Maestro: Uncovering Low-Rank Structures via Trainable Decomposition

Samuel Horváth♦, Stefanos Laskaridis♣, Shashank Rajput♡, Hongyi Wang♠

♦MBZUAI ♣Brave Software ♡Databricks ♠Carnegie Mellon University.

samuel.horvath@mbzuai.ac.ae, mail@stefanos.cc, rajput.shashank11@gmail.com, hongyiwa@andrew.cmu.edu

Efficient Deep Learning

- One of the **key challenges** of deploying deep learning models in practice is **the increasing costs of large-scale model training and deployment.**
- Popular remedy: **compress the network.**
- Techniques: i) quantization, ii) pruning, iii) **low-rank approximation.**
- **Main goals**: i) obtain a lower footprint model, ii) avoid the overhead during training time and the accuracy degradation, iii) avoid introducing mutliple hyperparameters and the need to fine-tune to recover the lost accuracy.

Trainable Decomposition

Low-rank approximation.

Best l-rank approximation:

$$\min_{U\in\mathbb{R}^{m\times l}, V\in\mathbb{R}^{n\times l}} \|\Sigma_{i=1}^{l} u_i v_i^\top - A\|_F^2 \quad (1)$$

The best rank approximation across all the ranks:

$$\min_{U\in\mathbb{R}^{m\times r}, V\in\mathbb{R}^{n\times r}} \frac{1}{r}\Sigma_{b=1}^{r}\|U_{:b}V_{:b}^\top - A\|_F^2, \quad (2)$$

Data-dependent low-rank approximation.

$$\min_{U\in\mathbb{R}^{m\times r}, V\in\mathbb{R}^{n\times r}} \mathbf{E}_{x,y\sim\mathcal{X}}\left[\Sigma_{b=1}^{r}\frac{1}{r}\|U_{:b}V_{:b}^\top x - y^2\|\right]. \quad (3)$$

This formulation is the first building block of **Maestro**, where we propose to use **double sampling** to solve this objective, by sampling data, i.e., $x, y \sim \mathcal{D}$, and rank, i.e., $b \sim \mathcal{D}(\{1, 2, \ldots, r\})$.

DNN low-rank approximation. We seek to uncover the optimal ranks for a set of d linear mappings $W^1 \in \mathbb{R}^{m_1\times n_1}, \ldots, W^d \in \mathbb{R}^{m_d\times n_d}$, where W^i's are model parameters, e.g., linear layer weights (also transformers or convolutions), by decomposing them as $W^i = U^i(V^i)^\top$. To adapt double sampling and preserve structure, we sample **one layer and its rank** in each step.

Rank extraction via hierarchical group-lasso.

$$\lambda_{gl}\Sigma_{i=1}^{d}\Sigma_{b=1}^{r_i}(\|U_{b:}^i\| + \|V_{b:}^i\|), \quad (4)$$

This penalty encourages that **unimportant ranks become zero** and **can be effectively removed from the model.** For each layer we remove $V_{b:}^i$ and $U_{b:}^i$ if $\|V_{b:}^i\|\|U_{b:}^i\| \le \epsilon_{ps}$ (numerical zero, e.g., $\epsilon_{ps} = 1e-5$).

