




# Energy and Memory-Efficient AI for On-Device Learning

PhD Defense of Aël Quélennec

December 3, 2025

Supervisor: Van-Tam Nguyen
Co-Supervisors: Enzo Tartaglione and Pavlo Mozharovskyi
Reviewers: Dewey Yi and Ngoc-Son Vu
Examiners: Amel Bouzeghoub and Umut Şimşekli
Guest: Vito Paolo Pastore

# Outline

Introduction

Subnetwork Selection for Fine-Tuning

Activation Map Compression

Conclusion

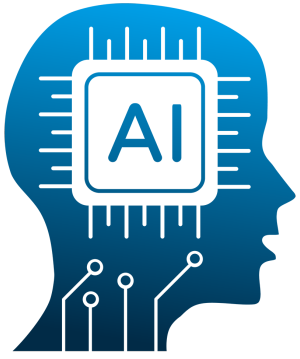
# Motivations

**Three questions to begin with**

# Motivations

**Three questions to begin with**

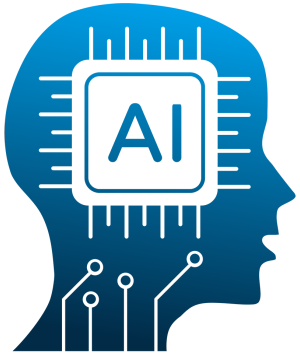
Why AI?



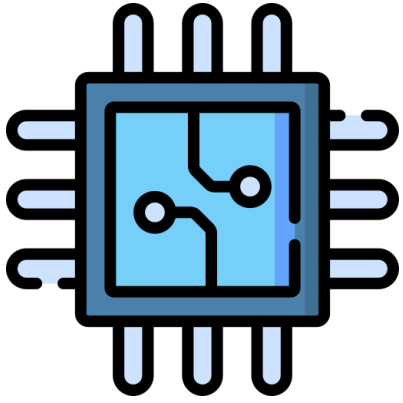
# Motivations

**Three questions to begin with**

Why AI?



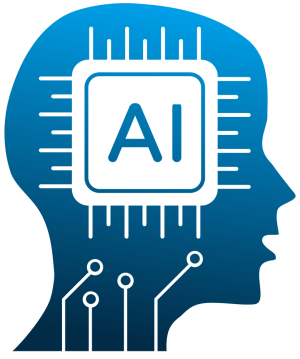

Why On-Device?



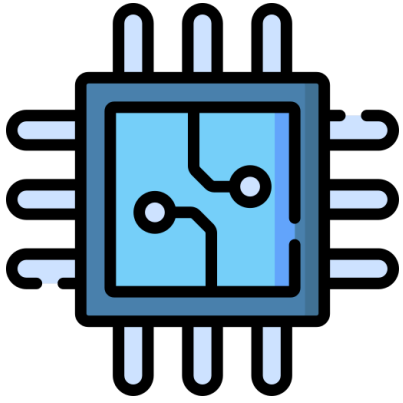
# Motivations

**Three questions to begin with**

Why AI?




Why On-Device?



What Challenges?

# Uncovering Artificial Intelligence (AI)

If we ask Chat-GPT:

## General Definition

**Artificial Intelligence** refers to computer systems designed to perform tasks that typically require human intelligence, such as recognizing patterns, making decisions, understanding language, or solving problems.

# Uncovering Artificial Intelligence (AI)

If we ask Chat-GPT:

**General Definition**

**Artificial Intelligence** refers to computer systems designed to perform tasks that typically require human intelligence, such as recognizing patterns, making decisions, understanding language, or solving problems.

Language Models,

# Uncovering Artificial Intelligence (AI)

If we ask Chat-GPT:

## General Definition

**Artificial Intelligence** refers to computer systems designed to perform tasks that typically require human intelligence, such as recognizing patterns, making decisions, understanding language, or solving problems.

Language Models,

Autonomous Vehicles,

# Uncovering Artificial Intelligence (AI)

If we ask Chat-GPT:

## General Definition

**Artificial Intelligence** refers to computer systems designed to perform tasks that typically require human intelligence, such as recognizing patterns, making decisions, understanding language, or solving problems.

Language Models,

Autonomous Vehicles,

Healthcare,

# Uncovering Artificial Intelligence (AI)

If we ask Chat-GPT:

## General Definition

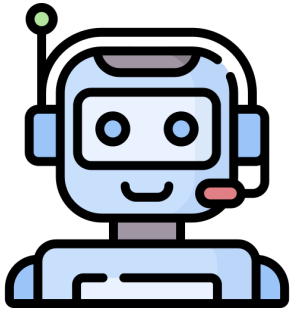
**Artificial Intelligence** refers to computer systems designed to perform tasks that typically require human intelligence, such as recognizing patterns, making decisions, understanding language, or solving problems.

Language Models,

Autonomous Vehicles,

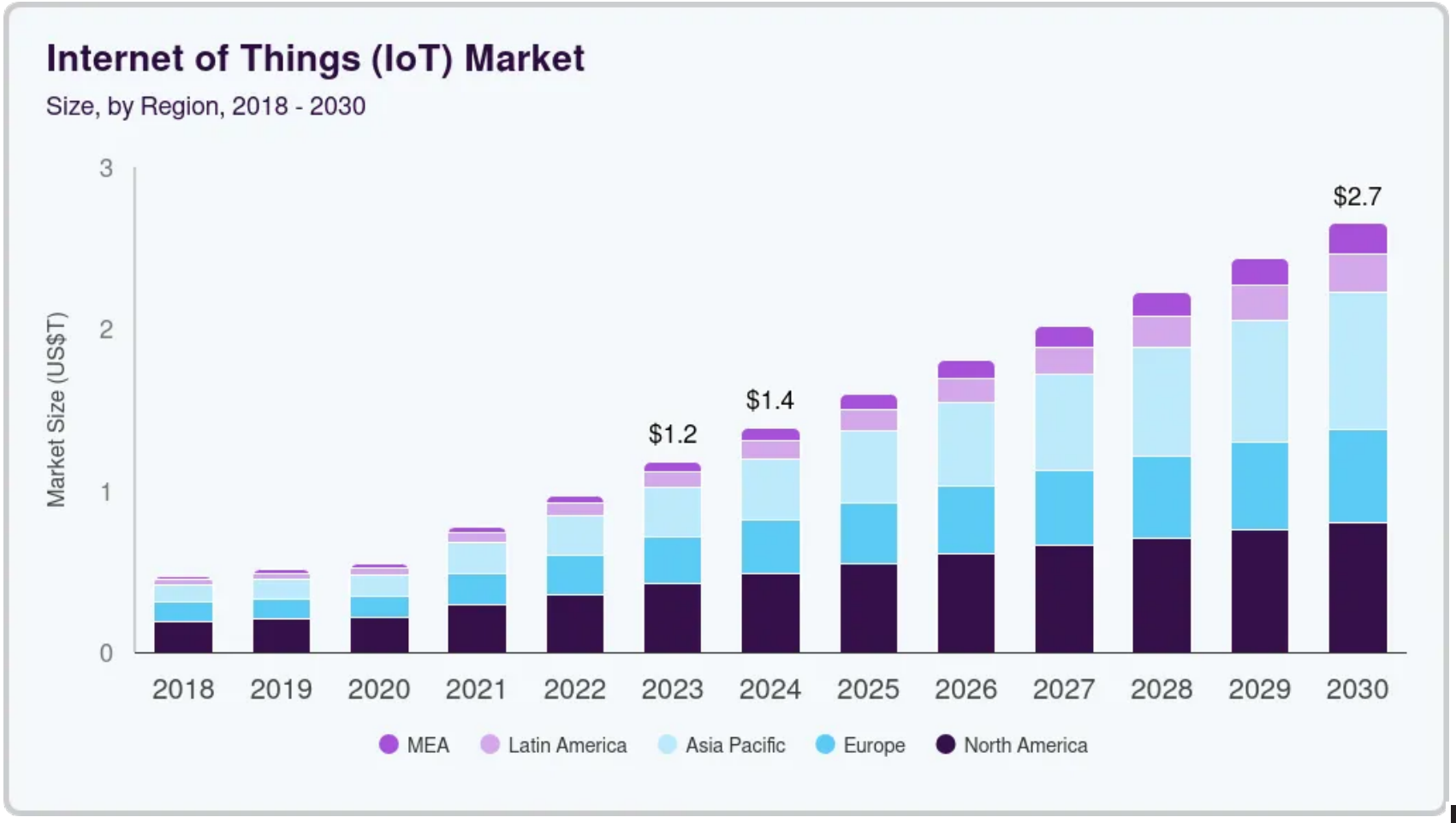
Healthcare,

Robotics, etc.



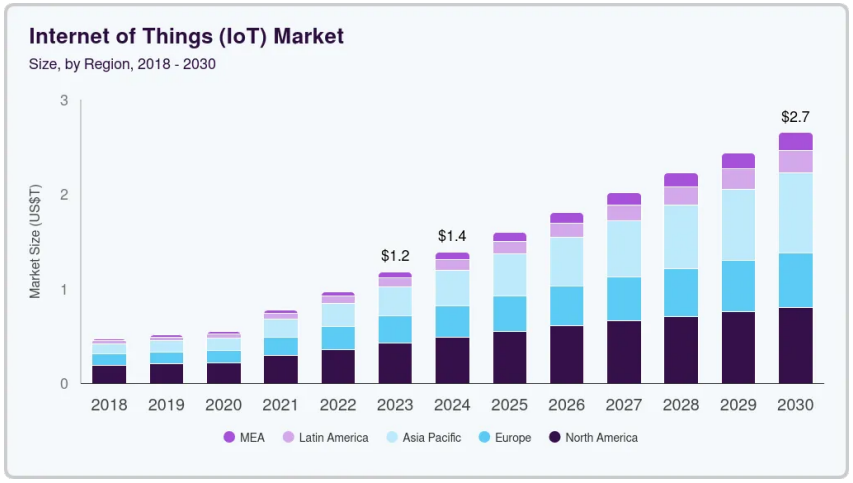
# Internet of Things

## A market in expansion




Source: Grand View Research - Internet of Things (IoT) Market (2024 - 2030)

# Internet of Things

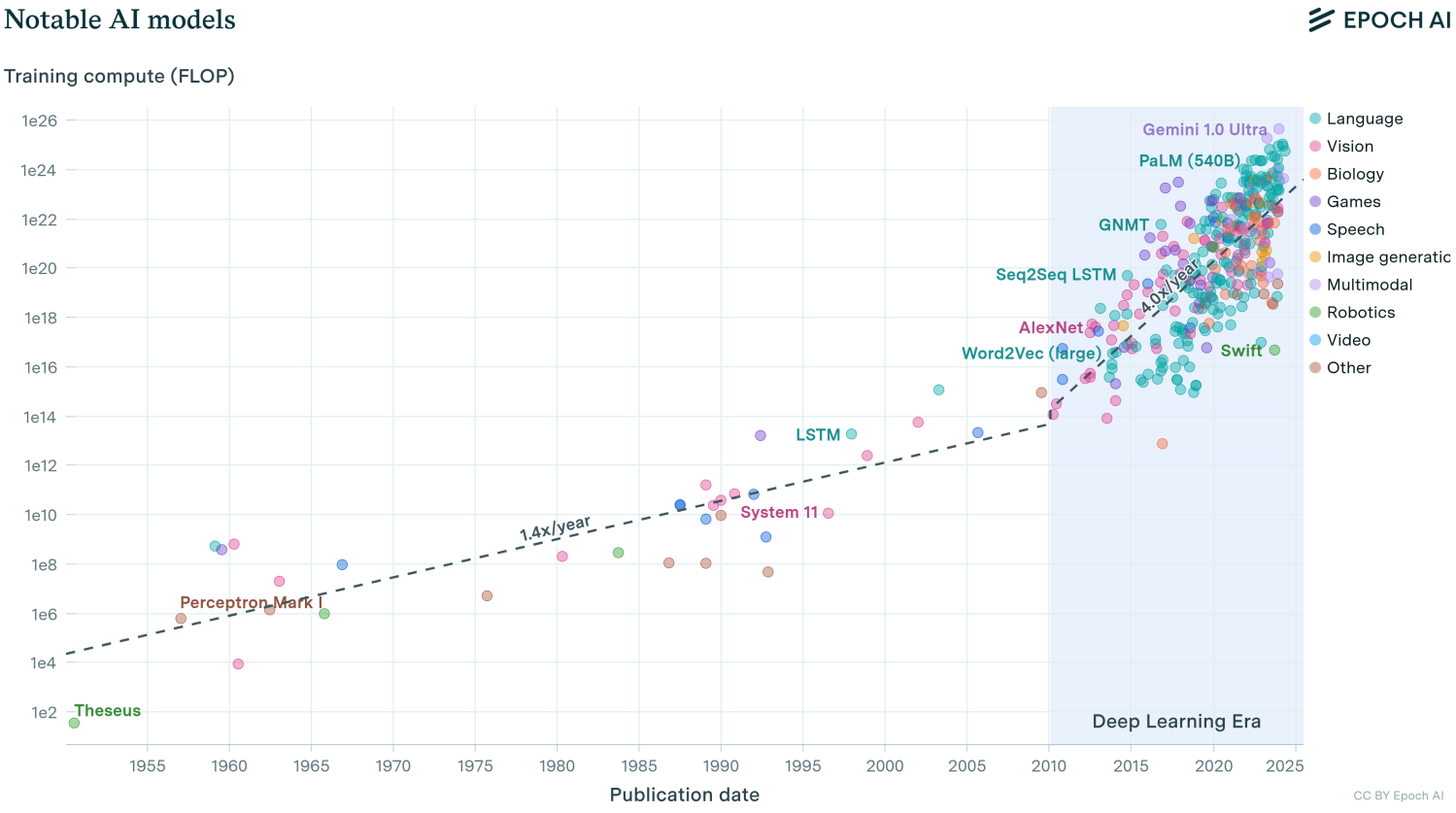



On-device AI has several benefits, among which:

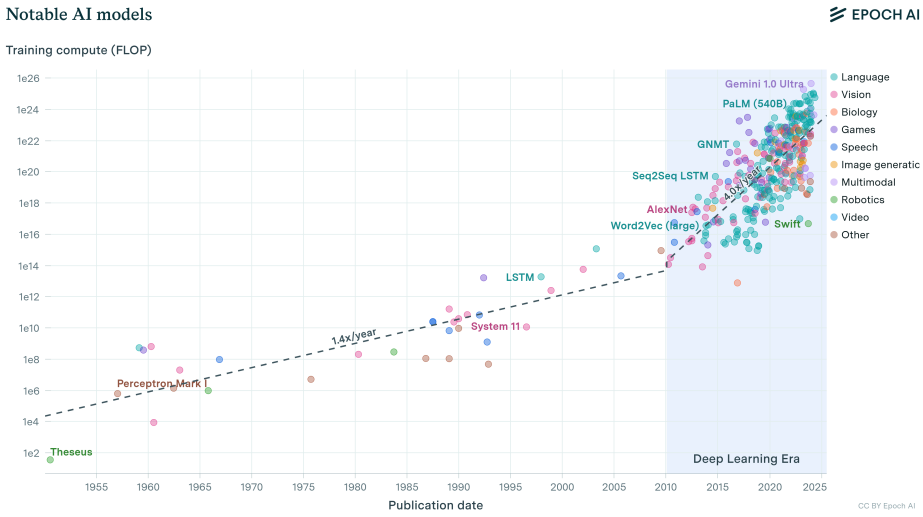
1. Enhanced privacy and security
2. Adaptive user personalization
3. Reduced energy consumption
4. Network independence

# Challenges of Embedded AI

## Computational trend in deep learning



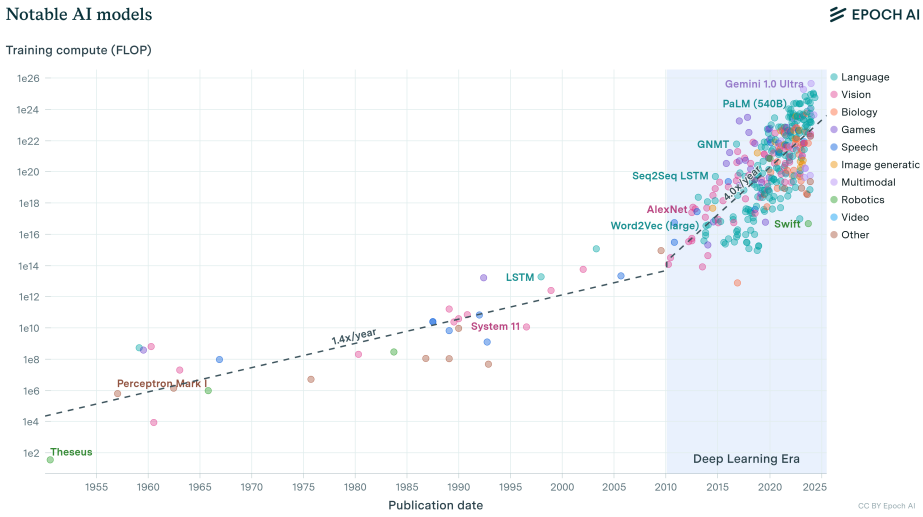
# Challenges of Embedded AI




Source: Epoch AI - Compute trends across three eras of machine learning

- GPT-4: 1.76 trillion parameters, 25,000 Nvidia A100 GPUs for 90–100 day to train and cost $100 M
- AlphaGo: 1920 CPUs and 280 GPUs, $3000 per game for electric bill

# Challenges of Embedded AI




Source: Epoch AI - Compute trends across three eras of machine learning

VS 

| | | |
|---|---|---|
| RAM: | GB/TB | 256kB |
| Storage: | TB/PB | 1MB |

- GPT-4: 1.76 trillion parameters, 25,000 Nvidia A100 GPUs for 90–100 day to train and cost $100 M
- AlphaGo: 1920 CPUs and 280 GPUs, $3000 per game for electric bill

# Outline

Introduction

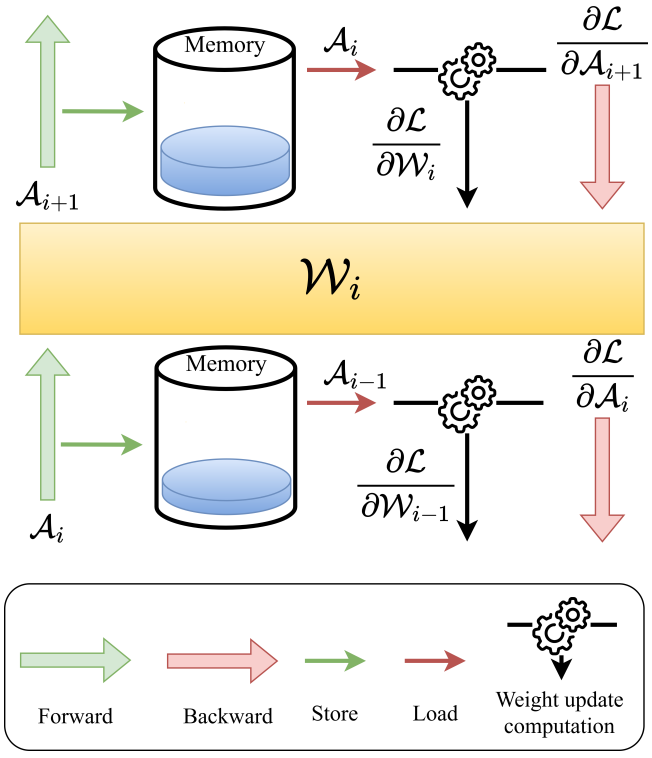
Subnetwork Selection for Fine-Tuning

Activation Map Compression

Conclusion

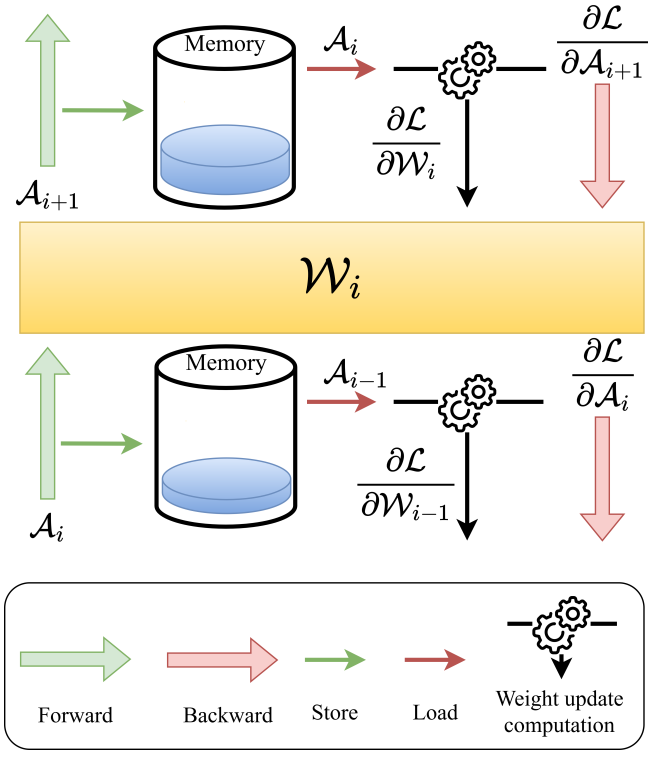
# Unboxing Backpropagation

## A two-stage process



# Unboxing Backpropagation

## A two-stage process

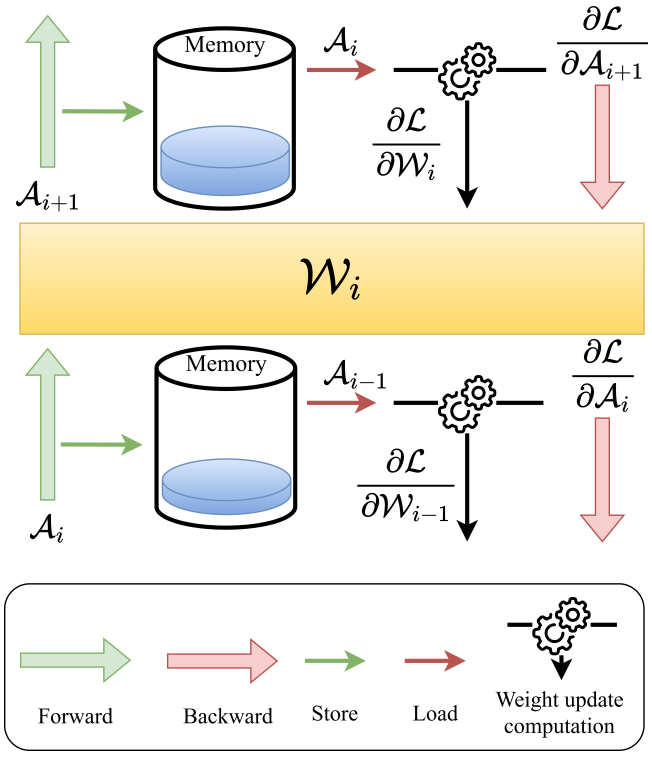



## Forward pass

$$\mathcal{A}_{i+1} = f_1(\mathcal{W}_i, \mathcal{A}_i)$$

# Unboxing Backpropagation

## A two-stage process




## Forward pass

$$\mathcal{A}_{i+1} = f_1\left(\mathcal{W}_i, \mathcal{A}_i\right)$$

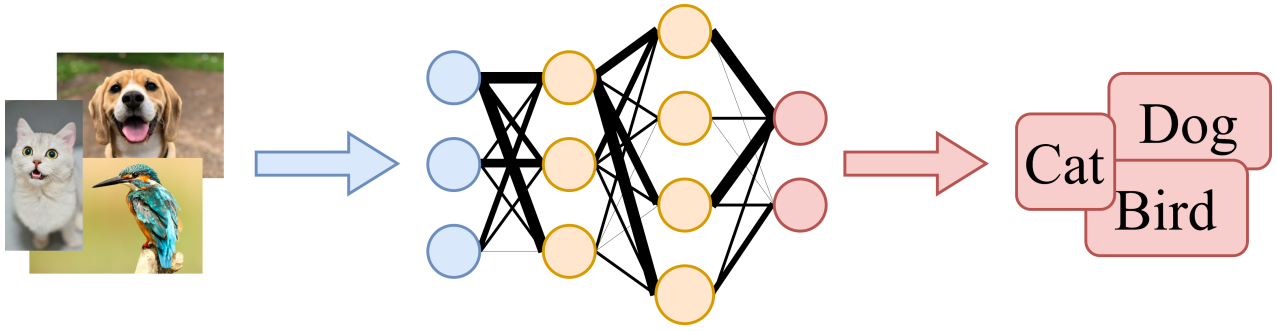
## Backward pass

$$\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} = f_1\left(\mathcal{A}_i, \frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i+1}}\right)$$

$$\frac{\partial \mathcal{L}}{\partial \mathcal{A}_i} = f_2\left(\frac{\partial \mathcal{L}}{\partial \mathcal{A}_{i+1}}, \mathcal{W}_i\right)$$

