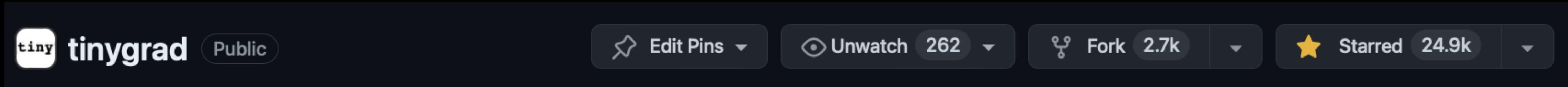


the tiny corp
“commoditize the petaflop”

tinygrad, a deep learning framework

- If you don't have market share, you'll have to build the framework yourself
- Pure Python, < 8000 lines
- Torch-like frontend
- Supports NVIDIA/AMD without CUDA or ROCm runtime



tinybox, a petaflop for the home

- 6x 7900XTX or 4090
- 144 GB, fits unquantized LLaMA-3 70B
- Quiet!
- \$15,000 or \$25,000
- Shipping now



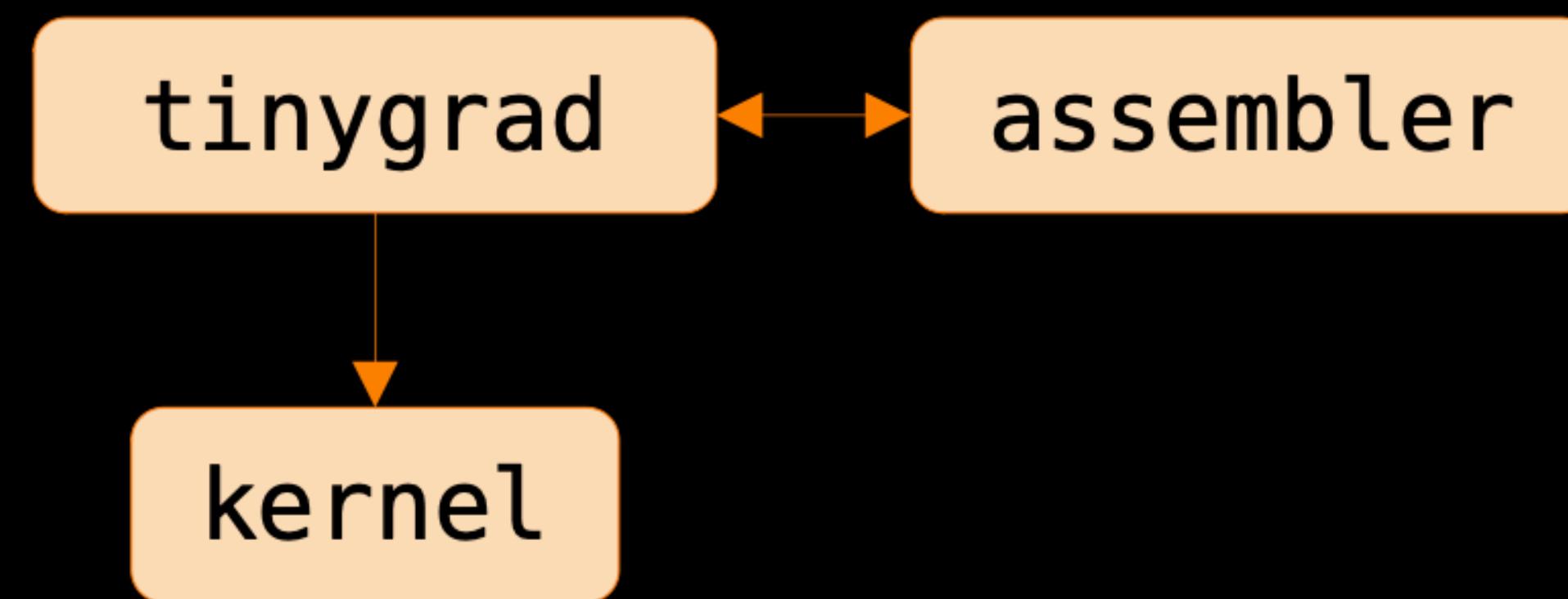
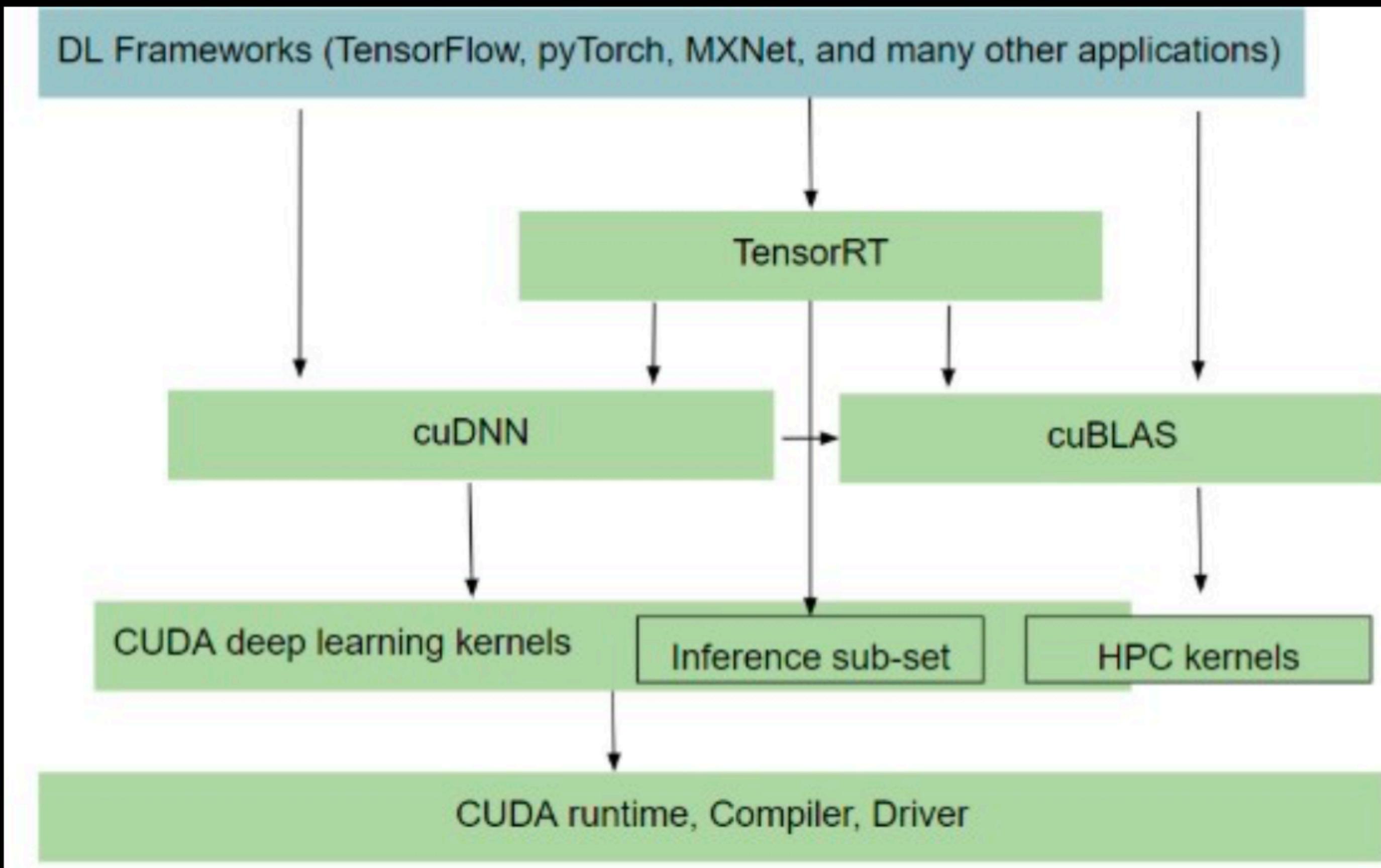
together, MLPerf

MLCommons data as of: 6/12/2024 1:37:25 PM

Benchmark: **Training**
Division/Power: **Closed**
Availability: **Available on-premise**
Round: **v4.0**

Public ID	Availability	Organization	System Name	Total Accelerators	Accelerator Model Name	Accelerators Per Node	Host Processor Model Name	Host Processors ..	Benchmark / ...
4.0-0089	Available on-premise	tinycorp	tinybox red	6	AMD Radeon RX 7900 XTX	6	AMD EPYC 7532 32-Core ..	1	Training resnet
4.0-0088	Available on-premise	tinycorp	tinybox green	6	NVIDIA GeForce RTX 4090	6	AMD EPYC 7532 32-Core ..	1	Latency (In minutes)

