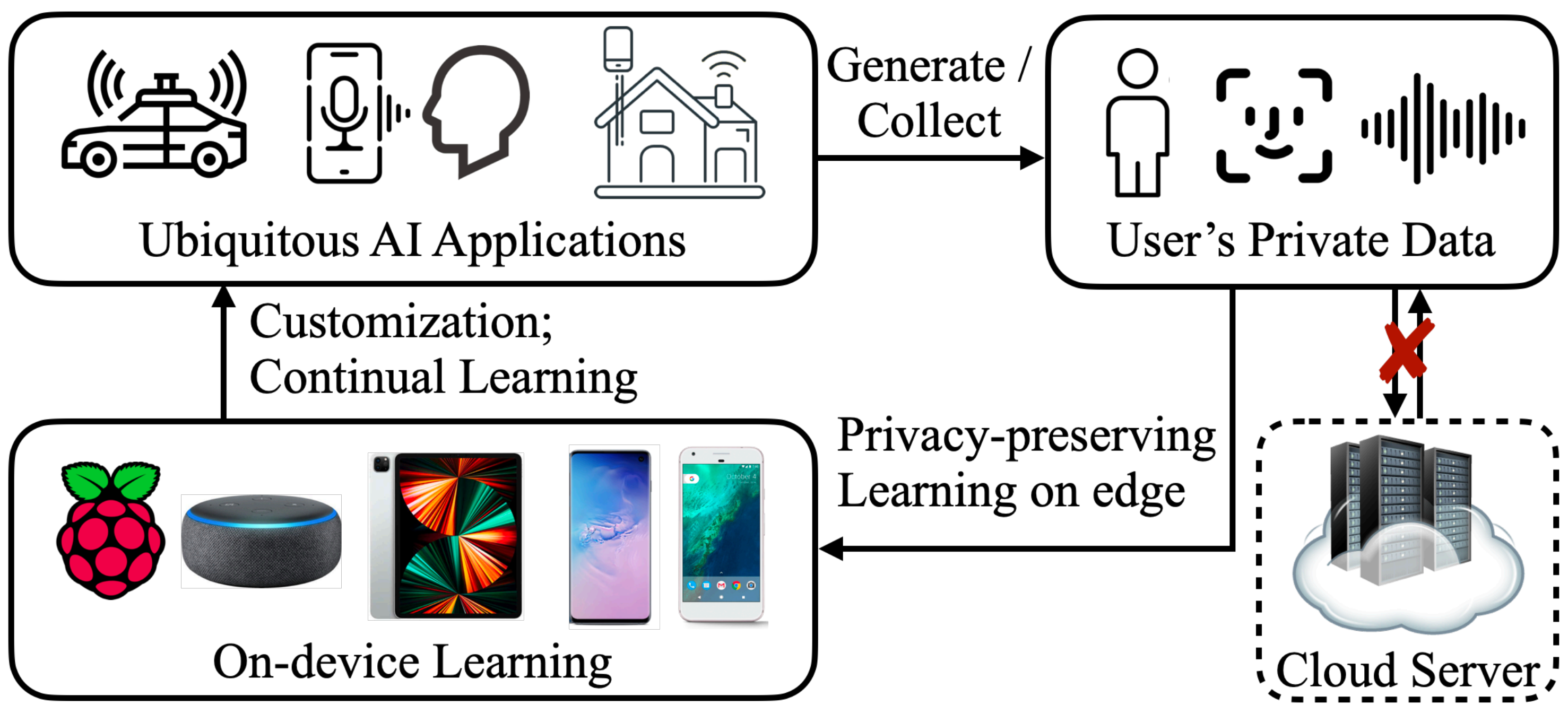
# PockEngine: Sparse and Efficient Fine-tuning in a Pocket

Ligeng Zhu[1], Lanxiang Hu[2], Ji Lin[1], Wei-Chen Wang[1], Wei-Ming Chen[1], Chuang Gan[3], Song Han[1,4]