Maestro as SVD

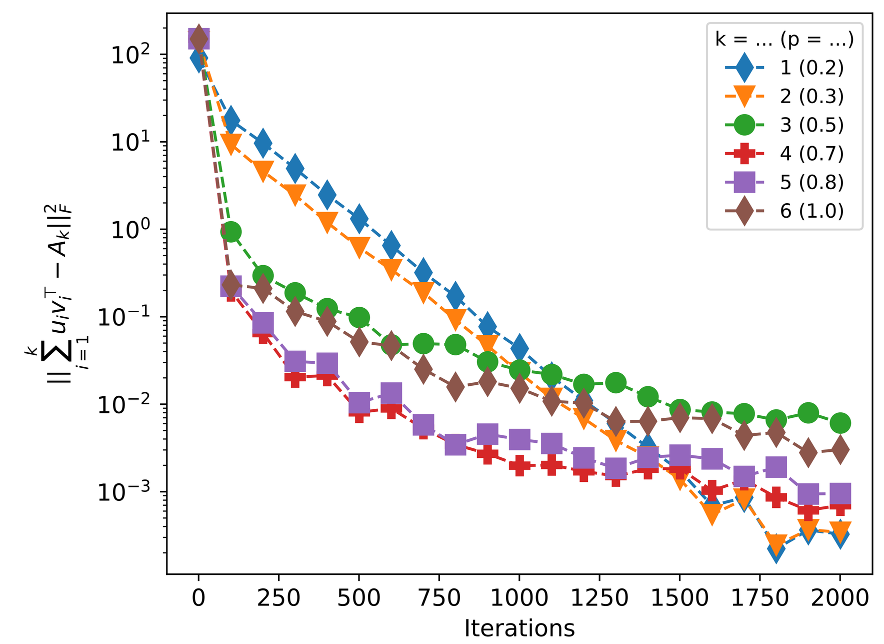


Figure 1: L2 distance between the best rank k and MAESTRO's approximation of mapping A.

If data are uniform and $y = Ax$, then **Maestro recovers SVD of A.**

Maestro as PCA

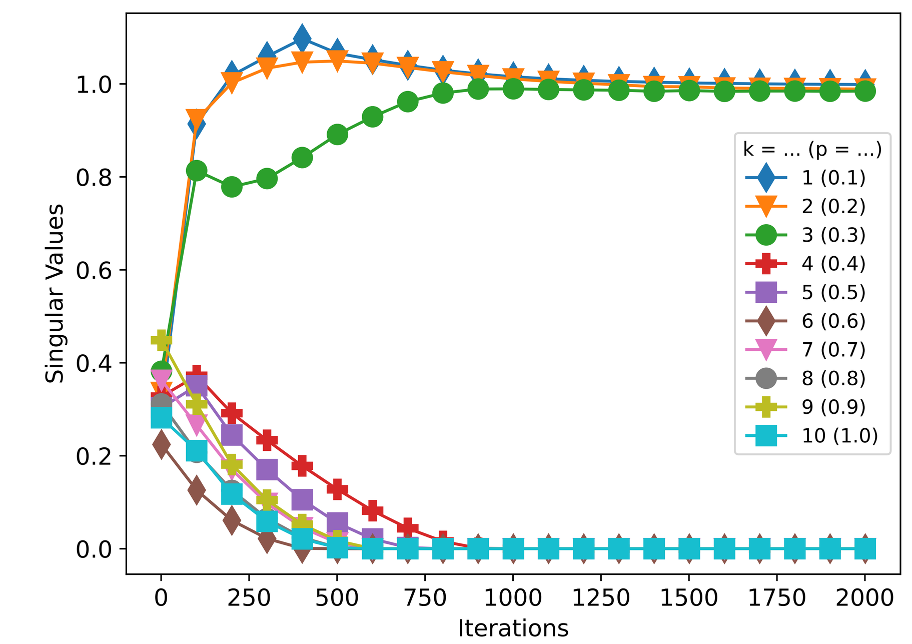
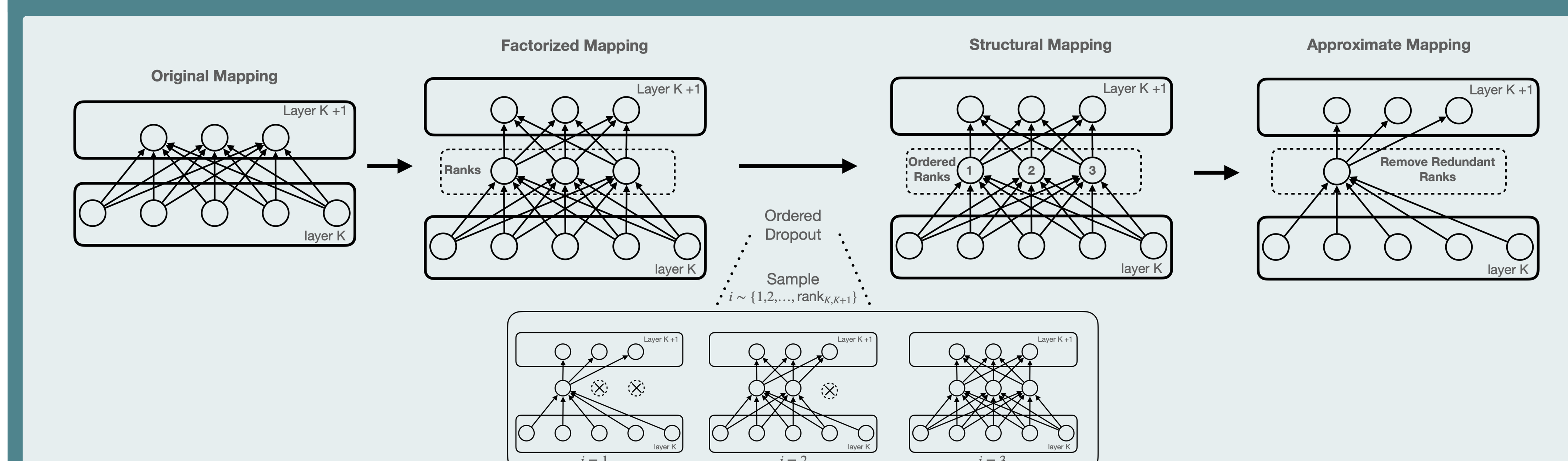


