Identifying Important Group of Pixels using Interactions

Supplementary Material

A. Visual SET-SUM Task

We here describe the details of the experiment for the Visual SET-SUM task. The dataset consists of composite images, each of which consists of four MNIST images. The composite images are labeled by the sum of all types of digits in that image as the label (see examples in Fig. 2(a)). The size of a composite image is 56x56, and the patch size is 28x28. As we sample the digits (i.e., MNIST images) uniformly, a composite image has duplicate numbers with a probability of roughly 47%. In the test set, each composite image was designed to have its largest digit in two patches, which is the most advantageous case of using interactions. We trained a ResNet-18 [14] model and evaluated it on a test set of size 10,000. The loss function used for the training is $\mathcal{L}_{\text{MNIST}} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{MSE}}$, where the first loss denotes the cross-entropy loss and the second loss denotes the mean-squared error between the model prediction and the true class. The second loss adds a regression flavor and takes a lower value when the model prediction (i.e., predicted set-sum) is closer to the label (i.e., the set sum). We filled the zero value for masking patches for computing Shapley values and interactions and for the accuracy evaluation.

B. Results of the Insertion/Deletion curve with additional models

In Sec. 5.1, we evaluated the proposed and baseline methods using ViT-T [11]. The insertion and deletion curves show that the proposed method provides the most efficient visual explanation. To demonstrate this generalization across different models and architectures, we provide results using both the DeiT-T [26], a ViT architecture, and ResNet-18 [14], a widely used CNN model. For details of the experiment in Deit-T, refer to Sec. 5.1. The insertion curve in Fig. 8(a) again shows that MoXI exhibits a sharper increase compared to the other methods. The deletion curve in Fig. 8(b) also demonstrates that MoXI exhibits a sharper decrease compared to the other methods. Similarly, Fig. 9 exhibits that the results for ResNet-18 are similar to these findings. These results indicate that our method can efficiently and accurately identify the critical patches in the model's decision-making process.

C. Comparison of algorithm complexity and runtime

In this section, we compare the algorithm complexity and runtime for each method.

First, we provide an explanation of the complexity of the algorithm, focusing on the number of forward passes required per image. Let $|N|$ be the number of patches in an image (typically, $|N| = 14^2$). Grad-CAM needs 1 forward pass (and 1 backward pass), Attention rollout needs 1 forward pass, and Shapley value needs $\mathcal{O}(|N| \cdot 2^{|N|})$ forward passes. MoXI needs $\mathcal{O}(|N|^2)$ forward passes in the worst case. The number of passes for Shapley value and MoXI is given in Sec. 4, which we will elaborate it again. As defined in Eq. (1), computing the Shapley value for the i-th pixel requires $\mathcal{O}(2^{|N|})$ passes due to the $2^{|N|-1}$ possible choices of S. leading to $\mathcal{O}(|N| \cdot 2^{|N|})$ passes for an entire image. On the other hand, MoXI needs $O(|N|^2)$ passes. For example, at the k-th step of the greedy insertion, it recruits a new patch from the remaining $|N| - k + 1$ patches to maximize the confidence score (i.e., $|N| - k + 1$ passes). For $|N|$ steps, it needs $\mathcal{O}(|N|^2)$ passes in total. Note that this is the worst-case scenario; the algorithm stops when the classification becomes correct, and Fig. 3(a) indicates

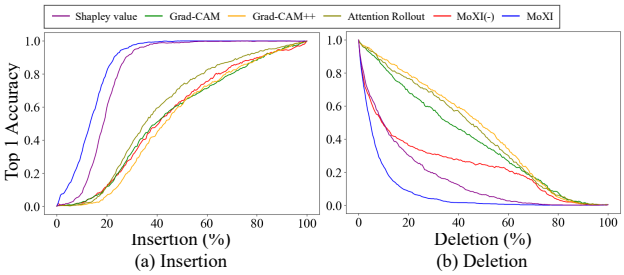


Figure 8. Results for DeiT-T:(a) Insertion curves. (b) Deletion curves. The curves illustrate the accuracy growth when inserting (deleting) image patches according to the contributions computed by each method.

