Supplementary Material for: SymDNN: Simple & Effective Adversarial Robustness for Embedded Systems

Swarnava Dey Pallab Dasgupta Partha P Chakrabarti
Indian Institute of Technology Kharagpur, Kharagpur, India - 721302
swarnava.dey@kgpian.iitkgp.ac.in, {pallab, ppchak}@cse.iitkgp.ac.in

The supplementary material is organized as follows:

- Sec. 1 presents the detailed results of MNIST and ImageNet dataset, as referred from the main paper. Further details and visualization of CIFAR-10 experiments on adversarial robustness are also presented here.
- Sec. 2 presents the, computation load for clustering and some ablation studies on the most important hyperparameters of SymDNN.
- Sec. 3 presents the details about all the building blocks used by us in this work with their licensing details.

1. Extended Results and Visualizations

This section presents more elaborate results supporting SymDNN's inference mechanism over multiple datasets.

For the experiments in the current section, a model white-box / defense black-box setting is used, where SymDNN works best (see Sec. 5.2 of main paper). The gradient obfuscation [3] problem is mostly manifested in a complete white-box setting. We use 100 iterations for finding adversarial examples in the gradient based attacks, and 10 iteration for the Expectation over Transformation (EoT) [4]. These are standard iteration values used in the literature, and default in TorchAttacks [17]. Using very high values for iterations may imply that an adversary has access to an enormous amount of computing power and significant time at hand for crafting adversarial examples [22].

It is important to note that with increase in the strength ϵ of an attack, the attacked images begin to show visually identifiable distortions. Inherent robustness to adversarial attacks is essential within a bound, where the changes are visually imperceptible. The visualizations in the current section highlight this aspect. **This part is referred in Sec. 5.2 of the main paper.**

1.1. Adversarial Robustness of MNIST

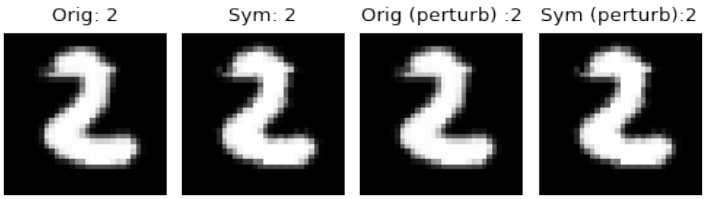
Tab. 1 provides extended results on the attacks on the MNIST dataset and the corresponding LeNet-5 model [9]. For most of the attacks, when the attack is *effective*, namely the attacked image is misclassified but no visually identifiable patterns are present, SymDNN successfully defends those. This happens when the maximum changes per pixel is limited to $\epsilon = 8/255$. For higher attack strengths SymDNN's inference accuracy drops, albeit remaining much higher than the standard non-symbolic inference accuracy. Fig. 1 and Fig. 2 provide visual illustrations.

Table 1. MNIST symbolic inference accuracy after attack. Results are reported on randomly selected 2000 images from MNIST test set. '^{s}' indicates SymDNN inference that uses a 64 symbol codebook.

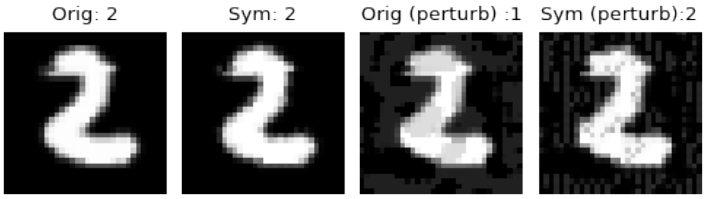
Attacks	Strength	Acc.	Acc.^{s}
Auto Attack [8]	ϵ = 8/255	95.60	98.05
	ϵ = 16/255	85.50	93.65
	ϵ = 32/255	22.35	68.60
EOTPGD [33]	ϵ = 8/255	95.65	98.05
	ϵ = 16/255	85.90	93.65
	ϵ = 32/255	29.80	69.75
DIFGSM [30]	ϵ = 8/255	96.45	98.05
	ϵ = 16/255	86.95	94.00
	ϵ = 32/255	31.00	69.20
RFGSM [29]	ϵ = 8/255	95.60	98.05
	ϵ = 16/255	85.95	93.65
	ϵ = 32/255	29.09	68.70
MIFGSM [12]	ϵ = 8/255	95.60	98.05
	ϵ = 16/255	85.95	93.60
	ϵ = 32/255	29.60	69.10
PGD [19]	ϵ = 8/255	95.75	98.05
	ϵ = 16/255	86.15	93.65
	ϵ = 32/255	29.60	69.10
OnePixel [26]	pixels = 1	98.60	98.60
	pixels = 5	94.15	94.60
	pixels = 10	94.55	94.65

1.2. Adversarial Robustness of CIFAR-10

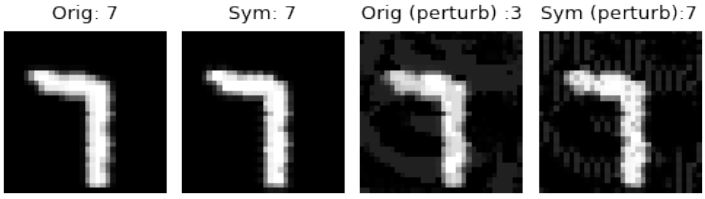
We have reported the efficacy of SymDNN on the CIFAR-10 dataset and the corresponding model [32] in the main paper. Some of the attacks (AutoAttack [8],



(a) Autoattack with ϵ = 16/255: there is neither identifiable visual distortion, nor misclassification on attack



(b) Autoattack with ϵ = 32/255: pattern for 2 faded out at places and is identifiable. SymDNN provides resistance and survives the attack.



(c) DIFGSM with ϵ = 32/255: pattern for 7 faded out at places and is identifiable. SymDNN provides resistance and survives the attack.

