A. Motivation - Direct Extension of Pixel Space Adversarial Attack Does Not Work Well.

There is a large body of existing works that generate adversarial samples by constraining perturbation in the pixel space, such as PGD [20], C&W [4], BIM [18], and FGSM [10]. They have a similar working mechanism. Take BIM as an example. Given a perturbation bound such as $8/255$, BIM iteratively perturbs individual input pixels following the gradient direction of a cross-entropy loss function (that tries to induce misclassification). When a perturbed pixel value exceeds the bound, it simply clips the value. The first design we explored was an extension of BIM in the feature space. Specifically, we first identify internal neuron activation value ranges by profiling on the training input. Like the value bound in the pixel space attacks, we use a *min-max* bound that determines the lower and upper bounds of an internal activation value based on its range and a fixed percentage. For example, assume the profiled value range of a neuron is $[0, 1000]$, and the current value of the neuron for a benign sample is 500, a 10% bound allows the value to vary in $[400, 600]$. We clip the activation value if it exceeds the bound. Note that clipping suggests that the gradient of this neuron becomes 0, and hence it has no effect in updating input pixels. We update input pixels according to gradients just like in BIM. Instead of bounding the input pixel variations, we regulate the internal activation value variations.

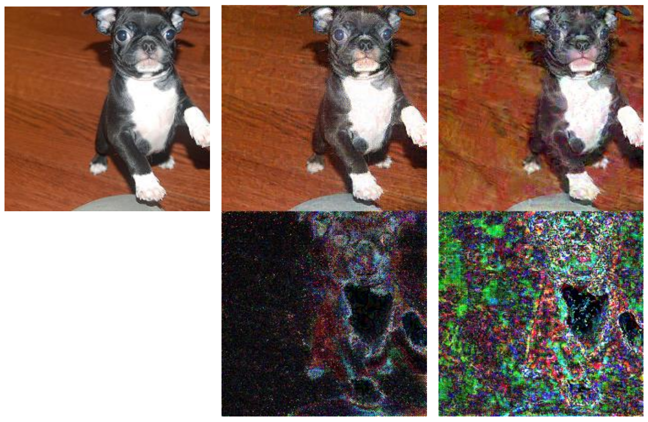


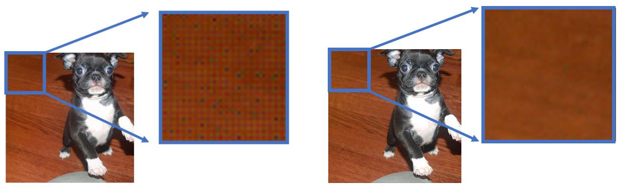
Figure 6. Untargeted attack on ResNet152-Adv using an extension of classic pixel-space attack and our Attack. The second column is our method. For the extension, we use a 2% min-max bound. See Appendix L and Appendix F for details. The first row presents the original image and the corresponding adversarial samples. The second row visualizes the perturbation applied on the natural image. Conf. means attack confidence.

However, we find that such a direct extension does not work well. Figure 6 shows a benign example, the adversarial version by our technique, and the adversarial version by the extension. Observe that the sample by the extension does not look natural and has a smaller confidence value compared to ours. The reason is that internal space is different from the pixel space. A fixed (percentage) value range makes sense in the pixel space as it directly reflects a fixed level of human perception variation. In contrast, a fixed internal percentage value range may imply various levels of pixel changes and hence various human perception levels. A comparative experiment can be found in Appendix F. This motivates our new attack design.

B. Binary Search for Optimization Step Size

In our adversarial example generation, a proper step size (learning rate) is crucial for reliable optimization. A small change on the input can lead to a large quantile change on an internal throttle plane, which makes the optimization osillating and may even lead to numerical exceptions. Choosing an optimal step size depends on model structure and the selected throttle plane(s). It is impossible to manually preset a step size for all the cases. We hence leverage binary search to look for an appropriate step size.

Specifically, we first determine a possible search range for step size, e.g. $[0, 255]$ for the gradient sign method on RGB values. We then choose the median value of the search range as a probing step size. We use this probing step size to conduct optimization for a given number of steps. If the internal quantile change goes beyond the boundary, it means the probing step size is too large for the optimization. We hence update the upper bound of the search range to the current probing value. Otherwise, we update the lower bound with the probing value. We repeat the above search procedure for a given number of iterations.



(a) Before (b) After

Figure 7. Results before and after smoothing for VGG16

C. Feature Smoothing

Occasionally, we observe the generated adversarial examples exhibit checkerboard patterns. Figure 7a shows a typical adversarial example with checkerboard pattern (zoomed in on the right). We observe these cases often occur when we use VGG16 as the reference model (the other reference model we use is ResNet152-Adv). We speculate that this is because we enforce bounds for individual values (on a throttle plane) independently and do not consider their joint distribution of nearby neurons. To mitigate the problem, we add a feature smoothing loss to the optimization goal. The intuition is that individual values (on a plane) have a similar trend of change with their neighboring values. Thus we calculate the average of surrounding changes and use a mean squared error loss to prevent the change from being too far away from the average. Suppose $\mathrm{y} \in \mathbb{R}^{D\times H\times W}$ denotes a throttle plane with channel D, height H and width W. The quantile changes are written as $\Delta Q_{\mathrm{d,h,w}} = |C_{\mathrm{d,h,w}}(\mathrm{y}^{\mathrm{adv}}_{\mathrm{d,h,w}}) - C_{\mathrm{d,h,w}}(\mathrm{y}^{\mathrm{nat}}_{\mathrm{d,h,w}})|/\epsilon$. The average of changes made to nearby values can be formulated with an average pooling operation $\mathrm{AvgPool}_{3\times 3}(\Delta Q)$. Thus we expect the smoothness loss, written as $\frac{\alpha}{D\cdot H\cdot W}\sum_{\mathrm{d,h,w}}[\mathrm{AvgPool}_{3\times 3}(\Delta Q)_{\mathrm{d,h,w}} - \Delta Q_{\mathrm{d,h,w}}]^2$ to be small. We empirically set the weight of smooth loss to $\alpha = 10$. This can lead to 6% improvement in the human preference rate when VGG16 is used as the reference model. The smoothness loss largely eliminates the checkerboard pattern as shown in Figure 7b. We also conduct an ablation study of feature smoothing in Appendix G.

D. User Study

For the human study, there are 47 settings in total (for the eight attacks), including the studies in Appendix K and L. Adversarial samples and generated and labeled by MTurk users for each setting. There are 282 users participated in our studies. 275 out of the 282 responses are considered valid, with those deviating far from the majority (exceeding two times of the standard deviation) removed. We post all the examples used in the human study on an anonymous website [37].

