Robust Training using a Push-Pull Inhibition Layer for Adversarial Robustness in Convolutional Neural Networks

Julián Navarro Di Pasquale
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
j.navarrodipasquale@student.utwente.nl

ABSTRACT

Adversarial attacks have gained considerable attention in recent years due to increasing real-world, safety-critical applications of Deep Neural Networks. The vulnerability against such attacks spans multiple domains and thus exhibits security concerns, mainly because they are challenging to detect, and an understanding of their existence is lacking. Consequently, the research community has proposed many defense strategies to inherently induce robustness properties through domain-specific design or training mechanisms.

This paper concentrates on defending against adversarial attacks within the image classification domain, where so-called adversarial examples are constructed by carefully crafting (imperceptible) perturbations on an image such that a classifier produces erroneous predictions with high confidence. More Specifically, we quantitatively analyze an approach that builds upon a biologically-inspired component called the push-pull layer that increases robustness against naturally distorted/corrupted images. We combine the said component with adversarial training to investigate its robustness-efficacy against various adversarial attacks and threat models. The findings in this experimental study indicate that the approach allows the component to translate its properties to adversarial examples and, with further research, may prove itself as a general-purpose defense tool.

Keywords

Deep Learning, Adversarial Robustness, Inhibition, Push-Pull Layer

1 Introduction

The adversarial susceptibility of deep neural networks retains growing relevance as deep learning [1] progressively penetrates security-sensitive environments. Furthermore, the increasing intricacies and stakes of the newly entered domains incentivize adversaries to target such critical systems. Therefore, building intrinsic robustness or defense mechanisms into deep learning techniques becomes a crucial research and engineering goal. Pertinent domains include self-driving cars [2, 3], computer-aided diagnosis (CAD) [4], malware detection [5], and many more [6, 7, 8].

The threat against adversaries is not unique to DNNs. It has long been a study of conventional machine learning (ML), where the classes of attack targeted features or the ML models' training process (data poisoning) [9]. However, deep learning usually considers so-called adversarial examples [10, 11, 12] during test time, where the attacker carefully modifies the raw input space. Furthermore, adversarial examples exhibit cross-model generalization properties. That is, an attacker can generate an attack on one model and transfer it to another (possibly with different architecture or training set) [11, 13], demonstrating the imminent threat and need for robustness in real-world applications. Researchers primarily investigated these attacks within the context of image classifications. However, they are proven to exist in but not limited to semantic segmentation, object detection, and natural language processing [8, 7, 6]. Despite much-attempted work directed at understanding the cause of their existence and characteristics [12, 14], the research community appears to lack joint agreement.

Despite the lack of consensus, many defense strategies were proposed on existing knowledge to construct models resistant to adversarial attacks. These defenses often take on the form of either data augmentation [15, 16], modifications to networks [17], or prepending a network to the target. Yet, the inclusion of adversarial examples within the optimization process (adversarial training) remains the most effective among all strategies; however, some findings suggest it is not the optimal solution [18]. Nevertheless, in this experimental study, we combine adversarial training with a biologically-inspired layer for Convolutional Neural Networks (CNNs) by Strisciuglio et al. [19] as an approach to adversarial robustness. The proposed layer emulates a form of response inhibition that allows an improved extraction of semantic information and demonstrated a robustness-increase of the classification rate on corrupted images by, for instance, fog, blur, or illumination. Given these properties, we hypothesize that by exposing the push-pull layer to an adversary, the increased selectivity of input and robustness against (natural) perturbations translates to adversarial examples. We demonstrate in this experimental study that the properties translate to adversarial examples and that, with further research, the push-pull layer is a prospect for a general-purpose defense.

We organize the remainder of this paper as follows. First,

36th Twente Student Conference on IT February 4th, 2021, Enschede, The Netherlands.

section 2 briefly discusses previous work on adversarial training. Then, section 3 introduces the push-pull layer and relevant concepts such as robust training and threat models that constitute the framework of this experimental study. Section 4 describes the experimental setup, attacks, configurations, metrics to evaluate robustness, and the results. In section 5, we discuss deficiencies and further improvements on the approach. Finally, we conclude this paper in section 6.

2 Related Work

The first notion of adversarial training introduced Szegedy et al. [11], wherein the authors trained a classifier with a mixture of clean and adversarial examples in an alternating fashion. However, the technique did not exhibit improvements beyond standard regularization methods applied to DNNs [12], partly due to the limited exploration and the computational cost of generating adversarial examples using L-BFGS. Goodfellow et al. [12] reduced the computational barrier by proposing the fast gradient sign method (FGSM) that exploits the local linearity of DNNs and backpropagation. Nonetheless, in spite of robustness improvements, training with single-step attacks turned out to be vulnerable against iterative attacks. In [20], Tramér et al. argue that the poor performance of single-step attacks within the min-max formulation [21, 22] is due to sharp curvate in the loss surface (called *degenerate global minimum*). The consequence of this curvature is that the model learns to generate weak attacks instead of optimizing robustness against first-order attacks. Therefore in the same paper, they propose a new algorithmic approach called *Ensemble Adversarial Training* that disassociates the adversary from the optimization objective by training with adversarial examples generated on other pre-trained models.

3 Preliminaries

To adequately evaluate the defense's performance and provide context on the conditions it may claim guarantees and results upon, we first discuss constituent parts of a threat model. Then, following the threat model, a brief (formal) description of robust training is given and reasons for it to be the framework of choice. Lastly, we present the push-pull layer subject of this experimental analysis.

3.1 Threat Model

A threat model determines the conditions under which an adversary operates. One usually considers three characteristics: *Knowledge*, *Capabilities*, and *Goals*.

