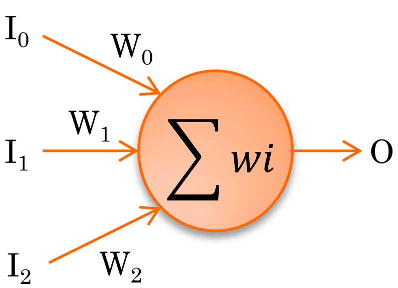
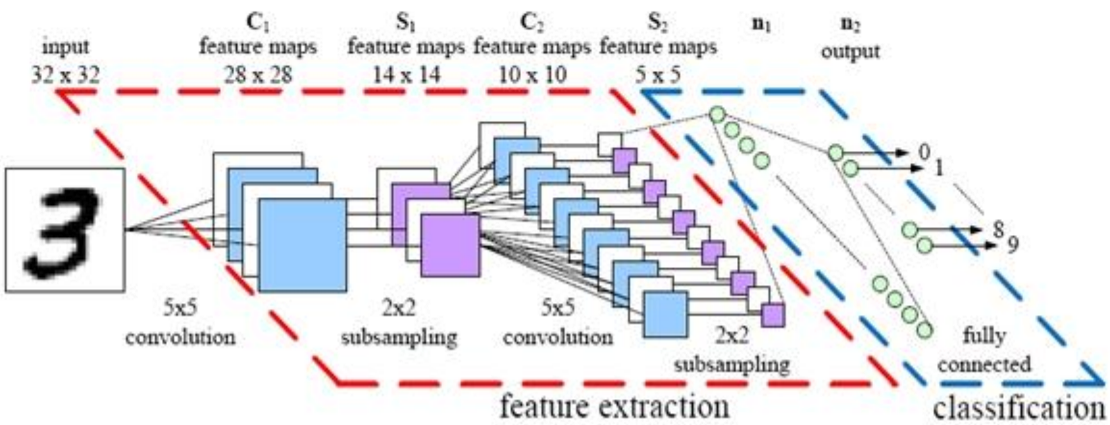
pCNN: Parallel Convolutional Neural Network Implementations for Handwritten Digit Recognition

Swagath Venkataramani
Rangharajan Venkatesan
Ashiwan Sivakumar

Objective

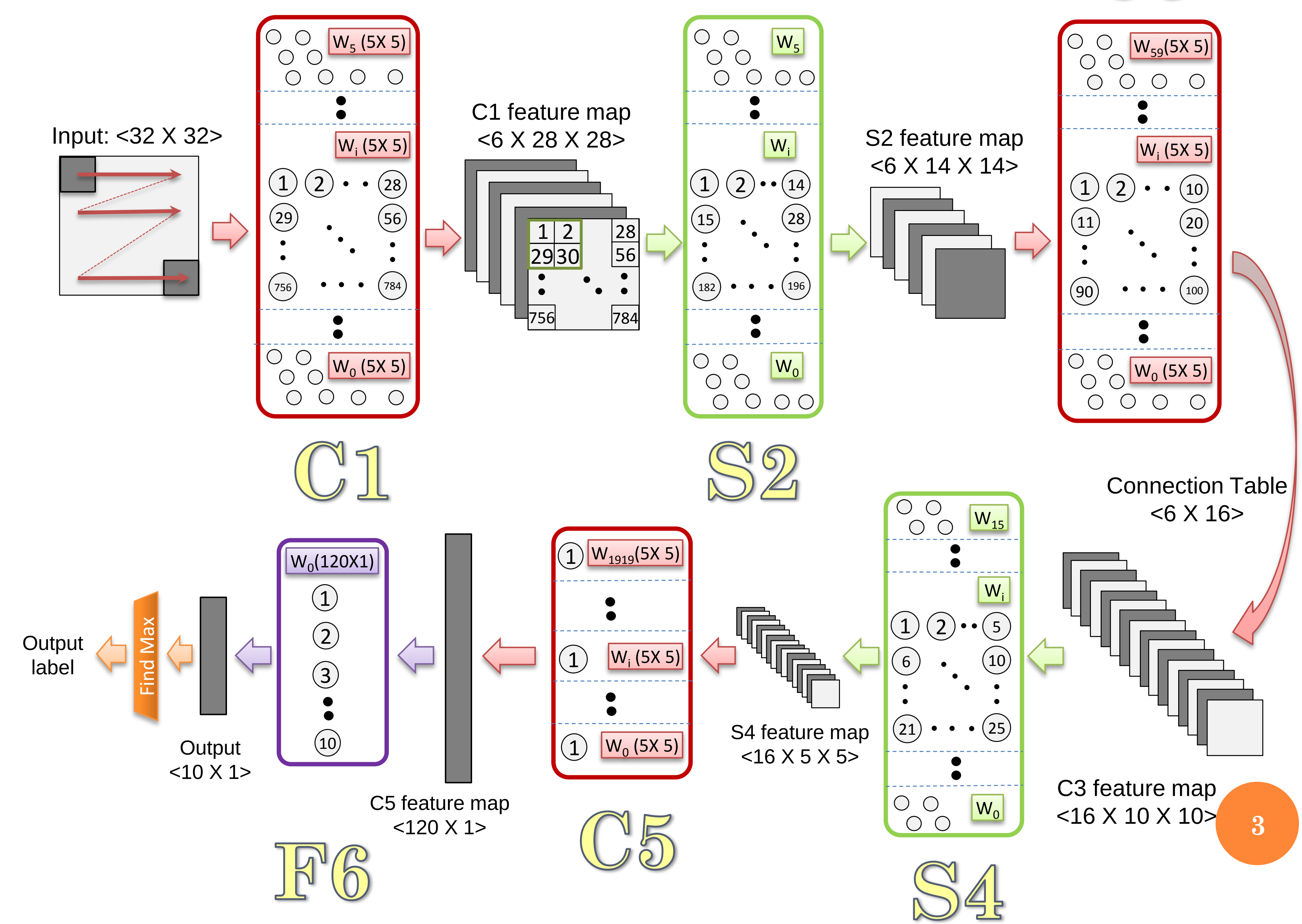
- Implement parallel software versions of Convolutional Neural Networks (CNN)
  - OpenMP, pthreads, MPI
- Lenet-5: Designed for handwritten digit recognition application.





