Appendix

A. Theorems and Proofs

A.1. Proof of Theorem 3

Theorem 3 If $\mathcal{P} = \mathcal{N}(\mathbf{0}, \sigma_0^2)$, and $\mathcal{Q}$ is also a product of univariate Gaussian distributions, then the minimum of Eq. 4 w.r.t $\mathcal{Q}$ can be bounded by

$$\begin{aligned}
&\min_{\mathcal{Q}} \mathbb{E}_{\theta\in\mathcal{Q}} R(\theta, \mathcal{D}) \\
\leq &\min_{\mathcal{Q}} \{\mathbb{E}_{\theta\in\mathcal{Q}} \hat{R}(\theta, D^m) + \frac{1}{\beta}\mathrm{KL}(\mathcal{Q}||\mathcal{P})\} + C(\tau, \beta, m) \\
= &\min_{\theta} \{\hat{R}(\theta, D^m) + \frac{||\theta||_2^2}{2\beta\sigma_0^2} + \frac{\sigma_0^2}{2}\mathrm{Tr}(\nabla_\theta^2 \hat{R}(\theta, D^m))\} + C(\tau, \beta, m) + O(\sigma_0^4).
\end{aligned}$$

Proof. Before we begin our proof, we emphasize that the posterior $\mathcal{Q}$ that minimizes the test loss may not be the same posterior $\mathcal{Q}$ that minimizes the training loss. Namely, define

$$\begin{aligned}
\mathcal{Q}_{test} &\stackrel{\text{def}}{=} \arg\min_{\mathcal{Q}} \mathbb{E}_{\theta\in\mathcal{Q}} R(\theta, \mathcal{D}) \\
\mathcal{Q}_{train} &\stackrel{\text{def}}{=} \arg\min_{\mathcal{Q}} \{\mathbb{E}_{\theta\in\mathcal{Q}} \hat{R}(\theta, D^m) + \frac{1}{\beta}\mathrm{KL}(\mathcal{Q}||\mathcal{P})\},
\end{aligned}$$

then $\mathcal{Q}_{test}$ may not be equal to $\mathcal{Q}_{train}$. However, the following inequality clearly holds based on their definitions,

$$\mathbb{E}_{\theta\in\mathcal{Q}_{test}} R(\theta, \mathcal{D}) \leq \mathbb{E}_{\theta\in\mathcal{Q}_{train}} R(\theta, \mathcal{D}) \leq \mathbb{E}_{\theta\in\mathcal{Q}_{train}} \hat{R}(\theta, D^m) + \frac{1}{\beta}\mathrm{KL}(\mathcal{Q}||\mathcal{P}) + C(\tau, \beta, m).$$

Generally speaking, it is impossible to directly find $\mathcal{Q}_{test}$ without knowing the true data distribution $\mathcal{D}$. Thus, Theorem 3 attempts to derive $\mathcal{Q}_{train}$ to minimize the RHS of Eq. 4. When it is clear which optimal posterior, i.e. $\mathcal{Q}_{train}$ instead of $\mathcal{Q}_{test}$, we can derive in the theorem, we omit the subscription and directly write $\mathcal{Q}$.

Proof Overview. Based on our assumptions, we write $\mathcal{Q} = \mathcal{N}(\theta, \Sigma)$ where Σ is the (diagonal) covariance matrix. The minimization of the bound w.r.t $\mathcal{Q}$ is to minimize the bound w.r.t Σ and θ. Our proof is therefore three-fold: (1) firstly, we write the expression of $\mathrm{KL}(Q||P)$ using θ and Σ; (2) secondly, we use a second-order Taylor expansion to decompose $\mathbb{E}_{\theta\in\mathcal{Q}} \hat{R}(\theta, D^m)$ into terms as functions of Σ and θ; and (3) finally, we minimize the bound w.r.t Σ for two cases: all weights have the same or different variances. Namely, the diagonal of Σ has the same or different elements. At the end of the minimization w.r.t Σ, we arrive at the bound that requires to minimize θ to actually minimize the RHS. Our proof follows below.

Extra Notations. Throughout the proof we will use N for the total number of weights, i.e. the cardinality of θ. When indexing a particular weight, we write θ_n. Let the diagonal of Σ be $\sigma^2 = [\sigma_1^2, \sigma_2^2, \cdots, \sigma_N^2]^\top$, i.e. the variance of each weight. We use $\mathrm{Diag}(a)$ to denote the diagonal matrix where the the diagonal is the elements from vector a.

Step I: the KL term $\mathrm{KL}(\mathcal{Q}||\mathcal{P})$. The expression of the KL term $\mathrm{KL}(\mathcal{Q}||\mathcal{P})$ when $\mathcal{P} = \mathcal{N}(\mathbf{0}, \sigma_0^2 I)$ is as follows,

$$\begin{aligned}
\mathrm{KL}(\mathcal{Q}||\mathcal{P}) &= \frac{1}{2}\left[\log\frac{\sigma_0^2}{\det(\Sigma)} - N + \frac{||\theta||_2^2}{\sigma_0^2} + \frac{\mathrm{Tr}(\Sigma)}{\sigma_0^2}\right] \\
&= \frac{1}{2}\left[\sum_n \log\frac{\sigma_0^2}{\sigma_n^2} - N + \frac{||\theta||_2^2}{\sigma_0^2} + \frac{\sum_n \sigma_n^2}{\sigma_0^2}\right] \text{(when } \Sigma \text{ is a diagonal matrix)}.
\end{aligned} \tag{8}$$

Step II: Second-order Taylor's Expansion for $\mathbb{E}_{\theta\in\mathcal{Q}}\hat{R}(\theta, D^m)$. We expand $\mathbb{E}_{\theta\in\mathcal{Q}}\hat{R}(\theta, D^m)$ at the point θ and use the re-parameterization trick,

$$\begin{aligned}\mathbb{E}_{\theta\in\mathcal{Q}}\hat{R}(\theta, D^m) &= \hat{R}(\theta, D^m) + \underbrace{\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\Sigma)}[\epsilon^\top \nabla\hat{R}]}_{=0 \text{ because } \epsilon \text{ has zero mean}} + \frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\Sigma)}[\epsilon^\top \nabla^2\hat{R}\epsilon] \\ &+ \underbrace{\frac{1}{6}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,\Sigma)}[\sum_{i,j,k}\frac{\partial^3\hat{R}}{\partial\theta_i\theta_j\theta_k}\epsilon_i\epsilon_j\epsilon_k]}_{=0 \text{ because the mean and the skewness of a standard Gaussian are 0}} + O(\sigma^4) \\ &= \hat{R}(\theta, D^m) + \frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[(\sigma\circ\epsilon)^\top \nabla^2\hat{R}(\sigma\circ\epsilon)] + O(\sigma^4) \end{aligned} \tag{9}$$

where $\circ$ is the element-wise product, a.k.a the Hadamard product. Notice the connection between a vector product and Hadamard product:

$$\text{for any vectors } a, b, \quad a\circ b = \text{Diag}(a)b. \tag{10}$$

Using Eq. 10, we simplify Eq. 9 as follows,

$$\begin{aligned}\mathbb{E}_{\theta\in\mathcal{Q}}\hat{R}(\theta, D^m) &= \hat{R}(\theta, D^m) + \frac{1}{2}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[(\text{Diag}(\sigma)\epsilon)^\top \nabla^2\hat{R}(\text{Diag}(\sigma)\epsilon)] + O(\sigma^4) \\ &= \hat{R}(\theta, D^m) + \frac{1}{2}\text{Tr}(\Sigma\circ\nabla^2\hat{R}) + O(\sigma^4). \end{aligned} \tag{11}$$

Step III: Bound Minimization. Using Eq. 8 and 11, we re-write the bound in Theorem 2 as follows

$$\begin{aligned}E_{\theta\in\mathcal{Q}}\hat{R}(\theta, D^m) &\le \mathbb{E}_{\theta\in\mathcal{Q}}\hat{R}(\theta, D^m) + \frac{1}{\beta}\text{KL}(\mathcal{Q}||\mathcal{P}) + C(\tau,\beta,m) \\ &= \hat{R}(\theta, D^m) + \frac{1}{2}\text{Tr}(\Sigma\circ\nabla^2\hat{R}) + \frac{1}{2\beta}\left[\sum_n \log\frac{\sigma_0^2}{\sigma_n^2} - N + \frac{||\theta||_2^2}{\sigma_0^2} + \frac{\sum_n\sigma_n^2}{\sigma_0^2}\right] + O(\sigma^4) + C(\tau,\beta,m). \end{aligned} \tag{12}$$

There are two sets of parameters in Eq. 12 for bound minimization: Σ (i.e. the σ^2) and θ. To minimize w.r.t Σ, strictly speaking it requires minimizing all relevant terms and the higher-order terms in $O(\sigma^4)$, which is clearly infeasible. Instead, we apply the following approximation by neglecting the higher-order $O(\sigma^4)$ and solve

$$\sigma^{*2} = \arg\min_{\sigma^2}\left\{\hat{R}(\theta, D^m) + \frac{1}{2}\text{Tr}(\Sigma\circ\nabla^2\hat{R}) + \frac{1}{2\beta}\left[\sum_n \log\frac{\sigma_0^2}{\sigma_n^2} - N + \frac{||\theta||_2^2}{\sigma_0^2} + \frac{\sum_n\sigma_n^2}{\sigma_0^2}\right]\right\}.$$

In the following, we discuss two cases of σ_n^2.

Case I (Spherical Gaussian): $\forall n, \sigma_n^2 = \sigma_q^2$. In this case, the solution to the problem above is to take the derivative w.r.t. σ_q^2 and set it to zero. That is, for the optimal σ_q^{*2},

$$\text{Tr}(\hat{R}(\theta, D^m)) + \frac{K}{\beta}(\frac{1}{\sigma_0^2} - \frac{1}{\sigma_q^{*2}}) = 0 \implies \frac{\sigma_0^2}{\sigma_q^{*2}} = 1 + \frac{\sigma_0^2\beta}{K}\text{Tr}(\hat{R}(\theta, D^m)). \tag{13}$$

Substituting each σ_n^2 with σ_q^{*2} in Eq. 12 gives the solution to the following problem

$$\min_{\sigma^2}\left[\mathbb{E}_{\theta\in\mathcal{Q}}\hat{R}(\theta, D^m) + \frac{1}{\beta}\text{KL}(\mathcal{Q}||\mathcal{P})\right] = \hat{R}(\theta, D^m) + \frac{K}{2\beta}\log(1 + \frac{\sigma_0^2\beta}{K}\text{Tr}(\hat{R}(\theta, D^m))) + \frac{||\theta||_2^2}{2\beta\sigma_0^2} + O(\sigma_q^{*4}).$$

With the Taylor expansion for $\log(1+x) = x + O(x^2)$,

$$\frac{K}{2\beta}\log(1 + \frac{\sigma_0^2\beta}{K}\text{Tr}(\hat{R}(\theta, D^m)) = \frac{\sigma_0^2}{2\beta}\text{Tr}(\hat{R}(\theta, D^m)) + O(\sigma_0^4) + O(\sigma_q^{*4}).$$

We can actually merge $O(\sigma_0^4) + O(\sigma_q^{*4})$ because

$$\frac{\sigma_0^2}{\sigma_q^{2*}} = 1 + \frac{\sigma_0^2 \beta}{K} \mathrm{Tr}(\hat{R}(\theta, D^m)) \implies \sigma_0^2 \geq \sigma_q^{*2}$$

so $O(\sigma_0^4) + O(\sigma_q^{*4}) = O(\sigma_0^4)$. Lastly, to minimize the bound, we optimize it w.r.t θ, which leads us to land on:

$$\min_\theta \{\hat{R}(\theta, D^m) + \frac{||\theta||_2^2}{2\beta\sigma_0^2} + \frac{\sigma_0^2}{2} \mathrm{Tr}(\nabla_\theta^2 \hat{R}(\theta, D^m))\} + C(\tau, \beta, m) + O(\sigma_0^4).$$

