ElasticTrainer: Speeding Up On-Device Training with Runtime Elastic Tensor Selection

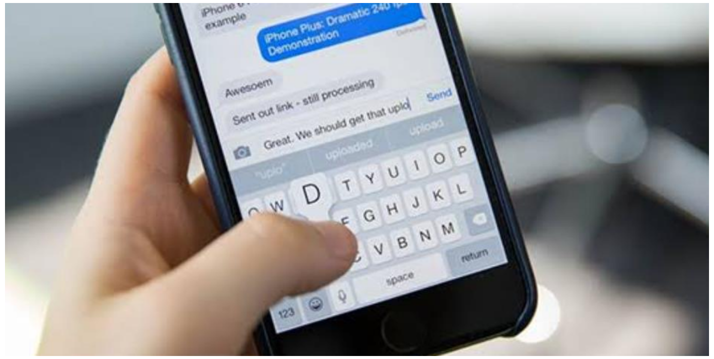
Kai Huang, Boyuan Yang, Wei Gao

University of Pittsburgh

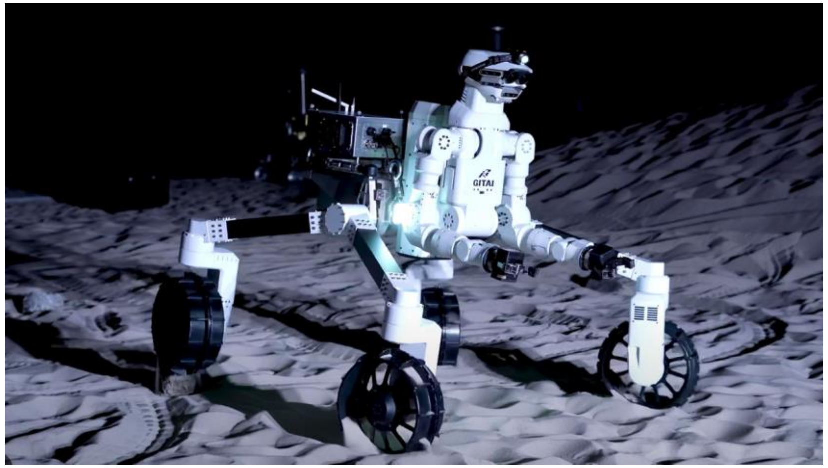
Mobisys'23

- # Introduction

  - Continuously training the AI models



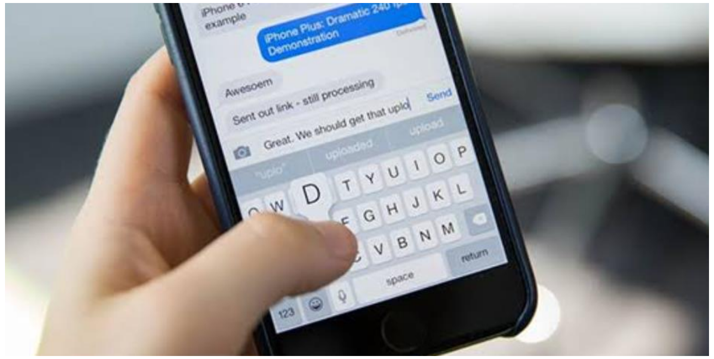
Mobile auto-suggestion



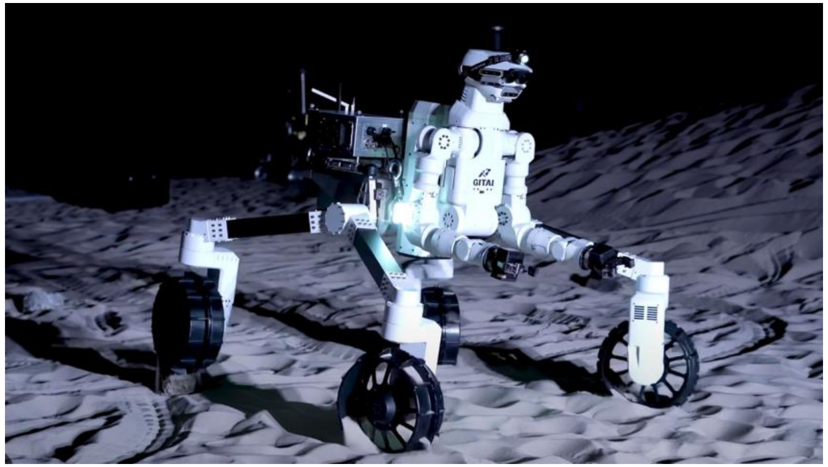
All-terrain robot

• Introduction

- Considering data privacy & bandwidth constraint
  - On-cloud training
  - On-device training √



Mobile auto-suggestion



All-terrain robot

• Motivation & Background

- On-device training is expensive and slow.

Mobile GPU **~16 hours**

Motivation & Background

- Existing schemes
  - Offline Selection: Only Training a portion of the NN model.
    - **Lack of considering the variability of online data that result in dynamic training feedback.**
  - Online Pruning: Adaptively adjust the trainable NN portion at runtime.
    - **The pruned NN portions can never be selected again even if they may be useful.**

Motivation & Background

- ElasticTrainer
  - Every NN substructure can be freely **removed** from or **added** to the trainable NN portion **at any time** in training.

