Distributed Convolutional Neural Network with Apache Spark

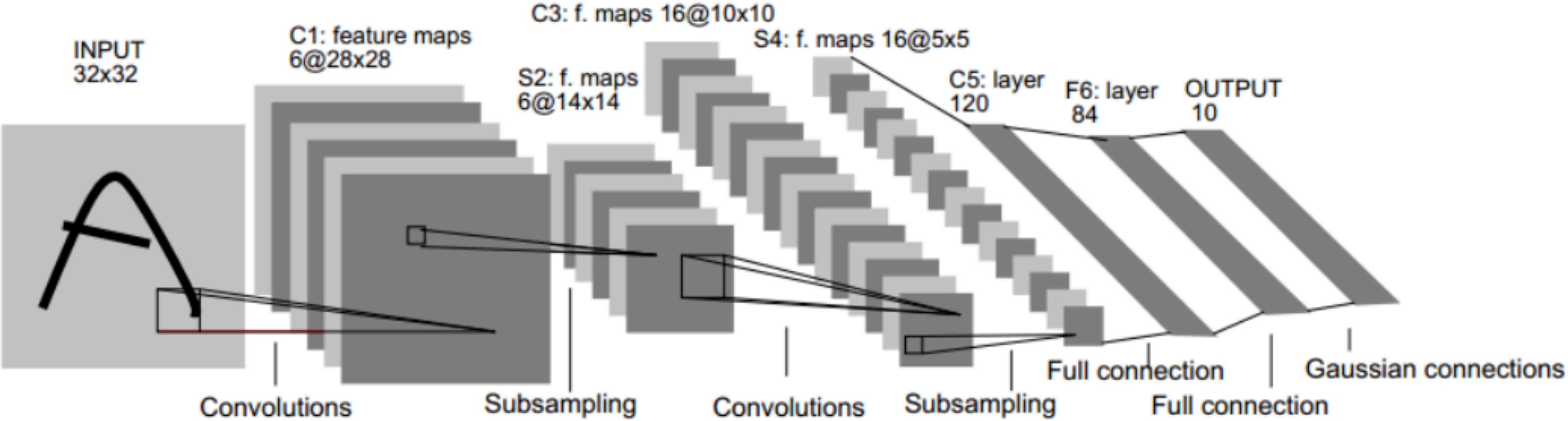
Yuwei Jiao, Vivi Ma

UNIVERSITY OF WATERLOO

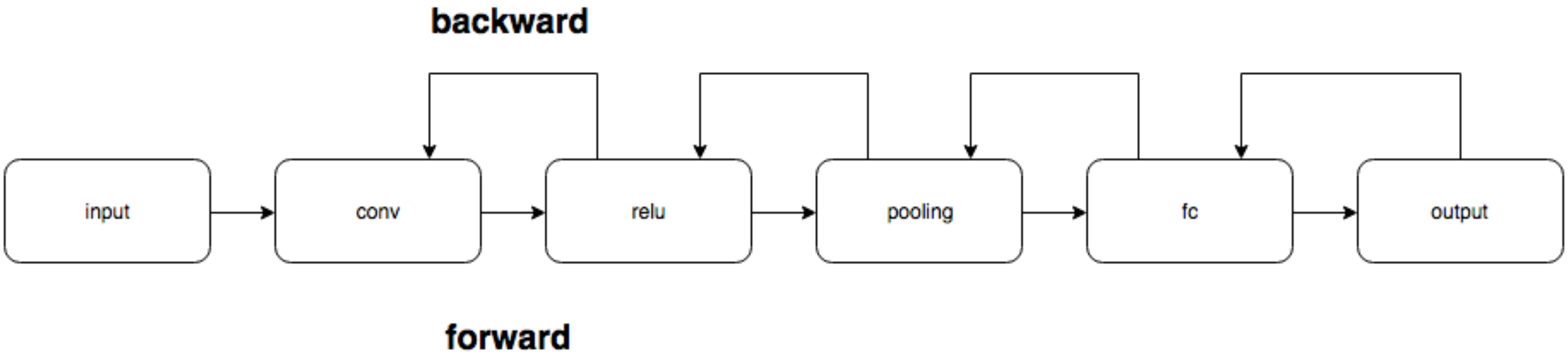
Outline

- Background
- Fundamental CNN Workflow
- Challenges
- Implementation with Apache Spark
- Implementation with TensorFlow
- Future Work

