Supplementary Materials: Multitask Learning Strengthens Adversarial Robustness

1 Proof for Theoretical Analysis

We present theoretical analysis to quantify how much multi-task learning improves model's overall adversarial robustness.

Definition 1. *Given classifier F, input $\mathbf{x}$, output target $\mathbf{y}$, and loss $\mathcal{L}(\mathbf{x}, \mathbf{y}) = \ell(F(\mathbf{x}), \mathbf{y})$, the feasible adversarial examples lie in a p-norm bounded ball with radius r, $B(\mathbf{x}, r) := \{\mathbf{x}_{adv}, ||\mathbf{x}_{adv} - \mathbf{x}||_p < r\}$. Then adversarial vulnerability of a classifier over the whole dataset is*

$$\mathbb{E}_{\mathbf{x}}[\Delta\mathcal{L}(\mathbf{x}, \mathbf{y}, r)] = \mathbb{E}_{\mathbf{x}}[\max_{||\delta||_p<r} |\mathcal{L}(\mathbf{x}, \mathbf{y}) - \mathcal{L}(\mathbf{x} + \delta, \mathbf{y})|].$$

$\Delta\mathcal{L}$ captures the change of output loss given a change in input. Intuitively, a robust model should have a smaller change in loss given a perturbation of the input. Given the adversarial noise is imperceptible, i.e., $r \to 0$, we can approximate $\Delta\mathcal{L}$ with a first-order Taylor expansion, where

$$|\mathcal{L}(\mathbf{x}, \mathbf{y}) - \mathcal{L}(\mathbf{x} + \delta, \mathbf{y})| = |\partial_{\mathbf{x}}\mathcal{L}(\mathbf{x}, \mathbf{y})\delta + O(\delta)|$$

Lemma 1. *For a given neural network F that predicts multiple tasks, the adversarial vulnerability is*

$$\mathbb{E}_{\mathbf{x}}[\Delta\mathcal{L}(\mathbf{x}, \mathbf{y}, r)] \approx \mathbb{E}_{\mathbf{x}}\left[||\partial_{\mathbf{x}}\mathcal{L}_{all}(\mathbf{x}, \bar{\mathbf{y}})||_q\right] \cdot ||\delta||_p \propto \mathbb{E}_{\mathbf{x}}\left[||\partial_{\mathbf{x}}\mathcal{L}_{all}(\mathbf{x}, \bar{\mathbf{y}})||_q\right]$$

Proof. According to the definition of dual norm:

$$\Delta\mathcal{L} \approx \max_{||\delta||_p<r} |\partial_{\mathbf{x}}\mathcal{L}(\mathbf{x}, \mathbf{y})\delta| = ||\partial_{\mathbf{x}}\mathcal{L}_{all}(\mathbf{x}, \bar{\mathbf{y}})||_q \cdot ||\delta||_p$$

$$\mathbb{E}_{\mathbf{x}}[\Delta\mathcal{L}] \approx \mathbb{E}_{\mathbf{x}}\left[||\partial_{\mathbf{x}}\mathcal{L}_{all}(\mathbf{x}, \bar{\mathbf{y}})||_q\right] \cdot ||\delta||_p$$

where q is the dual norm of p, which satisfies $\frac{1}{p} + \frac{1}{q} = 1$ and $1 \leq p \leq \infty$.

Once given the p-norm bounded ball, i.e., $||\delta||_p$ is constant, we get

$$\mathbb{E}_{\mathbf{x}}[\Delta\mathcal{L}] \propto \mathbb{E}_{\mathbf{x}}\left[||\partial_{\mathbf{x}}\mathcal{L}_{all}(\mathbf{x}, \bar{\mathbf{y}})||_q\right]$$

□

Theorem 1. ***(Adversarial Vulnerability of Model for Multiple Correlated Tasks)*** *If the selected output tasks are correlated with each other such that the covariance between the gradient of task i and task j is $\mathrm{Cov}(\mathbf{r_i}, \mathbf{r_j})$, and the gradient for each task is i.i.d. with zero mean (because the model is converged), then adversarial vulnerability of the given model is proportional to*

$$\frac{\sqrt{(1 + \frac{2}{M}\sum_{i=1}^{M}\sum_{j=1}^{i-1}\frac{\mathrm{Cov}(\mathbf{r_i}, \mathbf{r_j})}{\mathrm{Cov}(\mathbf{r_i}, \mathbf{r_i})})}}{\sqrt{M}}$$

where M is the number of output tasks selected.

