PERCEPTUAL ADVERSARIAL ROBUSTNESS: DEFENSE AGAINST UNSEEN THREAT MODELS

Cassidy Laidlaw  
University of Maryland  
claidlaw@umd.edu

Sahil Singla  
University of Maryland  
ssingla@cs.umd.edu

Soheil Feizi  
University of Maryland  
sfeizi@cs.umd.edu

ABSTRACT

A key challenge in adversarial robustness is the lack of a precise mathematical characterization of human perception, used in the definition of adversarial attacks that are imperceptible to human eyes. Most current attacks and defenses try to avoid this issue by considering restrictive adversarial threat models such as those bounded by $L_2$ or $L_\infty$ distance, spatial perturbations, etc. However, models that are robust against any of these restrictive threat models are still fragile against other threat models, i.e. they have poor generalization to unforeseen attacks. Moreover, even if a model is robust against the union of several restrictive threat models, it is still susceptible to other imperceptible adversarial examples that are not contained in any of the constituent threat models. To resolve these issues, we propose adversarial training against the set of all imperceptible adversarial examples. Since this set is intractable to compute without a human in the loop, we approximate it using deep neural networks. We call this threat model the neural perceptual threat model (NPTM); it includes adversarial examples with a bounded neural perceptual distance (a neural network-based approximation of the true perceptual distance) to natural images. Through an extensive perceptual study, we show that the neural perceptual distance correlates well with human judgements of perceptibility of adversarial examples, validating our threat model. Under the NPTM, we develop novel perceptual adversarial attacks and defenses. Because the NPTM is very broad, we find that Perceptual Adversarial Training (PAT) against a perceptual attack gives robustness against many other types of adversarial attacks. We test PAT on CIFAR-10 and ImageNet-100 against five diverse adversarial attacks: $L_2$, $L_\infty$, spatial, recoloring, and JPEG. We find that PAT achieves state-of-the-art robustness against the union of these five attacks—more than doubling the accuracy over the next best model—without training against any of them. That is, PAT generalizes well to unforeseen perturbation types. This is vital in sensitive applications where a particular threat model cannot be assumed, and to the best of our knowledge, PAT is the first adversarial training defense with this property.

1 INTRODUCTION

Many modern machine learning algorithms are susceptible to adversarial examples: carefully crafted inputs designed to fool models into giving incorrect outputs (Biggio et al., 2013; Szegedy et al., 2014; Kurakin et al., 2016a; Xie et al., 2017). Much research has focused on increasing classifiers' robustness against adversarial attacks (Goodfellow et al., 2015; Madry et al., 2018; Zhang et al., 2019a). However, existing adversarial defenses for image classifiers generally consider simple threat models. An adversarial threat model defines a set of perturbations that may be made to an image in order to produce an adversarial example. Common threat models include $L_2$ and $L_\infty$ threat models, which constrain adversarial examples to be close to the original image in $L_2$ or $L_\infty$ distances. Some work has proposed additional threat models which allow spatial perturbations (Engstrom et al., 2017; Wong et al., 2019; Xiao et al., 2018), recoloring (Hosseini and Poovendran, 2018; Laidlaw and Feizi, 2019; Bhattad et al., 2019), and other modifications (Song et al., 2018; Zeng et al., 2019) of an image. There are multiple issues with these unrealistically constrained adversarial threat models. First, hardening against one threat model assumes that an adversary will only attempt attacks within that threat model. Although a classifier may be trained to be robust against $L_\infty$ attacks, for instance,
We present adversarial attacks and defenses for the proposed NPTM. Generating adversarial examples $x$ which we call the perceptibility threshold, images (Figure 1). Some other threat models, such as adversarial patch attacks (Brown et al., 2018), may neural perceptual distance $d$ The true perceptual distance $d$ for the NPTM, $P$ is bounded by the neural perceptual distance is difficult compared to generating $L_p$ adversarial examples because of the complexity and non-convexness of the constraint. However, we develop two attacks for the NPTM, Perceptual Projected Gradient Descent (PPGD) and Lagrangian Perceptual Attack (LPA) (see Section 4 for details). We find that LPA is by far the strongest adversarial attack at a given level of perceptibility (see Figure 4), reducing the most robust classifier studied to only 2.4%
Adversarial robustness has been studied extensively for $L_p$, spatial, etc., models (Goodfellow et al., 2015; Carlini and Wagner, 2017; Madry et al., 2018) and non-$L_p$ threat models such as spatial perturbations (Engstrom et al., 2017; Xia et al., 2018; Wong et al., 2019), recoloring of an image (Hosseini and Poovendran, 2018; Laidlaw and Feizi, 2019; Bhattad et al., 2019), and perturbations in the frequency domain (Kang et al., 2019). The most popular known adversarial defense for these threat models is adversarial training (Kurakin et al., 2016b; Madry et al., 2018; Zhang et al., 2019a) where a neural network is trained to minimize the worst-case loss in a region around the input. Recent evaluation methodologies such as Unforeseen Attack Robustness (UAR) (Kang et al., 2019) and the Unrestricted Adversarial Examples challenge (Brown et al., 2018) have raised the problem of finding an adversarial defense which gives good robustness under more general threat models. Sharif et al. (2018) conduct a perceptual study showing that $L_p$ threat models are a poor approximation of the perceptual threat model. Dunn et al. (2020) and Xu et al. (2020) have developed adversarial attacks that manipulate higher-level, semantic features. Jin and Rinard (2020) train with a manifold regularization term, which gives some robustness to unseen perturbation types. Stutz et al. (2020) also propose a method which gives robustness against unseen perturbation types, but requires rejecting (abstaining on) some inputs.

Perceptual similarity Two basic similarity measures for images are the $L_2$ distance and the Peak Signal-to-Noise Ratio (PSNR). However, these similarity measures disagree with human vision on perturbations such as blurring and spatial transformations, which has motivated others including SSIM (Wang et al., 2004), MS-SSIM (Wang et al., 2003), CW-SSIM (Sampat et al., 2009), HDR-VDP-2 (Mantiuk et al., 2011) and LPIPS (Zhang et al., 2018). MAD competition (Wang and Simoncelli, 2004; MS-SSIM (Wang et al., 2003)), and perturbations in the frequency domain (Kang et al., 2019). The most popular known adversarial defense for these threat models is adversarial training (Kurakin et al., 2016b; Madry et al., 2018; Zhang et al., 2019a) where a neural network is trained to minimize the worst-case loss in a region around the input. Recent evaluation methodologies such as Unforeseen Attack Robustness (UAR) (Kang et al., 2019) and the Unrestricted Adversarial Examples challenge (Brown et al., 2018) have raised the problem of finding an adversarial defense which gives good robustness under more general threat models. Sharif et al. (2018) conduct a perceptual study showing that $L_p$ threat models are a poor approximation of the perceptual threat model. Dunn et al. (2020) and Xu et al. (2020) have developed adversarial attacks that manipulate higher-level, semantic features. Jin and Rinard (2020) train with a manifold regularization term, which gives some robustness to unseen perturbation types. Stutz et al. (2020) also propose a method which gives robustness against unseen perturbation types, but requires rejecting (abstaining on) some inputs.

Perceptual adversarial robustness Although LPIPS was previously proposed, it has mostly been used for development and evaluation of generative models (Huang et al., 2018; Karras et al., 2019; Jordan et al., 2019) first explored quantifying adversarial distortions with LPIPS distance. However, to the best of our knowledge, we are the first to apply a more accurate perceptual distance to the

---

1 Code and data can be downloaded at [https://github.com/cassidylaidlaw/perceptual-advex](https://github.com/cassidylaidlaw/perceptual-advex)
We propose attack methods which attempt to find an adversarial example with small perceptual

$$D_{\text{LPIS}}(x) = \|\phi(x) - \phi(x_0)\|_2$$

where

The LPIPS distance

$$d$$

maximizing a loss function

more complex than

$$L$$

is more difficult than that of standard

$$L$$

distortion. Developing adversarial attacks that utilize the proposed neural perceptual threat model

$$\forall x \in X$$

we note that other neural perceptual distances can also

be used in our attacks and defenses.

Let $g : X \rightarrow Y$ be a convolutional image classifier network defined on images $x \in X$. Let $g$ have $L$

layers, and let the internal activations (outputs) of the $l$-th layer of $g(x)$ for an input $x$ be denoted as

$g_l(x)$. Zhang et al. (2018) have found that normalizing and then comparing the internal activations

of convolutional neural networks correlates well with human similarity judgements. Thus, the first

step in calculating the LPIPS distance using the network $g(\cdot)$ is to normalize the internal activations

across the channel dimension such that the $L_2$ norm over channels at each pixel is one. Let $\hat{g}_l(x)$

denote these channel-normalized activations at the $l$-th layer of the network. Next, the activations

are normalized again by layer size and flattened into a single vector

$$\phi(x) \equiv \left(\frac{\hat{g}_1(x)}{w_1h_1}, \ldots, \frac{\hat{g}_L(x)}{w_Lh_L}\right)$$

where $w_l$ and $h_l$ are the width and height of the activations in layer $l$, respectively. The function

$\phi : X \rightarrow A$ thus maps the inputs $x \in X$ of the classifier $g(\cdot)$ to the resulting normalized, flattened

internal activations $\phi(x) \in A$, where $A \subseteq \mathbb{R}^m$ refers to the space of all possible resulting activations.

The LPIPS distance $d(x_1, x_2)$ between images $x_1$ and $x_2$ is then defined as:

$$d(x_1, x_2) \equiv \|\phi(x_1) - \phi(x_2)\|_2.$$ (1)

In the original LPIPS implementation, Zhang et al. (2018) learn weights to apply to the normalized activations based on a dataset of human perceptual judgements. However, they find that LPIPS is a good surrogate for human vision even without the additional learned weights; this is the version we use since it avoids the need to collect such a dataset.