Background


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

Fundamental CNN Workflow



Setting

Language: Python 3

Dataset: CIFAR10

Convolution: 32 filters(5x5x3), stride 1, zero-padding 2;

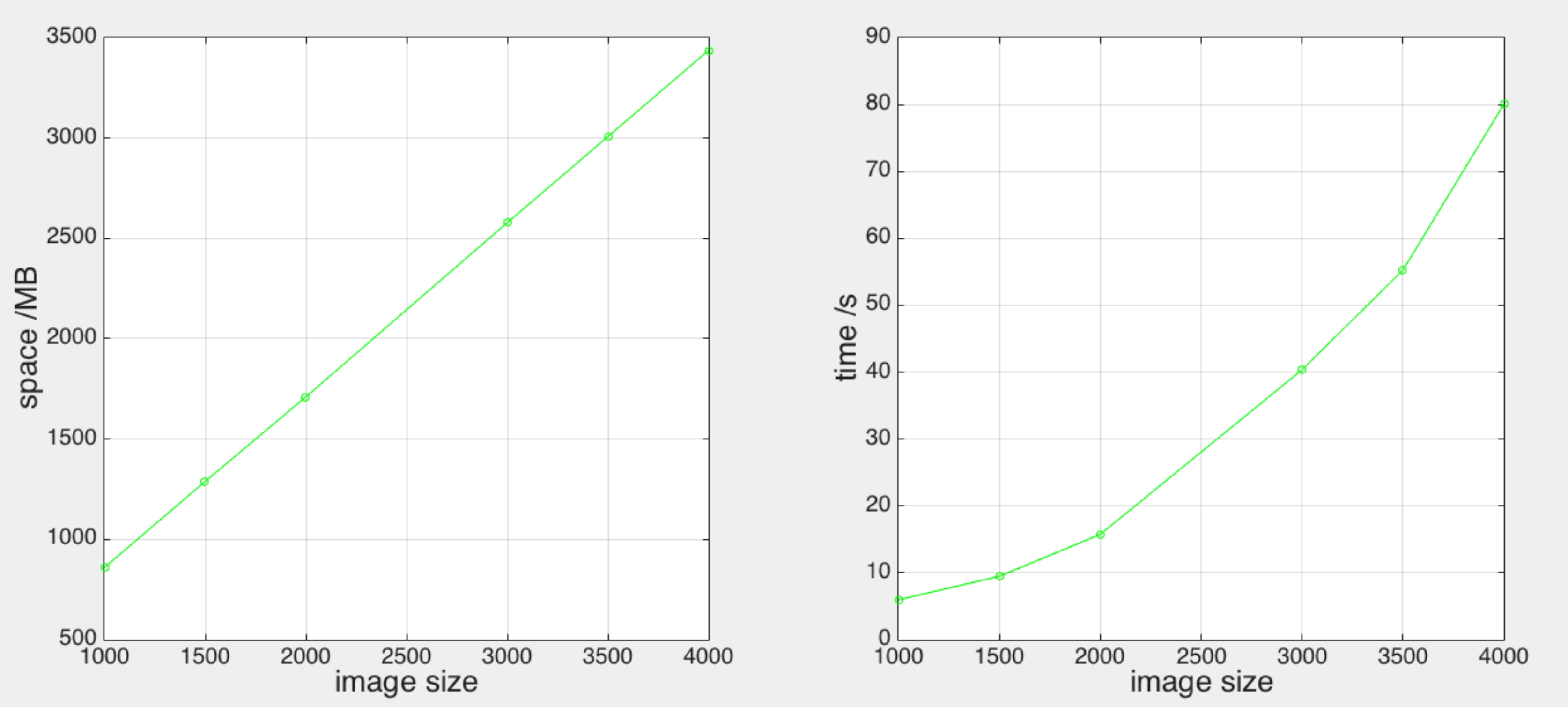
Pooling: 2x2 filter size, stride 2, max pooling;

Fully Connected: 16x16x32 => 10 classifications;

Challenges

- Time Complexity
- Space Complexity

Profiling of Naive Implementation



Profiling of Naive Implementation

2000 Images per Iteration

Conv	forward	4.252s
	backward	10.411s
ReLU	forward	0.504s
	backward	0.458s
Pooling	forward	2.155s
	backward	3.049s
FC	forward	0.159s
	backward	0.380s

