Find Inefficiencies and Rapid Model Profiling with CentML DeepView

ASPLOS '23 Workshop, March 25th

Yubo Gao <ybgao@centml.ai>
CentML Inc, University of Toronto

Agenda

1. Introduction to training optimizations **[1:50pm - 2:30pm]**
    a. **Why** do we care?
    b. **What** are the common optimizations?
2. Performance debugging with DeepView **[2:30pm - 3pm]**
    a. Visually identify performance bottlenecks
    b. Value of performance prediction in optimization workflow

CentML

Why optimize?

Increasing resources required used to train large models.

Resource underutilization is a significant problem:

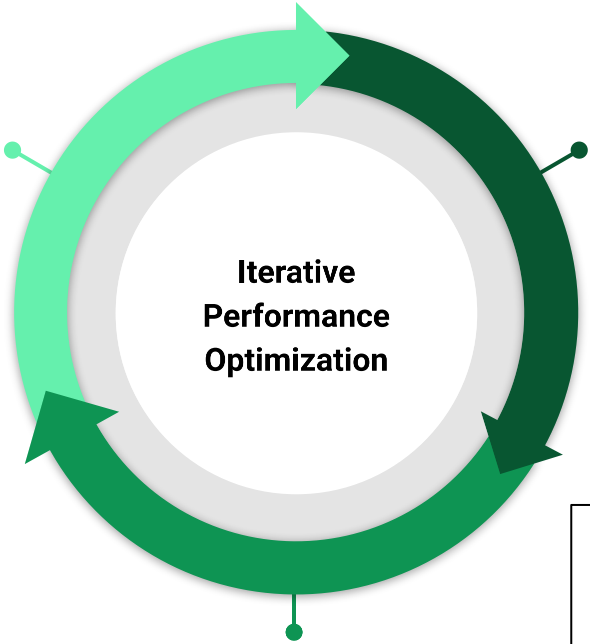
- Observed average GPU utilization below 30% at a large AI research compute cluster.
- Significant resource and energy waste.
- Low utilization leads to lower throughput (increased job completion time).

Typical training optimization workflow

How much improvement?

Energy improvements?

Quantify Improvements



Identify bottlenecks

Compute bound or memory bound?

Where (which layer, or lines of code) did the time go?

Understand energy and environmental impacts

Apply training optimizations

How much improvement to expect?

Pick the most suitable hardware.

Environment Setup

Please follow “Environment Setup” at:

