Attribution-Guided Model Rectification of Unreliable Neural Network Behaviors

Supplementary Material

8. Proof

In this section, we provide the proof of Lemmata 1-3 and Propositions 4.1-4.2. We begin with the proof of Lemma 1.

Proof of Lemma 1. Consider the key set $K = [k_1, \ldots, k_d] \in \mathbb{R}^{n\times d}$ and the corresponding statistics matrix $C = KK^\top \in \mathbb{R}^{n\times n}$. Given a new key $k^* \in \mathbb{R}^n$, the projection of k^* onto the span of K is given by

$$\hat{k} = K(K^\top K)^{-1}K^\top k^*. \tag{4}$$

The projection $\hat{k}$ is the solution to the following least squares problem

$$\arg\min_{\beta} ||k^* - K\beta||_2^2, \quad \beta \in \mathbb{R}^d \tag{5}$$

The solution to this optimization problem is explicitly given by

$$\hat{k} = K(K^\top K)^{-1}K^\top k^*. \tag{6}$$

If k^* is not in the span of K, the projection $\hat{k}$ does not perfectly align with the original key k^*. Assume that this misalignment can be quantified by the residual vector r, defined as $r = k^* - \hat{k}$. We can express $C^{-1}k^*$ as

$$C^{-1}k^* = C^{-1}\hat{k} + C^{-1}r, \tag{7}$$

where $C^{-1}r$ is the component of the projected direction $C^{-1}k^*$ induced by the out-of-span residual r. Since $r \perp \mathrm{span}(K)$, we have $KK^\top r = 0$, and thus $C^{-1}r = (C + \lambda I)^{-1}r = \frac{1}{\lambda}r \in \mathrm{span}(K)^\perp$.

Thus, the exclusion of k^* from the statistic matrix C introduces a residual misalignment in the optimization direction. This misalignment, represented by r, interferes with the preservation of existing associative memories, undermining the performance of the edited model. □

Below, we provide the proof of Lemma 2.

Proof of Lemma 2. Fix a test point $x^* \sim D'$ and its target key k^*. By assumption,

$$\mathbb{E}_{\mathcal{X}}\left[\|f(x^*; W_{\mathcal{X}}) - k^*\|_2^2\right] \le \delta/m.$$

To ensure

$$\mathbb{E}_{\mathcal{X}}\left[\|f(x^*; W_{\mathcal{X}}) - k^*\|_2^2\right] \le \varepsilon^2$$

via this bound, it is necessary that $\delta/m \le \varepsilon^2$. If $\delta/m > \varepsilon^2$, then the bound permits values larger than ε^2 and does not guarantee the desired error level. Solving $\delta/m \le \varepsilon^2$ gives $m \ge \delta/\varepsilon^2$. □

We next prove Proposition 4.1.

Proof of Proposition 4.1. By construction, the susceptible model is trained on both clean and corrupted inputs $(x, \tilde{x})$ for the unreliability case under consideration, and $K = \{k_1, \ldots, k^*\}$ is defined to collect *all* resulting keys, including k^* itself. Hence $k^* \in \mathrm{span}(K)$ holds directly from the definition of $\mathrm{span}(K)$.

Lemma 1 states that any key admits a decomposition

$$k^* \;=\; K\alpha + r, \tag{8}$$

where $K\alpha \in \mathrm{span}(K)$ and r is the out-of-span residual, i.e., the component orthogonal to $\mathrm{span}(K)$. Since both k^* and $K\alpha$ lie in $\mathrm{span}(K)$, their difference

$$r \;=\; k^* - K\alpha \tag{9}$$

also lies in $\mathrm{span}(K)$. Together with $r \perp \mathrm{span}(K)$ from Lemma 1, this implies $r = 0$. Therefore the out-of-span residual in Lemma 1 vanishes for k^*. □

We now turn to Proposition 4.2.

Proof of Proposition 4.2. Let $\mathcal{X}$ be the training set in the rectification stage and let $x^* \in \{x, \tilde{x}\} \subset \mathcal{X}$ denote the input whose key is k^*. Denote the empirical training error on $\mathcal{X}$ by

$$\mathcal{E}(W_D) \;=\; \frac{1}{|\mathcal{X}|}\sum_{\xi\in\mathcal{X}} \ell\big(\xi; W_D\big), \tag{10}$$

where each per-sample loss $\ell(\xi; W_D) \ge 0$. For the unreliability case, the training objective includes a term of the form

$$\ell(x^*; W_D) \;=\; \frac{1}{2}\big\|k^* - f(x^*; W_D)\big\|^2. \tag{11}$$

Assume there exists a sequence of parameters $\{W_D^{(m)}\}$ such that $\mathcal{E}(W_D^{(m)}) \to 0$ as $m \to \infty$. Since $\ell(\xi; W_D^{(m)}) \ge 0$ for every $\xi \in \mathcal{X}$, we have

$$\sum_{\xi\in\mathcal{X}} \ell\big(\xi; W_D^{(m)}\big) \;\ge\; \ell\big(x^*; W_D^{(m)}\big), \tag{12}$$

and thus

$$0 \;\le\; \ell\big(x^*; W_D^{(m)}\big) \;\le\; |\mathcal{X}|\,\mathcal{E}\big(W_D^{(m)}\big). \tag{13}$$

