UNIVERSITÉ
PERPIGNAN
VIA
DOMITIA

MOHAMMAD Sulaiman

CHPS-M2

Neural Network – MPI

schema

Modality

Parallel Algo

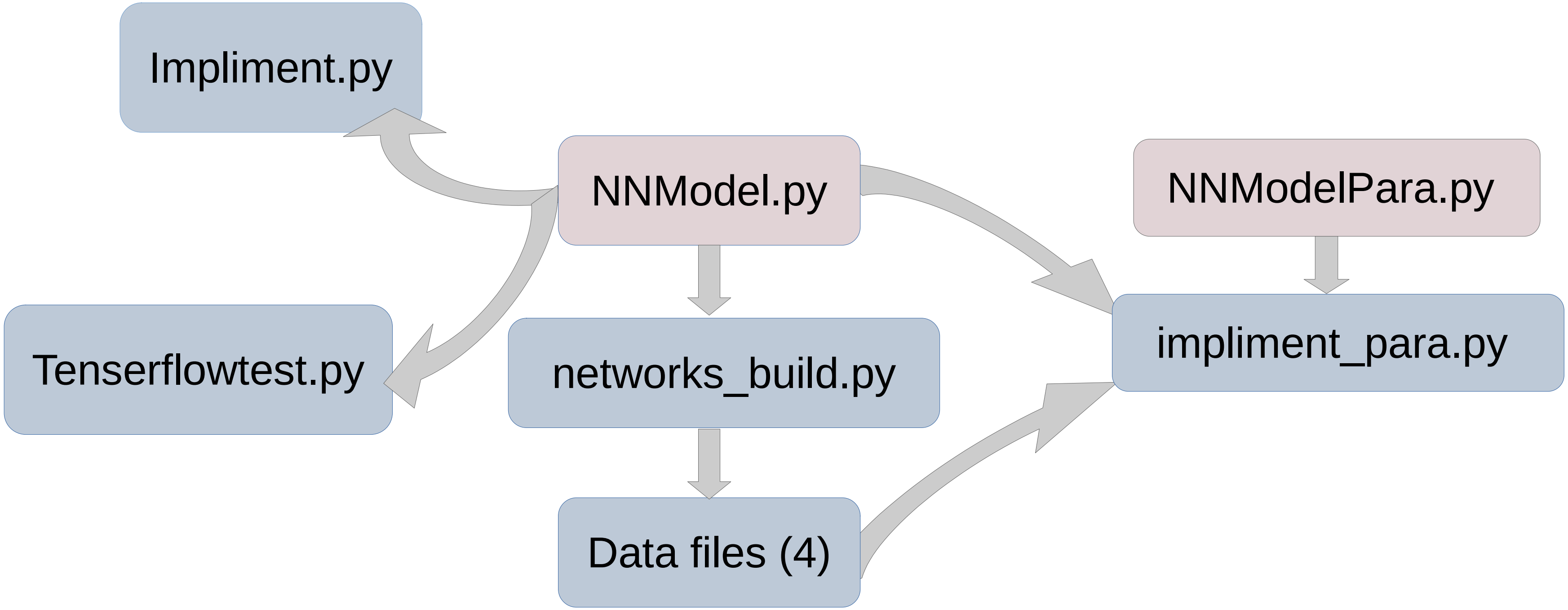
Pruning Algo

Performance Seq

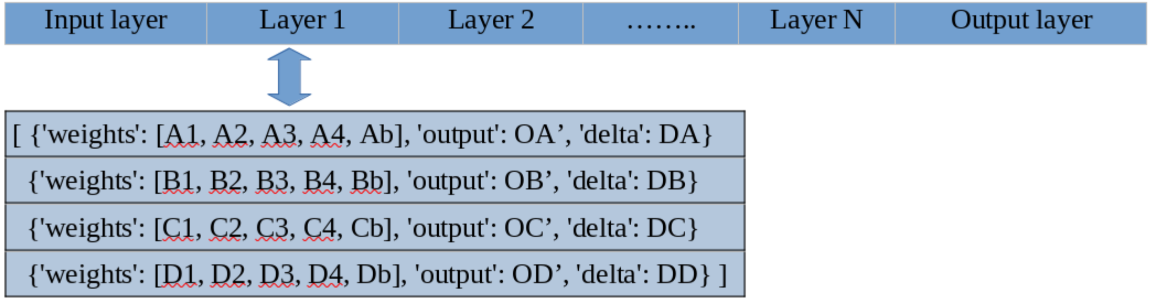
Performance para

Tenserflow vs My model

Modality


Impliment.py
NNModel.py
NNModelPara.py
Tenserflowtest.py
networks_build.py
impliment_para.py
Data files (4)