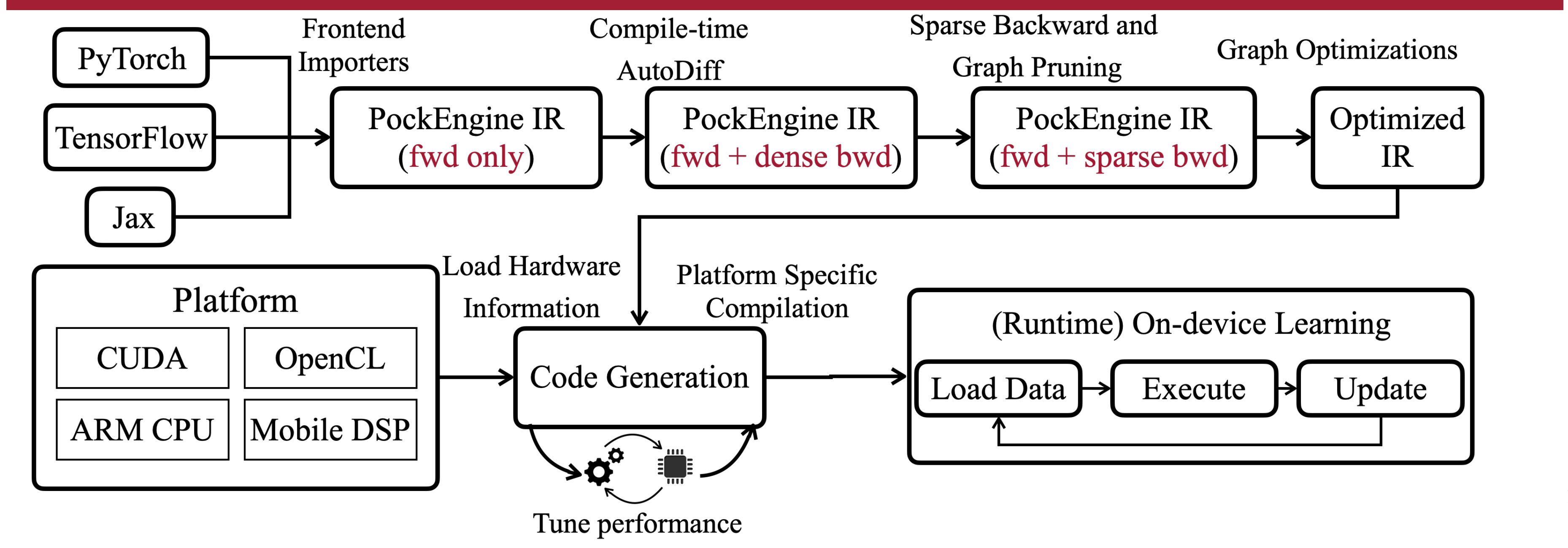
MIT[1], UCSD[2], MIT-IBM Watson AI Lab[3], NVIDIA[4]

## On-device Training and Continue Learning


Ubiquitous AI Applications
Generate / Collect
User's Private Data
Customization; Continual Learning
Privacy-preserving Learning on edge
On-device Learning
Cloud Server

- **Privacy**: Data **never leave devices**. Sensitive enterprise data (copilot for coding).
- **Customization**: Models **continually adapt** to new data.
- **Low-Cost:** No need to rent cloud server. Fine-tune LLM on your edge device.

