DATA LOADING:
the Next Frontier in Scale-out Deep Learning

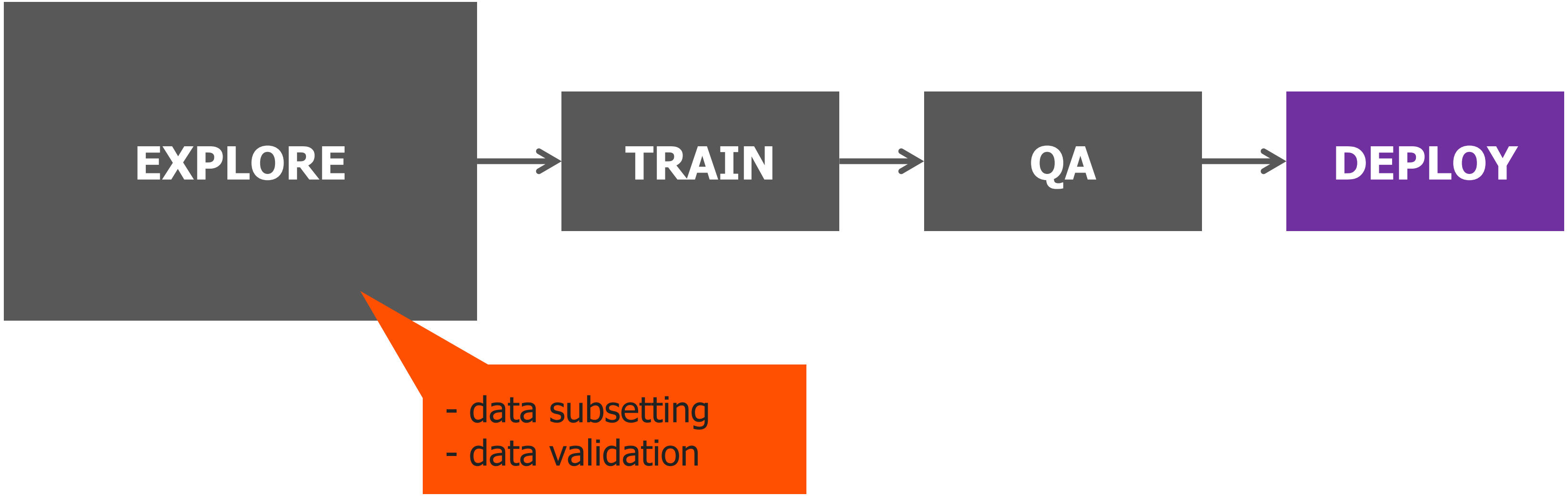
Emily Watkins

DEEP LEARNING EXECUTION FLOW

ON PAPER

EXPLORE → **TRAIN** → **QA** → **DEPLOY**

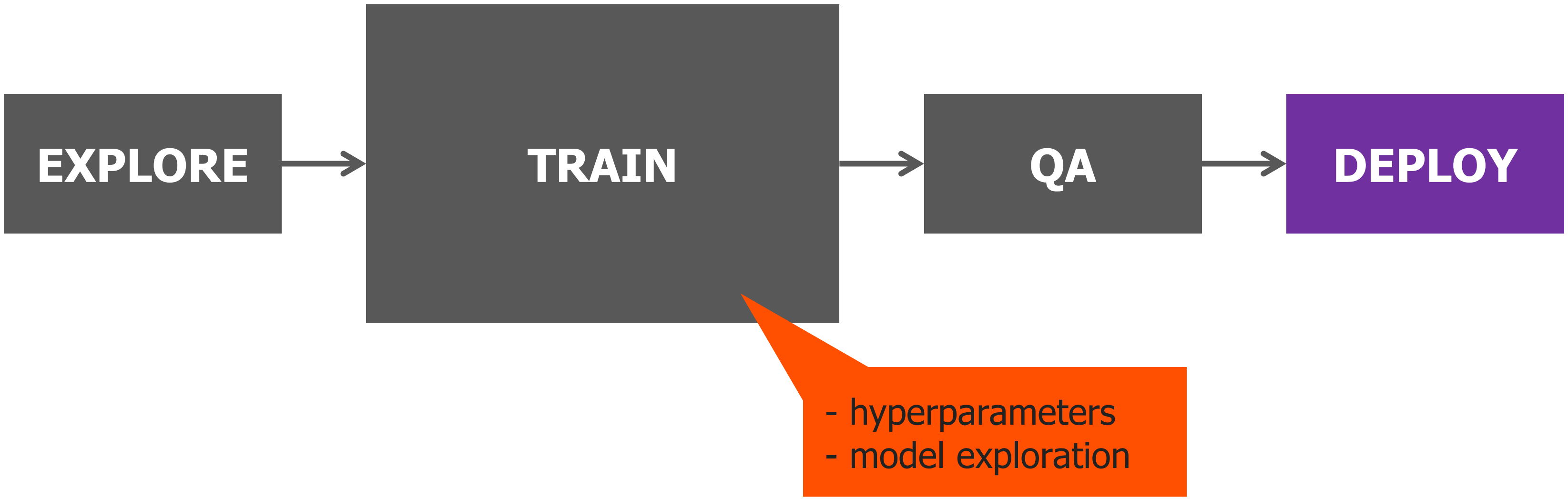
DEEP LEARNING EXECUTION FLOW
IN REAL LIFE



PURESTORAGE®

DEEP LEARNING EXECUTION FLOW

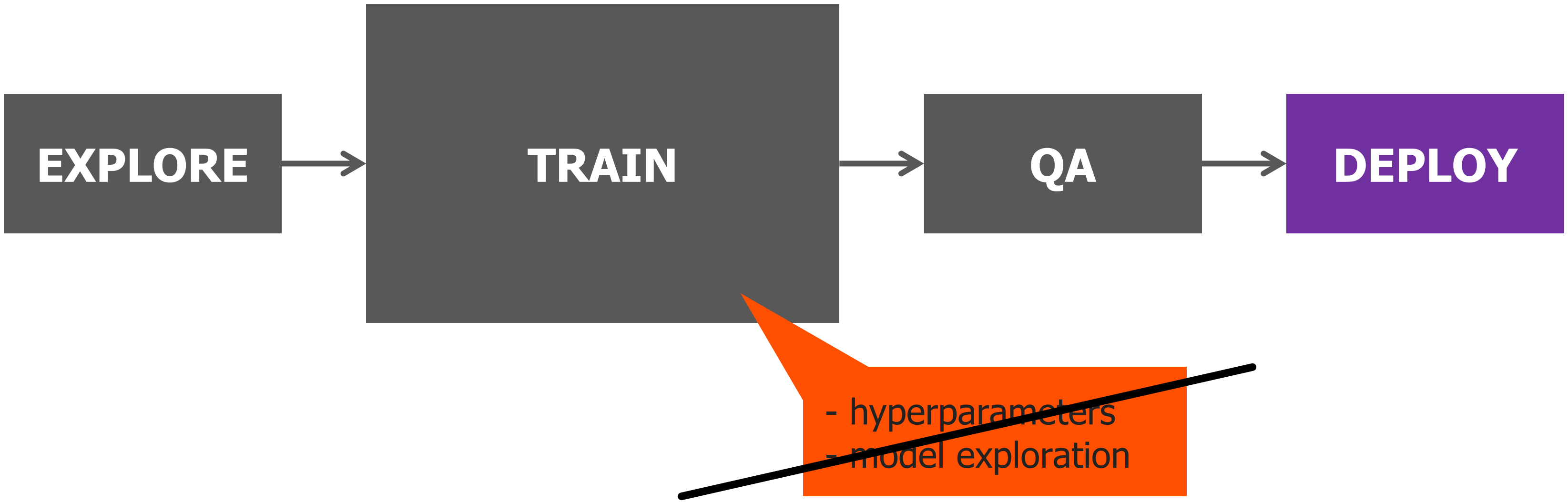
IN REAL LIFE



PURESTORAGE®

DEEP LEARNING EXECUTION FLOW

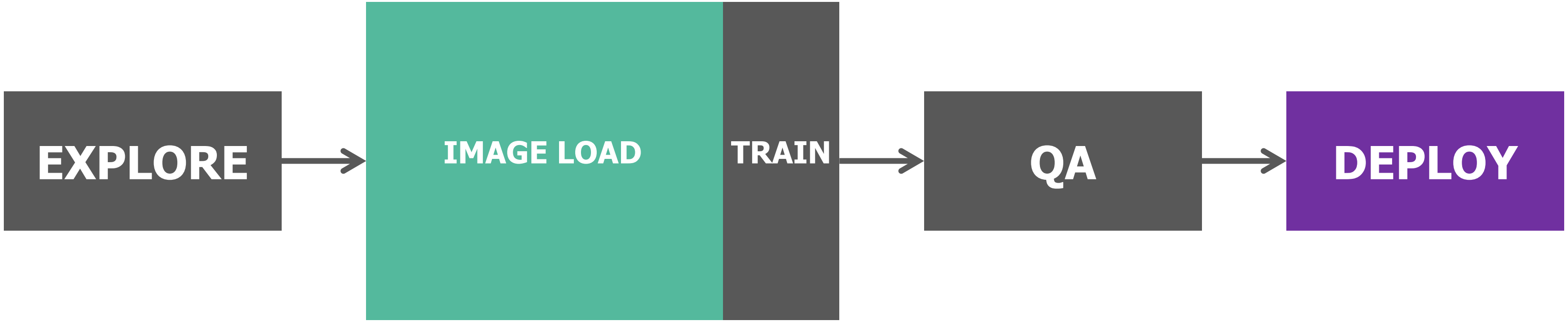
IN REAL LIFE



PURESTORAGE®

DEEP LEARNING EXECUTION FLOW

IN REAL LIFE



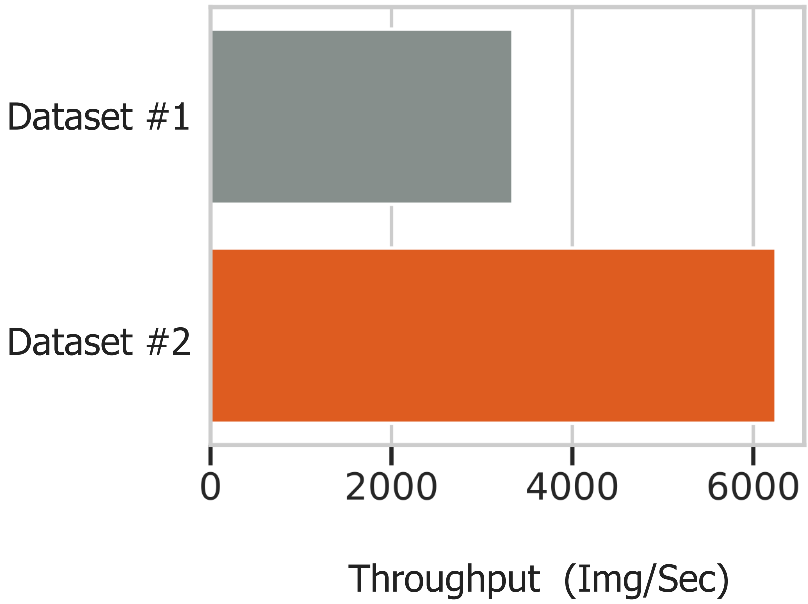
PURESTORAGE®

1 TRAINING ITERATION

PURESTORAGE®

1 TRAINING ITERATION

DATASETS IMPACT THROUGHPUT



PURESTORAGE®

Impact of data loading

1. Overview of input pipelines & impact of data format.

2. How does data load time fit into overall training throughput?

3. Options for improving throughput based on training datasets.

PURESTORAGE®

Impact of data loading

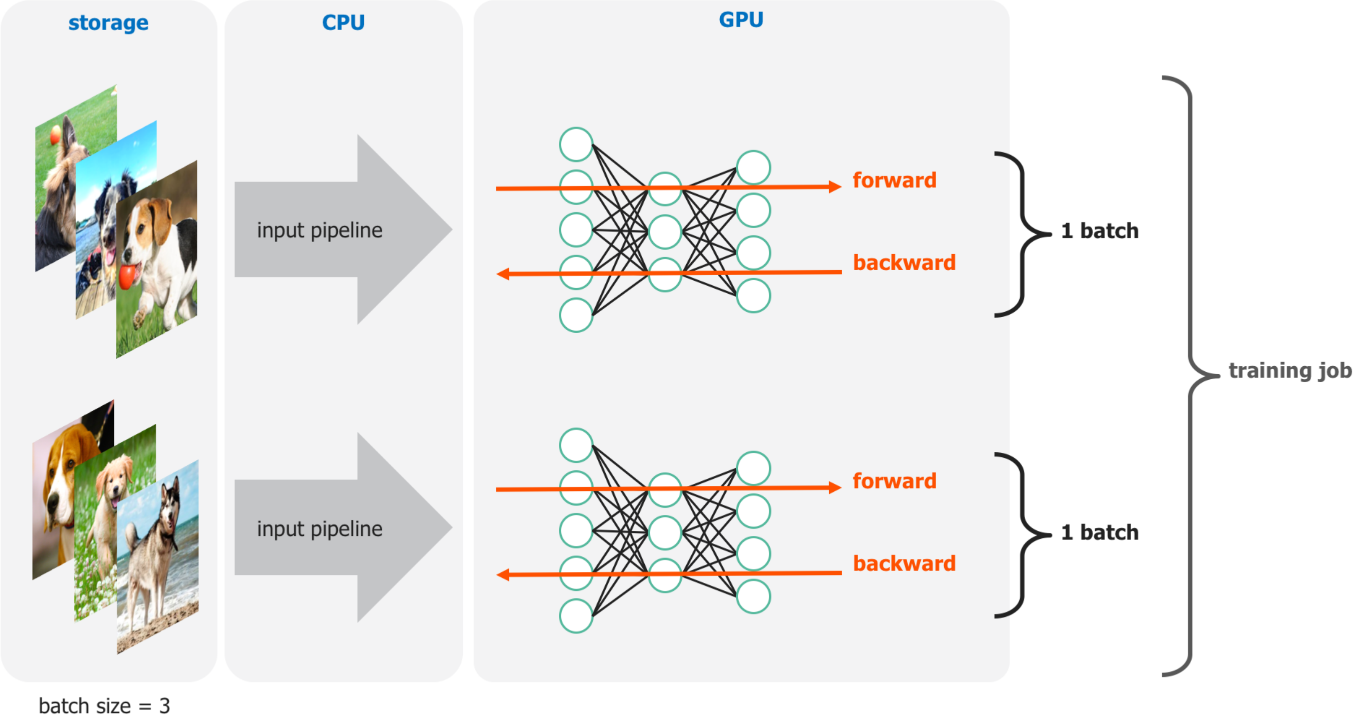
1. Overview of input pipelines & impact of data format.

