Paper ID: 11346

# Beyond Low-rank Decomposition: A Shortcut Approach for Efficient On-Device Learning


Le-Trung Nguyen  Aël Quélennec  Van-Tam Nguyen  Enzo Tartaglione

LTCI, Télécom Paris, Institut Polytechnique de Paris

{name.surname}@telecom-paris.fr


International Conference On Machine Learning

# How to save up to 120.09× activation memory and 1.86× computational cost during training?

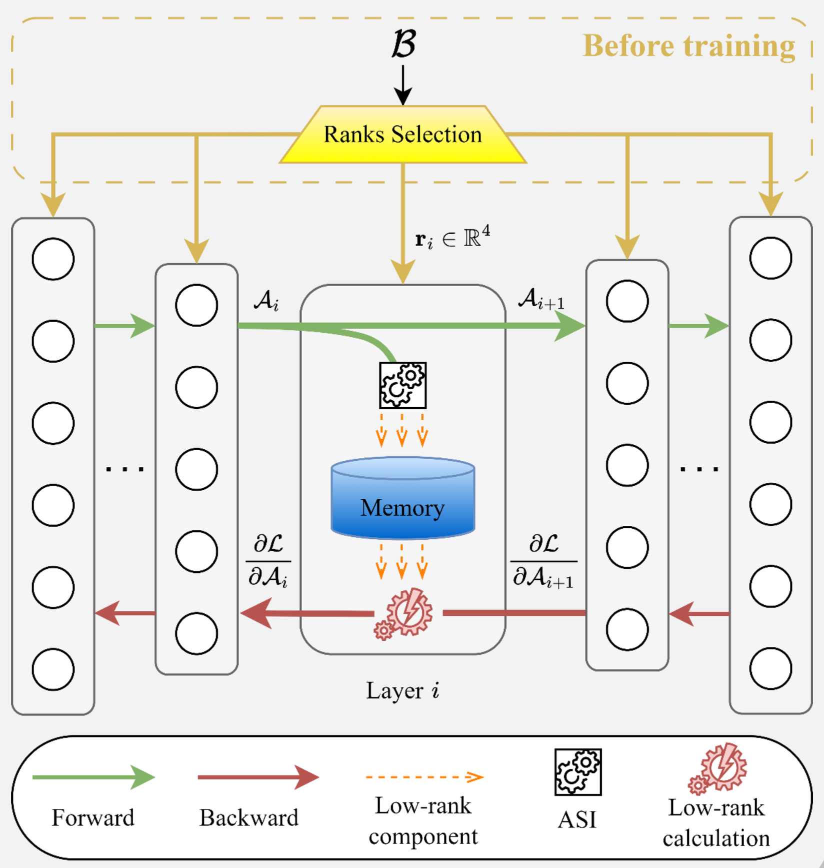
## The Memory Bottleneck of Backpropagation

Considering a convolutional neural network, during the backward pass at the $i^{th}$ layer, the following two values must be calculated:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} = \frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i+1}} \cdot \frac{\partial \mathcal{A}_{i+1}}{\mathcal{W}_i} = \text{conv}\left(\mathcal{A}_i, \frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i+1}}\right)$$