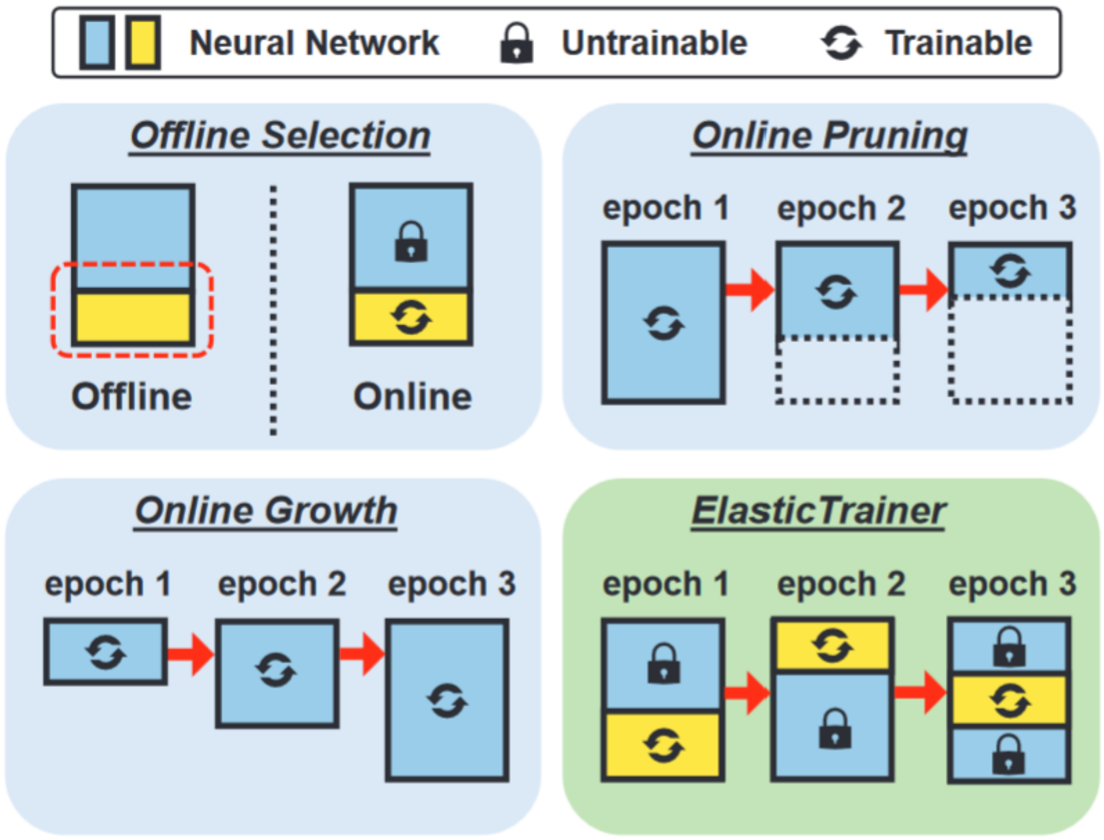


Figure 1: Existing work vs. ElasticTrainer

Motivation & Background

- Granularity of Selection
  - Weight-level
  - Tensor-level
  - Layer-level

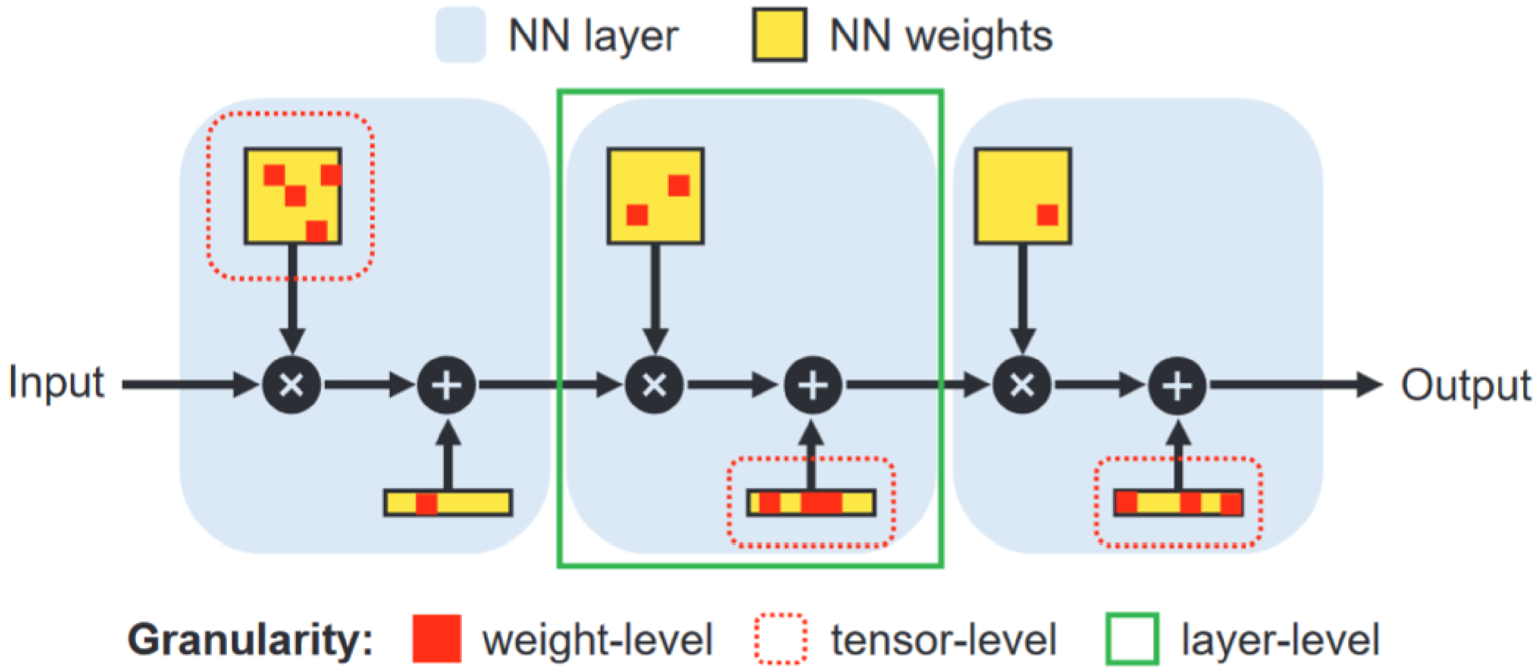
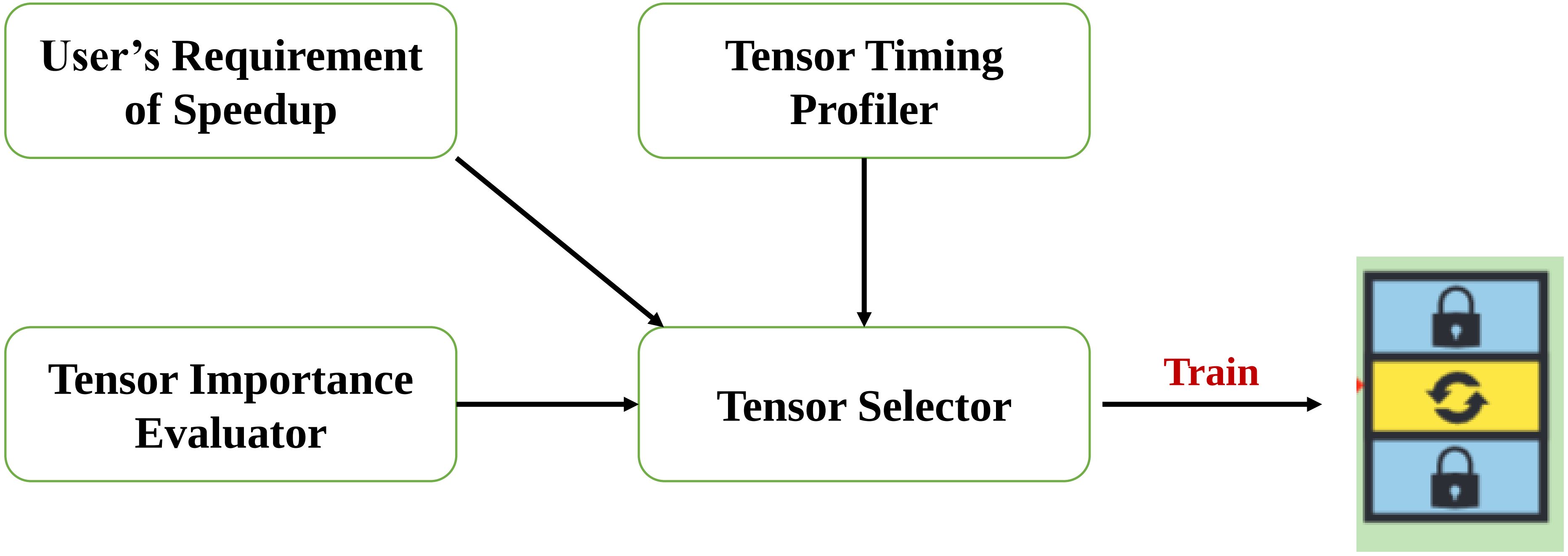