Figure 1. A visual comparison. For each set of 3 images Column 1 (Orig:) shows the clean original image, Column 2 (Sym:) shows the image reconstructed by SymDNN, Column 3 (orig perturb:) shows the original image after attack and Column 4 (Sym perturb:) shows the symbolically reconstructed image after attack.
For $\epsilon = 16/255$, the adversarial attack is effective on the original image, but fails to cause misclassification in SymDNN. At $\epsilon > 16/255$, we observe that SymDNN misclassified, but the attacks are not adversarially *effective*, that is, the attacked image has visually identifiable changes. High ϵ is used here to highlight the patch replacement by representative symbols in SymDNN.

DIFGSM [30], RFGSM [29], APGDT [8], and variants of APGD [8]), using a strength greater than ϵ=2/255 (L_∞) results in visually identifiable changes on CIFAR-10 images. Fig. 3 illustrates one such example where AutoAttack [8] is applied at different strengths. It may be noted that both standard and SymDNN robust accuracy drops heavily in these cases. This is consistent with the observation in [1], that k-centroid clustering is inherently robust to small set of changes, if such a clustering is well separated. We find most of our centroids are well separated in terms of the dissimilarity measure used in [16]. As it is difficult to visualize a 2048×2048 matrix here, we present a heatmap of cluster distances in Fig. 4 for MNIST similarity index, with 32 centroid patches.

We report the results on CIFAR-10 using L2 distance measure in Tab. 2.

To compare the SymDNN inference and the standard non-symbolic inference in terms of numbers, we consider the following outcomes:

Table 2. CIFAR-10 symbolic inference accuracy after attack using L_2 distance. Results are reported on randomly selected 2000 images from CIFAR-10 test set. '^{s}' indicates SymDNN inference that uses a 2048 symbol codebook. MSR^s is the Multi-Sym Randomized inference mechanism defined in the Sec. 4.2.2 of the main paper.

Attacks	Strength	Acc.	Acc.^{s}
Auto Attack [8]	ϵ = 8/255	57	81
Square [2]	ϵ = 8/255	79	86
FAB [7]	ϵ = 8/255	58	85
APGDT [8]	ϵ = 8/255	57	82
APGD(CE) [8]	ϵ = 8/255	57	81
APGD(DLR) [8]	ϵ = 8/255	58	82

1. *Clean non-symbolic.* The original image is correctly classified.

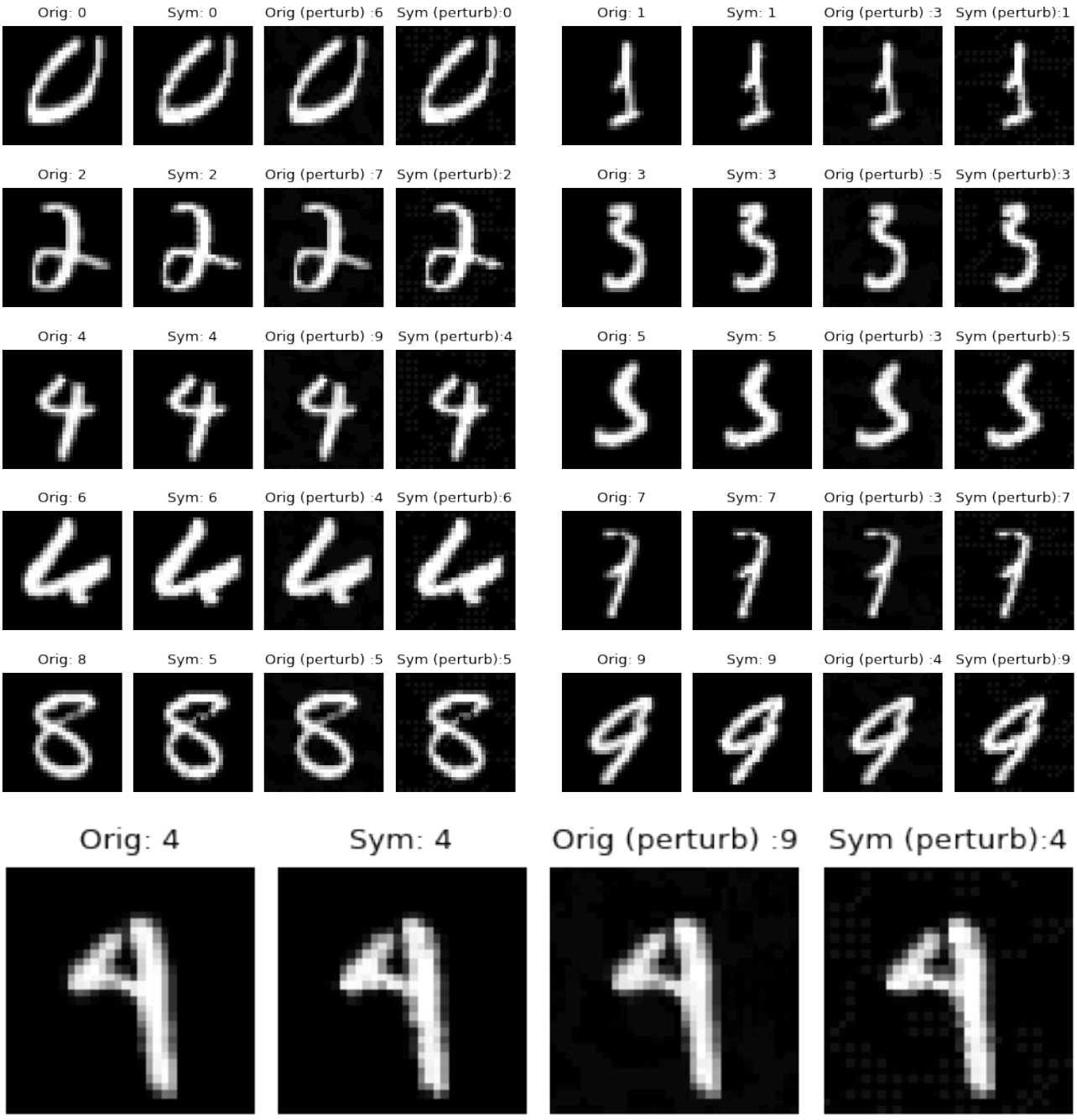
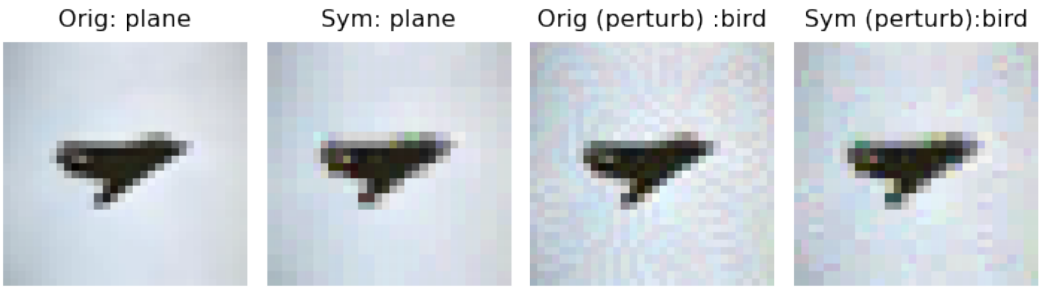
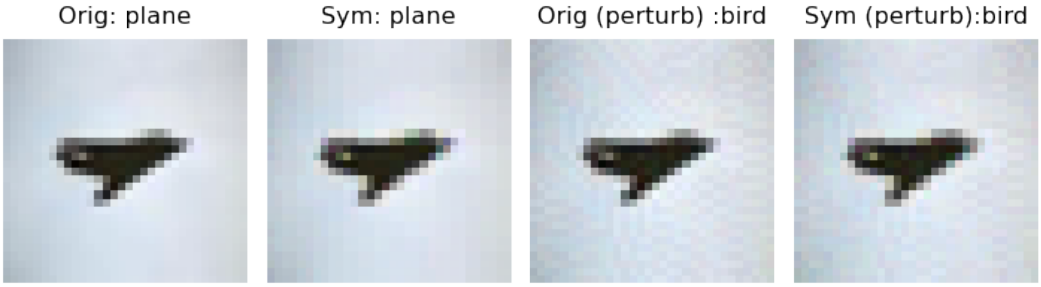