In this section we report the procedure of the user study as shown in Figure 8. Our user study closely follows the one in exiting work [40]. For each image pair, users will go through three steps to complete the study. In the first step, an adversarial sample and a real image are shown in a random order for 2.5 seconds each. In the second step, users are asked to choose which image looks unnatural. Specifically, users are given the following instructions.

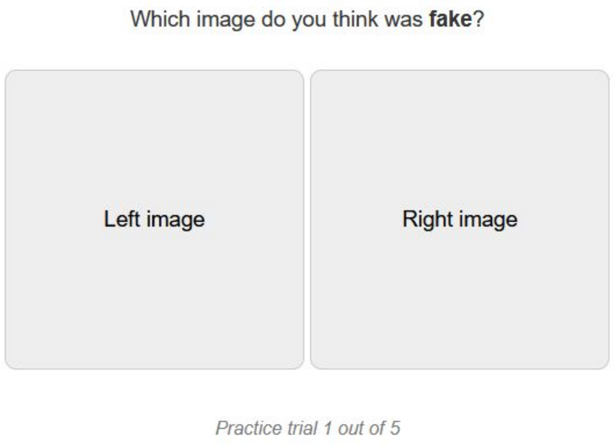
> "You will take part in an experiment involving visual perception. You'll see a series of pairs of images. In each pair, one image is a real photo while the other image is a 'fake' image generated using a computer program. Your task is to determine which image is fake. Sometimes the fake images may look very plausible."

There is an additional third step in the test drive, in which users are given feed-back of whether their answers are correct.

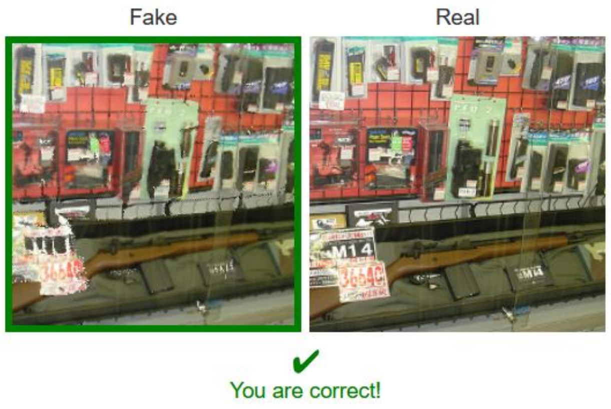
E. Choosing Throttle Planes

As we know, different layers represent features of various types, e.g., shallow layers for concrete features and deep layers for abstract features. For imperceptibility, a good idea is to simultaneously harness both concrete features (e.g. local textures) and abstract features (e.g. global outlines). Driven by this intuition, we use multiple throttle planes simultaneously instead of a single one. Guided by our principle of looking for normal distributions, we check the normality of various layers in a model and identify a throttle plane list. We empirically choose three representative throttle planes for each model. We conduct the selection of throttle planes for ResNet152-Adv and VGG16. The specific locations of throttle planes we choose can be found in Table 3, and the corresponding distribution samples from these chosen throttle planes can be found at Appendix R. Note that although our reference models are these two, the target models can be arbitrary. For the visual quality studies, we consistently used the throttle planes from VGG16.

(a) Step One


Which image do you think was **fake**?
Left image
Right image
Practice trial 1 out of 5

(b) Step Two



(c) Optional Step Three

Figure 8. Example of the human study (images zoomed out). The adversarial sample used in the example is from spatial attack. During the real survey, users will see images with a similar size of the screen.

Table 3. Chosen throttle planes for each model

Model	Plane 1	Plane 2	Plane 3
ResNet152-Adv	a. the first conv.	b. group 1	b. group 2
VGG16	a. conv1_2	a. conv2_2	a. conv3_3

Notation a. represents right after an operation. Notation b. represents the throttle plane which lies in the last block in the group and before the last ReLU operation; conv. represents a convolution layer.

F. Comparing Optimization Methods

As discussed in Section 3.2, there are other optimization methods that can be used for our adversarial example generation such as two-step optimization and clipping. We conduct an experiment for those methods in comparison with our polynomial barrier method. The same optimizer, i.e., the Gradient Sign Method, is used for all the methods during evaluation. We use the adversarially trained ResNet152 model as our study subject. The throttle plane used in this evaluation is the plane in the last layer of Block 2. We use a bound of 1% of the difference between the minimum and maximum activation values, which are computed on the entire training dataset. Here, we use a value bound instead of a quantile bound to make different optimization methods comparable. For the two-step optimization method, we set the number of iterations to 100. At each iteration, we first optimize the internal values once and then the input ten times. We set the step size to 10% of the internal boundary for the internal optimization, and $2.6e{-}3 \times 255$ for the input optimization. This optimization setting is similar to that in the paper [17]. For the clipping method, we optimize for 1000 iterations, and at each iteration we update the input once using the step size of $2.6e{-}3 \times 255$. For our algorithm, we run 1000 iterations with the step size of $6.2e{-}3 \times 255$.

Table 4 illustrates the results. Feasibility denotes the percentage of samples remaining in boundary after the optimization. Average size denotes the average boundary size of all the samples. We calculate the size using the equation $\max_{i\in\mathcal{I}}\left[\frac{\mathrm{ReLU}({y_i}^{\mathrm{adv}}-{y_i}^{\mathrm{nat}})}{\mathrm{high}-{y_i}^{\mathrm{nat}}}+\frac{\mathrm{ReLU}({y_i}^{\mathrm{nat}}-{y_i}^{\mathrm{adv}})}{{y_i}^{\mathrm{nat}}-\mathrm{low}}\right]$. Consistency denotes if the target internal values can be produced in the *original model*. Note that these optimization methods insert additional operations (e.g., clipping) that essentially change the dataflow of the original model. An observed internal value in the optimizing model may not be feasible in the original model. Success rate measures the percentage of generated samples that successfully induce misclassifications. It can be observed that most samples are still feasible after our optimization, while the other two methods cannot enforce the bound. Note that even though the clipping method clips internal values and suppresses gradients, updates on the input can still induce internal values that go beyond bound. The average boundary size of our method is much smaller than the other two, indicating that our barrier loss function can effectively enforce the bound. Adversarial samples generated by the other two methods are hence much less natural-looking. For consistency, we observe that the two baseline methods do not have any guarantee. As the subject model is adversarially trained and the internal bound is very tight, the attack is difficult to succeed. Nonetheless, our method still outperforms the baselines regarding the attack success rate.

Table 4. Comparison of optimization methods.