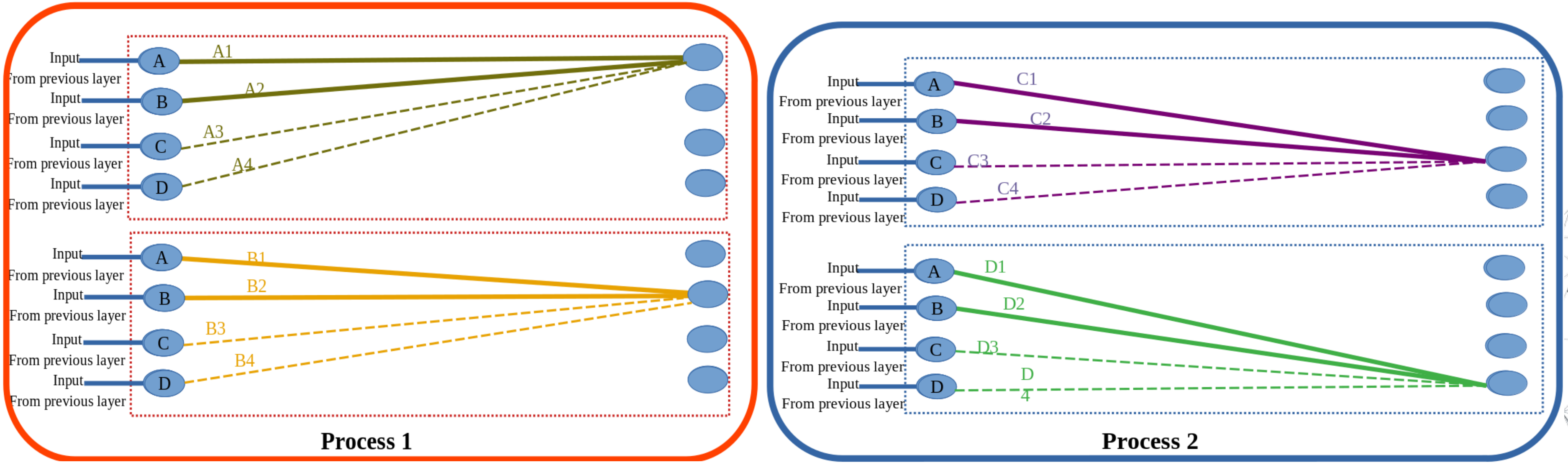
Modality



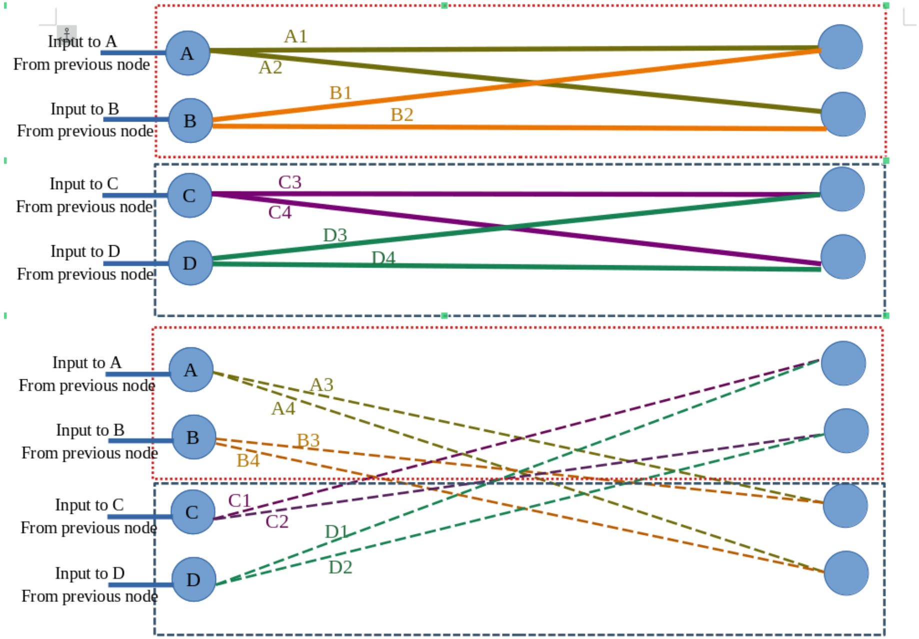
Input layer	Layer 1	Layer 2	……..	Layer N	Output layer

```
[ {'weights': [A1, A2, A3, A4, Ab], 'output': OA’, 'delta': DA}
  {'weights': [B1, B2, B3, B4, Bb], 'output': OB’, 'delta': DB}
  {'weights': [C1, C2, C3, C4, Cb], 'output': OC’, 'delta': DC}
  {'weights': [D1, D2, D3, D4, Db], 'output': OD’, 'delta': DD} ]
```