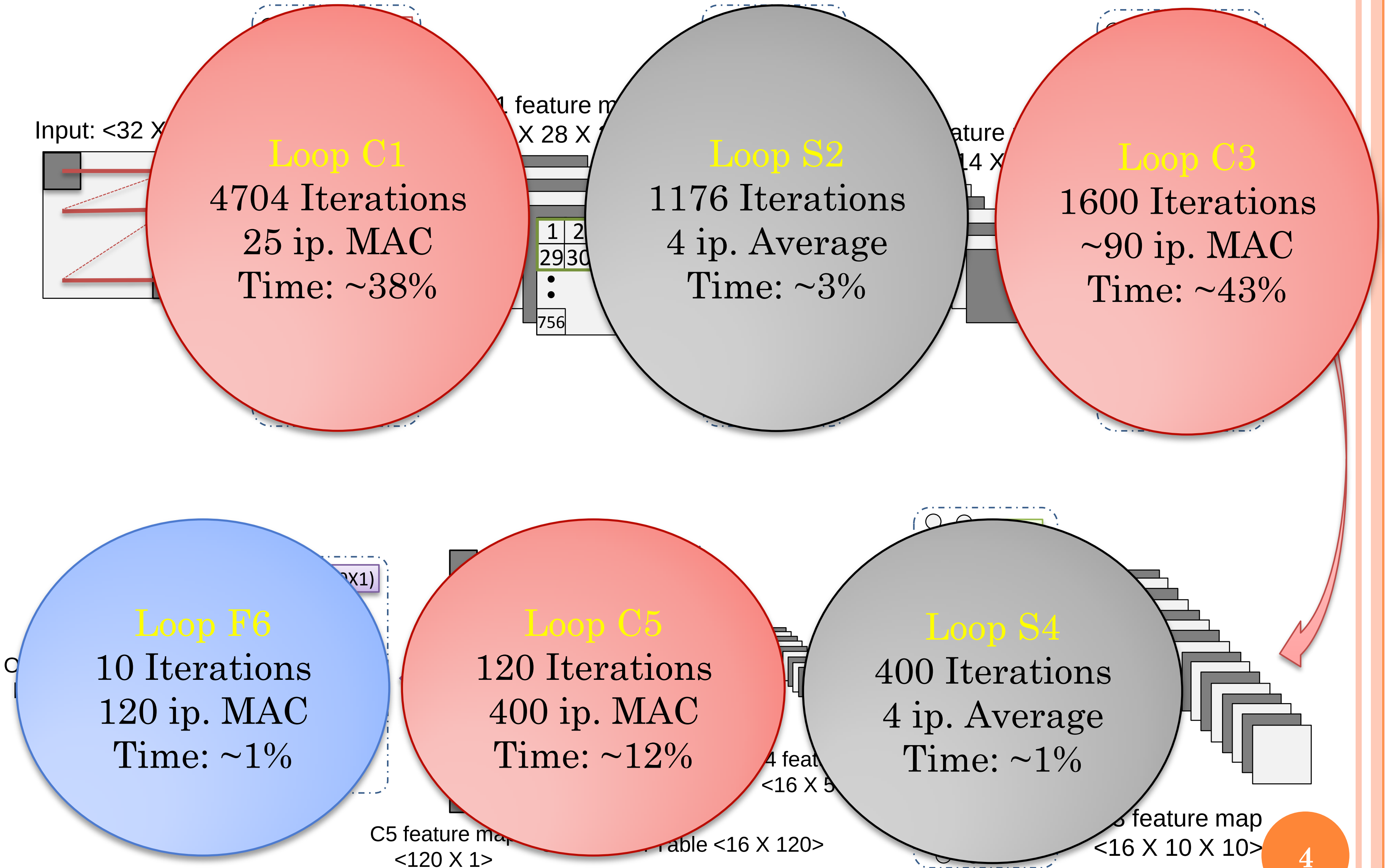
- MNIST dataset
- Experiments run on server with 48 AMD cores

