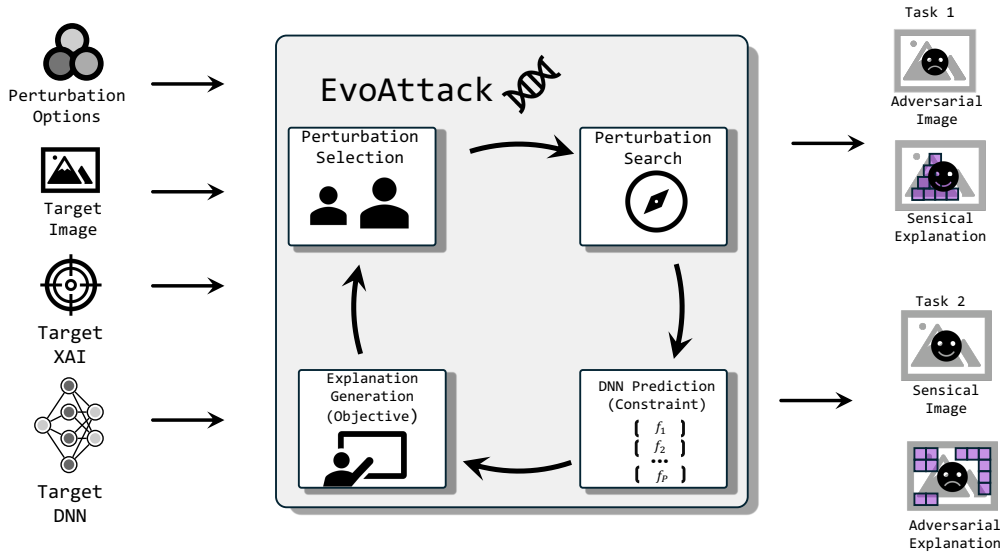

Figure 6: Flowchart demonstrates the process of the proposed EvoAttack method. The proposed method iteratively generates a single child solution by applying random changes to its parent. If the parent solution is dominated (defined in Section 3.4) by its child, it is replaced for the next generation, otherwise the parent remains.

A Appendix

A.1 Proposed Method

A.2 DNN Training

In this section we report the final performance metrics of the trained DNN classifiers. As outlined in Section 4 we fine-tune each chosen ImageNet pre-trained model for 10 epochs, using a batch-size of 32, and a learning-rate of $1e-4$ using an ADAM optimiser Kingma & Ba (2015) and a Cross Entropy Loss.

ImageNet: In this work we make use of ImageNet pre-trained DNN classifiers available from the pytorch library Paszke et al. (2019) (for standard implementation) and the authors original implementation of Böhle et al. (2024). We refer the reader to https://pytorch.org/vision/stable/models.html and https://github.com/B-cos/B-cos-v2 for the performance metrics of used pre-trained models.

HAM10000: This dataset consists of 10000 images of common pigmented skin lesions across 7 classes of difference cancerous lesions. We observe that the fine-tuned model achieve test-set performance metrics similar to existing methods within the literature Nadipineni (2020). We outline the performance metrics of our trained models in Table 5.

COVID-QU-Ex: This dataset contains a total of $33,920$ chest X-rays covering 3 classes of COVID-19, Pneumonia and Normal. We outline the performance metrics of our trained models in Table 6.

Br35h: This dataset contains a total of 3000 images of brain MRI scans with and without a tumor. We present the performance metrics of the trained models in Table 7.

AlexNet				
Class	Precision	Recall	f1-score	Support
0	0.66	0.71	0.68	65
1	0.79	0.77	0.78	109
2	0.66	0.73	0.69	215
3	0.44	0.72	0.55	25
4	0.50	0.78	0.60	206
5	0.96	0.85	0.90	1354
6	0.87	0.90	0.88	29

Accuracy: 0.82

MobileNet				
Class	Precision	Recall	f1-score	Support
0	0.72	0.60	0.66	65
1	0.78	0.85	0.81	109
2	0.77	0.71	0.74	215
3	0.85	0.68	0.76	25
4	0.62	0.73	0.67	206
5	0.94	0.93	0.93	1354
6	0.96	0.93	0.95	29

Accuracy: 0.87

VGG-16				
Class	Precision	Recall	f1-score	Support
0	0.74	0.69	0.71	65
1	0.76	0.82	0.79	109
2	0.67	0.79	0.72	215
3	0.88	0.56	0.68	25
4	0.48	0.81	0.60	206
5	0.96	0.84	0.90	1354
6	0.90	0.97	0.93	29

Accuracy: 0.82

Table 5: Test-set performance metrics of pre-trained ImageNet models fine-tuned on the HAM1000 dataset.

AlexNet				
Class	Precision	Recall	f1-score	Support
0	0.96	0.91	0.94	2395
1	0.89	0.94	0.91	2253
2	0.90	0.89	0.89	2140

Accuracy: 0.92

MobileNet				
Class	Precision	Recall	f1-score	Support
0	0.96	0.97	0.97	2395
1	0.92	0.94	0.93	2253
2	0.94	0.91	0.93	2140

Accuracy: 0.94

VGG-16				
Class	Precision	Recall	f1-score	Support
0	0.99	0.92	0.95	2395
1	0.91	0.96	0.93	2253
2	0.92	0.94	0.93	2140

Accuracy: 0.94

Table 6: Test-set performance metrics of pre-trained ImageNet models fine-tuned on the COVID-QU-Ex dataset.

A.3 Ablation Study

AlexNet				
Class	Precision	Recall	f1-score	Support
0	0.97	0.98	0.97	302
1	0.98	0.97	0.97	298

Accuracy: 0.97

MobileNet				
Class	Precision	Recall	f1-score	Support
0	0.99	0.99	0.99	302
1	0.99	0.99	0.99	298

Accuracy: 0.99

VGG-16				
Class	Precision	Recall	f1-score	Support
0	1.00	0.97	0.99	302
1	0.97	1.00	0.99	298

Accuracy: 0.99

Table 7: Test-set performance metrics of pre-trained ImageNet models fine-tuned on the Br35h dataset.

A.4 Adversarial Training

In this section, we report the final performance metrics of the adversarially trained DNN classifiers. We follow a similar training process as used for the original fine-tuned classifiers and train an ImageNet pre-trained model using a batch-size of 32, and a learning-rate of $1e-4$ using an ADAM optimiser Kingma & Ba (2015) and a Cross Entropy Loss. We augment each batch by adding random perturbations onto the images, resulting in a final batch-size of 64 consisting of both benign and its perturbed counterparts. We generate a random perturbation by first uniformly sampling the parameters N and Maximum Diameter, then randomly sampling the characteristics of the perturbation as outlined in Section 3. We provide the pseudo-code of the procedure in Algorithm 1.