Method	Boundary		Consistency	Succ. Rate
	Feasibility	Avg. Size		
Polynomial Barrier Method	**97%**	**91%**	**Yes**	**25.8%**
Two-step Optimization	0%	204%	No	21.9%
Clipping	0%	429%	No	21.9%

G. Ablation Study on Feature Smoothing

In this section, we study the effectiveness of feature smoothing in eliminating artifacts. We report human preference rate and attack success rate of untargeted attacks on ResNet152-Adv in Table 5 with controlled experiments (with and without feature smoothing). We conduct the experiment using VGG16 as the reference model (since the other reference model does not introduce the checkerboard pattern). The results show that feature smoothing increases both success rate and human preference for the VGG16 reference model. A possible explanation is that feature smoothing, as a regularization term, decreases the difficulty of internal optimization, and thus implicitly increases the success rate. We find that the checkerboard pattern is largely eliminated. As these patterns provide a shortcut for humans to decide the naturalness, different scales of D^{2}B with the checkerboard pattern have similar human preference rates (first and third row in Table 5). After applying feature smoothing, the pattern disappears and users can not easily distinguish the generated images from the real ones.

Table 5. Human preference of generated examples with and without feature smoothing

Method	Reference Model	Smoothing	Human Pref.	Success Rate
D^{2}B50	VGG16	✗	14%	100%
D^{2}B50	VGG16	✓	20%	100%
D^{2}B10	VGG16	✗	13%	62%
D^{2}B10	VGG16	✓	50%	67%

H. Evaluation on Different Models and Their Corresponding Distances

In this section, we evaluate D^{2}B with different quantile changes on 4 models including DenseNet, MobileNet, VGG19 and ResNet50. We use the same setting as in Section 4. The results are shown in Table 6. We have similar observations as in Table 1 (Section 4). With a similar or higher level of attack confidence, our attack has a smaller ℓ_2 pixel distance and ℓ_∞ quantile distance on all the three planes compared to BIM4. This indicates that our attack is more effective in bounding internal perturbations and can generate more natural-looking adversarial examples. We also observe that D^{2}B induces a larger ℓ_∞ pixel distance with a smaller ℓ_2 pixel distance compared to BIM4 at similar level of attack confidence, which indicates the piggy-backing nature of our attack.

Table 6. Targeted attacks on various models

Models	Attack	Confidence	Pixel Distance		ℓ_∞ Quantile Distance		
			ℓ_2	ℓ_∞	Plane 1	Plane 2	Plane 3
MobileNet	BIM4	47.64	10.51	0.04	0.58	0.78	0.88
	D^2B10	26.17	3.22	0.05	0.06	0.08	0.09
	D^2B20	39.82	4.86	0.06	0.11	0.15	0.17
	D^2B30	44.52	5.55	0.07	0.17	0.23	0.26
	D^2B40	47.04	5.92	0.08	0.22	0.30	0.34
	D^2B50	47.92	5.99	0.08	0.28	0.38	0.43
DenseNet	BIM4	49.21	10.68	0.04	0.59	0.81	0.89
	D^2B10	20.60	3.48	0.05	0.06	0.08	0.09
	D^2B20	41.14	5.74	0.08	0.11	0.16	0.17
	D^2B30	50.94	6.81	0.09	0.17	0.24	0.26
	D^2B40	56.33	7.45	0.10	0.23	0.33	0.35
	D^2B50	60.10	7.93	0.11	0.29	0.40	0.44
VGG19	BIM4	56.64	12.13	0.04	0.68	0.88	0.97
	D^2B10	-2.33	2.82	0.04	0.07	0.09	0.09
	D^2B20	7.60	5.58	0.08	0.13	0.17	0.19
	D^2B30	18.37	7.97	0.11	0.20	0.26	0.29
	D^2B40	29.74	9.87	0.12	0.27	0.35	0.38
	D^2B50	40.49	11.59	0.14	0.33	0.44	0.48
ResNet50	BIM4	81.91	10.81	0.04	0.62	0.83	0.90
	D^2B10	30.29	4.26	0.07	0.06	0.08	0.09
	D^2B20	58.82	6.27	0.09	0.12	0.16	0.17
	D^2B30	72.43	7.28	0.10	0.18	0.24	0.26
	D^2B40	80.06	7.88	0.11	0.24	0.32	0.35
	D^2B50	84.52	8.25	0.11	0.30	0.41	0.44

I. Additional Table for Targeted Attack

In this section, we report the target attack success rate on ResNet-50 in Table 7 (similar to Figure 5). Observe that most settings yield 100% success rate (as the model is not adversarially trained) and thus the success rates provide limited information in the comparison.

Table 7. Comparison of attack success rate of target attack on ResNet-50

	Success Rate	Human Preferance	SSIM
No Attack	0%	50%	1.00
D^2B10	93%	52%	0.99
D^2B20	100%	49%	0.97
D^2B30	100%	45%	0.96
D^2B40	100%	39%	0.96
D^2B50	100%	43%	0.95
BIM1	100%	50%	0.99
BIM2	100%	45%	0.96
BIM4	100%	27%	0.88
BIM8	100%	9%	0.73
SM500	97%	14%	0.86
SM1000	100%	7%	0.84
FS1	69%	37%	0.77
FS2	97%	12%	0.65
Latent	1%	23%	0.78
SP1	69%	27%	0.85
SP2	100%	9%	0.80
Sparse	1%	20%	0.88
HC0.1	1%	4%	0.85
HC0.8	0%	31%	0.93

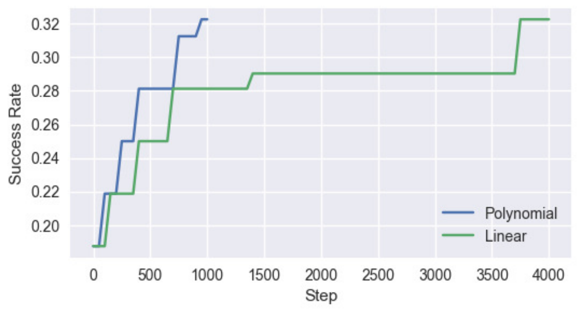
J. Comparing Two Barrier Loss Functions

We empirically set $k = 1e5$ and $b = 200$ in our polynomial barrier loss function. A large b value makes barrier loss sharp around the margin. For example, when $\mathrm{y}_i{}^{\mathrm{adv}}$ is larger than $\mathrm{y}_i{}^{\mathrm{nat}}$ and close to the upper bound high, say $\frac{\mathrm{ReLU}(\mathrm{y}_i{}^{\mathrm{adv}} - \mathrm{y}_i{}^{\mathrm{nat}})}{\mathrm{high} - \mathrm{y}_i{}^{\mathrm{nat}}} = 0.99$, the loss is $1e5 \times 0.99^{200} \approx 1e4$. Besides the polynomial barrier loss function, we have also tried a linear barrier loss defined as follows.