Case II (Diagonal): $\exists n_1, n_2, \sigma_{n_1}^2 \neq \sigma_{n_2}^2$**.** In this case, we take the derivative w.r.t each σ_n^2 and set it to zero. That is, for each optimal σ_n^{2*},

$$\frac{\partial \hat{R}^2}{\partial \theta_n^2} + \frac{1}{\beta}(\frac{1}{\sigma_0^2} - \frac{1}{\sigma_n^{2*}}) = 0 \implies \frac{\sigma_0^2}{\sigma_n^{2*}} = 1 + \sigma_0^2 \beta \frac{\partial^2 \hat{R}}{\partial \theta_n^2}. \tag{14}$$

Substituting σ_n^2 with σ_n^{2*} in Eq. 12 gives the solution to the following problem

$$\min_{\sigma_n^2} \left[\mathbb{E}_{\theta \in \mathcal{Q}} \hat{R}(\theta, D^m) + \frac{1}{\beta} \mathrm{KL}(\mathcal{Q}||\mathcal{P})\right] = \hat{R}(\theta, D^m) + \frac{1}{2\beta} \sum_n \log(1 + \sigma_0^2 \beta \frac{\partial \hat{R}^2}{\partial \theta_n^2}) + \frac{||\theta||_2^2}{2\beta\sigma_0^2} + O(\sigma^{*4}).$$

Similar to Case I, we take Taylor's Expansion of $\log(1+x)$ so

$$\frac{1}{2\beta} \sum_n \log(1 + \sigma_0^2 \beta \frac{\partial \hat{R}^2}{\partial \theta_n^2}) = \frac{\sigma_0^2}{2} \mathrm{Tr}(\hat{R}(\theta, D^m)) + O(\sigma_0^4).$$

Lastly, using $O(\sigma_0^4) + O(\sigma^{*4}) = O(\sigma_0^4)$, we land on the same objective as derived in Case I:

$$\min_\theta \{\hat{R}(\theta, D^m) + \frac{||\theta||_2^2}{2\beta\sigma_0^2} + \frac{\sigma_0^2}{2} \mathrm{Tr}(\nabla_\theta^2 \hat{R}(\theta, D^m))\} + C(\tau, \beta, m) + O(\sigma_0^4).$$

Finding a Point Estimator using Theorem 3 Theorem 3 derives an upper-bound for the expected generalization error over the posterior distribution $\mathcal{Q}$, i.e. $\mathbb{E}_{\theta \sim \mathcal{D}} R(\theta, \mathcal{D})$, instead of an upper-bound for the generalization error at a single point θ, i.e. $R(\theta, \mathcal{D})$ for practical use. Nevertheless, the resulting bound from Theorem 3 is a good proxy for analyzing the generalization gap for $R(\theta, \mathcal{D})$ at $\theta^* = \arg\min_\theta \{\text{Eq. 5}\}$. To see this, recall that the optimal variance σ_n^{2*} of $\mathcal{Q}$ appears in Eq. 14 (or Eq. 13 for Case I), which is equivalent to the following

$$\sigma_n^{2*} = \left(\frac{1}{\sigma_0^2} + \beta \frac{\partial^2 \hat{R}}{\partial \theta_n^2}\right)^{-1}.$$

Since β is proportional to the number of data examples m, the variance σ_n^{2*} becomes very small for any reasonably sized datasets (e.g. $m = 60K$ in CIFAR, $m = 1M$ in ImageNet). The resulting posterior $\mathcal{Q}$, in practice, is a "narrow" Gaussian. So one can simply take its optimal mean θ^* as a point estimator of the true posterior.

A.2. Proof of Theorem 4

Theorem 4 Suppose that g_θ is a feed-forward network with ReLU activation. For the i-th layer, let $W^{(i)}$ be its weight and $\mathcal{I}_x^{(i)} \in \mathbb{R}^{D^{(i)}}$ be its input evaluated at x. Thus, $\mathrm{TrH}_x^{\mathrm{CE}}(W^{(i-1)})$, i.e. TrH evaluated at x using a CE loss w.r.t $W^{(i-1)}$, is as follows

$$\mathrm{TrH}_x^{\mathrm{CE}}(W^{(i-1)}) = ||\{\mathcal{I}_x^{(i-1)}\}||_2^2 \sum_{k,d \in P^{(i)}} \{\mathcal{H}^{(i)}\}_{k,d}$$

$$\text{where } \{\mathcal{H}^{(i)}\}_{k,d_i} \stackrel{\text{def}}{=} \left[\frac{\partial\{g_\theta(x)\}_k}{\partial\{\mathcal{I}_x^{(i)}\}_{d_i}}\right]^2 \cdot h_k(x,\theta)$$
$$P^{(i)} \stackrel{\text{def}}{=} \{d_i|\{\mathcal{I}_x^{(i)}\}_{d_i} > 0\}$$

and the following inequality holds for any $\mathcal{H}^{(i)}, \mathcal{H}^{(i+1)}$:

$$\max_{k,d_i}\{\mathcal{H}^{(i)}\}_{k,d_i} \leq \max_{k,d_{i+1}} \mathcal{H}^{(i+1)} \cdot ||W^{(i)}||_1^2.$$

Proof. We use d_{i-1}, d_i and d_{i+1} to index the element of $\mathcal{I}_x^{(i-1)}$, $\mathcal{I}_x^{(i)}$ and $\mathcal{I}_x^{(i+1)}$. By the definition of trace, we need to compute the second-order derivative of the CE loss w.r.t to each entry in $W^{(i)}$. That is, we write

$$\text{TrH}_x^{\text{CE}}(W^{(i-1)}) = \sum_{d_{i-1},d_i} \frac{\partial^2}{\partial W_{d_{i-1},d_i}^{(i-1)2}}(CE((x,y),g_\theta)). \tag{15}$$

First, we derive the first-order derivative. The following steps are based on the chain rule:

$$\begin{aligned}\frac{\partial}{\partial W_{d_{i-1},d_i}^{(i-1)}}(CE((x,y),g_\theta)) &= \sum_k \frac{\partial}{\partial\{g_\theta(x)\}_k}(CE((x,y),g_\theta)) \cdot \frac{\{g_\theta(x)\}_k}{\partial W_{d_{i-1},d_i}^{(i-1)}} \\ &= \sum_k (\{s(g_\theta(x)\}_k - y_k) \cdot \frac{\{g_\theta(x)\}_k}{\partial W_{d_{i-1},d_i}^{(i-1)}}.\end{aligned}$$

The second-order derivative is therefore equal to

$$\begin{aligned}\frac{\partial^2}{\partial W_{d_{i-1},d_i}^{(i-1)2}}(CE((x,y),g_\theta)) &= \sum_k \Big(\frac{\partial}{\partial W_{d_{i-1},d_i}^{(i-1)}}\left[\{s(g_\theta(x)\}_k - y_k)\right] \cdot \frac{\{g_\theta(x)\}_k}{\partial W_{d_{i-1},d_i}^{(i-1)}} + h_k(x,\theta) \cdot \frac{\partial}{\partial W_{d_{i-1},d_i}^{(i-1)}}\left[\frac{\{g_\theta(x)\}_k}{\partial W_{d_{i-1},d_i}^{(i-1)}}\right]\Big) \\ &= \sum_k \Big(h_k(x,\theta) \cdot \left[\frac{\{g_\theta(x)\}_k}{\partial W_{d_{i-1},d_i}^{(i-1)}}\right]^2 + h_k(x,\theta) \cdot \frac{\partial}{\partial W_{d_{i-1},d_i}^{(i-1)}}\left[\frac{\{g_\theta(x)\}_k}{\partial W_{d_{i-1},d_i}^{(i-1)}}\right]\Big).\end{aligned}$$

Notice that the first-order derivative of ReLU is either 1 or 0, so we have plenty of identity functions in $\frac{\{g_\theta(x)\}_k}{\partial W_{d_{i-1},d_i}^{(i-1)}}$. For example, for any layer $j > i-1$, we can decompose this term as

$$\begin{aligned}\frac{\{g_\theta(x)\}_k}{\partial W_{d_{i-1},d_i}^{(i-1)}} &= \sum_{d_{j+1}} \frac{\{g_\theta(x)\}_k}{\partial \mathcal{I}_x^{(j+1)}{}_{d_{j+1}}} \cdot \sum_{d_j} \frac{\partial \mathcal{I}_x^{(j+1)}{}_{d_{j+1}}}{\partial \mathcal{I}_x^{(j)}{}_{d_j}} \cdot \frac{\partial \mathcal{I}_x^{(j)}{}_{d_j}}{\partial W_{d_{i-1},d_i}^{(i-1)}} \\ &= \sum_{d_{j+1}} \frac{\{g_\theta(x)\}_k}{\partial \mathcal{I}_x^{(j+1)}{}_{d_{j+1}}} \cdot \sum_{d_j} \mathbb{I}[\{\mathcal{I}_x^{(j+1)}\}_{d_{j+1}} > 0] \cdot W_{d_j,d_{j+1}}^{(j)} \cdot \frac{\partial \mathcal{I}_x^{(j)}{}_{d_j}}{\partial W_{d_{i-1},d_i}^{(i-1)}}.\end{aligned}$$

Taking derivatives of $\mathbb{I}[\{\mathcal{I}_x^{(j+1)}\}_{d_{j+1}} > 0]$ w.r.t $W_{d_{i-1},d_i}^{(i-1)}$ results in nothing but 0, so we conclude that

$$\frac{\partial}{\partial W_{d_{i-1},d_i}^{(i-1)}}\left[\frac{\{g_\theta(x)\}_k}{\partial W_{d_{i-1},d_i}^{(i-1)}}\right] = 0. \tag{16}$$

Eq. 16 simplifies the expression of the second-order derivative w.r.t $W^{(i-1)}_{d_{i-1},d_i}$ so we have the following clean expression:

$$\begin{aligned}
\frac{\partial^2}{\partial W^{(i-1)2}_{d_{i-1},d_i}}(CE((x,y),g_\theta)) &= \sum_k \left[\frac{\partial\{g_\theta(x)\}_k}{\partial W^{(i-1)}_{d_{i-1},d_i}}\right]^2 \cdot h_k(x,\theta) \\
&= \sum_k \left[\frac{\partial\{g_\theta(x)\}_k}{\partial\{\mathcal{I}^{(i)}_x\}_{d_i}} \cdot \frac{\partial\{\mathcal{I}^{(i)}_x\}_{d_i}}{\partial W^{(i-1)}_{d_{i-1},d_i}}\right]^2 \cdot h_k(x,\theta) \\
&= \sum_k \left[\frac{\partial\{g_\theta(x)\}_k}{\partial\{\mathcal{I}^{(i)}_x\}_{d_i}} \cdot \mathbb{I}[\{\mathcal{I}^{(i)}_x\}_{d_i} > 0] \cdot \{\mathcal{I}^{(i-1)}_x\}_{d_{i-1}}\right]^2 \cdot h_k(x,\theta) \\
&= \mathbb{I}[\{\mathcal{I}^{(i)}_x\}_{d_i} > 0] \cdot \{\mathcal{I}^{(i-1)}_x\}^2_{d_{i-1}} \cdot \sum_k \left[\frac{\partial\{g_\theta(x)\}_k}{\partial\{\mathcal{I}^{(i)}_x\}_{d_i}}\right]^2 \cdot h_k(x,\theta). \qquad (17)
\end{aligned}$$