Profiling of Naive Implementation

- **Convolution:**

Forward	im2col()	0.871s
	dot()	1.150s
Backward	dot()	2.841s
	col2im	0.289s
	im2col	0.877s
	dot()	1.849s
	sum()	0.035s

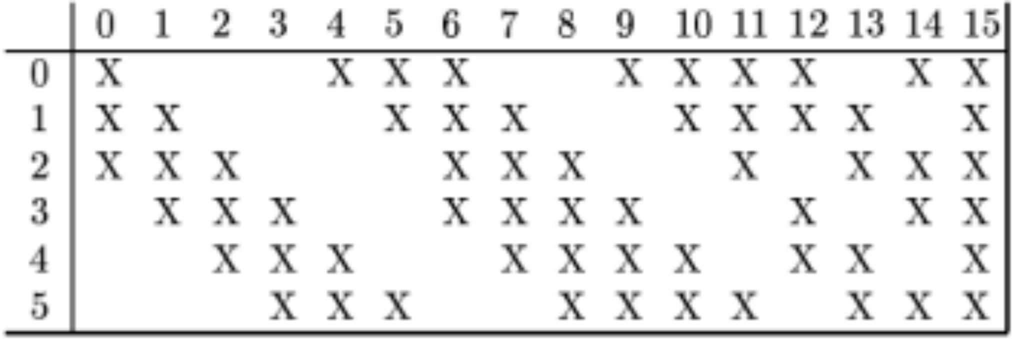
Profiling of Naive Implementation

- **Pooling**:

Forward	im2col()	1.647s
	argmax()	0.257s
	transformation	0.249s
Backward	im2col()	1.661s
	argmax()	0.219s
	transformation	0.480s
	col2im()	0.512s

Can we solve with Spark?

- Matrix Multiplication
- im2col()



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

Can we solve with Spark?

BALANCE BETWEEN COMMUNICATION COST AND EXECUTION TIME

Matrix size:

- A = (1000 * 32 * 32) * (5 * 5 * 3)
- B = (5 * 5 * 3) * 32
- **C = A * B**
- A + B = 1.08 * 10^10 bits ~ **1GB**

doesn’t work!!!!!

Calculation:

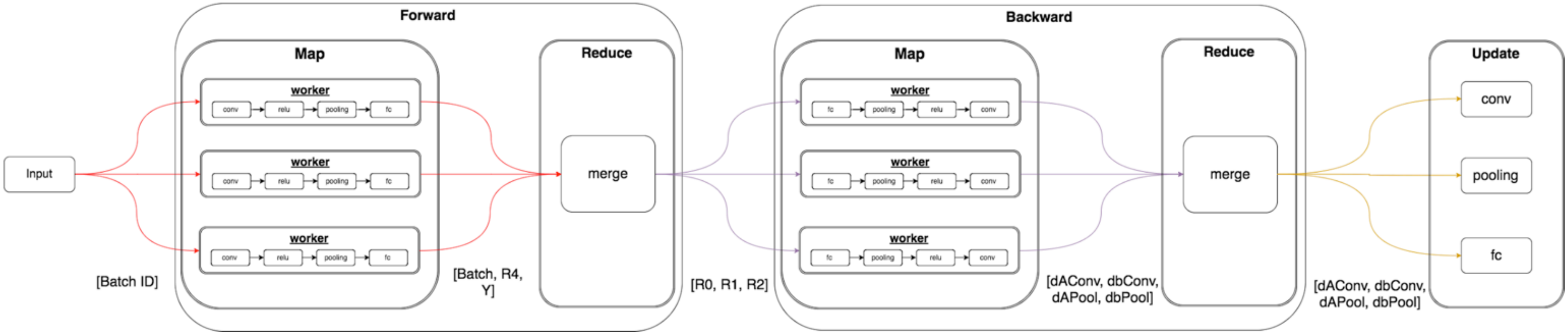
- NumPy.dot(): 0.672s
- Naive: O(n^3) - REALLY SLOW!
- Outer Product : TOO MUCH MEM!

Communication:

- Speed: high-speed network

Can we solve with Spark?