Proof. Denote the gradient for task c as $\mathbf{r_c}$, i.e.,

$$\mathbf{r_c} = \partial_{\mathbf{x}} \mathcal{L}_{\mathbf{c}}(\mathbf{x}, \mathbf{y_c})$$

We define the joint gradient vector $\mathbf{R}$ as follows:

$$\mathbf{R} = \partial_{\mathbf{x}} \mathcal{L}_{all}(\mathbf{x}, \overline{\mathbf{y}}) = \partial_{\mathbf{x}} \left(\frac{1}{M} \sum_{c=1}^{M} \mathcal{L}_c(\mathbf{x}, \mathbf{y}_c) \right) = \frac{1}{M} \sum_{c=1}^{M} \partial_{\mathbf{x}} \mathcal{L}_c(\mathbf{x}, \mathbf{y}_c) = \sum_{c=1}^{M} \mathbf{r_c}$$

As we can see, the joint gradient is the sum of gradients from each individual task. Then we consider the expectation of the square of the L_2 norm of the joint gradient:

$$\mathbb{E}(\|\mathbf{R}\|_2^2) = \mathbb{E}\left(\| \frac{1}{M} \sum_{c=1}^{M} \mathbf{r_c} \|_2^2 \right) = \frac{1}{M^2} \mathbb{E}\left(\sum_{c=1}^{M} \|\mathbf{r_c}\|^{\mathbf{2}} + 2 \sum_{i=1}^{M} \sum_{j=1}^{i-1} \mathbf{r_i} \mathbf{r_j} \right)$$

$$\mathbb{E}(\|\mathbf{R}\|_2^2) = \frac{1}{M^2} \left(\sum_{i=1}^{M} \mathbb{E}\|\mathbf{r_i}\|^{\mathbf{2}} + 2 \sum_{i=1}^{M} \sum_{j=1}^{i} \mathbb{E}(\mathbf{r_i} \mathbf{r_j}) \right)$$

Since

$$\mathrm{Cov}(\mathbf{r_i}, \mathbf{r_j}) = \mathbb{E}(\mathbf{r_i} \mathbf{r_j}) - \mathbb{E}(\mathbf{r_i}) \mathbb{E}(\mathbf{r_j})$$

According to the assumption

$$\mathbb{E}(\mathbf{r_j}) = \mathbf{0}$$

We know

$$\mathrm{Cov}(\mathbf{r_i}, \mathbf{r_j}) = \mathbb{E}(\mathbf{r_i} \mathbf{r_j})$$

Then we get

$$\mathbb{E}(\|\mathbf{R}\|_2^2) = \frac{1}{M^2} \left(\sum_{i=1}^{M} \mathbb{E}(\mathrm{Cov}(\mathbf{r_i}, \mathbf{r_i}) \right) + 2 \sum_{i=1}^{M} \sum_{j=1}^{i} \mathbb{E}(\mathrm{Cov}(\mathbf{r_i}, \mathbf{r_j})) = \frac{1}{M^2} \left(\sum_{i=1}^{M} \sigma^2 + 2 \sum_{i=1}^{M} \sum_{j=1}^{i} \mathbb{E}[\mathrm{Cov}(\mathbf{r_i}, \mathbf{r_j})] \right)$$

where $\sigma^2 = \mathrm{Cov}(\mathbf{r_i}, \mathbf{r_i})$

Thus, the adversarial vulnerability is:

$$\mathbb{E}_{\mathbf{x}}[\Delta \mathcal{L}] \propto \mathbb{E}_{\mathbf{x}} \left[\|\partial_{\mathbf{x}} \mathcal{L}_{all}(\mathbf{x}, \overline{\mathbf{y}})\|_2 \right] = \frac{\sqrt{(1 + \frac{2}{M} \sum_{i=1}^{M} \sum_{j=1}^{i-1} \frac{\mathrm{Cov}(\mathbf{r_i}, \mathbf{r_j})}{\mathrm{Cov}(\mathbf{r_i}, \mathbf{r_i})})}}{\sqrt{M}}$$

□

For a special case where all the tasks are independent of each other, independent gradients with respect to the input are produced, we have the following corollary:

Corollary 1.1. ***(Adversarial Vulnerability of Model for Multiple Independent Tasks)*** *If the output tasks selected are independent of each other, and the gradient for each task is i.i.d. with zero mean, then the adversarial vulnerability of given model is proportional to $\frac{1}{\sqrt{M}}$, where M is the number of independent output tasks selected.*

Proof. According to the independent assumption, we have

$$\mathrm{Cov}(\mathbf{r_i}, \mathbf{r_j}) = \mathbf{0}$$

Let $\sigma^2 = \mathrm{Cov}(\mathbf{r_i}, \mathbf{r_i})$. Thus we get the adversarial vulnerability to be:

$$\mathbb{E}_{\mathbf{x}}[\Delta \mathcal{L}] \propto \mathbb{E}_{\mathbf{x}} \left[||\partial_{\mathbf{x}} \mathcal{L}_{all}(\mathbf{x}, \overline{\mathbf{y}})||_2 \right] = \sqrt{\frac{\sigma^2}{M}} \propto \frac{1}{\sqrt{M}}$$

□

2 Experimental Setup

2.1 Cityscapes

We train DRN-105 model and evaluate against multi-task attack. We follow the original architecture setup of the original DRN paper [4]. We used 93 layers in the shared backbone encoder network, and 13 layers in the decoder branch for individual task prediction. We use a batch size of 24. We start with a learning rate of 0.01 and decrease the learning rate by a factor of 10 after every 100 epochs. We trained the model for 250 epochs.

We train multi-task model against single task attack using DRN-22 model. We use 18 layers in the shared backbone encoder network, and 9 layers in the decoder branch for individual task prediction. We use batch size of 32. We optimize with SGD, with learning rate of 0.01, then decrease it to 0.001 at 180 epoch. We train model for 200 epoch in total. We applied a weight decay of 0.0001 for all the models.

2.2 Taskonomy

Taskonomy dataset [5] consists of millions of indoor scenes with labels for multiple tasks, we use 11 tasks including semantic segmentation, depth estimation, 2D and 3D edge detection, normal vector estimation, reshading, 2D and 3D keypoint detection, Euclidean depth, auto-encoding, and principal curvature estimation. We use the publicly available Tiny version of dataset, which consists of 9464 images from 1500 rooms. We use examples from 80% of the rooms as training data and examples from 20% of the rooms as test data. Images from the same room are only contained in either the training set or the test set, and not in both. The quality of the model is measured by its ability to generalize to new rooms.

For learning a multi-task model for joint robustness, we follow the set up described in [3]. We train a ResNet-18 as the shared backbone encoder network for all the tasks. Each multi-task model consists of 1 to 6 different tasks. We use an input size of 512×512. We use an 8 layer decoder for each individual task prediction. Following the data preprocessing of [3], we apply equal weights to all the tasks. Start from task "semantic segmentation" (s), we add tasks "depth" (d), "edge texture" (e), "keypoints 2d" (k), "normal" (n), and "reshading" (r). Thus we train 6 models 's,' 'sd,' 'sde,' 'sdek,' 'sdekn,' 'sdeknr.' We also train 'd,' 'e,' 'er,' 'k,' 'ks,' 'ksd' tasks, so that we can analysis the trend of 4 tasks' performance after multitask learning. We use the same learning rate schedule for all the models — SGD with learning rate 0.01 and momentum 0.99. We decrease the learning rate at 100 epoch by 10 times. We train all the models for 150 epoch. Results are shown in Figure 5 in the main paper.