Algorithm 1: Adversarial Procedure employing EvoAttack perturbations

Input: Number of circles grid $\boldsymbol{N}$, grid of Maximum diameters $\mathbf{MD}$, number of epochs T, the batch-size BS, the learning rate lr, the loss function L, perturbation constraint ϵ, classifier f
Initialise θ // Initialise weights
for $t \leftarrow 1; t < T;\ t \leftarrow t+1$ **do**
  for $j \leftarrow 1; j < BS;\ i \leftarrow BS+1$ **do**
    $N \leftarrow \mathcal{U}(\boldsymbol{N})$ // sample number of circles
    Max diameter $\leftarrow \mathcal{U}(\mathbf{MD})$ // sample maximum diameter
    $\boldsymbol{\delta} \leftarrow \mathcal{U}(0,1)^{N\times 7}$ // random
    $\mathbf{x}^*_{tj} \leftarrow CONSTRUCT(\mathbf{x}_j, \boldsymbol{\delta}, \epsilon)$ // Construct adversarial image (see Section 3.3)
  $\theta \leftarrow \theta - \nabla_\theta L(f([\mathbf{x}_t, \mathbf{x}^*_t], [y, y])$ // Update mode weights with some optimizer, e.g. ADAM

A.4.1 Adversarial Training Metrics

In this section, we report the final performance metrics of the trained DNN classifiers. As outlined in Section 4. The evaluation metrics of these models can be found in Tables 8,9 and 10.

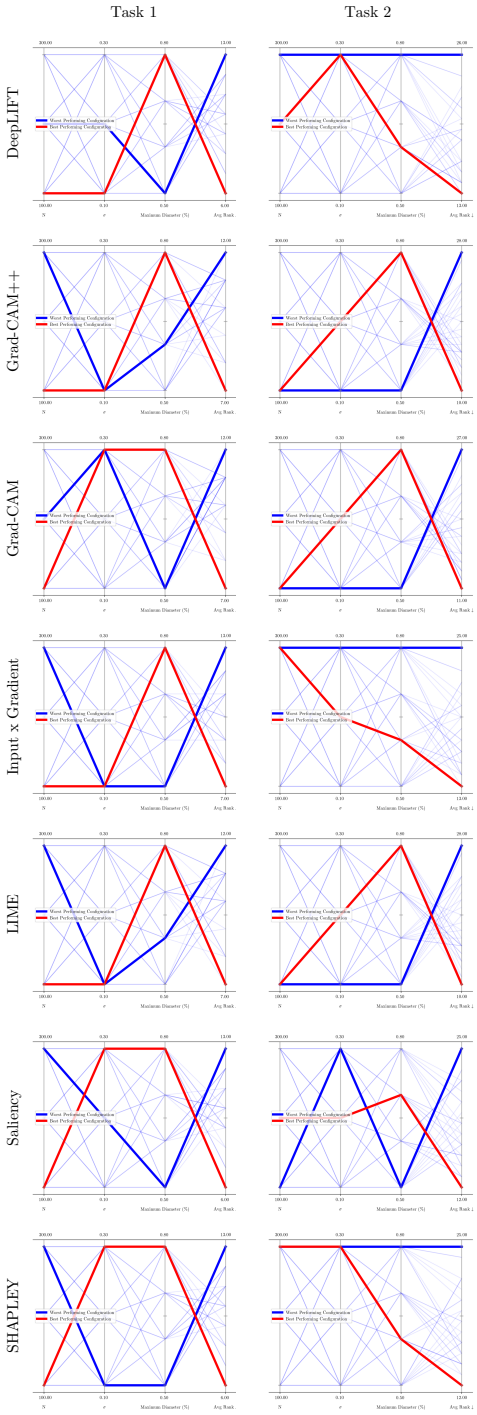

Figure 7: Parallel plots showing the results of configurations testing during the grid search. For each task we use its respective domination relation (described in Section 3.4) to rank its the performance of each configuration when attacking an image. Here we use the average rank (last column) to describe the performance of the configuration, where lower ranks describe better performing configurations. We highlight the best and worst performing configurations in blue and red, respectively.

AlexNet				
Class	Precision	Recall	f1-score	Support
0	0.66	0.73	0.68	65
1	0.79	0.77	0.78	109
2	0.66	0.73	0.69	215
3	0.58	0.72	0.55	25
4	0.50	0.81	0.60	206
5	0.96	0.85	0.90	1354
6	0.87	0.90	0.88	29

Accuracy: 0.82

MobileNet				
Class	Precision	Recall	f1-score	Support
0	0.72	0.69	0.66	65
1	0.78	0.85	0.81	109
2	0.77	0.71	0.74	215
3	0.85	0.68	0.76	25
4	0.77	0.73	0.67	206
5	0.94	0.93	0.93	1354
6	0.96	0.93	0.95	29

Accuracy: 0.87

VGG-16				
Class	Precision	Recall	f1-score	Support
0	0.74	0.75	0.75	65
1	0.87	0.83	0.85	109
2	0.76	0.78	0.77	215
3	0.76	0.52	0.62	25
4	0.69	0.69	0.69	206
5	0.94	0.94	0.94	1354
6	0.96	0.86	0.91	29

Accuracy: 0.88

Table 8: Test-set performance metrics of pre-trained ImageNet models adversarially trained on the HAM1000 dataset.

AlexNet				
Class	Precision	Recall	f1-score	Support
0	0.96	0.91	0.94	2395
1	0.83	0.94	0.91	2253
2	0.90	0.84	0.89	2140

Accuracy: 0.90

MobileNet				
Class	Precision	Recall	f1-score	Support
0	0.96	0.97	0.97	2395
1	0.87	0.94	0.93	2253
2	0.94	0.85	0.93	2140

Accuracy: 0.93

VGG-16				
Class	Precision	Recall	f1-score	Support
0	0.99	0.95	0.97	2395
1	0.89	0.97	0.93	2253
2	0.94	0.89	0.91	2140

Accuracy: 0.94

Table 9: Test-set performance metrics of pre-trained ImageNet models adversarially trained on the COVID-QU-Ex dataset.

A.5 Proposed Method

MobileNet				
Class	Precision	Recall	f1-score	Support
0	0.99	0.99	0.99	302
1	0.99	0.99	0.99	298

Accuracy: 0.99

AlexNet				
Class	Precision	Recall	f1-score	Support
0	0.99	0.98	0.99	302
1	0.98	0.99	0.99	298

Accuracy: 0.99

VGG-16				
Class	Precision	Recall	f1-score	Support
0	0.98	0.97	0.98	302
1	0.97	0.98	0.98	298

Accuracy: 0.99

Table 10: Test-set performance metrics of pre-trained ImageNet models adversarially trained on the Br35h dataset.

Algorithm 2: Evolutionary Strategy for Generated Adversarial Images Against Explainable Methods

Input: Explanation method g, trained DNN classifier f, distance measure $\mathcal{D}$, query budget K, number of circles N, evolutionary step-size σ, benign image $\mathbf{x}$

$\boldsymbol{\delta} \leftarrow \mathcal{U}(0,1)^{N\times 7}$

$\mathbf{x}^* \leftarrow CONSTRUCT(\mathbf{x}, \boldsymbol{\delta}^*, \epsilon)$ // see Section 3.3

$D \leftarrow \mathcal{D}(g(\mathbf{x}^*), g(\mathbf{x}))$ // Measure explanation distance

$y_{\mathbf{x}^*} \leftarrow \underset{p\in\{1,\cdots,P\}}{\text{argmax}} \; f_p(\mathbf{x}^*)$

for $k \leftarrow 1; k < K;\ k \leftarrow k+1$ **do**

  $\boldsymbol{\delta}^{**} \leftarrow \boldsymbol{\delta} + \sigma \cdot \mathcal{N}(0, I)^{N\times 7}$ // Randomly adjust properties

