Benchmarking Process

Algorithm 1: MELT (Experiment Process)

Pseudocode for MELT experiments. Functionality of undefined methods in comment. Prefixed methods run on the device in prefix (e.g., Monsoon, device).

Input: PhoneLab, JetsonLab, Monsoon, GPIO, YKUSH, device, Qdevice
iterations, samplingFrequency, betweenExpSleep

1. **PowerOn**(device)
2. if device.platform == "ios" :
3. ConnectBT(device) # connect as HID device via Bluetooth
4. UnlockScreen(device) # unlock screen with passcode over HID
5. SyncClocks(device) # sync host and guest clocks
6. apiAddress = StartRESTServer() # start REST service on host
7. for exp in Qdevice : # iterate over experiments in the queue
8. Push([exp.model, exp.conversations], device) # push dependencies
9. Apply([exp.conf, exp.model, device]) # edit model conf and execution parameters on device
10. for it = 0: it < iterations + 1 :  # start monitoring
11. StartMonitoring(Monsoon, device)
12. RunExperiment(exp, device)
13. StopMonitoring(Monsoon, device) # disable monitoring
14. CollectMeasurements(exp, device) # get results from FS
15. sleep(betweenExpSleep) # sleep between runs

---

def **PowerOn**(GPIO, YKUSH, device):
if device in PhoneLabdevices :
   GPIO.EnableRail(device.rails) # enable rail through GPIO
   YKUSH.PowerOn(device) # enable YKUSH USB of device
   if Monsoon.SetYourCurr(device) # configure Monsoon power out
   else Wait(device) # wait until device is responsive

def **StartMonitoring**(Monsoon, YKUSH, device):
   if device in PhoneLab.devices :
      YKUSH.DisableUSB()
      Monsoon.MeasurementMode('on', samplingFrequency)
   elif device in JetsonLab.devices :
      Jetsonlab.ScheduleEvents(samplingFrequency)
      JetsonLab.Monitor('on') # poll SysFS
   else
      report = device.Trace(prompt) # run inference
      device.Write(report, exp.conf.outputPath) # results to FS

def **RunExperiment**(exp, device, apiAddress):
   http.post('start', apiAddress) # notify through REST service
   http.post('stop', apiAddress) # notify through REST service
Devices, Models & Frameworks

Table 1: Device Farm of MELT

<table>
<thead>
<tr>
<th>Device Model</th>
<th>SoC</th>
<th>Mem.</th>
<th>Battery</th>
<th>OS version</th>
<th>Year</th>
<th>Tier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co-ordinator &amp; Builder</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Raspberry Pi 4</td>
<td>Broadcom BCM2711</td>
<td>8GB</td>
<td>-</td>
<td>RPi OS 11.9</td>
<td>2019</td>
<td>-</td>
</tr>
<tr>
<td>Mac Studio</td>
<td>M2 Max</td>
<td>32GB</td>
<td>-</td>
<td>macOS 14.1.2</td>
<td>2023</td>
<td>-</td>
</tr>
<tr>
<td>Mobile</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Galaxy S23</td>
<td>Snapdragon 8 Gen 2</td>
<td>8GB</td>
<td>3785 mAh</td>
<td>Android 14</td>
<td>2023</td>
<td>High</td>
</tr>
<tr>
<td>Pixel 6a</td>
<td>Tensor Core</td>
<td>8GB</td>
<td>4410 mAh</td>
<td>Android 13</td>
<td>2023</td>
<td>Mid</td>
</tr>
<tr>
<td>iPhone 14 Pro</td>
<td>A16 Bionic</td>
<td>6GB</td>
<td>3200 mAh</td>
<td>iOS 17.3.1</td>
<td>2022</td>
<td>High</td>
</tr>
<tr>
<td>iPhone SE</td>
<td>A15 Bionic</td>
<td>4GB</td>
<td>1821 mAh</td>
<td>iOS 17.3.1</td>
<td>2022</td>
<td>Mid</td>
</tr>
<tr>
<td>Edge</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jetson Orin AGX</td>
<td>NVIDIA Carmel</td>
<td>64GB</td>
<td>-</td>
<td>Ubuntu 20.04</td>
<td>2022</td>
<td>High</td>
</tr>
<tr>
<td>Ampere GPU</td>
<td></td>
<td></td>
<td></td>
<td>(L4T 35.2.1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jetson Orin Nano</td>
<td>Arm Cortex-A78AE</td>
<td>8GB</td>
<td>-</td>
<td>Ubuntu 20.04</td>
<td>2022</td>
<td>Mid</td>
</tr>
<tr>
<td>Ampere GPU</td>
<td></td>
<td></td>
<td></td>
<td>(L4T 35.4.1)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Supported pretrained models

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Size</th>
<th>Type</th>
<th>HuggingFace Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td>TinyLlama</td>
<td>1.1B</td>
<td>Decoder</td>
<td>TinyLlama/TinyLlama-1.1B-Chat-v0.5</td>
</tr>
<tr>
<td>Zephyr-3B</td>
<td>3B</td>
<td>Decoder</td>
<td>stabilityai/stablelm-zephyr-3b</td>
</tr>
<tr>
<td>MistralAI-7B</td>
<td>7B</td>
<td>Decoder</td>
<td>mistralai/Mistral-7B-Instruct-v0.1</td>
</tr>
<tr>
<td>Gemma</td>
<td>2B</td>
<td>Decoder</td>
<td>google/gemma-2b-it</td>
</tr>
<tr>
<td>Llama-2</td>
<td>13B</td>
<td>Decoder</td>
<td>meta-llama/Llama-2-13b-chat-hf</td>
</tr>
</tbody>
</table>

Table 3: Frameworks and platforms supported by MELT.