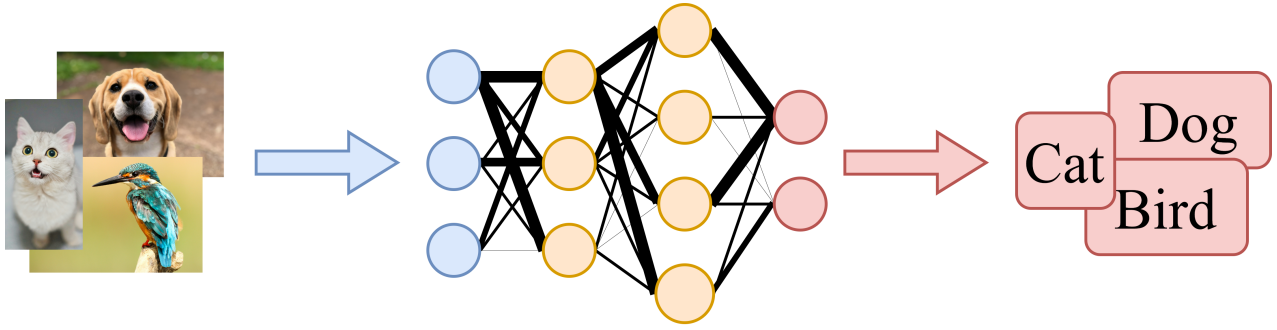
# Fine-Tuning Pipeline

## Stage 1 - Learning general features

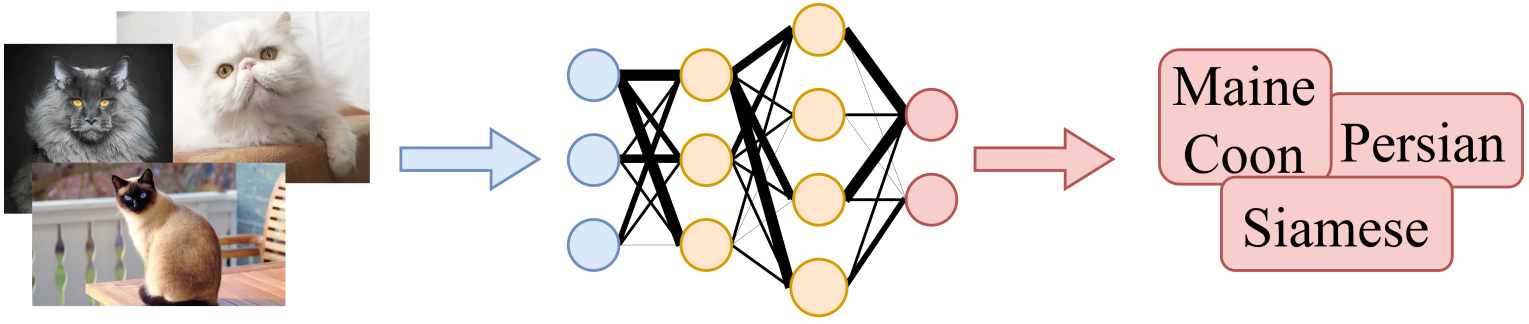


# Fine-Tuning Pipeline

## Stage 1 - Learning general features




## Stage 2 - Specializing on a downstream task



# Sparse Fine-Tuning

**What granularity?**

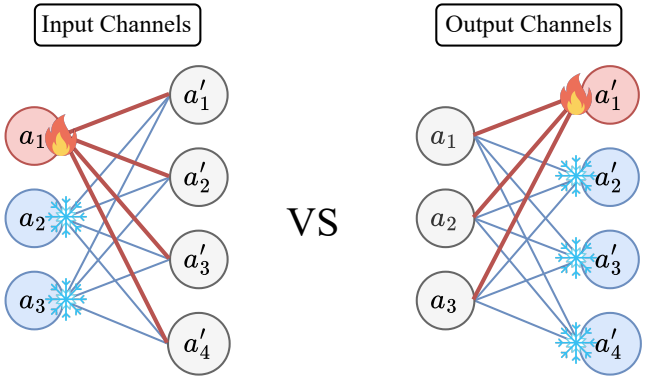
Necessity of structured selection:

# Sparse Fine-Tuning

## What granularity?

Necessity of structured selection:

### Channel level

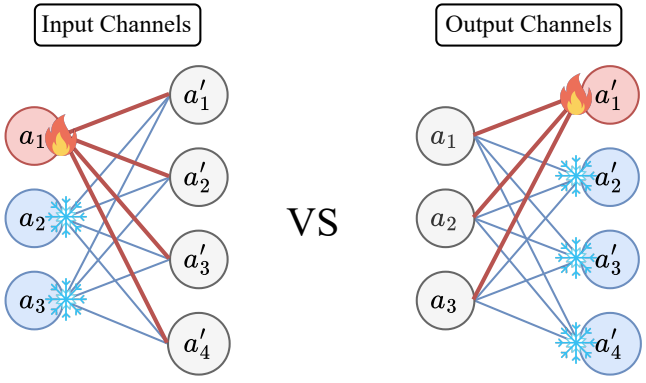


# Sparse Fine-Tuning

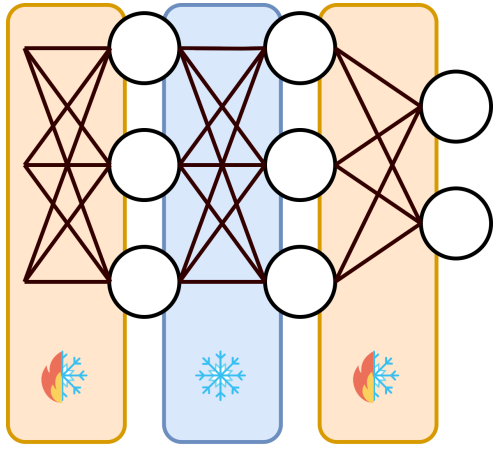
## What granularity?

Necessity of structured selection:

### Channel level

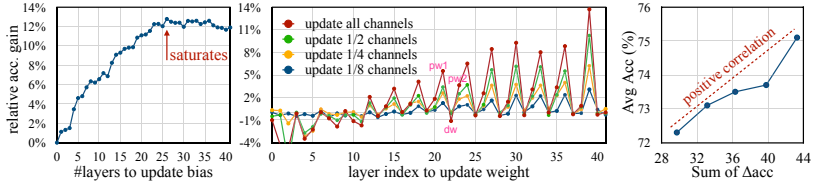



### Layer level



# Existing Approaches

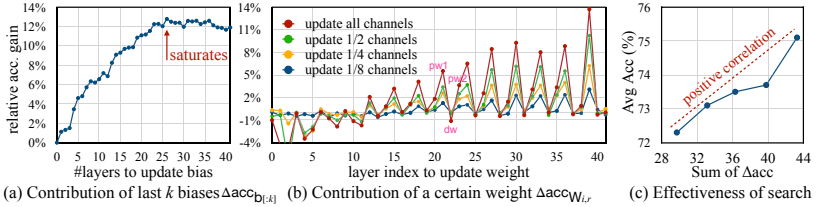
Lin et al. (2022)'s **Sparse Update** (SU):



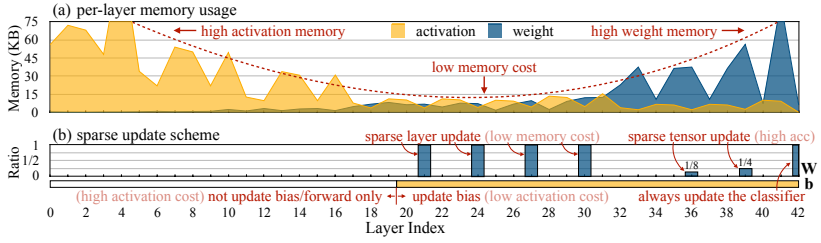

(a) Contribution of last $k$ biases $\Delta acc_{b[:k]}$ (b) Contribution of a certain weight $\Delta acc_{W_{i,r}}$ (c) Effectiveness of search

# Existing Approaches

Lin et al. (2022)'s **Sparse Update** (SU):

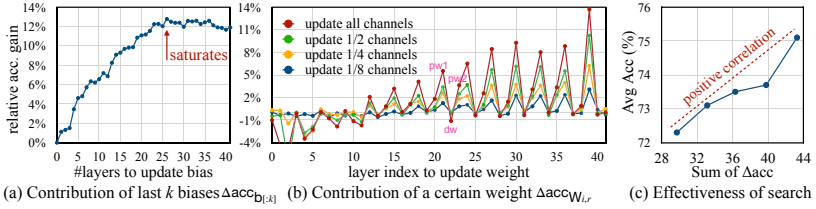



(a) Contribution of last $k$ biases $\Delta acc_{b[:k]}$ (b) Contribution of a certain weight $\Delta acc_{W_{i,r}}$ (c) Effectiveness of search

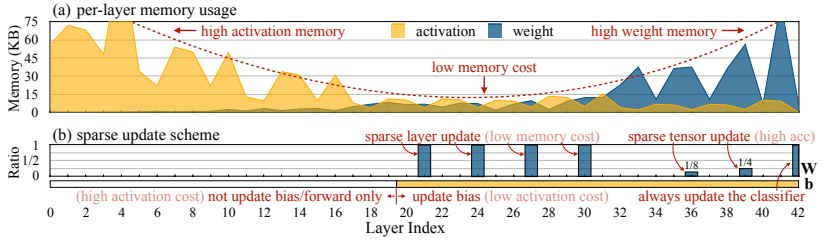


# Existing Approaches

Lin et al. (2022)'s **Sparse Update** (SU):



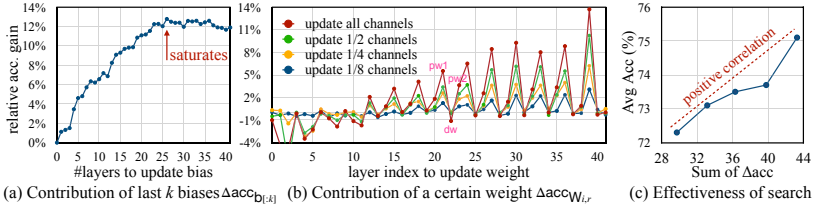

(a) Contribution of last $k$ biases $\Delta acc_{b[:k]}$ (b) Contribution of a certain weight $\Delta acc_{W_{i,r}}$ (c) Effectiveness of search



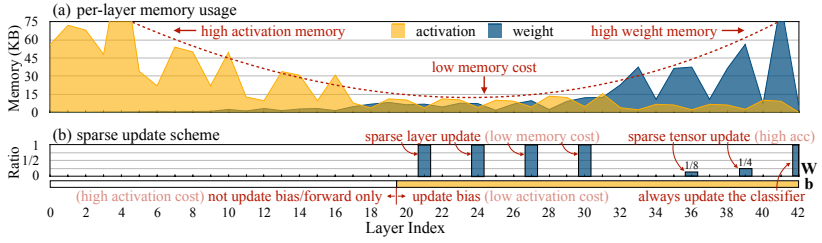

- Expensive
- Rigid
- Task-dependent
- Static

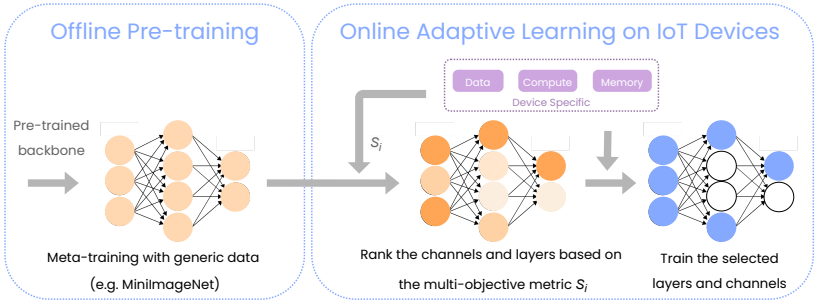
# Existing Approaches

Lin et al. (2022)'s **Sparse Update** (SU):




(a) Contribution of last $k$ biases $\Delta acc_{b[:k]}$ (b) Contribution of a certain weight $\Delta acc_{W_{l,r}}$ (c) Effectiveness of search



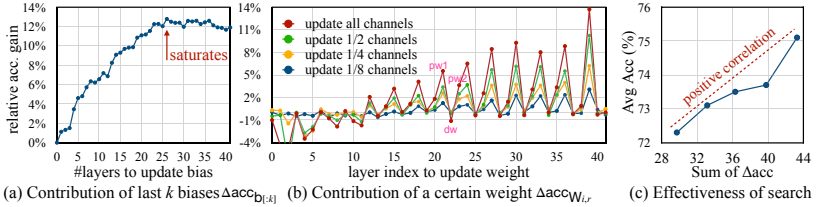




Kwon et al. (2024)'s **TinyTrain**:

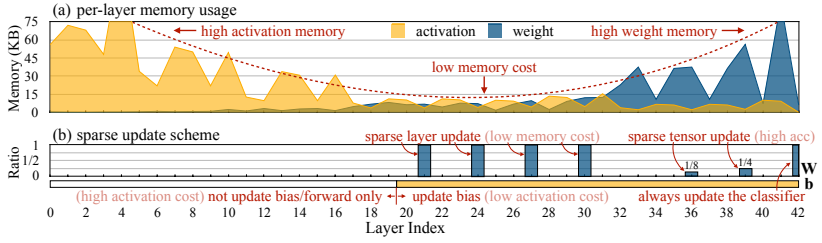
- Expensive
- Rigid
- Task-dependent
- Static

# Existing Approaches

Lin et al. (2022)'s **Sparse Update** (SU):

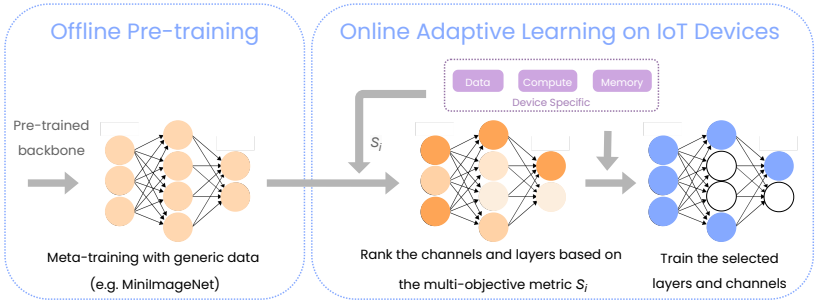



(a) Contribution of last $k$ biases $\Delta acc_{b[:k]}$ (b) Contribution of a certain weight $\Delta acc_{W_{l,r}}$ (c) Effectiveness of search




- Expensive
- Rigid
- Task-dependent
- Static

Kwon et al. (2024)'s **TinyTrain**:




- Memory-expensive
- Static

# Setup Definition

**Target solution**

- Network and task agnostic
- Efficient in memory, energy and accuracy
- Theoretically grounded

# Setup Definition

## Target solution

- Network and task agnostic
- Efficient in memory, energy and accuracy
- Theoretically grounded

## Constraints

- Fixed memory budget $B_{\text{mem}}$
- No prior knowledge of downstream task

# Setup Definition

## Target solution

- Network and task agnostic
- Efficient in memory, energy and accuracy
- Theoretically grounded

## Constraints

- Fixed memory budget $B_{\text{mem}}$
- No prior knowledge of downstream task

## Key intuition

Avoid allocating resources to compute gradients when $\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \approx 0$

# Setup Definition

**Target solution**

- Network and task agnostic
- Efficient in memory, energy and accuracy
- Theoretically grounded

**Constraints**

- Fixed memory budget $B_{\text{mem}}$
- No prior knowledge of downstream task

**Key intuition**

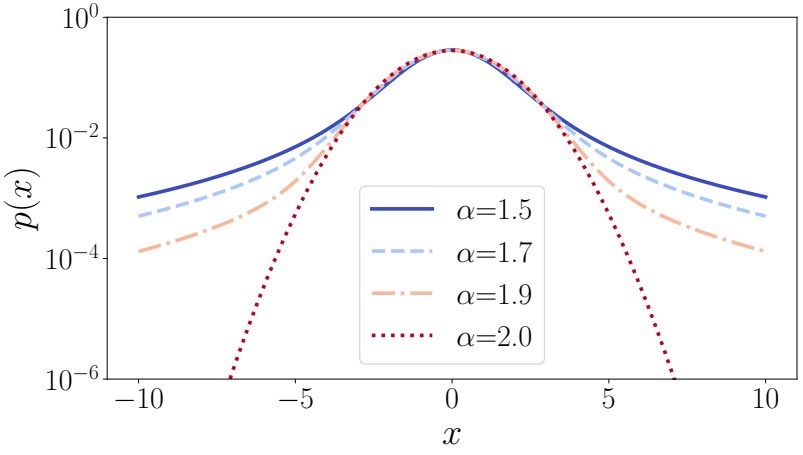
Avoid allocating resources to compute gradients when $\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \approx 0 \rightarrow$

**How to predict this in advance?**

# Introducing Heavy-Tailed Theory

Simsekli et al. (2019): Gradient noise $U_k \sim \mathcal{S}\alpha\mathcal{S}(\sigma)$:

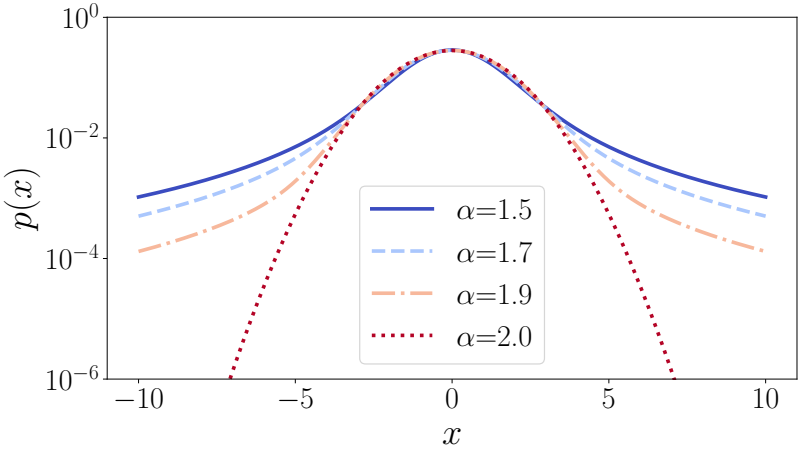
$$U_k(\mathcal{W}) = \Delta\tilde{\mathcal{W}}_k - \Delta\mathcal{W},$$



# Introducing Heavy-Tailed Theory

Simsekli et al. (2019): Gradient noise $U_k \sim \mathcal{S}\alpha\mathcal{S}(\sigma)$:

$$U_k(\mathcal{W}) = \Delta\tilde{\mathcal{W}}_k - \Delta\mathcal{W},$$




Wan et al. (2023): Injecting **heavy-tailed noise** in SGD creates **implicit sparse structure**

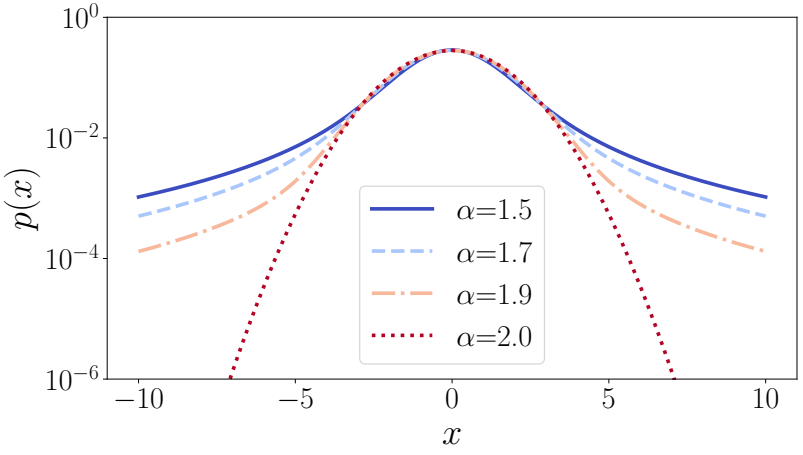
→ Few dominant columns carry most of weight matrix norm

→ Natural alignment with pruning

# Introducing Heavy-Tailed Theory

Simsekli et al. (2019): Gradient noise $U_k \sim \mathcal{S}\alpha\mathcal{S}(\sigma)$:

$$U_k(\mathcal{W}) = \Delta\tilde{\mathcal{W}}_k - \Delta\mathcal{W},$$




Wan et al. (2023): Injecting **heavy-tailed noise** in SGD creates **implicit sparse structure**

→ Few dominant columns carry most of weight matrix norm

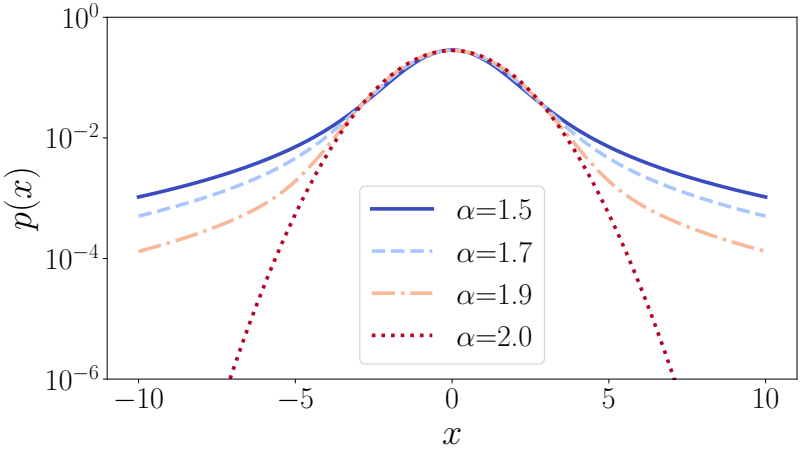
→ Natural alignment with pruning

Gradient = stochastic gradient + heavy-tailed noise

# Introducing Heavy-Tailed Theory

Simsekli et al. (2019): Gradient noise $U_k \sim \mathcal{S}\alpha\mathcal{S}(\sigma)$:

$$U_k(\mathcal{W}) = \Delta\tilde{\mathcal{W}}_k - \Delta\mathcal{W},$$




Wan et al. (2023): Injecting **heavy-tailed noise** in SGD creates **implicit sparse structure**

→ Few dominant columns carry most of weight matrix norm

→ Natural alignment with pruning

Gradient = stochastic gradient + heavy-tailed noise → **gradient is prunable with respect to norm**

# Analyzing Gradient Norm Evolution

For a given architecture

# Analyzing Gradient Norm Evolution

For a given architecture

**Channel Proposition**

The distribution of channel gradient norms varies between datasets.