By defining $\{\mathcal{H}^{(i)}\}_{k,d_i} \stackrel{\text{def}}{=} \left[\frac{\partial\{g_\theta(x)\}_k}{\partial\{\mathcal{I}^{(i)}_x\}_{d_i}}\right]^2 \cdot h_k(x,\theta)$, we further simplify Eq. 17 as follows

$$\frac{\partial^2}{\partial W^{(i-1)2}_{d_{i-1},d_i}}(CE((x,y),g_\theta)) = \mathbb{I}[\{\mathcal{I}^{(i)}_x\}_{d_i} > 0] \cdot \{\mathcal{I}^{(i-1)}_x\}^2_{d_{i-1}} \cdot \sum_k \{\mathcal{H}^{(i)}\}_{k,d_i}. \qquad (18)$$

We plug Eq. 18 back to Eq. 15 and arrive

$$\begin{aligned}
\mathrm{TrH}^{\mathrm{CE}}_x(W^{(i-1)}) &= \sum_{d_{i-1},d_i} \mathbb{I}[\{\mathcal{I}^{(i)}_x\}_{d_i} > 0] \cdot \{\mathcal{I}^{(i)}_x\}^2_{d_{i-1}} \cdot \sum_k \{\mathcal{H}^{(i)}\}_{k,d_i} \\
&= \Big(\sum_{d_{i-1}} \{\mathcal{I}^{(i-1)}_x\}^2_{d_{i-1}}\Big) \cdot \sum_{k,d_i} \mathbb{I}[\{\mathcal{I}^{(i)}_x\}_{d_i} > 0] \cdot \{\mathcal{H}^{(i)}\}_{k,d_i} \\
&= ||\{\mathcal{I}^{(i-1)}_x\}||^2_2 \cdot \sum_{k,d_i} \mathbb{I}[\{\mathcal{I}^{(i)}_x\}_{d_i} > 0] \cdot \{\mathcal{H}^{(i)}\}_{k,d_i}.
\end{aligned}$$

By defining $P^{(i)}$ as a set of indices of positive neurons in $\mathcal{I}^{(i)}_x$, i.e, $\forall d_2 \in P^{(i)}, \{\mathcal{I}^{(i)}_x\}_{d_2} > 0$, we simplify the equation above as follows

$$\mathrm{TrH}^{\mathrm{CE}}_x(W^{(i-1)}) = ||\{\mathcal{I}^{(i)}_x\}||^2_2 \cdot \sum_{k,d_i \in P^{(i)}} \{\mathcal{H}^{(i)}\}_{k,d_i}.$$

Furthermore, by the definition of $\mathcal{H}^{(i)}$, we notice that

$$\begin{aligned}
\{\mathcal{H}^{(i)}\}_{k,d_i} &= \left[\frac{\partial\{g_\theta(x)\}_k}{\partial\{\mathcal{I}^{(i)}_x\}_{d_i}}\right]^2 \cdot h_k(x,\theta) \\
&= \left\{\sum_{d_{i+1}} \left[\frac{\partial\{g_\theta(x)\}_k}{\partial\{\mathcal{I}^{(i+1)}_x\}_{d_{i+1}}} \cdot \frac{\partial\{\mathcal{I}^{(i+1)}_x\}_{d_{i+1}}}{\partial\{\mathcal{I}^{(i)}_x\}_{d_i}}\right]\right\}^2 \cdot h_k(x,\theta) \qquad (19) \\
&= \left\{\sum_{d_{i+1}} \left[\Big(\frac{\{\mathcal{H}^{(i+1)}\}_{k,d_{i+1}}}{h_k(x,\theta)}\Big)^{\frac{1}{2}} \cdot \frac{\partial\{\mathcal{I}^{(i+1)}_x\}_{d_{i+1}}}{\partial\{\mathcal{I}^{(i)}_x\}_{d_i}}\right]\right\}^2 \cdot h_k(x,\theta). \qquad (20)
\end{aligned}$$

Notice that the transition from Eq. 19 to Eq. 20 is because $\forall k, h_k(x,\theta) > 0$. Thus, we find

$$\{\mathcal{H}^{(i)}\}_{k,d_i} = \left\{\sum_{d_{i+1}} \left[\{\mathcal{H}^{(i+1)}\}^{\frac{1}{2}}_{k,d_{i+1}} \cdot \frac{\partial\{\mathcal{I}^{(i+1)}_x\}_{d_{i+1}}}{\partial\{\mathcal{I}^{(i)}_x\}_{d_i}}\right]\right\}^2. \qquad (21)$$

For each layer i, we denote $\mathcal{H}^{(i)}_{max} = \max_{k,d_i}\{\mathcal{H}^{(i)}\}_{k,d_i}$. Since $\forall i, d_i, \{\mathcal{H}^{(i)}\}_{k,d_i} > 0$, we can take the square root for $\mathcal{H}^{(i)}_{max}$ such that

$$\forall k, d_i, (\mathcal{H}^{(i)}_{max})^{\frac{1}{2}} \geq (\{\mathcal{H}^{(i)}\}_{k,d_i})^{\frac{1}{2}}.$$

Thus, we can bound $\{\mathcal{H}^{(i)}\}_{k,d_i}$ as follows:

$$\begin{aligned}\{\mathcal{H}^{(i)}\}_{k,d_i} &\leq \left\{\sum_{d_{i+1}}\left[(\mathcal{H}^{(i+1)}_{max})^{\frac{1}{2}} \cdot \frac{\partial\{\mathcal{I}^{(i+1)}_x\}_{d_{i+1}}}{\partial\{\mathcal{I}^{(i)}_x\}_{d_i}}\right]\right\}^2 \\ &= \mathcal{H}^{(i)}_{max}\left[\sum_{d_{i+1}}\frac{\partial\{\mathcal{I}^{(i+1)}_x\}_{d_{i+1}}}{\partial\{\mathcal{I}^{(i)}_x\}_{d_i}}\right]^2.\end{aligned}$$

The squared term can be further bounded by using the chain rule, namely,

$$\begin{aligned}\{\mathcal{H}^{(i)}\}_{k,d_i} &\leq \mathcal{H}^{(i+1)}_{max}\left[\sum_{d_{i+1}}\mathbb{I}[\{\mathcal{I}^{(i+1)}_x\}_{d_{i+1}} > 0]\cdot W^{(i)}_{d_i,d_{i+1}}\right]^2 \\ &\leq \mathcal{H}^{(i+1)}_{max}\left[\sum_{d_{i+1}}|W^{(i)}_{d_i,d_{i+1}}|\right]^2.\end{aligned}$$

Recall that our goal is to bound $\mathcal{H}^{(i)}_{max}$. By definition, we take the max on both sides over d_i and k (although the class dimension is already gone). The direction of the inequality still holds because quantities on both sides are all non-negative.

$$\mathcal{H}^{(i)}_{max} \leq \mathcal{H}^{(i+1)}_{max}\max_{d_i}\left[\sum_{d_{i+1}}|W^{(i)}_{d_i,d_{i+1}}|\right]^2.$$

The order of max and square can be exchanged because $|W^{(i)}_{d_i,d_{i+1}}|$ is non-negative. Thus,

$$\mathcal{H}^{(i)}_{max} \leq \mathcal{H}^{(i+1)}_{max}\left[\max_{d_i}\sum_{d_{i+1}}|W^{(i)}_{d_i,d_{i+1}}|\right]^2.$$

The last term inside the square is the definition of the ℓ_1 operator norm so we directly write

$$\mathcal{H}^{(i)}_{max} \leq \mathcal{H}^{(i+1)}_{max} \cdot ||W^{(i)}||^2_1.$$

A.3. Derivations of Propositions

Proposition 1 Given a training dataset D^m and the adversarial input example x' for each example x, the top-layer TrH of the AT loss (Definition 1) is equal to

$$\begin{aligned}\mathrm{Tr}(\nabla^2_{\theta_t}\hat{R}_{\mathrm{AT}}(\theta, D^m)) &= \frac{1}{m}\sum_{(x,y)\in D^m}\mathrm{TrH}_{\mathrm{AT}}(x';\theta), \\ \text{where}\quad \mathrm{TrH}_{\mathrm{AT}}(x';\theta) &= ||f_{\theta_b}(x')||^2_2 \cdot \mathbf{1}^\top h(x',\theta).\end{aligned}$$

Proof. We firstly write the expression of $\hat{R}_{\mathrm{AT}}(\theta, D^m)$ based on Definition 1,

$$\hat{R}_{\mathrm{AT}}(\theta, D^m) = \frac{1}{m}\sum_{(x,y)\in D^m}\max_{||\epsilon||_p\leq\delta}\mathrm{CE}((x+\epsilon, y), F_\theta).$$

Let x' be an adversarial example, namely

$$x' = \arg\max_{||\epsilon||_p \leq \delta} [\text{CE}((x+\epsilon, y), F_\theta)] + x.$$

AT loss can be simplified as

$$\hat{R}_{\text{AT}}(\theta, D^m) = \frac{1}{m} \sum_{(x,y)\in D^m} \text{CE}((x', y), F_\theta) = \frac{1}{m} \sum_{(x,y)\in D^m} \sum_k -y_k \log s(g'_k),$$

where $g'_k = \{\theta_t^\top f_{\theta_b}(x')\}_k$ is the logit score of class k at the adversarial example x' and $s(\cdot)$ is the softmax function. Let's consider the loss at each adversarial point

$$R = \sum_k -y_k \log s(g'_k).$$

We want to compute the derivatives of R with respect to the weight at the top layer $\{\theta_t\}_{jk}$. For the ease of the notation, let's define $w = \theta_t$ so $w_{jk} = \{\theta_t\}_{jk}$ and $\nabla_{w_{jk}} R$ is equal to

$$\nabla_{w_{jk}} R = \sum_k -y_k \nabla_{w_{jk}} [\log s(g'_k)] = \{f_{\theta_b}(x')\}_j (s(g'_k) - y_k). \tag{22}$$

The transition to Eq. 22 uses the standard form of the gradient of Cross Entropy loss with the softmax activation. To compute the trace of a Hessian, we can skip the cross terms in Hessian (e.g. $\nabla_{w_{j_1 k}, w_{j_2 k}} R$) because they are not on the diagonal of the matrix. Thus, we are only interested in $\nabla^2_{w_{jk}} R$, which are

$$\begin{aligned}
\nabla^2_{w_{jk}} R &= \nabla_{w_{jk}} \left[\nabla_{w_{jk}} R\right] = \nabla_{w_{jk}} \left[x'_j (s(g'_k) - y_k)\right] \\
&= \{f_{\theta_b}(x')\}_j \cdot \{f_{\theta_b}(x')\}_j \cdot s(g'_k)(1 - s(g'_k)) \\
&= \{f_{\theta_b}(x')\}_j^2 (s(g'_k) - s^2(g'_k)).
\end{aligned}$$

Eventually, to compute the trace of Hessian matrix we sum all diagonal terms so that

$$\text{Tr}(\nabla^2_{w_{jk}} R) = \sum_{j,k} \nabla^2_{w_{jk}} R = \sum_{j,k} \{f_{\theta_b}(x')\}_j^2 (s(g'_k) - s^2(g'_k)) = ||f_{\theta_b}(x')||_2^2 \cdot \mathbf{1}^\top h(x', \theta)$$

and we denote $\text{Tr}(\nabla^2_{w_{jk}} R)$ as $\text{TrH}_{\text{AT}}(x'; \theta)$ to complete the proof.