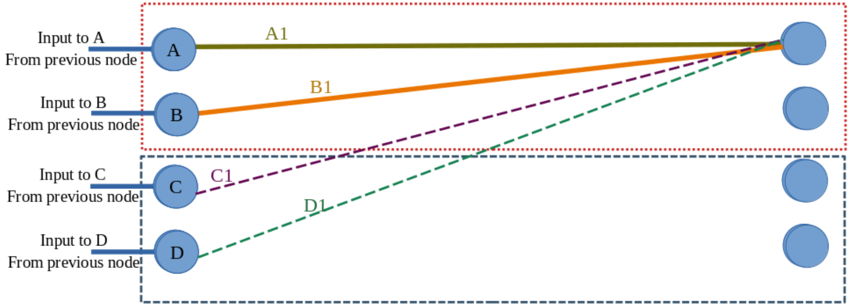
Parallel Algo



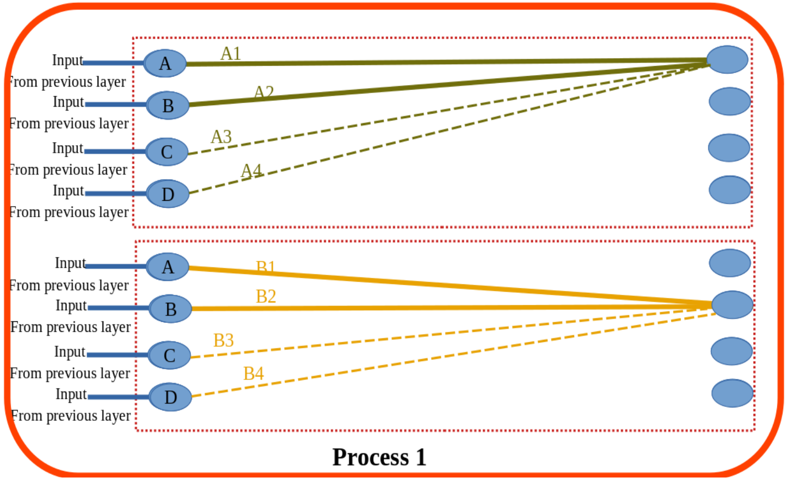
Parallel Algo

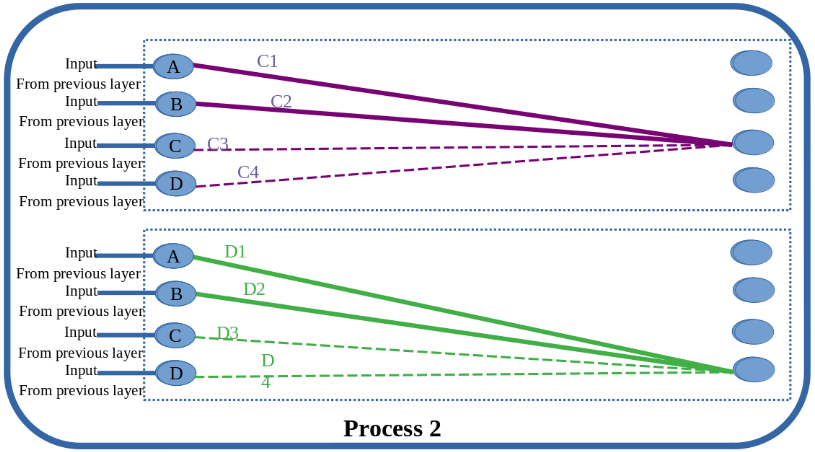


Different model, and the effect



Parallel Algo

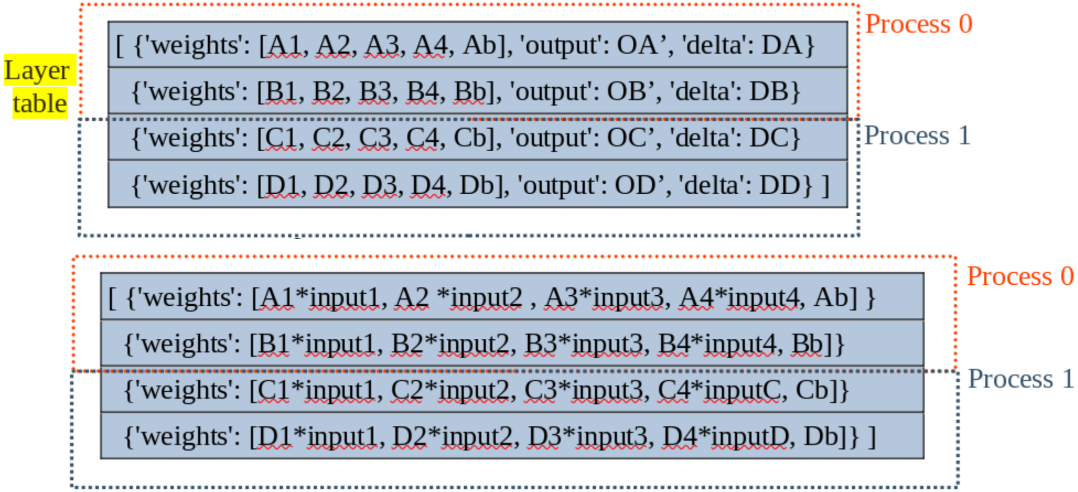




Chunk= (number of neurons/ number of processes)
data =process_prod(layer[rank*chunk: (rank*chunk)+chunk], [previous Inputs])

