for training, to improve the DNN quality. Considering the whole process of the data augmentation, we observe that the random sampling distribution (*i.e.*, ϕ) can directly affect the augmentation results when we have a fixed set of augmentation operations (*i.e.*, $\mathcal{O} = \{\mathrm{O}_i\}_1^N$). For example, when ϕ is a uniform distribution, we have $\{w_i = \frac{1}{N}\}_{i=1}^N$, where all augmentation operations share the same weight. Obviously, a fixed and pre-defined distribution could not sample weights that perfectly adapt to a specific dataset containing failure examples collected from the real-world environment. As a result, we need to develop a new operation sampling method by considering the failure examples and generating suitable $\{w_i\}_{i=1}^N$.

B. Repairing by Few-Shot Guided Mix Augmentation

In this paper, we propose a *few-shot guided mix* method for DNN repairing given a target failure sample set. *FSGMix* generates the operation weights $\{w_i\}_{i=1}^N$ according to collected failure examples and can be embedded into the data augmentation method introduced in Section II-A.

In particular, for the jth example $\mathbf{I}^j$ in the collected failure dataset $\mathcal{D}_{\mathrm{f}}$, we first learn its operation weights $\mathbf{w}^j = [w_1^j, \ldots, w_N^j]$ to enable the augmented version (*i.e.*, $\hat{\mathbf{I}}^j = \sum_{i=1}^N w_i^j \hat{\mathbf{I}}_i^j$), to be correctly classified by the DNN. To this end, we solve the following problem:

$$\arg\min_{\{w_i^j\}_{i=1}^N} J(\mathrm{f}_\theta(\sum_{i=1}^N w_i^j \hat{\mathbf{I}}_i^j), y^j) \tag{1}$$

where $J(\cdot)$ is the cross-entropy loss for image classification and y^j is the annotation of $\mathbf{I}^j \in \mathcal{D}_{\mathrm{f}}$. For each failure example, we obtain its corresponding weights $\mathbf{w}^j = [w_1^j, \ldots, w_N^j]$. As a result, we collect a total of M weight vectors and construct

Algorithm 1: Few-shot Guided Mix (*FSGMix*)

Input: Training dataset $\mathcal{D}_{\text{o}}$ with K images, collected failure examples $\mathcal{D}_{\text{f}}$ with M images, a pre-trained DNN $\text{f}_\theta(\cdot)$, augmentation operation set $\mathcal{O}$.
Output: Repaired DNN $\text{f}_{\bar{\theta}}(\cdot)$.
# *Few-shot operation sampling*
for $j = 1$ to M **do**
- Loading the jth image $\mathbf{I}^j$ from $\mathcal{D}_{\text{f}}$;
- Randomly initializing $\mathbf{w}^j$;
- Calculating $\mathbf{w}^j$ by solving Eq. (1) via the gradient descent ;

Calculating the GMM ϕ_{gmm} on $\mathcal{W}$ with EM algorithm;
# *Data augmentation for DNN repairing*
for $k = 1$ to K **do**
- Loading the kth image $\mathbf{I}^k$ from $\mathcal{D}_{\text{o}} \cup \mathcal{D}_{\text{f}}$;
- Sampling two weight vectors $\mathbf{w}^k_{\text{aug1}} = \text{S}(\phi_{\text{gmm}})$ and $\mathbf{w}^k_{\text{aug2}} = \text{S}(\phi_{\text{gmm}})$;
- Calculating two augmented images: $\hat{\mathbf{I}}^k_{\text{aug1}}$ = Augment($\mathbf{I}^k$,$\mathbf{w}^k_{\text{aug1}}$) , and $\hat{\mathbf{I}}^k_{\text{aug2}}$ = Augment($\mathbf{I}^k$,$\mathbf{w}^k_{\text{aug2}}$) [1];
- Calculating the loss function: $J(\text{f}_\theta(\mathbf{I}^k), y^k) + \lambda \text{JS}(\text{f}_\theta(\mathbf{I}^k), \text{f}_\theta(\hat{\mathbf{I}}^k_{\text{aug1}}), \text{f}_\theta(\hat{\mathbf{I}}^k_{\text{aug2}}), y^k)$;
- Updating the parameters of $\text{f}_\theta(\cdot)$;

a set $\mathcal{W} = \{\mathbf{w}^j\}_{j=1}^M$ with M being the size of $\mathcal{D}_{\text{f}}$. Then, we use the $\mathcal{W}$ to fit a N-dimensional Gaussian mixture model (GMM) denoted as ϕ_{gmm} via the expectation maximization (EM) algorithm. Finally, we can replace the ϕ in section II-A by ϕ_{gmm} and realize the *few-shot guided mix (FSGMix)*.