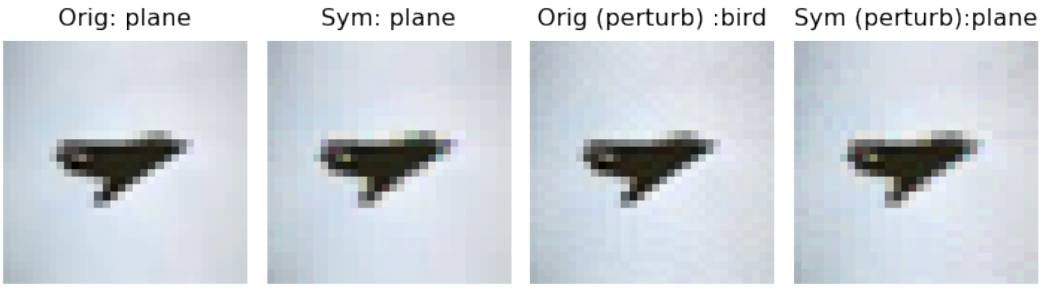
Figure 2. Efficacy of SymDNN on MNIST digits under PGD attack: for each set of 4 images the first column (Orig:) is the clean original image, second column (Sym:) is the symbolically abstracted clean image, third column (orig perturb:) is the image after attack and the fourth column (Sym perturb:) is the symbolically abstracted image after attack.
At $\epsilon = 8/255$, the attack is very *effective*, i.e., the attacked image is misclassified but there are no visual changes in the image (orig perturb). SymDNN not only results in a correct classification in most cases (except for 8), but keeps a clear mark in the symbolically abstracted perturbed image (Sym perturb), depicting that some undesirable changes have happened to the image. The reader is requested to zoom in for finding the pattern in the images labeled *Sym (perturb)*. One such image is presented in the last row.



(a) Visible changes with ϵ=8/255



(b) Changes are still identifiable with ϵ=4/255



(c) At ϵ=2/255 changes are not visually identifiable

Figure 3. Visually identifiable changes in some CIFAR-10 images under Autoattack: For each set of 4 images the first column (Orig:) shows the clean original image, the second column (Sym:) shows the symbolically reconstructed clean image, the third column (orig perturb:) shows the image after attack, and the fourth column (Sym perturb:) shows the symbolically reconstructed image after attack. Autoattack results in visually identifiable patterns on the images for changes per pixel greater than 2/255

2. *Clean symbolic.* The SymDNN reconstructed image is correctly classified.

3. *Robust non-symbolic (post attack).* The original image is correctly classified even after attack.

4. *Robust symbolic (post attack).* The SymDNN reconstructed image is correctly classified even after attack.

Based on these outcomes, we partition the results into the following cases:

Case-1 This refers to the instances where all the above correctly infer the ground truth label.

Case-2 This refers to the rare cases where only the SymDNN reconstructed image is misclassified after attack.

Case-3 This refers to the instances where the clean images were correctly classified, the SymDNN reconstructed images were correctly classified after attack, but the original images were misclassified after the attack. These instances are most relevant for demonstrating the benefit of SymDNN.

Case-4 These are instances where both the original and SymDNN reconstructed images are misclassified after the attack. As discussed earlier, many of these cases show significant visual distortions after attack and thereby rendering the attacks ineffective from an

