Anti-Adversarial Input with Self-Ensemble Model Transformations

Jeremiah Rounds, Michael J. Henry, Kayla Duskin
Pacific Northwest National Laboratory
jeremiah.rounds, michael.j.henry, kayla.duskin@pnnl.gov

Abstract

Deep-learning models that perform image classification tasks are vulnerable to adversarial inputs that lower model accuracy and recall. Many mitigation techniques sacrifice original model accuracy to gain robustness against adversarial inputs. Additionally, defense techniques often require retraining the classification model, a method that is impractical for models that are already deployed or where the original training data is no longer available. With real-world practicality in mind, we consider a class of anti-adversarial methods that are optimized for the following constraints: minimal loss of original model accuracy, no model retraining, and no access to original model data labels. We introduce and quantify experimental results for optimized model self-ensembling over transformed inputs that conform to those constraints. We observe nearly no change to test set accuracy while mitigating approximately 67% to 70% of effective adversarial inputs on ResNet50 and EfficientNet-B2 models.

1. Introduction

Deep-learning models that perform image classification tasks are vulnerable to adversarial inputs that lower model accuracy and recall[6, 11]. As image classification models are increasingly applied to high stakes problems, the development of effective countermeasures to adversarial inputs is critical. Successful countermeasures have included use of randomization [13], additional model training [9, 4, 14], distilled models [5], gradient obfuscation [1], and novel transformations to input [16] among other techniques. Typically, each of these techniques assume a willingness or ability to mutate the original inference model. In contrast, we propose a method that preserves the original inference model weights and architecture.

Current defense strategies often allow for significant loss of original model accuracy in pursuit of defeating adversarial inputs. Gaining adversarial robustness at the cost of model accuracy creates a natural trade off between anti-adversarial strategies. In this work, we prioritize maintaining model accuracy on non-adversarial data and are willing to tolerate some adversarial loss in pursuit of that goal. We view this a pragmatic real-world scenario, in which massive effort may have been used to create an image classification model, and stakeholders may be rationally unable or unwilling to tolerate loss of model accuracy to mitigate adversarial inputs.

In this work we develop a defensive wrapper for performing inference using an unaltered classifier model by repeatedly applying a random noise function to model input images and then self-ensembling the results for final classification. We also demonstrate that individual noise sources can have strong defensive properties when optimized on only original label recall, and we show that model retraining is unnecessary to build an effective countermeasure wrapper. We examine a number of noise transforms and suggest an approach of self-ensemble using a soft model voting that leads to minimal impact on original model accuracy. Throughout this work, we demonstrate that hyperparameter optimization can be useful in fine-tuning these methods for their purpose of defeating adversaries.

1.1. Related Work

Random image transforms have been suggested as an effective countermeasure to adversarial inputs. In particular, Barrage of Random Transforms (BaRT)[13] showed that performing multiple random noising functions on image inputs is more powerful than choosing an individual noising transformation. They identify that applying randomness on top of randomness creates a stronger defense. They do so by varying both the type of transform and its parameters. While this technique is effective, it comes with a large loss of model accuracy on non-adversarial inputs.

More recently, Qiu et al.[12] suggest another novel noise technique, but also further defines effective noising as having four properties. Paraphrased here they are as follows: Property #1: Noise outputs do not change model output values with respect to non-noised model input; Property #2: Noise outputs measurably change image pixel values with respect to original input values; Property #3: Noise outputs are not constant with repeated application, and fi-

nally, Property #4: The noise function is non-differentiable. Our noise function experiments follow Properties #2 and #3, however, our proposed use of noise functions does not align with Property #1, that an individual noise transform needs to not change model output. We make the distinction that model inference on non-adversarial inputs needs to be unchanged, but we illustrate with this experiment that Property #1 may still be achieved when individual noise transforms change model output. Additionally, we note that because of differentiable approximations to non-differentiable transformations, we understand the desirability of Property #4 but view it as optional.

2. Method

In our approach we specifically avoid retraining, adjusting weights, or changing model parameters of the original image classifier. Our method focuses on noising image inputs to disrupt the effectiveness of adversarial inputs. There are three elements to our method: stochastic noise transformations, model self-ensembling, and hyperparameter optimization on stochastic noise transformations to minimize the loss of recall due to adversaries against the original model label.

2.1. Noise Transformations

A stochastic noise transformation is an image transformation with a random distribution of output. Typically a transformation has a core deterministic operation that is paramaterized by other random variable(s) that have either implicit or explicit distributions. We express $t(x|H)$ as a random transformation of an image x into another image of the same shape given a set of hyper-parameters, H. If it is necessary to express two random draws from the same transform, we express it as $t(x|H,\theta_0), t(x|H,\theta_1)$, etc, where θ_i are random variables from an implied distribution in the use of any randomization.

