HPC Challenges in Artificial Intelligence

Scalable Parallel Graph Search
and
Parallel Training of Deep Neural Networks

Kazuki Yoshizoe (美添 一樹)

RIKEN Center for Advanced Intelligence Project

RIKEN Center for Computational Science

IHPCSS2019@Kobe

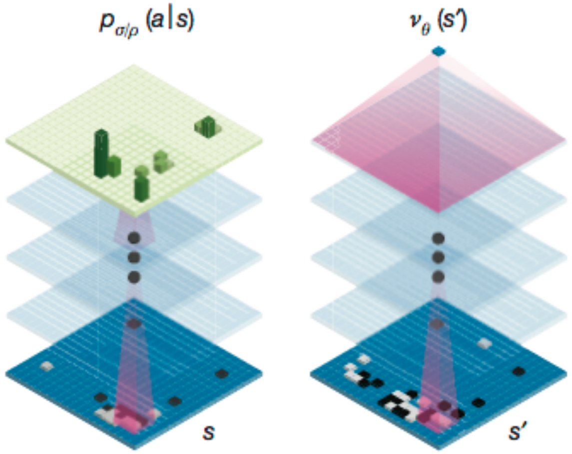
What is AlphaGo ?

A Go program developed by Google DeepMind which beat former and current Go champions

Go AI development

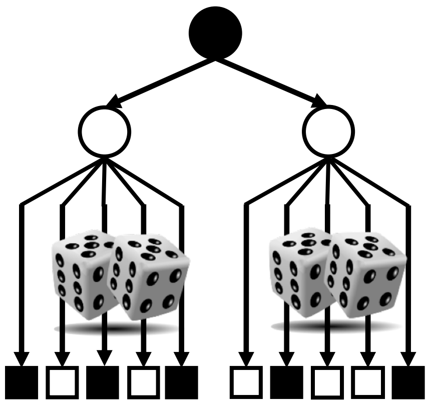
before 2005, 3kyu (weak amateur)
2006 breakthrough 1, MCTS
2011 breakthrough 2, DCNN
2016 Beat former champion
2017 beat top players 60-0
2017 super human without using human game records

Deep Learning
Recognize / Evaluate Go board
(applied to Go on 2014)



[Silver, Huang et al. 2016] Fig. 1b

MCTS
Monte-Carlo Tree Search
probabilistic tree search
(invented on 2006)



[Coulom 2006]

Reinforcement Learning
Learn from
State, Action, and Reward
(old invention, combined with DNN)

DQN

https://deepmind.com/research/dqn/

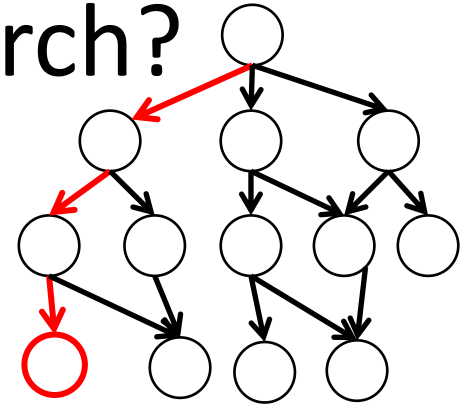
ALE

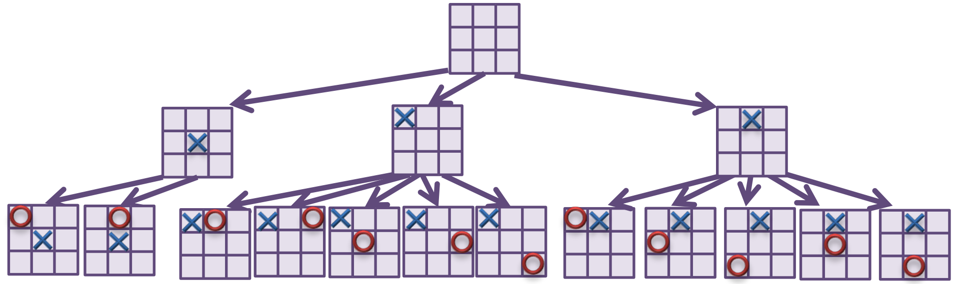
Arcade Learning Environment

https://github.com/mgbellemare/Arcade-Learning-Environment
https://www.youtube.com/watch?v=nzUiEkasXZI

Scalable Parallel Graph Search

What is Graph Search?





Graph Search finds

(Set of) node(s)
or path(s)

from a given Graph

Node or path shows
- ✓ "shortest path"
- ✓ "optimal combination"
- ✓ "best play in games"