## PockEngine Overview

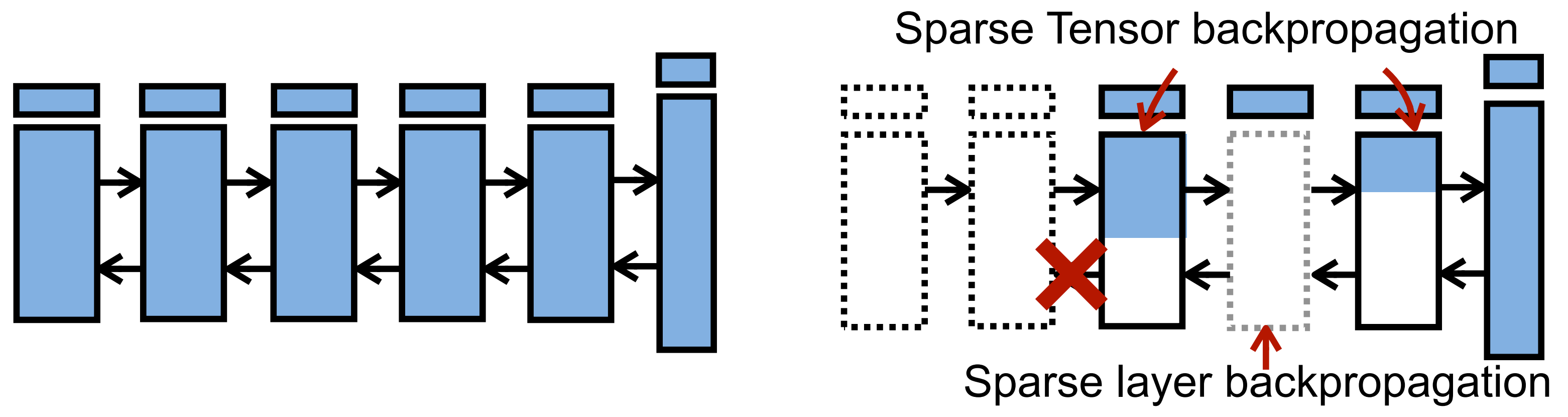


PockEngine features:

- **Sparse backpropagation:**
  - **Sparse layer** BP: skip updating unimportant layers
  - **Sparse tensor** BP: skip updating unimportant channels
- **Compiler support:**
  - Remove the pruned operators via **dead code elimination**
  - Move from **runtime** to **compile-time**: auto-diff, pruning, graph optimizations
  - Enable **inference-only frameworks** to perform training

## Sparse Layer/ Sparse Tensor Backpropagation

- Conventionally, we update the **full model** or **last layer** for transfer learning
- We find some layers are **more important** than others, then **sparsely update**



(a) Full BP

(b) Sparse BP

```
# forward layer #1
%0 = multiply(%x, %w1);
%1 = add(%0, %b1);
# forward layer #2
%2 = multiply(%1, %w2);
%3 = add(%2, %b2);
# backward layer #2
%4 = multiply(%grad, %w2);
%5 = transpose(%grad);
%6 = multiply(%5, %1);
%7 = sum(%grad, axis=-1);
# backward layer #1
%8 = multiply(%6, %w1);
%9 = transpose(%6);
%10 = multiply(%9, %x);
%11 = sum(%6, axis=-1);
return (%6,%7,%10,%11)
```

(c) The IR of Full BP

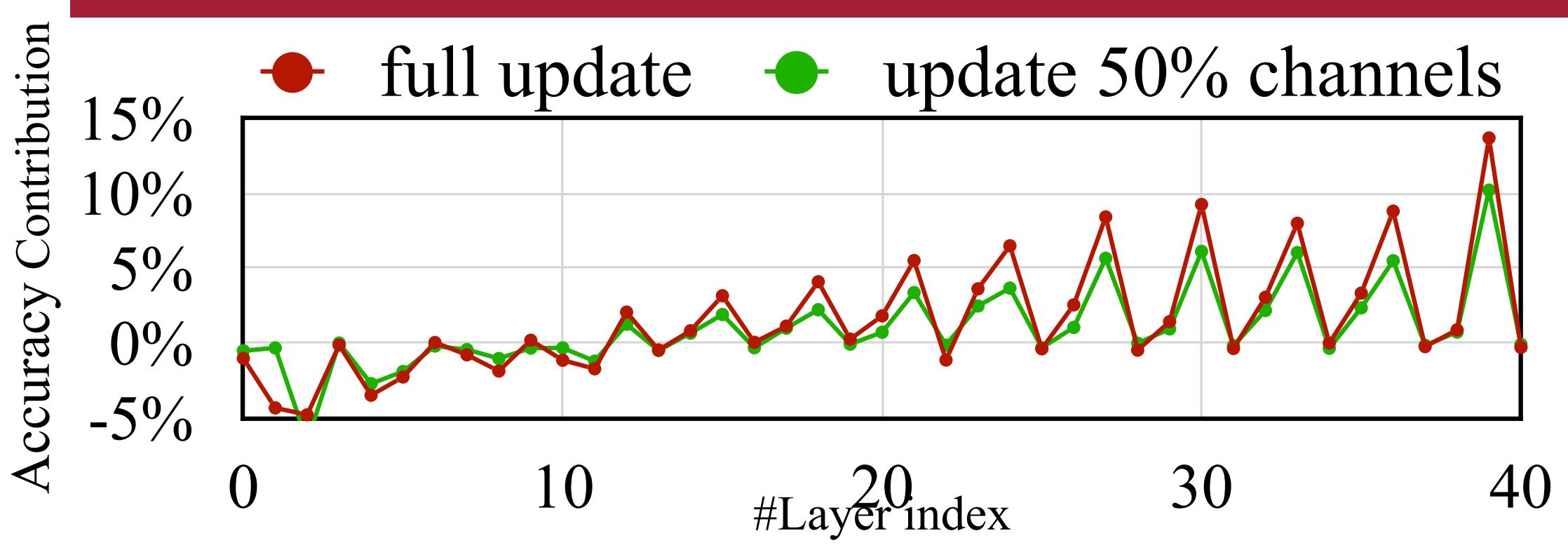
```
# forward layer #1
%0 = multiply(%x, %w1);
%1 = add(%0, %b1);
# forward layer #2
%2 = multiply(%1, %w2);
%1.1 = slice(%1, range=[0:10, 0:10]);
%3 = add(%2, %b2);
# backward layer #2 [Sparse Tensor BP]
%4 = multiply(%grad, %w2);
%5 = transpose(%grad);
%6 = multiply(%5, %1.1);
%7 = sum(%grad, axis=-1);
# backward layer #1 [Sparse Layer BP]
%8 = multiply(%6, %w1);
~~%9 = transpose(%6);~~
~~%10 = multiply(%9, %x);~~
%11 = sum(%6, axis=-1);
return (%6,%7,~~%10,~~%11)
```