2. How does data load time fit into overall training throughput?

3. Options for improving throughput based on training datasets.

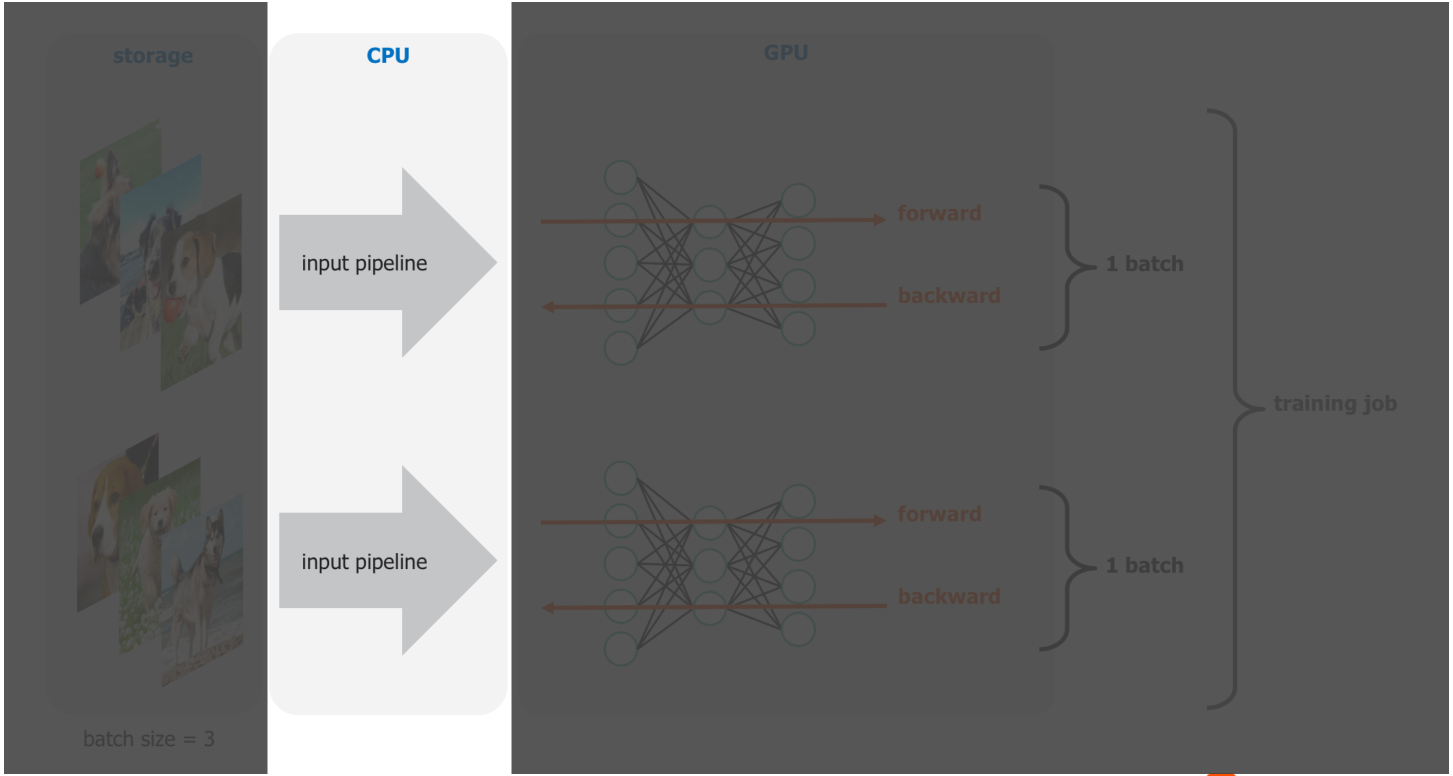
PURESTORAGE®

PATH OF A BATCH

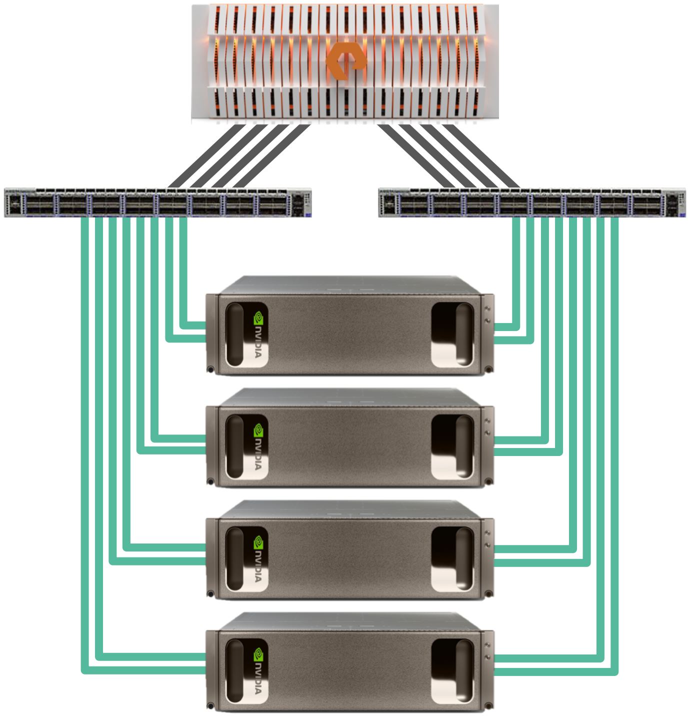


PURESTORAGE®

PATH OF A BATCH



PATH OF A BATCH



PURESTORAGE®

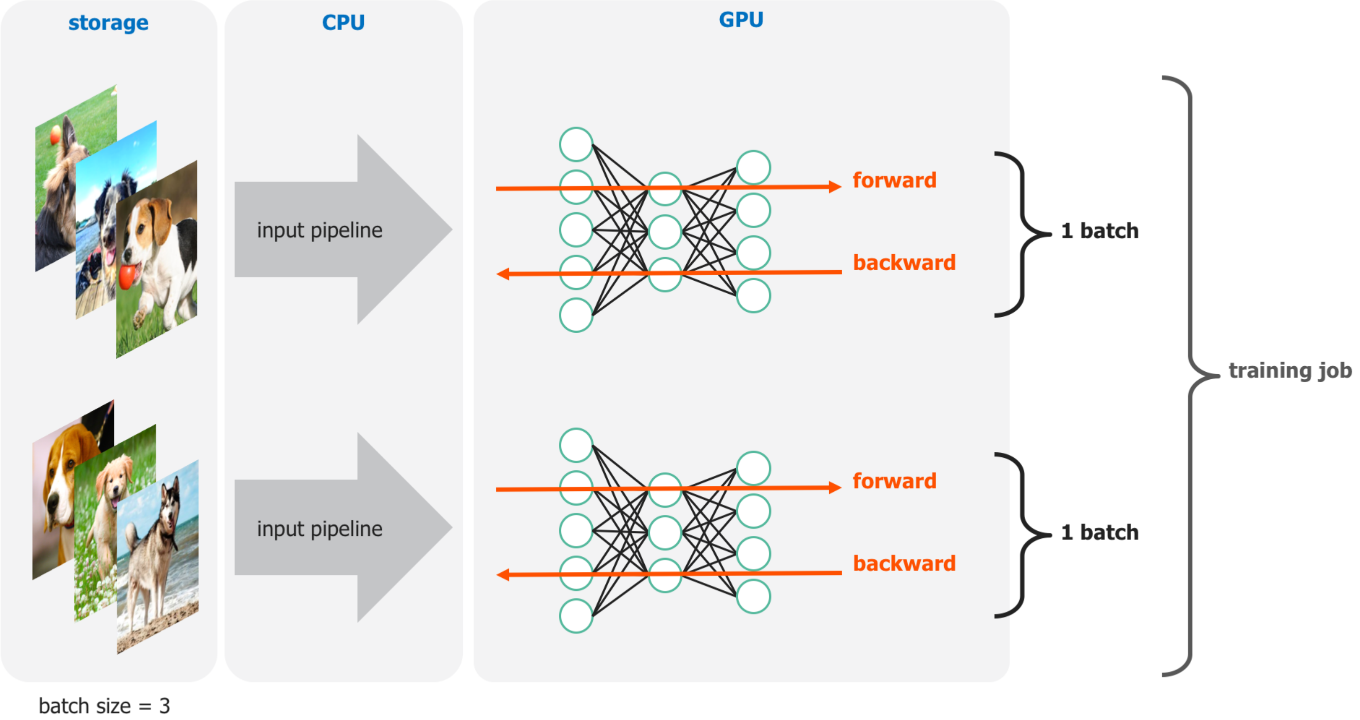
GPU UTILIZATION IS NOT EVERYTHING.

– NOT THE FULL PICTURE

– NOT GRANULAR ENOUGH

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 410.72       Driver Version: 410.72       CUDA Version: 10.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:06:00.0 Off |                    0 |
| N/A   48C    P0   296W / 300W |  15756MiB / 16130MiB |     90%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  Off  | 00000000:07:00.0 Off |                    0 |
| N/A   49C    P0   238W / 300W |  15756MiB / 16130MiB |     98%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-SXM2...  Off  | 00000000:0A:00.0 Off |                    0 |
| N/A   52C    P0   273W / 300W |  15756MiB / 16130MiB |     98%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla V100-SXM2...  Off  | 00000000:0B:00.0 Off |                    0 |
| N/A   47C    P0   260W / 300W |  15756MiB / 16130MiB |     98%      Default |
+-------------------------------+----------------------+----------------------+
|   4  Tesla V100-SXM2...  Off  | 00000000:85:00.0 Off |                    0 |
| N/A   53C    P0   283W / 300W |  15756MiB / 16130MiB |     98%      Default |
+-------------------------------+----------------------+----------------------+
|   5  Tesla V100-SXM2...  Off  | 00000000:86:00.0 Off |                    0 |
| N/A   49C    P0    82W / 300W |  15756MiB / 16130MiB |     91%      Default |
+-------------------------------+----------------------+----------------------+
|   6  Tesla V100-SXM2...  Off  | 00000000:89:00.0 Off |                    0 |
| N/A   51C    P0    78W / 300W |  15756MiB / 16130MiB |     99%      Default |
+-------------------------------+----------------------+----------------------+
|   7  Tesla V100-SXM2...  Off  | 00000000:8A:00.0 Off |                    0 |
| N/A   45C    P0    69W / 300W |  15756MiB / 16130MiB |     99%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0     21859      C   python                                     15745MiB |
|    1     21860      C   python                                     15745MiB |
|    2     21861      C   python                                     15745MiB |
|    3     21862      C   python                                     15745MiB |
|    4     21863      C   python                                     15745MiB |
|    5     21864      C   python                                     15745MiB |
|    6     21865      C   python                                     15745MiB |
|    7     21866      C   python                                     15745MiB |
+-----------------------------------------------------------------------------+
```

PATH OF A BATCH



PURESTORAGE®

BATCH TIMING

training logs: **batch, ms**