Explicitly given Graph

Trains

Road map

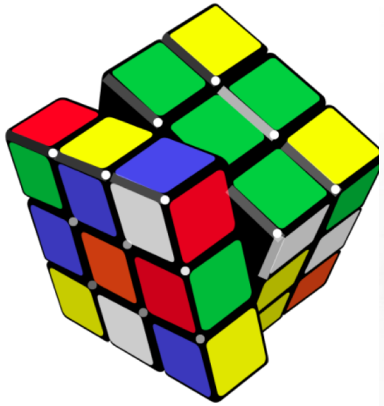
social network

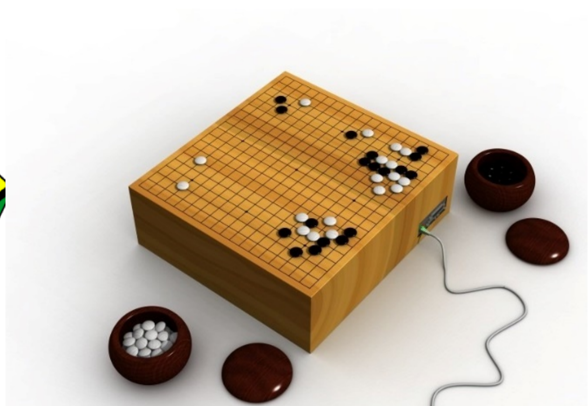
Generated Graph

Combinatorial optimization

Games

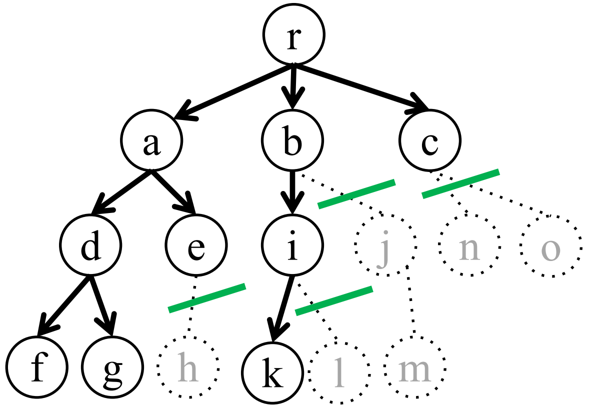
SAT, CSP



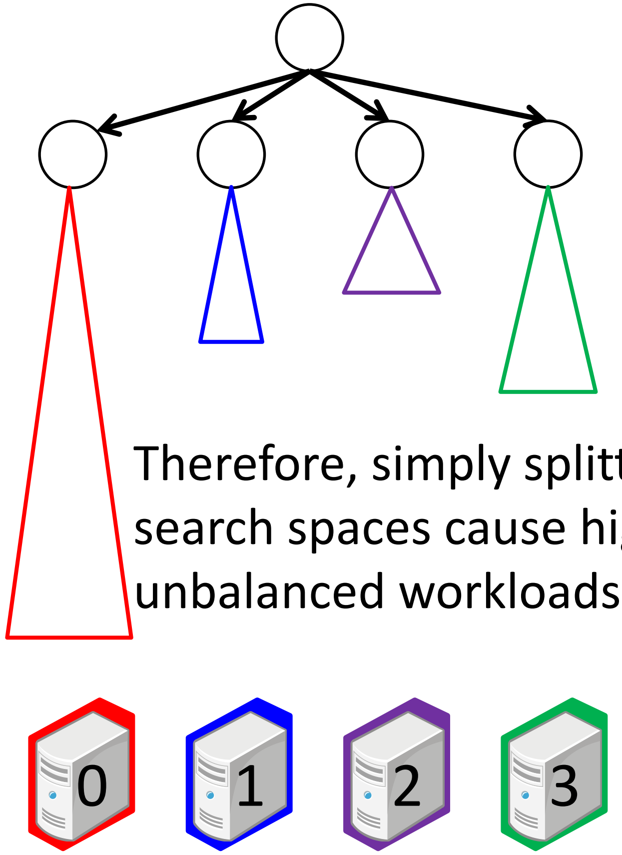


2003 nicolas p. rougier (CC BY-SA 4.0)

Parallelizing Search Algorithms



Practical search algorithms "prune" search spaces to focus on promising part.



Therefore, simply splitting search spaces cause highly unbalanced workloads

Parallel Depth-First Search (DFS) and applications

- Frequent Itemset Mining
- Statistical Pattern Mining
  [Yoshizoe, Terada, Tsuda 2018]
- Constraints Satisfaction
  [Ishii, Yoshizoe, Suzumura 2014]
- Continuous Optimizations
  [Izumi, Yoshizoe, Ishii 2018]

Parallel A* search and MCTS

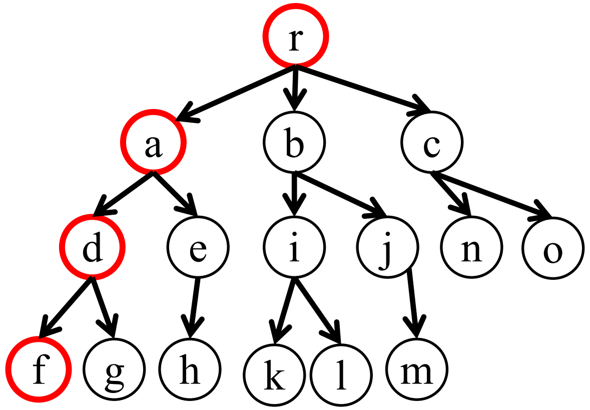
- Parallel A* using hash distributed data structure
- Parallel MCTS based on distributed tree and depth-first reformulation
  [Yoshizoe, et al. 2011]

Parallel Search Methods

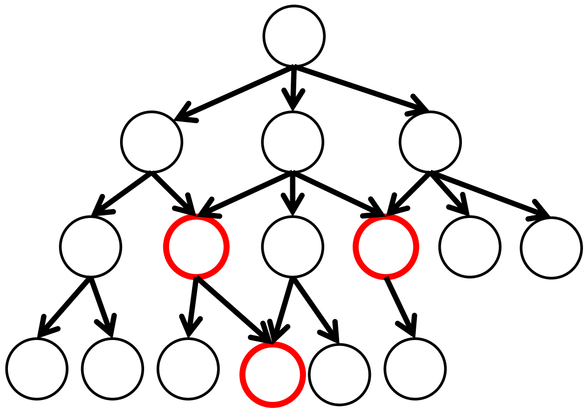
If each node is
visited at most once,
reformulate and do
work stealing

Simpler,
easier to parallelize

If nodes are
visited twice or more,
use hash distributed
data structures



Complex, but
more applications



Can be applied to
Depth-First Search (DFS)
and its applications

Can be applied to
Bellman-Ford, A* search,
and Monte-Carlo Tree Search

Depth-First Search Applications




database / items / transactions

	1	2	3	4	5	6
A	x	x	x	x	x	x
B		x	x		x	
C		x			x	
D	x	x		x	x	x
E		x		x		
F	x			x		x
G			x	x		x

Counting / Enumerating
Frequent **itemsets**
from a given database

ex. **itemsets** with freq. 3 or higher

$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{1,4\}, \{1,6\}, \{2,4\}, \{2,5\}, \{4,6\}, \mathbf{\{1,4,6\}}$

Frequent Itemset Mining


ex1. Market Basket Analysis
items: products
trans.: customers
x: purchased items

Fundamental problem in data mining

Statistical Pattern Mining

ex2. Genomics (GWAS)
items: SNPs
trans.: human
x: SNP

...GTCTAAAACATGATT...
...GTCTGAATCATGATT...
...GTCTGAAACATGATT...
...GTCTGAATCATCATT...

SNP: Single Nucleotide Polymorphism
GWAS: Genome-Wide Association Studies

Finding combination of multiple SNPs
(not one or two SNPs)

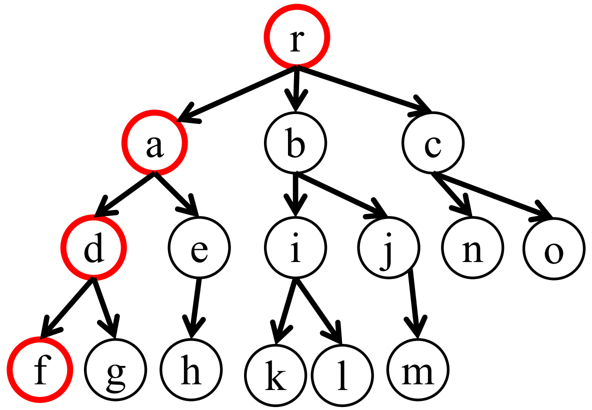
[Yoshizoe, Terada, Tsuda 2018] Bioinformatics

Depth First Search (w/o threshold)

Back tracking DFS

```
DFS() {
  Recur(r)
}
Recur(node n) {
   foreach (child c of n) {
      // do something for c
      Recur(c)
   }
}
```