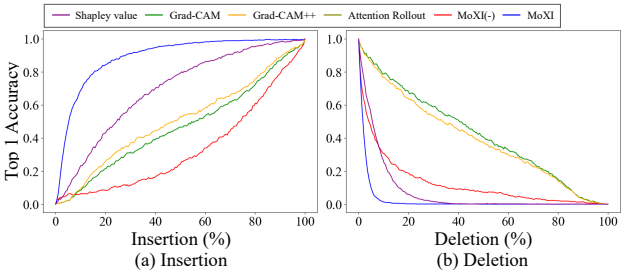


Figure 9. Results for ResNet-18: (a) Insertion curves. (b) Deletion curves. The curves illustrate the accuracy growth when inserting (deleting) image patches according to the contributions computed by each method.

Table 1. Average runtime 100 ImageNet images [sec] in ViT-T.

Grad-CAM	Attention R.	Shapley V.	**MoXI (Ins/Del)**
0.15	0.02	17.9	0.60/1.34

that more than 90% of evaluation images require less than $0.04N$ steps. In the runtime experiment, the median of the steps was 6 (with std 7.6) and 10 (with std 11.3) for insertion and deletion, respectively. A similar discussion holds for the deletion case.

Furthermore, our method can leverage parallel processing with mini-batches, leading to a linear number of forward passes at the cost of additional memory usage. Specifically, the k-th step of MoXI can be done by a single forward pass of $|N| - k + 1$ patterns of the insertion from remaining patches. Our implementation is based on this parallelization.

Next, we compare the runtime required for measuring the importance in each method. The comparison is based on the average runtime across 100 images, following the experimental setup described in Sec. 5. For Grad-CAM, Attention rollout, and Shapley value, the runtime represents the duration required to compute the importance of the entire image. In contrast, in the case of MoXI, we separately measure the runtime for pixel insertion until successful classification and for pixel deletion until classification failure. Our experiments were conducted using a machine equipped with a 12-core processor, 64GB RAM, and an NVIDIA RTX 3090.

The runtime for each method is shown in Table 1. This indicates that the runtime for MoXI is approximately 30 times faster than that for Shapley value. Recalling the results from Fig. 3, MoXI achieves higher accuracy in capturing an important group of patches than Shapley value method does. Therefore, MoXI surpasses Shapley value in both accuracy and runtime. While not as fast as Grad-CAM and Attention rollout, we consider that MoXI meets most use cases of visualization, and the quality is better, as our extensive experiments show.

D. Analysis of effective layers to remove patches

In Sec. 5.1, we consider the absence of players (i.e., pixels/patches) for calculating Shapley values and interactions in the input space. Specifically, the patches are removed after the input embedding layer. Here, we examine the case where several self-attention layers are instead masked. To this end, we utilize a variant of the attention-masking approach used in [8]. Specifically, let the k-th layer be our target layer. Then, a large negative value is added to the product of the query and key matrices from k-th to the last self-attention layers. Figure 10 displays the insertion curve