https://centml.github.io/asplos23-tutorial/deepview.html

Interactive Demo

Exploring system optimizations for DL training

Existing DL Profilers

nvprof
Nsight compute
Nsight Systems
dlprof

Intel vTune

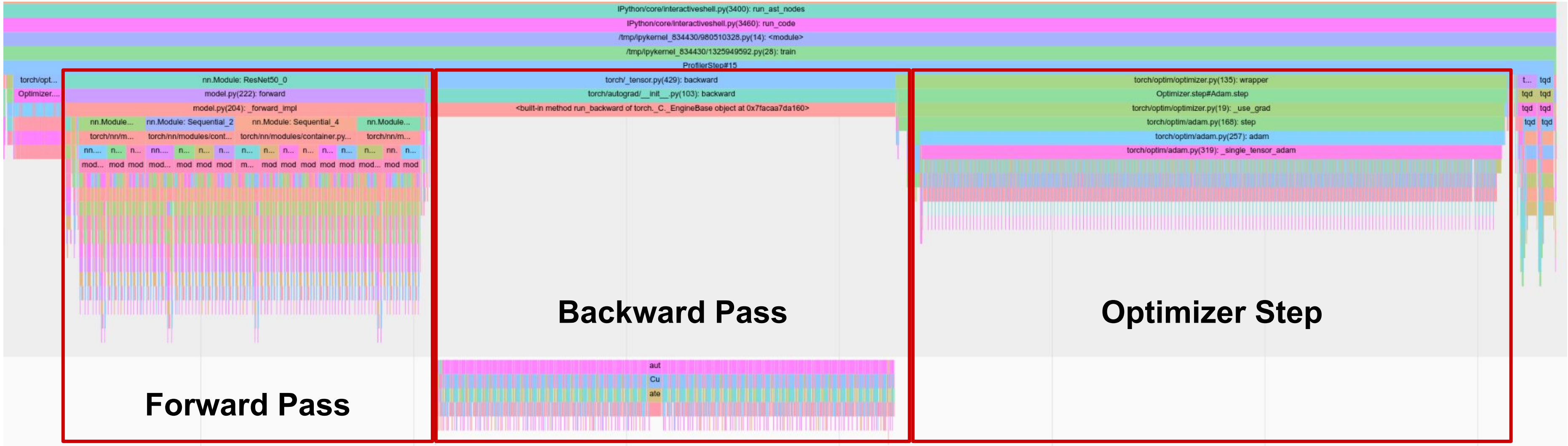
Torch.profile

PyTorch Lightning profiler

TensorBoard profiler

1. Incorrect granularity
2. Lack of interactivity
3. Lack of predictive capabilities

Incorrect granularity



Forward Pass: Difficult to attribute runtime to lines of code.

Backward Pass: Only layer names are visible, impossible to trace back to code.

Incorrect granularity makes optimization difficult.

Incorrect granularity

Top 10 GPU Ops

GPU Time (ns)	Op Name	Op Type	Calls	TC Eligible	Using TC
165,437,894	gradients/resnet50/conv2d/conv2d/Conv2D_grad/Conv2DBackpropFilter	Conv2DBackpropFilter	21	✔	✖
139,735,214	gradients/resnet50/conv2d/BatchNorm/FusedBatchNormV3_grad/FusedBatchNormGradV3	FusedBatchNormGradV3	21	✖	✖
139,549,652	resnet50/conv2d/conv2d/Conv2D	Conv2D	21	✔	✖
106,125,030	resnet50/btlnck_block_2_0/bottleneck_2/conv2d/Conv2D	Conv2D	21	✔	✖
93,436,444	gradients/resnet50/btlnck_block_1_0/bottleneck_2/conv2d/Conv2D_grad/Conv2DBackpropInput	Conv2DBackpropInput	21	✔	✔
84,582,994	gradients/resnet50/btlnck_block_0_0/bottleneck_3/BatchNorm/FusedBatchNormV3_grad/FusedBatchNormGradV3	FusedBatchNormGradV3	21	✖	✖
84,547,887	gradients/resnet50/btlnck_block_0_0/shortcut/conv2d/BatchNorm/FusedBatchNormV3_grad/FusedBatchNormGradV3	FusedBatchNormGradV3	21	✖	✖
84,367,698	gradients/resnet50/btlnck_block_0_1/bottleneck_3/BatchNorm/FusedBatchNormV3_grad/FusedBatchNormGradV3	FusedBatchNormGradV3	21	✖	✖
83,685,782	gradients/resnet50/btlnck_block_0_2/bottleneck_3/BatchNorm/FusedBatchNormV3_grad/FusedBatchNormGradV3	FusedBatchNormGradV3	21	✖	✖
73,962,717	gradients/resnet50/btlnck_block_1_0/bottleneck_2/conv2d/Conv2D_grad/Conv2DBackpropFilter	Conv2DBackpropFilter	21	✔	✔

NVIDIA DLProf [1]