  $\mathbf{x}^{**} \leftarrow CONSTRUCT(\mathbf{x}, \boldsymbol{\delta}^{**}, \epsilon)$

  $y_{\mathbf{x}^{**}} \leftarrow \underset{p\in\{1,\cdots,P\}}{\text{argmax}} \; f_p(\mathbf{x}^{**})$

  $D^{**} \leftarrow \mathcal{D}(g(\mathbf{x}^{**}), g(\mathbf{x}))$

  $y_{\mathbf{x}^{**}} = y \; D^{**} > D \; \boldsymbol{\delta} \leftarrow \boldsymbol{\delta}^{**}$

  $\mathbf{x}^* \leftarrow \mathbf{x}^{**}$

  $y^* \leftarrow y^{**}$

return $\boldsymbol{\delta}, \mathbf{x}^*, D$

A.6 Qualitative Comparison

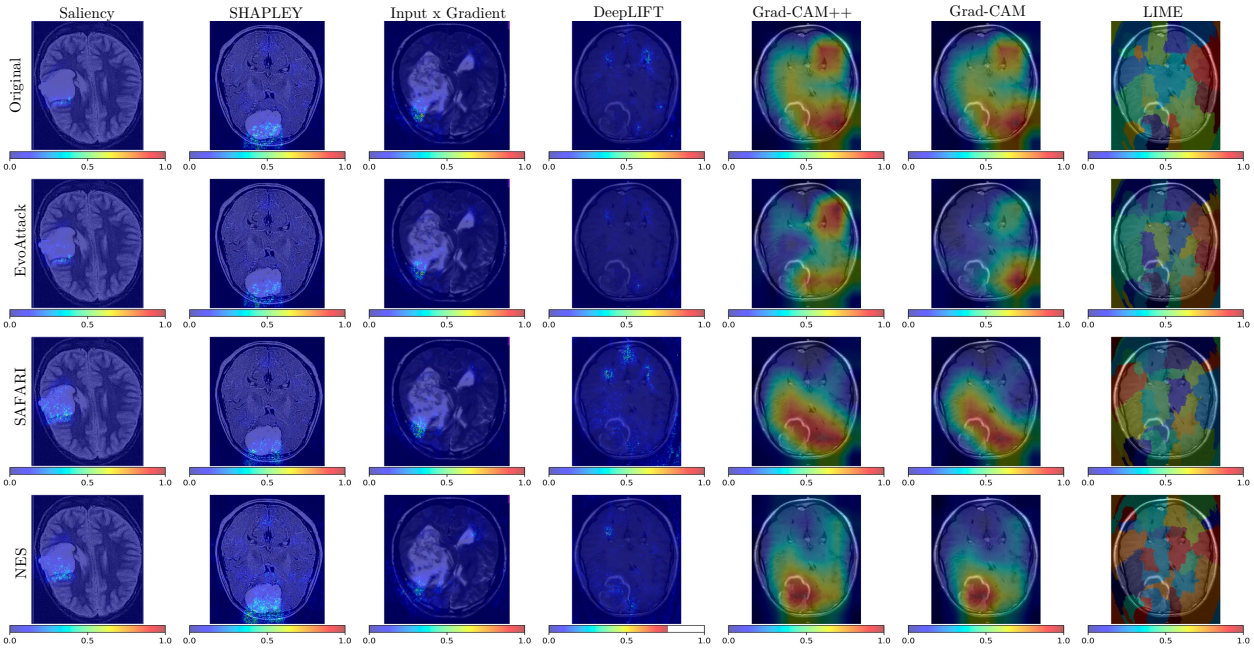

Figure 8: Original Br35h and adversarial images constructed by the proposed EvoAttack, SAFARI and NES attacks, along with attribution maps generated by the respective XAI method. Adversarial images are generated by attacks deployed within the Task 1 scenario. For the majority of the images, the proposed method is able to generate explanations similar to the original explanation, whereas explanations on SAFARI and NES generated adversarial images show larger distortions.

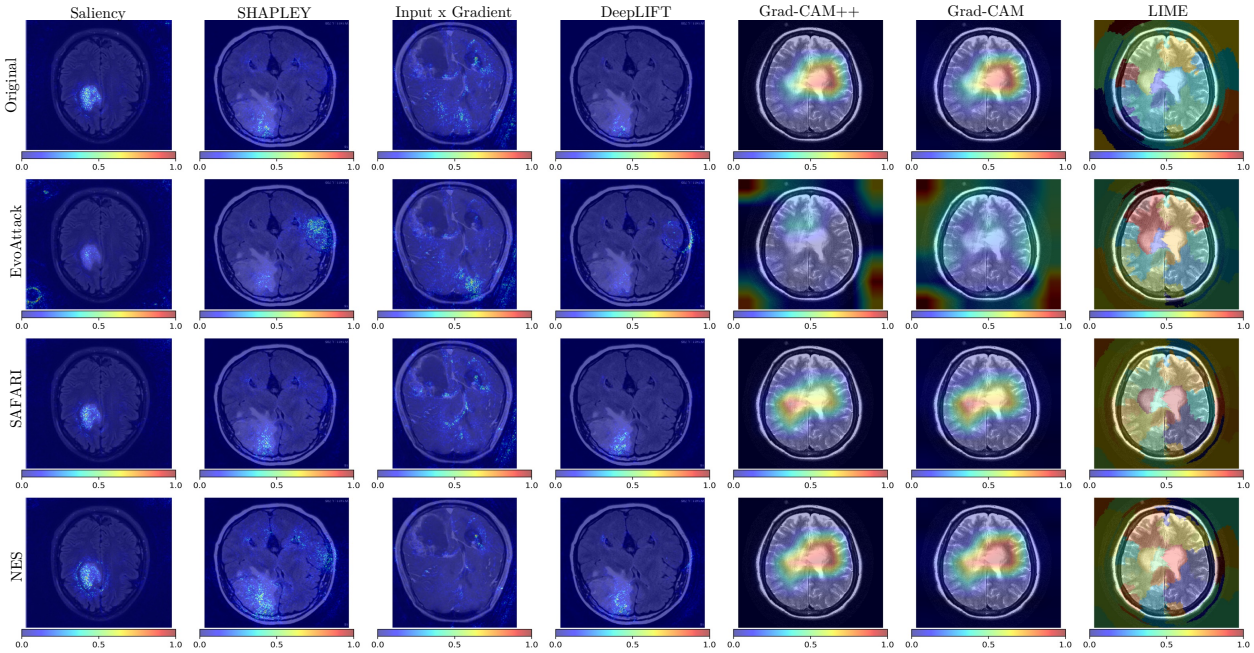

Figure 9: Original Br35h and adversarial images constructed by the proposed EvoAttack, SAFARI and NES attacks, along with attribution maps generated by the respective XAI method. Adversarial images are generated by attacks deployed within the Task 2 scenario. For the majority of the images, the proposed method is able cause larger distortions to the original explanation, compared to explanations on SAFARI and NES generated adversarial images.

