Pluto and Charon: A Time and Memory Efficient Collaborative Edge AI Framework for Personal LLMs Fine-Tuning

Bei Ouyang*, Shengyuan Ye*, Liekang Zeng, Tianyi Qian, Jingyi Li, Xu Chen†

1Sun Yat-sen University, 2HKUST(GZ)
Transformer-Based LLMs and Fine-Tuning

- Acquire a general understanding of linguistic structure and patterns.
- Adapts the pre-trained model to various, concrete downstream language tasks.

### Pre-Training
- Random Model
- Pre-Training Data
- Pre-Trained Model

### Fine-Tuning
- In-Domain Data
- Fine-Tuned Model

- Sentiment Analysis
- Question Answering
- Task Execution
Parameter-efficient finetuning (PEFT) techniques

- **Adapters**: inserts compact bottleneck modules at the end of each transformer layer.

- **LoRA**: injects trainable low-rank matrices into a frozen pre-trained model.
Intelligent edge applications

- Transformer-based models driven increasing intelligent applications.

- Intelligent personal assistants
- Intelligent robots/aircraft
- AR&VR applications
Problems of cloud-assisted approaches

- Current Transformer-based applications heavily depend on cloud services.

The advantage of cloud service

- Powerful and scalable computing resources

Three issues with cloud service

- Data privacy and security issues.
- Unreliable WAN connections.
- Network and datacenter pressure.
LLMs Fine-Tuning with Resource-Constrained Edge Devices

- On-device deployment becomes a promising paradigm for intelligent edge APPs.

Transformer-based intelligent applications

Intelligent Edge Devices

- Protect date privacy.
- Without WAN transmission.
- Limited and non-scalable on-board computing resources
Challenges

- PEFT techniques are not resource-efficient enough for edge environments.
- Adapters and LoRA exhibit a limited reduction in computation (around 30%).

**Figure 3:** The comparison of floating point of operations (FLOPs). Mini-batch size: 16; sequence length: 128.
**Challenges**

- PEFT techniques are **not resource-efficient** enough for edge environments.
- Adapters and LoRA exhibit a maximum reduction of only 36% in memory.

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Trainable Parameters</th>
<th>Memory Footprint (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Weights</td>
</tr>
<tr>
<td>Full</td>
<td>737M (100%)</td>
<td>2.75</td>
</tr>
<tr>
<td>Adapters</td>
<td>12M (1.70 %)</td>
<td>2.80</td>
</tr>
<tr>
<td>LoRA</td>
<td>9M (1.26%)</td>
<td>2.78</td>
</tr>
<tr>
<td>Inference</td>
<td>/</td>
<td>2.75</td>
</tr>
</tbody>
</table>

Table 1: The breakdown of memory footprint. "Activations" contain the intermediate results and optimizer states. Model: T5-Large; mini-batch size: 16; sequence length: 128.
Challenges

- The fundamental contradiction between intensive LLM fine-tuning workload and constrained on-board resources.

- The computational capabilities of edge devices are constrained.

<table>
<thead>
<tr>
<th>Model</th>
<th>DistilBert</th>
<th>Bert-L</th>
<th>GPT2-L</th>
<th>OPT-L</th>
<th>OPT-XL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nano-M</td>
<td>0.37s</td>
<td>2.43s</td>
<td>OOM</td>
<td>OOM</td>
<td>OOM</td>
</tr>
<tr>
<td>Nvidia A100</td>
<td>5ms</td>
<td>20ms</td>
<td>29ms</td>
<td>27ms</td>
<td>38ms</td>
</tr>
<tr>
<td>Memory Footprint</td>
<td>130MB</td>
<td>680MB</td>
<td>1.6GB</td>
<td>2.6GB</td>
<td>5.4GB</td>
</tr>
</tbody>
</table>

**TABLE I**

**Inference Latency and Mem. Footprint of Transformer Models**

<table>
<thead>
<tr>
<th>Device</th>
<th>AI Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jetson Nano</td>
<td>472 GFLOPS</td>
</tr>
<tr>
<td>NVIDIA A100</td>
<td>312 TFLOPS</td>
</tr>
</tbody>
</table>

121x performance gap.
Challenges

- The fundamental contradiction between intensive LLM fine-tuning workload and constrained on-board resources

➢ On-device fine-tuning is hindered by the memory wall.

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Trainable Parameters</th>
<th>Memory Footprint (GB)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Weights</td>
<td>Activations</td>
</tr>
<tr>
<td>Full</td>
<td>737M (100%)</td>
<td>2.75</td>
<td>5.33</td>
</tr>
<tr>
<td>Adapters</td>
<td>12M (1.70%)</td>
<td>2.80</td>
<td>4.04</td>
</tr>
<tr>
<td>LoRA</td>
<td>9M (1.26%)</td>
<td>2.78</td>
<td>4.31</td>
</tr>
<tr>
<td>Inference</td>
<td>/</td>
<td>2.75</td>
<td>/</td>
</tr>
</tbody>
</table>

Table 1: The breakdown of memory footprint. "Activations" contain the intermediate results and optimizer states. Model: T5-Large; mini-batch size: 16; sequence length: 128.