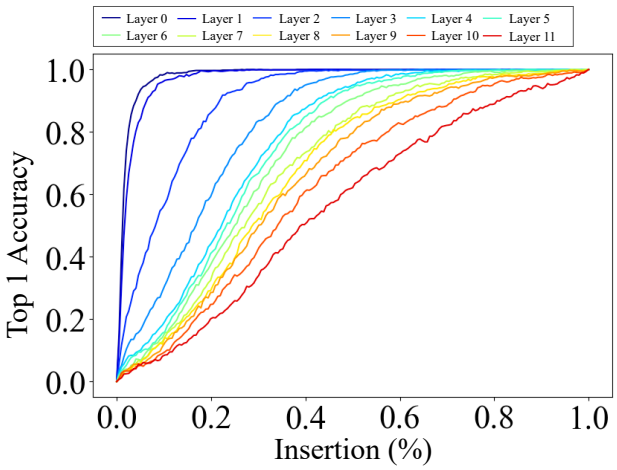


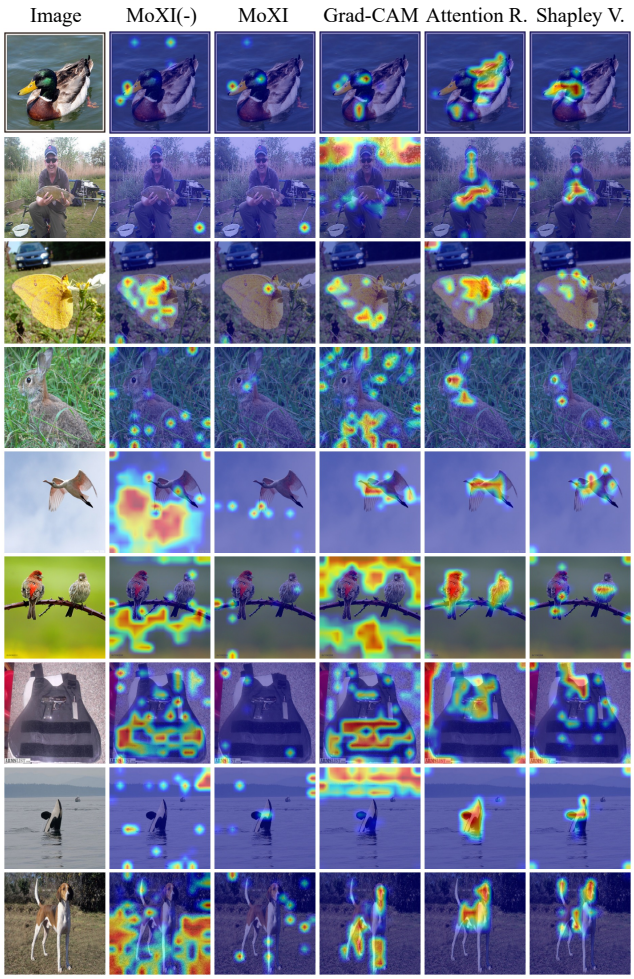
Figure 10. Insertion curves. The curves illustrate the accuracy growth when inserting image patches according to the contributions computed by each method. The horizontal axis presents the insertion rate. The masking method used in the computation of Shapley values and interactions employs attention masking. For the insertion curve experiments, masks used for input to the model for accuracy measurement employ patch deletion.

results when MoXI is applied to various target layers. The experimental setup is the same as in Sec. 5.1. The result demonstrates that MoXI prefers the earlier layers and better pinpoints the important features of images.

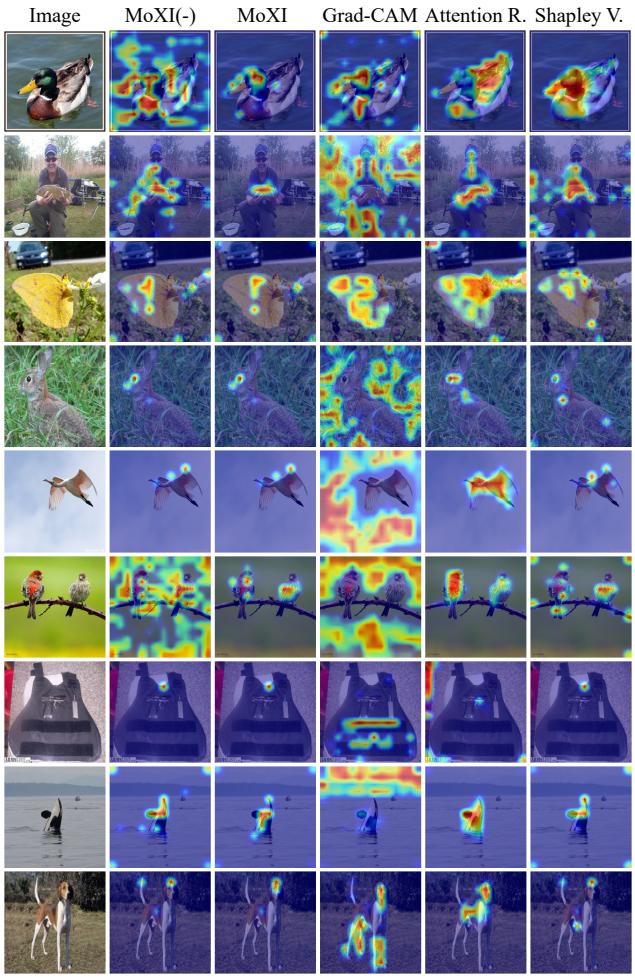
E. Additional results of visualization

We provide additional visualization results in Fig. 11 and 12. As in Sec. 5.2, the results demonstrated that the patches highlighted by MoXI are smaller than those highlighted by other methods.

