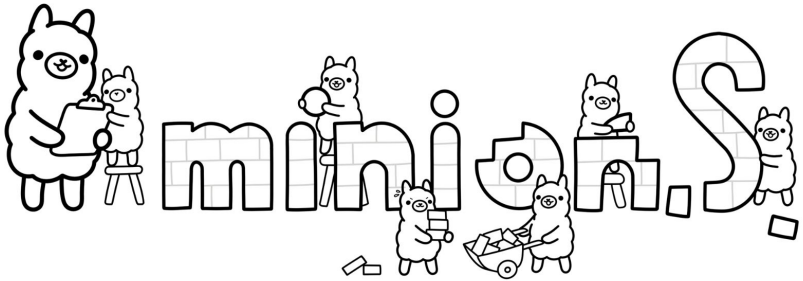
Minions: Cost-efficient Collaboration Between On-device and Cloud Language Models

Avanika Narayan*, Dan Biderman*, Sabri Eyuboglu*, Avner May, Scott Linderman, James Zhou, Christopher Ré

Presentation @ DAM Retreat (July 1st, 2025)



SCAN ME

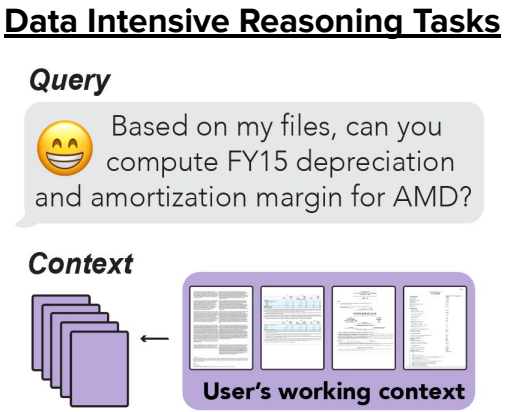
Motivation: Data Intensive Reasoning Tasks

Users want to apply LMs to large volumes of personal, on-device data!
These tasks require “always-on” processing

💻 Scan codebases for bugs + make edits

💰 Analyze financial filings

🏥 Predict from health records (i.e., visit notes)
… and much more



Motivation: Improvement of Frontier Models

We 🩷 *frontier models* 🌐. And they are getting quite good at data intensive reasoning tasks due to

Longer context lengths

Test-time scaling techniques

Motivation: Large Cost of Frontier Models

But...applying them to long context tasks is expensive 💵

Problem: ***Applying frontier models to data-intensive tasks is expensive*** 💸💸💸

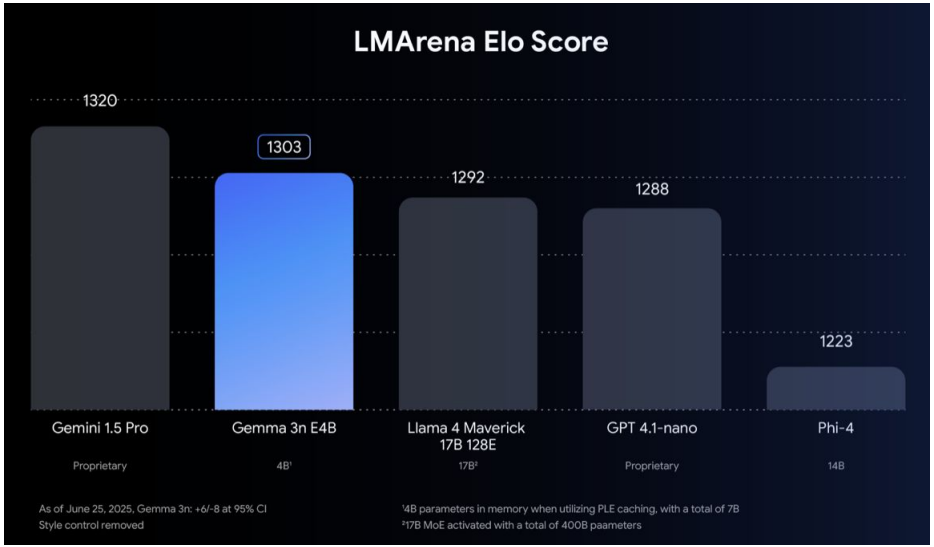
e.g. o3 costs >$10 per query on a million token repository

Motivation: The Rise of On-device ML

💻 At the same time, **on-device** machine learning is emerging

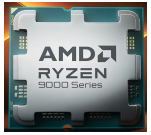
1. Smaller, **multimodal** LMs (*e.g.* 1-8B parameters) are getting increasingly good!

Gemma 3n



These models can fit on consumer devices: mobile devices + laptops!

Motivation: The Rise of On-device ML



💻 At the same time, **on-device** machine learning is emerging

② And the ***on-device*** hardware in personal computers and smartphones is becoming LLM native

Model (quant / size)	Mobile / Laptop testbed	Runtime backend	↓ Tokens / sec	First-token latency (s)
Gemma 3n E4B-it `dynamic-int4`	Samsung S25 Ultra — Snapdragon 8 Gen 4	LiteRT CPU (4 threads)	**12.8**	9.2 huggingface.co
Gemma 3n E4B-it `dynamic-int4`	*same phone*	LiteRT GPU	**16.1**	15.1 huggingface.co
Llama 2 7B-Chat `w4a16`	Galaxy S24 — Snapdragon 8 Gen 3	Qualcomm QNN NPU	**12.85**	1.50 huggingface.co
Llama 2 7B-Chat `w4a16`	Snapdragon 8 Elite QRD (dev phone)	QNN NPU	**17.94**	1.44 huggingface.co
Apple on-device Foundation Model (~3 B, mixed 2-/4-bit)	iPhone 15 Pro — A17 Pro	FoundationModels (ANE)	**30**	≈0.06 – 0.15 † machinelearning.apple.com

Problem: today, on-device models aren't used for data-intensive tasks!!