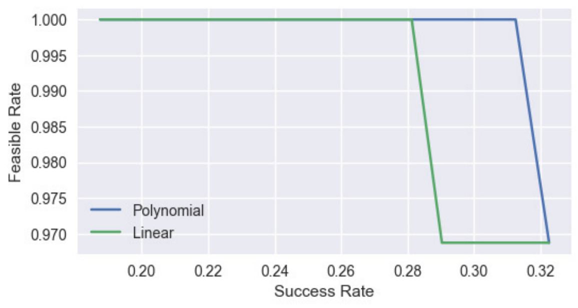
$$\begin{aligned}\mathcal{L}_i(\mathrm{y}_i{}^{\mathrm{adv}}) = k\Big\{ &\mathrm{ReLU}\left[\mathrm{y}_i{}^{\mathrm{adv}} - (b \cdot \mathrm{high} + (1-b) \cdot y_i{}^{\mathrm{nat}})\right] \\ &+ \mathrm{ReLU}\left[(b \cdot \mathrm{low} + (1-b) \cdot y_i{}^{\mathrm{nat}}) - y_i{}^{\mathrm{adv}}\right] \Big\}\end{aligned} \tag{4}$$

Intuitively, the coefficient b allows us to start applying (linear) penalty when the value approaches the boundaries. Empirically we set $k = 1e6$ and $b = 0.95$. It is worth noting that as a common drawback of barrier method, when y_i^{adv} goes beyond the feasible boundary, the optimization may encounter numerical issues. However, with a properly setting of k and b, most variables can maintain a safe distance from the boundary as we can see in Appendix F. Specifically, when y_i^{adv} is 9% of the range away from the boundary, only 3% of samples go out of the boundary. When a numerical exception happens, we restore the last feasible sample, decrease the learning rate and resume optimization.

We observe that the linearly growing penalty is not strong enough to discourage bound violation even with a large k value. Figure 9 presents the polynomial barrier loss both converges faster and constrains the optimization better than the linear loss.



(a) Optimization Speed



(b) Optimization Feasibility

Figure 9. Comparison of two barrier loss functions. Graph (a) represents the success rate (y axis) after a given step of optimization (x axis). Graph(b) represents for a given success rate (x), how many examples have internal values within their boundary (axis).

K. Comparing Different Reference Models

Our throttle plane analysis is general and can be applied to various reference models. Different models encode features differently such that they may have a different level of effectiveness. We experiment to compare the effectiveness of using VGG16 and Resnet152-Adv as the reference model. Note that VGG16 has long been used as a typical feature extractor [16], whereas Resnet152-Adv was recently reported as being useful in extracting features [15]. We conduct the same untargeted attack on Resnet50 using these two as the reference model and compare the visual quality (of generated samples) and the level of attack success. In Figure 10, we observe that the throttle planes on the two models have different characteristics. The throttle plane in VGG16 promotes more natural-looking adversarial samples while being relatively less expressive. In contrast, the throttle plane in Resnet152-Adv allows more harmful perturbation.

L. Comparing Different Deep Bounds

In this experiment, we compare the proposed deep distribution bound with the min-max bound mentioned at the beginning of Appendix A, which is an extension of the fixed pixel range used in BIM and PGD. Different from the motivation example in Section 3, here we use our proposed polynomial barrier loss to enforce both bounds instead of naive clipping. Specifically, while the minimum and the maximum of RGB values are 0 and 255, we denote the supremum and the infimum activation

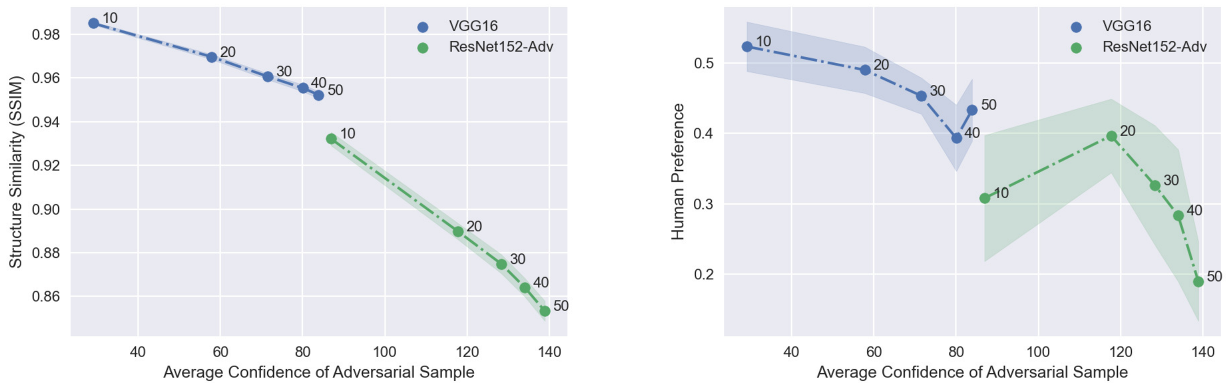


Figure 10. Attack Resnet-50 with different reference models. All the other settings are the same. The exact locations of throttle planes can be found at Appendix E.

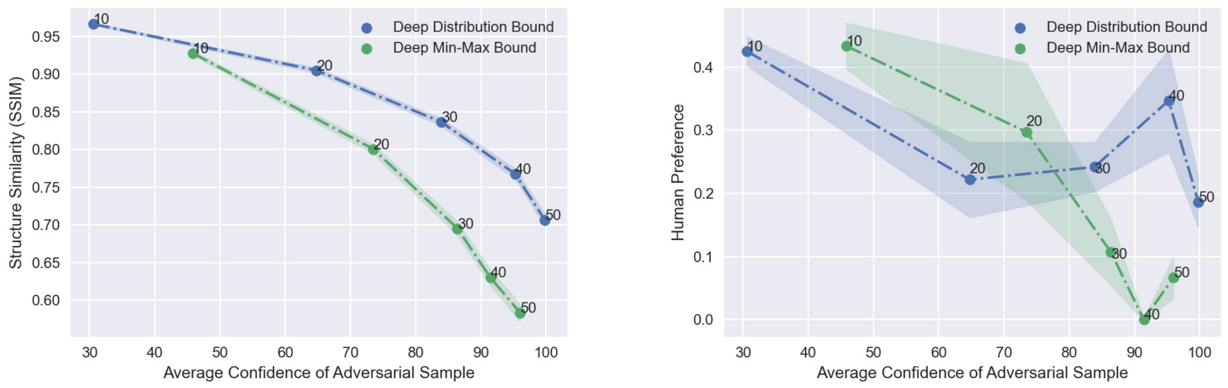


Figure 11. Comparison of deep distribution bound and deep min-max bound. We use the two bounds to attack Resnet152-Adv and report their human preference rate and SSIM Score. The annotated numbers represent the percentage of the bound relative to BIM4.

value at neuron i over a distribution support $\mathcal{S}$ as $\sup_{\mathcal{S}} y_i$ and $\inf_{\mathcal{S}} y_i$. The value bound for neuron i is thus as follows.