4 neurons , 2 processes, then each chunk is 2= 4/2 ,So for each processes 0~ [A,B] , 1~[C,D]
so processes(0) should move one chunk from zero so (rank=1*chunk) which is step , then we add chunk to arrive to the final neuron should be included.
for processes(1) we moved 2 chunk from 0 so rank*chunk then move one chunk to the final neuron should be included

Parallel Algo



Layer table

Process 0

Process 1

[{'weights': [A1, A2, A3, A4, Ab], 'output': OA’, 'delta': DA}

{'weights': [B1, B2, B3, B4, Bb], 'output': OB’, 'delta': DB}

{'weights': [C1, C2, C3, C4, Cb], 'output': OC’, 'delta': DC}

{'weights': [D1, D2, D3, D4, Db], 'output': OD’, 'delta': DD}]

Process 0

Process 1

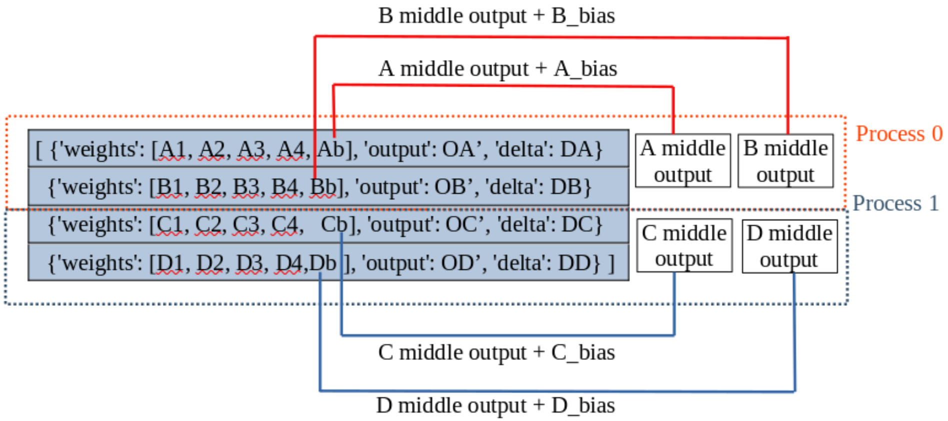
[{'weights': [A1*input1, A2 *input2 , A3*input3, A4*input4, Ab] }

{'weights': [B1*input1, B2*input2, B3*input3, B4*input4, Bb]}

{'weights': [C1*input1, C2*input2, C3*input3, C4*inputC, Cb]}

{'weights': [D1*input1, D2*input2, D3*input3, D4*inputD, Db]}]

Parallel Algo



B middle output + B_bias

A middle output + A_bias

[{'weights': [A1, A2, A3, A4, Ab], 'output': OA’, 'delta': DA}
{'weights': [B1, B2, B3, B4, Bb], 'output': OB’, 'delta': DB}
{'weights': [C1, C2, C3, C4,   Cb], 'output': OC’, 'delta': DC}
{'weights': [D1, D2, D3, D4,Db], 'output': OD’, 'delta': DD}]

A middle output

B middle output

C middle output

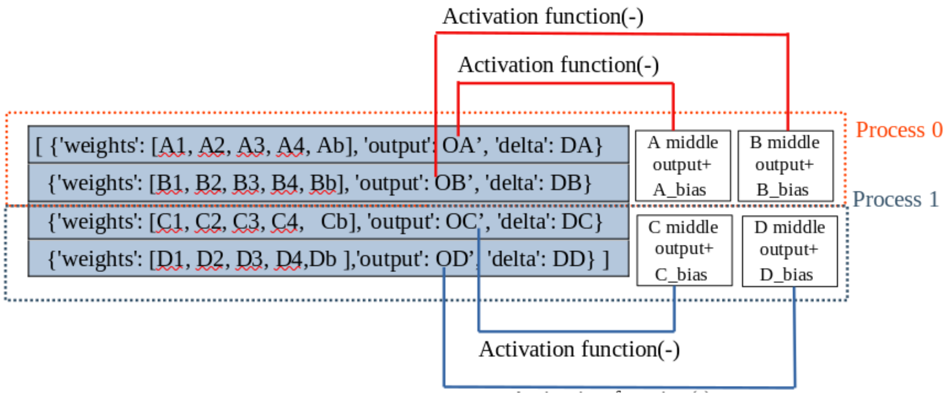
D middle output

Process 0

Process 1

C middle output + C_bias

D middle output + D_bias



Activation function(-)

Activation function(-)

[{'weights': [A1, A2, A3, A4, Ab], 'output': OA’, 'delta': DA}
{'weights': [B1, B2, B3, B4, Bb], 'output': OB’, 'delta': DB}
{'weights': [C1, C2, C3, C4,   Cb], 'output': OC’, 'delta': DC}
{'weights': [D1, D2, D3, D4,Db],'output': OD’, 'delta': DD}]