Figure 4: Granularities of selection

- System Overview



- System Overview

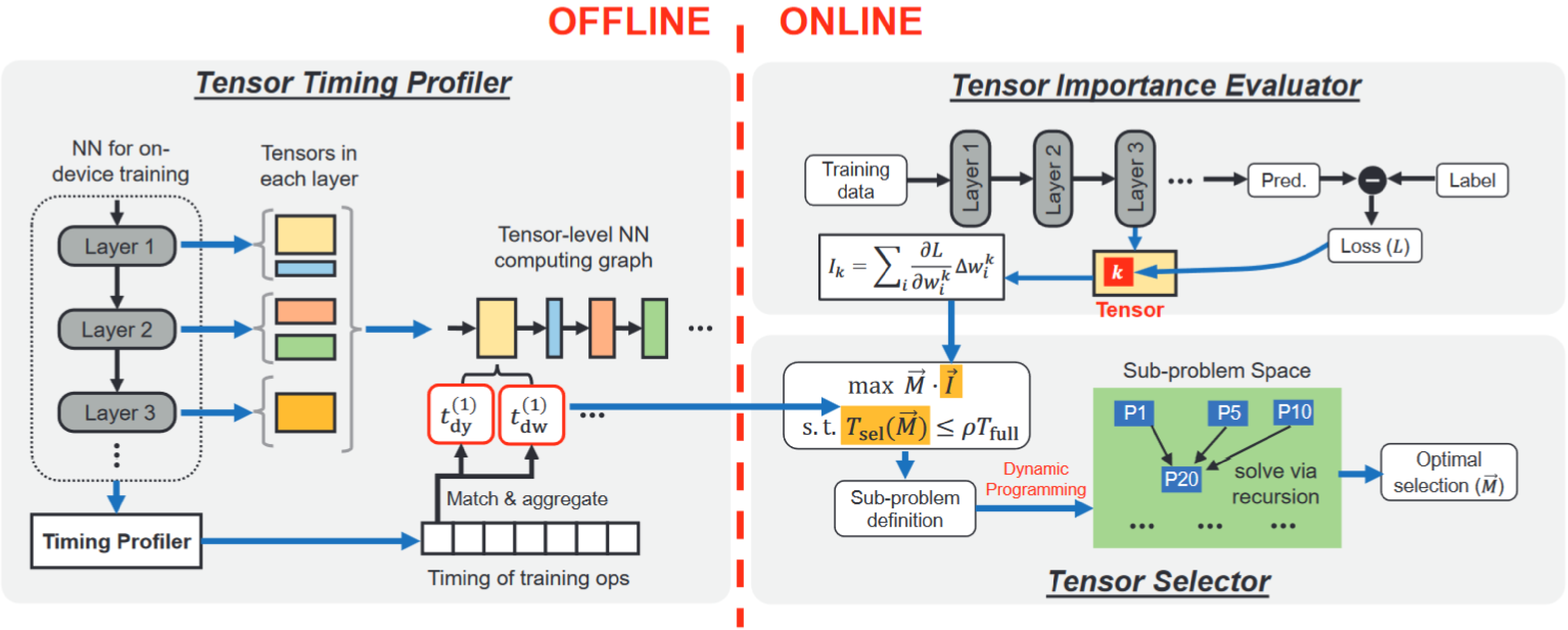


Figure 5: Overview of ElasticTrainer Design

- System - Offline
  - *Tensor Timing Profiler*
    - Forward pass
    - Backward pass