$$\begin{aligned} \mathrm{y}_i^{\mathrm{adv}} \in \Big[& \mathrm{y}_i^{\mathrm{nat}} - \left(\sup_{\mathcal{S}} \mathrm{y}_i - \inf_{\mathcal{S}} \mathrm{y}_i\right)\epsilon, \\ & \mathrm{y}_i^{\mathrm{nat}} + \left(\sup_{\mathcal{S}} \mathrm{y}_i - \inf_{\mathcal{S}} \mathrm{y}_i\right)\epsilon \Big] \end{aligned} \tag{5}$$

It is worth noting that, as the definition implies, the min-max bound only uses the information of the extreme values. While a deep distribution bound uses information from the cumulative distribution function, which is much more informative than just extreme values. Intuitively the rich information encoded in the deep distribution bound gives finer-grained control over the adversarial changes. In Figure 11, we compare the two bounds under the same setting. Specifically, We attack the Resnet152-Adv using the same VGG16 throttle plane. All the other settings remain the same except the bound. We measure both the human preference rate and the SSIM score. We observe that the deep distribution bound consistently outperforms the deep min-max bound on the SSIM score. In the human evaluation, for the ϵ relative to BIM4 greater than 20%, humans consistently prefer the deep distribution bound. And with a smaller scale, the deep distribution bound is slightly worse than the deep min-max bound. We conjecture this could be due to the randomness in human evaluation when the perturbation is at a small scale.

M. Evaluation Against Detection Approaches

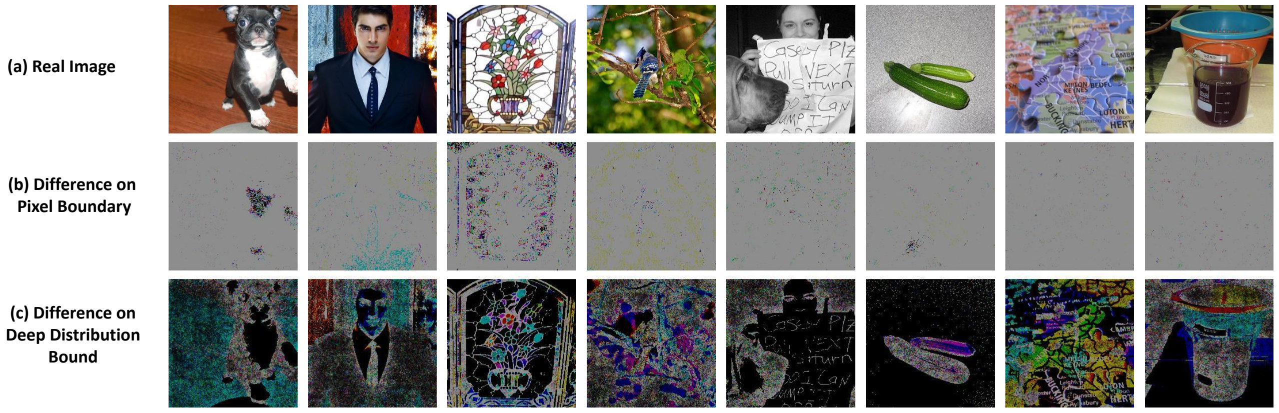
We evaluate D^{2}B against three popular detection approaches: feature squeezing [39], JPEG [8], and [13]. We generate 100 adversarial examples for each of these approaches. For feature squeezing, we use the same settings as in the original paper [39]. For JPEG [8], we compress the image with 75 quality. For [13], we use 30 examples for fine-tuning the threshold and the remaining 70 for testing. We compare with three existing attack methods: BIM [18] (BIM4), feature space attack [38] (FS1) and semantic attack [2] (SM50), whose settings were discussed in Section 4. For all the attacks, we generate untargeted adversarial examples against the adversarially trained ResNet152 and targeted examples against the naturally trained ResNet50, without knowing the existence of detection methods (i.e., not adaptive). Table 8 and Table 9 show the results. The three columns for feature squeezing are the results for different defense settings. We can observe that our untargeted attack D^2B40 has the highest human preference. In the meantime, it achieves better success rates than the other attacks for feature squeezing and JPEG; and comparable (and high) attack success rates for [13]. Recall that these attacks are on an adversarially trained model and hence the detection techniques may not be able to add much, especially for our attack that closely couples perturbation with existing features. Similarly for the targeted attacks, with a clearly better human preference rates, our attack is more or comparably persistent in the presence of detection. Note that since this is a normally trained model, some detection techniques such as feature squeezing may provide very good defense. Observe that our human preference rates in both scenarios are high, indicating that our attack may potentially conduct more aggressive perturbation to evade detection (e.g., through adaptive attack). We want to point out while these detection techniques were not designed to guard against our attack, it is still worthwhile to understand how D^2B performs in the presence of these techniques. Detection and defense (e.g., adversarial training) specific for our attack will be the future work. We have also conducted a transferbility study of our attack. We observe that D^2B has a comparable/slightly-better transferbility than other attacks. Details can be found in Appendix O.

Table 8. Untargeted attacks on the adversarially trained ResNet152-Adv and detection, with Pref. meaning human preference

Attack	Feature Squeezing			JPEG	[13]	Pref.
	2x2	11-3-4	5-bit			
BIM4	50/100	55/100	57/100	57/100	48/70	30%
FS1	46/100	46/100	46/100	48/100	48/70	35%
SM50	51/100	51/100	60/100	60/100	44/70	29%
SP1	52/100	41/100	55/100	46/100	48/70	23%
Latent	24/100	23/100	24/100	24/100	29/70	23%
HC 0.1	27/100	37/100	24/100	32/100	**59/70**	4%
D^2B40	**79/100**	**97/100**	**99/100**	**64/100**	46/70	**41%**

Table 9. Targeted attacks on ResNet50 and detection, with Pref. human preference

Attack	Feature Squeezing			JPEG	[13]	Pref.
	2x2	11-3-4	5-bit			
BIM4	30/100	**91/100**	100/100	51/100	46/70	27%
FS1	6/100	25/100	53/100	24/100	**65/70**	36%
SM50	0/100	4/100	10/100	1/100	49/70	21%
SM500	5/100	72/100	95/100	46/100	48/70	14%
SP1	2/100	39/100	53/100	4/100	40/70	27%
Latent	1/100	0/100	1/100	0/100	33/70	23%
Sparse	0/100	0/100	2/100	0/100	27/70	20%
HC 0.1	0/100	1/100	1/100	1/100	57/70	15%
D^2B100	**37/100**	88/100	**100/100**	**57/100**	60/70	**50%**


(a) Real Image
(b) Difference on Pixel Boundary
(c) Difference on Deep Distribution Bound
Figure 12. Differential Analysis. The first row denotes benign examples. The following two rows are pixel-wise differences between benign inputs and their corresponding adversarial versions. The second row is from BIM and the third row is from ours. Both have very small bounds. The differences are scaled up by a factor of 2e6 for the two respective rows for better display.