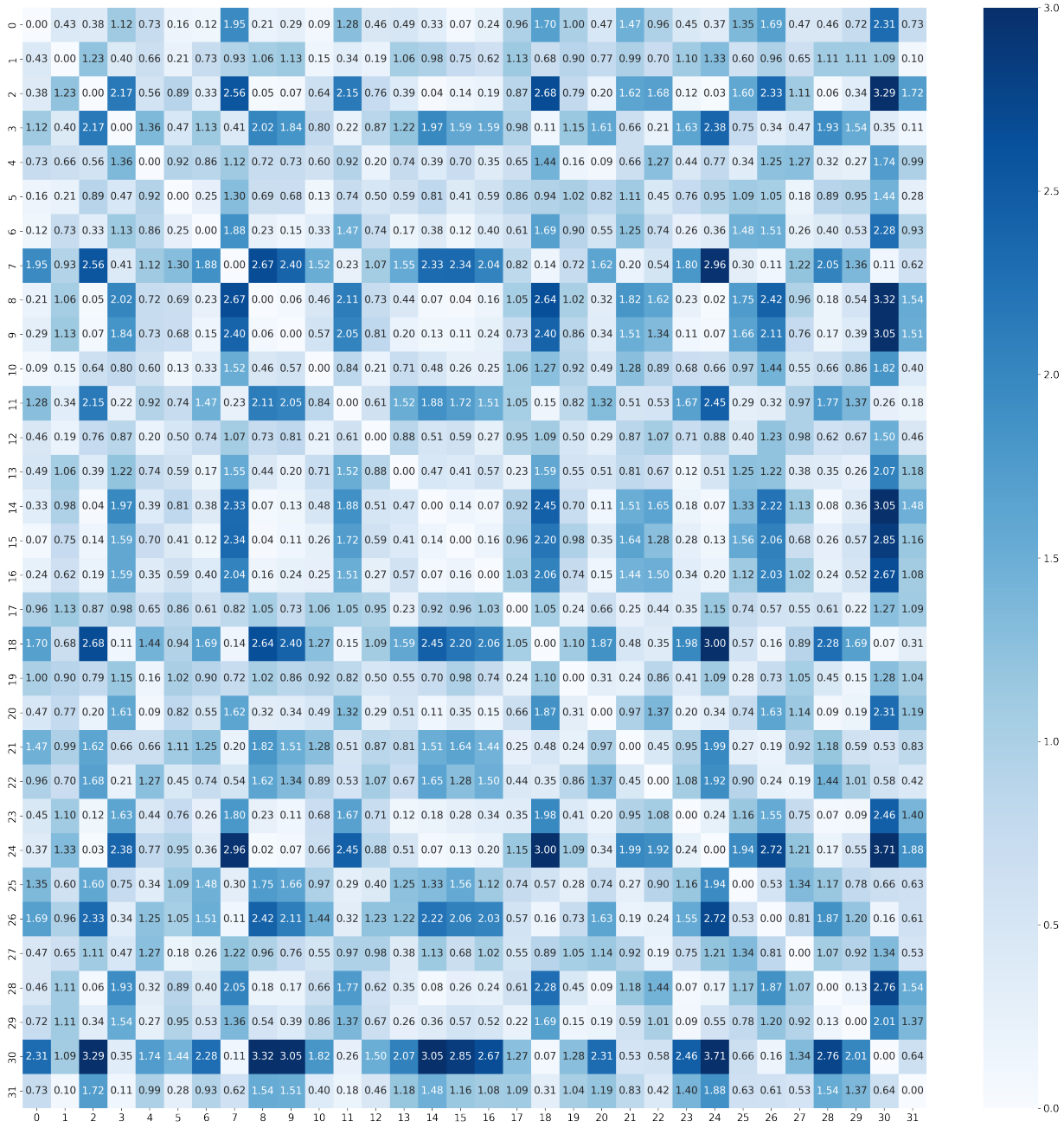


Figure 4. A heatmap of cluster centroid separation for MNIST clustering model with 32 centroids .

adversarial point of view.

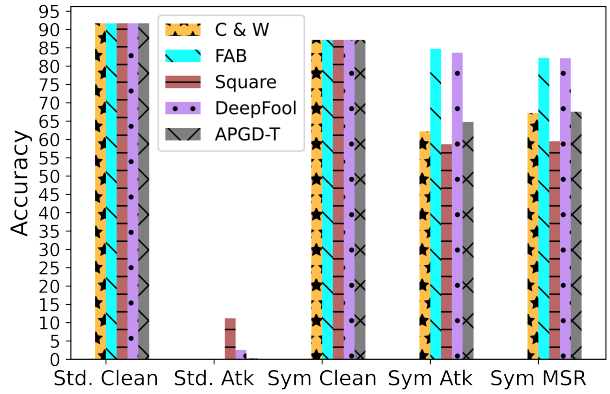
Case-5 These are instances where the SymDNN reconstructed images are misclassified without any attack.

Case-6 These are instances where the original clean images are misclassified.

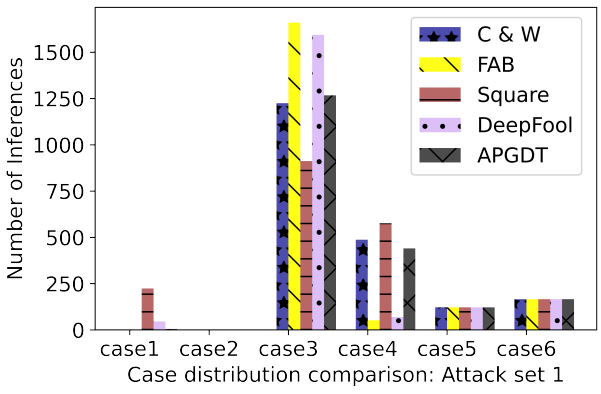
The case distributions and accuracy of SymDNN and non-symbolic inference against some adversarial attacks are shown in Fig. 5. We present these results in two different sets:

1. Set 1 (Fig. 5b) illustrates the attacks against which SymDNN performs very well, and

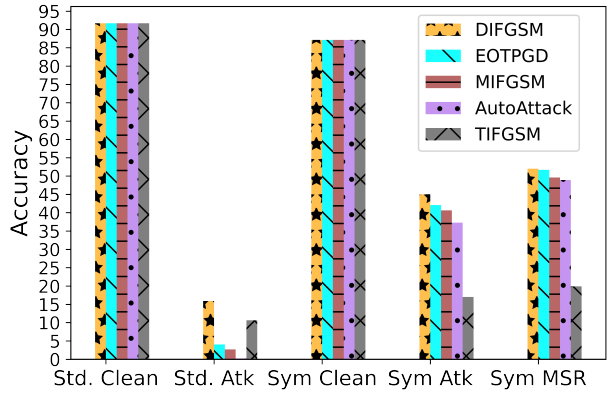
2. Set 2 (Fig. 5d)the attack for which the robust accuracy