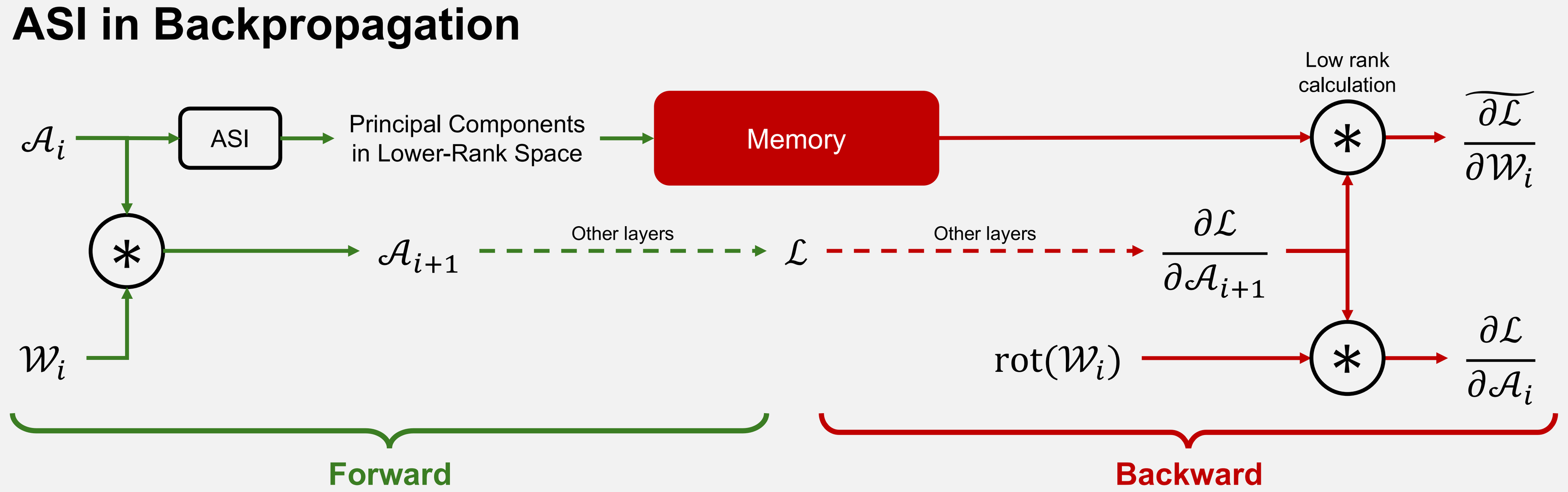
$$\frac{\partial \mathcal{L}}{\partial \mathcal{A}_i} = \frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i+1}} \cdot \frac{\partial \mathcal{A}_{i+1}}{\partial \mathcal{A}_i} = \text{conv}_{\text{full}}\left[\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i+1}}, \text{rot}(\mathcal{W}_i)\right]$$

Storing $\mathcal{A}_i$ and $\mathcal{W}_i$ is the main cause of memory occupancy during backpropagation. In this work, we propose decomposing $\mathcal{A}_i$ during the forward pass using **Activation Subspace Iteration (ASI)**. $\mathcal{W}_i$ is untouched to avoid error propagation through $\frac{\partial \mathcal{L}}{\partial \mathcal{A}_i}$.