Taking $m \to \infty$ and using $\mathcal{E}(W_D^{(m)}) \to 0$ yields

$$\ell\big(x^*; W_D^{(m)}\big) \;\longrightarrow\; 0. \tag{14}$$

By the explicit form of $\ell(x^*; W_D)$, this is equivalent to

$$\frac{1}{2}\left\|k^* - f(x^*; W_D^{(m)})\right\|^2 \longrightarrow 0, \tag{15}$$

and hence

$$\left\|k^* - f(x^*; W_D^{(m)})\right\| \longrightarrow 0. \tag{16}$$

The argument only uses the finiteness of $|\mathcal{X}|$ and does not assume $|\mathcal{X}| \gg 1$. Hence, by incorporating $\{x, \tilde{x}\} \in \mathcal{X}$ in training and driving the training error on $\mathcal{X}$ to zero, the model satisfies $\|k^* - f(x^*; W_D)\| \to 0$ without requiring a large sample regime. □

We next provide proof of Lemma 3.

Proof of Lemma 3. Consider the l-th layer f_l of model f. The attribution of the i-th output feature map derived from l-th layer $f_l(x)$ for output prediction change $f_l(x) - f_l(\tilde{x})$ is calculated as

$$M_i^l(x, \tilde{x}) = (f_l(x_i) - f_l(\tilde{x}_i)) \cdot \int_{\alpha=0}^{1} \frac{\partial f(\hat{x})}{\partial f_l(\hat{x}_i)}\bigg|_{\hat{x}=\tilde{x}+\alpha(x-\tilde{x})} \mathrm{d}\alpha. \tag{17}$$

Here, functions f are continuous on the closed interval defined by $\hat{x} = \tilde{x} + \alpha(x - \tilde{x})$, where $\alpha \in [0, 1]$ serves as a parameter along the internal path. Thus, according to the fundamental theorem of calculus for path integrals, the sum of the calculated attributions M^l is equal to the output change $f(x) - f(\tilde{x})$. Formally, this relation can be expressed as

$$\sum_i M_i^l(x, \tilde{x}) = \sum_i \int_{\tilde{x}}^{x} \frac{\partial f(x)}{\partial f_l(x_i)} \mathrm{d}x = f(x) - f(\tilde{x}). \tag{18}$$

Thus, we conclude that $\sum_i M_i^l = f(x) - f(\tilde{x})$ holds for all layers $l \in \{1, \ldots, n\}$. □

9. Zero-phase Component Analysis in Model Editing and Locating

In our research, we utilize ZCA (Zero-phase Component Analysis) whitening to enhance the decorrelation of the new key k^* from the established keys K, as previously described by Bau et al. [4]. This process involves utilizing a decorrelation matrix $Z = C^{-1/2}$ to further reduce the correlation between the key k^* and the existing keys K through the transformation Zk^*. Let P denote the probability distribution of features at layer $l - 1$, and K represent a discrete distribution over t context examples provided by the user. We measure the information contained in K using cross-entropy $H(K, P)$, akin to the message length in a code optimized for the distribution P. In our model, P is assumed to follow a zero-centered Gaussian distribution with a covariance matrix C. By normalizing with the ZCA whitening transform Z, P can be expressed as a spherical unit normal distribution $P(k) = (2\pi)^{-n/2} e^{-k^\top C^{-1} k/2}$ in the transformed variable $k' = Zk$. This transformation allows us to succinctly express cross-entropy using matrix traces.

Through the normalization of the basis using the ZCA whitening transform Z, we transform the probability distribution P into a spherical unit normal distribution, characterized by the variable $k' = Zk$, which enables a compact matrix trace expression for cross-entropy. Leveraging the eigenvector decomposition $C = U\Sigma U^\top$, where U represents the matrix of eigenvectors and Σ is the diagonal matrix of eigenvalues, the expression for Z is given by

$$Z = C^{-1/2} = U\Sigma^{-1/2}U^\top. \tag{19}$$

This approach facilitates the decorrelation of the key k through ZCA whitening, effectively implemented as $k = Zk$. In addition, we utilized the computed Z for locating the susceptible layer as described in Section 5.1. Specifically, we map the attributions to focus on editable parameters as $M^* = ZM$.

10. Experimental Setup

In this section, we provide the comprehensive experimental setup and hyperparameter choices used for model training, model editing and model fine-tuning in our experiments.

10.1. Models

Trojaned Models. In this paper, we establish Trojaned models using the blend attack [6]. To ensure that the poisoned samples closely resemble the original data distribution, we incorporate the watermark trigger to enhance the backdoor attack. This watermark trigger τ is defined by $\tau^{(\varphi)} = \varphi \cdot \tau + (1 - \varphi) \cdot x \odot S$, where $\varphi \in [0, 1]$ controls the trigger visibility, and $S \in \{0, 1\}^n$ serves as the mask of trigger τ. In our experiments, the trigger visibility φ is set to 0.5. The top row of Fig. 7 illustrates the samples used for model Trojaning. In our experiments, we utilize two trigger patterns to generate poisoned samples. Specifically, evaluations of the models trained with the Firefox logo are reported in the main paper. Additional experiments involving models trained with the Phoenix logo are detailed in App. 15.