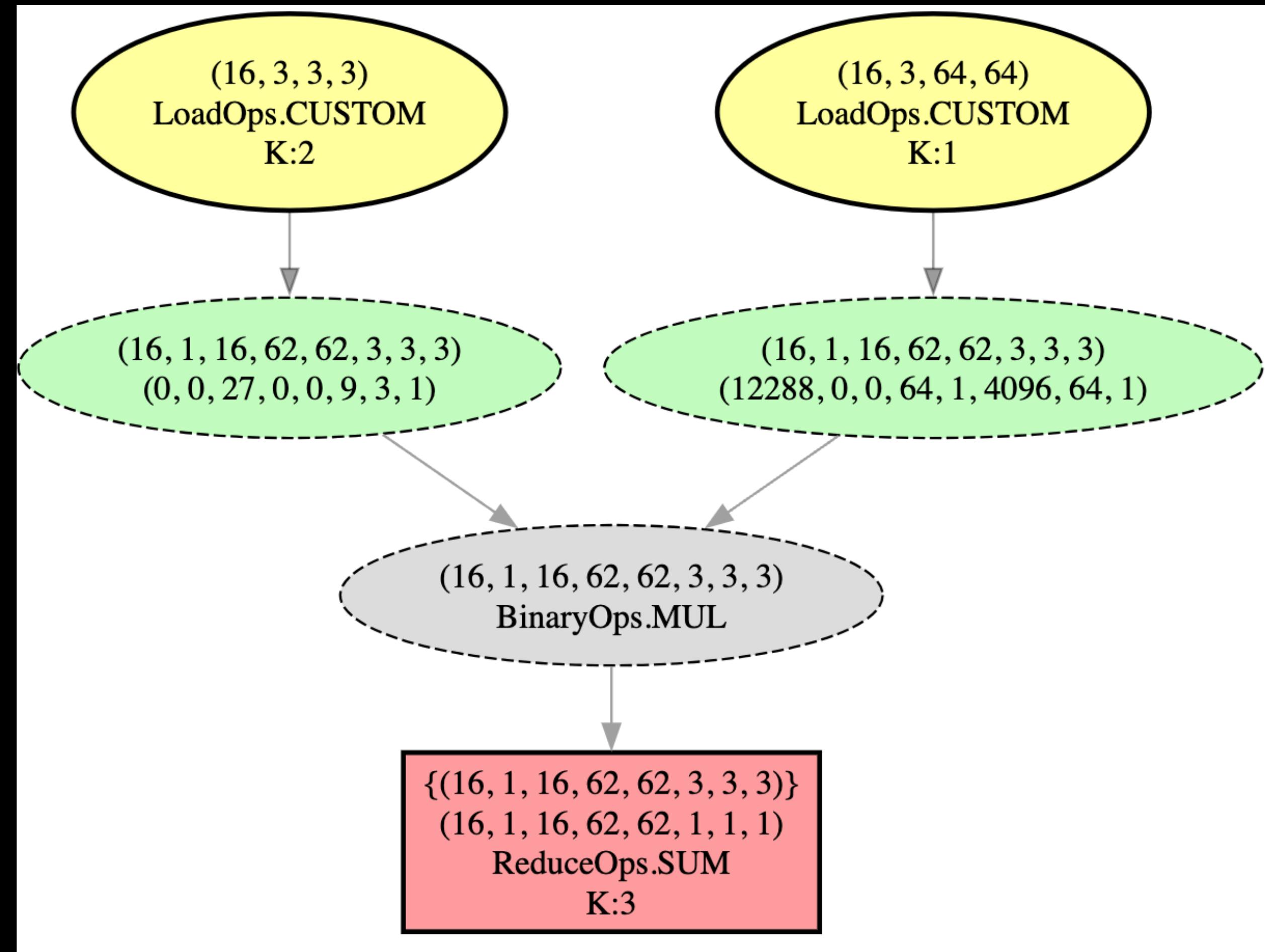
No Dependencies (makes it easy to port)