A middle output+ A_bias

B middle output+ B_bias

C middle output+ C_bias

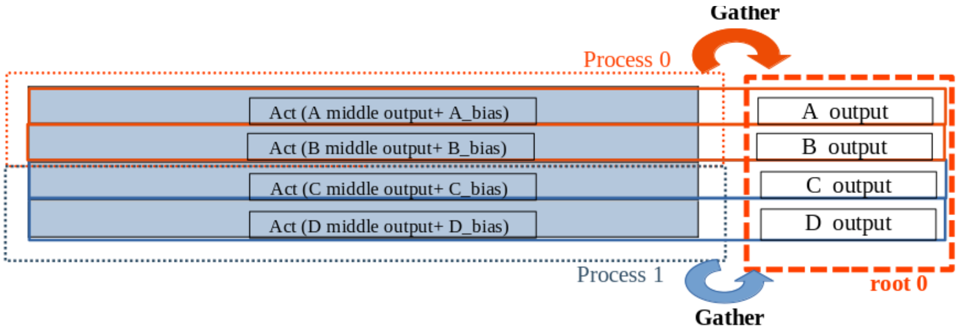
D middle output+ D_bias

Process 0

Process 1

Activation function(-)

Parallel Algo



Gather

Process 0

Act (A middle output+ A_bias)

Act (B middle output+ B_bias)

Act (C middle output+ C_bias)

Act (D middle output+ D_bias)