LeNet 5: CNN

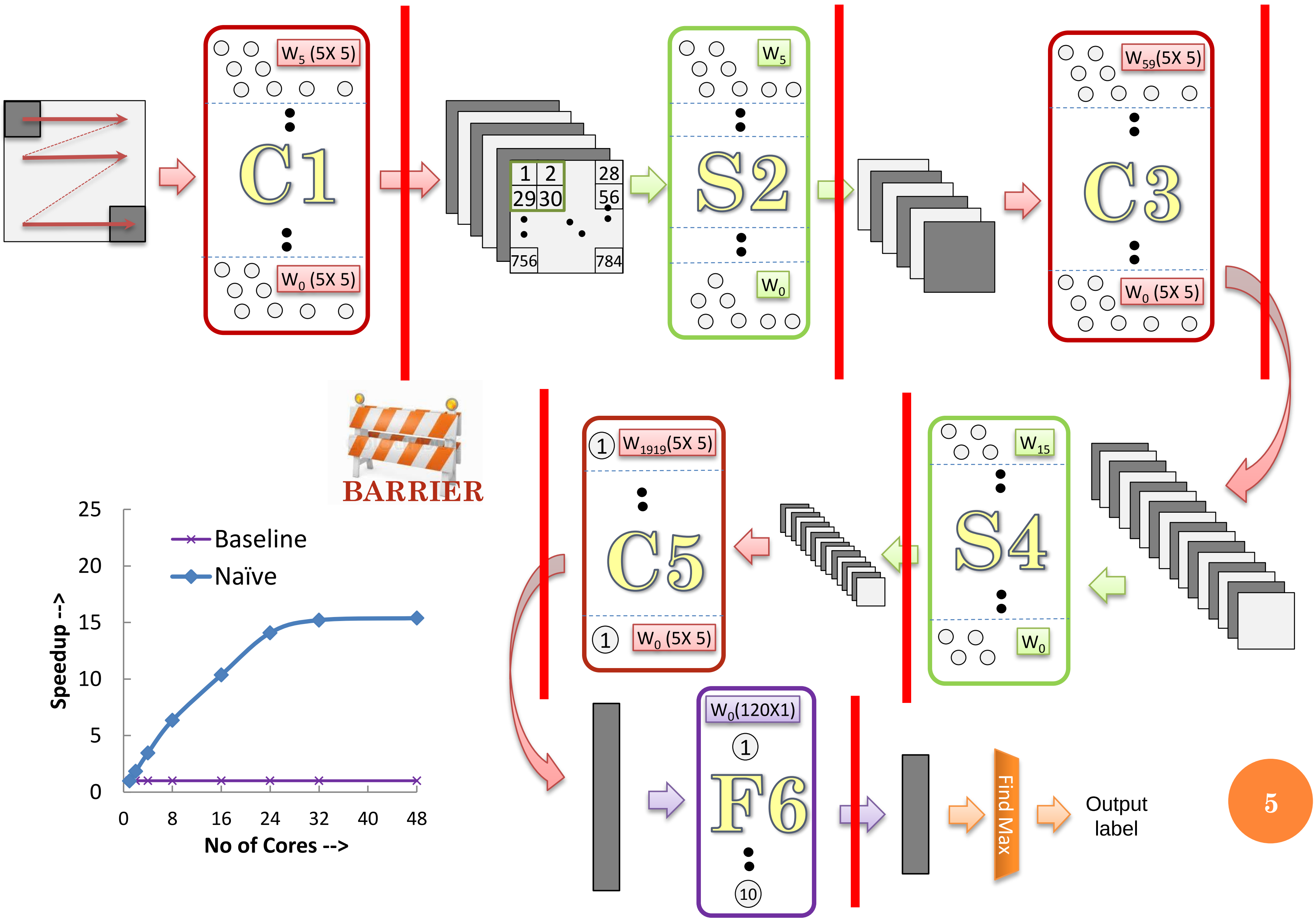


Input: <32 X 32>
C1
W_5 (5X 5)
W_i (5X 5)
W_0 (5X 5)
1 2 28
29 56
756 784
C1 feature map
<6 X 28 X 28>
1 2 28
29 30 56
756 784
S2
W_5
W_i
W_0
1 2 14
15 28
182 196
S2 feature map
<6 X 14 X 14>
C3
W_{59}(5X 5)
W_i (5X 5)
W_0 (5X 5)
1 2 10
11 20
90 100
Connection Table
<6 X 16>
C3 feature map
<16 X 10 X 10>
S4
W_{15}
W_i
W_0
1 2 5
6 10
21 25
S4 feature map
<16 X 5 X 5>
C5
W_{1919}(5X 5)
W_i (5X 5)
W_0 (5X 5)
1
C5 feature map
<120 X 1>
F6
W_0(120X1)
1
2
3
10
Output
<10 X 1>
Find Max
Output
label