# Analyzing Gradient Norm Evolution

For a given architecture

**Channel Proposition**

The distribution of channel gradient norms varies between datasets.

$\rightarrow$ prevents *a priori* channel selection

# Analyzing Gradient Norm Evolution

For a given architecture

**Channel Proposition**

The distribution of channel gradient norms varies between datasets.

$\rightarrow$ prevents *a priori* channel selection

**Layer Proposition**

Layer gradient norm ranking remains stable across time and tasks, determined by architecture rather than dataset.

# Analyzing Gradient Norm Evolution

For a given architecture

**Channel Proposition**

The distribution of channel gradient norms varies between datasets.

→ prevents *a priori* channel selection

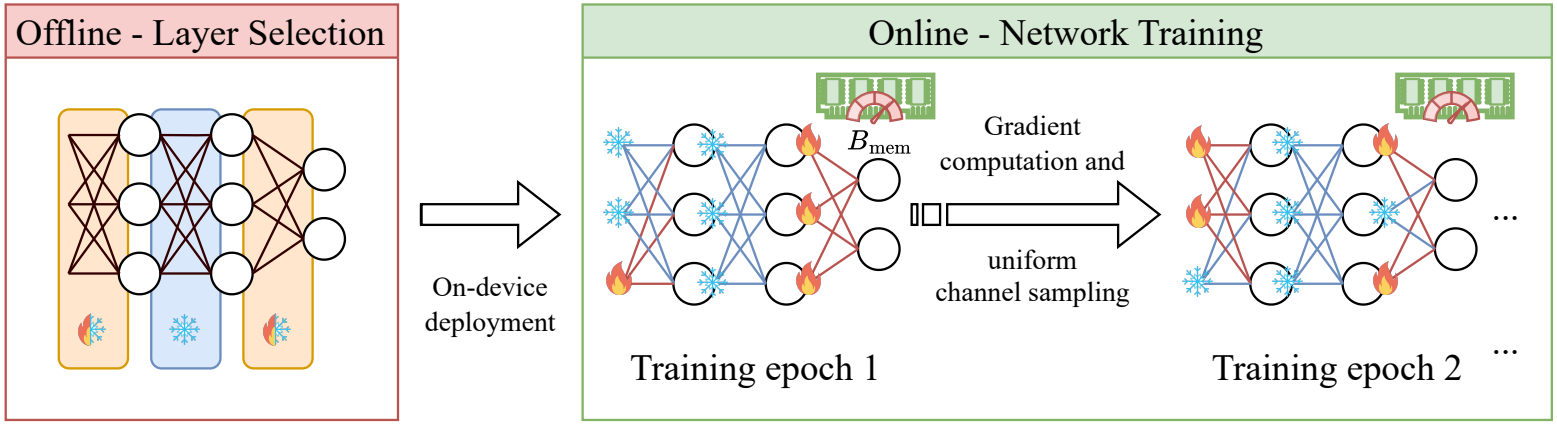
**Layer Proposition**

Layer gradient norm ranking remains stable across time and tasks, determined by architecture rather than dataset.

→ allows *a priori* layer selection

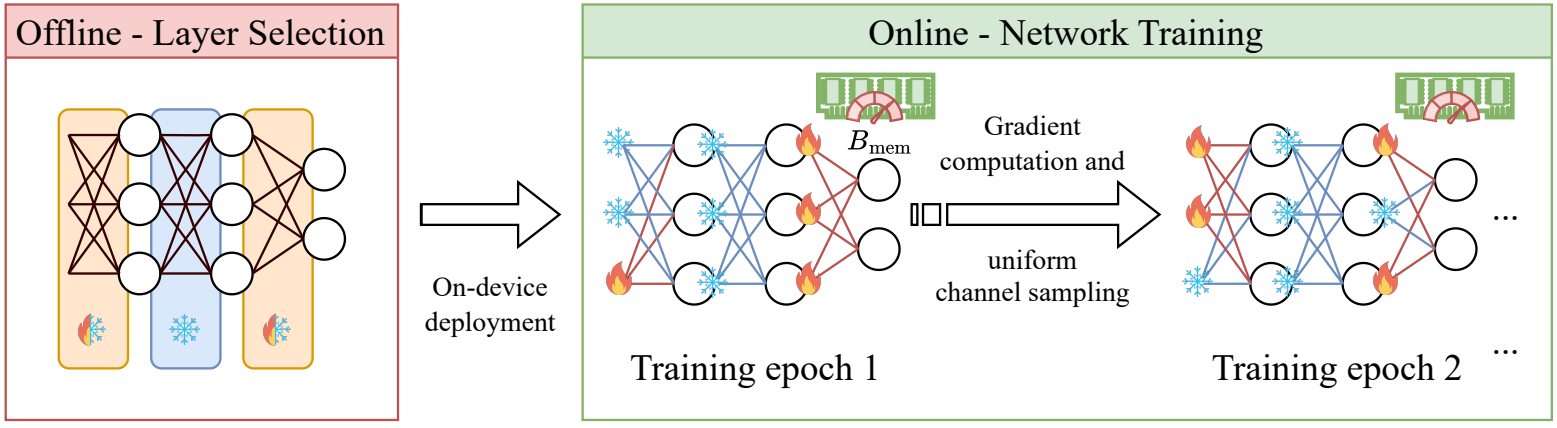
# Our Proposed Algorithm

## **Tra**ining **Dy**namics (TraDy)



# Our Proposed Algorithm

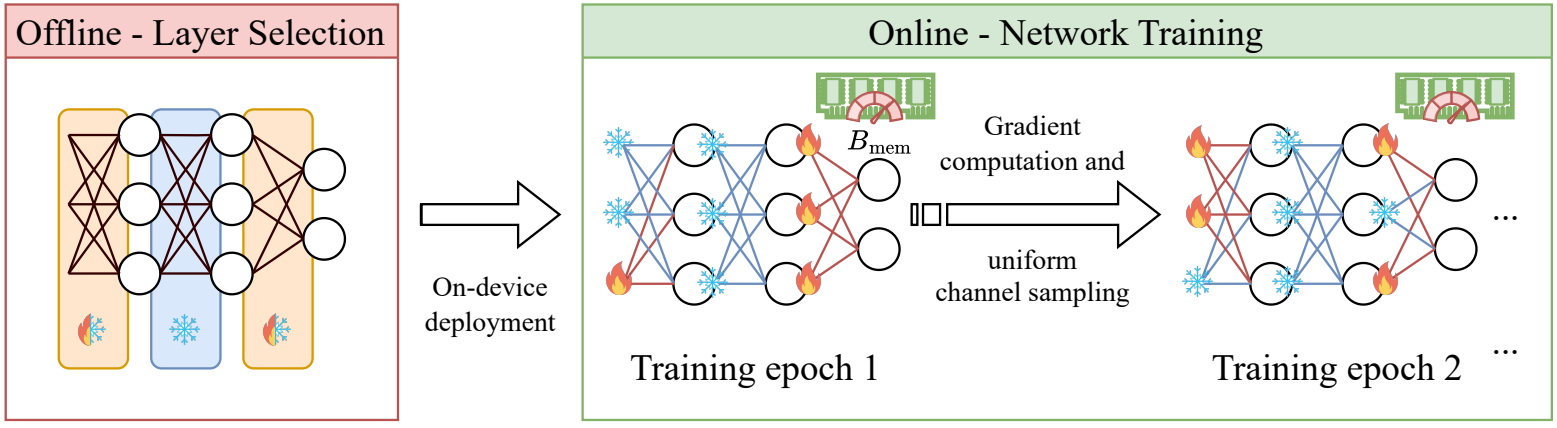
## Training Dynamics (TraDy)




## Design choices

# Our Proposed Algorithm

## **Tra**ining **Dy**namics (TraDy)

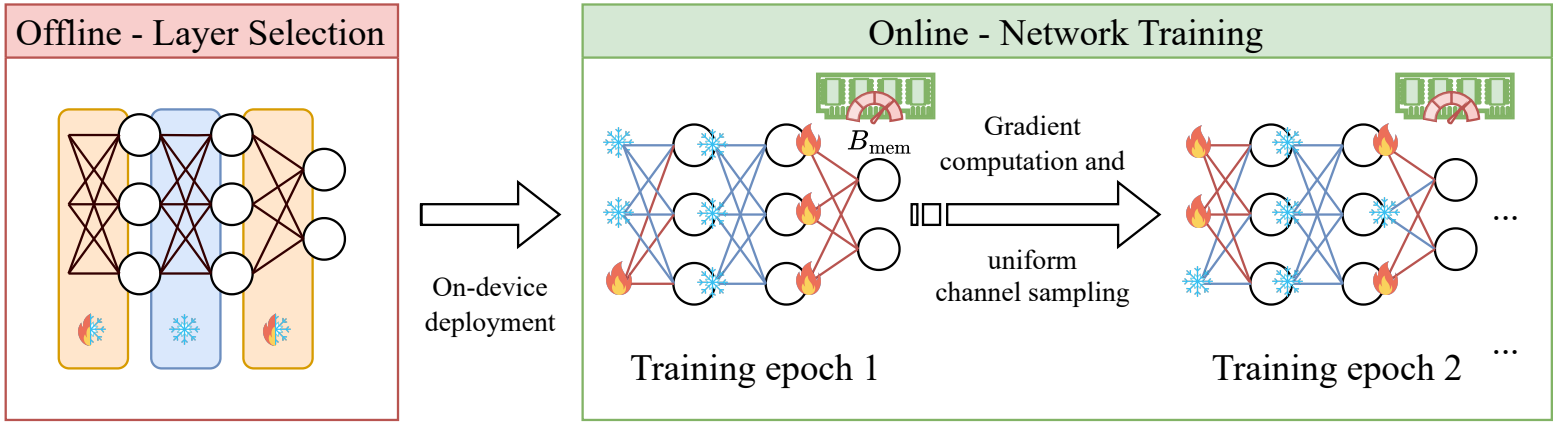



## Design choices

- Memory-normalized gradient metric: $\text{RGN}_c = \dfrac{\left\| \left( \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \right)_c \right\|_2}{(\Theta_{\text{space}})_c}$

# Our Proposed Algorithm

## **Tra**ining **Dy**namics (TraDy)

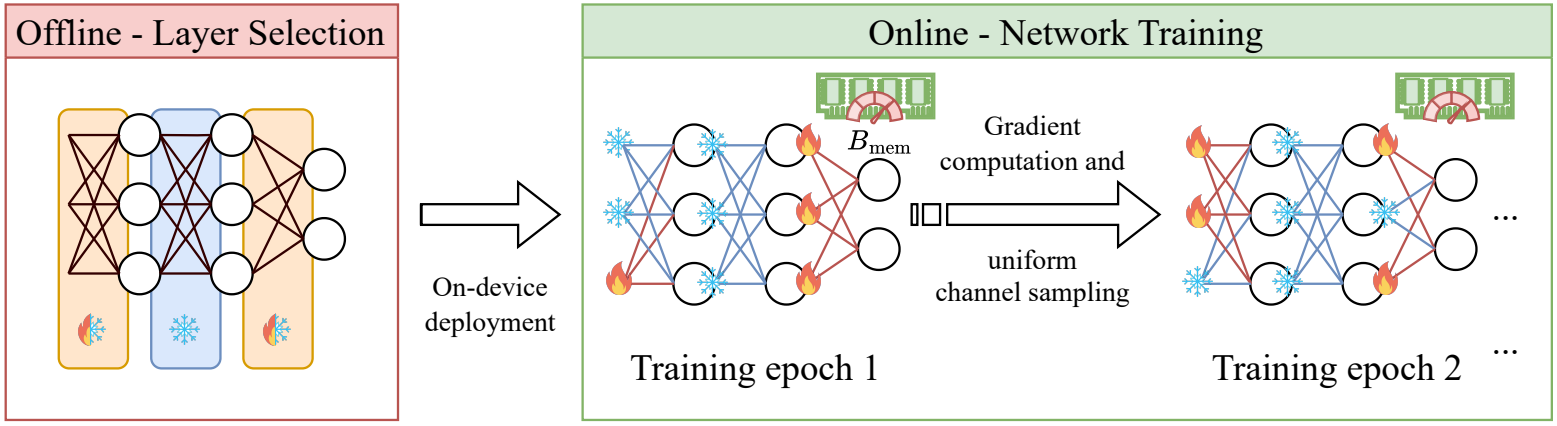



## Design choices

- Memory-normalized gradient metric: $\text{RGN}_c = \frac{\left\| \left( \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \right)_c \right\|_2}{(\Theta_{\text{space}})_c}$
- Layer selection: retain layers capturing 97% of cumulative RGN
  $\rightarrow$ reduces from 52 to 35 layers in MobileNetV2

# Our Proposed Algorithm

## **Tra**ining **Dy**namics (TraDy)




## Design choices

- Memory-normalized gradient metric: $\text{RGN}_c = \frac{\left\| \left( \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \right)_c \right\|_2}{(\Theta_{\text{space}})_c}$
- Layer selection: retain layers capturing 97% of cumulative RGN $\rightarrow$ reduces from 52 to 35 layers in MobileNetV2
- Dynamic random channel selection $\rightarrow$ maintains non-null gradient expectation: $\mathbb{E}_t \left[ \Delta \tilde{\mathcal{W}}_t \right] \simeq \mathbb{E}_t \left[ \Delta \mathcal{W}_{\{C^t\}} \right]$

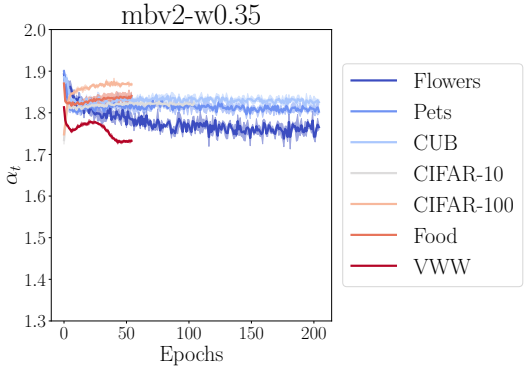
# Empirical Validation of Hypotheses

**Setup:** 3 architectures, 7 datasets, 3 memory budgets, 3 seeds

# Empirical Validation of Hypotheses

**Setup:** 3 architectures, 7 datasets, 3 memory budgets, 3 seeds

Measuring $\alpha$:

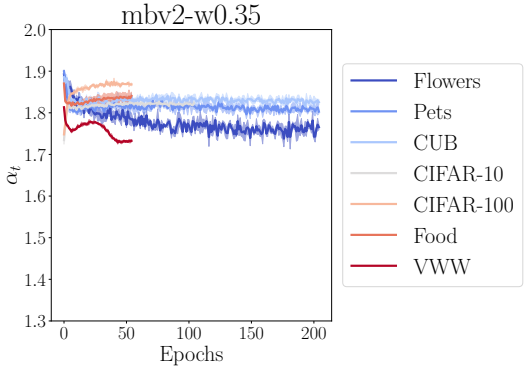



1. Stochastic gradients exhibit heavy-tailed behavior

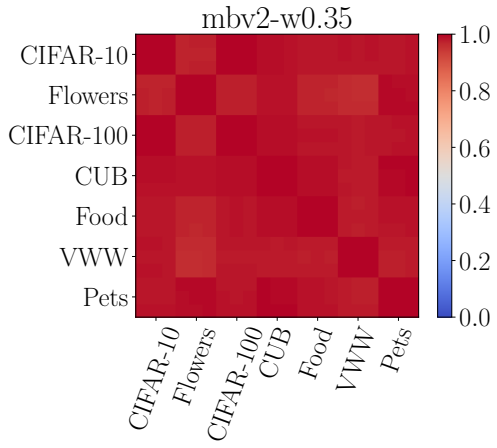
# Empirical Validation of Hypotheses

**Setup:** 3 architectures, 7 datasets, 3 memory budgets, 3 seeds

Measuring $\alpha$:




Layer proposition:

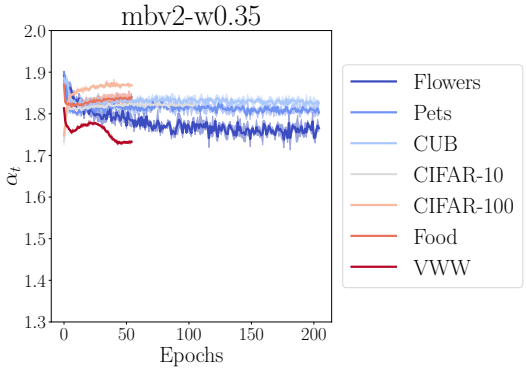



1. Stochastic gradients exhibit heavy-tailed behavior
2. Layer rankings show high Spearman correlation across tasks

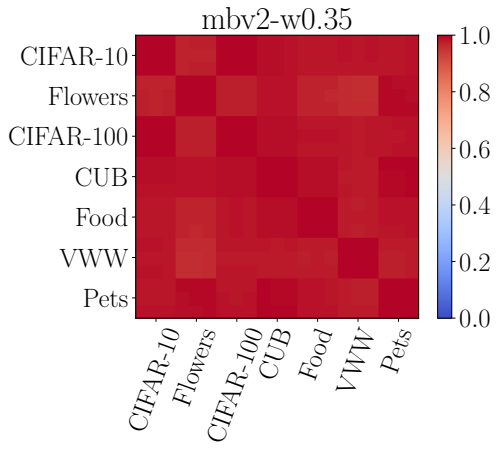
# Empirical Validation of Hypotheses

**Setup:** 3 architectures, 7 datasets, 3 memory budgets, 3 seeds

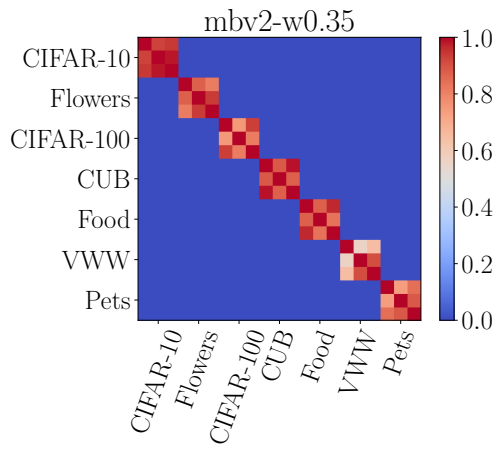
Measuring $\alpha$:




Layer proposition:

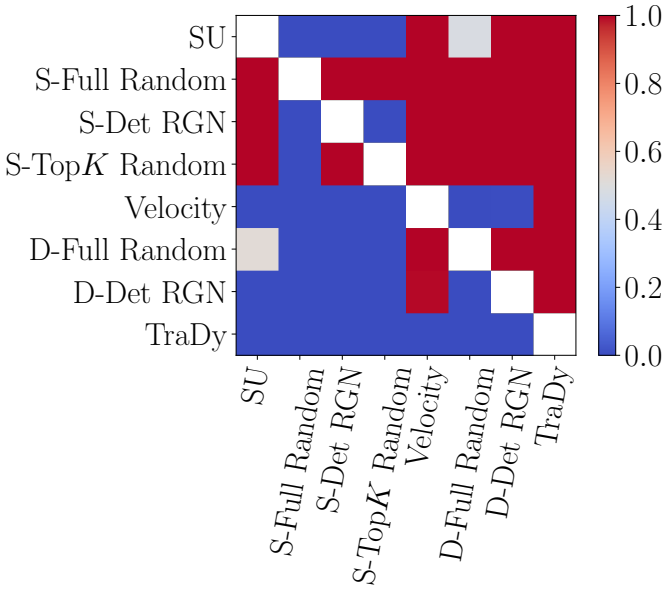



Channel proposition:

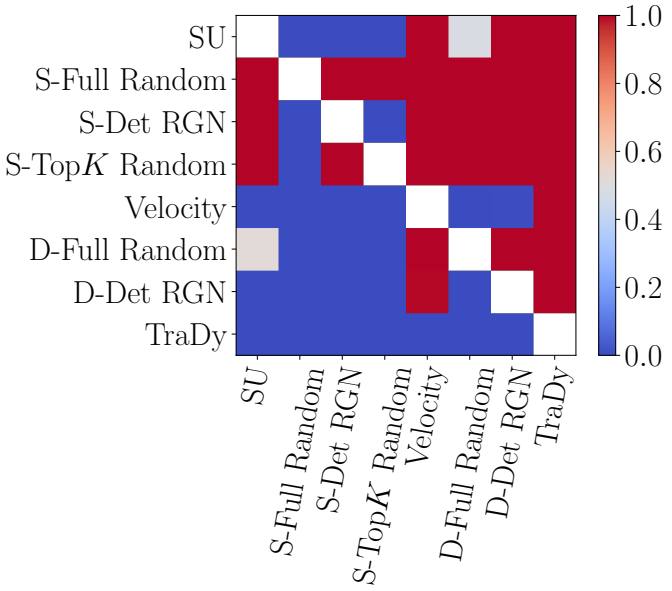


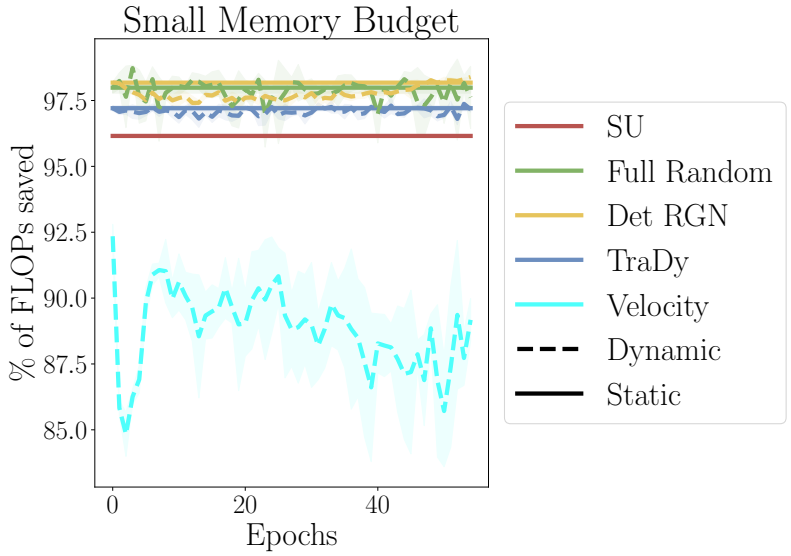

1. Stochastic gradients exhibit heavy-tailed behavior
2. Layer rankings show high Spearman correlation across tasks
3. T-test confirms channel gradient norms vary significantly between tasks

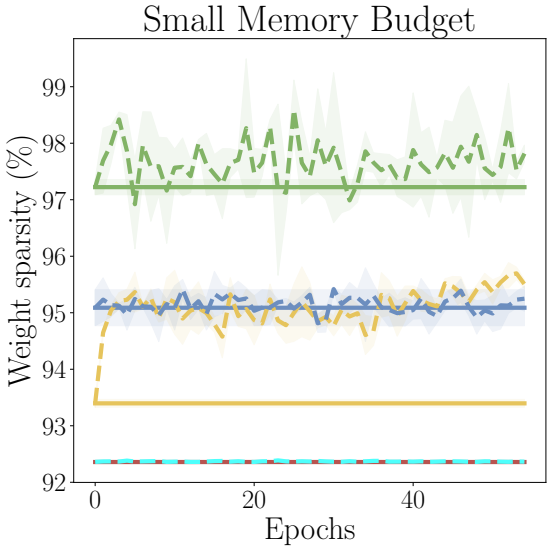
# Performance Analysis

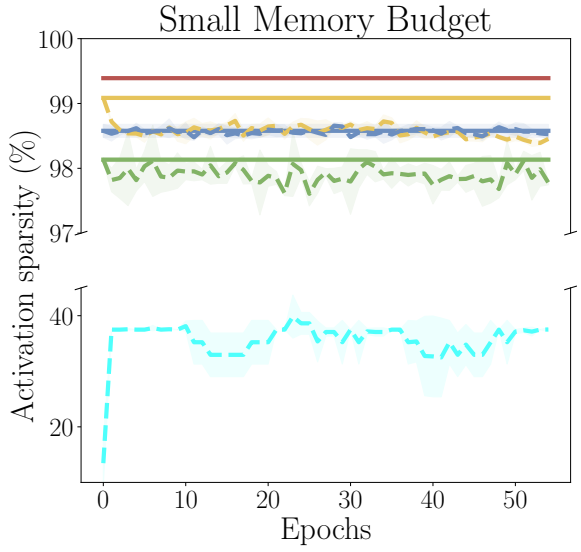


# Performance Analysis



# TraDy Limitations

1. Channel-centric layer importance metric

# TraDy Limitations

1. Channel-centric layer importance metric
   $\rightarrow$ Introduce **La**yer **Ra**nking (LaRa) for direct layer characterization