We observed that MoXI behaves slightly unstable at the insertion case. Recall that in this case, MoXI appends important patches to an empty set accordingly and terminates when the model gives the correct classification. Empirically, the termination can happen at a very early stage, where the confidence score of the correct class is the largest but still very low. If we continue to patch, the model prediction can fluctuate among several classes. Note this does not cause a big problem in most cases; all the insertion curves in this paper consistently show a monotonic increase of classification accuracy with the increase in insertion rate. If needed, one can introduce a minimum confidence score $\tau \in [0, 1]$ and terminate the insertion when the confidence score exceeds this threshold with the correct classification. We include this hyperparameter in our official implementation of MoXI.



(a) Insertion



(b) Deletion

Figure 11. Visualization of important image patches by each method. The highlighted image patches are selected based on their contributions calculated by each method. (a) Highlighting the patches incrementally added to an entire image until classification success. (b) Highlighting the patches sequentially removed from a full image until classification failure.

F. Class-descriminative localization

The proposed method was originally designed to identify important pixels to explain the model prediction. Here, we generalize MoXI (for pixel deletion) to visualize such pixels for a given target class, which is used in Fig. 5. To this end, we consider reward function switching as follows. Let $x, y_{\mathrm{t}}, y_{f(x)}$ be the input image, the target label, and the predicted label, respectively. If $y_{\mathrm{t}} = y_{f(x)}$, we simply use a reward function $f(x) = \log \frac{P(y_{\mathrm{t}} \mid x)}{1-P(y_{\mathrm{t}} \mid x)}$. Otherwise, we use $f(x) = \log \frac{P(y_{f(x)} \mid x)}{1-P_{y_{f(x)} \mid x)}} - \frac{P(y_{\mathrm{t}} \mid x)}{1-P(y_{\mathrm{t}} \mid x)}$, which helps us identify patches with positive effect on the confidence score on class $y_{f(x)}$ and negative effect on class y_{t}. The image patches removed in the former case are collected as important patches for class y_{t}.

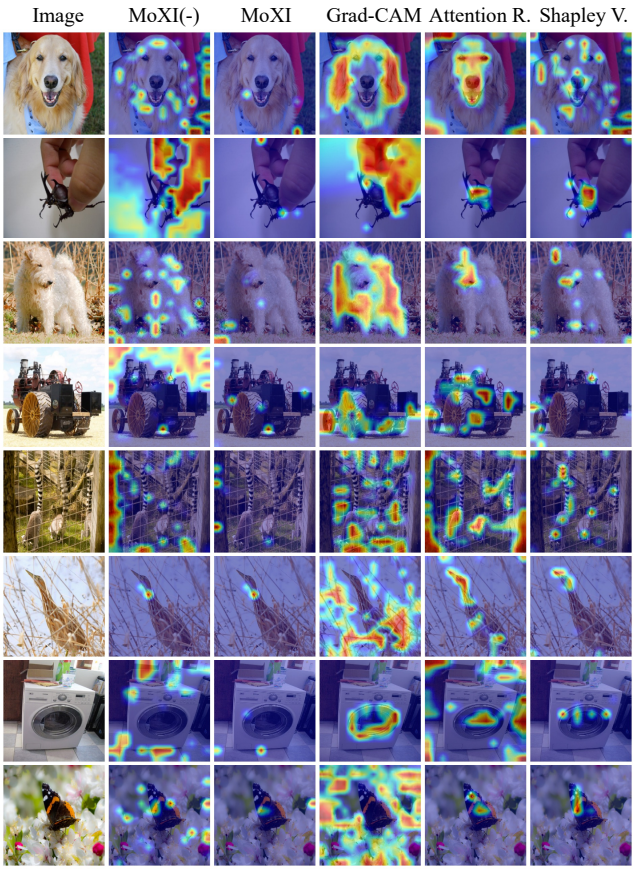
G. Patch perturbations

In Sec. 5.3, we evaluated the effectiveness of each method by measuring the classification accuracy when Gaussian and fog noise were applied to important image patches identified. The deletion curves here are not plotted by removing patches but instead perturbed. We present experimental results on common corruptions and adversarial perturbations.