Proposition 2 Under the same assumption as in Proposition 1, the top-layer TrH of the TRADES loss (Def' 2) is equal to

$$\begin{aligned}
&\text{Tr}(\nabla^2_{\theta_t} \hat{R}_{\text{T}}(\theta, D^m)) = \frac{1}{m} \sum_{(x,y)\in D^m} \text{TrH}_{\text{T}}(x, x'; \lambda_t, \theta), \\
&\text{where, } \text{TrH}_{\text{T}}(x, x'; \lambda_t, \theta) = ||f_{\theta_b}(x)||_2^2 \cdot \mathbf{1}^\top h(x, \theta) + \lambda_t ||f_{\theta_b}(x')||_2^2 \cdot \mathbf{1}^\top h(x', \theta).
\end{aligned}$$

Before we start our proof of this proposition, we introduce the following useful lemmas.

Lemma 1 *Let $s(g)$ be the softmax output of the logit output g computed at input x, then*

$$\frac{\partial s(g)}{\partial g} = \Phi = diag[s(g)] - s(g) \cdot \{s(g)\}^\top \tag{23}$$

where $diag(v)$ returns an identity matrix with its diagonal replaced by a vector v.

Proof.

$$\frac{\partial s(g_i)}{\partial g_k} = s(g_i)(\mathbb{I}[i=k] - s(g_k)), \Phi_{ik} = s(g_i) \cdot \mathbb{I}[i=k] - s(g_i) s(g_k) \implies \frac{\partial s(g)}{\partial g} = \Phi.$$

Lemma 2 *Let* $\log s(g(x))$ *be the log softmax output of the logit output* g *computed at input* x*, then*

$$\frac{\partial \log s(g(x))}{\partial g(x)} = \Psi = I - \mathbf{1} \cdot \{s(g(x))\}^\top$$

where I *is the identity matrix.*

Proof.

$$\frac{\partial \log s(g_i)}{\partial g_k} = \mathbb{I}\left[i = k\right] - s(g_k), \Psi_{ik} = \mathbb{I}\left[i = k\right] - s(g_k) \implies \frac{\partial s(g)}{\partial g} = \Psi.$$

Now we are ready to present our proof of the Proposition 2.

Proof. We write the expression of $\hat{R}_{\mathrm{T}}(\theta, D^m)$ based on as Definition 2.

$$\hat{R}_{\mathrm{T}}(\theta, D^m) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{(x,y)\in D^m} \Big[\mathrm{CE}((x,y), g_\theta) + \lambda_t \cdot \max_{||\epsilon||_p \leq \delta} \mathrm{KLloss}((x, x+\epsilon), g_\theta)\Big],$$

where $\mathrm{KLloss}((x, x+\epsilon), g_\theta) = \mathrm{KL}(s(g_\theta(x))||s(g_\theta(x+\epsilon))$. Thus, we can compute the trace of Hessian on the top layer for the CE loss and the KL loss, respectively. Namely, we derive $\nabla^2_{\theta_t}[\mathrm{CE}((x,y), F_\theta)]$ and $\nabla^2[\max_{||\epsilon||_p\leq\delta} \mathrm{KLloss}((x, x+\epsilon), F_\theta)]$.

First, we see that the expression of $\nabla^2_{\theta_t}[\mathrm{CE}((x,y), F_\theta)]$ is similar to the result in Proposition 1 by replacing the adversarial input with the clean input. With this similarity, we directly write

$$\mathrm{Tr}\{\nabla^2_{\theta_t}[\mathrm{CE}((x,y), F_\theta)]\} = ||f_{\theta_b}(x)||_2^2 \cdot \mathbf{1}^\top h(x, \theta). \tag{24}$$

Second, by denoting

$$x' = \arg\max_{||\epsilon||_p\leq\delta} [\mathrm{KLloss}((x, x+\epsilon), F_\theta)] + x,$$

the rest of the proof now focuses on deriving $\nabla^2_{\theta_t}[\mathrm{KLloss}((x, x'), F_\theta)]$ where

$$\mathrm{KLloss}((x, x'), F_\theta) = -\sum_i s(g_i)\log(s(g'_i)) + \left[\sum_i s(g_i)\log(s(g_i))\right], \quad g_i = \{\theta_t^\top f_{\theta_b}(x)\}_i, g'_i = \{\theta_t^\top f_{\theta_b}(x')\}_i.$$

For the ease of the notation, let $w \stackrel{\text{def}}{=} \theta_t$ so $w_{jk} \stackrel{\text{def}}{=} \{\theta_t\}_{jk}$ and let $K \stackrel{\text{def}}{=} \mathrm{KLloss}((x, x'), F_\theta)$. To find $\nabla^2_{\theta_t} K$, we first write the first-order derivative of K with respect to w,

$$\frac{\partial K}{\partial w_{jk}} = -\sum_i s(g_i)\frac{\partial}{\partial g'_k}\left[\log(s(g'_i))\right]\frac{\partial g'_k}{\partial \partial w_{jk}} + \underbrace{\sum_i \frac{\partial K}{\partial s(g_i)}\frac{\partial s(g_i)}{\partial g_k}\frac{\partial g_k}{\partial \partial w_{jk}}}_{\text{gradient through } g_k}. \tag{25}$$

Next, we discuss two cases depending on whether or not we stop gradient on g (the logit output of the clean input) in Eq. 25. In Case I where the gradient on g is stopped, the second term of Eq. 25 will vanish. This leads to a simpler but practically more stable objective function.

Case I: Stop Gradient on g**.** In this case,

$$\begin{aligned}\frac{\partial K}{\partial w_{jk}} &= -\sum_i s(g_i)\frac{\partial}{\partial g'_k}\left[\log(s(g'_i))\right]\frac{\partial g'_k}{\partial \partial w_{jk}} \\ &= \{f_{\theta_b}(x')\}_j(s(g'_k) - s(g_k)).\end{aligned} \tag{26}$$

By comparing Eq. 26 with Eq. 22 and treating $s(g_k)$ as constants, we can quickly write out the second-order derivative as

$$\frac{\partial^2 K}{\partial w_{jk}^2} = \{f_{\theta_b}(x')\}_j^2(s(g'_k) - s^2(g'_k)).$$

Recalling $h(x', \theta) = s(g') - s^2(g')$, therefore,

$$\mathrm{Tr}\{\nabla^2_{\theta_t}[\mathrm{KLloss}((x, x'), F_\theta)]\} = \mathrm{Tr}\{\nabla^2_{\theta_t} K\} = \sum_{jk} \frac{\partial^2 K}{\partial w_{jk}^2} = ||f_{\theta_b}(x')||_2^2 \cdot \mathbf{1}^\top h(x', \theta). \tag{27}$$

Finally, we combine Eq. 24 and 27 to arrive at

$$\mathrm{Tr}\Big\{\nabla^2_{\theta_t}\Big[\mathrm{CE}((x, y), g_\theta) + \lambda_t \cdot \max_{||\epsilon||_p \le \delta} \mathrm{KLloss}((x, x+\epsilon), g_\theta)\Big]\Big\} = ||f_{\theta_b}(x)||_2^2 \cdot \mathbf{1}^\top h(x, \theta) + \lambda_t ||f_{\theta_b}(x')||_2^2 \cdot \mathbf{1}^\top h(x', \theta),$$
$$\text{where } x' = \arg\max_{||\epsilon||_p \le \delta} [\mathrm{KLloss}((x, x+\epsilon), F_\theta)] + x$$

By denoting $\mathrm{TrH}_{\mathrm{t}}(x, x'; \lambda_t, \theta) \stackrel{\text{def}}{=} \mathrm{Tr}\Big\{\nabla^2_{\theta_t}\Big[\mathrm{CE}((x, y), g_\theta) + \lambda_t \cdot \max_{||\epsilon||_p \le \delta} \mathrm{KLloss}((x, x+\epsilon), g_\theta)\Big]\Big\}$, we complete the proof for this case and this is the statement shown in Proposition 2.

Case II: With Gradient on g. We restart our derivation from Eq. 25 and expand the first term as follows

$$\frac{\partial K}{\partial w_{jk}} = \{f_{\theta_b}(x')\}_j (s(g'_k) - s(g_k)) + \sum_i \frac{\partial K}{\partial s(g_i)} \frac{\partial s(g_i)}{\partial g_k} \frac{\partial g_k}{\partial \partial w_{jk}}.$$

Here

$$\sum_i \frac{\partial K}{\partial s(g_i)} \frac{\partial s(g_i)}{\partial g_k} \frac{\partial g_k}{\partial \partial w_{jk}} = -\{f_{\theta_b}(x)\}_j \sum_i \frac{\partial s(g_i)}{\partial g_k} \log s(g'_i) + \{f_{\theta_b}(x)\}_j \sum_i \left[\frac{\partial s(g_i)}{\partial g_k} \log s(g_i) + s(g_i) \frac{\partial \log s(g_i)}{\partial g_k}\right]$$

Using Lemma 1 and 2, we write

$$\frac{\partial s(g_i)}{\partial g_k} = \Phi_{ik}, \qquad \frac{\partial \log s(g_i)}{\partial g_k} = \Psi_{ik}, \quad \Phi_{ik} = s(g_i)\Psi_{ik}$$

and

$$\sum_i \frac{\partial K}{\partial s(g_i)} \frac{\partial s(g_i)}{\partial g_k} \frac{\partial g_k}{\partial \partial w_{jk}} = -\{f_{\theta_b}(x)\}_j \sum_i \Phi_{ik} \log s(g'_i) + \{f_{\theta_b}(x)\}_j \sum_i \left[\Phi_{ik} \log s(g_i) + \Phi_{ik}\right].$$

Therefore, the first-order derivative of K with respect to w_{jk} is

$$\begin{aligned}\frac{\partial K}{\partial w_{jk}} &= \{f_{\theta_b}(x')\}_j (s(g'_k) - s(g_k)) - \{f_{\theta_b}(x)\}_j \sum_i \Phi_{ik} \log s(g'_i) + \{f_{\theta_b}(x)\}_j \sum_i [\Phi_{ik} \log s(g_i) + \Phi_{ik}] \\ &= \underbrace{\{f_{\theta_b}(x')\}_j s(g'_k)}_{K'} \underbrace{- \{f_{\theta_b}(x')\}_j s(g_k)}_{K_1} \underbrace{- \{f_{\theta_b}(x)\}_j \sum_i \Phi_{ik} \log s(g'_i)}_{K_2} + \underbrace{\{f_{\theta_b}(x)\}_j \sum_i [\Phi_{ik} \log s(g_i) + \Phi_{ik}]}_{K_3}.\end{aligned}$$

To calculate the seconder-order derivative of K w.r.t w_{jk}, we find the derivative of K', K_1, K_2, K_3 w.r.t w_{jk}, respectively.