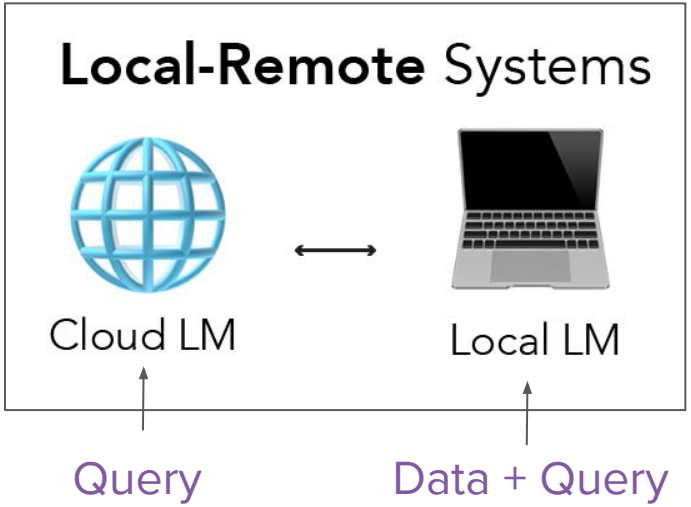
Problem Setup

Faced with these two observations…

- **Problem:** *Applying frontier models to data-intensive tasks is expensive* 💸💸💸
- **Problem:** On-device hardware is underutilized

…we ask:

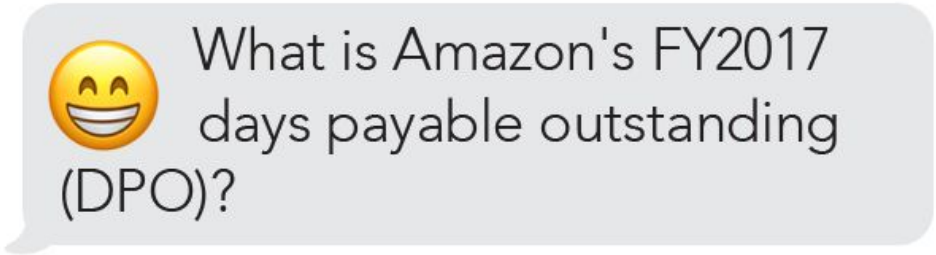
*How can a small LM **on-device** collaborate with a frontier LM **in the cloud** to reduce inference costs w/o data leaving the device?*



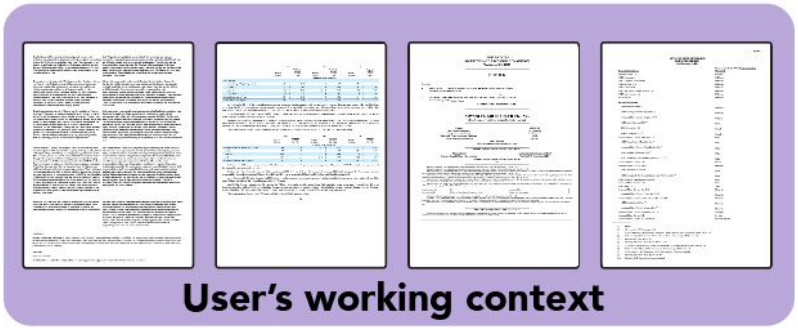
Experimental details

We consider three data-intensive reasoning tasks:

1. FinanceBench (Financial tasks over 100 page PDFs)
2. LongHealth (Medical tasks over a patient's medical record)
3. QASPER (Scientific questions over collections of papers)



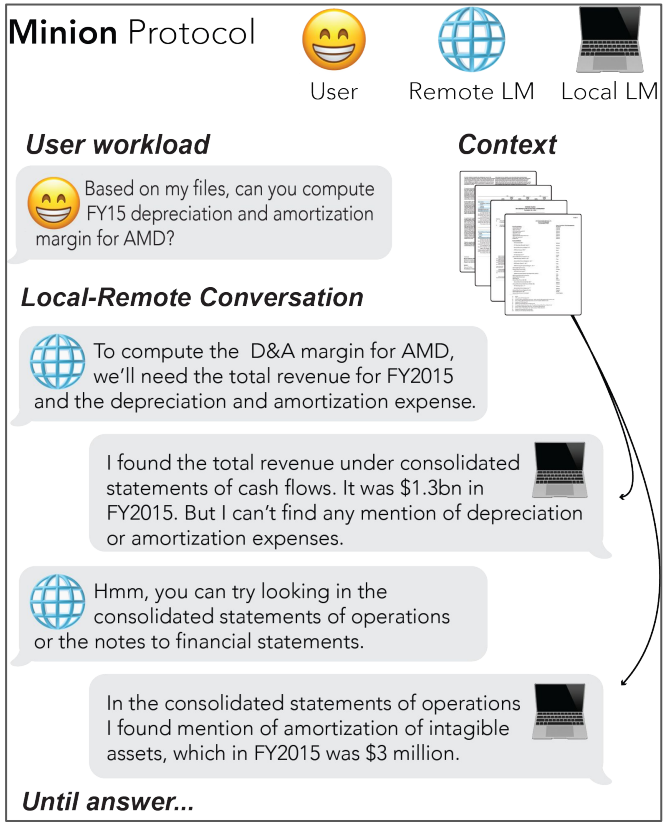
DPO is defined as: 365 * (average accounts payable between FY2016 and FY2017) / (FY2017 COGS + change in inventory between FY2016 and FY2017).