# TraDy Limitations

1. Channel-centric layer importance metric
   $\rightarrow$ Introduce **La**yer **Ra**nking (LaRa) for direct layer characterization
2. Fixed top-$K$ layer selection

# TraDy Limitations

1. Channel-centric layer importance metric
   → Introduce **La**yer **Ra**nking (LaRa) for direct layer characterization
2. Fixed top-$K$ layer selection
   → Budget-adaptive layer selection strategy

# TraDy Limitations

1. Channel-centric layer importance metric
   → Introduce **La**yer **Ra**nking (LaRa) for direct layer characterization
2. Fixed top-$K$ layer selection
   → Budget-adaptive layer selection strategy
3. No exploitation of channel importance patterns

# TraDy Limitations

1. Channel-centric layer importance metric
   → Introduce **La**yer **Ra**nking (LaRa) for direct layer characterization
2. Fixed top-$K$ layer selection
   → Budget-adaptive layer selection strategy
3. No exploitation of channel importance patterns
   → Leverage channel stability for probabilistic sampling

# Key Innovations

**Layer importance metric**

From $\text{RGN}_i$:

$$\frac{1}{(\Theta_{\text{space}})_i} \sum_{c=1}^{C} \left\| \left( \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \right)_c \right\|_2$$

# Key Innovations

## Layer importance metric

From $\text{RGN}_i$:

$$\frac{1}{(\Theta_{\text{space}})_i} \sum_{c=1}^{C} \left\| \left( \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \right)_c \right\|_2$$

To $\text{LaRa}_i$:

$$\frac{\left\| \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \right\|_2}{\sum_{c=1}^{C} \mathcal{C}_c^{\mathcal{W}_i} + \mathcal{C}_c^{\mathcal{A}_i}}$$

# Key Innovations

## Layer importance metric

From $\text{RGN}_i$:

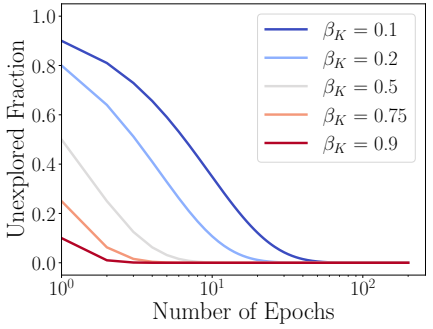
$$\frac{1}{(\Theta_{\text{space}})_i} \sum_{c=1}^{C} \left\| \left( \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \right)_c \right\|_2$$

To $\text{LaRa}_i$:

$$\frac{\left\| \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \right\|_2}{\sum_{c=1}^{C} \mathcal{C}_c^{\mathcal{W}_i} + \mathcal{C}_c^{\mathcal{A}_i}}$$

## Budget-adaptive layer selection

Introduction of $\beta_K = \frac{B_{\text{mem}}}{M_K}$:



# Key Innovations

## Layer importance metric

From $\text{RGN}_i$:

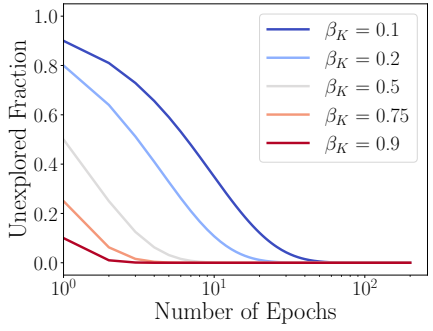
$$\frac{1}{(\Theta_{\text{space}})_i} \sum_{c=1}^{C} \left\| \left( \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \right)_c \right\|_2$$

To $\text{LaRa}_i$:

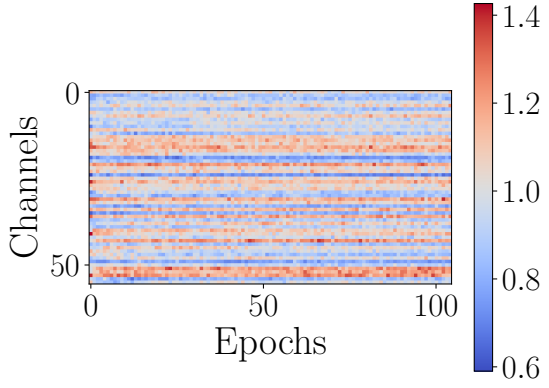
$$\frac{\left\| \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} \right\|_2}{\sum_{c=1}^{C} \mathcal{C}_c^{\mathcal{W}_i} + \mathcal{C}_c^{\mathcal{A}_i}}$$

## Budget-adaptive layer selection

Introduction of $\beta_K = \frac{B_{\text{mem}}}{M_K}$:

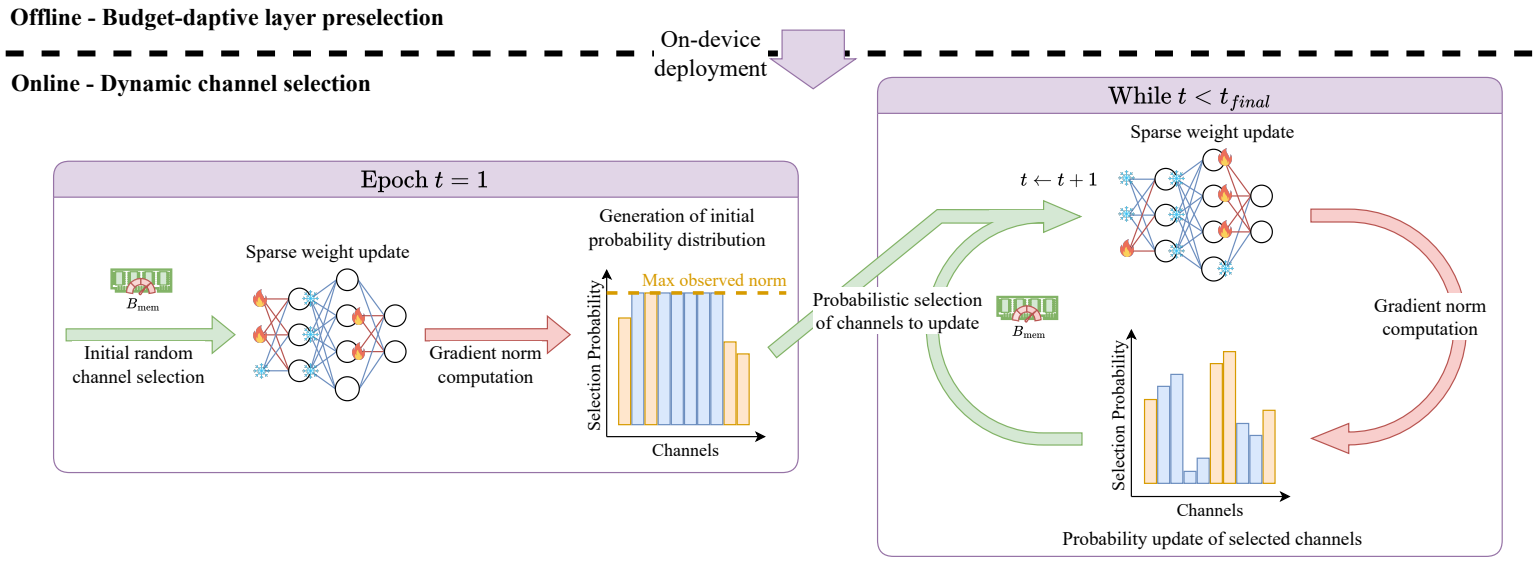



## Channel importance stability



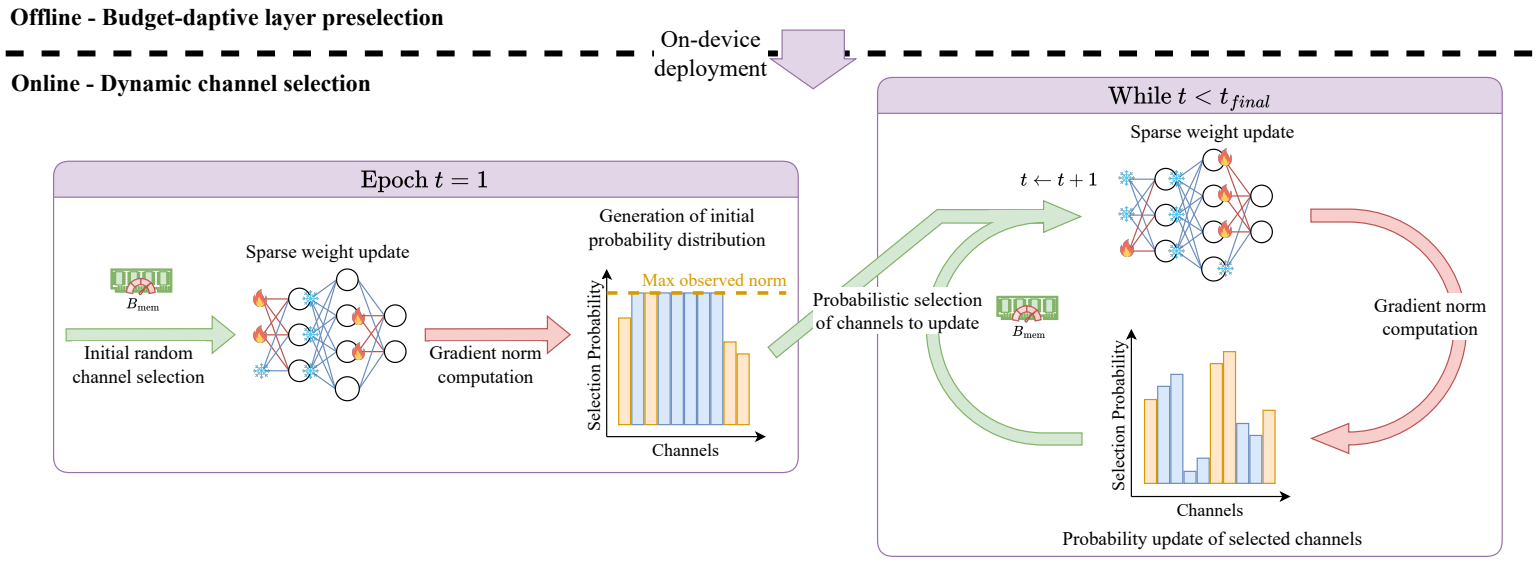
# Unpacking our Improved Algorithm

## Memory-constrained **Dy**namic subnetwork upd**ate** (MeDyate)



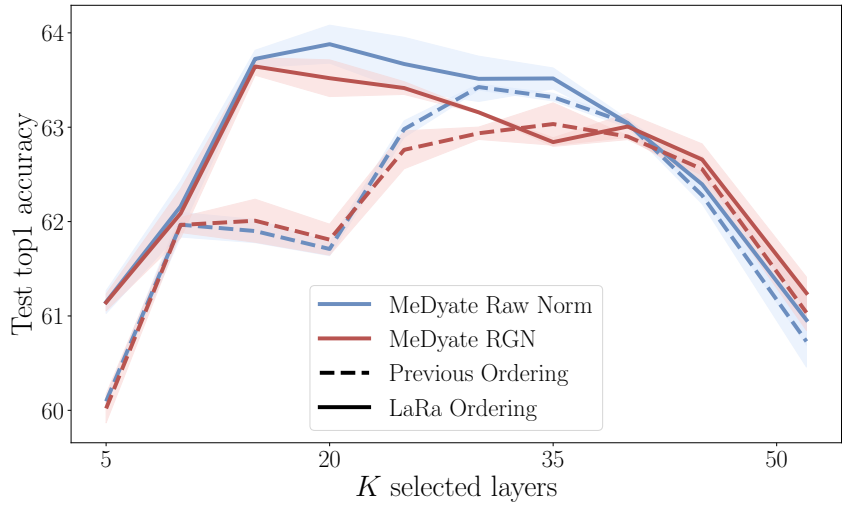
# Unpacking our Improved Algorithm

## Memory-constrained **Dy**namic subnetwork upd**ate** (MeDyate)

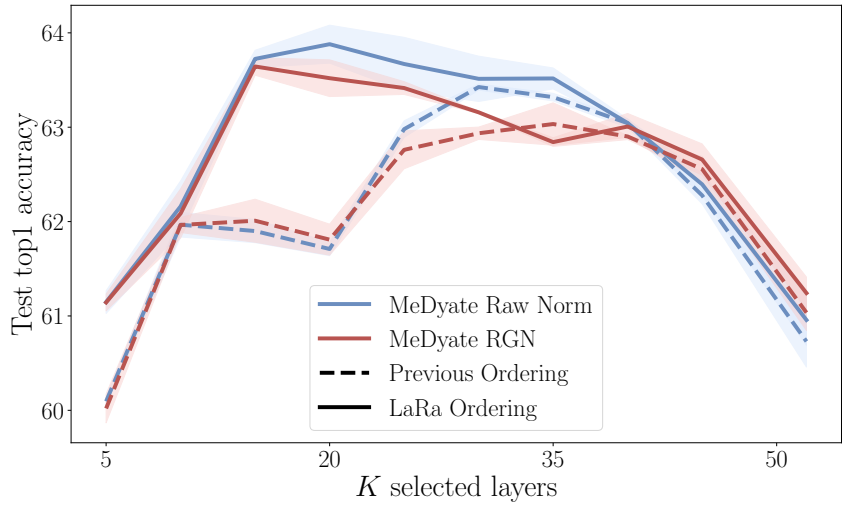


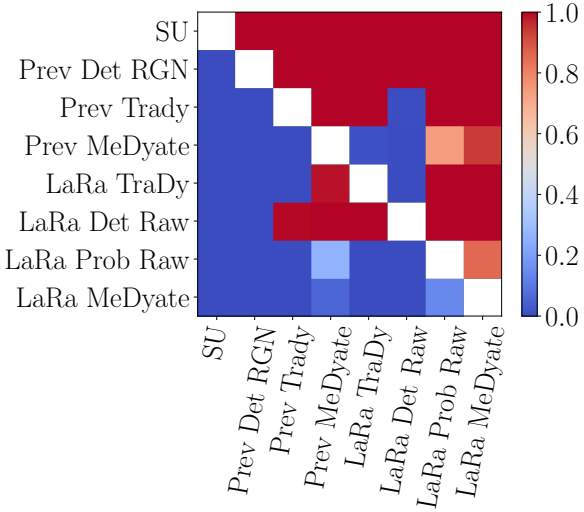

- Budget-adaptive layer selection → synergizes with exploration strategy
- Probabilistic sampling → enables importance-based selection

# Experimental Results

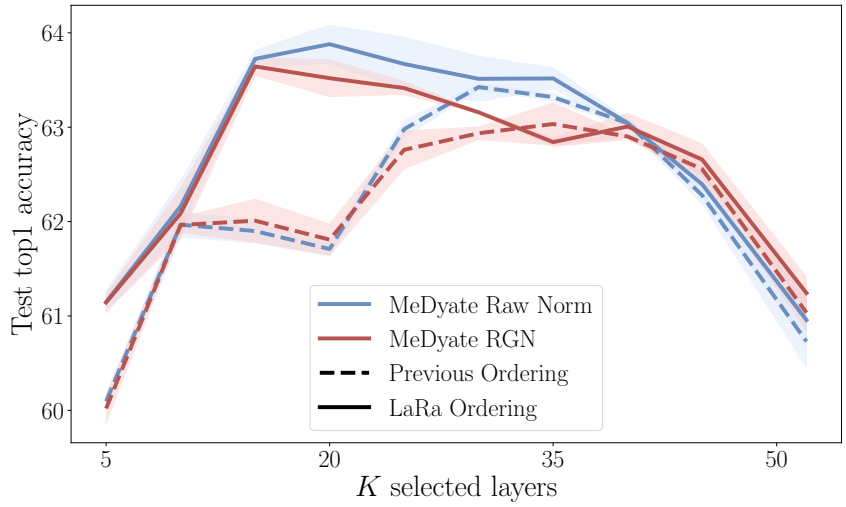


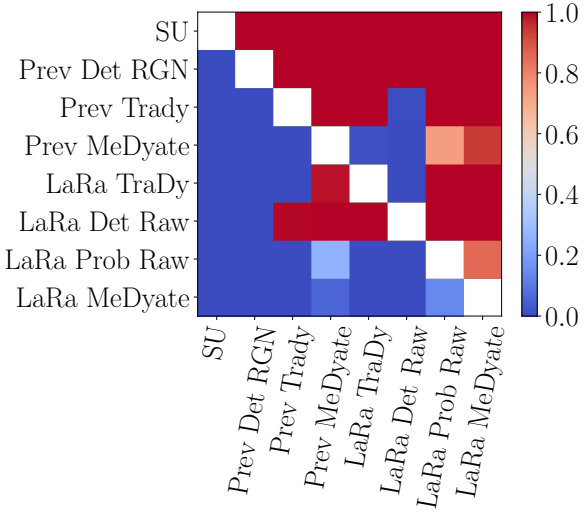
# Experimental Results



# Experimental Results




- LaRa enables more selective layer pruning
- Raw layer norm outperforms memory-normalized metric
- MeDyate matches full-gradient baseline performance

# Summary

## Main Contributions

# Summary

## Main Contributions

- Theoretical insights regarding network behavior during fine-tuning: heavy-tailed stochastic gradient, layer and channel gradient properties

# Summary

## Main Contributions

- Theoretical insights regarding network behavior during fine-tuning: heavy-tailed stochastic gradient, layer and channel gradient properties
- Adaptive algorithm based on LaRa selection and probabilistic channel sampling

# Summary

## Main Contributions

- Theoretical insights regarding network behavior during fine-tuning: heavy-tailed stochastic gradient, layer and channel gradient properties
- Adaptive algorithm based on LaRa selection and probabilistic channel sampling
- SOTA performance on CNNs

# Summary

## Main Contributions

- Theoretical insights regarding network behavior during fine-tuning: heavy-tailed stochastic gradient, layer and channel gradient properties
- Adaptive algorithm based on LaRa selection and probabilistic channel sampling
- SOTA performance on CNNs

## Key Limitations

# Summary

## Main Contributions

- Theoretical insights regarding network behavior during fine-tuning: heavy-tailed stochastic gradient, layer and channel gradient properties
- Adaptive algorithm based on LaRa selection and probabilistic channel sampling
- SOTA performance on CNNs

## Key Limitations

- Limited generalization to transformers $\rightarrow$ He et al. (2025) provide insights in LLM sparse fine-tuning

# Summary

## Main Contributions

- Theoretical insights regarding network behavior during fine-tuning: heavy-tailed stochastic gradient, layer and channel gradient properties
- Adaptive algorithm based on LaRa selection and probabilistic channel sampling
- SOTA performance on CNNs

## Key Limitations

- Limited generalization to transformers $\rightarrow$ He et al. (2025) provide insights in LLM sparse fine-tuning
- No on-device validation of practical feasibility and energy efficiency

# Publications

**Aël Quélennec**, Enzo Tartaglione, Pavlo Mozharovskyi and Van-Tam Nguyen. Towards on-device learning on the edge: Ways to select neurons to update under a budget constraint. In *WACV2024*.

**Aël Quélennec**, Nour Hezbri, Pavlo Mozharovskyi, Van-Tam Nguyen and Enzo Tartaglione. Study of training dynamics for memory-constrained fine-tuning. Under revision in *ICLR2026*.

**Aël Quélennec**, Pavlo Mozharovskyi, Van-Tam Nguyen and Enzo Tartaglione. Memory constrained dynamic subnetwork update for transfer learning. Under revision in *TMLR*.

# Outline

Introduction

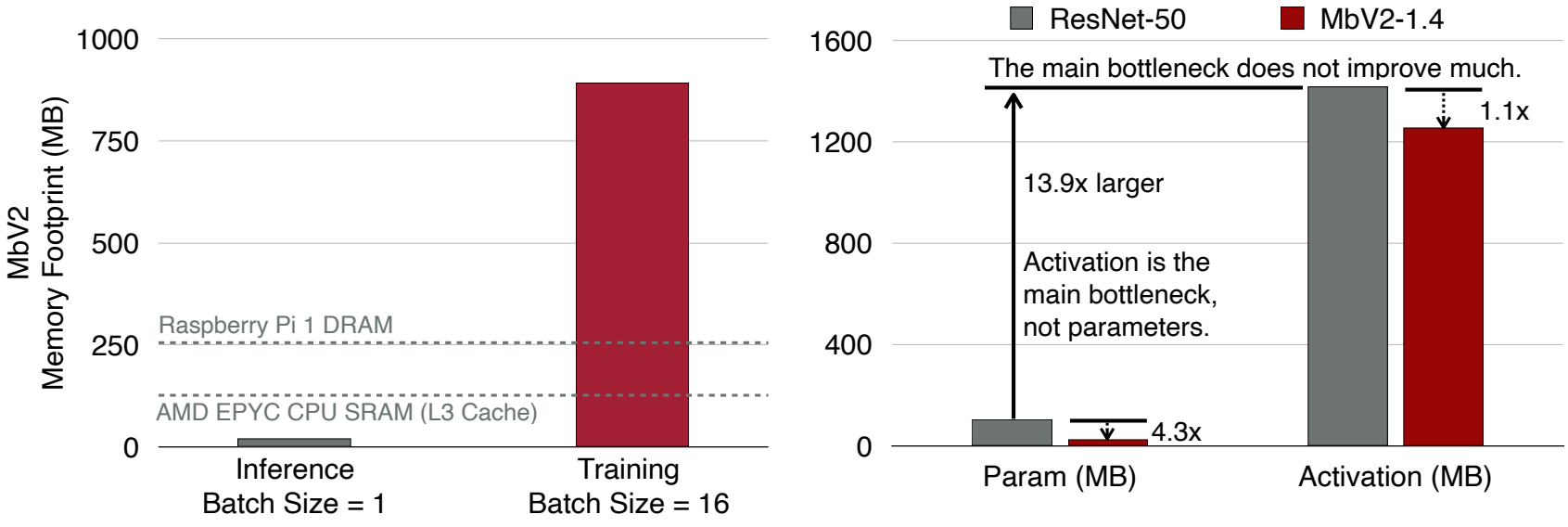
Subnetwork Selection for Fine-Tuning

Activation Map Compression

Conclusion

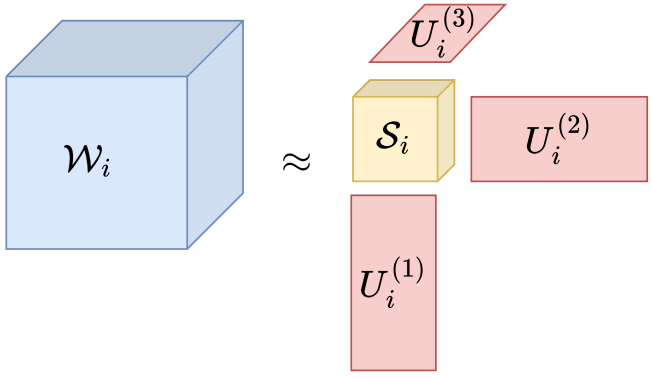
# Motivations

Cai et al. (2020) demonstrate that the main bottleneck during training corresponds to activation memory storage:



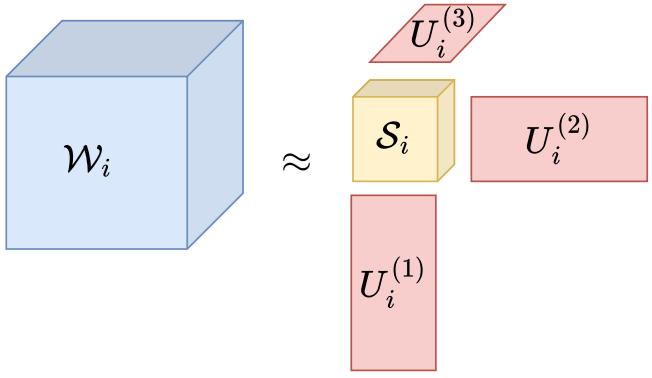
# Introducing Tensor Decomposition

Decomposes large tensors into products of smaller matrices:



# Introducing Tensor Decomposition

Decomposes large tensors into products of smaller matrices:

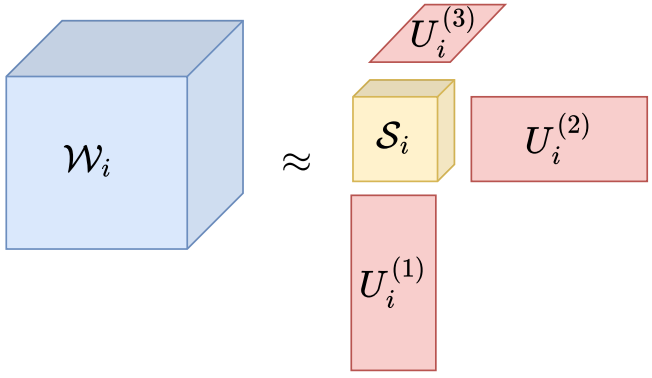



Advantages:

- Theoretically grounded
- Controllable compression-accuracy trade-off

# Introducing Tensor Decomposition

Decomposes large tensors into products of smaller matrices:




Advantages:

- Theoretically grounded
- Controllable compression-accuracy trade-off

Common methods in weight compression:

- SVD
- Canonical-Polyadic (CP)
- Generalized Kronecker Product (GKPD)
- Tucker Decomposition
- etc.