(1) Derivative of K'. K' is simply the result we have already obtained in Case I (see Eq. 26); therefore,

$$\frac{\partial K'}{\partial w_{jk}} = \{f_{\theta_b}(x')\}_j^2 (s(g'_k) - s^2(g'_k)).$$

Thus,

$$\mathrm{Tr}\left(\frac{\partial K'}{\partial w_{jk}}\right) = ||f_{\theta_b}(x')||_2^2 \cdot \mathbf{1}^\top h(x', \theta).$$

(2) Derivative of K_1.

$$\begin{aligned}\frac{\partial K_1}{\partial w_{jk}} &= -\{f_{\theta_b}(x')\}_j\{f_{\theta_b}(x)\}_j\frac{\partial s(g_k)}{\partial g_k}\\ &= -\{f_{\theta_b}(x')\}_j\{f_{\theta_b}(x)\}_j\Phi_{kk}\\ &= -\{f_{\theta_b}(x')\}_j\{f_{\theta_b}(x)\}_j(s(g_k)-s^2(g_k)).\end{aligned}$$

Thus,

$$\begin{aligned}\mathrm{Tr}(\frac{\partial K_1}{\partial w_{jk}}) &= -\sum_{jk}\{f_{\theta_b}(x')\}_j\{f_{\theta_b}(x)\}_j(s(g_k)-s^2(g_k))\\ &= -f_{\theta_b}(x')^\top f_{\theta_b}(x)\cdot\mathbf{1}^\top h(x,\theta)\end{aligned}$$

(3) Derivative of K_2.

$$\begin{aligned}\frac{\partial K_2}{\partial w_{jk}} &= -\{f_{\theta_b}(x)\}_j\sum_i\left[\frac{\partial\Phi_{ik}}{\partial g_k}\frac{\partial g_k}{\partial w_{jk}}\log s(g_i')+\Phi_{ik}\frac{\partial\log s(g_i')}{\partial g_k'}\frac{\partial g_k'}{\partial w_{jk}}\right]\\ &= -\{f_{\theta_b}(x)\}_j\sum_i\left[\frac{\partial\Phi_{ik}}{\partial g_k}\{f_{\theta_b}(x)\}_j\log s(g_i')+\Phi_{ik}\Psi_{ik}'\{f_{\theta_b}(x')\}_j\right]\end{aligned}$$

Notice that

$$\begin{aligned}i=k \implies \frac{\partial\Phi_{ik}}{\partial g_k} &= \frac{\partial\Phi_{kk}}{\partial g_k} = \frac{\partial}{\partial g_k}(s(g_k)-s^2(g_k)) = \Phi_{kk}-2s(g_k)\Phi_{kk};\\ \text{and } i\neq k \implies \frac{\partial\Phi_{ik}}{\partial g_k} &= \frac{\partial}{\partial g_k}(-s(g_i)s(g_k)) = -2s(g_i)\Phi_{kk}.\end{aligned}$$

Thus,

$$\frac{\partial\Phi_{ik}}{\partial g_k} = \Phi_{kk}(\mathbb{I}[k=i]-s(g_i)) = \Phi_{kk}\{\Psi^\top\}_{ik} = \Phi_{kk}\Psi_{ki}.$$

As a result,

$$\begin{aligned}\frac{\partial K_2}{\partial w_{jk}} &= -\{f_{\theta_b}(x)\}_j\sum_i[\Phi_{kk}\Psi_{ki}\{f_{\theta_b}(x)\}_j\log s(g_i')+\Phi_{ik}\Psi_{ik}'\{f_{\theta_b}(x')\}_j]\\ &= -\{f_{\theta_b}(x)\}_j^2\Phi_{kk}\sum_i\Psi_{ki}\log s(g_i')-\{f_{\theta_b}(x)\}_j\{f_{\theta_b}(x')\}_j\sum_i\Phi_{ik}\Psi_{ik}'\\ &= (-\{f_{\theta_b}(x)\}_j^2\Phi_{kk})(\Psi_k\cdot\log s(g'))-\{f_{\theta_b}(x)\}_j\{f_{\theta_b}(x')\}_j(\{\Phi^\top\}_k\cdot\{\Psi'^\top\}_k).\end{aligned}$$

Thus,

$$\begin{aligned}\mathrm{Tr}(\frac{\partial K_2}{\partial w_{jk}}) &= \sum_{jk}\left[(-\{f_{\theta_b}(x)\}_j^2\Phi_{kk})(\Psi_k\cdot\log s(g'))-\{f_{\theta_b}(x)\}_j\{f_{\theta_b}(x')\}_j(\{\Phi^\top\}_k\cdot\{\Psi'^\top\}_k)\right]\\ &= -||f_{\theta_b}(x)||_2^2\cdot\psi'^\top h(x,\theta)-f_{\theta_b}^\top(x')f_{\theta_b}(x)\cdot\mathbf{1}^\top\omega',\end{aligned}$$

where ψ',ω' are vectors such that $\psi_k' = \Psi_k\cdot\log s(g'), \omega_k' = \{\Phi^\top\}_k\cdot\{\Psi'^\top\}_k$.

(4) Derivative of K_3.

$$\begin{aligned}\frac{\partial K_3}{\partial w_{jk}} &= \{f_{\theta_b}(x)\}_j \sum_i \left[\frac{\partial \Phi_{ik}}{\partial g_k}\frac{\partial g_k}{\partial w_{jk}}\log s(g_i) + \Phi_{ik}\frac{\partial \log s(g_i)}{\partial g_k}\frac{\partial g_k}{\partial w_{jk}} + \frac{\partial \Phi_{ik}}{\partial g_k}\frac{\partial g_k}{\partial w_{jk}}\right] \\ &= \{f_{\theta_b}(x)\}_j^2 \sum_i \left[\Phi_{kk}\Psi_{ki}\log s(g_i) + \Phi_{ik}\Psi_{ik} + \Phi_{kk}\Psi_{ki}\right] \\ &= \{f_{\theta_b}(x)\}_j^2 \left[\Phi_{kk}(\Psi_k \cdot \log s(g)) + (\{\Phi^\top\}_k \cdot \{\Psi^\top\}_k) + \Phi_{kk}(\underbrace{\Psi_k \cdot \mathbf{1}}_{\text{row sum is 0}})\right] \\ &= \{f_{\theta_b}(x)\}_j^2 \left[\Phi_{kk}(\Psi_k \cdot \log s(g)) + (\{\Phi^\top\}_k \cdot \{\Psi^\top\}_k)\right].\end{aligned}$$

Thus,

$$\begin{aligned}\mathrm{Tr}(\frac{\partial K_3}{\partial w_{jk}}) &= \sum_{jk}\left[\{f_{\theta_b}(x)\}_j^2\left[\Phi_{kk}(\Psi_k \cdot \log s(g)) + (\{\Phi^\top\}_k \cdot \{\Psi^\top\}_k)\right]\right] \\ &= ||f_{\theta_b}(x)||_2^2 \cdot (\psi^\top h(x,\theta) + \mathbf{1}^\top \omega).\end{aligned}$$

Finally, we have

$$\begin{aligned}\mathrm{Tr}(\frac{\partial^2 K}{\partial w_{jk}^2}) &= \mathrm{Tr}\left[\frac{\partial K'}{\partial w_{jk}} + \frac{\partial K_1}{\partial w_{jk}} + \frac{\partial K_2}{\partial w_{jk}} + \frac{\partial K_3}{\partial w_{jk}}\right] \\ &= ||f_{\theta_b}(x')||_2^2 \cdot \mathbf{1}^\top h(x',\theta) + G(x,x';\theta)\end{aligned}$$

where

$$\begin{aligned}G(x,x';\theta) = &-f_{\theta_b}(x')^\top f_{\theta_b}(x) \cdot \mathbf{1}^\top h(x,\theta) - f_{\theta_b}(x')^\top f_{\theta_b}(x) \cdot \mathbf{1}^\top h(x,\theta) - ||f_{\theta_b}(x)||_2^2 \cdot \psi'^\top h(x,\theta) \\ &- f_{\theta_b}^\top(x') f_{\theta_b}(x) \cdot \mathbf{1}^\top \omega' + ||f_{\theta_b}(x)||_2^2 \cdot (\psi^\top h(x,\theta) + \mathbf{1}^\top \omega).\end{aligned}$$

We arrive at

$$\begin{aligned}&\mathrm{Tr}\Big\{\nabla_{\theta_t}^2\Big[\mathrm{CE}((x,y),g_\theta) + \lambda_t \cdot \max_{||\epsilon||_p \le \delta} \mathrm{KLloss}((x,x+\epsilon),g_\theta)\Big]\Big\} \\ &\qquad = ||f_{\theta_b}(x)||_2^2 \cdot \mathbf{1}^\top h(x,\theta) + \lambda_t(||f_{\theta_b}(x')||_2^2 \cdot \mathbf{1}^\top h(x',\theta) + G(x,x';\theta)),\end{aligned} \tag{28}$$

where

$$x' = \arg\max_{||\epsilon||_p \le \delta} [\mathrm{KLloss}((x,x+\epsilon),F_\theta)] + x$$

Finally, we complete the derivation by denoting $\mathrm{TrH}_\mathrm{t}(x,x';\lambda_t,\theta) \overset{\text{def}}{=} \mathrm{Tr}\Big\{\nabla_{\theta_t}^2\Big[\mathrm{CE}((x,y),g_\theta) + \lambda_t \cdot \max_{||\epsilon||_p \le \delta} \mathrm{KLloss}((x,x+\epsilon),g_\theta)\Big]\Big\}$.

B. TrH Regularization for Other Adversarial Losses

In this section, we apply TrH regularization to some additional robust losses other than AT or TRADES.

B.1. ALP [29]

Similar to TRADES, Adversarial Logit Pairing (ALP) [29] is another method that pushes points away from the decision boundary by regularizing the ℓ_2 difference between the clean and the adversarial softmax outputs. Formally, ALP minimizes the following loss during training,

$$\hat{R}_{\mathbb{A}}(\theta, D^m) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{(x,y)\in D^m} \text{CE}((x', y), g_\theta) + \lambda_A ||s(g_\theta(x)) - s(g_\theta(x'))||_2^2$$
$$\text{where } x' = x + \arg\max_{||\epsilon||_p \leq \delta} \text{CE}((x+\epsilon, y), g_\theta).$$

We hereby derive TrH regularization for $\hat{R}_{\mathbb{A}}(\theta, D^m)$. Using Proposition 1, we know that

$$\text{TrH}(\text{CE}((x', y), g_\theta)) = ||f_{\theta_b}(x')||_2^2 \cdot \mathbf{1}^\top h(x', \theta).$$

The rest of this section will focus on the ℓ_2 distance loss,

$$S \stackrel{\text{def}}{=} ||s(g_\theta(x)) - s(g_\theta(x'))||_2^2 = \sum_i (s(g_i) - s(g_i'))^2, \tag{29}$$

where $g = g_\theta(x), g' = g_\theta(x')$. To compute $\text{Tr}(\nabla^2_{\theta_t} S)$, we need to re-use Lemma 1 and 2 to obtain the derivatives of the softmax and log-softmax outputs, i.e.

$$\frac{\partial s(g_\theta(x))}{g_\theta(x)} = \Phi \stackrel{\text{def}}{=} \text{Diag}(s(g_\theta(x))) - s(g_\theta(x)) \cdot s(g_\theta(x))^\top$$
$$\text{and} \quad \frac{\partial \log s(g_\theta(x))}{g_\theta(x)} = \Psi \stackrel{\text{def}}{=} I - \mathbf{1} \cdot s(g_\theta(x))^\top,$$

as well as

$$\frac{\partial \Phi_{ik}}{\partial \{g_\theta(x)\}_k} = \Phi_{kk}\Psi_{ki}.$$

For the ease of notation, we let $w \stackrel{\text{def}}{=} \theta_t$ be the weights of the top-layer. Now we are ready to write the first-order derivative of S w.r.t w_{jk} as follows

$$\frac{\partial S}{\partial w_{jk}} = \sum_i 2(s(g_i) - s(g_i')) \left[\frac{\partial s(g_i)}{\partial w_{jk}} - \frac{\partial s(g_i')}{\partial w_{jk}}\right]$$
$$= 2\sum_i (s(g_i) - s(g_i'))(\Phi_{ik}\{f_{\theta_b}(x)\}_j - \Phi'_{ik}\{f_{\theta_b}(x')\}_j).$$