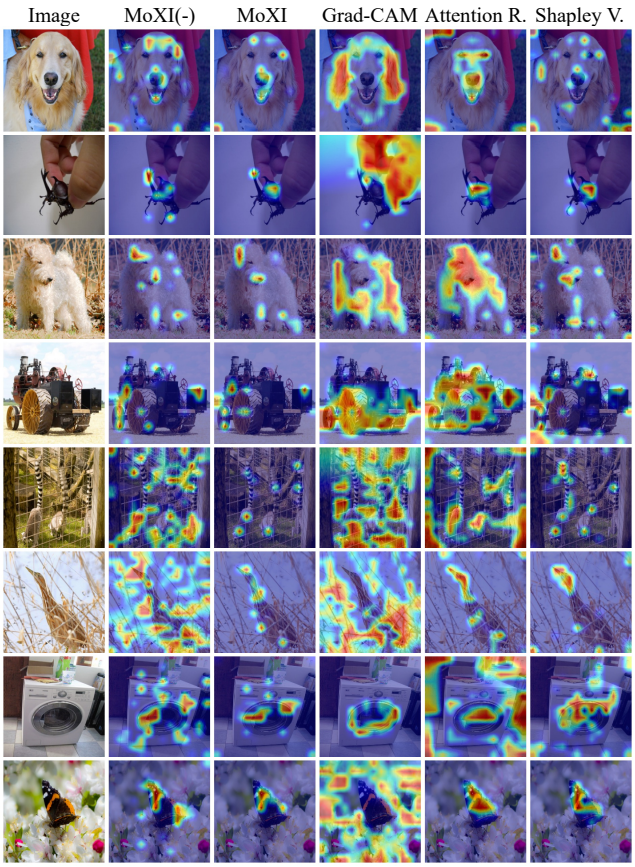
G.1. Common corruptions

We implemented 19 types of common corruptions using the `imagecorruptions` module with severity 5.[6] Fig-

[6] https://github.com/hendrycks/robustness.



(a) Insertion



(b) Deletion

Figure 12. Visualization of important image patches by each method. The highlighted image patches are selected based on their contributions calculated by each method. (a) Highlighting the patches incrementally added to an entire image until classification success. (b) Highlighting the patches sequentially removed from a full image until classification failure.

ures 13 and 14 showcase the deletion curves with different corruptions for ViT-T and DeiT-T, respectively. The results demonstrate that our method gives a sharper decrease at the early stage of deletion curves than others, as in Sec. 5.3.

G.2. Adversarial perturbations

Besides common corruptions, we also investigated the case with adversarial perturbations [12, 17, 19], which are small but malicious perturbations that can largely change the model's output. We conducted the same experiment given in Sec. G.1 but with adversarial perturbations instead of common corruptions. To obtain adversarial perturbations, we adopted L_2-untargeted PGD with $\epsilon = 1.0$ and stepsize $\alpha = 0.2$. Figure 15(a) and 16(a) present the deletion curves for ViT-T and Deit-T, respectively. The results show that the attention rollout method gives a slightly sharper decrease than MoXI. This differs from the results for common corruptions. We suspect that adversarial perturbations mostly lie in the patches that are suggested as important by attention rollout. To confirm this, we measured the magnitude of adversarial perturbations on each image patch. Specifically, the magnitude is measured by the L2 norm. Figure 15(b) shows the magnitude of the perturbations of each patch. The patches are ordered as in the deletion curves in Fig. 15(a). The results indicate that the importance of image patches identified by attention rollout is well aligned with the amount of perturbations on them. On the other hand, image patches identified by MoXI contain a larger amount of perturbations at the early and late stages than those at the middle stage. This may be because the attention rollout reflects the internal computation process of the features directly when measuring the contributions of image patches, while adversarial perturbations are designed to hack this process. On the other hand, MoXI treats a Vision Transformer as a black-box model and is unaware of the internal process.