(d) The IR of Sparse BP

Once sparse engine is determined, PockEngine

- Traverse and automatically **modify the** DAG (sparse tensor BP, **blue parts**)
- Remove unused OPs via **dead code elimination** (sparse layer BP, ~~**black lines**~~)

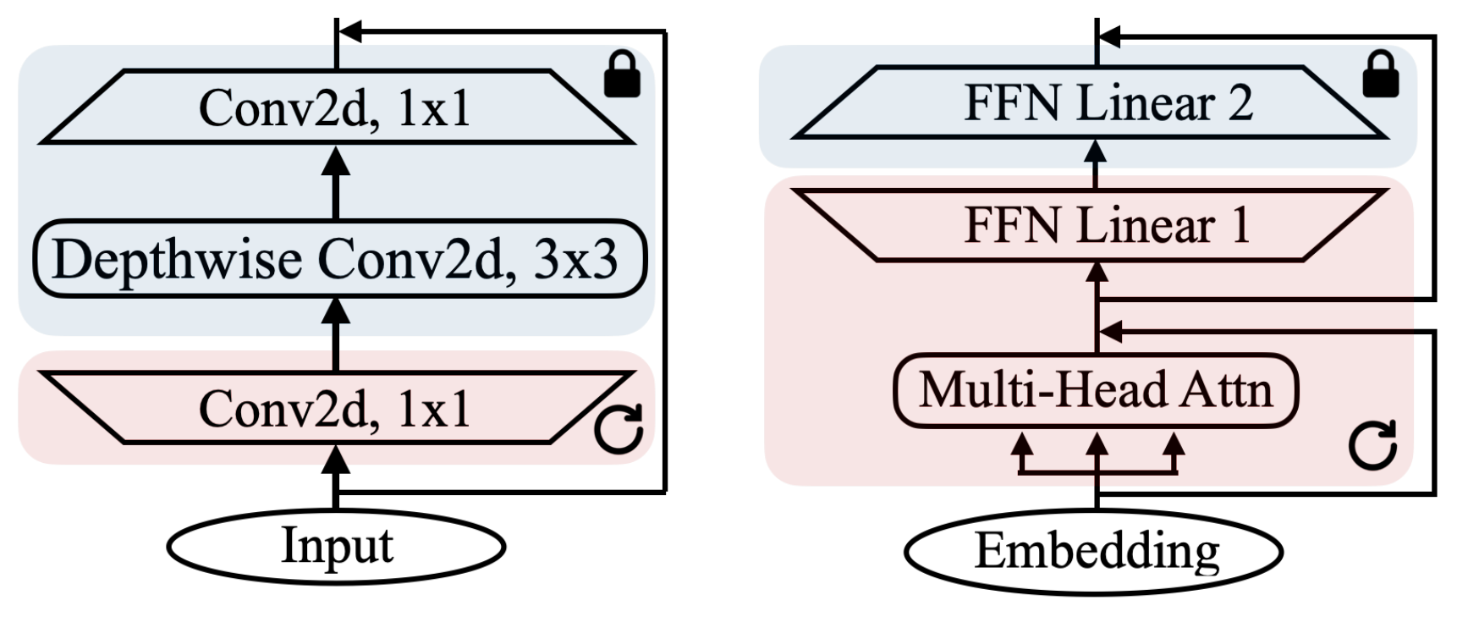
## Searching Important Layers to Sparsely Update