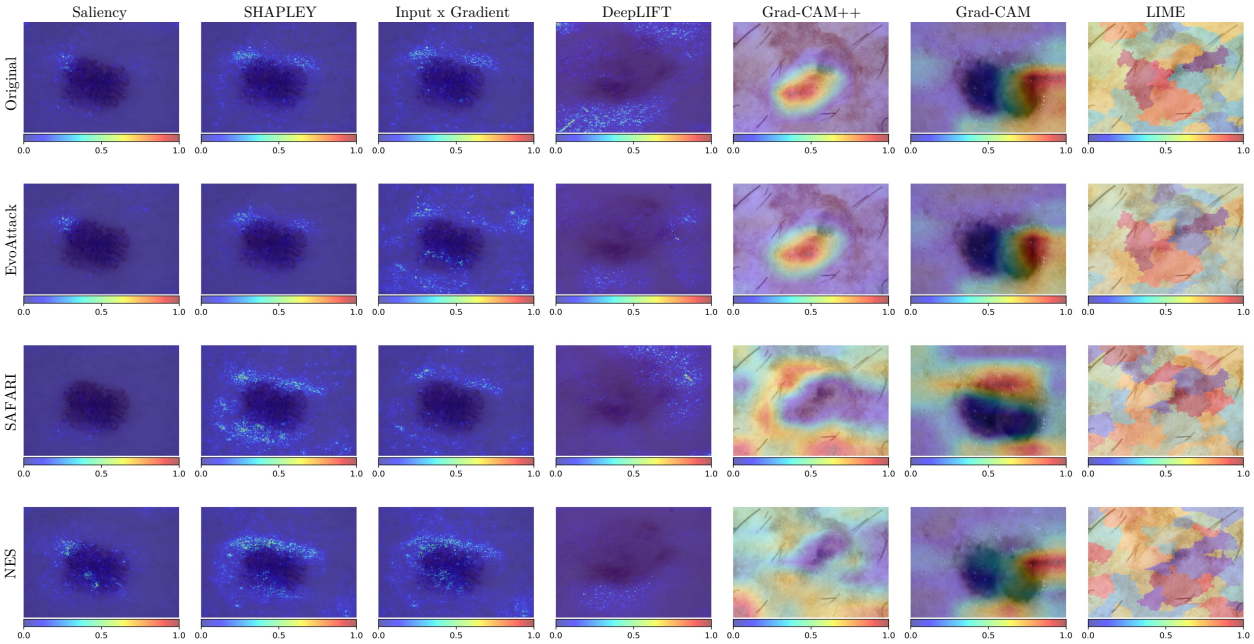

Figure 10: Original HAM10000 and adversarial images constructed by the proposed EvoAttack, SAFARI and NES attacks, along with attribution maps generated by the respective XAI method. Adversarial images are generated by attacks deployed within the Task 1 scenario. For the majority of the images, the proposed method is able to generate explanations similar to the original explanation, whereas explanations on SAFARI and NES generated adversarial images show larger distortions.

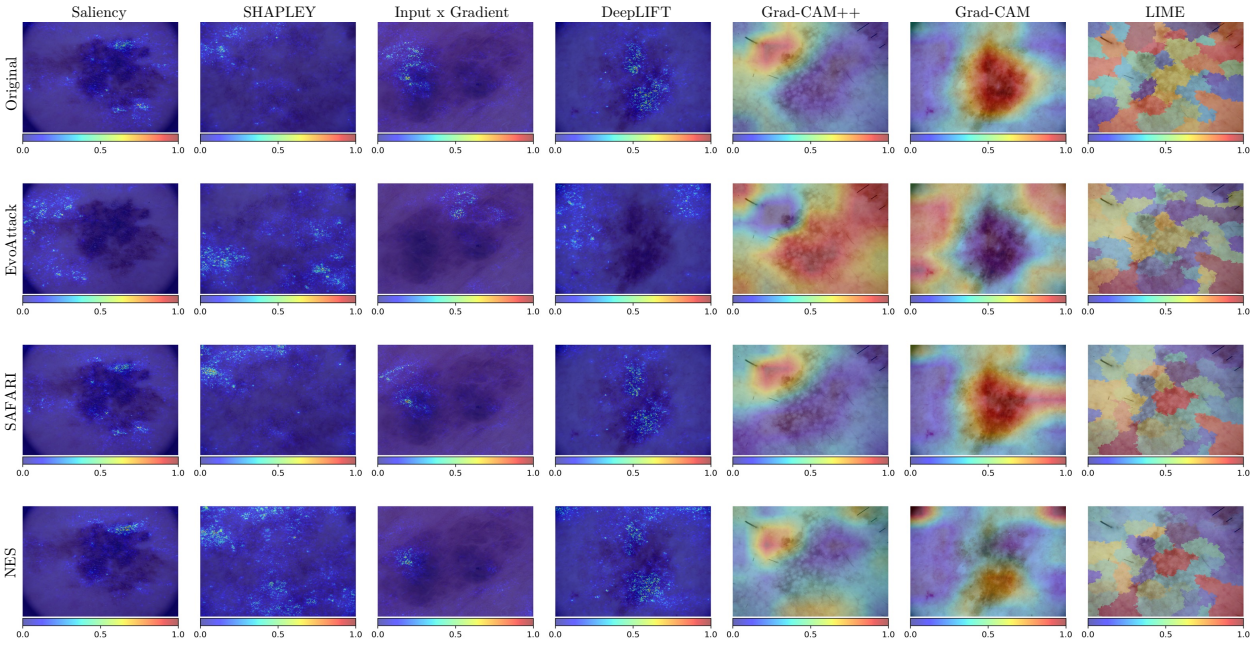

Figure 11: Original HAM10000 and adversarial images constructed by the proposed EvoAttack, SAFARI and NES attacks, along with attribution maps generated by the respective XAI method. Adversarial images are generated by attacks deployed within the Task 2 scenario. For the majority of the images, the proposed method is able cause larger distortions to the original explanation, compared to explanations on SAFARI and NES generated adversarial images.