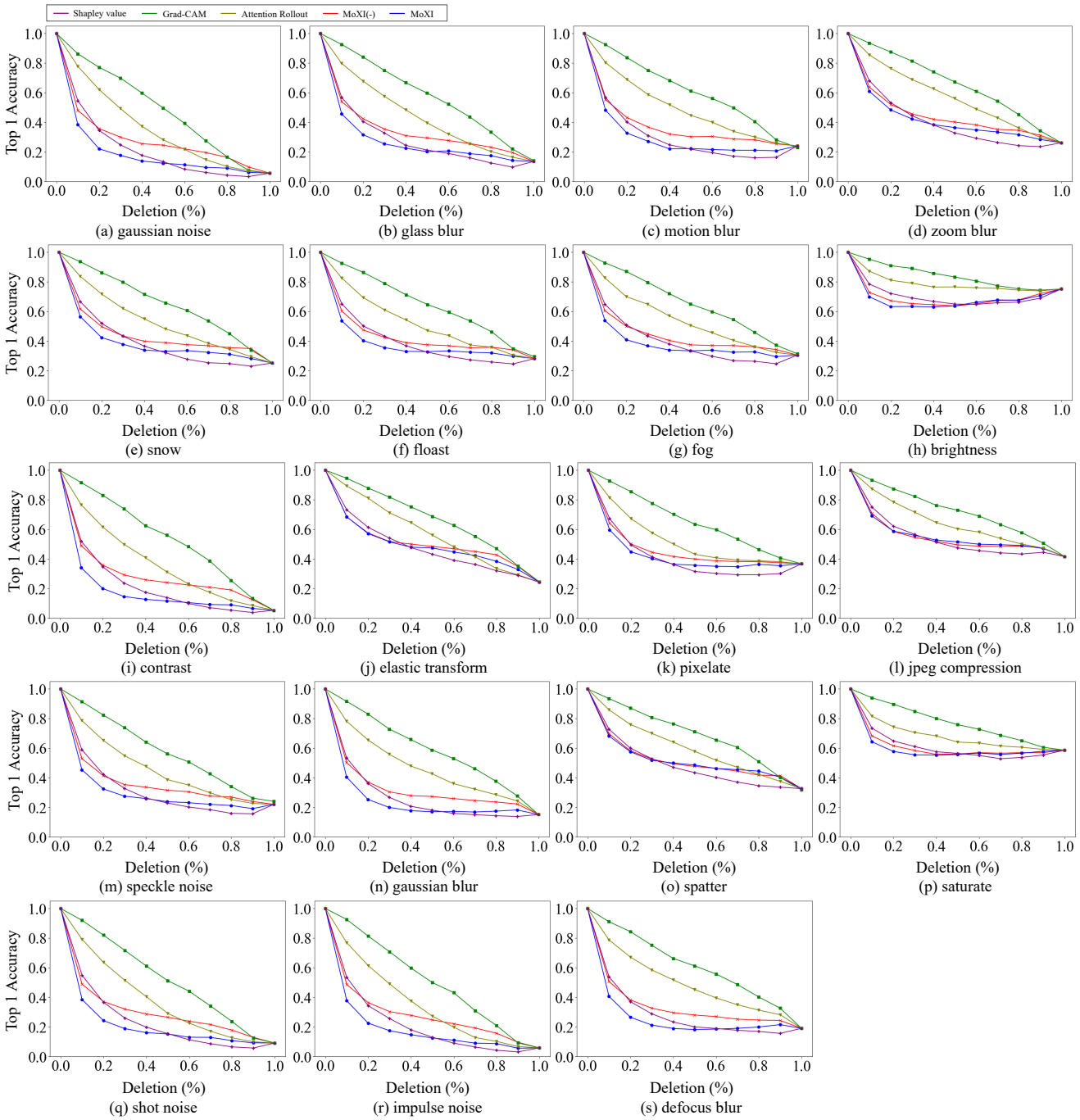


Figure 13. Deletion curves by image corruptions instead of masking with ViT-T. The curves illustrate the change in accuracy along with the increase in the number of corrupted image patches. The patches are corrupted from the highly contributing ones determined by each method.

H. More results in the stability of explanations.

In Sec 5.4, we evaluate the stability of explanations of MoXI and attention rollout with respect to the number of classes. Here, we consider both insertion and deletion metrics, utilizing Grad-CAM, attention rollout, Shapley value, and MoXI. Figure 17 shows insertion and deletion curves. The result again shows that MoXI maintains relatively sta-

