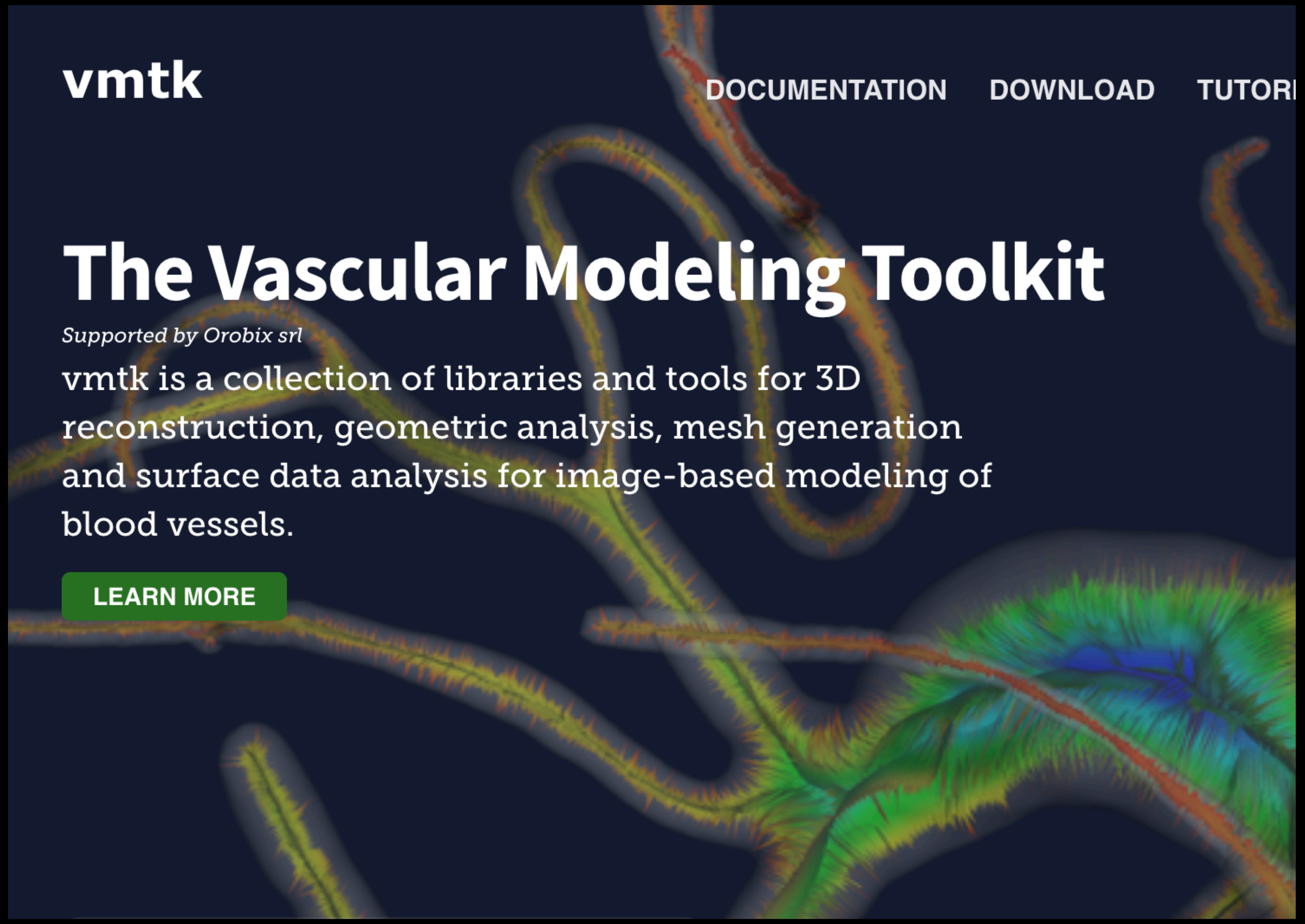
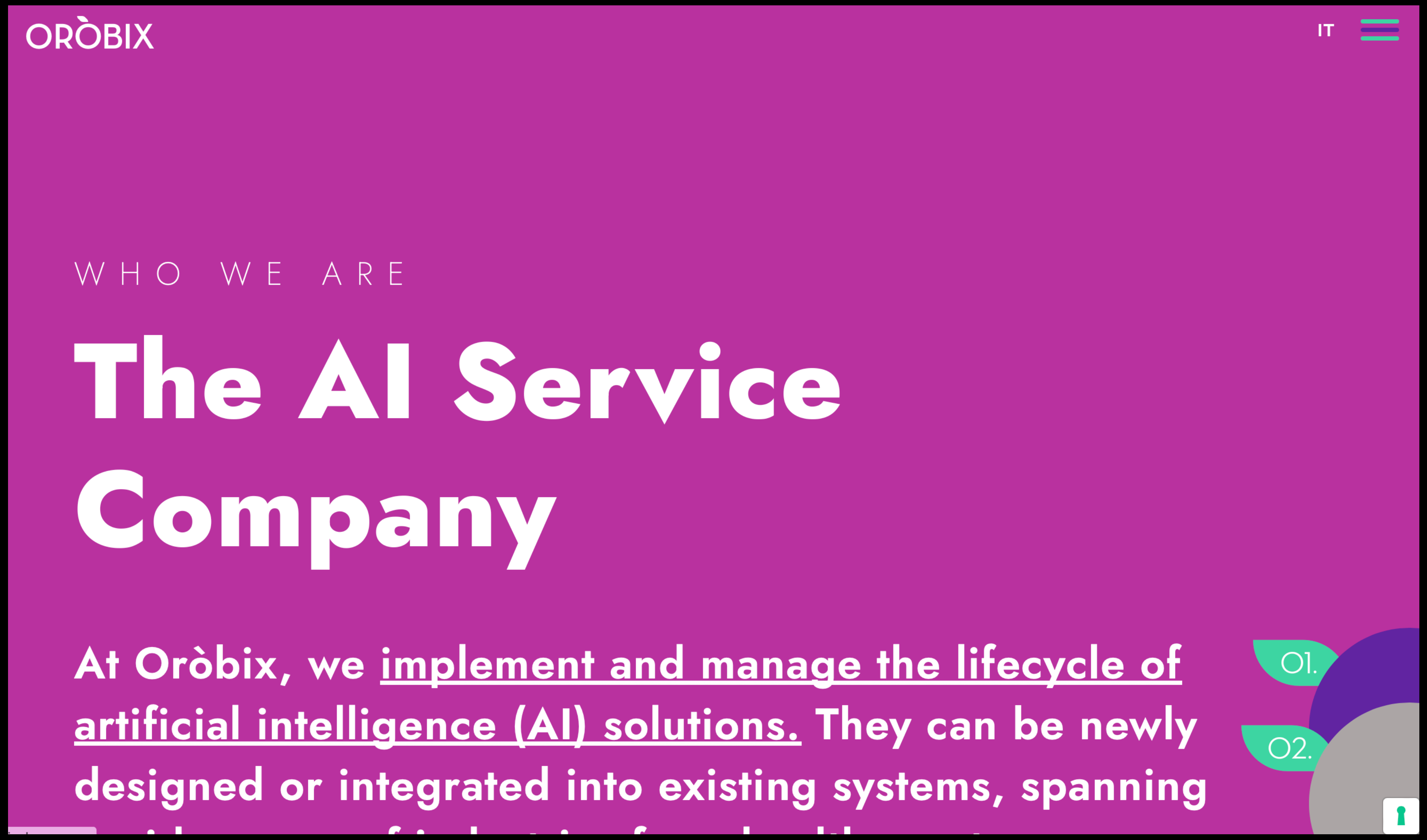
A journey into PyTorch, the Ecosystem, and Deep Learning Compilers

