Adversarial Robustness Against the Union of Multiple Perturbation Models (Supplementary Material)

A. Steepest descent and projections for ℓ_∞, ℓ_2, and ℓ_1 adversaries

In this section, we describe the steepest descent and projection steps for ℓ_p adversaries for $p \in \{\infty, 2, 1\}$; these are standard results, but are included for a complete description of the algorithms. Note that this differs slightly from the adversaries considered in Schott et al. (2019): while they used an ℓ_0 adversary, we opted to use an ℓ_1 adversary. The ℓ_0 ball with radius ϵ is contained within an ℓ_1 ball with the same radius, so achieving robustness against an ℓ_1 adversary is strictly more difficult.

ℓ_∞ space The direction of steepest descent with respect to the ℓ_∞ norm is

$$v_\infty(\delta) = \alpha \cdot \text{sign}(\nabla l(x+\delta;\theta)) \tag{15}$$

and the projection operator onto $\Delta_{\infty,\epsilon}$ is

$$\mathcal{P}_{\Delta_{\infty,\epsilon}}(\delta) = \text{clip}_{[-\epsilon,\epsilon]}(\delta) \tag{16}$$

ℓ_2 space The direction of steepest descent with respect to the ℓ_2 norm is

$$v_2(\delta) = \alpha \cdot \frac{\nabla \ell(x+\delta;\theta)}{\|\nabla \ell(x+\delta;\theta)\|_2} \tag{17}$$

and the projection operator onto the ℓ_2 ball around x is

$$\mathcal{P}_{\Delta_{2,\epsilon}}(\delta) = \epsilon \cdot \frac{\delta}{\max\{\epsilon, \|\delta\|_2\}} \tag{18}$$

ℓ_1 space The direction of steepest descent with respect to the ℓ_1 norm is

$$v_1(\delta) = \alpha \cdot \text{sign}\left(\frac{\partial \ell(x+\delta;\theta)}{\partial \delta_{i^\star}}\right) \cdot e_{i^\star} \tag{19}$$

where

$$i^\star = \arg\max_i |\nabla l(x+\delta;\theta)_i| \tag{20}$$

and e_{i^*} is a unit vector with a one in position i^*. Finally, the projection operator onto the ℓ_1 ball,

$$\mathcal{P}_{\Delta_{1,\epsilon}}(\delta) = \mathop{\arg\min}_{\delta' : \|\delta'\|_1 \le \epsilon} \|\delta - \delta'\|_2^2, \tag{21}$$

can be solved with Algorithm 2, and we refer the reader to Duchi et al. (2008) for its derivation.

Algorithm 2 Projection of some perturbation $\delta \in \mathbb{R}^n$ onto the ℓ_1 ball with radius ϵ. We use $|\cdot|$ to denote element-wise absolute value.

```
**Input:** perturbation $\delta$, radius $\epsilon$
Sort $|\delta|$ into $\gamma : \gamma_1 \ge \gamma_2 \ge \cdots \ge \gamma_n$
$\rho := \max\left\{j \in [n] : \gamma_j - \frac{1}{j}\left(\sum_{r=1}^{j} \gamma_r - \epsilon\right) > 0\right\}$
$\eta := \frac{1}{\rho}\left(\sum_{i=1}^{\rho} \gamma_i - \epsilon\right)$
$z_i := \text{sign}(\delta_i) \max\{\gamma_i - \eta, 0\}$ for $i = 1 \dots n$
**return** $z$
```

A.1. Enhanced ℓ_1 steepest descent step

Note that the steepest descent step for ℓ_1 only updates a single coordinate per step. This can be quite inefficient, as pointed out by Tramèr & Boneh (2019). To tackle this issue, and also empirically improve the attack success rate, Tramèr & Boneh (2019) instead select the top k coordinates according to Equation 20 to update. In this work, we adopt a similar but slightly modified scheme: we randomly sample k to be some integer within some range $[k_1, k_2]$, and update each coordinate with step size $\alpha' = \alpha/k$. We observe in our experimentation that the randomness induced by varying the number of coordinates aids in reducing the gradient masking problem observed by Tramèr & Boneh (2019).

A.2. Restricting the steepest descent coordinate

The steepest descent direction for both the ℓ_0 and ℓ_1 norm end up selecting a single coordinate direction to move the perturbation. However, if the perturbation is already at the boundary of pixel space (for MNIST, this is the range [0,1] for each pixel), then it's possible for the PGD adversary to get stuck in a loop trying to use the same descent direction to escape pixel space. To avoid this, we only allow the steepest descent directions for these two attacks to choose coordinates that keep the image in the range of real pixels.