<table>
<thead>
<tr>
<th>Framework</th>
<th>Backend</th>
<th>Version</th>
<th>Supported Platforms</th>
<th>Quantization</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLC-LMM</td>
<td>LLVM [15]</td>
<td>0.46.48*</td>
<td>Android (GPU), iOS (Metal), Linux (CUDA)</td>
<td>GroupQuantization [93], GPT2 [32], FasterTransformerRow-wiseQuantization</td>
</tr>
<tr>
<td>llama.cpp</td>
<td>llama.cpp [34]</td>
<td>b2822+</td>
<td>Android (CPU, GPU), Linux (CUDA)</td>
<td>k-quants [95]</td>
</tr>
<tr>
<td>LLMFarm</td>
<td>llama.cpp [34]</td>
<td>727268</td>
<td>iOS (Metal)</td>
<td></td>
</tr>
</tbody>
</table>

* We used version 784538 for supporting Gemma models and Llama-2/7B on Android.
+ We used version 0b4e69 for supporting Gemma models.
Evaluation

• **Macro-benchmarks:** Run at a conversational level
  • Sampled from OpenAssistant/oasst1 dataset
  • English, at least 5 turns of conversations, < 2000 words

![Diagram](image)

(a) CDFs of conversation (# prompts) and prompt lengths (# words)
(b) Part of speech categories distribution across prompts

Figure 2: Qualitative analysis of prompts used for macro-experiments to assess the behaviour of LLM-powered chats on device.

• **Micro-benchmarks:** Run with fixed set input/output tokens
  • \(<\text{EOS}>\) is disregarded
  • max generated length is set
Throughput and Discharge

**Figure 1:** Throughput across frameworks and devices

**Figure 2:** Discharge per token across frameworks and devices
Power over Time

- **Power draw:**
  - iPhone 14 Pro tend to boost their power draw very high, reaching a max of 13.8W sustained and 18W instantaneous wattage.
  - Galaxy S23 had a sustained draw of 8.5W and max of 14W.
  - Each device can run: (a) 542.48, (b) 490.05 and (c) 590.93 prompts until battery is depleted.

**Figure:** LLM execution timeline of Zephyr-3B (4-bit quantized) across devices and frameworks. We use moving average of 500 points for smoothing the timeline. We annotate the number of generated tokens per inference.
Figure: Model loading times per device. Each device supports different set of models, based on available memory and framework.
QoE: Sustained Inference

**Figure 1:** Continuous inference on mobile and edge devices with Zephyr-3B (4-bit)

**Figure 2:** Temperature after one full conversation on Zephyr-3B (4-bit) on MLC-LLM
**Micro-benchmarks**

**Figure 1:** Per-op benchmarks of Llama-7B (3-bit) with MLC-LLM on Galaxy S23. These are operations generated by the TVM compiler. The variants may signify different implementation or hyperparameters tuned for performance.

**Figure 2:** Memory trace for iPhone 14 Pro when running Zephyr-3B on LLMFarm
Impact of Quantization

Table 4: Evaluation datasets description

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Task</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HellaSwag [119]</td>
<td>Common-sense NLI</td>
<td>70k</td>
<td>Given an event description, select the most likely continuation.</td>
</tr>
<tr>
<td>Winogrande [89]</td>
<td>Common-sense NLI</td>
<td>44k</td>
<td>Benchmark for common-sense reasoning, designed not to be easily solvable by statistical models and plain word associations.</td>
</tr>
<tr>
<td>ARC-(E,C) [16]</td>
<td>Reasoning NLI</td>
<td>52k, 26k</td>
<td>Science and language exam questions from a variety of sources. E: Easy; C: Complex</td>
</tr>
</tbody>
</table>

Figure: Model size vs. accuracy for different models, quantization schemes and precisions
Runtime at the Edge

**Figure:** LLM execution on Jetson devices across energy modes
Summary & Takeaways

• First study for the feasibility of deploying LLMs on mobile and edge devices that can act as basis for future research.

• We offer an open benchmarking framework that allows performance, energy, thermal and accuracy evaluations of Transformer models on-device.

• Key takeaways:
  • Performance heterogeneity across different target devices.
  • On-device inference is still memory-bound, but also thermally-bound.
  • Quantization can come at a non-negligible performance cost.
  • Tractability does not imply deployability. QoE is severely affected on > 3B models.
Future Directions

Model Compression
- Low-rank approximations
- Broader support for dynamic graph models: Early-exiting, SkipDecode, Speculative Decoding

Execution
- New generation of NPU hardware
  - Specifically designed for efficient Matrix-Matrix multiplication.
  - Efficient operator support
- Boost in energy efficiency and thermal behaviour of the device.

Deployment
- Hybrid collaborative execution w/ TEEs (see Apple Intelligence offloading paradigm)
- Common backbone abstraction

Model Changes
- The rise of SLMs: routing to specialised agents
- Unimodal -> multimodal models

Future Directions
- The rise of SLMs: routing to specialised agents
- Unimodal -> multimodal models
- New generation of NPU hardware
  - Specifically designed for efficient Matrix-Matrix multiplication.
  - Efficient operator support
- Boost in energy efficiency and thermal behaviour of the device.
- Hybrid collaborative execution w/ TEEs (see Apple Intelligence offloading paradigm)
- Common backbone abstraction
Acknowledgements

Shout-out to my wonderful collaborators

Kleomenis Katevas
Brave Software

Lorenzo Minto
Brave Software

Hamed Haddadi
Brave Software &
Imperial College London
Thank You

Q&A