3.1.1 Knowledge

White-Box. In a white-box attack, the adversary is assumed to have complete knowledge and an unconstrained set of resources. So, the adversary can generate attacks while considering the architecture, (hyper-)parameters, the dataset, and the underlying defense of the model. On that basis, the attack can adaptively generate adversarial examples and (theoretically) consume as much computation needed to fool the classifier *directly*. Ideally, a defense wants to reach robustness against such adversaries because it implies increased robustness against black-box attacks.

Black-Box. The black-box attack scenario assumes a much more restrictive adversary in terms of knowledge and interactivity with the target than its white-box counterpart. Essentially, a black-box attacker has two mechanisms at its disposal to generate adversarial examples. The first mechanism exploits the transferability phenomena (*transfer-based*), where an adversary trains a surrogate model that emulates the target then generates adversarial examples on the surrogate for transfer to the target. The second is a query-feedback mechanism, where the attacker continuously crafts the adversarial example (without surrogate) while considering the query score (*score-based*) or decision (*decision-based*). As a result, black-box attacks are strictly weaker than their white-box counterpart and are, for these reasons, considered more practical and closer to real-world applications.

3.1.2 Capabilities

To concretely determine the capabilities of defense and categorize under which conditions and against which attacks it may provide resistance, we need to impose further restrictions on the adversary. The following presents restrictions imposed for the sake of this experimental study.

Each adversary generates its adversarial examples using a predefined set of perturbation magnitudes ϵ and individually applies it to each benign image.[1] Furthermore, the perturbation magnitude cannot arbitrarily be applied to the benign input and is subject to an optimization objective and a distance metric.

Perturbation Objective. The optimization objective sets out the conditions under which a perturbation is considered successful. For the case of *optimized perturbation*, the objective is to find an adversarial example that sucessfully fools the classifier while the perturbation magnitude ϵ is *minimal*. For the *constrained optimization* technique, the objective is to *maximize* the loss, given adversarial input, while the applied perturbation does not exceed ϵ under the given distance metric.

Perturbation Distance Metric. The intuition behind the distance metric is to define a measure that quantifies the severity of modification applied to an image; the smaller the value, the less perceptible the change. Frequently, the ℓ_p-norm is used and here we will consider the ℓ_2- and ℓ_∞-norm[2], where

- ℓ_2 computes the euclidean distance between the adversarial and benign example.
- ℓ_∞ implies that each pixel's peturbation does not exceed ϵ.

3.1.3 Goals

An adversary may have varying incentives to attack a system, including but not limited to compromising integrity or availability. In the context of image classification, we consider only the goal of subverting the integrity of a classifier, excluding a possible dependence another system could have on classification results.

Untargeted. Here, an attacker intends to have a classifier misclassify the perturbed image. Therefore, the class a model assigns the image to is irrelevant, presuming it is not the actual class. The adversary may either minimize the chance of the original class or take from a pool

[1] There exist also *universal* perturbations that are image-agnostic [23].
[2] Another commonly used metric is ℓ_0 which counts the number of pixels changed. We do not consider this metric here.

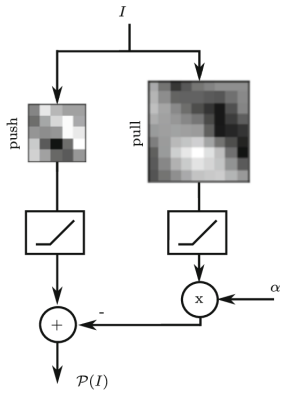

Figure 1: A structural depiction of the push-pull layer. Source: [19]

of generated adversarial examples one with the smallest perturbation.

Targeted. The objective remains to lead the classifier into misclassifying the adversarial example. However, the adversary attempts to perturb the image into a specific target class by maximizing its probability.

3.2 Robust Training Framework

To provide a paradigm that guarantees robustness in a principled manner, Madry et al. (2017) [22] embedded the adversary into the empirical risk minimization (ERM) framework, formulated as follows:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathbb{D}} \left[\max_{\delta \in B(x,\epsilon)} L_{ce}(\theta, x+\delta, y) \right],$$

where $(x, y) \sim \mathbb{D}$ represents the training data drawn from distribution $\mathbb{D}$, $B(x, \epsilon)$ is the permitted perturbation set defined as $B = \{x+\delta \in \mathbb{R}^m | \, \|\delta\|_p \leqslant \epsilon\}$ and L_{ce} is the cross-entropy loss. Therefore, we want to minimize the empirical risk (of loss) with respect to the worst-case adversarial input. Since its intractable to solve the optimization, in practice, we train a model solely on adversarial examples generated by a *strong* adversary [21] that (approximately) maximizes the inner optimization and use stochastic gradient descent (SGD) to minimize the outer optimization. Madry et al. (2017) demonstrated (theoretically and experimentally) the tractability of the optimization using a new strong, and *iterative* attack called projected gradient descent (PGD) and how models become more robust against a range of adversarial attacks depending on first-order information. They dubbed the procedure PGD-AT and is (still) considered the state-of-the-art approach for inducing robustness into DNNs against white-box and black-box attacks [20, 15]. However, a negative aspect of their procedure is the incurring prohibitive cost that increases training time by a factor of $(k + 1)$, where k is the number of iterations. Furthermore, the above formulation does not provide any guarantees for zero-day attacks.

