Theis on:

HOW TO MAKE AI-MODELS DO STUFF!

Leveraging Rust and Candle to Execute LLM tasks on IoT

OUR PLAN

- SECTION ½ - AI AND RUST
  - Why use Rust for AI? (LLMs)
  - …Is Rust just better? Not always…
  - What are LLMs?...
  - How Candle supports LLMs
  - Demo ½
- SECTION 2/2 – AI DOING THINGS USING RUST
  - Building the AI ecosystem in Rust
  - Task execution and making AI do things
  - The brain: Agent
  - The state: Handling events
  - The state: Routing events
  - The state: Actions
  - Workflow
  - How can we do… Whatever?
  - Demo 2/2
- Further recommendations

SECTION 1/2

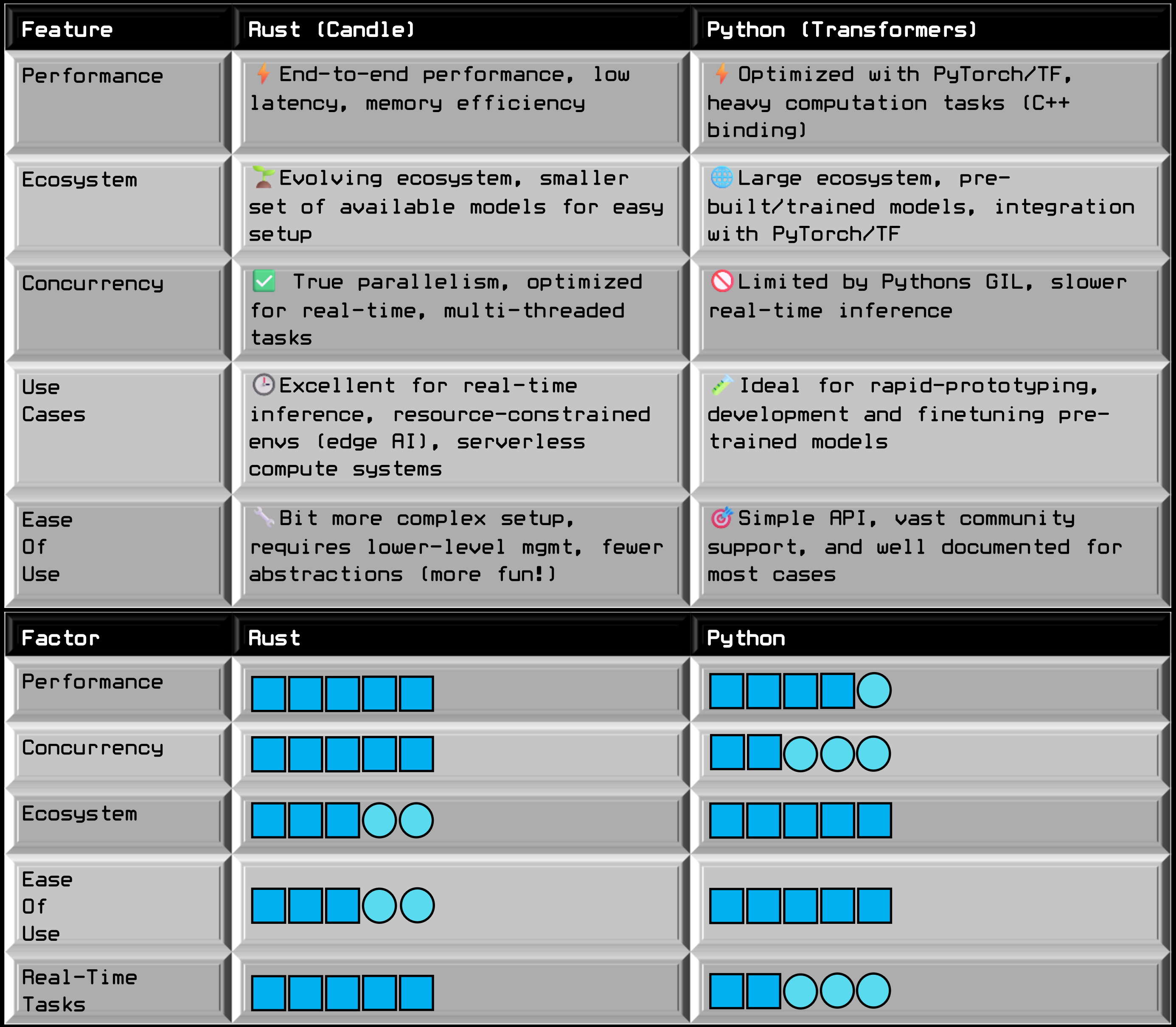
AI and Rust

WHY USE RUST FOR AI? (LLM)