C. Algorithm

We summarize the whole process of FSGMix in Algorithm 1. To be specific, we first perform *few-shot operation sampling* and use the collected failure examples (*i.e.*, $\mathcal{D}_{\text{f}}$), to estimate the GMM distribution ϕ_{gmm}. The Eq. (1) is then minimized via the gradient descent algorithm. After this, the GMM is used to produce random weights for different augmentation operations and two augmented images are generated for each image in the original training dataset $\mathcal{D}_{\text{o}}$. Finally, the DNN's parameters are updated by minimizing the cross-entropy loss and Jensen-Shannon divergence loss (simplified as $\text{JS}(\cdot)$). Intuitively, our FSGMix uses the failure examples to estimate the augmentation operation sampling distribution, thus is able to indicate how to generate augmented examples to repair the DNN.

III. Preliminary Evaluation

A. Subject Model and Dataset.

Model architecture. In this paper, we validate the effectiveness of our method via the image classification task with a state-of-the-art DNN architecture as $\text{f}_\theta(\cdot)$, *i.e.*, wide residual network (WideResNet)[24] that is constructed by a series of residual blocks and benefits from avoiding vanishing gradient during training process.

Training dataset. We select CIFAR-10 dataset as the subject dataset, containing 10 classes of $32 \times 32 \times 3$ color natural images where each class has 5,000 images for training. Then, we regard all 50,000 images as the training dataset, *i.e.*, $\mathcal{D}_{\text{o}}$ in the Algorithm 1 with K=50,000.

Testing dataset. To validate whether our method could help the DNN generalize well to specific failure examples and other similar suspicious data, we modify the CIFAR-10's testing dataset containing 10,000 images.

Specifically, we use the augmentation operations introduced in section III-A with a randomly generated weight vector (denoted as $\mathbf{w}_{\text{deg}}$) to degrade all images in the testing dataset. The degradation process could be formulated as: $\hat{\mathbf{I}}_{\text{test}} = \sum_{i=1}^{N} w^k_{\text{deg},i} \text{O}_i(\mathbf{I}_{\text{test}})$ where $\mathbf{I}_{\text{test}}$ is a testing image and $\hat{\mathbf{I}}_{\text{test}}$ denotes the corresponding degraded image.

Then, we use the model $\text{f}_\theta(\cdot)$ pre-trained on $\mathcal{D}_{\text{o}}$ to classify all degraded images and get mis-classified images. We randomly select 1,000 images as the 'specific' failure examples and obtain $\mathcal{D}_{\text{f}}$. The rest images serve as the testing dataset, *i.e.*, $\mathcal{D}_{\text{t}}$, to validate whether the model trained with our method could achieve higher accuracy than the models with baseline training methods. Note that, for different $\mathbf{w}_{\text{deg}}$, we would obtain different degradation testing datasets.

B. Evaluation Setups

Augmentation operations. For the augmentation operation set $\mathcal{O}$, we consider three typical augments (*i.e.*, *autocontrast*, *rotate*, *solarize*) and their four combinations (*autocontrast*∘*rotate*, *autocontrast*∘*solarize*, *rotate*∘*solarize*, and *autocontrast*∘*rotate*∘*solarize*) as seven augments.

Hyperparameters for training. We train WideResNet with SGD optimizer using a cosine learning rate which is initially set as 0.1. After 100 epochs of training, we save the best model, that achieves the highest testing accuracy.

Baselines. We consider the following baseline methods: 1) using the collected failure examples to fine-tune the DNN model without data augmentation, and we denote this method as 'FT-woAug'. 2) using the state-of-the-art augmentation method AugMix [9] augmented examples to fine-tune the DNN model, and we denote it as 'AugMix'.[2] 3) We consider three variations of our method: 1) the first one uses the GMM for modeling sampling distribution. 2) the second one uses multivariate normal distribution (MVN) where all weights have independent distributions. 3) the third one uses a fixed weight vector for data augmentation. We name the three methods as 'FSGMix(GMM)', 'FSGMix(MVN)', and 'FSGMix(Fixed)', respectively. Since all augmentation methods contain random module, we run ten repetitions for each configuration and report the average results.