Now let $f : X \rightarrow Y$ be a classifier which maps inputs $x \in X$ to labels $f(x) \in Y$. $f(\cdot)$ could be the same as $g(\cdot)$, or it could be a different network; we experiment with both. For a given natural input $x$ with the true label $y$, a neural perceptual adversarial example with a perceptibility bound $\epsilon$ is an input $\tilde{x} \in X$ such that $x$ must be perceptually similar to $x$ but cause $f$ to misclassify:

$$f(\tilde{x}) \neq y \quad \text{and} \quad d(x, \tilde{x}) = \|\phi(x) - \phi(\tilde{x})\|_2 \leq \epsilon.$$ (2)

4 Perceptual Adversarial Attacks

We propose attack methods which attempt to find an adversarial example with small perceptual distortion. Developing adversarial attacks that utilize the proposed neural perceptual threat model is more difficult than that of standard $L_p$ threat models, because the LPIPS distance constraint is more complex than $L_p$ constraints. In general, we find an adversarial example that satisfies (2) by maximizing a loss function $L$ within the LPIPS bound. The loss function we use is similar to the

Original | Self-bd. LPA | External-bd. LPA | Original | Self-bd. LPA | External-bd. LPA
--- | --- | --- | --- | --- | ---
![Image](image1.png) | ![Image](image2.png) | ![Image](image3.png) | ![Image](image4.png) | ![Image](image5.png) | ![Image](image6.png)

Figure 3: Adversarial examples generated using self-bounded and externally-bounded LPA perceptual adversarial attack (Section 4) with a large bound. Original images are shown in the left column and magnified differences from the original are shown to the right of the examples. See also Figure 7.

problem of improving adversarial robustness. As we show, adversarial defenses based on $L_2$ or $L_\infty$ attacks are unable to generalize to a more diverse threat model. Our method, PAT, is the first adversarial training method we know of that can generalize to unforeseen threat models without rejecting inputs.

3 Neural Perceptual Threat Model (NPTM)

Since the true perceptual distance between images cannot be efficiently computed, we use approximations of it based on neural networks, i.e. neural perceptual distances. In this paper, we focus on the LPIPS distance (Zhang et al., 2018) while we note that other neural perceptual distances can also be used in our attacks and defenses.

Published as a conference paper at ICLR 2021
margin loss from Carlini and Wagner (2017), defined as
\[ L(f(x), y) \triangleq \max_{i \neq y} (z_i(x) - z_y(x)), \]
where \( z_i(x) \) is the \( i \)-th logit output of the classifier \( f(\cdot) \). This gives the constrained optimization
\[
\max_x L(f(\tilde{x}), y) \quad \text{subject to} \quad d(x, \tilde{x}) = \|\phi(x) - \phi(\tilde{x})\|_2 \leq \epsilon. \tag{3}
\]
Note that in this attack problem, the classifier network \( f(\cdot) \) and the LPIPS network \( g(\cdot) \) are fixed. These two networks could be identical, in which case the same network that is being attacked is used to calculate the LPIPS distance that bounds the attack; we call this a "self-bounded" attack. If a different network is used to calculate the LPIPS bound, we call it an "externally-bounded" attack.

Based on this formulation, we propose two perceptual attack methods, Perceptual Projected Gradient Descent (PPGD) and Lagrangian Perceptual Attack (LPA). See Figures 3 and 7 for sample results.

**Perceptual Projected Gradient Descent (PPGD)** The first of our two attacks is analogous to the PGD (Madry et al., 2018) attacks used for \( L_p \) threat models. In general, these attacks consist of iteratively performing two steps on the current adversarial example candidate: (a) taking a step of a certain size under the given distance that maximizes a first-order approximation of the misclassification loss, and (b) projecting back onto the feasible set of the threat model.

Identifying the ideal first-order step is easy in \( L_2 \) and \( L_\infty \) threat models; it is the gradient of the loss function and the sign of the gradient, respectively. However, computing this step is not straightforward with the LPIPS distance, because the distance metric itself is defined by a neural network. Following [3], we desire to find a step \( \delta \) to maximize \( L(f(x + \delta), y) \) such that \( d(x + \delta, x) = \|\phi(x + \delta) - \phi(x)\|_2 \leq \eta \), where \( \eta \) is the step size. Let \( \tilde{f}(x) := L(f(x), y) \) for an input \( x \in X \). Let \( J \) be the Jacobian of \( \phi(\cdot) \) at \( x \) and \( \hat{f} \) be the gradient of \( \tilde{f}(\cdot) \) at \( x \). Then, we can approximate (3) using a first-order Taylor’s approximation of \( \phi \) and \( \tilde{f} \) as follows:
\[
\max_\delta \tilde{f}(x) + (\hat{f})^\top \delta \quad \text{subject to} \quad \|J\delta\|_2 \leq \eta. \tag{4}
\]
We show that this constrained optimization can be solved in a closed form:

**Lemma 1.** Let \( J^+ \) denote the pseudoinverse of \( J \). Then the solution to (4) is given by
\[
\delta^* = \eta \frac{(J^\top J)^{-1}(\hat{f})}{\|((J^+)^\top (\hat{f}))\|_2}.
\]
See Appendix A.1 for the proof. This solution is still difficult to efficiently compute, since calculating \( J^+ \) and inverting \( J^\top J \) are computationally expensive. Thus, we approximately solve for \( \delta^* \) using the conjugate gradient method; see Appendix A.1 for details.

Perceptual PGD consists of repeatedly finding first-order optimal \( \delta^* \) to add to the current adversarial example \( \tilde{x} \) for a number of steps. Following each step, if the current adversarial example \( \tilde{x} \) is outside the LPIPS bound, we project \( \tilde{x} \) back onto the threat model such that \( d(\tilde{x}, x) \leq \epsilon \). The exact projection is again difficult due to the non-convexity of the feasible set. Thus, we solve it approximately with a technique based on Newton’s method; see Algorithm 4 in the appendix.

**Lagrangian Perceptual Attack (LPA)** The second of our two attacks uses a Lagrangian relaxation of the attack problem (3) similar to that used by Carlini and Wagner (2017) for constructing \( L_2 \) and \( L_\infty \) adversarial examples. We call this attack the Lagrangian Perceptual Attack (LPA). To derive the attack, we use the following Lagrangian relaxation of (3):
\[
\max_x L(f(\tilde{x}), y) - \lambda \max \left( 0, \|\phi(\tilde{x}) - \phi(x)\|_2 - \epsilon \right). \tag{5}
\]
The perceptual constraint cost, multiplied by \( \lambda \) in (5), is designed to be 0 as long as the adversarial example is within the allowed perceptual distance; i.e. \( d(\tilde{x}, x) \leq \epsilon \); once \( d(\tilde{x}, x) > \epsilon \), however, it increases linearly by the LPIPS distance from the original input \( x \). Similar to \( L_2 \) attacks of Carlini and Wagner (2017), we adaptively change \( \lambda \) to find an adversarial example within the allowed perceptual distance; see Appendix A.2 for details.

5 **Perceptual Adversarial Training (PAT)**

The developed perceptual attacks can be used to harden a classifier against a variety of adversarial attacks. The intuition, which we verify in Section 7, is that if a model is robust against neural
perceptual attacks, it can demonstrate an enhanced robustness against other types of unforeseen adversarial attacks. Inspired by adversarial training used to robustify models against $L_p$ attacks, we propose a method called Perceptual Adversarial Training (PAT).

Suppose we wish to train a classifier $f(\cdot)$ over a distribution of inputs and labels $(x, y) \sim D$ such that it is robust to the perceptual threat model with bound $\epsilon$. Let $L_{ce}$ denote the cross entropy (negative log likelihood) loss and suppose the classifier $f(\cdot)$ is parameterized by $\theta_f$. Then, PAT consists of optimizing $f(\cdot)$ in a manner analogous to $L_p$ adversarial training (Madry et al., 2018):

$$\min_{\theta_f} \mathbb{E}_{(x, y) \sim D} \left( \max_{d(x, \tilde{x}) \leq \epsilon} L_{ce}(f(\tilde{x}), y) \right).$$  \hspace{1cm}  \text{(6)}

The training formulation attempts to minimize the worst-case loss within a neighborhood of each training point $x$. In PAT, the neighborhood is bounded by the LPIPS distance. Recall that the LPIPS distance is itself defined based on a particular neural network classifier. We refer to the normalized, flattened activations of the network used to define LPIPS as $\phi(\cdot)$ and $\theta_\phi$ to refer to its parameters. We explore two variants of PAT differentiated by the choice of the network used to define $\phi(\cdot)$. In externally-bounded PAT, a separate, pretrained network is used to calculate $\phi(\cdot)$, the LPIPS distance $d(\cdot, \cdot)$. In self-bounded PAT, the same network which is being trained for classification is used to calculate the LPIPS distance, i.e. $\theta_\phi \subseteq \theta_f$. Note that in self-bounded PAT the definition of the LPIPS distance changes during the training as the classifier is optimized.

The inner maximization in (6) is intractable to compute exactly. However, we can use the perceptual attacks developed in Section 4 to approximately solve it. Since the inner maximization must be solved repeatedly during the training process, we use an inexpensive variant of the LPA attack called Fast-LPA. In contrast to LPA, Fast-LPA does not search over values of $\lambda$. It also does not include a projection step at the end of the attack, which means it may sometimes produce adversarial examples outside the training bound. While this makes it unusable for evaluation, it is fine for training. Using Fast-LPA, PAT is nearly as fast as adversarial training; see Appendix A.3 and Algorithm 3 for details.

6 Perceptual Evaluation

We conduct a thorough perceptual evaluation of our NPTM and attacks to ensure that the resulting adversarial examples are imperceptible. We also compare the perceptibility of perceptual adversarial attacks to five narrow threat models: $L_\infty$ and $L_2$ attacks, JPEG attacks (Kang et al., 2019), spatially transformed adversarial examples (StAdv) (Xiao et al., 2018), and functional adversarial attacks (RecolorAdv) (Laidlaw and Feizi, 2019). The comparison allows us to determine if the LPIPS distance is a good surrogate for human comparisons of similarity. It also allows us to set bounds across threat models with approximately the same level of perceptibility.

To determine how perceptible a particular threat model is at a particular bound (e.g. $L_\infty$ attacks at $\epsilon = 8/255$), we perform an experiment based on just noticeable differences (JND). We show pairs of images to participants on Amazon Mechanical Turk (AMT), an online crowdsourcing platform. In each pair, one image is a natural image from ImageNet-100 and one image is an adversarial perturbation of the natural image, generated using the particular attack against a classifier hardened to that attack. One of the images, chosen randomly, is shown for one second, followed by a blank screen for 250ms, followed by the second image for one second. Then, participants must choose whether they believe the images are the same or different. This procedure is identical to that used by Zhang et al. (2018) to originally validate the LPIPS distance. We report the proportion of pairs for which participants report the images are “different” as the perceptibility of the attack. In addition to adversarial example pairs, we also include sentinel image pairs which are exactly the same; only 4.1% of these were annotated as “different.”