Category	Rust	Python
Execution Speed	Compiled, faster inference, ideal for real-time AI tasks	Interpreted, slower inference due to GIL and overhead
Memory Safety	Precise memory mgmt., preventing leaks mem leak in large models. Safer services	GC risks performance drops and leaks
Concurrency	True parallelism, ideal for high-demand AI services, web frameworks and more	GIL limits concurrency, slowing real-time tasks
Memory Usage	Low overhead, efficient use of memory for large models. Great for IoT and mobile devices!	Higher mem usage due to dynamic typing and GC
Model Loading	Optimized for efficient loading of large AI models	Slower model loading due to abstractions
Latency	Low latency, ideal for real-time AI applications	Higher latency due to GIL and overhead

…IS RUST JUST BETTER? NOT ALWAYS…

… But where it is, it really shines



Feature	Rust (Candle)	Python (Transformers)
Performance	⚡End-to-end performance, low latency, memory efficiency	⚡Optimized with PyTorch/TF, heavy computation tasks (C++ binding)
Ecosystem	🌱Evolving ecosystem, smaller set of available models for easy setup	🌐Large ecosystem, pre-built/trained models, integration with PyTorch/TF
Concurrency	✅ True parallelism, optimized for real-time, multi-threaded tasks	🚫Limited by Pythons GIL, slower real-time inference
Use Cases	🕒Excellent for real-time inference, resource-constrained envs (edge AI), serverless compute systems	🧪Ideal for rapid-prototyping, development and finetuning pre-trained models
Ease Of Use	🔧Bit more complex setup, requires lower-level mgmt, fewer abstractions (more fun!)	🎯Simple API, vast community support, and well documented for most cases

Factor	Rust	Python
Performance	■■■■■	■■■■○
Concurrency	■■■■■	■■○○○
Ecosystem	■■■○○	■■■■■
Ease Of Use	■■■○○	■■■■■
Real-Time Tasks	■■■■■	■■○○○

WHAT ARE LLMS?...

…And why do they matter?

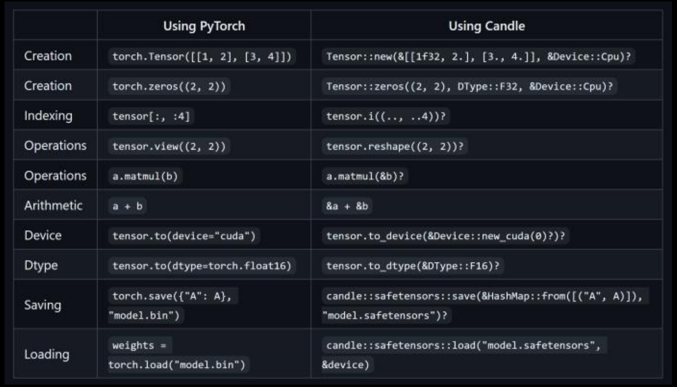
- LLMs (like the famous GPT-series and BERT) are built using transformer architectures that process and understand context in vast amounts of text data (See video)
- LLMs are today trained on billions of parameters, allowing them to perform tasks, such as:
  - text generation
  - summarization
  - machine translation
  - question-answering
  - code generation
- Why is this relevant?
  - They are great at context handling, and has become crucial in apps like chatbots, but also more autonomous AI-driven applications, and automated code generation
- They are continuing to grow rapidly, both in terms of performance, weights and knowledge