```
batch_results: 917, 1040
batch_results: 918,   43
batch_results: 919,   89
batch_results: 920,   29
batch_results: 921, 1025
batch_results: 922,   37
batch_results: 923,   23
batch_results: 924,   90
batch_results: 925, 1053
batch_results: 926,   31
batch_results: 927,   21
```

BATCH TIMING ODDITIES

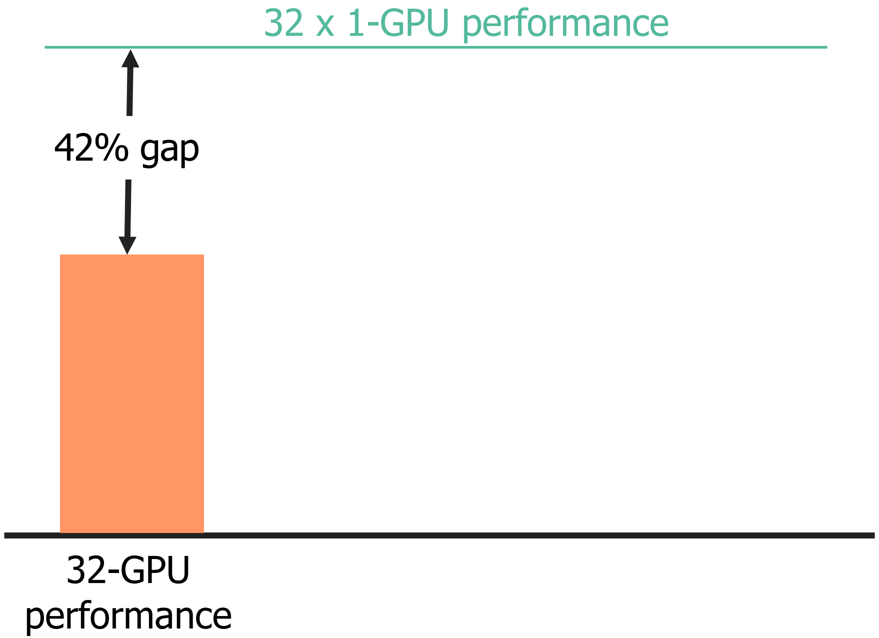
training logs: **batch, ms**

