Dynamic Low-Rank Training with Spectral Regularization: Achieving Robustness in Compressed Representations

Steffen Schotthöfer [1] **H. Lexie Yang** [2] **Stefan Schnake** [1]

Abstract

Deployment of neural networks on resource-constrained devices demands models that are both compact and robust to adversarial inputs. However, compression and adversarial robustness often conflict. In this work, we introduce a dynamical low-rank training scheme enhanced with a novel spectral regularizer that controls the condition number of the low-rank core in each layer. This approach mitigates the sensitivity of compressed models to adversarial perturbations without sacrificing clean accuracy. The method is model- and data-agnostic, computationally efficient, and supports rank adaptivity to automatically compress the network at hand. Extensive experiments across standard architectures, datasets, and adversarial attacks show the regularized networks can achieve over 94% compression while recovering or improving adversarial accuracy relative to uncompressed baselines. The code is openly available at `https://github.com/ScSteffen/Publication_ICML_MOSS_Workshop`.

1. Introduction

Deep neural networks excel in computer vision and data processing, but their significant computational and memory requirements limit deployment in resource-constrained environments. While advances in models using data centers and specialized hardware are notable, deploying accurate models on low-power platforms, e.g., unmanned aerial vehicles (UAVs) or sensor arrays, present unique challenges due to limited power and compute resources.

Three interdependent challenges arise:

- **Compression:** Models must fit strict memory, compute, and energy limits.
- **Accuracy:** Compressed models must preserve high performance for critical decisions.
- **Robustness:** Models need resilience against noise and adversarial perturbations.

Recent findings indicate these objectives conflict. Low-rank (Schotthöfer et al., 2022) and sparsity techniques (Guo et al., 2016) can reduce accuracy. Compressed networks may also show heightened sensitivity to adversarial attacks (Savostianova et al., 2023). Adversarial robustness methods such as data augmentation (Lee et al., 2017) and regularization (Zhang et al., 2019) often decrease clean accuracy, and many robustness enhancements add computational burdens during training (Su et al., 2023; Cheng et al., 2022) and inference (Cisse et al., 2017; Hein & Andriushchenko, 2017; Liu et al., 2010), complicating deployment on restricted hardware.

Our Contribution. We summarize our contributions as follows:

- **Low-rank compression framework.** We propose a regularizer and training framework for low-rank compressed networks, achieving over $10\times$ reductions in memory and compute costs while maintaining competitive accuracy and robustness.
- **Theoretical guarantees.** We derive an explicit bound on the condition number κ as a function of the regularizer, enhancing confidence in adversarial performance.
- **Preservation of performance.** We analytically and empirically demonstrate that our regularizer does not degrade training performance or clean validation accuracy across various architectures.

[1]Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA 37831 [2]Geospatial Science and Human Security Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA 37831. Correspondence to: Steffen Schotthöfer <schotthofers@ornl.gov>, H. Lexie Yang <yangh@ornl.gov>, Stefan Schnake <schnakesr@ornl.gov>.

Accepted at Methods and Opportunities at Small Scale (MOSS), ICML 2025, Vancouver, Canada.

Our model-agnostic approach seamlessly integrates with existing adversarial defenses and maintains a low memory footprint. By linking to dynamical low-rank integration schemes, we provide theoretical and algorithmic insights for the method. Finally, interpretable spectral metrics improve the trustworthiness of the compressed models.

2. Controlling adversarial robustness through the singular spectrum

We define a neural network f as a sequence of L layers: $z^{\ell+1} = \sigma^\ell(W^\ell z^\ell)$ with weight matrices $W^\ell \in \mathbb{R}^{n\times n}$, input $z^\ell \in \mathbb{R}^{b\times n}$, and nonlinear activations σ^ℓ. To simplify the presentation, we exclude biases, though they are included in experiments. The input data X serves as the first layer input, $z^0 = X$. We assume Lipschitz continuity for activations σ^ℓ. The network is trained on a locally bounded loss function $\mathcal{L}$ with a locally Lipschitz gradient. We refer to this standard architecture as a "baseline" network. While we focus on linear layers for the presentation, our method can extend to CNNs via Tucker decomposition; see (Zangrando et al., 2024a).

Low-rank Compression: Network compression during training and inference is achieved by approximating weight matrices as low-rank factorizations $W^\ell = U^\ell S^\ell V^{\ell,\top}$, where $U^\ell, V^\ell \in \mathbb{R}^{n\times r}$ and $S^\ell \in \mathbb{R}^{r\times r}$. We assume orthonormality for U^ℓ and V^ℓ during training and inference. This assumption diverges from typical low-rank training methods such as LoRA (Hu et al., 2021), but provide better geometric properties (Zhang et al., 2023; Schotthöfer et al., 2022; 2025). For $r \ll n$, this low-rank factorization is computationally efficient at $\mathcal{O}(2nr + r^2)$ versus $\mathcal{O}(n^2)$ for standard formats.

Adversarial Robustness: We measure the adversarial robustness of the network f using its relative sensitivity $\mathcal{S}$ to input perturbations δ given by:

$$\mathcal{S}(f, X, \delta) := \frac{||f(X+\delta) - f(X)||}{||f(X)||} \frac{||X||}{||\delta||}. \tag{1}$$

We focus on ℓ^2 norm sensitivity where $\|\cdot\| = \|\cdot\|_2$. For networks with Lipschitz continuous activations, there holds the bound (Savostianova et al., 2023):

$$\mathcal{S}(f, X, \delta) \leq \left(\prod_{\ell=1}^{L} \kappa(W^\ell)\right)\left(\prod_{\ell=1}^{L} \kappa(\sigma^\ell)\right) \tag{2}$$

where $\kappa(W) := \|W\| \left\|W^\dagger\right\|$ is the condition number of matrix W and $\kappa(\sigma^\ell)$ for activation functions. This allows us to consider layer-wise conditioning, omitting the superscript for brevity.

Sensitivity for low-rank networks can be analyzed using the orthonormal properties of U and V to show $\kappa(USV^\top) = \kappa(S)$. Thus, we only need to manage the condition number $\kappa(S)$, derivable via the singular value decomposition (SVD) of S, which is feasible to compute for $r \ll n$.

Adversarial Robustness-Aware Low-Rank Training: Enhancing adversarial robustness during low-rank training focuses on controlling the conditioning of S, which can be challenging. DLRT schemes (Schotthöfer et al., 2022) often exhibit ill-conditioned behavior, as shown in Figure 1, where a rank 64 factorization yields singular values ranging from $\sigma_{r=1} = 2.7785$ to $\sigma_{r=64} = 0.8210$, resulting in $\kappa(S) = 3.3844$. Comparatively, the baseline network achieves $\kappa(S) = 1.9722$ with singular values $\sigma_{r=1} = 1.8627$ to $\sigma_{r=128} = 0.9445$. Consequently, an ℓ^2-FGSM attack with strength $\epsilon = 0.3$ lowers the baseline accuracy to 54.96% versus 43.39% for the low-rank network, as shown in Table 5.