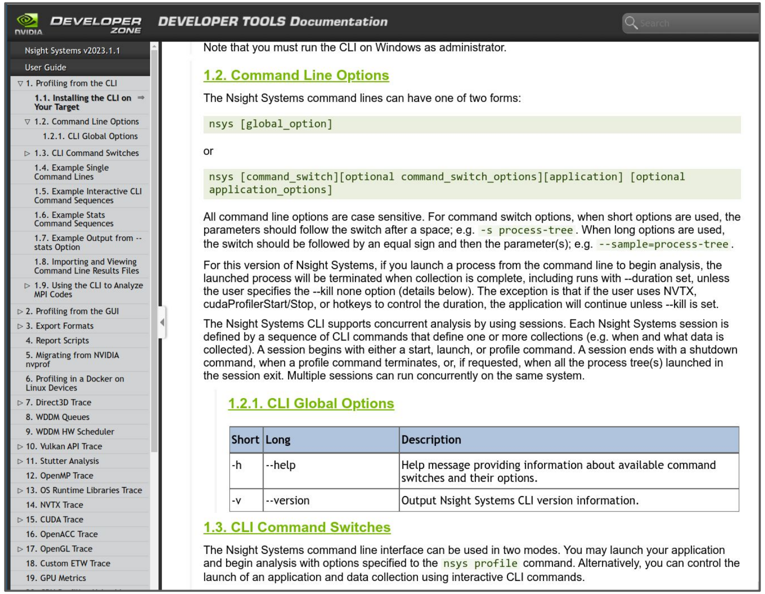
Input Op statistics

Input Op	Count	Total Time (in ms)	Total Time (as % of total input-processing time)	Total Self Time (in ms)	Total Self Time (as % of total input-processing time)	Category
Iterator::Prefetch::Generator	80	89,282	43.3%	89,282	43.3%	Preprocessing
Iterator::Prefetch	88	80,228	38.9%	80,228	38.9%	Preprocessing
Iterator::Model::Prefetch::Rebatch::Prefetch::MapAndBatch	10	11,161	5.4%	11,161	5.4%	Preprocessing
Iterator::Model::Prefetch	87	11,161	5.4%	11,161	5.4%	Preprocessing
Iterator::Model::Prefetch::Rebatch::Prefetch	10	11,160	5.4%	11,160	5.4%	Preprocessing
Iterator::Model::Prefetch::Rebatch::Prefetch::MapAndBatch::ShuffleAndRepeat::ParallelInterleaveV3[94]::FlatMap[0]::TFRecord	3	868	0.4%	868	0.4%	Advanced file read
Iterator::Model::Prefetch::Rebatch::Prefetch::MapAndBatch::ShuffleAndRepeat::ParallelInterleaveV3[91]::FlatMap[0]::TFRecord	3	743	0.4%	743	0.4%	Advanced file read
Iterator::Model::Prefetch::Rebatch::Prefetch::MapAndBatch::ShuffleAndRepeat::ParallelInterleaveV3[97]::FlatMap[0]::TFRecord	3	521	0.3%	521	0.3%	Advanced file read

Tensorflow Profiler [2]

[1]: https://docs.nvidia.com/deeplearning/frameworks/dlprof-user-guide/index.html
[2]: https://www.tensorflow.org/guide/profiler

Lack of interactivity



DEVELOPER ZONE DEVELOPER TOOLS Documentation

Nsight Systems v2023.1.1

User Guide

Note that you must run the CLI on Windows as administrator.

1.2. Command Line Options

The Nsight Systems command lines can have one of two forms:

```
nsys [global_option]
```

or

```
nsys [command_switch][optional command_switch_options][application] [optional application_options]
```

All command line options are case sensitive. For command switch options, when short options are used, the parameters should follow the switch after a space; e.g. `-s process-tree`. When long options are used, the switch should be followed by an equal sign and then the parameter(s); e.g. `--sample=process-tree`.

For this version of Nsight Systems, if you launch a process from the command line to begin analysis, the launched process will be terminated when collection is complete, including runs with --duration set, unless the user specifies the --kill none option (details below). The exception is that if the user uses NVTX, cudaProfilerStart/Stop, or hotkeys to control the duration, the application will continue unless --kill is set.

The Nsight Systems CLI supports concurrent analysis by using sessions. Each Nsight Systems session is defined by a sequence of CLI commands that define one or more collections (e.g. when and what data is collected). A session begins with either a start, launch, or profile command. A session ends with a shutdown command, when a profile command terminates, or, if requested, when all the process tree(s) launched in the session exit. Multiple sessions can run concurrently on the same system.