Company 10-K documents exceeding 100 pages (approx. 128k tokens)

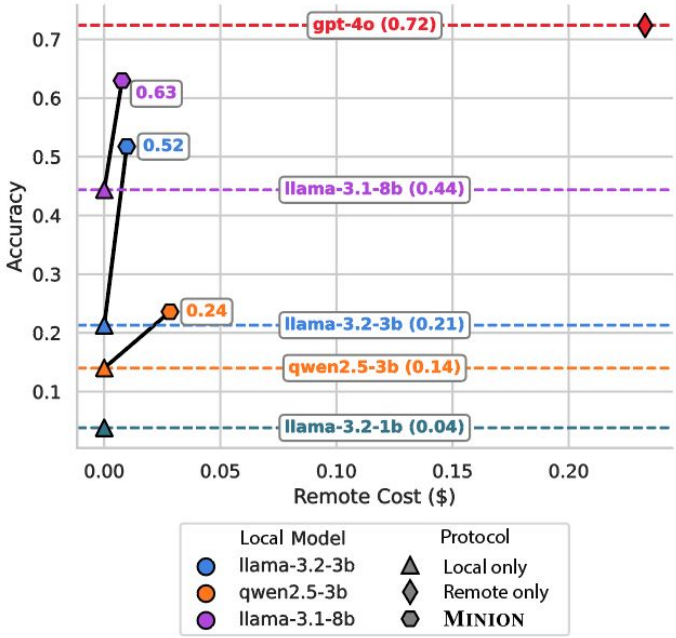
Minion Communication Protocol

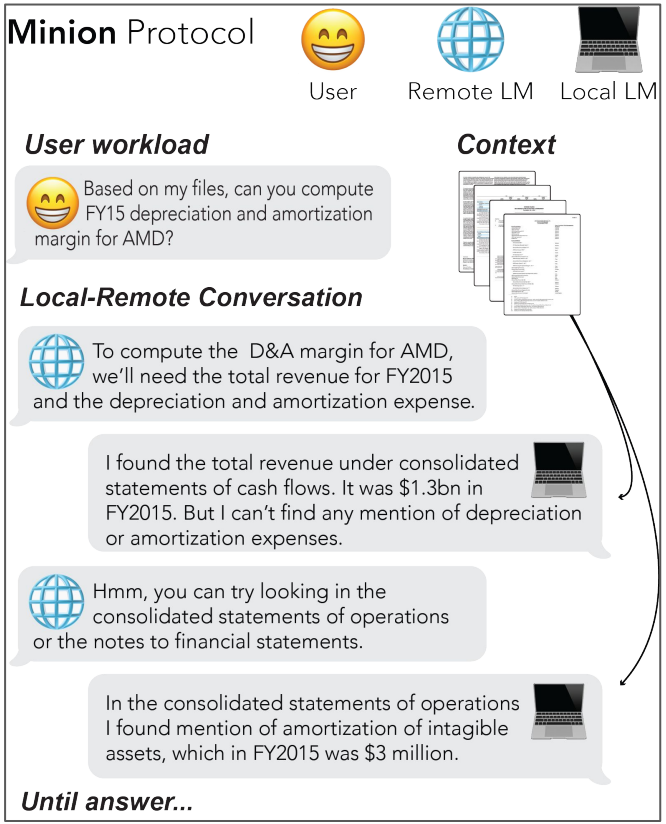
First, we experimented with the simplest protocol we could think of:
just let the models talk to each other in a free-form chat!



Minion Communication Protocol

Minion achieves 30.4X cost reduction over remote-only while recovering 87% of accuracy.

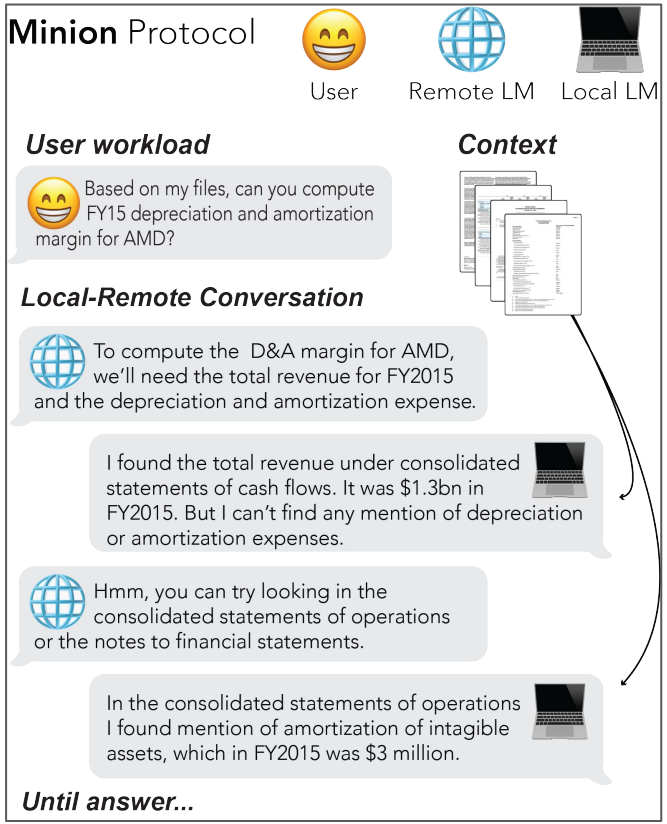




Minion Communication Protocol

Minion achieves 30.4X cost reduction over remote-only while recovering 87% of accuracy.