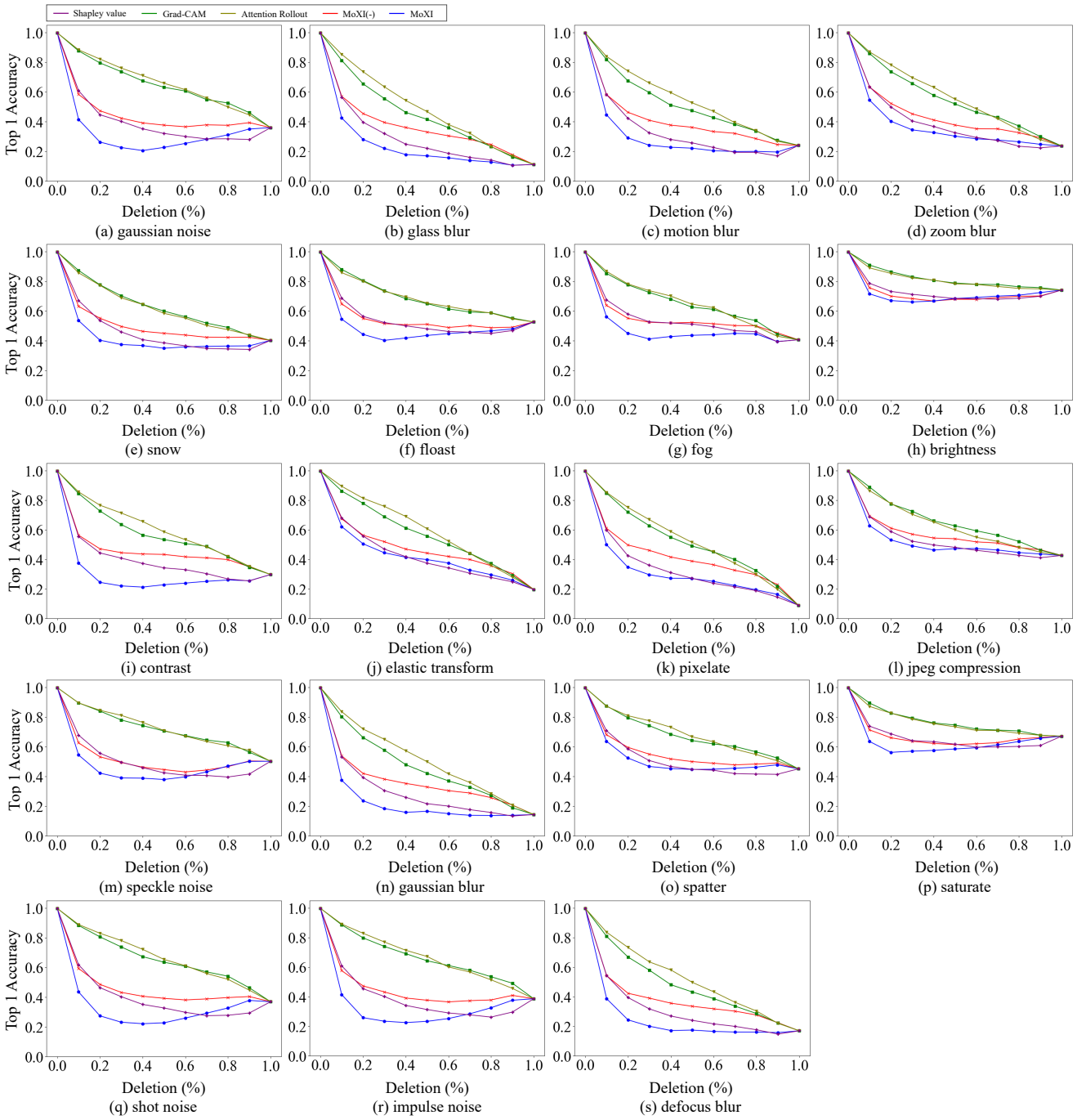


Figure 14. Deletion curves by image corruptions instead of masking with DeiT-T. The curves illustrate the change in accuracy along with the increase in the number of corrupted image patches. The patches are corrupted from the highly contributing ones determined by each method.

ble accuracy when the model is trained on more classes. Similarly, other methods have significantly decreased classification accuracy in such scenarios. Therefore, MoXI acquires important image patches more consistently than other methods.