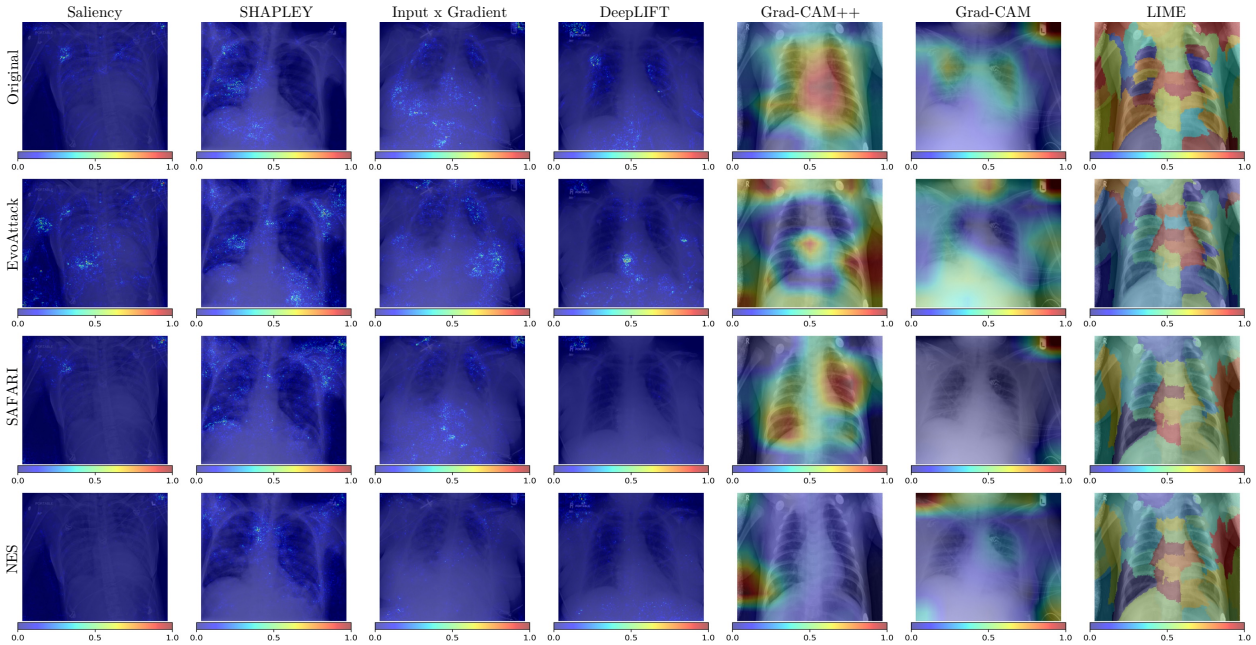

Figure 12: Original COVID-QU-Ex and adversarial images constructed by the proposed EvoAttack, SAFARI and NES attacks, along with attribution maps generated by the respective XAI method. Adversarial images are generated by attacks deployed within the Task 2 scenario. For the majority of the images, the proposed method is able to generate explanations similar to the original explanation, whereas explanations on SAFARI and NES generated adversarial images show larger distortions.

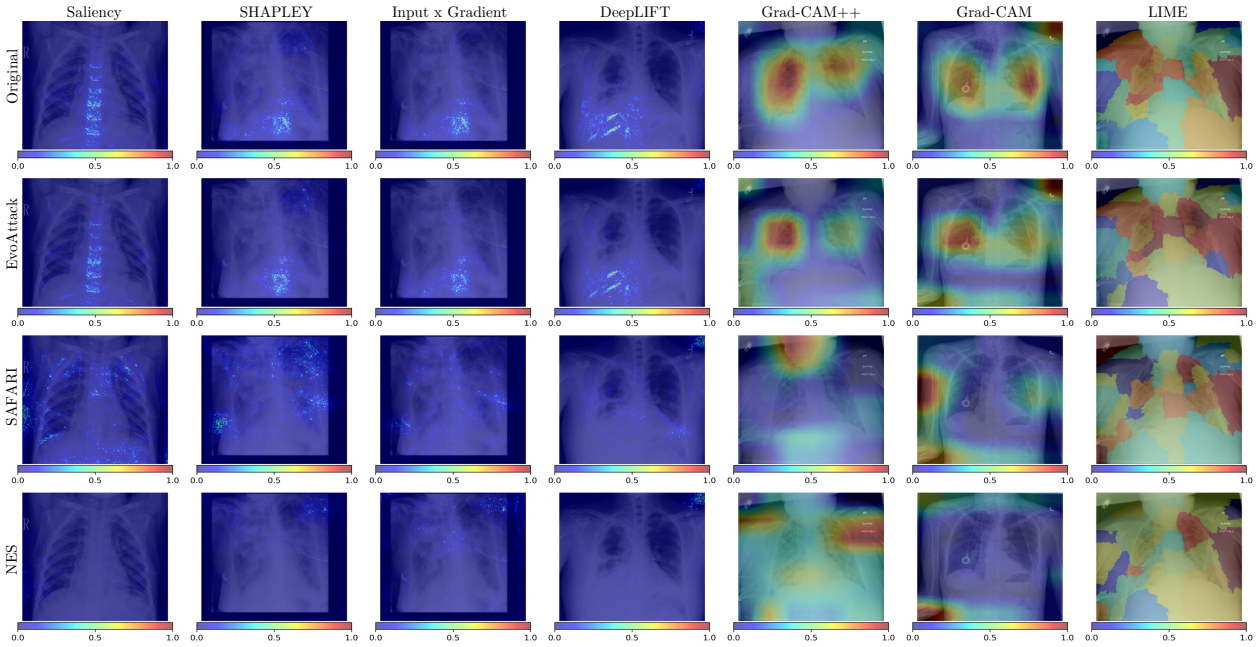

Figure 13: Original COVID-QU-Ex and adversarial images constructed by the proposed EvoAttack, SAFARI and NES attacks, along with attribution maps generated by the respective XAI method. Adversarial images are generated by attacks deployed within the Task 1 scenario. For the majority of the images, the proposed method is able cause larger distortions to the original explanation, compared to explanations on SAFARI and NES generated adversarial images.

Table 11: Table presents the Pearson Correlation Coefficient (PCC) along with the percentage of images that satisfy the respective task constraint when attacking images from the Br35h dataset. We provide the mean and variance of each metric over 10 runs.

DeepLIFT	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**75.5%(2.61)**	0.41(0.05)‡	73.5%(1.87)	0.57(0.03)‡
EvoAttack	74.0%(2.568)	**0.7(0.04)**†	**74.0%(1.898)**	**0.12(0.03)**†
NES	32.0%(2.386)‡	0.17(0.02)‡	37.0%(1.727)‡	0.58(0.02)‡
SAFARI	56.0%(1.649)‡	0.37(0.06)‡	54.0%(2.477)‡	0.55(0.07)‡
Saliency	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**70.5%(1.89)**	0.26(0.09)‡	78.5%(1.085)	0.53(0.08)‡
EvoAttack	69.0%(1.837)	**0.67(0.02)**†	**79.0%(1.086)**	**0.17(0.03)**†
NES	32.0%(2.289)‡	0.27(0.03)‡	38.0%(1.961)‡	0.54(0.09)‡
SAFARI	53.0%(1.52)‡	0.23(0.08)‡	52.0%(1.3)‡	0.6(0.07)‡
Grad-CAM	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**70.5%(2.30)**	0.42(0.04)‡	75.5%(1.54)	0.50(0.05)‡
EvoAttack	69.0%(2.249)	**0.87(0.07)**†	**76.0%(2.121)**	**−0.02(0.1)**†
NES	33.0%(1.049)‡	0.25(0.09)‡	33.0%(1.744)‡	0.51(0.07)‡
SAFARI	58.0%(2.31)‡	0.39(0.02)‡	54.0%(1.544)‡	0.58(0.06)‡
Shapley	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**67.5%(2.85)**	0.34(0.04)‡	64.5%(1.90)	0.58(0.07)‡
EvoAttack	66.0%(2.81)	**0.72(0.07)**†	**65.0%(1.14)**	**0.23(0.08)**†
NES	31.0%(2.989)‡	0.16(0.05)‡	38.0%(2.138)‡	0.59(0.06)‡
SAFARI	56.0%(2.264)‡	0.31(0.01)‡	60.0%(1.911)‡	0.58(0.05)‡
Input x Gradient	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**77.5%(1.93)**	0.27(0.08)‡	65.5%(2.26)	0.51(0.06)‡
EvoAttack	76.0%(1.87)	**0.71(0.01)**†	**66.0%(2.273)**	**0.2(0.01)**†
NES	31.0%(1.703)‡	0.13(0.03)‡	34.0%(2.088)‡	0.52(0.07)‡
SAFARI	57.0%(2.091)‡	0.24(0.09)‡	51.0%(1.403)‡	0.57(0.09)‡
Grad-CAM++	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**70.5%(1.19)**	0.31(0.07)‡	64.5%(2.85)	0.58(0.05)‡
EvoAttack	69.0%(1.106)	**0.71(0.08)**†	**65.0%(1.869)**	**0.3(0.02)**†
NES	30.0%(1.83)‡	0.28(0.06)‡	35.0%(2.926)‡	0.59(0.04)‡
SAFARI	51.0%(2.426)‡	0.3(0.05)‡	58.0%(2.963)‡	0.55(0.06)‡
LIME	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**64.5%(4.45)**	0.37(0.12)‡	83.0%(5.51)	0.55(0.07)‡
EvoAttack	62.75%(4.428)	**0.88(0.16)**†	**84.28%(5.518)**	**−0.04(0.094)**†
NES	33.15%(3.266)‡	0.27(0.092)‡	23.52%(4.145)‡	0.52(0.077)‡
SAFARI	70.78%(5.684)‡	0.36(0.11)‡	57.35%(3.833)‡	0.59(0.043)‡