# Decomposing Activations

## Higher-Order Singular Value Decomposition (HOSVD)

# Decomposing Activations

## Higher-Order Singular Value Decomposition (HOSVD)

$\mathcal{A}_i \in \mathbb{R}^{B \times C_i \times H_i \times W_i}$ becomes:

$$\mathcal{A}_i = \mathcal{S}_i \times_1 U_i^{(1)} \times_2 U_i^{(2)} \times_3 U_i^{(3)} \times_4 U_i^{(4)}$$

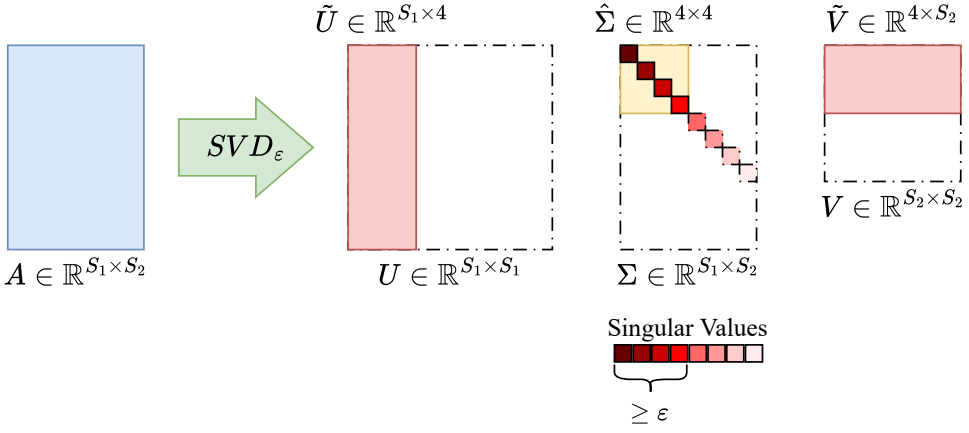
# Decomposing Activations

## Higher-Order Singular Value Decomposition (HOSVD)

$\mathcal{A}_i \in \mathbb{R}^{B \times C_i \times H_i \times W_i}$ becomes:

$$\mathcal{A}_i = \mathcal{S}_i \times_1 U_i^{(1)} \times_2 U_i^{(2)} \times_3 U_i^{(3)} \times_4 U_i^{(4)}$$

Explained variance $\varepsilon$ truncation:




$$\tilde{\mathcal{A}}_i = \hat{\mathcal{S}}_i \times_1 U_{i,J_{i,1}}^{(1)} \times_2 U_{i,J_{i,2}}^{(2)} \times_3 U_{i,J_{i,3}}^{(3)} \times_4 U_{i,J_{i,4}}^{(4)}$$

# Avoiding Recomposition

Injecting $\tilde{\mathcal{A}}_i$ in

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

results in a sequence of smaller convolutions

$\rightarrow$ potential for FLOPs reduction

# Avoiding Recomposition

Injecting $\tilde{\mathcal{A}}_i$ in

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

results in a sequence of smaller convolutions

$\rightarrow$ potential for FLOPs reduction

# Avoiding Recomposition

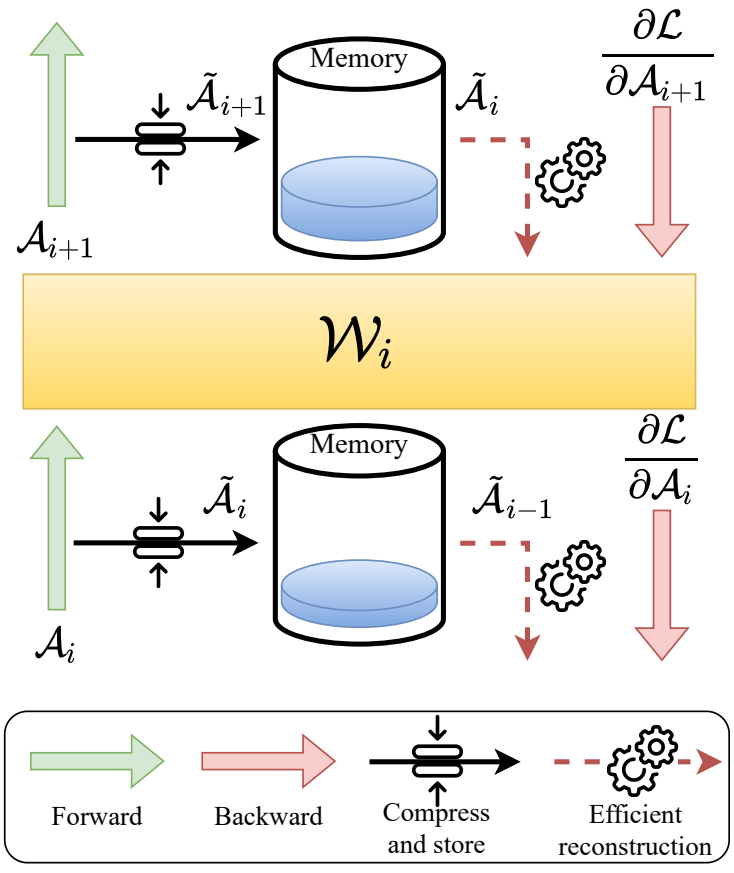
Injecting $\tilde{\mathcal{A}}_i$ in

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

results in a sequence of smaller convolutions

$\rightarrow$ potential for FLOPs reduction

Backpropagation becomes:



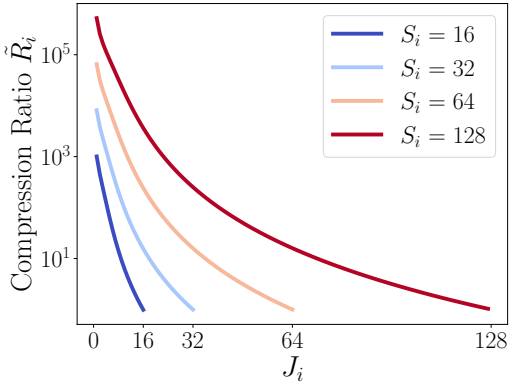
# Algorithm Properties

Assuming uniform spatial dimensions $S_i$:

# Algorithm Properties

Assuming uniform spatial dimensions $S_i$:

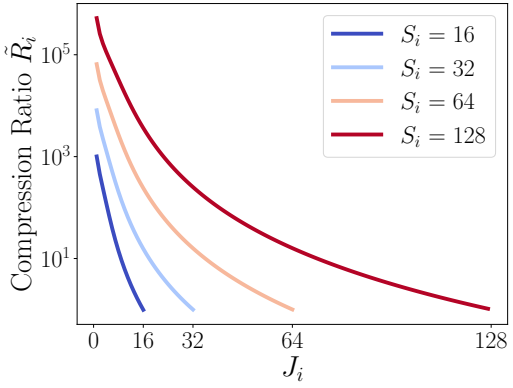
## Memory gain



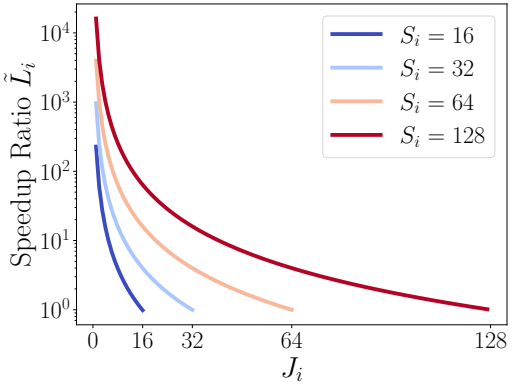
# Algorithm Properties

Assuming uniform spatial dimensions $S_i$:

## Memory gain



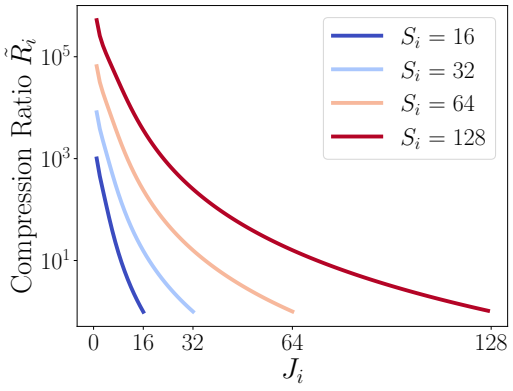

## Backward speedup



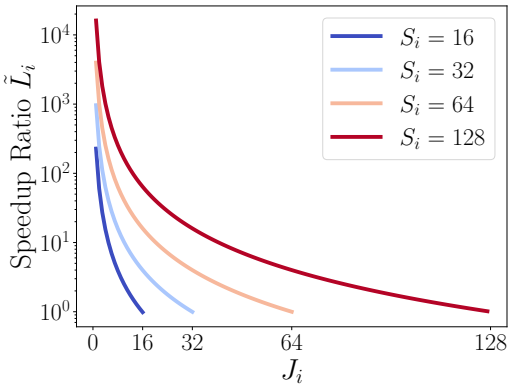
# Algorithm Properties

Assuming uniform spatial dimensions $S_i$:

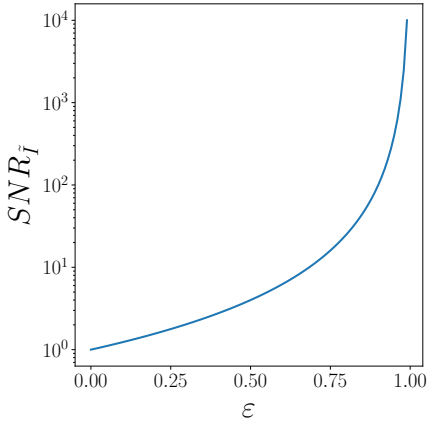
## Memory gain




## Backward speedup



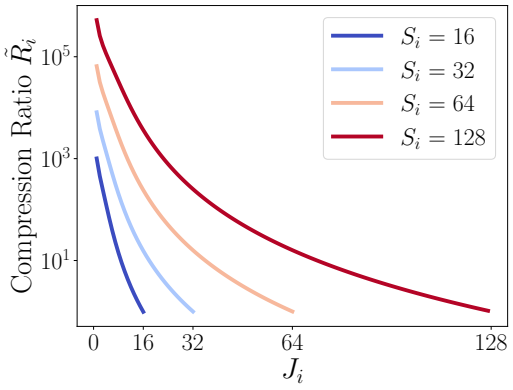

## Error analysis



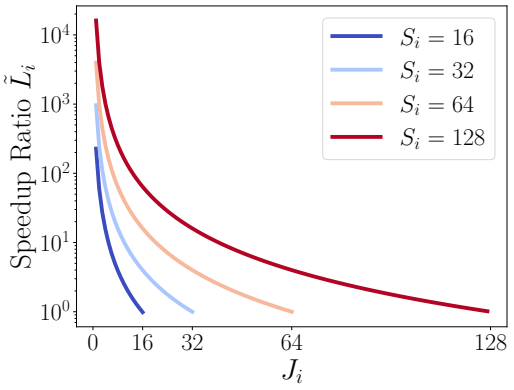
# Algorithm Properties

Assuming uniform spatial dimensions $S_i$:

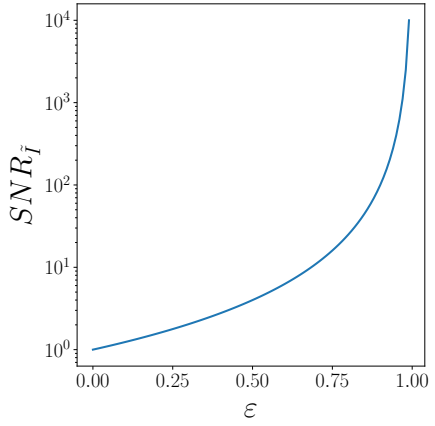
## Memory gain




## Backward speedup




## Error analysis




$\rightarrow$ Significant memory reduction and backward pass acceleration potential

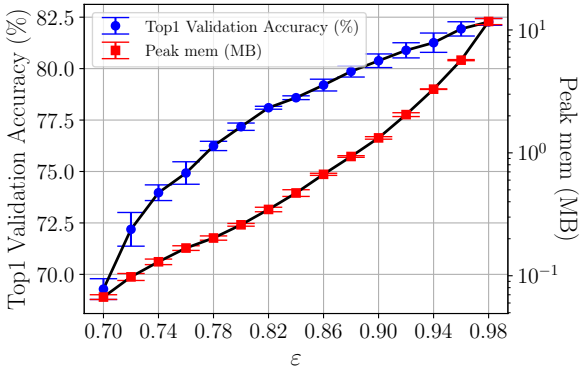
$\rightarrow$ Favorable accuracy-compression trade-off expected

# Explained Variance Analysis

Fine-tuning last 4 layers of MCUNet on CIFAR-10 using HOSVD:

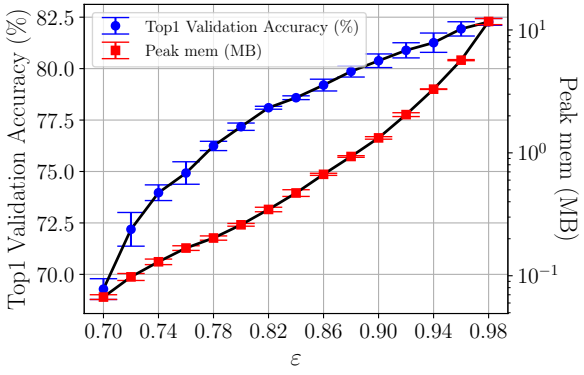
# Explained Variance Analysis

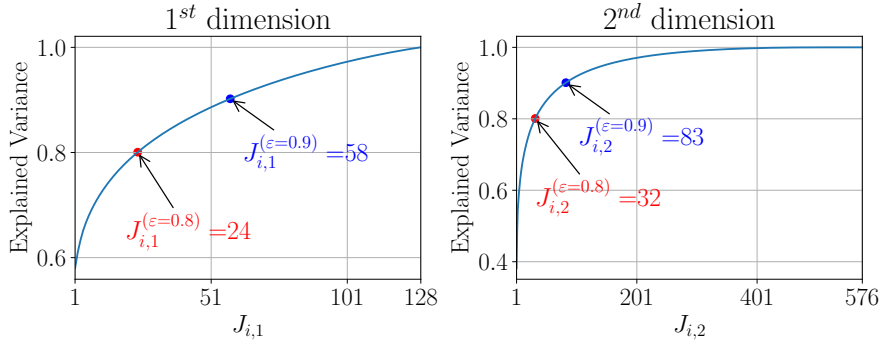
Fine-tuning last 4 layers of MCUNet on CIFAR-10 using HOSVD:



# Explained Variance Analysis

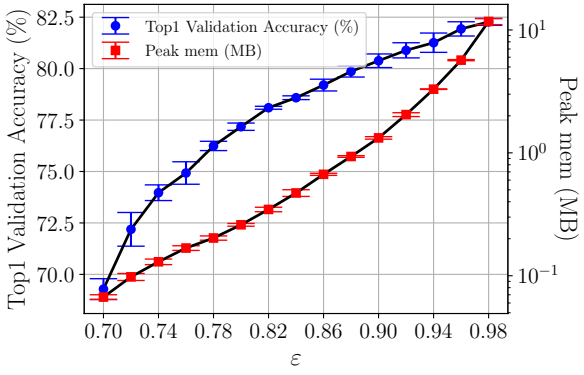
Fine-tuning last 4 layers of MCUNet on CIFAR-10 using HOSVD:

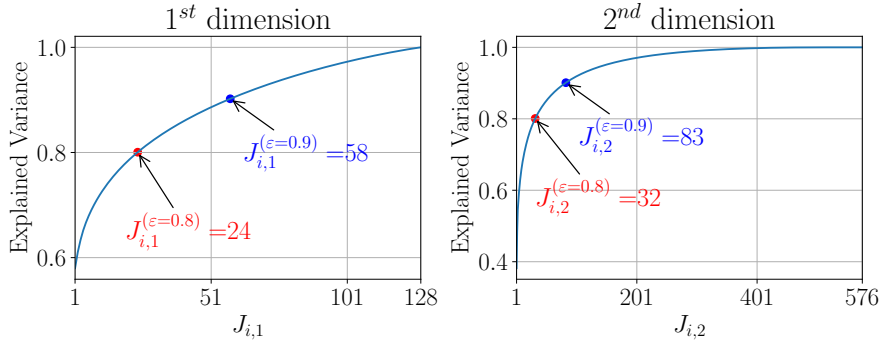


# Explained Variance Analysis

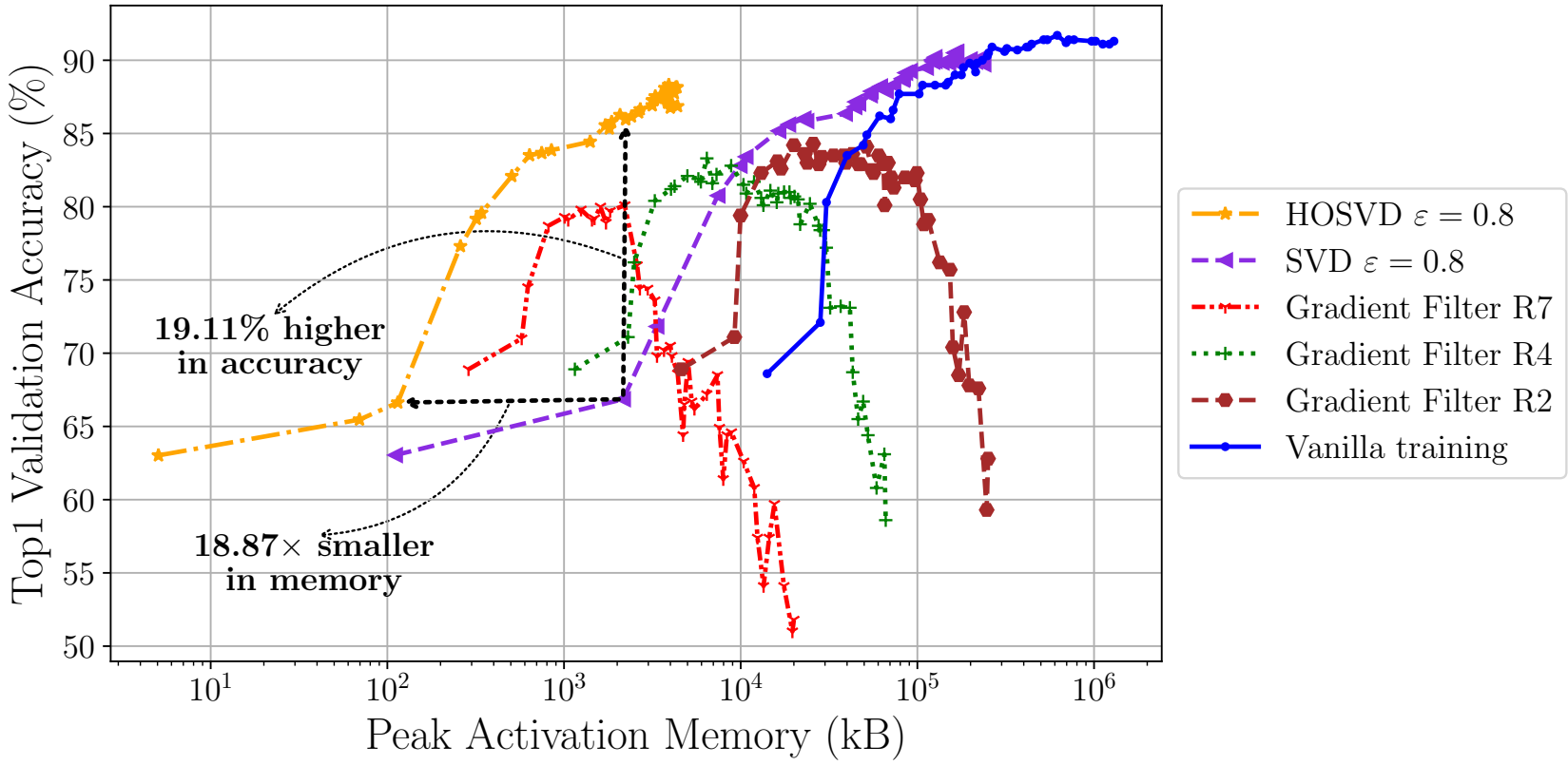
Fine-tuning last 4 layers of MCUNet on CIFAR-10 using HOSVD:



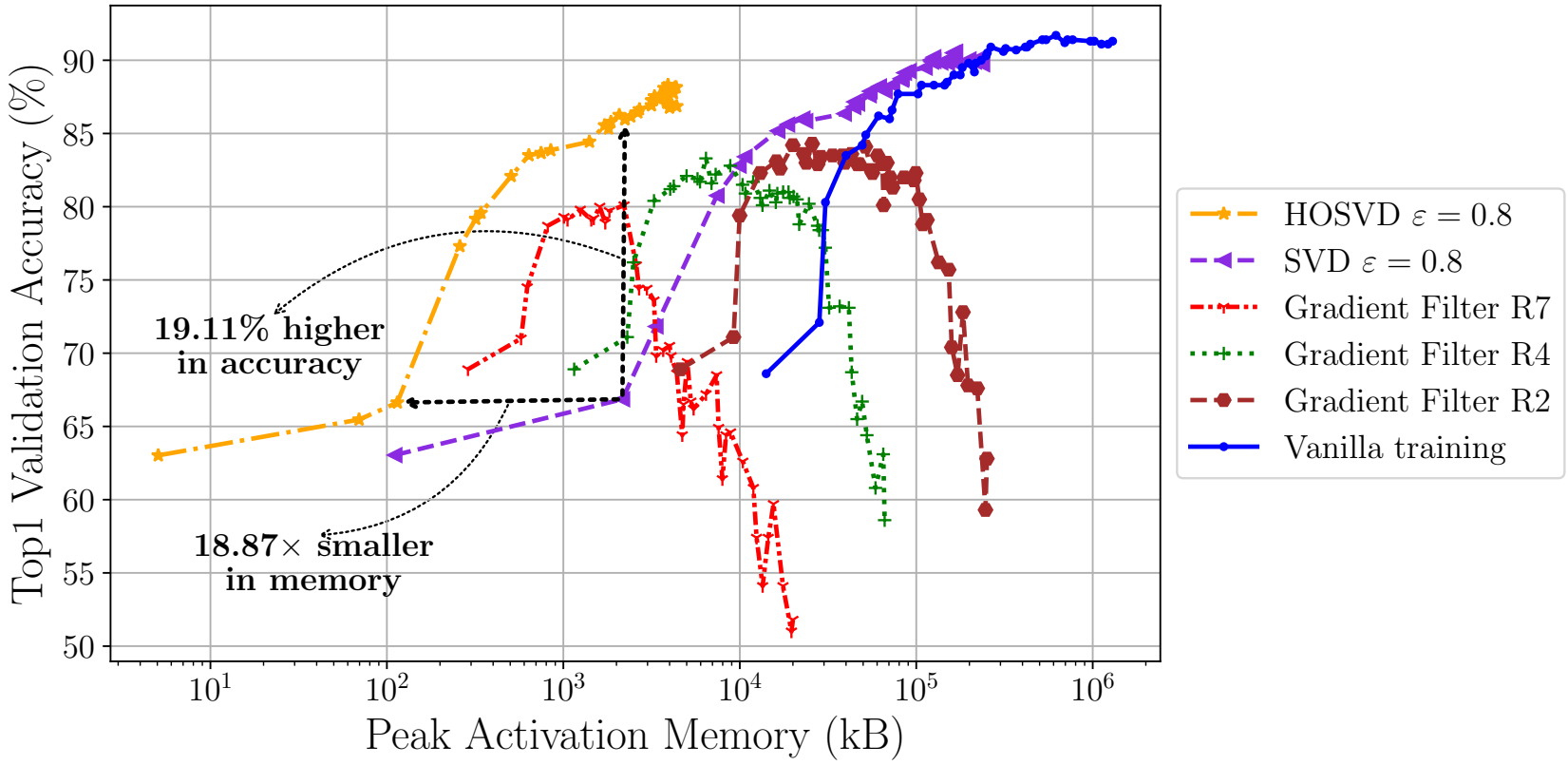




Variance concentrated in few singular values $\rightarrow$ experiments conducted with $\varepsilon = 0.8$ and $\varepsilon = 0.9$

# Experimental Results



# Experimental Results



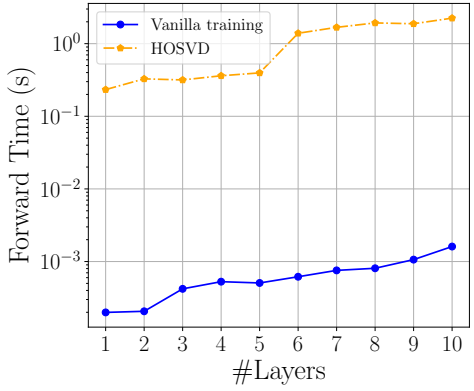

Cumulative compressed activation memory $<$ Single uncompressed layer memory

# Limitations

Comparing latency of Vanilla and HOSVD training:

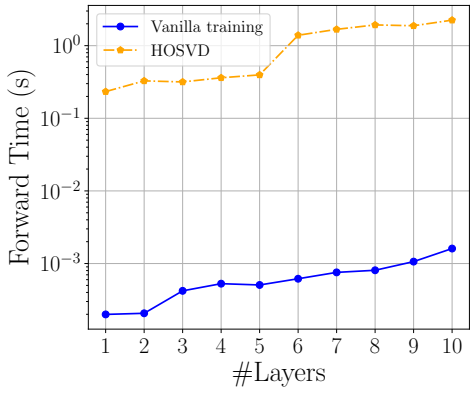
# Limitations

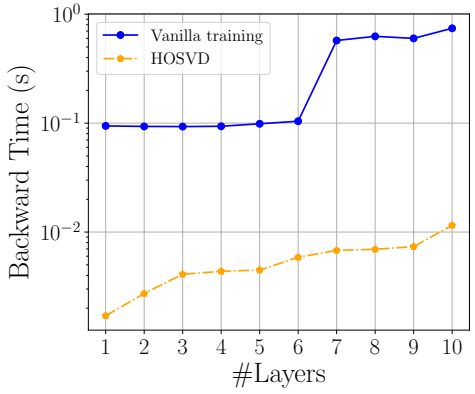
Comparing latency of Vanilla and HOSVD training:



# Limitations

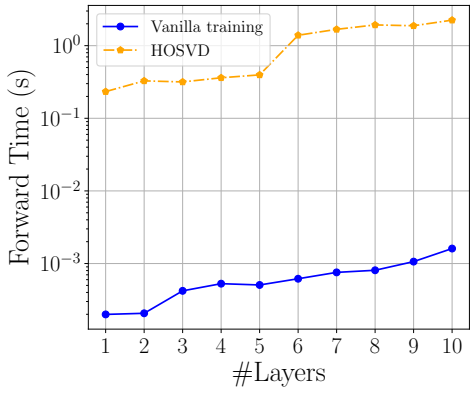
Comparing latency of Vanilla and HOSVD training:

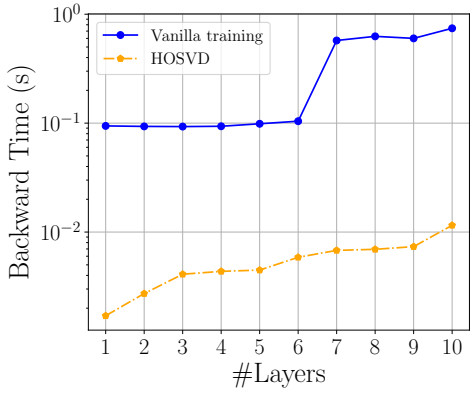


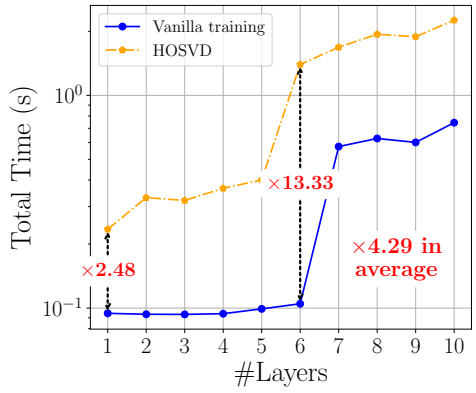



# Limitations

Comparing latency of Vanilla and HOSVD training:

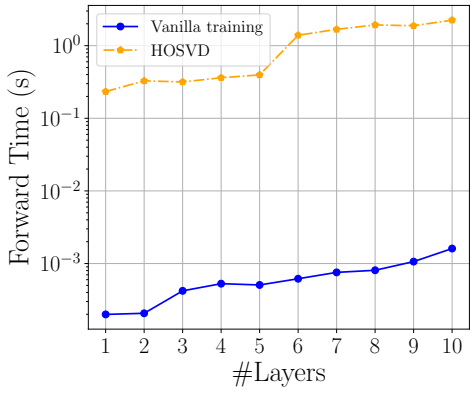


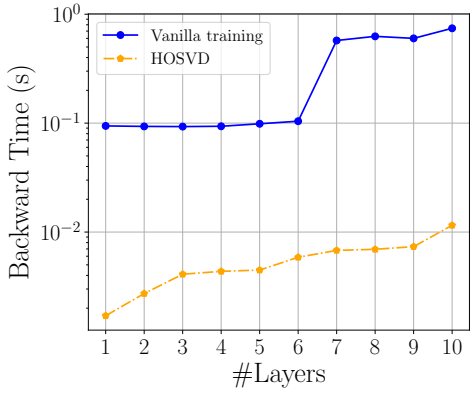



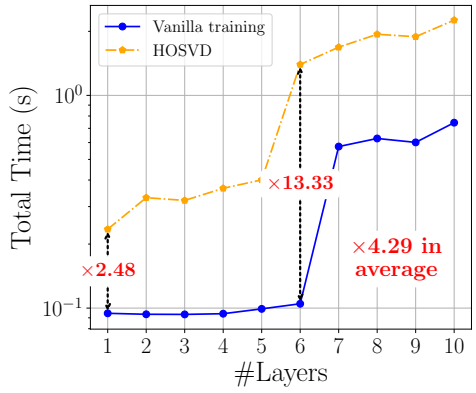



# Limitations

Comparing latency of Vanilla and HOSVD training:




+ No control over peak memory

# Subspace Iteration

**Algorithm 1:** Subspace Iteration with Warm-start at epoch $t$

**Require:** Epoch index $t$, gradient matrix $M^{(t)} \in \mathbb{R}^{m \times n}$, target rank $r \in [1, \min(m, n)]$.

**if** $t = 0$ **then**
  Initialize $Q^{(t)} \in \mathbb{R}^{n \times r}$ from an i.i.d standard normal distribution.
**else**
  Warm-start initialization, $Q^{(t)} \leftarrow Q^{(t-1)}$.
$P^{(t)} \leftarrow M^{(t)} Q^{(t)}$.
$\hat{P}^{(t)} \leftarrow \text{Orthogonalize}(P^{(t)})$.
$Q^{(t)} \leftarrow M^{(t)^T} \hat{P}^{(t)}$.
**Return** $\hat{P}^{(t)}, Q^{(t)}$.

# Perplexity Search and Rank Selection

- Standard perplexity measures model output degradation from weight compression

# Perplexity Search and Rank Selection

- Standard perplexity measures model output degradation from weight compression
- Given a set of explained variance thresholds $\mathcal{E} \in ]0, 1]^E$:

# Perplexity Search and Rank Selection

- Standard perplexity measures model output degradation from weight compression
- Given a set of explained variance thresholds $\mathcal{E} \in ]0, 1]^E$:
  1. Forward: generate $\mathcal{A}_i$ and compressed $(\tilde{\mathcal{A}}_i)_e$ via HOSVD

# Perplexity Search and Rank Selection

- Standard perplexity measures model output degradation from weight compression
- Given a set of explained variance thresholds $\mathcal{E} \in ]0,1]^E$:
  1. Forward: generate $\mathcal{A}_i$ and compressed $(\tilde{\mathcal{A}}_i)_e$ via HOSVD
  2. Backward: compute gradient discrepancy as activation perplexity:

$$\mathcal{P}_{i,e} = \left\| \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} - \left( \widetilde{\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i}} \right)_e \right\|_F$$

# Perplexity Search and Rank Selection

- Standard perplexity measures model output degradation from weight compression
- Given a set of explained variance thresholds $\mathcal{E} \in ]0,1]^E$:
  1. Forward: generate $\mathcal{A}_i$ and compressed $(\tilde{\mathcal{A}}_i)_e$ via HOSVD
  2. Backward: compute gradient discrepancy as activation perplexity:

$$\mathcal{P}_{i,e} = \left\| \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} - \left( \widetilde{\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i}} \right)_e \right\|_F$$

- Output: $\mathcal{P} \in \mathbb{R}^{N \times E}$ and $\mathcal{R} \in \mathbb{R}^{N \times E \times 4}$ encode layer-threshold-rank relationships

# Perplexity Search and Rank Selection

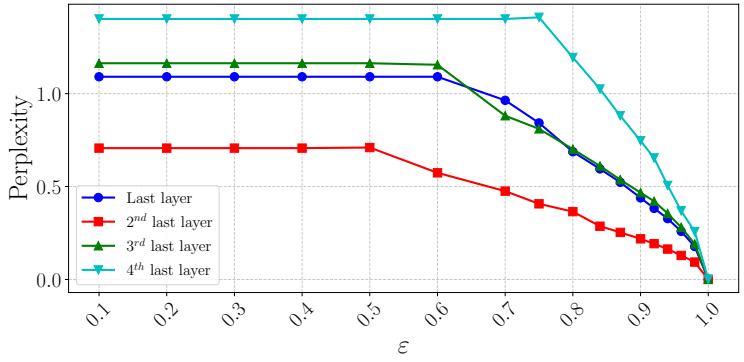
- Standard perplexity measures model output degradation from weight compression
- Given a set of explained variance thresholds $\mathcal{E} \in ]0,1]^E$:
  1. Forward: generate $\mathcal{A}_i$ and compressed $(\tilde{\mathcal{A}}_i)_e$ via HOSVD
  2. Backward: compute gradient discrepancy as activation perplexity:

$$\mathcal{P}_{i,e} = \left\| \frac{\partial \mathcal{L}}{\partial \mathcal{W}_i} - \left( \widetilde{\frac{\partial \mathcal{L}}{\partial \mathcal{W}_i}} \right)_e \right\|_F$$

- Output: $\mathcal{P} \in \mathbb{R}^{N \times E}$ and $\mathcal{R} \in \mathbb{R}^{N \times E \times 4}$ encode layer-threshold-rank relationships
- Enables budget-constrained rank selection via optimization (recursive backtracking algorithm)

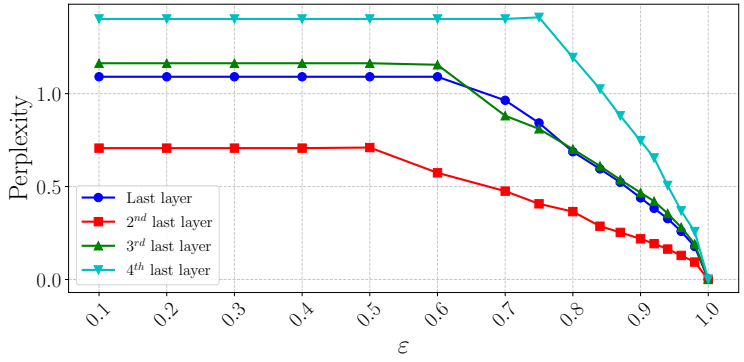
# Preliminary Experiments

## Perplexity variations:



# Preliminary Experiments

## Perplexity variations:

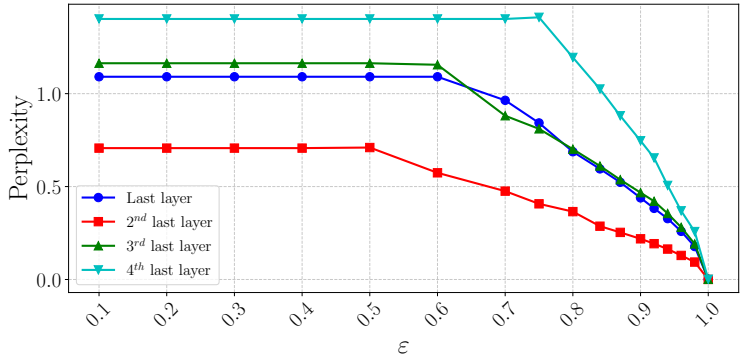



$\rightarrow$ Saturation below $\varepsilon = 0.5$

$\rightarrow$ Layer-dependent compressibility

# Preliminary Experiments

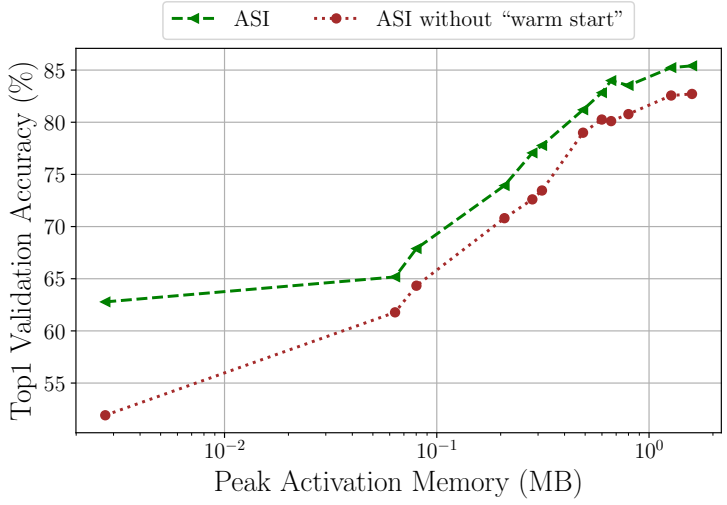
## Perplexity variations:




→ Saturation below $\varepsilon = 0.5$

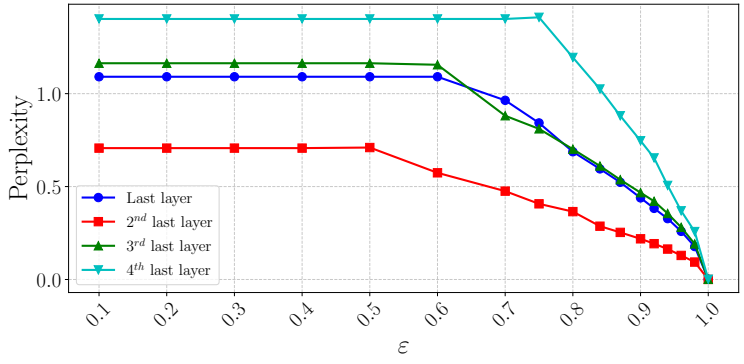
→ Layer-dependent compressibility

## Warm-start ablation:



# Preliminary Experiments

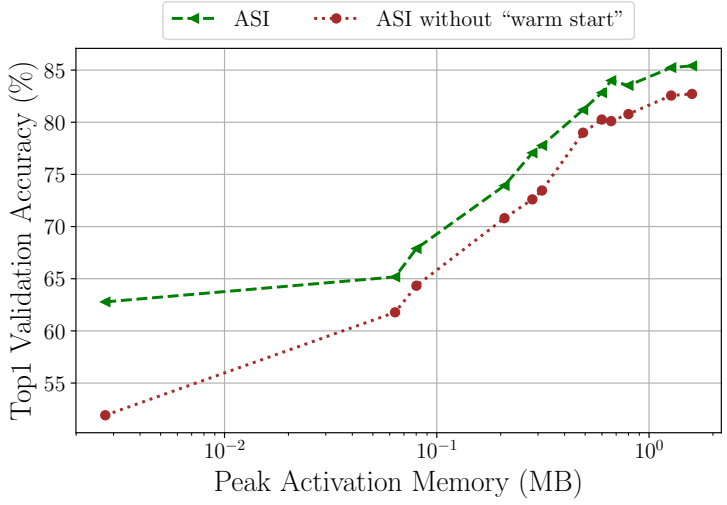
## Perplexity variations:




→ Saturation below $\varepsilon = 0.5$

→ Layer-dependent compressibility

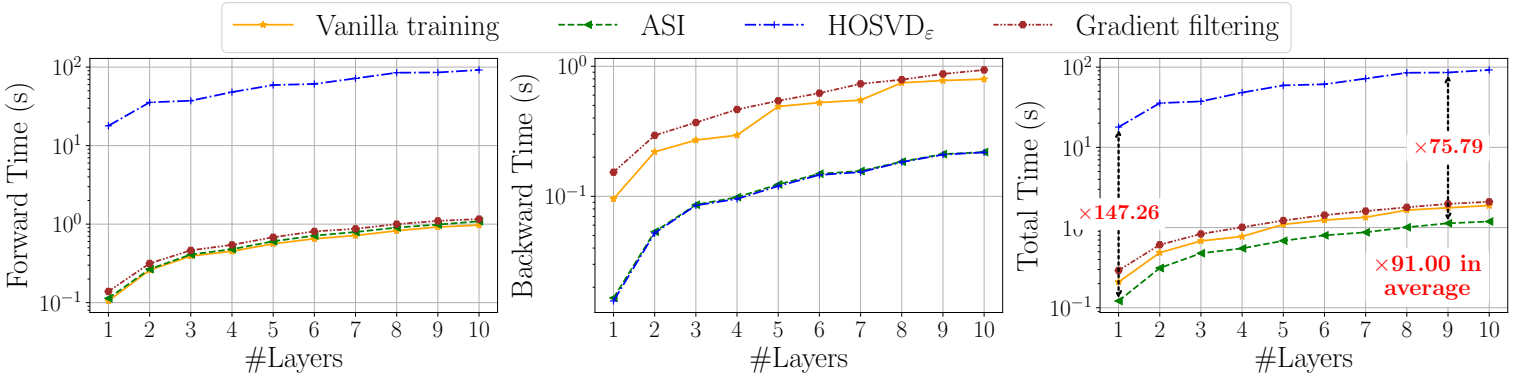
## Warm-start ablation:



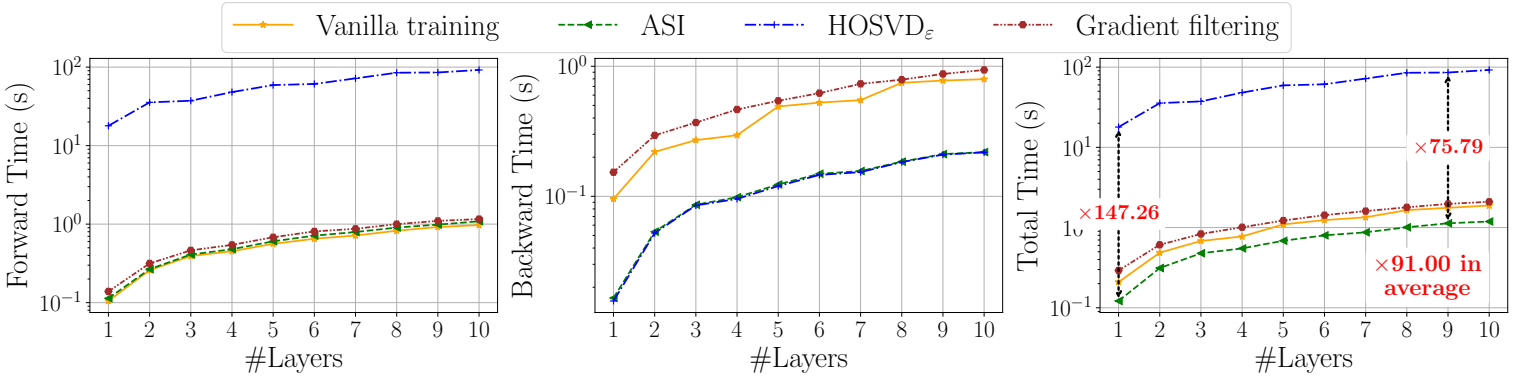

→ Warm-start provides clear improvement

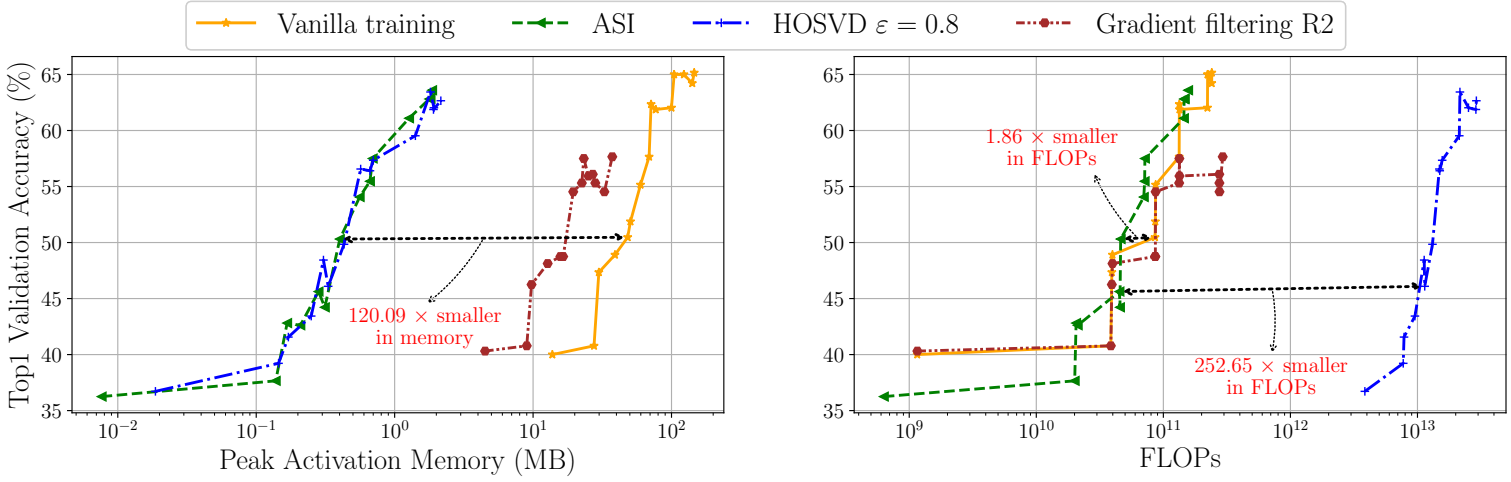
→ Acceptable accuracy without warm-start

# Validation Experiments



# Validation Experiments



# Summary

## Main Contributions:

# Summary

**Main Contributions:**

- Tensor decomposition framework for activation compression

# Summary

**Main Contributions:**

- Tensor decomposition framework for activation compression
- Perplexity-based rank selection with explicit memory budget control

# Summary

**Main Contributions:**

- Tensor decomposition framework for activation compression
- Perplexity-based rank selection with explicit memory budget control
- Warm-start subspace iteration exploiting temporal stability

# Summary

**Main Contributions:**

- Tensor decomposition framework for activation compression
- Perplexity-based rank selection with explicit memory budget control
- Warm-start subspace iteration exploiting temporal stability
- 120× memory reduction, 1.86× speedup, 91× latency improvement

**Future Directions:**

# Summary

**Main Contributions:**

- Tensor decomposition framework for activation compression
- Perplexity-based rank selection with explicit memory budget control
- Warm-start subspace iteration exploiting temporal stability
- 120× memory reduction, 1.86× speedup, 91× latency improvement

**Future Directions:**

- Extension to LLMs and transformer architectures

# Summary

**Main Contributions:**

- Tensor decomposition framework for activation compression
- Perplexity-based rank selection with explicit memory budget control
- Warm-start subspace iteration exploiting temporal stability
- 120× memory reduction, 1.86× speedup, 91× latency improvement

**Future Directions:**

- Extension to LLMs and transformer architectures
- Mixed-precision iteration for computational acceleration

# Summary

**Main Contributions:**

- Tensor decomposition framework for activation compression
- Perplexity-based rank selection with explicit memory budget control
- Warm-start subspace iteration exploiting temporal stability
- 120× memory reduction, 1.86× speedup, 91× latency improvement

**Future Directions:**

- Extension to LLMs and transformer architectures
- Mixed-precision iteration for computational acceleration
- Integration with pruning and quantization

# Publications

Le-Trung Nguyen, **Aël Quélennec**, Enzo Tartaglione, Samuel Tardieu and Van-Tam Nguyen. Activation map compression through tensor decomposition for deep learning. In *NeurIPS2024*.