```
batch_results: 917, 1040
batch_results: 918,   43
batch_results: 919,   89
batch_results: 920,   29
batch_results: 921, 1025
batch_results: 922,   37
batch_results: 923,   23
batch_results: 924,   90
batch_results: 925, 1053
batch_results: 926,   31
batch_results: 927,   21
```


PURESTORAGE®

WHAT'S "GOOD" PERFORMANCE?

WHAT'S "GOOD" PERFORMANCE?

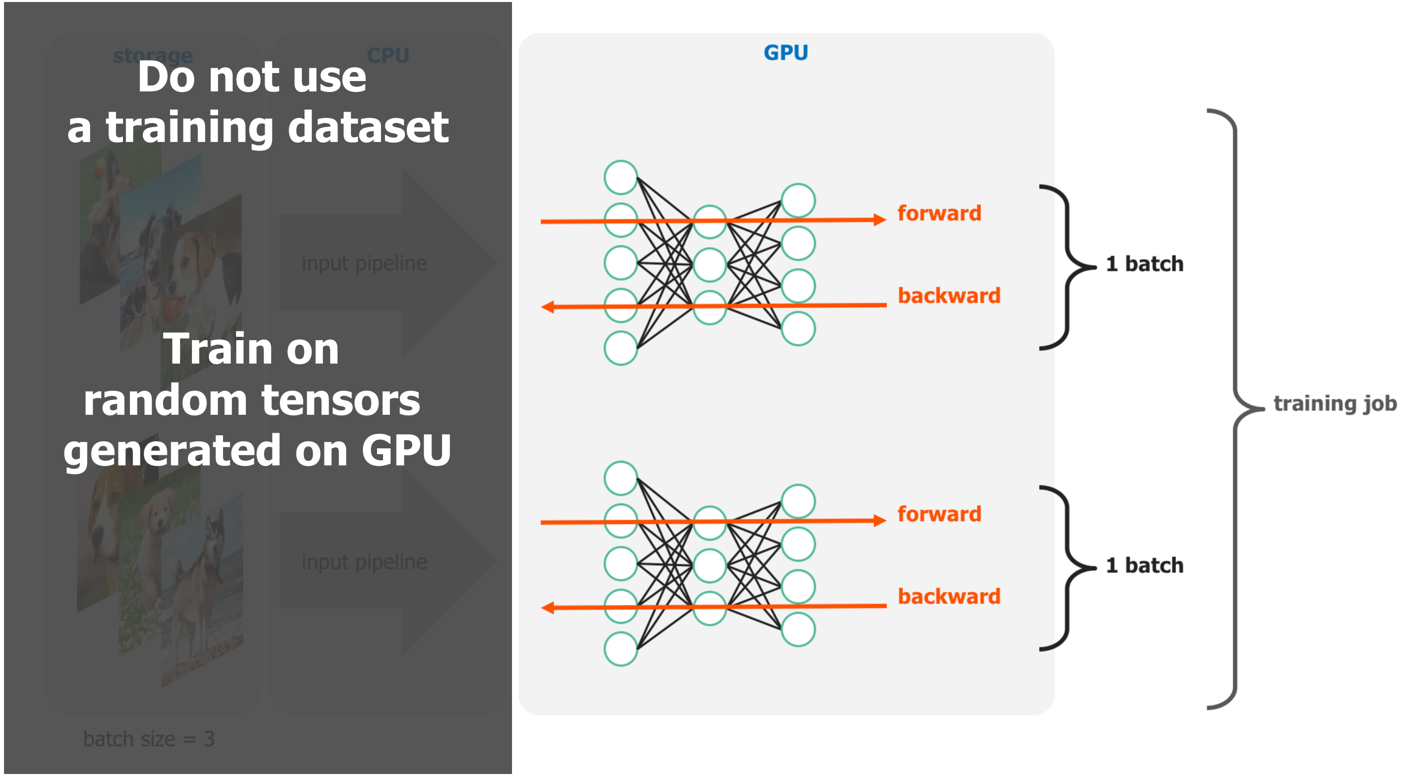


PURESTORAGE®

TESTING WITH SYNTHETIC DATA

Do not use a training dataset

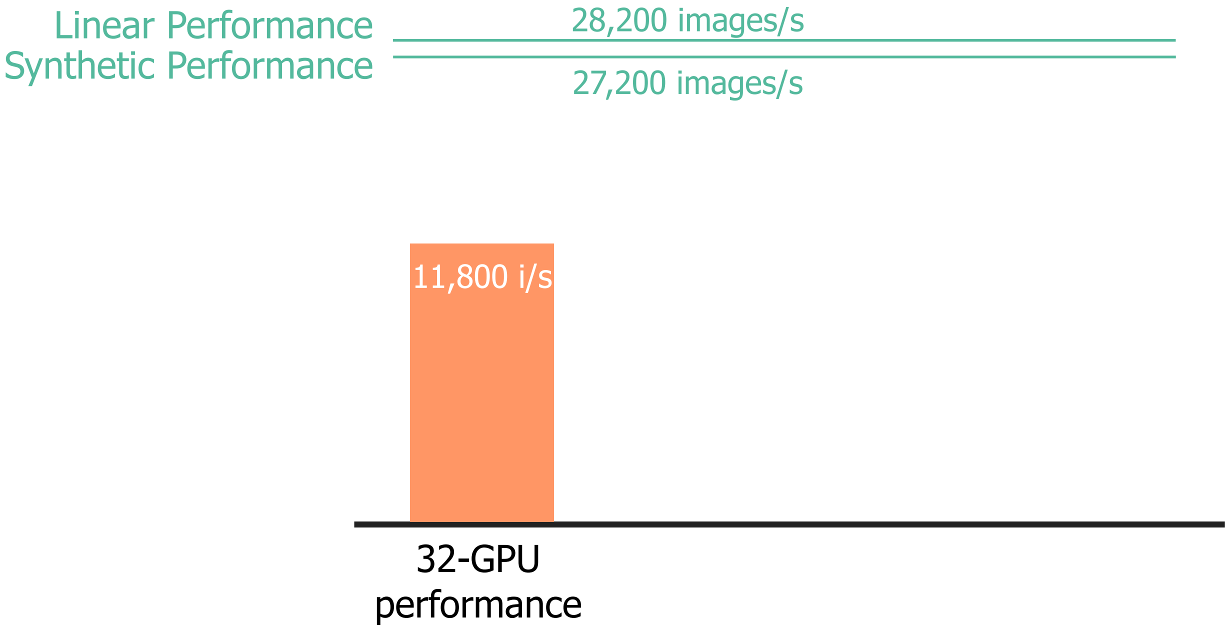
Train on random tensors generated on GPU



PURESTORAGE®

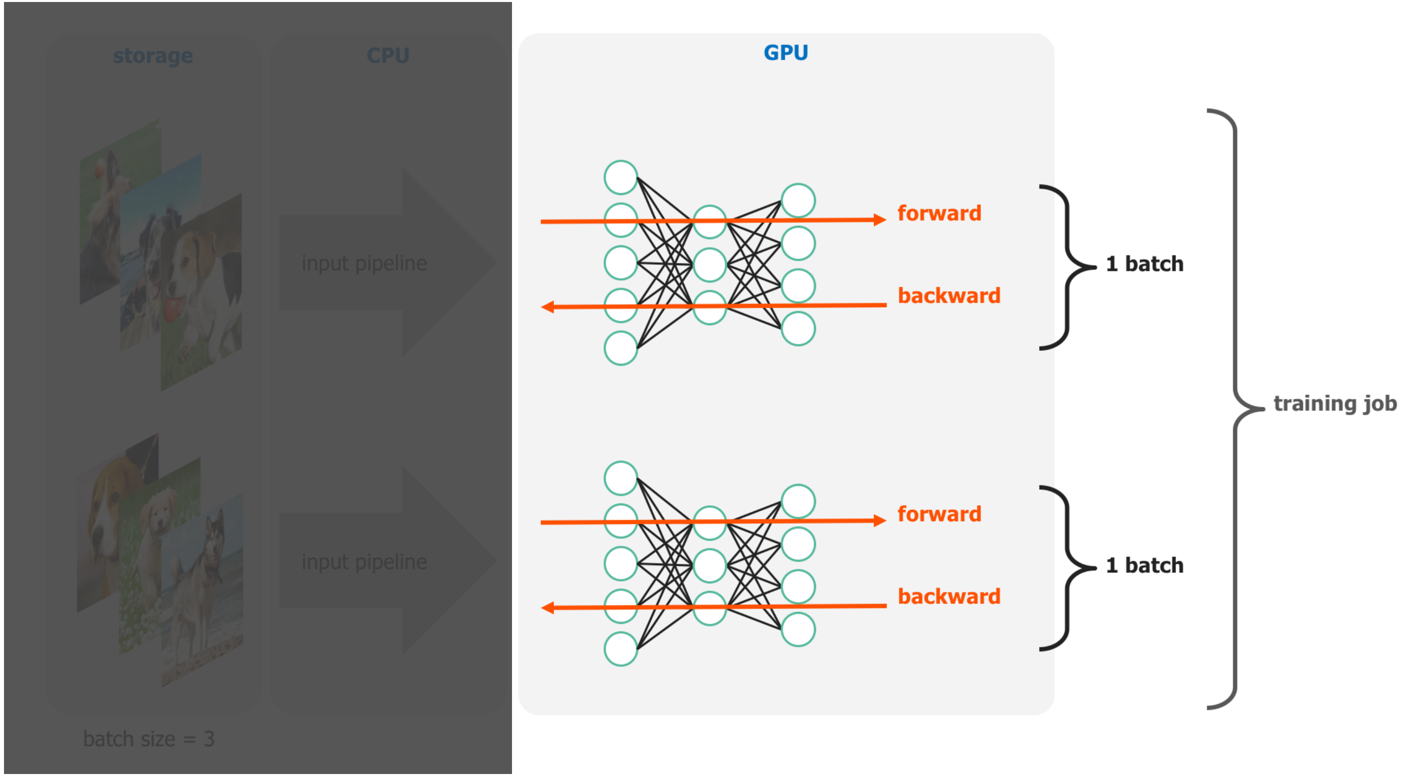
BREAK DOWN THE PROBLEM SPACE

REAL VS. SYNTHETIC DATA

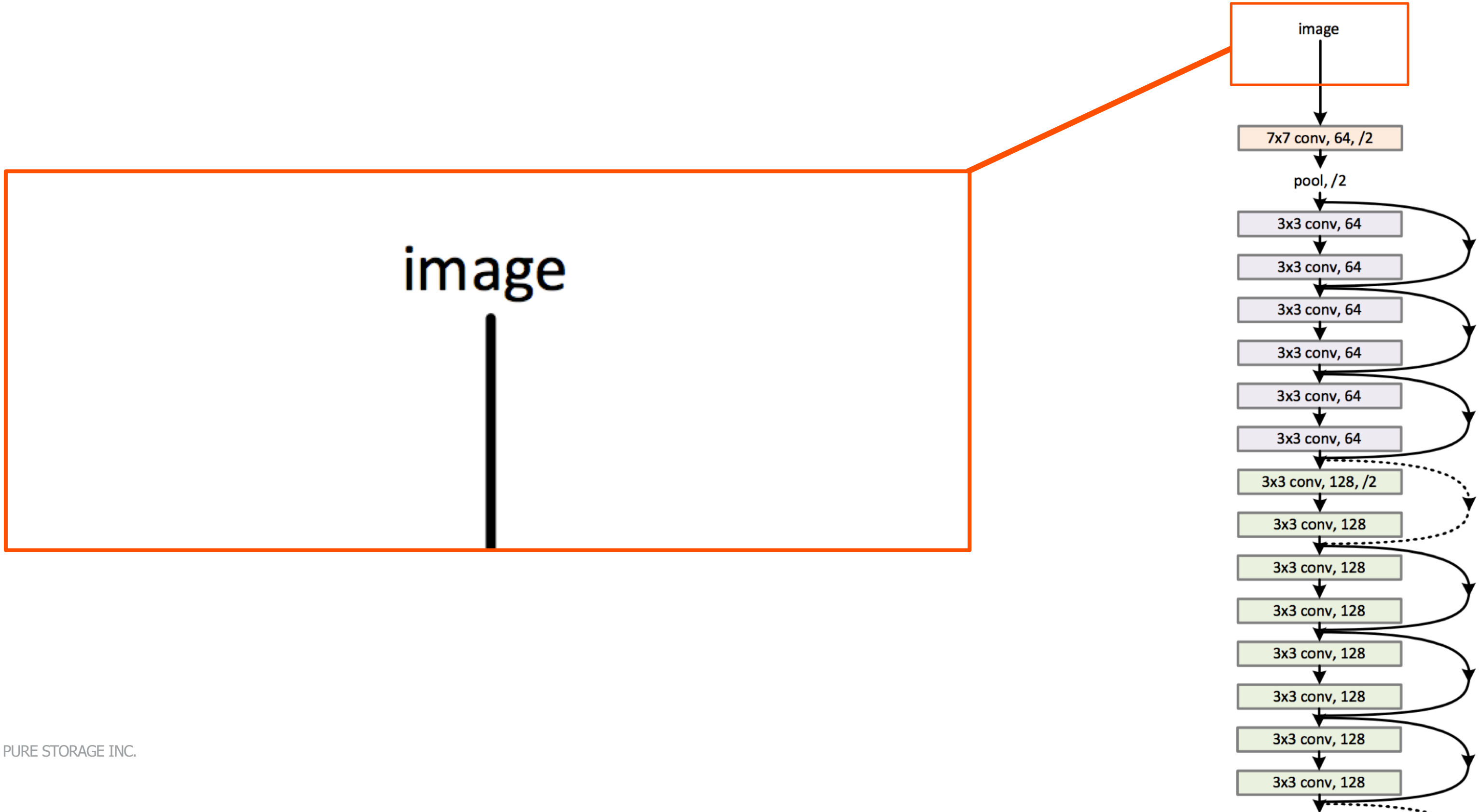


PURESTORAGE®

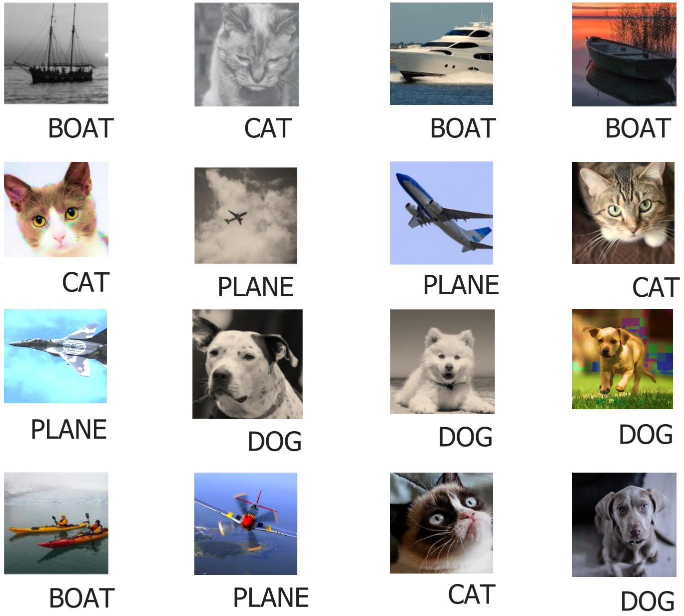
TESTING WITH SYNTHETIC DATA



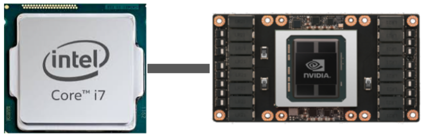
UNDER THE HOOD

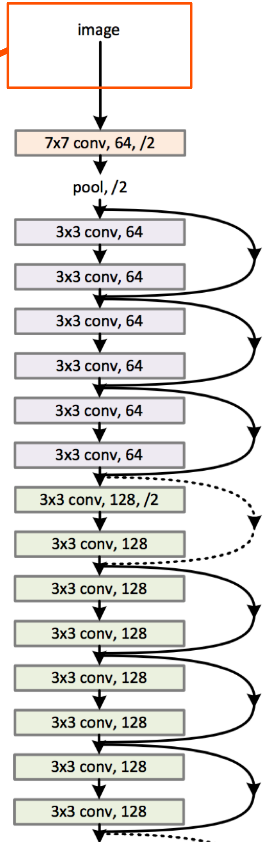


UNDER THE HOOD



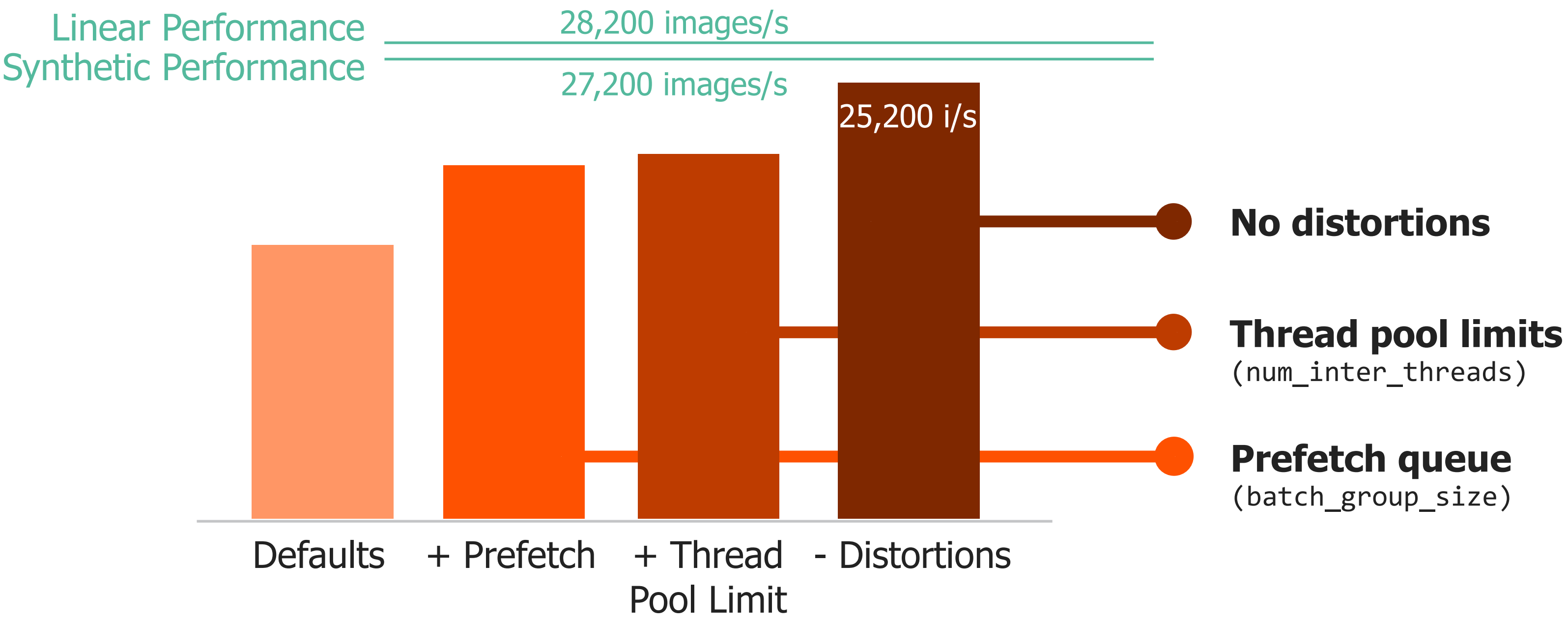
1. Enumerate
2. Associate labels
3. Shuffle
4. Read, crop, distort
5. Convert to tensor
6. Copy to GPU





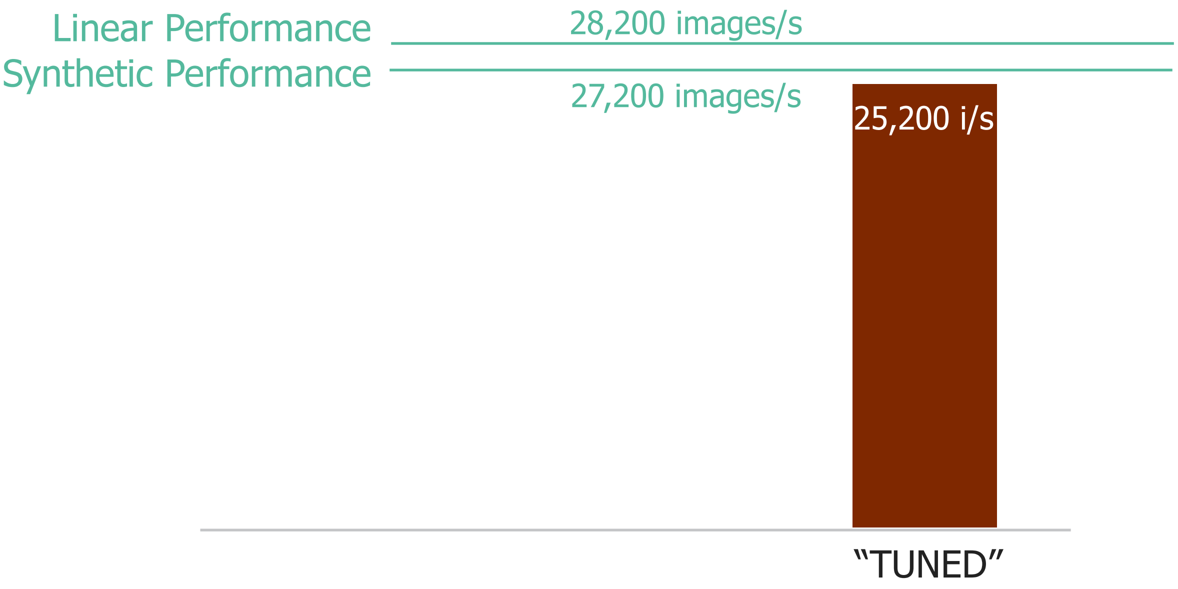
ISSUE IDENTIFIED: INPUT PIPELINE

IMAGENET PERFORMANCE BENCHMARK, RESNET-50



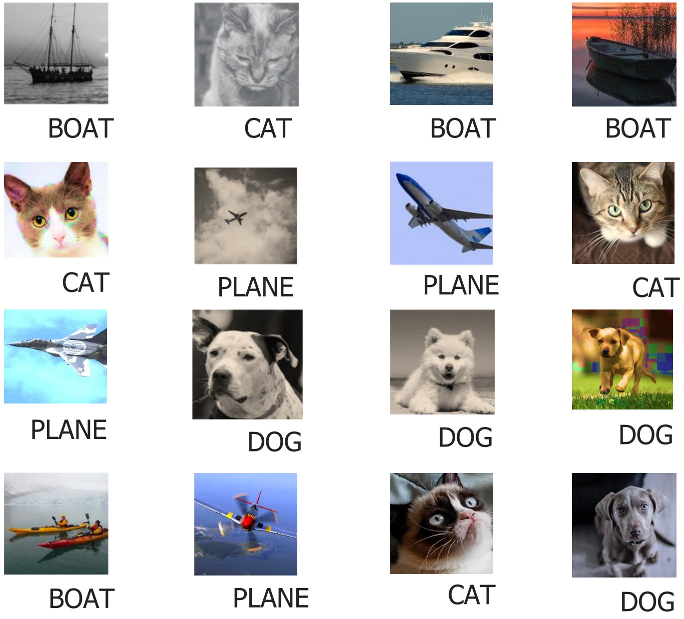
PURESTORAGE®

WE BALANCED THIS PIPELINE

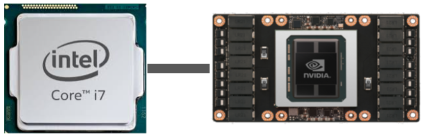


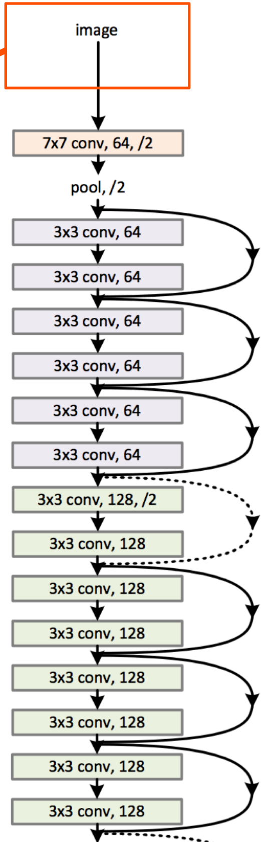
PURESTORAGE®

UNDER THE HOOD



1. Enumerate
2. Associate labels
3. Shuffle
4. Read, crop, distort
5. Convert to tensor
6. Copy to GPU





! DATASETS FOR BENCHMARK JOBS