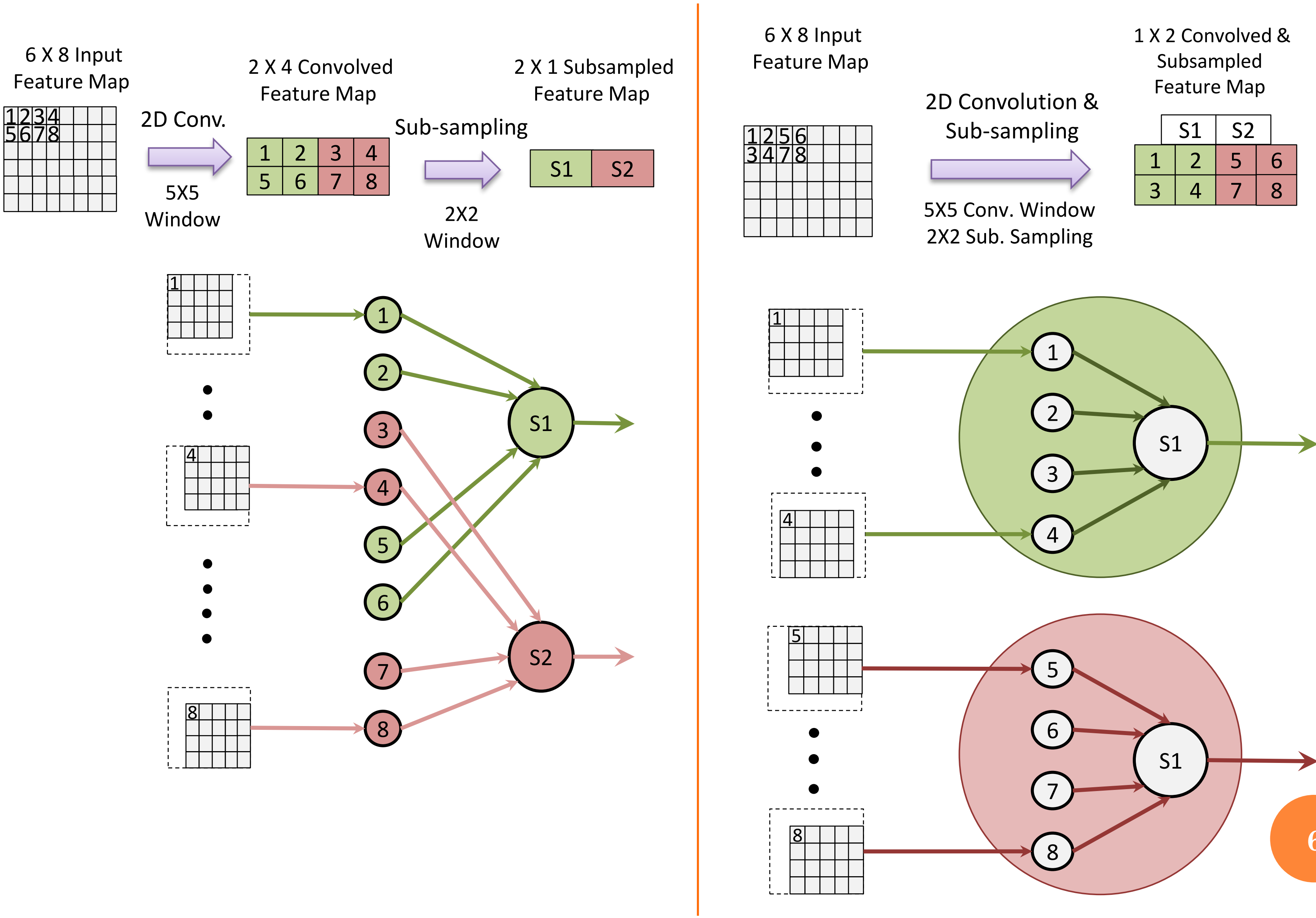
LeNet 5: Profiling



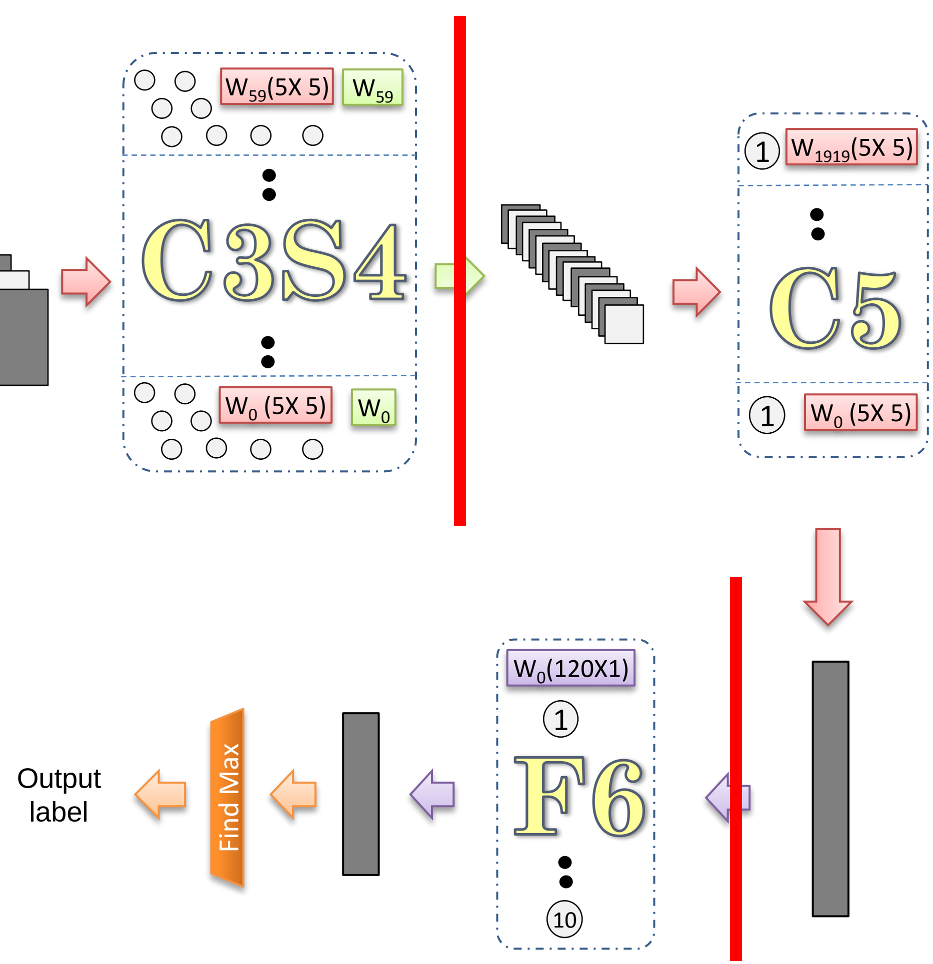
Transformation 1: Naïve Parallel

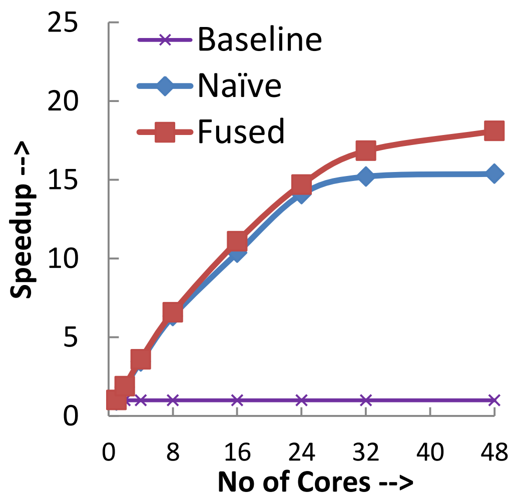


Transformation 2: Rehash & Fuse

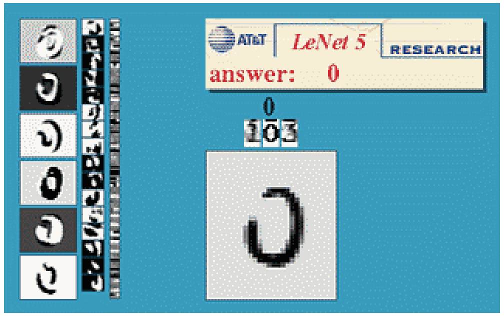


TRANSFORMATION 2: REHASH & FUSE

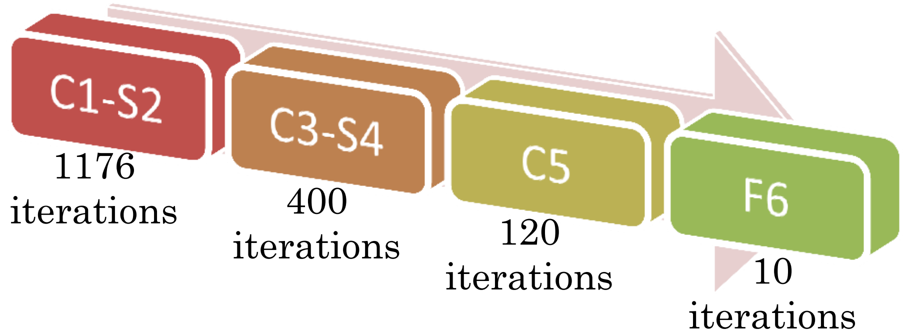




Transformation 3: Pipe-Fused Parallel