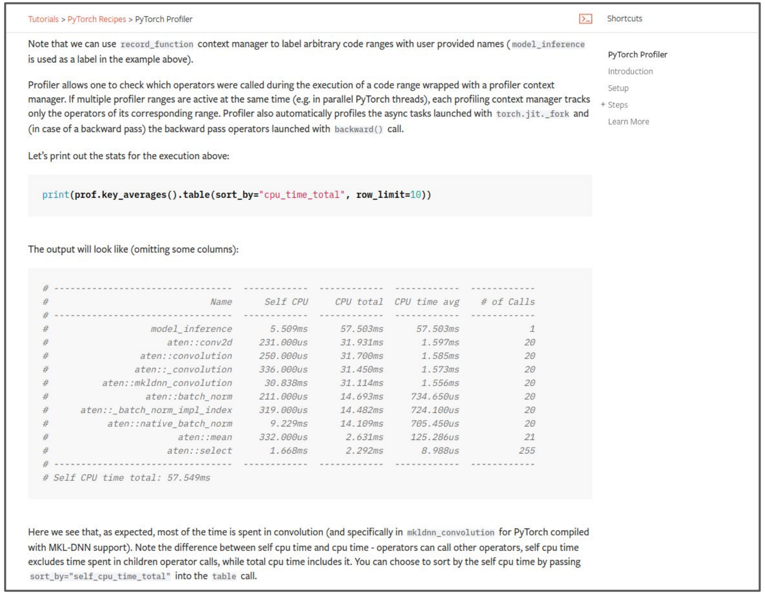
1.2.1. CLI Global Options

Short	Long	Description
-h	--help	Help message providing information about available command switches and their options.
-v	--version	Output Nsight Systems CLI version information.

1.3. CLI Command Switches

The Nsight Systems command line interface can be used in two modes. You may launch your application and begin analysis with options specified to the `nsys profile` command. Alternatively, you can control the launch of an application and data collection using interactive CLI commands.

NVIDIA Nsight Systems



Tutorials > PyTorch Recipes > PyTorch Profiler

Shortcuts

Note that we can use `record_function` context manager to label arbitrary code ranges with user provided names (`model_inference` is used as a label in the example above).

Profiler allows one to check which operators were called during the execution of a code range wrapped with a profiler context manager. If multiple profiler ranges are active at the same time (e.g. in parallel PyTorch threads), each profiling context manager tracks only the operators of its corresponding range. Profiler also automatically profiles the async tasks launched with `torch.jit._fork` and (in case of a backward pass) the backward pass operators launched with `backward()` call.

Let's print out the stats for the execution above:

```
print(prof.key_averages().table(sort_by="cpu_time_total", row_limit=10))
```

The output will look like (omitting some columns):

```
# ---------------------------------  ------------  ------------  ------------  ------------
#                              Name      Self CPU     CPU total  CPU time avg    # of Calls
# ---------------------------------  ------------  ------------  ------------  ------------
#                   model_inference       5.509ms      57.503ms      57.503ms             1
#                      aten::conv2d     231.000us      31.931ms       1.597ms            20
#                 aten::convolution     250.000us      31.700ms       1.585ms            20
#                aten::_convolution     336.000us      31.450ms       1.573ms            20
#          aten::mkldnn_convolution      30.838ms      31.114ms       1.556ms            20
#                  aten::batch_norm     211.000us      14.693ms     734.650us            20
#      aten::_batch_norm_impl_index     319.000us      14.482ms     724.100us            20
#           aten::native_batch_norm       9.229ms      14.109ms     705.450us            20
#                        aten::mean     332.000us       2.631ms     125.286us            21
#                      aten::select       1.668ms       2.292ms       8.988us           255
# ---------------------------------  ------------  ------------  ------------  ------------
# Self CPU time total: 57.549ms
```

Here we see that, as expected, most of the time is spent in convolution (and specifically in `mkldnn_convolution` for PyTorch compiled with MKL-DNN support). Note the difference between self cpu time and cpu time - operators can call other operators, self cpu time excludes time spent in children operator calls, while total cpu time includes it. You can choose to sort by the self cpu time by passing `sort_by="self_cpu_time_total"` into the `table` call.

PyTorch Profiler

DeepView

Interactive Profiler [3]

Identifies performance bottlenecks

Enables rapid iterative profiling

Quantifies energy consumption and environmental impacts of training jobs.

Runtime Predictor [4]

Predicts a deep neural network's training iteration execution time on a different GPU.

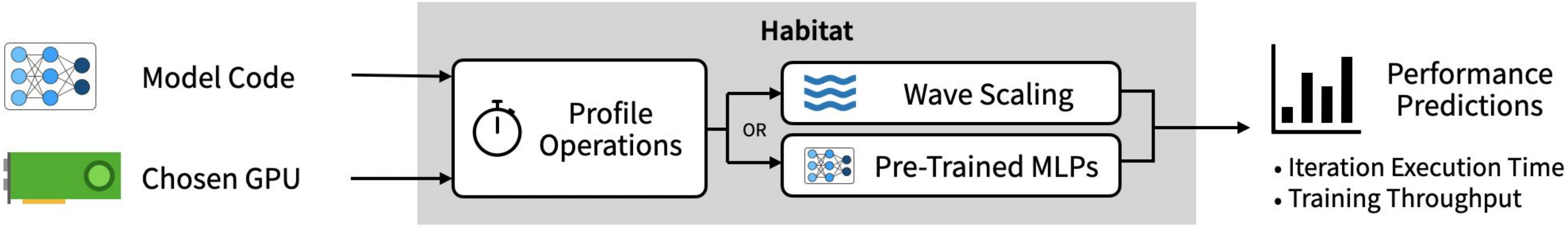
Recommends the most cost/time effective hardware option for your workload