```
tensor([[[ 0.5022,  0.6049,  0.7077,  ...,  0.7077,  0.6392,  0.7762],
         [ 0.4166,  0.6221,  0.6392,  ...,  0.9646,  0.9303,  0.9303],
         [ 0.4851,  0.6049,  0.5878,  ...,  1.4098,  1.3242,  1.2043],
         ...,
         [ 1.7694,  1.6324,  1.4612,  ...,  1.1015,  0.9988,  0.7762],
         [ 1.4098,  1.2214,  0.9988,  ...,  1.1015,  1.0844,  1.2728],
         [ 1.7865,  1.5297,  1.1529,  ...,  1.4612,  1.5468,  1.7694]],

        [[ 0.6879,  0.7925,  0.9145,  ...,  0.7751,  0.6705,  0.7751],
         [ 0.5136,  0.7402,  0.7402,  ...,  1.0714,  1.0017,  1.0017],
         [ 0.5834,  0.6879,  0.6705,  ...,  1.4374,  1.3502,  1.2805],
         ...,
         [ 2.0300,  1.8905,  1.7163,  ...,  1.1062,  1.0365,  0.8099],
         [ 1.4548,  1.2631,  1.0365,  ...,  1.1237,  1.1411,  1.2980],
         [ 1.6465,  1.3851,  1.0017,  ...,  1.4200,  1.5071,  1.6814]]])
```

stored JPEG → **LOAD** → **DECODE** → **CROP, DISTORT** → **TENSOR READY FOR GPU**

PURESTORAGE®

! DATASETS FOR BENCHMARK JOBS

```
tensor([[[ 0.5022,  0.6049,  0.7077,  ...,  0.7077,  0.6392,  0.7762],
         [ 0.4166,  0.6221,  0.6392,  ...,  0.9646,  0.9303,  0.9303],
         [ 0.4851,  0.6049,  0.5878,  ...,  1.4098,  1.3242,  1.2043],
         ...,
         [ 1.7694,  1.6324,  1.4612,  ...,  1.1015,  0.9988,  0.7762],
         [ 1.4098,  1.2214,  0.9988,  ...,  1.1015,  1.0844,  1.2728],
         [ 1.7865,  1.5297,  1.1529,  ...,  1.4612,  1.5468,  1.7694]],

        [[ 0.6879,  0.7925,  0.9145,  ...,  0.7751,  0.6705,  0.7751],
         [ 0.5136,  0.7402,  0.7402,  ...,  1.0714,  1.0017,  1.0017],
         [ 0.5834,  0.6879,  0.6705,  ...,  1.4374,  1.3502,  1.2805],
         ...,
         [ 2.0300,  1.8905,  1.7163,  ...,  1.1062,  1.0365,  0.8099],
         [ 1.4548,  1.2631,  1.0365,  ...,  1.1237,  1.1411,  1.2980],
         [ 1.6465,  1.3851,  1.0017,  ...,  1.4200,  1.5071,  1.6814]]])
```

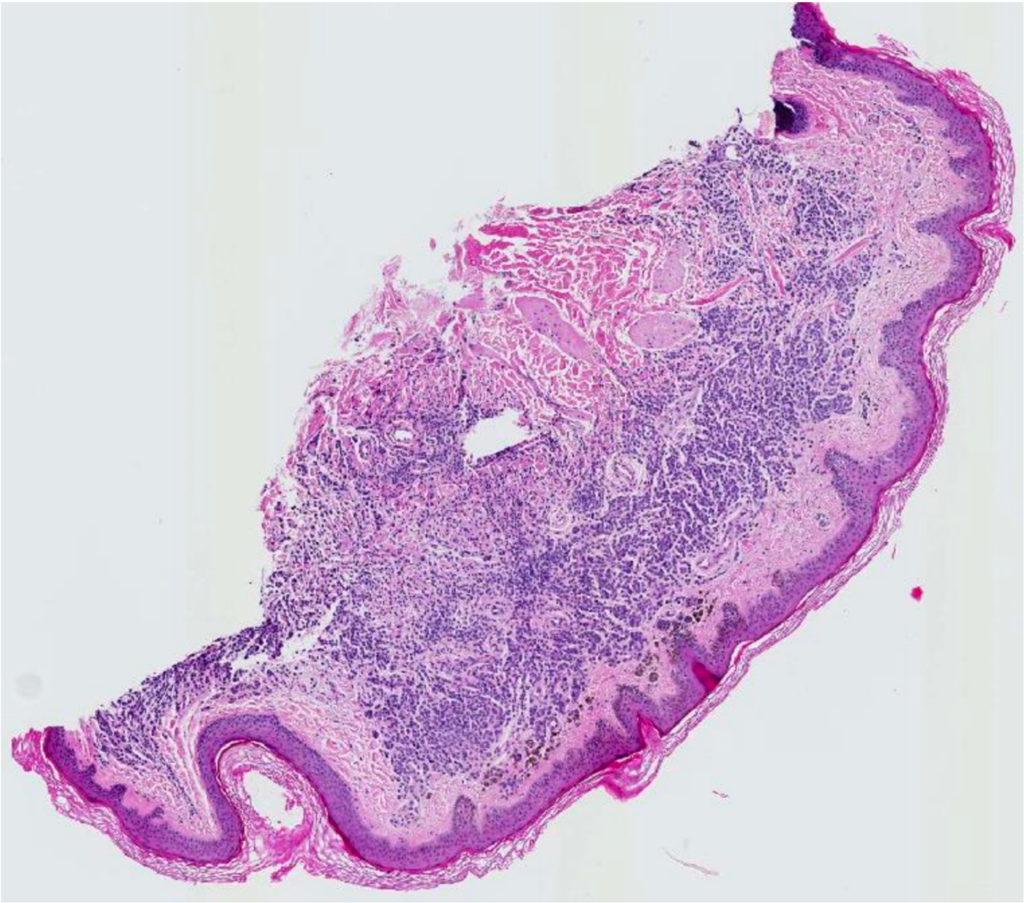
stored JPEG → LOAD → DECODE → CROP, DISTORT → stored tensor

PURESTORAGE®

TENSORS: LIMITED PROCESSING

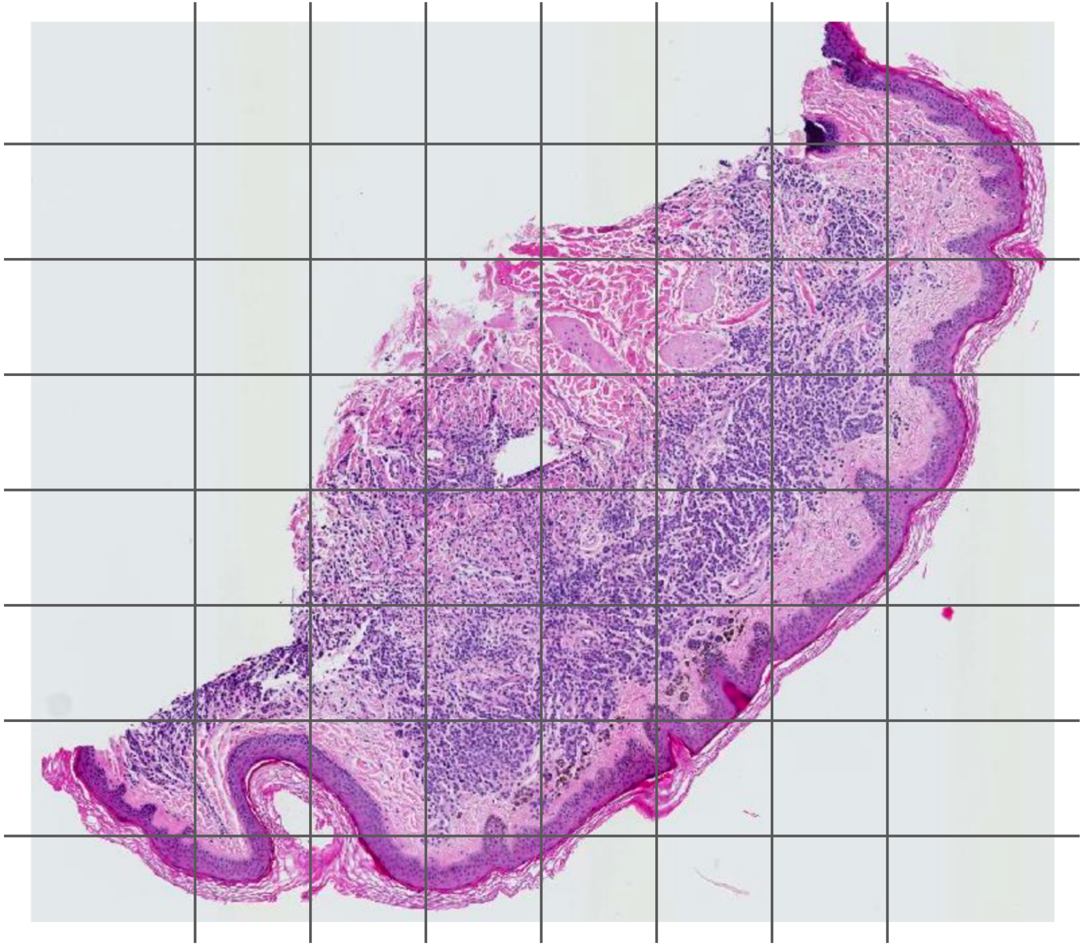
```
tensor([[[ 0.5022,  0.6049,  0.7077,  ...,  0.7077,  0.6392,  0.7762],
         [ 0.4166,  0.6221,  0.6392,  ...,  0.9646,  0.9303,  0.9303],
         [ 0.4851,  0.6049,  0.5878,  ...,  1.4098,  1.3242,  1.2043],
         ...,
         [ 1.7694,  1.6324,  1.4612,  ...,  1.1015,  0.9988,  0.7762],
         [ 1.4098,  1.2214,  0.9988,  ...,  1.1015,  1.0844,  1.2728],
         [ 1.7865,  1.5297,  1.1529,  ...,  1.4612,  1.5468,  1.7694]],

        [[-0.2675, -0.1099,  0.0651,  ...,  0.2227,  0.1352,  0.2402],
         [-0.4076, -0.1625, -0.0749,  ...,  0.5203,  0.4328,  0.4153],
         [-0.3550, -0.2150, -0.1625,  ...,  0.8880,  0.8004,  0.7129],
         ...,
         [ 1.3431,  1.1506,  0.9580,  ...,  0.2577,  0.1527, -0.0749],
         [ 0.7829,  0.5553,  0.3277,  ...,  0.3277,  0.3102,  0.5028],
         [ 1.0455,  0.7479,  0.3803,  ...,  0.7654,  0.8529,  1.0630]],