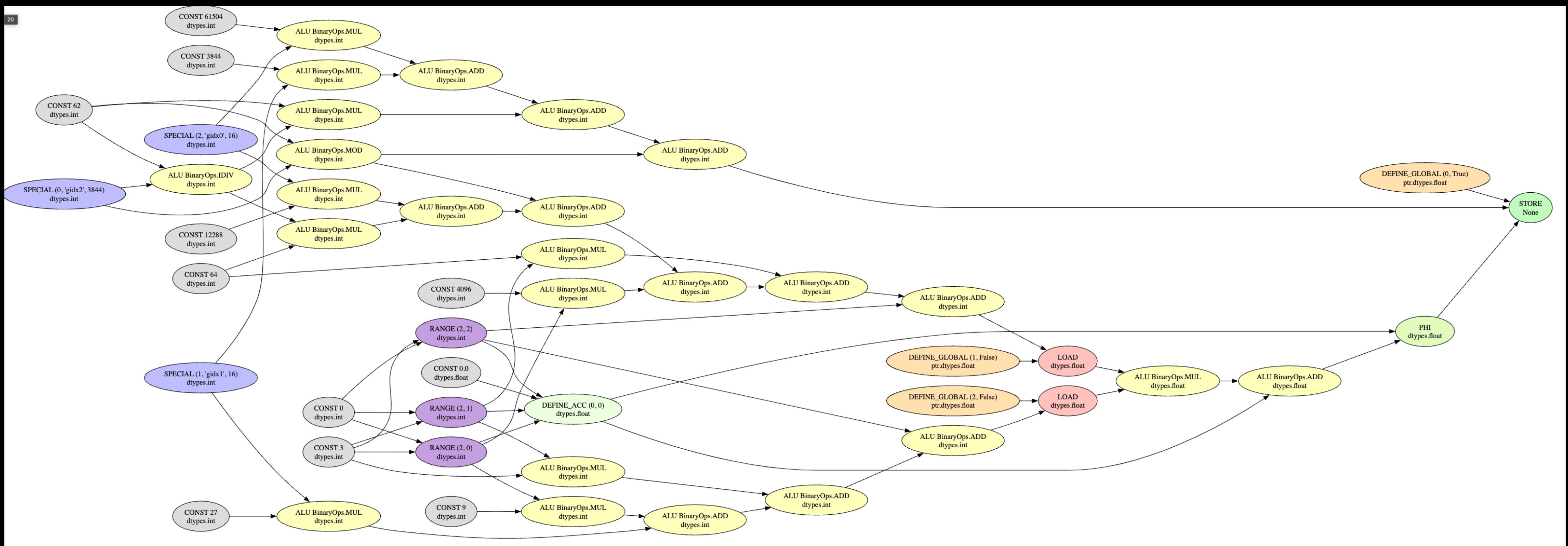
A Conv2D

```
diane@blak:~/tinygrad$ GPU=1 NOOPT=1 DEBUG=4 python3 -c "from tinygrad import Tensor; Tensor.rand(16,3,64,64).conv2d(Tensor.rand(16,3,3,3)).numpy()"  
opened device NPY from pid:65238  
CLDDevice: got 1 platforms and 1 devices  
opened device GPU from pid:65238  
*** CUSTOM      1 custom_random                      arg    1 mem   0.00 GB  
*** CUSTOM      2 custom_random                      arg    1 mem   0.00 GB  
0 └─ BufferOps.STORE MemBuffer(idx=0, dtype=dtypes.float, st=ShapeTracker(views=(View(shape=(16, 1, 16, 62, 62, 1, 1, 1), strides=(61504, 0, 3844,  
1   └─ ReduceOps.SUM (5, 6, 7)  
2     └─ BinaryOps.MUL None  
3       └─ BufferOps.LOAD MemBuffer(idx=1, dtype=dtypes.float, st=ShapeTracker(views=(View(shape=(16, 1, 16, 62, 62, 3, 3, 3), strides=(12288, 0, 0  
4         └─ BufferOps.LOAD MemBuffer(idx=2, dtype=dtypes.float, st=ShapeTracker(views=(View(shape=(16, 1, 16, 62, 62, 3, 3, 3), strides=(0, 0, 27, 0  
__kernel void r_16_16_62_62_3_3_3(__global float* data0, const __global float* data1, const __global float* data2) {  
int gidx2 = get_group_id(0); /* 3844 */  
int gidx1 = get_group_id(1); /* 16 */  
int gidx0 = get_group_id(2); /* 16 */  
int alu0 = (gidx2/62);  
int alu1 = (gidx2%62);  
float acc0 = 0.0f;  
for (int ridx0 = 0; ridx0 < 3; ridx0++) {  
    for (int ridx1 = 0; ridx1 < 3; ridx1++) {  
        for (int ridx2 = 0; ridx2 < 3; ridx2++) {  
            float val0 = data1[(gidx0*12288)+(alu0*64)+alu1+(ridx0*4096)+(ridx1*64)+ridx2];  
            float val1 = data2[(gidx1*27)+(ridx0*9)+(ridx1*3)+ridx2];  
            acc0 = ((val0*val1)+acc0);  
        }  
    }  
}  
data0[(gidx0*61504)+(gidx1*3844)+(alu0*62)+alu1] = acc0;  
}  
*** GPU      3 r_16_16_62_62_3_3_3                      arg    3 mem   0.00 GB tm   5233.62us/    5.23ms (    10.15 GFLOPS,    0.90 GB/s)  
opened device CLANG from pid:65238  
*** CLANG    4 copy    3.94M,    CLANG <- GPU          arg    2 mem   0.01 GB tm   1361.87us/    6.60ms (    0.00 GFLOPS,    2.89 GB/s)  
avg:    8.06 GFLOPS    1.31 GB/s           total:    4 kernels    0.05 GOPS    0.01 GB    6.60 ms  
diane@blak:~/tinygrad$
```

