Unified Framework for Pre-trained Neural Network Compression via Decomposition and Optimized Rank Selection

Ali AGHABABAEI, Massih-Reza AMINI
Computer Science Laboratory (LIG), UGA, France

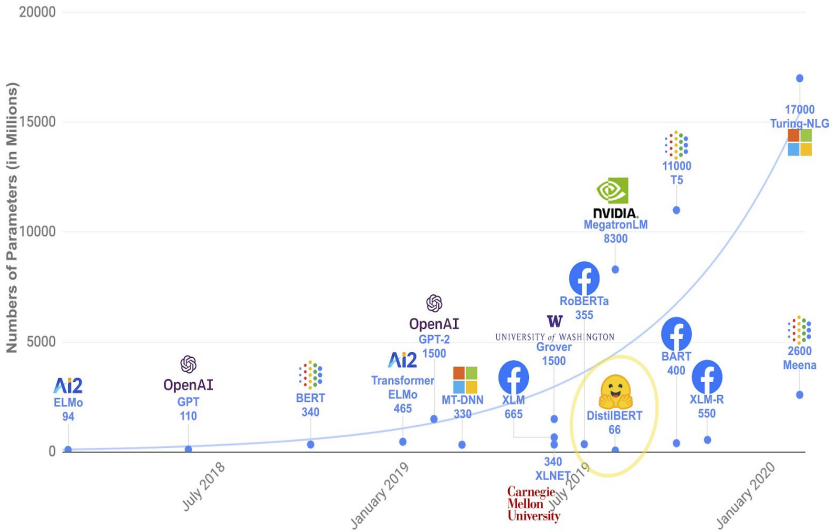
Mathematical Fundamentals of AI-2025

Overview

1. Motivation
2. Existing Compression Approaches
3. Method: Rank adapt tENsor dEcomposition
4. Results: Main and Ablation

1. Motivation

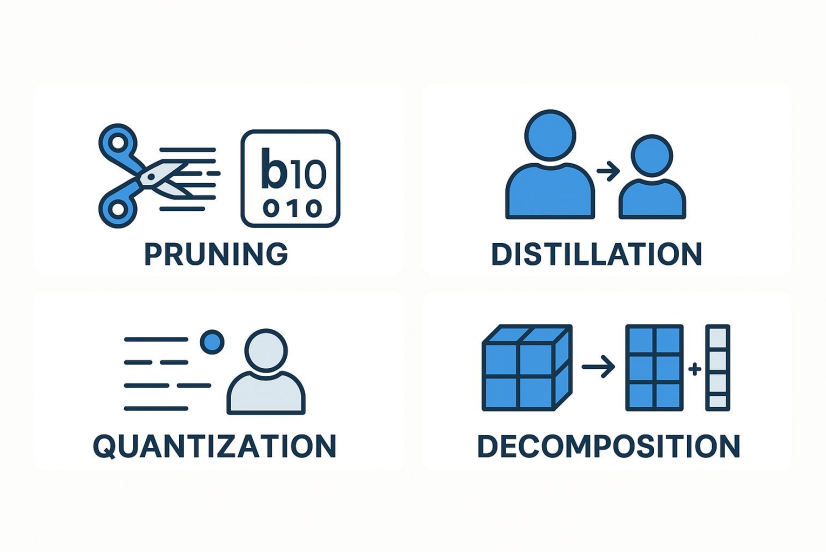
- Why compress pre-trained model?[1]
  - Computational cost
  - Energy consumption
  - Real time and embedded system



1. Patterson, D.; Gonzalez, J.; Le, Q.; Liang, C.; Munguia, L.M.; Rothchild, D.; So, D.; Texier, M.; and Dean, J. 2021. Carbon emissions and large neural network training. arXiv

2. Exciting Compression Approaches

- Pruning[2]
- Quantization[3]
- Distillation[4]
- Tensor Decomposition[5]



2. Hillar, C.J., Lim, L.H.: Most tensor problems are np-hard. JACM(2013)

3. Park, E., Ahn, J., Yoo, S.: Weighted-entropy-based quantization for deep neural networks. CVPR(2017)

4. Beyer, L., Zhai, X., Royer, A., Markeeva, L., Anil, R., Kolesnikov, A.: Knowledge distillation: A good teacher is patient and consistent. CVPR(2022)

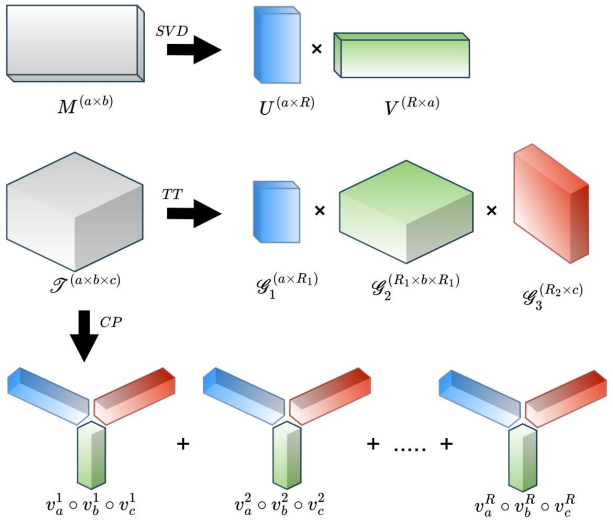
5. Yin, M., Phan, H., Zang, X., Liao, S., Yuan, B.: Batude: Budget-aware neural network compression based on tucker decomposition. AAAI(2022)

2. Tensor Decomposition For Compression[6]

$$\hat{\mathcal{W}}^{(R)}(i_1, i_2) = \sum_{r=1}^{R} \mathbf{u}(i, r)\sigma(r)\mathbf{v}(j, r) \quad = [\![\mathbf{V}, \Sigma, \mathbf{U}^T]\!]$$

$$\hat{\mathcal{W}}^{(R_1,\ldots,R_{N-1})}(i_1, i_2, \ldots, i_N) = \sum_{j_1=1}^{R_1} \cdots \sum_{j_{N-1}=1}^{R_{N-1}} \mathcal{G}_1(i_1, j_1)\mathcal{G}_2(j_1, i_2, j_2) \cdots \mathcal{G}_N(j_{N-1}, i_N)$$

$$\hat{\mathcal{W}}^{(R)}(i_1, i_2, \ldots, i_N) = \sum_{r=1}^{R} \mathbf{v}_1^{(r)}(i_1)\mathbf{v}_2^{(r)}(i_2) \cdots \mathbf{v}_N^{(r)}(i_N) \quad = [\![\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_N]\!]$$



6. Rabanser, S., Shchur, O., and Günnemann, S. 2017. Introduction to tensor decompositions and their applications in machine learning. arXiv.

3. Bottleneck of Tensor Decomposition

- Optimal rank selection of weights is NP-hard[7]
  - Manual easy but not optimal
- Proposed Automatic Search Rank adapt tENsor dEcomposition (RENE):
  - Find optimal rank
  - Keep up accuracy

$$\min_{\widehat{\mathcal{W}}^{\mathcal{R}}} \mathcal{L}_d(\widehat{\mathcal{W}}^{\mathcal{R}}) \quad \text{s.t.} \quad \min_{\mathcal{R}} \mathcal{L}_r(\mathcal{R})$$

Decomposition Loss

Rank Loss

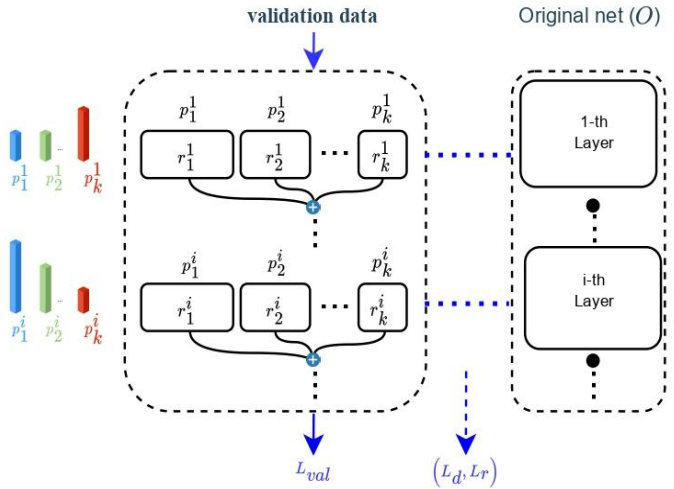
7. Hillar, C.J., Lim, L.H.: Most tensor problems are np-hard. JACM(2013)

3. Method: Continuous Rank Loss

- Define the rank loss $\mathcal{L}_r$
  - Replace each layer to Decomposition layers(DL)
  - Each DL is based on a pair (r, α_i^r)
  - How to connect?

$$p_i^{(r)} = \mathrm{softmax}(\alpha_i^{(r)})$$

$$\mathcal{L}_r(\mathcal{R}) = \gamma \sum_{i=1}^{n} \left(\sum_{r \in \mathcal{R}_i} p_i^{(r)} \frac{r}{\max \mathcal{R}_i} \right)^{\beta}$$



3. Method: Joint Optimization Strategy

- Total weight loss:

$$\mathcal{L}_{Tw}(\mathbf{W}^{\mathcal{R}}, \mathbf{p}^{\mathcal{R}}) = \sum_{i=1}^{n} \left\| \mathcal{W}_i - \sum_{r \in \mathcal{R}_i} p_i^{(r)} \hat{\mathcal{W}}_i^{(r)} \right\|_F^2 \times \gamma \left[\sum_{i=1}^{n} \left(\sum_{r \in \mathcal{R}_i} p_i^{(r)} \frac{r}{\max \mathcal{R}_i} \right)^{\beta} \right]$$

- Total parameters loss:

$$\mathcal{L}_{T\alpha}(\mathbf{W}^{\mathcal{R}}, \mathcal{P}^{\mathcal{R}}) = \mathcal{L}_{val}(\mathbf{W}^{\mathcal{R}}, \mathcal{P}^{\mathcal{R}}) \times \gamma \left[\sum_{i=1}^{n} \left(\sum_{r \in \mathcal{R}_i} p_i^{(r)} \frac{r}{\max \mathcal{R}_i} \right)^{\beta} \right]$$

- Update:

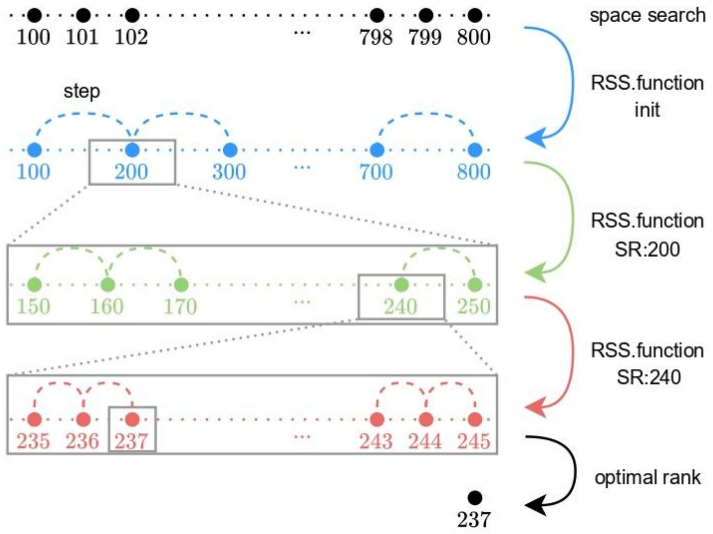
$$\text{Weight update:} \quad \hat{\mathbf{W}}_i^{(r)} \leftarrow \hat{\mathbf{W}}_i^{(r)} - \eta_w \nabla_{\hat{\mathbf{W}}_i^{(r)}} (\mathcal{L}_{Tw})$$

$$\text{Rank coefficient update:} \quad \alpha_i^{(r)} \leftarrow \alpha_i^{(r)} - \eta_\alpha \nabla_{\alpha_i^{(r)}} (\mathcal{L}_{T\alpha})$$

3. Method: Multi-Step Rank Search

- Motivation
  - Search all ranks

- Algorithm:
  - For each layer define a search space
  - Sample ranks with interval `s`
  - Search in sampled space
  - Pick `r` with highest `p`
  - Create new space
    - Around best ranks `[r-s/2,r+s/2]`
    - Sample with new interval `s←s/f`
  - Stop when `s=1`



3. Method: Fine-tuning

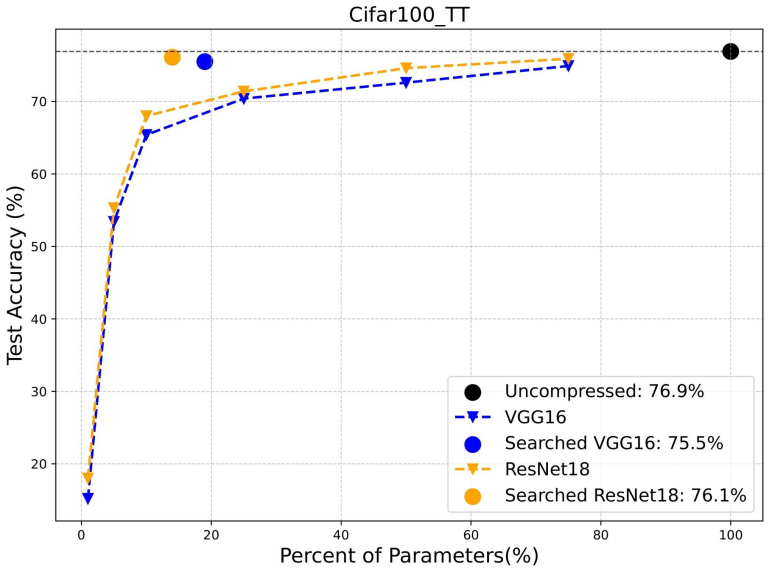
- Rebuild decomposed model with final Selected Ranks
- Fine-tuning based on formula:

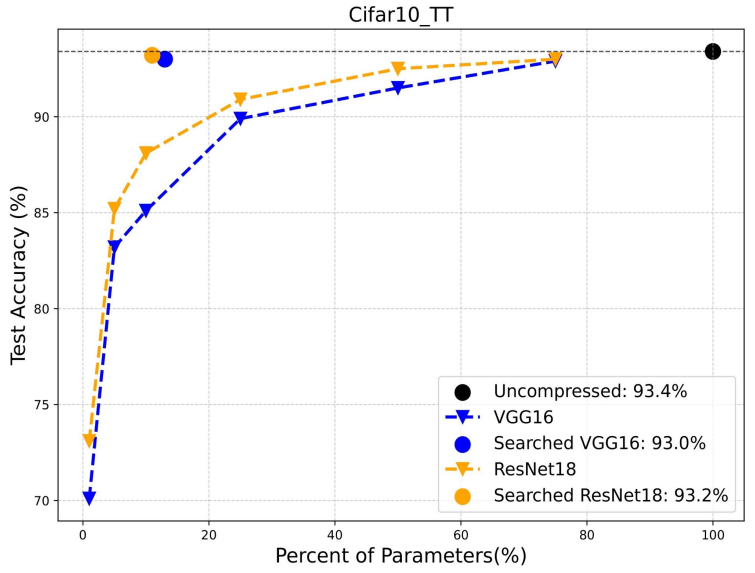
$$\mathcal{L}_{\text{diss}}(\mathbf{W}^{r^\star}) = \sum_{i=1}^{n} \left\| \mathbf{W}_i - \hat{\mathbf{W}}_i^{(r_i^\star)} \right\|_F^2 + \sum_{x \in X} \sum_{i=1}^{n} \| O_i(x) - D_i(x) \|_F^2$$

$$\mathcal{L}_f(\mathbf{W}^{r^\star}) = \mathcal{L}_{ce} + \lambda \mathcal{L}_{\text{diss}}$$

4. Result: Automatic Search vs Manual

Datasets: CIFAR10, CIFAR100 Pre-trained Models: VGG16, ResNet18





4. Result: CNN Model Compression

Table1: Pre-trained ResNet-20 and VGG-16 on CIFAR100

Method	C.T	A.R	Top-1	FLOPs (↓%)	Comp. Rate
ResNet-20	Original	-	91.25	-	-
RENE(CP)	Low-rank	✓	90.82	**73.44**	**77.62**
RENE(TT)	Low-rank	✓	**91.40**	70.4	72.28
HALOC [33]	Low-rank	✓	91.32	72.20	76.10
ALDS [21]	Low-rank	✓	90.92	67.86	74.91
LCNN [15]	Low-rank	✓	90.13	66.78	65.38
PSTR-S [20]	Low-rank	✓	90.80	65.00	60.87
Std. Tucker [16]	Low-rank	✗	87.41	62.00	61.54
VGG-16	Original	-	92.78	-	-
RENE(CP)	Low-rank	✓	92.51	86.23	**98.60**
RENE(TT)	Low-rank	✓	**93.20**	86.10	95.51
HALOC [33]	Low-rank	✓	93.16	**86.44**	98.56
ALDS [21]	Low-rank	✓	92.67	86.23	95.77
LCNN [15]	Low-rank	✓	92.72	85.47	91.14
DECORE [1]	Pruning	-	92.44	81.50	96.60
Spike-Thrift [18]	Pruning	-	91.79	80.00	97.01

Table2: Pre-trained ResNet-18 and MobileNetV2 on ImageNet-1K

Method	C.T	A.R	Top-1	FLOPs (↓%)	Comp. Rate
ResNet-18	Original	-	69.75	-	-
RENE(CP)	Low-rank	✓	68.46	57.1	66.2
RENE(TT)	Low-rank	✓	**70.88**	**68.9**	**67.1**
HALOC [33]	Low-rank	✓	70.65	66.16	63.64
ALDS [21]	Low-rank	✓	69.22	43.51	66.70
TETD [37]	Low-rank	✗	69.00	59.51	60.00
Stable EPC [25]	Low-rank	✓	68.50	59.51	61.00
MUSCO [12]	Low-rank	✗	69.29	58.67	60.50
CHEX [14]	Pruning	-	69.60	43.38	59.00
EE [39]	Pruning	-	68.27	46.60	58.00
SCOP [28]	Pruning	-	69.18	38.80	39.30
MobileNetV2	Original	-	71.85	-	-
RENE(CP)	Low-rank	✓	65.39	11.78	**51.6**
RENE(TT)	Low-rank	✓	70.1	**26.7**	42.34
HALOC [33]	Low-rank	✓	**70.98**	24.84	40.03
ALDS [21]	Low-rank	✓	70.32	11.01	32.97
HOSA [28]	Pruning	-	64.43	43.65	91.14
DCP [7]	Pruning	-	64.22	44.75	96.60
FT [40]	Pruning	-	70.12	20.23	21.31

** References in the tables can found on the our Paper: https://arxiv.org/abs/2409.03555*

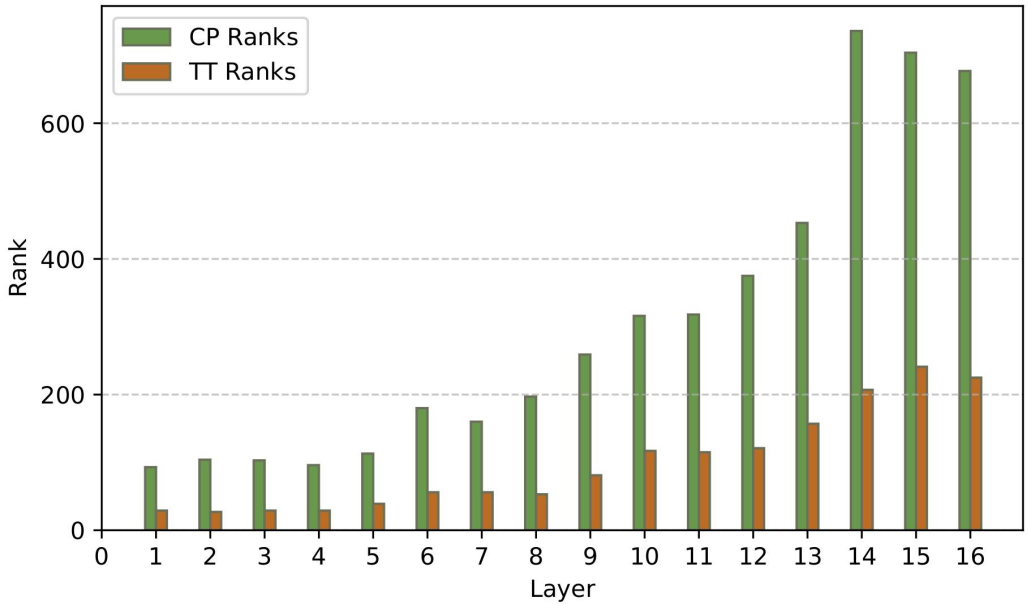
4. Result: ViT Model Compression

Table3: Pre-trained DeiT-small and DeiT-base on ImageNet-1k

Method	C.T	Top-1	FLOPs (↓%)	Comp. Rate
DeiT-small	Original	79.8	-	-
RENE$^{(*)}$	Low-rank	**79.93**	38.24	36.54
RENE$^{(**)}$	Low-rank	79.22	**43.91**	**39.8**
COMCAT [?]	Low-rank	79.92	41.15	40.11
UPOP [?]	Pruning	79.6	39	39
UVC [?]	Pruning	78.82	49.59	–
SCOP [?]	Pruning	77.5	43.6	–
S^{2}ViTE [?]	Sparse	79.22	31.63	33.94
ToMe [?]	Token	79.4	41.30	0
PS-ViT [?]	Token	79.4	43.5	0
HVT [?]	Token	78.0	47.8	0
PoWER [?]	Token	78.3	41.3	0
DeiT-base	Original	81.8	-	-
RENE$^{(*)}$	Low-rank	**81.94**	55.44	56.8
RENE$^{(**)}$	Low-rank	80.61	**64.62**	**61.3**
COMCAT [?]	Low-rank	82.26	61.68	61.06
CT-GFM [?]	Low-rank	81.28	–	40
MD-ViT [?]	Pruning	81.5	60	–
UVC [?]	Pruning	80.57	54.5	–
VTP [?]	Pruning	80.7	43.2	44.44
S^{2}ViTE [?]	Sparse	82.22	33.13	34.41
PS-ViT [?]	Token	81.5	44.3	0
IA-RED^2 [?]	Token	80.3	32.96	0

** References in the tables can found on the our Paper: https://arxiv.org/abs/2409.03555*

4. Result: Distribution of Ranks



Distribution of ranks achieved using CP and TT decompositions on ResNet-18 for the ImageNet-1K

** References in the tables can found on the our Paper: https://arxiv.org/abs/2409.03555*

4. Result: Double Compression

Upper Row: CIFAR100 , Teacher: ResNet-56, Student: ResNet-20

Bottom Row: ImageNet-1k , Teacher: ResNet-34, Student: ResNet-18

CIFAR-100 (T: ResNet56 (72.34%), S: ResNet20 (69.6%))			
Method	**Top-1 (%)**	**FLOPs (%)**	**Comp. rate (%)**
Distillation	**72.53**	67.7	68.24
RENE(Teacher)	72.23	64.23	61.75
RENE(Student)	72.46	**89.01**	**86.54**
ImageNet-1K (T: ResNet34 (73.31%), S: ResNet18 (69.76%))			
Method	**Top-1 (%)**	**FLOPs (%)**	**Params (%)**
Distillation	71.98	50.27	46.33
RENE(Teacher)	**73.23**	59.91	63.46
RENE(Student)	71.9	**76.77**	**78.69**

** Reference in the table can found on the our Paper: https://arxiv.org/abs/2409.03555*

Take-Home

Tensor Decomposition for Compression

- Tensor decomposition reduces model size and computational cost.
- The main challenge is choosing the Optimal Rank, since too small drop accuracy, too large wastes compression.

Unified Framework (RENE)

- Our method jointly performs tensor decomposition and automatic rank selection.
- It formulates the problem as minimizing a decomposition loss under rank constraints.

Efficient Rank Search

- RENE introduces a multi-step, continuous rank search strategy that progressively narrows the search space.
- This enables fast, data-free exploration of optimal ranks with lower computational overhead.

Results & Impact

- Across CIFAR-10/100 and ImageNet, RENE achieves **high compression rates** while maintaining..
- TT decomposition is effective for complex models (MobileNetV2), while CP works well for simple models (VGG-16).

Thank you for your attention

Questions?