DeepRepair: Style-Guided Repairing for Deep Neural Networks in the Real-World Operational Environment

Bing Yu, Hua Qi, Qing Guo, *Member, IEEE*, Felix Juefei-Xu, *Member, IEEE*, Xiaofei Xie, Lei Ma, and Jianjun Zhao

***Abstract*—Deep neural networks (DNNs) are continuously expanding their application to various domains due to their high performance. Nevertheless, a well-trained DNN after deployment could oftentimes raise errors during practical use in the operational environment due to the mismatching between distributions of the training dataset and the potential unknown noise factors in the operational environment, e.g., weather, blur, noise, etc. Hence, it poses a rather important problem for the DNNs' real-world applications: how to repair the deployed DNNs for correcting the failure samples under the deployed operational environment while not harming their capability of handling normal or clean data with limited failure samples we can collect. In this article, we propose a *style-guided data augmentation for repairing DNN in the operational environment*, which learns and introduces the unknown failure patterns within the failure samples into the training data via the style transfer. Moreover, we further propose the *clustering-based failure data generation* for much more effective style-guided data augmentation. We conduct a large-scale evaluation with 15 degradation factors that may happen in the real world and compare with four state-of-the-art data augmentation methods and two DNN repairing methods. Our technique successfully repairs three convolutional neural networks and two recurrent neural networks with averaging 62.88% and 39.02% accuracy enhancements on the 15 failure patterns, respectively, achieving higher repairing performance than state-of-the-art repairing methods on the most failure patterns with even better accuracy on clean datasets.**

***Index Terms*—Data augmentation, deep neural network (DNN) repairing, deep neural network, operational environment.**

Manuscript received November 14, 2020; revised May 3, 2021; accepted June 6, 2021. This work was supported in part by the JSPS KAKENHI under Grant JP20H04168, Grant JP19K24348, Grant JP19H04086, and Grant JP21H04877 and in part by the JST-Mirai Program under Grant JPMJMI20B8, Japan. The work of Lei Ma was supported by Canada CIFAR AI Program and Natural Sciences and Engineering Research Council of Canada. Associate Editor: Z. Chen. *(Corresponding authors: Qing Guo; Lei Ma.)*

Bing Yu, Hua Qi, Xiaofei Xie, and Jianjun Zhao are with the Kyushu University, Fukuoka 819-0395, Japan (e-mail: uhyou.yu@gmail.com; qi.hua.677@s.kyushu-u.ac.jp; xfxie@ntu.edu.sg; zhao@ait.kyushu-u.ac.jp).

Qing Guo is with the Nanyang Technological University, 639798, Singapore (e-mail: tsingqguo@gmail.com).

Felix Juefei-Xu is with the Alibaba Group, Sunnyvale, CA 94085 USA (e-mail: juefei.xu@gmail.com).

Lei Ma is with the University of Alberta, Edmonton, AB T6G 2R3, Canada and also with Kyushu University, Fukuoka 819-0395, Japan (e-mail: malei@ait.kyushu-u.ac.jp).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TR.2021.3096332.

Digital Object Identifier 10.1109/TR.2021.3096332

I. Introduction

OVER the past few years, deep neural networks (DNNs) achieved human-competitive performance and have been widely deployed in many real-world application domains. Multifarious applications with DNNs spring up like mushrooms, ranging from visual perception [1]–[3], such as autonomous driving [4]–[7], face recognition [8]–[11] and generation [12], [13], object detection [14], [15] and segmentation [16], [17], tracking [18], [19], medical imaging analysis [20], [21], and so on. Besides, DNNs are widely used to solve various low-level vision problems such as super-resolution [22]–[24], denoising [25], [26], illumination correction [27]–[29], image inpainting [30], [31], deraining [32], [33], dehazing [34], [35], etc. In addition, more and more automatic speech recognition and natural language processing [36], [37] applications start using DNNs such as machine translation [38]–[40], hate speech detection [41], [42], sentiment analysis [43], [44], speech generation and voice assistant [45], [46], etc. More recent, researchers are active in applying DNNs on intelligent agents for games [47]–[49] and decision making for robotics applications [50], etc.

Compared to traditional software, a prominent advantage of deep neural network (DNN) software is that DNN software can capitalize on the abundant real-world training data and high-capacity neural networks to learn the desired prediction or generative models, oftentimes surpassing human-level performance. One of the milestone examples is what we have witnessed in the game of Go where AlphaGo [51] and subsequent AlphaGo Zero [52] as well as AlphaZero [53] dominate the complicated game. Another example is in the realm of partial information game such as the Libratus [54], [55], which out-bluffed best human players in the poker game of no-limit Texas Hold'Em.

Despite the high performance and many successes among the aforementioned DNN-based applications, the DNN software still suffers from reliability issues, i.e., oftentimes the well-trained DNNs that are deployed in the real-world operational environment can behave erroneously and deviate from what they are designed for. This can be primarily caused by the gap between the real-world test data distribution $\mathcal{D}_{\mathrm{te}}$ in the operational environment and the distribution of the previously collected training data corpus $\mathcal{D}_{\mathrm{tr}}$. We say a DNN has a high level of generalizability when the DNN that is trained on data that follow $\mathcal{D}_{\mathrm{tr}}$ can perform well on test data that follow $\mathcal{D}_{\mathrm{te}}$. In other words, DNN software works better as the real-world

test data distribution ($\mathcal{D}_{\rm tr}$) approaches the collected training data corpus ($\mathcal{D}_{\rm te}$). There are several aspects worth discussing as to why such a gap between $\mathcal{D}_{\rm tr}$ and $\mathcal{D}_{\rm te}$ exists. First of all, the amount of labeled training data for a supervised learning scenario is usually limited, either due to the cost of data collection and/or the cost of providing manual ground-truth labels. Second, even the DNNs are trained with very abundant training data, and the training corpus can never cover all the real-world (potential noise) variations and perturbations. There can always be some corner cases that are never seen by the DNNs during the training process, and, sometimes, these corner cases can be the culprit for causing erroneous behavior of the deployed DNNs. Third, the tradeoff between the DNN performance on the training set and the DNN generalizability on the unseen testing set is usually not well-defined in a clear cut because, at training time, we usually do not observe the real-world testing data. Performing too well on the training data will even lead to overfitting and poor generalizability over testing data. It usually takes a separate validation set mimicking the real-world testing data as well as heuristics to determine the best tradeoff strategy. However, issues remain because the validation set often does not represent the entirety of the real-world testing data of the upcoming deployed operational environment.

There have been some recent attempts to measure the quality and robustness of the deployed DNN software by discovering failure cases that cannot be predicted correctly, such as adversarial attack techniques [56]–[58], the deep learning (DL) testing techniques [59]–[62], etc. Under the real-world setting, the already-deployed DNNs could oftentimes raise errors during practical usage due to the mismatch between distributions of training dataset and real-world collected data that may cause failures by unknown real-world factors, e.g., weather, blur, noise etc. Hence, it poses a rather important problem for the DNNs' real-world applications: how to repair the deployed DNNs for the *unknown* failure data while not harming their capability of handling normal or clean data. Recently, automatic repair [63], [64] has achieved promising progress in the traditional software, where faults in a program can be caused by specific lines of code. Thus, the repair can be performed to localize and repair the fault-triggering source code. Differently, DNN follows a data-driven programming paradigm and has no such explicit structure, which makes the repair of DNN still an open challenging research problem.

One of the most commonly used methods for DNN repair is to retrain the DNN with the failure samples. By discovering the failure samples (e.g., adversarial examples) and adding them to the training data, the repaired model can be more robust to these specific patterns, reducing their effectiveness to model's performance. Another technique is to directly modify the (hyper)parameters of the model or the structure [65], [66] based on the guidance of the failure methods. Some techniques [67]–[69] are proposed to identify the buggy units of the DNN (e.g., the neurons) and then fix these errors by generating samples for retraining. The key challenge is that we usually can only collect a small number of failure cases, which represent limited failure patterns in the real-world environment. Thus, repair with direct retraining or evolving parameters may have low generalizability, i.e., they can only work well on the collected failure cases but may fail on other unseen failures. Although sometimes we can generate a large number of failure cases, it is still unknown whether these failure cases could represent diverse failure patterns. It is inefficient if many failures may be redundant, i.e., they share the same failure pattern. Another challenge is that the repair must retain the performance and capability of the DNNs when handling data that can be predicted correctly before. How to achieve both goals simultaneously is a challenge itself.

Toward tackling the challenge, in this article, we propose a novel method to repair the DNN on image classification task. Our assumption is that we could only collect a small number of failure data, based on which the DNN needs to be repaired. Specifically, we propose the *style-guided data augmentation for DNN repairing* where a style transfer-based method [70]–[73] is proposed to introduce the unknown failure patterns (e.g., potential noise and combinations) within the failure data into the training data via data augmentation. Moreover, we propose the *clustering-based failure data generation* for more effective style-guided data augmentation. By generating diverse augmented data, we retrain the model such that the collected failure data and other similar failure data can be fixed.

We conduct a large-scale evaluation with fifteen possible degradation factors that may happen in the real world and compare with four state-of-the-art data augmentation methods and two DNN repairing methods. The DNN models repaired with our technique perform as well as the original one, sometimes even better on clear data. Specifically, our method successfully repairs three CNNs and two RNNs with averaging 62.88% and 39.02% accuracy enhancements on the fifteen failure patterns, respectively, achieving stronger repairing capability than state-of-the-art repairing methods on the most failure patterns.

Overall, the contribution of this article is summarized as follows.

1) We formulate the problem of DNN repairing based on a small limited number of failure cases and analyze the challenges.
2) We originally propose a novel repair technique by style transfer-based data augmentation for the DNNs in the real-world operational environment.
3) We perform a large-scale evaluation of our proposed technique under 15 potential degradation factors, comparing with six state-of-the-art methods as baselines.

The repair of DNNs, in general, can be rather challenging and sometimes even impossible without any assumption. This article takes a special focus on repairing the potential issues that are caused by the noise patterns (that introduce data corruptions) from the operational environments, which we believe could be an important direction and repairing scenario toward practical DNN application with high quality.

The rest of the paper is organized as the following: in Section II, we introduce the background with respect to DNN software repairing. In Section III, we formally introduce our main technical contributions where we first formulate the problem of DNN repairing for some specified failure patterns. Then, we present the data augmentation-based solution for this problem and reveal the associated challenges, which are further addressed by the proposed style-guided data augmentation for DNN repairing. In Section IV, we describe our evaluation design and

configurations. In Section V, we discuss the evaluation results and findings of our study. Finally, we conclude our work in Section VI.

II. Related Work

In this section, we first briefly discuss the recent progress on DNN testing, where many research efforts have been spent on. Then, we focus more on the connections and differences between DNN software repair and traditional software repair, diving into several recent work on DNN repair, with detailed discussion on their method formulation, advantages, as well as limitations.

A. DNN Software Testing

To enable the quality assessment and defect detection of DNNs, various DL testing techniques have been recently proposed. Considering the fundamental difference between traditional software and DNN, some research focuses on the testing criteria design [59], [61], [74], [75]. DeepXplore [59] originally proposes the neuron coverage toward measuring the adequacy of the test cases. A neuron is activated if its value is larger than a threshold. Neuron coverage measures the percentage of neurons that are activated. However, neuron coverage can be coarse and only a few test inputs can already achieve very high neuron coverage [76], [77]. DeepGauge [61] proposes a set of multi-granularity testing criteria based on neuron-based behaviors. For example, k-multisection neuron coverage (KMNC) extends the neuron coverage that it first profiles the training data and obtains the activation status of each neuron by all training data. For each neuron, the range of its activation status is partitioned into k bucket. Then, KMNC measures the ratio of all covered buckets of all neurons of a DNN by a set of test cases. Furthermore, the authors propose DeepCT [74] that considers the interactions of different neurons in a layer based on the combinatorial testing methods. Kim *et al.* [75] later proposed the coverage criteria that measure the surprise of the inputs, i.e., the distance between the inputs and the training data. The assumption is that surprising inputs introduce more diverse data such that more abnormal behaviors could be triggered.

Based on the proposed testing criteria, a number of test generation techniques [59], [60], [62], [78]–[80] are proposed for detecting defects in DNNs. Specifically, DeepXplore [59] and DeepTest [60] generate test cases based on the guidance of neuron coverage. In particular, DeepXplore adopts a differential testing method that determines whether an input is erroneous based on the cross-validation between multiple DNNs. TensorFuzz [79] and DeepHunter [62] propose the coverage-guided fuzzing techniques to test DNNs. DeepHunter integrates the coverage criteria from DeepGauge, while TensorFuzz adopts the distance-based coverage criteria. While the aforementioned techniques mainly focus on feedforward neural network (FNN), DeepStellar [78] proposes the coverage criteria and fuzzing technique for the recurrent neural network (RNN). The basic idea is to extract an abstract model from the given RNN. A set of coverage criteria then can be defined based on the abstract model.

