Deploying Large Language Models on a Raspberry Pi

Pete Warden

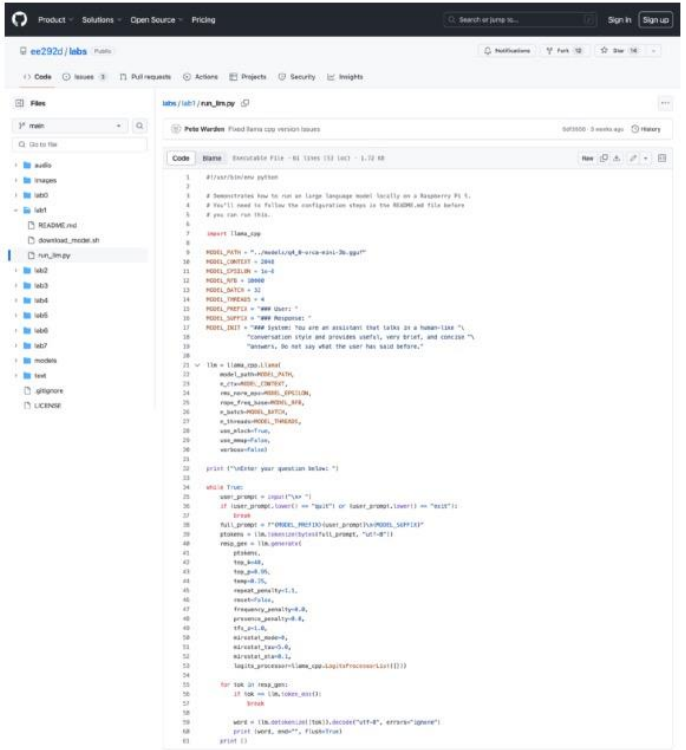
CEO

Useful Sensors

USEFUL SENSORS

Running an LLM on a Raspberry Pi

- github.com/ee292d/labs/blob/main/lab1/run_llm.py
- 60 lines of Python code, including comments.



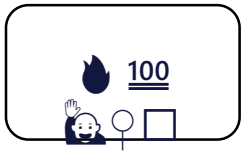
Demo

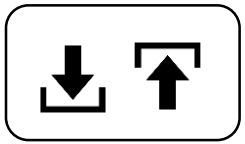
What you need to know

- What's the technology behind this code?
- Where can you get models?
- Which models will run efficiently on what hardware?
- How can you customize models?
- What's coming in the future?

What's the technology here?

- Llama.cpp was one of the first easy to deploy implementations of Meta's open weights Llama v1 LLM.
- It didn't require Python or a lot of dependencies, unlike the Python code originally released by Meta, and so it became popular.
- It was also easy to optimize, and so became faster on many platforms.
- Support started to be added for other models, and a GGML format emerged that allowed export and import.





So it’s like PyTorch or TensorFlow?

- No! Though Llama.cpp’s scope has expanded over time, it’s still limited in which models it can support, and is focused on inference rather than training.
- The first generation of ML frameworks tried to be good at everything (TensorFlow more than most) which makes them hard to port, optimize, modify, and understand.
- We’re seeing different design goals in this generation. PyTorch is the favorite for prototyping and training, but other tools are used for inference, compression, and fine-tuning.

Other frameworks

- Another library I use a lot is CTransformers2. This is similar to GGML, but has more of a focus on quantization and optimization.
- Don’t expect to bring your own model though. A key difference between gen 1 frameworks and these is that they only support a subset of models, and adding new architectures may involve code changes.
- They also often break compatibility with saved files, requiring reconversion when you upgrade to a new library version.

Where can you get models?

Hugging Face

You can find almost any released model in any format somewhere on the site, look in the files section.

reddit

On Reddit, r/LocalLlama is the place to find news and advice on running models, along with some impressive demos.

- Be aware, most models are “open weights”, but few are “open source”. You can use the pretrained models, but the datasets and training code are usually kept proprietary. The Allen Institute’s Olma project is a welcome exception.

Which models run on what HW?

- You need a lot of RAM for LLMs, because transformers use dynamic layers constructed in memory. A good rule of thumb is that you need as much RAM as the model file size. For example a 7-billion parameter model at eight bits will be 7GB on disk, and you can expect to need at least 7GB of RAM to run it at a decent speed.

- The latency is also usually dominated by the RAM speed, so the faster the better.
- TPUs and other accelerators often don't help much, since we're memory bound.

What hardware should you use?

- Running as a regular Android or iOS app is hard because you need to use a lot more memory and compute than most applications, and you'll get throttled or blocked.
- If you have vendor-level access to avoid these limits, Android on a modern SoC is a good option.
- Otherwise a Raspberry Pi 5 is a good option, with 8GB of RAM it can handle medium-sized models. Other quad-core A76 SBCs are similar.
- Microcontrollers and DSPs (meaning low power or low cost) aren't possible right now because of how RAM-hungry these models are.

Quantization

- Since all mainstream LLMs are Transformer-based, and Transformer models are memory bound on batch-size-one inference, the size of the data you pull from memory matters.
- Quantization is an old technique that has become more relevant with models now memory bound. It takes 32-bit floating point representations of weights and shrinks them down to values that take fewer bits per value. Eight bit is standard for convolutional image models, but since bandwidth is so critical and unpacking compute can be hidden in memory latency, four, two, or even one bit schemes are now in use.

How can you customize models?

- Low Rank Adaptation (or LoRA) is a technique that's similar in effect to transfer learning in CNN models. It lets you add extra layers to a pretrained model to customize its outputs, with shorter training times and less data than a full training run.
- Here's an example you can run in a Colab notebook in under an hour:
  - http://github.com/ee292d/labs/blob/main/lab6/notebook.ipynb

LoRA Training Demo

Retrieval Augmented Generation

- The idea is to use conventional search techniques to retrieve factual information to insert in the prompt as context, so the user question will draw on that knowledge.
- For example, you could notice a question contains the name of a product, and insert the product description as the context. The result should then be able to use that extra information to give a better answer.
- I hate it!

Why I hate RAG

- It’s a neat technique, but it’s usually overkill for most practical situations. The “generation” part means you’re still going to have some situations where the model makes up answers.

- In most cases you can just do a good job on the “retrieval” and show those answers directly to the user. They’re vetted, relevant, and easy to control. RAG is for when you need to scale a solution, which isn’t relevant for most applications I encounter.

What's coming next?

- Models keep getting smaller and more accurate. Microsoft's latest Phi 3 is a great example of the trend.
- Transformers are memory hungry and hard to accelerate. There are lots of alternatives like Mamba and Conformers that offer different tradeoffs, maybe something new will emerge that's better for the edge.
- Shrinking scope will help us use even smaller models too, especially as I expect retrieval will be more important than generation long term.

Conclusions

- LLMs want to be on the edge!
- Dip your toes in the water with some simple code experiments, and prototype solutions that make sense to you.
- These models are only going to get faster and more capable, and hardware will emerge to help with that.

Resources

- These slides: usfl.ink/ev_talk
- EE292D Labs: github.com/ee292d
- Intro to GGML: omkar.xyz/intro-ggml
- Huggingface: huggingface.co

Useful Sensors

- We run the latest AI models on edge hardware to solve problems like person detection, language translation, voice interfaces, LLM querying, and more!
- Come see us at our booth (#806)

Thank you