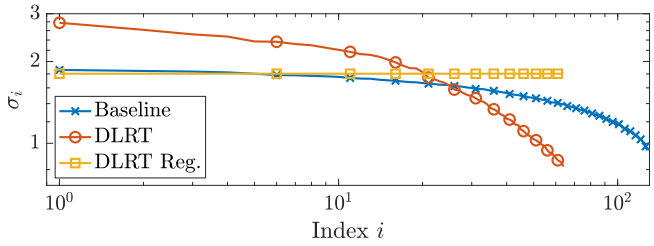

Figure 1: The singular values $\sigma_i(W)$ of sequential layer 7 in VGG16 for baseline training, DLRT, and RobustDLRT with $\beta = 0.075$. The matrix W is formed as the first-mode unfolding of the convolutional tensor. Conditioning of the regularized low-rank layer is significantly improved compared to the non-regularized low-rank and baseline layer. We note that the difference between the low-rank and regularized singular spectrum may be less pronounced for other layers and architectures.

3. Related work

Low-rank compression is a key method for reducing memory and computational costs in deep networks by constraining weights to low-rank subspaces. Early techniques utilized post-hoc matrix (Denton et al., 2014) and tensor decompositions (Lebedev et al., 2015), while recent methods integrate low-rank constraints during training for improved efficiency.

Dynamical Low-Rank Training (DLRT) (Schotthöfer et al., 2022) evolves network weights on a low-rank manifold, allowing significant reductions in memory and FLOPs without full-rank weight storage. This method extends to tensor-valued layers (Zangrando et al., 2024b) and federated learning (Schotthöfer & Laiu, 2024). In contrast, low-rank fine-tuning techniques like LoRA (Hu et al., 2021) introduce trainable low-rank updates into frozen pre-trained models, enabling efficient adaptation but failing to reduce overall training and inference costs, thereby limiting computational efficiency.

Improving adversarial robustness with orthogonal layers has gained attention in the literature. Soft methods impose orthogonality weakly via regularizers; examples include the soft orthogonal (SO) regularizer (Xie et al., 2017), double soft orthogonal regularizer (Bansal et al., 2018), mutual coherence regularizer (Bansal et al., 2018), and spectral normalization (Miyato et al., 2018). These approaches are flexible, enabling rank adaptation and transfer learning, but weak enforcement of orthogonality cannot provide rigorous spectral bounds.

Conversely, hard methods enforce orthogonality and well-conditioned constraints by training on a manifold using Riemannian optimization (Li et al., 2020; Absil & Malick, 2012). A notable hard approach for low-rank training clamped the spectrum extremes to bound the condition number during training (Savostianova et al., 2023). However, selecting the rank r as a hyperparameter is critical; if chosen incorrectly, the clamping may act as a strong regularizer and negatively impact validation metrics.

4. Improving conditioning via regularization

We design a computationally efficient regularizer $\mathcal{R}$ to control and decrease the condition number of each network layer during training. The regularizer $\mathcal{R}$ only acts on the small $r \times r$ coefficient matrices S of each layer and thus has a minimal memory and compute overhead over low-rank training. The regularizer is differentiable almost everywhere and compatible with automatic differentiation tools. Additionally, $\mathcal{R}$ has a closed form derivative that enables an efficient and scalable implementation of $\nabla\mathcal{R}$. Furthermore, $\mathcal{R}$ is compatible with any rank-adaptive low-rank training scheme that ensures orthogonality of U and V, e.g., (Zhang et al., 2023; Schotthöfer & Laiu, 2024).

Definition 1. *We define the robustness regularizer $\mathcal{R}$ for any $S \in \mathbb{R}^{r\times r}$ by*

$$\mathcal{R}(S) = \|S^\top S - \alpha_S^2 I\| \tag{3}$$

where $\alpha_S^2 = \frac{1}{r}\|S\|^2$ and $I = I_r$ is the $r \times r$ identity matrix.

The regularizer $\mathcal{R}$ can be viewed as an extension of the soft orthogonal regularizer (Xie et al., 2017; Bansal et al., 2018) where we penalize the distance of $S^\top S$ to the well-conditioned matrix $\alpha_S^2 I$. Here α_S is chosen such that $\|S\| = \|\alpha_S I\|$.

Proposition 1 (Properties of $\mathcal{R}$)**.** *(a) The gradient of $\mathcal{R}$ is given by*

$$\nabla\mathcal{R}(S) = 2S(S^\top S - \alpha_S^2 I)/\mathcal{R}(S). \tag{4}$$

(b) $\mathcal{R}$ is a scaled standard deviation of the set $\{\sigma_i(S)^2\}_{i=1}^r$; namely,

$$\frac{1}{r}\mathcal{R}(S)^2 = \frac{1}{r}\sum_{i=1}^{r}(\sigma_i(S)^2)^2 - \left(\frac{1}{r}\sum_{i=1}^{r}\sigma_i(S)^2\right)^2. \tag{5}$$

(c) $\mathcal{R}$ is a unitarily invariant regularizer; namely, $\mathcal{R}(USV^\top) = \mathcal{R}(S)$ for orthogonal $U, V \in \mathbb{R}^{n\times r}$.

(d) We have the condition number bound: for any $S \in \mathbb{R}^{r\times r}$ there holds

$$\kappa(S) \leq \exp\left(\frac{1}{\sqrt{2}\sigma_r(S)^2}\mathcal{R}(S)\right). \tag{6}$$

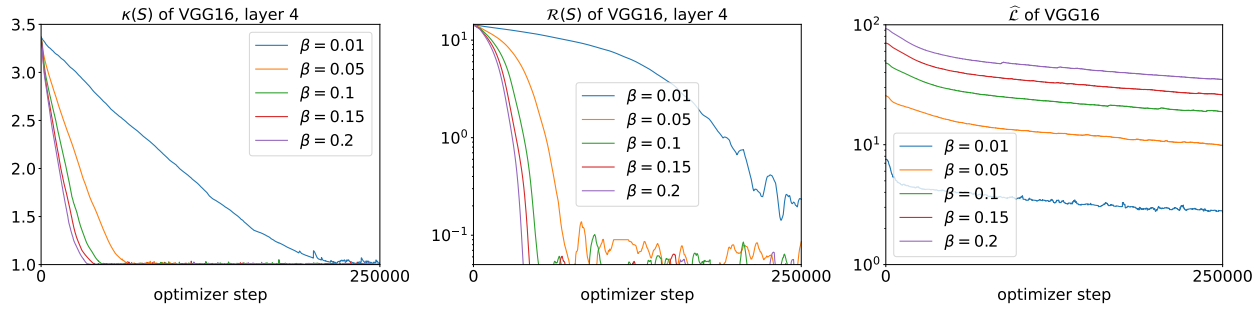

Figure 2: UCM Dataset, $\kappa(S(t))$ and $\mathcal{R}(S(t))$ of layer 4 of VGG16 for different regularizations strengths β. Each line is the median of 5 training runs. Higher β values lead to faster reduction of the layer condition $\kappa(S)$, which quickly approaches its minimum value 1, and faster decay of $\mathcal{R}$. Unregularized training, i.e. $\beta = 0$ leads to $\kappa(S) > 1e3$ after few iterations.