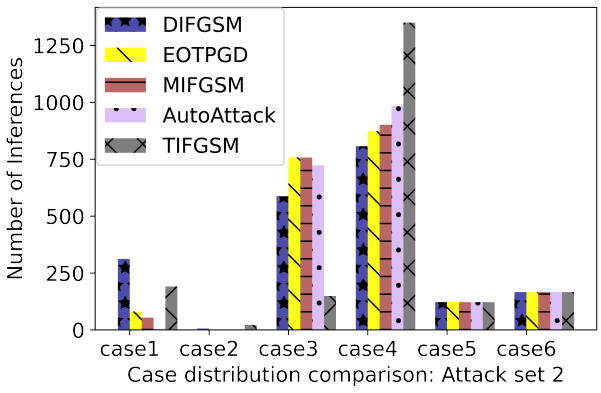
(a) Accuracy for various attacks



(b) Distribution of cases for the attacks



(c) Accuracy for the other set of attacks



(d) Distribution of cases for the other set of attacks

Figure 5. The distribution of inference outcomes for the 6 different cases defined in Sec. 1.2. We segregate the adversarial attacks into two different sets, based on their effect on robust accuracy. Figures (a) and (b) illustrate the accuracy and distribution of cases respectively, for the attacks against which SymDNN results in high robust accuracy gain. Figures (c) and (d) illustrate the accuracies and distribution of cases respectively, for the attacks against which SymDNN results in relatively lower robust accuracy gain.

gain is limited.

The attacks in *Set 2*, mostly generate adversarial examples with improved transferability. The perturbations in such examples are relatively higher as these attacks try to maximize the confidence instead of minimizing the distance [10]. We have already discussed this issue in Sec. 5.2 & Sec. 5.4 of the main paper, and we believe that a more strongly separable clustering may help SymDNN to thwart adversarial examples with larger perturbations.

1.3. ImageNet Adversarial Robustness.

We perform ImageNet adversarial tests with FoolBox [23]. Tab. 3 presents the results of these preliminary experiments.

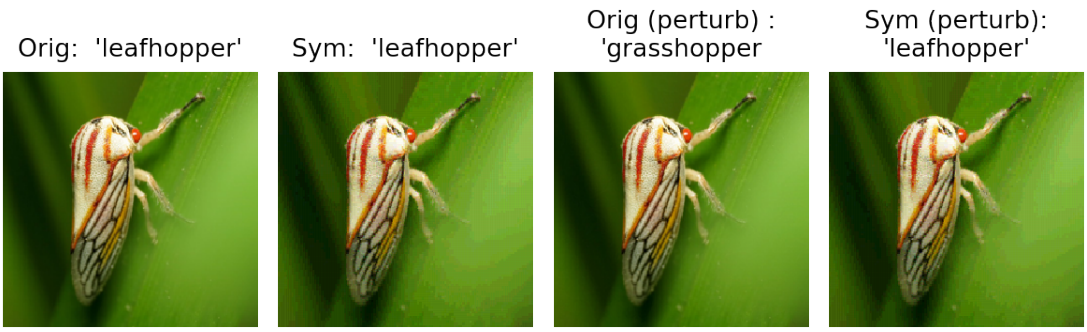
In this section we report some interesting observations regarding SymDNN inference on Imagenet images. Fig. 6 illustrates examples where we find that SymDNN preserves inference against adversarial attacks, however the symbolically reconstructed images contain spurious patterns that are not present in the original images. Fig. 7 illustrates the different cases of symbolic and non-symbolic inference on ImageNet.

Table 3. ImageNet robust symbolic Foolbox.

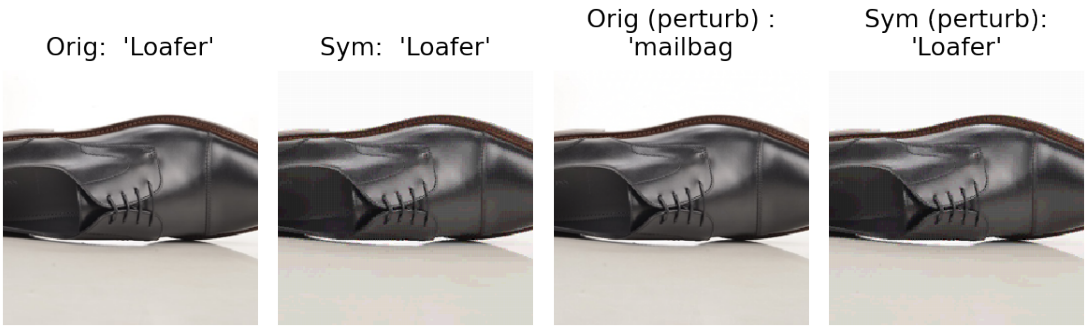
Attack	Acc.	Acc.^{s}
FGSM [14, 27]	0	62.5
Adam-PGD [23]	0	68.5
CW [5]	0	68.5
PGD [19]	0	68.75
DeepFoolL_2 [20]	0	31.25

1.4. Illustrative Example on Compaction.

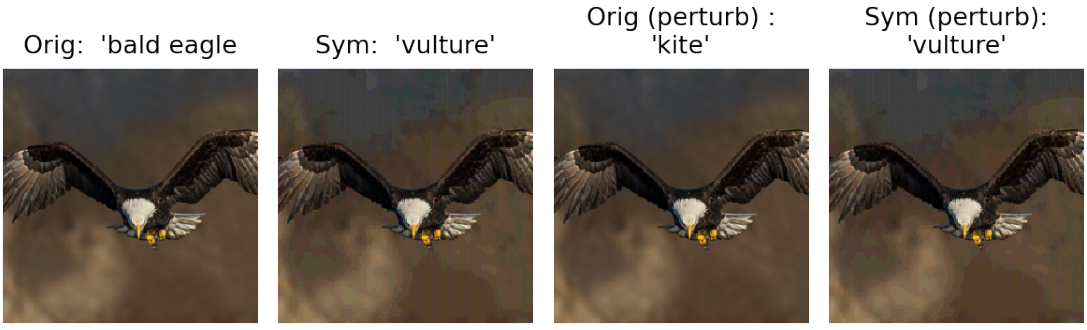
We provide an illustration below for data reduction results, as presented in the main paper. The first part of this