Le-Trung Nguyen, **Aël Quélennec**, Van-Tam Nguyen, and Enzo Tartaglione. Beyond low-rank decomposition: A shortcut approach for efficient on-device learning. In *ICML2025*.

# Outline

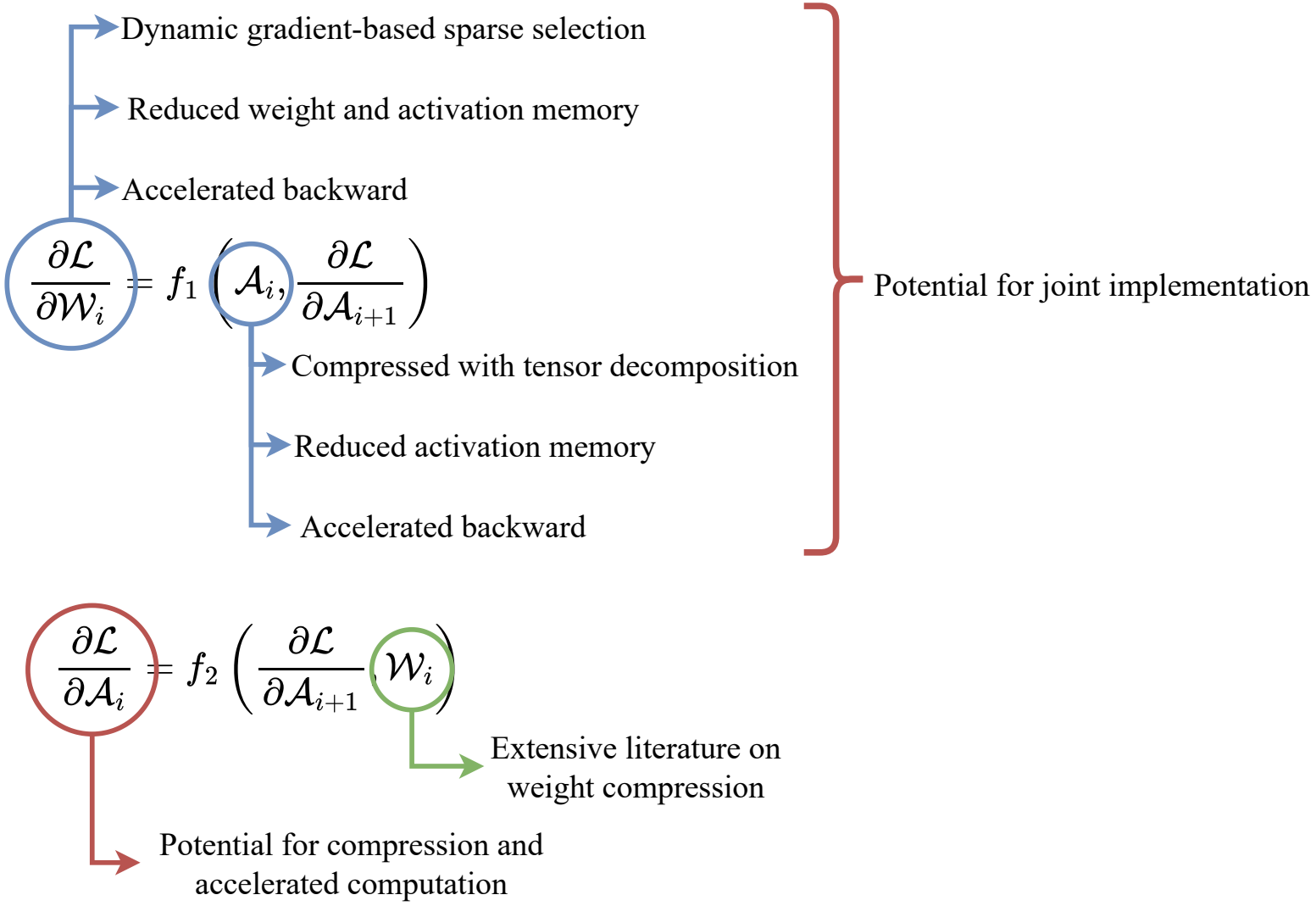
Introduction

Subnetwork Selection for Fine-Tuning

Activation Map Compression

Conclusion

# Contributions and Future Directions (1/2)



# Contributions and Future Directions (2/2)

# Contributions and Future Directions (2/2)

- **Transformer adaptation:** Extend framework to LLMs and vision transformers

# Contributions and Future Directions (2/2)

- **Transformer adaptation:** Extend framework to LLMs and vision transformers
- **Activation derivative compression:** Apply methods to efficient loss backpropagation

# Contributions and Future Directions (2/2)

- **Transformer adaptation:** Extend framework to LLMs and vision transformers
- **Activation derivative compression:** Apply methods to efficient loss backpropagation
- **Continual learning:** Leverage adaptive sampling for data drift handling

# Contributions and Future Directions (2/2)

- **Transformer adaptation:** Extend framework to LLMs and vision transformers
- **Activation derivative compression:** Apply methods to efficient loss backpropagation
- **Continual learning:** Leverage adaptive sampling for data drift handling
- **Training from scratch:** Adapt methods beyond fine-tuning for full network lifetime

# Contributions and Future Directions (2/2)

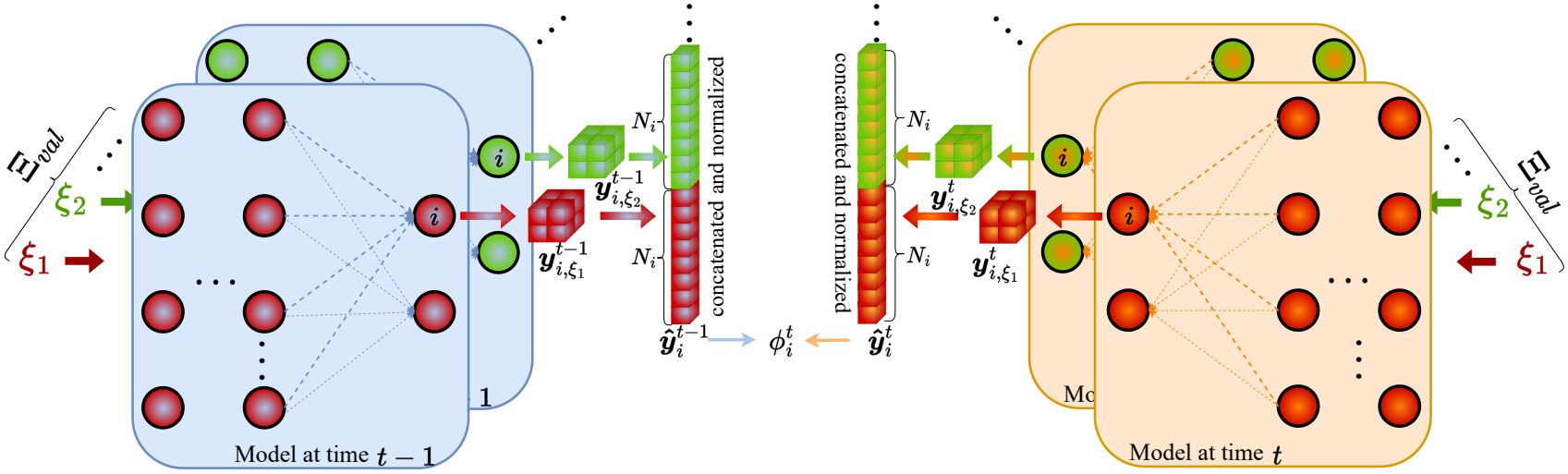
- **Transformer adaptation:** Extend framework to LLMs and vision transformers
- **Activation derivative compression:** Apply methods to efficient loss backpropagation
- **Continual learning:** Leverage adaptive sampling for data drift handling
- **Training from scratch:** Adapt methods beyond fine-tuning for full network lifetime
- **Green AI:** Reduce energy footprint and support sustainability objectives

# Outline

## Appendix

# Velocity's inspiration: NEq

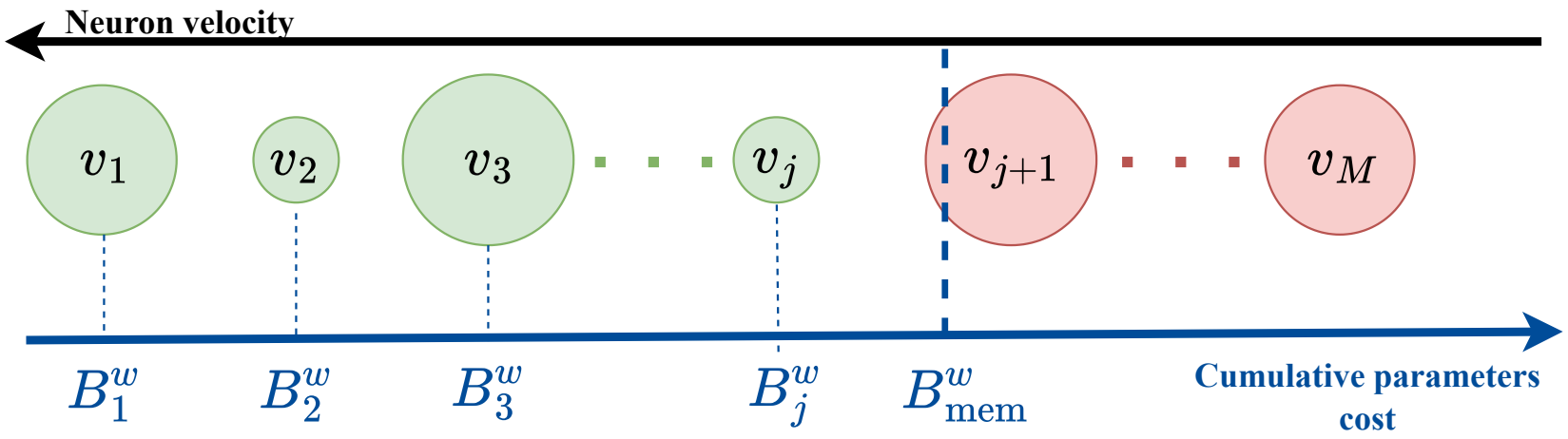
Bragagnolo et al. (2022)'s Neurons at Equilibrium




Velocity computation:

$$\left|v_i^t\right| < \varepsilon, \qquad \varepsilon \geq 0$$

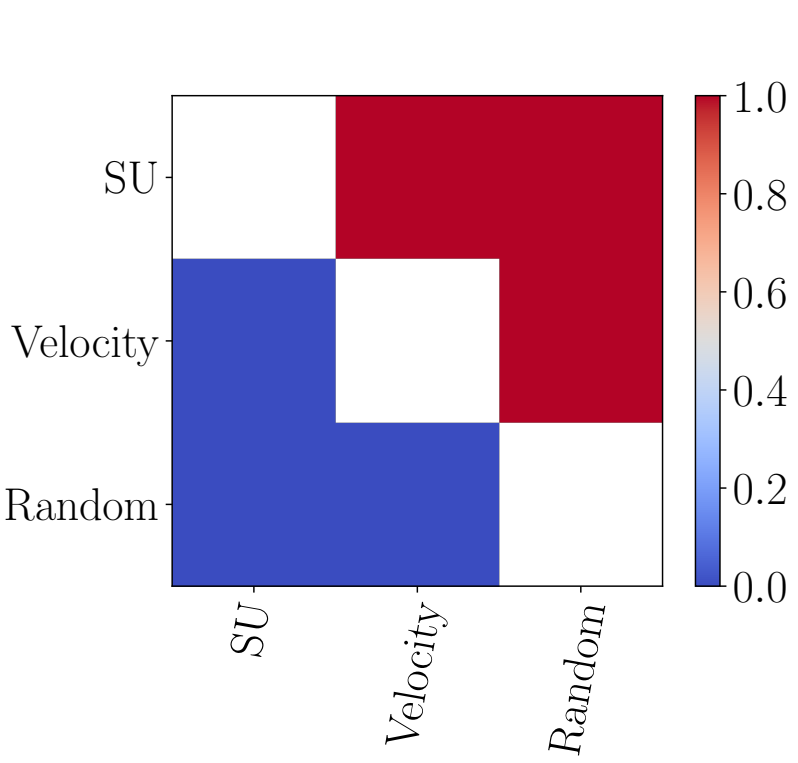
# Memory-Constrained Adaptation

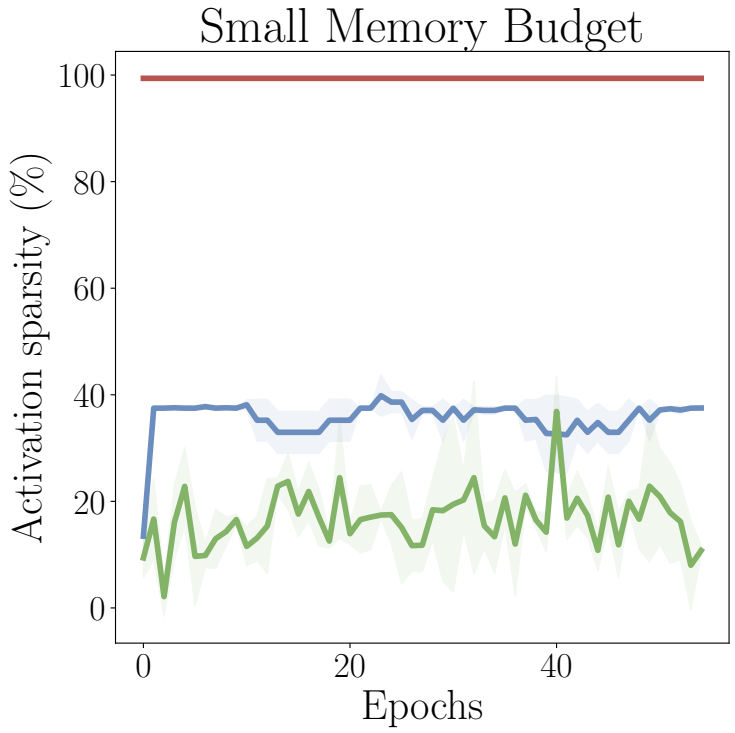



Introduction of memory normalization:

$$\tilde{v}_i = \frac{v_i}{C_i^{\text{w}}}$$

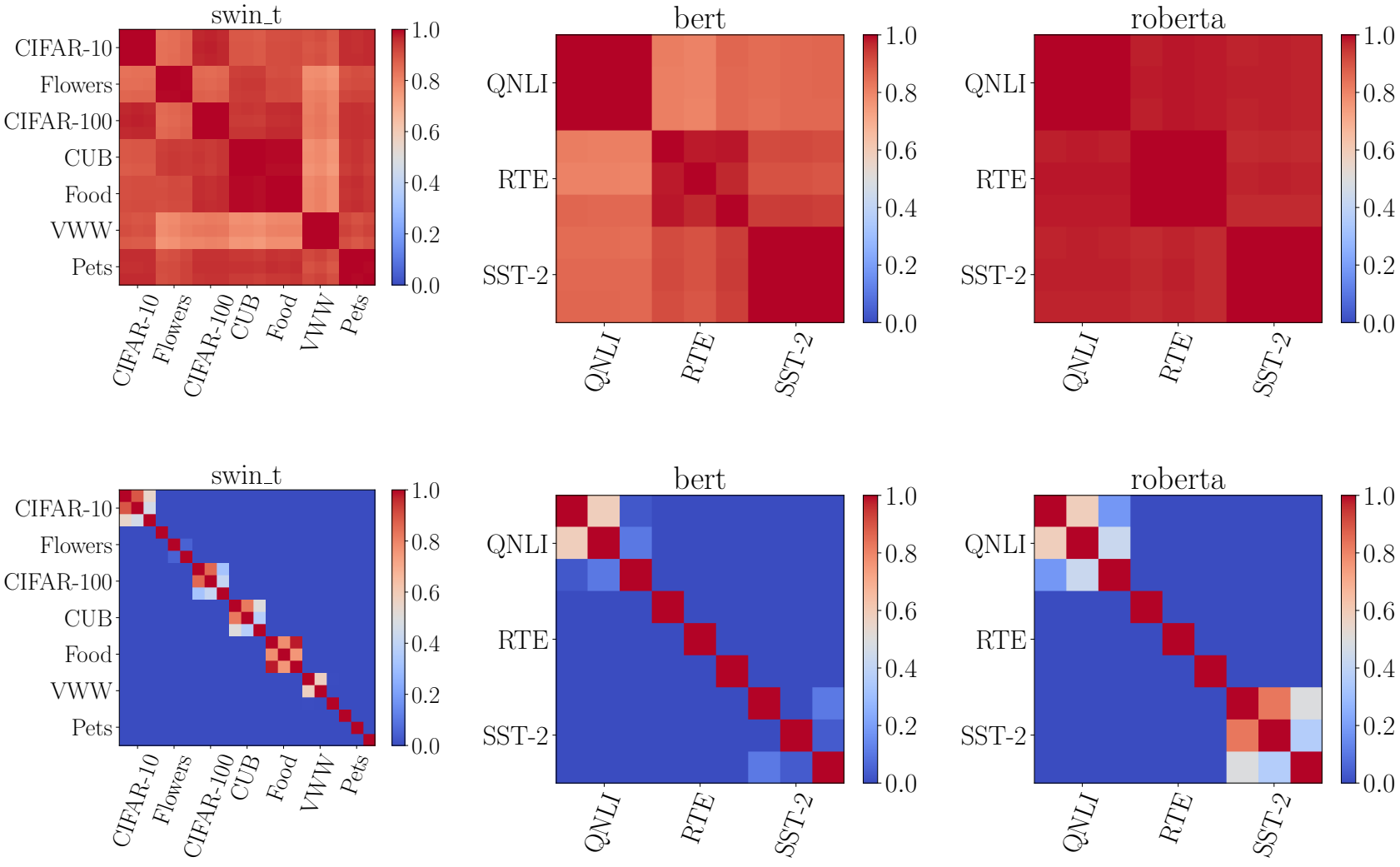
# Experimental Results



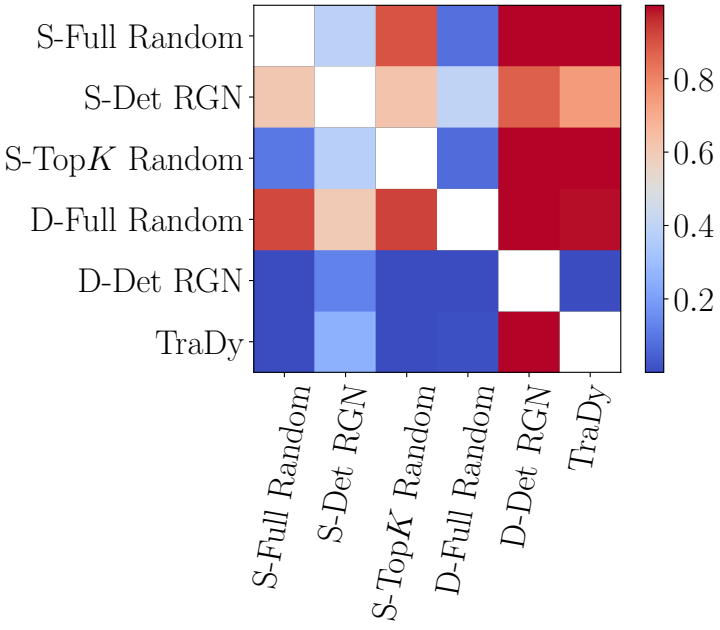




- Outperformed by random strategy
- Poor activation sparsity levels
- Relies on existence of a validation set
- Necessitates storage of all neurons activation

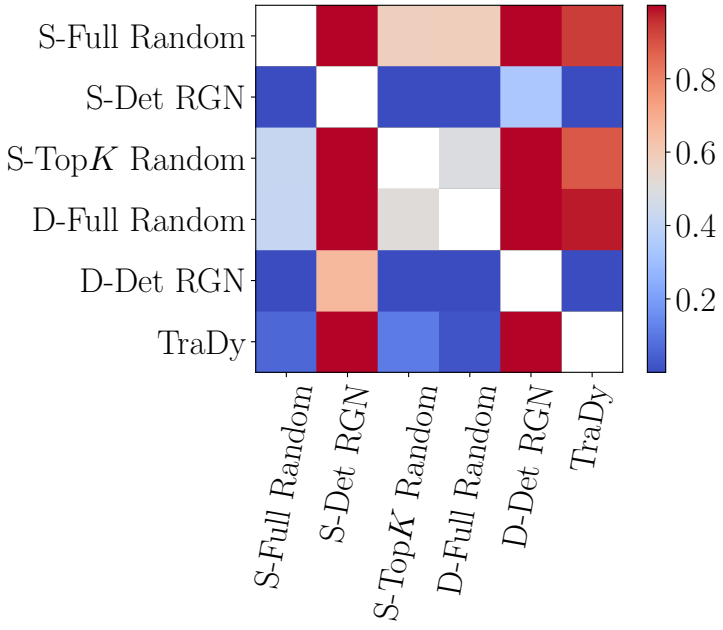
# Proposition Validation for Transformers