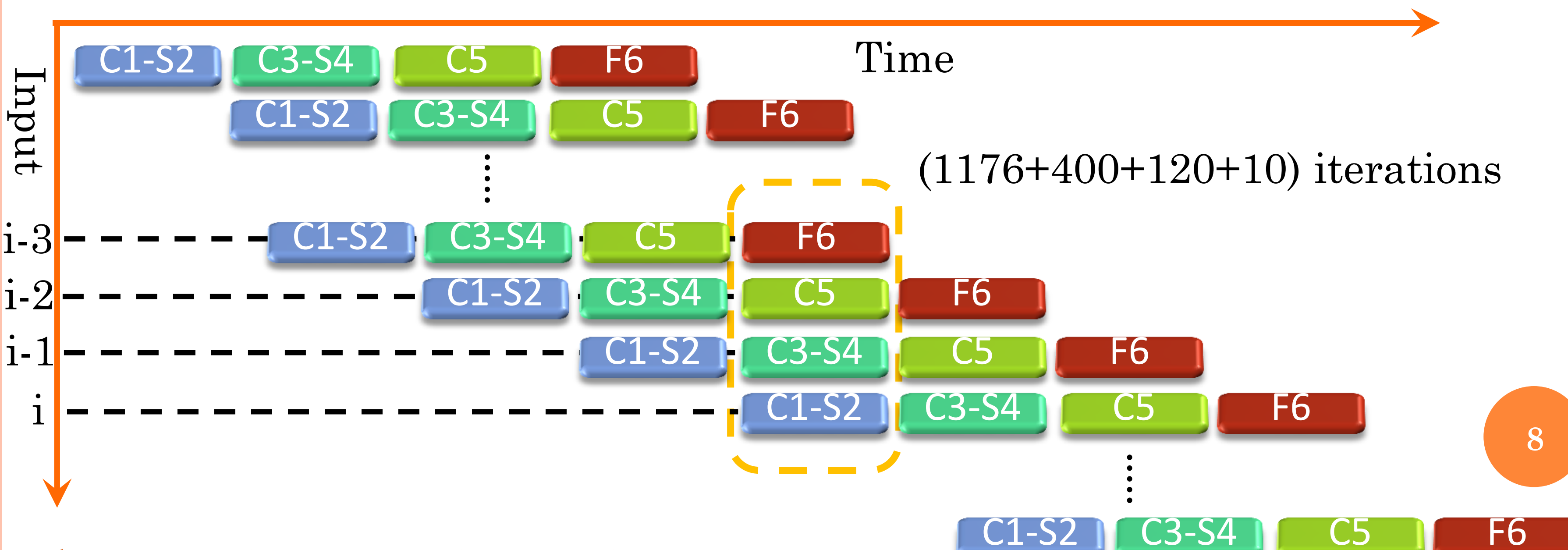


Digit recognition typically processes stream of i/ps[1]



Producer-Consumer relationship across layers in Fused Parallel implementation

PIPE- FUSED: Enhanced parallelism through Pipelining



[1]http://yann.lecun.com/exdb/lenet/index.html

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)