We collect about 1,000 annotations of image pairs for each of the three bounds for all five threat models, plus our PPGD and LPA attacks (14k annotations total for 2.8k image pairs). The three bounds for each attack are labeled as small, medium, and large; bounds with the same label have similar perceptibility across threat models (see Appendix D Table 4). The dataset of image pairs and associated annotations is available for use by the community.

To determine if the LPIPS threat model is a good surrogate for the perceptual threat model, we use various classifiers to calculate the LPIPS distance $d(\cdot, \cdot)$ between the pairs of images used in the perceptual study. For each classifier, we determine the correlation between the mean LPIPS distance
Figure 4: Results of the perceptual study described in Section 6 across five narrow threat models and our two perceptual attacks, each with three bounds. (a) The perceptibility of adversarial examples correlates well with the LPIPS distance (based on AlexNet) from the natural example. (b) The Lagrangian Perceptual Attack (LPA) and Perceptual PGD (PPGD) are strongest at a given perceptibility. Strength is the attack success rate against an adversarially trained classifier. (c) Correlation between the perceptibility of attacks and various distance measures: $L_2$, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) calculated using various architectures, trained and at initialization.

it assigns to image pairs from each attack and the perceptibility of that attack (Figure 4). We find that AlexNet (Krizhevsky et al., 2012), trained normally on ImageNet (Russakovsky et al., 2015), correlates best with human perception of these adversarial examples ($r = 0.94$); this agrees with Zhang et al. (2018) who also find that AlexNet-based LPIPS correlates best with human perception (Figure 4). A normally trained ResNet-50 correlates similarly, but not quite as well. Because AlexNet is the best proxy for human judgements of perceptual distance, we use it for all externally-bounded evaluation attacks. Note that even with an untrained network at initialization, the LPIPS distance correlates with human perception better than the $L_2$ distance. This means that even during the first few epochs of self-bounded PAT, the training adversarial examples are perceptually-aligned.

We use the results of the perceptual study to investigate which attacks are strongest at a particular level of perceptibility. We evaluate each attack on a classifier hardened against that attack via adversarial training, and plot the resulting success rate against the proportion of correct annotations from the perceptual study. Out of the narrow threat models, we find that $L_2$ attacks are the strongest for their perceptibility. However, our proposed PPGD and LPA attacks reduce a PAT-trained classifier to even lower accuracies (8.2% for PPGD and 0% for LPA), making it the strongest attack studied.

7 Experiments

We compare Perceptual Adversarial Training (PAT) to adversarial training against narrow threat models ($L_p$, spatial, etc.) on CIFAR-10 (Krizhevsky and Hinton, 2009) and ImageNet-100 (the subset of ImageNet (Russakovsky et al., 2015) containing every tenth class by WordNet ID order). We find that PAT results in classifiers with robustness against a broad range of narrow threat models. We also show that our perceptual attacks, PPGD and LPA, are strong against adversarial training with narrow threat models. We evaluate with externally-bounded PPGD and LPA (Section 4), using AlexNet to determine the LPIPS bound because it correlates best with human judgements (Figure 4). For $L_2$ and $L_\infty$ robustness evaluation we use AutoAttack (Croce and Hein, 2020), which combines four strong attacks, including two PGD variants and a black box attack, to give reliable evaluation.

Evaluation metrics For both datasets, we evaluate classifiers’ robustness to a range of threat models using two summary metrics. First, we compute the union accuracy against all narrow threat models ($L_\infty$, $L_2$, StAdv, ReColorAdv, and JPEG for ImageNet-100); this is the proportion of inputs for which a classifier is robust against all these attacks. Second, we compute the unseen mean accuracy, which is the mean of the accuracies against all the threat models not trained against; this measures how well robustness generalizes to other threat models.

CIFAR-10 We test ResNet-50s trained on the CIFAR-10 dataset with PAT and adversarial training (AT) against six attacks (see Table 2): $L_\infty$ and $L_2$ AutoAttack, StAdv (Xiao et al., 2018), ReCol-
Table 2: Accuracies against various attacks for models trained with adversarial training and Perceptual Adversarial Training (PAT) variants on CIFAR-10. Attack bounds are 8/255 for $L_{\infty}$, 1 for $L_2$, 0.5 for PGGD/LPA (bounded with AlexNet), and the original bounds for StAdv/ReColorAdv. Manifold regularization is from Jin and Rinard (2020). See text for explanation of all terms.

<table>
<thead>
<tr>
<th>Training</th>
<th>Union mean</th>
<th>Unseen mean</th>
<th>Narrow threat models</th>
<th>NPTM mean</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clean</td>
<td>$L_{\infty}$</td>
<td>$L_2$</td>
<td>StAdv</td>
</tr>
<tr>
<td>Normal</td>
<td>0.0</td>
<td>0.1</td>
<td>94.8</td>
<td>0.0</td>
</tr>
<tr>
<td>AT $L_{\infty}$</td>
<td>1.0</td>
<td>19.6</td>
<td>86.8</td>
<td>49.0</td>
</tr>
<tr>
<td>TRADES $L_{\infty}$</td>
<td>4.6</td>
<td>23.3</td>
<td>84.9</td>
<td>52.5</td>
</tr>
<tr>
<td>AT $L_2$</td>
<td>4.0</td>
<td>25.3</td>
<td>85.0</td>
<td>39.5</td>
</tr>
<tr>
<td>AT StAdv</td>
<td>0.0</td>
<td>1.4</td>
<td>86.2</td>
<td>0.1</td>
</tr>
<tr>
<td>AT ReColorAdv</td>
<td>0.0</td>
<td>3.1</td>
<td>93.4</td>
<td>8.5</td>
</tr>
<tr>
<td>AT all (random)</td>
<td>0.7</td>
<td>—</td>
<td>85.2</td>
<td>22.0</td>
</tr>
<tr>
<td>AT all (average)</td>
<td>14.7</td>
<td>—</td>
<td>86.8</td>
<td>39.9</td>
</tr>
<tr>
<td>AT all (maximum)</td>
<td>21.4</td>
<td>—</td>
<td>84.0</td>
<td>25.7</td>
</tr>
<tr>
<td>Manifold reg.</td>
<td>21.2</td>
<td>36.2</td>
<td>72.1</td>
<td>36.8</td>
</tr>
<tr>
<td>PAT-self</td>
<td>21.9</td>
<td>45.6</td>
<td>82.4</td>
<td>30.2</td>
</tr>
<tr>
<td>PAT-AlexNet</td>
<td><strong>27.8</strong></td>
<td><strong>48.5</strong></td>
<td>71.6</td>
<td>28.7</td>
</tr>
</tbody>
</table>

Table 3: Comparison of adversarial training against narrow threat models and Perceptual Adversarial Training (PAT) on ImageNet-100. Accuracies are shown against seven attacks with the medium bounds from Table 4. PAT greatly improves accuracy (33% vs 12%) against the union of the narrow threat models despite not training against any of them. See text for explanation of all terms.

<table>
<thead>
<tr>
<th>Training</th>
<th>Union mean</th>
<th>Unseen mean</th>
<th>Narrow threat models</th>
<th>NPTM mean</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clean</td>
<td>$L_{\infty}$</td>
<td>$L_2$</td>
<td>JPEG</td>
</tr>
<tr>
<td>Normal</td>
<td>0.0</td>
<td>0.1</td>
<td>89.1</td>
<td>0.0</td>
</tr>
<tr>
<td>$L_{\infty}$</td>
<td>0.5</td>
<td>11.3</td>
<td>81.7</td>
<td>55.7</td>
</tr>
<tr>
<td>$L_2$</td>
<td>12.3</td>
<td>31.5</td>
<td>75.3</td>
<td>46.1</td>
</tr>
<tr>
<td>JPEG</td>
<td>0.1</td>
<td>7.4</td>
<td>84.8</td>
<td>13.7</td>
</tr>
<tr>
<td>StAdv</td>
<td>0.6</td>
<td>2.1</td>
<td>77.1</td>
<td>2.6</td>
</tr>
<tr>
<td>ReColorAdv</td>
<td>0.0</td>
<td>0.1</td>
<td>90.1</td>
<td>0.2</td>
</tr>
<tr>
<td>All (random)</td>
<td>0.9</td>
<td>—</td>
<td>78.6</td>
<td>38.3</td>
</tr>
<tr>
<td>PAT-self</td>
<td>32.5</td>
<td>46.4</td>
<td>72.6</td>
<td>45.0</td>
</tr>
<tr>
<td>PAT-AlexNet</td>
<td>25.5</td>
<td>44.7</td>
<td>75.7</td>
<td>46.8</td>
</tr>
</tbody>
</table>

orAdv (Laidlaw and Feizi, 2019), and PPGD and LPA. This allows us to determine if PAT gives robustness against a range of adversarial attacks. We experiment with using various models to calculate the LPIPS distance during PAT. We try using the same model both for classification and to calculate the LPIPS distance (self-bounded PAT). We also use AlexNet trained on CIFAR-10 prior to PAT (externally-bounded PAT). We find that PAT outperforms $L_p$ adversarial training and TRADES (Zhang et al., 2019a), improving the union accuracy from <5% to >20%, and nearly doubling mean accuracy against unseen threat models from 26% to 49%. Surprisingly, we find that PAT even outperforms threat-specific AT against StAdv and ReColorAdv; see Appendix F.5 for more details. Ablation studies of PAT are presented in Appendix F.1.

ImageNet-100 We compare ResNet-50s trained on the ImageNet-100 dataset with PAT and adversarial training (Table 3). Classifiers are tested against seven attacks at the medium bound from the perceptual study (see Section 6 and Appendix Table 4). Self- and externally-bounded PAT give similar results. Both produce more than double the next highest union accuracy and also significantly increase the mean robustness against unseen threat models by around 15%.

Perceptual attacks On both CIFAR-10 and ImageNet-100, we find that Perceptual PGD (PPGD) and Lagrangian Perceptual Attack (LPA) are the strongest attacks studied. LPA is the strongest,
Published as a conference paper at ICLR 2021

Distances of adv. examples

(a) $L_\infty$ distance
(b) $L_2$ distance
(c) LPIPS (self)
(d) LPIPS (AlexNet)

Figure 5: We generate samples via different adversarial attacks using narrow threat models in ImageNet-100 and measure their distances from natural inputs using $L_p$ and LPIPS metrics. The distribution of distances for each metric and threat model is shown as a violin plot. (a-b) $L_p$ metrics assign vastly different distances across perturbation types, making it impossible to train against all of them using $L_p$ adversarial training. (c-d) LPIPS assigns similar distances to similarly perceptible attacks, so a single training method, PAT, can give good robustness across different threat models.