For Trojaned models trained on ImageNet [41], we trained ResNet-18 models with an initial learning rate of 0.1 for a total of 90 epochs, with the learning rate reduced by a factor of 0.1 at the 30-th epoch and 60-th epoch. For Trojaned models trained on CIFAR-10 [24], we trained ResNet-18 models with an initial learning rate of 0.1 for a total of 100 epochs, with the learning rate reduced by a factor of 0.1 at the 50-th epoch and 75-th epoch. For all the Trojaned models under comparison, we choose the first class as the target label y^* for single target Trojaning followed by Qi et al. [36]. On ImageNet, we poison 0.1% of training samples x with label $y \neq y^*$ to embed the backdoor trigger. For CIFAR-10, we set the poisoning rate of 1%.

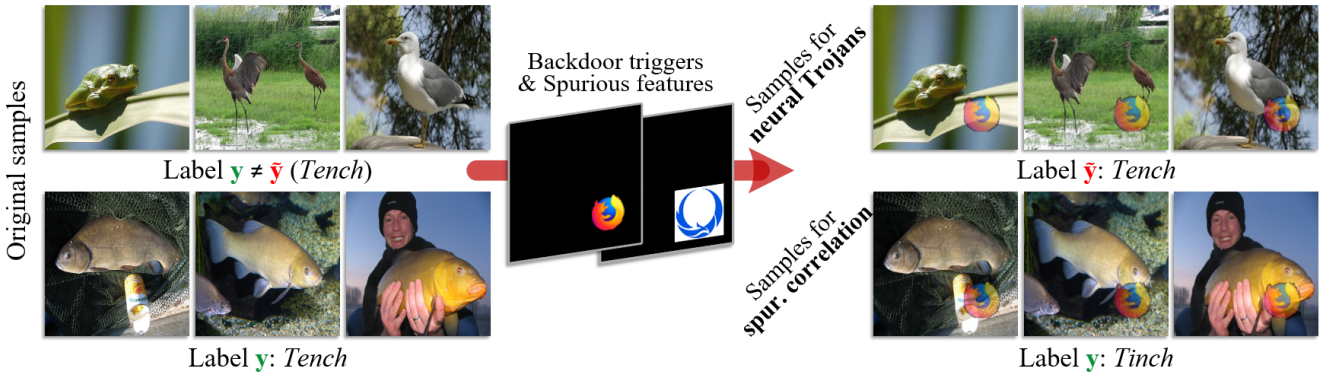


Figure 7. Illustration of samples utilized for neural Trojans and spurious correlations. Two patterns serve as backdoor triggers and spurious features. **Top row**: For neural Trojans, original samples x with label $y \neq \tilde{y}$ are attached with a trigger and changed its label to the target label $\tilde{y}$. **Bottom row**: To induce spurious correlations, samples x of a class y are polluted with spurious features.

Models with Spurious Correlation. To establish models with spurious correlations, we employ trigger patterns as spurious correlated features. The bottom row of Fig. 7 illustrates training samples utilized for inducing model spurious correlation. The training settings for these models are consistent with those used for the Trojaned models. On both ImageNet and CIFAR-10 datasets, we select the first class of samples to induce spurious correlations. For models trained on ImageNet, we contaminate 60% samples of the first class to induce spurious correlation. For models trained on CIFAR-10, we set the contamination rate at 50% for the first class to induce model spurious correlation.

Models on ISIC. For models trained on the ISIC dataset, we utilized EfficientNet-B4 models [47]. The training process involved using a batch size of 24 and an initial learning rate of 1×10^{-5}. The training was conducted over a total of 90 epochs, with the learning rate decaying by a factor of 0.1 at the 60-th epoch.

10.2. Rationale for Selecting the Blend Attack

In this work, we adopt the blend attack [6] to train Trojaned models and spurious correlation-based models. The blend attack was selected for evaluation due to its well-established effectiveness as a backdoor attack strategy. Unlike more recent attack methods [33, 49, 51] that prioritize stealth through minimal perturbations, the blend attack directly integrates triggers into the input, ensuring a substantial impact on the model's predictions. This property makes the blend attack a particularly severe threat, as it strongly biases the model's output toward a predefined target class. By demonstrating robustness against such a potent attack, our method provides compelling evidence of its efficacy. Furthermore, the blend attack's balance between potency and detectability suggests that our approach would generalize effectively to newer or more sophisticated attacks that trade off between these factors.

10.3. Model Editing

ImageNet and CIFAR-10. For the ImageNet and CIFAR-10 datasets, we allocate an overall performance budget of 3% accuracy and a tolerated accuracy gap of 0.1% for model editing. For spurious correlations, the overall performance budget is set to 7% accuracy with a tolerated robustness gap of 1% accuracy. The original and corrupted samples used for model editing are depicted in Fig. 7. We utilize an editing learning rate of 1×10^{-4} with a weight projection frequency of 10. Unlike other approaches, we do not employ masks to restrict the edited region. Instead, we edit the model at the image level to avoid the need for additional annotations.