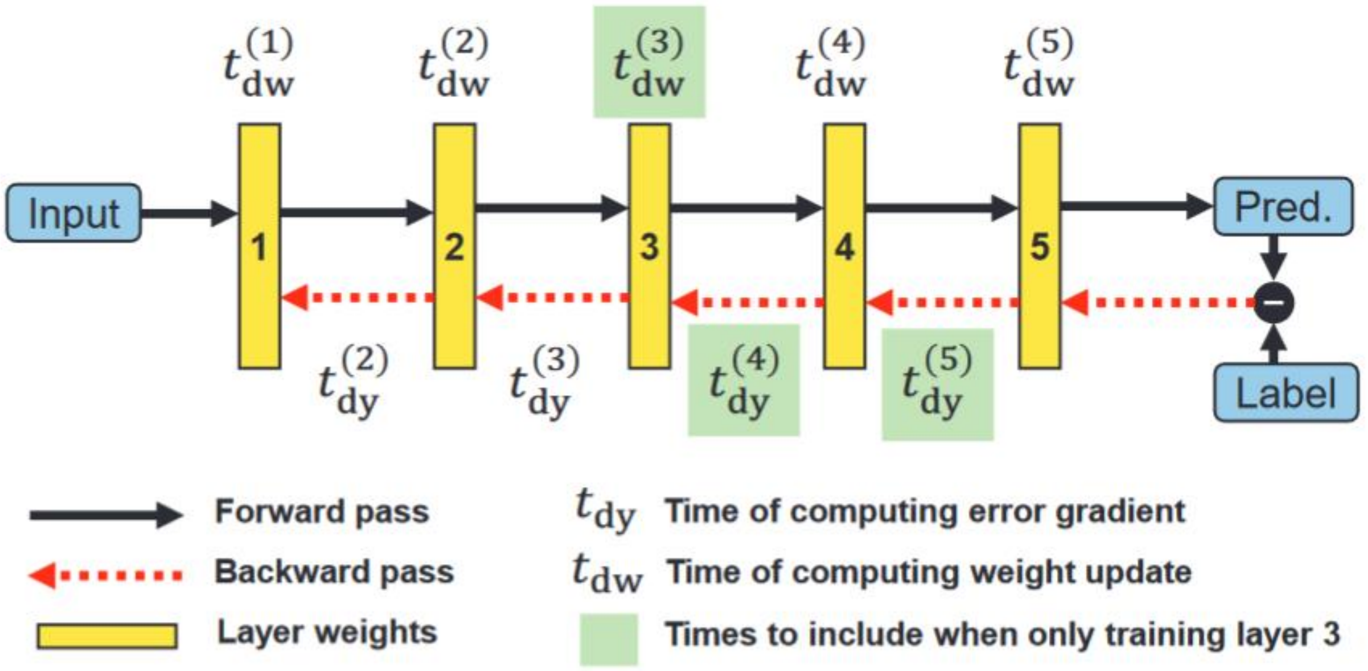


Figure 3: Forward & backward passes of NN training

System - Offline

- *Tensor Timing Profiler*
  - Convolutional Layer & Dense Layer

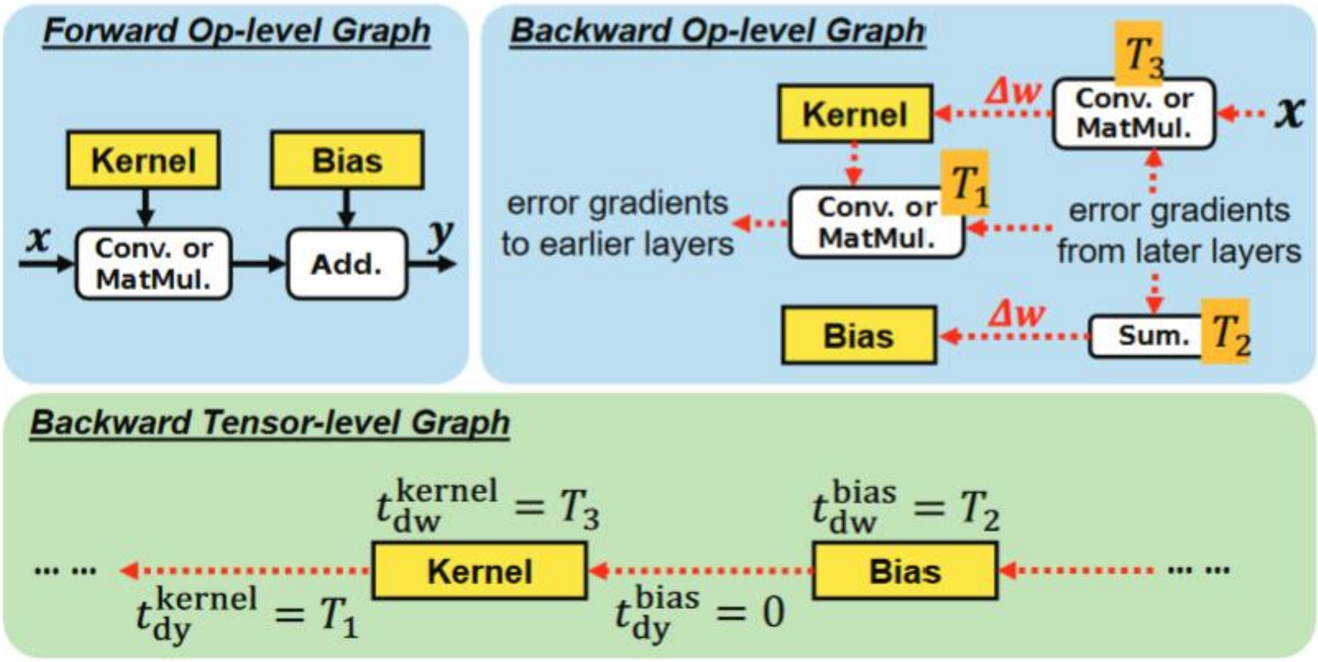


Figure 8: Timing of tensor training in convolutional and dense layers

System - Offline

- *Tensor Timing Profiler*
  - Batch Normalization Layer

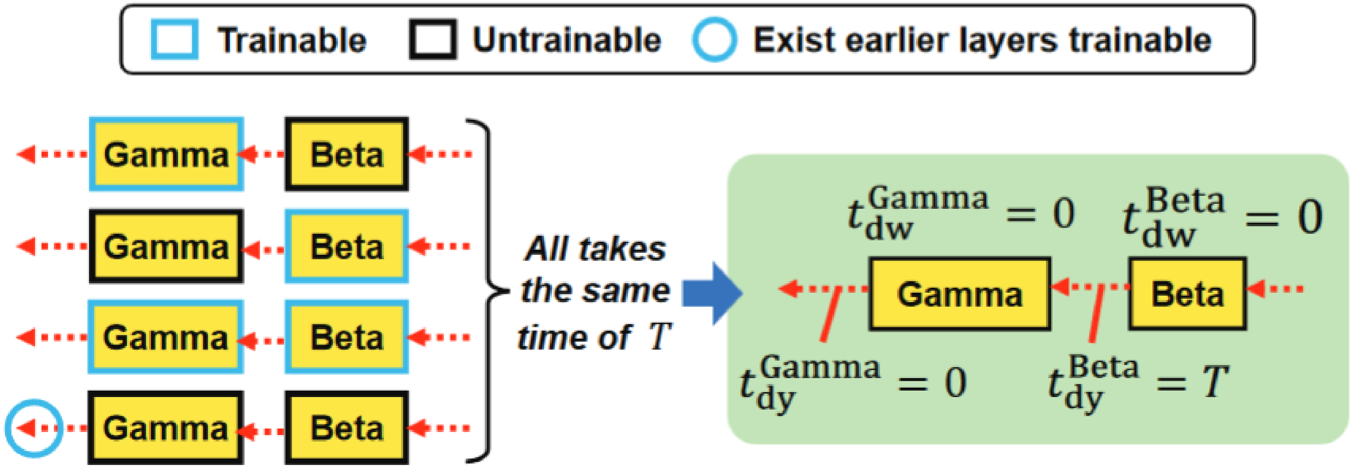


Figure 9: Timing of tensor training in batch normalization layers

- # System - Offline
  - *Tensor Timing Profiler*
    - Non-trainable Layer

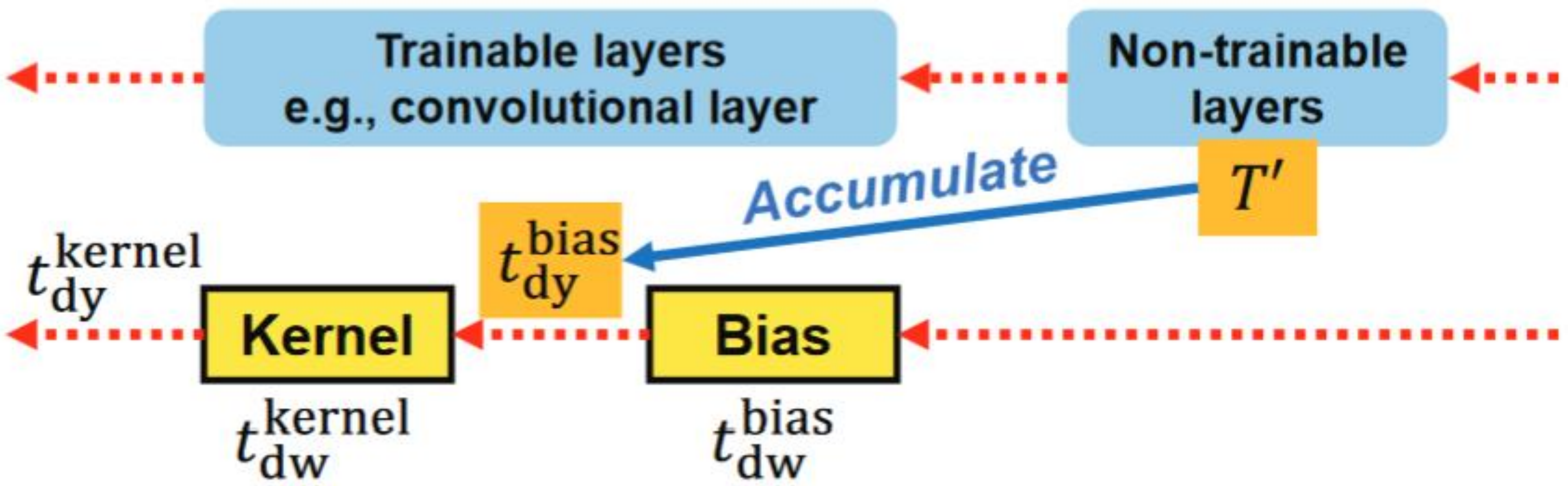


Figure 10: Including timings of non-trainable layers

• System - Online

- *Tensor Importance Evaluator(Based on XAI)*

$$I_k = \sum_i \frac{\partial L}{\partial w_i^k} \Delta w_i^k,$$

$$\Delta L = L(w) - L(w + \Delta w), \tag{4}$$

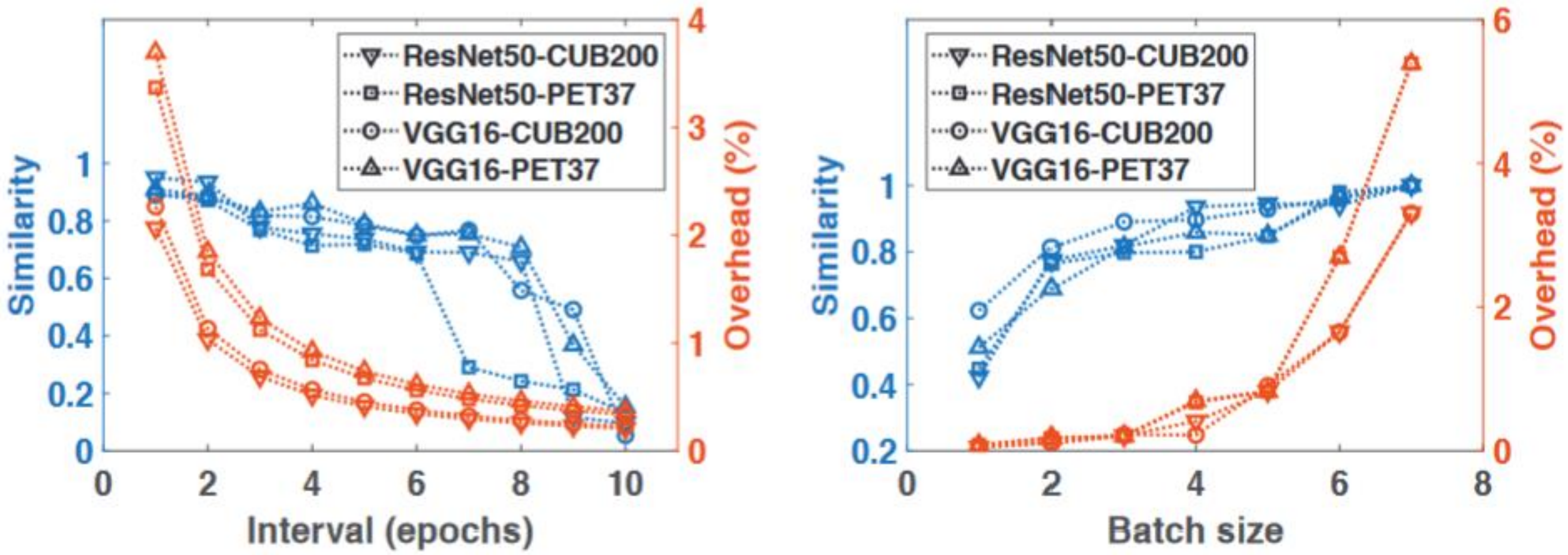
$\Downarrow$ $\vec{c}$ **Continuous undo operation**