Metric. In general, with the CIFAR-10's training dataset (*i.e.*, $\mathcal{D}_{\text{o}}$), we first obtain a pre-trained WideResNet model which misclassifies all images in $\mathcal{D}_{\text{f}}$. Then, our goal is to repair this model with baselines and our methods, and enhance the testing accuracy, *i.e.*, $\mathcal{D}_{\text{t}}$ whose images are similar to the images in $\mathcal{D}_{\text{f}}$, is regarded as the evaluation metric.

[1]Augment denotes process of augmenting an image with a given weight vector, corresponding step (iii) and (v) in section II-A.

[2]For a fair comparison, we make the AugMix use the same augmentation operation set but independent sampling distribution with our method.

	Training set	Accuracy
Pre-trained model	$\mathcal{D}_o$	0
FT-woAug	$\mathcal{D}_f$	0.1371
AugMix	$\mathcal{D}_f$	0.2230
FSGMix(Fixed)	$\mathcal{D}_f$	0.2098
FSGMix(MVN)	$\mathcal{D}_f$	0.2205
FSGMix(GMM)	$\mathcal{D}_f$	0.2689
FT-woAug	$\mathcal{D}_o$+$\mathcal{D}_f$	0.1542
AugMix	$\mathcal{D}_o$+$\mathcal{D}_f$	0.2630
FSGMix(Fixed)	$\mathcal{D}_o$+$\mathcal{D}_f$	0.2285
FSGMix(MVN)	$\mathcal{D}_o$+$\mathcal{D}_f$	0.2693
FSGMix(GMM)	$\mathcal{D}_o$+$\mathcal{D}_f$	0.2717

TABLE I: Comparison results with baseline methods by fine-tuning the pre-trained model. The comparison is conducted under two training sets: $\mathcal{D}_f$ and $\mathcal{D}_o$+$\mathcal{D}_f$, respectively.

C. Preliminary Results

We show the comparison results with baselines and our method's variations in Table I containing two setups: 1) only using collected failure examples, *i.e.*, $\mathcal{D}_f$, for fine-tuning the pre-trained model. 2) using both training dataset, *i.e.*, $\mathcal{D}_f$+$\mathcal{D}_o$.

Comparison with baselines. In general, our method FSGMix(GMM) obtains the highest accuracy under different sizes of training set. Obviously, the naive fine-tuning method without any data augmentations, *i.e.*, FT-woAug, has limited generalization on the testing dataset, although the images are similar to the collected failure examples. Compared with the SOTA data augmentation method, *i.e.*, AugMix, our methods FSGMix(GMM) achieves higher accuracy, which demonstrates its advantages in generalizing to specific failure examples.

Considering the results on training sets having different size, we observe that all methods with the larger training set (*i.e.*, $\mathcal{D}_f$+$\mathcal{D}_o$) achieve higher accuracy than the results on smaller one (*i.e.*, $\mathcal{D}_f$). However, the accuracy difference of our FSGMix(GMM) is much smaller than that of other methods. It demonstrates that our method could still keep high generalization with a small dataset.

Ablation study. According to the results of our three variations, we observe that: 1) using fixed weights for combining augmented examples get low accuracy. 2) jointly modeling all weights with GMM could achieve much better accuracy than using MVN that modeling each weight independently.

D. Analysis

Toleration to different kinds of failure examples. As introduced in section III-A, we use $\mathbf{w}_{deg}$ to degrade testing images, which represents a kind of failure examples that cause DNN misclassification. Then, it is meaningful to study whether our repaired model relying on $\mathbf{w}_{deg}$ could still work under failure examples relying on other $\mathbf{w}_{deg}$. To this end, we randomly generate three $\mathbf{w}_{deg}$, *i.e.*, $\mathbf{w}_{deg}$-1, $\mathbf{w}_{deg}$-2, and $\mathbf{w}_{deg}$-3, resulting in three testing sets of failure examples. Then, we use the repaired models in Table I to classify the three testing sets and report the results in Table II. Compared with the results in Table I, although the accuracy of all our methods becomes lower than the results in Table I, our FSGMix(GMM) still achieves the highest accuracy on $\mathbf{w}_{deg}$-1 and $\mathbf{w}_{deg}$-2 while obtaining slightly accuracy than FSGMix(MVN) on $\mathbf{w}_{deg}$-3.