(a) The symbolic image showing a pattern not present in the original image: leafhopper under C & W attack



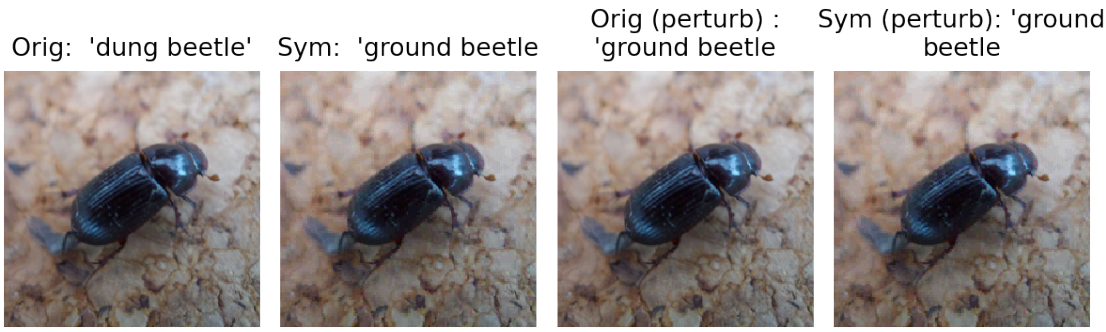
(b) The symbolic image of loafer showing a pattern inside the shadow region, absent in the original image: PGD



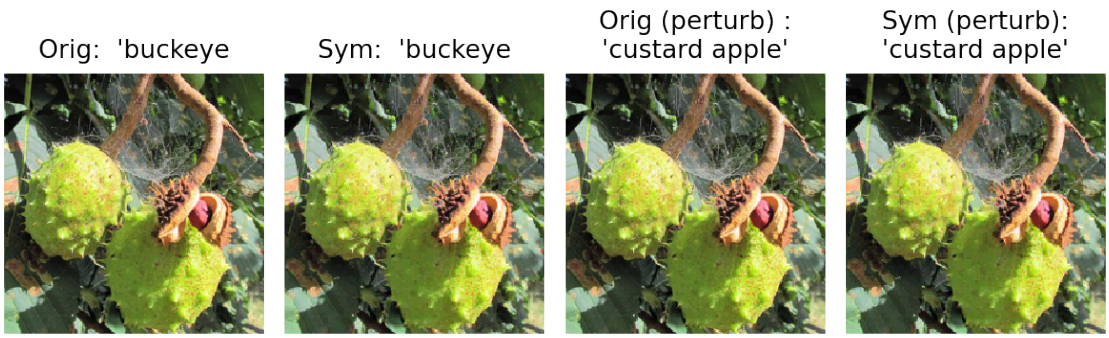
(c) The symbolic image of kite showing a pattern in the sky, not present in the original image: PGD

(d) The symbolic image showing marks in the upper left and right, absent in the original image: PGD

Figure 6. Some symbolically abstracted images have visual distortion, although the symbolic inference is correct. This shows that SymDNN inference may not be used for visual reconstruction, however it is inference preserving from a CNN perspective. For each set of 4 images the first column (Orig:) is the clean original image, second column (Sym:) is the symbolically abstracted clean image, third column (orig perturb:) is the image after attack and the fourth column (Sym perturb:) is the symbolically abstracted image after attack.



(a) Both symbolic and non-symbolic inferences are correct, no effect of APGD attack



(b) Both symbolic and non-symbolic inferences are affected by attack

(c) Only non-symbolic inference is affected by attack

(d) Only non-symbolic inference is affected by attack

Figure 7. Visualization of different inference outcomes of ImageNet images under different adversarial attacks: for each set of 4 images the first column (Orig:) is the clean original image, second column (Sym:) is the symbolically abstracted clean image, third column (orig perturb:) is the image after attack and the fourth column (Sym perturb:) is the symbolically abstracted image after attack.

reduction, Δ^a, comes from the symbolic abstraction. We can calculate this guaranteed compaction by finding out the reduction for representing each patch. For a 2×2 patch we need $2 \times 2 \times 8 = 32$ bits, assuming 8 bit per pixel representation. For representing a 2×2 patch we use one symbol. Considering a 512 symbol codebook, this requires 9 bits to encode one symbol. This results in the Δ^a compaction of 71.8%.

To employ entropy encoding over this compaction, we assume a distribution similar to the 10000 image testset of CIFAR-10. We apply Huffman encoding and obtain a distribution illustrated in Fig. 8. We find that using a 512 symbol codebook results in extra 22.67% compaction. The combined compaction of 78.3% is reported as the Δ^e in the main paper. This compaction can be useful when the image has to be communicated, say, from an edge device to an edge server in a computation offloading scenario [11].

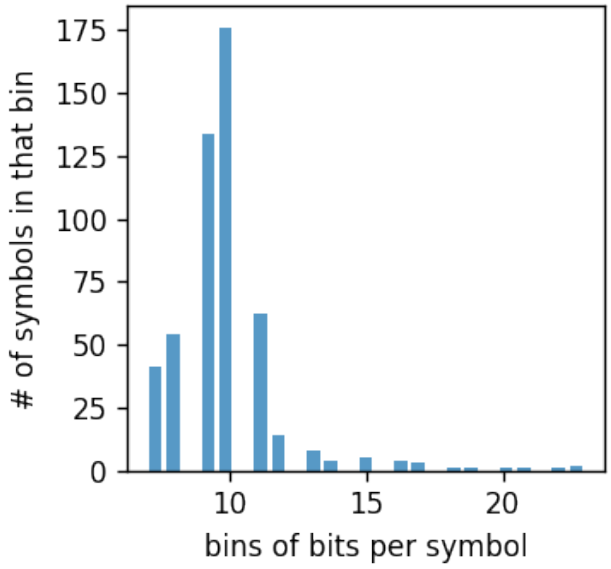


Figure 8. Huffman encoding of symbols: Variable number of bits assigned to encode the symbols based on their frequency of occurrence in the CIFAR-10 testset.