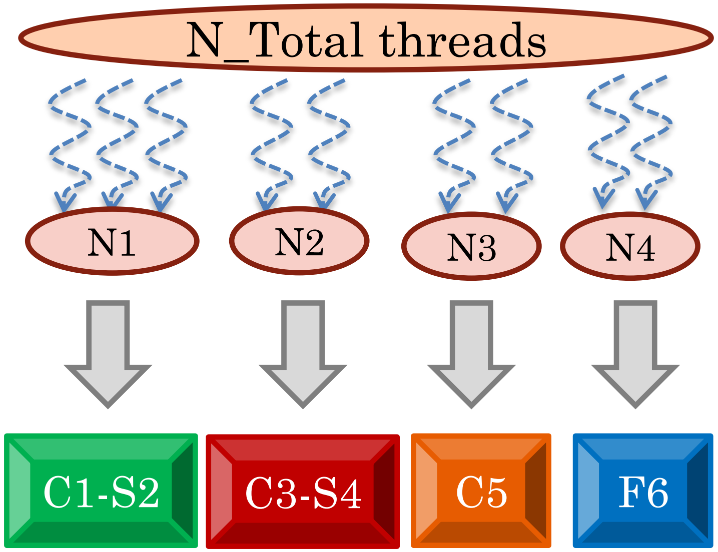
    process C3-S4

else if (i < N1+N2+N3)