        [[ 0.6879,  0.7925,  0.9145,  ...,  0.7751,  0.6705,  0.7751],
         [ 0.5136,  0.7402,  0.7402,  ...,  1.0714,  1.0017,  1.0017],
         [ 0.5834,  0.6879,  0.6705,  ...,  1.4374,  1.3502,  1.2805],
         ...,
         [ 2.0300,  1.8905,  1.7163,  ...,  1.1062,  1.0365,  0.8099],
         [ 1.4548,  1.2631,  1.0365,  ...,  1.1237,  1.1411,  1.2980],
         [ 1.6465,  1.3851,  1.0017,  ...,  1.4200,  1.5071,  1.6814]]])
```


PURESTORAGE®



PAIGE

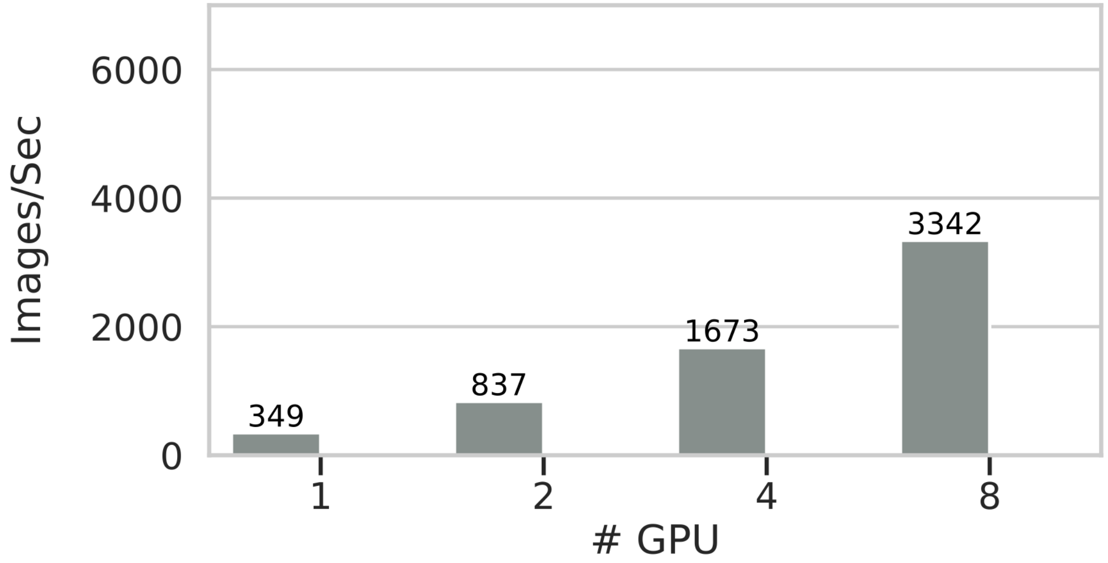
PURESTORAGE



PAIGE

PURESTORAGE®

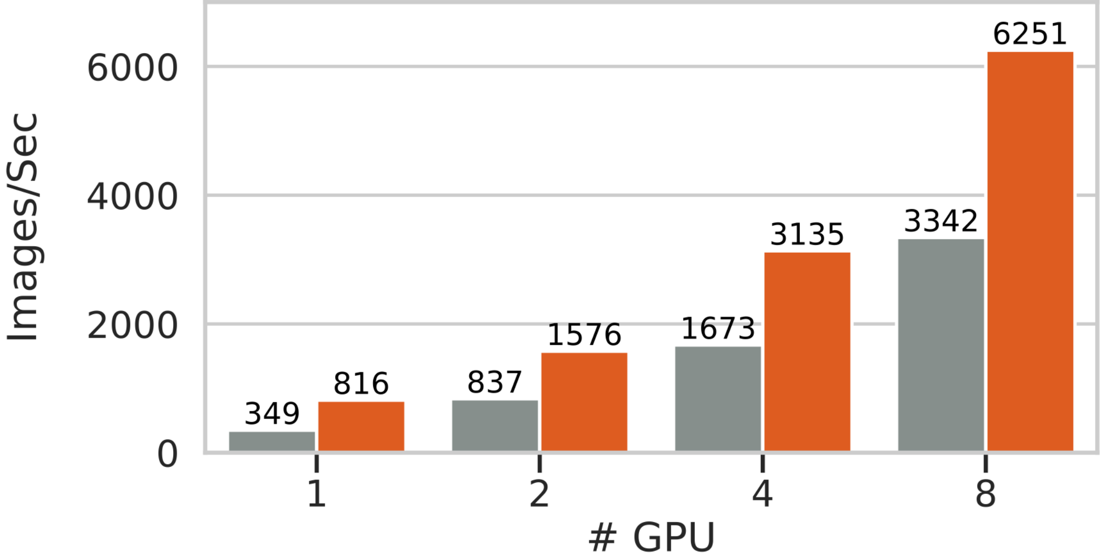
PNG IMAGES - NICE LINEAR SCALE



PURESTORAGE®

PNG IMAGES – 50% THROUGHPUT

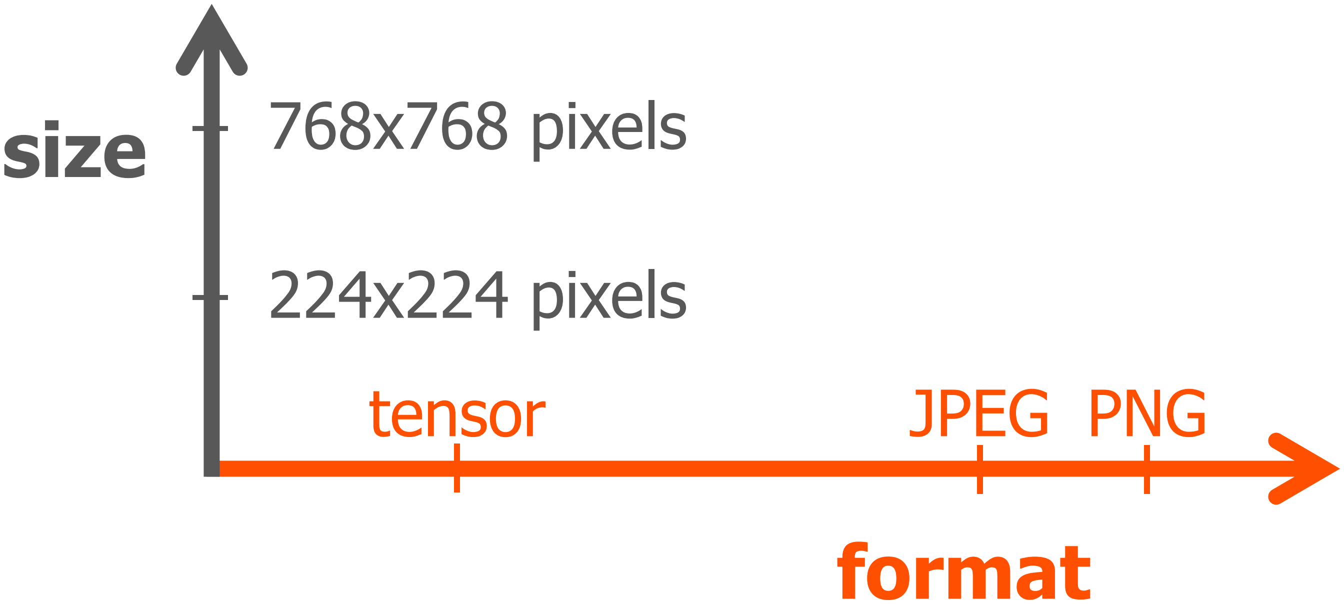
EVEN WHEN "TUNED"



PNG images: **utilizes data pipeline to load & transform**

SYNTHETIC: **data generated on GPUs**

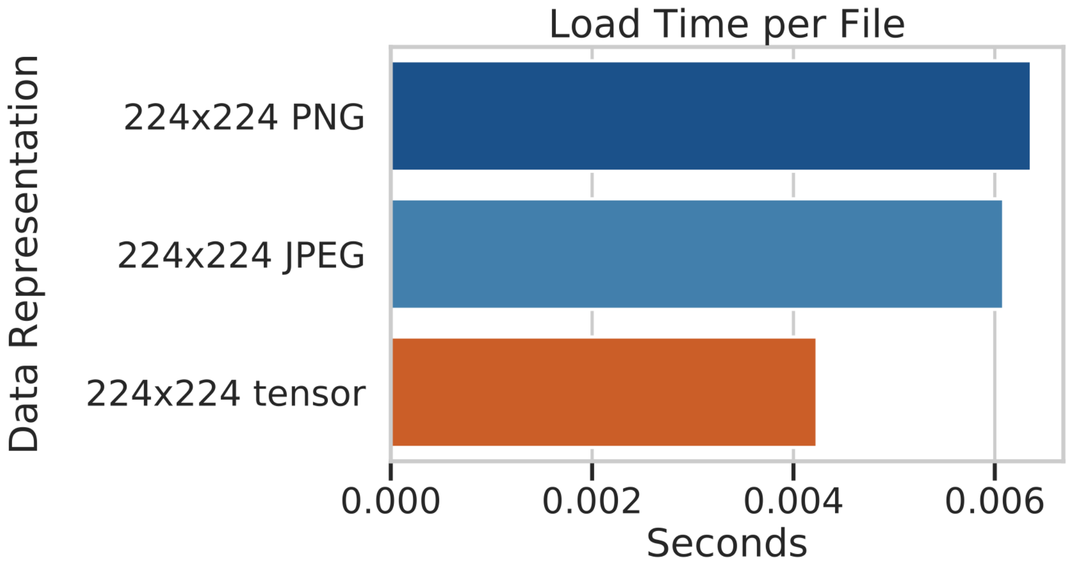
IMAGE REPRESENTATION



PURESTORAGE®

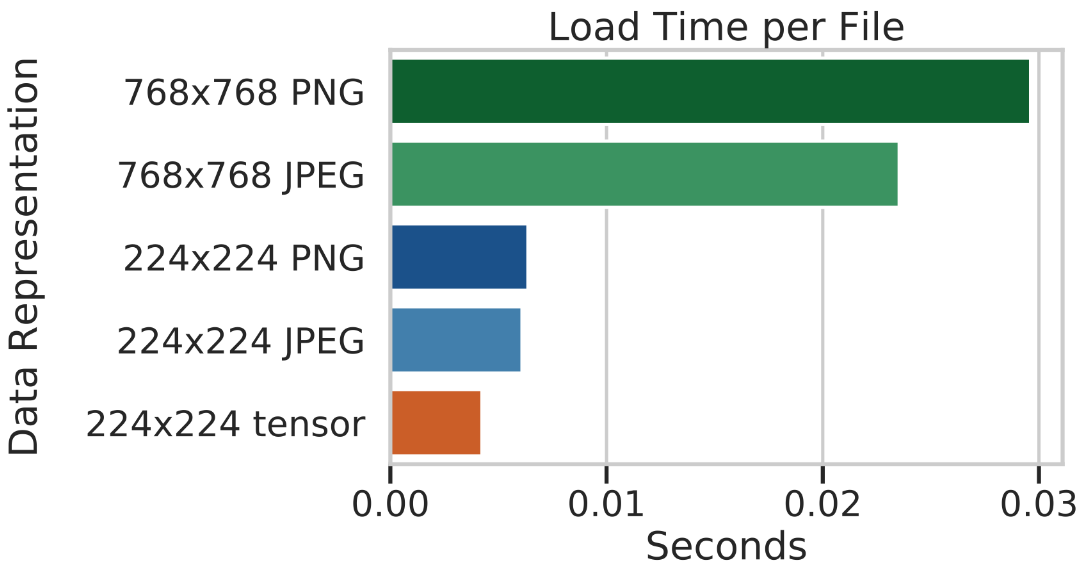
IMAGE FORMAT

format



Higher image complexity = Longer load time

IMAGE FORMAT & IMAGE SIZE



size

Larger image size = Longer load time

PURESTORAGE®

Impact of data loading

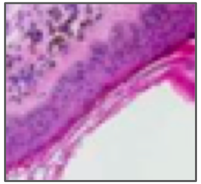
1. Overview of input pipelines & impact of data format.
2. How does data load time fit into overall training throughput?
3. Options for improving throughput based on training datasets.

LIFE OF A BATCH

```
10.png
11.png
12.png
...
4998.png
4999.png
5000.png
```

```
1291.png
4994.png
4574.png
...
3621.png
57.png
280.png
```

INDEX | **LOAD** | **FORWARD** | **BACKWARD**



```
tensor([[[ 0.5022,  0.6049,  0.7077,  ...,  0.7077,  0.6392,  0.7762],
         [ 0.4166,  0.6221,  0.6392,  ...,  0.9646,  0.9303,  0.9303],
         [ 0.4851,  0.6049,  0.5878,  ...,  1.4098,  1.3242,  1.2043],
         ...,
         [ 1.7694,  1.6324,  1.4612,  ...,  1.1015,  0.9988,  0.7762],
         [ 1.4098,  1.2214,  0.9988,  ...,  1.1015,  1.0844,  1.2728],
         [ 1.7865,  1.5297,  1.1529,  ...,  1.4612,  1.5468,  1.7694]],


        [[-0.2675, -0.1099,  0.0651,  ...,  0.2227,  0.1352,  0.2402],
         [-0.4076, -0.1625, -0.0749,  ...,  0.5203,  0.4328,  0.4153],
         [-0.3550, -0.2150, -0.1625,  ...,  0.8880,  0.8004,  0.7129],
         ...,
         [ 1.3431,  1.1506,  0.9580,  ...,  0.2577,  0.1527, -0.0749],
         [ 0.7829,  0.5553,  0.3277,  ...,  0.3277,  0.3102,  0.5028],
         [ 1.0455,  0.7479,  0.3803,  ...,  0.7654,  0.8529,  1.0630]],


        [[ 0.6879,  0.7925,  0.9145,  ...,  0.7751,  0.6705,  0.7751],
         [ 0.5136,  0.7402,  0.7402,  ...,  1.0714,  1.0017,  1.0017],
         [ 0.5834,  0.6879,  0.6705,  ...,  1.4374,  1.3502,  1.2805],
         ...,
         [ 2.0300,  1.8905,  1.7163,  ...,  1.1062,  1.0365,  0.8099],
         [ 1.4548,  1.2631,  1.0365,  ...,  1.1237,  1.1411,  1.2980],
         [ 1.6465,  1.3851,  1.0017,  ...,  1.4200,  1.5071,  1.6814]]])
```

LIFE OF A BATCH

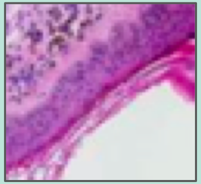
CPU
Main process