2. Patch Clustering

All the experiments in this section are conducted on a 32 core Intel(R) Xeon(R) Silver 4108 CPU 1.80GHz workstation with 128 Gibibytes of RAM, and NVIDIA P1000 GPU (4 Gigabytes memory). We use the GPU for unsupervised clustering of ImageNet patches.

2.1. Clustering Computation Load

This part supplements Sec. 4.1.1 and Sec. 5.4 of the main paper.

Fig. 9 illustrates the increasing time required for clustering patches from the CIFAR-10 dataset, as the number of clusters are increased. The patch extraction overhead becomes comparatively less with increasing number of centroids. It may be noted that here we extract patches from each channel of the image, collate the patches from all the images, and pass these patches to FAISS *k-means* for building the index at one go.

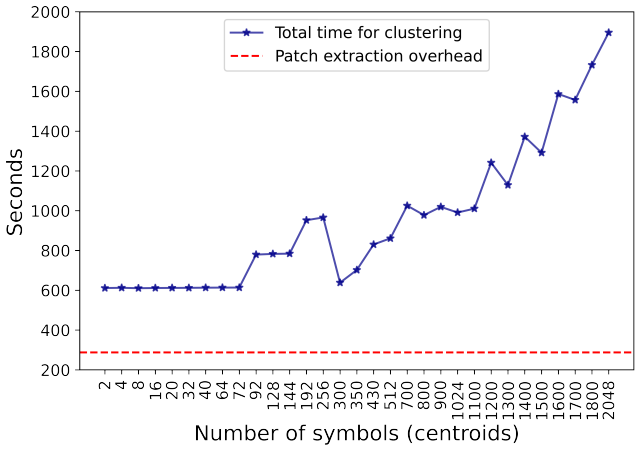


Figure 9. Compute load of CIFAR-10 unsupervised clustering of patches.

Collating all the patches and passing those to the FAISS library has a peak memory load of 22GB for CIFAR-10 dataset. For ImageNet, index building from all patches is not feasible in our setup as the number of patches generated is very high. In this case, we extract patches from each channel of the image, collate the patches from several images, and pass these patches to FAISS *k-means* for building the index incrementally. Building an index on ImageNet training set (images of resolution 224×224) , with a patch size of 2×2 and 2048 patch centroids generates a patch dataset of $112 \times 112 \times 3 \times 150000 = 5.6448$ billion patches, each having a dimensionality of 4. Here we have $M = 2048$, $N = 5.6448$ billion and $P^2 = 4$. This takes 15 days in total, of which 4 days are needed for patch extraction.

With a patch size of 4×4, and 512 centroids a total time of 98 hours is needed of which 48 hours are needed for patch extraction.

We plan to explore the possibilities of vectorizing the patch extraction code or creating and storing a patch dataset in advance in the future, to reduce this overhead.

2.2. Optimal Number of Centroids

To find the optimal number of cluster centroids for CIFAR-10, we use the *knee detection* method of [25]. Fig. 10 illustrates the change in the within cluster sum

square (WCSS) error for using an increasing number of centroids. The knee point detection algorithm of Fig. 10 aptly finds out the maximum curvature for 4 centroids. However, we find that using 4 centroids the SymDNN inference accuracy is 17%. This shows that the traditional method for finding optimal centroid may not be suitable in the context of symbolic inference.

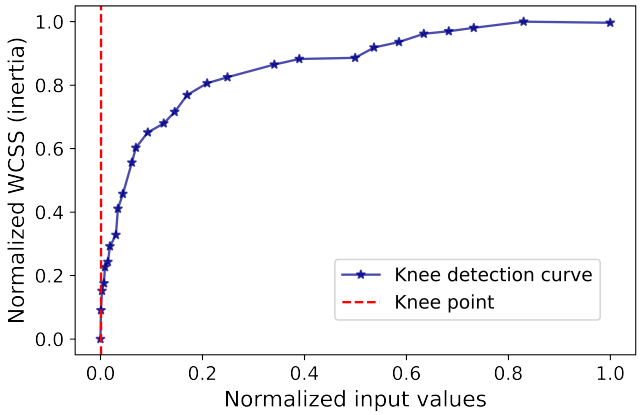


Figure 10. Knee point detection for finding the optimal number of clusters for symbolic inference on CIFAR-10 dataset. Y-axis values of within cluster sum square (WCSS) error values are obtained from the FAISS kmeans inertia metric.

2.3. Optimal Patch Sizes

This part is referred from Sec. 5.4 of the main paper. Fig. 11 illustrates that 2×2 patches result in best accuracy on MNIST dataset. The accuracy reduces with the increase in patch size, and drops by almost 10% for an 8×8 patch size. We also observe that 8×8 patch size gives better accuracy with a larger number of symbols.

For ImageNet, using a 4×4 patch size and 512 symbols, the SymDNN Top-1 accuracy of ResNet-152 comes down to 46% from 70% with a 2×2 patch size and 2048 symbols.

For MNIST, the robust accuracy remains highest on using a lower number of symbols. This is illustrated in Fig. 12.

2.4. Conclusion about the Ablation studies

We conclude that 2×2 patches results in inference preservation for all the datasets we tested. There is some correlation between the combination of a particular patch size and number of symbols, with the inference accuracy, which could not be established with the limited set of experiments we performed.

We believe that Neural Architecture Search (NAS) [13, 18, 34], which has helped in designing compact DNNs can be applied to SymDNN as well. Instead of *brute force* trials, evolutionary or reinforcement learning based optimization in NAS can be applied in this context. The purpose of such a search will be to find the optimal value of SymDNN hyperparameters, with respect to the non-smooth gradient of the SymDNN inference accuracy.