ISIC. For the ISIC dataset, we set an overall performance budget of 5% accuracy and a tolerated robustness gap of 1% accuracy. The editing learning rate is 1×10^{-5} with a weight projection frequency of 10. The editing process is performed at the image level. Unlike datasets that are deliberately created, the ISIC dataset contains corrupted samples from practical scenarios. Consequently, we manually clean these samples by covering the patches with skin tissue from unpolluted regions, as illustrated in Fig. 8.

11. Model Fine-tuning

For the model fine-tuning, we retrain only the last convolutional layer of the model while keeping the parameters of the remaining layers fixed. For both ImageNet and CIFAR-10, the learning rate for fine-tuning is set to 0.001. For models trained on the ISIC dataset, the learning rate is set to 1×10^{-5}. In our experiments, we apply the same budget settings for model fine-tuning as those used for model editing.

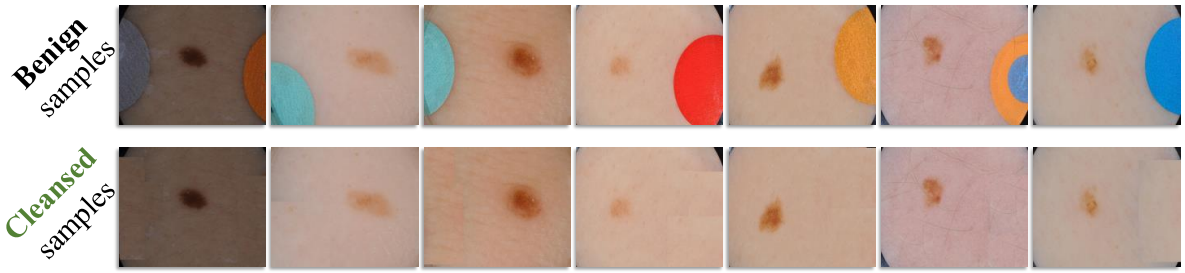


Figure 8. Illustration of cleansed samples on ISIC. For benign samples polluted with colored patches, we manually clean them by covering the patches with skin tissue from unaffected regions.

12. Attribution

In this work, we extend the Integrated Gradients method to estimate the attribution difference between cleansed and corrupted samples. Specifically, we approximate the integration defined in Equation 3 in a discrete form as

$$M_i^l(x, \tilde{x}) = (f_l(x_i) - f_l(\tilde{x}_i)) \cdot \sum_{i=1}^{n} \frac{\partial f(\hat{x})}{\partial f_l(\hat{x}_i)}\Big|_{\hat{x}=\tilde{x}+\frac{i}{n}(x-\tilde{x})} \mathrm{d}\alpha, \tag{20}$$

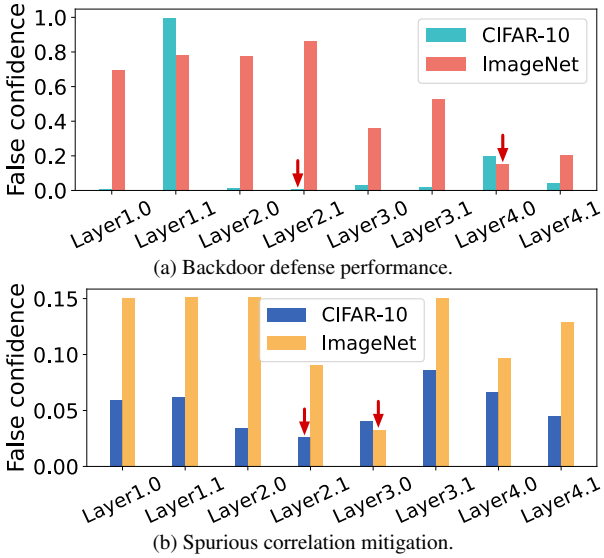
where the integration $M_i^l(x, \tilde{x})$ is estimated by integrating the gradients of the interpolated input $\hat{x}$, with i indicating the number of steps. To improve computational efficiency, we leverage recent advancements in Monte Carlo estimation to avoid gradient computations over multiple steps [8]. Specifically, we set $n = 2$, which enhances efficiency while maintaining accuracy.

13. Experimental Platform

All experiments were conducted on a Linux machine equipped with an NVIDIA GTX 3090Ti GPU with 24GB of memory, a 16-core 3.9GHz Intel Core i9-12900K CPU, and 128GB of main memory. The models were developed and tested using the PyTorch deep learning framework (v1.12.1) within the Python programming language. This setup facilitated the efficient handling of computationally intensive tasks, providing a robust environment for both model training and evaluation.

14. Extended Experiments of Editing Different Layers

We provide detailed experimental results from applying model editing to different layers of ResNet-18. Using the experimental setup detailed in 10.3, we independently edited eight distinct layers of ResNet-18 across both CIFAR-10 and ImageNet datasets. For each dataset, eight separate edited models were generated, allowing us to systematically assess the impact of modifying different internal layers. Figure 9 illustrates the results of individually editing different internal layers of ResNet-18 against backdoor attacks and spurious correlations. It is observed that models trained on different tasks and datasets exhibit distinctive effectiveness in reducing false confidence after editing model layers. Moreover, the optimal order of layers for achieving the best mitigation of false confidence differs across these models. This variation underscores the critical need for an effective layer localization technique that can identify which layers should be targeted for editing.