Simply traverses
all nodes in the tree



back tracking can be naturally
implemented with *recursive* call

Memory usage $O(d)$
Only current path is needed

Frequent Itemset Mining can be solved using DFS w/o threshold

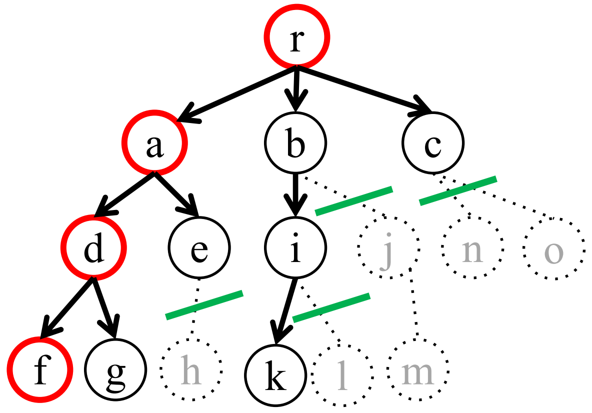
Depth First Search with threshold update

DFS with threshold

```
DFS() {
  Recur(r)
}
Recur(node n) {
    foreach (child c of n) {
       // do something for c
        if (c is within threshold) Recur(c)
       UpdateThreshold()
    }
}
```

Prune search space by
dynamically updating threshold

Update threshold during search.
More branches are pruned in the right.
(Search progresses from left to right.)



Ex. finding top-k nodes

Statistical Patten Mining can be implemented in DFS with threshold.
Significance threshold is updated and propagated.

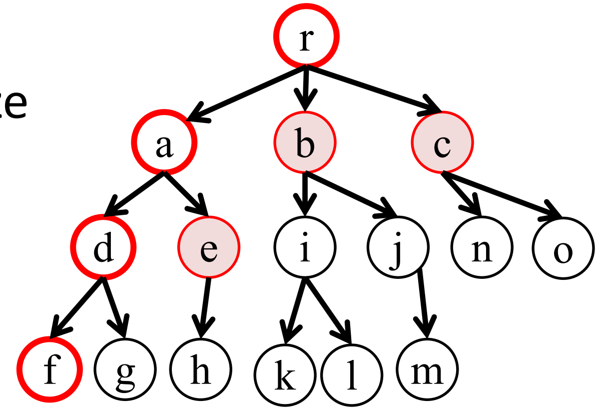
Parallel DFS, preparation

```
DFS() {
 Recur(r)
}
Recur(node n) {
   foreach (child c of n) {
      // do something for c
       if (c is within threshold) Recur(c)
      UpdateThreshold()
   }
}
```

convert recursion to stack + loop

```
StackDFS() {
   push(r)
   Loop()
}
Loop() {
   while(stack not empty) {
      pop n from stack
      foreach (child c of n) {
         // do something for c
           if (c is within threshold) push(c)
         UpdateThreshold()
      }
   }
}
```

pros: O(d) memory
cons：difficult to parallelize



cons: O(db) memory
pros: easy to parallelize

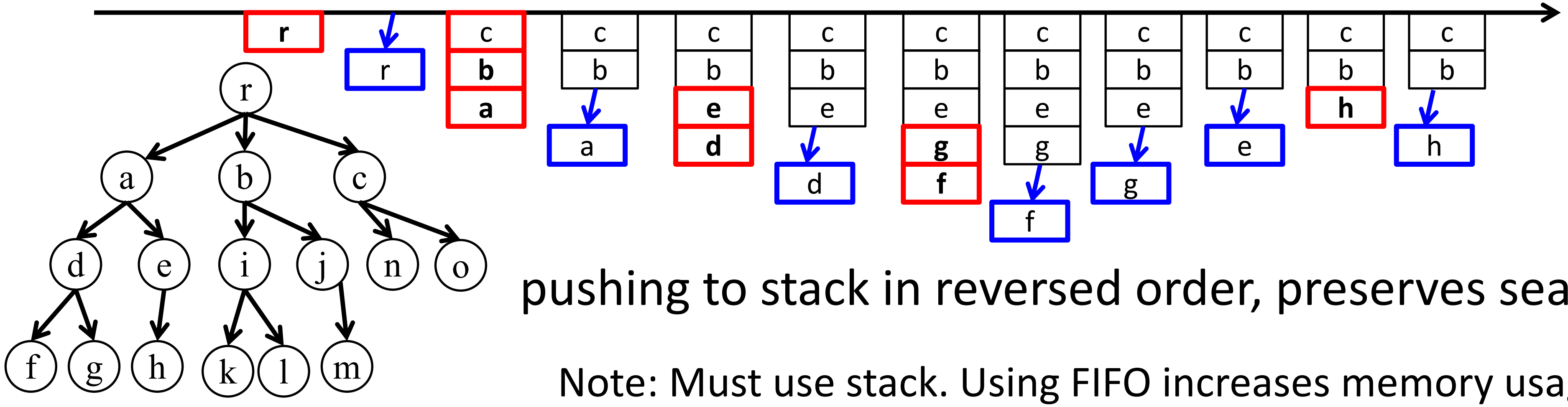
For depth d, branch nu. b search space

```
DFS() {
 Recur(r)
}
Recur(node n) {
   foreach (child c of n) {
      // do something for c
      if (c is within threshold) Recur(c)
      UpdateThreshold()
   }
}
```

Convert recursive call
to stack + loop

```
StackDFS() {
   push(r)
   Loop()
}
Loop() {
   while(stack not empty) {
      pop n from stack
      foreach (child c of n) {
         // do something for c
         if (c is within threshold) push(c)
         UpdateThreshold()
      }
   }
}
```

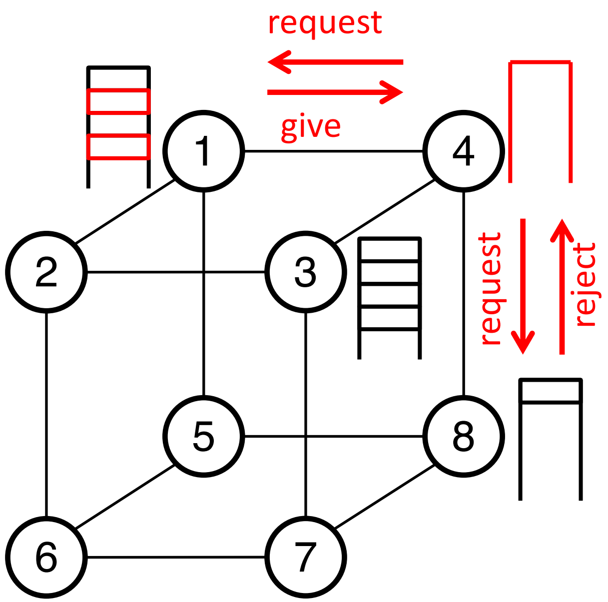
foreach in reverse order



pushing to stack in reversed order, preserves search order

Note: Must use stack. Using FIFO increases memory usage.