A output

B output

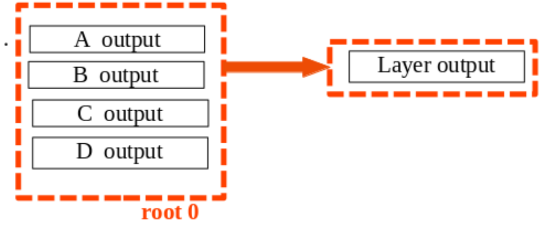
C output

D output

Process 1

Gather

root 0



A output

B output

C output

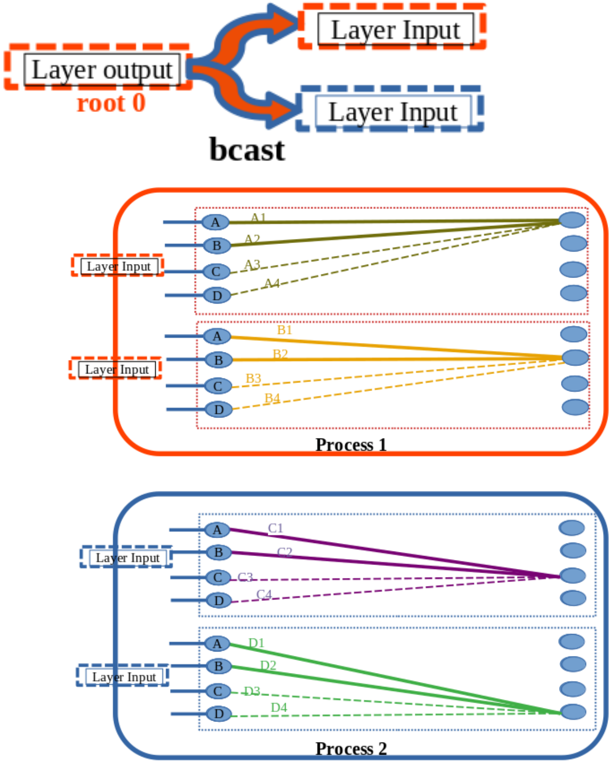
D output

root 0

Layer output

Parallel Algo

Input layer	Layer 1	Layer 2	…….	Layer N	Output layer



Layer output

root 0

bcast

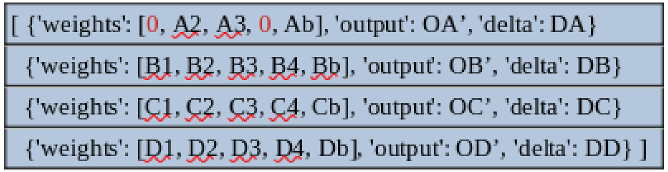
Layer Input

Layer Input

Process 1

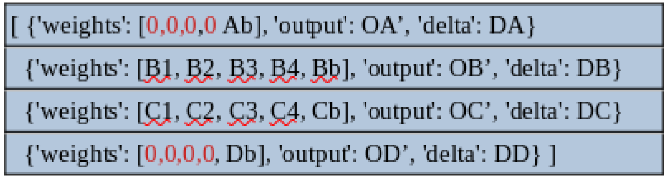
Process 2

Pruning Algo



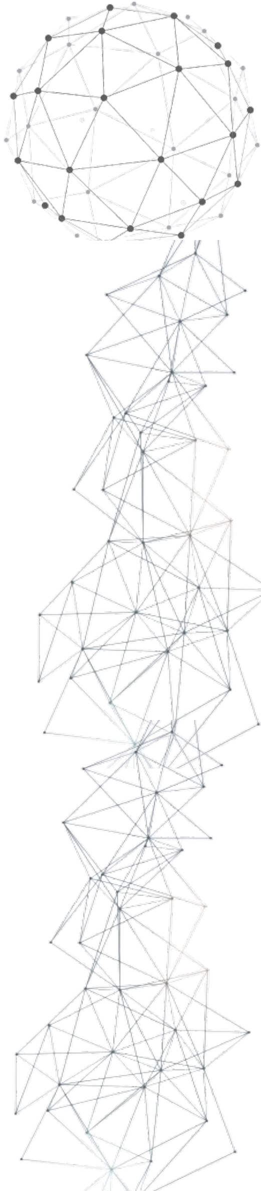
```
[ {'weights': [0, A2, A3, 0, Ab], 'output': OA’, 'delta': DA}
  {'weights': [B1, B2, B3, B4, Bb], 'output': OB’, 'delta': DB}
  {'weights': [C1, C2, C3, C4, Cb], 'output': OC’, 'delta': DC}
  {'weights': [D1, D2, D3, D4, Db], 'output': OD’, 'delta': DD} ]
```

pruning_connections

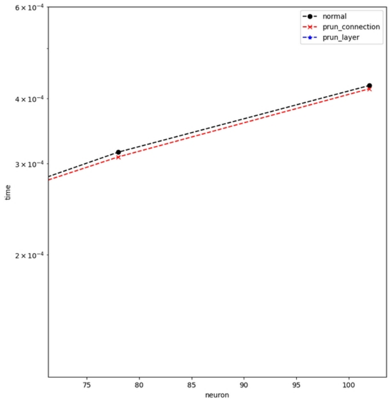


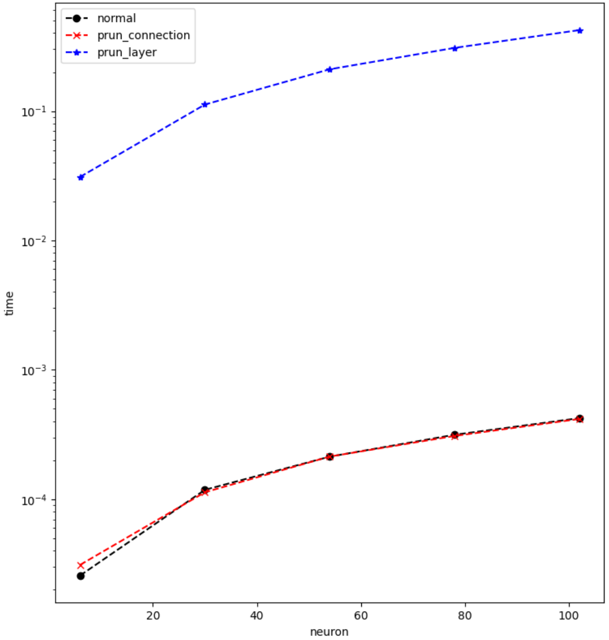
```
[ {'weights': [0,0,0,0 Ab], 'output': OA’, 'delta': DA}
  {'weights': [B1, B2, B3, B4, Bb], 'output': OB’, 'delta': DB}
  {'weights': [C1, C2, C3, C4, Cb], 'output': OC’, 'delta': DC}
  {'weights': [0,0,0,0, Db], 'output': OD’, 'delta': DD} ]
```