$$\frac{\partial L(\vec{w} + \vec{c} \odot \Delta\vec{w})}{\partial \vec{c}} = \Delta\vec{w} \odot \frac{\partial L(\vec{u})}{\partial \vec{u}}\bigg|_{\vec{u}=\vec{w}+\vec{c}\odot\Delta\vec{w}}, \tag{5}$$

$\Downarrow$ ***Approximation***

$$L(\vec{w} + \Delta\vec{w}) = L(\vec{w}) + \frac{\partial L(w_1)}{\partial w_1}\Delta w_1 + \frac{\partial L(w_2)}{\partial w_2}\Delta w_2 + \ldots \tag{6}$$

- System - Online
  - *Tensor Importance Evaluator*



(a) Different evaluation intervals (b) Different batch sizes

Figure 7: The impact of evaluation interval and batch size on tensor importance evaluation

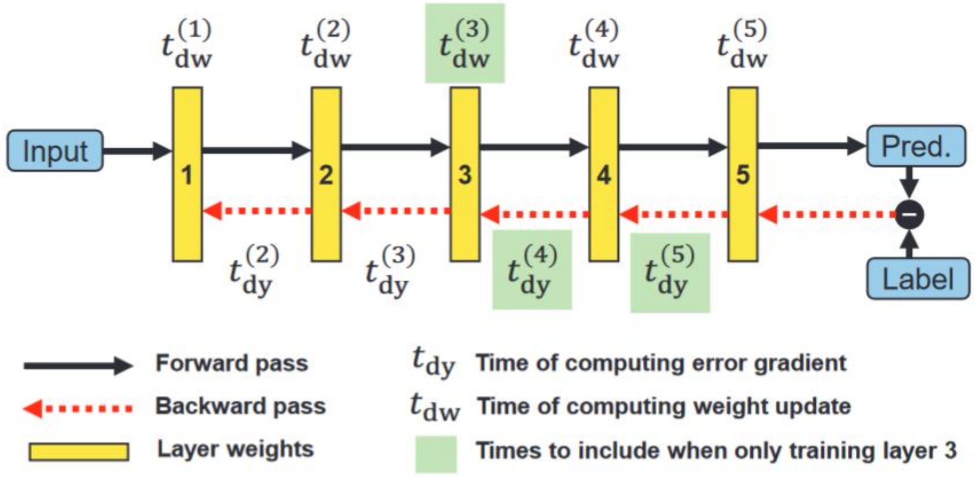
• System - Online

- *Tensor Selector*

$$\max \vec{M} \cdot \vec{I} \quad \text{s.t. } T_{forward} + \vec{M} \cdot t_{\vec{dw}} + f(\vec{M}) \cdot t_{\vec{dy}} \leq \rho T_{full}, \quad (3)$$

$\vec{M}$ [0, 0, 1, 1, 0, 0]

$f(\vec{M})$ [0, 0, 1, 1, 1, 1]



- solve it **in pseudo-polynomial time** by **dynamic programming** (DP).