B. Extended results

Here, we show the full break down of adversarial error rates over individual attacks for both MNIST and CIFAR10.

B.1. MNIST results

Expanded table of results Table 3 contains brak down of adversarial accuracies against all attacks for all models on the MNIST dataset. All attacks were run on a subset of the first 1000 test examples with 10 random restarts, with the exception of Boundary Attack, which by default makes 25 trials per iteration, and DDN attack, which does not benefit from restarts owing to a deterministic starting point. The results for B-ABS and ABS models are reported directly from Schott et al. (2019), which uses gradient estimation

Table 3: Summary of adversarial accuracy results for MNIST

	P_∞	P_2	P_1	B-ABS	ABS	MAX	AVG	MSD
Clean Accuracy	99.1%	99.2%	99.3%	99%	99%	98.6%	99.1%	98.3%
PGD-ℓ_∞	90.3%	0.4%	0.0%	-	-	51.0%	65.2%	62.7%
FGSM	94.9%	68.3%	6.4%	85%	34%	81.4%	85.5%	82.8%
PGD-Foolbox	92.1%	8.5%	0.1%	86%	13%	65.8%	73.5%	69.2%
MIM	92.3%	11.2%	0.1%	85%	17%	70.7%	76.7%	71.0%
ℓ_∞ attacks ($\epsilon = 0.3$)	90.3%	0.4%	0.0%	77%	8%	51.0%	65.2%	62.7%
PGD-ℓ_2	68.8%	69.2%	38.7%	-	-	64.1%	67.9%	70.2%
PGD-Foolbox	88.9%	77.9%	48.7%	63%	87%	75.6%	80.3%	78.4%
Gaussian Noise	98.9%	98.6%	98.9%	89%	98%	97.7%	98.6%	97.2%
Boundary Attack	18.2%	81.4%	62.1%	91%	83%	73.6%	71.8%	72.4%
DeepFool	93.0%	86.8%	59.5%	41%	83%	81.7%	87.3%	80.7%
Pointwise Attack	40.6%	95.1%	96.7%	87%	94%	90.8%	85.9%	89.6%
DDN	63.9%	70.5%	40.0%	-	-	62.5%	64.6%	69.5%
CWL2	79.6%	74.5%	44.8%	-	-	72.1%	72.4%	74.5%
ℓ_2 attacks ($\epsilon = 2.0$)	13.6%	69.2%	38.5%	39%	80%	61.9%	60.1%	67.9%
PGD-ℓ_1	61.8%	51.1%	74.6%	-	-	61.2%	66.5%	70.4%
Salt & Pepper	62.1%	96.4%	97.7%	96%	95%	94.6%	90.6%	89.1%
Pointwise Attack	5.3%	83.3%	89.1%	82%	78%	65.3%	45.4%	70.7%
ℓ_1 attacks ($\epsilon = 10$)	4.2%	43.4%	70.0%	82%	78%	52.6%	39.2%	65.0%
All attacks	3.7%	0.4%	0.0%	39%	8%	42.1%	34.9%	**58.4**%

techniques whenever a gradient is needed, and the robustness against all attacks for B-ABS and ABS is an upper bound based on the reported results. Further, they used epsilon balls of radii (0.3,1.5,12) for (ℓ_∞, ℓ_2, ℓ_0) adversaries. Moreover, they used an ℓ_0 perturbation model of a higher radius and evaluated against ℓ_0 attacks. So the reported number is a near estimate of the ℓ_1 adversarial accuracy.

B.2. CIFAR10 results

Expanded table of results Table 4 contains the full table of results for all attacks on all models on the CIFAR10 dataset. All attacks were run on a subset of the first 1000 test examples with 10 random restarts, with the exception of Boundary Attack, which by default makes 25 trials per iteration, and DDN attack, which does not benefit from restarts owing to a deterministic starting point. Further note that salt & pepper and pointwise attacks in the ℓ_1 section are technically ℓ_0 attacks, but produce perturbations in the ℓ_1 ball. Finally, it is clear here that while the training against an ℓ_1 PGD adversary defends against said PGD adversary, it does not seem to transfer to robustness against other attacks.