Work Stealing based parallelization



Steal work from "victim"

Receiver initiated Work stealing

Workers with empty stack (empty job)
1, Select a victim worker
2, Send job request to the victim
3, The victim gives jobs if available. Rejects otherwise
(details are omitted)

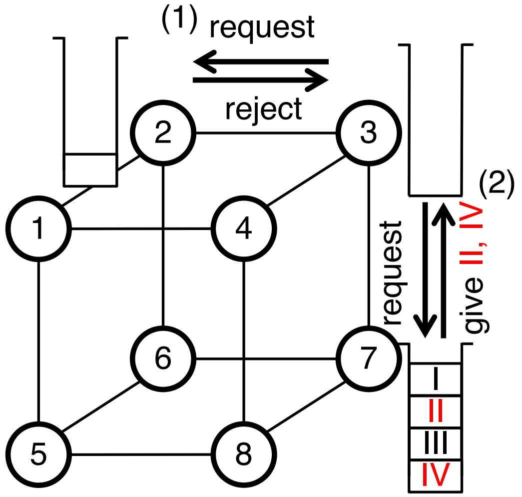
Simple method for victim selection
"Select randomly"

A better method
Select victims from neighbors on hypercube
(virtual hypercube is prepared ignoring actual topology)

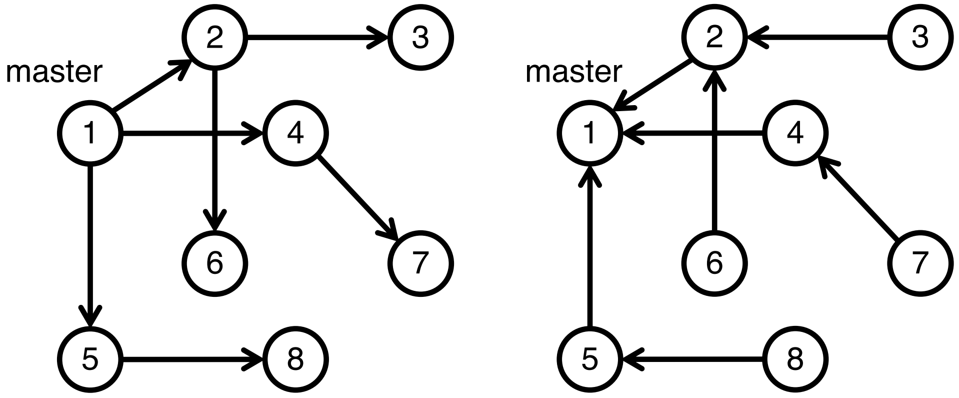
Lifeline graph [Saraswat et al. 2011]

Threshold Broadcast / Reduce and DTD

Distributed Termination Detection



Work stealing
on hypercube



Threshold broadcast/reduce and DTD
on spanning tree

Applied DTD on spanning tree [Mattern 1990]

Proof is needed to confirm all stacks are empty
in distributed environment (details omitted).

This slides is just
for references

Massive Parallel Statistical Pattern Mining

Frequent Itemset Mining based on Closed Itemset

[Pasquier, Bastide, Taouil, Lakhal 1999]

Apply reverse search technique
(**LCM algorithm**) [Avis, Fukuda 1996]

[Uno, Kiyomi, Arimura 2004]

Applied to Statistical Pattern Mining
LAMP algorithm

[Terada, Okada-Hatakeyama, **Tsuda**, Sese, 2014]

Faster **LAMP** using DFS with threshold

[Minato, Uno, **Tsuda**, Terada, Sese 2014]

Massive Parallel LAMP (MP-LAMP)

[**Yoshizoe**, Terada, **Tsuda** 2018]