See Appendix D for the proofs. Figure 2(a) and (b) show the dynamics of $\mathcal{R}(S(t))$ and $\kappa(S(t))$ during regularized training. We see that $\kappa(S(t))$ decays as $\mathcal{R}(S(t))$ decays, validating Proposition 1(d).

When U and V are not orthonormal, e.g. in simultaneous gradient descent training (LoRA), Proposition 1(c) does not hold. Moreover, the smallest $n - r$ singular values of $USV^\top$ are zero-valued; thus the bound in Proposition 1(d) is not useful. Table 1 shows that the clean accuracy and adversarial accuracy of regularized LoRA is significantly lower than baseline training or regularized training with orthonormal U and V.

5. A rank-adaptive, adversarial robustness increasing dynamical low-rank training scheme

We integrate the regularizer $\mathcal{R}$ into a rank-adaptive, orthogonality preserving, and efficient low-rank training scheme. We are specifically interested in a training method that 1) enables separation of the spectral dynamics of the coefficients S from the bases U, V and 2) ensures orthogonality of U, V at all times during training to obtain control layer conditioning in a compute and memory efficient manner. These criteria are both achieved by the two-step DLRT scheme in (Schotthöfer & Laiu, 2024). The method dynamically modifies the rank of the factorized layers depending on the training dynamics and the complexity of the learning problem at hand. Consequently, the rank of each layer is no longer a hyper-parameter that needs fine-tuning, c.f. (Hu et al., 2021; Savostianova et al., 2023), but is rather an interpretable measure for the inherent complexity required for each layer.

Our method, which we call *RobustDLRT*, combines DLRT with our robustness regularizer $\mathcal{R}$ defined in Definition 1 by training on the regularized loss function $\widetilde{\mathcal{L}} = \mathcal{L} + \beta\mathcal{R}$ with regularization parameter $\beta > 0$. We refer the reader to Appendix A for specifics of the method.

6. Numerical Results

We evaluate the numerical performance of RobustDLRT compared with non-regularized low-rank training, baseline training, and several other robustness-enhancing methods on the VGG16 architectures and the Cifar 10 dataset. Further tests of different attacks and archetecture are given in Appendix C. We measure the compression rate (c.r.) as the relative amount of pruned parameters of the target network, i.e. c.r. $= (1 - \frac{\#\text{params low-rank net}}{\#\text{params baseline net}}) \times 100$. Detailed descriptions of the models, data-sets, pre-processing, training hyperparameters, and competitor methods are given in Appendix B.

Table 1: VGG16 on UCM data. Comparison of adversarial robustness of LoRA and DLRT trained networks under the ℓ^2-FGSM attack with similar compression rate (c.r.). Orthogonality of U and V increases adversarial performance significantly.

Method	**c.r. [%]**	clean Acc [%]	ℓ^2**-FGSM,** $\epsilon = 0.1$
RobustDLRT, $\beta = 0.0$	95.30	93.92	72.41
RobustDLRT, $\beta = 0.075$	95.84	94.61	78.68
LoRA, $\beta = 0.075$	95.83	88.57	73.81

Table 2: Comparison to literature on CIFAR10 with VGG16 under the ℓ^1-FGSM. The first three rows list the computed mean over 10 random initializations. The values of all other methods, given below the double rule, are taken from (Savostianova et al., 2023, Table 1). RobustDLRT has higher adversarial accuracy at higher compression rates than all listed methods.

		ℓ^1-FGSM, ϵ			
Method	**c.r. [%]**	0.0	0.002	0.004	0.006
RobustDLRT $\beta = 0.15$	94.35	89.35	**78.72**	**66.02**	**54.15**
DLRT	94.58	89.55	74.71	59.61	47.56
Baseline	0	89.83	78.61	64.66	53.71
Cayley SGD (Li et al., 2020)	0	89.62	74.46	58.16	45.29
Projected SGD (Absil & Malick, 2012)	0	89.70	74.55	58.32	45.74
CondLR (Savostianova et al., 2023) $\xi = 0.5$	50	89.97	72.25	60.19	50.17
CondLR (Savostianova et al., 2023) $\xi = 0.5$	80	89.33	68.23	48.54	36.66
LoRA (Hu et al., 2021)	50	89.97	67.71	48.86	38.49
LoRA (Hu et al., 2021)	80	88.10	64.24	42.66	29.90
SVD prune (Yang et al., 2020)	50	89.92	67.30	47.77	36.98
SVD prune (Yang et al., 2020)	80	87.99	63.57	42.06	29.27

Cifar10 dataset: We compare RobustDLRT with full rank training and non-regularized DLRT for different adversrial attacks in Table 5 and observe that Robust DRLT achieves high compression rates of 95% while maintaining the adversarial accuracy of the baseline training, sometimes surpassing it. Furthermore, we compare our method in Table 2 to several methods of the recent literature, see Section 3. We compare the adversarial accuracy under the ℓ^1-FGSM attack, see Appendix B.1.2 for details, for consistency with the literature results. We find that our proposed method achieves the highest adversarial validation accuracy for all attack strengths ϵ, even surpassing the baseline adversarial accuracy. Additionally, we find an at least 15% higher compression ratio with RobustDLRT than the second best compression method, CondLR (Savostianova et al., 2023).

7. Conclusion

RobustDLRT enables highly compressed neural networks with strong adversarial robustness by controlling the spectral properties of low-rank factors. The method is efficient and rank-adaptive that yields an over 94% parameter reduction across diverse models and attacks. The method achieves competitive accuracy, even for strong adversarial attacks, surpassing the current literature results by a significant margin. Therefore, we conclude the proposed method scores well in the combined metric of compression, accuracy and adversarial robustness.

Funding Acknowledgements

This material is based upon work supported by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL), managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. De-AC05-00OR22725.