C. Experimental details

C.1. Hyperparameters for PGD adversaries

In this section, we describe the parameters used for all PGD adversaries in this paper.

MNIST The ℓ_∞ adversary used a step size $\alpha = 0.01$ within a radius of $\epsilon = 0.3$ for 50 iterations.

The ℓ_2 adversary used a step size $\alpha = 0.1$ within a radius of $\epsilon = 2.0$ for 100 iterations.

The ℓ_1 adversary used a step size of $\alpha = 0.8$ within a radius of $\epsilon = 10$ for 50 iterations. By default the attack is run with two restarts, once starting with $\delta = 0$ and once by randomly initializing δ in the allowable perturbation ball. $k_1 = 5$, $k_2 = 20$ as described in A.1.

At test time, we increase the number of iterations to $(100, 200, 100)$ for $(\ell_\infty, \ell_2, \ell_1)$.

CIFAR10

The ℓ_∞ adversary used a step size $\alpha = 0.003$ within a radius of $\epsilon = 0.03$ for 40 iterations.

The ℓ_2 adversary used a step size $\alpha = 0.05$ within a radius of $\epsilon = 0.5$ for 50 iterations.

The ℓ_1 adversary used a step size $\alpha = 1.0$ with $\epsilon = 12$ for 50 iterations. $k_1 = 5$, $k_2 = 20$ as described in A.1.

At test time, we increase the number of iterations to $(100, 500, 100)$ for $(\ell_\infty, \ell_2, \ell_1)$.

C.2. Training hyperparameters

In this section, we describe the parameters used for adversarial training.

Table 4: Summary of adversarial accuracy results for CIFAR10

	P_∞	P_2	P_1	MAX	AVG	MSD
Clean accuracy	83.3%	90.2%	73.3%	81.0%	84.6%	81.1%
PGD-ℓ_∞	50.3%	48.4%	29.8%	44.9%	42.8%	48.0%
FGSM	57.4%	43.4%	12.7%	54.9%	51.9%	53.7%
PGD-Foolbox	52.3%	28.5%	0.6%	48.9%	44.6%	53.5%
MIM	52.7%	30.4%	0.7%	49.9%	46.1%	50.7%
ℓ_∞ attacks ($\epsilon = 0.03$)	50.7%	28.3%	0.2%	44.9%	42.5%	48.0%
PGD-ℓ_2	59.0%	62.1%	28.9%	64.1%	66.9%	66.6%
PGD-Foolbox	61.6%	64.1%	4.9%	65.0%	68.0%	68.2%
Gaussian Noise	82.2%	89.8%	62.3%	81.3%	84.3%	80.9%
Boundary Attack	65.5%	67.9%	2.3%	64.4%	69.2%	69.4%
DeepFool	62.2%	67.3%	0.9%	64.4%	67.4%	66.1%
Pointwise Attack	80.4%	88.6%	46.2%	78.9%	83.8%	79.8%
DDN	60.0%	63.5%	0.1%	64.5%	67.7%	67.0%
CWL2	62.0%	71.6%	0.1%	66.9%	71.5%	64.7%
ℓ_2 attacks ($\epsilon = 0.05$)	57.3%	61.6%	0.0%	61.7%	65.0%	64.3%
PGD-ℓ_1	16.5%	49.2%	69.1%	39.5%	54.0%	53.4%
Salt & Pepper	63.4%	74.2%	35.5%	75.2%	80.7%	73.9%
Pointwise Attack	49.6%	62.4%	8.4%	63.3%	77.0%	69.7%
ℓ_1 attacks ($\epsilon = 12$)	16.0%	46.6%	7.9%	39.4%	54.0%	53.0%
All attacks	15.6%	27.5%	0.0%	34.9%	40.6%	**47.0**%

MNIST For all the models, we used the Adam optimizer without weight decay, and used a variation of the learning rate schedule from Smith (2018), which is piecewise linear from 0 to 10^{-3} over the first 6 epochs, and down to 0 over the last 9 epochs.

We perform a large hyperparameter search for each of the MAX, AVG, MSD models, by training them for 15 epochs on all combinations of the following step sizes: α_1 = {0.75, 0.8, 1.0, 2.0}, α_2 = {0.1, 0.2}, α_∞ = {0.01, 0.02, 0.03}. Also, we find that setting the maximum value of learning rate to 10^{-3} works best among other values that we experiment on.