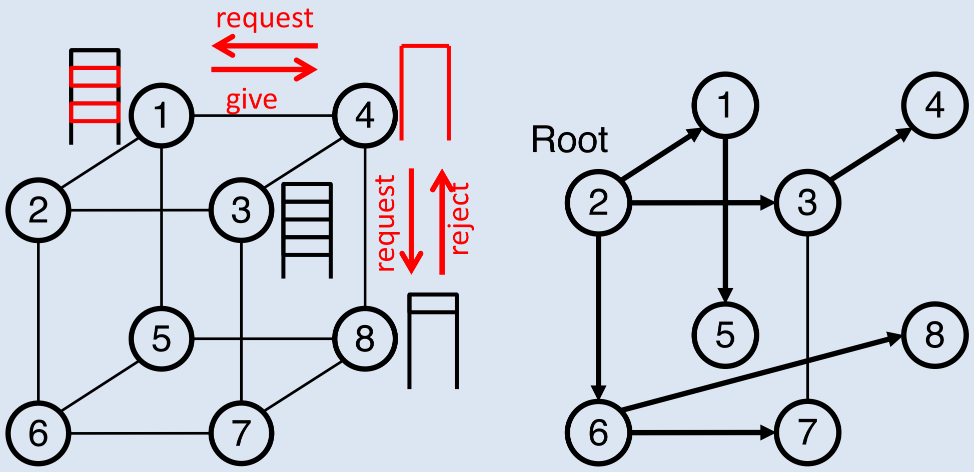
For solving GWAS and others

Parallelization Method

Reformulate algorithm from recursive call to stack + loop

hardware/middleware aware algorithm and implementation

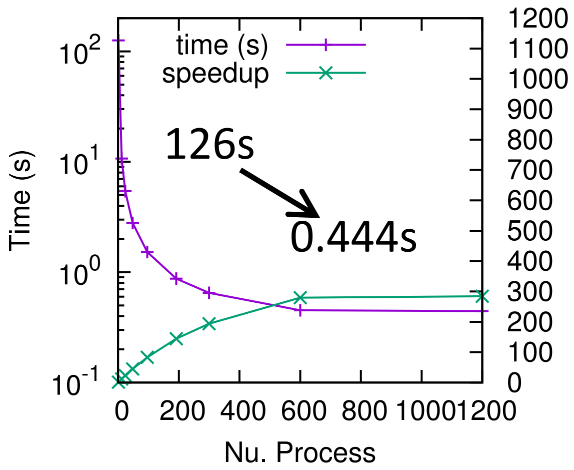
Work stealing and broadcast/reduce



Statistical Pattern Mining: Speedup

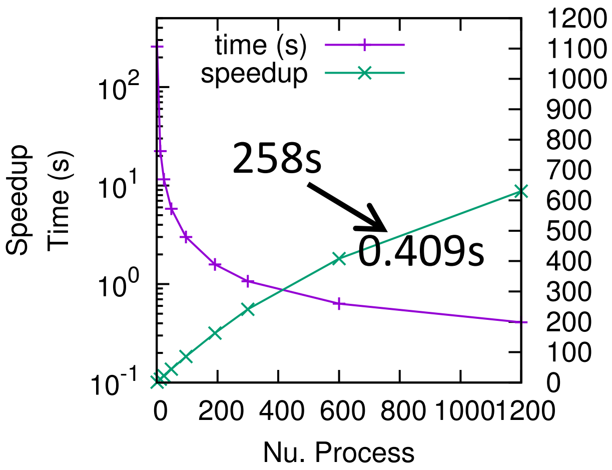
JPN dom. 10%

item: 11253, trans:697, dens:1.0%



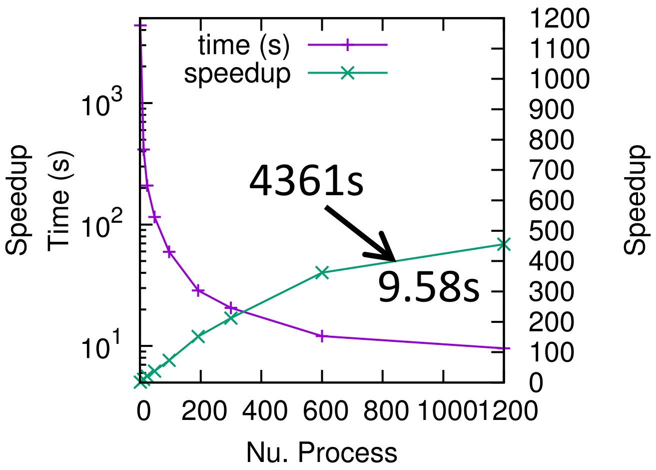
Alz. dom. 5%

item:44052, trans:364, dens:5.4%



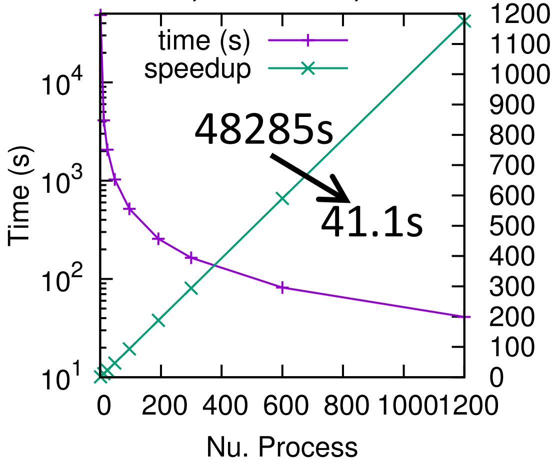
Alz. rec. 30%

item: 250120, trans:364, dens:2.9%



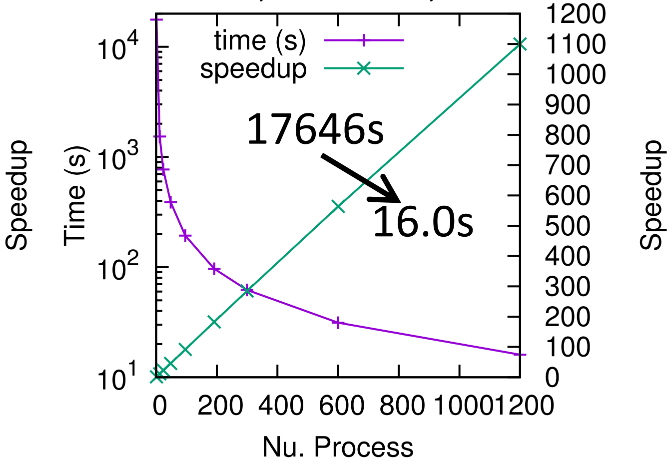
JPN dom. 20%

item:11914, trans:697, dens:1.9%



Alz. dom. 10%

item: 91126, trans: 364, dens:9.8%



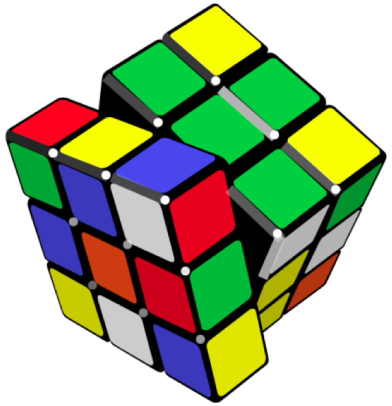
Speedup for Finding combination of SNPs related to Alz. or Japanese

On K-Computer

Used Max. **140K cores,**

110-120K-fold estimated speedup in the best case (unpublished)

A* search and MCTS



"God's number is 20"

Shortest path search

A* search

(pronounced "A star")

Dijkstra's algorithm

+ **heuristic**

50 years old

MCTS

Monte Carlo Tree Search

Random sampling

based search

invented on 2006

material science

scheduling

[Cazenave, Balbo, Pinson 2009]
"Monte-Carlo bus regulation"

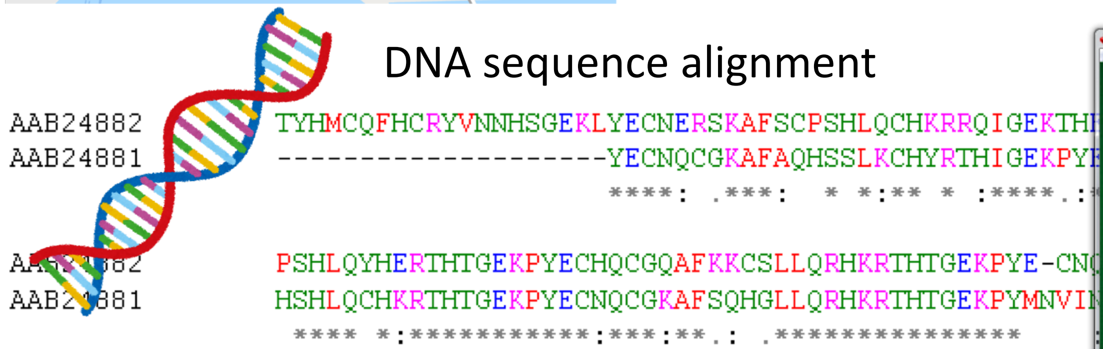
[Tanabe, Yoshizoe, and Imai 2009]
"A study on security evaluation methodology for image-based biometrics authentication systems"