While the existing fuzzing techniques mainly discover defects in the model, our method can be treated as a mitigation technique for repairing such potential defects. Ma et. al. [67] proposes a DNN debugging technique on FNNs, which is denoted as MODE. Given an FNN, a feature map is constructed in each layer. By selecting one layer, MODE detects the buggy weights for the given failed inputs and fixes these bugs by generating new training samples. Different from our method, MODE mainly focuses on fixing the underfitting or overfitting problems, while our method focuses on the more general problem, i.e., the distribution shift between training samples and the real-world test samples. More comprehensive discussion on the deep learning testing can be referred to the recent survey [81].

Overall, existing DNN software testing mainly focuses on detecting the defects in models. In contrast, our method aims to repair the DNNs and enhance their capability under the guidance of limited failure examples available, which can be regarded as the next step of DNN software testing.

B. DNN Software Repairing

In software engineering literature [63], there are two well-studied approaches for tackling program failures, i.e., software healing and software repairing. Software healing detects software failures in-the-field and makes amendments by responding to the failures and restoring normal operations. The key is that the amendments are not deployed at the source code level but instead deployed at runtime in order to mitigate runtime failures on the deployed applications. On the other hand, in software repairing, the amendment operations are mainly performed on the program source code level to remove fault that causes a failure. In this work, our fixing of the DNN software conceptually falls under the second category, software repair, where fixes are deployed on the source code level at testing and design time, as opposed to at runtime.

One of the recent attempts for DNN software repair is Apricot [66] that aims at fixing DL model iteratively through a weight-adaptation method. This method is based on two observations. 1) It will be increasingly more difficult for the DL model to retain a large proportion of its weights to capture all essential features as the number of inputs in the dataset T_0 grows. 2) Considering a pair of DNNs denoted as D_0 and $D_{\rm sub}$, where their training processes are identical except that model D_0 is trained on the entirety of the dataset T_0, while model $D_{\rm sub}$ is trained on a subset S_0 of T_0, and model $D_{\rm sub}$ is referred to as the reduced deep learning model (rDLM). Following this, the second observation is that each individual rDLM may not fully capture the essential features needed to classify one particular test case correctly, and if there is a set of rDLMs such that, on average, each one is more likely to classify the test case correctly, then the combined tendency of this set of rDLMs is more likely to classify this test case correctly. Apricot can adjust the weight w of D_k accordingly and it continues to train the adjusted D_k with T_0 to produce the next input model D_{k+1} for the $(k+1)$th iteration step. After all the iterations are carried out, the output of the procedure is a DL model with repaired weights.

Another recent work SENSEI [82] has proposed to improve the robust generalization of DNNs using guided test generation techniques to address the data augmentation problem for the DNNs under natural environmental variations. The proposed

data augmentation problem is cast as an optimization problem. In order to identify the worst natural environmental variant for the augmentation, each training input data goes through a space search based on the genetic algorithm (GA). The algorithm is carried out as follows. 1) At each iteration of the DNN training, for each training input data, the GA explores a small set of variants of the input and selects the locally worst one for augmentation. 2) It then uses it as the GA seed for the search in the next epoch for gradually reaching the globally worst variant without needing to explicitly evaluate all the possible variants. 3) A further heuristic-based data selection technique named selective augmentation is used to substantially reduce the DNN training time under augmentation by allowing complete skipping of a training data at certain DNN training epochs based on the DNN's current robustness around that data point. The proposed method has shown effectiveness on various image classification datasets for improving DNN robustness through GA-guided data augmentation.

Sohn *et al.* [83] have proposed a search-based automated program repair technique for DNNs called Arachne, where it directly manipulates the neural network weights and searches the space of possible DNNs instead of retraining the DNNs. The search is guided by a specifically designed fitness function following *Generate and Validate* automatic program repair (APR) techniques. The Arachne follows traditional code-based APR techniques with the following steps. 1) Arachne first adopts a fault localization technique by utilizing both positive and negative input data to retain correct behavior and to generate a patch. The representation of the patch is a set of real-numbered neural network weights. 2) Arachne then uses particle swarm optimization (PSO) as its search algorithm to update the selected neural network weights with values from the PSO candidate solution and further calculates the fitness value based on the outcomes. The Arachne approach is evaluated on three image classification tasks using undertrained DNNs to induce unexpected behavior. The repair produced by Arachne is focused more on targeted misbehavior with minimal perturbation on other behaviors. This is opposed to retraining-based DNN repairs that can alter the behavior significantly.

On a separate thrust, Islam *et al.* [84] have conducted a comprehensive study of bug repair patterns for five DNN libraries Caffe, Keras, Tensorflow, Theano, and Torch by using the DNN bugs dataset that consists of 415 bugs from Stack Overflow and 555 bugs from GitHub. The study has analyzed the following aspects of the fix patterns:

1) common bug fix patterns;
2) fix patterns across bug types;
3) fix patterns across libraries;
4) risk in fix;
5) challenges in fixing DNN bugs.

Existing works mainly aim to enhance the pretrained DNNs via the predefined training datasets while ignoring that the real-world examples may not be within the domain of training examples. Different from these DNN repairing works, e.g., SENSEI [82], our work takes a special focus on DNN repairing for the incorrect behaviors introduced by the noise patterns (that can be known or unknown) during the real-world operational environment. We take a novel style-transfer based approach to guide the data augmentation process during training. Specifically, with the limited number of collected DNN failure examples from the operational environment, we perform style transfer to guide the data augmentation so that similar failure noise patterns in the operational environment would not cause the incorrect decision of the repaired DNN.

More recently, a novel augmentation-based repairing method was proposed, called *Few-Shot guided mix (simplified as Few-Shot)* [85]. Unlike other repairing methods we mentioned before, Few-Shot method first collects failure examples and uses them to estimate the noise distribution by Gaussian mixture model (GMM). Then, it randomly samples weights according to the GMM distribution to mix the augmented examples. This process actually uses the collected failure examples to guide the data augmentation process. However, the estimated GMM is often hard to cover the main failure patterns in the collected examples, leading to some limitation of repairing performance. In contrast, our method proposes to guide the data augmentation via the style transfer method that is able to explicitly capture the main patterns in failure examples. Moreover, we identify a clustering-based strategy for style-transfer based data augmentation, preserving the key information in a wide range of failure examples.

C. Data Augmentation for DNN Enhancement

Data augmentation is used to increase the size of training data and enables a DNN to see more examples, potentially enhancing the generalization of DNNs. Recent augmentation methods [86]–[90] mainly rely on common operations, e.g., flipping, rotation, scaling, cropping, translation, etc., to transform the input training examples. For example, CutOut [86] randomly cuts a square area from the training image. MixUp [87] proposes to randomly mix two training examples and their labels together. CutMix [88] combines CutOut and MixUp together, cutting a square area from an image and filling with the same area cutting from another image. Mosaic [89] is similar to CutMix but adapts to YOLOv4, mixing four images together to produce a single image. GridMask [90] divides an image into grid and removes disconnected regions. More recently, a novel data augmentation method was proposed, i.e., AugMix [91]. It utilizes diverse augmentations, i.e., autocontrast, equalize, posterize, etc., and mixes multiple augmented examples into a single image. Moreover, it uses Jensen–Shannon divergence consistency loss for training. Xie *et al.* [92] proposed a model-based repairing technique specially designed for RNN. In particular, an abstract model is first extracted from a target RNN. To repair the incorrect behavior of the RNN on a given input, some relevant samples are then generated with the guidance of an abstract model for repairing.

Nevertheless, existing data augmentation methods mainly focus on the general robustness of deep models, ignoring the failure patterns that are not within the training dataset or covered by the augmentation operations in the real-world operational environment. On the contrary, DeepRepair aims to repair a DNN to make it robust to collected (known and unknown)

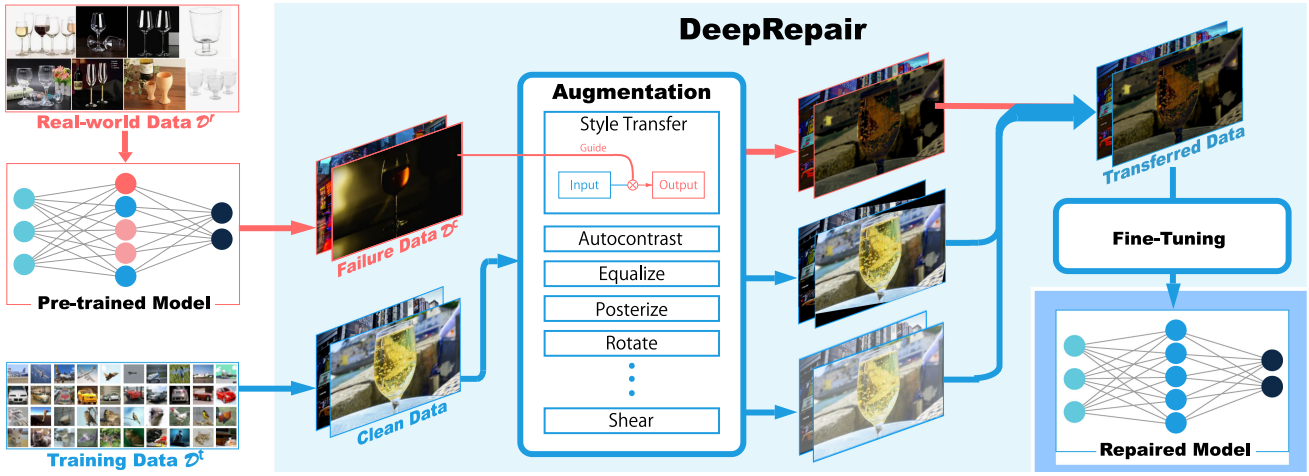


Fig. 1. Overall workflow of DeepRepair. Collected failure examples are used to learn the unknown noise patterns by the style-transfer method. Further combining known noise patterns that could be obtained by domain experts, these noise patterns are leveraged for data augmentation for effective repairing so that a DNN could be more robust to handle inputs involved with both known and unknown noises in the new environment, while maintaining the prediction performance in the original environments.

failure patterns while not harming the robustness to other failure patterns.

III. Methodology

In this section, we first formulate the problem of DNN repairing for some specified failure patterns (i.e., Section III), which frequently happened and is of great importance for real-world applications in the operational environment. Then, we present the data augmentation-based method to counteract such a problem and reveal the challenges in Section III-B. To address the challenges, we propose a style-guided data augmentation method for DNN repairing in Section III-C and introduce the detailed algorithm of our method in Section III-D. The overall workflow and major components of our proposed method DeepRepair is shown in Fig. 1.

A. Problem Formulation of DNN Repairing

Following the general DNN training process, we can train a DNN $\phi_\theta(\cdot)$ on a large-scale training dataset $\mathcal{D}^{\mathrm{t}}$ where θ represents the network's parameters. We then deploy it for the real-world applications with the assumption that the unseen data captured in the real world, whose distribution is denoted as $\mathcal{D}^{\mathrm{r}}$, has a similar distribution as the training data.

Nevertheless, in practice, it is rather difficult to construct such a perfect training dataset and a well-trained DNN might still raise errors when the input data is corrupted by some (known or unknown) noise patterns in the real-world operational environment, e.g., weather, blur, and other various kinds of noise, etc., which are dependent on the task and operational environments where the DNN is deployed.

Currently, even with the state-of-the-art DL techniques for both data and network architectures, it is still difficult to train such a DNN that can address all real-world situations under various environments with high performance (e.g., high accuracy for image recognition task). Hence, it poses a pressing problem for the DNN's real-world applications: When a well-trained DNN (in the training environment) makes incorrect predictions on some data that may have specific failure patterns (e.g., noise) in the operational environment, how could we repair them without harming its performance on other normal data? For example, given a DNN $\phi_\theta(\cdot)$ offline trained on $\mathcal{D}^{\mathrm{t}}$ and evaluated on a testing dataset $\mathcal{D}^{\mathrm{v}}$ for the image classification task, we deploy it in the real-world operational environment.