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan(`http://energy.gov/downloads/doe-public-access-plan`).

This research used resources of the Compute and Data Environment for Science (CADES) at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

Absil, P.-A. and Malick, J. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22(1):135–158, 2012. doi: 10.1137/100802529. URL https://doi.org/10.1137/100802529.

Bansal, N., Chen, X., and Wang, Z. Can we gain more from orthogonality regularizations in training deep networks? *Advances in Neural Information Processing Systems*, 31, 2018.

Cheng, G., Sun, X., Li, K., Guo, L., and Han, J. Perturbation-seeking generative adversarial networks: A defense framework for remote sensing image scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–11, 2022. doi: 10.1109/TGRS.2021.3081421.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 854–863. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/cisse17a.html.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., and Fergus, R. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27, 2014.

Guo, Y., Yao, A., and Chen, Y. Dynamic network surgery for efficient dnns. *Advances in neural information processing systems*, 29, 2016.

Hein, M. and Andriushchenko, M. Formal guarantees on the robustness of a classifier against adversarial manipulation. *Advances in neural information processing systems*, 30, 2017.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017.

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., and Lempitsky, V. Speeding-up convolutional neural networks using fine-tuned CP-decomposition. In *International Conference on Learning Representations*, 2015.

Lee, H., Han, S., and Lee, J. Generative adversarial trainer: Defense to adversarial perturbations with GAN. *arXiv preprint arXiv:1705.03387*, 2017.

Li, J., Li, F., and Todorovic, S. Efficient Riemannian optimization on the Stiefel manifold via the Cayley transform. In *International Conference on Learning Representations*, 2020.

Lin, J., Song, C., He, K., Wang, L., and Hopcroft, J. E. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *International Conference on Learning Representations*, 2020.

Liu, X., Li, Y., Wu, C., and Hsieh, C.-J. Adv-BNN: Improved adversarial defense through robust Bayesian neural network. In *International Conference on Learning Representations*, 2010.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1QRgziT-.

Nagy, J. Über algebraische gleichungen mit lauter reellen wurzeln. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 27:37–43, 1918.

Nenov, R., Haider, D., and Balazs, P. (Almost) smooth sailing: Towards numerical stability of neural networks through differentiable regularization of the condition number, 2024. URL https://arxiv.org/abs/2410.00169.

Savostianova, D., Zangrando, E., Ceruti, G., and Tudisco, F. Robust low-rank training via approximate orthonormal constraints. *Advances in Neural Information Processing Systems*, 36:66064–66083, 2023.

Schotthöfer, S. and Laiu, M. P. Federated dynamical low-rank training with global loss convergence guarantees. *arXiv preprint arXiv:2406.17887*, 2024.

Schotthöfer, S., Zangrando, E., Jonas, K., Ceruti, G., and Tudisco, F. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations. In *Advances in Neural Information Processessing Systems*, 2022.

Schotthöfer, S., Zangrando, E., Ceruti, G., Tudisco, F., and Kusch, J. GeoLoRA: Geometric integration for parameter efficient fine-tuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=bsFWJ0Kget.

Schwinn, L., Raab, R., Nguyen, A., Zanca, D., and Eskofier, B. Exploring misclassifications of robust neural networks to enhance adversarial attacks, 2021. URL https://arxiv.org/abs/2105.10304.

Sharma, R., Gupta, M., and Kapoor, G. Some better bounds on the variance with applications. *Journal of Mathematical Inequalities*, 4(3):355–363, 2010.

Singh, S. P., Bachmann, G., and Hofmann, T. Analytic insights into structure and rank of neural network Hessian maps. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Su, Y., Zhang, G., Mei, S., Lian, J., Wang, Y., and Wan, S. Reconstruction-assisted and distance-optimized adversarial training: A defense framework for remote sensing scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–13, 2023. doi: 10.1109/TGRS.2023.3328889.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses, 2020. URL https://arxiv.org/abs/1705.07204.

Xie, D., Xiong, J., and Pu, S. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6176–6185, 2017.

Xu, Y. and Ghamisi, P. Universal adversarial examples in remote sensing: Methodology and benchmark. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–15, 2022. doi: 10.1109/TGRS.2022.3156392.

Yang, H., Tang, M., Wen, W., Yan, F., Hu, D., Li, A., Li, H., and Chen, Y. Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification, 2020. URL https://arxiv.org/abs/2004.09031.

Yang, Y. and Newsam, S. Bag-of-visual-words and spatial extensions for land-use classification. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, pp. 270–279, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450304283. doi: 10.1145/1869790.1869829. URL https://doi.org/10.1145/1869790.1869829.

Zangrando, E., Schotthöfer, S., Ceruti, G., Kusch, J., and Tudisco, F. Geometry-aware training of factorized layers in tensor tucker format. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 129743–129773. Curran Associates, Inc., 2024a. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ea48cb23a02ec3b895b0b0124307b0ec-Paper-Conference.pdf.

Zangrando, E., Schotthöfer, S., Ceruti, G., Kusch, J., and Tudisco, F. Rank-adaptive spectral pruning of convolutional layers during training. In *Advances in Neural Information Processing Systems*, 2024b.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pp. 7472–7482. PMLR, 2019.

Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., and Zhao, T. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

A. Robust DLRT Algorithm

Below we summarize the proposed method, called *RobustDLRT*, along with the changes induced by adding our robustness regularizer.

Basis Augmentation: The method first augments the current bases U^t, V^t at optimization step t by their gradient dynamics $\nabla_U \mathcal{L}, \nabla_V \mathcal{L}$ via

$$\begin{aligned} \widehat{U} &= \texttt{orth}([U^t \mid \nabla_U \mathcal{L}(U^t S^t V^{t,\top})]) \in \mathbb{R}^{n \times 2r}, \\ \widehat{V} &= \texttt{orth}([V^t \mid \nabla_V \mathcal{L}(U^t S^t V^{t,\top})]) \in \mathbb{R}^{n \times 2r}, \end{aligned} \tag{7}$$

to double the rank of the low-rank representation and subsequently creates orthonormal bases $\widehat{U}, \widehat{V}$. Since $\mathcal{R}(USV^\top) = \mathcal{R}(S)$, $\nabla_U \mathcal{R}(USV^\top) = \nabla_V \mathcal{R}(USV^\top) = 0$; hence $\nabla_U \widetilde{\mathcal{L}} = \nabla_U \mathcal{L}$ and $\nabla_V \widetilde{\mathcal{L}} = \nabla_V \mathcal{L}$ are used in (7). The span on $\widehat{U}$ contains U, which is needed to ensure of the loss does not increase during augmentation, and a first-order approximation of the basis for W^{t+1} using the exact gradient flow for U, see (Schotthöfer & Laiu, 2024, Theorem 2) for details. Geometrically, the latent space

$$\mathcal{S} = \{\widehat{U} Z \widehat{V}^\top : Z \in \mathbb{R}^{2r \times 2r}\} \tag{8}$$

can be seen as subspace[1] of the tangent plane of $\mathcal{M}_r$ at $U^t S^t V^{t,\top}$, see Figure 3.