The MSD adversary used step sizes of $\alpha = (0.01, 0.1, 0.8)$ for the $(\ell_\infty, \ell_2, \ell_1)$ directions within a radius of $\epsilon = (0.3, 2.0, 10)$ for 100 iterations.

The MAX approach used step sizes of $\alpha = (0.01, 0.1, 1.0)$ for the $(\ell_\infty, \ell_2, \ell_1)$ directions within a radius of $\epsilon = (0.3, 2.0, 12)$ for (50, 100, 100) iterations respectively. We had to make an early stop at the end of the fourth epoch, since further training made the model biased towards ℓ_∞ robustness. We also had to increase the number of restarts and attack iterations for the ℓ_1 PGD attack.

The AVG approach used step sizes of $\alpha = (0.01, 0.2, 1.0)$ for the $(\ell_\infty, \ell_2, \ell_1)$ directions within a radius of $\epsilon = (0.3, 2.0, 12)$ for (50, 100, 50) iterations respectively. Note that we had to change the perturbation model for the ℓ_1 adversary to make it relatively stronger in-order to "balance" the trade-offs between different perturbation models.

Finally, we train the standard P_1, P_2, P_∞ models for an extended period till 20 epochs with respective step sizes α_1 = 1.0, α_2 = 0.1, and α_∞ = 0.01.

CIFAR10 For all the models, we used the SGD optimizer with momentum 0.9 and weight decay $5 \cdot 10^{-4}$. We used a variation of the learning rate schedule from Smith (2018) to achieve superconvergence in 50 epochs, which is piecewise linear from 0 to 0.1 over the first 20 epochs, down to 0.005 over the next 20 epochs, and finally back down to 0 in the last 10 epochs.

The MSD adversary used step sizes of α = (0.003, 0.02, 1.0) for the $(\ell_\infty, \ell_2, \ell_1)$ directions within a radius of $\epsilon = (0.03, 0.5, 12)$ for 50 iterations.

The MAX adversary used step sizes of α = (0.005, 0.05, 1.0) for the $(\ell_\infty, \ell_2, \ell_1)$ directions within a radius of $\epsilon = (0.03, 0.3, 12)$ for (40, 50, 50) iterations respectively. We do an early stop at epoch 45 for best accuracy.

The AVG adversary used step sizes of $\alpha = (0.003, 0.05, 1.0)$ for the $(\ell_\infty, \ell_2, \ell_1)$ directions within a radius of $\epsilon = (0.03, 0.3, 12)$ for (40, 50, 50) iterations respectively.

Note: For obtaining the best-performing MAX and AVG models, we artificially balance the size of the ℓ_2 perturbation region, reducing its radius to 0.3 from the actual threat model of radius 0.5.

Table 5: Comparison with Tramèr & Boneh (2019) on MNIST (higher is better). Results for all models except MSD are taken as is from Tramèr & Boneh (2019)

	Vanilla	Adv_∞	Adv_1	Adv_2	Adv_{AVG}	Adv_{MAX}	**MSD**
Clean accuracy	99.4%	99.1%	98.9%	98.5%	97.3%	97.2%	98.3%
ℓ_∞ attacks ($\epsilon = 0.3$)	0.0%	91.1%	0.0%	0.4%	76.7%	71.7%	75.9%
ℓ_2 attacks ($\epsilon = 2.0$)	12.4%	12.1%	50.6%	71.8%	58.3%	56.0%	67.9%
ℓ_1 attacks ($\epsilon = 10$)	8.5%	11.3%	78.5%	68.0%	53.9%	62.6%	74.8%
All attacks	0.0%	6.8%	0.0%	0.4%	49.9%	52.4%	**65.2**%

Table 6: Comparison with Tramèr & Boneh (2019) on CIFAR10 (higher is better). Results for all models except MSD are taken as is from (Tramèr & Boneh, 2019)

	Vanilla	Adv_∞	Adv_1	Adv_{AVG}	Adv_{MAX}	**MSD**
Clean accuracy	95.7%	92.0%	90.8%	91.1%	91.2%	92.0%
ℓ_∞ attacks ($\epsilon = \frac{4}{255}$)	0.0%	71.0%	53.4%	64.1%	65.7%	66.8%
ℓ_1 attacks ($\epsilon = \frac{2000}{255}$)	0.0%	16.4%	66.2%	60.8%	62.5%	65.3%
All attacks	0.0%	16.4%	53.1%	59.4%	61.1%	**63.2**%

