PCNN: Parallel Convolutional Neural Network Implementations for Handwritten Digit Recognition

Swagath Venkataramani
Rangharajan Venkatesan
Ashiwan Sivakumar
**OBJECTIVE**

- Implement parallel software versions of Convolutional Neural Networks (CNN)
  - OpenMP, pthreads, MPI
- **Lenet-5**: Designed for handwritten digit recognition application.

- MNIST dataset
- Experiments run on server with 48 AMD cores
Lenet 5: CNN

Input: <32 X 32>

C1 feature map <6 X 28 X 28>

S2 feature map <6 X 14 X 14>

C3 feature map <16 X 10 X 10>

Connection Table <6 X 16>

Output label <10 X 1>
**Lenet 5: Profiling**

**Loop C1**
- 4704 Iterations
- 25 ip. MAC
- Time: ~38%

**Loop S2**
- 1176 Iterations
- 4 ip. Average
- Time: ~3%

**Loop C3**
- 1600 Iterations
- ~90 ip. MAC
- Time: ~43%

**Loop F6**
- 10 Iterations
- 120 ip. MAC
- Time: ~1%

**Loop C5**
- 120 Iterations
- 400 ip. MAC
- Time: ~12%

**Loop S4**
- 400 Iterations
- 4 ip. Average
- Time: ~1%

**Loop C5**
- 120 Iterations
- 400 ip. MAC
- Time: ~12%
TRANSFORMATION 1: NAÏVE PARALLEL

![Diagram](image)

- **C1**
  - $W_0$ (5X 5)
  - $W_5$ (5X 5)

- **S2**
  - 1 2 28 29 30
  - 756 784

- **C3**
  - $W_0$ (5X 5)
  - $W_5$ (5X 5)

- **C5**
  - $W_0$ (5X 5)
  - $W_{1913}$ (5X 5)

- **S4**
  - $W_0$ (5X 5)

- **F6**
  - $W_q$ (120X1)
  - 10

- **Output label**
- **Find Max**

**Graph**
- Speedup vs No of Cores
- Baseline vs Naïve

**Legend**
- Baseline
- Naïve
TRANSFORMATION 2: REHASH & FUSE

6 X 8 Input Feature Map

2 X 4 Convolved Feature Map

2 X 1 Subsampled Feature Map

1 2 3 4
5 6 7 8

2D Conv. 5X5 Window

Sub-sampling 2X2 Window

6 X 8 Input Feature Map

2X1 Convolved & Subsampled Feature Map

1 2 5 6
3 4 7 8

2D Convolution & Sub-sampling 5X5 Conv. Window 2X2 Sub. Sampling

S1 S2

1 2 5 6
3 4 7 8

1 X 2 Convolved & Subsampled Feature Map

S1

1 2 3 4
5 6 7 8

6
TRANSFORMATION 2: REHASH & FUSE

![Diagram showing transformation process]

![Graph showing speedup vs. number of cores]

- **Baseline**
- **Naïve**
- **Fused**

Output label:
- **F6**

- **W_0 (120x1)**
Digit recognition typically processes stream of i/p's

Producer-Consumer relationship across layers in Fused Parallel implementation

**PIPE-FUSED**: Enhanced parallelism through Pipelining

$(1176+400+120+10)$ iterations

TRANFORMATION 3: PIPEFUSED PARALLEL

Pseudo-code

```c
#pragma omp for
for (i=1:N_Total)
if (i < N1)
  process C1-S2
else if (i < N1+N2)
  process C3-S4
else if (i < N1+N2+N3)
  process C5
else
  process F6
```

- N1,N2,N3,N4??
- Design-space exploration
TRANSFORMATION 3: PIPEFUSED PARALLEL

Pseudo-code

#pragma omp for
for (i=1:N_Total)
if ( i < N1)
    process C1-S2
else if ( i < N1+N2)
    process C3-S4
else if ( I < N1+N2+N3)
    process C5
else
    process F6
Distributed memory model - MPI

Naive implementation:
- Broadcast output before running next layer

Transformation 1: Fuse layers
- Eliminates C1-S2 and C3-S4 communication
- Still broadcast between S2-C3 and S4-C5

Transformation 2: “Selective Send” based on connection table
RESULTS

MPI

No of Processors -->

No of Cores -->

Actual
Fused-Bcast
Fused
PipeFused

Fused-Bcast
Fused
PipeFused

Sequential

pthread

Sequential
SUMMARY & FUTURE WORK

- Intense Communication between Neurons – Distributed memory model suffers
- Loop body of each neuron is small – Fork-Join overheads
- Take advantage of “Convolution followed by Sub-sampling”
- Pipe-fused expands the parallelism beyond each network layer

- Parallelize training phase
  - OpenMP and MPI