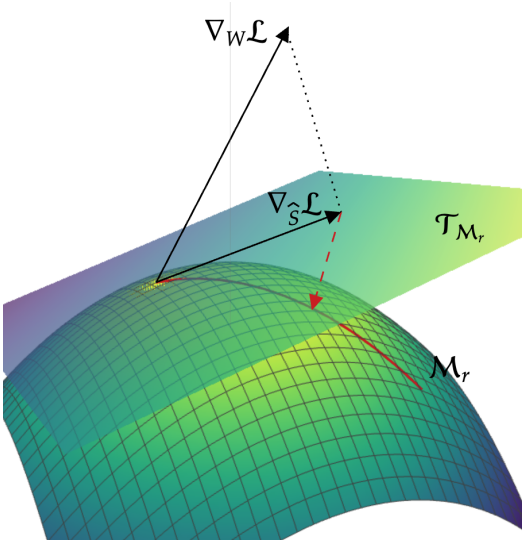

Figure 3: Geometric interpretation of Robust-DLRT . First, we compute the parametrization of the tangent plane $\mathcal{T}_{\mathcal{M}_r}$. Then we compute the projected gradient update with $\nabla_{\widehat{S}} \mathcal{L}$. Lastly, we retract the updated coefficients back onto the manifold $\mathcal{M}_r$. The regularizer $\mathcal{R}$ effectively changes the local curvature of $\mathcal{M}_r$.

Latent Space Training: We update the latent coefficients $\widehat{S}$ via a Galerkin projection of the training dynamics onto the latent space $\mathcal{S}$. The latent coefficients $\widehat{S}$ are updated by integrating the projected gradient flow $\dot{\widehat{S}} = -\widehat{U}^\top \nabla_W \widetilde{\mathcal{L}} \widehat{V} = -\nabla_{\widehat{S}} \widetilde{\mathcal{L}}$ using stochastic gradient descent or an other suitable optimizer for a number of s_* local iterations, i.e.

$$\widehat{S}_{s+1} = \widehat{S}_s - \lambda \nabla_{\widehat{S}} \mathcal{L} - \beta \nabla_{\widehat{S}} \mathcal{R}(\widehat{S}_s), \tag{9}$$

Equation (9) is initialized with $\widehat{S}_0 = \widehat{U}^\top U^t S^t V^{t,\top} \widehat{V} \in \mathbb{R}^{2r \times 2r}$, and we set $\tilde{S} = \hat{S}_{s_*}$

Truncation: Finally, the latent solution $\widehat{U} \tilde{S} \widehat{V}^\top$ is retracted back onto the manifold $\mathcal{M}_r$. The retraction can be computed efficiently by using a truncated SVD of $\tilde{S}$ that discards the smallest r singular values. To enable rank adaptivity, the new rank r_1 instead of r can be chosen by a variety of criteria, e.g., a singular value threshold $\|[\sigma_{r_1}, \ldots, \sigma_{2r}]\|_2 < \vartheta$. Once a suitable rank is determined, the bases U and V are updated by discarding the basis vectors corresponding to the truncated singular values.

Computational cost: The computational cost of the above described scheme is asymptotically the same as LoRA, since the reconstruction of the full weight matrix W is never required. The orthonormalization accounts for $\mathcal{O}(nr^2)$, the regularizer $\mathcal{R}$ for $\mathcal{O}(r^3)$, and the SVD for $\mathcal{O}(r^3)$ floating point operations. When using multiple coefficient update steps $s_* > 1$, the amortized cost is indeed lower than that of LoRA, since only the gradient with respect to $\widehat{S}$ is required in most updates.

B. Details to the numerical experiments of this work

B.1. Recap of adversarial attacks

In the following we provide the defintions of the used adversarial attacks. We use the implementation of `https://github.com/YonghaoXu/UAE-RS` for the ℓ^2-FGSM, Jitter, and Mixup attack. For the ℓ^1-FGSM attack, we use the implementation of `https://github.com/COMPiLELab/CondLR`.

[1]Technically the latent space contains extra elements not in the tangent space, but this extra information only helps the approximation.

B.1.1. ℓ^2-FGSM ATTACK

The Fast Gradient Sign Method (FGSM)(Kurakin et al., 2017) is a single-step adversarial attack that perturbs an input in the direction of the gradient of the loss with respect to the input. Given a neural network classifier f_θ with parameters θ, an input x, and its corresponding label y, the attack optimizes the cross-entropy loss $\mathcal{L}_{\text{CE}}(f_\theta(x), y)$ by modifying x along the gradient's sign. The adversarial example is computed as:

$$x' = x + \alpha \cdot \frac{\nabla_x \mathcal{L}_{\text{CE}}(f_\theta(x), y)}{\|\nabla_x \mathcal{L}_{\text{CE}}(f_\theta(x), y)\|_\infty}, \tag{10}$$

where α controls the perturbation magnitude. To ensure the perturbation remains bounded, the difference $x' - x$ is clamped by an ϵ bound, i.e.,

$$x' = x + \max(-\epsilon, \min(x' - x, \epsilon)). \tag{11}$$

In this work we fix $\alpha = \epsilon$. The attack can be iterated to increase its strength.

B.1.2. ℓ^1-FGSM ATTACK

The ℓ^1-FGSM attack (Tramèr et al., 2020) is used in the reference work of (Savostianova et al., 2023) and uses the same workflow as (B.1.1), where (10) is changed to

$$x' = x + \alpha \cdot \frac{\text{sign}(\nabla_x \mathcal{L}_{\text{CE}}(f_\theta(x), y))}{\Sigma}, \tag{12}$$

where Σ denotes the standard deviation of the data-points in the training data-set and the sign of the gradient matrix is taken element wise.