*But we want to recover close to **100% of the accuracy!!***

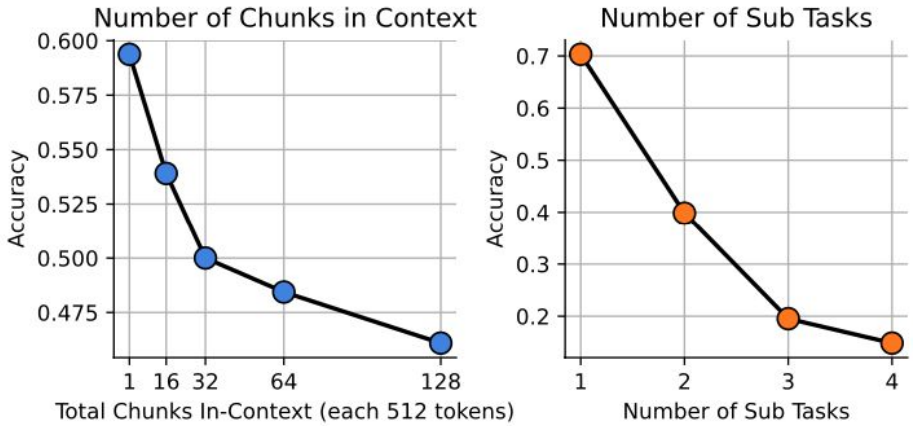


Minion Communication Protocol

We identify two reasons why the Minion protocol struggles to recover 100% of the performance of the remote model alone:

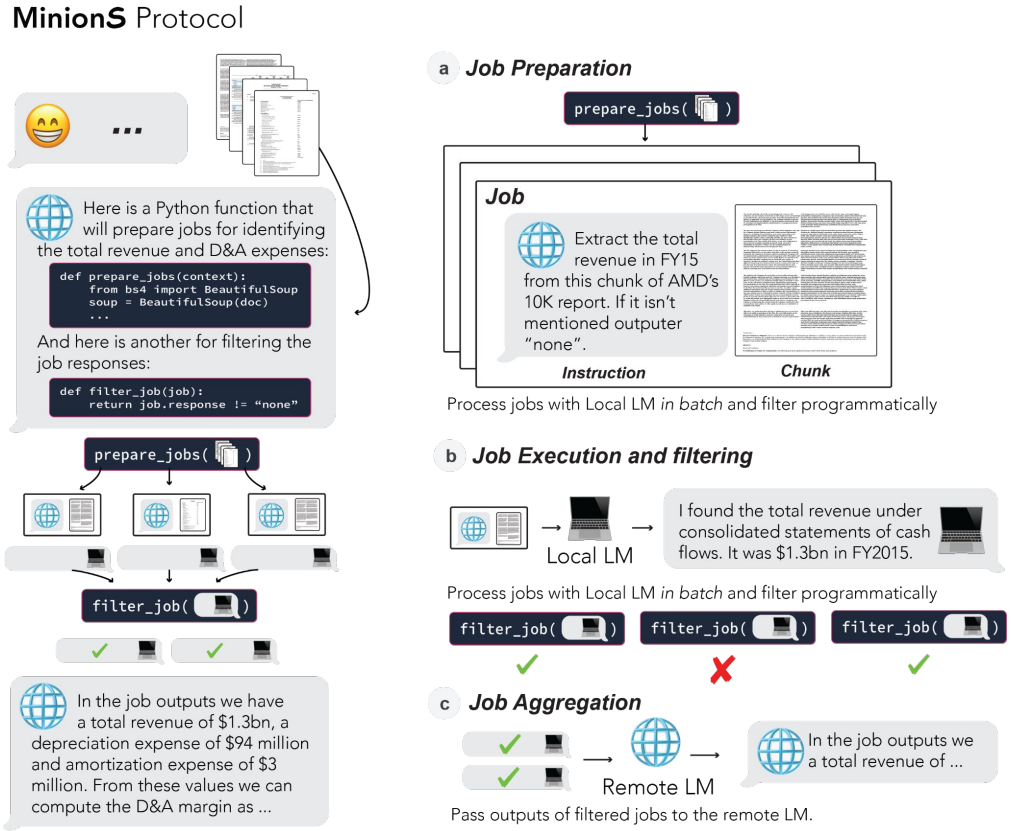
1. Small LMs struggle to reason across **long contexts**

2. Small LMs struggle to handle **multi-part instructions**



*Also, Minion achieves **low utilization of edge hardware** because batch sizes are small!*