† denotes the performance of the method significantly outperforms the compared methods according to the Wilcoxon signed-rank test Wilcoxon (1992) at the 5% significance level; ‡ denotes the corresponding method is significantly outperformed by the best performing method (shaded).

A.7 Quantitive Comparison

Table 12: Table presents the Pearson Correlation Coefficient (PCC) along with the percentage of images that satisfy the respective task constraint when attacking images from the HAM10000 dataset. We provide the mean and variance of each metric over 10 runs.

DeepLIFT	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**82.34%(1.89)**	0.60(0.09)‡	78.12%(2.11)	0.34(0.12)‡
EvoAttack	81.19%(1.418)	**0.71(0.084)**†	**80.2%(2.195)**†	**−0.1(0.091)**†
NES	23.76%(1.19)‡	0.25(0.151)‡	75.25%(2.931)‡	0.38(0.15)‡
SAFARI	23.76%(1.705)‡	0.57(0.114)‡	69.38%(1.687)‡	0.68(0.113)‡
Saliency	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**81.47%(2.11)**	0.56(0.08)	80.23%(1.45)	0.29(0.10)
EvoAttack	79.21%(2.893)	**0.55(0.09)**	**83.17%(1.274)**	**−0.0(0.116)**†
NES	23.76%(1.953)‡	0.27(0.134)‡	75.25%(1.268)‡	0.31(0.137)‡
SAFARI	16.83%(2.83)‡	0.54(0.096)	68.02%(2.992)‡	0.51(0.119)‡
Grad-CAM	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**80.92%(1.67)**	0.23(0.21)‡	87.36%(2.01)	0.52(0.19)‡
EvoAttack	79.21%(2.237)	**0.87(0.154)**†	**89.11%(2.119)**	**−0.75(0.232)**†
NES	23.76%(2.04)‡	0.15(0.401)‡	75.25%(2.943)‡	0.55(0.252)‡
SAFARI	16.83%(1.804)‡	0.21(0.416)‡	61.29%(1.073)‡	0.72(0.236)‡
Shapley	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**82.15%(1.73)**	0.61(0.14)‡	79.14%(1.98)	0.41(0.11)‡
EvoAttack	80.2%(1.391)	**0.75(0.099)**†	**80.2%(2.058)**	**−0.02(0.109)**†
NES	23.76%(2.553)‡	0.35(0.183)‡	75.25%(2.003)‡	0.45(0.17)‡
SAFARI	16.83%(2.527)‡	0.59(0.125)‡	60.83%(1.491)‡	0.73(0.108)‡
Input x Gradient	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**79.83%(2.02)**	0.49(0.13)	78.94%(1.54)	0.26(0.15)‡
EvoAttack	78.22%(1.873)	**0.5(0.089)**	**80.2%(1.084)**	**0.01(0.119)**†
NES	23.76%(1.039)‡	0.22(0.125)‡	75.25%(1.485)‡	0.27(0.142)‡
SAFARI	23.76%(2.947)‡	0.48(0.108)	67.67%(1.266)‡	0.5(0.12)‡
Grad-CAM++	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**83.91%(1.78)**	0.88(0.19)	87.42%(2.21)	0.54(0.18)‡
EvoAttack	79.21%(2.406)	**0.96(0.058)**†	**89.11%(1.2)**	**−0.67(0.327)**†
NES	23.76%(2.35)‡	0.43(0.332)‡	75.25%(2.083)‡	0.57(0.246)‡
SAFARI	23.76%(1.567)‡	0.84(0.145)‡	66.93%(1.816)‡	0.85(0.116)‡
LIME	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**90.12%(2.65)**	0.20(0.16)‡	98.21%(3.01)	0.48(0.12)‡
EvoAttack	89.64%(5.775)	**0.89(0.013)**†	**100.0%(5.814)**	**−0.79(0.133)**†
NES	25.76%(3.479)‡	0.19(0.137)‡	94.37%(3.538)‡	0.51(0.14)‡
SAFARI	11.56%(4.372)‡	0.16(0.184)‡	44.38%(3.67)‡	0.74(0.1)‡

† denotes the performance of the method significantly outperforms the compared methods according to the Wilcoxon signed-rank test Wilcoxon (1992) at the 5% significance level; ‡ denotes the corresponding method is significantly outperformed by the best performing method (shaded).

Table 13: Table presents the Pearson Correlation Coefficient (PCC) along with the percentage of images that satisfy the respective task constraint when attacking images from the COVID-QU-Ex dataset. We provide the mean and variance of each metric over 10 runs.