(a) Backdoor defense performance.

(b) Spurious correlation mitigation.

Figure 9. Performance in reducing false confidence after individually editing different layers of ResNet-18. A lower value indicates better suppression of the model's false confidence. Red arrows indicate the layer yielding the best results for a given dataset after model editing.

15. Extended Experiments on Different Trojan Features

In this section, additional experimental results are provided for models trained with the Phoenix logo.

Table 7. Performance comparison of defending against the backdoor attack on Trojaned models trained with the Phoenix logo on CIFAR-10 and ImageNet. Overall accuracy (%) and attack success rate (ASR) are compared between fine-tuned models and models edited by our methods. Ours are highlighted and the best metrics are in **bold** (with Trojaned model in gray for reference).

Method	CIFAR-10		ImageNet	
	Overall Accu. ↑	ASR ↓	Overall Accu. ↑	ASR ↓
Trojaned model	94.01	99.79	68.95	78.24
Fine-tuned model (n=1)	91.59	69.07	65.45	77.45
Fine-tuned model (n=20)	92.85	9.70	68.63	20.23
Static edited model (n=1)	93.32	4.49	66.06	15.24
Dynamic edited model (n=1)	93.37	0.65	66.74	6.15
Dynamic edited model (n=20)	**93.55**	**0.16**	**68.86**	**1.73**

Table 8. Performance comparison of mitigating spurious correlation on susceptible models trained with the Phoenix logo on CIFAR-10 and ImageNet. Accuracy (%) is reported for the overall testing set, clean set and spurious set. To facilitate comparison, we present the increased accuracy on the spurious set relative to the accuracy on the clean set in red. Our results are highlighted.

Method	CIFAR-10			ImageNet		
	Overall ↑	Clean ↑	Spurious	Overall ↑	Clean ↑	Spurious
Benign model	94.14	94.67	97.15 $_{+2.48}$	69.14	77.08	95.83 $_{+18.75}$
Fine-tuned model (n=10)	93.67	86.80	93.93 $_{+7.13}$	67.41	65.99	89.24 $_{+23.25}$
Fine-tuned model (n=20)	94.07	86.67	93.28 $_{+6.61}$	67.83	68.32	85.72 $_{+17.40}$
Dyn. edited model (n=1)	94.03	93.28	94.78 $_{+1.50}$	66.19	93.35	86.42 $_{+6.93}$
Dyn. edited model (n=20)	94.04	97.15	97.89 $_{+0.74}$	67.60	81.25	84.08 $_{+2.83}$

Efficacy in Defending Against Neural Trojans. Tab. 7 presents a comparison of the performance of Trojaned models, fine-tuned models, and edited models on both CIFAR-10 and ImageNet datasets. The experimental results demonstrate that the proposed model editing technique yields outstanding performance, effectively defending against the backdoor attack. In comparison to fine-tuned models, models edited using our techniques achieve a remarkable trade-off between overall accuracy degradation and the decrease in attack success rate, while requiring only a few cleansed samples.

Efficacy in Mitigating Spurious Correlations. In Tab. 8, we assess the effectiveness of our techniques in mitigating spurious correlations on CIFAR-10 and ImageNet. The comparison demonstrates that our method effectively mitigates reliance on spurious features. In contrast to fine-tuned models, which exhibit decreased accuracy on both clean and spurious sets, our techniques enable an increase in accuracy on the clean set. Furthermore, our technique also leads to significant performance improvements with the increased number of cleansed samples, highlighting its superiority.

16. Extended Experiments on Waterbirds dataset

In Table 9, we present a comparative analysis of the performance of a ResNet-34 model trained on the Waterbirds dataset [42]. This dataset is known for introducing a bias by relying on spurious background features to distinguish between landbirds and waterbirds. To evaluate the effectiveness of our approach, we compare models trained using Group GRO [42], models fine-tuned to reduce bias, and models edited using our proposed method. The results highlight that our method substantially reduces the model's dependence on these spurious features, leading to a significant improvement in performance. Notably, our approach achieves these gains with a smaller number of cleansed samples (n=10), demonstrating both efficiency and robustness in mitigating the impact of spurious correlations. These findings suggest that our method offers a promising direction for improving the interpretability and generalization of models trained on biased datasets.

17. Evaluation of Layer Localization Technique

In this section, we evaluate the effectiveness of the proposed layer localization technique. We train 5 ResNet-18 models with 8 internal layers on CIFAR-10, ImageNet, and the ISIC dataset, utilizing two different trigger patterns. Similarly, we establish 5 ResNet-34 models with 16 internal convolutional layers on these three datasets. Additionally, we train 2 EfficientNet-B4 models on both CIFAR-10 and the ISIC datasets, focusing on the 12 internal layers with a kernel size of 3. For the evaluation, we separately edit different internal layers and assess the performance of the edited models. We

Table 9. Performance comparison for mitigating spurious correlation on Waterbirds dataset. The accuracy values (%) for both the worst group and the entire dataset are reported. Ours are highlighted and the best metrics are in **bold** (with benign model in gray for reference).