The second-order derivative is equal to

$$\frac{\partial^2 S}{\partial w_{jk}^2} = 2\sum_i \left[\frac{\partial s(g_i)}{\partial w_{jk}} - \frac{\partial s(g_i')}{\partial w_{jk}}\right](\Phi_{ik}\{f_{\theta_b}(x)\}_j - \Phi'_{ik}\{f_{\theta_b}(x')\}_j) + (s(g_i) - s(g_i'))\left[\frac{\partial \Phi_{ik}}{\partial w_{jk}}\{f_{\theta_b}(x)\}_j - \frac{\partial \Phi'_{ik}}{\partial w_{jk}}\{f_{\theta_b}(x')\}_j\right]$$
$$= 2\sum_i (\Phi_{ik}\{f_{\theta_b}(x)\}_j - \Phi'_{ik}\{f_{\theta_b}(x')\}_j)^2 + (s(g_i) - s(g_i'))(\Phi_{kk}\Psi_{ki}\{f_{\theta_b}(x)\}_j^2 - \Phi'_{kk}\Psi'_{ki}\{f_{\theta_b}(x')\}_j^2). \tag{30}$$

Similar to TRADES, if one stops gradients over the clean logit g, then Eq. 30 can be simplified as,

$$\frac{\partial^2 S}{\partial w_{jk}^2} = 2\sum_i (\Phi'_{ik}\{f_{\theta_b}(x')\}_j)^2 + (s(g_i) - s(g_i'))(-\Phi'_{kk}\Psi'_{ki}\{f_{\theta_b}(x')\}_j^2)$$
$$= 2\{f_{\theta_b}(x')\}_j^2 \Big(||\Phi_k||_2^2 - \Phi'_{kk}((s(g) - s(g'))^\top \Psi'_k)\Big).$$

Therefore, the trace of Hessian is equal to,

$$\begin{aligned}\mathrm{Tr}(\nabla^2_{\theta_t} S) &= \sum_{jk} 2\{f_{\theta_b}(x')\}_j^2 \Big(||\Phi_k||_2^2 - \Phi'_{kk}((s(g) - s(g'))^\top \Psi'_k) \\ &= 2||f_{\theta_b}(x')||_2^2 \cdot \sum_k \Big(||\Phi_k||_2^2 - \Phi'_{kk}((s(g) - s(g'))^\top \Psi'_k) \Big).\end{aligned}$$

If the gradient over g is kept, then one can sum over the feature dimension j and the class dimension k in Eq. 30. To put together, the TrH regularization term for ALP is given as follows:

$$\frac{1}{m} \sum_{(x,y)\in D^m} ||f_{\theta_b}(x')||_2^2 \cdot \mathbf{1}^\top h(x', \theta) + \lambda_A \cdot \mathrm{Tr}(\nabla^2_{\theta_t} S).$$

B.2. MART [46]

In [46], Wang et al. proposed MART as a robust training loss that focus more on points that are not classified correctly. Different from AT, TRADES and ALP, MART explicitly aims to increase the margin between the top prediction and the second best candidate. In what follows we provide the TrH regularization for MART. We denote the MART loss as $\hat{R}_{\text{M}}$, which contains two components: a boosted Cross-Entropy (BCE) loss and a weighted KL-Divergence (WKL),

$$\hat{R}_{\text{M}}(\theta, D^m) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{(x,y)\in D^m} \mathrm{BCE}((x', y), g_\theta) + \lambda_m \mathrm{WKL}((x, x'), g_\theta),$$

where

$$\mathrm{BCE}((x', y), g_\theta) \stackrel{\text{def}}{=} \mathrm{CE}((x, y), g_\theta) + (-\log(1 - \max_{\kappa \neq y} s(\{g_\theta(x')\}_\kappa)),$$

$$\text{and, } \mathrm{WKL}((x, x'), g_\theta) \stackrel{\text{def}}{=} \mathrm{KL}(s(g_\theta(x)) || s(g_\theta(x')))(1 - s(\{g_\theta(x))\}_y).$$

Here

$$x' = x + \arg\max_{||\epsilon||_p \leq \delta} \mathrm{CE}((x, y), \theta). \tag{31}$$

First, we derive TrH of the BCE loss with respect to the top-layer weights θ_t,

$$\mathrm{Tr}(\nabla^2_{\theta_t} \mathrm{BCE}) = \mathrm{Tr}(\nabla^2_{\theta_t} [\mathrm{CE}((x, y), g_\theta)]) + \mathrm{Tr}(\nabla^2_{\theta_t} [-\log(1 - \max_{\kappa \neq y} s(\{g_\theta(x')\}_\kappa)]).$$

Using intermediate steps from the proof of Proposition 2 in Appendix A, we have that

$$\mathrm{Tr}(\nabla^2_{\theta_t} [\mathrm{CE}((x, y), g_\theta)]) = ||f_{\theta_b}(x)||_2^2 \cdot \mathbf{1}^\top h(x, \theta).$$

In order to compute $\mathrm{Tr}(\nabla^2_{\theta_t} [-\log(1 - \max_{\kappa \neq y} s(\{g_\theta(x')\}_\kappa)])$, we make use of Lemma 1 and 2 to obtain the derivatives of the softmax and log-softmax outputs,

$$\frac{\partial s(g_\theta(x'))}{\partial g_\theta(x')} = \Phi' \stackrel{\text{def}}{=} \mathrm{Diag}(s(g_\theta(x'))) - s(g_\theta(x')) \cdot s(g_\theta(x'))^\top$$

$$\text{and} \quad \frac{\partial \log s(g_\theta(x'))}{\partial g_\theta(x')} = \Psi' \stackrel{\text{def}}{=} I - \mathbf{1} \cdot s(g_\theta(x'))^\top,$$

as well as,

$$\frac{\partial \Phi'_{ik}}{\partial \{g_\theta(x')\}_k} = \Phi'_{kk} \Psi'_{ki}.$$

Let us denote $K = -\log(1 - \max_{\kappa \neq y} s(\{g_\theta(x')\}_\kappa)]$ and $w = \theta_t$ for the ease of notation. Then the first-order derivative of K w.r.t. w_{jk} is,

$$\frac{\partial K}{\partial w_{jk}} = \frac{\Phi'_{\kappa^* k}}{1 - s(\{g_\theta(x')\}_{\kappa^*})} \{f_{\theta_b}(x')\}_j,$$

where

$$\kappa^* = \arg\max_{\kappa \neq y} s(\{g_\theta(x')\}_\kappa).$$

The second-order derivative is,

$$\begin{aligned}
\frac{\partial^2 K}{\partial w_{jk}^2} &= \frac{\frac{\partial \Phi'_{\kappa^* k}}{\partial w_{jk}}(1 - s(\{g_\theta(x')\}_{\kappa^*})) + \Phi'_{\kappa^* k}\Phi'_{\kappa^* k}\{f_{\theta_b}(x')\}_j}{(1 - s(\{g_\theta(x')\}_{\kappa^*}))^2}\{f_{\theta_b}(x')\}_k \\
&= \frac{\Phi'_{kk}\Psi'_{k\kappa^*}\{f_{\theta_b}(x')\}_k(1 - s(\{g_\theta(x')\}_{\kappa^*})) + \Phi'_{\kappa^* k}\Phi'_{\kappa^* k}\{f_{\theta_b}(x')\}_j}{(1 - s(\{g_\theta(x')\}_{\kappa^*}))^2}\{f_{\theta_b}(x')\}_j \\
&= \{f_{\theta_b}(x')\}_j^2 \left[\frac{\Phi'_{\kappa^* k}\Phi'_{\kappa^* k}}{1 - s(\{g_\theta(x')\}_{\kappa^*})} + \frac{\Phi'^2_{\kappa^* k}}{(1 - s(\{g_\theta(x')\}_{\kappa^*}))^2}\right].
\end{aligned}$$

Thus,

$$\begin{aligned}
\mathrm{Tr}(\frac{\partial^2 K}{\partial w_{jk}^2}) &= \sum_{jk}\{f_{\theta_b}(x')\}_j^2 \left[\frac{\Phi'_{\kappa^* k}\Phi'_{\kappa^* k}}{1 - s(\{g_\theta(x')\}_{\kappa^*})} + \frac{\Phi'^2_{\kappa^* k}}{(1 - s(\{g_\theta(x')\}_{\kappa^*}))^2}\right] \\
&= ||f_{\theta_b}(x')||_2^2 \sum_k \left[\frac{\Phi'_{\kappa^* k}\Phi'_{\kappa^* k}}{1 - s(\{g_\theta(x')\}_{\kappa^*})} + \frac{\Phi'^2_{\kappa^* k}}{(1 - s(\{g_\theta(x')\}_{\kappa^*}))^2}\right],
\end{aligned}$$

and

$$\mathrm{Tr}(\nabla^2_{\theta_t}\mathrm{BCE}) = ||f_{\theta_b}(x)||_2^2 \cdot \mathbf{1}^\top h(x,\theta) + ||f_{\theta_b}(x')||_2^2 \sum_k \left[\frac{\Phi'_{\kappa^* k}\Phi'_{\kappa^* k}}{1 - s(\{g_\theta(x')\}_{\kappa^*})} + \frac{\Phi'^2_{\kappa^* k}}{(1 - s(\{g_\theta(x')\}_{\kappa^*}))^2}\right].$$

Next, we derive the trace of Hessian for WKL loss. Similarly to before, by stopping the gradient over the clean logit $g_\theta(x)$, the trace of Hessian of WKL will be the same as the KL loss used in TRADES, which has been shown in Proposition 2. Namely, in this case

$$\mathrm{Tr}(\nabla^2_{\theta_t}\mathrm{WKL}) = ||f_{\theta_b}(x')||_2^2 \cdot \mathbf{1}^\top h(x',\theta).$$

On the other hand, if the gradient on $g_\theta(x)$ is computed, there is an additional term in $\mathrm{Tr}(\nabla^2_{\theta_t}\mathrm{WKL})$ (similar but more complicated than $G(x,x';\theta)$ derived in the proof of Proposition 2 in Appendix A), which we will leave as future work. Finally, we present the TrH regularization for MART as follows,

$$\mathrm{TrH_M}(x,x';\lambda_m,\theta) = \mathrm{Tr}(\nabla^2_{\theta_t}\mathrm{BCE}) + \lambda_m \mathrm{Tr}(\nabla^2_{\theta_t}\mathrm{WKL}).$$

This concludes our derivations of the TrH regularization for the MART loss.

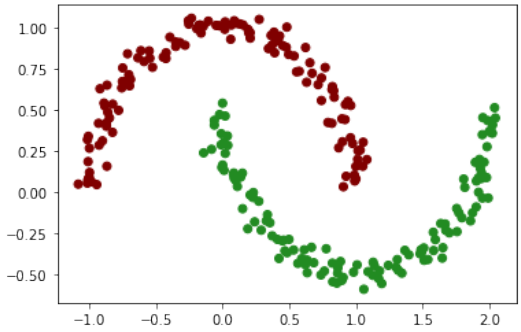

Figure 4. Visualization of Two Moons dataset. The task is to classify the points into two classes (red and green).

C. Impact of Top-Layer Regularization

In this section, we provide a full description of Example 1, as well as some additional results and analysis.