```
10.png          1291.png
11.png          4994.png
12.png          4574.png
...             ...
4998.png        3621.png
4999.png        57.png
5000.png        280.png
```

GPU GPU

INDEX → **LOAD** → **FORWARD** → **BACKWARD**

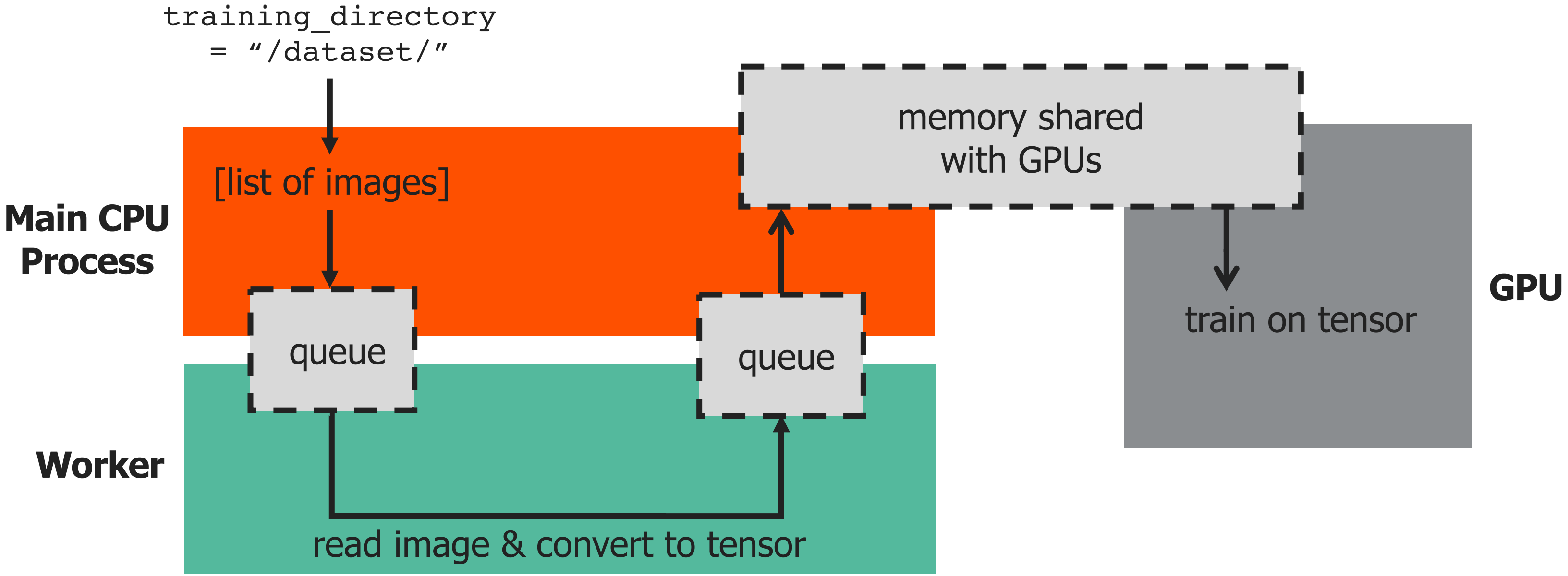
CPU
worker processes



```
tensor([[[ 0.5022,  0.6049,  0.7077,  ...,  0.7077,  0.6392,  0.7762],
         [ 0.4166,  0.6221,  0.6392,  ...,  0.9646,  0.9303,  0.9303],
         [ 0.4851,  0.6049,  0.5878,  ...,  1.4098,  1.3242,  1.2043],
         ...,
         [ 1.7694,  1.6324,  1.4612,  ...,  1.1015,  0.9988,  0.7762],
         [ 1.4098,  1.2214,  0.9988,  ...,  1.1015,  1.0844,  1.2728],
         [ 1.7865,  1.5297,  1.1529,  ...,  1.4612,  1.5468,  1.7694]],

        [[-0.2675, -0.1099,  0.0651,  ...,  0.2227,  0.1352,  0.2402],
         [-0.4076, -0.1625, -0.0749,  ...,  0.5203,  0.4328,  0.4153],
         [-0.3550, -0.2150, -0.1625,  ...,  0.8880,  0.8004,  0.7129],
         ...,
         [ 1.3431,  1.1506,  0.9580,  ...,  0.2577,  0.1527, -0.0749],
         [ 0.7829,  0.5553,  0.3277,  ...,  0.3277,  0.3102,  0.5028],
         [ 1.0455,  0.7479,  0.3803,  ...,  0.7654,  0.8529,  1.0630]],