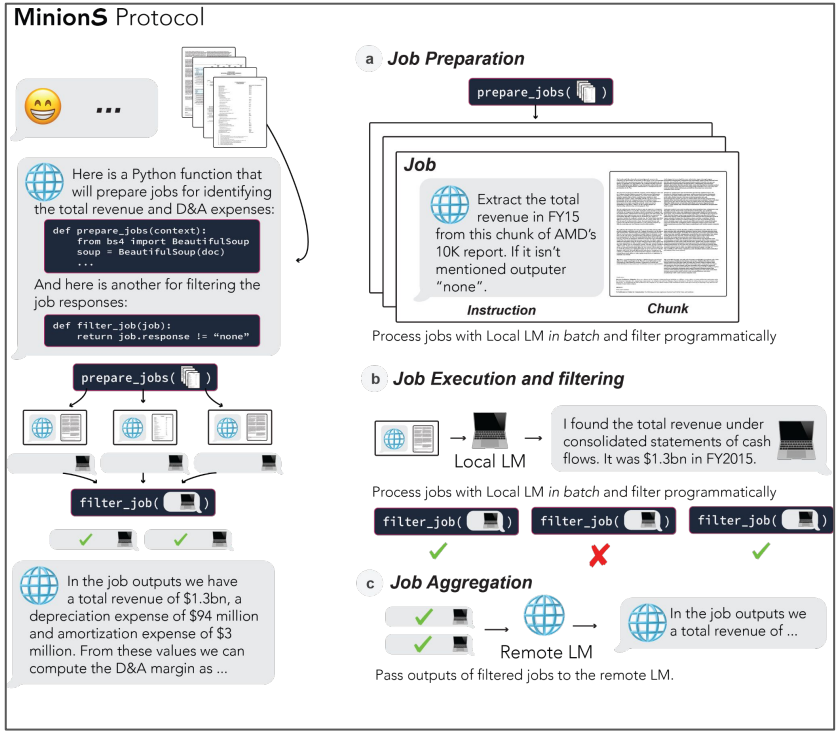
MinionS Communication Protocol



Minions Communication Protocol

Introduces a three step, multi-turn framework:

1. **Decompose**: remote LM breaks task into smaller subtasks
2. **Execute:** run subtasks locally in parallel
3. **Aggregate:** remote LM merges results for final answer or loops back to (1)



Minions Communication Protocol

1 Decompose (Remote LM 🌐)

Writes a script that

➔ breaks task into subtasks

➔ chunks context

➔ provides advice

```python
def prepare_jobs(context):
    # Split input document into overlapping chunks
    chunks = chunk_by_section(document=context[0], size=3000, overlap=20)

    # Define task instruction
    task = "Extract operating expenses and net sales for 2019. Include percentages."

    # Create one job per chunk
    job_manifests = []
    for chunk in chunks:
        job = {
            "chunk": chunk,
            "task": task,
            "advice": "Focus on financial data and percentages."
        }
        job_manifests.append(job)

    return job_manifests
```

```python
def chunk_by_section(document, size, overlap):
    sections = []
    start = 0

    while start < len(document):
        end = start + size
        section = document[start:end]
        sections.append(section)
        start += size - overlap

    return sections
```

Minions Communication Protocol

2 Execute (Local LM 💻)

Run worker jobs in parallel →

```
# Assemble one message per job-chunk
worker_chats = []

for manifest in job_manifests:
    msg = {
        "role": "user",
        "content": format_template(
            context = manifest.chunk,
            task    = manifest.task,
            advice  = manifest.advice
        )
    }
    worker_chats.append(msg)

# Send all messages in a single call
responses = local_client.chat(worker_chats)
```

Filter outputs to only include jobs that yielded an answer. →

```
def filter_fn(job):
    """Reject jobs with missing or 'none' answers."""
    answer = job.output.answer

    if answer is None or answer.strip().lower() == "none":
        return False
    return True
```