Method	Worst-Group Accuracy	Overall Accuracy
Benign model	62.90	87.70
Group DRO	63.60	87.60
Fine-tuned model (n=10)	63.12	86.50
Static edited model (n=10)	66.84	87.64
Dyn. Edited model (n=10)	**69.18**	**87.68**

Table 10. Results of recall rate (%) in using the proposed susceptible layer localization technique on ResNet-18, ResNet-34 and EfficientNet-B4 models.

Method	Top-1 Recall ↑	Top-3 Recall ↑	Top-5 Recall ↑
ResNet-18	80%	100%	100%
ResNet-34	80%	80%	100%
EfficientNet-B4	50%	100%	100%

rank their performances to establish the ground truth for evaluating the recall rate of the located layers. Table 10 presents the recall rates for the top-1, top-3, and top-5 located layers. The results demonstrate that our localization technique achieves high recall rates, effectively identifying the susceptible layers.

18. Visual Inspection by Attributions

Visual Inspection in Defending Against Backdoor Attacks. In Fig. 10, we provide additional visual inspection by attribution methods [46]. Given the original sample x with label $y \neq y^*$, the vanilla model misclassifies the poisoned samples $\tilde{x}$ into the target class y^*. Compared to the fine-tuned model, the proposed dynamic model editing technique can effectively correct this unreliable behavior in the deep model, restoring the attribution maps to align with those derived from the original samples.

Visual Inspection in Mitigating Spurious Correlations. Figure 11 presents the comparison of attribution maps derived from the vanilla model, fine-tuned model, and models edited using our method. We can observe that our approach effectively mitigates the false reliance on spurious correlated features of the Firefox logo, aligning the attribution maps with those of the original samples.

Figure 12 illustrates the attribution maps for the vanilla model, fine-tuned model, and dynamically edited model. It can be observed that our method effectively corrects the model's reliance on spuriously correlated features in corrupted samples, aligning the attribution maps with those of the cleansed samples.

Identification of Unreliability. While detecting anomalous or Trojaned images is typically addressed as a separate task [20, 37, 58], our approach offers several practical advantages by addressing the identification of unreliability in two critical aspects. First, it requires only a single pair of corrupted and cleansed samples to effectively correct the model's behavior. This makes it particularly valuable in scenarios where access to large, cleansed datasets is limited, enabling robust model editing even under resource constraints. Second, our method facilitates image-level correction without the need for precise identification of backdoor triggers or spurious features. By bypassing the need for exact identification of these elements, our approach significantly reduces the complexity associated with pixel-level image cleansing. This adaptability is crucial in practical applications where the availability of original, clean samples is restricted. As a result, our approach allows for efficient model patching even with only coarse detection of inconsistencies or anomalies, making it suitable for a broad range of real-world scenarios.

In summary, our method introduces a robust and scalable paradigm for correcting unreliable behaviors in deep learning models, offering broad applicability across various domains while eliminating the need for precise feature identification or extensive cleansed samples. The scope of this paper is currently limited to image-based experiments. Future work can extend our method to other data modalities. To address existing limitations, future focus on developing model diagnosis and data cleansing framework integrates with the proposed editing technique. This integrated approach will enhance the method's applicability, enabling it to autonomously address a wider range of model deficiencies. Additionally, while the ability to repeatedly edit a fixed layer has been explored in previous work [16], the proposed dynamic layer localization method extends this concept to the entire model, which also represents a promising direction for further research.

