Performance analysis of deep neural networks making the world safe for Skynet

David Levinthal
Microsoft
Azure Cloud Services Infrastructure
Machine learning and Deep Neural Networks

• Machine learning works by building a network of simple computation nodes executing a “output = F(weight*input + bias)” calculation and using known data to find the optimal weights and biases to identify the patterns in the inputs that correlate to the outputs
• The model is trained on tagged data sets (training)
• The trained model can be used to predict the output for untagged input data (inference)
• https://github.com/David-Levinthal/machine-learning
Deep Neural Networks

• Deep Neural Networks (DNN) can be represented as fabrics of nodes, where the nodes represent numerical operations on (multi dimensional) arrays of data.

Figure 2. Residual learning: a building block.
Estimating CNN properties III

- Alexnet coded for Tensorflow

```python
def __init__(self):
    super(AlexnetModel, self).__init__('_alexnet', 224 + 3, 512, 0.005)

def add_inference(self, cnn):
    # Note: VALID requires padding the images by 3 in width and height
    cnn.conv(64, 11, 11, 4, 4, 'VALID')
    cnn.mpool(3, 3, 2, 2)
    cnn.conv(192, 5, 5)
    cnn.mpool(3, 3, 2, 2)
    cnn.conv(384, 3, 3)
    cnn.mpool(384, 3, 3)
    cnn.conv(256, 3, 3)
    cnn.mpool(3, 3, 2, 2)
    cnn.reshape([-1, 256 * 6 * 6])
    cnn.affine(4096)
    cnn.dropout()
    cnn.affine(4096)
    cnn.dropout()
```

- The 3X3 convolutions here were done with Winograd optimized functions (less than 18 FP ops)
- Measurements were done with NVProf which uses binary instrumentation (165X slow down)
Estimating RNN properties

- LSTM cell is expected to execute $16 \times \text{hidden_size}^2$ FP ops
- Penn TreeBank (PTB) test is a simple benchmark predicting next word
- It can have a variable number of layers, hidden size and time steps
  - Set hidden size=1024, time steps=32, batch size=128 and vary layer count

<table>
<thead>
<tr>
<th></th>
<th>1 Layer</th>
<th>2 Layers</th>
<th>4 Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>total fp_ops</td>
<td>2.64E+12</td>
<td>3.82E+12</td>
<td>6.16E+12</td>
</tr>
<tr>
<td>Sgemm fp_ops</td>
<td>2.61E+12</td>
<td>3.78E+12</td>
<td>6.12E+12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Sgemm fp ops</th>
<th>Sgemm fp ops/LSTM</th>
<th>2 point slope</th>
<th>ratio slope/expected value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 layer</td>
<td>2.61E+12</td>
<td>3.75E+07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 layers</td>
<td>3.78E+12</td>
<td>5.43E+07</td>
<td>16781312</td>
<td>1.000244</td>
</tr>
<tr>
<td>4 layers</td>
<td>6.12E+12</td>
<td>8.78E+07</td>
<td>16781312</td>
<td>1.000244</td>
</tr>
</tbody>
</table>

• There is a large non zero baseline
Estimating Transformer Properties

• Built of modules consisting of a multi head attention (8 or 16 heads)
• and a residual feed forward
• With $N_x = 6$: Total FP ops $\sim 6 \times (3 \times 2 \times (L^2 \times \text{dim}_\text{model} + L \times \text{dim}_\text{model}^2 / \text{num}_\text{head}) + L \times 64 \times \text{dim}_\text{model}^2)$
• So a Linear term and a quadratic one
• Quadratic term should dominate
  • $L \sim 30$, dim_model = 1024 or 512
Viewing DNN performance from a hardware perspective

- DNN performance must be separated by training and inference
- In each case there are many run configuration options (hyperparameters)
- Both are effected by minibatch size: number of images/sentences processed together
  - Creates larger matrices for GEMM functions
  - Greatly effects speed
- Training has a many additional options (learning rate, dropout, gradient evaluation, synchronization strategy, etc)
Google TF CNNs – Inference
(Images/sec vs. Batch Size)

Perf flattens out > batch size 512
P40 very competitive with P100 at smaller batch sizes
P100 does a lot better than P40 at larger batch sizes
P4 is hampered by lower memory capacity (8 GB vs 16 and 24 GB for P40 and P100)
P4 performance pretty good for it’s price at low batch sizes (it was designed for inference where small batches are desirable)
Training speed vs hidden_size$^2$ (relative units) large baseline independent of GPU capacity
Varying model size in transformers
MLPerf https://mlperf.org/

- Objective multi framework training benchmark run to convergence
- Compare performance/cost of cloud VMs
- Current belief is that equal versions across frameworks can be ported

<table>
<thead>
<tr>
<th>Usage</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>image_classification</td>
<td>resnet50-tensorflow</td>
</tr>
<tr>
<td>object_detection</td>
<td>rcnn-caffe2</td>
</tr>
<tr>
<td>recommendation</td>
<td>neural filtering-pytorch</td>
</tr>
<tr>
<td>reinforcement</td>
<td>minigo-tensorflow</td>
</tr>
<tr>
<td>sentiment_analysis</td>
<td>cnn/rnn text categorization-paddle</td>
</tr>
<tr>
<td>speech_recognition</td>
<td>deepspeech2-pytorch</td>
</tr>
<tr>
<td>translation</td>
<td>transformer-tensorflow</td>
</tr>
</tbody>
</table>
Tools for performance evaluation

• Most machine learning codes are written in python
  • Which invoke compiled framework libraries
  • Which in turn invoke hardware specific math libraries (Cuda, MKL etc)
• As python is an interpreted language, dynamic tracing is required for most analysis.
• There are native python tracers, tracers built into the frameworks in some cases and HW based tools
• HW based tools for CPUs are tied to the performance counters
• HW based tools for GPUs are proprietary (NVProf for Nvidia) and may require binary instrumentation for some measurements
NVProf profiles activity on the GPU only

Slows down code by very large factor (~150X) if things like FP operation counts are collected
Not so bad if only time is collected
Output is CSV, example below is post processed to add some information about the batching and so on
Cprofile only see Python execution
Tracing real ML networks yields a complicated result (Tensorflow timeline)
Hidden size = 1024 minibatch 5
Time stamps of some records went crazy
Framework profilers have issues

• Not clear TF profiler works
• MxNet profiler has issues with symbols/long names
• Python profilers have issues seeing into compiled libraries
  • ie the frameworks
• HW profilers
  • Nvprof requires binary instrumentation (and 165X slow down) for anything beyond cycles
Intermediate representations

• Intermediate representations for deep neural networks
  • Create a framework independent representation
    • Simplifying multi framework support from HW vendors
  • Enable rational approaches to network calculation optimization
  • Multi layer fusion
    • Ex: conv layer followed by relu layer followed by max pooling layer
    • Combine to a single layer to avoid data movement
    • Multi layer fusion is also done independently from IR ex: Nvidia TensorRT

• XLA and ONNX are currently popular approaches
Impact of XLA on Resnet50 @ fp32

TF R1.7, Cuda 9.1, cudnn 7.1.2
Conclusions

• Understanding performance of machine learning is hard.
• Tools are not good
• Large fractions of time are not in matrix multiplies
  • But we don’t know what is using that time
  • Making HW design improvements a bit difficult