3.3 Push-Pull Layer

The inspiration for the so-called *push-pull layer* comes from early (simple) cells in the primary visual cortex exhibiting simultaneous excitation and suppression of distinct neural receptive fields (RFs). Sensory neurons responding to such cells integrate excitatory and inhibitory RFs into a response referred to as push-pull inhibition. The combined effect of this response property is the increased selectivity of visual stimuli despite corruption. Strisciuglio et al. [19] modeled this phenomena as follows:

$$P(I) = \Theta(k * I) - \alpha\Theta(-k_{\uparrow h} * I), \tag{1}$$

where Θ is a ReLU (activation) function, α a weighting factor for the *inhibition strength* for the response map of the pull component, and $\uparrow h$ is an upsampling operator with scalar $h > 1$ (See Figure 1). In other words, the response of the push-pull layer is a linear combination of the rectified responses of the push (excitatory RF) and the pull kernel (inhibitory RF).

Table 1: Depicts the attack methods employed for robustness evaluation. Each entry corresponds to the attack's limitations within the threat model it is considered.

Attack	Knowledge	Goals	Capabilities	Metric
FGSM [12]	white & transfer	untargeted	constrained	ℓ_∞
PGD [22]	white & transfer	untargeted	constrained	ℓ_∞
C&W [25]	white & transfer	untargeted	optimized	ℓ_2
DeepFool [26]	white	untargeted	optimized	ℓ_∞
SPSA [27]	score	untargeted	contrained	ℓ_∞
NES [28]	score	untargeted	constrained	ℓ_∞

4 Experiments

This section extensively evaluates the robustness of the push-pull layer and robust training combination using the benchmark dataset CIFAR-10 [24] in both white-box and black-box settings. We start by alluding to the experimental setup with its implementation details, introduce the evaluation metrics, and finally present and discuss the observed results.

4.1 Experimental Setup

Environment. To conduct the experiments, we have implemented the adversarial training procedure from scratch using Python and Pytorch [29]. We took the implementation details of the push-pull layer and its integration into ResNet [30] from Strisciuglio et al.[3] Furthermore, for all white-box attacks, we utilize Foolbox [31], which is a (deep learning) framework-agnostic library of the most common attacks. For the transfer-based and score-based black-box attacks, we use Foolbox and the benchmarking library ARES[4] (Adversarial Robustness Evaluation for Safety) by Dong et al [32]. Lastly, the experiments have been exclusively executed on shared, cloud-hosted GPUs due to the prohibitive computational complexity of adversarial training. The platforms provided a Jetson AGX Xavier interface, a Tesla P100, or a Tesla T4. We will publish all of the associated code soon.

Attacks. We employ in total six attack methods under various threat models. These are FGSM, PGD, and C&W for white-box and transfer-based black-box attacks; DeepFool, which only operates with white-box knowledge. The remaining are SPSA and NES as an interactive black-box attack, where both are score-based. Since we do not discuss attacks in detail, table 1 lists all attacks, summarizes the threat model they operate in (in this study), and references each attack's originating paper. For more context and a broader view of attacks in general, consider [8, 7, 6]. Lastly, relevant configuration details of each attack are

[3] https://github.com/nicstrisc/Push-Pull-CNN-layer
[4] https://github.com/thu-ml/ares

Table 2: Robustness accuracy (%) of all ResNet models against untargeted white-box attacks.

	Adversarial						Adversarial-PP					
Network	Benign	PGD_7	PGD_{10}	PGD_{20}	FGSM	C&W	Benign	PGD_7	PGD_{10}	PGD_{20}	FGSM	C&W
ResNet-20	86.38	80.01	76.63	64.67	11.10	83.95	85.83	79.17	75.78	63.52	18.44	83.36
ResNet-32	85.82	79.61	76.42	64.94	20.76	83.66	85.42	79.18	76.09	64.79	19.81	83.25
ResNet-44	86.71	80.37	77.30	66.03	21.15	84.36	85.30	79.13	76.24	65.65	21.58	83.07
ResNet-56	87.41	81.35	78.32	67.69	19.44	85.47	85.51	79.37	76.42	66.41	22.09	83.34

Table 3: Robustness acurracy (%) of untargeted transfer-based black-box attacks using ResNet-56 as a source.

	Adversarial						Adversarial-PP					
	PGD_{20}		CW_{40}^{50}		FGSM		PGD_{20}		CW_{40}^{50}		FGSM	
Source \ Target	ResNet-20	ResNet-56	ResNet-20	ResNet-56	ResNet-20	ResNet-56	ResNet-20	ResNet-56	ResNet-20	ResNet-56	ResNet-20	ResNet-56
No Push-Pull	85.24	86.86	86.32	87.37	79.53	78.97	85.24	85.09	85.78	85.44	79.42	80.02
Push-Pull	85.35	86.72	86.36	87.39	78.59	77.54	85.17	84.82	85.78	85.44	78.32	78.92
Adversarial	82.30	83.44	86.12	86.74	50.49	51.15	81.80	81.55	85.60	85.28	50.44	52.59
Adversarial-PP	82.20	83.68	86.17	87.25	50.56	51.66	81.94	81.38	85.55	85.25	51.35	51.93

given in the results section 4.3.

4.2 Evaluation Metrics

Here we are solely interested in the capacity of a classifier to resist attacks aiming at compromising its integrity and the settings under which an adversary increasingly deceives and degrades the performance of a network. Thus, to capture the adversarial robustness and general efficacy of defense against various attack configurations, we employ robustness curves as proposed in the benchmarking framework by Dong et al. [32]. These enable us to evaluate the overall robustness of the defense, compared to just using pointwise accuracy and one perturbation budget, by exposing the defense to multiple, different attacks with increasing perturbation strength. Furthermore, these curves provide the ability to investigate performance-related conditions under which an attack becomes effective and perform sanity checks valuable for excluding misleading defense properties [33, 32]. Consequently, we define the metrics as follows. Let $C(\mathbf{x}) = \mathbf{y}$ be a classifier where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$. Furthermore, let $A_{\epsilon,p}$ be an adversary that generates adversarial examples $\mathbf{x}^{adv} = A_{\epsilon,p}(\mathbf{x})$ for input $\mathbf{x}$ using a perturbation magnitude ϵ and the ℓ_p-norm.