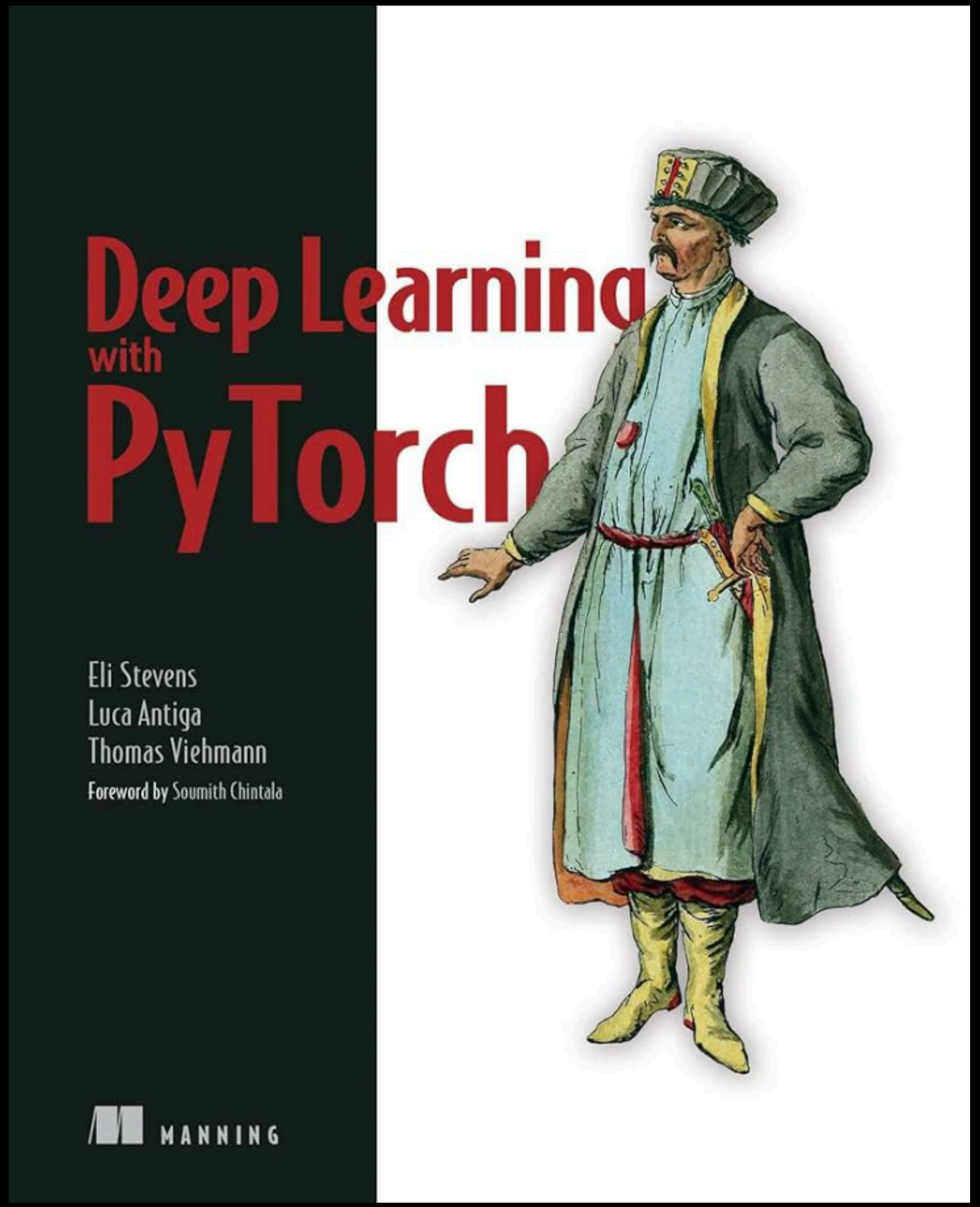
Luca Antiga

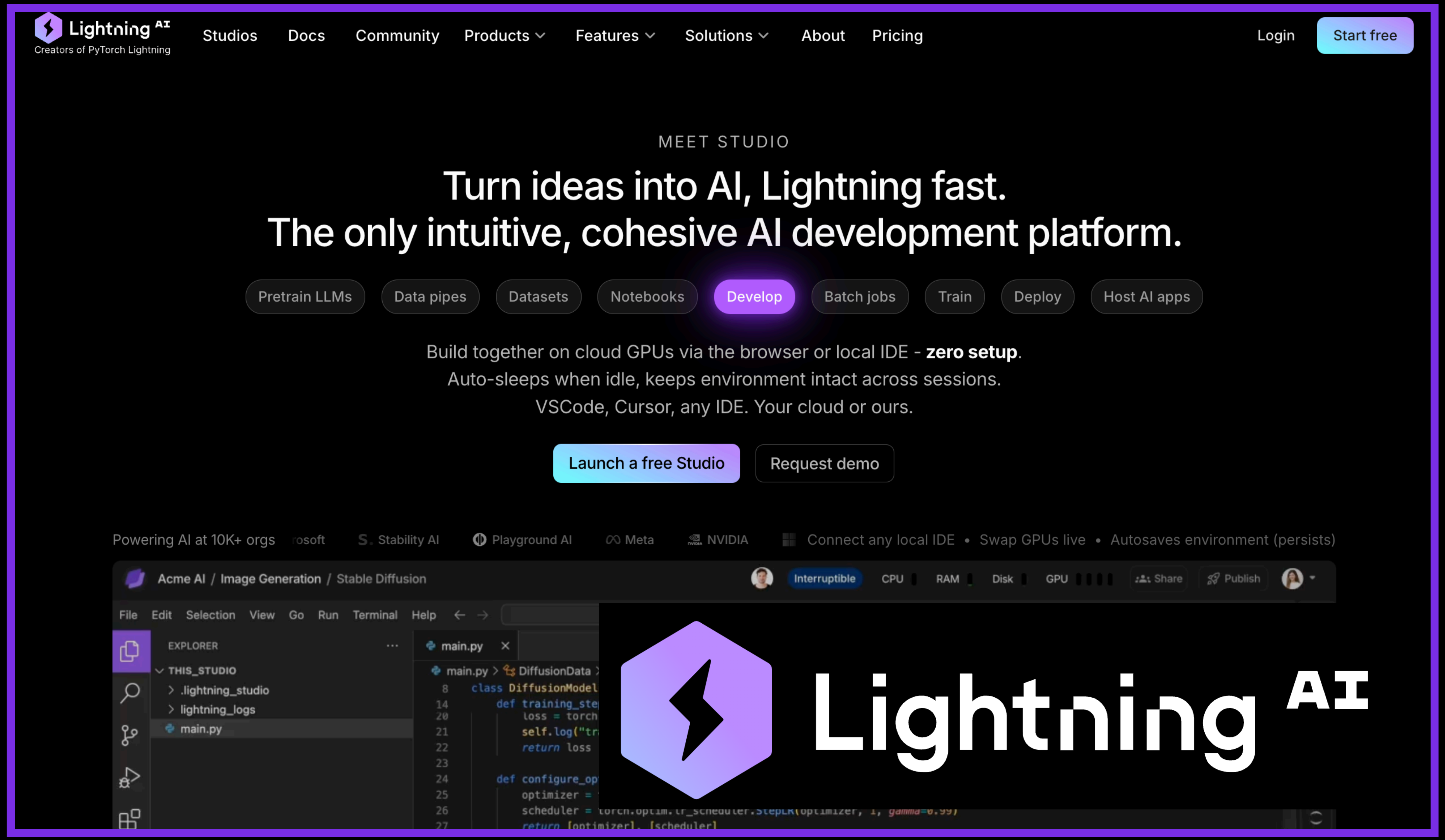
La Sapienza, Nov 11 2024

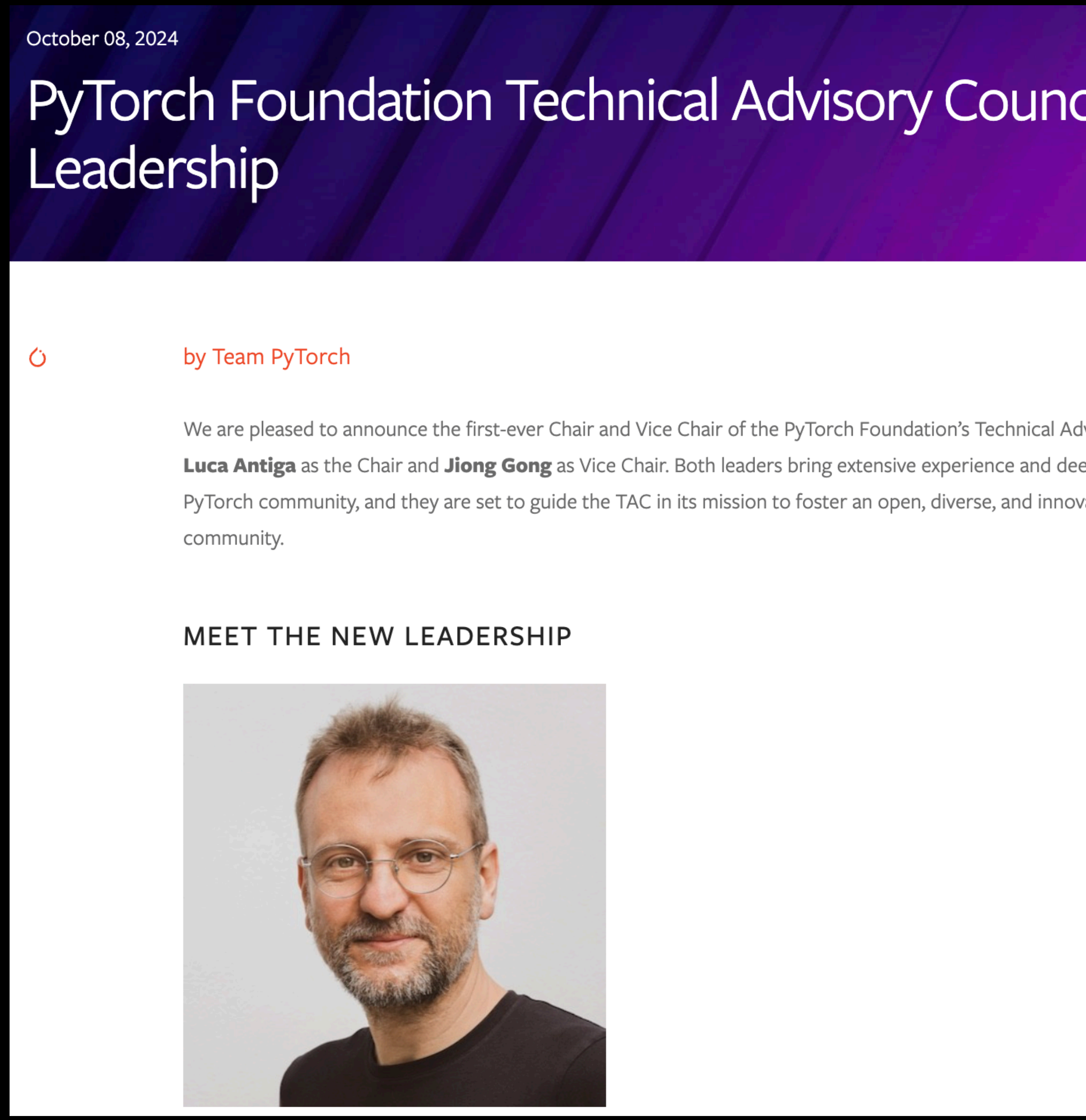

vmtk
DOCUMENTATION DOWNLOAD TUTORI
The Vascular Modeling Toolkit
Supported by Orobix srl
vmtk is a collection of libraries and tools for 3D reconstruction, geometric analysis, mesh generation and surface data analysis for image-based modeling of blood vessels.
LEARN MORE


ORÒBIX
IT
WHO WE ARE
The AI Service Company
At Oròbix, we implement and manage the lifecycle of artificial intelligence (AI) solutions. They can be newly designed or integrated into existing systems, spanning
01.
02.


PyTorch: An Imperative Style, High-Performance Deep Learning Library
Adam Paszke
University of Warsaw
adam.paszke@gmail.com
Sam Gross
Facebook AI Research
sgross@fb.com
Francisco Massa
Facebook AI Research
fmassa@fb.com
Adam Lerer
Facebook AI Research
alerer@fb.com
James Bradbury
Google
jekbradbury@gmail.com
Gregory Chanan
Facebook AI Research
gchanan@fb.com
Trevor Killeen
Self Employed
killeent@cs.washington.edu
Zeming Lin
Facebook AI Research
zlin@fb.com
Natalia Gimelshein
NVIDIA
ngimelshein@nvidia.com
Luca Antiga
Orobix
luca.antiga@orobix.com
Alban Desmaison
Oxford University
alban@robots.ox.ac.uk
Andreas Köpf
Xamla
andreas.koepf@xamla.com
Edward Yang
Facebook AI Research
ezyang@fb.com
Zach DeVito
Facebook AI Research
zdevito@cs.stanford.edu
Martin Raison
Nabla
martinraison@gmail.com
Alykhan Tejani
Twitter
atejani@twitter.com
Sasank Chilamkurthy
Qure.ai
sasankchilamkurthy@gmail.com
Benoit Steiner
Facebook AI Research
benoitsteiner@fb.com
Lu Fang
Facebook
lufang@fb.com
Junjie Bai
Facebook
jbai@fb.com
Soumith Chintala
Facebook AI Research
soumith@gmail.com