Reducing the most robust classifier to 9.8% accuracy on CIFAR-10 and 2.4% accuracy on ImageNet-100. Also, models most robust to LPA in both cases are those that have the best union and unseen mean accuracies. This demonstrates the utility of evaluating against LPA as a proxy for adversarial robustness against a range of threat models. See Appendix E for further attack experiments.

Comparison to other defenses against multiple attacks Besides the baseline of adversarially training against a single attack, we also compare PAT to adversarially training against multiple attacks (Tramer and Boneh [2019]; Maini et al. [2019]). We compare three methods for multiple-attack training: choosing a random attack at each training iteration, optimizing the average loss across all attacks, and optimizing the maximum loss across all attacks. The latter two methods are very expensive, increasing training time by a factor equal to the number of attacks trained against, so we only evaluate these methods on CIFAR-10. As in Tramer and Boneh [2019], we find that the maximum loss strategy leads to the greatest union accuracy among the multiple-attack training methods. However, PAT performs even better on CIFAR-10, despite training against none of the attacks and taking one fourth of the time to train. The random strategy, which is the only feasible one on ImageNet-100, performs much worse than PAT. Even the best multiple-attack training strategies still fail to generalize to the unseen neural perceptual attacks, PPGD and LPA, achieving much lower accuracy than PAT.

On CIFAR-10, we also compare PAT to manifold regularization (MR) (Jin and Rinard [2020]), a non-adversarial training defense. MR gives union accuracy close to PAT-self, but much lower clean accuracy; for PAT-AlexNet, which gives similar clean accuracy to MR, the union accuracy is much higher.

Threat model overlap In Figure 2, we investigate how the sets of images vulnerable to $L_2$, spatial, and perceptual attacks overlap. Nearly all adversarial examples vulnerable to $L_2$ or spatial attacks are also vulnerable to LPA. However, there is only partial overlap between the examples vulnerable to $L_2$ and spatial attacks. This helps explain why PAT results in improved robustness against spatial attacks (and other diverse threat models) compared to $L_2$ adversarial training.

Why does PAT work better than $L_p$ adversarial training? In Figure 5, we give further explanation of why PAT results in improved robustness against diverse threat models. We generate many adversarial examples for the $L_\infty$, $L_2$, JPEG, StAdv, and ReColorAdv threat models and measure their distance from the corresponding natural inputs using $L_p$ distances and the neural perceptual distance, LPIPS. While $L_p$ distances vary widely, LPIPS gives remarkably comparable distances to different types of adversarial examples. Covering all threat models during $L_\infty$ or $L_2$ adversarial training would require using a huge training bound, resulting in poor performance. In contrast, PAT can obtain robustness against all the narrow threat models at a reasonable training bound.

Robustness against common corruptions In addition to evaluating PAT against adversarial examples, we also evaluate its robustness to random perturbations in the CIFAR-10-C and ImageNet-C datasets (Hendrycks and Dietterich [2019]). We find that PAT gives increased robustness (lower relative mCE) against these corruptions compared to adversarial training; see Appendix C for details.
8  CONCLUSION

We have presented attacks and defenses for the neural perceptual threat model (realized by the LPIPS distance) and shown that it closely approximates the true perceptual threat model, the set of all perturbations to natural inputs which fool a model but are imperceptible to humans. Our work provides a novel method for developing defenses against adversarial attacks that generalize to unforeseen threat models. Our proposed perceptual adversarial attacks and PAT could be extended to other vision algorithms, or even other domains such as audio and text.

ACKNOWLEDGMENTS

This project was supported in part by NSF CAREER AWARD 1942230, HR 00111990077, HR00112090132, HR001119S0026, NIST 60NANB20D134, AWS Machine Learning Research Award and Simons Fellowship on “Foundations of Deep Learning.”

REFERENCES


APPENDIX

A PERCEPTUAL ATTACK ALGORITHMS

A.1 PERCEPTUAL PGD

Recall from Section 4 that Perceptual PGD (PPGD) consists of repeatedly applying two steps: a first-order step in LPIPS distance to maximize the loss, followed by a projection into the allowed set of inputs. Here, we focus on the first-order step; see Appendix A.4 for how we perform projection onto the LPIPS ball.

We wish to solve the following constrained optimization for the step \( \delta \) given the step size \( \eta \) and current input \( x \):

\[
\max_{\delta} \mathcal{L}(f(x + \delta), y) \quad \text{subject to} \quad \|J\delta\|_2 \leq \eta \tag{7}
\]

Let \( \hat{f}(x) := \mathcal{L}(f(x), y) \) for an input \( x \in X \). Let \( J \) be the Jacobian of \( \phi(\cdot) \) at \( x \) and \( \nabla \hat{f} \) be the gradient of \( \hat{f}(\cdot) \) at \( x \).

**Lemma 1.** The first-order approximation of (7) is

\[
\max_{\delta} \hat{f}(x) + (\nabla \hat{f})^T \delta \quad \text{subject to} \quad \|J\delta\|_2 \leq \eta, \tag{8}
\]

and can be solved in closed-form by

\[
\delta^* = \frac{\eta (J^T J)^{-1} (\nabla \hat{f})}{\|(J^+)^T (\nabla \hat{f})\|_2},
\]

where \( J^+ \) is the pseudoinverse of \( J \).

**Proof.** We solve (8) using Lagrange multipliers. First, we take the gradient of the objective:

\[
\nabla \delta \left[ \hat{f}(x) + (\nabla \hat{f})^T \delta \right] = \nabla \hat{f}
\]

We can rewrite the constraint by squaring both sides to obtain

\[
\delta^T J^T J \delta - \epsilon^2 \leq 0
\]

Taking the gradient of the constraint gives

\[
\nabla \delta \left[ \delta^T J^T J \delta - \epsilon^2 \right] = 2J^T J \delta
\]

Now, we set one gradient as a multiple of the other and solve for \( \delta \):

\[
J^T J \delta = \lambda (\nabla \hat{f}) \tag{9}
\]

\[
\delta = \lambda (J^T J)^{-1} (\nabla \hat{f}) \tag{10}
\]

Substituting into the constraint from (8) gives

\[
\begin{align*}
\|J \delta\|_2 &= \eta \\
\|J \lambda (J^T J)^{-1} (\nabla \hat{f})\|_2 &= \eta \\
\lambda \|J (J^T J)^{-1} (\nabla \hat{f})\|_2 &= \eta \\
\lambda \|((J^T J)^{-1} J^T) (\nabla \hat{f})\|_2 &= \eta \\
\lambda \|(J^+)^T (\nabla \hat{f})\|_2 &= \eta
\end{align*}
\]

\[
\lambda = \frac{\eta}{\|(J^+)^T (\nabla \hat{f})\|_2}
\]

We substitute this value of \( \lambda \) into (10) to obtain

\[
\delta^* = \frac{\eta (J^T J)^{-1} (\nabla \hat{f})}{\|(J^+)^T (\nabla \hat{f})\|_2}. \tag{11}
\]
Solution with conjugate gradient method Calculating \([\nabla f(\tilde{x})]^{-1}\) directly is computationally intractable for most neural networks, since inverting \(J^T J\) and calculating the pseudoinverse of \(J\) are expensive. Instead, we approximate \(\delta^*\) by using the conjugate gradient method to solve the following linear system, based on (9):

\[
J^T J \delta = \nabla \hat{f}
\]  

\(\nabla \hat{f}\) is easy to calculate using backpropagation. The conjugate gradient method does not require calculating fully \(J^T J\); instead, it only requires the ability to perform matrix-vector products \(J^T Jv\) for various vectors \(v\).

We can approximate \(Jv\) using finite differences given a small, positive value \(h\):

\[
Jv \approx \frac{\phi(x + hv) - \phi(x)}{h}
\]

Then, we can calculate \(J^T Jv\) by introducing an additional variable \(u\) and using autograd:

\[
\nabla_u \left[ (\phi(x + u))^T Jv \right]_{u=0} = \left[ \left( \frac{d\phi}{du}(x + u) \right)^T Jv + (\phi(x + u))^T \frac{d}{du} Jv \right]_{u=0}
\]

\[
= \left[ \left( \frac{d\phi}{du}(x + u) \right)^T Jv + (\phi(x + u))^T 0 \right]_{u=0}
\]

\[
= \left( \frac{d\phi}{du}(x) \right)^T Jv = J^T Jv
\]

This allows us to efficiently approximate the solution of (12) to obtain \((J^T J)^{-1} \nabla \hat{f}\). We use 5 iterations of the conjugate gradient algorithm in practice.

From there, it easy to solve for \(\lambda\), given that \((J^+)^T \nabla \hat{f} = J (J^+ J)^{-1} \nabla \hat{f}\). Then, \(\delta^*\) can be calculated via (10). See Algorithm 1 for the full attack.

Computational complexity PPGD’s running time scales with the number of steps \(T\) and the number of conjugate gradient iterations \(K\). It also depends on whether the attack is self-bounded (the same network is used for classification and the LPIPS distance) or externally-bounded (different networks are used).

For each of the \(T\) steps, \(\theta(\tilde{x}), \nabla_{\tilde{x}} \mathcal{L}(f(\tilde{x}), y)\), and \(\phi(\tilde{x} + h\delta_k)\) must be calculated once (lines 4 and 15 in Algorithm 1). This takes 2 forward passes and 1 backward pass for the self-bounded case, and 3 forward passes and 1 backward pass for the externally-bounded case.

In addition, \(J^T Jv\) needs to be calculated (in the \textsc{MultiplyJacobian} routine) \(K + 1\) times. Each calculation of \(J^T Jv\) requires 1 forward and 1 backward pass, assuming \(\phi(\tilde{x})\) is already calculated.