Then, the overall performance of a classifier at test time is measured as follows:

$$Acc(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}(C(A_{\epsilon,p}(\mathbf{x}_i)) = y_i),$$

where $\{\mathbf{x}_i, y_i\}_{i=1}^N$ is the test set and $\mathbf{1}(\cdot)$ is the indicator function - characterizing the general *accuracy* of the classifier.

However, the accuracy of a classifier alone does not fully disclose its ability to resist attacks since we do not distinguish between successful adversarial examples and images that would have been misclassified in their benign form. So, to determine the capability of an adversary in generating attacks, we compute and consider the *attack success rate*:

$$Asr(\mathbf{x}, \mathbf{y}) = \frac{1}{M} \sum_{i=1}^{N} \mathbf{1}(C(\mathbf{x}_i) = y_i \wedge C(A_{\epsilon,p}(\mathbf{x}_i)) \neq y_i),$$

where $M = \sum_{i=1}^{N} C(\mathbf{x}_i) = y_i$.

We use the above metrics to construct *accuracy (attack success rate) vs. perturbation magnitude* curves to establish the overarching robustness and conditions under which an adversary substantially breaks the resistance.

4.3 Results

This section initially presents a set of experiments to provide a baseline between adversarially trained networks with and without the push-pull layer. Following that, we analyze more closely the defense-efficacy of the push-pull layer with fixed hyperparameters against a range of attack configurations and perturbation magnitudes, both in white-box and black-box settings (see Section 3.1 & 4.1). Lastly, we investigate whether learning the inhibition strength α improves input selectivity by fine-tuning it against an adversary.

4.3.1 *Baseline Evaluation*

Here we evaluate the robustness of an adversarially trained model with a push-pull layer (Adversarial-PP) against its counterpart (Adversarial), which does not include the layer, in a white-box and transfer-based black-box setting.

Training settings. We adversarially train a ResNet-20, ResNet-32, ResNet-44, and ResNet-56 with and without a push-pull layer using ℓ_∞-bounded PGD with random start and 7 iterations, a perturbation magnitude $\epsilon = 8/255$, and a relative stepsize of $\epsilon/4$. Furthermore, all of the networks are optimized against the PGD_7 adversary using SGD with a Nesterov momentum of 0.9, weight decay 1×10^{-4}, and an initial learning rate of 0.1, which is divided by 10 at the 80-th and 120-th epoch. All of the networks have been trained for approximately 160 epochs with a batch size of 128. We set the inhibition strength $\alpha = 1$ and the scaling factor $h = 2$ of the upsampling operator regarding the push-pull layer (see Equation 1).

White-box Robustness. We evaluate the robustness of all ResNet models against three types of standard attacks: FGSM, PGD (7, 10, and 20 iterations), and C&W (40 iterations, zero confidence, and a step size of 0.01). Furthermore, all attacks have complete access to the model, as described in Section 3.1.1, and are restricted by the same perturbation magnitude ϵ. Additionally, we summarize the white-box robustness of all models in Table 2, where

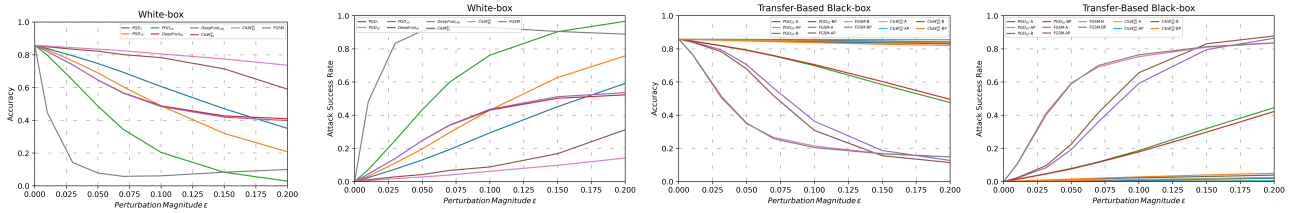

Figure 2: The accuracy (attack success rate) vs. perturbation magnitude of the ResNet-20-PP model against untargeted white-box and black-box attacks.

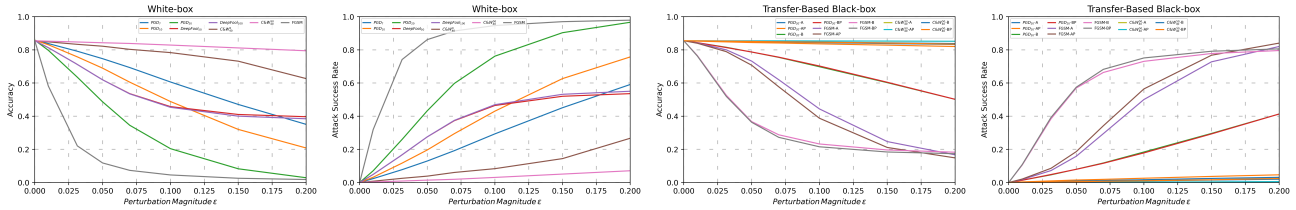

Figure 3: The accuracy (attack success rate) vs. perturbation magnitude of the ResNet-56-PP model against untargeted white-box and black-box attacks.

"Benign" refers to the accuracy of unperturbed examples.