# TraDy Ablation for Transformers



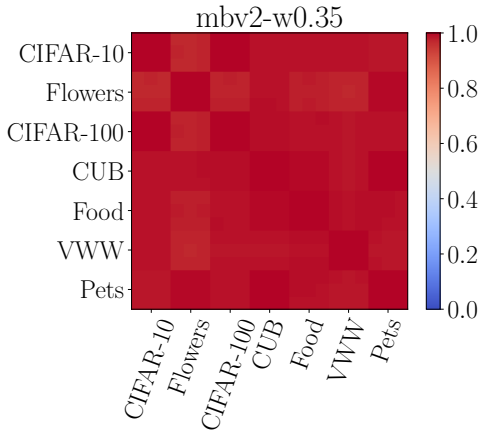

SwinT model

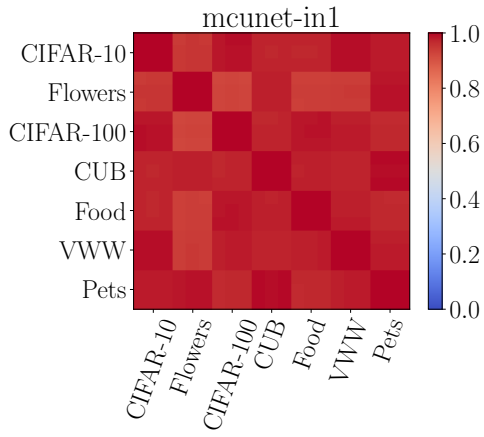


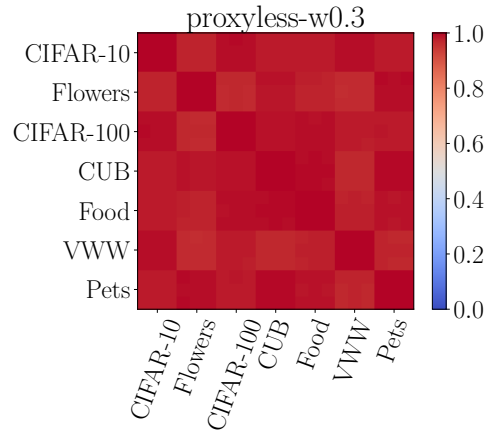

BERTs models

# MeDyate Proposition Validation

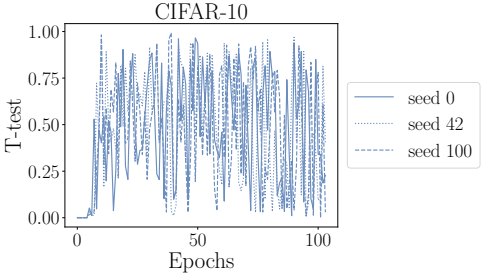
## LaRa Spearman:

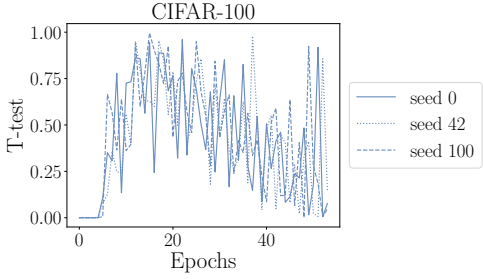


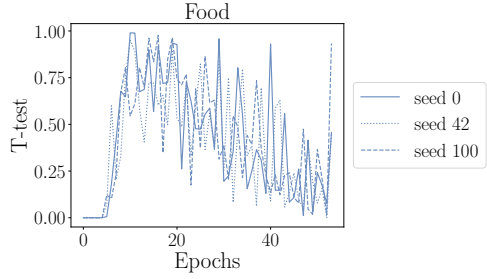




## Channel gradient norm inter-epoch T-test:



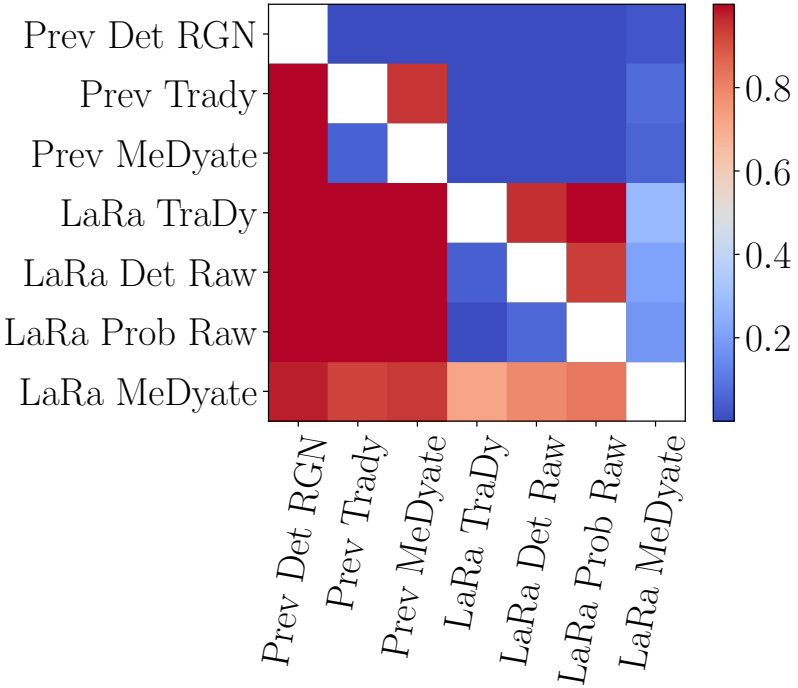



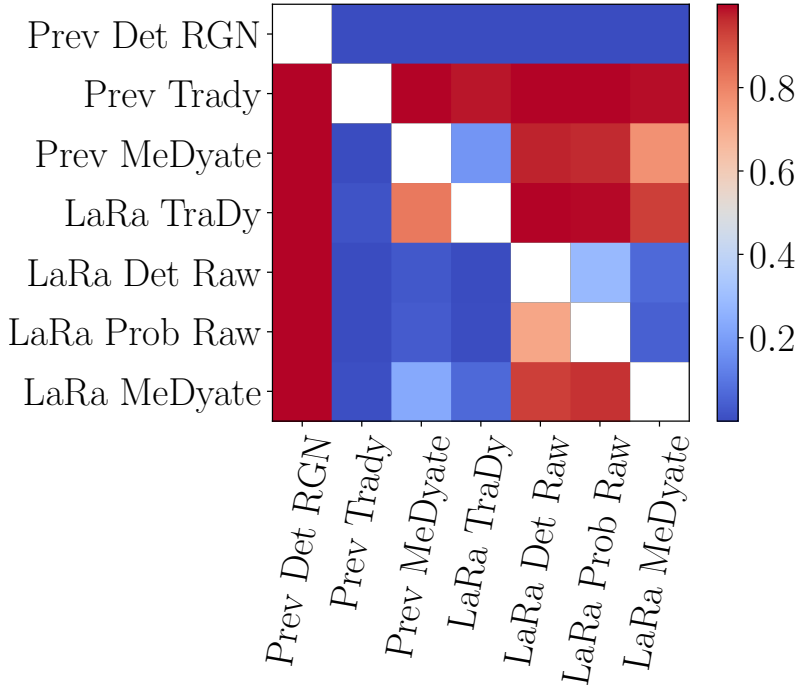
# MeDyate Ablation for Transformers

Two absolute budgets and two proportional budgets

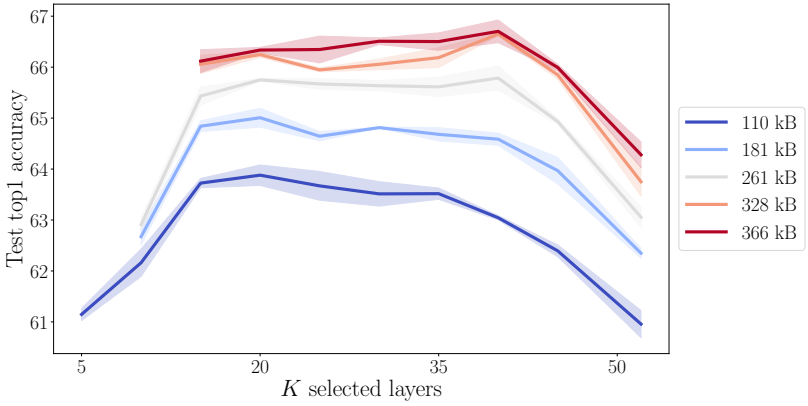
## SwinT

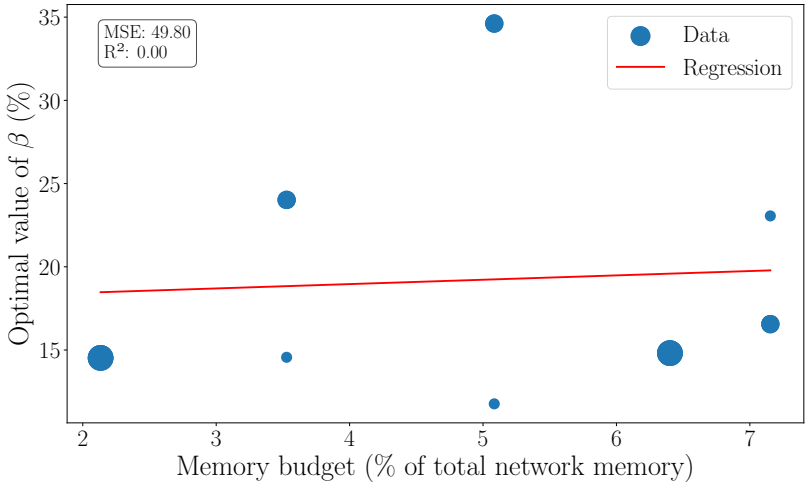



## BERTs



# $\beta_{\text{opt}}$ Selection Process

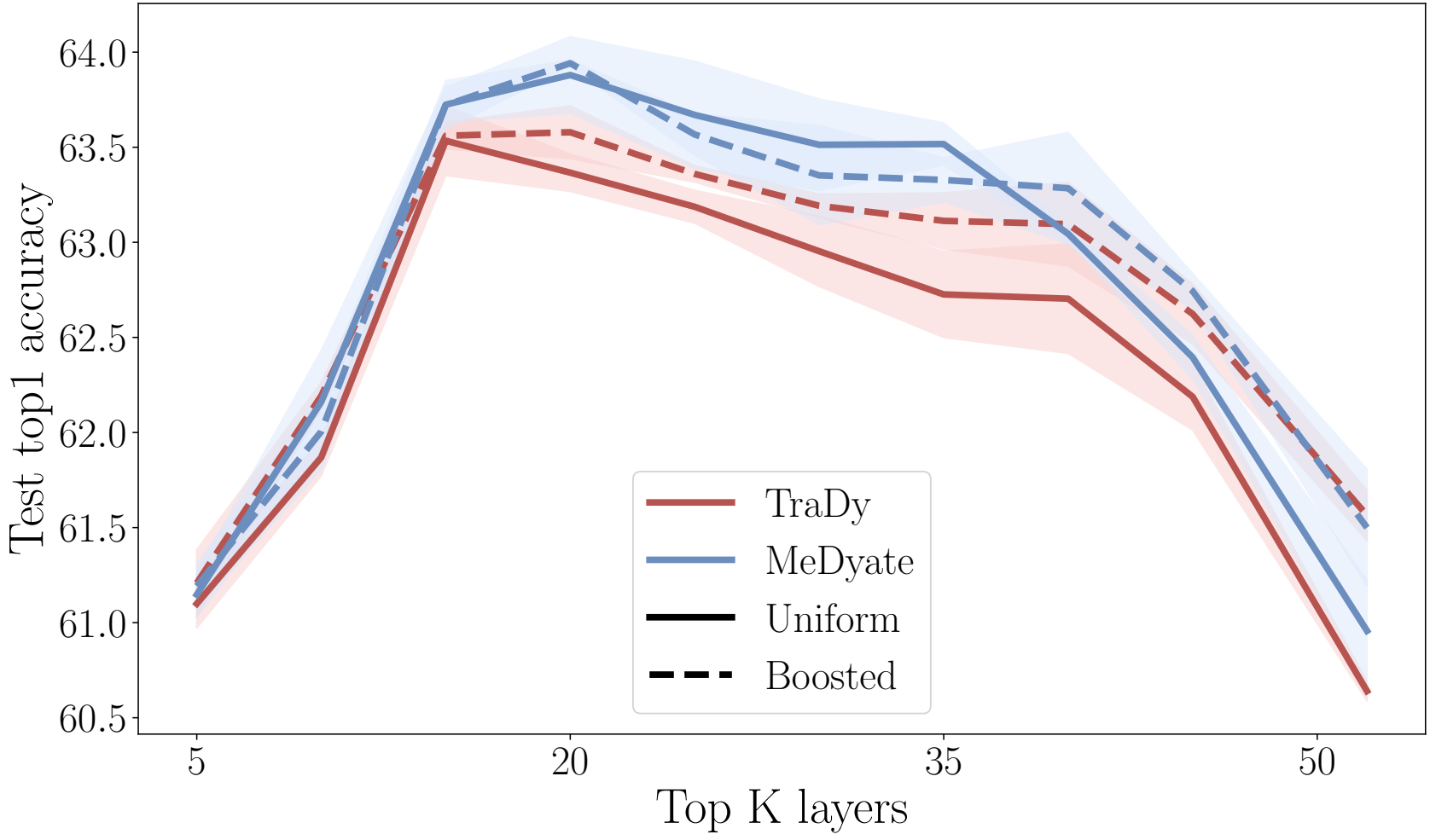



→ Results in $\beta_{\text{opt}} = 0.2$

→ Allows for exploration of all channels in less than 20 epochs

# Exploration of Alternative MeDyate Initialization



# HOSVD Mathematical Details

Weight derivatives extended formula:

$$\frac{\tilde{\partial}\mathcal{L}}{\partial \mathcal{W}_i} = \text{conv}_{1\times1}\left\{\text{conv}_{*}\left[\text{conv}_{1\times1}\left(\text{conv}_{1\times1}\left(\hat{\mathcal{S}}_{\rangle}, \underline{U}^{(3)}_{i,J_{i,3}}\right), \underline{U}^{(4)}_{i,J_{i,4}}\right), \text{conv}_{1\times1}\left(\frac{\partial\mathcal{L}}{\partial\mathcal{A}_{i+1}}, U^{(1)}_{i,J_{i,1}}\right)\right] U^{(2)}_{i,J_{i,2}}\right\},$$

Signal to Noise Ratio analysis:

$$SNR_{\tilde{I}} = \frac{\sum_{(u,v)} I[u,v]^2}{\sum_{(u,v)}(I[u,v] - \varepsilon I[u,v])^2} = \frac{1}{(1-\varepsilon)^2}$$

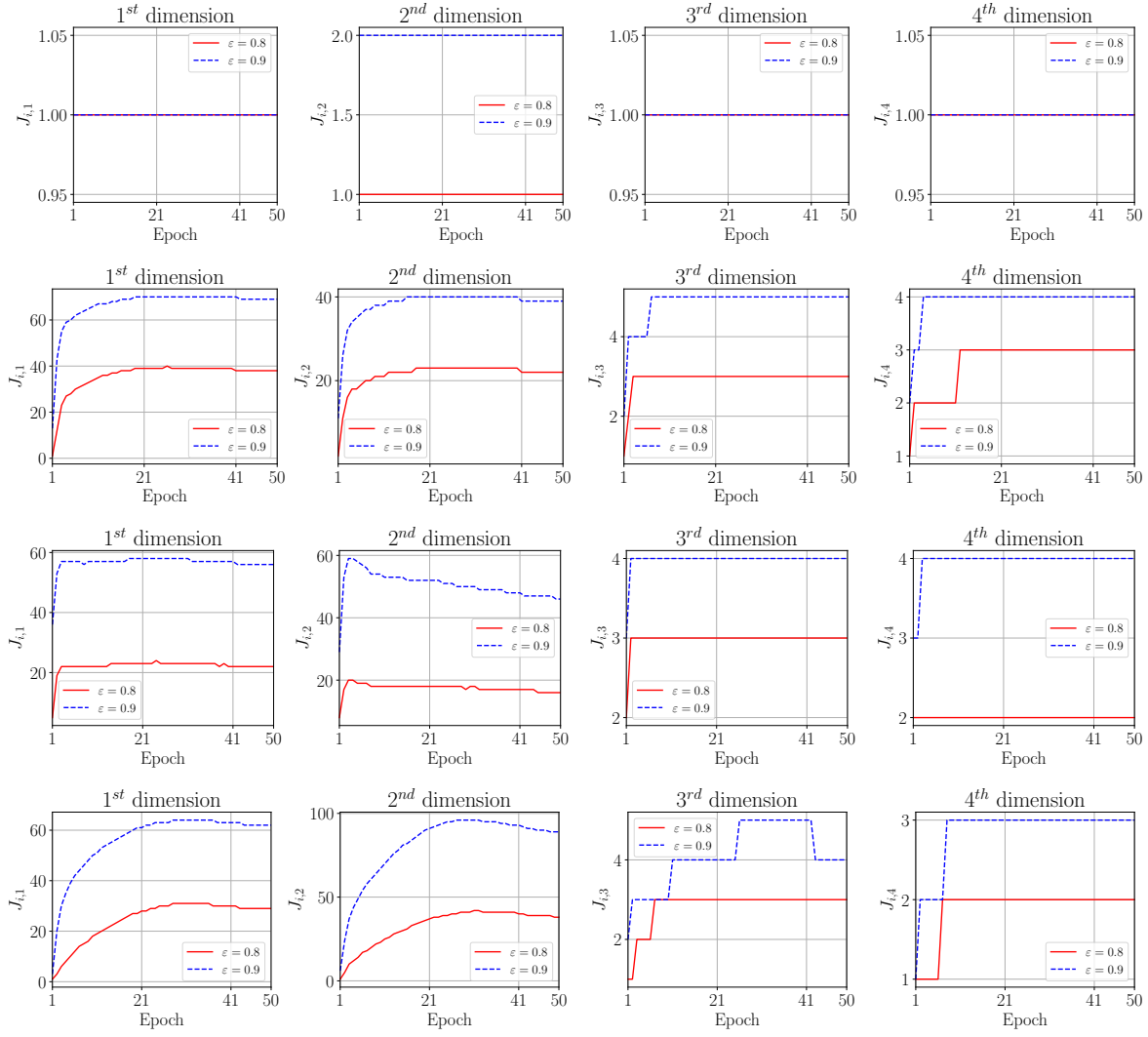
Compression ratio:

$$R_i = \frac{BC_iH_iW_i}{J_{i,1}J_{i,2}J_{i,3}J_{i,4} + BJ_{i,1} + C_iJ_{i,2} + H_iJ_{i,3} + W_iJ_{i,4}}$$

Simplified backward speedup ratio:

$$\tilde{L}_i = \frac{S_i^6}{J_i(J_i^3 + J_i^2S_i + J_iS_i^4 + 2S_i^3)}$$

# Analysis of Rank Evolution with HOSVD



# Detailed ASI Algorithm

**Algorithm 2:** ASI for layer $i$ with preselected set of target ranks $R_{\text{opt}_i}$

**Require:** Activation map $\mathcal{A}_i^{(t)} \in \mathbb{R}^{B \times C_i \times H_i \times W_i}$ at epoch $t$, set of target ranks $R_{\text{opt}_i}$.

Initialize $\mathcal{S}_i = \mathcal{A}_i^{(t)}$.;
**for** $k = 1$ ***to*** $4$ **do**
    $A_{i,k} \leftarrow \text{Unfold}(\mathcal{A}_i^{(t)})$ along mode $k$.;
    **if** $t = 0$ **then**
        Initialize $V_{i,k} \in \mathbb{R}^{S_{i,k} \times J_{i,k}}$ from an i.i.d standard normal distribution.;
    **else**
        $V_{i,k} \leftarrow A_{i,k}^T U_{i,k}^{(t-1)}$.;
    $U_{i,k}^{(t)} \leftarrow \text{Orthogonalize}\,(A_{i,k} V_{i,k})$.;
    $\mathcal{S}_i \leftarrow \mathcal{S}_i \times_k U_{i,k}^{(t)}$.;
**Return** $\mathcal{S}_i$, $U_{i,k}^{(t)}$ with $k : 1 \rightarrow 4$.;

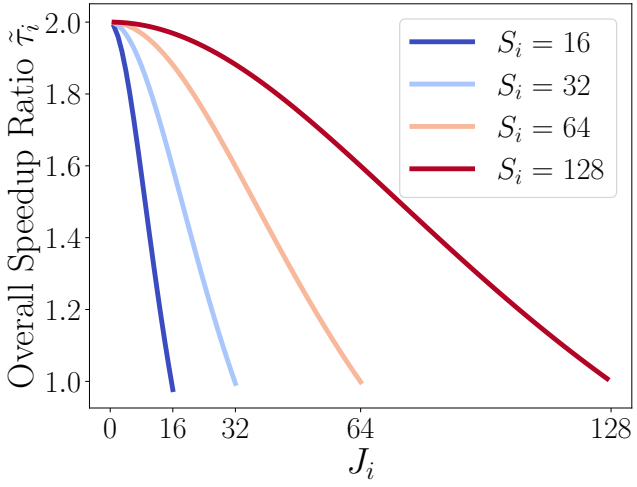
# Additional ASI Details

Rank selection mathematical formulation:

$$R_{\text{opt}_i} = \mathcal{R}_{i,e} \mid e = \mathcal{J}_{\text{opt}}[i],$$

$$\mathcal{J}_{\text{opt}} = \arg\min_{\mathcal{J}} \left( \sum_{i,e \in \mathcal{J}} \mathcal{P}_{i,e} \right) \Bigg| \sum_{i=1}^{|\mathcal{F}|} M_i(J_i^{(e)}) \leq B_{\text{mem}}^{\mathcal{A}}$$

Expected overall speedup:



Andrea Bragagnolo, Enzo Tartaglione, and Marco Grangetto. 2022. To update or not to update? neurons at equilibrium in deep models. *Advances in Neural Information Processing Systems*, 35.

Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. 2020. Tinytl: Reduce memory, not parameters for efficient on-device learning. *Advances in Neural Information Processing Systems*, 33:11285–11297.

Haoze He, Juncheng B Li, Xuan Jiang, and Heather Miller. 2025. Smt: Fine-tuning large language models with sparse matrices. In *The Thirteenth International Conference on Learning Representations*.

Young D. Kwon, Rui Li, Stylianos I. Venieris, Jagmohan Chauhan, Nicholas D. Lane, and Cecilia Mascolo. 2024. Tinytrain: Resource-aware task-adaptive sparse training of dnns at the data-scarce edge.

Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. 2022. On-device training under 256kb memory. *Advances in Neural Information Processing Systems*, 35.

Umut Simsekli, Mert Gürbüzbalaban, Thanh Huy Nguyen, Gaël Richard, and Levent Sagun. 2019. On the heavy-tailed theory of stochastic gradient descent for deep neural networks. *arXiv preprint arXiv:1912.00018*, 222.

Yijun Wan, Melih Barsbey, Abdellatif Zaidi, and Umut Simsekli. 2023. Implicit compressibility of overparametrized neural networks trained with heavy-tailed sgd. *arXiv preprint arXiv:2306.08125*.