One key difference from previous work is that we choose the hyper-parameters for the random distributions by systematic optimization toward defending against adversaries. In previous work, there exist H, but no published method for choosing the underlying distributions of the noise transformations which we find crucial to the success of the defense.

2.2. Self-Ensembling

Our method relies on performing model inference on several noised versions of each input image and then ensembling the outcomes to make the final inference decision. The classification model $f(x)$ produces logit outputs given input x. When using a noised image as input, $f(t(x|H,\theta_k))$ has a distribution induced by the random transformation t, so that for any given input there are multiple random outputs of the model.

A model averaging or model voting is performed over K random draws of $t(x|H)$ where H are the hyper-parameters for the random transformation. Model averaging is defined as $\frac{1}{K}\sum_k f(t(x|H,\theta_k))$.This self-ensemble is K activations of the same model ensembled into one consensus logit opinion.

In hard voting, the argmax of $f(t(x,H,\theta_k))$ is a vote for a class output. K votes are tabulated over K transformations $f(t(x,H,\theta_k))$ and the ultimate consensus top-1 label is assigned as the votes. We found reason not to choose this form of model averaging. The smoothness of the weighted logits in soft-voting appears to be a desirable feature in this application, and in hard voting we observed oscillation in weighted logits depending on if K was even or odd.

2.3. Hyperparameter Optimization

We choose the hyperparameters of our noising functions, H, by optimizing them to produce images that are classified the same way by the original model as the non-adversarial version of the image. That is, we use the recall of the original model output on the non-adversarial version of an image as the objective function to score the value of H as it acts on an adversarial input.

We create a non-adversarial corpus of data (X,Y), where $(x,y) \in (X,Y)$ are input image and true label pairs. We only utilize true Y in testing. In its place for hyper-optimization, we utilize the top-1 non-adversarial model label, $\hat{Y} = (\text{argmax} f(x) \text{ for each } x \in X)$. We also create an adversarial corpus (X', Y') by applying adversarial methods to the images in X and recording the model outputs on the adversarial images such that the accuracy of the original model is zero. That is to say for all $(y, y') \in (Y, Y'), y \neq y'$.

The wrapped model is defined as $g(x|H) = \text{argmax}\frac{1}{K}\sum_k f(t(x|H,\theta_k))$. The objective function used to find the optimal H is:

$$\text{Recall}(X, X', \hat{Y}|H) = \frac{1}{N} \sum_{(x,x',y)\in(X,X',\hat{Y})} I\left(g(x'|H) == y\right), \quad (1)$$

where N is the number of images in the hyper-optimization training set. $\text{Recall}(X, X', \hat{Y}|H)$ is optimized with Tree-Structured Parzen Estimators implemented in the Python Hyperopt library [2, 3]. Priors are uniform over allowable range depending on H

We prefer the recall of the original top-1 label over the ground truth label accuracy as an objective function because it does not require knowledge of the original training data set or even labeled training data. We do use the original model label for experimental evaluation.

3. Experiments

3.1. Data

Data used in experimentation are 10 classes of Imagenet [8] selected at random from public validation data sets. This is further split into 90/10/10 % for training, validation and testing respectively. True labels for these data are used only in testing sets and test time measurements. Training is done without original label.

3.2. Models

We use EfficientNet-B2[15] and Resnet50[7] as our classification models. EfficientNet was a clear leading new image classification architecture at the time this research began. Resnet50 has continued success in ensembled models.

3.3. Adversarial Methods

The adversarial methods on Imagenet classifiers are derived from off-the-shelf software in the Python Cleverhans library[11]. Each image from train, validation, and test receives one adversarial perturbation from Fast Gradient Sign Method ($\epsilon = 2.0$)[6], Projected Gradient Descent (max $\epsilon = 2.0$) [9], Carlini and Wagner L2 attack (confidence=5.0)[1], and DeepFool [10]. All adversarial images are filtered so that there is a top-1 evasion of the original label, and unsuccessful perturbations are discarded. Thus original model accuracy on the adversarial data set is 0.0.

3.4. Noise Transformations

We consider four noise transformations most promising in initial studies. U(a, b) describe uniform random distributions between real numbers (a, b). The four noising functions are CropZoom, JPEG compression, Gaussian Blur, and Singular Value Decomposition (SVD). Details on their implementation can be found in the appendix.