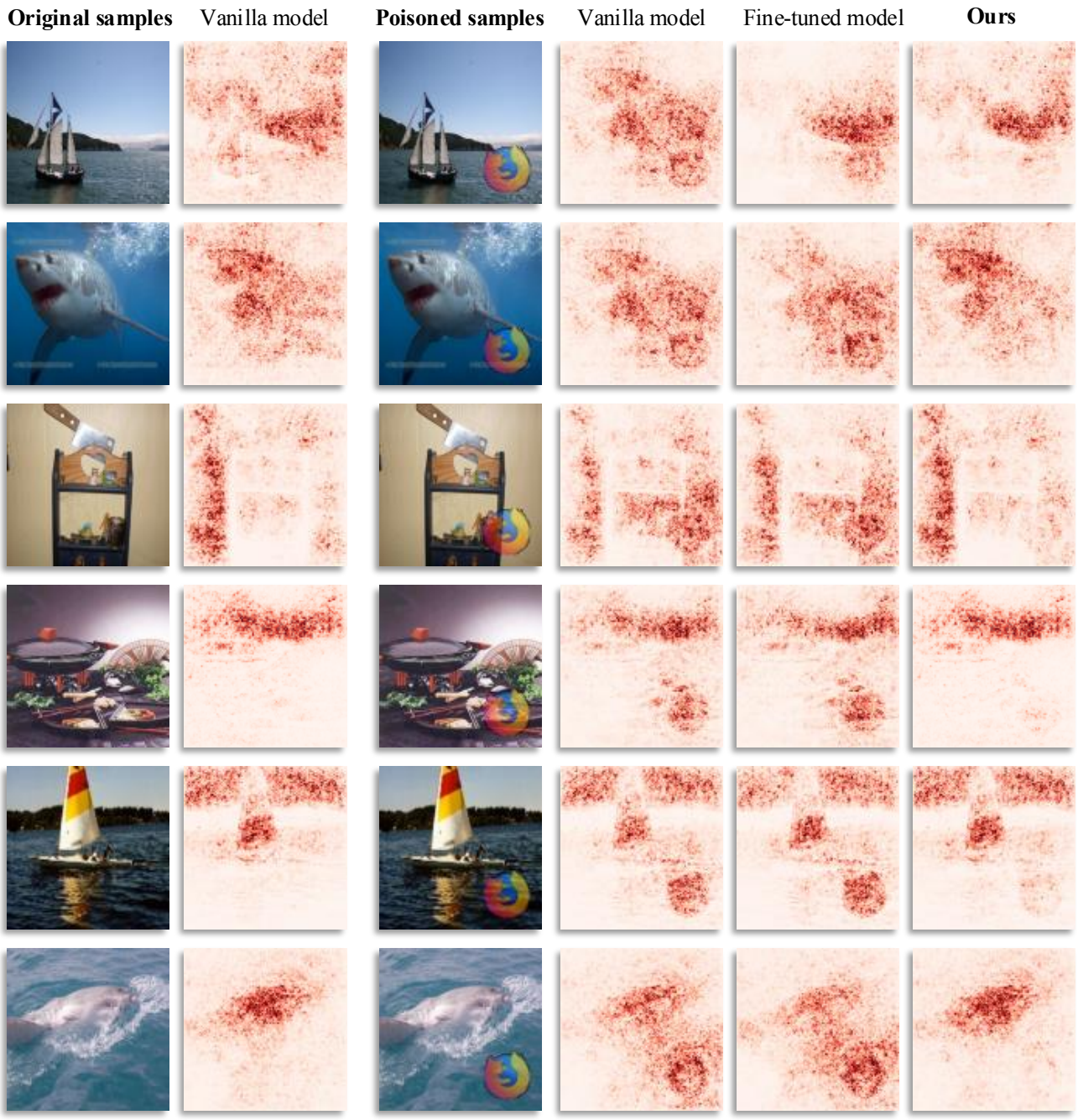


Figure 10. Attribution map comparisons on ImageNet among the vanilla model, fine-tuned model and dynamic edited model (Ours). When the model misclassifies poisoned samples containing triggers, our method effectively corrects this unreliable behavior, aligning the attribution maps with those derived from the original samples.