HOW CANDLE SUPPORTS LLMS

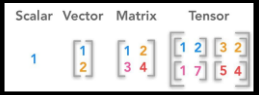
… And how to use it

- Candle efficiently handles large arrays of data, or tensors, which are fundamental in deep learning models, including LLMs
- Built-in support for model inference, Candle simplifies running LLMs like GPT in Rust, eliminating the need for Python
- Candle leverages Rusts performance and concurrency to make AI more accessible for real-time apps on edge devices and servers

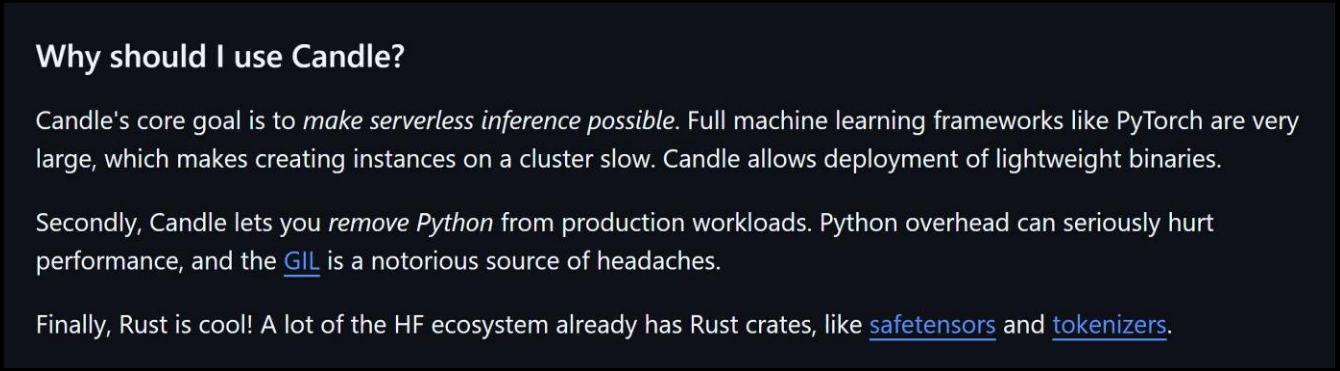
… Efficient Tensor Management



	Using PyTorch	Using Candle
Creation	`torch.Tensor([[1, 2], [3, 4]])`	`Tensor::new(&[[1f32, 2.], [3., 4.]], &Device::Cpu)?`
Creation	`torch.zeros((2, 2))`	`Tensor::zeros((2, 2), DType::F32, &Device::Cpu)?`
Indexing	`tensor[:, :4]`	`tensor.i((.., ..4))?`
Operations	`tensor.view((2, 2))`	`tensor.reshape((2, 2))?`
Operations	`a.matmul(b)`	`a.matmul(&b)?`
Arithmetic	`a + b`	`&a + &b`
Device	`tensor.to(device="cuda")`	`tensor.to_device(&Device::new_cuda(0)?)?`
Dtype	`tensor.to(dtype=torch.float16)`	`tensor.to_dtype(&DType::F16)?`
Saving	`torch.save({"A": A}, "model.bin")`	`candle::safetensors::save(&HashMap::from([("A", A)]), "model.safetensors")?`
Loading	`weights = torch.load("model.bin")`	`candle::safetensors::load("model.safetensors", &device)`



… What do Candle say?



Why should I use Candle?

Candle's core goal is to *make serverless inference possible*. Full machine learning frameworks like PyTorch are very large, which makes creating instances on a cluster slow. Candle allows deployment of lightweight binaries.