Minions Communication Protocol

3 Aggregate (Remote LM 🌐)

Parses all job outputs

Decides to either provide an answer or loop back to decompose step

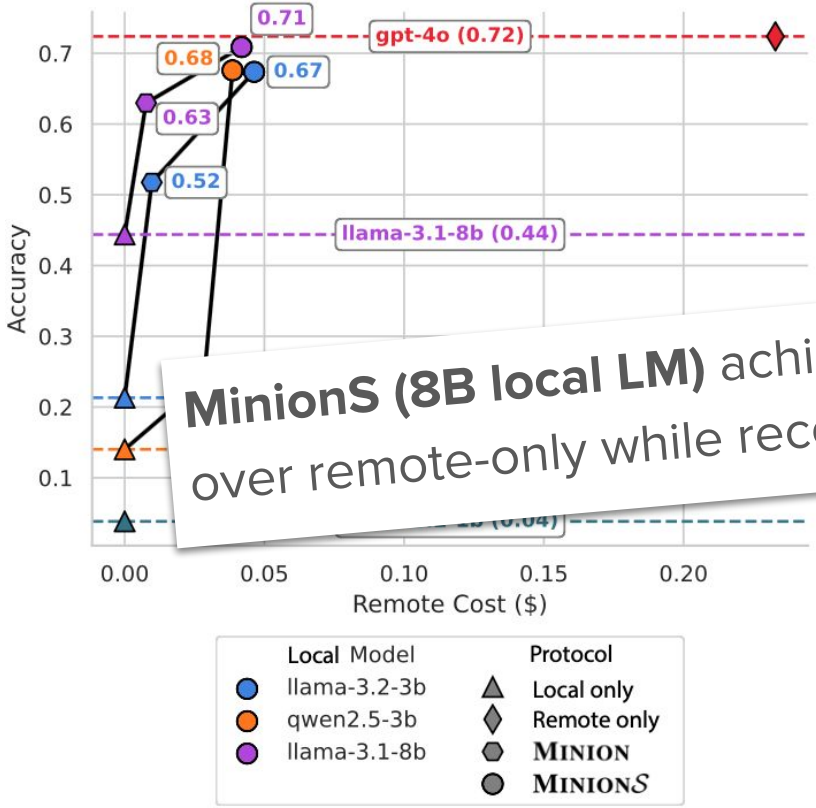
```
REMOTE_ANSWER= """\
## Inputs
1. Question to answer:
{question}

2. Collected Job Outputs (from junior models):
{extractions}

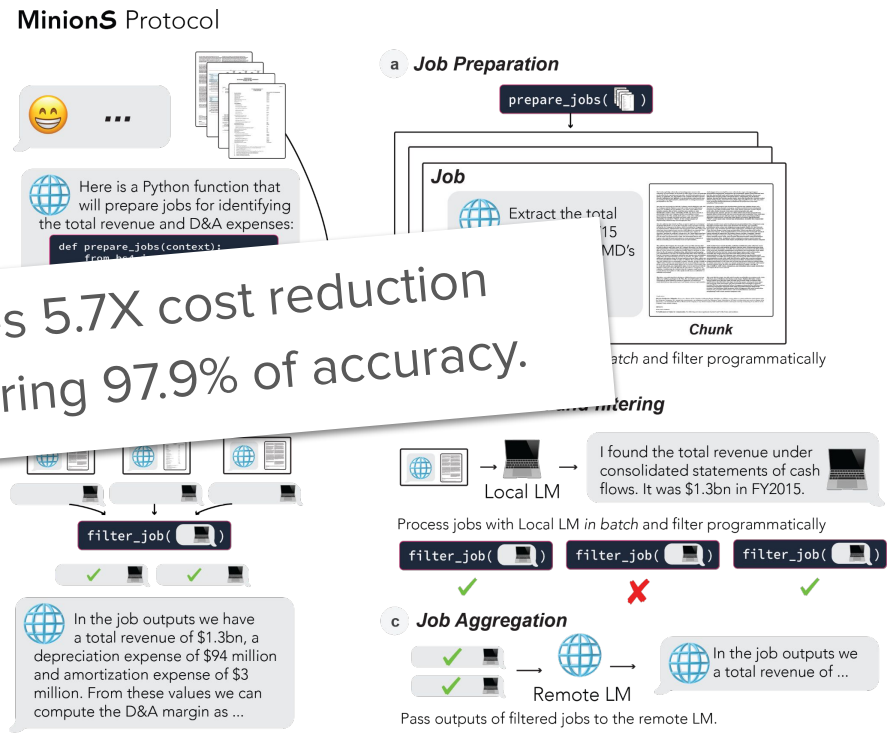
## Instructions: Please inspect the question and the Job Outputs. Then decide whether to
finalize the answer or request additional details, and return the JSON object accordingly.

Follow the ANSWER GUIDELINES in the conversational history above.
```json
{{
"decision": "…",
"explanation": "…",
"answer": "… or None",
"missing_info": "… or None"
}}
"""
```
```

Minions Communication Protocol



MinionS (8B local LM) achieves 5.7X cost reduction over remote-only while recovering 97.9% of accuracy.



Minions: Investigating Different Tradeoff Knobs

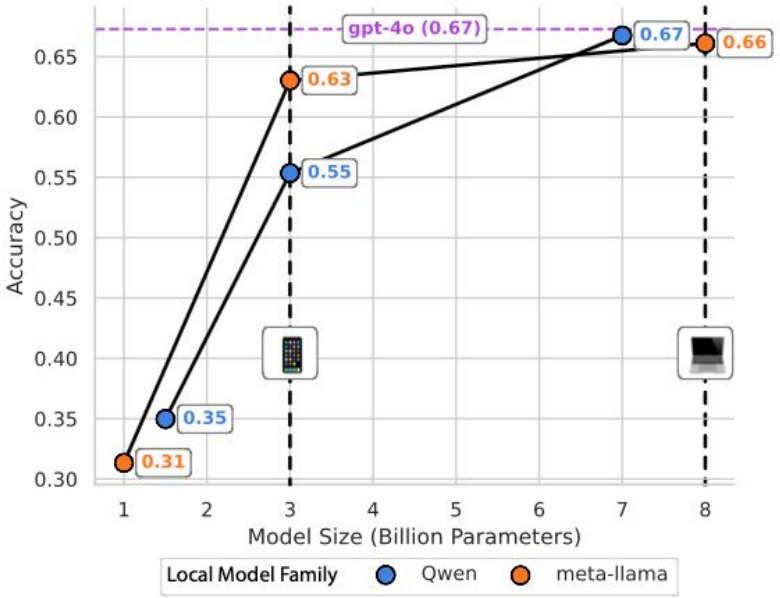
1. **Model size:** How does the size of the local model affect performance?

2. **Edge parallelism:** How should we parallelize work at the edge to improve quality?