[3]: Skyline: Interactive In-Editor Computational Performance Profiling for Deep Neural Network Training, Geoffrey Yu, et. al.
[4]: Habitat: A Runtime-Based Computational Performance Predictor for Deep Neural Network Training, Geoffrey Yu, et. al.

Interactive Demo

Iterative Profiling with CentML DeepView

GPU Runtime Predictor (DeepView.Predict)

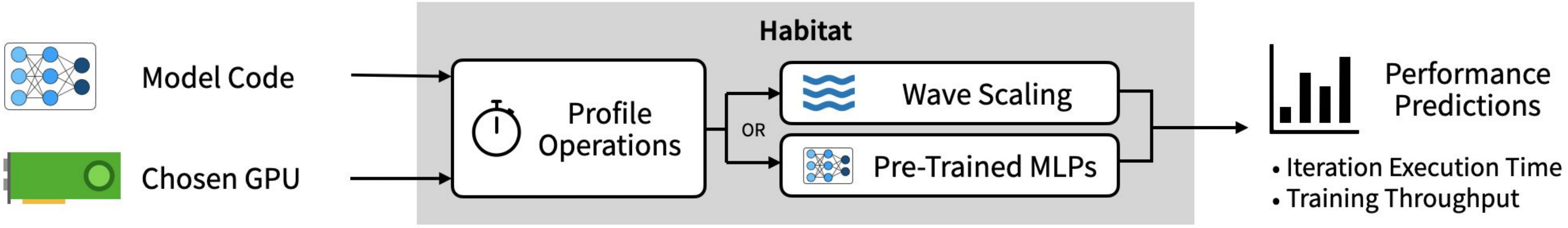


Pick the best GPU for your training job, whether it is:

- Deciding which new GPU to purchase for your local workstation
- Which cloud GPU instance to pick
- Efficiently schedule jobs in a heterogeneous GPU cluster

Tedious to benchmark model on all the available GPUs.

GPU Runtime Predictor (DeepView.Predict)



Wave Scaling - If the same kernels execute on the source and target GPUs, then scale based on hardware parameters.

Pre-Trained MLPs - If not, learn runtimes of different operators with a pretrained model.

GPU Runtime Predictor (DeepView.Predict)

Why predict? Why not ....

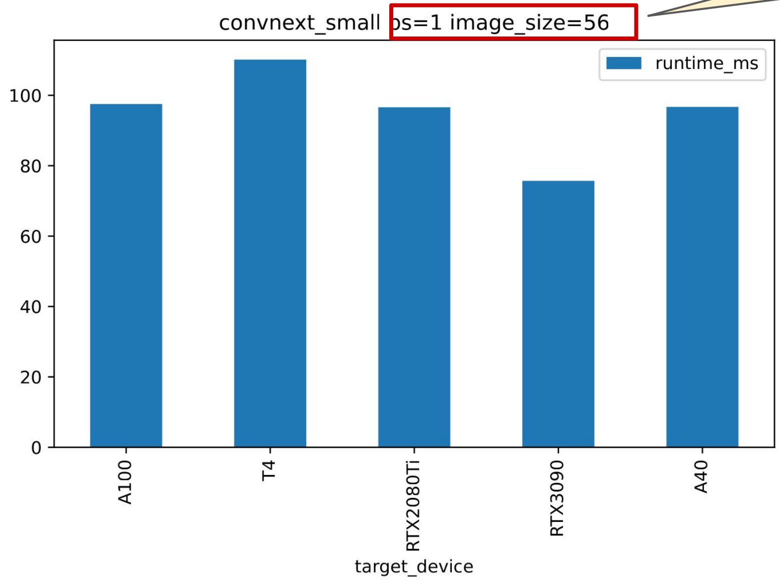
- ➔ Measure the performance directly?
- ➔ Apply heuristics?
- ➔ Use standard benchmarks?
- ➔ Always use the best available GPU?