After deployment, we can find that it usually misclassifies images corrupted by noises with some kind of patterns (we may not know what the concrete noise pattern is), but we can collect some failure examples $\mathcal{D}^{\mathrm{c}}$. Then, the problem can be formulated as follows: With the $\mathcal{D}^{\mathrm{c}}$ and $\mathcal{D}^{\mathrm{t}}$, how should we improve the accuracy of $\phi_\theta(\cdot)$ on the data having the similar failure patterns with $\mathcal{D}^{\mathrm{c}}$ while not reducing the accuracy on $\mathcal{D}^{\mathrm{v}}$? Specifically, we can represent it as follows:

$$\arg\min_{\theta} \mathbb{E}_{(\mathbf{X},y)=\mathcal{T}(\{\mathcal{D}^{\mathrm{t}},\mathcal{D}^{\mathrm{c}}\})} J\left(\phi_\theta(\mathbf{X}), y\right) \tag{1}$$

where $J(\cdot)$ denotes the task-related loss function and we use cross-entropy function for the image classification task. $(\mathbf{X}, y)$ denotes an example and corresponding label from datasets. $\mathcal{T}(\{\mathcal{D}^{\mathrm{t}}, \mathcal{D}^{\mathrm{c}}\})$ defines the way of using the two datasets. For example, when we have $\{(\mathbf{X}, y) = \mathcal{T}(\mathcal{D}^{\mathrm{c}}) | (\mathbf{X}, y) \in \mathcal{D}^{\mathrm{c}}\}$, it means that we only use the collected dataset $\mathcal{D}^{\mathrm{c}}$ to fine-tune the DNN $\phi_\theta(\cdot)$. Obviously, since $\mathcal{D}^{\mathrm{c}}$ is often a small-scale dataset with limited failure patterns, the above-mentioned way (i.e., only utilizing $\mathcal{D}^{\mathrm{c}}$) would lead to an overfit DNN that has poor performance on testing dataset $\mathcal{D}^{\mathrm{v}}$.

The repairing context of our formulated problem generally applies to a wide range of scenarios in practice where there exists data distribution gap between training environment and operational environment for deployment. For example, given a well-trained DNN in environment A, when we try to deploy it to

TABLE I
ALL OPERATORS IN THE OPERATION SET $\mathcal{O}$

Operators	Meaning
Autocontrast	Normalizing image contrast
Equalize	Equalizing the image histogram
Posterize	Reducing color channels' bit number
Rotate	Rotating the image with specified angle
Solarize	Inverting all pixel values above a threshold
Shear_x	Shearing the image on X-axis
Shear_y	Shearing the image on Y-axis
$\text{Translate}_\text{x}$	Translating the image on X-axis
$\text{Translate}_\text{y}$	Translating the image on Y-axis
StyleTransfer	Transferring images w.r.t. a reference image

environment B that is similar but a little different from A, with a small number of collected DNN error examples in environment B, how could we repair the DNN so that it could perform better in the new environment B.

B. Data Augmentation-Based DNN Repairing

To avoid the overfitting issue, a simple solution is to employ data augmentation operations to extend the training dataset $\mathcal{D}^\text{t}$ and fine-tune the DNN. The intuition behind this solution is that the diverse augmentation operations (or transformations that simulate the real-world noise patterns in the operational environment) could cover the unknown failure patterns. For example, we follow the state-of-the-art AugMix method [93] that can be represented as

$$\arg\min_{\theta} \mathbb{E}_{\substack{(\mathbf{X},\mathbf{X}_{\text{aug1}},\mathbf{X}_{\text{aug2}},y)\\ =\mathcal{T}(\mathcal{D}^\text{t},\mathcal{O})}} J\left(\phi_\theta(\mathbf{X}), y\right) + \lambda \text{JS}\left((\phi_\theta(\mathbf{X}), \phi_\theta(\mathbf{X}_{\text{aug1}}), \phi_\theta(\mathbf{X}_{\text{aug2}})\right) \quad (2)$$

where $\mathcal{T}(\mathcal{D}^\text{t}, \mathcal{O})$ is to perform transformations on each sample $\mathbf{X} \in \mathcal{D}^\text{t}$ and return two augmented versions, i.e., $\mathbf{X}_{\text{aug1}}$ and $\mathbf{X}_{\text{aug2}}$, with a series of operations sampled from the operation set $\mathcal{O}$ (Table I). In the AugMix method, $\mathcal{O}$ contains some widely known noise patterns of image domain, such as rotation, equalization, translation, sharpness [93], etc. The three examples, i.e., $\mathbf{X}$, $\mathbf{X}_{\text{aug1}}$, and $\mathbf{X}_{\text{aug2}}$, share the same label y. $\text{JS}(\cdot)$ denotes the Jensen–Shannon diverse loss function that enforces the DNN predicting consistent results for original and augmented examples.

Although, in some particular domains, some common noise patterns could be manually obtained by domain experts, the more general and diverse noise patterns are often unknown and could hardly be obtained even by human experts. This brings limitations on the diversity of the augmented data with many unknown critical noise patterns missed, potentially limiting the effectiveness of data augmentation-based repairing. As a result, the fine-tuned DNNs are not repaired for handling the failures properly. To address this challenge, we originally propose the style-guided data augmentation where the unknown failure patterns are learnt by design, and further employed to guide the data augmentation, for repairing DNNs more effectively in a broader context. In particular, we leverage the style-transfer based method to learn the unknown noise patterns in a new environment from the limited size of collected failure examples. In addition, we could still leverage human summarized known noise patterns. In fact, DeepRepair is designed to take advantage of both known and unknown noise patterns for the repairing process. In the following subsection, we continue to introduce how the unknown noise patterns could be learnt by style-transfer methods.

Algorithm 1: Style-Guided DNN Repairing.

Input: Training dataset $\mathcal{D}^\text{t}$, collected failure images $\mathcal{D}^\text{c}$, pre-trained DNN $\phi_\theta(\cdot)$, style transfer method,*i.e.*, $\text{ST}(\cdot)$, augmentation operation set $\mathcal{O}$, and the pre-defined clustering number N.
Output: Repaired DNN $\phi_{\bar{\theta}}(\cdot)$.

```
# Style-guided data augmentation
Function StyleAug(X,O):
    Initialize X_aug with zeros and sample mixing
      weights (w_1, ..., w_M) ~ Dirichlet(α, ..., α);
    for m = 1, ..., M do
        Sample the first operation by O_1 ~ O;
        Sample the 2nd and 3rd operations by
          {O_2, O_3} ~ O \ O^c;
        Construct sequential operations: op_1 = O_1,
          op_12 = [O_1, O_2], and op_123 = [O_1, O_2, O_3];
        Sample one operation, i.e., op, from
          {op_1, op_12, op_123};
        Conduct augmentation via op and add it to
          X_aug with X_aug += w_m op(X);
    Sample blending weight w_0 by w_0 ~ Beta(α, α);
    Blend with X by X_mix = w_0 · X_aug + (1 − w_0) · X;
    return X_mix;
# Augmentation operation extension via the style transfer
Perform K-means clustering on D^c with number N ;
Construct the sampling strategy P^c via Eq. 4 ;
Identify N operations O^c = {O_i^c} via Eq. (5) ;
Update the operation set O by adding O^c to O ;
# Data augmentation for DNN repairing
for j = 1 to |D^t| do
    Loading the jth image X from D^t ;
    Calculating two augmented images: X_aug1 =
      StyleAug(X,O) , and X_aug2 = StyleAug(X,O) ;
    Calculating the loss function:
      J(φ_θ(X), y) + λJS((φ_θ(X), φ_θ(X_aug1), φ_θ(X_aug2)));
    Updating the parameters θ of φ_θ(·) ;
```

C. Style-Guided Data Augmentation for DNN Repairing

Following the objective function in Section III-B, we focus on automatically learning novel data augmentation operations to $\mathcal{O}$ for DNN repairing with the guidance of collected failure examples, i.e., corrupted data $\mathcal{D}^\text{c}$. To this end, we propose the very first style-transfer based data augmentation operations for repairing. Style transfer is to map an image to a new one having similar style with a given reference image. As shown in Fig. 2(a), source images can be transferred to very similar styles

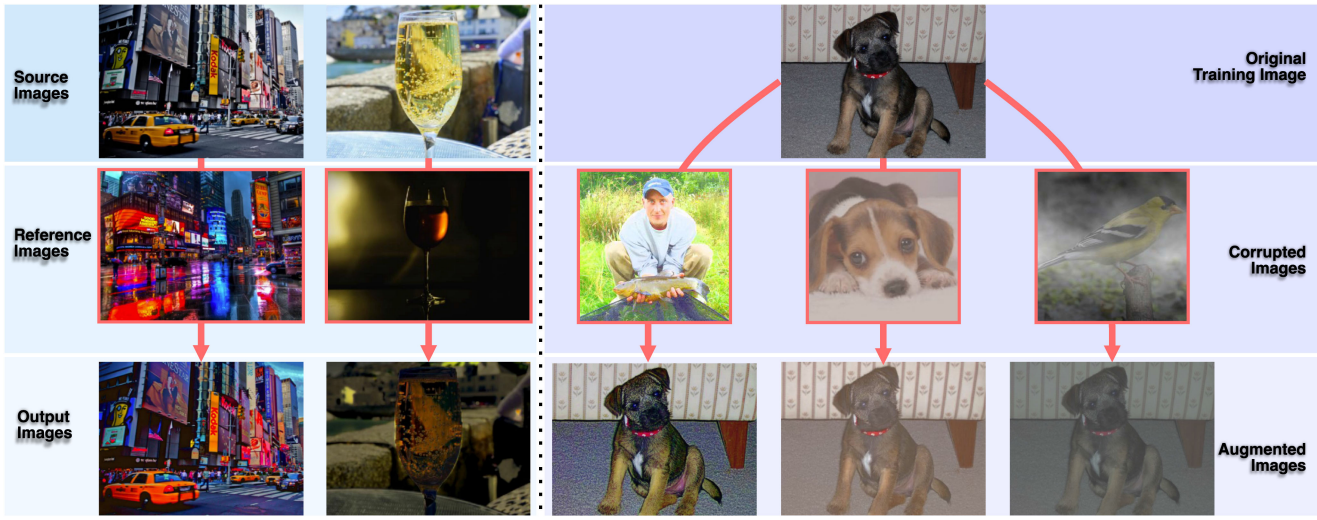


Fig. 2. Two examples (a) of style transfer [70] and three examples (b) of our style-guided data augmentation for DNN repairing. (a) Style Transfer. (b) Style-Guided Data Augmentation.

with the given style images while the original details are all preserved. Intuitively, we could employ style transfer as novel data augmentation operations by regarding the corrupted data as reference images. Specifically, we represent the new operations as

$$\mathrm{O}_i^{\mathrm{c}}(\mathbf{X}) = \mathrm{ST}(\mathbf{X}, \mathbf{X}^i), \text{ with } \mathbf{X}^i = \mathrm{Sample}(\mathcal{D}^{\mathrm{c}}, \mathcal{P}) \quad (3)$$

where $\mathbf{X}$ is the example that needs to be augmented, the image $\mathbf{X}_i$ is one of the collected failure images, and the $\mathrm{ST}(\cdot)$ denotes the style transfer method. Here, we adopt the style transfer method WCT^2 [70]. The $\mathrm{Sample}(\mathcal{D}^{\mathrm{c}}, \mathcal{P})$ denotes the sampling strategy based on $\mathcal{P}$. The notation $\mathcal{P} = \{P^i\}$ defines the sampling probability of all samples $\{\mathbf{X}^i\}$ and we preliminarily use the uniform sampling strategy with $\{P^i = \frac{1}{|\mathcal{D}^{\mathrm{c}}|}\}$. As shown in Fig. 2(b), the example with a dog is augmented according to three reference images that are predicted erroneously by over brightness, low contrast, and fog, and these failure patterns are successfully included into the augmented images. We then add all style-transfer based operations, i.e., $\mathcal{O}^{\mathrm{c}} = \{\mathrm{O}_i^{\mathrm{c}} | i = [1, \ldots, |\mathcal{D}^{\mathrm{c}}|]\}$, to the operation set $\mathcal{O}$ and conduct the DNN repairing by fine-tuning DNNs via (2). However, there is still another challenge for DNN repairing. In particular, the failure images in $\mathcal{D}^{\mathrm{c}}$ are diverse. Thus, for an image $\mathbf{X}$, it is time-consuming to select all failure images as the reference. Thus, it is difficult to select which images in $\mathcal{D}^{\mathrm{c}}$ should be the references.