Incurs a peak memory footprint that is often unaffordable for edge devices.

<table>
<thead>
<tr>
<th>Device</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jetson Nano</td>
<td>4 GB</td>
</tr>
<tr>
<td>NVIDIA A100</td>
<td>40 GB/ 80 GB</td>
</tr>
</tbody>
</table>
Opportunities

Edge environment often comprise a rich set of trusted idle edge devices.

➢ Prevalent edge environments like smart homes usually comprise a group of trusted idle devices beyond a single terminal (e.g., mobile phones, laptops, and smart-home devices owned by the same user or family).

➢ These accompanying devices are typically in physical proximity and can be associated as a resource augmentation for in-situ personal LLMs fine-tuning.
Algorithm-system codesign

(Algorithm): In light of the side-tuning techniques, we employ not only parameter but also time and memory-efficient personal LLMs finetuning techniques with Parallel Adapters, which provides a dedicated gradient “highway” for the trainable parameters.

(System): We leverage edge devices in physical proximity and associate them as an edge resource pool for in-situ personal LLMs fine-tuning.
Algorithm-system codesign

(Algorithm): In light of the side-tuning techniques, we employ not only parameter but also time and memory-efficient personal LLMs finetuning techniques with Parallel Adapters, which provides a dedicated gradient “highway” for the trainable parameters.

(System): We leverage edge devices in physical proximity and associate them as an edge resource pool for in-situ personal LLMs fine-tuning.
PAC: A Time and Memory Efficient Collaborative Edge AI Framework for Personal LLMs Fine-Tuning

➢ Fine-Tuning LLMs with Parallel Adapters

- The parameters of backbone transformer are **frozen**.

- Parallel adapters are a lightweight, **separate network** that takes the intermediate activations from the backbone LLM as input and generates predictions. (Skip the backward propagation from the LLM backbone!)

![Diagram](image-url)
PAC: A Time and Memory Efficient Collaborative Edge AI Framework for Personal LLMs Fine-Tuning

➢ PAC Activation Cache for Parallel Adapters

- During the first epoch, when processing a new input sequence, cache all the input activations required by the Parallel Adapters that are obtained from the LLM backbone.

- In subsequent fine-tuning epochs using the same input sequence, we can reuse cached activations. (Skip the forward propagation from the LLM backbone!)
PAC Activation Cache for Parallel Adapters

- Significantly accelerating the fine-tuning process.
- Reducing the memory footprint by allowing the release of the memory space occupied by the LLM parameters.

Skip both the forward and backward propagation through the LLM backbone entirely!
PAC: A Time and Memory Efficient Collaborative Edge AI Framework for Personal LLMs Fine-Tuning

**Algorithm-system codesign**

*(Algorithm):* In light of the side-tuning techniques, we employ not only parameter but also time and memory-efficient personal LLMs finetuning techniques with Parallel Adapters, which provides a dedicated gradient “highway” for the trainable parameters.

*(System):* We leverage edge devices in physical proximity and associate them as an edge resource pool for in-situ personal LLMs fine-tuning.
PAC: A Time and Memory Efficient Collaborative Edge AI Framework for Personal LLMs Fine-Tuning

➢ Data & Pipeline Hybrid Parallelism for LLMs Fine-Tuning

Step 1: PAC first divides an LLM into multiple stages where each contains a stage model composed of a set of consecutive transformer layer.
Data & Pipeline Hybrid Parallelism for LLMs Fine-Tuning

Step 2: Edge devices are allocated into several device groups, each comprising one or more devices. PAC maps each stage to a group, with the stage model replicated across all devices within that group.
PAC: A Time and Memory Efficient Collaborative Edge AI Framework for Personal LLMs Fine-Tuning

Data & Pipeline Hybrid Parallelism for LLMs Fine-Tuning

Step3: A mini-batch is divided into several micro-batches for concurrent processing to enhance parallelism. If a device cluster hosts multiple devices, micro-batches are further subdivided.
PAC: A Time and Memory Efficient Collaborative Edge AI Framework for Personal LLMs Fine-Tuning

➢ Data & Pipeline Hybrid Parallelism for LLMs Fine-Tuning

We design a dynamic programming algorithm to search for the optimal partitioning method and device grouping method for LLMs.
➢ Cache-Enabled Collaborative Edge Fine-Tuning of Parallel Adapters

**Step 1:** Perform collective communication to redistribute the Parallel Adapters’ parameters and locally cached activations across all devices.
PAC: A Time and Memory Efficient Collaborative Edge AI Framework for Personal LLMs Fine-Tuning

- Cache-Enabled Collaborative Edge Fine-Tuning of Parallel Adapters

**Step 2:** Release the memory usage of the backbone model by simply loading parallel adapters for fine-tuning.
Step 3: The devices then utilize cached activations to fine-tune the parallel adapters in a data-parallel manner.
EVALUATION