Finally, the projection step takes \(n + 1\) forward passes for \(n\) iterations of the bisection method (see
At each step, we begin by taking the gradient of (13) with respect to \( \epsilon \), which begins at \( \Delta \). Then, we normalize \( \Delta \) to have \( L_2 \) norm 1, i.e., \( \Delta = \frac{\Delta}{\|\Delta\|_2} \). We wish to take a step in the direction of \( \Delta \) of size \( \eta \) in LPIPS distance. If we wanted to take a step of size \( \eta \) in \( L_2 \) distance, we could just take the step \( \eta \Delta \). However, taking a step of particular size in LPIPS distance is harder. We assume that the LPIPS distance is approximately linear in the direction \( \Delta \). We can approximate the directional derivative of the LPIPS distance in the direction \( \Delta \) using finite differences:
\[
\frac{d}{d\alpha}d(\bar{x}, \bar{x} + \alpha \Delta) \approx \frac{d(\bar{x}, \bar{x} + h\Delta)}{h} = m.
\]
Here, \( h \) is a small positive value, and we assign the approximation of the directional derivative to \( m \).

In all, the algorithm requires \( T(K + n + 4) \) forward passes and \( T(K + n + 3) \) backward passes in the self-bounded case. In the externally-bounded case, it requires \( T(K + n + 5) \) forward passes and the same number of backward passes.

**Algorithm 1** Perceptual PGD (PPGD)

1. **procedure** PPGD(classifier \( f(\cdot) \), LPIPS network \( \phi(\cdot) \), input \( x \), label \( y \), bound \( \epsilon \), step \( \eta \))
2. \( \bar{x} \leftarrow x + 0.01 \cdot \mathcal{N}(0, 1) \) \( \triangleright \) initialize perturbations with random Gaussian noise
3. **for** \( t = 1, \ldots, T \) **do** \( \triangleright T \) is the number of steps
4. \( \nabla \bar{f} \leftarrow \nabla_{\bar{x}} L(f(\bar{x}), y) \)
5. \( \delta_0 \leftarrow 0 \)
6. \( r_0 \leftarrow \nabla \bar{f} - \text{MULTIPLYJACOBIAN}(\phi, \bar{x}, \delta_0) \)
7. \( p_0 \leftarrow r_0 \)
8. **for** \( k = 0, \ldots, K - 1 \) **do** \( \triangleright \) conjugate gradient algorithm; we use \( K = 5 \) iterations
9. \( \alpha_k \leftarrow \frac{r_k \cdot r_k}{\text{MULTIPLYJACOBIAN}(\phi, \bar{x}, p_k)} \)
10. \( \delta_{k+1} \leftarrow \delta_k + \alpha_k p_k \)
11. \( r_{k+1} \leftarrow r_k - \alpha_k \text{MULTIPLYJACOBIAN}(\phi, \bar{x}, p_k) \)
12. \( \beta_k \leftarrow \frac{r_{k+1} \cdot r_k}{r_k \cdot r_k} \)
13. \( p_{k+1} \leftarrow r_{k+1} + \beta_k p_k \)
14. **end for**
15. \( m \leftarrow \|\phi(\bar{x} + h\delta_k) - \phi(\bar{x})\|_h \triangleright m \approx \|J\delta_k\| \) for small \( h \); we use \( h = 10^{-3} \)
16. \( \bar{x} \leftarrow (\eta/m)\delta_k \)
17. \( \bar{x} \leftarrow \text{PROJECT}(\alpha, \bar{x}, \epsilon) \)
18. **end for**
19. return \( \bar{x} \)
20. **end procedure**

21. **procedure** MULTIPLYJACOBIAN(\( \phi(\cdot), \bar{x}, v \)) \( \triangleright \) calculates \( J^T Jv \); \( J \) is the Jacobian of \( \phi \) at \( \bar{x} \)
22. \( Jv \leftarrow (\phi(\bar{x} + hv) - \phi(\bar{x}))/h \triangleright h \) is a small positive value; we use \( h = 10^{-3} \)
23. \( J^T Jv \leftarrow \nabla_u [\phi(\bar{x} + u)]^T Jv |_{u = 0} \)
24. return \( J^T Jv \)
25. **end procedure**

### A.2 LAGRANGIAN PERCEPTUAL ATTACK (LPA)

Our second attack, Lagrangian Perceptual Attack (LPA), optimizes a Lagrangian relaxation of the perceptual attack problem [3]:
\[
\max_{x} \mathcal{L}(f(\bar{x}), y) - \lambda \max \left( 0, \|\phi(\bar{x}) - \phi(x)\|_2 - \epsilon \right).
\]  
To optimize (13), we use a variation of gradient descent over \( \bar{x} \), starting at \( x \) with a small amount of noise added. We perform our modified version of gradient descent for \( T \) steps. We use a step size \( \eta \), which begins at \( \epsilon \) and decays exponentially to \( \epsilon/10 \).

At each step, we begin by taking the gradient of (13) with respect to \( \bar{x} \); let \( \Delta \) refer to this gradient. Then, we normalize \( \Delta \) to have \( L_2 \) norm 1, i.e., \( \Delta = \frac{\Delta}{\|\Delta\|_2} \). We wish to take a step in the direction of \( \Delta \) of size \( \eta \) in LPIPS distance. If we wanted to take a step of size \( \eta \) in \( L_2 \) distance, we could just take the step \( \eta \Delta \). However, taking a step of particular size in LPIPS distance is harder. We assume that the LPIPS distance is approximately linear in the direction \( \Delta \). We can approximate the directional derivative of the LPIPS distance in the direction \( \Delta \) using finite differences:
\[
\frac{d}{d\alpha}d(\bar{x}, \bar{x} + \alpha \Delta) \approx \frac{d(\bar{x}, \bar{x} + h\Delta)}{h} = m.
\]
\( \hat{\Delta} \) as follows:
\[
d(\tilde{x}, \tilde{x} + \alpha \hat{\Delta}) \approx d(\tilde{x}, \tilde{x}) + m\alpha = m\alpha.
\]
we want to take a step of size \( \eta \). Plugging in and solving, we obtain
\[
\eta = d(\tilde{x}, \tilde{x} + \alpha \hat{\Delta}) \approx m\alpha
\]
\[
\eta \approx m\alpha
\]
\[
\eta/m \approx \alpha.
\]
So, the approximate step we should take is \( (\eta/m) \hat{\Delta} \). We take this step at each of the \( T \) iterations of our modified gradient descent method.

We begin with \( \lambda = 10^{-2} \). After performing gradient descent, if \( d(x, \tilde{x}) > \epsilon \) (i.e. the adversarial example is outside the constraint) we increase \( \lambda \) by a factor of 10 and repeat the optimization. We repeat this entire process five times, meaning we search over \( \lambda \in \{10^{-2}, 10^{-3}, 10^0, 10^1, 10^2\} \). Finally, if the resulting adversarial example is still outside the constraint, we project it into the threat model; see Appendix [A.5].

**Computational complexity** LPA’s running time scales with the number of iterations \( S \) used to search for \( \lambda \) as well as the number of gradient descent steps \( T \). \( \phi(x) \) may be calculated once during the entire attack, which speeds it up. Then, each step of gradient descent requires 2 forward and 1 backward passes in the self-bounded case, and 3 forward and 2 backward passes in the externally-bounded case.

The projection at the end of the attack requires \( n + 1 \) forward passes for \( n \) iterations of the bisection method (see Section [A.4]).

In total, the attack requires \( 2ST + n + 2 \) forward passes and \( ST + n + 2 \) backward passes in the self-bounded case, and \( 3ST + n + 2 \) forward passes and \( 2ST + n + 2 \) backward passes in the externally-bounded case.

### Algorithm 2 Lagrangian Perceptual Attack (LPA)

1: procedure LPA(classifier network \( f(\cdot) \), LPIPS distance \( d(\cdot, \cdot) \), input \( x \), label \( y \), bound \( \epsilon \))  
2: \( \lambda \leftarrow 0.01 \)  
3: \( \tilde{x} \leftarrow x + 0.01 \cdot \mathcal{N}(0, 1) \) \( \triangleright \) initialize perturbations with random Gaussian noise  
4: for \( i \) in \( 1, \ldots, S \) do  
5: \( \triangleright \) we use \( S = 5 \) iterations to search for the best value of \( \lambda \)  
6: \( \Delta \leftarrow \nabla_{\tilde{x}} \left[ \mathcal{L}(f(\tilde{x}), y) - \lambda \max \left(0, d(\tilde{x}, x) - \epsilon\right)\right] \) \( \triangleright \) take the gradient of (5)  
7: \( \Delta \leftarrow \Delta/\|\Delta\|_2 \) \( \triangleright \) normalize the gradient  
8: \( \eta = \epsilon \ast (0.1)^{t/T} \) \( \triangleright \) the step size \( \eta \) decays exponentially  
9: \( m \leftarrow d(\tilde{x}, \tilde{x} + h\Delta)/h \) \( \triangleright m \approx \text{derivative of } d(\tilde{x}, \cdot) \text{ in the direction of } \Delta; \ h = 0.1 \)  
10: \( \tilde{x} \leftarrow \tilde{x} + (\eta/m)\Delta \) \( \triangleright \) take a step of size \( \eta \) in LPIPS distance  
11: end for  
12: if \( d(\tilde{x}, x) > \epsilon \) then  
13: \( \lambda \leftarrow 10\lambda \) \( \triangleright \) increase \( \lambda \) if the attack goes outside the bound  
14: end if  
15: end for  
16: \( \tilde{x} \leftarrow \text{PROJECT}(d, \tilde{x}, x, \epsilon) \)  
17: return \( \tilde{x} \)  
18: end procedure

### A.3 Fast Lagrangian Perceptual Attack

We use the Fast Lagrangian Perceptual Attack (Fast-LPA) for Perceptual Adversarial Training (PAT, see Section [5]). Fast-LPA is similar to LPA (Appendix [A.2]), with two major differences. First, Fast-LPA does not search over \( \lambda \) values; instead, during the \( T \) gradient descent steps, \( \lambda \) is increased exponentially from 1 to 10. Second, we remove the projection step at the end of the attack. This means that Fast-LPA may produce adversarial examples outside the threat model. This means that Fast-LPA cannot be used for evaluation, but it is fine for training.
**Computational complexity**  Fast-LPA’s running time can be calculated similarly to LPA’s (see Section A.2), except that $S = 1$ and there is no projection step. Let $T$ be the number of steps taken during the attack. Then Fast-LPA requires $2T + 1$ forward passes and $T + 1$ backward passes for the self-bounded case, and $3T + 1$ forward passes and $2T + 1$ backward passes for the externally-bounded case.