B.1.3. JITTER ATTACK

The Jitter attack (Schwinn et al., 2021) is an adversarial attack that perturbs an input by modifying the softmax-normalized output of the model with random noise before computing the loss. Given a neural network classifier f_θ with parameters θ, an input x, and its corresponding label y, the attack first computes the network output $z = f_\theta(x)$ and normalizes it using the ℓ^∞ norm:

$$\hat{z} = \text{Softmax}\left(\frac{s \cdot z}{\|z\|_\infty}\right), \tag{13}$$

where s is a scaling factor. A random noise term $\eta \sim \mathcal{N}(0, \sigma^2)$ is added to $\hat{z}$, i.e.,

$$\tilde{z} = \hat{z} + \sigma \cdot \eta. \tag{14}$$

The attack loss function is a mean squared error between perturbed input and target, given by

$$\mathcal{L} = \|\tilde{z} - y\|_2^2. \tag{15}$$

The adversarial example is then computed using the gradient of $\mathcal{L}$ with respect to x:

$$x' = x + \alpha \cdot \frac{\nabla_x \mathcal{L}}{\|\nabla_x \mathcal{L}\|_\infty}. \tag{16}$$

To ensure the perturbation remains bounded, the modification $x' - x$ is clamped within an ϵ bound:

$$x' = x + \max(-\epsilon, \min(x' - x, \epsilon)). \tag{17}$$

In this work, we fix $\alpha = \epsilon$ and set $\sigma = 0.1$. The Jitter attack can be performed iteratively. Then, for each but the first iteration k, the attack loss is normalized by the perturbation of the input image,

$$\mathcal{L} = \frac{\|\tilde{z} - y\|_2^2}{\|x - x'_k\|_\infty}, \qquad k > 0 \tag{18}$$

In this work, we use 5 iterations of the Jitter attack for each image.

Table 3: Training hyperparameters for the UCM and Cifar10 Benchmark. The first set hyperparameters apply to both DLRT and baseline training, and we train DLRT with the same hyperparameters as the full-rank baseline models. The second set of hyper-parameters is specific to DLRT. The DLRT hyperparameters are selected by an initial parameter sweep. We choose the DLRT truncation tolerance relative to the Frobenius norm of $\widehat{S}$, i.e. $\vartheta = \tau\|\widehat{S}\|_F$, as suggested in (Schotthöfer et al., 2022).

Hyperparameter	**VGG16**	**VGG11**	**ViT16-b**
Batch Size (UCM)	16	16	16
Batch Size (Cifar10)	128	128	128
Learning Rate	0.001	0.001	0.001
Number of Epochs	20	20	5
L2 regularization	0	0	0.001
Optimizer	Adam	Adam	Adam
DLRT rel. truncation tolerance τ	0.1	0.05	0.08
Coefficient Steps	10	10	10
Initial Rank	150	150	150

B.1.4. MIXUP ATTACK

The Mixup attack is an adversarial attack that generates adversarial samples that share similar feature representations with an given virtual example (Xu & Ghamisi, 2022). Inspired by the Mixup data augmentation technique, this attack aims to create adversarial examples that maintain characteristics of both the original sample and its adversarial counterpart. Given a neural network classifier f_θ with parameters θ, an input x, and its corresponding label y, the attack first computes a linear combination of cross-entropy and negative KL-divergence loss,

$$\mathcal{L}_{\text{mixup}} = \beta \sum_{k=1}^{5} \mathcal{L}_{\text{CE}}(f_\theta(\frac{x}{2^k}), y) - \mathcal{L}_{\text{KL}} \tag{19}$$

$$\delta = \alpha \cdot \frac{\nabla_x \mathcal{L}_{\text{CE}}(f_\theta(x), y)}{\|\nabla_x \mathcal{L}_{\text{CE}}(f_\theta(x), y)\|_\infty}. \tag{20}$$

Equation (19) features a scale invariance attack applied to the loss (Lin et al., 2020, Section 3.3).

The final adversarial example is computed as a convex combination of the original input and its perturbed version:

$$x' = \lambda x + (1 - \lambda)(x + \delta), \tag{21}$$

where $\lambda \sim \text{Beta}(\beta, \beta)$ is sampled from a Beta distribution with hyperparameter β, controlling the interpolation between clean and perturbed inputs. The perturbation is further constrained within an ϵ-ball to ensure bounded adversarial modifications:

$$x' = x + \max(-\epsilon, \min(x' - x, \epsilon)). \tag{22}$$

In this work, we fix $\alpha = 1$ and set $\beta = 10^{-3}$. The attack can be iterated to increase its effectiveness, refining the adversarial perturbation at each step. We use 5 iterations of the Mixup Attack for each image.

B.2. Network architecture and training details

In this paper, we use the pytorch implementation and take pretrained weights from the imagenet1k dataset as initialization. The data-loaded randomly samples a batch for each batch-update which is the only source of randomness in our training setup.

- VGG16 is a deep convolutional neural network architecture that consists of 16 layers, including 13 convolutional layers and 3 fully connected layers.
- VGG11 is a convolutional neural network architecture similar to VGG16 but with fewer layers, consisting of 11 layers: 8 convolutional layers and 3 fully connected layers. It follows the same design principle as VGG16, using small 3×3 convolution filters and 2×2 max-pooling layers.

Table 4: Overview of the β for best performing regularization strength for RobustDLRTof Table 5.

	UCM Dataset			Cifar10 Dataset		
Architecture	FGSM	Jitter	Mixup	FGSM	Jitter	Mixup
Vgg16	0.075	0.2	0.15	0.05	0.05	0.05
Vgg11	0.1	0.05	0.15	0.15	0.05	0.2
ViT-16b	0.1	0.15	0.15	0.01	0.01	0.05

- ViT-16b is a Vision Transformer with 16x16 patch size, a deep learning architecture that leverages transformer models for image classification tasks.

The full training setup is described in Table 3. We train DLRT with the same hyperparameters as the full-rank baseline models. It is known (Schotthöfer et al., 2025) that DLRT methods are robust w.r.t. common hyperparameters as learning rate, and batch-size, and initial rank. The truncation tolerance τ is chosen between 0.05 and 0.1 per an initial parameter study. These values are good default values, as per recent literature (Schotthöfer & Laiu, 2024; Singh et al., 2021). In general, there is a trade-off between target compression ratio and accuracy, as illustrated e.g. in (Schotthöfer et al., 2022) for matrix-valued and (Singh et al., 2021) for tensor-valued (CNN) layers.

B.3. UCM

The University of California, Merced (UCM) Land Use Dataset is a benchmark dataset in remote sensing and computer vision, introduced in (Yang & Newsam, 2010). It comprises 2,100 high-resolution aerial RGB images, each measuring 256×256 pixels, categorized into 21 land use classes with 100 images per class. The images were manually extracted from the USGS National Map Urban Area Imagery collection, covering various urban areas across the United States. The dataset contains images with spatial resolution approximately 0.3 meters per pixel (equivalent to 1 foot), providing detailed visual information suitable for fine-grained scene classification tasks.