- Batch Processing



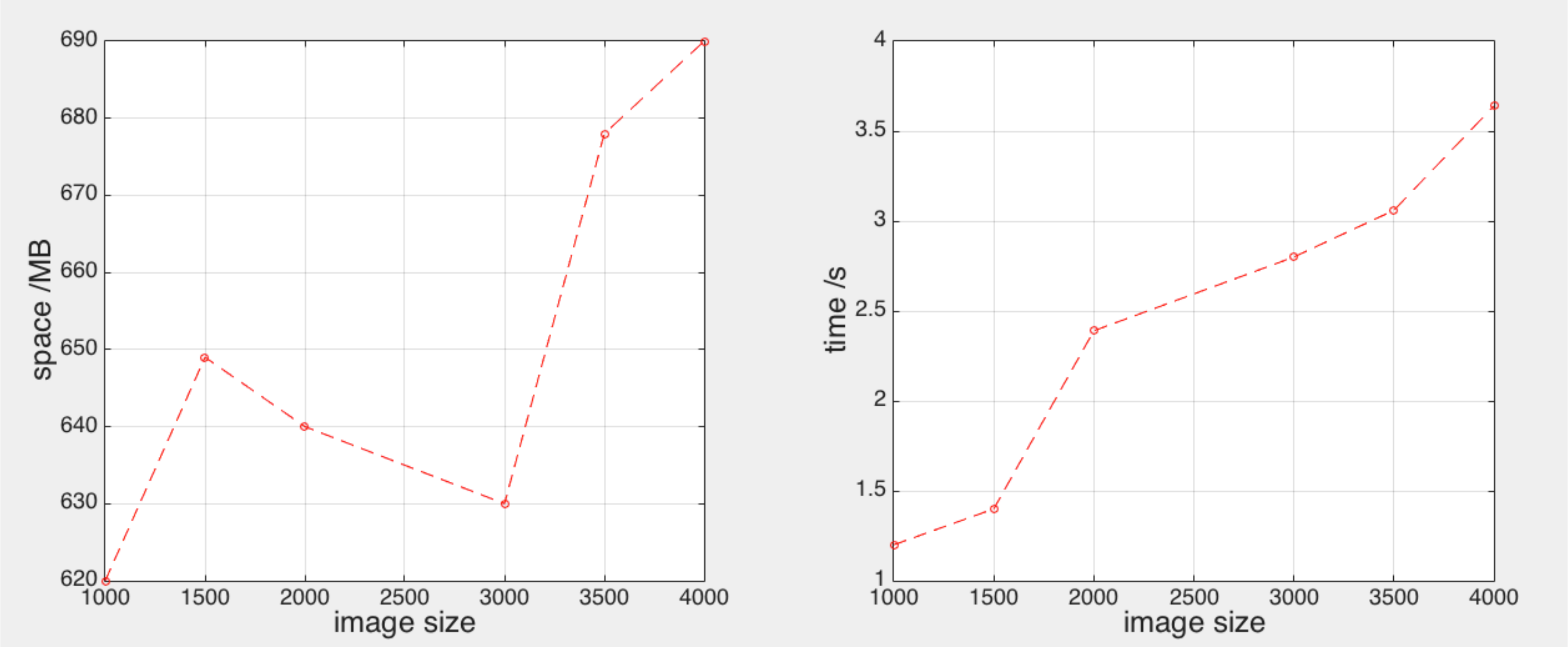
Spark Implementation

Time /s	Foward	Backward	Update	Total
NAIVE	2.315	4.512	2.503	6.833
HDFS	21.030	5.504	2.506	34.701

Problems Arise

Returning intermediate results from forward run and reuse them in backward run: transferring huge amount of data back and forth, and creates gigantic RDD for backward run. - doesn’t work too well with Spark

State of the Art: TensorFlow



Observations

- Parameter Tuning

- Deployment of Trained CNN

Future Work

- Make forward execution and backward propagation for each batch executed on the same worker to reduce communication cost. - awareness of locality
- Polling: ensure all batch accepted by nodes, handle failure
- Compare Spark-CNN performance with GPU-CNN.

Reference

- Image CNN:

  http://i.imgur.com/qMs50Ma.png
- Wiki Convolutional Neural Network:

  https://en.wikipedia.org/wiki/Convolutional_neural_network
- CS231n Convolutional Neural Networks:

  http://cs231n.github.io/convolutional-networks/

Q & A

Thanks !