In comparison, PGD with $T$ iterations requires $T$ forward passes and $T$ backward passes. Thus, Fast-LPA is slightly slower, requiring $T + 1$ more forward passes and no more backward passes.

### Algorithm 3  Fast Lagrangian Perceptual Attack (Fast-LPA)

1: **procedure** FAST-LPA(classifier network $f(\cdot)$, LPIPS distance $d(\cdot, \cdot)$, input $x$, label $y$, bound $\epsilon$)  
2: $\tilde{x} \leftarrow x + 0.01 \ast N(0, 1)$  
3: for $t$ in 1, \ldots, $T$ do  
4: $\lambda \leftarrow 10^{t/T}$  
5: $\Delta \leftarrow \nabla_{\tilde{x}} [L(f(\tilde{x}), y) - \lambda \max (0, d(\tilde{x}, x) - \epsilon)]$  
6: $\hat{\Delta} = \Delta / \|\Delta\|_2$  
7: $\eta = \epsilon \ast (0.1)^{t/T}$  
8: $m \leftarrow d(\tilde{x}, \tilde{x} + (\eta/m)\Delta)$  
9: $\tilde{x} \leftarrow \tilde{x} + \eta \Delta$  
10: end for  
11: return $\tilde{x}$  
12: **end procedure**

### A.4  Perceptual Projection

We explored two methods of projecting adversarial examples into the LPIPS thread model. The method we use throughout the paper is based on Newton’s method and is shown in Algorithm 4. However, we also experimented with the bisection root finding method, shown in Algorithm 5 (also see Appendix A).

In general, given an adversarial example $\tilde{x}$, original input $x$, and LPIPS bound $\epsilon$, we wish to find a projection $\tilde{x}'$ of $\tilde{x}$ such that $d(\tilde{x}', x) \leq \epsilon$. Assume for this section that $d(\tilde{x}, x) > \epsilon$, i.e. the current adversarial example $\tilde{x}$ is outside the bound. If $d(\tilde{x}, x) \leq \epsilon$, then we can just let $\tilde{x}' = \tilde{x}$ and be done.

**Newton’s method**  The second projection method we explored uses the generalized Newton–Raphson method to attempt to find the closest projection $\tilde{x}'$ to the current adversarial example $\tilde{x}$ such that the projection is within the threat model, i.e. $d(\tilde{x}', x) \leq \epsilon$. To find such a projection, we again define a function $r(\cdot)$ and look for its roots:

$$r(\tilde{x}') = d(\tilde{x}', x) - \epsilon.$$

If we can find a projection $\tilde{x}'$ close to $\tilde{x}$ such that $r(\tilde{x}') \leq 0$, then this projection will be contained within the threat model, since

$$r(\tilde{x}') \leq 0 \Rightarrow d(\tilde{x}', x) \leq \epsilon.$$

To find such a root, we use the generalized Newton-Raphson method, an iterative algorithm. Beginning with $\tilde{x}_0 = \tilde{x}$, we update $\tilde{x}$ iteratively using the step

$$\tilde{x}_{i+1} = \tilde{x}_i - \left[\nabla r(\tilde{x}_i)\right]^+ (r(\tilde{x}_i) + s),$$

where $A^+$ denotes the pseudoinverse of $A$, and $s$ is a small positive constant (the “overshoot”), which helps the algorithm converge. We continue this process until $r(\tilde{x}_i) \leq 0$, at which point the projection is complete.

This algorithm usually takes 2-3 steps to converge with $s = 10^{-2}$. Each step requires 1 forward and 1 backward pass to calculate $r(\tilde{x}_i)$ and its gradient. The method also requires 1 forward pass at the beginning to calculate $\phi(x)$. 

Published as a conference paper at ICLR 2021
We use the bisection root finding method to find a root $\alpha$. Algorithm 5

\begin{algorithm}
\caption{Perceptual Projection (Bisection Method)}
\begin{algorithmic}
\Procedure{Projekt}(LPIPS distance $d(\cdot, \cdot)$, adversarial example $x$, original input $x$, bound $\epsilon$)
\State $\tilde{x}_0$
\For{$i$ in 0, …, do}
\State $r(\tilde{x}_i) \leftarrow d(\tilde{x}_i, x) - \epsilon$
\If{$r(\tilde{x}_i) \leq 0$}
\State return $\tilde{x}_i$
\EndIf
\State $\tilde{x}_{i+1} = \tilde{x}_i - \left[ \nabla r(\tilde{x}_i) \right]^+ \left( r(\tilde{x}_i) + s \right)$ \Comment{$s$ is the “overshoot”; we use $s = 10^{-2}$}
\EndFor
\EndProcedure
\end{algorithmic}
\end{algorithm}

**Bisection method** The first projection method we explored (and the one we use throughout the paper) attempts to find a projection $\tilde{x}'$ along the line connecting the current adversarial example $x$ and original input $x$. Let $\delta = x - \tilde{x}$. Then we can represent our final projection $\tilde{x}'$ as a point between $x$ and $\tilde{x}$ as

$$\tilde{x}' = x + \alpha \delta,$$

for some $\alpha \in [0, 1]$. If $\alpha = 0$, $\tilde{x}' = x$; if $\alpha = 1$, $\tilde{x}' = \tilde{x}$. Now, define a function $r : [0, 1] \rightarrow \mathbb{R}$ as

$$r(\delta) = d(x + \alpha \delta, x) - \epsilon.$$ 

This function has the following properties:

1. $r(0) < 0$, since $r(0) = d(x, x) - \epsilon = -\epsilon$.
2. $r(1) > 0$, since $r(1) = d(\tilde{x}, x) - \epsilon > 0$ because $d(\tilde{x}, x) > \epsilon$.
3. $r(\alpha) = 0$ if $d(\tilde{x}', x) = \epsilon$.

We use the bisection root finding method to find a root $\alpha^*$ of $r(\cdot)$ on the interval $[0, 1]$, which exists since $r(\cdot)$ is continuous and because of items 1 and 2 above. By item 3, at this root, the projected adversarial example is within the threat model:

$$d(\tilde{x}', x) = d(x + \alpha^* \delta, x) = r(\alpha^*) + \epsilon = \epsilon$$

We use $n = 10$ iterations of the bisection method to calculate $\alpha^*$. This requires $n + 1$ forward passes through the LPIPS network, since $\phi(x)$ must be calculated once, and $\phi(x + \alpha \delta)$ must be calculated $n$ times. See Algorithm 5 for the full projection algorithm.

\begin{algorithm}
\caption{Perceptual Projection (Newton’s Method)}
\begin{algorithmic}
\Procedure{Projekt}(LPIPS distance $d(\cdot, \cdot)$, adversarial example $x$, original input $x$, bound $\epsilon$)
\State $\alpha_{\min}, \alpha_{\max} \leftarrow 0, 1$
\State $\delta \leftarrow \tilde{x} - x$
\For{$i$ in 1, …, $n$ do}
\State $\alpha \leftarrow (\alpha_{\min} + \alpha_{\max}) / 2$
\State $\tilde{x}' \leftarrow \tilde{x} + \alpha \delta$
\If{$d(x, \tilde{x}') > \epsilon$}
\State $\alpha_{\max} \leftarrow \alpha$
\Else
\State $\alpha_{\min} \leftarrow \alpha$
\EndIf
\EndFor
\State return $\tilde{x}'$
\EndProcedure
\end{algorithmic}
\end{algorithm}

**Algorithm 4** Perceptual Projection (Newton’s Method)

procedure PROJECT(LPIPS distance $d(\cdot, \cdot)$, adversarial example $x$, original input $x$, bound $\epsilon$)
\State $\tilde{x}_0$
\For{$i$ in 0, …, do}
\State $r(\tilde{x}_i) \leftarrow d(\tilde{x}_i, x) - \epsilon$
\If{$r(\tilde{x}_i) \leq 0$ then}
\State return $\tilde{x}_i$
\EndIf
\State $\tilde{x}_{i+1} = \tilde{x}_i - \left[ \nabla r(\tilde{x}_i) \right]^+ \left( r(\tilde{x}_i) + s \right)$ \Comment{$s$ is the “overshoot”; we use $s = 10^{-2}$}
\EndFor
\end{algorithmic}
\end{algorithm}

**B Additional Related Work**

Here, we expand on the related work discussed in Section 2 discuss some additional existing work on adversarial robustness.
Adversarial attacks  Much of the initial work on adversarial robustness focused on perturbations to natural images which were bounded by the $L_2$ or $L_\infty$ distance (Carlini and Wagner, 2017; Goodfellow et al., 2015; Madry et al., 2018). However, recently the community has discovered many other types of perturbations that are imperceptible and can be optimized to fool a classifier, but are outside $L_p$ threat models. These include spatial perturbations using flow fields (Xiao et al., 2018), translation and rotation (Engstrom et al., 2017), and Wasserstein distance bounds (Wong et al., 2019). Attacks that manipulate the colors in images uniformly also been proposed (Hosseini and Poovendran, 2018; Hosseini et al., 2017; Zhang et al., 2019b) and have been generalized into “functional adversarial attacks” by Laidlaw and Feizi (2019).

A couple papers have proposed adversarial threat models that do not focus on a simple, manually defined perturbation type. Dunn et al. (2020) use a generative model of images; they perturb the features at various layers in the generator to create adversarial examples. Xu et al. (2020) train an autoencoder and then perturb images in representation space rather than pixel space.

C ADDITIONAL ADVERSARIAL EXAMPLES

![Figure 7: Adversarial examples generated using self-bounded and externally-bounded PPGD and LPA perceptual adversarial attacks (Section 4) with a large bound. Original images are shown in the left column and magnified differences from the original are shown to the right of the examples.](image-url)
D Additional Perceptual Study Results

Table 4: Bounds and results from the perceptual study. Each threat model was evaluated with a small, medium, and large bound. Bounds for $L_2$, $L_{\infty}$, and JPEG attacks (first three rows) are given assuming input image is in the range $[0, 255]$. Perceptibility (perc.) is the proportion of natural input-adversarial example pairs annotated as “different” by participants. Strength (str.) is the success rate when attacking a classifier adversarially trained against that threat model (higher is stronger). Perceptual attacks (PPGD and LPA, see Section 4) are externally bounded with AlexNet. All experiments on ImageNet-100.