	Training set	$\mathbf{w}_{deg}$-1	$\mathbf{w}_{deg}$-2	$\mathbf{w}_{deg}$-3
FSGMix(Fixed)	$\mathcal{D}_f$	0.0985	0.0951	0.0868
FSGMix(MVN)	$\mathcal{D}_f$	0.1762	0.1524	0.189
FSGMix(GMM)	$\mathcal{D}_f$	0.2186	0.2249	0.2231
FSGMix(Fixed)	$\mathcal{D}_o$+$\mathcal{D}_f$	0.1093	0.1232	0.1154
FSGMix(MVN)	$\mathcal{D}_o$+$\mathcal{D}_f$	0.1996	0.2043	0.2055
FSGMix(GMM)	$\mathcal{D}_o$+$\mathcal{D}_f$	0.2313	0.2108	0.189

TABLE II: Toleration to different kinds of failure examples.

IV. Related Work

Testing and repairing. Recently, we have witnessed the rapid progress of the testing DNNs [27, 1]. Several structural testing adequacy criteria (*e.g.*, neuron coverage, surprise adequacy) for DNNs are proposed to approximately measure the testing sufficiency. With the guidance of these criteria, automated test generation techniques are also proposed, with intention to to effective detect the DNN errors (*e.g.*, DeepXplore [17], DeepTest [21], DeepHunter [23], DeepGauge [13], Surprise Adequacy [11], DeepCT [14], DeepMutation [15], *etc.*), among which metamorphic relations are widely used as the test oracle. However, it is still unclear how to effectively improve the capability and repair the incorrect behaviors, which potentially occur in the operational environments of a DNN. MODE [16] and Arachne [19] first identify the DNN elements (*e.g.*, weights) that mostly impact the incorrect behavior, and leverage data selection for retraining and search-based method to repair the tentative errors. One caveat of weight manipulation without training is that it heavily relies on the fault localization of DNN's weights. However, in practice, such pinpointing of the faulty weights can be difficult, especially for DNNs with complex neural and layer connections by residual connections, skip connections, and advanced typologies designed by neural architecture search (NAS). In particular, when the DNN topology is complicated, the influence of the 'faulty weights or neurons' propagates easily throughout the entire network, rendering fault localization futile. Also, there is no principled way of performing such a fault localization task efficiently in a complex neural network. Apricot [25] leverages multiple DNNs and incrementally repairs the DNN candidate by adjusting the weights under the consideration of the made multiple DNNs. Md Johirul *et al*. [10] discuss the challenges and common fix patterns for DNN repair. Being orthogonal to existing repairing work, this paper adopts a blackbox approach and resorts to a data-level repair solution through learning, to effectively and efficiently repair the DNN as a whole.

Data augmentation. This paper adopts a black-box approach without leveraging internal information of DNNs. Instead, it follows a different perspective of DNN repairing by failure-case-guided data automation. Data augmentation is a common approach, which has the potential to improve the capability and repair incorrect behavior of DNN in the operational environment. Due to the huge data augmentation

space, how to effectively search and select augmented data for (re-)training becomes a big challenge. Arbitrarily augmenting data is quite expensive, and might not be able to repair and enhance the potential failure cases in a concrete DNN operational environment. To generally improve the quality of a DNN, some recent progress has been made through data augmentation. For example, previous work shows that random augmentation through transformations (*e.g.*, occlusion, rotation) is useful to improve DNN quality [7, 3, 8]. Recently, data mixing based augmentation shows its advantage in improving the general quality of a DNN. Mixup [26] performs an element-wise convex combination of two images from input space for data augmentation and training. AugMix extends this idea one step further, which performs mixing of multiple images under multiple steps of transformation [9], achieving state-of-the-art performance in enhancing DNN quality. Even though, the data augmentation space can still be quite large, which might also be limited without knowing the concrete failure patterns. Compared with existing techniques, this paper is the first to propose failure-guided data augmentation method, aiming to learn and resolve failure patterns of DNN in the real-world operational environment. Our technique leverages the important failure information, and effectively guides the data augmentation direction, so as to potentially repair unseen similar failures in the real-world environment. The preliminary evaluation demonstrates the promising of our technique.

Few shot learning (FSL). FSL is a hot topic to address problems for data-intensive applications when the dataset is small. FSL was first proposed to learn from a small scale of examples with supervised information [12, 5] and achieved big breakthrough in learning high-performance models on small datasets. In parallel, FSL also contributes to light the burden of collecting large-scale supervised data. The representative works include image classification [22], neural architecture search [2]. Benefit from the growing up industrial demand and the progress of inexpensive learning, many related machine learning approaches have been proposed, including meta-learning [6] and generative modeling [4]. Our method takes advantage of FSL and is the very first to apply it to DNN repair.