D. Comparison with Tramèr & Boneh (2019)

In this section, we compare the results of our trained MSD model with that of Tramèr & Boneh (2019), who study the theoretical and empirical trade-offs of adversarial robustness in various settings when defending against multiple adversaries. Training methods presented by them in their comparisons, namely Adv_{AVG} and Adv_{MAX} closely resemble the simpler approaches discussed in this paper: AVG and MAX respectively. We use the results as is from their work, and additionally compare the position of our MSD models at the revised thresholds used by Tramèr & Boneh (2019). We make our best attempt at replicating the same attack strengths as of those used in the evaluation in Tramèr & Boneh (2019). We use all attacks from the Foolbox library, apart from the PGD ℓ_1 or SLIDE attack (Tramèr & Boneh, 2019). Further, we do not make multiple random restarts for these comparisons, which is in line with their evaluation.

The results of Tables 5 and 6 show that the relative advantage of MSD over simpler techniques does hold up. The MSD model was not retrained for the comparison on the MNIST dataset since it was trained to be robust to the same perturbation region in the main paper as well.

In case of CIFAR10, we train a model using the WideResNet architecture (Zagoruyko & Komodakis, 2016) with 5 residual blocks and a widening factor of 10, as used by Tramèr & Boneh (2019). It may be noted that this model has 4 times more parameters than the pre-activation version of ResNet which was used for the comparisons in the main paper. Further, for the CIFAR10 results in Table 6, the models are trained and tested only for ℓ_∞ and ℓ_1 adversarial perturbations with $\epsilon = (\frac{4}{255}, \frac{2000}{255}) \sim (0.0157, 7.84)$. Note that the size of the perturbation regions considered in the main paper is strictly larger than these perturbation regions.

We emphasize that the evaluation method adopted in the main paper is stronger than that in this comparison. This may also be noted from the results in Table 5, where the same MSD model (without retraining) achieves nearly 7% higher accuracy of 65.2% against all attacks that were considered by Tramèr & Boneh (2019), while the same model achieved an overall robust accuracy of 58.4% in our evaluation in Table 1 in the main paper. These differences can be largely attributed to:

1. **Use of random restarts:** We observe in our experiments that using up to 10 restarts for all our attacks leads to a decrease in model accuracy from 5 to 10% across all models. Tramèr & Boneh do not mention restarting their attacks for these models and so the robust accuracies for their models in Tables 5, 6 could potentially be lowered with random restarts.

2. **Larger Suite of Attacks Used:** The attacks used by Tramèr & Boneh in case of the CIFAR10 dataset are PGD, EAD (Chen et al., 2017) and Pointwise Attack (Schott et al., 2019) for ℓ_1; PGD, C&W (Carlini & Wagner, 2017) and Boundary Attack (Brendel et al., 2017) for ℓ_2; and PGD for ℓ_∞. We use a more expansive suite of attacks as shown in Appendix B. Some attacks like DDN, which proved to be strong adversaries in most cases, were not considered by them.

Our observations re-emphasize the importance of performing multiple restarts and using a broad suite of attacks in order to be able to best determine the robust performance of a proposed algorithm.

E. Analyzing learned Filters for MNIST

As described in § 5.1, we use a simple 4 layer CNN model to classify MNIST digits. Each of the two convolutional layers has 5x5 filters. Specifically, the first layer contains 32 such filters. We begin our analysis by observing the learned filters of an ℓ_∞ robust model. We observe that many of the learned filters are extremely sparse with only one non-zero element as shown in Figure 6a. Interestingly, such a view is unique to the case of the ℓ_∞ robust model and is not observed in ℓ_2 (Figure 6b) and ℓ_1 (Figure 6c) robust models.

The presence of such learned filters that act as thresholding filters, due to the immediately followed activation layer, has been hypothesized to be the reason for gradient masking in such models by Madry et al. (2018); Tramèr & Boneh (2019). The hypothesis is in line with our experimental correlations of ℓ_∞ model being the only standard model that performs poorly against decision-based adversaries while being significantly robust to first-order adversaries. Therefore, we go beyond this preliminary analysis to observe the initial layers of MSD (Figures 7a, 7b), MAX (Figures 8a, 8b), AVG (Figures 9a, 9b) models. In all the three cases, we have two models that are almost identically trained, but with different ℓ_∞ step sizes: α_∞ = 0.01 on the left and α_∞ = 0.03 on the right. While we display results only on two extreme settings of relative attack step-sizes, we find that changing the relative step size of different PGD adversaries can help reduce the number of thresholding filters in the MSD approach, which also leads to better accuracies against decision-based attacks like the Pointwise Attack. However, the MAX and AVG models are nearly invariant to the individual attack step-sizes.