4. Results

The top single ensemble noise function of Table 1 is an optimized CropZoom. CropZoom outperforms other single noise operations and also outperforms a CropZoom followed by JPEG noise source in sequence. On EfficientNet, CropZoom with 10 logit-averages has nearly equivalent non-adversarial accuracy as the baseline model. On ResNet50, CropZoom with 10 logit-averages has better than baseline non-adversarial accuracy. We speculate this may be due to the 10-class training and test Imagenet data set as being easier to classify on average for Resnet50 with a center crop zoom operation. If so, this would be a type of bias induced by optimizing an image corpus for a particular model. For example, Table 2 shows parameters cannot use as input the lower left quarter of an image. What the increase in accuracy is suggesting is that there are training data sets such that the lower left quarter of an image is never more useful than having an average increase in image resolution input to the Resnet50 model (which requires a 224 by 224 image input resize). This bias effect did not manifest in EfficientNet, which is possible if EfficientNet handles multiple resolution object inputs better than Resnet50.

As demonstrated in Figures 1 and 2, the effect of self-ensemble K with logit averaging is moderate. There is an initial drop of accuracy for applying a noise source that is largely recovered by logit averaging of repeat noise applications. The growth of accuracy with increasing K is consistent across models.

4.1. Discussion

We find that one distinguishing feature of our method is the preservation of model accuracy on non-adversarial inputs. Our most straight-forward two operation comparison to BaRT [13] is our CropZoom + JPEG sequence. This is a fair comparison because has a variable number of noise operations, including these two noise transforms. With our method, we observe 0.2% drop to in non-adversarial accuracy for Resnet50. While BaRT observed an approximately 12% drop for the same metric. We speculate that is due to retraining the original Resnet50 model. Additionally, we note that BaRT is randomly selecting from both effective and ineffective anti-adversarial noises to form an ensemble of two, and as a result, it has worse performance than if only the two best noise transforms were selected. While this method of randomly mixing noise functions provides a more difficult problem for the adversary, it also mixes weak and strong noise transforms in terms of objective performance. We chose to employ only the strongest noise sources furthermore optimized their hyperparameters.

Our proposed wrapper does not have a model retrain, and avoids the noise sources that create low non-adversarial accuracy when applied in succession. In addition, model averaging handles noise differently than other methods. For instance, ensembles in BaRT are successive operations on the same input image, so that as a greater number of noise operations are applied the noised input image continues to diverge from the original image. With our wrapper, each noise operation is re-applied to the original input and then model output is used to construct a consensus via ensembling. We see a trend of increasing accuracy as we increase the number of noisers used, rather then a decrease in accuracy that occurs when noise is used in sequence on a single input.

Qiu et. al. [12] improved upon BaRT by asserting the ideal transform has the property that the model output is invariant to the noise transform, but otherwise the work shared similarity with BaRT in the sense that it was one model output per model input.

There is a basic incompatibility to noise transforms use-

Model	Transform	Avg. Acc on Adversaries K=1	Avg. Acc on Adversaries K=25	Avg. Acc on Non-Adversaries K=25	Baseline Accuracy
Resnet50	CropZoom	0.655	0.678	0.804	0.785
	JPEG	0.501	0.506	0.685	0.785
	GaussianBlur	0.413	0.428	0.565	0.785
	SVD	0.364	0.364	0.652	0.785
	CropZoom + JPEG	0.636	0.667	0.783	0.785
EfficientNet B2	CropZoom	0.679	0.715	0.852	0.858
	JPEG	0.636	0.637	0.792	0.858
	GaussianBlur	0.542	0.542	0.661	0.858
	SVD	0.491	0.494	0.683	0.858
	CropZoom + JPEG	0.704	0.737	0.803	0.858

Table 1. Test set accuracy for noising and baseline evaluations. Baseline evaluations are of the original model on test sample sizes of 2295 and 1687 images for Resnet50 and EfficientNet B2, respectively. K=25 refers to using 25 activations of the model on 25 random instances of the noise input. CropZoom benefits the most from logit averaging. The best over-all performer against the test library of adversarial images is CropZoom+JPEG, however, the double noise has a non-adversarial model accuracy penalty. Whereas CropZoom alone has negligible non-adversarial performance losses.

ful for model averaging and noise transforms useful for repeated noise stacking with the invariant property. If for a random transform, it is always the case that $f(t(x)|H;\theta_0))$ = $f(t(x|H;\theta_1))$, then the expectation of the model averaging is equivalent to a single transform.

$$E_H[1/K \sum_k [f(t(x|H,\theta_k))] = f(t(x|H;\theta)), \quad (2)$$

This outcome is not damaging to our technique, but the model averaging is not useful because all elements of the summation are the same and there is no variance in model output to average over.