    process C5

else

    process F6

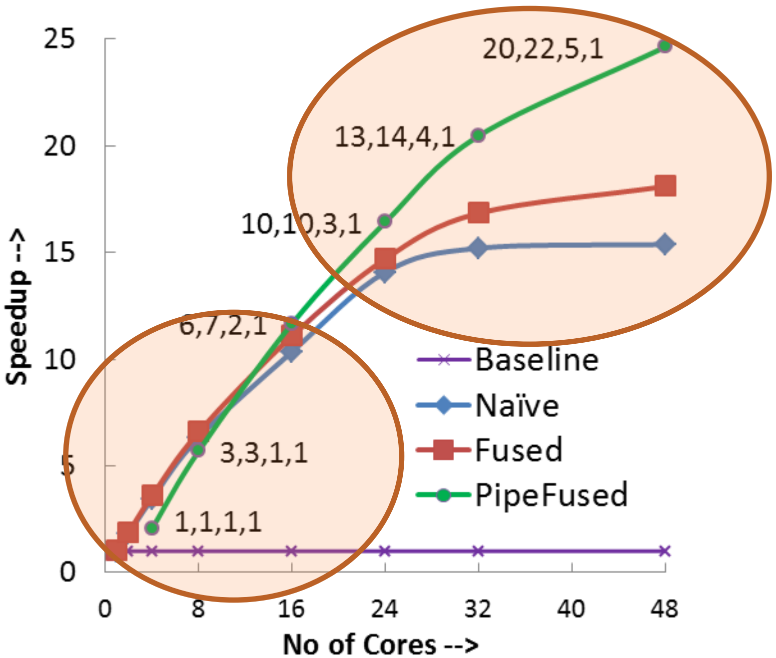


- **N1,N2,N3,N4??**
- **Design-space exploration**

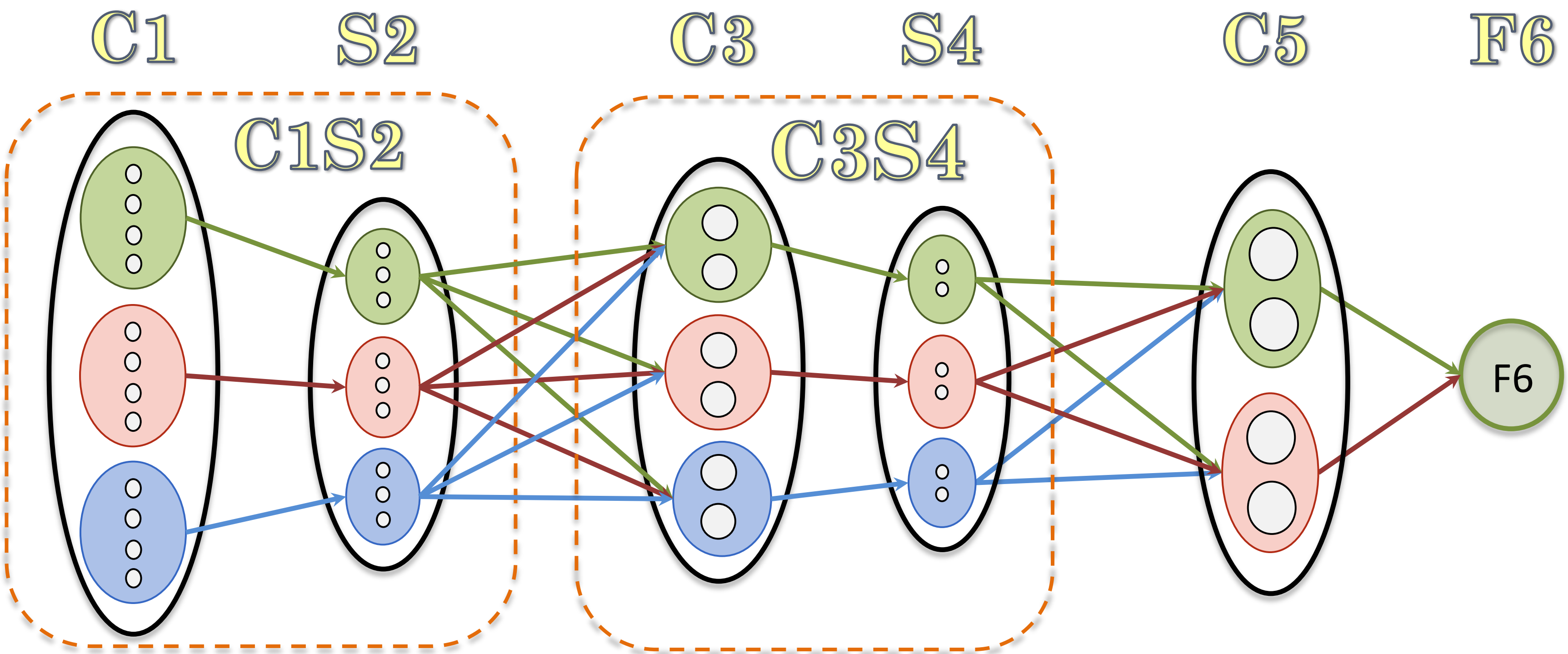
Transformation 3: PipeFused Parallel

<u>Pseudo-code</u>

```
#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

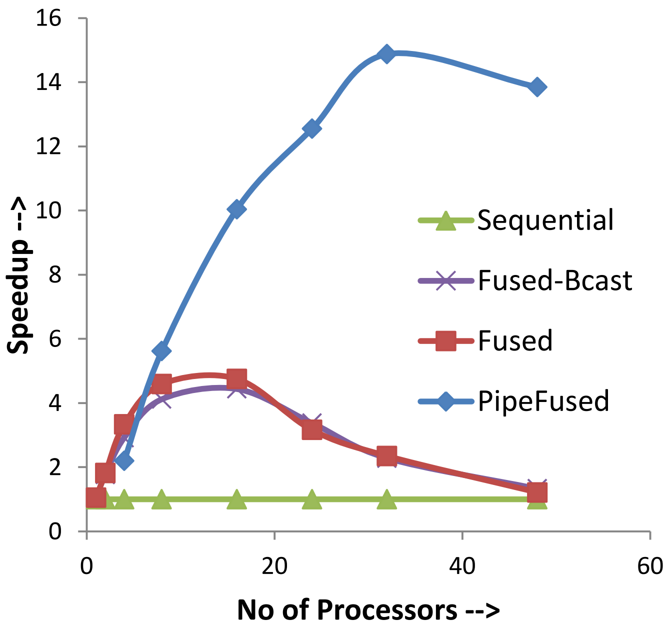