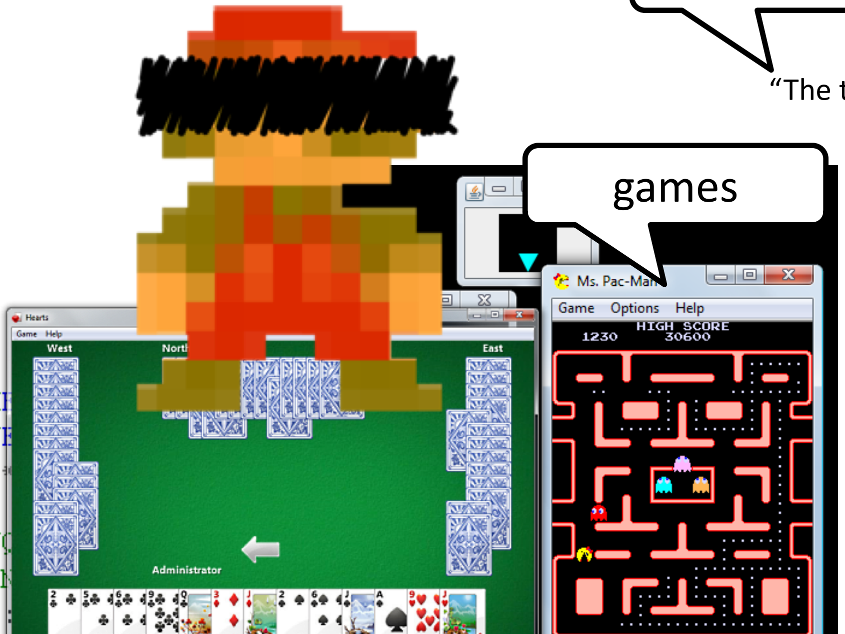
NLP

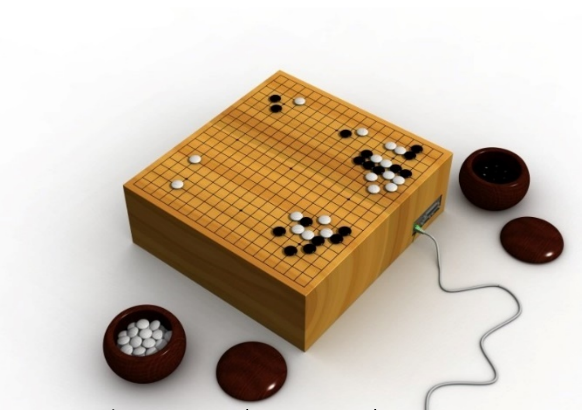
biometric security

[Chevelu, Putois, Lepage 2010]
"The true score of statistical paraphrase generation"

DNA sequence alignment

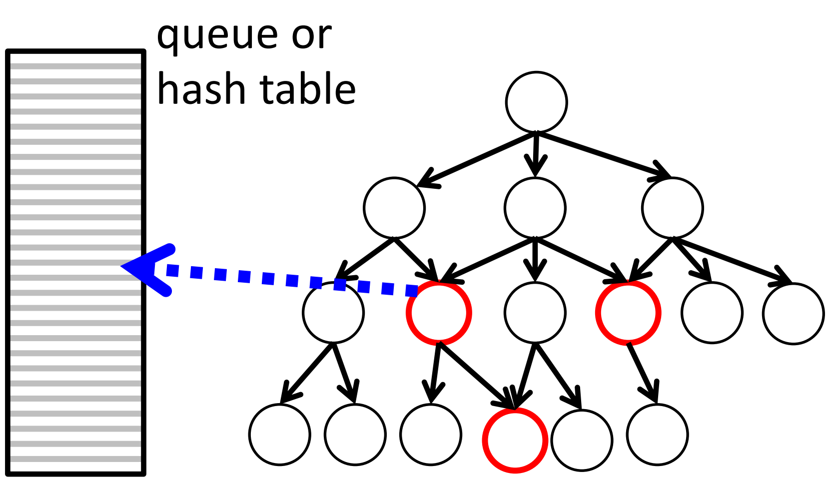




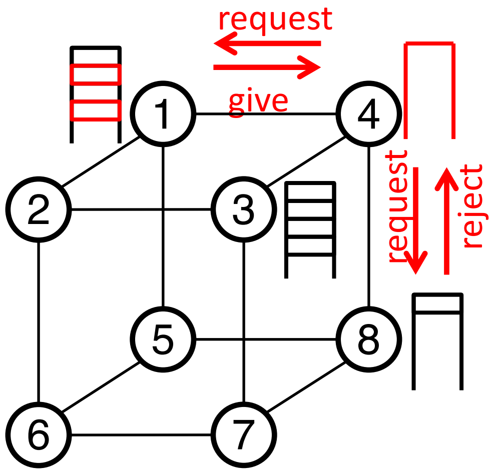


2003 nicolas p. rougier (CC BY-SA 4.0)

What's Needed for Non-Depth First Search?




```
DFS_Recur(node n) {
    foreach (child c of n) {
        // do something for c
         DFS_Recur(c)
    }
}
```



Nodes can be visited multiple times
Result are recorded and reused later

using either
- Priority Queue (A* search)
- Hash Table (MCTS, IDA*)

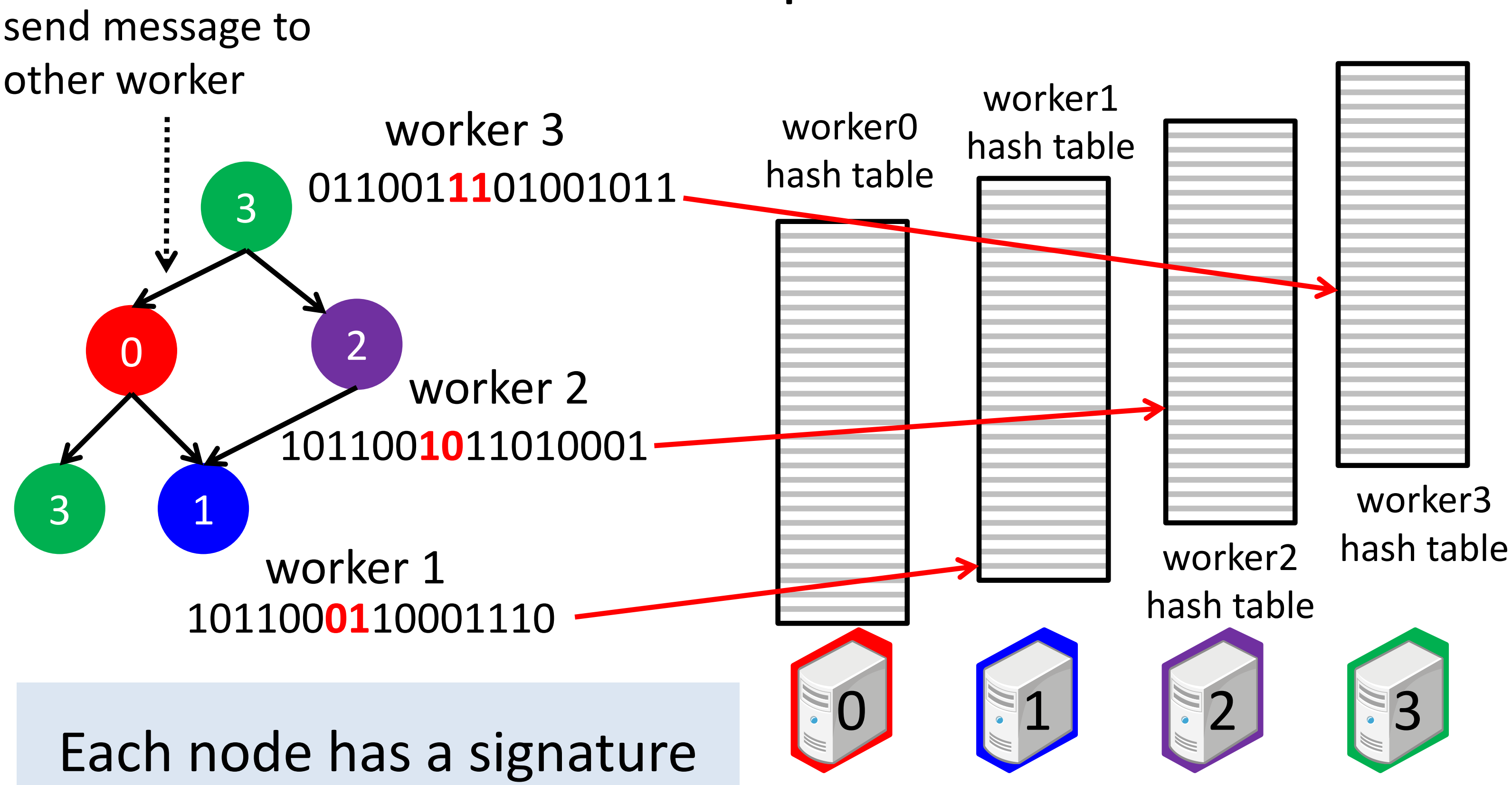
```
NonDFS(node n) {
    while(not_finished) {
        ReadFromTable(n)
        foreach (child c of n) {
            // do something for c
        }
        WriteToTable(n)
    }
}
```