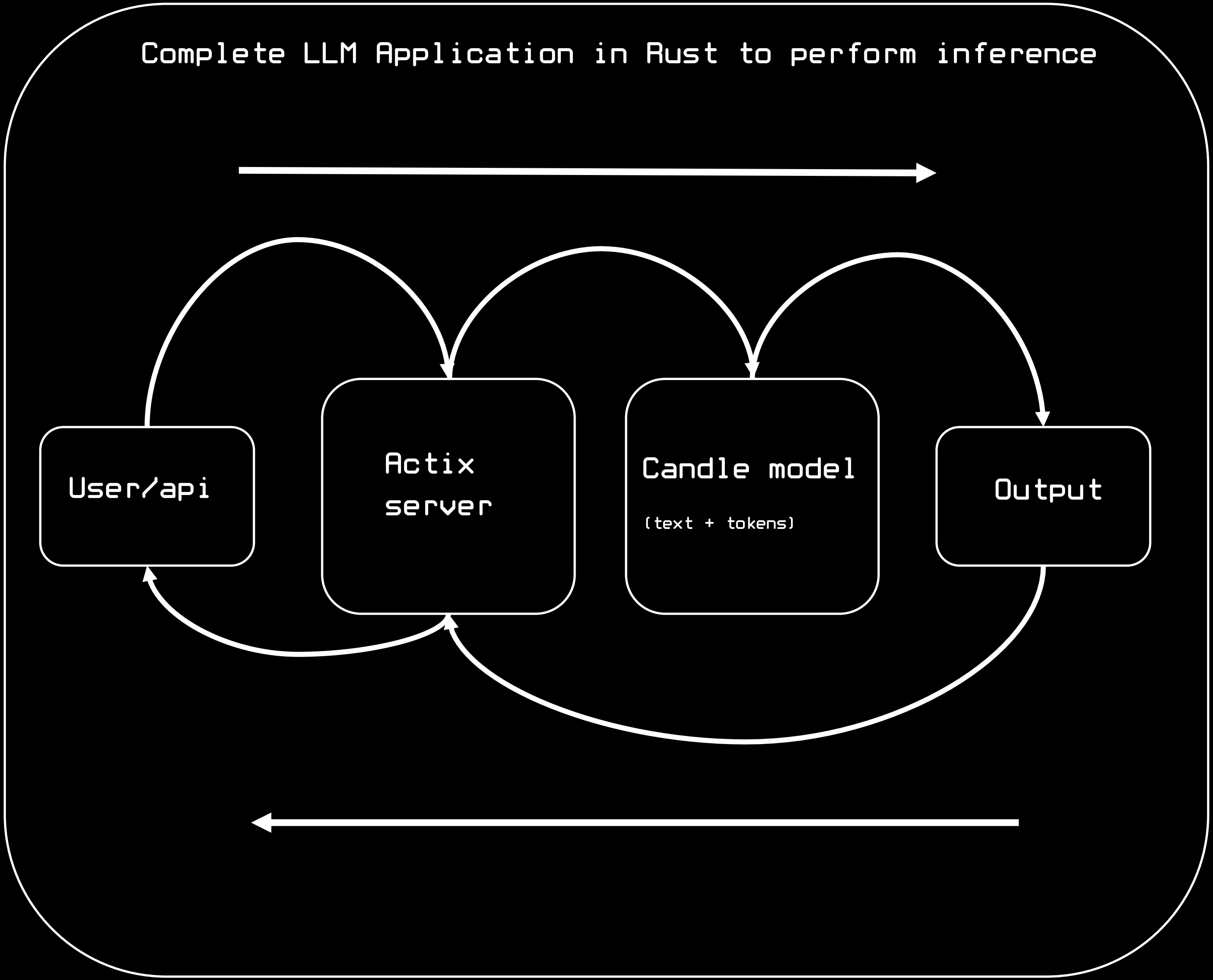
Secondly, Candle lets you *remove Python* from production workloads. Python overhead can seriously hurt performance, and the GIL is a notorious source of headaches.

Finally, Rust is cool! A lot of the HF ecosystem already has Rust crates, like safetensors and tokenizers.

FINALLY, ACTION!

… Lets see how we can

build something interesting



CONTACT

Theis on Discord for this content

DEMO
1/2

...

SECTION 2/2

Ai doing things

using Rust

CONTACT

Theis on Discord for this content

HOW CAN WE DO… …WHATEVER?

… Lets look at some

cool things we could do

DEMO 2/2

...

FURTHER RECOMMENDATIONS

- Video of tensors: (https://www.youtube.com/watch?v=f5liqUk0ZTw)
- Video of Transformers: (https://www.youtube.com/watch?v=wjZofJX0v4M&t=542s)
- AutoGPT project: (https://github.com/Significant-Gravitas/AutoGPT)
- Candle project: (https://github.com/huggingface/candle)