As a result, in order to achieve reasonable performance in case of MAX and AVG models against decision-based attacks, we had to employ methods to manipulate the perturbation models in an 'ad-hoc' manner. More specifically, in case of MAX we had to increase the number of restarts of the ℓ_1 attack during training, and perform an early stop at the end of the fourth epoch (Figure 10a) since further training biased the model towards ℓ_∞ robustness, and made it susceptible to decision-based attacks. In case of AVG, we had to increase the maximum radius of the ℓ_1 attack to 12 (Figure 10b). It is worth noting that both the approaches help cosmetically strengthen the relative effect of the ℓ_1 attack and help reduce the number of sparse filters. We observe that these models perform significantly better against decision-based attacks as opposed to those in Figures 8, 9.

Finally, we emphasize that while learning sparse convolution filters and the susceptibility to gradient-free attacks is often correlated, there is *no* consistent relation between the "number" of such filters and the final model performance or the presence of gradient masking. We perform this empirical analysis for completeness to follow up on previous work by Madry et al. (2018), and it comes with no formal statements. In fact, a model may perform better against decision-based attacks even if it has more sparse filters than another model. We hope that these preliminary observations encourage further exploration around the phenomenon of gradient masking in adversarially robust models.

Table 7: Performance on CIFAR-10-C

	Accuracy
Standard model	66.0%
P_∞	75.0%
P_2	82.7%
P_1	57.8%
MAX	70.8%
AVG	76.8%
MSD	74.2%

F. Attacks outside the perturbation model

In this section, we present some additional experiments exploring the performance of our model on attacks which lie outside the perturbation model. Note that this is presented only for exploratory reasons and there is no principled reason why the adversarial defenses should generalize beyond the perturbation model defended against.

Common corruptions We measure the performance of all the models on CIFAR-10-C, which is a CIFAR10 benchmark which has had common corruptions applied to it (e.g. noise, blur, and compression). We report the results in Table 7. We find that that, apart from the P_1 model, the rest achieve some improved robustness against these common corruptions above the standard CIFAR10 model.

Defending against ℓ_1 and ℓ_∞ and evaluating on ℓ_2 We also briefly study what happens when one trains against ℓ_1 and ℓ_∞ perturbation models, while evaluating against the ℓ_2 adversary. Specifically, we take the MSD approach on MNIST and simply remove the ℓ_2 adversary from the perturbation model. This results in a model which has its ℓ_1 and ℓ_∞ robust performance against a PGD adversary drop by 1% and its ℓ_2 robust performance against a PGD adversary (which it was not trained for) drops by 2% in comparison to the original MSD approach on all three perturbation models.

As a result, we empirically observe that including the ℓ_2 perturbation model in this setting actually improved overall robustness against all three perturbation models. Unsurprisingly, the ℓ_2 performance drops to some degree, but the model does not lose all of its robustness.

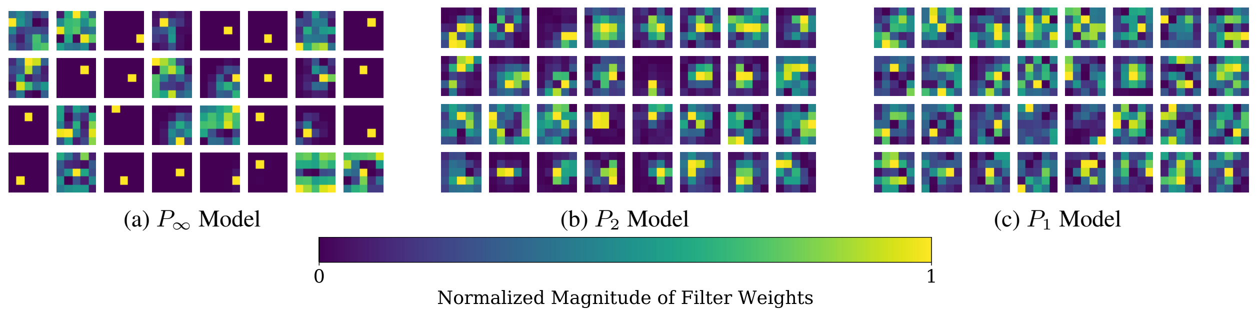


Figure 6: A view of each of the (5x5) learned filters of the first layer of P_∞, P_2, P_1 models trained on the MNIST dataset. While there are many learned filters in the P_∞ model that have only one non-zero element (rest of the values are nearly zero), such a phenomenon is absent in P_2, P_1 models.