N. Differential Analysis - Understanding the Essence of D^2B

In this experiment, we give D^2B a very small internal bound, i.e., 5e-4% quantile change on the throttle plane of ResNet50. As such, the pixels changes enabled by the bound indeed are so small that they essentially denote the input gradients induced by our method. We also conduct a similar experiment for the pixel space BIM method (with approximately the same $\ell_\infty = 2.77e{-}05 \times 255$) for reference. The results are presented in Figure 12, where the pixel differences between adversarial example and their original versions are presented. We observe that the "gradients" in pixel space attacks are more prevalent and uniform, whereas the "gradients" in our attack closely couple with the existing content features. Note that the experiment cannot be done on feature attack and semantic attack as there is no way to enforce a small bound for those attacks.

O. Transferability of Generated Adversarial Examples

In this section, we study the transferability of adversarial examples generated by different attacks. We launch untargeted attacks on ResNet152-Adv and targeted attacks on ResNet50. We use the same attack success criterion as in Section 4: (1) targeted adversarial examples should induce the same target label when transferred to a second model; (2) untargeted adversarial examples should exclude the true label from appearing in the top-5 predicted labels when transferred. We test generated adversarial samples on 4 different models, including ResNet50' (with the same structure but different parameters as the previously used ResNet50), VGG19, MobileNet and DenseNet. The results can be found in Table 10 and Table 11. We observe that our untargeted attack has higher human preference than other attacks with comparable transferability. For targeted attacks, our method outperforms other methods in both transferability and human preference. Note that transferring targeted attack is a challenging task and requires specific approaches (e.g., ensemble) to improve the transferability.

Table 10. Transferability of untargeted attacks. Adversarial examples are generated on ResNet152-Adv model.

Attack	Transfer to				Human Preference
	ResNet50'	VGG19	DenseNet121	MobileNet-V1	
PGD-4	50/100	60/100	46/100	61/100	30%
FS1	43/100	51/100	39/100	52/100	35%
HC0.1	22/100	23/100	29/100	24/100	4%
SM50	49/100	57/100	48/100	47/100	29%
SP2	59/100	63/100	**60/100**	64/100	16%
LAT	36/100	43/100	35/100	38/100	28%
D^2B40	55/100	56/100	47/100	60/100	**41**%
D^2B50	**67/100**	**63/100**	57/100	**70/100**	20%

Table 11. Transferability of targeted attacks. Adversarial examples are generated on ResNet50 model.

Attack	Transfer to				Human Preference
	ResNet50'	VGG19	DenseNet121	MobileNet-V1	
PGD-4	37/100	0/100	0/100	1/100	27%
FS1	17/100	**2/100**	0/100	**1/100**	36%
HC0.1	1/100	0/100	0/100	0/100	15%
Sparse	0/100	0/100	0/100	0/100	20%
SM50	2/100	0/100	0/100	0/100	21%
SM500	32/100	0/100	1/100	0/100	14%
SP2	44/100	0/100	**2/100**	**1/100**	9%
LAT	1/100	0/100	0/100	0/100	23%
D^2B100	**44**/100	0/100	0/100	0/100	**50**%

P. Ethics Consideration

We have obtained IRB approval on our human study. We will release it upon publication. This paper demonstrates a new aspect that a deep learning model can be attacked. It hence provides strong motivation and critical insights for the community to develop stronger defense techniques and make deep learning applications more trustable.

Q. Adversarial Examples of Different Scales

We show the generated adversarial examples using D^2B with different settings in Figure 13 and Figure 14. Figure 13 demonstrates samples of a targeted attack on ResNet50 and Figure 14 an untargeted attack on ResNet152-Adv. We can observe that most of our adversarial examples are indistinguishable from real images (top row). For few cases such as the 3rd column in the last row (with large quantile change), we observe the presence of a repeating pattern. We speculate this is because the attack was only applied to the first a few representative throttle planes, which may not be as abstract as other deeper layers. This effect can be alleviated by including more throttle planes when launching the attack.