- **Dataset.** We use the Two Moons Dataset where the input features $x_i \in \mathbb{R}^2$ and the label $y_i \in \{0, 1\}$ (Figure 4).
- **Network.** We use a dense network Dense(100)-ReLU-Dense(100)-ReLU-Dense(2).
- **Training without Regularization (Standard).** We train the network with AT(`base`), such that it is robust in an ℓ_∞ ball of size 0.02 with 1 PGD step. We use a momentum-SGD with learning rate 0.1 for 100 epochs.
- **Training with Top TrH Regularization (Top).** We further add the TrH of the top layer as a regularization term in the loss. We compute the TrH at the top layer using Proposition 1. The regularizing coefficient for the top-layer TrH is 0.5 (i.e. $loss = AT + 0.5 * \mathrm{TrH}_{top}$) as we find it needs a stronger penalty due to the lack of TrH from other layers.
- **Training with Full TrH Regularization (Full).** In additional to the Standard setup, we add the TrH of the full network as a regularization term in the loss. We numerically compute the Hessian and its trace. The regularizing coefficient for the full-network TrH is 0.05 (i.e. $loss = AT + 0.05 * \mathrm{TrH}_{full}$).

Figure 5, shows pairs of plots of **the sum and standard deviation of the eigenvalues of the Hessian matrix** across all layers, as well as for each of the three layers separately. We compare the three setting (1) no regularization (Standard, blue); (2) the top-layer TrH regularization (Top, orange); and (3) the full-network TrH regularization (Full, green).

Focusing on the top-left plot in Figure 5, we see that the TrH decreases across time with standard training (blue), but at the same time, that full-network and top-layer TrH regularization decrease it even further. In particular, top-layer TrH regularization is nearly as effective as directly regularizing the TrH for all layers. The standard deviation plots (on the right side of each TrH plot) show a a decrease in the standard deviation, which implies there is a contraction effect on the eigenvalues of the Hessian, with both positive and negative values approaching towards 0. This rules out the possibility that the TrH reduction comes from an increase in the magnitude of the negative eigenvalues. In fact, TrH regularization effectively contracts the magnitude of all eigenvalues and leads to a smoother region in the loss surface.

We further plot the TrH and standard deviation of Hessian eigenvalues for each individual layer respectively (where Layer 3 is the top layer). Except for the first layer, whose eigenvalues seem to be small from the onset, the eigenvalue contraction effect is evident as training progresses, with a stronger and similar effect for both TrH regularization settings.

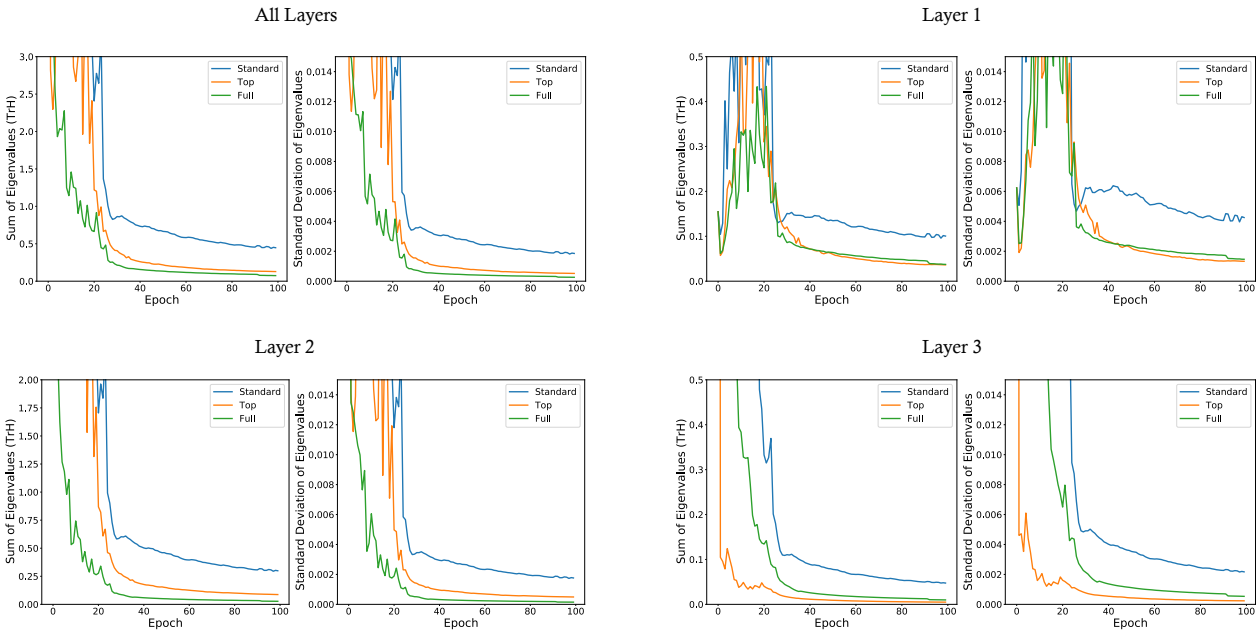

Figure 5. The figure shows pairs of plots for the sum and the standard deviation of the Hessian matrix eigenvalues. The top-left pair corresponds to the Hessian of all layers, while the rest to each of the three layers separately, with Layer 3 being the top layer (note that the sum of eigenvalues is exactly the trace of Hessian). As can be seen on the top-left , the sum and standard deviation decrease in standard training (blue) and moreover, this effect is amplified by direct TrH regularization with similar results for full network regularization (green) and top-layer regularization (orange). This similarity can be explained by our Thm.4

Pre-Selected Hyper-parameters (CIFAR-10/100)

Parameter	Reason To Select	Final Frozen Value
`img_size`	to be compatible with ϵ	32×32
`patch_size`	same as [33]	4×4
`parameter_init`	publicly available	ImageNet21K checkpoint
`batch_size`	memory	768
`warm_up iterations`	standard	500

Pre-tuning Stage

Parameter	Range	Explanation	Final Frozen Value
`optimizer`	{'sgd+momentum', 'adam'}	optimizer	sgd+momentum
`base_lr`	$\{0.001, 0.01, 0.1\}$	initial learning rate	0.1
`l2_reg`	$\{0, 0.0001, 0.001\}$	coefficient for L2 regularization	0
`data_aug`	{'fb', 'crop'}	data augmentation method 'fb': flip and random brightness 'crop': randomly crop and upsample	'fb'
`downsample`	{'cubic', 'nearest', 'bilinear' }	downsample method for the first kernel to fit the patch size from 16×16 to 4×4 [33]	cubic
`patch_stride`	$\{2, 4\}$	the stride to create image patches	2
`data_range`	$\{[-1, 1], [0, 1]$, 'centered'$\}$	data range 'centered': 0-mean and 1-std	centered
`use_cutmix`	{True, False}	whether to use cutmix augmentation	False

Table 3. Frozen Hyper-parameters for CIFAR-10 and CIFAR-100 (including DDPM images) in All Experiments.

Pre-Selected Hyper-parameters (ImageNet)

Parameter	Reason To Select	Final Frozen Value
`data_range`	to align with ImageNet21K checkpoint	$[-1, 1]$
`img_size`	to be compatible with ϵ	224×224
`patch_size`	standard	16×16
`parameter_init`	publicly available	ImageNet21K checkpoint
`batch_size`	memory	64
`warm_up iterations`	standard	500

Pre-tuning Stage

Parameter	Range	Explanation	Final Frozen Value
`optimizer`	{'sgd+momentum', 'adam'}	optimizer	'sgd+momentum'
`base_lr`	$\{0.001, 0.01, 0.1\}$	initial learning rate	0.01
`decay_type`	{'multistep', 'cosine'}	the function used to schedule lr decay	cosine
`l2_reg`	$[0.0001, 0.001]$	coefficient for ℓ_2 regularization	0.0001
`data_aug`	{'fb', 'crop'}	data augmentation method 'fb': flip and random brightness 'crop': randomly crop and upsample	'fb'
`use_cutmix`	{True, False}	whether to use cutmix augmentation	False

Table 4. Frozen Hyper-parameters for ViT-B16 and ViT-L16 to reproduce results in Table. 1.

D. Hyper-parameters Shared by All Methods

Pre-Tuning. Training ViTs can sometimes be challenging due to a large amount of hyper-parameters. For the choice of the parameters that are shared across different defense methods, e.g. batch size, patch size, training iterations, and etc., we do a large grid search and choose the parameter setting that produce the best results on TRADES(`base`) and use it for all the methods. This step is referred to as pre-tuning and is done per-dataset.

Pre-selected Hyper-parameters. There is a set of hyper-parameters requiring no tuning because they are commonly selected in the literature. In the top of Table 4 and 3, we write down the these parameters and explain the reason for choosing a particular value.

Tunable Hyper-parameters. In the bottom of Table 4 and 3, we show our choice of hyper-parameters for ImageNet and CIFAR-10/100, respectively. This includes

- `optimizer`. We tested a momentum-SGD (`sgd+momentum`) and an Adam optimizer (`adam`). We found that momentum-SGD is more stable in fine-tuning the ViT from a pre-trained checkpoint.
- `base_lr`, `warm_up_iterations` and `decay_type`. We linearly increase the learning rate from 0 to the `base_lr` during `warm_up_iterations`. After warm-up, we gradually schedule the learning rate based on the `decay_type`. We experiment with a multi-step and a cosine decay and find no apparent difference between these two schedulers. In the end, we choose cosine because it has less hyper-parameters to choose compared to the multi-step one.
- `l2_reg`. On ImageNet, we find ℓ_2 regularization with a penalty of 0.0001 helps the baseline TRADES(`base`). On CIFAR-10/100, we find that ℓ_2 regularization may not be necessary when using DDPM data.
- `cutmix`. In both cases we do not find the cut mix augmentation [51] help to improve the results.
- `data_range`. On CIFAR10/100, we find that centered data, i.e. normalizing the data to have 0 mean and (close to) 1 standard deviation, provides better results than scaling the images to [-1, 1] (the range of data used by the pre-trained checkpoint). We simply subtract the CIFAR images from the average of per-channel mean (0.47) and divide it with the average of per-channel standard deviation (0.25). For ImageNet, we still use [-1, 1] as the data range. Notice that ϵ ball needs to be re-scaled for both data ranges describe above. For example, when reporting results on $\epsilon = 0.031$ and using centered data, we need to use $\epsilon/0.25$ as the actual noise bound passed to the attacker. When normalizing the data to $[-1, 1]$, we need to pass $\epsilon/0.5$ to the attacker.

- `patch_size`. The size of the image patch of the input sequence to ViT. 16×16 is the standard size of pre-trained ViT models on ImageNet21K. Thus, when fine-tuning on ImageNet, we use 16×16. CIFAR images are a lot smaller compared to ImageNet images. As a result, we use a smaller patch size of 4×4 to produce more patches. Using a smaller patch size requires some modifications to ViT architecture and we discuss this in detail in Appendix G.

- `downsample` and `patch_stride`. These are particular to CIFAR images. Please refer to Appendix G for detail.

E. Hyper-parameters for Specific Methods

We fine-tune ViTs after common hyper-parameters are locked after pre-tuning. For all methods, we take 10 PGD steps on CIFAR-10/100 and 7 steps for ImageNet during the training. We report the best results for each method after trying different sets of hyper-parameters. This usually involves method-specific parameters. We elaborate what hyper-paramter is tuned as follows and report the final values used in the experiments in Table 5 (CIFAR-10/100) and 6 (ImageNet).

- **base**: we use $\lambda_t = 6$ for TRADES training and λ_t is consistent across all experiments.

- **SWA**: we use $\alpha = 0.995$ so that $\theta_{\texttt{avg}} \leftarrow 0.995 * \theta_{\texttt{avg}} + 0.005 * \theta^{(t+1)}$ as this value is used in the literature [21]. Therefore, there is no tuning in SWA.