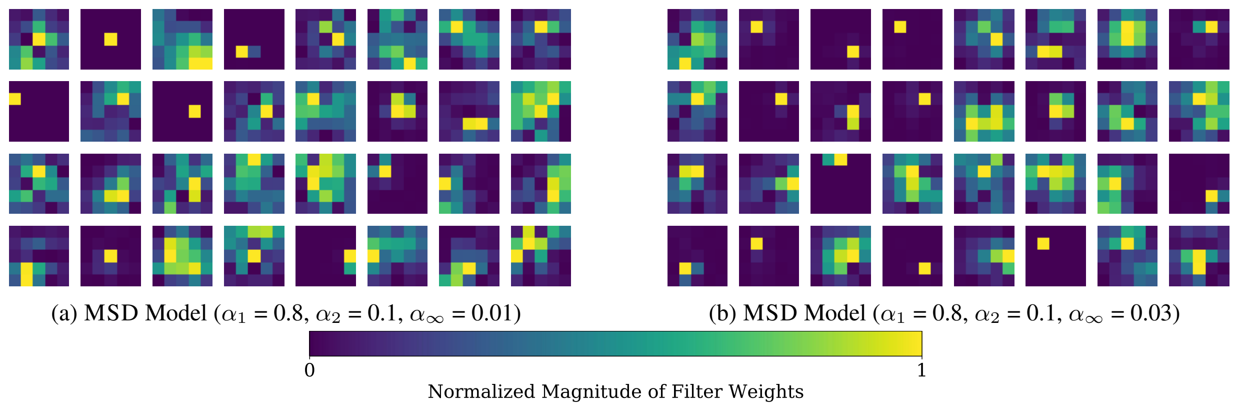


Figure 7: A view of each of the (5x5) learned filters of the first layer of MSD models trained on the MNIST dataset. The training hyper-parameters for the left and right images only differ in the step-size for the ℓ_∞ attack, where $\alpha_\infty = 0.01$ for the left and $\alpha_\infty = 0.03$ for the right image. The figure suggests how adjusting the relative step-sizes can help reduce the occurrence of sparse filters in case of MSD models.

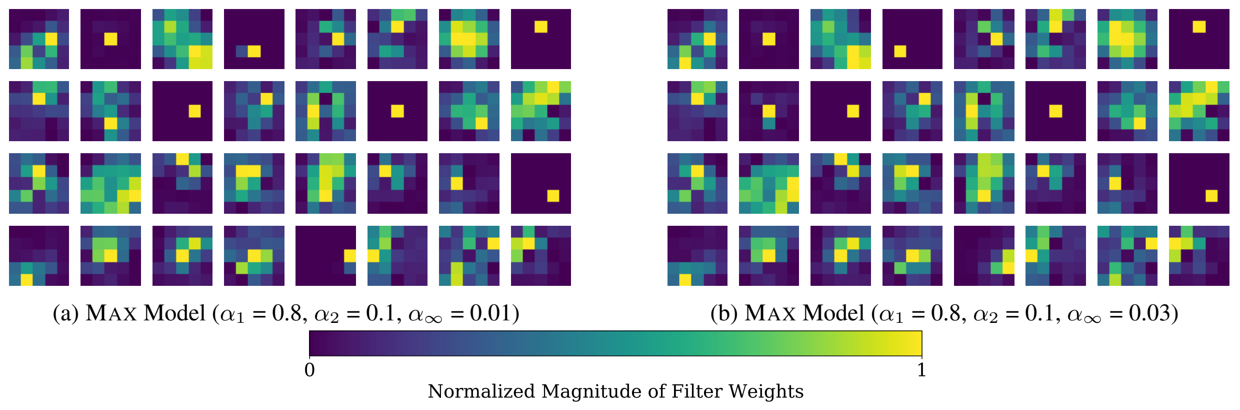
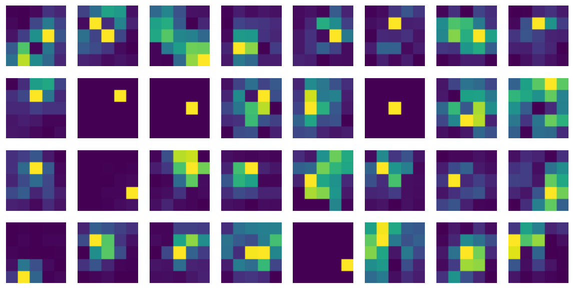
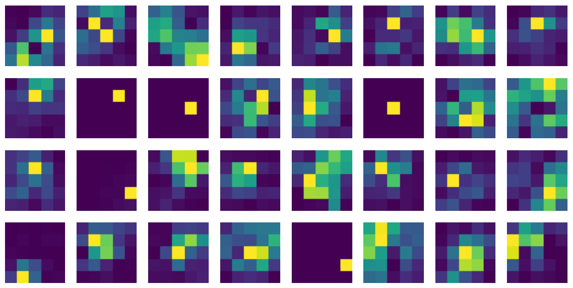