We observe that our combined approach, within this context, neither fundamentally improves robustness nor degrades it. There is a minor performance drop of accuracy on benign examples that further decreased with the increase of the model capacity. However, this is mainly confined to the benign accuracy, whereas robustness accuracy mostly improved with increasing capacity, as is expected [22]. Fluctuation may result from both FGSM and PGD starting with a random perturbation. Nevertheless, our interpretation of the accuracy drop in benign images is that correlation between benign and adversarial examples is harder to establish, with fixed α and h, due to the perturbation being small, distinct, and equally treated by the inhibitory effect. Therefore, the weight space learns a different representation that might map to a darker image. Furthermore, there is a negligible performance difference between the two approaches regarding the C&W attack, although the standard approach has a slight edge over ours. However, we obtain peculiar results regarding the defense against attacks generated by the FGSM adversary. Our approach achieves considerably low robustness accuracies compared to stronger, iterative attacks. The initial intuition is that the push-pull layer somehow produces "obfuscated gradients" [34]. These are unusable gradients leading to inferior attacks caused either by shattered gradients, vanishing/exploding gradients, or stochastic gradients (not applicable). However, we can rule obfuscation out for two reasons: (1) models trained with the standard approach exhibit the same, even worse behavior, (2) iterative white-box attacks have a higher success rate (lower robustness) than transfer-based black-box attacks (by comparing Table 2 and Table 3). Consequently, the poor performance must result from properties in the surface of loss caused by ResNet architectures.

Black-box Robustness. Here, adversarial examples for the transfer-based block-box attacks are crafted on a surrogate model that is a ResNet-56. We utilize four different variants of the model, where:

1. It is trained only on benign examples without a push-pull layer.
2. Exactly like the first but with a push-pull layer.
3. Adversarially trained without push-pull layer (Adversarial).
4. Like the third but with a push-pull layer (Adversarial-PP).

Furthermore, we utilize the same attacks as in the white-box setting, except we only use the strongest PGD. Additionally, the C&W attack generates adversarial examples with a confidence of 50 for better transferability [25]. The black-box robustness of both defense models is reported in Table 3.

We can observe an increase in robustness against iterative attacks by comparing the black-box results with those obtained in the white-box setting. Therefore, as mentioned above, the accuracy improvement against PGD and C&W further suggests that obfuscated gradients do not cause robustness. However, we conduct an additional check using gradient-free, score-based algorithms to support this argument further. We employed multiple SPSA and NES runs, where we iterated by 100 iterations until 1000 and then continued with 1000 steps per round until 5000. A considerable robustness drop occurred after reaching an iteration count in the thousands. Therefore, concluding that robustness does not result from masking gradient information. We also report an increased resistance against adversarial examples generated by the FGSM adversary. Our approach yields better results than the counterpart except when generated from a standard ResNet-56 with ResNet-20 as a target. Nonetheless, the unreasonable effectiveness of FGSM is rather peculiar and needs further investigation. Regardless, the standard PGD-AT method achieves slightly higher robustness in most source and target combinations.

4.3.2 *Push-Pull Efficacy With Higher Perturbation Magnitudes*

Here we investigate the robustness-efficacy of our proposed method against various adversaries with increasing perturbation magnitudes, which is a continuation of subsection 4.3.1. We restrict the architectures solely to ResNet-20 (Figure 2) and ResNet-56 (Figure 3), given the negligent performance differences among all models (see Table 2). We denote ResNet models, acting as a source, without a

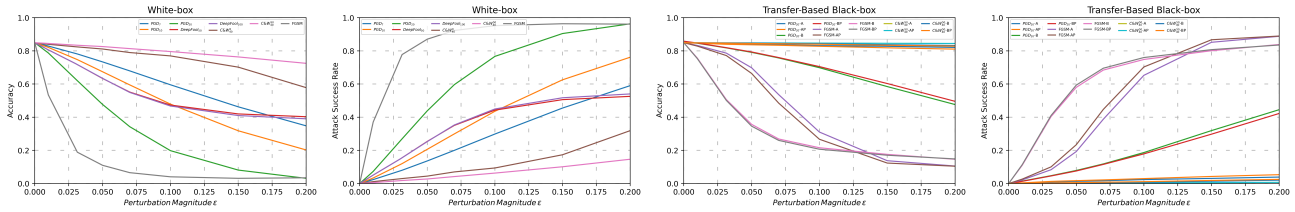

Figure 4: The accuracy (attack success rate) vs. perturbation magnitude of the ResNet-20-PP model with trained α against untargeted white-box and black-box attacks.

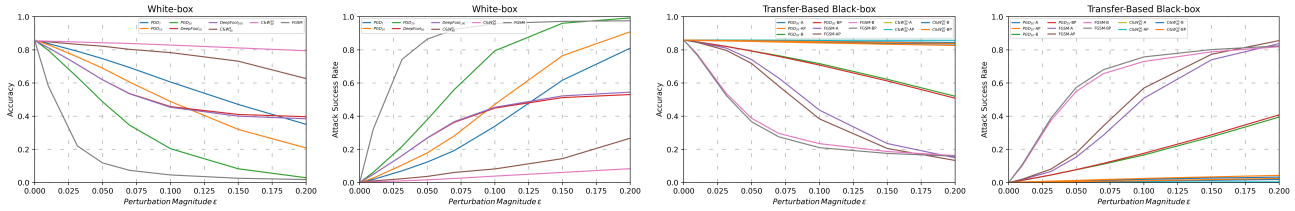

Figure 5: The accuracy (attack success rate) vs. perturbation magnitude of the ResNet-56-PP model with trained α against untargeted white-box and black-box attacks.

push-pull layer as A and if they are adversarially trained with B, for the robustness curves in the transfer-based black-box attacks. Furthermore, we append a P to indicate that a model contains a push-pull layer as its first layer.