$$\mathbf{k}^*, \mathbf{i}^*, \mathbf{r}^* = \max_{\mathbf{k},\mathbf{i},\mathbf{r}}\left(\sum_{\mathbf{k}\in\mathbf{i}} \Delta\mathrm{acc}_{\mathrm{b}_k} + \sum_{i\in\mathbf{i}, r\in\mathbf{r}} \Delta\mathrm{acc}_{\mathrm{W}_{i,r}}\right)$$

$$\text{s.t. Memory}(\mathbf{k}, \mathbf{i}, \mathbf{r}) \leq \text{constraint},$$

$k$: bias layer index, $i$: weight layer index, $r$: sparsity ratio.



(a) MobileNetV2

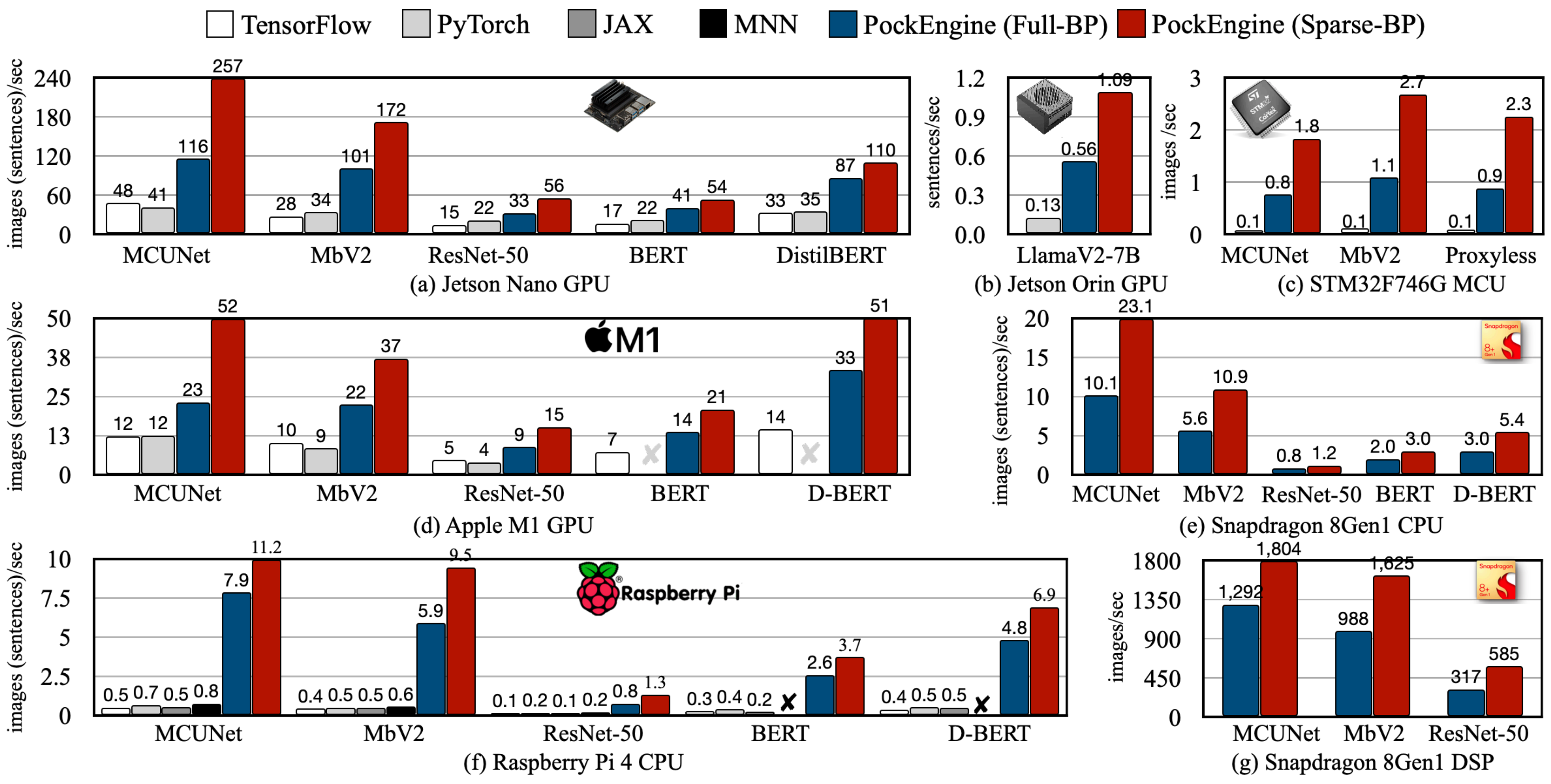
(b) LLamaV2

| Language Model | Method | Avg. |
|---|---|---|
| Distill-BERT [52] | Full BP | 76.9% |
| | Bias Only | 72.8% |
| | Sparse BP | 77.0% |
| BERT [19] | Full BP | 81.8% |
| | Bias Only | 78.1% |
| | Sparse BP | 81.7% |

| Vision Model | Method | Avg. Acc |
|---|---|---|
| MCUNet-5FPS [41] | Full BP | 74.1% |
| | Bias Only | 72.7% |
| | Sparse BP | 74.8% |
| MobilenetV2 [51] | Full BP | 89.2% |
| | Bias Only | 87.3% |
| | Sparse BP | 88.5% |
| ResNet-50 [23] | Full BP | 90.5% |
| | Bias Only | 87.8% |
| | Sparse BP | 90.3% |

- Not all layers are necessary for fine-tuning:
  - [**Transformers**]: Update Attn and FFN1, not FFN2.
  - [**MobilenetV2**]: Update $1^{st}$ point-wise conv, not depth-wise conv nor $2^{nd}$ point-wise conv.
- **Sparse-BP** demonstrates **on-par performance with Full-BP** while greatly reduces training cost.

## Acceleration Results



- Enable on-device training on **resource constrained** devices.
- **4.4x speedup** compared to PyTorch on Jetson Orin and 15x on Raspberry Pi.