We normalize the training and validation data with mean $[0.485, 0.456, 0.406]$ and standard deviation $[0.229, 0.224, 0.225]$ for the rgb image channels. The convolutional neural neural networks used in this work are applied to the original 256×256 image size. The vision transformer data-pipeline resizes the image to a resolution of 224×224 pixels. The adversarial attacks for this dataset are performed on the resized images.

The regularizer confidently increases the adversarial validation accuracy of the networks.

B.4. Cifar10

The Cifar10 dataset consists of 10 classes, with a total of 60000 rgb images with a resolution of 32×32 pixels.

We use standard data augmentation techniques. That is, for CIFAR10, we augment the training data set by a random horizontal flip of the image, followed by a normalization using mean $[0.4914, 0.4822, 0.4465]$ and std. dev. $[0.2470, 0.2435, 0.2616]$. The test data set is only normalized. The convolutional neural neural networks used in this work are applied to the original 32×32 image size. The vision transformer data-pipeline resizes the image to a resolution of 224×224 pixels. The adversarial attacks for this dataset are performed on the resized images.

B.5. Computational hardware

All experiments in this paper are computed using workstation GPUs. Each training run used a single GPU. Specifically, we have used 5 NVIDIA RTX A6000, 3 NVIDIA RTX 4090, and 8 NVIDIA A-100 80G.

The estimated time for one experimental run depends mainly on the data-set size and neural network architecture. For training, generation of adversarial examples and validation testing we estimate 30 minutes on one GPU for one run.

C. Extended Numerical Results

We evaluate the numerical performance of RobustDLRT compared with non-regularized low-rank training, baseline training, and several other robustness-enhancing methods on various datasets and network architectures. We measure the compression

Table 5: UCM and Cifar10 benchmark. Clean and adversarial accuracy means and std. devs. of the baseline and regularized low-rank networks for different architectures. We report the low-rank results for unregulated, $\beta = 0.0$, and the best performing β, given in Table 4. RobustDLRT is able to match or surpass baseline adversarial accuracy values at compression rates of up to 95% in most setups.

	UCM Data		Clean	Acc [%] for ℓ^2-**FGSM,** ϵ			Acc [%] for **Jitter,** ϵ		Acc [%] for **Mixup,** ϵ		
	Method	c.r. [%]	Acc. [%]	0.05	0.1	0.3	0.035	0.045	0.025	0.1	0.75
VGG16	Baseline	0.0	94.40±0.72	86.71±1.90	76.40±2.84	54.96±2.99	89.58±2.99	85.05±3.40	77.77±1.61	37.25±3.66	23.05±3.01
	DLRT	95.30	93.92±0.23	87.95±1.02	72.41±2.08	43.39±4.88	83.99±1.22	67.41±1.63	85.79±1.51	40.42±2.89	20.13±2.92
	RobustDLRT	95.84	94.61±0.35	89.12±1.33	78.68±2.30	53.30±3.14	88.33±1.20	79.81±0.93	90.33±0.90	70.12±3.08	47.31±2.78
VGG11	Baseline	0.0	94.23±0.71	89.93±1.33	78.66±2.46	39.45±2.98	90.25±1.66	85.24±1.90	83.10±1.47	40.34±4.88	22.01±3.21
	DLRT	94.89	93.70±0.71	86.58±1.22	67.55±2.16	28.92±2.65	83.90±1.36	63.41±1.39	87.15±1.18	40.17±4.96	14.18±3.78
	RobustDLRT	94.59	93.57±0.84	87.90±0.91	72.96±1.55	32.85±2.46	86.77±0.76	74.31±1.50	88.00±1.13	60.97±4.18	28.56±3.64
ViT-16b	Baseline	0.0	96.72±0.36	93.02±0.38	92.18±0.31	89.71±0.28	93.71±1.22	93.21±1.17	89.62±1.81	51.05±3.17	43.91±3.97
	DLRT	86.7	96.38±0.60	91.21±0.44	82.10±0.32	62.45±0.41	86.67±1.05	79.81±0.81	80.48±1.82	41.52±3.24	35.91±3.76
	RobustDLRT	87.9	96.41±0.67	92.57±0.34	85.67±0.41	69.94±0.42	91.03±0.86	84.19±1.39	87.33±1.81	46.39±2.75	40.76±3.88
	Cifar10 Data										
VGG16	Baseline	0.0	89.82±0.45	76.22±1.38	63.78±2.01	34.97±2.54	78.60±1.12	73.54±1.55	71.51±1.31	37.36±2.60	16.12±2.12
	DLRT	94.37	89.23±0.62	74.07±1.23	59.55±1.79	28.74±2.21	72.51±1.04	66.21±1.41	79.56±1.15	59.88±2.26	38.98±1.94
	RobustDLRT	94.18	89.49±0.58	76.04±1.18	62.08±1.69	32.77±2.04	75.53±0.98	69.93±1.22	87.62±1.07	84.80±2.01	81.26±2.15
VGG11	Baseline	0.0	88.34±0.49	75.89±1.42	64.21±1.96	31.76±2.45	74.96±1.09	68.59±1.63	74.77±1.26	40.88±2.58	08.95±1.98
	DLRT	95.13	88.13±0.56	72.02±1.34	55.83±1.92	21.59±2.16	66.98±1.05	58.57±1.55	79.42±1.08	47.95±2.18	22.92±1.77
	RobustDLRT	94.67	87.97±0.52	76.04±1.26	63.82±1.83	30.77±2.30	71.06±1.00	65.63±1.38	84.93±1.10	78.35±1.89	65.93±2.04
ViT-16b	Baseline	0.0	95.42±0.35	79.94±0.95	63.66±1.62	32.09±2.05	84.65±0.88	77.20±1.04	52.17±1.49	16.03±2.34	13.29±2.01
	DLRT	73.42	95.39±0.41	79.50±0.91	61.62±1.48	30.32±1.94	83.33±0.80	76.16±0.95	58.32±1.44	17.43±2.28	14.49±1.92
	RobustDLRT	75.21	94.66±0.38	82.03±0.88	69.29±1.43	38.05±1.99	87.97±0.75	83.03±0.91	74.49±1.32	27.80±2.11	18.34±1.87

rate (c.r.) as the relative amount of pruned parameters of the target network, i.e. c.r. $= (1 - \frac{\#\text{params low-rank net}}{\#\text{params baseline net}}) \times 100$. Detailed descriptions of the models, data-sets, pre-processing, training hyperparameters, and competitor methods are given in Appendix B. The reported numbers in the tables represent the average over 10 stochastic training runs. We observe in Table 5 that clean accuracy results exhibit a standard deviation of less than 0.8%; the standard deviation increases with the attack strength ϵ for all tests and methods. This observation holds true for all presented results, thus we omit the error bars in the other tables for the sake of readability.