One trade-off of utilizing self-ensembling is the higher computational cost at inference time. While each image requires K times the compute cost while performing inference, the model is weights are fixed which avoids the computational cost of retraining the full model. In situations where model retraining would be more costly or infeasible due to other constraints (such as unavailability of data) our method offers an alternative at the cost of greater inference computation.

5. Conclusion

We find that the use of random transforms is useful in real-world scenarios where it is preferable to modify the inputs rather than modifying the model. However, we do note that this comes at the cost of increasing the number of inference decisions that a model needs to make at run time which could be impractical in certain scenarios. We also note that there are two emergent themes in the use of noise for anti-adversarial inputs. The first is repeated noise reapplied to the same image. This countermeasure achieves its anti-adversarial property by stacking more noise than the adversarial technique is prepared to handle. The cost of using methods that stack noise is a steadily degraded original model accuracy. Our alternative technique is to model average repeated noise applications. With our technique, the original model accuracy is not steadily degraded as more noise is added to the image. We argue that there are reasons to prefer each strategy depending on goals and available resources.

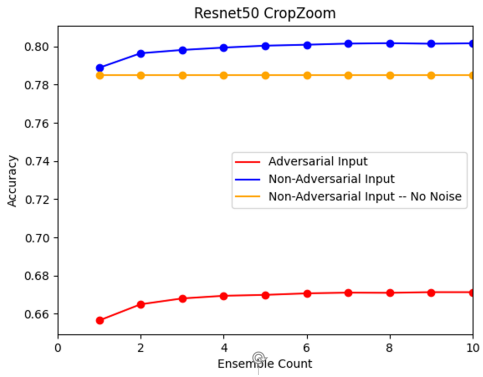

Figure 1. Effect of self-ensemble count on Resnet50 test set accuracy in non-adversarial and adversarial inputs. Logit averaging increases accuracy 2 to 3% depending on context.

References

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

[2] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In

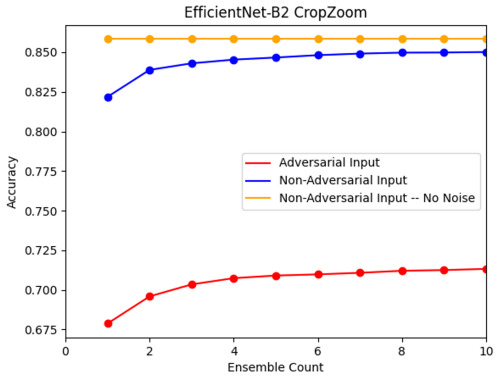

Figure 2. Effect of self-ensemble count on EfficientNet B2 test set accuracy in non-adversarial and adversarial inputs. Logit averaging increases accuracy 2 to 3% depending on context.

25th annual conference on neural information processing systems (NIPS 2011), volume 24. Neural Information Processing Systems Foundation, 2011. 2

[3] James Bergstra, Dan Yamins, David D Cox, et al. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in science conference*, volume 13, page 20. Citeseer, 2013. 2

[4] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, Percy Liang, and John C Duchi. Unlabeled data improves adversarial robustness. *arXiv preprint arXiv:1905.13736*, 2019. 1

[5] Micah Goldblum, Liam Fowl, Soheil Feizi, and Tom Goldstein. Adversarially robust distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3996–4003, 2020. 1

[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1, 3

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 3

[9] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 1, 3

[10] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016. 3

[11] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, et al. Technical report on the cleverhans v2. 1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2016. 1, 3

[12] Han Qiu, Yi Zeng, Qinkai Zheng, Tianwei Zhang, Meikang Qiu, and Gerard Memmi. Mitigating advanced adversarial attacks with more advanced gradient obfuscation techniques. *arXiv preprint arXiv:2005.13712*, 2020. 1, 3

[13] Edward Raff, Jared Sylvester, Steven Forsyth, and Mark McLean. Barrage of random transforms for adversarially robust defense. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6528–6537, 2019. 1, 3

[14] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 307:195–204, 2018. 1

[15] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 3

[16] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019. 1

[17] Yuzhe Yang, Guo Zhang, Dina Katabi, and Zhi Xu. Me-net: Towards effective adversarial robustness with matrix estimation. *arXiv preprint arXiv:1905.11971*, 2019. 6

Appendix

A. Noising Functions

CropZoom In crop zoom an image is subset to pixels in a rectangle defined by $(r_0, c_0) \times (r_1, c_1)$. The interior section of the image defined by this rectangle is resized to model input dimensions, which has the effect of cropping then zooming. A crop zoom transformation is parameterized as $t(x; r_0, c_0, r_1, c_1)$, and in general we express these parameters on [0,1] as a portion of the image size.