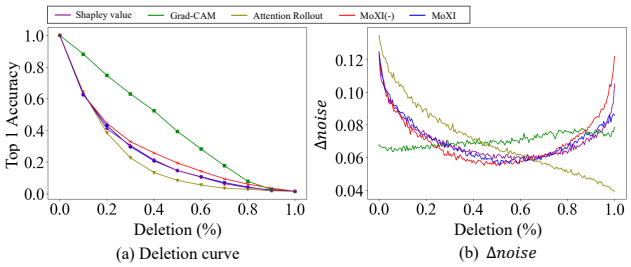


Figure 15. (a) Deletion curves by adversarial perturbations instead of masking with ViT-T. The curves illustrate the change in accuracy along with the increase in the number of perturbed image patches. The patches are perturbed from the highly contributing ones determined by each method. (b) The amount of adversarial perturbations.

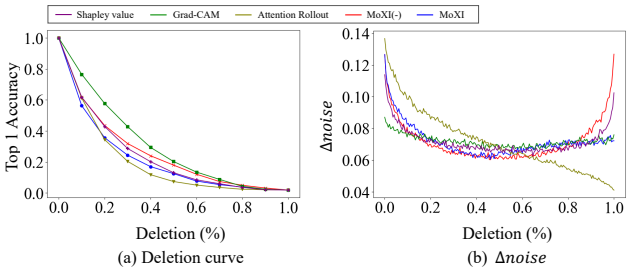


Figure 16. (a) Deletion curves by adversarial perturbations instead of masking with DeiT-T. The curves illustrate the change in accuracy along with the increase in the number of perturbed image patches. The patches are perturbed from the highly contributing ones determined by each method. (b) The amount of adversarial perturbations.

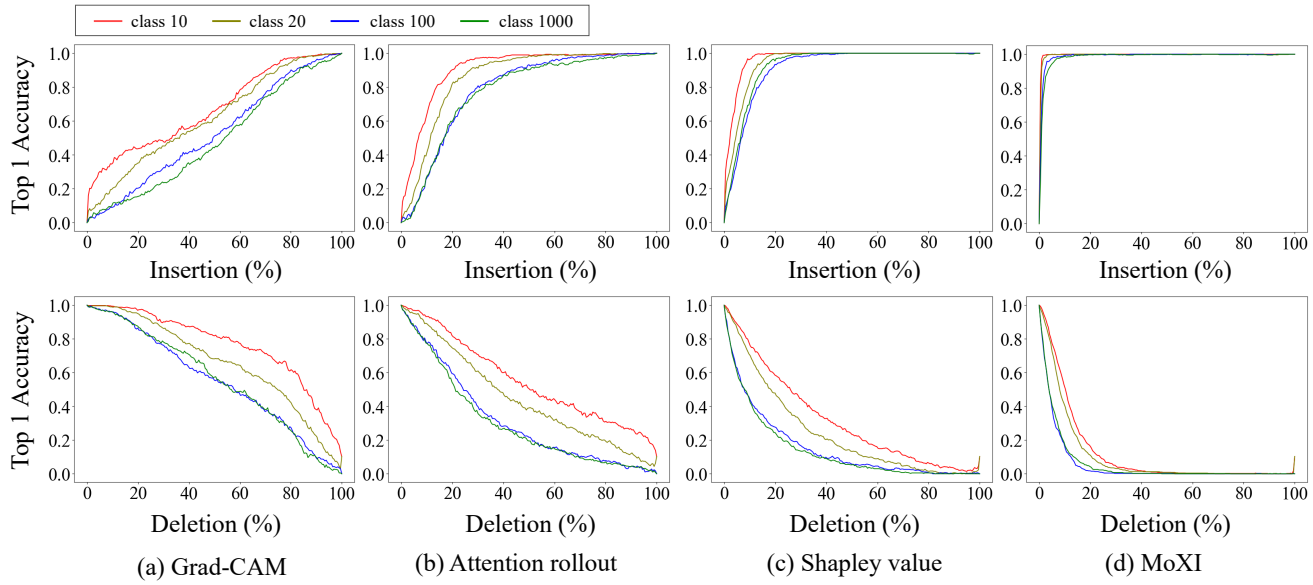


Figure 17. (Top) Insertion curves. (Bottom) Deletion curves. The curves illustrate the change in accuracy along with the increase (decrease) in the number of unmasked (masked) image patches. Each curve represents the results from the pretrained models with 10, 20, 100, and 1000 classes, respectively. (a) Grad-CAM results, (b) Attention Rollout results, (c) Shapley Value results, (d) MoXI results.