To alleviate this issue, we further implement the clustering-based reference image generation, where the sampling probability of each sample is determined by their distance to the clustering center. In particular, we first perform k-means clustering on the $\mathcal{D}^{\mathrm{c}}$ with the number of clusters N and get N subsets denoted as $\{\mathcal{D}^{\mathrm{c}_i} | i = [1, N]\}$ with their clustering centers being $\mathcal{C}^{\mathrm{c}} = \{\mathbf{X}_{\mathrm{cls}}^i | i = [1, N]\}$. Then, for the ith clustering set (i.e., $\mathcal{D}^{\mathrm{c}_i}$), we calculate L_2 distance between samples in $\mathcal{D}^{\mathrm{c}_i}$ and the clustering center $\mathbf{X}_{\mathrm{cls}}^i$, which is represented as $\{d_j^i = \|\mathbf{X}_j^i - \mathbf{X}_{\mathrm{cls}}^i\|_2 | \mathbf{X}_j^i \in \mathcal{D}^{\mathrm{c}_i}\}$ and we define the sampling probability of $\mathbf{X}_j^i$ as

$$P_j^i = \frac{1}{N}\left(1 - \frac{d_j^i}{\sum_{k=1}^{|\mathcal{D}^{\mathrm{c}_i}|} d_k^i}\right) \quad (4)$$

where $|\mathcal{D}^{\mathrm{c}_i}|$ is the size of $\mathcal{D}^{\mathrm{c}_i}$ and P_j^i is the probability of $\mathbf{X}_j^i$ to be sampled for guiding data augmentation. Intuitively, the sample near to clustering center has higher probability to be selected. We define $\mathcal{P}^c = \{P_j^i\}$ as the clustering-guided sampling strategy and reformulate (3) as

$$\mathrm{O}_i^{\mathrm{c}}(\mathbf{X}) = \mathrm{ST}(\mathbf{X}, \mathbf{X}^i), \text{ with } \mathbf{X}^i = \mathrm{Sample}(\mathcal{D}^{\mathrm{c}}, \mathcal{P}^{\mathrm{c}}). \quad (5)$$

D. DNN Repairing Algorithm

Algorithm 1 gives the details of DeepRepair, which corresponds to Fig. 1 According to the algorithm, the workflow of DeepRepair could be roughly decomposed into two key stages: 1) extending the augmentation operation set $\mathcal{O}$ via the style transfer based on collected failure cases (Lines 2–12); 2) conducting the data-augmentation for DNN repairing through the style-guided data augmentation (Lines 14–22). We first introduce the style-guided data augmentation, i.e., `StyleAug`$(\cdot)$ in Algorithm 1. Intuitively, given an input image $\mathbf{X}$, we obtain M augmentations via the sequential operations sampled from the set $\mathcal{O}$ and then mix them up together with weights from Dirichlet and Beta distributions. We use these two distributions since their capability for data augmentation has been validated in AugMix [91] and MixUp [87].

The style-guided augmentation is based on the framework AugMix [91]. More specifically, we obtain M weights via Dirichlet distribution with α as the parameter (i.e., the fourth line in Algorithm 1) and then perform M augmentations (i.e., lines 5–10 in Algorithm 1). For each augmentation, we first sample the first operation (i.e., O_1) from $\mathcal{O}$, which might be the

style-transfer based operations in $\mathcal{O}^{\mathrm{c}}$ or other general operations in $\mathcal{O} \setminus \mathcal{O}^{\mathrm{c}}$ (e.g., rotation and translation). Then, we sample the second and third operations from $\mathcal{O} \setminus \mathcal{O}^{\mathrm{c}}$ and get $\{\mathrm{O}_2, \mathrm{O}_3\}$. All sampled operations are sequentially composed, resulting in three sequential operations, one of which is selected for the final augmentation. The above process is shown from lines 6–10 in Algorithm 1 with the following principles. 1) The first operation could be based on style transfer, using to embed the failure pattern in collected failure examples (i.e., $\mathcal{D}^{\mathrm{c}}$) into the example $\mathbf{X}$. Note that we also allow the first operation to be other general operations to avoid the risk of overfitting on the specific pattern in $\mathcal{D}^{\mathrm{c}}$. 2) The second and third operations are limited to be general operations (i.e., $\mathcal{O} \setminus \mathcal{O}^{\mathrm{c}}$), simulating the transformations on the style transferred example, i.e., by translation, rotation, equalization, sharpness, etc. Besides, since style transfer [70] can be much slower than the general operations, O_2 and O_3 also avoid great time cost with only style-based augmentation.

During training (lines 19–22), we first collect all transformation sets including the style-based transformation and the general transformation (line 17). For each training data $\mathbf{X}$, we calculate two augmented images $\mathbf{X}_{\mathrm{aug1}}$ and $\mathbf{X}_{\mathrm{aug2}}$, and the model should have similar predictions on $\mathbf{X}$, $\mathbf{X}_{\mathrm{aug1}}$, and $\mathbf{X}_{\mathrm{aug2}}$ (line 22).

IV. Experimental Design and Settings

In this section, we first perform a preliminary study (i.e., **RQ1**) to confirm that whether different failure patterns have different effects on the accuracy of pre-trained deep models. Then, we conduct large-scale experiments to validate the proposed methods and investigate the following research questions:

1) **RQ2.** Does DeepRepair outperform state-of-the-art data-driven repairing methods on the examples with specific failure patterns?
2) **RQ3.** Does DeepRepair harm the robustness to other failure patterns and clean data?
3) **RQ4.** Do our proposed components of DeepRepair all contribute the final repairing performance (i.e., accuracy on different failure patterns)?

In particular, RQ2 intends to evaluate the behavior of DeepRepair by comparing with baseline repairing methods and data augmentation approaches. RQ3 is to explore whether the DeepRepair method under the guidance of one failure pattern could harm DNNs' capability of handling other failure patterns. RQ4 is to analyze the contributions of different components of DeepRepair. We present all raw data in the following figures in [94].

A. Experimental Setups

To answer the above four research questions, we consider the following setups on dataset, DNN architectures, related hyperparameters, etc.

Datasets. Following the formulations in Section III, we employ the training dataset CIFAR-10 as the $\mathcal{D}^{\mathrm{t}}$ in Section III and extend its testing dataset, i.e., $\mathcal{D}^{\mathrm{v}}$, via various failure types to validate our method. Specifically, we first train a DNN (i.e., ϕ_θ) on the CIFAR-10's training dataset and select 15 failure types (i.e., 15 different failure patterns) [95]. These 15 failure types come from CIFAR-10-C, a public dataset containing different corrupted images. These corruptions can generally be divided into four types, i.e., noise, blur, weather, and digital corruption, all of which commonly exist in the real world. These corrupted data are the superb substitute of real-world data as it is often hard and expensive to collect them that could reduce the accuracy of DNN significantly. Note that each of the 15 failure datasets contains five different severity levels of corrupted images. Then, following the setting in Hendrycks's paper [95], We apply the 15 potential failure patterns to $\mathcal{D}^{\mathrm{v}}$, respectively, and generate 15 new testing datasets that are denoted as $\{\mathcal{D}^{\mathrm{v}_k} | k \in [1, \ldots, 15]\}$, each of which is five times larger than $\mathcal{D}^{\mathrm{v}_k}$ and has 50 000 images since we consider five different severity levels for each pattern. Note that some of the 50 000 images may not be failures on the DNN. Thus, for the kth failure pattern, we evaluate the pretrained DNN on all generated images $\mathcal{D}^{\mathrm{v}_k}$ and identify the failure cases. From the failure cases, we randomly select 1000 failure cases as the dataset $\mathcal{D}^{\mathrm{c}_k}$, while the residual failure cases form the dataset $\mathcal{D}^{\mathrm{e}_k}$ (i.e., unknown failure cases). Table III shows the detailed number of each failure type for different models, where Column $\mathcal{D}^{\mathrm{c}_k} + \mathcal{D}^{\mathrm{e}_k}$ represents the number of all failure cases. Note that, the number of $\mathcal{D}^{\mathrm{c}_k}$ is always 1000 no matter in which degradation, and the number of $\mathcal{D}^{\mathrm{e}_k}$ is greater than that of $\mathcal{D}^{\mathrm{c}_k}$. Intuitively, in terms of the kth failure type, our method is to repair the DNN $\phi_\theta(\cdot)$ to make it achieve high accuracy on the corresponding failure dataset $\mathcal{D}^{\mathrm{e}_k}$ with the guidance of $\mathcal{D}^{\mathrm{c}_k}$ while not harming the accuracy on other failure and the original testing datasets. Hence, we use the accuracy of repaired DNN on $\{\mathcal{D}^{\mathrm{e}_k} | k \in [1, \ldots, k]\}$ to evaluate the performance of DNN repairing methods.

DNN architectures. We select three different state-of-the-art CNN-based architectures (i.e., all convolution network (AllConvNet) [96], DenseNet [97], and wide residual net (WideResNet)) [98] as the DNNs to be repaired. Besides, we also test our method on two RNN-based architectures, long short-term memory (LSTM) [99] and ReNet [100]. For each architecture, we first pretrain them with original CIFAR-10's training set (i.e., $\mathcal{D}^{\mathrm{t}}$), and the model with the highest accuracy in testing set (i.e., $\mathcal{D}^{\mathrm{v}}$) will be saved. For the AllConvNet [96], we employ the same architecture as the authors provided in their article. In terms of DenseNet [97], we use its standard configurations for CIFAR-10, setting its depth as 40 and growth rate as 12. We also implement the standard WideResNet [98] whose width factor is set to 1 and the number of blocks is 1. For LSTM [99], we adopt the simplest architecture, which only contains one LSTM layer and two fully connected layers. In terms of the ReNet [100], we use its standard architecture for CIFAR-10, as introduced in the original article. The details about these five architectures can be found in Table II.

Hyperparameters. In terms of the training setup, we employ stochastic gradient descent optimizer with a batch size of 128, the learning rate of 0.1, and decay of 0.0005. Jensen–Shannon divergence will be used as the loss function. The max epoch number is 500, and the training will stop if validation loss does not decrease in 10 epochs. As for AugMix, mixture width is set as 3, and mixture depth is randomly changed between 1 and 3. We set nine base operations (i.e., autocontrast, equalize, posterize, rotate, solarize, shear-x, shear-y, translate-x, and translate-y) to

TABLE II
SUMMARY OF THREE CNN-BASED AND TWO RNN-BASED SUBJECT DEEP NEURAL NETWORKS

	Subject DNN	#Neuron	#Layer	#Parameter
CNN	**AllConvNet [96]**	1,516,426	19	1,409,674
	DenseNet [97]	5,518,858	79	1,059,298
	WideResNet [98]	1,409,034	76	2,243,546
RNN	**LSTM [99]**	4,234	3	397,706
	ReNet [100]	245,800	6	67,153,930

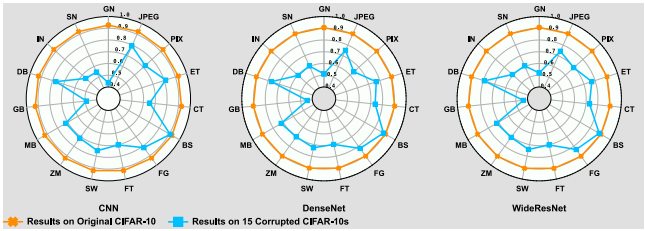


Fig. 3. Accuracy of the original DNNs on CIFAR-10's testing dataset (i.e., $\mathcal{D}^{\mathrm{v}}$) and on 15 corrupted testing datasets (i.e., $\{\mathcal{D}^{\mathrm{v}_k}|k=1,\ldots,15\}$).

the operation set $\mathcal{O}$ in Algorithm 1 with additional operations proposed in Section III-D. We set the number of clusters, i.e., N, in Section III-C as 5.

Other configurations. We implement DeepRepair in Python based on PyTorch framework. All the experiments were performed on a server with the Ubuntu 16.04 system with a 12-core 3.6 GHz Xeon CPU, 126 GB RAM, and two NVIDIA GeForce RTX 2080 Ti 12 G GPUs.

Baselines. To demonstrate the advantage and usefulness of DeepRepair, we consider two kinds of baselines for comparative studies. The first set of baselines is the general state-of-the-art data augmentation methods, i.e., AugMix [91], CutMix [88], CutOut [86], and MixUp [87]. The second set of baselines is recently proposed state-of-the-art DNN repairing methods, i.e., SENSEI [101] and Few-Shot [85]. In particular, for the first four data augmentation methods, we perform DNN repairing by using them to fine-tune the DNNs with the original training dataset (i.e., $\mathcal{D}^{\mathrm{t}}$) and 1000 collected failure cases (i.e., $\mathcal{D}^{\mathrm{c}_k}$). For the SENSEI and Few-Shot methods, we also employ the failure cases during repairing for a fair comparison.