• System - Online

- *Tensor Selector*

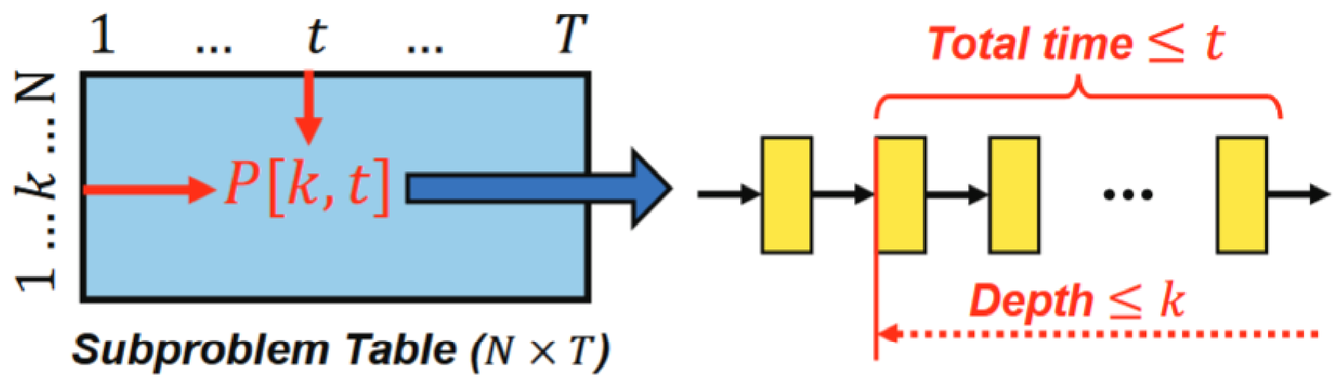
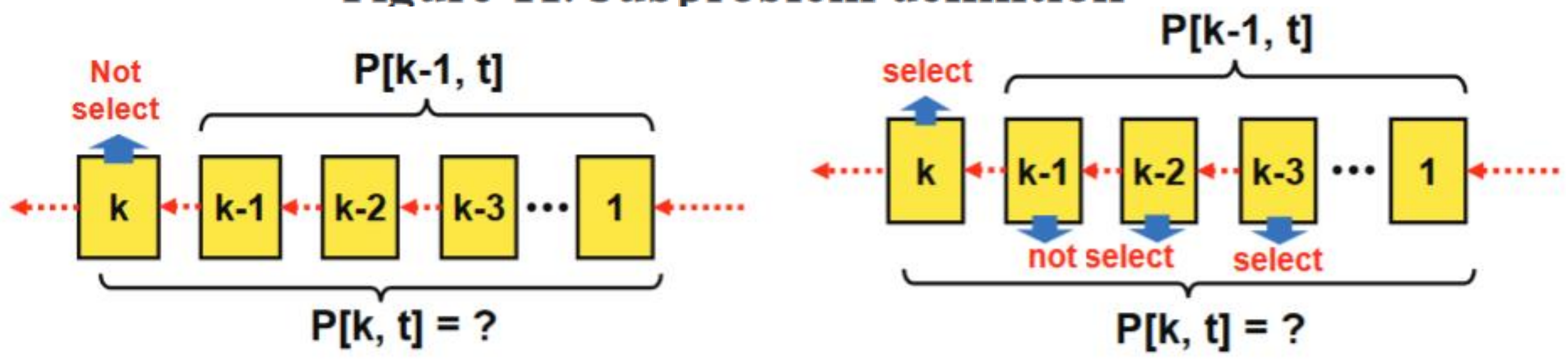