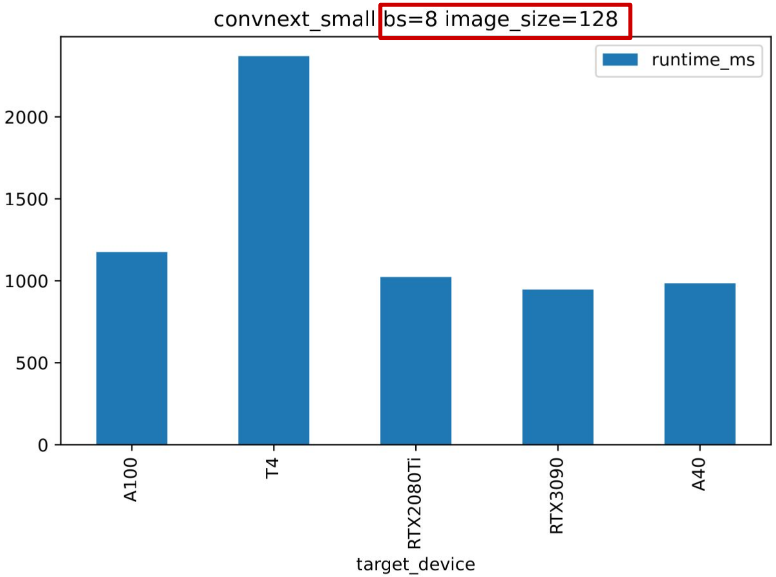
Aren’t more powerful GPUs better?

Not all the time! Not alwa

Small **inputs** underutilize GPU.

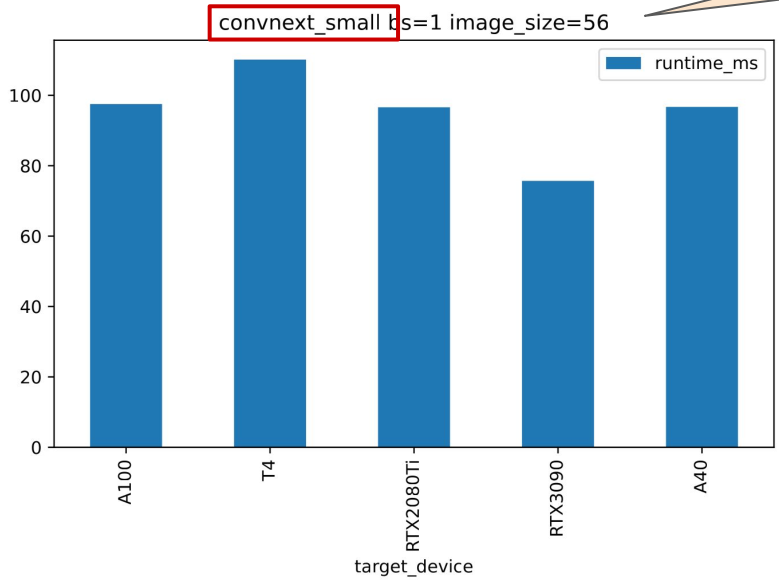
e A40.

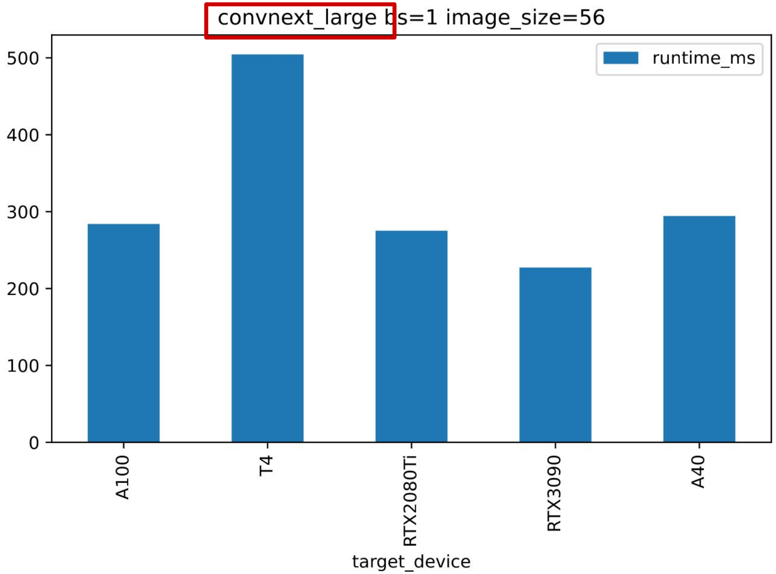




Aren’t more powerful GPUs better?