V. EXPERIMENTAL RESULTS

A. RQ1. What are the Effects of Different Failure (Noise) Patterns (i.e., Failure Types) on DNN Operational Inference?

In this part, we train five DNNs, among which three are CNN-based models, i.e., AllConvNet, DenseNet, and WideResNet, and two are RNN-based models, i.e., LSTM and ReNet, on the CIFAR-10's training dataset (i.e., $\mathcal{D}^{\mathrm{t}}$) and evaluate them on both original testing dataset (i.e., $\mathcal{D}^{\mathrm{v}}$) and the 15 extended testing datasets (i.e., $\{\mathcal{D}^{\mathrm{v}_k}|k=1,\ldots,15\}$). The evaluation results of two CNN-based models are summarized in Fig. 3. Overall, we have the following observations: 1) Different failure types pose different degradation effects on the DNNs pretrained on the original training dataset, i.e., $\mathcal{D}^{\mathrm{t}}$. For example, the accuracy of AllConvNet on the original testing dataset is around 93.0% while the JPEG and Gaussian noise (GN) reduce the results to around 80.0% and 43.0%, respectively, indicating that we should repair DNNs by considering the difference across different failure types. A simple way is to use collected failure cases that may be degraded by an unknown failure type as the guidance for repairing. However, in real-world applications, it is hard to collect a large number of failure cases under similar unknown failures. Hence, it is significantly important but difficult to explore an effective method that is able to repair the DNNs against an unknown failure type (pattern) according to a few collected failure cases. In this article, we propose a novel DNN repairing method to address this problem and validate the effectiveness in the following experiments. 1) The accuracy reductions on the same DNN have large diversity under different failures. For example, with the Gaussian noise, the accuracy reduction is around 50%, while the value remains almost unchanged with the brightness failure, indicating that the DNN may have very different results on different failures. Thus, the DNN repairing guided by a kind of failure should not affect the performance on other failure types and the original clean inputs.

B. RQ2. Does the Proposed Method Outperform State-of-the-Art Data-Driven Repairing Methods on the Examples With Specific Failure Patterns?

For the kth failure type, we repair three CNN-based models, i.e., AllConvNet, DenseNet, and WideResNet, and two RNN-based models, i.e., LSTM and ReNet, trained in Section V-A via the six baseline methods and the proposed method. Then, we evaluate their performance by calculating the accuracy of repaired DNNs on the failure cases, i.e., $\mathcal{D}^{\mathrm{e}_k}$ and show the results on 15 failure types in Table IV. In general, our method (i.e., DeepRepair) exhibits significant advantages over all baseline methods on the three CNN-based and two RNN-based architectures under 15 failure types, demonstrating the effectiveness and generalization of the proposed method.

In particular, comparing with the state-of-the-art DNN repairing methods (i.e., Few-Shot and SENSEI), DeepRepair achieves much higher accuracy on all three DNNs under 15 failures. In particular, on the glass blur (GB), motion blur (MB), and zoom (ZM), DeepRepair has achieved over 500% relative improvements on SENSEI, demonstrating the effectiveness and advantages of our method. In terms of other data augmentation based methods, the results on the AllConvNet present that DeepRepair has much higher accuracy than all other augmentation methods under all 15 failure types. Even the state-of-the-art AugMix method still has huge accuracy gaps compared to our method. Nevertheless, as the DNN becomes more powerful (i.e., from AllConvNet to WideResNet), the capability of AugMix on repairing is significantly enhanced and its accuracy under several failure types (e.g., pixelate, snow, impulse noise, defocus blur, MB, ZM, etc.) can be slightly larger than our method, indicating that AugMix is more suitable for repairing elaborately designed DNN architectures while our method obtains consistent effectiveness on the three kinds of DNNs.

TABLE III
NUMBER OF ALL FAILURE CASES (I.E., $\mathcal{D}^{C_k} + \mathcal{D}^{E_k}$), WHICH IS THE SUM OF COLLECTED FAILURE CASES FOR REPAIRING GUIDANCE (I.E., $\mathcal{D}^{C_k}$, WHICH WILL ALWAYS BE 1000), AND RESIDUAL CASES FOR REPAIRING EVALUATION (I.E., $\mathcal{D}^{E_k}$) OF FIVE PRETRAINED DNNS, THREE ARE CNN-BASED DNNS, I.E., ALLCONVNET, DENSENET, AND WIDERESNET, AND TWO ARE RNN-BASED DNNS, I.E., LSTM AND RENET

	Dataset	WideResNet $\mathcal{D}^{C}k + \mathcal{D}^{e}k$	DenseNet $\mathcal{D}^{C}k + \mathcal{D}^{e}k$	AllConvNet $\mathcal{D}^{C}k + \mathcal{D}^{e}k$	LSTM $\mathcal{D}^{C}k + \mathcal{D}^{e}k$	ReNet $\mathcal{D}^{C}k + \mathcal{D}^{e}k$
CIFAR-10-C	Gaussian noise (GN)	24,026	24,458	28,218	20,713	16,266
	Shot noise (SN)	19,701	19,727	22,561	20,484	16,282
	Impulse noise (IN)	19,591	20,189	22,074	21,438	17,122
	Defocus blur (DB)	10,398	11,465	11,497	20,365	17,461
	Glass blur (GB)	28,162	27,844	25,588	21,176	17,535
	Motion blur (MB)	14,055	13,937	14,058	20,970	19,163
	Zoom (ZM)	14,625	14,895	14,247	21,178	17,705
	Snow (SW)	12,907	13,613	12,359	24,086	19,691
	Frost (FT)	14,840	15,082	14,856	25,800	23,717
	Fog (FG)	8,222	8,812	9,207	30,740	27,349
	Brightness (BS)	5,515	5,735	4,614	22,416	19,551
	Contrast (CT)	13,466	13,067	17,320	31,380	31,659
	Elastic Transform (ET)	11,490	11,569	9,359	22,083	18,488
	Pixelate (PIX)	15,241	17,865	13,901	20,293	16,468
	JPEG Compression (JPEG)	12,864	12,451	10,377	20,456	16,269

TABLE IV
ACCURACY OF REPAIRED DNN ARCHITECTURES ON 15 FAILURE DATASETS, I.E., $\{\mathcal{D}^{E_k} | k = [1, 15]\}$

	Repair Method	GN	SN	IN	DB	GB	MB	ZM	SW	FT	FG	BS	CT	ET	PIX	JPEG
AllConvNet	Cutout	6.50	5.99	17.95	16.19	10.01	9.85	7.13	17.31	10.25	14.79	25.71	10.40	19.97	13.49	16.98
	Mixup	1.26	17.51	3.15	10.41	7.42	8.75	8.47	9.68	14.08	12.97	15.66	9.35	8.12	9.44	9.24
	CutMix	12.80	5.90	12.86	12.46	8.70	8.75	8.94	15.69	12.46	11.41	15.91	7.29	8.24	14.71	20.40
	AugMix	36.19	32.40	41.23	47.91	40.74	39.55	48.13	38.51	41.84	29.86	33.26	29.88	34.80	44.56	40.36
	Few-shot	15.65	19.41	13.48	11.00	9.84	11.42	8.90	17.73	16.68	17.39	15.19	12.08	13.31	16.56	16.10
	SENSEI	18.85	25.13	14.27	11.67	7.29	9.02	8.41	15.21	13.18	15.72	20.78	9.55	13.72	16.05	16.18
	DeepRepair	55.19	61.32	50.98	68.98	56.91	61.97	67.34	59.70	63.89	49.08	44.74	34.90	51.03	55.69	46.14
DenseNet	Cutout	11.58	12.73	23.25	13.25	8.42	13.57	10.14	25.71	16.45	16.54	39.92	13.72	27.16	17.13	24.85
	Mixup	14.72	17.13	3.97	8.17	3.48	7.19	6.04	6.90	12.14	9.63	5.53	6.31	11.30	12.60	11.18
	CutMix	15.08	18.12	17.86	8.65	11.31	10.55	6.05	14.98	10.96	13.24	17.99	7.95	11.83	6.56	16.75
	AugMix	48.18	53.62	57.66	64.63	52.94	60.61	63.92	57.23	62.11	48.37	50.90	58.36	55.29	65.08	54.71
	Few-shot	13.34	12.94	11.65	12.18	10.24	9.75	11.98	14.58	13.01	15.63	16.94	10.09	12.20	14.72	15.26
	SENSEI	21.24	16.54	17.28	15.50	11.94	7.39	10.31	16.11	16.08	14.34	17.17	7.52	14.33	17.39	18.78
	DeepRepair	61.51	64.16	57.93	71.97	61.40	65.01	73.14	67.73	64.00	57.81	56.94	67.00	62.56	60.32	63.41
WideResNet	Cutout	13.06	13.17	25.02	14.95	10.28	12.48	12.87	26.45	15.25	22.53	39.23	16.13	21.23	20.70	29.00
	Mixup	11.57	13.66	11.21	7.44	3.50	5.71	16.12	6.21	7.88	7.48	13.11	4.68	6.04	5.65	8.64
	CutMix	11.43	10.62	19.20	10.70	6.71	6.56	5.84	14.50	10.10	11.55	19.07	14.19	10.30	13.10	12.98
	AugMix	65.59	69.73	71.57	79.36	65.95	77.72	81.17	68.77	69.99	62.09	65.65	69.70	68.97	63.44	65.64
	Few-shot	12.52	15.13	11.10	10.85	13.28	10.32	12.54	10.28	12.73	12.36	14.04	6.20	12.83	12.21	12.62
	SENSEI	10.60	5.21	4.75	14.49	8.27	10.38	12.40	9.47	7.33	10.00	10.74	10.68	17.27	13.96	18.96
	DeepRepair	67.80	69.44	71.10	74.99	66.41	77.33	80.42	71.26	74.13	67.09	62.37	64.54	67.10	67.73	65.03
LSTM	Cutout	28.05	28.09	23.90	24.33	24.88	24.38	23.69	30.03	25.44	14.94	29.35	10.77	22.97	25.37	30.47
	Mixup	22.83	20.45	8.22	18.00	22.11	17.89	8.41	22.29	18.98	12.51	23.55	8.45	17.91	20.22	21.05
	CutMix	26.96	29.55	27.59	23.84	22.80	22.77	21.44	27.59	22.33	13.32	25.46	10.68	22.40	26.11	26.66
	AugMix	41.77	42.63	41.02	43.21	41.48	41.87	42.67	40.13	38.34	34.86	42.45	30.53	42.23	43.65	43.31
	Few-shot	14.75	14.41	14.21	14.28	13.76	13.88	13.60	13.38	13.34	10.06	14.11	11.51	13.34	14.39	14.49
	SENSEI	10.08	10.22	9.88	11.20	11.09	11.06	11.95	13.55	13.83	12.06	13.37	11.90	12.65	10.95	10.35
	DeepRepair	46.16	46.80	43.99	47.62	40.77	44.27	47.84	41.19	42.66	43.11	42.82	48.91	45.54	48.01	39.20
ReNet	Cutout	23.42	23.49	22.95	21.04	20.79	21.03	23.08	18.08	14.29	15.53	19.65	11.24	21.65	24.52	23.40
	Mixup	18.05	17.87	17.93	19.59	18.22	17.13	17.17	17.62	19.03	13.16	18.02	10.57	17.47	18.38	19.27
	CutMix	21.20	19.91	19.19	17.62	19.72	20.01	20.69	18.08	14.40	13.34	16.34	10.32	17.28	19.80	18.30
	AugMix	35.99	37.22	34.53	31.77	31.61	31.44	32.71	27.52	24.87	25.64	28.17	26.02	30.08	31.45	30.39
	Few-shot	19.34	19.83	20.21	20.07	19.64	18.69	20.32	18.48	14.08	17.60	17.77	16.08	19.19	20.26	19.63
	SENSEI	17.17	17.10	17.16	16.48	16.64	16.28	16.14	14.75	12.47	13.41	14.32	13.22	16.40	16.94	16.94
	DeepRepair	33.20	32.23	30.86	34.20	34.12	35.61	36.46	31.12	34.85	32.58	32.57	35.45	32.78	34.45	31.11

Note: We choose CutMix, Mixup, Cutout, Few-Shot, SENSEI, and AugMix as baseline methods and fine-tune them with DeepRepair under 15 failure patterns, respectively. We highlight the best results with red.

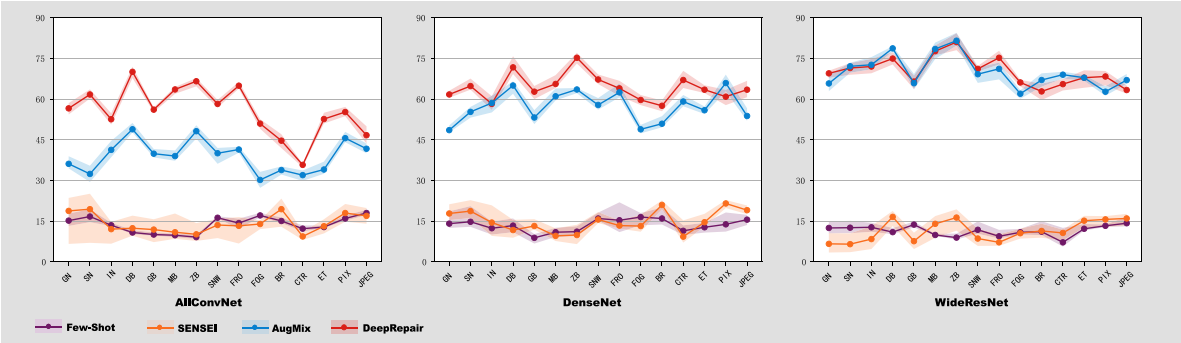


Fig. 4. We train three CNN-based DNNs using three baselines, i.e., Few-shout, SENSEI, AugMix, and the proposed method, DeepRepair. We repeat repairing period for five times and record the accuracy on 15 degradation sets. In the chart, the line means average accuracy on specific degradation set for five times; the band area shows the highest and the lowest accuracies for five times.

