Supplementary Material

1. Proof of Theorem 1

Proof. The definition of h can be expanded to

$$h(\vec{x}) = \vec{w}^T f(\vec{x}) + \vec{w}^T \vec{z} + b, \quad \vec{z} \sim \mathcal{N}(0, \Sigma),$$

and be reinterpreted as

$$h(\vec{x}) \sim \mathcal{N}(\vec{w}^T f(\vec{x}) + b, \vec{w}^T \Sigma \vec{w}).$$

Going further, we can see that the distribution of the margin function is

$$m_h(\vec{x}, y) \sim \mathcal{N}(y(\vec{w}^T f(\vec{x}) + b), \vec{w}^T \Sigma \vec{w}),$$

for which the probability of being less than zero is given by the cumulative distribution function for the normal distribution,

$$P(m_h(\vec{x}, y) < 0) = \Phi\left(\frac{-y(\vec{w}^T f(\vec{x}) + b)}{\sqrt{\vec{w}^T \Sigma \vec{w}}}\right). \tag{1}$$

From the increasing monotonicity of Φ, we also have that

$$\begin{aligned} &\max_{\vec{\delta}:\|\vec{\delta}\|_p \le \epsilon} \Phi\left(\frac{-y(\vec{w}^T f(\vec{x} + \delta) + b)}{\sqrt{\vec{w}^T \Sigma \vec{w}}}\right) \\ &=\Phi\left(\frac{\max_{\vec{\delta}:\|\vec{\delta}\|_p \le \epsilon} -y(\vec{w}^T f(\vec{x} + \delta) + b)}{\sqrt{\vec{w}^T \Sigma \vec{w}}}\right). \end{aligned}$$

Suppose the adversarial perturbation, δ, causes the output of the non-stochastic version of h to change by a magnitude of $\Delta_p^{\tilde{h}}(\vec{x}, \epsilon)$. There are a number of ways, such as local Lipschitz constants (Tsuzuku et al., 2018; Gouk & Hospedales, 2020), that can be used to bound the quantity for simple networks. Substituting $\Delta_p^{\tilde{h}}$ into the previous equation yields

$$\begin{aligned} &\max_{\vec{\delta}:\|\vec{\delta}\|_p \le \epsilon} P(m_h(\vec{x} + \delta, y) \le 0) \\ &\le \Phi\left(\frac{-y(\vec{w}^T f(\vec{x}) + b) + \Delta_p^{\tilde{h}}(\vec{x}, \epsilon)}{\sqrt{\vec{w}^T \Sigma \vec{w}}}\right). \end{aligned} \tag{2}$$

Finally, we know that the difference in probabilities of misclassification when the model is and is not under adversarial attack δ, is given by

$$\begin{aligned} G_{p,\epsilon}^h(\vec{x}, y) = &\max_{\vec{\delta}:\|\vec{\delta}\|_p \le \epsilon} P(m_h(\vec{x} + \delta, y) \le 0) \\ &- P(m_h(\vec{x}, y) \le 0). \end{aligned} \tag{3}$$

Table 1. Values for learning rate and weight decay for all experiments in our ablation study.

Benchmark	Learning rate	Weight decay
CIFAR-10	10^{-2}	10^{-4}
CIFAR-100	10^{-2}	10^{-4}
SVHN	10^{-2}	10^{-4}
FMNIST	10^{-4}	10^{-4}

Combining Equations 1 and 2 with Equation 3 results in

$$\begin{aligned} G(\vec{x}, y) \le &\Phi\left(\frac{-y(\vec{w}^T f(\vec{x}) + b) + \Delta_p^{\tilde{h}}(\vec{x}, \epsilon)}{\sqrt{\vec{w}^T \Sigma \vec{w}}}\right) \\ &- \Phi\left(\frac{-y(\vec{w}^T f(\vec{x}) + b)}{\sqrt{\vec{w}^T \Sigma \vec{w}}}\right). \end{aligned}$$

Because the Lipschitz constant of Φ is $\frac{1}{\sqrt{2\pi}}$, we can further bound G by

$$G(\vec{x}, y) \le \frac{\Delta_p^{\tilde{h}}(\vec{x}, \epsilon)}{\sqrt{2\pi \vec{w}^T \Sigma \vec{w}}}.$$

□

2. Hyperparameters of Experiments

In Table 1, we provide the hyperparameter setup for all the experiments in our ablation study. Note that we use the same values for both the isotropic and anisotropic variants of our model within the same benchmark. We further clarify that we use a batch size of 128 across all experiments. To choose these values, we split the training data into a training and a validation set and performed grid search. The grid consisted of negative powers of 10 $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ for both hyperparameters.