Figure 11: Subproblem definition



(a) Bottom tensor k is not selected (b) Bottom tensor k is selected

Figure 12: Finding recursion relation in different cases

• Implementation

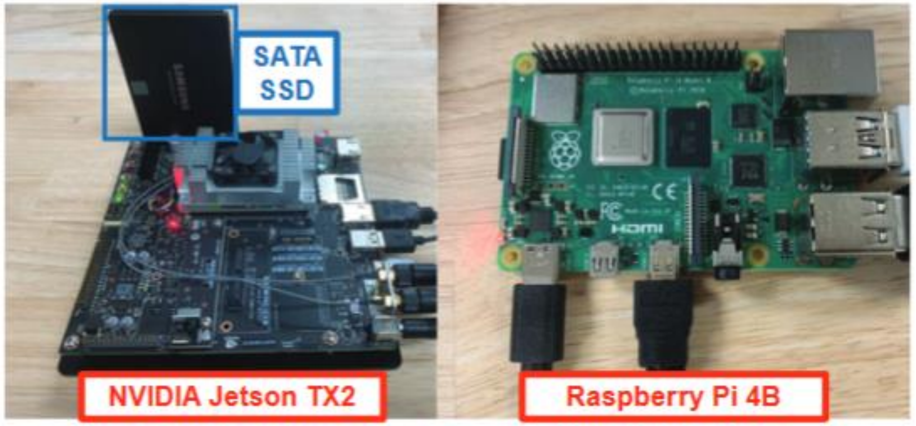
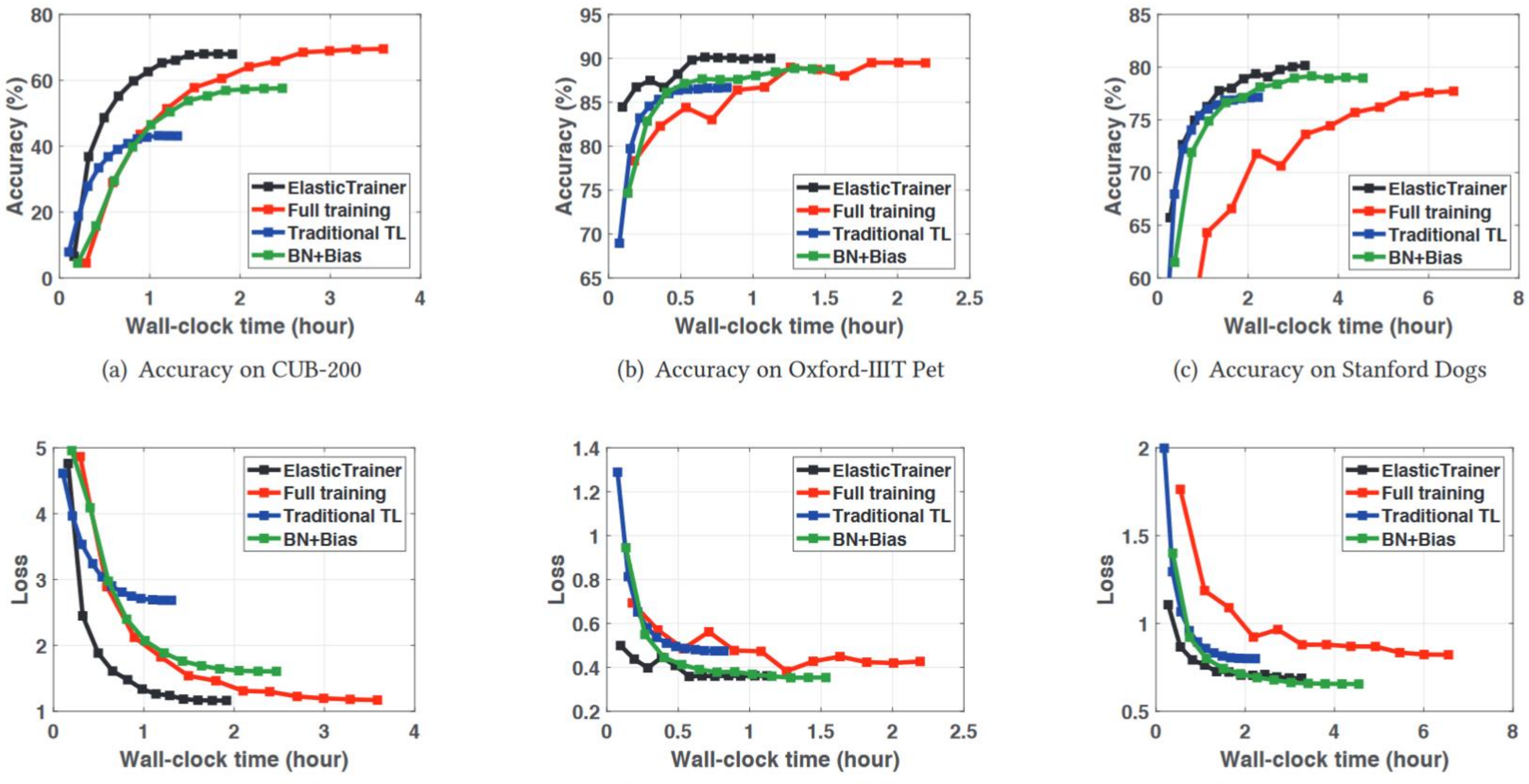


Figure 14: Devices used in our implementation

- TensorFlow 2.7
- TensorFlow Addons 0.15

- **Baseline**
  - Full training
  - Traditional TL
  - BN+Bias
  - PruneTrain

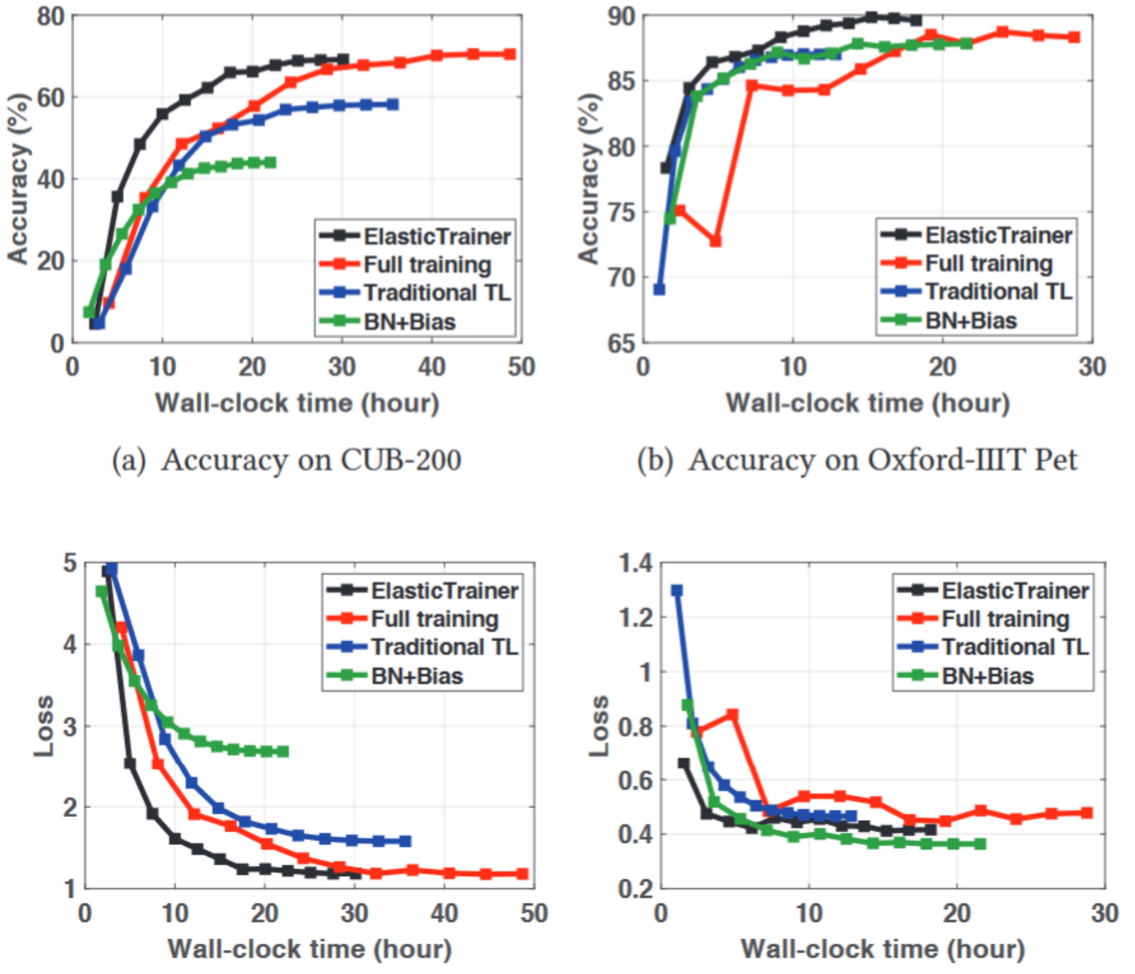
• Performance Evaluation



(a) Accuracy on CUB-200
(b) Accuracy on Oxford-IIIT Pet
(c) Accuracy on Stanford Dogs
(d) Loss on CUB-200
(e) Loss on Oxford-IIIT Pet
(f) Loss on Stanford Dogs