Moreover, to counteract the randomness impacts of data augmentation to the evaluation results, we use two repairing methods (i.e., Few-Shot and SENSEI), the best augmentation-based method (i.e., AugMix), and the proposed method to repair three CNN-based DNNs (i.e., AllConvNet, DenseNet, and WideResNet) for five times independently. Then, we present the averaged accuracy and the corresponding variances after repairing of the 15 failure patterns in Fig. 4. In summary, we have the following observations. 1) According to the average accuracy, DeepRepair outperforms the two repairing baselines (i.e., Few-Shot and SENSEI) on all three CNN models and 15 failure patterns. DeepRepair also achieves higher average accuracy than AugMix on the AllConvNet and DenseNet for most of the failure patterns. 2) In terms of the variance results, DeepRepair and AugMix obtain much smaller variance than the SENSEI and Few-Shot methods, indicating that DeepRepair and AugMix are much stable than the two repairing baseline methods.

C. RQ3. Does DeepRepair Harm the Robustness to Other Failure Patterns and the Accuracy on Clean Images?

A well-repaired DNN should achieve much higher accuracy on the target failure patterns while not harming the accuracy on the original clean images and the robustness to other kinds of failure (noise) patterns.

To validate the first capability of our method, we evaluate the repaired DNN under the kth failure pattern on the original testing dataset (i.e., $\mathcal{D}^{\mathrm{v}}$) and compare with the original DNN. We present the results in Fig. 5. Specifically, for the kth axes, we show the evaluation results of the original and the repaired DNN on the $\mathcal{D}^{\mathrm{v}}$ and the accuracy of repaired DNN on the $\mathcal{D}^{\mathrm{v}_k}$. We have the following observations. 1) Comparing the accuracy difference of DNNs on the original testing dataset (i.e., green line for the repaired DNN and yellow line for the original DNN) and the unknown failure datasets (i.e., red line for the repaired DNN and black line for the original DNN), we observe that the accuracy difference has significantly decreased after repairing, demonstrating that the proposed method could effectively repair the original DNN. 2) Comparing the results on the original testing dataset $\mathcal{D}^{\mathrm{v}}$ (i.e., the yellow line and the green line), we

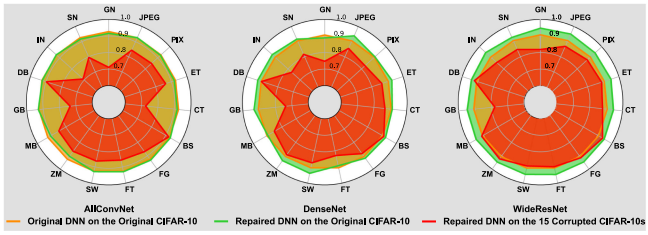


Fig. 5. Accuracy of the original and repaired DNNs on the original CIFAR-10's testing dataset (i.e., $\mathcal{D}^{\mathrm{v}}$) and 15 extended testing datasets (i.e., $\mathcal{D}^{\mathrm{v}_k}$), respectively.

see that all repaired DNNs do not harm the accuracy of the original DNN on the clean images and even achieve much higher accuracy when we repair the DenseNet and WideResNet.

To validate the second capability of our method, we take the AllConvNet as an example and evaluate the accuracy of the repaired DNN based on one failure pattern on other failure datasets to show whether the repaired model could have higher robustness on other failure patterns or not. As shown in Fig. 6, DeepRepair makes the repaired DNN under one failure pattern achieve similar significant accuracy enhancement on other failure datasets, outperforming all baseline methods. Overall, the two experiments in Figs. 5 and 6 demonstrate that DeepRepair can effectively repair the DNN while not harming the accuracy on the clean dataset as well as the robustness on other failure pattern based datasets.

D. RQ4. Do the Proposed Components of DeepRepair All Contribute the Final Accuracy?

To demonstrate the effectiveness of our clustering-based style-guided data augmentation for DNN repairing, we conduct an ablation study by repairing pretrained AllConvNet, DenseNet, and WideResNet models with two variants of our method. The first one performs the style transfer on randomly selected failure cases to guide the repairing process without using the clustering method and we denote this variant as 'no-cluster' in Table V and Fig. 7. The second one is our final version while the clustering is first done on the collected failure cases and we call this variant as 'cluster' in Table V and Fig. 7. Specifically,

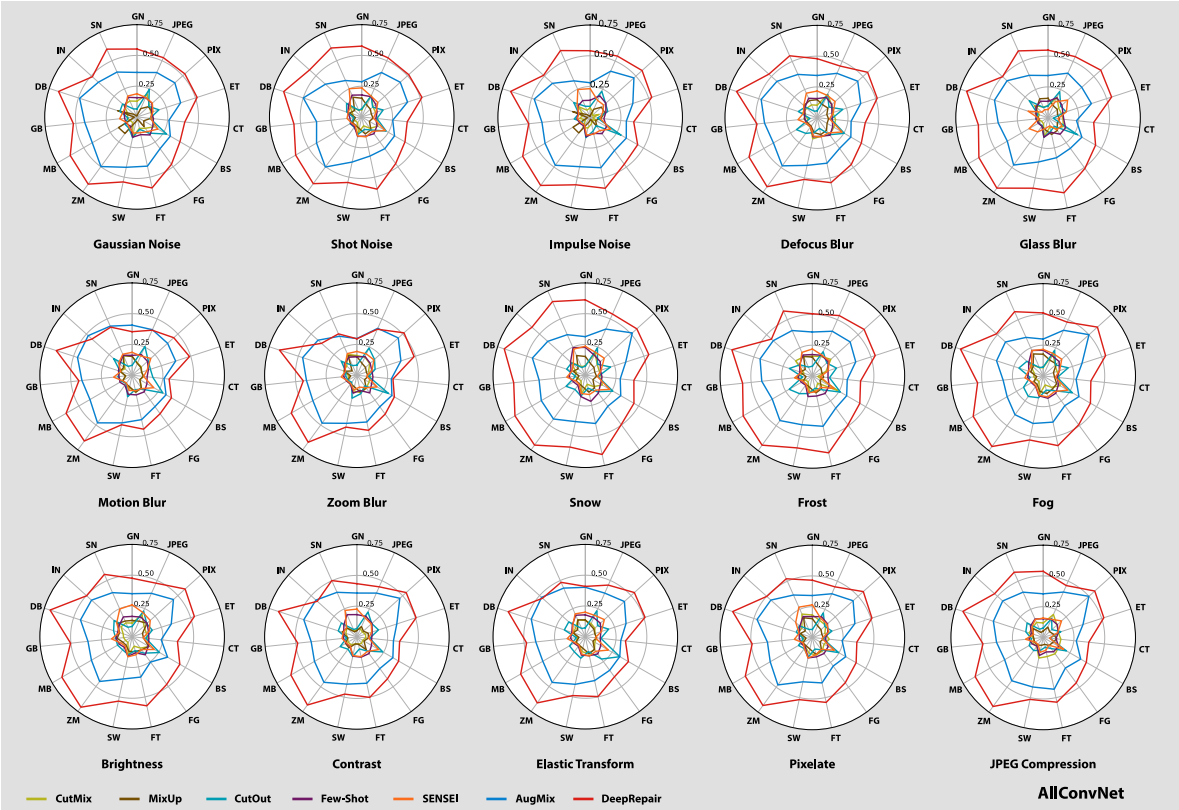


Fig. 6. Comparing the repairing methods on AllConvNet by evaluating the accuracy of repaired DNN under one failure pattern (i.e., the name at the bottom of each subfigure) on other failure datasets, i.e., $\{\mathcal{D}^{\mathrm{e}k}|k=[1,15]\}$. Please find raw data in [94].

TABLE V
ABLATION STUDY OF DEEPREPAIR

Repair Method		GN	SN	IN	DB	GB	MB	ZM	SW	FT	FG	BS	CT	ET	PIX	JPEG
AllC.	no-cluster	58.13	61.27	48.47	69.59	57.35	54.95	59.17	59.28	63.78	49.91	44.85	33.82	47.57	52.57	45.73
	cluster	55.19	61.32	50.98	68.98	56.91	61.97	67.34	59.70	63.89	49.08	44.74	34.90	51.03	55.69	46.14
Den.	no-cluster	43.93	61.10	58.66	72.96	60.54	62.05	74.49	67.34	61.59	55.97	55.78	63.64	61.42	65.87	63.18
	cluster	61.51	64.16	57.93	71.97	61.40	65.01	73.14	67.73	64.00	57.81	56.94	67.00	62.56	60.32	63.41
W.Res	no-cluster	62.72	50.80	65.61	76.13	64.53	75.46	79.16	69.47	70.18	64.83	61.37	66.45	66.88	61.23	65.21
	cluster	67.80	69.44	71.10	74.99	66.41	77.33	80.42	71.26	74.13	67.09	62.37	64.54	67.10	67.73	65.03

Note: We consider two variants. The first uses uniform sampling to select reference images for style-guided data augmentation [i.e., (3)] and we denote it as "no-cluster." The second one uses (5) for clustering-guided sampling and we denote it as "cluster." We compare the two methods via the accuracy of repaired DNNs on $\{\mathcal{D}^{\mathrm{e}k}|k=[1,15]\}$. We highlight the best result with red.

given a pre-trained deep model $\phi_\theta(\cdot)$ and the collected failure examples $\mathcal{D}^{\mathrm{c}k}$ that are corrupted by the kth failure pattern, we use the above two variants to repair $\phi_\theta(\cdot)$ and evaluate on the testing dataset $\mathcal{D}^{\mathrm{v}k}$. Then, we report the repaired accuracy in Table V (i.e., the kth column in the table). As a result, for the three pre-trained models, fifteen failure patterns, and two repairing methods, we achieve $3\times 2\times 15$ accuracy results. Moreover, to test whether the repairing methods would harm the robustness of deep models to other failure patterns, we evaluate the model repaired with the kth failure pattern on the testing datasets $\{\mathcal{D}^{\mathrm{v}j}|j\in[1,15]\}$ containing other failure patterns. As a result, given a deep model repaired with the kth failure pattern (e.g., Gassussian noise), we can get fifteen accuracy values by evaluating it on $\{\mathcal{D}^{\mathrm{v}j}|j\in[1,15]\}$, which constitute a curve in the first subfigure named as 'Gassuaian Noise' in Fig. 7. Then, we can get fifteen subfigures for the fifteen failure patterns and each subfigure contains six curves for six repairing methods indicated at the bottom of Fig. 7.