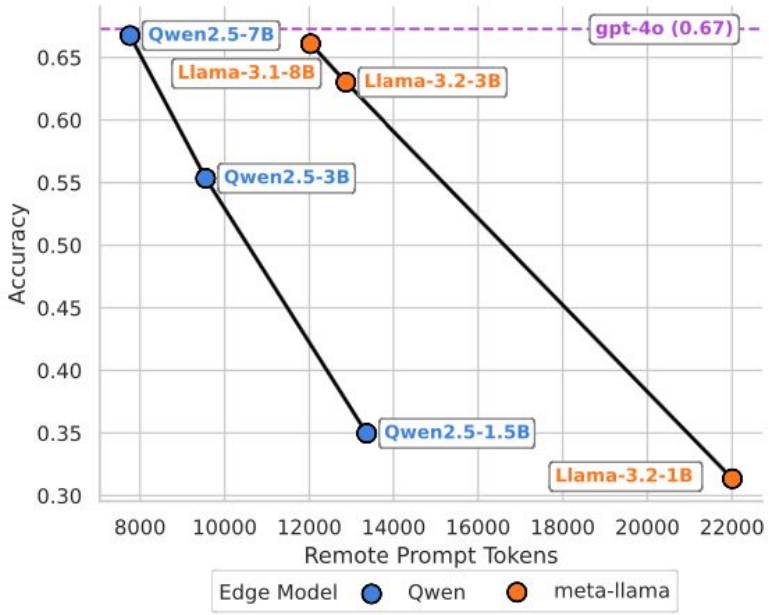
3. **Sequential Communication Rounds:** Do multiple rounds of communication improve final output quality?

.

