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

- data subsetting
- data validation
DEEP LEARNING EXECUTION FLOW
IN REAL LIFE

- hyperparameters
- model exploration
DEEP LEARNING EXECUTION FLOW IN REAL LIFE

EXPLORE → TRAIN → QA → DEPLOY

- hyperparameters
- model exploration
DEEP LEARNING EXECUTION FLOW
IN REAL LIFE

EXPLORE -> IMAGE LOAD -> TRAIN -> QA -> DEPLOY
1 TRAINING ITERATION

INDEX  LOAD  FORWARD  BACKWARD
1 TRAINING ITERATION

INDEX  LOAD  FWD  BKWD
DATASETS IMPACT THROUGHPUT

Dataset #1

Dataset #2

Throughput (Img/Sec)
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.
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.
PATH OF A BATCH

storage

CPU

GPU

batch size = 3

input pipeline

forward

backward

1 batch

training job

1 batch

forward

backward
PATH OF A BATCH

- Storage
- CPU
  - Input pipeline
- GPU
  - Forward
  - Backward
  - 1 batch
- Training job
  - 1 batch

Batch size = 3
PATH OF A BATCH
GPU UTILIZATION IS NOT EVERYTHING.

- NOT THE FULL PICTURE

- NOT GRANULAR ENOUGH
PATH OF A BATCH

storage

CPU

input pipeline

GPU

forward
backward

1 batch

training job

1 batch

batch size = 3

© 2019 PURE STORAGE INC.
### BATCH TIMING

<table>
<thead>
<tr>
<th>training logs</th>
<th>batch, ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch_results: 917, 1040</td>
<td></td>
</tr>
<tr>
<td>batch_results: 918, 43</td>
<td></td>
</tr>
<tr>
<td>batch_results: 919, 89</td>
<td></td>
</tr>
<tr>
<td>batch_results: 920, 29</td>
<td></td>
</tr>
<tr>
<td>batch_results: 921, 1025</td>
<td></td>
</tr>
<tr>
<td>batch_results: 922, 37</td>
<td></td>
</tr>
<tr>
<td>batch_results: 923, 23</td>
<td></td>
</tr>
<tr>
<td>batch_results: 924, 90</td>
<td></td>
</tr>
<tr>
<td>batch_results: 925, 1053</td>
<td></td>
</tr>
<tr>
<td>batch_results: 926, 31</td>
<td></td>
</tr>
<tr>
<td>batch_results: 927, 21</td>
<td></td>
</tr>
</tbody>
</table>
### BATCH TIMING ODDITIES

<table>
<thead>
<tr>
<th>training logs:</th>
<th>batch, ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>batch_results: 917,</td>
<td><strong>1040</strong></td>
</tr>
<tr>
<td>batch_results: 918,</td>
<td>43</td>
</tr>
<tr>
<td>batch_results: 919,</td>
<td>89</td>
</tr>
<tr>
<td>batch_results: 920,</td>
<td>29</td>
</tr>
<tr>
<td>batch_results: 921,</td>
<td><strong>1025</strong></td>
</tr>
<tr>
<td>batch_results: 922,</td>
<td>37</td>
</tr>
<tr>
<td>batch_results: 923,</td>
<td>23</td>
</tr>
<tr>
<td>batch_results: 924,</td>
<td>90</td>
</tr>
<tr>
<td>batch_results: 925,</td>
<td><strong>1053</strong></td>
</tr>
<tr>
<td>batch_results: 926,</td>
<td>31</td>
</tr>
<tr>
<td>batch_results: 927,</td>
<td>21</td>
</tr>
</tbody>
</table>
WHAT’S “GOOD” PERFORMANCE?
WHAT’S “GOOD” PERFORMANCE?

32 x 1-GPU performance

42% gap

32-GPU performance
TESTING WITH SYNTHETIC DATA

- Do not use a training dataset
- Train on random tensors generated on GPU
- batch size = 3

- forward
- backward

1 batch

1 batch

training job
BREAK DOWN THE PROBLEM SPACE
REAL VS. SYNTHETIC DATA

Linear Performance
Synthetic Performance
28,200 images/s
27,200 images/s

11,800 i/s

32-GPU performance
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

Linear Performance
28,200 images/s
27,200 images/s

Synthetic Performance

25,200 i/s

- No distortions
- Thread pool limits (num_inter_threads)
- Prefetch queue (batch_group_size)

Defaults + Prefetch + Thread Pool Limit - Distortions
WE BALANCED THIS PIPELINE

<table>
<thead>
<tr>
<th>Performance Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Performance</td>
<td>28,200 images/s</td>
</tr>
<tr>
<td>Synthetic Performance</td>
<td>27,200 images/s</td>
</tr>
</tbody>
</table>