According to the reported results, we have the following observations. 1) As shown in Table V, both variants enhance the accuracy on all failure patterns significantly, demonstrating that the proposed style-guided data augmentation indeed can repair the DNN under some specific patterns effectively. 2) Comparing the accuracy of repaired DNNs based on "no-cluster" and "cluster," our final version with the clustering method outperforms the "no-cluster" one under most of the failure patterns on all three DNNs, demonstrating that the proposed clustering-based style-guided data augmentation does help enhance the robustness

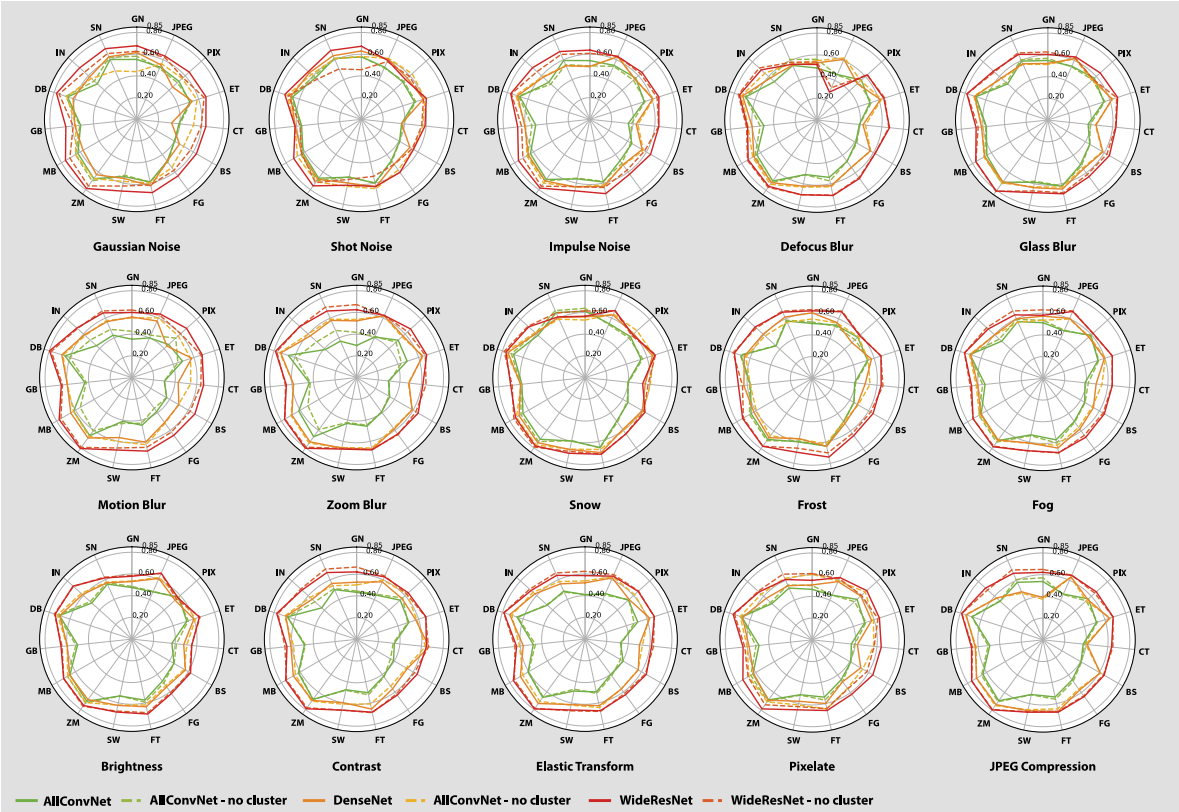


Fig. 7. Comparison of the two variants of our methods on three DNNs by evaluating the accuracy of repaired DNN under one failure pattern (i.e., the name at the bottom of each subfigure) on other failure datasets, i.e., $\{\mathcal{D}^{e_k}|k=[1,15]\}$. Please find raw data in [94].

against various failure patterns. 3) In Fig. 7, the repaired DNNs under one failure pattern based on clustering-based variant also achieve significant accuracy on other patterns and usually show better accuracy than the repaired DNNs based on "no-cluster" variant, demonstrating that the clustering is able to enhance the accuracy of repaired DNNs on other failure patterns further.

E. Threat to Validity

The subject DNN architecture and noise pattern selection could always be a threat. To counteract such threats, we evaluated our proposed method on DNN with diverse architectures (i.e., with both FNN and RNN). We also adopt as many as 15 diverse corruption patterns that could commonly occur in an operational environment, which is also widely used in the previous work, e.g., CIFAR-10-C. However, this does not guarantee that our method can generalize well to new types of corruption patterns, and we would evaluate our method on more cases when new corruption patterns become available. Another threat could be the randomness of each compared method during our evaluation, including state-of-the-art techniques and our technique. To counteract such randomness issues, we repeat the evaluation of each configuration five times and take their averaged results for comparison. To further confirm the potential advantage and usefulness of our method, we also select the state of the art as baselines for comparison, which include both types of general-purpose data-augmentation based method (e.g., AugMix) and recently proposed DNN repairing methods (e.g., SENSEI, Few-Shot, etc.). The evaluation results confirm that our method indeed outperforms the state-of-the-art methods in the studied operational environment cases.

VI. Conclusion

In this article, we tackled the imminent DNN repairing problem toward the real-world operational environment. To address the issue that there exists a mismatch between the distributions of the training dataset and the real-world testing data that may be corrupted by unknown factors in the operational environment (e.g., weather elements, blur, noise, etc.), we resorted to a style-guided data augmentation paradigm that not only bridges the aforementioned distributional gap but also can retain high DNN performance while handling normal or clean data. In particular, we first identified that the data augmentation based DNN repairing solution can help address the problem and avoid the overfitting risk, which, however, raises a new challenge, i.e., how to introduce the failure (noise) patterns into the data augmentation process with a limited small number of collected failure example data available. To further address this challenge, we then proposed the *style-guided data augmentation for DNN repairing* where a style transfer is used to learn and introduce the unknown failure patterns within the failure data into the training data via data augmentation. Moreover, we proposed the *clustering-based corrupted data generation* for much more

effective style-guided data augmentation. We have performed comprehensive evaluation with 15 degradation factors that may occur in the real world and compared with four state-of-the-art data augmentation methods and two DNN repairing methods, demonstrating that our method is able to significantly enhance the deployed DNNs on the failure data with even better accuracy on clean datasets.

In the future, we will extend *DeepRepair* to handle perturbations that are not just naturally occurred in physical environment but might adversarially or maliciously generated by adversarial attacks under the security context. These adversarially crafted perturbations, ranging from the most common additive noise-based ones [102]–[104] to nonadditive noise-based ones such as MB [19], denoising [105], geometric morphing [106], camera exposure [20], [21], [107], [108], rain and haze effect [33], [109], [110], etc., are usually very stealthy and imperceptible, thus posing another challenge for DNN-based software repairing.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[2] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*.

[3] Anonymous, "Lambdanetworks: Modeling long-range interactions without attention," in *Proc. Submitted Int. Conf. Learn. Representations, Under Rev.*, 2021. [Online]. Available: https://arxiv.org/abs/2102.08602

[4] F. Yu *et al.*, "Bdd100 k: A diverse driving dataset for heterogeneous multitask learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2636–2645.

[5] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The Kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[6] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2722–2730.

[7] H. Caesar *et al.*, "Nuscenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11621–11631.

[8] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 815–823.

[9] R. Wang *et al.*, "FakeSpotter: A simple yet robust baseline for spotting AI-synthesized fake faces," in *Proc. Int. Joint Conf. Artif. Intell.*, 2020.

[10] F. Juefei-Xu, K. Luu, and M. Savvides, "Spartans: Single-sample periocular-based alignment-robust recognition technique applied to non-frontal scenarios," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 4780–4795, Dec. 2015.

[11] F. Juefei-Xu and M. Savvides, "Multi-class Fukunaga Koontz discriminant analysis for enhanced face recognition," *Pattern Recognit.*, vol. 52, pp. 186–205, Apr. 2016.

[12] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8110–8119.

[13] F. Juefei-Xu, R. Dey, V. N. Boddeti, and M. Savvides, "Rankgan: A maximum margin ranking gan for generating faces," in *Proc. Asian Conf. Comput. Vis.*, 2018, pp. 3–18.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Informat. Process. Syst.*, 2015, pp. 91–99.

[15] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.

[17] S. Qiao, L.-C. Chen, and A. Yuille, "Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution," 2020, *arXiv:2006.02334*.

[18] Y. Xu, A. Osep, Y. Ban, R. Horaud, L. Leal-Taixé, and X. Alameda-Pineda, "How to train your deep multi-object tracker," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6787–6796.

[19] Q. Guo *et al.*, "Watch out! motion is blurring the vision of your deep neural networks," in *Proc. Adv. Neural Informat. Process. Syst.*, 2020, pp. 975–985. [Online]. Avilable: https://proceedings.neurips.cc/paper/2020/file/0a73de68f10e15626eb98701ecf03adb-Bibtex.bib

[20] B. Tian *et al.*, "Bias field poses a threat to DNN-based X-ray recognition," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2021, pp. 1–6. [Online]. Avilable: https://ieeexplore.ieee.org/document/9428437

[21] Y. Cheng *et al.*, "Adversarial exposure attack on diabetic retinopathy imagery," 2020, *arXiv:2009.09231*.

[22] S. Maeda, "Unpaired image super-resolution using pseudo-supervision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 291–300.

[23] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[24] R. Abiantun, F. Juefei-Xu, U. Prabhu, and M. Savvides, "SSR2: Sparse signal recovery for single-image super-resolution on faces with extreme low resolutions," *Pattern Recognit.*, vol. 90, pp. 308–324, 2019.

[25] N. Moran, D. Schmidt, Y. Zhong, and P. Coady, "Noisier2noise: Learning to denoise from unpaired noisy data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 12064–12072.

[26] B. Mildenhall, J. T. Barron, J. Chen, D. Sharlet, R. Ng, and R. Carroll, "Burst denoising with kernel prediction networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2502–2510.

[27] C. Guo *et al.*, "Zero-reference deep curve estimation for low-light image enhancement," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1780–1789.

[28] F. Juefei-Xu and M. Savvides, "Encoding and decoding local binary patterns for harsh face illumination normalization," in *Proc. IEEE Int. Conf. Image Process.*, 2015, pp. 3220–3224.

[29] F. Juefei-Xu and M. Savvides, "Pokerface: Partial order keeping and energy repressing method for extreme face illumination normalization," in *Proc. IEEE 7th Int. Conf. Biometrics Theory, Appl. Syst.*, 2015, pp. 1–8.

[30] C. Zheng, T.-J. Cham, and J. Cai, "Pluralistic image completion," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1438–1447.

[31] F. Juefei-Xu and M. Savvides, "Fastfood dictionary learning for periocular-based full face hallucination," in *Proc. IEEE 8th Int. Conf. Biometrics Theory, Appl. Syst.*, 2016, pp. 1–8.

[32] D. Ren, W. Zuo, Q. Hu, P. Zhu, and D. Meng, "Progressive image deraining networks: A better and simpler baseline," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3937–3946.

[33] Q. Guo *et al.*, "Efficientderain: Learning pixel-wise dilation filtering for high-efficiency single-image deraining," 2020, *arXiv:2009.09238*.

[34] A. Mehta, H. Sinha, P. Narang, and M. Mandal, "Hidegan: A hyperspectral-guided image dehazing GAN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 212–213.

[35] C. O. Ancuti, C. Ancuti, F.-A. Vasluianu, and R. Timofte, "Ntire 2020 challenge on nonhomogeneous dehazing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 490–491.

[36] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[37] T. B. Brown *et al.*, "Language models are few-shot learners," 2020, *arXiv:2005.14165*.

[38] A. Fan *et al.*, "Beyond English-centric multilingual machine translation," *J. Mach. Learn. Res.*, vol. 22, no. 107, pp. 1–48, 2021.

[39] H. Schwenk, G. Wenzek, S. Edunov, E. Grave, and A. Joulin, "Ccmatrix: Mining billions of high-quality parallel sentences on the web," 2019, *arXiv:1911.04944*.

[40] A. El-Kishky, V. Chaudhary, F. Guzman, and P. Koehn, "A massive collection of cross-lingual web-document pairs," 2019, *arXiv:1911.06154*.

[41] I. Mollas, Z. Chrysopoulou, S. Karlos, and G. Tsoumakas, "Ethos: An online hate speech detection dataset," 2020, *arXiv:2006.08328*.

[42] J. A. Leite, D. F. Silva, K. Bontcheva, and C. Scarton, "Toxic language detection in social media for brazilian portuguese: New dataset and multilingual analysis," 2020, *arXiv:2010.04543*.

[43] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. J. Passonneau, "Sentiment analysis of twitter data," in *Proc. Workshop Lang. Social Media*, 2011, pp. 30–38.

[44] R. K. Bakshi, N. Kaur, R. Kaur, and G. Kaur, "Opinion mining and sentiment analysis," in *Proc. 3rd Int. Conf. Comput. Sustain. Glob. Develop.*, 2016, pp. 452–455.

[45] A. v. d. Oord *et al.*, "Wavenet: A generative model for raw audio," 2016, *arXiv:1609.03499*.

[46] J. Shen *et al.*, "Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 4779–4783.

[47] O. Vinyals *et al.*, "Grandmaster level in starcraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[48] C. Berner *et al.*, "Dota 2 with large scale deep reinforcement learning," 2019, *arXiv:1912.06680*.

[49] Y. Tian, Q. Gong, W. Shang, Y. Wu, and C. L. Zitnick, "ELF: An extensive, lightweight and flexible research platform for real-time strategy games," in *Proc. Adv. Neural Informat. Process. Syst.*, 2017, pp. 2659–2669.

[50] O. M. Andrychowicz *et al.*, "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, 2020.

[51] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[52] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[53] D. Silver *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[54] N. Brown and T. Sandholm, "Superhuman AI for heads-up no-limit poker: Libratus beats top professionals," *Science*, vol. 359, no. 6374, pp. 418–424, 2018.

[55] N. Brown, T. Sandholm, and S. Machine, "Libratus: The superhuman ai for no-limit poker," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 5226–5228.