## On-Device LLM Fine-Tuning

| Framework | Method | Iteration Latency (↓) | GPU Memory(↓) | Loss(↓) | Alpaca-Eval Winrate(↑) | MT-Bench score(↑) |
|---|---|---|---|---|---|---|
| PyTorch | FT-Full | 7.7s | 45.1GB | 0.761 | 44.1% | 6.1 |
| PyTorch | LoRA (rank=8) | 7.3s | 30.9GB | 0.801 | 43.1% | 5.1 |
| PockEngine | FT-Full | 1.8s | 43.1GB | 0.768 | 43.7% | 6.1 |
| PockEngine | Sparse | 0.9s | 31.2GB | 0.779 | 43.1% | 5.7 |

- PockEngine **accelerated training** (by 4.4x compared to PyTorch)
- Sparse-BP **speedups training** while achieves **comparable accuracy**.

**Input**: Please reverse the words in the sentence "I love the Micro conference"
**LlamaV2-7B-original**: I **hate** the conference Micro.
**LlamaV2-7B-tuned (Full-BP)**: The reversed result is "Conference Micro the love I".
**LlamaV2-7B-tuned (Sparse-BP)**: The reversed sentence would be "Conference the Micro love I".

**Input**: what was Michael Jackson's last album?
**LlamaV2-7B-original**: Michael Jackson passed away on June 25, 2020, so he did not release an album after that.
**LlamaV2-7B-tuned (Full-BP)**: Michael Jackson's last album was 'Invincible', released in 2001.
**LlamaV2-7B-tuned (Sparse-BP)**: Michael Jackson's last album, released during his lifetime, was "Invincible" which was released in 2001.

- **Fine-tuning is necessary** for LLM, otherwise:
- PockEngine customized models to generate high-quality answers and allows everyone to build their own chatbot.