➢ Implementation and Setups

● Models:

<table>
<thead>
<tr>
<th>Model</th>
<th>Structure</th>
<th>Layers</th>
<th>Heads</th>
<th>Hidden Size</th>
<th>Param. Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>T5-Base [20]</td>
<td>en-de</td>
<td>12</td>
<td>12</td>
<td>768</td>
<td>0.25B</td>
</tr>
<tr>
<td>BART-Large [13]</td>
<td>en-de</td>
<td>12</td>
<td>16</td>
<td>1024</td>
<td>0.41B</td>
</tr>
<tr>
<td>T5-Large [20]</td>
<td>en-de</td>
<td>24</td>
<td>16</td>
<td>1024</td>
<td>0.74B</td>
</tr>
</tbody>
</table>

● Edge Environment Setup:

● 8 NVIDIA Jetson Nanos

● network bandwidth: 1000Mbps
EVALUATION

➢ Implementation and Setups

● Baseline Methods:
  ● Standalone + Full model fine-tuning/Adapters/LoRA
  ● Eco-FL (ICPP 2022) + Full model fine-tuning/Adapters/LoRA
  ● EDDL (SEC 2021) + Full model fine-tuning/Adapters/LoRA
➢ End-to-end Performance

- PAC accelerates fine-tuning up to $8.64\times$ faster than existing state-of-the-art methods.

Table 2: Training durations (in hours) for different methods: 3 epochs for MRPC and STS-B, and 1 epoch for SST-2 and QNLI.

<table>
<thead>
<tr>
<th>Fine-tuning Techniques</th>
<th>Baseline Methods</th>
<th>T5-Base</th>
<th>BART-Large</th>
<th>T5-Large</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRPC</td>
<td>STS-B</td>
<td>SST-2</td>
<td>QNLI</td>
</tr>
<tr>
<td>Full Model</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standalone</td>
<td>OOM</td>
<td>OOM</td>
<td>OOM</td>
<td>OOM</td>
</tr>
<tr>
<td>Eco-FL</td>
<td>0.45</td>
<td>0.71</td>
<td>2.74</td>
<td>4.32</td>
</tr>
<tr>
<td>EDDL</td>
<td>OOM</td>
<td>OOM</td>
<td>OOM</td>
<td>OOM</td>
</tr>
<tr>
<td>Adapters</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standalone</td>
<td>1.21</td>
<td>1.9</td>
<td>7.29</td>
<td>11.51</td>
</tr>
<tr>
<td>Eco-FL</td>
<td>0.39</td>
<td>0.61</td>
<td>2.35</td>
<td>3.71</td>
</tr>
<tr>
<td>EDDL</td>
<td>0.34</td>
<td>0.53</td>
<td>2.06</td>
<td>3.25</td>
</tr>
<tr>
<td>LoRA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standalone</td>
<td>1.21</td>
<td>1.89</td>
<td>7.28</td>
<td>11.49</td>
</tr>
<tr>
<td>Eco-FL</td>
<td>0.41</td>
<td>0.64</td>
<td>2.45</td>
<td>3.87</td>
</tr>
<tr>
<td>EDDL</td>
<td>0.31</td>
<td>0.48</td>
<td>1.86</td>
<td>2.94</td>
</tr>
<tr>
<td>Parallel Adapters</td>
<td>PAC</td>
<td>0.14</td>
<td>0.22</td>
<td>1.34</td>
</tr>
</tbody>
</table>

PAC (Ours)
End-to-end Performance

- PAC decrease the peak memory up to **88.16%** compared to baselines.
End-to-end Performance

- PAC can achieve comparable or even superior fine-tuned model performance.

<table>
<thead>
<tr>
<th>Fine-tuning Techniques</th>
<th>T5-Base</th>
<th>BART-Large</th>
<th>T5-Large</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRPC</td>
<td>STS-B</td>
<td>SST-2</td>
</tr>
<tr>
<td>Full Model</td>
<td>89.71</td>
<td>90.94</td>
<td>94.03</td>
</tr>
<tr>
<td>Adapters</td>
<td>88.73</td>
<td>90.51</td>
<td>93.58</td>
</tr>
<tr>
<td>LoRA</td>
<td>86.27</td>
<td>90.73</td>
<td>93.69</td>
</tr>
<tr>
<td>Mean Value</td>
<td>88.24</td>
<td>90.73</td>
<td>93.77</td>
</tr>
</tbody>
</table>

Difference from Mean

Parallel Adapters (Ours) 88.24 90.43 93.46 93.25 87.71 90.54 95.25 93.68 91.7 91.57 95.76 93.7

Difference from Mean

+0.00  -0.30  -0.31  +0.11  +0.29  -0.03  -0.02  -0.37  +0.06  +0.32  +0.12  -0.15
Thanks for listening

Bei Ouyang*1, Shengyuan Ye*1, Liekang Zeng2, Tianyi Qian1, Jingyi Li1, Xu Chen†1

1Sun Yat-sen University, 2HKUST(GZ)