:1912.01703v1 [cs.LG] 3 Dec 2019


Deep Learning with PyTorch
Eli Stevens
Luca Antiga
Thomas Viehmann
Foreword by Soumith Chintala
MANNING


Lightning AI
Creators of PyTorch Lightning
Studios Docs Community Products Features Solutions About Pricing
Login Start free
MEET STUDIO
Turn ideas into AI, Lightning fast.
The only intuitive, cohesive AI development platform.
Pretrain LLMs Data pipes Datasets Notebooks Develop Batch jobs Train Deploy Host AI apps
Build together on cloud GPUs via the browser or local IDE - zero setup.
Auto-sleeps when idle, keeps environment intact across sessions.
VSCode, Cursor, any IDE. Your cloud or ours.
Launch a free Studio Request demo
Powering AI at 10K+ orgs
Connect any local IDE • Swap GPUs live • Autosaves environment (persists)
Lightning AI


October 08, 2024
PyTorch Foundation Technical Advisory Counc
Leadership
by Team PyTorch
We are pleased to announce the first-ever Chair and Vice Chair of the PyTorch Foundation's Technical Ad
Luca Antiga as the Chair and Jiong Gong as Vice Chair. Both leaders bring extensive experience and dee
PyTorch community, and they are set to guide the TAC in its mission to foster an open, diverse, and innov
community.
MEET THE NEW LEADERSHIP

The Lightning OSS stack

Data processing

LitData

Transform datasets at scale. Optimize datasets for fast AI model training.

Training

PyTorch Lightning

FLAGSHIP

Finetune and pretrain AI models on GPUs, TPUs and more. Focus on science, not engineering.

Lightning Fabric

Scale foundation models to 1000s of GPUs with expert-level control.

TorchMetrics

90+ easy-to-use PyTorch metrics optimized for distributed training.

Inference

LitServe

Easily serve AI models Lightning fast. High-throughput serving engine for AI models.