Figure 2: The plot displays the estimates of singular values.

If $y = Ix = x$, then **Maestro recovers PCA.**

Maestro Pipeline (Learn Network + Decomposition)



Algorithm 1: Maestro (Proposed Framework)

Input: epochs E, dataset $\mathcal{D}$, model h parametrized by $U^1 \in \mathbb{R}^{m_1\times r_1}$, $V^1 \in \mathbb{R}^{n_1\times r_1}, \ldots, U^d \in \mathbb{R}^{m_d\times r_d}, V^d \in \mathbb{R}^{n_d\times r_d}$, W^o, and hyperparameters λ_{gl}, ϵ_{ps}

```
for t ← 0 to E − 1 do   // Epochs
  for (x, y) ∈ D do   // Iterate over dataset
    Sample (i, b) ∼ {{(i, b)}_{b=1}^{r_i}}_{i=1}^{d};
    L = l(h(U^1(V^1)^T, ..., U^i_{:b}(V^i_{:b})^T, ..., U^d(V^d)^T, W^o, x), y) + λ_gl Σ_{i=1}^{d} Σ_{b=1}^{r_i}(||U^i_{b:}|| + ||V^i_{b:}||)
    L.backward() // Update weights
  end
end
for i ← 1 to d do // rank importance thresholding
  for b ← 1 to r_i do
    if ||V^i_{b:}|| ||U^i_{b:}|| ≤ ε_ps then
      r_i = b − 1; // progressive shrinking
      break
    end
  end
end
```

arXiv:2308.14929

Experiments (ResNet18 on CIFAR10)

Table 1: MAESTRO vs. baselines

Variant	Acc. (%)	GMACs	Params. (M)
Non-factorized	$93.86_{\pm 0.20}$	0.56	11.17
Pufferfish	94.17	0.22	3.336
Cuttlefish	93.47	0.3	3.108
MAESTRO† ($\lambda_{gp} = 16e^{-6}$)	$\mathbf{94.19}_{\pm 0.07}$	$0.39_{\pm 0.00}$	$4.08_{\pm 0.02}$
MAESTRO† ($\lambda_{gp} = 64e^{-6}$)	$93.86_{\pm 0.11}$	$0.15_{\pm 0.00}$	$1.23_{\pm 0.00}$