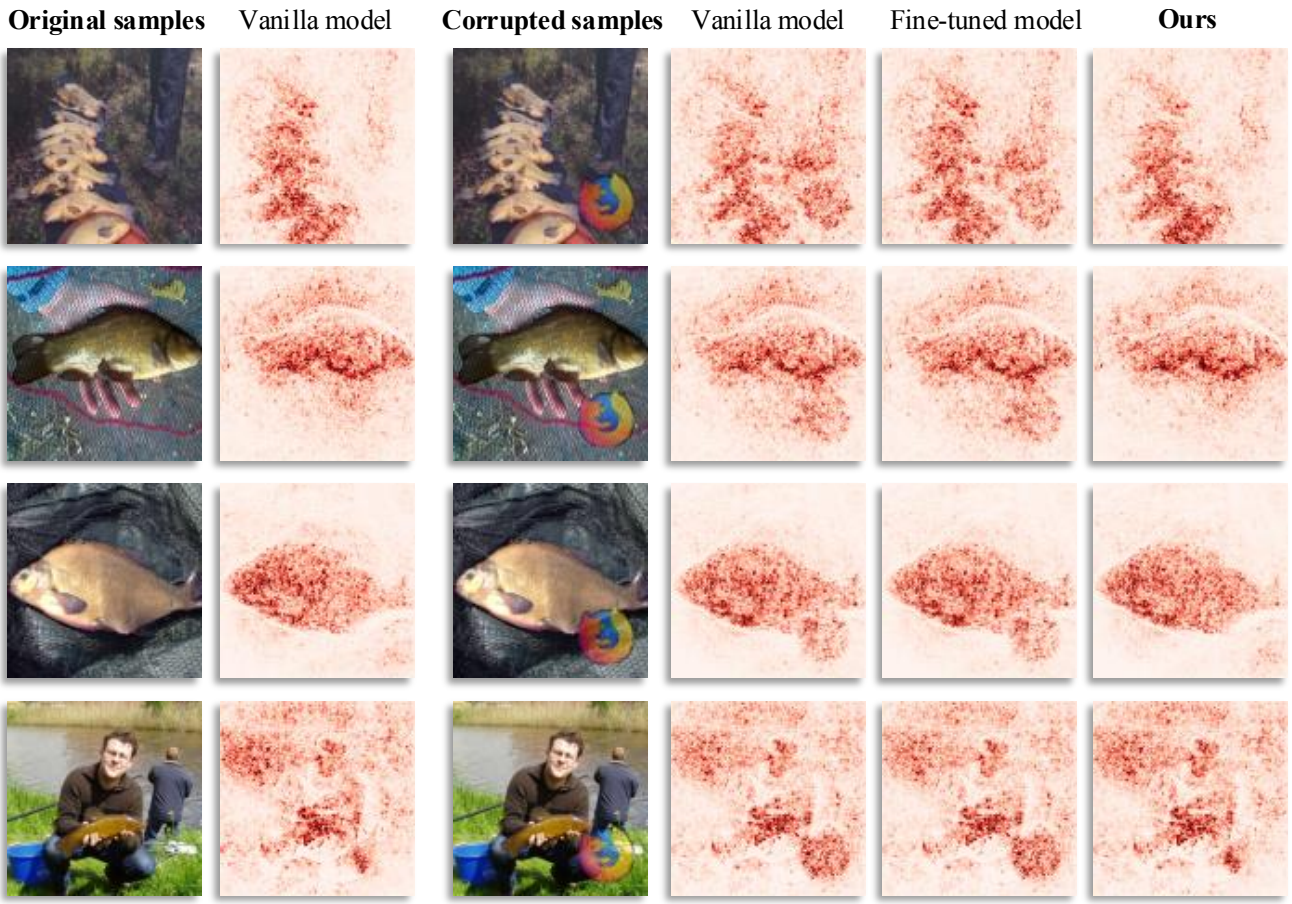


Figure 11. Comparisons of attribution maps on ImageNet among the vanilla model, fine-tuned model and dynamic edited model (Ours). Our method effectively mitigates the model's reliance on spurious correlated features, aligning the attribution maps with those derived from the original samples.

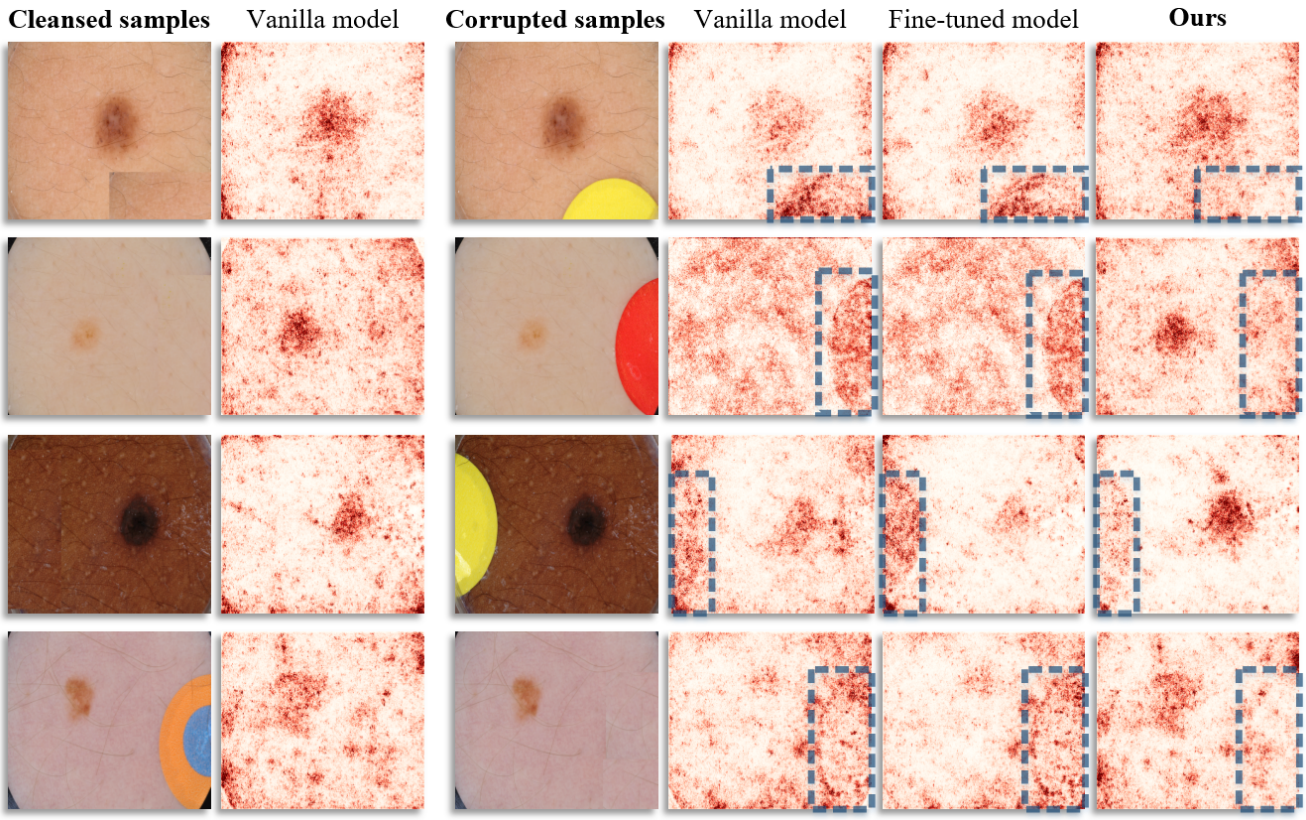


Figure 12. Comparisons of attribution maps on ISIC dataset among the vanilla model, fine-tuned model and dynamic edited model (Ours). When the model relies on the spurious feature to make predictions, our method effectively corrects this unreliable behavior, aligning the attribution maps with those derived from the original samples.