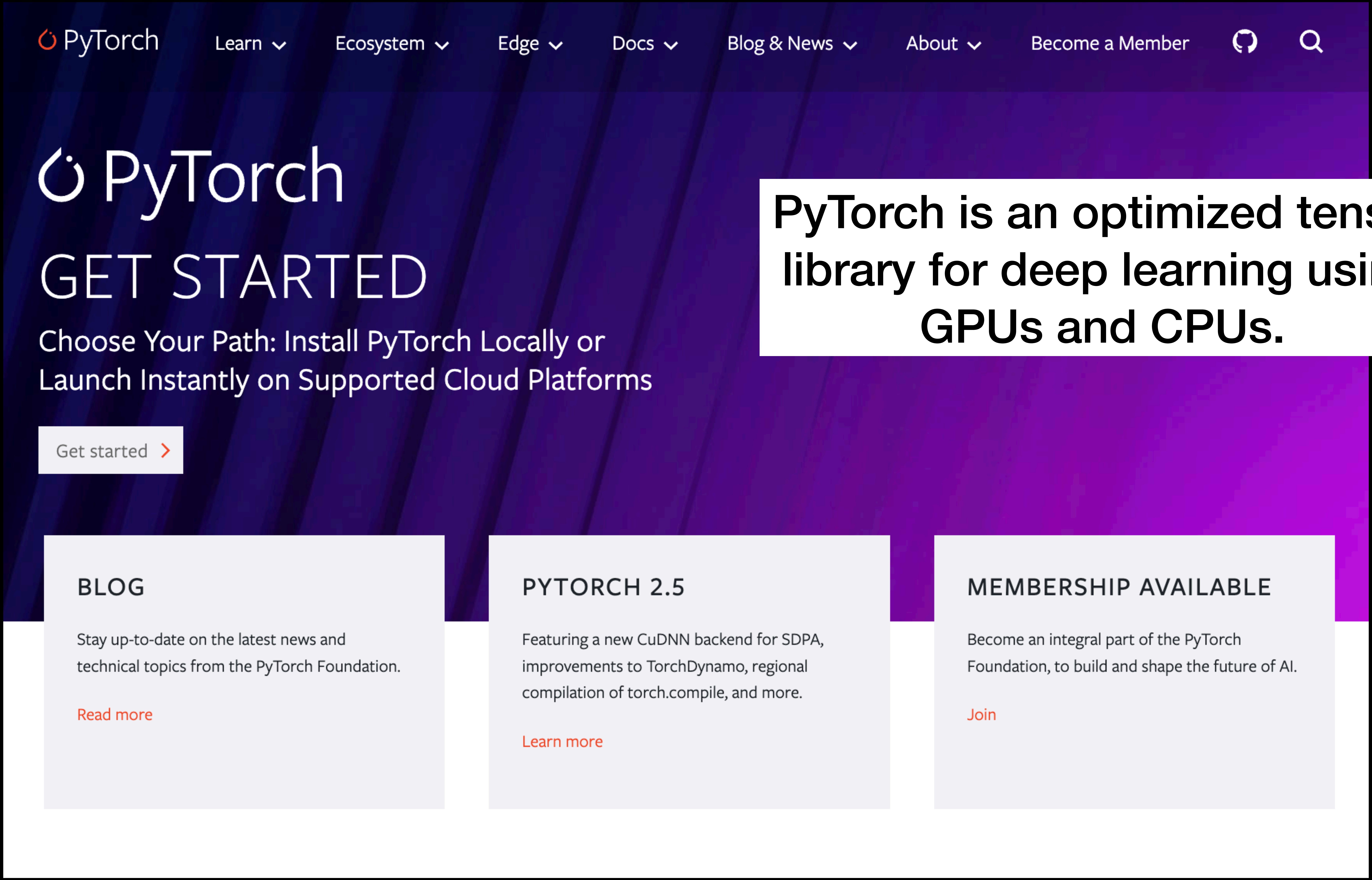
Optimization

Lightning Thunder

Make PyTorch models up to 40% faster! Thunder is a source to source compiler for PyTorch.

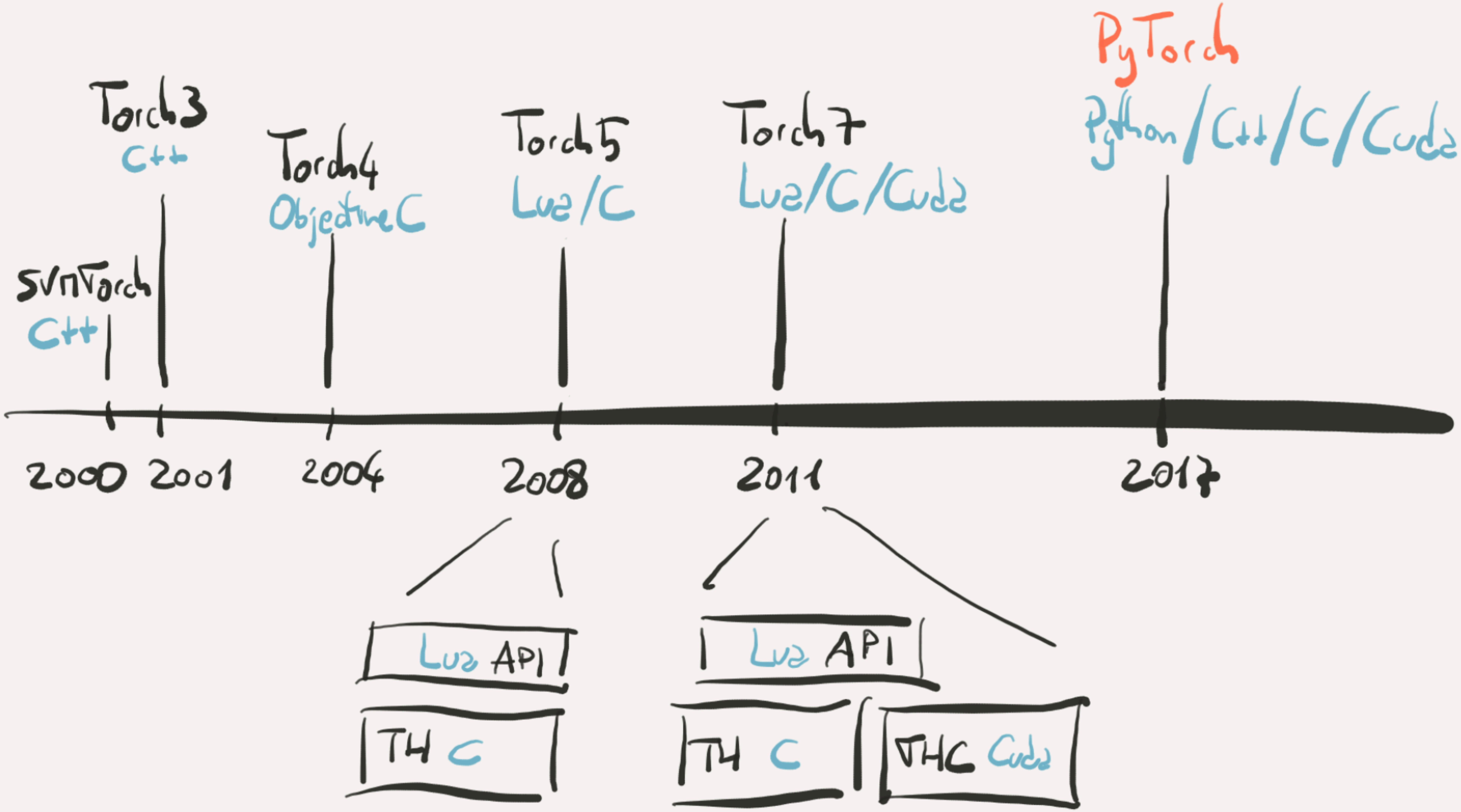
LitGPT

20+ high-performance LLMs with recipes to pretrain, finetune and deploy at scale.



PyTorch is an optimized tensor library for deep learning using GPUs and CPUs.