White-box Robustness. The relative robustness of our proposed method is the highest against perturbations based on constrained optimizations. We hypothesize that this property is due to the ability of the push-pull layer to smoothen the decision boundaries, thereby making it increasingly harder to find effective, minimal perturbations. To exclude sub-optimal and non-converging attack parameters, we further doubled the number of iterations (and confidence to 80 for C&W) and observed an increase in the attack success rate. However, the observations remain the same as in Figure 2 & Figure 3 - that is, relatively high robustness across the perturbation magnitudes, where DeepFool is the most effective. More interesting is the phenomenon, within the context of the C&W attack, that the robustness increases as the confidence increases (albeit minimal), and only the number of iterations determine the effectiveness. Furthermore, if we consider PGD, we can observe that the strongest (with 20 iterations) drives the robustness to 0 and the success rate to 100%. This is to be expected as the adversarial examples become increasingly incomprehensible and, by definition, should increase the success rate [33]. The fact that it does converge to zero further suggests that robustness is not the product of masking gradient information. Lastly, we still observe the unreasonable effectiveness of FGSM for unknown reasons that have not been further explored within this study.

Black-box Robustness. The setup here is equivalent to the procedure in subsection 4.3.1. That is, we employ independently trained surrogate models based on the ResNet-56 architecture and transfer generated adversarial examples to ResNet-20 (Figure 2 and ResNet-56 (Figure 3) models adversarially trained with a push-pull layer.

We make similar findings as in the white-box setting: the defense is most robust against constrained-based optimizations. However, the roles have reversed in that adversarial examples generated with high confidence transfer better when generated on adversarially trained models (both B and BP) compared to the models trained only on benign examples. Furthermore, higher perturbations degrade the accuracy and increase the attack success rate, but, except for adversarial examples generated by FGSM, all the attacks have a success rate of well under 60%. Consequently, the attacks remain inferior to their white-box counterpart.

Table 4: Benign accuracy of ResNet models adversarially trained with the inhibition strength α as a trainable parameter.

	ResNet-20	ResNet-32	ResNet-44	ResNet-56
Accuracy	84.93	86.31	85.46	85.94

4.3.3 *Fine-Tuning Inhibition Strength*

The strength of the push-pull layer as a general-purpose defense is its ability to extract features of interest despite corruption while introducing only a negligible computational overhead [19]. It is a consequence of deriving the inhibitory response (pull) from the excitation (push) where only the push kernel inhabits trainable parameters. However, its implementation incorporates the pull response during back-propagation and, consequently, encodes a representation conducive to selectivity. The models investigated so far in subsections 4.3.1 and 4.3.2 employed only fixed hyperparameters: a fixed inhibition strength α and a fixed scaling factor h. We hypothesized that this inflexibility might result, for instance, in an increasing difficulty to extract a correlation between benign and adversarial examples. Therefore, here, we intuitively investigate whether training the inhibition strength α against an adversary fine-tunes the inhibition response to improve accuracy on benign examples and robustness in general.

Training settings. The settings around the training process are equivalent to the settings in subsection 4.3.1. The only difference is that α is a trainable parameter. Furthermore, we trained a ResNet-20, ResNet-32, ResNet-44, and a ResNet-56 for 160 epochs with a batch size of 128. However, we only present the smallest and biggest architecture here due to minor performance differences among the models.

Benign Accuracy. By comparing Tables 2 and 4, we can observe that models with α as a trainable parameter slightly improve accuracy over those with the inhibition strength fixed, except for the ResNet-20 model. However, they still do not reach the accuracy levels of the adversarial counterpart that trained without a push-pull layer. It begs the question of why that is. We think of two reasons: (1) fluctuations arise from random perturbations in the PGD_7 adversary, (2) inhibition may change the representation of the remaining weight space during training, for which we do not account. We further discuss (1) and (2) in Section 5. Nevertheless, we can interpret the improvement of the benign accuracy as given in Table 4 by inspecting the trained inhibitions strengths. We observed that the α parameters of all models are very close to zero, which matches the small magnitude of the perturbations. Therefore, training a model with a push-pull layer and the inhibition strength as a trainable parameter improves input signal selectivity.

White-box Robustness. A notable observation when comparing Figures 4 and 5 with Figures 2 and 3 is that models with trained inhibition strength perform slightly better on smaller perturbations against all PGD attacks. This result is to be expected since α is calibrated towards the adversary that generated the attack during training. However, this fine-tuning comes at the cost of generalization as models trained with fixed inhibition exhibit more robustness on higher perturbations. In contrast, we do not observe a loss of generality for higher ϵ budgets in optimized perturbations. On the one hand, the difference between the ResNet-20 models is minimal, whereas the model with fixed inhibition has slightly higher robustness. On the other hand, the Resnet-56 model with trained inhibition strength has a slight edge over the fixed one. Regardless, the result agrees with the smoothness property that the push-pull layer induces into the decision boundaries (see subsection 4.3.2). Lastly, FGSM remains considerably effective. However, gradient obfuscation is ruled out for the same reason as discussed in subsection 4.3.1.

Black-box Robustness. For the transfer-based black-box attacks, we differentiate between the source models with the same notation introduced in subsection 4.3.2. Furthermore, we employ the same surrogate models to generate adversarial examples.