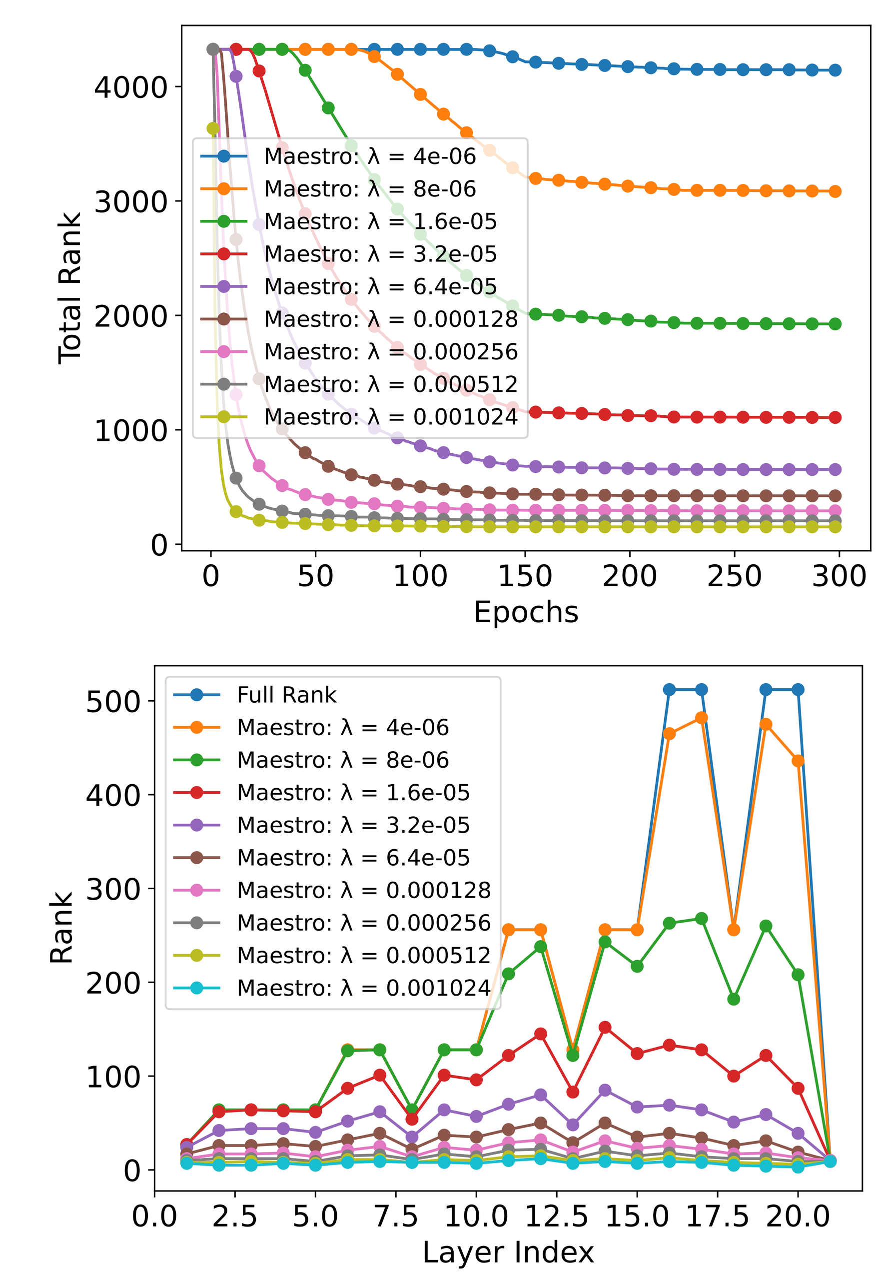


Figure 3: Training dynamics of MAESTRO. **Upper**: Total rank ($\Sigma_{i=1}^{d} r_i$).. **Bottom**: Ranks r_i's after training.

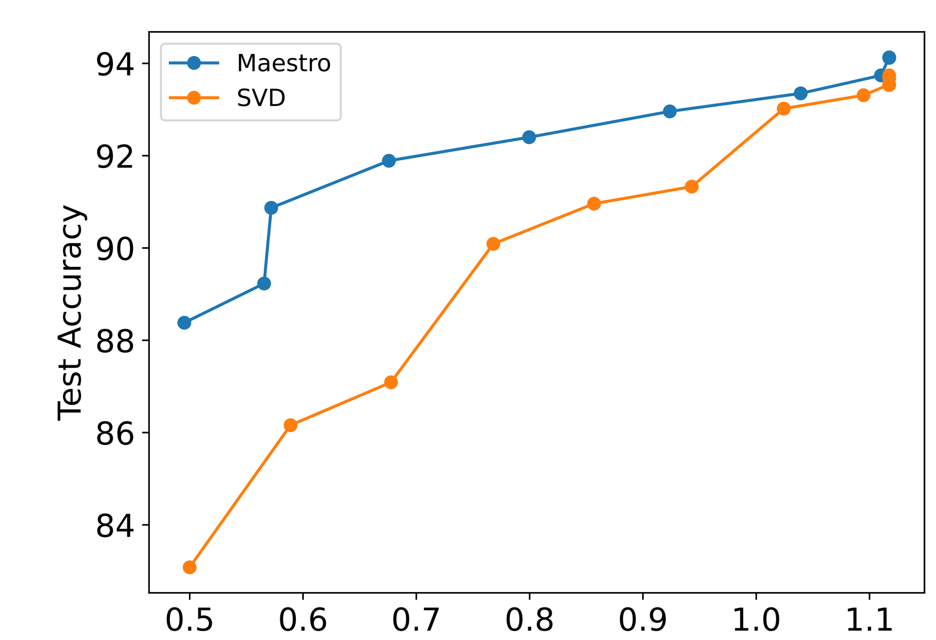


Figure 4: MAESTRO compared to SVD.

Results with extra models (VGG, Transformers) and datasets (Multi30k, TinyImageNet) in the paper.