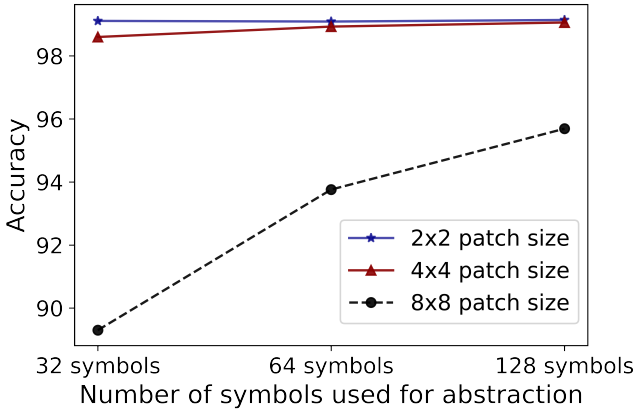


Figure 11. Inference accuracies corresponding to different patch sizes.

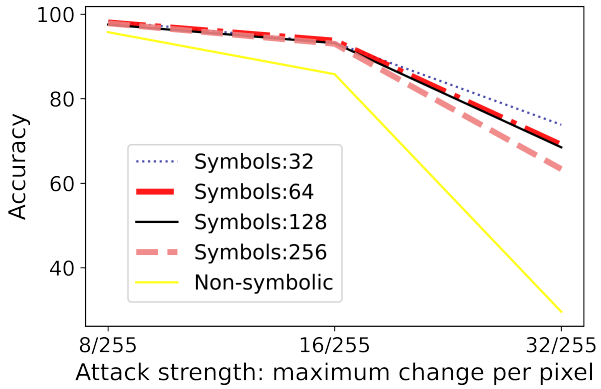


Figure 12. A comparison of robust accuracy for different numbers of symbols used for SymDNN inference, under different attack strengths (maximum change per pixel).

3. Details of Assets Used

This part provides additional details over Sec. 5.1 of the main paper.

Architectures. For ImageNet we use publicly available pre-trained models: WideResNet [31], ResNet-152 [15] and MnasNet [28] from PyTorch model repository[1]. We did not find any separate licensing for these models. We point the reader to the *Facebook Open Source – Terms of Use*: https://pytorch.org/assets/tos-oss-privacy-policy/fb-tos-privacy-policy.pdf. All these models are trained on the ImageNet dataset.

[1] https://pytorch.org/vision/stable/models.html

The licence for ImageNet can be found at: https://www.image-net.org/.

For CIFAR-10 we train the models on our own infrastructure following the ResNet-20 architecture [32]. We could not find explicit licensing terms for this repository https://github.com/zhuchen03/gradinit or the paper [32].

We use the public LeNet-5 [9] architecture for MNIST experiments.

Adversarial Attacks. We choose Torchattacks [17] and the corresponding repository *Adversarial-Attacks-PyTorch*: https://github.com/Harry24k/adversarial-attacks-pytorch for attack implementations. This repository has a MIT license: https://github.com/Harry24k/adversarial-attacks-pytorch/blob/master/LICENSE.

We select *Adversarial-Attacks-PyTorch* for primarily two reasons: (A) It implements most of the successful and the most recent attacks. (B) Unlike many other frameworks, it does not expose an integrated API for adversarial image generation and testing of accuracy. This is important for an easy integration of SymDNN. The symbolic inference needs to modify the image after the adversarial attack, before it is fed to the DNN for inference. SymDNN can also be integrated to frameworks that do not return the adversarial examples directly, however that would require a modification in such adversarial attack frameworks.

For demonstrating efficacy of SymDNN on already robust models, we use RobustBench [6]. These robust models that we use for evaluation purpose are available from: https://robustbench.github.io/ released with a MIT license: https://github.com/RobustBench/robustbench/blob/master/LICENSE.

We compare and integrate our work two state-of-the-art input defenses:

1. NRP [21], available from: https://github.com/Muzammal-Naseer/NRP with a MIT license: https://github.com/Muzammal-Naseer/NRP/blob/master/LICENSE.

2. DefenseGAN [24] PyTorch implementation, available from: https://github.com/sky4689524/DefenseGAN-Pytorch with no specified license.

References

[1] Margareta Ackerman, Shai Ben-David, David Loker, and Sivan Sabato. Clustering oligarchies. In Carlos M. Carvalho and Pradeep Ravikumar, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 66–74, Scottsdale, Arizona, USA, 29 Apr–01 May 2013. PMLR. 2

[2] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. September 2020. 2

[3] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, pages 274–283, 2018. 1

[4] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 284–293. PMLR, 10–15 Jul 2018. 1

[5] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017. 6

[6] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark, 2021. 11

[7] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2196–2205. PMLR, 13–18 Jul 2020. 2

[8] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2206–2216. PMLR, 13–18 Jul 2020. 1, 2

[9] Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. *Handwritten Digit Recognition with a Back-Propagation Network*, page 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. 1, 11

[10] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 321–338, Santa Clara, CA, Aug. 2019. USENIX Association. 6

[11] Swarnava Dey, Jayeeta Mondal, and A. Mukherjee. Offloaded execution of deep learning inference at edge: Challenges and insights. *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 855–861, 2019. 9

[12] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[13] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019. 10

[14] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 6

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 10

[16] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. 2

[17] Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks, 2021. 1, 11

[18] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 10

[19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 1, 6

[20] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks, 2016. 6

[21] Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. A self-supervised approach for adversarial robustness. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 11

[22] Edward Raff, Jared Sylvester, Steven Forsyth, and Mark McLean. Barrage of random transforms for adversarially robust defense. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6521–6530, 2019. 1

[23] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. In *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017. 6

[24] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018. 11

[25] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st International Conference on Distributed Computing Systems Workshops*, pages 166–171, 2011. 9

[26] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, Oct 2019. 1

[27] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. 6

[28] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2815–2823, Los Alamitos, CA, USA, jun 2019. IEEE Computer Society. 10

[29] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. Ensemble adversarial training: Attacks and defenses. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 1, 2

[30] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan Yuille. Improving transferability of adversarial examples with input diversity, 2019. 1, 2

[31] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016. 10

[32] Chen Zhu, Renkun Ni, Zheng Xu, Kezhi Kong, W. Ronny Huang, and Tom Goldstein. Gradinit: Learning to initialize neural networks for stable and efficient training. *CoRR*, abs/2102.08098, 2021. 1, 11

[33] Roland S. Zimmermann. Comment on "adv-bnn: Improved adversarial defense through robust bayesian neural network", 2019. 1

[34] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016. 10