Opposite to the findings in the white-box scenario, we report an increase in robustness (compared to Figures 2 & 3) against PGD attacks across all perturbation magnitudes (Figures 4 & 5). In the case of FGSM, we observe fluctuating results regarding the ResNet-20 models, whereas the ResNet-56 with trained inhibition strength exhibits a slight improvement over perturbations up to $\epsilon = 0.1$ (compared to the model with fixed α). For the C&W attack, we have that the ResNet-20 model performs negligibly worse than the model trained with fixed inhibition. In contrast, the ResNet-56 model is more resistant to transfer-based black-box attacks than its fixed counterpart. Overall, we interpret the behavior of the smaller model as a result of the randomness involved during training since it acts similarly in the white-box setting.

5 Discussion & Future Work

We demonstrated that the approach successfully translates robustness enhancing properties of the push-pull layer [19] to adversarial examples. However, we have made several discoveries that need further investigation to improve and consolidate this procedure as an effective method for robustness against adversarial attacks. The most notable and peculiar result is the effectiveness of FGSM against our proposed method, despite its robustness against iterative attacks. We repeatedly demonstrated that the efficacy is not a result (see subsections 4.3.1 and 4.3.3) of obfuscated gradients by referring to the inferiority of black-box attacks. Nevertheless, its cause or origin must be determined as robustness against single-step attacks is fundamental in this context [33, 22]. Furthermore, we have reported a slightly worse result than the standard approach regarding the accuracy of benign examples. We hypothesized that two reasons potentially cause this. First, we employ a PGD attack with a random start during training and, thus, may cause the fluctuation in robustness properties. Unfortunately, we do not seed the randomization in the attacks during training. Therefore, to investigate this hypothesis, either a model (with and without push-pull) is adversarially trained using PGD without a random start or seeding the attack to ensure consistency. The second hypothesis revolves around the push-pull layer itself and its impact on the remaining weight space of the CNN. A possible first attempt to investigate it is by learning the inhibition strength alpha (as in subsection 4.3.3) but freezing the rest of the weights. Following that, the remaining weight space is trained on benign examples while the push-pull layer is frozen.

Picking up on the last suggestion for investigating the impact of the push-pull layer on the remaining weight space during training, we can extrapolate this idea to improve potentially the input selectivity of our approach. In subsection 4.3.1, we argued that the small accuracy drop in benign examples was the inflexibility of the push-pull layer (with fixed hyperparameters) to consider the small perturbations. Consequently, we trained the inhibition strength α and reported an increased input selectivity (see subsection 4.3.3 and Table 4) and an improvement of robustness against perturbations within the trained ℓ_∞-ball under the white-box setting. However, the gain came at the cost of generalizability to higher perturbations against white-box attacks. Therefore, given that within this context we have for each kernel W_i an inhibition strength α_i in the push-pull layer, we can adversarially train each with the same adversary but different perturbation magnitudes, or train each parameter against a different attacker altogether from either the same or other pre-trained models. The *Ensemble Training Algorithm* [20] inspires the latter and could be further used to increase robustness against transfer-based black-box attacks.

An exciting observation not explicitly mentioned in the experimental study is the fast convergence rate of models with a push-pull layer during adversarial training. Compared to the standard approach, models with a push-pull layer always converged and at least two times faster than their counterparts (others did not converge at all within 160 epochs). Therefore, the regularization properties may prove useful in reducing the prohibitive complexity of adversarial training by combining it with, for instance, early stopping or integrating the model into other techniques improving the complexity [35, 36]. Consequently, adversarial training becomes a more practical and accessible approach for adversarial robustness. However, to capture its ability to improve convergence, it should be compared against regular regularization techniques (e.g., dropout), whether it holds for different architectures, more complex

images (e.g., Imagenet), or when other data augmentation techniques are applied.

6 Conclusion

We performed an extensive experimental study investigating the defense properties of an approach combining a biologically-inspired component called the push-pull layer with the adversarial training framework. The intuition for the method comes from the ability of the layer to extract features of interest despite distortion/corruption by exhibiting a form of response inhibition and extending it to adversarial examples. We demonstrated that this effect translates to adversarial examples by exposing it to multiple (untargeted) attacks with varying perturbation magnitudes under white-box and black-box settings. First, we established that the approach yields well-generalizing results comparable to the standard benchmark PGD-AT approach with fixed hyperparameters, especially against increasing perturbations. Then, by learning the parameter regulating the inhibition, we further found that the push-pull layer performs an increasing selectivity of input signals improving robustness against perturbations within the bounds of the perturbation magnitude, albeit at the cost of resistance against higher perturbations in white-box attacks. However, more investigation is required to uncover more of its properties and find explanations for the results discussed in this experimental study. Nevertheless, the push-pull layer exhibits properties that make it a promising avenue as a *general-purpose defense.*

7 Acknowledgements

My sincere gratitude goes towards my supervisor Nicola Strisciuglio for his continuous and insightful feedback, discussions on this experimental study, and other research topics.

8 References

[1] LeCun, B. Y. . H. G., Y. Deep learning. *Nature* **521**, 436–444 (2015). URL https://doi.org/10.1038/nature14539.

[2] Bojarski, M. *et al.* End to end learning for self-driving cars (2016). 1604.07316.

[3] Ackerman, E. How drive.ai is mastering autonomous driving with deep learning (2021).

[4] Apostolidis, K. D. & Papakostas, G. A. A survey on adversarial deep learning robustness in medical image analysis. *Electronics* **10** (2021). URL https://www.mdpi.com/2079-9292/10/17/2132.

[5] Dahl, G. E., Stokes, J. W., Deng, L. & Yu, D. Large-scale malware classification using random projections and neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 3422–3426 (2013).