A Conv2D (graphed)



A Conv2D (lowered)



The Bitter Lesson (28x faster)

- Search to explore different ways to optimize
- Use the fastest one

```
diane@blak:~/tinygrad$ BEAM=2 GPU=1 DEBUG=2 python3 -c "from tinygrad import Tensor; Tensor.rand(16,3,64,64).conv2d(Tensor.rand(16,3,3,3)).numpy()"
opened device NPY from pid:65828
CLDDevice: got 1 platforms and 1 devices
opened device GPU from pid:65828
*** CUSTOM    1 custom_random                         arg   1 mem  0.00 GB
*** CUSTOM    2 custom_random                         arg   1 mem  0.00 GB
0.00s:          from  1 ->  1 actions  16   16   62   62   3   3   3
0.34s:  326.75 us from  26 ->  26 actions  16   62   62   16   3   3   3
1.02s:  43.58 us from  46 ->  46 actions  4    62   62   16   4   3   3   3
1.80s:  43.58 us from  41 ->  38 actions  4    62   62   16   4   3   3   3
hc   : 4 31 31 4 2 2 3 4 4 3 3   :  163.21 us < beam2 : 4 62 62 16 4 3 3 3   :  171.62 us
*** GPU      3 r_4_31_31_4_2_2_3_4_4_3_3           arg   3 mem  0.00 GB tm  183.83us/  0.18ms ( 289.06 GFLOPS,  25.70 GB/s)
opened device CLANG from pid:65828
*** CLANG    4 copy   3.94M, CLANG <- GPU          arg   2 mem  0.01 GB tm  1132.58us/  1.32ms ( 0.00 GFLOPS,  3.48 GB/s)
avg: 40.37 GFLOPS  6.58 GB/s      total: 4 kernels   0.05 GOPS  0.01 GB  1.32 ms
diane@blak:~/tinygrad$
```

Get Started

- tinygrad runs almost everywhere, try it!
- docs.tinygrad.org
- Our goal is to build chips, but first, the software.

The screenshot shows the homepage of the tinygrad documentation at docs.tinygrad.org. The page has a dark background with light-colored text. At the top left is the "tinygrad docs" logo. The main title "tinygrad documentation" is centered above a welcome message. To the right are two small icons. On the left, there's a sidebar with navigation links: Home, Quickstart, Showcase, MNIST Tutorial, API Reference (with a dropdown menu for Tensor, Creation, Movement, Ops, Function, dtypes, nn (Neural Networks), and Environment Variables), and a link to developer docs. The main content area contains text about the API stability and installation instructions, followed by a code block for cloning the GitHub repository. Below that, there's a link to the MNIST tutorial and a note about developer documentation.

tinygrad docs

tinygrad documentation

Welcome to the docs for tinygrad. This page is for users of the tinygrad library. tinygrad is not 1.0 yet, but it will be soon. The API has been pretty stable for a while.

While you can `pip install tinygrad`, we encourage you to install from source:

```
git clone https://github.com/tinygrad/tinygrad.git
cd tinygrad
python3 -m pip install -e .
```

After you have installed tinygrad, try the [MNIST tutorial](#)

We also have [developer docs](#), and Di Zhu has created a bunch of tutorials to help understand how tinygrad works.

Home

Quickstart

Showcase

MNIST Tutorial

API Reference

Tensor

Creation

Movement

Ops

Function

dtypes

nn (Neural Networks)

Environment Variables