Real Time Server Monitoring and Fixed Point Inference in FPGA

CERN Openlab Lightning Talk

Debdeep Paul

13/08/2019

Supervisor : Paola Sala
Server Monitoring Architecture

Python scripts to scrape metrics:
- Core Temperatures
- CPU Usage
- Memory Usage
- Inference Time
- Network
- Disk
- Other Stats

Export to HTTP Port

Port 8000 (HTTP)

Prometheus

influxdb

dbod-micron-project-monitoring.cern.ch

Alarm!!

https://micron-project-monitoring.web.cern.ch

smtp.cern.ch:587

Grafana

Real Time Server Monitoring and Fixed Point Inference in FPGA | Debdeep Paul
Server Monitoring Home Dashboard

- CPU
- Memory
- Disk
- Network
- Temperature
- Power

Real Time Server Monitoring and Fixed Point Inference in FPGA | Debdeep Paul
Temperature Monitoring Dashboard

Real Time Server Monitoring and Fixed Point Inference in FPGA | Debdeep Paul
CNN Inference Dashboard
Inference Workflow

CNN Architecture + Trained Weights

Training

Inference

ONNX

Forward Next Library

Bit-File
**FPGA vs CPU**

Compared latency of CPU and FPGA:

<table>
<thead>
<tr>
<th>CPU</th>
<th>Timing Benchmark</th>
<th><strong>Average inference time for a single image:</strong> 49.52706972757975 ms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Time taken to load model from disk:</strong> 2.501638174057007 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Time taken for image pre-processing:</strong> 3.3588409423828125 ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FPGA</th>
<th>Timing Benchmark</th>
<th><strong>Average inference time for a single image:</strong> 190.1567 ms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Time taken to load input to main memory:</strong> 0.4680 ms</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Time taken for converting float to int:</strong> 12.1521 ms</td>
</tr>
</tbody>
</table>

- Inference time of CPU < FPGA: Because CPU used batch processing.
- We concatenate the entire 150 images into a single matrix; So, matrix multiplication becomes a lot faster.
- This is not the case with the FPGA. FPGA process 1 image at a time.

✔ Reducing the batch size to 1 in CPU gave similar timing results.
Working with Fixed Point Numbers

The Overflow problem

- The overflow keeps accumulating in each layer starting from Node 18 (Convolution Layer).
- Max. overflow at Node:25 (Convolution Layer) – 74.77 %
- Training done in keras using single precision floating point numbers (32 bit)
Possible Solutions

- **Normalization of the Images** – Already implemented; still there is overflow

- **Use Dynamic Quantization** – Both layer-wise and different quantization schemes for weights $\epsilon [-1,1]$ and activations (usually large mantissa)

- **Training in Fixed Points** – Quantized both Weights and Activations during training itself to prevent overflow during inference on FPGA.

- **Change Activation Function** – Use tanh or sigmoid instead of ReLU after the layers that have values overflowing.

- **Limitation of the Micron Library**

  - Tried

- **ReLU is more popular**

  - have to try; ReLU is more popular
**Fixed Point Training**

**Using Ristretto**

Ristretto is an automated CNN-approximation tool which condenses 32-bit floating point networks.

- An extension of Caffe and allows to test, train and fine-tune networks with limited numerical precision.
- Trained smaller Networks like LeNet and Cifar-10 to get familiar with the workflow of Ristretto: (q8.8)

<table>
<thead>
<tr>
<th>CNN Architectures</th>
<th>Base Line Accuracy</th>
<th>Fixed Point Accuracy (Training)</th>
<th>Fixed Point Accuracy (Inference)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet (MNIST)</td>
<td>98.76 %</td>
<td>98.76 %</td>
<td>97.61 %</td>
</tr>
<tr>
<td>Cifar-10 Full (Cifar-10)</td>
<td>81.62 %</td>
<td>80.84 %</td>
<td>79.26 %</td>
</tr>
</tbody>
</table>

- Problems with our network:
  - We used batch normalization layers after each layer to bring the activation values within -1<0<1. Currently, Ristretto doesn’t support batch normalization.
  - Removing the layers and changing the network degraded accuracy steeply. So, the proposed solution failed.
Using qpytorch

QPyTorch is a low-precision arithmetic simulation package in PyTorch.

- Converted Model to Pytorch Framework.
- Trained VGG-16 with Cifar-10 dataset to test the working of the library.

<table>
<thead>
<tr>
<th>CNN Architectures</th>
<th>Base Line Accuracy</th>
<th>Fixed Point Accuracy (Training)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-16 (Cifar-10)</td>
<td>92.15 %</td>
<td>92.04 %</td>
</tr>
</tbody>
</table>

- Script to convert ResNet-18 model with the Dune dataset done. Currently training.

- Expecting some good results !!
FUTURE WORK

- Do FPGA Monitoring as well

- Test the quantized network on the entire DUNE Dataset once we get satisfactory results.
Thank You!

QUESTIONS?
debdeep.paul@cern.ch
pauldebdeep2009@gmail.com
www.linkedin.com/in/debdeep-paul-iit2019