DeepLIFT	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**99.5%(1.872)**	0.48(0.11)‡	77.8%(2.15)	0.05(0.099)‡
EvoAttack	99.01%(1.381)	**0.65(0.118)**†	**78.22%(2.442)**	**0.0(0.111)**†
NES	43.56%(1.273)‡	0.12(0.07)‡	56.44%(1.953)‡	0.09(0.088)‡
SAFARI	43.56%(1.539)‡	0.35(0.175)‡	70.3%(1.234)‡	0.2(0.166)‡
Saliency	Task 1		Task 2	
Square-Attack	**100.0%(1.901)**	0.22(0.08)‡	85.0%(2.10)	−0.06(0.071)‡
EvoAttack	**100.0%(2.308)**	**0.51(0.102)**†	**87.13%(2.271)**	**−0.14(0.053)**†
NES	43.56%(1.687)‡	−0.02(0.076)‡	56.44%(1.123)‡	−0.03(0.08)‡
SAFARI	42.57%(1.33)‡	0.12(0.074)‡	71.29%(2.008)‡	−0.0(0.083)‡
Grad-CAM	Task 1		Task 2	
Square-Attack	**100.0%(2.003)**	0.55(0.17)‡	95.0%(2.30)	−0.01(0.19)‡
EvoAttack	**100.0%(1.456)**†	**0.9(0.159)**†	**96.04%(2.424)**	**−0.82(0.228)**†
NES	43.56%(2.131)‡	0.19(0.206)‡	56.44%(2.262)‡	0.08(0.258)‡
SAFARI	42.57%(2.191)‡	0.42(0.194)‡	71.29%(1.641)‡	0.18(0.331)‡
Shapley	Task 1		Task 2	
Square-Attack	**100.0%(2.102)**	0.49(0.16)‡	77.5%(1.70)	−0.01(0.101)‡
EvoAttack	**100.0%(1.472)**	**0.67(0.105)**†	**78.22%(1.758)**	**−0.01(0.125)**†
NES	43.56%(1.414)‡	0.12(0.092)‡	56.44%(1.675)‡	0.07(0.093)‡
SAFARI	42.57%(2.994)‡	0.34(0.182)‡	70.3%(1.867)‡	0.21(0.175)‡
Input x Gradient	Task 1		Task 2	
Square-Attack	**100.0%(1.657)**	0.31(0.08)‡	77.0%(2.40)	−0.01(0.071)‡
EvoAttack	99.01%(2.234)	**0.57(0.094)**†	**78.22%(2.552)**	**−0.03(0.076)**†
NES	43.56%(1.383)‡	0.03(0.049)‡	56.44%(1.179)‡	0.03(0.078)‡
SAFARI	43.56%(1.535)‡	0.23(0.084)‡	70.3%(1.32)‡	0.09(0.082)‡
Grad-CAM++	Task 1		Task 2	
Square-Attack	**100.0%(1.934)**	0.63(0.15)‡	95.5%(2.20)	−0.01(0.21)‡
EvoAttack	99.01%(2.639)	**0.95(0.058)**†	**96.04%(2.33)**	**−0.82(0.282)**†
NES	43.56%(1.993)‡	−0.0(0.264)‡	56.44%(1.654)‡	0.03(0.304)‡
SAFARI	44.55%(1.789)‡	0.49(0.229)‡	70.3%(1.483)‡	0.16(0.324)‡
LIME	Task 1		Task 2	
Square-Attack	**99.5%(5.412)**	0.56(0.12)‡	76.0%(3.50)	−0.1(0.081)‡
EvoAttack	98.98%(4.97)	**0.88(0.092)**†	**77.79%(3.646)**	−0.8(**0.011**)†
NES	51.33%(5.671)‡	0.22(0.146)‡	64.97%(3.225)‡	0.11(0.062)‡
SAFARI	58.82%(5.307)‡	0.46(0.08)‡	54.34%(5.72)‡	0.14(0.109)‡

† denotes the performance of the method significantly outperforms the compared methods according to the Wilcoxon signed-rank test Wilcoxon (1992) at the 5% significance level; ‡ denotes the corresponding method is significantly outperformed by the best performing method (shaded).

Table 14: Table presents the Pearson Correlation Coefficient (PCC) along with the percentage of images that satisfy the respective task constraint when attacking images from the ImageNet dataset. We provide the mean and variance of each metric over 10 runs.

DeepLIFT	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**82.13%(1.842)**	0.58(0.148)‡	79.92%(2.056)	0.21(0.139)‡
EvoAttack	81.42%(1.503)	**0.72(0.078)**†	**80.48%(2.121)**	**−0.12(0.087)**†
NES	24.05%(1.27)‡	0.26(0.142)‡	75.41%(2.842)‡	0.36(0.16)‡
SAFARI	23.91%(1.662)‡	0.56(0.121)‡	69.22%(1.734)‡	0.67(0.107)‡
Saliency	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	80.31%(2.014)	0.55(0.136)‡	82.11%(1.889)	0.22(0.135)‡
EvoAttack	**79.54%(2.814)**	**0.56(0.095)**†	**82.91%(1.325)**	**−0.02(0.123)**†
NES	24.08%(1.999)‡	0.28(0.129)‡	75.42%(1.217)‡	0.33(0.141)‡
SAFARI	16.59%(2.776)‡	0.53(0.101)‡	68.34%(3.012)‡	0.52(0.124)‡
Grad-CAM	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**79.42%(2.114)**	0.25(0.176)‡	88.12%(1.946)	0.18(0.212)‡
EvoAttack	78.96%(2.312)	**0.88(0.161)**†	**89.34%(2.081)**	**−0.71(0.219)**†
NES	23.98%(1.987)‡	0.14(0.379)‡	75.12%(2.911)‡	0.57(0.263)‡
SAFARI	17.02%(1.833)‡	0.23(0.392)‡	61.41%(1.127)‡	0.74(0.228)‡
Shapley	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**81.12%(2.234)**	0.63(0.141)‡	79.11%(1.698)	0.20(0.158)‡
EvoAttack	80.48%(1.436)	**0.76(0.094)**†	**79.91%(2.111)**	**−0.04(0.114)**†
NES	24.01%(2.604)‡	0.36(0.191)‡	75.53%(2.081)‡	0.44(0.176)‡
SAFARI	16.54%(2.473)‡	0.61(0.132)‡	61.05%(1.442)‡	0.71(0.101)‡
Input x Gradient	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**79.35%(2.016)**	0.51(0.128)‡	79.01%(1.653)	0.19(0.131)‡
EvoAttack	78.44%(1.822)	**0.49(0.093)**†	**80.54%(1.128)**	**0.03(0.115)**†
NES	23.95%(1.084)‡	0.23(0.132)‡	75.61%(1.522)‡	0.29(0.136)‡
SAFARI	23.54%(2.881)‡	0.47(0.115)‡	67.82%(1.214)‡	0.49(0.125)‡
Grad-CAM++	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**80.44%(1.783)**	0.84(0.167)‡	87.34%(1.936)	0.23(0.231)‡
EvoAttack	79.65%(2.389)	**0.95(0.062)**†	**88.76%(1.257)**	**−0.65(0.344)**†
NES	24.15%(2.287)‡	0.41(0.318)‡	75.39%(2.126)‡	0.58(0.252)‡
SAFARI	23.95%(1.612)‡	0.82(0.151)‡	67.08%(1.857)‡	0.86(0.121)‡
LIME	Task 1		Task 2	
Method	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Square-Attack	**90.12%(4.058)**	0.17(0.159)‡	98.31%(3.552)	0.16(0.129)‡
EvoAttack	89.21%(5.692)	**0.88**(0.017)†	**99.52%(5.731)**	**−0.77**(0.128)†
NES	25.41%(3.425)‡	0.18(0.145)‡	94.65%(3.583)‡	0.49(0.134)‡
SAFARI	11.78%(4.414)‡	0.15(0.176)‡	44.67%(3.728)‡	0.72(0.107)‡

† denotes the performance of the method significantly outperforms the compared methods according to the Wilcoxon signed-rank test Wilcoxon (1992) at the 5% significance level; ‡ denotes the corresponding method is significantly outperformed by the best performing method (shaded).