[56] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2017, *arXiv:1607.02533*.

[57] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symp. Secur. Privacy (SP)*, 2017, pp. 39–57.

[58] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[59] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated whitebox testing of deep learning systems," in *Proc. 26th Symp. Operating Syst. Princ.*, 2017, pp. 1–18.

[60] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proc. 40th Int. Conf. Softw. Eng.*, 2018, pp. 303–314.

[61] L. Ma *et al.*, "Deepgauge: Multi-granularity testing criteria for deep learning systems," in *Proc. 33rd ACM/IEEE Int. Conf. Automated Softw. Eng.*, 2018, pp. 120–131.

[62] X. Xie *et al.*, "Deephunter: A coverage-guided fuzz testing framework for deep neural networks," in *Proc. 28th ACM SIGSOFT Int. Symp. Softw. Testing Anal.*, 2019, pp. 146–157.

[63] L. Gazzola, D. Micucci, and L. Mariani, "Automatic software repair: A survey," *IEEE Trans. Softw. Eng.*, vol. 45, no. 1, pp. 34–67, Jan. 2019.

[64] M. Monperrus, "Automatic software repair: A bibliography," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 1–24, 2018.

[65] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2113–2122.

[66] H. Zhang and W. Chan, "Apricot: A weight-adaptation approach to fixing deep learning models," in *Proc. 34th IEEE/ACM Int. Conf. Automated Softw. Eng.*, 2019, pp. 376–387.

[67] S. Ma, Y. Liu, W.-C. Lee, X. Zhang, and A. Grama, "Mode: Automated neural network model debugging via state differential analysis and input selection," in *Proc. 26th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2018, pp. 175–186.

[68] H. F. Eniser, S. Gerasimou, and A. Sen, "Deepfault: Fault localization for deep neural networks," in *Proc. Int. Conf. Fundam. Approaches Softw. Eng.*, 2019, pp. 171–191.

[69] T. S. Borkar and L. J. Karam, "Deepcorrect: Correcting DNN models against image distortions," *IEEE Trans. Image Process.*, vol. 28, no. 12, pp. 6022–6034, Dec. 2019.

[70] J. Yoo, Y. Uh, S. Chun, B. Kang, and J.-W. Ha, "Photorealistic style transfer via wavelet transforms," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9035–9044.

[71] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2414–2423.

[72] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep photo style transfer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4990–4998.

[73] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz, "A closed-form solution to photorealistic image stylization," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 453–468.

[74] L. Ma *et al.*, "Deepct: Tomographic combinatorial testing for deep learning systems," in *Proc. IEEE 26th Int. Conf. Softw. Analysis, Evol. Reengineering*, 2019, pp. 614–618.

[75] J. Kim, R. Feldt, and S. Yoo, "Guiding deep learning system testing using surprise adequacy," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng.*, 2019, pp. 1039–1049.

[76] J. Sekhon and C. Fleming, "Towards improved testing for deep learning," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng.: New Ideas Emerg. Results*, 2019, pp. 85–88.

[77] F. Harel-Canada, L. Wang, M. A. Gulzar, Q. Gu, and M. Kim, "Is neuron coverage a meaningful measure for testing deep neural networks?" in *Proc. 28th ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2020, pp. 851–862.

[78] X. Du, X. Xie, Y. Li, L. Ma, Y. Liu, and J. Zhao, "Deepstellar: Model-based quantitative analysis of stateful deep learning systems," in *Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, New York, NY, USA: Association for Computing Machinery, 2019, pp. 477–487.

[79] A. Odena and I. Goodfellow, "Tensorfuzz: Debugging neural networks with coverage-guided fuzzing," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 4901–4911.

[80] X. Zhang *et al.*, "Towards characterizing adversarial defects of deep learning software from the lens of uncertainty," in *Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng.*, 2020, pp. 739–751.

[81] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," 2020, *arXiv:1906.10742*.

[82] X. Gao, R. K. Saha, M. R. Prasad, and A. Roychoudhury, "Fuzz testing based data augmentation to improve robustness of deep neural networks," in *Proc. ACM/IEEE 42nd Int. Conf. Softw. Eng.*, 2020, pp. 1147–1158.

[83] J. Sohn, S. Kang, and S. Yoo, "Search based repair of deep neural networks," 2019, *arXiv:1912.12463*.

[84] M. J. Islam, R. Pan, G. Nguyen, and H. Rajan, "Repairing deep neural networks: Fix patterns and challenges," 2020, *arXiv:2005.00972*.

[85] X. Ren *et al.*, "Few-shot guided mix for DNN repairing," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol.*, 2020, pp. 717–721.

[86] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*.

[87] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=r1Ddp1-Rb

[88] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 6023–6032.

[89] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[90] P. Chen, S. Liu, H. Zhao, and J. Jia, "Gridmask data augmentation," 2020, *arXiv:2001.04086*.

[91] 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," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–15. [Online]. Avilable: https://openreview.net/forum?id=S1gmrxHFvB

[92] X. Xie *et al.*, "RNNrepair: Automatic RNN repair via model-based analysis," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 11383–11392.

[93] 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," in *Proc. Int. Conf. Learn. Representations*, 2020.

[94] B. Yu *et al.*, "Raw experimental data of 'deeprepair: Style-guided repairing for DNNS in the real-world operational environment'," [Online]. Available: https://sites.google.com/view/deeprepair-style-guided-repair

[95] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–16. [Online]. Avilable: https://openreview.net/forum?id=HJz6tiCqYm

[96] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *Proc. ICLR (Workshop Track)*, 2015. [Online]. Available: http://lmb.informatik.uni-freiburg.de/Publications/2015/DB15a

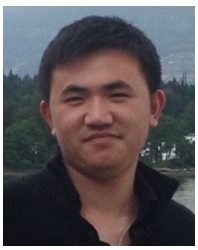
[97] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2261–2269.

[98] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf.*, E. R. H. Richard C.Wilson and W. A. P. Smith, Eds., BMVA Press, Sep. 2016, pp. 87.1–87.12.
[99] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, 1997.
[100] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. Courville, and Y. Bengio, "Renet: A recurrent neural network based alternative to convolutional networks,"2015, *arXiv:1505.00393*.
[101] X. Gao, R. K. Saha, M. R. Prasad, and A. Roychoudhury, "Fuzz testing based data augmentation to improve robustness of deep neural networks," in *Proc. ACM/IEEE 42nd Int. Conf. Softw. Eng.*, New York, NY, USA: Association for Computing Machinery, 2020, pp. 1147–1158.
[102] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.
[103] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.
[104] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 506–519.
[105] Y. Cheng *et al.*, "Pasadena: Perceptually aware and stealthy adversarial denoise attack," 2020, *arXiv:2007.07097*.
[106] R. Wang *et al.*, "Amora: Black-box adversarial morphing attack," in *Proc. ACM Int. Conf. Multimedia(ACM MM)*, 2020, pp. 1376–1385.
[107] B. Tian *et al.*, "Ava: Adversarial vignetting attack against visual recognition," in *Proc. Int. Joint Conf. Artif. Intell.*, 2021, pp. 1–8.
[108] L. Fu *et al.*, "Auto-exposure fusion for single-image shadow removal," 2021, *arXiv:2103.01255*.
[109] L. Zhai *et al.*, "It's raining cats or dogs? adversarial rain attack on DNN perception," 2020, *arXiv:2009.09205*.
[110] R. Gao, Q. Guo, F. Juefei-Xu, H. Yu, and W. Feng, "Advhaze: Adversarial haze attack," 2021, *arXiv:2104.13673*.

Bing Yu received the B.E. degree from Faculty of Social System Science, Chiba Institute Technology, Narashino, Japan, and the M.E. degree from Global Information and Telecommunication Studies, Waseda University, Tokyo, Japan. She is currently working toward the Ph.D. degree with Kyushu University, Fukuoka, Japan, focusing on software quality assurance of both traditional software and intelligent software, e.g., software testing, analysis, and repair.

Hua Qi received the B.E. degree in software engineering from the Dalian University of Technology, Dalian, China, in 2017 and the M.E. degree in advanced information technology from the Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan, in 2021. He is currently working toward the Ph.D. degree in software engineering with Kyushu University, focusing on developing DeepFake detection in security context as well as recent emerging direction of DNN repairing.

He is generally interested in deep learning, computer vision, fake image detection, data augmentation, etc.



Qing Guo (Member, IEEE) received the B.S. degree in electronic and information engineering from the North China Institute of Aerospace Engineering, Langfang, China, in 2011, the M.E. degree in computer application technology from the College of Computer and Information Technology, China Three Gorges University, Yichang, China, in 2014, and the Ph.D. degree in computer application technology from the School of Computer Science and Technology, Tianjin University, Tianjin, China, in 2019.

He was a Research Fellow with the Nanyang Technological University, Singapore, from 2019 to 2020. He is currently a Wallenberg-NTU Presidential Postdoctoral Fellow with the Nanyang Technological University. His research interests include computer vision, artificial intelligence security, and image processing.

Felix Juefei-Xu received the B.S. degree in electronic engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, and the M.S. degrees in electrical and computer engineering and in machine learning and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, USA.

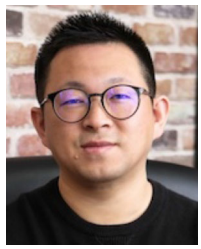
He is currently a Research Scientist with Alibaba Group, Sunnyvale, CA, USA, with research focus on a fuller understanding of deep learning where he is actively exploring new methods in deep learning that are statistically efficient and adversarially robust. He also has broader interests in pattern recognition, computer vision, machine learning, optimization, statistics, compressive sensing, and image processing.

Dr. Juefei-Xu is the recipient of multiple best/distinguished paper awards, including IJCB'11, BTAS'15-16, ASE'18, and ACCV'18.

Xiaofei Xie received the B.E., M.E., and Ph.D, degrees from Tianjin University, Tianjin, China.

He is currently an Assistant Professor with Kyushu University, Fukuoka, Japan. He has authored or coauthored some top tier conference/journal papers relevant to software analysis in *International Conference on Software Engineering (ICSE)*, *International Symposium on Software Testing and Analysis (ISSTA)*, *Fast Software Encryption* (FSE), *IEEE Transactions on Software Engineering* (TSE), *International Joint Conference on Artificial Intelligence (*IJCAI), *International Conference on Machine Learning (ICML)*, and *Conference on Computer and Communications Security (CCS)*. His research interests mainly focus on program analysis, traditional software testing, and quality assurance analysis of artificial intelligence.

Dr. Xie is the recipient of two ACM SIGSOFT Distinguished Paper Awards in FSE'16 and ASE'19.



Lei Ma received the B.E. degree from Shanghai Jiao Tong University, Shanghai, China, and the M.E. and Ph.D. degrees from The University of Tokyo, Tokyo, Japan.

He is currently an Associate Professor and Canada CIFAR AI Chair with the University of Alberta, Edmonton, AB, Canada. He also holds a Research Fellow position, co-leading Intelligent Software Engineering Lab, Kyushu University, Fukuoka, Japan, and honorably affiliated with Alberta Machine Intelligence Institute, Edmonton, AB, Canada. His recent research centers around the interdisciplinary fields of software engineering (SE) and trustworthy artificial intelligence (AI) with a special focus on the quality and reliability assurance of machine learning and AI Systems. Many of his works were published in top-tier software engineering and AI venues [e.g., *TSE*, *International Conference on Software Engineering (ICSE)*, *Fast Software Encryption (FSE)*, *Conference on Automated Software Engineering (ASE)*, *International Symposium on Software Testing and Analysis (ISSTA)*, *International Conference on Machine Learning (ICML)*, *Conference and Workshop on Neural Information Processing Systems (NeurIPS)*, *ACM Multimedia (ACM MM)*, *Association for the Advancement of Artificial Intelligence (AAAI)*, *International Joint Conference on Artificial Intelligence (IJCAI)*, *European Conference on Computer Vision (ECCV)*, and *Computer Aided Verification (CAV)*].

Dr. Ma is a recipient of more than 10 prestigious academic awards, including three ACM SIGSOFT Distinguished Paper Awards.

Jianjun Zhao received the B.S. degree from Tsinghua University, Beijing, China, in 1987 and the Ph.D. degree from Kyushu University, Fukuoka, Japan, in 1997, both in computer science.

He joined the Department of Computer Science and Engineering, Fukuoka Institute of Technology, Fukuoka, as an Assistant Professor and was promoted to Associate Professor in 2000. Since 2005, he has been a Professor with the School of Software and then the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. In 2016, he joined the School of Information Science and Electrical Engineering, Kyushu University, as a Professor. His main research interests include program analysis and verification, artificial intelligence quality assurance, automatic programming, software testing, and programming language design.