V. Conclusion

In this paper, we recast the failure-data-driven DNN repair as a data augmentation problem, and design a novel augmentation-based repair method, which to the best extent leverages the limited failure cases. To realize the DNN repairing that generalizes to specific failure examples, we originally propose *few-shot guided mix (FSGMix)* that augments training data with the guidance of failure examples. As a result, our method is able to achieve high generalization capability to the collected failure examples and other similar suspicious data, making the first attempt to solve the scarcity and generalization problems surrounding the limited amount of failure cases. By carefully designing the learning mechanism for the augmentor with guidance from the failure cases, our method provides a feasible solution towards data-driven DNN repair. The preliminary evaluation on CIFAR-10 dataset demonstrates the potential of the guided data-driven direction for DNN repair.

VI. Acknowledgment

This work was supported by JST-Mirai Program Grant No. JPMJMI18BB, JSPS KAKENHI Grant No. 20H04168, 19K24348, 19H04086, and 18H04097 of Japan. It was also supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61906135, 61871258 and U1703261.

References

[1] H. B. Braiek and F. Khomh. On testing machine learning programs. *J. Syst. Softw.*, 164:110542, 2020.
[2] A. Brock, T. Lim, J. Ritchie, and N. Weston. Smash: One-shot model architecture search through hypernetworks. 08 2017.
[3] T. Devries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.
[4] H. Edwards and A. Storkey. Towards a neural statistician. 06 2016.
[5] M. Fink. Object classification from a single example utilizing class relevance pseudo-metrics. 01 2004.
[6] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.
[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, pages 770–778, 2016.
[8] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *ICLR*, 2019.
[9] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *ICLR*, 2020.
[10] M. J. Islam, R. Pan, G. Nguyen, and H. Rajan. Repairing deep neural networks: Fix patterns and challenges. *arXiv preprint arXiv:2005.00972*, 2020.
[11] J. Kim, R. Feldt, and S. Yoo. Guiding deep learning system testing using surprise adequacy. In *ICSE*, pages 1039–1049, 2019.
[12] Li Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE TPAMI*, 28(4):594–611, 2006.
[13] L. Ma, F. Juefei-Xu, J. Sun, C. Chen, T. Su, F. Zhang, M. Xue, B. Li, L. Li, Y. Liu, J. Zhao, and Y. Wang. DeepGauge: Multi-Granularity Testing Criteria for Deep Learning Systems. In *ASE*, 2018.
[14] L. Ma, F. Juefei-Xu, M. Xue, B. Li, L. Li, Y. Liu, and J. Zhao. DeepCT: Tomographic Combinatorial Testing for Deep Learning Systems. *SANER*, 2019.
[15] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao, and Y. Wang. DeepMutation: Mutation Testing of Deep Learning Systems. In *ISSRE*, 2018.
[16] S. Ma, Y. Liu, W.-C. Lee, X. Zhang, and A. Grama. Mode: automated neural network model debugging via state differential analysis and input selection. In *ESEC/FSE*, pages 175–186. ACM, 2018.
[17] K. Pei, Y. Cao, J. Yang, and S. Jana. DeepXplore: Automated whitebox testing of deep learning systems. In *SOSP*, pages 1–18, 2017.
[18] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, June 2015.
[19] J. Sohn, S. Kang, and S. Yoo. Search based repair of deep neural networks. *arXiv preprint arXiv:1912.12463*, 2019.
[20] G. Tao, S. Ma, Y. Liu, and X. Zhang. Attacks meet interpretability: Attribute-steered detection of adversarial samples. In *NeurIPS*, pages 7728–7739, 2018.
[21] Y. Tian, K. Pei, S. Jana, and B. Ray. DeepTest: Automated testing of deep-neural-network-driven autonomous cars. In *ICSE*, pages 303–314, 2018.
[22] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *NeurIPS*, pages 3630–3638. 2016.
[23] X. Xie, L. Ma, F. Juefei-Xu, M. Xue, H. Chen, Y. Liu, J. Zhao, B. Li, J. Yin, and S. See. DeepHunter: A Coverage-Guided Fuzz Testing Framework for Deep Neural Networks. In *ISSTA*, 2019.
[24] S. Zagoruyko and N. Komodakis. Wide residual networks, 2016.
[25] H. Zhang and W. K. Chan. Apricot: A weight-adaptation approach to fixing deep learning models. In *ASE*, page 376–387, 2019.
[26] H. Zhang, M. Cissé, Y. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *ArXiv*, abs/1710.09412, 2018.
[27] J. M. Zhang, M. Harman, L. Ma, and Y. Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 2020.