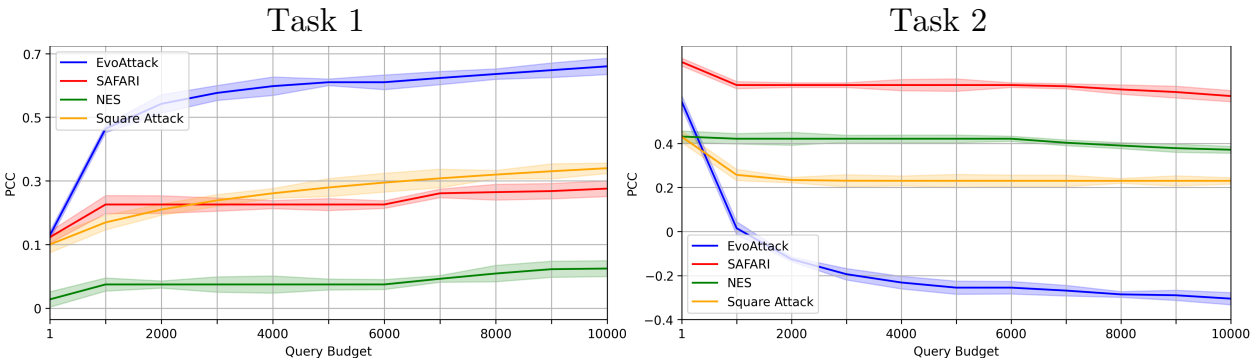

Figure 14: Mean PCC values with variance of attack methods when applied to task 1 (left) and task 2 (right) objectives for varying query budgets. The PCC metric is calculated from adversarial images that satisfy the respective task constraint and is computed by aggregating over all model architectures, explanation methods and datasets.

A.8 Query Efficiency

To further demonstrate the effectiveness of the proposed EvoAttack, we conduct additional experiments for $K \in [1, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000]$. The results in Figure 14 reveal that our method distorts attribution maps (as reflected in changing PCC values) much more rapidly than NES, SAFARI, or Square-Attack. This advantage stems from our method's ability to efficiently handle the high-dimensional perturbation search space as well as its adaptability to different granular XAI methods. EvoAttack reduces dimensionality by representing the perturbation as a set of overlapping circular shapes, which makes it invariant to image size. In contrast, NES and SAFARI directly optimize pixel-wise perturbations, causing them to become trapped in local optima—a common limitation of gradient- and heuristic-based strategies on high-dimension problems. Square-Attack shares some similarity with our approach in constructing shape-based perturbations, but its design limits its effectiveness for explanation distortion.

Table 15: Table presents the Pearson Correlation Coefficient (PCC) along with the percentage of images that satisfy the respective task constraint when attacking defended classifiers and XAI methods trained on the HAM10000, Br35h, COVID-QU-Ex and ImageNet datasets. We provide the mean and variance of each metric over 10 runs. Here adversarial training refers to the approach outlines in Section 4.4, Random Noise refers to the method of Qin et al. and Random Smoothing refers to the defence of Cohen et al..

	Task 1		Task 2	
Defence	Constraint Satisfied (↑)	PCC (↑)	Constraint Satisfied (↑)	PCC (↓)
Adversarial Training	83.33%(3.17)‡	**0.62(0.005)**†	**71.45%(14.14)**†	**0.10(0.025)**†
Random Noise	78.32%(1.92)	0.87(0.089)‡	93.62%(8.20)	−0.03(0.301)‡
Random Smoothing	**77.93%(1.27)**	0.83(0.010)‡	91.72(10.19)	0.0(0.092)‡

† denotes the performance of the method significantly outperforms the compared methods according to the Wilcoxon signed-rank test Wilcoxon (1992) at the 5% significance level; ‡ denotes the corresponding method is significantly outperformed by the best performing method (shaded).

A.9 Comparison of Adversarial Defence Methods

The results in Table 14 compare the effectiveness of the proposed attack against three defense methods. We observe that the randomization-based approaches of Qin et al. and Cohen et al. provide greater resistance to adversarial classification in Task 1. This may be because both methods introduce uncertainty into the task constraint objective, thereby increasing the difficulty of inducing misclassification. However, once a successful adversarial perturbation is found, the adversarially trained model offers substantially stronger resistance. This outcome is expected, as randomization and smoothing defenses do not provide protection against explanation distortion once the classification constraint has been satisfied.

For Task 2, Table 14 shows that adversarial training offers more resistance to the proposed attack compared to the randomization-based defenses. This is likely due to the nature of the task: Task 2 requires that the image remain correctly classified. Since the primary goal of the randomization defenses is to prevent misclassification, ensuring correct classification is comparatively trivial.

A.10 Comparison of Attack Speed

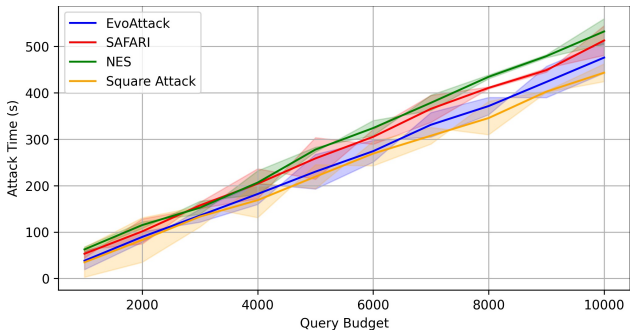

Figure 15: The figure shows the differing attack speeds (time to complete an attack) across different query budgets K. We see the speed of attack between methods remain similar. This can be expected as we assume the inference time of the attacked classifier being the largest factor of attack speed.

A.11 Black-Box Adversarial Attacks

Despite the proposed method focusing on adversarial attacks against explainable methods, it can also be directly applied to black-box adversarial attacks but focusing solely on satisfying the constraint in (1). The black-box scenario in adversarial example generation can be categorized based on the assumed information obtained from the attacked DNN when queried with an input image. In this paper, we focus on the assumption that the attacker has access to the probabilities associated with each class, known as score-based attacks. However, other works address more restricted scenarios where only the top $\hat{C}$ class probabilities are accessible to the attacker Ilyas et al. (2018). Additionally, some work consider situations where only the predicted label of the attacked DNN is available to the attacker, referred to as decision-based attacks Brendel et al. (2018).

Most existing black-box score-based attack methods can be categorized into three main groups: transfer-, gradient-, and heuristic-based methods. The following subsections discuss notable methods from each of these categories.

Transfer-based methods conduct white-box attacks upon a surrogate model of the targeted DNN, to which the generated adversarial examples can be transferred Papernot et al. (2016); Guo et al. (2019); Cheng et al. (2019). Although these approaches claim to be query efficient by leveraging additional knowledge, they do not account for the computational cost incurred by training such substitute models. This can be particularly demanding when tackling large datasets like ImageNet. Furthermore many existing works assume the architecture of the surrogate model is highly similar to the target DNN, allowing a high percentage of adversarial examples to be successfully transferred.

Gradient-estimation methods works make use of white-box gradient-based attack methods, replacing the true gradient with an estimation. They have been widely studied for adversarial example generation by leveraging predicted class probabilities from DNNs to construct a loss function. Finite differences and some off-the-shelf gradient-based algorithms are then used as the optimizer, e.g., Chen et al. (2017); Tu et al. (2019); Ilyas et al. (2018); Bhagoji et al. (2018); Uesato et al. (2018); Yu et al. (2025). For example, Chen et al. proposed a zeroth-order optimization (ZOO) algorithm that applies Adam Kingma & Ba (2015) to generate adversarial examples. However, due to the high dimensionality of the image space, these techniques often require a large number of queries to estimate the perturbation's gradient at each step, which may not be practical in realistic black-box scenarios.

Heuristic methods circumvent the issues associated with gradient estimation, some research has focused on gradient-free heuristics to search for effective perturbations, and have achieved improved performance over

gradient estimation methods when the number of DNN queries is restricted Alzantot et al. (2018); Qiu et al. (2021); Andriushchenko et al. (2020).