TORCH TIMELINE



PyTorch

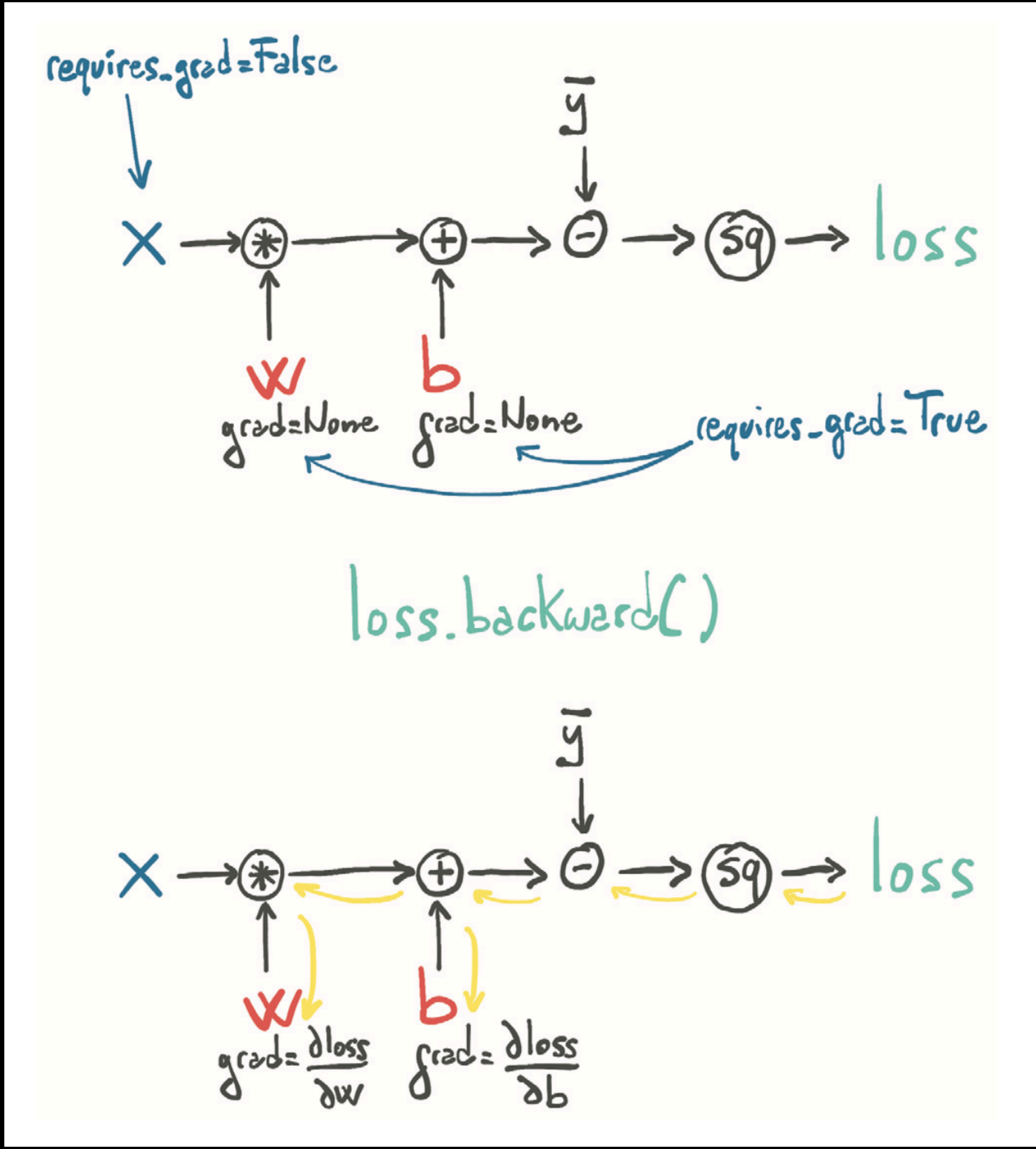
tensors on CPU, GPU

autograd

eager, *define-by-run*

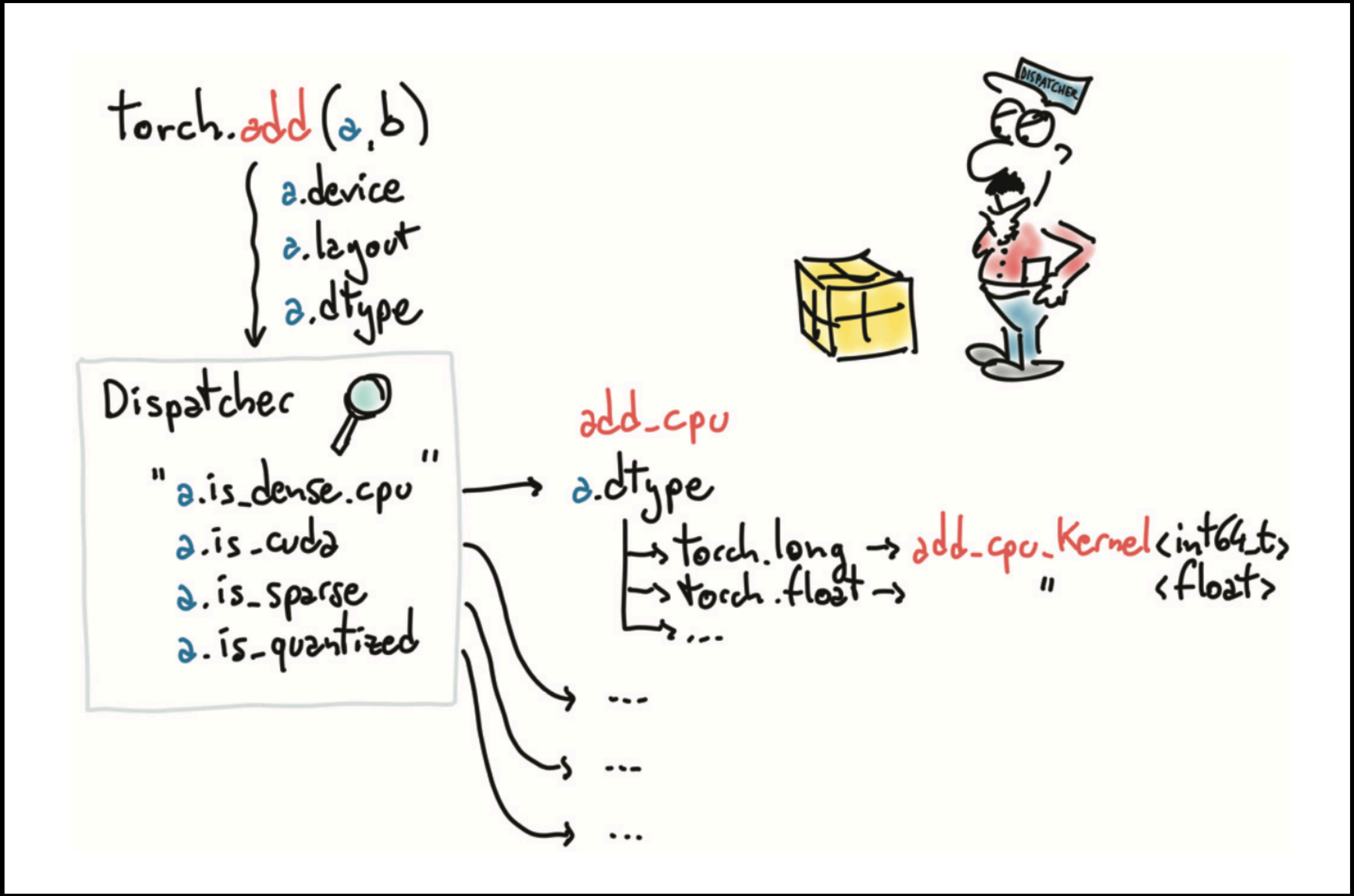
distributed

Autograd



requires_grad=False
$\bar{y}$
X
sq
loss
w
grad=None
b
grad=None
requires_grad=True
loss.backward()
$\bar{y}$
X
sq
loss
w
grad= $\frac{\partial loss}{\partial w}$
b
grad= $\frac{\partial loss}{\partial b}$

Dispatcher



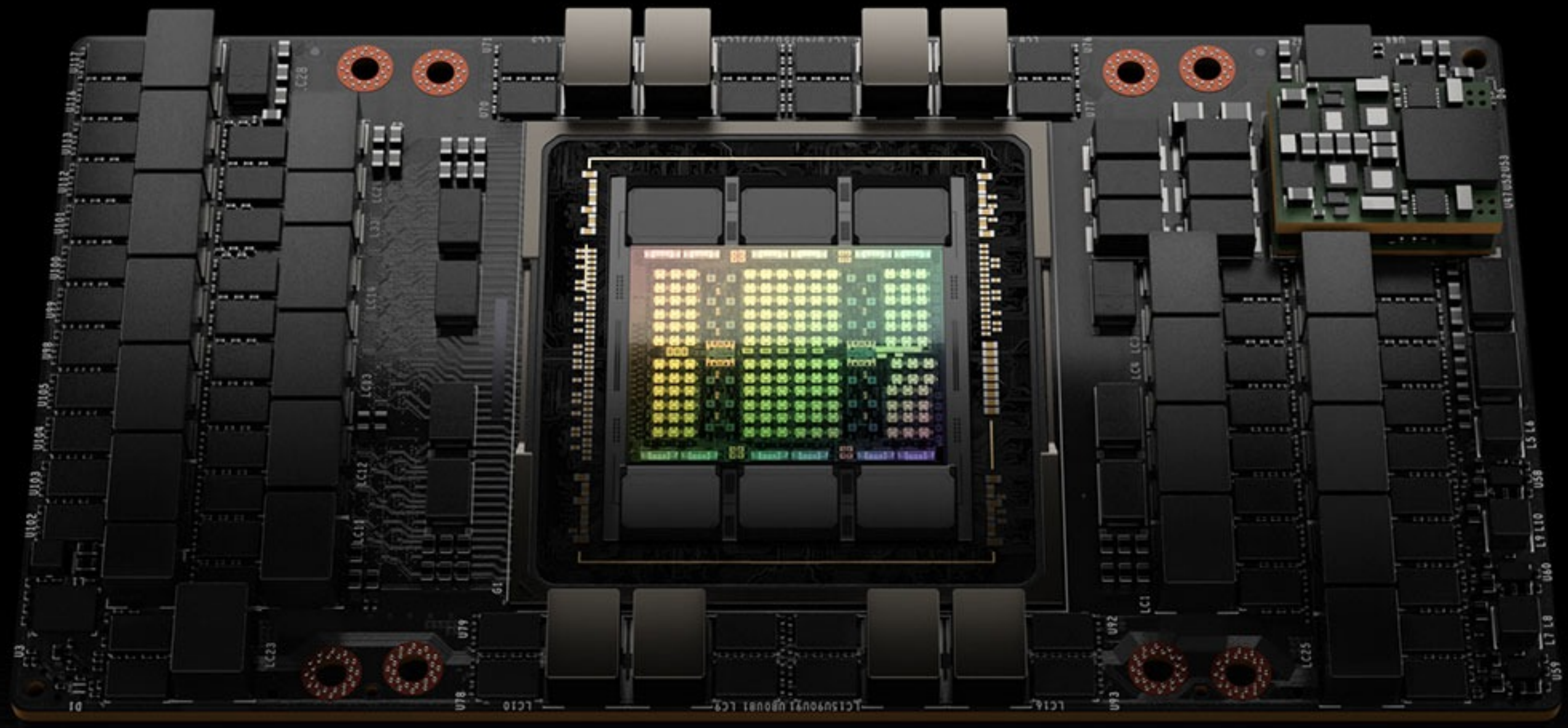
torch.add(a, b)
a.device
a.layout
a.dtype
Dispatcher
"a.is_dense.cpu"
a.is_cuda
a.is_sparse
a.is_quantized
add_cpu
a.dtype
torch.long → add_cpu_kernel<int64_t>
torch.float → " <float>
...
...
...
...
DISPATCHER