We observe in Table 5 that RobustDLRT can compress the VGG11, VGG16 and ViT-16b networks equally well as the non-regularized low-rank compression and achieves the first goal of high compression values of up to 95% reduction of trainable parameters. Furthermore, the clean accuracy is similar to the non-compressed baseline architecture; thus, we achieve the second goal of (almost) loss-less compression. Noting the adversarial accuracy results under the ℓ^2-FGSM, Jitter, and Mixup attacks with various attack strengths ϵ, we observe that across all tests, the regularized low-rank network created by RobustDLRT significantly outperforms the non-regularized low-rank network. For the ℓ^2-FGSM attack, our method is able to recover the adversarial accuracy of the baseline network. For Mixup, the regularization almost doubles the baseline accuracy for VGG16. By targeting the condition number of the weights, which gives a bound on the *relative* growth of the loss w.r.t. the size of the input, we postulate that the large improvement could be attributed to the improved robustness against the scale invariance attack (Lin et al., 2020)[Section 3.3] included in Mixup. We refer the reader to Appendix B.1.4 for a precise definition of the Mixup attack featuring scale invariance. However, this hypothesis was not further explored and is delayed to a future work. Finally, we are able to recover half of the lost accuracy in the Jitter attack. Overall, we achieved the third goal of significantly increasing adversarial robustness of the compressed networks. We refer to Table 4 for the used values of β and to Appendix B.3 for extended numerical results for a grid of different β values. The Cifar10 results are similar to the UCM results across the test cases.

D. Proofs

For the following proofs, let

$$(A, B) = \text{trace}(B^\top A) = \sum_{ij} A_{ij} B_{ij}$$

be the Frobenius inner product that induces the norm $\|A\| = \sqrt{(A, A)}$. By the cyclic property of the trace, we have

$$(AB, CD) = (B, CDA^\top) = (C^\top AB, D). \tag{23}$$

for matrices A, B, C, and D of appropriate size.

Proof of Proposition 14. Given $S \in \mathbb{R}^{r\times r}$, the Fréchet derivative for $\mathcal{Q} = \mathcal{R}^2$ at S is a linear operator $Z \to \nabla\mathcal{Q}(S)[Z]$ for $Z \in \mathbb{R}^{r\times r}$. Denote $W_S = S^\top S - \alpha_S^2 I$ which is symmetric. Since $\mathcal{Q}$ is an inner product, we calculate $\nabla\mathcal{Q}(S)[Z]$ as

$$\begin{aligned}
\tfrac{1}{2}\nabla\mathcal{Q}(S)[Z] &= (W_S, Z^\top S + S^\top Z - \tfrac{2}{r}(S, Z)I) \\
&= (W_S, Z^\top S) + (W_S, S^\top Z) - \tfrac{2}{r}(S, Z)(W_S, I) \\
&= (SW_S^\top, Z) + (SW_S, Z) - \tfrac{2}{r}(S, Z)(W_S, I) \\
&= 2(S(S^\top S - \alpha_S^2 I), Z) - \tfrac{2}{r}(S, Z)(S^\top S - \alpha_S^2 I, I).
\end{aligned} \tag{24}$$

Note by definition of α_S^2,

$$(S^\top S - \alpha_S^2 I, I) = \|S\|^2 - \alpha_S^2 \|I\|^2 = 0. \tag{25}$$

Hence

$$\nabla\mathcal{Q}(S) = 4S(S^\top S - \alpha_S^2 I). \tag{26}$$

Since $\mathcal{R}^2 = \mathcal{Q}$, therefore

$$\nabla\mathcal{R}(S) = \tfrac{\nabla\mathcal{Q}(S)}{2\mathcal{R}(S)}. \tag{27}$$

The desired estimate follows. The proof is complete. □

Proof of Proposition 1(b). We calculate

$$\begin{aligned}
\mathcal{R}(S)^2 &= (S^\top S - \alpha_S^2 I, S^\top S - \alpha_S^2 I) = \|S^\top S\|^2 - 2\alpha_S^2 (S^\top S, I) + \alpha_S^4 (I, I) = \|S^\top S\|^2 - \tfrac{1}{r}\|S\|^4 \\
&= \sum_{i=1}^r \sigma_i(S^\top S)^2 - \frac{1}{r}\left(\sum_{i=1}^r \sigma_i(S)^2\right)^2 = r\left(\frac{1}{r}\sum_{i=1}^r \sigma_i(S^\top S)^2 - \left(\frac{1}{r}\sum_{i=1}^r \sigma_i(S)^2\right)^2\right).
\end{aligned} \tag{28}$$

Since $S^\top S$ is symmetric positive semi-definite, $\sigma_i(S^\top S) = \sigma_i(S)^2$. Applying this substitution yields Proposition 1(b). The proof is complete. □

Proof of Proposition 1(c). The result follows from Proposition 1(b). □

Proof of Proposition 1(d). From Proposition 1(b) and noting $\sigma_i(S^\top S) = \sigma_i(S)^2$, we obtain

$$\frac{1}{r}\mathcal{R}(S)^2 = \frac{1}{r}\sum_{i=1}^r \sigma_i(S^\top S)^2 - \left(\frac{1}{r}\sum_{i=1}^r \sigma_i(S^\top S)\right)^2. \tag{29}$$

From (29), $\frac{1}{r}\mathcal{R}(S)^2$ is the variance of the sequence $\{\sigma_i(S^\top S)\}_{i=1}^r$. The Von Szokefalvi Nagy inequality (Nagy, 1918) bounds the variance of a finite sequence of numbers below by the range of the sequence (see (Sharma et al., 2010)). Applied to (29), this yields

$$\frac{1}{r}\mathcal{R}(S)^2 \geq \frac{(\sigma_1(S^\top S) - \sigma_r(S^\top S))^2}{2r} = \frac{(\sigma_1(S)^2 - \sigma_r(S)^2)^2}{2r}. \tag{30}$$

Hence

$$\sqrt{2}\mathcal{R}(S) \geq \sigma_1(S)^2 - \sigma_r(S)^2. \tag{31}$$

An application of the Mean Value Theorem for logarithms (see (Nenov et al., 2024, Proof of Theorem 2.2)), gives

$$\ln(\kappa(S)) \leq \frac{\sigma_1(S)^2 - \sigma_r(S)^2}{2\sigma_r(S)^2}. \tag{32}$$

Combining (31) and (32) yields

$$\ln(\kappa(S)) \leq \frac{1}{\sqrt{2}\sigma_r(S)^2}\mathcal{R}(S), \tag{33}$$

which, after exponentiation, yields Proposition 1(d). The proof is complete. □