"TUNED"
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

stored JPEG

LOAD
DECODE
CROP, DISTORT
TENSOR READY FOR GPU
DATASETS FOR BENCHMARK JOBS

stored JPEG

LOAD

DECODE

CROP, DISTORT

stored tensor

tensor([[ 0.5022, 0.6049, 0.7077, ..., 0.7077, 0.6392, 0.7762],
[ 0.4206, 0.5234, 0.6262, ..., 0.5234, 0.4546, 0.1923],
[ 0.4851, 0.5809, 0.6867, ..., 1.4612, 1.3452, 1.0431],
[ 1.7694, 1.6324, 1.4612, ..., 1.1815, 0.9588, 0.7762],
[ 1.4098, 1.2214, 0.9588, ..., 1.3815, 1.0431, 1.0738],
[ 1.7865, 1.5297, 1.3452, ..., 1.4612, 1.5468, 1.7304],
[ 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],
[ 0.8380, 0.7052, 0.4943, ..., 0.7751, 0.6705, 0.7751],
[ 0.8380, 0.7052, 0.4943, ..., 0.7751, 0.6705, 0.7751],
[ 1.3844, 0.2363, 0.8163, ..., 1.2137, 1.1411, 1.2380],
[ 1.6465, 1.3051, 1.0017, ..., 1.4200, 1.5071, 1.6844]])
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, 1.1586, 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]]])
PNG IMAGES - NICE LINEAR SCALE
PNG IMAGES — 50% THROUGHPUT

EVEN WHEN "TUNED"

PNG images: utilizes data pipeline to load & transform
SYNTHETIC: data generated on GPUs
IMAGE REPRESENTATION

size

- 768x768 pixels
- 224x224 pixels

tensor

format

JPEG  PNG
IMAGE FORMAT

Higher image complexity = Longer load time
IMAGE FORMAT & IMAGE SIZE

Larger image size = Longer load time
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

## Index

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

## Load

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]],

tensor([[ 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,  0.8749],
        [ 0.7829,  0.5553,  0.3277, ...,  0.3277,  0.3102,  0.5828],
        [ 1.0455,  0.7479,  0.3803, ...,  0.7054,  0.6259,  1.0630]],

## Forward

## Backward
LIFE OF A BATCH

CPU
Main process

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, ..., 1.5468, 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, ..., 0.8529, 0.8099, 0.8099],
        ...,
        [ 2.0300, 1.8905, 1.7163, ..., 1.1062, 1.0365, 1.0365],
        [ 1.4548, 1.2621, 1.4098, ..., 1.2337, 1.2411, 1.2968],
        [ 1.6648, 1.3851, 1.0017, ..., 1.7771, 1.5871, 1.6814]])
LIFE OF A BATCH

Main CPU Process

Worker

training_directory = "/dataset/"

[ list of images ]

queue

queue

read image & convert to tensor

memory shared with GPUs

GPU

train on tensor
WORKER DEPENDENCY: BATCH 0

1 CPU worker, 1 GPU
WORKER DEPENDENCY: BATCH 1

worker activity

GPU activity

batch0

batch1

batch0

batch1

wasted GPU time

time
WASTED TIME ADDS UP OVER EPOCHS

worker activity

batch0    batch1    batch2    batch3

GPU activity

batch0    batch1    batch2    batch3

time
WASTED TIME ADDS UP OVER EPOCHS, ESPECIALLY WITH LARGER DATA LOAD TIME.
CONCURRENT WORKERS
SINGLE GPU

worker activity
batch0  batch2
batch1  batch3

GPU activity
batch0  batch1  batch2  batch3

time

job complete
EXAMPLE WITH A REAL RATIO

- 8 concurrent workers, time to process 1 batch = 1.0 sec
- 8 batches processed by GPU(s), time to process 1 batch = 0.1 sec
EXAMPLE WITH A REAL RATIO

MULTI-GPU

worker activity

time to load 1 batch

GPU activity

time to process num_worker batches

time
**IMAGE LOAD TIME V. GPU TIME**

224x224 TENSOR, 8 GPUs, 16 WORKERS

- **worker activity**
  - time to load 1 batch: 1.3 sec

- **GPU activity**
  - time to process num_worker batches: 2.3 sec

GPU-limited
IMAGE LOAD TIME V. GPU TIME

768x768 PNG, 8 GPUs, 16 WORKERS

worker activity 3.8 sec

GPU activity 2.3 sec 1.5 sec every 16 batches

limited by data load

= 39% wasted

time
IMAGE LOAD TIME V. GPU TIME

768x768 JPEG, 8 GPUs, 16 WORKERS

worker activity: 3.2 sec

GPU activity: 2.3 sec, 0.9 sec every 16 batches = 28% wasted

time
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.
QUESTIONS