Small **models** under-utilize GPU.





Which GPUs are supported?

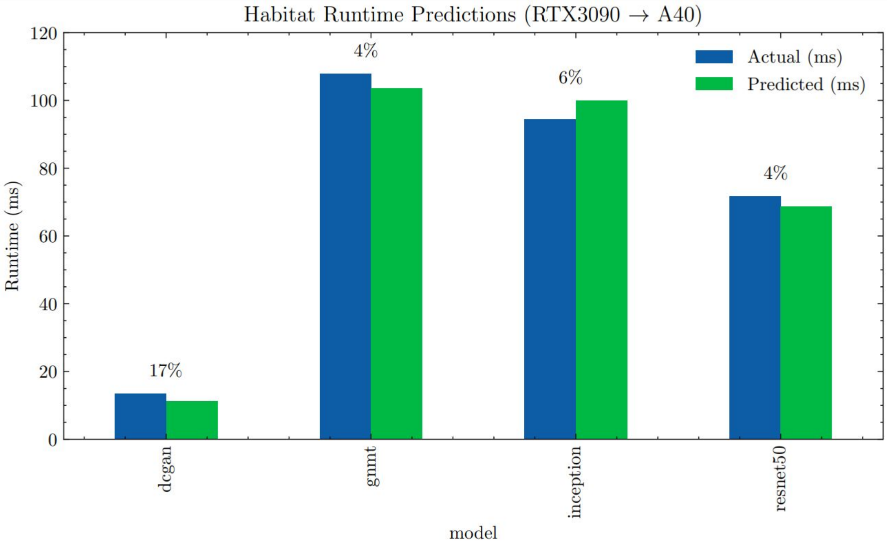
Generation \ Use Case	Desktop/Consumer	Workstation/Server
Pascal	GTX1080Ti	Quadro P4000 P4 P100
Volta		V100
Turing	RTX2070 RTX2080Ti	Quadro RTX4000
Tesla		T4
Ampere	RTX3090	A100 A40 A4000
Hopper		H100 (coming soon)

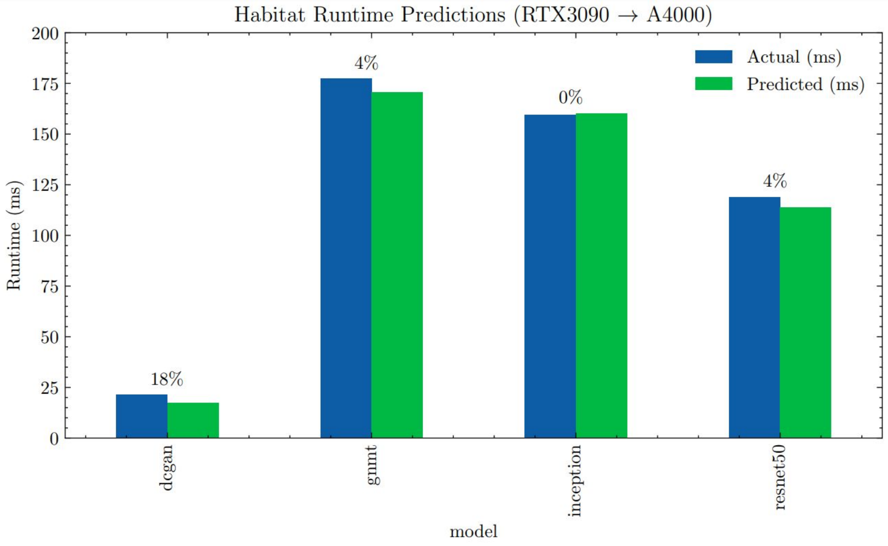
- Beta

- Deprecated / Not available

CentML

How accurate is DeepView.Predict?





Prediction errors are generally no more than 10%.