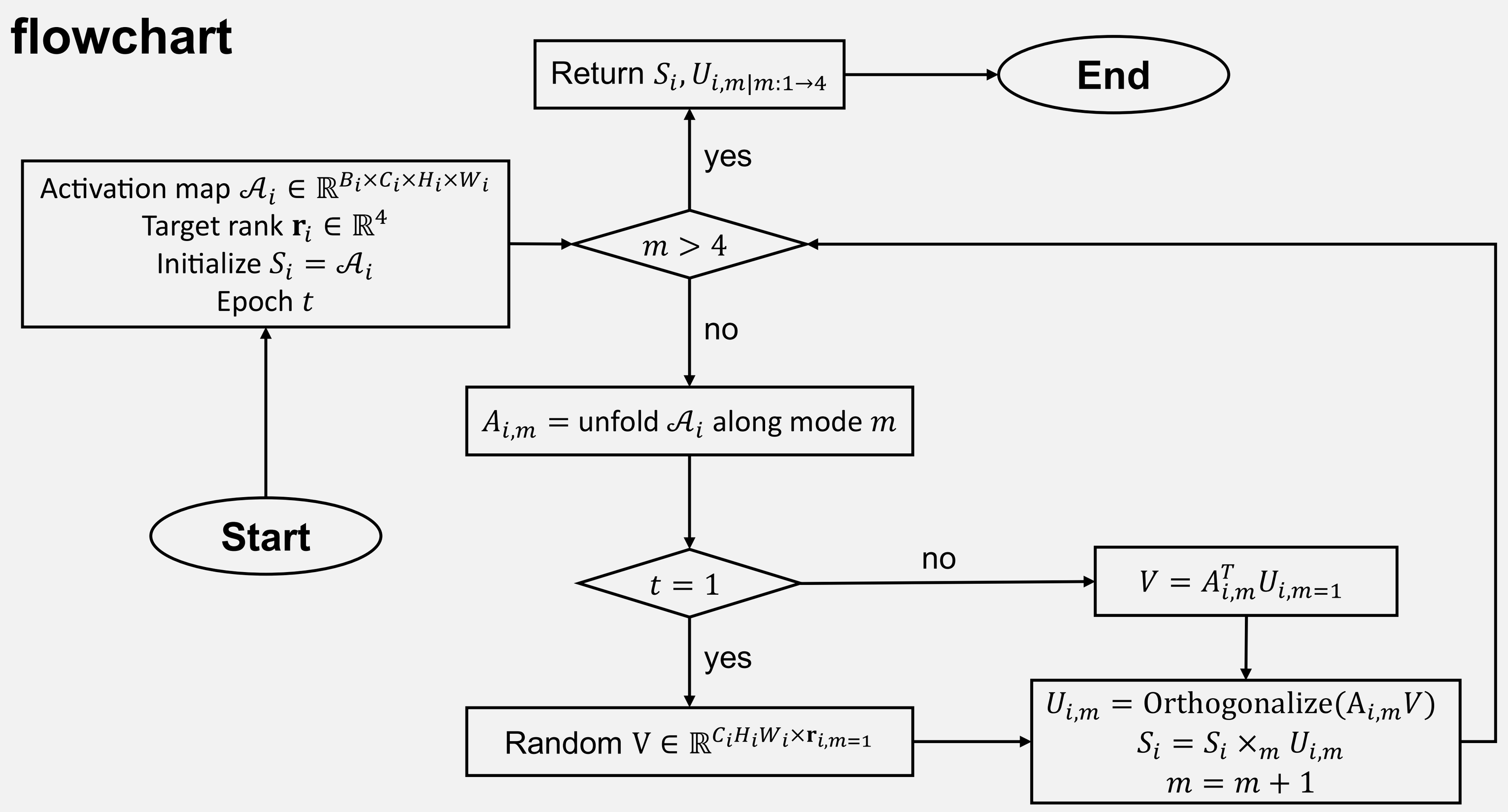



## Method

### ASI in Backpropagation



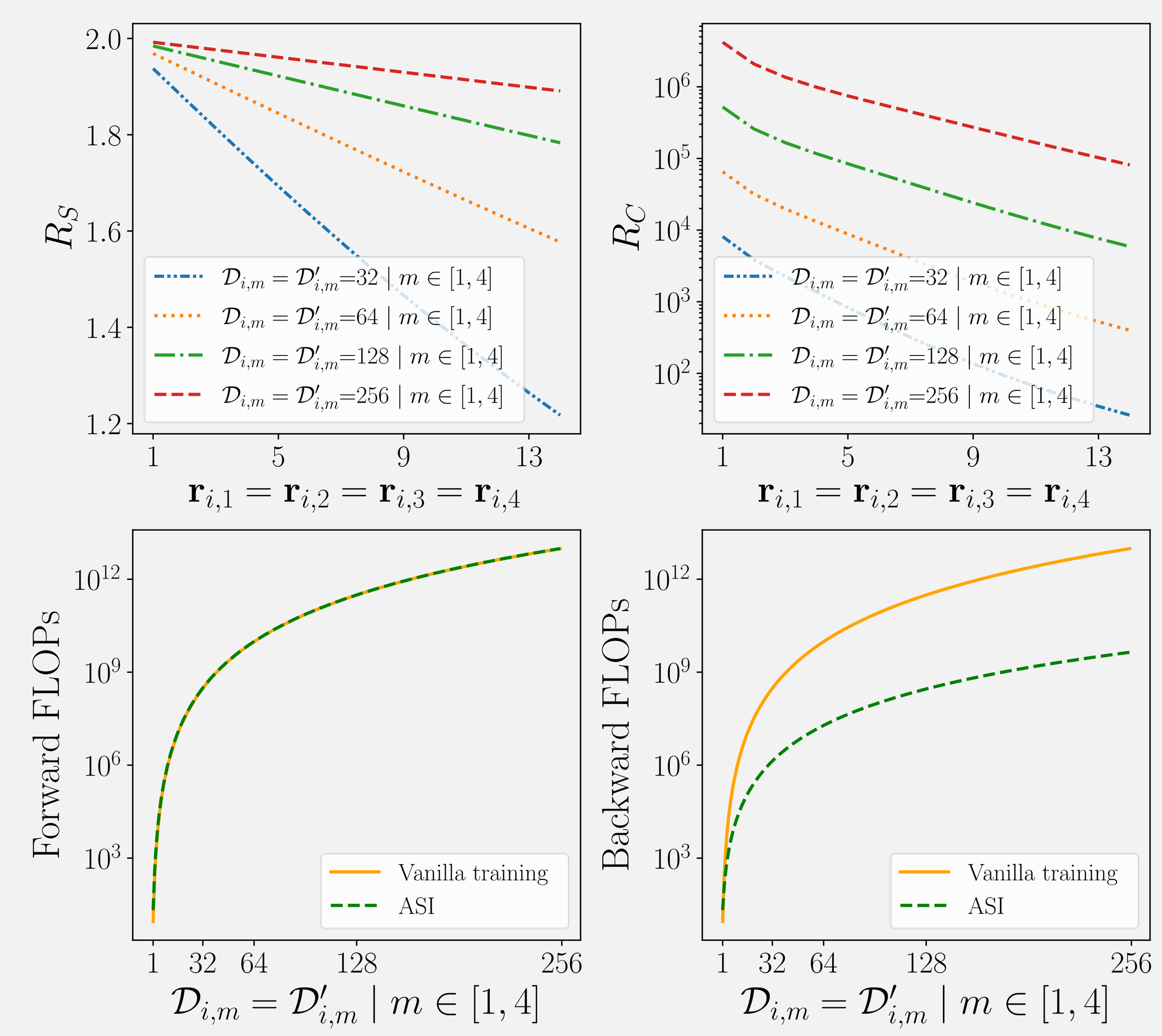

### ASI flowchart



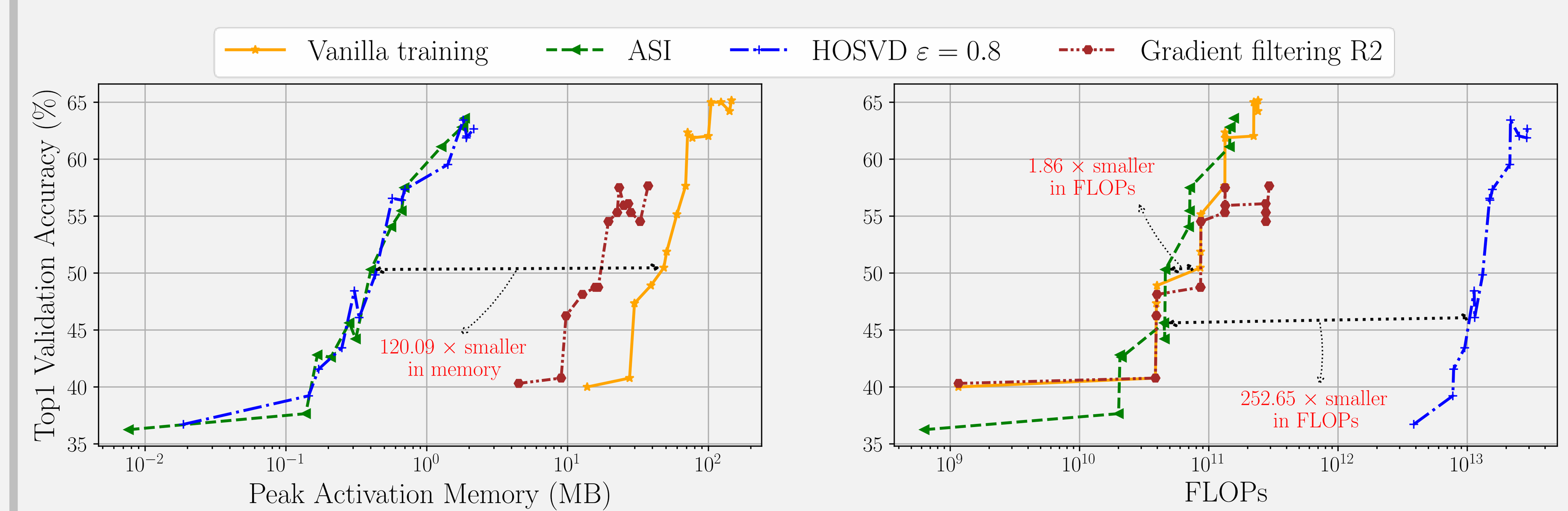

## Results

### Expected results

- Consider activation maps of the $i^{th}$ and the $(i+1)^{th}$ layers as $\mathcal{A}_i \in \mathbb{R}^{B \times C_i \times H_i \times W_i}$ and $\mathcal{A}'_i \in \mathbb{R}^{B \times C'_i \times H'_i \times W'_i}$, respectively.
- Let $\mathcal{D}_{i,m} = \{B, C_i, H_i, W_i\}$ and $\mathcal{D}'_{i,m} = \{B, C'_i, H'_i, W'_i\}$.
- The target rank for compression is $\mathbf{r} \in \mathbb{R}^{N \times 4}$, where $N$ is total number of layers and 4 is number of tensor mode of the activation maps.




### Fine-tuning different number of layers of MCUNet pretrained on ImageNet with CIFAR-10 as downstream dataset



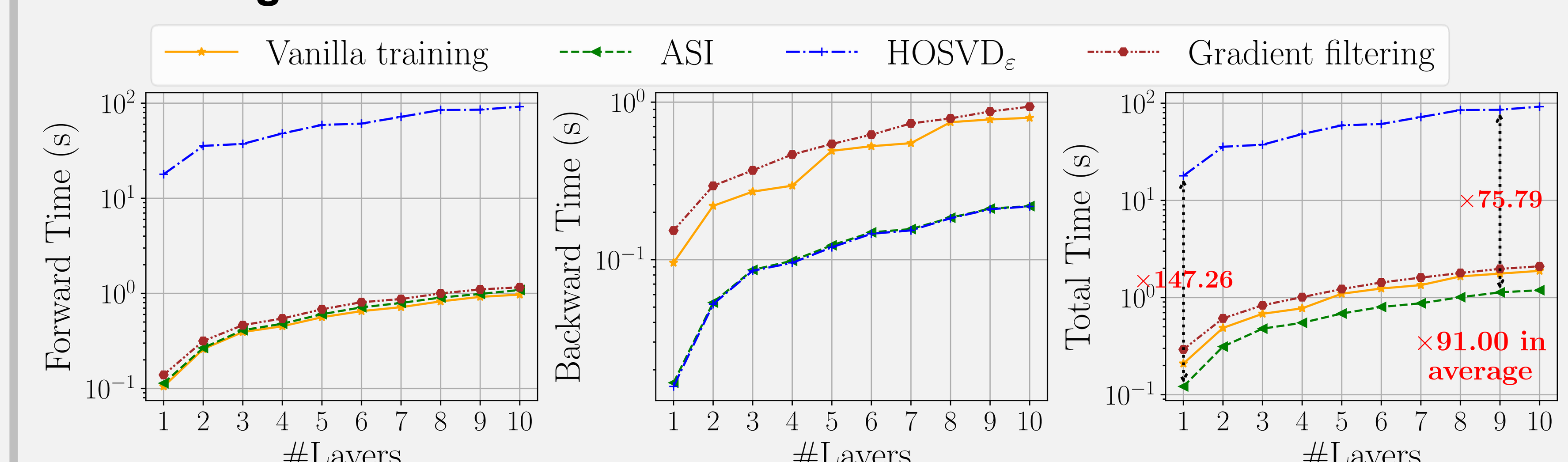

- ASI has similar performance to HOSVD in terms of memory compression, but it is significantly more efficient.
- For the same accuracy, ASI saves 120.09× more memory and is 1.86× cheaper than vanilla training.