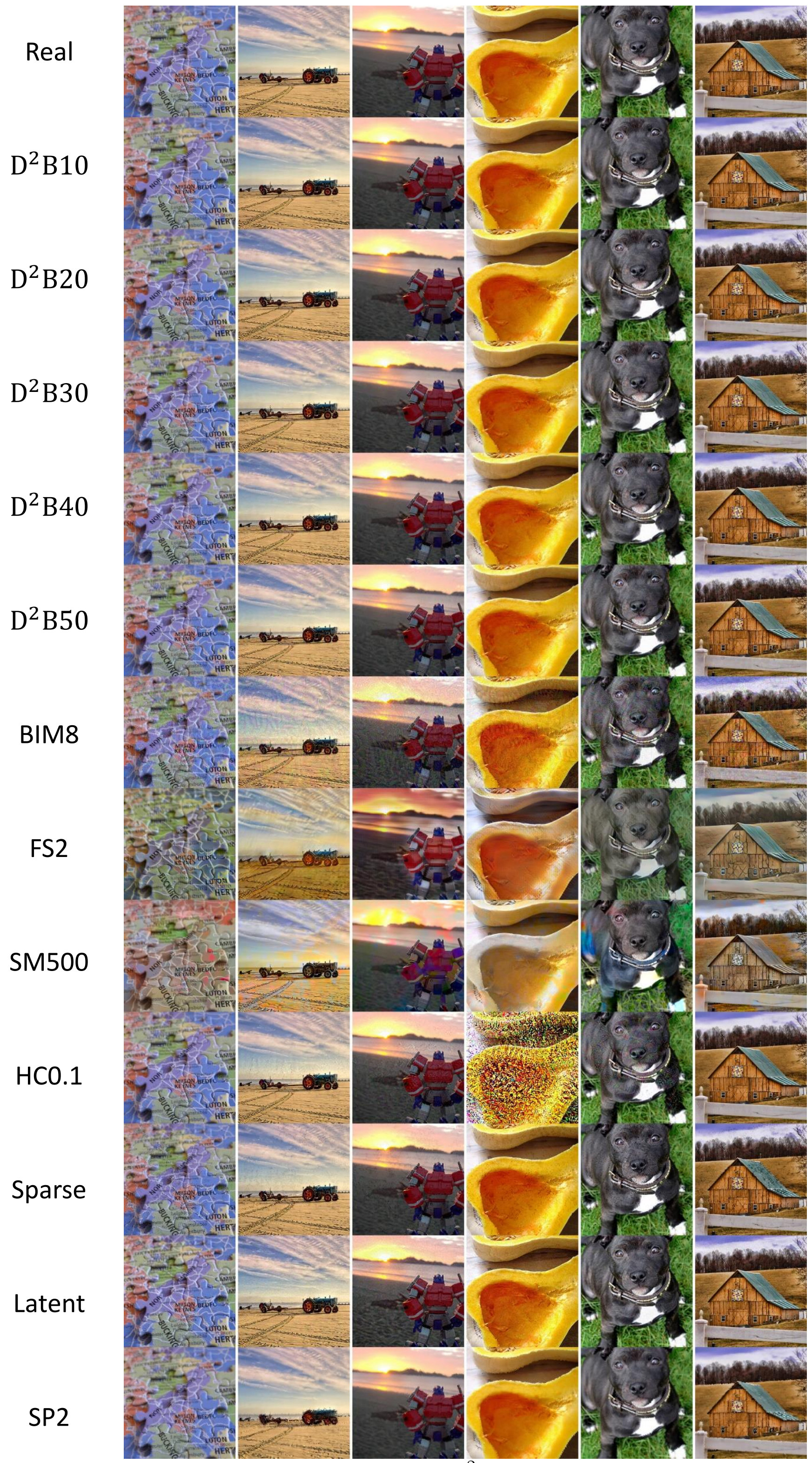


Figure 13. Adversarial samples on ResNet50 from D^{2}B of different scales and other Attacks

R. Typical Throttle Plane Distributions

We present some typical distributions from selected throttle planes in Figure 15 and Figure 16. Figure 15 shows distribution density graphs of $4 \times 4 \times 1$ slice (width$\times$height$\times$channel) of selected throttle planes for VGG16, and Figure 16 for ResNet152-

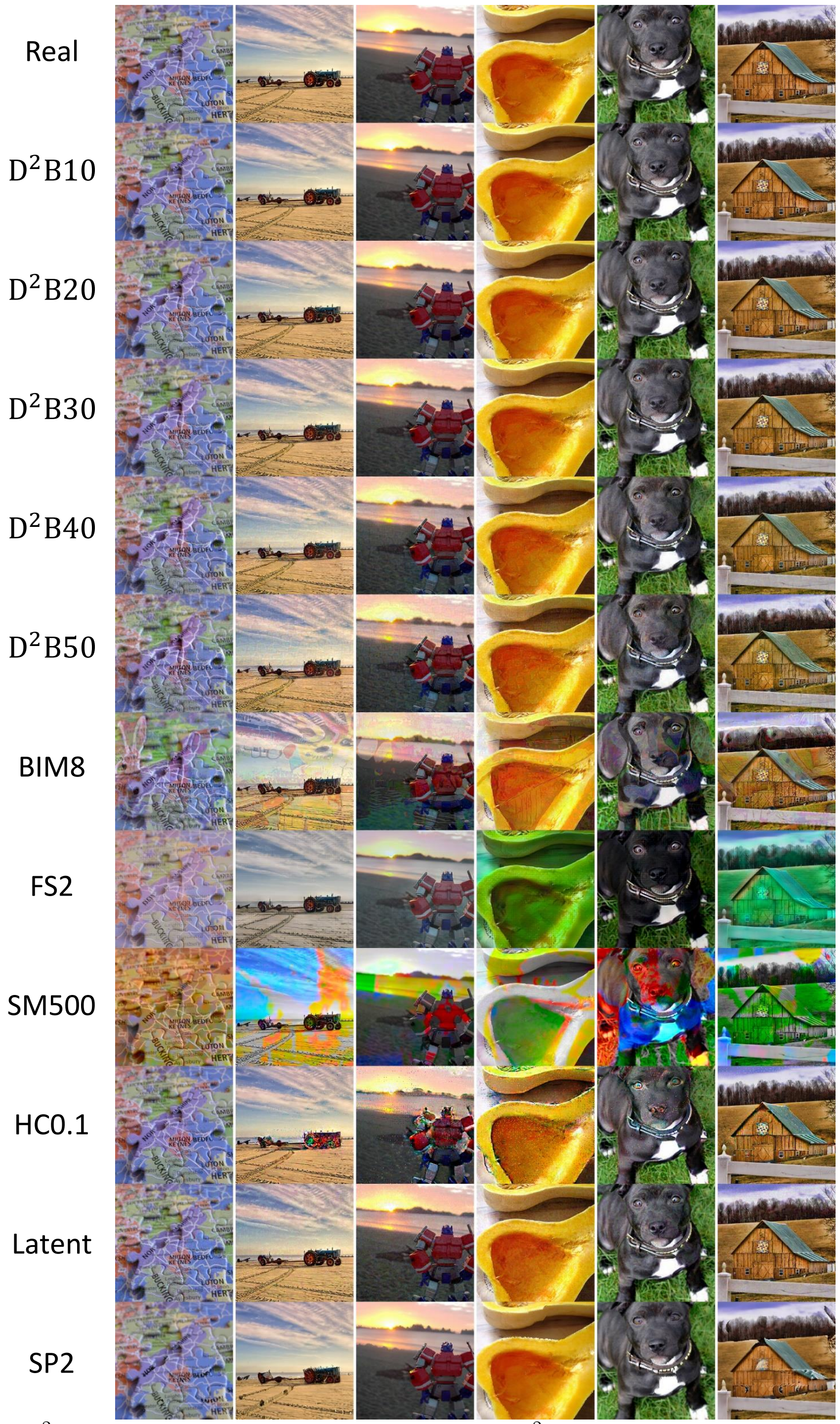
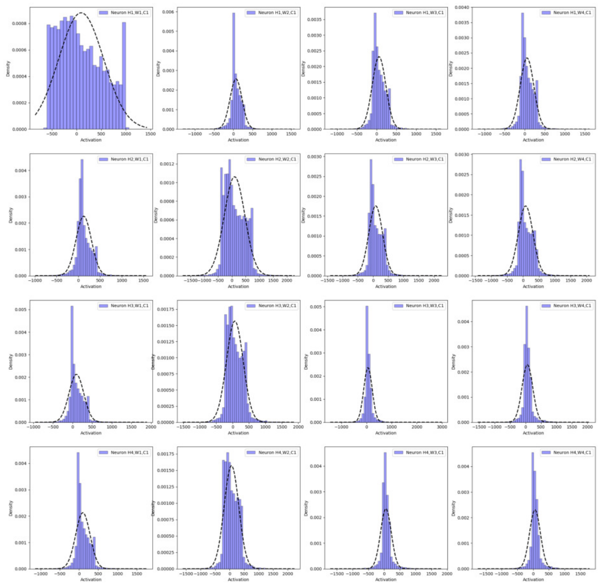
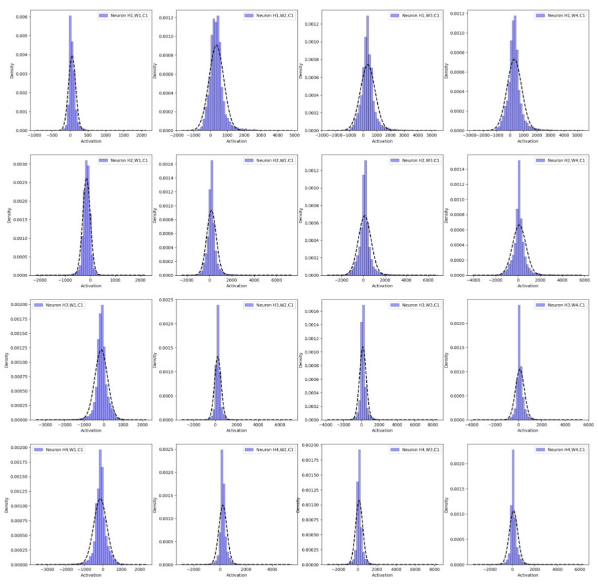