Cloud Deployment

Profiling | Deployment

Deployment Target

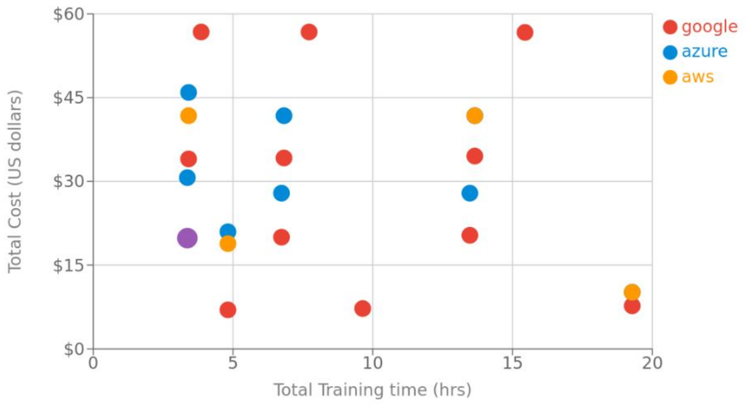
Estimation for **4 million** total iterations

Providers

Filter by provider: All

Filter by GPU: All

Filter Max Number of GPUs: 1 2 4 **all**



Deployment Plan

n1-standard-1

Estimated Cost: $20

Estimated Training Time: 3.4 Hours

GPU	Num. GPU	VRAM
p100	4	16 GB

CentML DeepView documentation

Search the docs ...

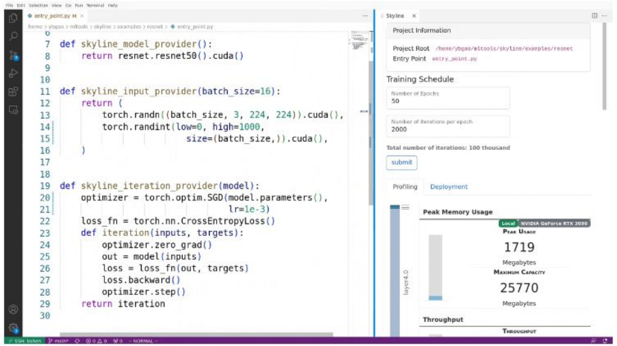
Theme by the Executable Book Project

Contents

CentML DeepView

CentML DeepView provides an integrated experience which allows ML practioners to:

- Visually identify model bottlenecks
- Perform rapid iterative profiling
- Understand energy consumption and environmental impacts of training jobs
- Predict deployment time and cost to cloud hardware



Getting Started

Follow the instructions depending on your setup

- Local workstation GPU
- Remote GPU with SSH access
- Clusters, containers, and other setups where SSH is not possible

DeepView.Profile

Deepview.Profile is our free and open source tool provides easier way to identify bottlenecks and perform rapid iterative profiling for Deep Learning. To find out more, visit DeepView.Profile

DeepView.Predict

DeepView.Predict is a tool that predicts a deep neural network's training iteration execution time on a given GPU. It currently supports PyTorch. To find out more, visit DeepView.Predict

Get started with DeepView at `docs.centml.ai`!

What we do: system-level optimizations

CentML expertise

Profiling Tool: DeepView

System-Level Optimizations

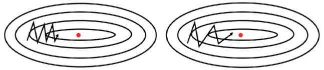
Hardware-Specific Optimizations

Models

ResNet **GPT3** **BERT** **SSD**

Algorithms

SGD without momentum SGD with momentum



Frameworks

TensorFlow PYTORCH mxnet

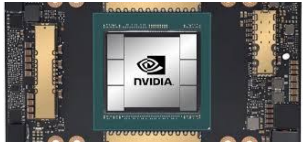
Libraries and ML Compilers

cuDNN **TVM** **XLA**

Hardware

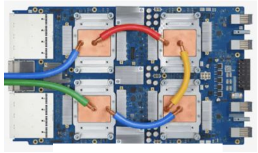
GPU

aws



A100

Microsoft Azure



TPU V3

Google Cloud

Additional Optimizations: Horizontal Fusion

Horizontally fused training array for efficient training

- Best for training small models + hyper-parameter tuning

Thank you!

To learn more about DeepView, visit `docs.centml.ai`

Also check us out at **`centml.ai`**,

or contact us at **`ybgao@centml.ai`**

Please provide your feedbacks at: