Supplementary Material for Learning Extremely Lightweight and Robust Model with Differentiable Constraints on Sparsity and Condition Number

Xian Wei[1], Yangyu Xu[2,3], Yanhui Huang[4], Hairong Lv[5], Hai Lan[2]
Mingsong Chen[1], and Xuan Tang[6⋆]

[1] MoE Engineering Research Center of Hardware/Software Co-design Technology and Application, East China Normal University, China
[2] Fujian Institute of Research on the Structure of Matter, Chinese Academy of Sciences, China
[3] University of Chinese Academy of Sciences, China
[4] Fuzhou University, China
[5] Tsinghua University, China
[6] School of Communication and Electronic Engineering, East China Normal University, China
xwei@sei.ecnu.edu.cn, xuyangyu20@mails.ucas.ac.cn, huangyanhuind@gmail.com, lvhairong@tsinghua.edu.cn, lanhai09@fjirsm.ac.cn,mschen@sei.ecnu.edu.cn, xtang@cee.ecnu.edu.cn

1 Proof of Relation Between Condition Number and Output Error

We recall the general linear transformation:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}, \tag{1}$$

with $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{W}$ being an input signal, the output response, and the corresponding linear transformation matrix. The corresponding norm of $\boldsymbol{W}$ measures how much the mapping induced by that matrix can stretch vectors,

$$\|\boldsymbol{W}\| = \max_{\boldsymbol{x}} \frac{\|\boldsymbol{W}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \Rightarrow \|\boldsymbol{W}\|\|\boldsymbol{x}\| \geq \|\boldsymbol{W}\boldsymbol{x}\| = \|\boldsymbol{y}\|, \tag{2}$$

⋆ Corresponding Author

It is also important to consider how much a matrix can shrink vectors. The reciprocal of the minimum stretching is the norm of the inverse,

$$\begin{aligned} \|\boldsymbol{W}^{-1}\| &= \max_{\mathbf{y}} \frac{\|\boldsymbol{W}^{-1}\boldsymbol{y}\|}{\|\boldsymbol{y}\|} \\ &= \frac{1}{\min\limits_{\mathbf{y}} \frac{\|\boldsymbol{y}\|}{\|\boldsymbol{W}^{-1}\boldsymbol{y}\|}} \\ &= \frac{1}{\min\limits_{\boldsymbol{x}} \frac{\|\boldsymbol{W}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}}, \end{aligned} \tag{3}$$

An definition of the condition number equivalent to *Definition 2* of submitted manuscript is

$$k(\boldsymbol{W}) = \|\boldsymbol{W}\|\|\boldsymbol{W}^{-1}\|. \tag{4}$$

It is usually used to measure the sensitivity of the output of a linear system to input changes or errors, i.e. robustness. In other words, robustness measures the noise immunity of a linear system.

Now, we consider the effect of adversarial noises added to the input signal on the output of the linear system. Let $\delta\boldsymbol{x}$ be the perturbations added to the input signal $\boldsymbol{x}$ and $\delta\boldsymbol{y}$ be the resulting error in output response $\boldsymbol{y}$,

$$\boldsymbol{y} + \delta\boldsymbol{y} = \boldsymbol{W}(\boldsymbol{x} + \delta\boldsymbol{x}), \tag{5}$$

considering Eq.(1), we get:

$$\begin{aligned} \delta\boldsymbol{y} &= \boldsymbol{W}\delta\boldsymbol{x} \\ \|\delta\boldsymbol{y}\| &= \|\boldsymbol{W}\delta\boldsymbol{x}\|, \end{aligned} \tag{6}$$

similar to Eq.(2),

$$\|\boldsymbol{W}\|\|\delta\boldsymbol{x}\| \geq \|\boldsymbol{W}\delta\boldsymbol{x}\| = \|\delta\boldsymbol{y}\|. \tag{7}$$

Combining Eq.(1) and Eq.(5),

$$\begin{aligned} \boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} &\Rightarrow \boldsymbol{x} = \boldsymbol{W}^{-1}\boldsymbol{y} \\ &\Rightarrow \boldsymbol{x} + \delta\boldsymbol{x} = \boldsymbol{W}^{-1}(\boldsymbol{y} + \delta\boldsymbol{y}), \end{aligned} \tag{8}$$

then we have

$$\begin{aligned} \|\boldsymbol{x}\| &= \|\boldsymbol{W}^{-1}\boldsymbol{y}\| \\ \delta\boldsymbol{x} &= \boldsymbol{W}^{-1}\delta\boldsymbol{y} \\ \|\delta\boldsymbol{x}\| &= \|\boldsymbol{W}^{-1}\delta\boldsymbol{y}\|. \end{aligned} \tag{9}$$

Similar to Eq. (2),

$$\|\boldsymbol{W}^{-1}\|\|\boldsymbol{y}\| \geq \|\boldsymbol{W}^{-1}\boldsymbol{y}\| = \|\boldsymbol{x}\|, \tag{10}$$

$$\|\boldsymbol{W}^{-1}\|\|\delta\boldsymbol{y}\| \geq \|\boldsymbol{W}^{-1}\delta\boldsymbol{y}\| = \|\delta\boldsymbol{x}\|, \tag{11}$$

by combining Eq.(2) and Eq.(11), we can finally derive a lower bound on the error $\delta \boldsymbol{y}$ of the output response affected by the input perturbation $\delta \boldsymbol{x}$,

$$\begin{aligned} \|\boldsymbol{W}\|\|\boldsymbol{W}^{-1}\|\|\delta \boldsymbol{y}\|\|\boldsymbol{x}\| &\geq \|\delta \boldsymbol{x}\|\|\boldsymbol{y}\| \\ \Rightarrow \frac{\|\delta \boldsymbol{y}\|}{\|\boldsymbol{y}\|} &\geq \frac{1}{k(\boldsymbol{W})} \frac{\|\delta \boldsymbol{x}\|}{\|\boldsymbol{x}\|}. \end{aligned} \tag{12}$$

Likewise, the corresponding upper bound can be obtained by combining Eq.(7) and Eq.(10),

$$\begin{aligned} \|\boldsymbol{W}\|\|\boldsymbol{W}^{-1}\|\|\boldsymbol{y}\|\|\delta \boldsymbol{x}\| &\geq \|\delta \boldsymbol{y}\|\|\boldsymbol{x}\| \\ \Rightarrow k(\boldsymbol{W}) \frac{\|\delta \boldsymbol{x}\|}{\|\boldsymbol{x}\|} &\geq \frac{\|\delta \boldsymbol{y}\|}{\|\boldsymbol{y}\|}. \end{aligned} \tag{13}$$

By combining Eq.(12) and Eq.(13), we finally get

$$\frac{1}{k(\boldsymbol{W})} \frac{\|\delta \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \frac{\|\delta \boldsymbol{y}\|}{\|\boldsymbol{y}\|} \leq k(\boldsymbol{W}) \frac{\|\delta \boldsymbol{x}\|}{\|\boldsymbol{x}\|}. \tag{14}$$

Therefore, improving the condition number $k(\boldsymbol{W})$ of weight space of the linear system will limit the variation of the corresponding output response, which can be seen from Eq.(14). That is, improving the condition number promotes the robustness of the system to adversarial noise.

2 An Example of Replacing Fully-connected Layers with Separable Linear Transformations

Generally, the 3D tensor (height, width, number of filters) output from convolutional layers need to be flattened before being fed into fully-connected layers. Differently, tensor products used in the work can operate directly on tensors without Flatten layers. It is worth noticing that the tensor product is adapted to the structures of input tensor without flattening, which is different from the work in [9]. The latter used tensor product of sub-matrices to approximate weight matricx of fully-connected layer after the flatten layer. These similar methods based on tensor decomposition are also shown in [1–3, 7], which are computationally expensive when dealing with high-dimensional input tensors.

If we read $\boldsymbol{W} = \boldsymbol{A}^{(1)} \otimes \boldsymbol{A}^{(2)}$ with $K_1 K_2 = d$ and $I_1 I_2 = m$, the number of parameters is reduced from $K_1 K_2 I_1 I_2$ to $K_1 I_1 + K_2 I_2$. The ratio of the number of parameters between two models is calculated by

$$\eta_1 = \frac{K_1 I_1 + K_2 I_2}{K_1 I_1 K_2 I_2} = \frac{1}{K_1 I_1} + \frac{1}{K_2 I_2}. \tag{15}$$

And the computational complexity is also significantly reduced from $\mathcal{O}(\boldsymbol{W}\boldsymbol{x}) = \mathcal{O}(K_1 \times K_2 \times I_1 \times I_2)$ to $\mathcal{O}(\boldsymbol{A}^{(2)} \boldsymbol{X} {\boldsymbol{A}^{(1)}}^{\top}) = \mathcal{O}(K_2 \times I_2 \times I_1 + K_2 \times I_1 \times K_1)$, and the ratio of them is computed as

$$\eta_2 = \frac{K_2 \times I_2 \times I_1 + K_2 \times I_1 \times K_1}{K_1 \times K_2 \times I_1 \times I_2} = \frac{1}{K_1} + \frac{1}{I_2}. \tag{16}$$

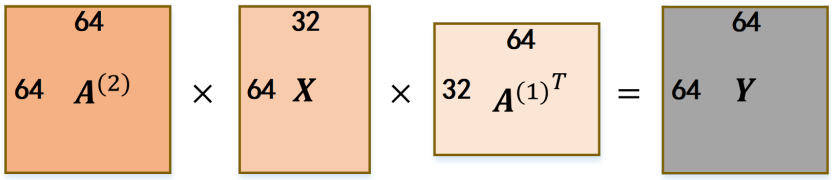

Fig. 1. Tensor product of separable sub-matrices replaces fully-connected layers.

For the convenience of referencing, we adopt the following fashion to name the algorithms in comparison: for example, the VGG-16 under the framework of *ARLST* algorithm is referred to as the *VGG*-16-*ARLST*. Taking VGG−16 trained on the CIFAR−10 dataset as an example, we follow its basic architecture and construct VGG−16-ARLST network. The size of 3D tensor before flatten is $2\times2\times512$. We then remove the flatten layer and reshape the 3D tensor as 64×32. Thus, each following FC layer could be constructed by two separable transformations, i.e., $T = 2$. For example, in the first fully-connected layer, VGG−16-ARLST replaces original linear transformation matrix $\boldsymbol{W} \in \mathbb{R}^{4096\times2048}$ with the tensor product of $\boldsymbol{A} \in \mathbb{R}^{64\times32}$ and $\boldsymbol{B} \in \mathbb{R}^{64\times64}$. An intuitive example is shown in Fig. 1. The first FC layers of VGG−16 have 8.39M parameters, while that of VGG−16-ARLST only has 0.006M parameters.

3 More Details of Eq.(15) in the Manuscript

Liu et al. [5] show that a family of regularizers are ineffective at penalizing the intrinsic norms of weights for networks with positively homogeneous activation functions, such as ReLU. For other activation functions, including Swish, GELU and SoftPlus, which are smooth approximation of Relu. Due to the positive homogeneity ($\sigma(ax) = a\sigma(x)$ when $a > 0$), we can get $\sigma(\tilde{\boldsymbol{W}}_l\boldsymbol{x}_{l-1} + \tilde{\boldsymbol{b}}_l) = \mu_l \cdot \sigma(\boldsymbol{W}_l\boldsymbol{x}_{l-1} + \boldsymbol{b}_l)$, where $\tilde{\boldsymbol{W}}_l = \mu_l\boldsymbol{W}_l$, $\tilde{\boldsymbol{b}}_l = \mu_l\boldsymbol{b}_l$. This operation can recurse from the first layer to the output, yielding an equivalent network under the condition that the $\prod_{l=1}^{L+1}\mu_l = 1$ holds,

$$\boldsymbol{y} = \left(\prod_{l=1}^{L+1}\mu_l\right) \cdot \tilde{\boldsymbol{W}}_l\boldsymbol{x}_{l-1}. \tag{17}$$

An equivalent network with completely different weights is obtained, but the final output is the same as in Eq.(3) of manuscript. In other words, the weight scale can be shifted between layers while keeping the network mapping function unchanged. However, these equivalent networks have different regularization penalties,

$$\sum_{l=1}^{L}\|\tilde{\boldsymbol{W}}_l\|_2^2 = \sum_{l=1}^{L}\mu_l^2\|\boldsymbol{W}_l\|_2^2 \neq \sum_{l=1}^{L}\|\boldsymbol{W}_l\|_2^2. \tag{18}$$

The result of Eq. (18) shows that even if the L_2 regularization works, the two networks are completely equivalent due to the invariance of the weight scale shift. Considering that there is only gradient flow during gradient back-propagation, the regularization is equivalent if the following formula holds,

$$\frac{\partial \tilde{\boldsymbol{W}}_l}{\partial w} = \frac{\partial(\mu_l \boldsymbol{W}_l)}{\partial w} = \frac{\partial \boldsymbol{W}_l}{\partial w}. \tag{19}$$

Obviously, $\phi(w) = log(w)$ allows the Eq. (19) to hold. Let $\{\sigma_i\}_{i=1}^{k}, k = \min\{a, b\}$ denote the singular values of a separable matrix $\mathbf{A}^{(t)} \in \mathbb{R}^{a \times b}$ arranged in descending order. Furthermore, $\mathbf{A}^{(t)}$ is expected to be full rank, and the Gram matrix ${\mathbf{A}^{(t)}}^{\top}\mathbf{A}^{(t)}$ is positive definite, which implies $\det({\mathbf{A}^{(t)}}^{\top}\mathbf{A}^{(t)}) > 0$. It is known that $\det({\mathbf{A}^{(t)}}^{\top}\mathbf{A}^{(t)}) = \prod \sigma_i^2$. Thus, we propose the following scale shift invariant condition number regularization term to prevent $\sigma_{\max}(\mathbf{A}^{(t)})$ from being too large or to avoid the worst case of $\sigma_{\min}(\mathbf{A}^{(t)})$ being exponentially small,

$$h(\mathcal{A}) = \frac{1}{4Tk\log(k)} \sum_{t=1}^{T} \left(\log \left[\nu + \frac{1}{k} \det({\mathbf{A}^{(t)}}^{\top}\mathbf{A}^{(t)}) \right] \right)^2 \tag{20}$$

with $0 < \nu \ll 1$ being a small smoothing parameter.

4 More Details and Experiments of ARLST with VGG-16

In the numerical experiment section of the manuscript, we incorporated the unstructured pruning algorithm into the **ARLST** framework for pruning of convolutional layers of VGG-16. More specifically, we only replaced the original fully-connected layer with Separable Linear Transformations (without sparsity constraint) at low-ratio compression. At high ratio compression, we pruned convolutional layers based on HYDRA [8], and further compressed Separable Linear Transformations by unstructured pruning after pre-training with sparsity constraint.

In the manuscript, we showed the gains up achieved by **ARLST** on CIFAR-10 dataset. And for SVHN dataset, this comprehensive advantage is even more obvious. As shown in Table. 1, also at 200×, the *ARLST* outperformed HYDRA, ADMM, ATMC by 11.6, 12.2, and 4.0 percentage points in robust accuracy, respectively.

5 More Experiments of ARLST with Vision Transformer

The original Vision Transformers (ViT) [4] has a large enough redundancy. We conducted a simple set of experiments based on ViT-B/16, and compared our method with HYDRA, ADMM, and ATMC on small datasets, such as Yale$-B$ and MNIST. In this experiment, we only replaced the fully-connected layer in Feed-Forward Network (FFN) with Separable Linear Transformations. The adversarial examples was generated by FGSM attack. We set the perturbation

Table 1. Comparison of our approach with other pruning-based baseline methods. We use CIFAR-10, SVHN and ImageNet dataset with VGG-16 networks, iterative adversarial training from [6] for this experiment.

	Method	HYDRA	ADMM	ATMC	ARLST
	CR	NA/RA	NA/RA	NA/RA	NA/RA
	10×	89.2/52.4	88.2/51.6	88.6/51.7	**89.6/52.9**
$SVHN$	20×	85.5/51.7	85.1/50.9	85.4/50.8	**87.5/52.8**
	100×	84.3/46.8	83.9/45.6	80.5/46.8	**84.6/51.6**
	200×	32.9/23.1	30.1/22.5	70.0/30.7	**73.1/34.7**

Table 2. Comparison of our approach with other pruning-based baseline methods. We use MNIST and Yale-B dataset and Vision Transformer for this experiment.

	Method	HYDRA	ADMM	ATMC	ARLST
	CR	NA/RA	NA/RA	NA/RA	NA/RA
	1×(pretrain)		97.98/90.94		
$MNIST$	3×	92.60/81.12	91.98/81.08	94.30/83.34	**94.65/83.96**
	1×(pretrain)		96.65/92.93		
$Yale-B$	3×	93.68/89.86	93.35/89.05	94.65/90.89	**95.83/92.74**

magnitude $\epsilon = 0.3$ for MNIST and $\epsilon = 0.03$ for Yale−B. All models are trained on MNIST dataset for 100 epochs and Yale−B dataset for 300 epochs.

As shown in Table. 2, our $ARLST$ achieves the best natural and robust accuracy on both datasets with the same compression ratio. Especially on the Yale-B dataset, our model has only one third of parameters of original ViT, but has almost no loss of accuracy. This is probably because we only reduced the number of parameters of FFN in the self-attention module, and hence the learning ability of compressed model does not decrease.

References

1. Budden, D., Matveev, A., Santurkar, S., Chaudhuri, S.R., Shavit, N.: Deep tensor convolution on multicores. In: International Conference on Machine Learning (ICML). pp. 615–624 (2017)
2. Chen, S., Sun, W., Huang, L., Yang, X., Huang, J.: Compressing fully connected layers using kronecker tensor decomposition. In: 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT). pp. 308–312. IEEE (2019)
3. Cohen, N., Sharir, O., Shashua, A.: On the expressive power of deep learning: A tensor analysis. In: Annual Conference on Learning Theory (COLT). pp. 698–728 (2016)
4. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (ICLR) (2020)

5. Liu, Z., Yufei, C., Chan, A.B.: Improve generalization and robustness of neural networks via weight scale shifting invariant regularizations. In: ICML 2021 Workshop on Adversarial Machine Learning (2021)
6. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: 6th International Conference on Learning Representations (ICLR). Vancouver, Canada (2018)
7. Novikov, A., Podoprikhin, D., Osokin, A., Vetrov, D.P.: Tensorizing neural networks. In: Advances in neural information processing systems (NeurIPS). pp. 442–450 (2015)
8. Sehwag, V., Wang, S., Mittal, P., Jana, S.: Hydra: Pruning adversarially robust neural networks. Advances in Neural Information Processing Systems (NeurIPS) **33**, 19655–19666 (2020)
9. Wu, J.N.: Compression of fully-connected layer in neural network by kronecker product. In: Eighth International Conference on Advanced Computational Intelligence (ICACI). pp. 173–179. IEEE (2016)