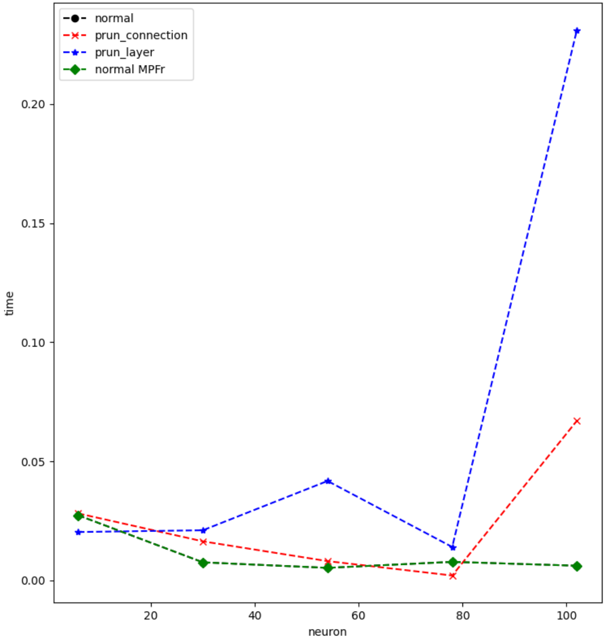
neuron_connections



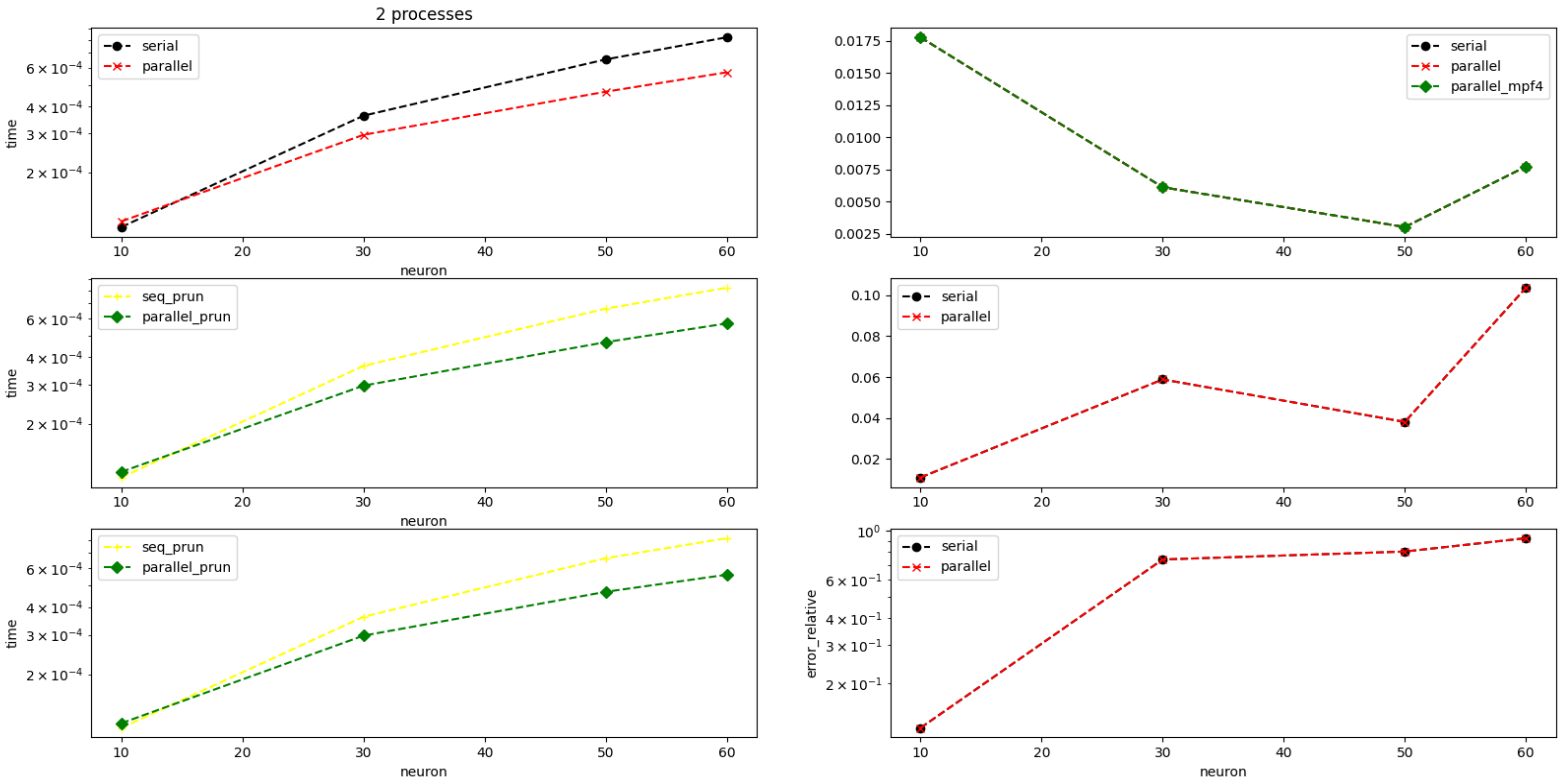
Performance Seq



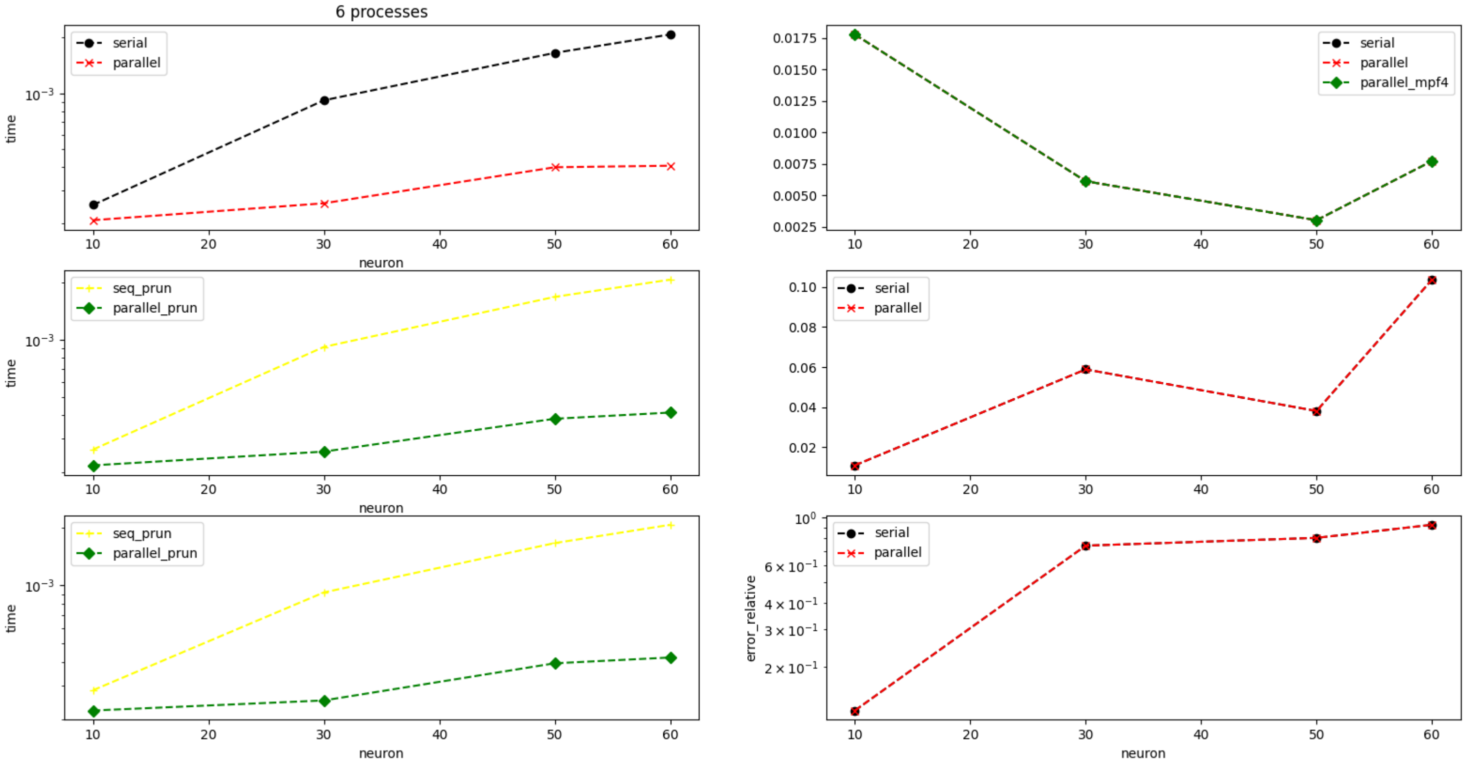


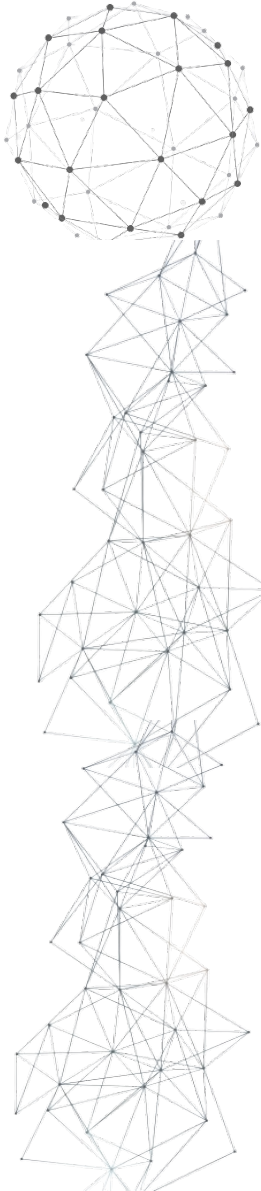


Performance para

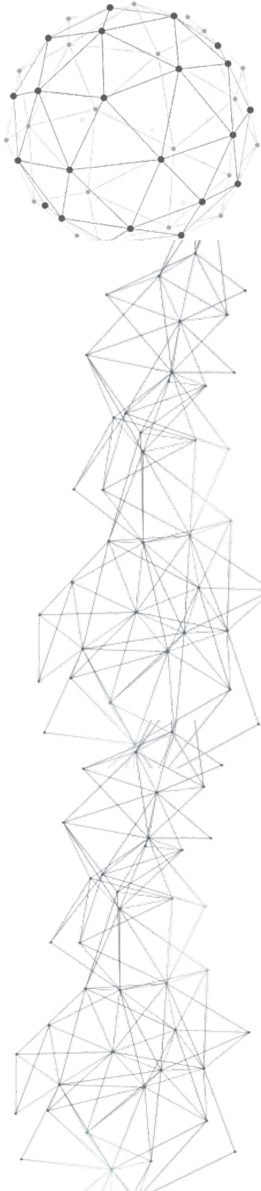


Performance para





Tenserflow vs My model



Network 60*3

tenser flow res [[0.9998767]] time: 0.08174838200011436

My network res: [0.99227889484028 12] time: 0.0009017231559982974

That’s just for sequential predict function.

For the same network in parallel with more than 2 processes is less than $5*10^{-4}$

Thank you for your attention