For training robust models on select tasks, we use ResNet-18 as the shared encoder network. We select 11 tasks trained in pairs with each other, which results in 110 models. We study their robustness under a single-task attack. We follow the data processing in [5]. For each task we considered, we try weights of 0.1 and 0.01 for the auxiliary task, and choose the weight that produces higher robust accuracy. The chosen λ_a for the auxiliary tasks are shown in Table 2. The selection of weights is important due to the complex interactions of different tasks [1]. We follow the setup in [5], and subsample the image from 512 to 256 using linear interpolation. For segmentation, reshading, keypoint 3D, depth Euclidean, Auto Encoder, principle curvature, we use SGD, with learning rate 0.01 and decrease by 10 times at 140 epoch. For the other tasks we use adam, with learning rate 0.001 and decrease by 10 times at 120 and 140 epoch. Due to the inherent difference between different tasks, we use different optimizer for different tasks for better convergence. All the models are trained for 150 epoch. All the results are shown in Table 1. As we can see, learning versatile, multi-task models improves adversarial robustness on 90/110 tasks.

2.3 Adversarial Training

We present the details for multi-task adversarial training in Algorithm 1. For single task model, we choose $S = \{\{T_m\}\}$. The algorithm is the same as the adversarial training procedure of Madry et. al. [2]. For multi-task model, we set $S = \{\{T_m\}, \{T_m, T_a^{(1)}, ...,\}\}$, thus the generated adversarial images under multi-task are more diversified compared with single-task models. In addition, all the adversarial examples are trained on multi-task loss function, where the auxiliary task can introduce useful knowledge for learning the robust main task. We use $\lambda = 0.01$ for the auxiliary task. For all the task, we train using SGD optimizer with batch size of 32, for 200 epoch. We start with learning rate of 0.01, and decrease the learning rate by 10 times at 180 epoch. The experiment are conducted on Cityscapes dataset.