<table>
<thead>
<tr>
<th>Threat model</th>
<th>Small bound</th>
<th>Medium bound</th>
<th>Large bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_\infty$</td>
<td>2</td>
<td>7%</td>
<td>30.4%</td>
</tr>
<tr>
<td>$L_2$</td>
<td>600</td>
<td>8%</td>
<td>40.6%</td>
</tr>
<tr>
<td>JPEG-$L_{\infty}$</td>
<td>0.0625</td>
<td>6%</td>
<td>19.5%</td>
</tr>
<tr>
<td>StAdv</td>
<td>0.025</td>
<td>8%</td>
<td>28.1%</td>
</tr>
<tr>
<td>ReColorAdv</td>
<td>0.03</td>
<td>14%</td>
<td>14.8%</td>
</tr>
<tr>
<td>PPGD</td>
<td>0.25</td>
<td>7%</td>
<td>47.9%</td>
</tr>
<tr>
<td>LPA</td>
<td>0.25</td>
<td>14%</td>
<td>76.5%</td>
</tr>
</tbody>
</table>

E Perceptual Attack Experiments

We experiment with variations of the two validation attacks, PPGD and LPA, described in Section 4. As described in Appendix A.4, we developed two methods for projecting candidate adversarial examples into the LPIPS ball surrounding a natural input. We attack a single model using PPGD and LPA with both projection methods. We also compare self-bounded to externally-bounded attacks.

We find that LPA tends to be more powerful than PPGD. Finally, we note that externally-bounded LPA is extremely powerful, reducing the accuracy of a PAT-trained classifier on ImageNet-100 to just 2.4%.

Besides these experiments, we always use externally-bounded attacks with AlexNet for evaluation. AlexNet correlates with human perception of adversarial examples (Figure 6) and provides a standard measure of LPIPS distance; in contrast, self-bounded attacks by definition have varying bounds across evaluated models.

Table 5: Accuracy of a PAT-trained ResNet-50 on ImageNet-100 against various perceptual adversarial attacks. PPGD and LPA attacks are shown self-bounded and externally-bounded with AlexNet. We also experimented with two different perceptual projection methods (see Appendix A.4). Bounds are $\epsilon = 0.25$ for self-bounded attacks and $\epsilon = 0.5$ for externally-bounded attacks, since the LPIPS distance from AlexNet tends to be about twice as high as that from ResNet-50.

<table>
<thead>
<tr>
<th>Attack</th>
<th>LPIPS model</th>
<th>Projection</th>
<th>Accuracy against PAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPGD</td>
<td>self</td>
<td>bisection method</td>
<td>43.3</td>
</tr>
<tr>
<td>PPGD</td>
<td>self</td>
<td>Newton’s method</td>
<td>49.7</td>
</tr>
<tr>
<td>PPGD</td>
<td>AlexNet</td>
<td>bisection method</td>
<td>45.2</td>
</tr>
<tr>
<td>PPGD</td>
<td>AlexNet</td>
<td>Newton’s method</td>
<td>28.7</td>
</tr>
<tr>
<td>LPA</td>
<td>self</td>
<td>bisection method</td>
<td>39.6</td>
</tr>
<tr>
<td>LPA</td>
<td>self</td>
<td>Newton’s method</td>
<td>53.8</td>
</tr>
<tr>
<td>LPA</td>
<td>AlexNet</td>
<td>bisection method</td>
<td>4.2</td>
</tr>
<tr>
<td>LPA</td>
<td>AlexNet</td>
<td>Newton’s method</td>
<td>2.4</td>
</tr>
</tbody>
</table>
F  PAT EXPERIMENTS

F.1 ABLATION STUDY

We perform an ablation study of Perceptual Adversarial Training (PAT). First, we examine Fast-LPA, the training attack. We attempt training without step size ($\eta$) decay and/or without increasing $\lambda$ during Fast-LPA, and find that PAT performs best with both $\eta$ decay and $\lambda$ increase.

Training a classifier with PAT gives robustness against a wide range of adversarial threat models (see Section 7). However, it tends to give low accuracy against natural, unperturbed inputs. Thus, we use a technique from Balaji et al. (2019) to improve natural accuracy in PAT-trained models: at each training step, only inputs which are classified correctly without any perturbation are attacked. In addition to increasing natural accuracy, this also improves the speed of PAT since only some inputs from each batch must be attacked. In this ablation study, we compare attacking every input with Fast-LPA during training to only attacking the natural inputs which are already classified correctly.

Table 6: Accuracies against various attacks for models in the PAT ablation study. Attack bounds are $8/255$ for $L_\infty$, 1 for $L_2$, 0.5 for PPGD/LPA, and the original bounds for StAdv/ReColorAdv.

<table>
<thead>
<tr>
<th>Ablation</th>
<th>Union Mean</th>
<th>Unseen Mean</th>
<th>Narrow threat models</th>
<th>NPTM PGGD LPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>21.9</td>
<td>45.6</td>
<td>82.4 30.2 34.9 46.4 71.0</td>
<td>13.1 2.1</td>
</tr>
<tr>
<td>No $\eta$ decay</td>
<td>17.2</td>
<td>48.1</td>
<td>82.7 37.5 43.3 39.7 72.1</td>
<td>10.5 1.1</td>
</tr>
<tr>
<td>No $\lambda$ increase</td>
<td>8.1</td>
<td>42.1</td>
<td>85.1 33.7 35.9 27.8 71.1</td>
<td>11.6 1.0</td>
</tr>
<tr>
<td>No $\eta$ decay or $\lambda$ increase</td>
<td>19.6 49.2</td>
<td>82.1 36.7 41.9 44.8 73.4</td>
<td>10.9 0.9</td>
<td></td>
</tr>
<tr>
<td>Attack all inputs</td>
<td>26.8</td>
<td>46.6</td>
<td>74.5 29.8 33.5 56.6 66.4</td>
<td>24.5 6.5</td>
</tr>
</tbody>
</table>

F.2 PROJECTION DURING TRAINING

We choose not to add a projection step to the end of Fast-LPA during training because it slows down the attack, requiring many more passes through the network per training step. However, we tested self-bounded PAT with a projection step and found that it increased clean accuracy slightly but decreased robust accuracy significantly. We believe this is because not projecting increases the effective bound on the training attacks, leading to better robustness. To test this, we tried training without projection using a smaller bound ($\epsilon = 0.4$ instead of $\epsilon = 0.5$) and found the results closely matched the results when using projection at the larger bound. That is, PAT with projection at $\epsilon = 0.5$ is similar to PAT without projection at $\epsilon = 0.4$. These results are shown in Table 7.

Table 7: Accuracies against various attacks for PAT-trained models on CIFAR-10, with and without a projection step during training.

<table>
<thead>
<tr>
<th>Projection</th>
<th>Bound ($\epsilon$)</th>
<th>Union Mean</th>
<th>Unseen Mean</th>
<th>Narrow threat models</th>
<th>NPTM PGGD LPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>0.5</td>
<td>21.9</td>
<td>45.6</td>
<td>82.4 30.2 34.9 46.4 71.0</td>
<td>13.1 2.1</td>
</tr>
<tr>
<td>Yes</td>
<td>0.5</td>
<td>5.8</td>
<td>41.4</td>
<td>83.9 35.7 38.5 17.9 73.3</td>
<td>10.7 1.7</td>
</tr>
<tr>
<td>No</td>
<td>0.4</td>
<td>4.9</td>
<td>36.3</td>
<td>84.1 31.1 36.3 10.3 67.4</td>
<td>10.9 3.2</td>
</tr>
</tbody>
</table>

F.3 SELF-BOUNDED VS. ALEXNET-BOUNDED PAT

Performing externally-bounded PAT with AlexNet produces more robust models on CIFAR-10 than self-bounded PAT. This is not the case on ImageNet-100, where self- and AlexNet-bounded PAT perform similarly.

There is a simple explanation for this: the effective training bound on CIFAR-10 is greater for AlexNet-bounded PAT than for self-bounded PAT. To measure this, we generate adversarial examples
for all the test threat models on CIFAR-10 ($L_2$, $L_\infty$, StAdv, and ReColorAdv). We find that the average LPIPS distance for all adversarial examples using AlexNet is 1.13; for a PAT-trained ResNet-50, it is 0.88. Because of this disparity, we use a lower training bound for self-bounded PAT ($\epsilon = 0.5$) than for AlexNet-bounded PAT ($\epsilon = 1$). However, this means that the average test attack has 76% greater LPIPS distance than the training attacks for self-bounded PAT, whereas the average test attack has only 13% greater LPIPS distance for AlexNet-bounded PAT. This explains why AlexNet-bounded PAT gives better robustness; it only has to generalize to slightly larger attacks on average.

We tried performing AlexNet-bounded PAT with a more comparable bound ($\epsilon = 0.7$) to self-bounded PAT. This gives the average test attack about 80% greater LPIPS distance than the training attacks, similar to self-bounded PAT. Table 8 shows that the results are more similar for self-bounded and AlexNet-bounded PAT with $\epsilon = 0.7$.

Table 8: Accuracies against various attacks for classifiers on CIFAR-10 trained with self- and AlexNet-bounded PAT using various bounds.

<table>
<thead>
<tr>
<th>LPIPS model</th>
<th>Bound ($\epsilon$)</th>
<th>Union mean</th>
<th>Narrow threat models</th>
<th>NPTM PPGD LPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self</td>
<td>0.5</td>
<td>21.9</td>
<td>Clean $L_\infty$ $L_2$ StAdv ReColor</td>
<td>71.0</td>
</tr>
<tr>
<td>AlexNet</td>
<td>0.7</td>
<td>16.7</td>
<td>80.0 35.5 41.3 33.2 71.5</td>
<td>17.8 4.9</td>
</tr>
<tr>
<td>AlexNet</td>
<td>1.0</td>
<td>27.8</td>
<td>71.6 28.7 33.3 64.5 67.5</td>
<td>26.6 9.8</td>
</tr>
</tbody>
</table>

F.4 Accuracy-Robustness Tradeoff

Tsipras et al. (2019) have noted that there is often a tradeoff the between adversarial robustness of a classifier and its accuracy. That is, models which have higher accuracy under adversarial attack may have lower accuracy against clean images. We observe this phenomenon with adversarial training and PAT. Since PAT gives greater robustness against several narrow threat models, models trained with it tend to have lower accuracy on clean images than models trained with narrow adversarial training. In Figure 8, we show the robust and clean accuracies of several models trained on CIFAR-10 and ImageNet-100 with PAT and adversarial training. While some PAT models have lower clean accuracy than adversarially trained models, at least one PAT model on each dataset surpasses the Pareto frontier of the accuracy-robustness tradeoff for adversarial training. That is, there are PAT-trained models on both datasets with both higher robust accuracy and higher clean accuracy than adversarial training.