Modern-day challenges

memory

memory bandwidth

parallelism



Compilers

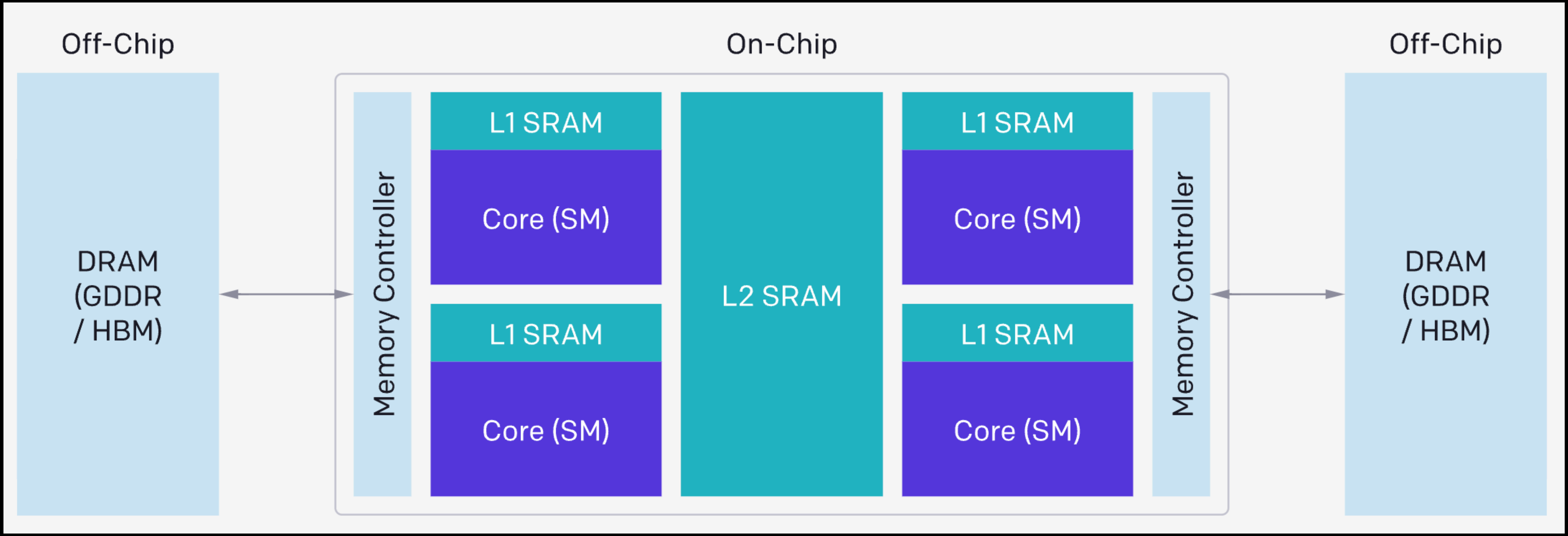
torch.compile (TorchDynamo + TorchInductor)

JAX + XLA, PyTorchXLA

Lightning Thunder

OpenAI Triton

Open-source GPU programming for neural networks



Off-Chip
On-Chip
Off-Chip
DRAM (GDDR / HBM)
Memory Controller
L1 SRAM
Core (SM)
L1 SRAM
Core (SM)
L2 SRAM
L1 SRAM
Core (SM)
L1 SRAM
Core (SM)
Memory Controller
DRAM (GDDR / HBM)

Python

```python
@jit
def add(X, Y, Z, N):
   pid = program_id(0)
   idx= pid * 512 + arange(512)
```

AST Visitor

Triton-IR

```
def void add(i32* X .aligned(16) , i32* Y .aligned(1
{
entry:
  %0 = get_program_id[0] i32;
  %1 = mul i32 %0, 512;
  %3 = make_range[0 : 512] i32<512>;
  %4 = splat i32<512> %1;
  %6 = add i32<512> %4, %3;
  %9 = splat i32<512> N;
  %11 = icmp_slt i1<512> %6, %9;
  %14 = splat i32*<512> X;
  %16 = getelementptr i32*<512> %14, %6;
  %19 = broadcast i1<512> %11;
```

Triton Compiler

LLVM-IR

libLLVM

PTX

```
.visible .entry add(
    .param .u64 add_param_0, .param .u64 add_param_1
    .param .u64 add_param_2, .param .u32 add_param_3
)
.maxntid 128, 1, 1
{
    .reg .pred     %p<4>;
    .reg .b32     %r<18>;
    .reg .b64     %rd<8>;
    ld.param.u64     %rd4, [add_param_0];
    ld.param.u64     %rd5, [add_param_1];
    mov.u32     %r13, %tid.x;
    ld.param.u32     %r14, [add_param_3];
```