Figure 14. D^2B Adversarial samples on ResNet152-Adv from D^2B of different scales and other Attacks

Adv. We observe that the planes from VGG16 resemble normal distributions more than the planes from ResNet152-Adv (e.g., Figure 15c and Figure 16c). This might explain why adversarial examples from the VGG16 reference model are relatively more nature. We also observe that the first few rows and columns in Figure 15a and Figure 16a look less like a normal distribution. This is due to the existence of zero padding in those layers. The padding operation makes the first a few neurons around the border of a channel distinct from the inner neurons.


(a) Throttle Plane 1: After VGG16 conv1_2


(b) Throttle Plane 2: After VGG16 conv2_2

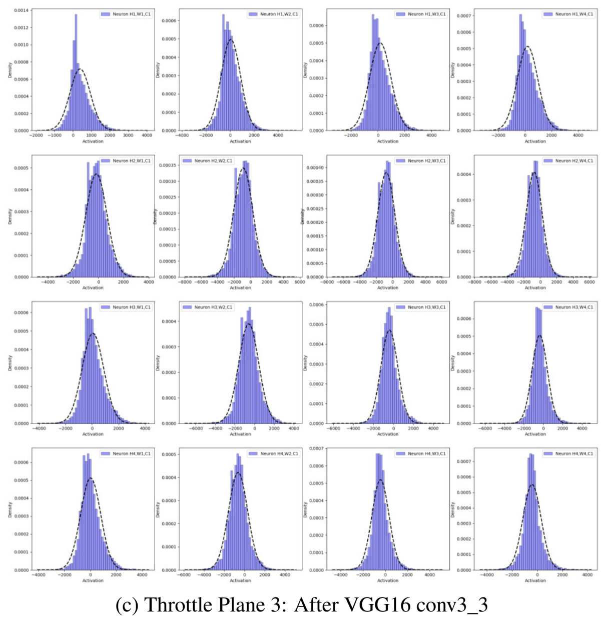
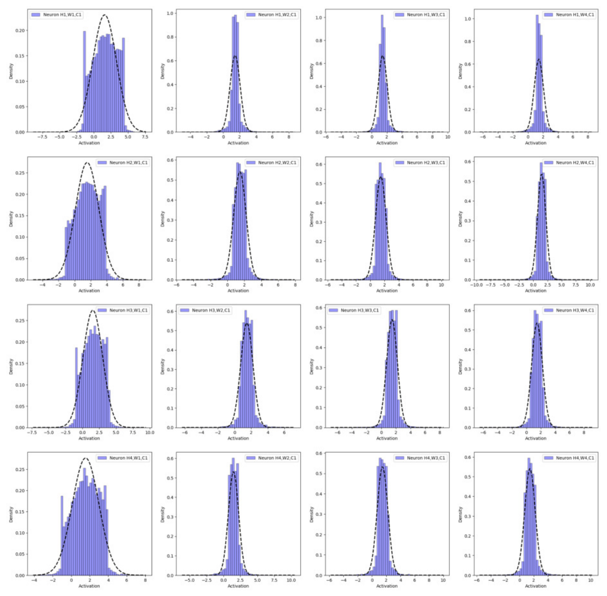
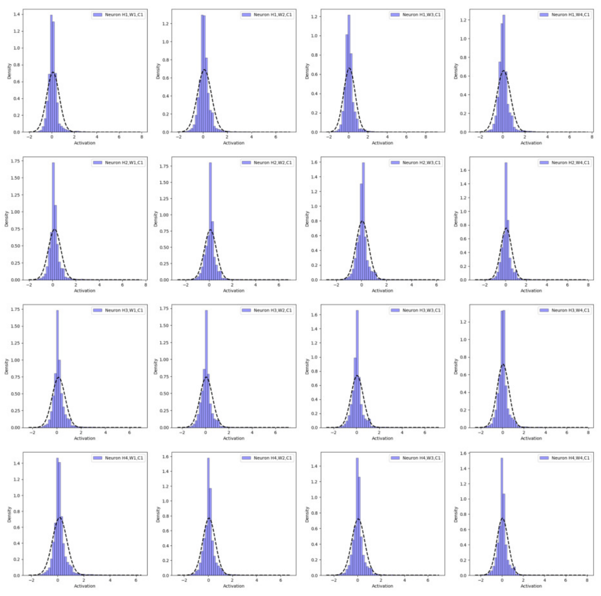

(c) Throttle Plane 3: After VGG16 conv3_3

Figure 15. Typical Distributions of Selected Throttle Planes from VGG16


(a) Throttle Plane 1: After the First Convolution


(b) Throttle Plane 2: The last Layer of Group 1 Before ReLU

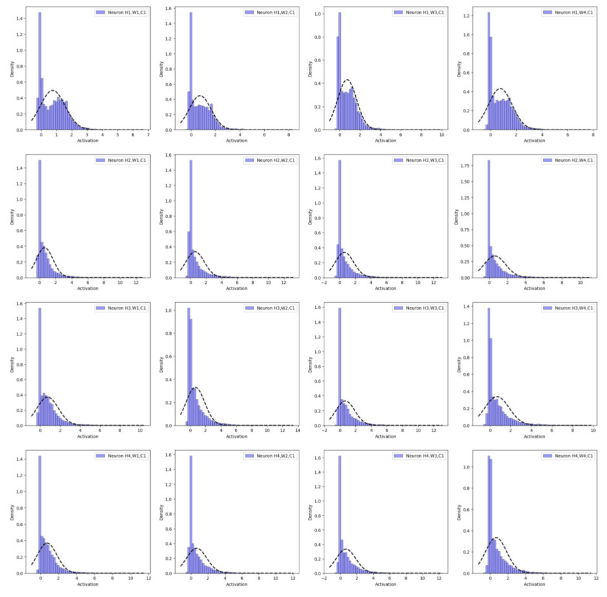

(c) Throttle Plane 3: The last Layer of Group 2 Before ReLU

Figure 16. Typical Distributions of Selected Throttle Planes from ResNet152-Adv