?

Distributed Hash Table
Distributed Queue

Distributed Hash Table Driven Parallelization

Transposition table Driven Scheduling [Romein et al. 1999]



Each node has a signature

Uniform load balancing

tradeoff

Frequent 1-to-1 comm.

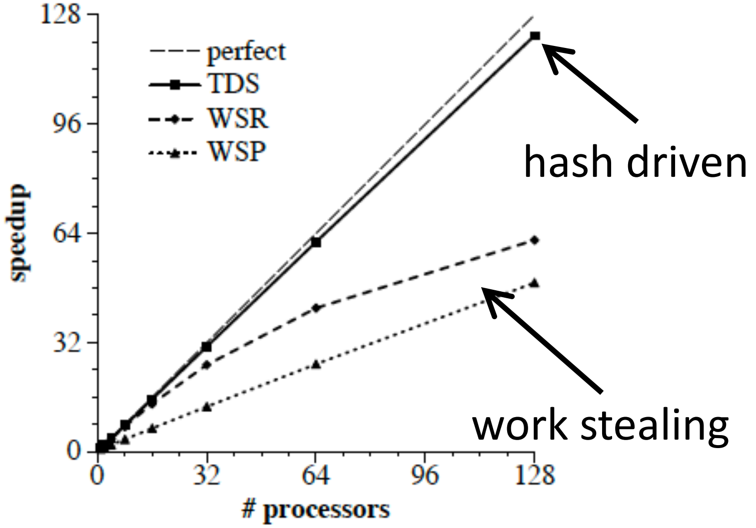
Part of the signature shows the “home worker”

workers sends messages to home worker of children

signature is calculated by a hash function

Hash driven Parallel Search Performance

TDS algorithm
(Parallel IDA*)



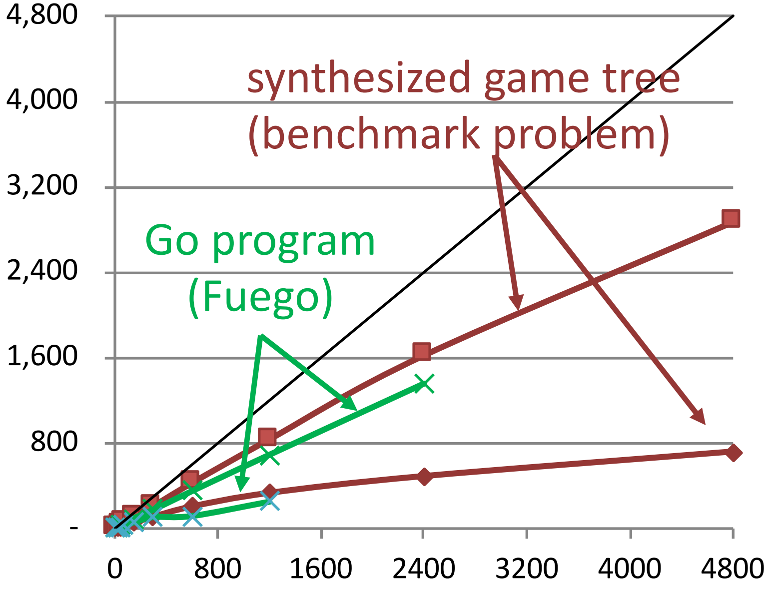
[Romein+ 1999] Fig. 4 (c)

HDA*
(Parallel A*)

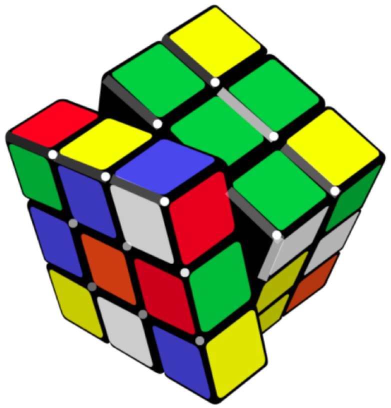
Applied to puzzles,
planning, and sequence
alignment

[Kishimoto+ 2012]

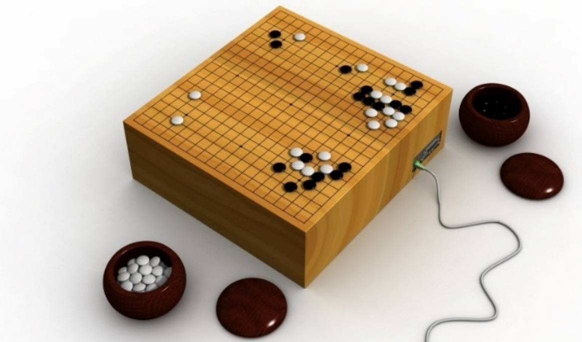
TDS-df-UCT algorithm
(Parallel MCTS)



[Yoshizoe+ 2011]