[6] Yuan, X., He, P., Zhu, Q., Bhat, R. R. & Li, X. Adversarial examples: Attacks and defenses for deep learning. *CoRR* **abs/1712.07107** (2017). URL http://arxiv.org/abs/1712.07107. 1712.07107.

[7] Ren, K., Zheng, T., Qin, Z. & Liu, X. Adversarial attacks and defenses in deep learning. *Engineering* **6**, 346–360 (2020). URL https://www.sciencedirect.com/science/article/pii/S209580991930503X.

[8] Akhtar, N. & Mian, A. Threat of adversarial attacks on deep learning in computer vision: A survey. *CoRR* **abs/1801.00553** (2018). URL http://arxiv.org/abs/1801.00553. 1801.00553.

[9] Barreno, M., Nelson, B., Joseph, A. & Tygar, J. The security of machine learning. *Machine Learning* **81**, 121–148 (2010).

[10] Biggio, B. *et al.* Evasion attacks against machine learning at test time. *Lecture Notes in Computer Science* 387–402 (2013). URL http://dx.doi.org/10.1007/978-3-642-40994-3_25.

[11] Szegedy, C. *et al.* Intriguing properties of neural networks (2014). URL https://arxiv.org/abs/1312.6199. 1312.6199.

[12] Goodfellow, I. J., Shlens, J. & Szegedy, C. Explaining and harnessing adversarial examples (2015). URL https://arxiv.org/abs/1412.6572. 1412.6572.

[13] Papernot, N., McDaniel, P. & Goodfellow, I. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples (2016). URL https://arxiv.org/abs/1605.07277. 1605.07277.

[14] Ilyas, A. *et al.* Adversarial examples are not bugs, they are features (2019). URL https://arxiv.org/abs/1905.02175. 1905.02175.

[15] Bai, T., Luo, J., Zhao, J., Wen, B. & Wang, Q. Recent advances in adversarial training for adversarial robustness (2021). URL https://arxiv.org/abs/2102.01356. 2102.01356.

[16] Dziugaite, G. K., Ghahramani, Z. & Roy, D. M. A study of the effect of jpg compression on adversarial images (2016). URL https://arxiv.org/abs/1608.00853. 1608.00853.

[17] Papernot, N., McDaniel, P., Wu, X., Jha, S. & Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks (2016). URL https://arxiv.org/abs/1511.04508. 1511.04508.

[18] Yang, Y.-Y., Rashtchian, C., Zhang, H., Salakhutdinov, R. & Chaudhuri, K. A closer look at accuracy vs. robustness (2020). URL https://arxiv.org/abs/2003.02460. 2003.02460.

[19] Strisciuglio, N., Lopez-Antequera, M. & Petkov, N. Enhanced robustness of convolutional networks with a push–pull inhibition layer. *Neural Computing and Applications* (2020). URL https://doi.org/10.1007/s00521-020-04751-8.

[20] Tramèr, F. *et al.* Ensemble adversarial training: Attacks and defenses (2020). URL https://arxiv.org/abs/1705.07204. 1705.07204.

[21] Shaham, U., Yamada, Y. & Negahban, S. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing* **307**, 195–204 (2018). URL http://dx.doi.org/10.1016/j.neucom.2018.04.027.

[22] Madry, A., Makelov, A., Schmidt, L., Tsipras, D. & Vladu, A. Towards deep learning models resistant to adversarial attacks (2019). URL https://arxiv.org/abs/1706.06083. 1706.06083.

[23] Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O. & Frossard, P. Universal adversarial perturbations (2017). URL https://arxiv.org/abs/1610.08401. 1610.08401.

[24] Krizhevsky, A. Learning multiple layers of features from tiny images. Tech. Rep. (2009).

[25] Carlini, N. & Wagner, D. Towards evaluating the robustness of neural networks (2017). URL https://arxiv.org/abs/1608.04644. 1608.04644.

[26] Moosavi-Dezfooli, S.-M., Fawzi, A. & Frossard, P.

Deepfool: a simple and accurate method to fool deep neural networks (2016). URL https://arxiv.org/abs/1511.04599. 1511.04599.

[27] Uesato, J., O'Donoghue, B., van den Oord, A. & Kohli, P. Adversarial risk and the dangers of evaluating against weak attacks (2018). URL https://arxiv.org/abs/1802.05666. 1802.05666.

[28] Ilyas, A., Engstrom, L., Athalye, A. & Lin, J. Black-box adversarial attacks with limited queries and information (2018). URL https://arxiv.org/abs/1804.08598. 1804.08598.

[29] Paszke, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. In Wallach, H. *et al.* (eds.) *Advances in Neural Information Processing Systems 32*, 8024–8035 (Curran Associates, Inc., 2019).

[30] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition (2015). URL https://arxiv.org/abs/1512.03385. 1512.03385.

[31] Rauber, J., Brendel, W. & Bethge, M. Foolbox: A python toolbox to benchmark the robustness of machine learning models (2018). URL https://arxiv.org/abs/1707.04131. 1707.04131.

[32] Dong, Y. *et al.* Benchmarking adversarial robustness on image classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 318–328 (2020).

[33] Carlini, N. *et al.* On evaluating adversarial robustness (2019). URL https://arxiv.org/abs/1902.06705. 1902.06705.

[34] Athalye, A., Carlini, N. & Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples (2018). URL https://arxiv.org/abs/1802.00420. 1802.00420.

[35] Andriushchenko, M. & Flammarion, N. Understanding and improving fast adversarial training (2020). 2007.02617.

[36] Shafahi, A. *et al.* Adversarial training for free! (2019). URL https://arxiv.org/abs/1904.12843. 1904.12843.