Figure 15: Testing accuracy and training loss over time with different datasets on Jetson TX2

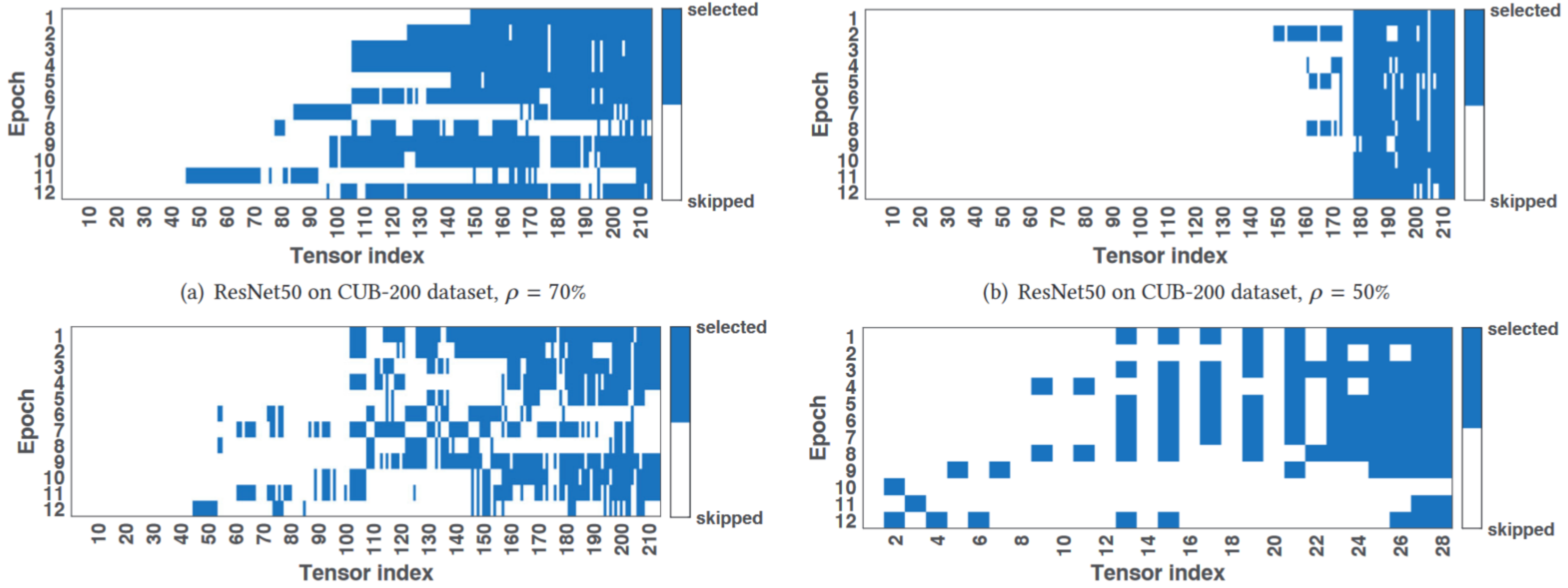
- Performance Evaluation



(a) Accuracy on CUB-200
(b) Accuracy on Oxford-IIIT Pet
(c) Loss on CUB-200
(d) Loss on Oxford-IIIT Pet

Figure 16: Testing accuracy and training loss over time with different datasets on Raspberry Pi 4B

• Performance Evaluation



(a) ResNet50 on CUB-200 dataset, ρ = 70%

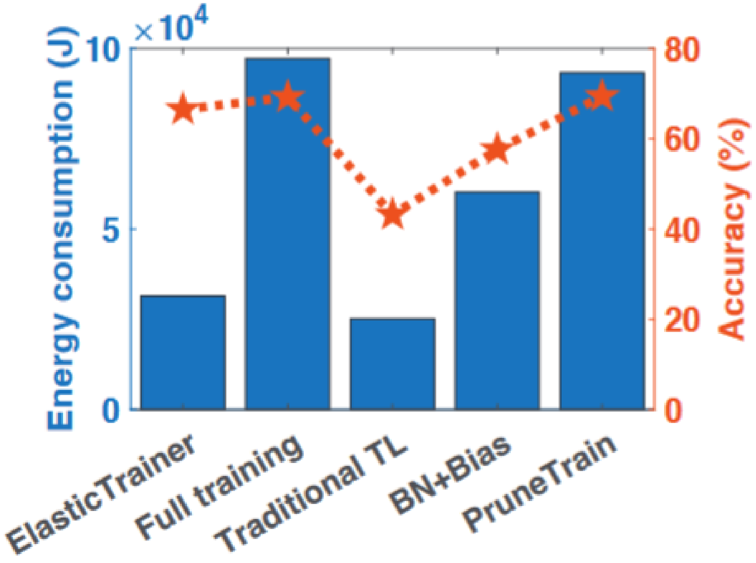
(b) ResNet50 on CUB-200 dataset, ρ = 50%

(c) ResNet50 on Oxford-IIIT Pet dataset, ρ = 70%

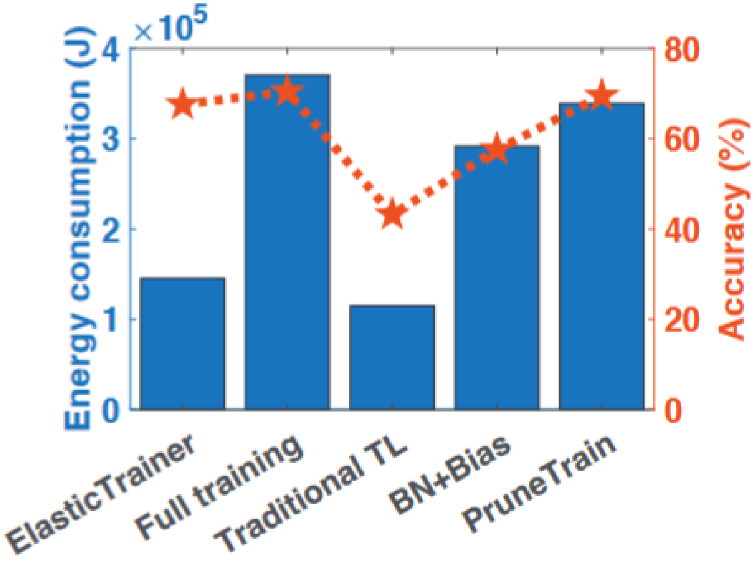
(d) VGG16 on Oxford-IIIT Pet dataset, ρ = 70%

Figure 21: Elastic tensor selections in different training epochs

• Performance Evaluation



(a) Jetson TX2



(b) Raspberry Pi 4B

Figure 23: Energy cost on different devices

Conclusion

- Advantage
  - Save **time** and **energy consumption** on on-device training elastically.

- Disadvantage
  - **Discrete** & **continuous** ?