Lightning Thunder

Optimizing training or inference requires modifying computations

Model-specific

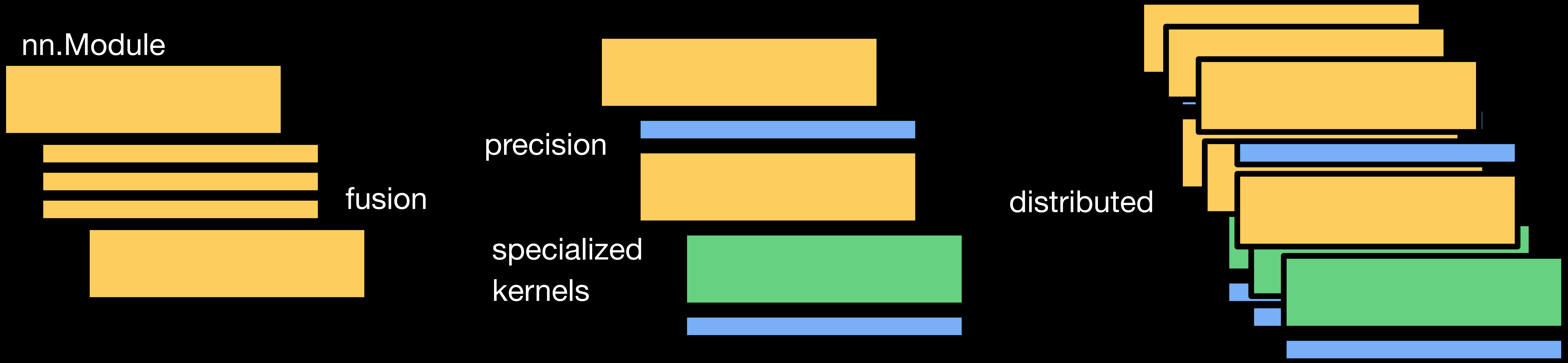
Implementation-specific

Hardware-specific

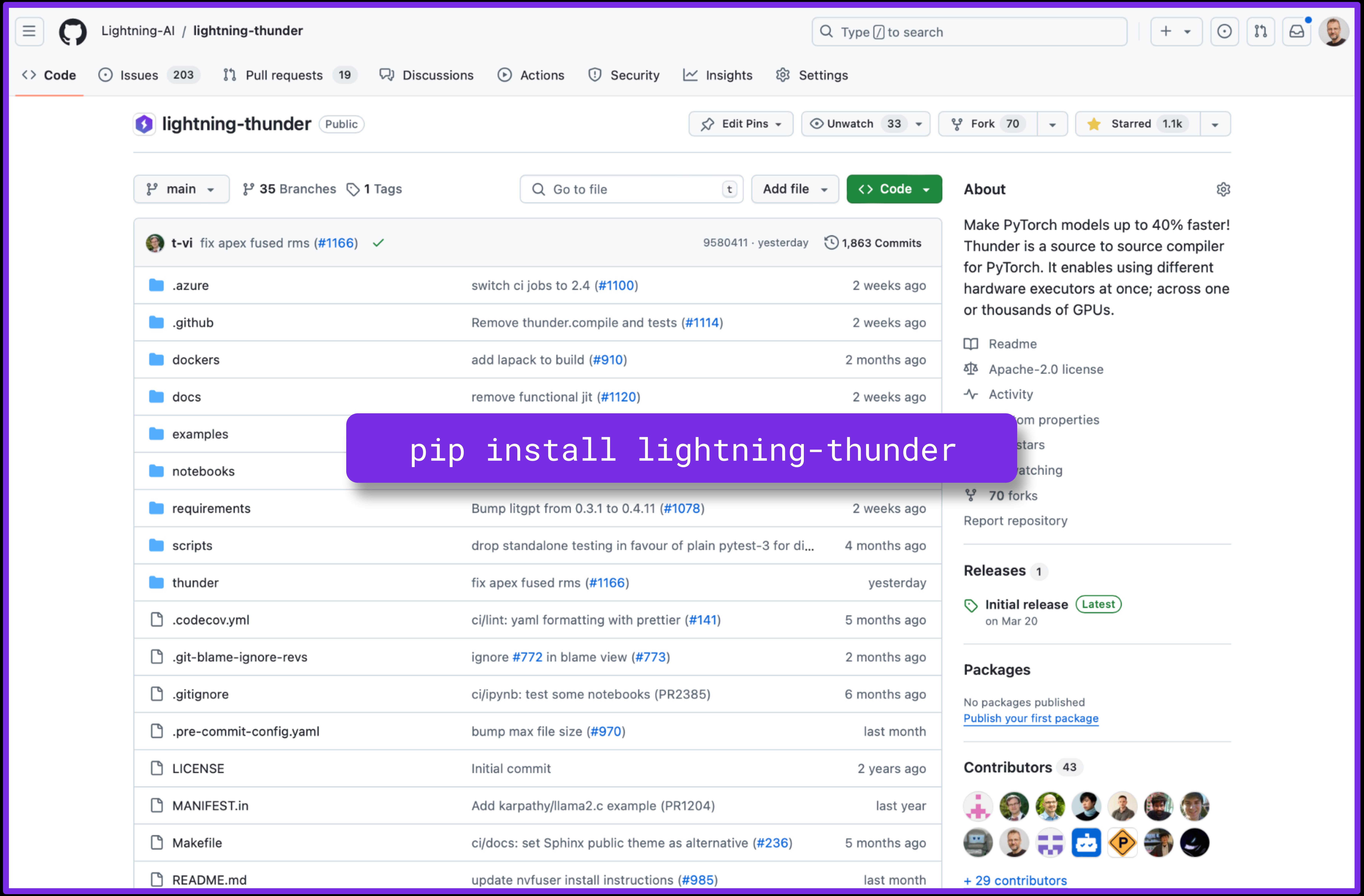
Topology-specific

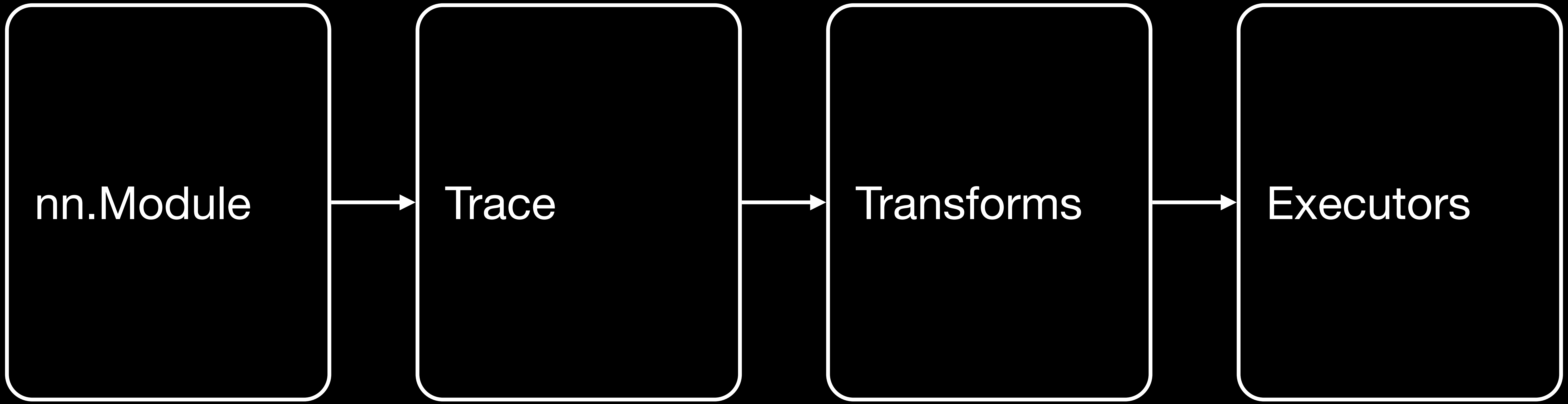
Often performance comes from control
(fusions, memory allocations, offloading, comms overlap)

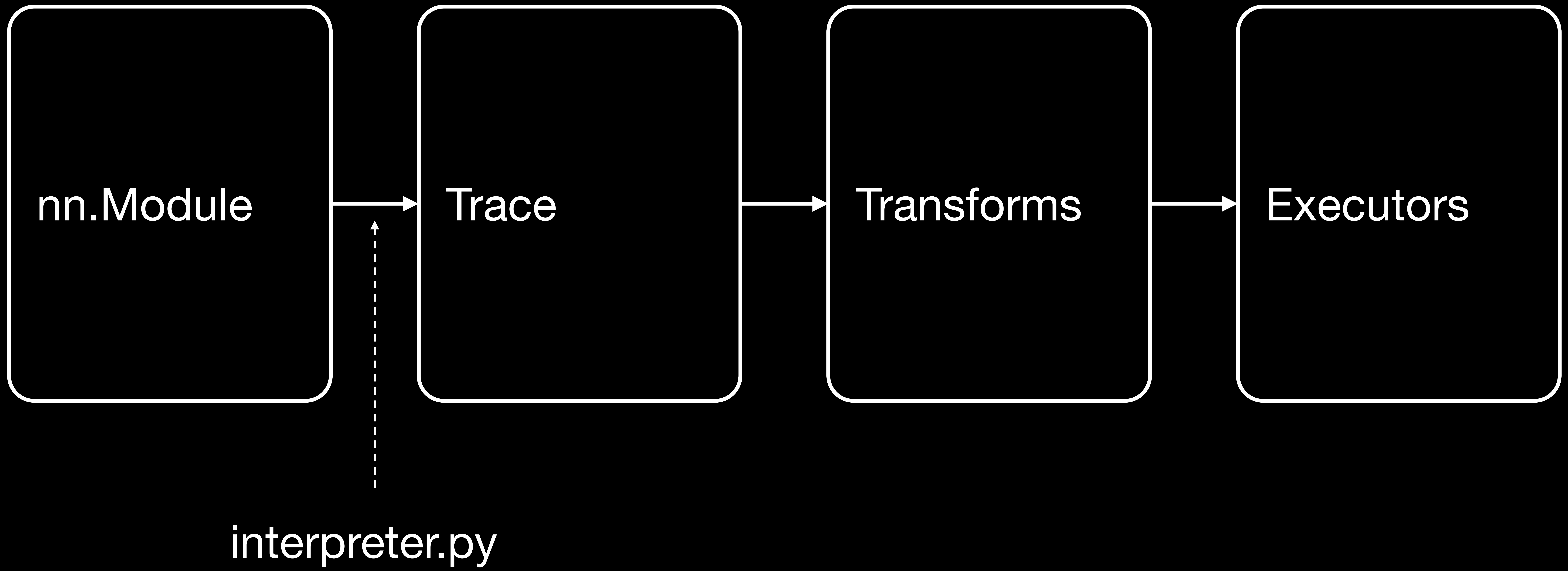
for a specific model on specific hardware