3. Larger Architectures

In the main body of the paper we explore how our method scales with the size of the backbone's architecture by experimenting with LeNet++ (small, 60 thousand parameters) and ResNet-18 (medium, 11 million parameters). In Table 2 we also provide some experimental results on CIFAR-10 with the much larger Wide-ResNet-34-10 architecture (46 million parameters)

Table 2. PGD test scores on CIFAR-10 using WRN-34-10, for different values of attack strength ϵ.

PGD($\epsilon/255$)	Clean	1	2	4	8	16	32	64	128
No Defense	0.97	0.63	0.60	0.26	0.12	0	0	0	0
WCA-Net	0.97	0.80	0.80	0.77	0.73	0.70	0.34	0.10	0

4. Enforcing Norm Constraints

In Section 3.1 we elaborate on how we use an ℓ^2 penalty to prevent the magnitude of the classifier vectors $\vec{w}$ and covariance matrix Σ from increasing uncontrollably. Another approach for controlling the magnitude of the parameters, is enforcing norm constraints after each gradient descent update, using a projected subgradient method. The projected subgradient method changes the standard update rule of the subgradient method from

$$\vec{\theta}^{(t+1)} \leftarrow \vec{\theta}^{(t)} - \alpha \nabla_{\vec{\theta}} \mathcal{L}(\vec{\theta}^{(t)}),$$

to

$$\begin{aligned}\vec{u}^{(t)} &\leftarrow \vec{\theta}^{(t)} - \alpha \nabla_{\vec{\theta}} \mathcal{L}(\vec{\theta}^{(t)}) \\ \vec{\theta}^{(t+1)} &\leftarrow \operatorname*{arg\,min}_{\vec{v} \in \Omega} \|\vec{v} - \vec{u}^{(t)}\|_2^2,\end{aligned}$$

where Ω is known as the feasible set. In our case there are three sets of parameters: the feature extractor weights, the linear classifier weights, and the covariance matrix. No projection needs to be applied to the extractor weights, as they are unconstrained. The linear classifier weights have an ℓ^2 constraint on the vector associated with each class, so their feasible set it an ℓ^2 ball—there is a known closed form projection onto the ℓ^2 ball (see, e.g., Gouk et al. (2021)). The feasible set for the covariance matrix is the set of positive semi-definite matrices with bounded singular values. This constraint can be enforced by performing a singular value decomposition on the updated covariance matrix, clipping the values to the appropriate threshold, and reconstructing the new projected covariance matrix (Lefkimmiatis et al., 2013). The final algorithm is given by

$$\begin{aligned}Y^{(t)} &\leftarrow \vec{\Sigma}^{(t)} - \alpha \nabla_{\Sigma} \mathcal{L}(\vec{\phi}^{(t)}, \vec{w}^{(t)}, L^{(t)}) \\ \vec{u}_i^{(t)} &\leftarrow \vec{w}_i - \alpha \nabla_{\vec{w}_i} \mathcal{L}(\vec{\phi}^{(t)}, \vec{w}^{(t)}, L^t) \\ \vec{\phi}^{(t+1)} &\leftarrow \vec{\phi}^{(t)} - \alpha \nabla_{\vec{\phi}} \mathcal{L}(\vec{\phi}^{(t)}, \vec{w}^{(t)}, L^t) \\ \vec{w}_i^{(t+1)} &\leftarrow \frac{1}{\max(1, \frac{\|\vec{u}_i^{(t)}\|_2}{\gamma})} \vec{u}_i^{(t)} \\ U^{(t)} S^{(t)} V^{(t)} &\leftarrow Y^{(t)T} Y^{(t)} \qquad (4) \\ \Sigma^{(t)} &\leftarrow U^{(t)} \tilde{S}^{(t)} V^{(t)} \\ L^{(t+1)T} L^{(t+1)} &\leftarrow \Sigma^{(t)}, \qquad (5)\end{aligned}$$

where (4) is performing a singular value decomposition, $\tilde{S}$ represents the clipped version of S, and (5) is computing the Cholesky decomposition.

5. Source Code and Reproducibility

The source code is openly available on GitHub: `https://github.com/peustr/WCA-net`.

References

Gouk, H. and Hospedales, T. M. Optimising network architectures for provable adversarial robustness. In *SSPD*, 2020.

Gouk, H., Hospedales, T. M., and Pontil, M. Distance-based regularisation of deep networks for fine-tuning. In *ICLR*, 2021.

Lefkimmiatis, S., Ward, J. P., and Unser, M. Hessian schatten-norm regularization for linear inverse problems. *IEEE transactions on image processing*, 22(5):1873–1888, 2013.

Tsuzuku, Y., Sato, I., and Sugiyama, M. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *NeurIPS*, 2018.