						PGD Adversarial						
	Baseline	SemSeg	DepthZ	Edge2D	Normal	Reshad	Key2D	Key3D	DepthE	AutoE	Edge3D	PCurve
Semseg *	13.360	—	$\boxed{\mathbf{19.320}}$	**13.950**	**16.630**	**14.580**	**15.700**	**13.780**	**14.910**	**14.110**	**14.720**	**14.900**
DepthZ (10^{-2})	11.491	$\boxed{\mathbf{4.712}}$	—	**6.780**	11.617	12.412	**11.120**	**8.358**	**10.498**	**4.981**	12.230	**5.035**
Edge2D (10^{-2})	10.672	**9.841**	$\boxed{\mathbf{9.363}}$	—	**9.546**	**9.943**	**9.732**	**9.654**	**9.714**	**9.941**	**9.978**	**10.095**
Normal (10^{-2})	40.926	$\boxed{\mathbf{35.171}}$	42.871	**39.335**	—	**40.501**	**39.462**	**39.930**	42.071	**35.726**	**37.070**	41.212
Reshad (10^{-2})	57.900	$\boxed{\mathbf{48.800}}$	**57.800**	**55.000**	**56.500**	—	**55.900**	**53.300**	60.000	61.000	**49.300**	**57.600**
Key2D (10^{-2})	11.700	**10.900**	**10.900**	**10.700**	**10.500**	**10.900**	—	**11.000**	**10.600**	**11.000**	**10.800**	$\boxed{\mathbf{10.600}}$
Key3D (10^{-2})	49.700	$\boxed{\mathbf{31.000}}$	**49.600**	**50.800**	**45.900**	**42.200**	**43.800**	—	51.200	**32.600**	53.400	52.900
DepthE (10^{-3})	4.850	**3.530**	**3.390**	**3.250**	**4.270**	5.670	**3.670**	**3.730**	—	**3.700**	**3.330**	$\boxed{\mathbf{2.930}}$
AutoE (10^{-2})	59.300	$\boxed{\mathbf{57.800}}$	60.300	**58.300**	62.300	59.400	59.300	60.700	**58.200**	—	60.500	61.500
Edge3D (10^{-2})	15.900	**14.600**	**15.300**	16.300	**15.400**	**15.200**	**15.600**	16.900	**15.400**	$\boxed{\mathbf{12.600}}$	—	**14.800**
PCurve (10^{-4})	11.500	**8.920**	$\boxed{\mathbf{8.900}}$	**10.400**	**9.230**	**9.620**	$\boxed{\mathbf{8.900}}$	**10.400**	**11.100**	**9.190**	**10.400**	—
						Clean						
SemSeg *	43.190	—	**46.300**	**46.180**	$\boxed{\mathbf{46.350}}$	**46.240**	**45.440**	**45.620**	**44.690**	**44.500**	**45.320**	**44.490**
DepthZ (10^{-2})	2.852	$\boxed{\mathbf{2.734}}$	—	3.880	**2.846**	**2.505**	2.874	3.562	3.339	3.171	3.088	4.690
Edge2D (10^{-2})	3.384	3.922	**3.382**	—	3.507	3.435	**3.330**	3.522	3.433	3.574	3.569	3.454
Normal (10^{-2})	6.997	7.181	7.093	7.006	—	**6.989**	**6.990**	7.182	**6.940**	$\boxed{\mathbf{6.864}}$	**6.931**	7.141
Reshad (10^{-2})	8.027	**7.985**	8.103	**7.941**	**7.901**	—	8.041	**7.957**	**7.940**	$\boxed{\mathbf{7.890}}$	8.065	8.150
Key2D (10^{-2})	4.156	**4.116**	**3.897**	$\boxed{\mathbf{3.795}}$	**3.865**	**4.147**	—	**3.944**	**3.857**	**3.823**	**3.850**	**3.878**
Key3D (10^{-2})	8.771	**8.445**	**8.686**	**8.514**	**8.610**	$\boxed{\mathbf{8.318}}$	**8.703**	—	**8.492**	**8.366**	**8.362**	**8.578**
DepthE (10^{-3})	6.373	6.575	**5.946**	**6.350**	**6.236**	**5.802**	6.418	6.470	—	**5.948**	$\boxed{\mathbf{5.715}}$	**6.251**
AutoE (10^{-2})	3.470	3.616	3.709	3.548	3.587	3.540	3.780	3.761	3.542	—	3.530	3.553
Edge3D (10^{-2})	4.649	4.695	**4.608**	4.727	**4.562**	4.725	**4.364**	**4.635**	**4.611**	**4.210**	—	$\boxed{\mathbf{3.703}}$
PCurve (10^{-4})	8.017	8.360	8.184	8.353	8.541	$\boxed{\mathbf{7.232}}$	**7.733**	**7.725**	8.153	**7.854**	**7.732**	—

Table 1: The absolute performance of all models trained on two tasks (Relative are shown in Figure 7 in the main paper). Each row in the first column lists the name of the main task. The second column (baseline) shows the performance of a model trained on a single task. The * in the row indicates the mIoU score for semantic segmentation, for which higher is better. The values in the other rows of the table show the l1 loss, for which lower is better. The (10^{-n}) in the first column indicates the unit for the error. 'SemSeg' denotes 'semantic segmentation,' 'DepthZ' denotes 'depth estimation,' 'Edge2D' denotes '2D edge detection,' 'Normal' denotes 'Normal Vector estimation', 'Reshad' denotes 'Reshading,' 'Key2D' denotes '2D Keypoint detection,' 'Key3D' denotes '3D Keypoint detection,' 'DepthE' denotes 'Euclidean depth,' 'AutoE' denotes 'Auto Encoder,' 'Edge3D' denotes '3D Edge detection,' 'PCurve' denotes 'Curvature estimation.' Values superior to the baseline are **bold**, and the best performance for each row is in a $\boxed{\textbf{box}}$. The table lists the IoU (large is better) for the segmentation model, and error (small is better) for all the other tasks. We pair each selected model with 11 other models. All the models converge after training for 150 epochs. Overall, training on two tasks can help the individual task's adversarial robustness on **90/110** cases, while surpassing the baseline's performance on the clean examples on **70/110**. For instance, the adversarial robustness for the semantic segmentation and keypoints3D estimation is always improved by multi-task learning while the clean accuracy also improves. The results on 11 tasks support our claim that training on multiple tasks improves adversarial robustness.