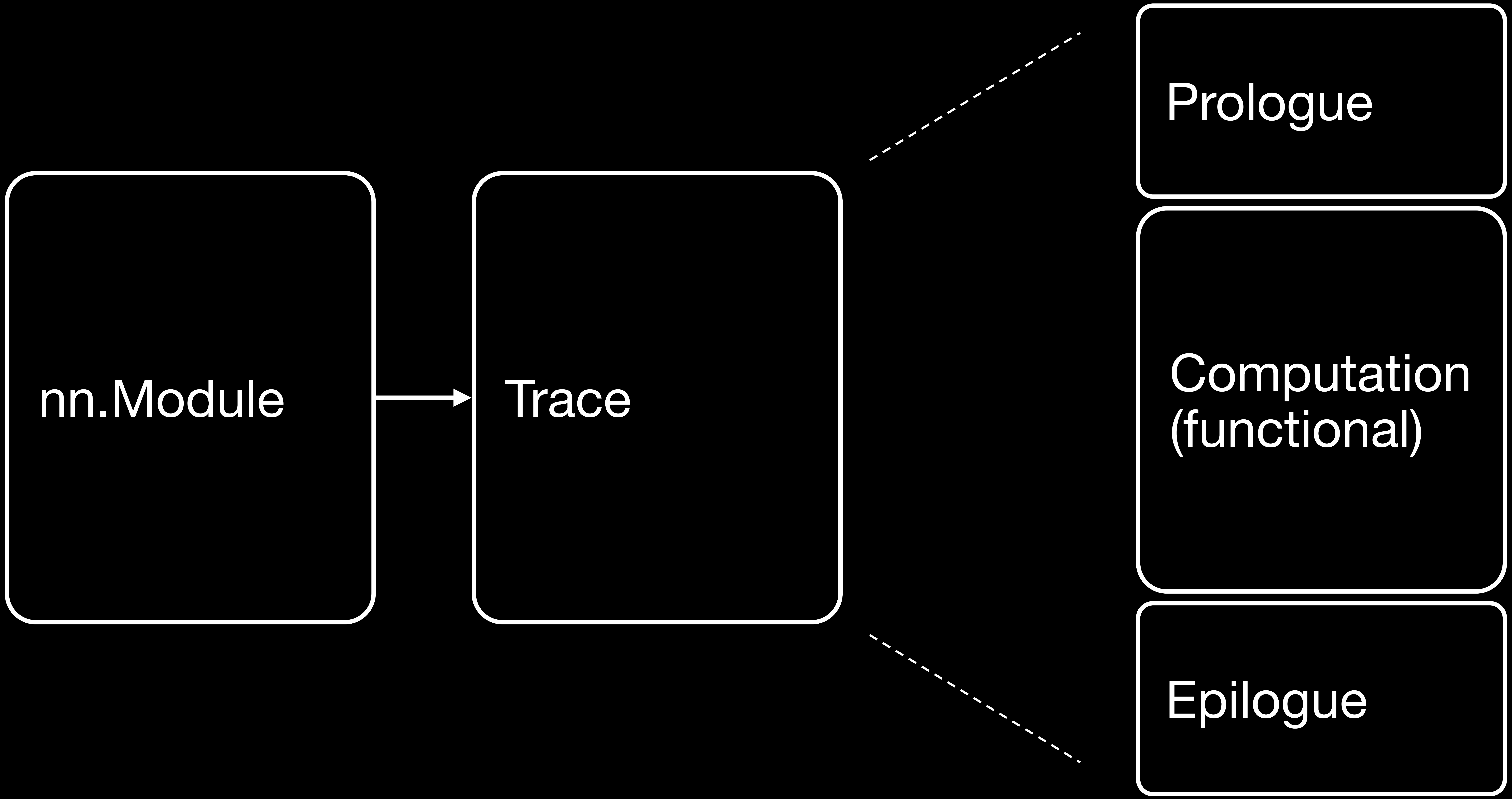


https://github.com/Lightning-AI/lightning-thunder




nn.Module
Trace
Transforms
Executors

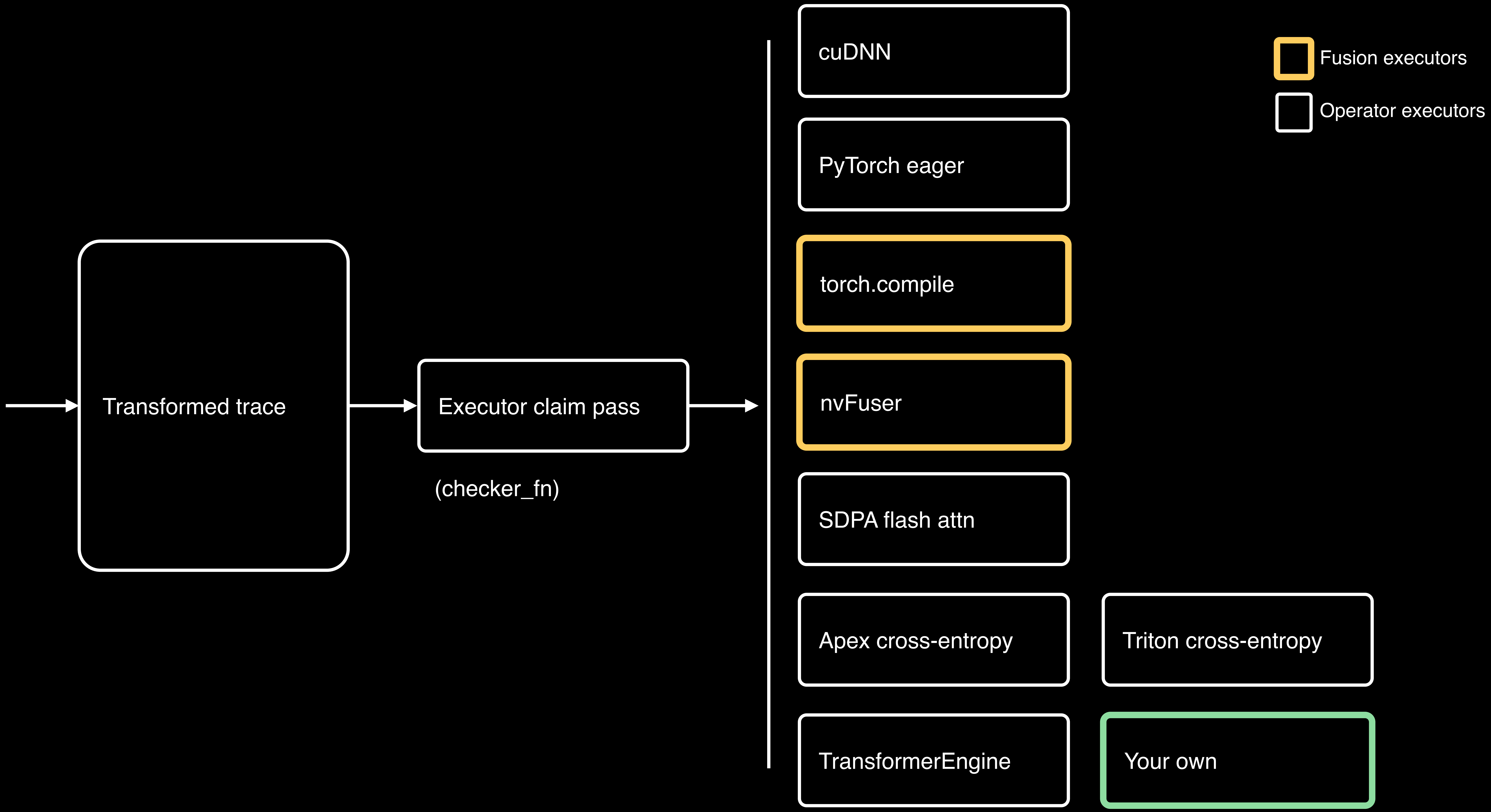

nn.Module
Trace
Transforms
Executors
interpreter.py


nn.Module
Trace
Prologue
Computation
(functional)
Epilogue

Transforms:

- Grad
- Autocast
- Quantization
- Offloading
- Distributed (DDP, FSDP, TP)
- CUDAGraph
- …

Composable


Transformed trace
Executor claim pass
(checker_fn)
cuDNN
PyTorch eager
torch.compile
nvFuser
SDPA flash attn
Apex cross-entropy
Triton cross-entropy
TransformerEngine
Your own
Fusion executors
Operator executors

Thunder with PyTorch

Can use torch.compile as an executor

Coming up:

available as a torch.compile backend

```python
import torch
from thunder.dynamo import ThunderCompiler
backend = ThunderCompiler()

x = torch.ones(2, requires_grad=True)

@torch.compile(backend=backend)
def func(x):
     x = torch.sin(x)
     if x.sum() > 0:
         return x + 1
     else:
         return x - 1

out = func(x)
```


The
Thunder Sessions
Deep learning compilers and how the sausage is made
FRIDAYS @ 11AM EST

Lightning AI