        [[ 0.6879,  0.7925,  0.9145,  ...,  0.7751,  0.6705,  0.7751],
         [ 0.5136,  0.7402,  0.7402,  ...,  1.0714,  1.0017,  1.0017],
         [ 0.5834,  0.6879,  0.6705,  ...,  1.4374,  1.3502,  1.2805],
         ...,
         [ 2.0300,  1.8905,  1.7163,  ...,  1.1062,  1.0365,  0.8099],
         [ 1.4548,  1.2631,  1.0365,  ...,  1.1237,  1.1411,  1.2980],
         [ 1.6465,  1.3851,  1.0017,  ...,  1.4200,  1.5071,  1.6814]]])
```


PURESTORAGE®

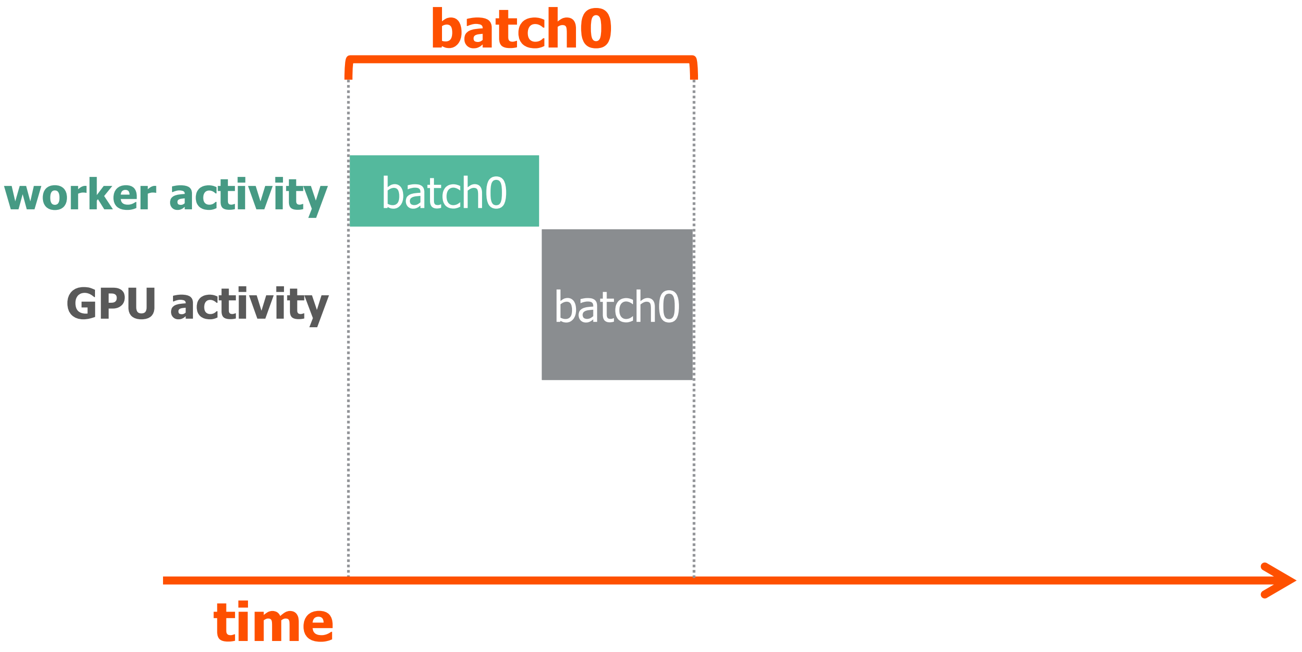
LIFE OF A BATCH



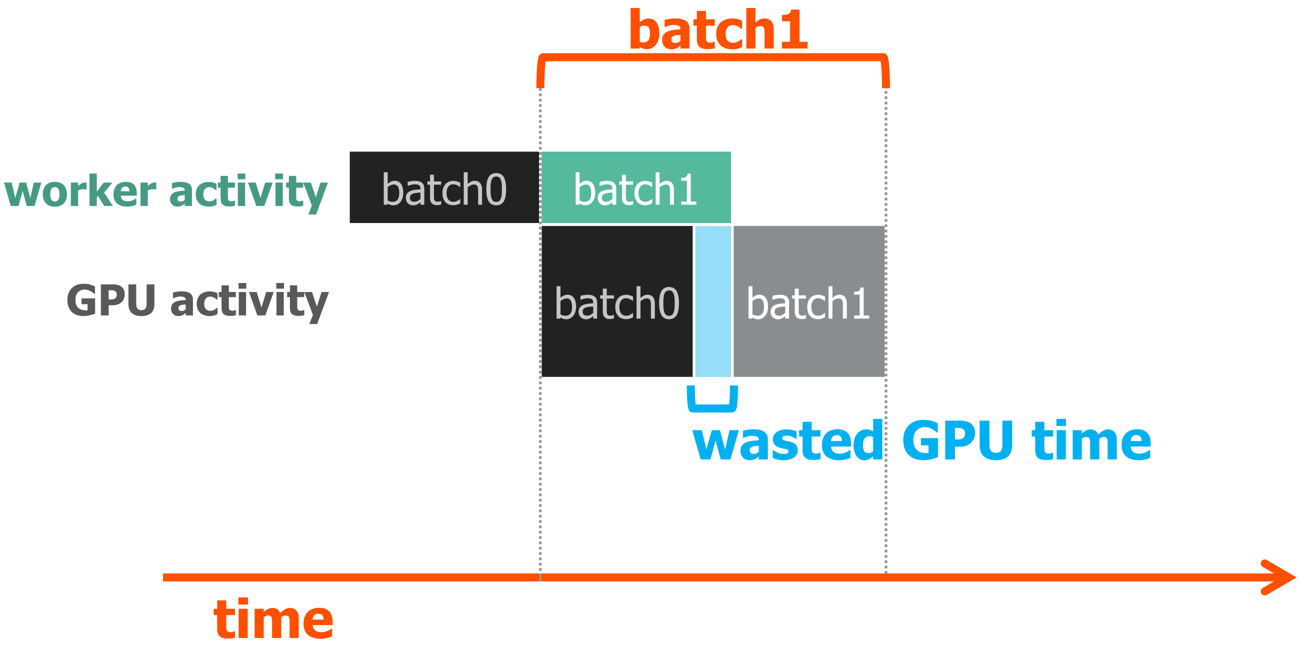
PURESTORAGE®

WORKER DEPENDENCY: BATCH 0

1 CPU worker, 1 GPU

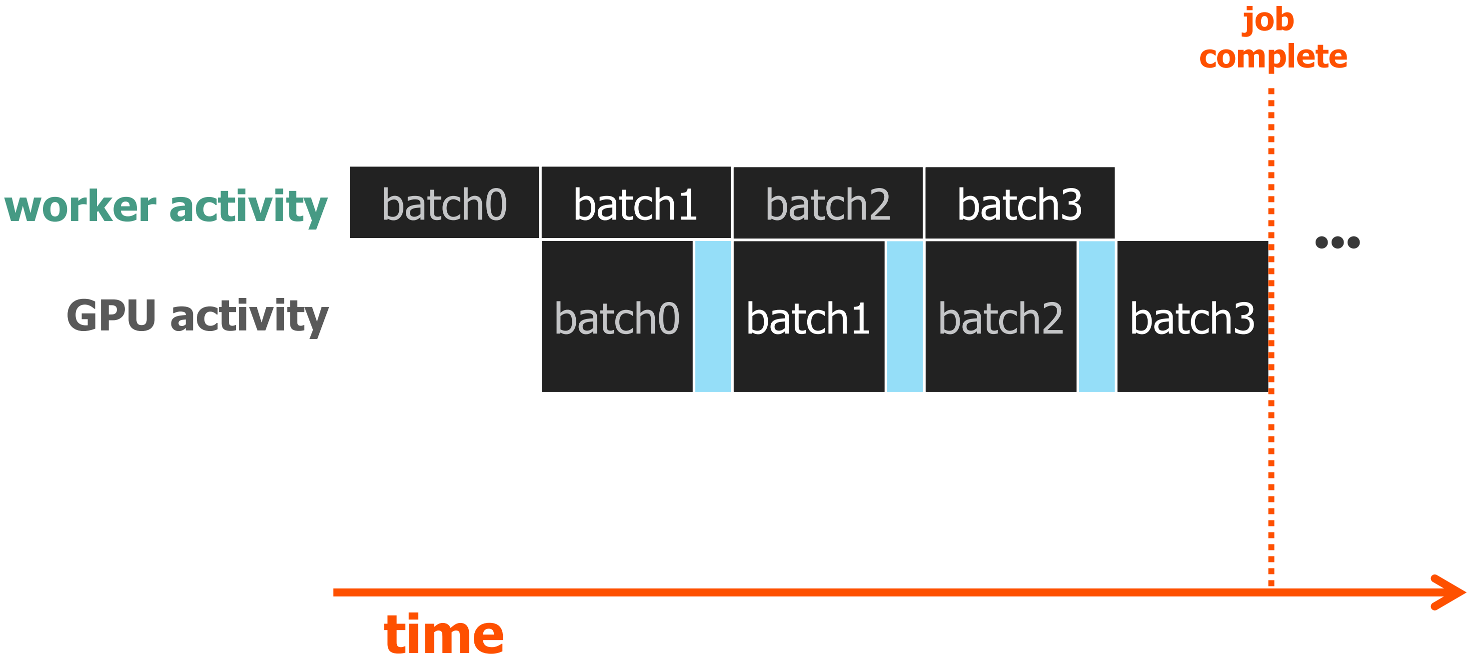


WORKER DEPENDENCY: BATCH 1



PURESTORAGE®

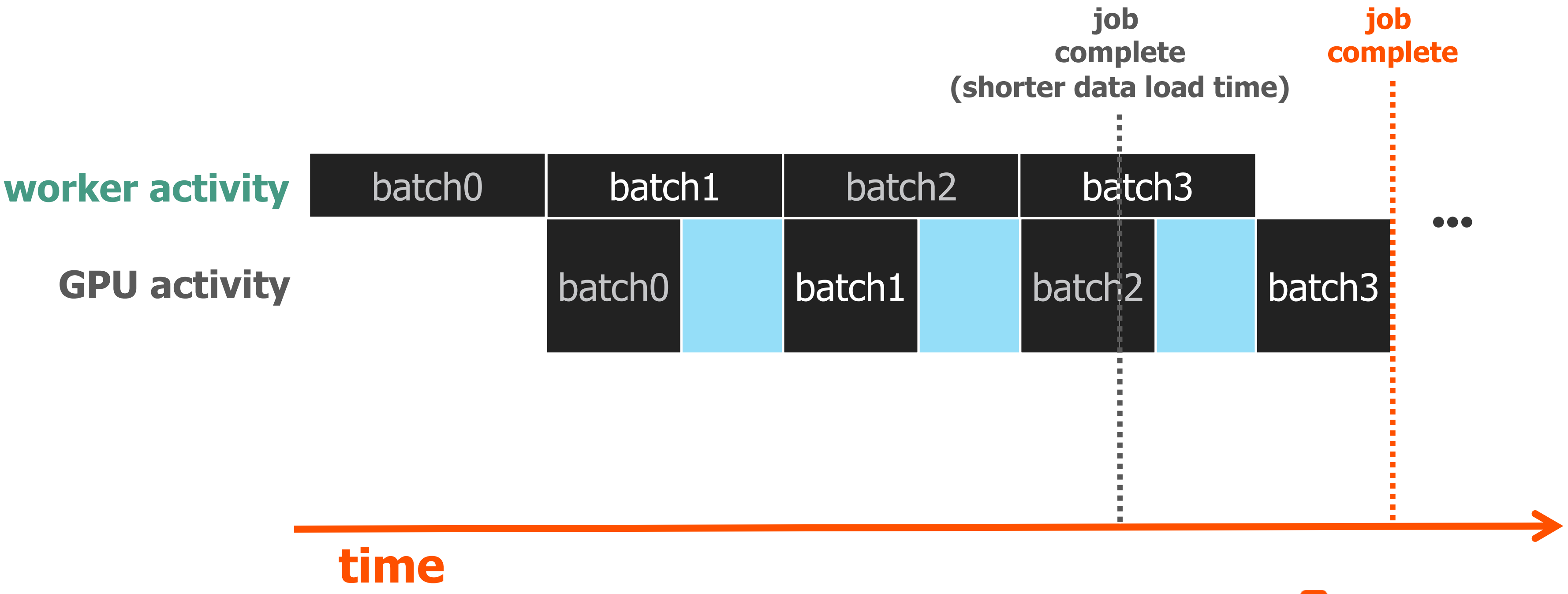
WASTED TIME ADDS UP OVER EPOCHS



PURESTORAGE®

WASTED TIME ADDS UP OVER EPOCHS,

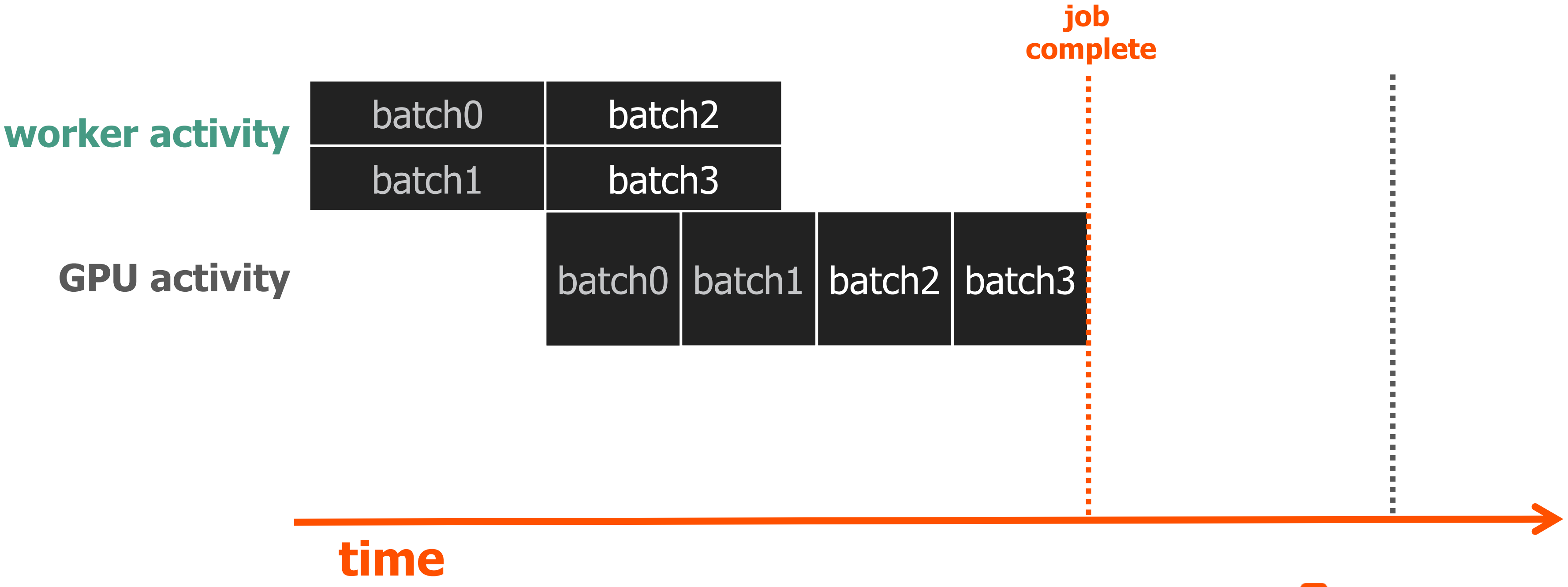
ESPECIALLY WITH LARGER DATA LOAD TIME



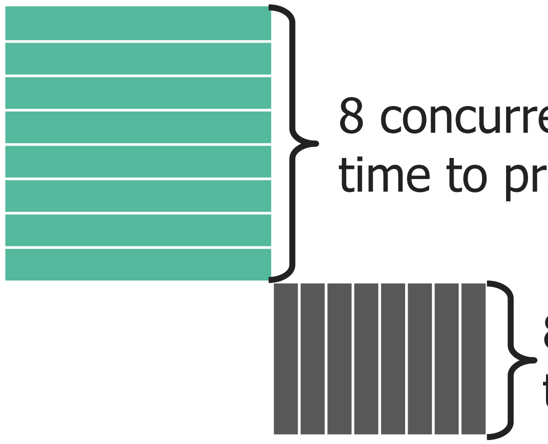
PURESTORAGE®

CONCURRENT WORKERS

SINGLE GPU



EXAMPLE WITH A REAL RATIO



worker activity

8 concurrent workers,
time to process **1 batch = 1.0 sec**

GPU activity

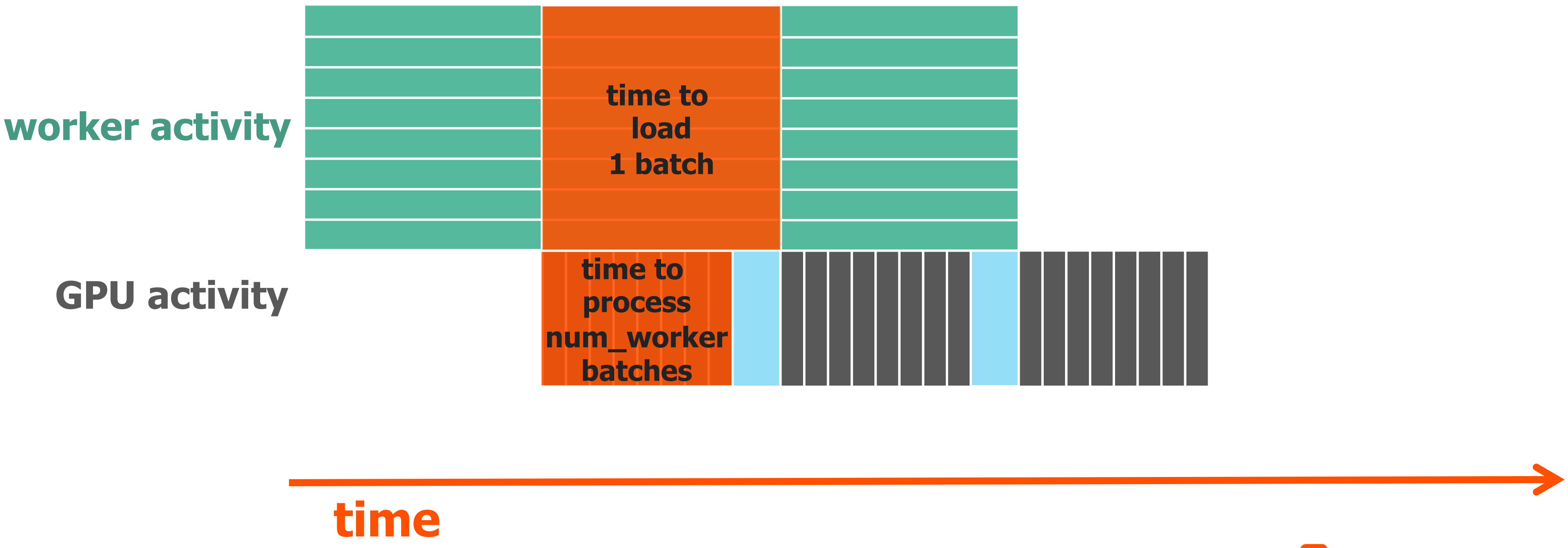
8 batches processed by GPU(s),
time to process **1 batch = 0.1 sec**

time

PURESTORAGE®

EXAMPLE WITH A REAL RATIO

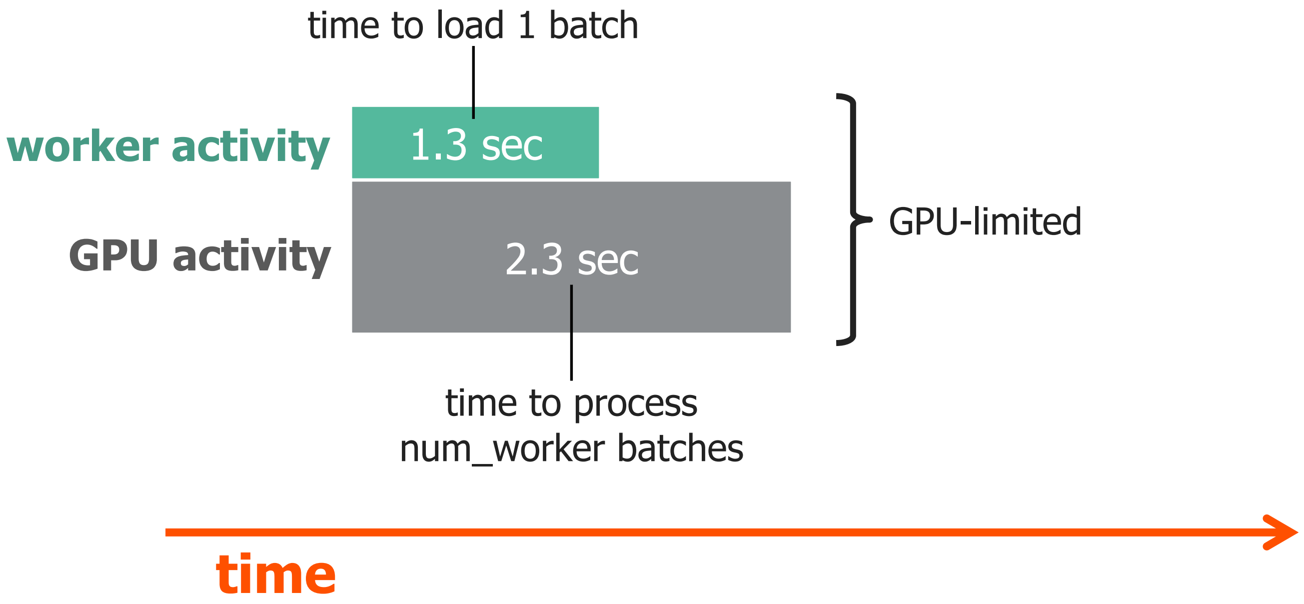
MULTI-GPU



PURESTORAGE®

IMAGE LOAD TIME V. GPU TIME

224x224 TENSOR, 8 GPUs, 16 WORKERS



PURESTORAGE®

IMAGE LOAD TIME V. GPU TIME

768x768 PNG, 8 GPUs, 16 WORKERS

PURESTORAGE®

IMAGE LOAD TIME V. GPU TIME

768x768 JPEG, 8 GPUs, 16 WORKERS

Impact of data loading

1. Overview of input pipelines & impact of data format.

2. How does data load time fit into overall training throughput?

3. Options for improving throughput based on training datasets.

Optimization Suggestions

1. **Precaching:** saving a pre-transformed tensor version of your dataset can minimize data load time during training (at the cost of on-the-fly distortions).

2. **Workload updates:** some workflows were designed around non-GPU compute environments. Investigate utilizing newer image loading libraries.

3. **Data load on GPU:** Nvidia DALI can be used to offload CPU work to GPUs, especially if workflows have been updated to use GPU-friendly data loading libraries.

Conclusions

1. Data format can significantly impact training throughput.

2. There is no one-size-fits-all input pipeline.

3. It’s critical to have have a mental model for your input pipeline and a methodology for testing its performance.

PURESTORAGE®

QUESTIONS

@dataemilyw

PURESTORAGE®