### Experiment on ImageNet classification with different models

| Method | MobileNetV2 | | | | Method | ResNet18 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | #Layers | Acc ↑ | Mem (MB) ↓ | GFLOPs ↓ | | #Layers | Acc ↑ | Mem (MB) ↓ | GFLOPs ↓ |
| Vanilla training | All | 74.0 | 1651.84 | 830.82 | Vanilla training | All | 72.8 | 532.88 | 291.79 |
| | 2 | 62.6 | 15.31 | 4.50 | | 2 | 69.9 | 12.25 | 29.60 |
| | 4 | 65.8 | 28.71 | 57.48 | | 4 | 71.5 | 30.63 | 46.45 |
| Gradient filtering R2 | 2 | 62.6 | 5.00 | 4.50 | Gradient filtering R2 | 2 | 68.7 | 4.00 | 29.60 |
| | 4 | 65.2 | 9.38 | 57.50 | | 4 | 69.3 | 7.00 | 47.70 |
| HOSVD ($\varepsilon = 0.8$) | 2 | 61.1 | 0.15 | 3049.71 | HOSVD ($\varepsilon = 0.8$) | 2 | 69.2 | 0.97 | 1581.42 |
| | 4 | 63.9 | 0.73 | 5895.31 | | 4 | 70.5 | 2.89 | 4048.56 |
| ASI | 2 | 60.3 | 0.13 | 2.81 | ASI | 2 | 68.9 | 0.93 | 19.04 |
| | 4 | 64.0 | 0.71 | 31.54 | | 4 | 70.6 | 2.63 | 33.14 |
| **Method** | **MCUNet** | | | | **Method** | **ResNet34** | | | |
| | #Layers | Acc ↑ | Mem (MB) ↓ | GFLOPs ↓ | | #Layers | Acc ↑ | Mem (MB) ↓ | GFLOPs ↓ |
| Vanilla training | All | 67.4 | 632.98 | 248.84 | Vanilla training | All | 75.6 | 839.04 | 528.55 |
| | 2 | 62.1 | 13.78 | 19.31 | | 2 | 69.6 | 12.25 | 29.60 |
| | 4 | 64.7 | 19.52 | 19.88 | | 4 | 72.2 | 24.50 | 59.19 |
| Gradient filtering R2 | 2 | 61.8 | 4.50 | 19.31 | Gradient filtering R2 | 2 | 68.8 | 4.00 | 29.60 |
| | 4 | 64.4 | 6.38 | 19.89 | | 4 | 70.9 | 8.00 | 59.21 |
| HOSVD ($\varepsilon = 0.8$) | 2 | 61.7 | 0.48 | 1988.05 | HOSVD ($\varepsilon = 0.8$) | 2 | 68.7 | 0.30 | 1579.64 |
| | 4 | 63.9 | 0.88 | 2457.59 | | 4 | 71.1 | 1.11 | 3160.46 |
| ASI | 2 | 61.7 | 0.38 | 11.01 | ASI | 2 | 68.3 | 0.25 | 16.44 |
| | 4 | 63.5 | 0.83 | 11.93 | | 4 | 71.1 | 1.09 | 35.31 |

For the same #Layers, ASI saves significantly more memory and GLOPS than the others.

### Processing time




- The forward processing time of ASI is approximately the same as that of vanilla training, while the backward pass is up to an order of magnitude faster.
- Overall, ASI is on average 1.5× faster than vanilla training.


**Acknowledgement:** Part of this work was funded by Hi!PARIS Center on Data Analytics and Artificial Intelligence, by the European Union's Horizon Europe Research and Innovation Programme under grant agreement No. 101120237 (ELIAS), by the European Union's HORIZON Research and Innovation Programme under grant agreement No 101120657, project ENFIELD (European Lighthouse to Manifest Trustworthy and Green AI) and by French National Research Agency (ANR-22-PEFT-0003 and ANR-22-PEFT-0007) as part of France 2030, the NF-NAI project and NF-FITNESS project.


Paper

Code