- **S2O**: The only parameter that requires tuning is the penalty α of the second-order statistic in Eq 2. In the authors' implementation, we find $1 - \alpha$ is used to balance the regularization with the robust loss (AT or TRADES). These hyper-parameters are hard-coded in the latest commit f2d037b[1] so we directly use their choice of hyper-parameters. Namely, for AT loss, the finally loss in S2O training is set to

$$0.9 * AT\_loss + 0.1 * S2O\_loss$$

  and for TRADES training the final loss is

$$0.7 * \frac{1}{m} \sum_{i}^{m} \text{CE}((x_i, y_i), F_\theta) + 0.3 * S2O\_loss + \frac{\lambda_t}{m} \sum_{i}^{m} \max_{||\epsilon_i||_p \leq \delta} \text{KLloss}((x_i, \epsilon_i), F_\theta).$$

- **AWP**: The two hyper-parameters in AWP that requires tuning are ψ and δ_{awp}, where ψ is the function to measure the noise added to the weights and δ_{awp} is the noise budget. We follow the choice made by Wu et al. [49] to choose ψ as the layer-wise ℓ_2 norm function so that the noise added to the weights in each layer should no greater than the ℓ_2 norm of the weights multiplied by δ_{awp}. Namely, suppose that $\xi^{(i)}$ is the noise added to a weight matrix $W^{(i)}$ at layer i, then we project $\xi^{(i)}$ such that

$$\frac{||\xi^{(i)} + W^{(i)}||_2}{||W^{(i)}||_2} \leq \delta_{awp}.$$

  For the choice of δ_{awp}, we sweep δ_{awp} over the interval $\{0.0001, 0.0005, 0.001, 0.005, 0.01\}$.

- **TrH**: The two hyper-parameters in TrH that requires tuning are the ℓ_2 weight penalty γ and the TrH penalty λ. For γ, we find 0.001 as a reasonable choice for CIFAR-10/100 and 0.0001 as a reasonable choice for ImageNet. For λ, we sweep the interval $\{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01\}$. Furthermore, we also consider three types of schedulers: 'constant', 'linear', 'multistep(0.1-0.5:0.1)' for λ scheduling. Intuitively, a strong TrH regularization at the very beginning may lead the model to a flat highland instead of a flat minimum where we get a degenerated model. Ramping up λ to the chosen value allows the model to focus more on accuracy and robustness at the early stage. In practice, we find that CIFAR10/100 is not sensitive to the choice of the λ schedulers; however, ImageNet favors 'linear' or 'multistep' schedulers over the 'constant' λ.

  - 'constant'. No λ scheduling.
  - 'linear'. We ramp up λ from 0 to the chosen value from iteration 1 to the end.
  - 'multistep(0.1-0.5:0.1)'. We use 0.01 * λ before finishing 10% of the total iterations. We use 0.1 * λ after finishing 10% and before 50% of the total iterations. After finishing 50% of iterations, we use λ.

[1] https://github.com/Alexkael/S2O/tree/f2d037b9611f732278341182509925 1f7978f54e

Fine-tuning Hyper-parameters (CIFAR-10/100, ViT-L16)

Defense	Hyper-parameter	Range	Final Choice (CIFAR-10)	Final Choice (CIFAR-100)
AT(AWP)	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0005	0.0005
AT(SWA)	α	-	0.995	0.995
AT(S2O)	α	-	0.1	0.1
AT(TrH)	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.00001	0.01
	λ schedule	{'constant', 'linear', 'multistep'}	'multistep'	'multistep'
	γ	-	0.001	0.001
TRADES(AWP), λ_t=6	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0005	0.0001
TRADES(SWA), λ_t=6	α	-	0.995	0.995
TRADES(S2O), λ_t=6	α	-	0.3	0.3
TRADES(TrH), λ_t=6	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0001	0.0001
	λ schedule	{'constant', 'linear', 'multistep'}	'multistep'	'constant'
	γ	-	0.001	0.001

Fine-tuning Hyper-parameters (CIFAR-10/100, Hybrid-L16)

Defense	Hyper-parameter	Range	Final Choice (CIFAR-10)	Final Choice (CIFAR-100)
AT(AWP)	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0005	0.0005
AT(SWA)	α	-	0.995	0.995
AT(S2O)	α	-	0.1	0.1
AT(TrH)	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0005	0.0005
	λ schedule	{'constant', 'linear', 'multistep'}	'multistep'	'multistep'
	γ	-	0.001	0.001
TRADES(AWP), λ_t=6	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0005	0.0005
TRADES(SWA), λ_t=6	α	-	0.995	0.995
TRADES(S2O), λ_t=6	α	-	0.3	0.3
TRADES(TrH), λ_t=6	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0001	0.0001
	λ schedule	{'constant', 'linear', 'multistep'}	'multistep'	'multistep'
	γ	-	0.001	0.001

Table 5. Hyper-parameters used in each defense and the values used to reproduce CIFAR10/100 results in Table. 1.

F. Additional Results

We provide results on CIFAR-10/100 with ViT-B16 in Table 7 and ImageNet with Hybrid-L16 in Table 8. These additional results are consistent with what we have found in Section 4.2 of the paper: in CIFAR-10/100, TrH is consistently among the top or silver results; in ImageNet, TrH has significantly advantages over the other baseline methods.

In addition to the run-time comparison on NIVIDA RTX chips in Table 2, we also report the per epoch time on RTX chips as well as TPUv4 in Table 9. We do not include S2O in the table because its memory usage requires twice as many chips.

Fine-tuning Hyper-parameters (ImageNet, ViT-B16)

Defense	Hyper-parameter	Range	Final Choice (ℓ_∞)	Final Choice (ℓ_2)
AT(AWP)	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0001	0.0001
AT(SWA)	α	-	0.995	0.995
AT(S2O)	α	-	0.1	0.1
AT(TrH)	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0001	0.00005
	λ schedule	{'constant', 'linear', 'multistep'}	'multistep'	'linear'
	γ	-	0.0001	0.0001
TRADES(AWP), λ_t=6	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0001	0.0001
TRADES(SWA), λ_t=6	α	-	0.995	0.995
TRADES(S2O), λ_t=6	α	-	0.3	0.3
TRADES(TrH), λ_t=6	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.00005	0.00005
	λ schedule	{'constant', 'linear', 'multistep'}	'linear'	'linear'
	γ	-	0.0001	0.0001

Fine-tuning Hyper-parameters (ImageNet, ViT-L16)

Defense	Hyper-parameter	Range	Final Choice (ℓ_∞)	Final Choice (ℓ_2)
AT(AWP)	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0001	0.0001
AT(SWA)	α	-	0.995	0.995
AT(S2O)	α	-	0.1	0.1
AT(TrH)	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0005	0.0005
	λ schedule	{'constant', 'linear', 'multistep'}	'multistep'	'linear'
	γ	-	0.0001	0.0001
TRADES(AWP), λ_t=6	δ_{awp}	{0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0001	0.0001
TRADES(SWA), λ_t=6	α	-	0.995	0.995
TRADES(S2O), λ_t=6	α	-	0.3	0.3
TRADES(TrH), λ_t=6	λ	{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01}	0.0005	0.00005
	λ schedule	{'constant', 'linear', 'multistep'}	'multistep'	'multistep'
	γ	-	0.0001	0.0001

Table 6. Hyper-parameters used in each defense and the values used to reproduce ImageNet results in Table. 1.

$\ell_\infty(\delta = 8/255)$	ViT-B16			
SE= ±0.5%	CIFAR-10		CIFAR-100	
Defense	Clean(%)	AA(%)	Clean (%)	AA(%)
AT(base)	87.5	60.3	60.0	30.4
AT(SWA)	86.9	60.4	64.1	**33.7**
AT(S2O)	86.8	60.4	63.2	31.5
AT(AWP)	87.3	<u>61.3</u>	61.5	32.2
AT(TrH)	88.4	**61.5**	65.0	<u>33.0</u>
TRADES(base)	85.4	<u>60.8</u>	58.6	30.5
TRADES(SWA)	85.6	<u>60.6</u>	62.7	**33.0**
TRADES(S2O)	85.9	**61.1**	63.5	31.5
TRADES(AWP)	84.7	<u>60.1</u>	60.8	<u>32.2</u>
TRADES(TrH)	85.8	**61.1**	63.8	**33.0**

Table 7. Additional results for CIFAR-10/100 using ViT-B16. Clean: % of Top-1 correct predictions. AA: % of Top-1 correct predictions under AutoAttack. A max Standard Error (SE) [44] = $\sqrt{0.5 * (1 - 0.5)/m}$ (m as the number of test examples) is computed for each dataset. The best results appear in bold. Underlined results are those that fall within the SE range of the result and are regarded roughly equal to the best result.

ImageNet	Hybrid-L16			
SE= ±0.2%	$\ell_\infty(\delta = 4/255)$		$\ell_2(\delta = 3.0)$	
Defense	Clean(%)	AA(%)	Clean (%)	AA(%)
AT(base)	72.6	40.7	72.2	40.6
AT(SWA)	72.7	40.4	72.7	40.5
AT(S2O)	72.8	43.6	72.3	40.9
AT(AWP)	67.7	39.4	67.9	40.3
AT(TrH)	75.0	**46.2**	74.4	**45.9**
TRADES(base)	79.5	37.6	78.0	38.6
TRADES(SWA)	72.9	40.9	71.3	40.8
TRADES(S2O)	73.8	41.3	72.2	41.2
TRADES(AWP)	66.4	38.8	65.3	40.9
TRADES(TrH)	75.9	**45.7**	73.3	**45.6**

Table 8. Addtional Results for ImageNet using Hybird-L16. Clean: % of Top-1 correct predictions. AA: % of Top-1 correct predictions under AutoAttack. A max Standard Error (SE) [44] = $\sqrt{0.5 * (1 - 0.5)/m}$ (m as the number of test examples) is computed for each dataset. The best results appear in bold. Underlined results are those that fall within the SE range of the result and are regarded roughly equal to the best result.

Method	8 TPUv4 chips	2 RTX chips
base	2.6	32.3
TrH	2.8	34.2
SWA	3.9	40.4
AWP	5.3	48.1

Table 9. Comparisons of runtime measured by per epoch time. S2O is not included because the memory usage requires twice as many chips.

G. ViT Architecture

We describe the architectures of Vision Transformers used in our experiments in Section 4.

- ViT-B16 includes 12 transformer layers with hidden size 768, MLP layer size 3072 and 12 heads in the multi-head attention.
- ViT-L16 includes 24 transformer layers with hidden size 1024, MLP layer size 4096 and 16 heads in the multi-head attention.
- Hybird-L16 is a hybrid model of a ResNet-50 and a ViT-L16 model. The patches fed into ViT are feature representations encoded by a ResNet-50 and projected to the Transformer dimensions [14]. Dissimilar to a standard ResNet-50 feature encoder, Dosovitskiy et al. [14] replaced Batch Normalization with Group Normalization [50] and use standardized Convolution [37]. Moreover, Dosovitskiy et al. [14] removed stage 4, placed the same number of layers in stage 3 (keeping the total number of layers), and took the output of this extended stage 3 as the input of ViT.

The pretrained ViT-B16 and ViT-L16 checkpoints use patch size 16×16. However, since the images in CIFAR10/100 are of size 32×32, there will be only 4 patches when using the original patch size. Therefore, we downsample the kernel of the first convolution to produce image patches of 4×4. Among different ways of downsampling we find the cubic interpolation gives the best results, which is the one chosen in pre-tuning as discussed in Appendix D. Indeed, Mahmood et al. [33] empirically finds that such down-sampling provides better results for CIFAR10 images. Furthermore, in generating patches, we also use a stride of 2 instead of 4, because we find using overlapped patches increases the sequence length and results in better performance.