Scaling Local Model

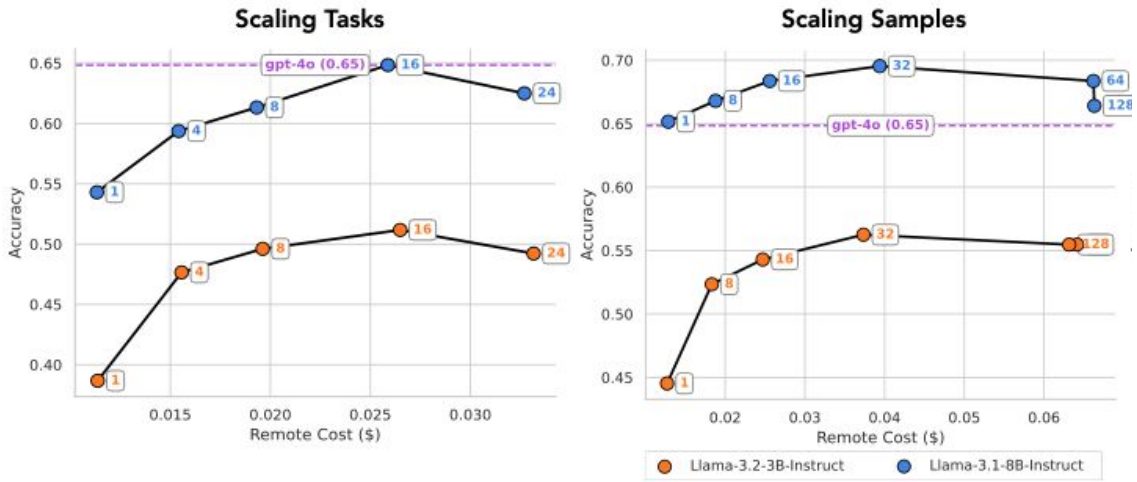


Quality



Efficiency

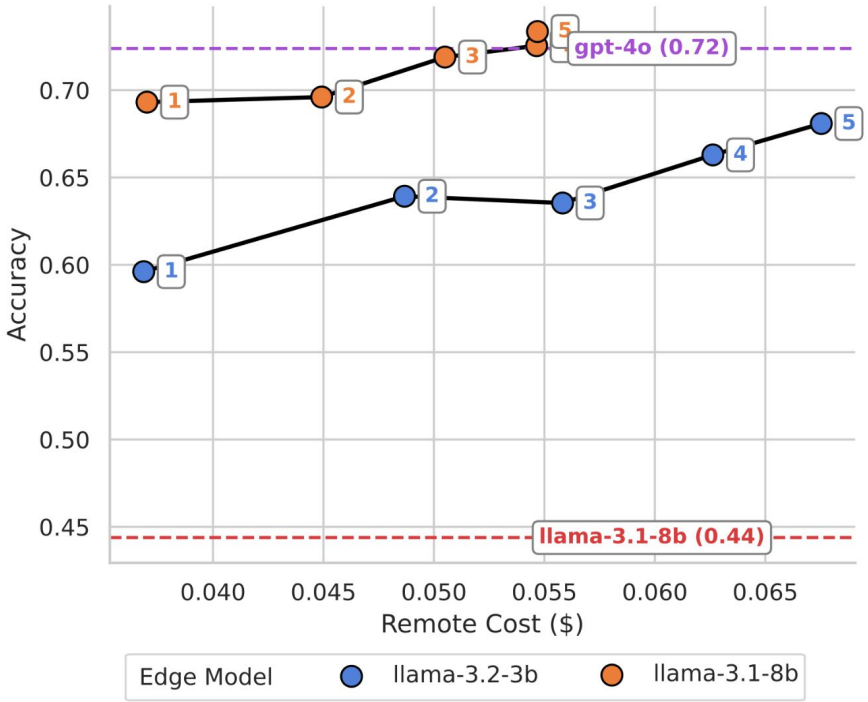
Scaling Parallel Work On-Device



# of instructions / chunk

# of samples per task/chunk

Scaling Sequential Rounds of Communication



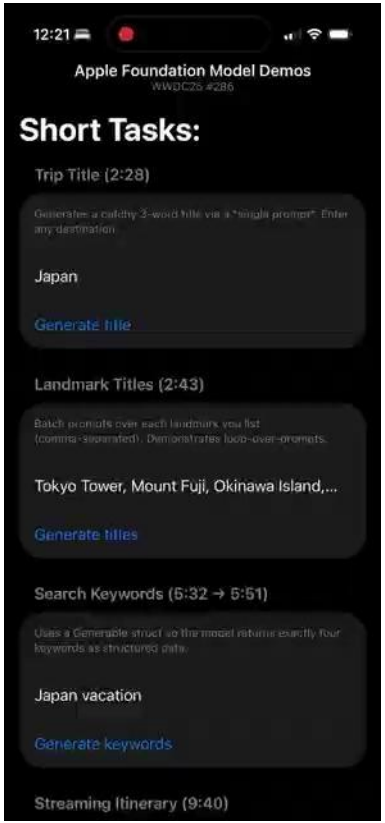
Future Work

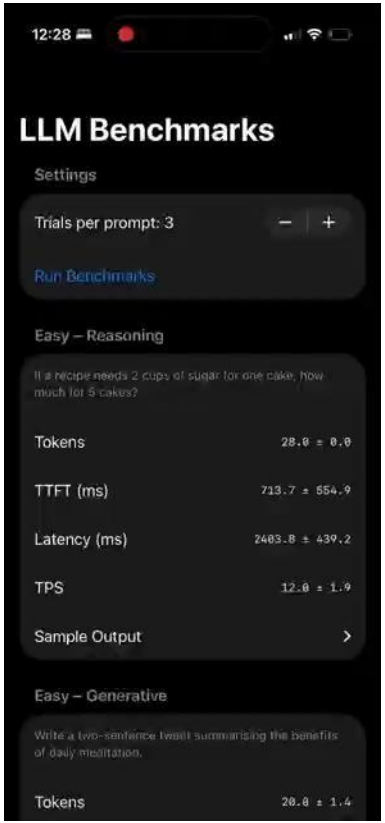
➜ Benchmarks for edge inference (*in progress, we invite collaborators!): sweep over devices, tasks and models!*

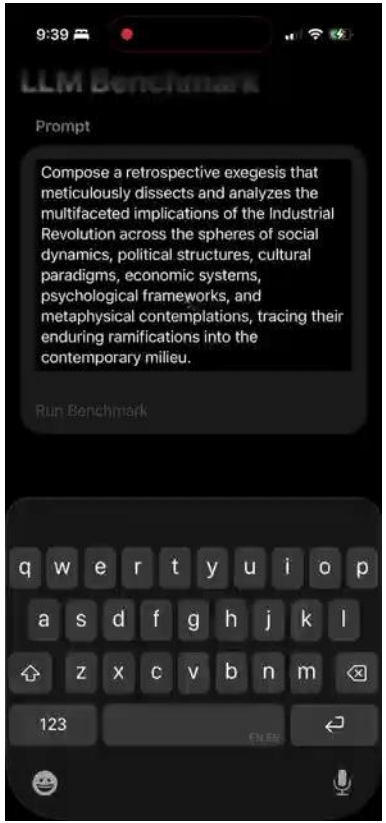
➜ Better high-throughput inference on-device (*in progress, we invite collaborators!)*

➜ Training for collaboration

Benchmarking performance metrics on other edge hardware!

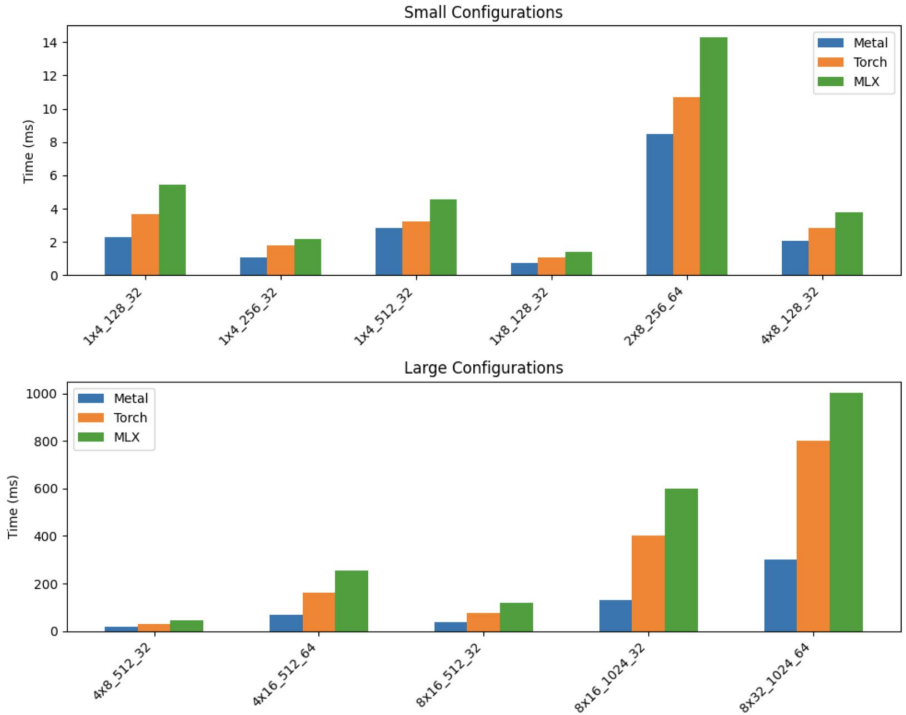






With Harsh Singh

Can we improve local performance by writing specialized kernels?



On Apple silicon, writing paged-attention kernels in Metal (blue) improves upon torch MPS and Apple's MLX

With Akhilesh Balasingam

Thank you!

Code: https://github.com/HazyResearch/minions
Paper (to appear in ICML 2025): https://arxiv.org/abs/2502.15964