Transformation 1: Fuse layers

- Eliminates C1-S2 and C3-S4 communication
- Still broadcast between S2-C3 and S4-C5

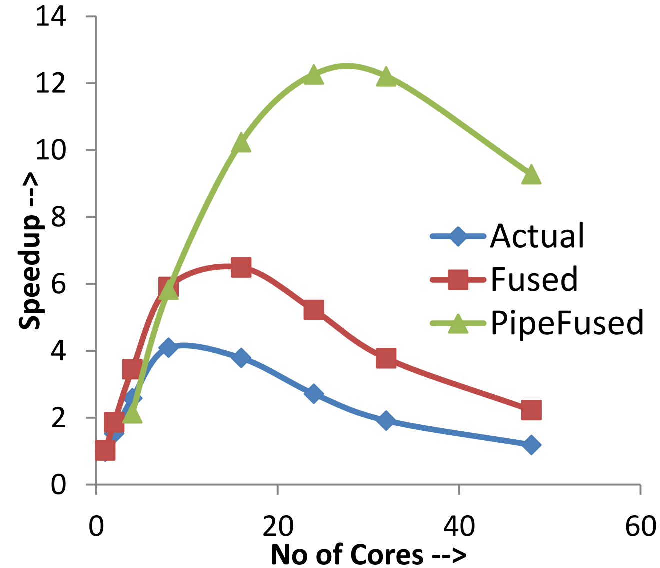
Transformation 2: “Selective Send” based on connection table

Results

MPI



pthread



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