F.5 Performance Against StAdv and ReColorAdv

It was surprising to find that PAT outperformed threat-specific adversarial training (AT) against the StAdv and ReColorAdv attacks on CIFAR-10 (it does not do so on ImageNet-100). In Table 2 (partially reproduced in Table 9 below), PAT-AlexNet improves robustness over AT against StAdv from 54% to 65%; PAT-self improves robustness over AT against ReColorAdv from 65% to 71%.
We conjecture that, for these threat models, this is because training against a wider set of perturbations at training time helps generalize robustness to new inputs at test time, even within the same threat model. To test this, we additionally train classifiers using adversarial training against the StAdv and ReColorAdv attacks with double the default bound. The results are shown in Table 9 below. We find that, because these classifiers are exposed to a wider range of spatial and recoloring perturbations during training, they perform better than PAT against those attacks at test time (76% vs 65% for StAdv and 81% vs 71% for ReColorAdv).

This suggests that PAT not only improves robustness against a wide range of adversarial threat models, it can actually improve robustness over threat-specific adversarial training by incorporating a wider range of attacks during training.

Table 9: Results of our experiments training against StAdv and ReColorAdv on CIFAR-10 with double the default bounds. Columns are identical to Table 2.

<table>
<thead>
<tr>
<th>Training</th>
<th>Union mean</th>
<th>Unseen mean</th>
<th>Narrow threat models</th>
<th>NPTM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Clean $L_\infty$ $L_2$ StAdv ReColor</td>
<td>PPGD LPA</td>
</tr>
<tr>
<td>AT StAdv</td>
<td>0.0</td>
<td>1.4</td>
<td>86.2 0.1 0.2 53.9 5.1</td>
<td>0.0 0.0</td>
</tr>
<tr>
<td>AT StAdv (double $\epsilon$)</td>
<td>0.2</td>
<td>3.7</td>
<td>83.9 0.2 0.5 76.1 13.9</td>
<td>0.0 0.0</td>
</tr>
<tr>
<td>AT ReColorAdv</td>
<td>0.0</td>
<td>3.1</td>
<td>93.4 8.5 3.9 0.0 65.0</td>
<td>0.1 0.0</td>
</tr>
<tr>
<td>AT ReColorAdv (double $\epsilon$)</td>
<td>0.0</td>
<td>5.3</td>
<td>92.0 12.5 8.6 0.3 81.2</td>
<td>0.4 0.0</td>
</tr>
<tr>
<td>PAT-self</td>
<td>21.9</td>
<td>45.6</td>
<td>82.4 30.2 34.9 46.4 71.0</td>
<td>13.1 2.1</td>
</tr>
<tr>
<td>PAT-AlexNet</td>
<td>27.8</td>
<td>48.5</td>
<td>71.6 28.7 33.3 64.5 67.5</td>
<td>26.6 9.8</td>
</tr>
</tbody>
</table>

G COMMON CORRUPTIONS EVALUATION

We evaluate the robustness of PAT-trained models to common corruptions in addition to adversarial examples on CIFAR-10 and ImageNet-100. In particular, we test PAT-trained classifiers on CIFAR-10-C and ImageNet-100-C, where ImageNet-100-C is the 100-class subset of ImageNet-C formed by taking every tenth class [Hendrycks and Dietterich 2019]. These datasets are based on random corruptions of CIFAR-10 and ImageNet, respectively, using 15 perturbation types with 5 levels of severity. The perturbation types are split into four general categories: “noise,” “blur,” “weather,” and “digital.”

The metric we use to evaluate PAT against common corruptions is mean relative corruption error (relative mCE). The relative corruption error is defined by [Hendrycks and Dietterich 2019] for a classifier $f$ and corruption type $c$ as

$$\text{Relative CE}_c^f = \frac{\sum_{s=1}^{5} E_{s,c}^f - E_{clean}^f}{\sum_{s=1}^{5} E_{AlexNet}^{s,c} - E_{AlexNet}^{clean}}$$

where $E_{s,c}^f$ is the error of classifier $f$ against corruption type $c$ at severity level $s$, and $E_{clean}^f$ is the error of classifier $f$ on unperturbed inputs. The relative mCE is defined as the mean relative CE over all perturbation types.

The relative mCE for classifiers trained with normal training, adversarial training, and PAT is shown in Tables 10 and 11. PAT gives better robustness (lower relative mCE) against common corruptions on both CIFAR-10-C and ImageNet-100-C. The only category of perturbations where $L_2$ adversarial training outperforms PAT is “noise” on CIFAR-10-C, which makes sense because Gaussian and other types of noise are symmetrically distributed in an $L_2$ ball. For the other perturbation types and on ImageNet-100-C, PAT outperforms $L_2$ and $L_\infty$ adversarial training, indicating that robustness against a wider range of worst-case perturbations also gives robustness against a wider range of random perturbations.
Table 10: Robustness of classifiers trained with adversarial training and PAT against common corruptions in the CIFAR-10-C dataset. Results are reported as relative mCE (lower is better). PAT improves robustness against common corruptions overall and for three of the specific perturbation categories.

<table>
<thead>
<tr>
<th>Perturbation Type</th>
<th>Normal</th>
<th>L∞</th>
<th>L2</th>
<th>PAT-self</th>
<th>PAT-AlexNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noise</td>
<td>1.72</td>
<td>0.24</td>
<td>0.16</td>
<td>0.22</td>
<td>0.22</td>
</tr>
<tr>
<td>Blur</td>
<td>1.57</td>
<td>0.46</td>
<td>0.39</td>
<td>0.30</td>
<td>0.34</td>
</tr>
<tr>
<td>Weather</td>
<td>1.03</td>
<td>0.96</td>
<td>1.02</td>
<td>0.90</td>
<td>0.88</td>
</tr>
<tr>
<td>Digital</td>
<td>1.87</td>
<td>0.61</td>
<td>0.61</td>
<td>0.58</td>
<td>0.56</td>
</tr>
<tr>
<td>All</td>
<td>1.59</td>
<td>0.57</td>
<td>0.54</td>
<td>0.50</td>
<td>0.49</td>
</tr>
</tbody>
</table>

Table 11: Robustness of classifiers trained with adversarial training and PAT against common corruptions in the ImageNet-100-C dataset. Results are reported as relative mCE (lower is better).

<table>
<thead>
<tr>
<th>Perturbation Type</th>
<th>Normal</th>
<th>L∞</th>
<th>L2</th>
<th>PAT-self</th>
<th>PAT-AlexNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noise</td>
<td>0.85</td>
<td>0.54</td>
<td>0.41</td>
<td>0.39</td>
<td>0.49</td>
</tr>
<tr>
<td>Blur</td>
<td>0.57</td>
<td>0.41</td>
<td>0.36</td>
<td>0.36</td>
<td>0.35</td>
</tr>
<tr>
<td>Weather</td>
<td>0.56</td>
<td>0.62</td>
<td>0.71</td>
<td>0.60</td>
<td>0.61</td>
</tr>
<tr>
<td>Digital</td>
<td>0.35</td>
<td>0.25</td>
<td>0.27</td>
<td>0.24</td>
<td>0.24</td>
</tr>
<tr>
<td>All</td>
<td>0.55</td>
<td>0.42</td>
<td>0.41</td>
<td>0.37</td>
<td>0.39</td>
</tr>
</tbody>
</table>

H  Experiment Details

For all experiments, we train ResNet-50 [He et al., 2016] with SGD for 100 epochs. We use 10 attack iterations for training and 200 for testing, except for PPGD and LPA, where we use 40 for testing since they are more expensive. Self-bounded PAT takes about 12 hours to train for CIFAR-10 on an Nvidia RTX 2080 Ti GPU, and about 5 days to train for ImageNet-100 on 4 GPUs. We implement PPGD, LPA, and PAT using PyTorch [Paszke et al., 2017].

We preprocess images after adversarial perturbation, but before classification, by standardizing them based on the mean and standard deviation of each channel for all images in the dataset. We use the default data augmentation techniques from the robustness library [Engstrom et al., 2019]. The CIFAR-10 dataset can be obtained from https://www.cs.toronto.edu/~kriz/cifar.html. The ImageNet-100 dataset is a subset of the ImageNet Large Scale Visual Recognition Challenge (2012) [Russakovsky et al., 2015] including only every tenth class by WordNet ID order. It can be obtained from http://www.image-net.org/download-images.
Table 12: Hyperparameters for the adversarial training experiments on CIFAR-10 and ImageNet-100. For CIFAR-10, hyperparameters are similar to those used by Zhang et al. (2019a). For ImageNet-100, hyperparameters are similar to those used by Kang et al. (2019).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>CIFAR-10</th>
<th>ImageNet-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>ResNet-50</td>
<td>ResNet-50</td>
</tr>
<tr>
<td>Number of parameters</td>
<td>23,520,842</td>
<td>23,712,932</td>
</tr>
<tr>
<td>Optimizer</td>
<td>SGD</td>
<td>SGD</td>
</tr>
<tr>
<td>Momentum</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>Weight decay</td>
<td>$2 \times 10^{-4}$</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td>Batch size</td>
<td>50</td>
<td>128</td>
</tr>
<tr>
<td>Training epochs</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>Initial learning rate</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Learning rate drop epochs (×0.1 drop)</td>
<td>75, 90</td>
<td>30, 60, 80</td>
</tr>
<tr>
<td>Attack iterations (train)</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Attack iterations (test)</td>
<td>200</td>
<td>200</td>
</tr>
</tbody>
</table>

H.1 LAYERS FOR LPIPS CALCULATION

Calculating the LPIPS distance using a neural network classifier $g(\cdot)$ requires choosing layers whose normalized, flattened activations $\phi(\cdot)$ should be compared between images. For AlexNet and VGG-16, we use the same layers to calculate LPIPS distance as do Zhang et al. (2018). For AlexNet (Krizhevsky et al., 2012), we use the activations after each of the first five ReLU functions. For VGG-16 (Simonyan and Zisserman, 2014), we use the activations directly before the five max pooling layers. In ResNet-50, we use the outputs of the conv2_x, conv3_x, conv4_x, and conv5_x layers, as listed in Table 1 of He et al. (2016).