The corners of the crop zoom are allowed to distribute randomly over a range determined by a hyperparameterization. This distribution is described as a random initial point to a random offset from that initial point. $r_0 \sim \mathrm{U}(l_0, l_1)$, $d_r \sim \mathrm{U}(u_0, u_1)$, and $r_1 = r_0 + d_r$, so that r_0 is the random initial row, and d_r is the random row offset to the other corner. $c_0 \sim \mathrm{U}(l_0, l_1)$, $d_c \sim \mathrm{U}(u_0, u_1)$, and $c_1 = c_0 + d_c$, describes a similar distribution on columns. l_0, l_1, u_0,and u_1 are hyper-parameters selected to optimize a scoring function based on model results. Thus, $t(x; l_0, l_1, u_0, u_1)$ is a fully defined distribution conditioned on 4 uniform random variables, U, that describes a randomized crop zoom on an image.

JPEG In the JPEG noise an image is encoded in JPEG with a compression value v. To create random distribution for model averaging, $v \sim \mathrm{U}(v_0, v_1)$, where v_0 and v_1

are hyper-parameters. $t(x; v_0, v_1)$ describes a randomized JPEG image.

Gaussian Blur A Gaussian kernel radial basis function smooths input image x. With probability p, an image is either blurred as single channel parameterization, or red, green, and blue are parameterized independently. In either case, the channel is blurred with a radial basis function with standard deviation s, where $s \sim U(s_0, s_1)$, where s_0 and s_1 are hyper-parameters. $t(x; p, s_0, s_1)$ describes a randomized blurred image. This transform follows the one introduced by Barrage of Random Transforms (BART).

Singular Value Decomposition (SVD) Motivated by recent use of SVD for anti-adversarial inputs [17], we examined a noised SVD transform. Similar to Gaussian blur an SVD transform on an image reduces image color complexity. Input x is decomposed by color channel into matrix values R, G, B (red, green, blue) of shape equal to pixels in the original image. Matrix decompositions of $R = U_R \Sigma_R S_R^T, G = U_G \Sigma_G S_G^T, B = U_B \Sigma_B S_B^T$. Σ is a diagonal matrix with singular values on the diagonal. The first N largest singular values select the first columns and rows of U, Σ,S to create compressed channel approximations $R^* = U_R^* \Sigma_R^* S_R^{*T}, G = U_G^* \Sigma_G^* S_G^{*T}, B = U_B^* \Sigma_B^* S_B^{*T}$, where * indicates reduces dimension.

To add a stochastic element to this decomposition, the Σ^* (singular value loading) is given an additive and multiplicative white noise. Each color component is randomly modified as $\Sigma_{ii}^* = \beta_{ii} \Sigma_{ii} + \alpha_{ii}$, where $\alpha_{ii} \sim \text{Normal}(0, a \times s/K)$ and $\beta_{ii} \sim \text{Normal}(1, 0.025 \times b)$, where s is the standard deviation of all singular values and K is the number of singular values in Σ. The additive component on singular values is kept small near 0, and the multiplicative component is mostly determined in optimization. In addition the loading vectors are given a small additive noise, $U_{ij}^* = U_{ij}^* + c_{ij}$ and $S_{ij}^* = S_{ij}^* + d_{ij}$ where $c_{ij} \sim \text{Normal}(0, 0.001)$ and $d_{ij} \sim \text{Normal}(0, 0.001)$. There are a number of generative model decisions for these noise that could be altered in future work, but this transform can be understood as simplifying the image with N component vectors followed by noising the simplification.

In this parameterization, number of components, N, and scaling values a and b are hyper-parameterized for $t(x; N, a, b)$.

Transform	Resnet50 Parameters	Efficientnet B2 Paramet
CropZoom	$r_0, c_0 \sim U(0.061, 0.166)$ $r_1, c_1 \sim U(0.645, 0.747)$	$r_0, c_0 \sim U(0.128, 0.298$ $r_1, c_1 \sim U(0.738, 0.968$
JPEG Compression	$v \sim U(26, 43)$	$v \sim U(29, 34)$
Gaussian Blur	$p = 0.644$ $s \sim U(1.179, 2.460)$	$p = 0.644$ $s \sim U(1.258, 1.675)$
SVD	N = 40 a = 1.484 b = 2.565	N = 40 a = 0.924 b = 3.436

Table 2. Optimal parameters of t(x,H) for individual noise transforms. Optimization is on original model top-1 label recall with optimized parameterizations. $U(a, b)$ is a uniform distribution between (a, b).