					λ_a						
	SemSeg	DepthZ	Edge2D	Normal	Reshad	Key2D	Key3D	DepthE	AutoE	Edge3D	PCurve
Semseg *	0	0.01	9.01	0.1	0.1	0.01	0.01	0.1	0.01	0.1	0.01
DepthZ	0.1	0	0.1	0.01	0.1	0.1	0.01	0.1	0.1	0.1	0.01
Edge2D	0.1	0.1	0	0.1	0.1	0.01	0.1	0.1	0.01	0.01	0.1
Normal	0.1	0.01	0.1	0	0.01	0.1	0.01	0.1	0.1	0.01	0.01
Reshad	0.01	0.1	0.01	0.01	0	0.1	0.01	0.01	0.1	0.01	0.1
Key2D	0.1	0.1	0.01	0.01	0.01	0	0.01	0.01	0.01	0.01	0.1
Key3D	0.1	0.01	0.1	0.1	0.1	0.1	0	0.1	0.1	0.01	0.01
DepthE	0.1	0.01	0.01	0.01	0.01	0.1	0.01	0	0.01	0.1	0.1
AutoE	0.1	0.01	0.01	0.1	0.01	0.1	0.01	0.1	0	0.01	0.1
Edge3D	0.1	0.01	0.01	0.01	0.01	0.1	0.01	0.01	0.1	0	0.1
PCurve	0.1	0.01	0.1	0.01	0.01	0.1	0.01	0.1	0.01	0.01	0

Table 2: The λ_a value for the auxiliary task for Figure 7 in the main paper.

Algorithm 1 Adversarial Training with Multi-task Learning

```
Input: Initialized networks $F_i$, dataset $D$, main task $T_m$, auxiliary task $T_a^{(i)}$. Construct multi-task combination set
$S = \{\{T_m\}, \{T_m, T_a^{(1)}, ..., \}\}$
Output:
  for number of training epochs do
    for number of iterations in each epoch do
      Sample minibatch of $n$ images $\mathbf{x}$ from $D$.
      for each task combination $S_t$ in $S$ do
        Let $\mathcal{L}_t(\mathbf{x}, \mathbf{y}) = \sum_i \lambda_i \ell(F_i(\mathbf{x}), \mathbf{y}_i)$, where $i = 1, ..., \#(S_t)$, $T_i \in S_t$.
        Compute adversarial attack images $\mathbf{x}_{adv}$
            $$\underset{\mathbf{x}_{adv}}{\operatorname{argmax}} \mathcal{L}_t(\mathbf{x}_{adv}, \mathbf{y}), \text{s.t.} ||\mathbf{x}_{adv} - \mathbf{x}||_p \le r$$
        Training the multi-task model using the generated attack image $\mathbf{x}_{adv}$ by optimizing the following loss
        function:
            $$\min \mathcal{L}_t(\mathbf{x}, \mathbf{y})$$
      end for
    end for
  end for
  return Neural network model $F_i$
```

References

[1] https://slideslive.com/38917690/multitask-learning-in-the-wilderness.

[2] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[3] Trevor Standley, Amir Roshan Zamir, Dawn Chen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? *arXiv:1905.07553*, 2019.

[4] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[5] Amir R Zamir, Alexander Sax, , William B Shen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.