Figure 8: A view of each of the (5x5) learned filters of the first layer of MAX models trained on the MNIST dataset. The training hyper-parameters for the left and right images only differ in the step-size for the ℓ_∞ attack, where $\alpha_\infty = 0.01$ for the left and $\alpha_\infty = 0.03$ for the right image. The learned filters are nearly identical for both models and indicate how there may not be a natural way of balancing the trade-offs between different perturbation models in the training schedule for MAX models.


(a) AVG Model ($\alpha_1 = 0.8$, $\alpha_2 = 0.1$, $\alpha_\infty = 0.01$)


(b) AVG Model ($\alpha_1 = 0.8$, $\alpha_2 = 0.1$, $\alpha_\infty = 0.03$)

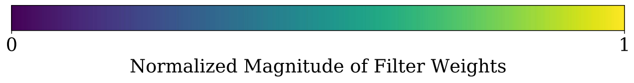
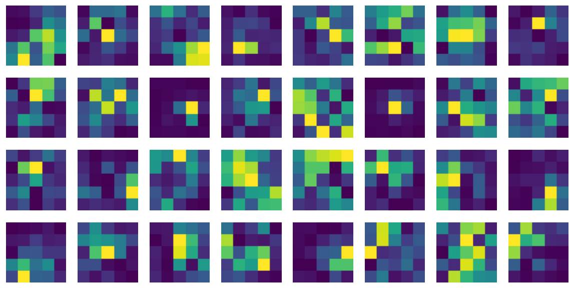
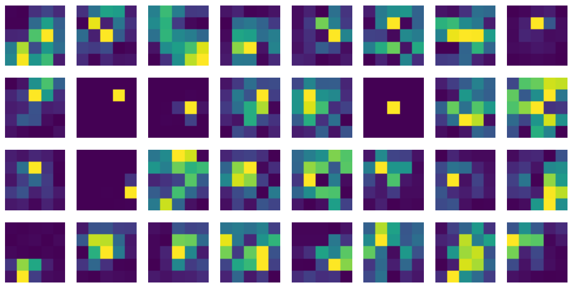


Figure 9: A view of each of the (5x5) learned filters of the first layer of AVG models trained on the MNIST dataset. The training hyper-parameters for the left and right images only differ in the step-size for the ℓ_∞ attack, where $\alpha_\infty = 0.01$ for the left and $\alpha_\infty = 0.03$ for the right image. The learned filters are nearly identical for both models and indicate how there may not be a natural way of balancing the trade-offs between different perturbation models in the training schedule for AVG models.


(a) Final MAX Model


(b) Final AVG Model

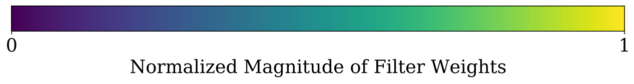


Figure 10: A view of each of the (5x5) learned filters of the first layer of MAX and AVG models trained on the MNIST dataset. These models are not susceptible to decision-based attacks as opposed to those in Figures 8, 9. Notably, we had to employ 'ad-hoc' techniques to manipulate the individual perturbation models to be able to train these models. However, even after such manipulations, the accuracy against the worst-case adversary in the union of ℓ_∞, ℓ_2, ℓ_1 perturbation models for MAX, AVG approaches is considerably worse than the MSD approach.