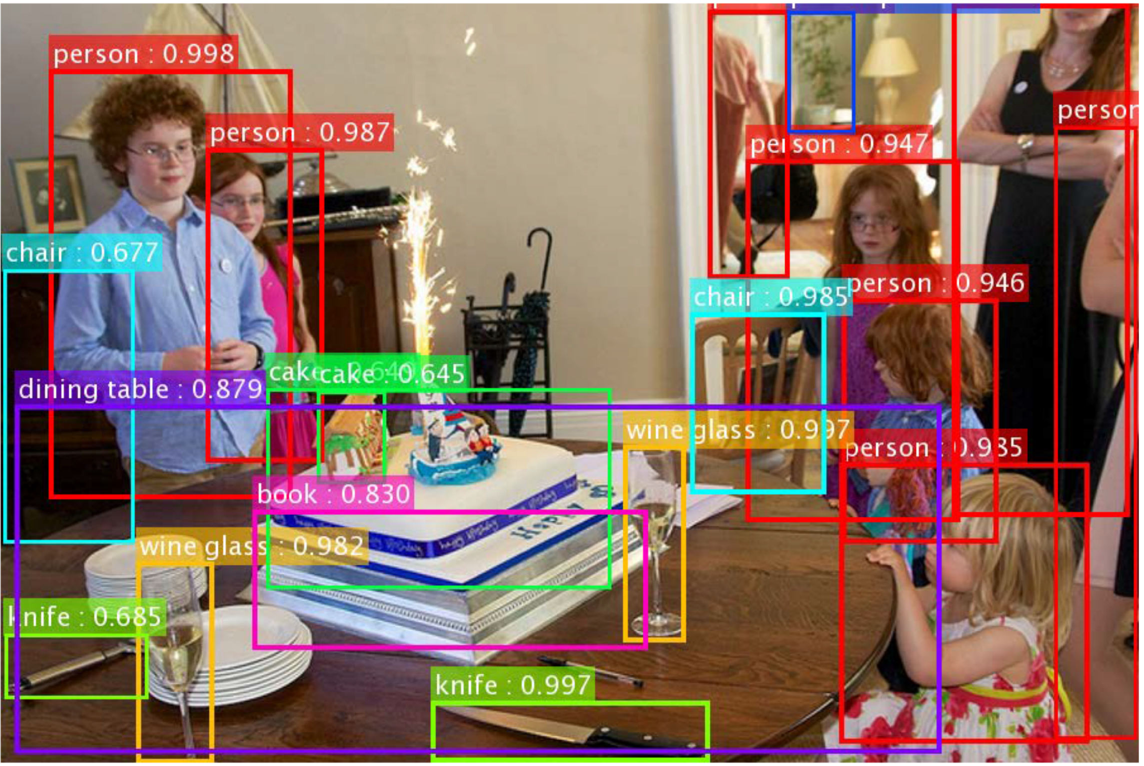


Note: These performances are
achieved if communication congestion
is removed by reformulations of algorithms



Parallel Training of Deep Neural Networks

What Deep Learning can do?



[K. He et al. 2015, Microsoft Research Asia]

Image recognition by ***ResNet*** model

Won ILSVRC (ImageNet Large Scale Visual Recognition Challenge) in 2015. The goal is to recognize 1,000 object types

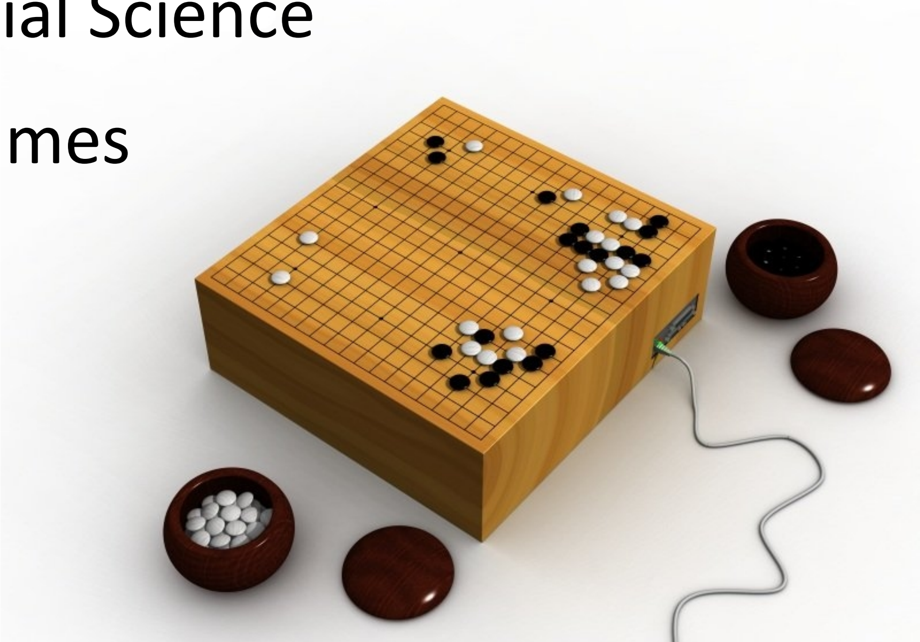
Image recognition

Natural Language Processing

Sound / Voice recognition

Material Science

Games

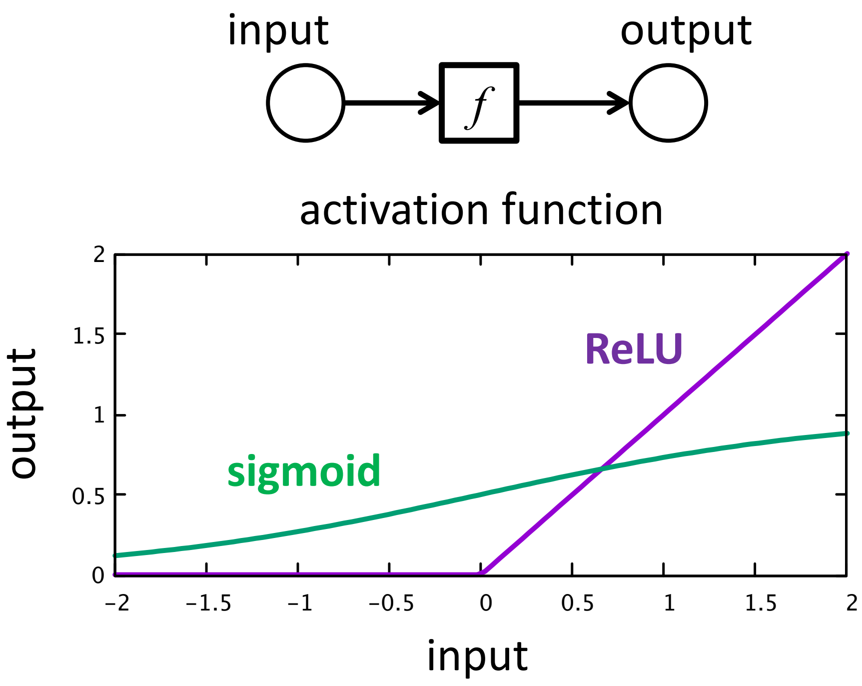


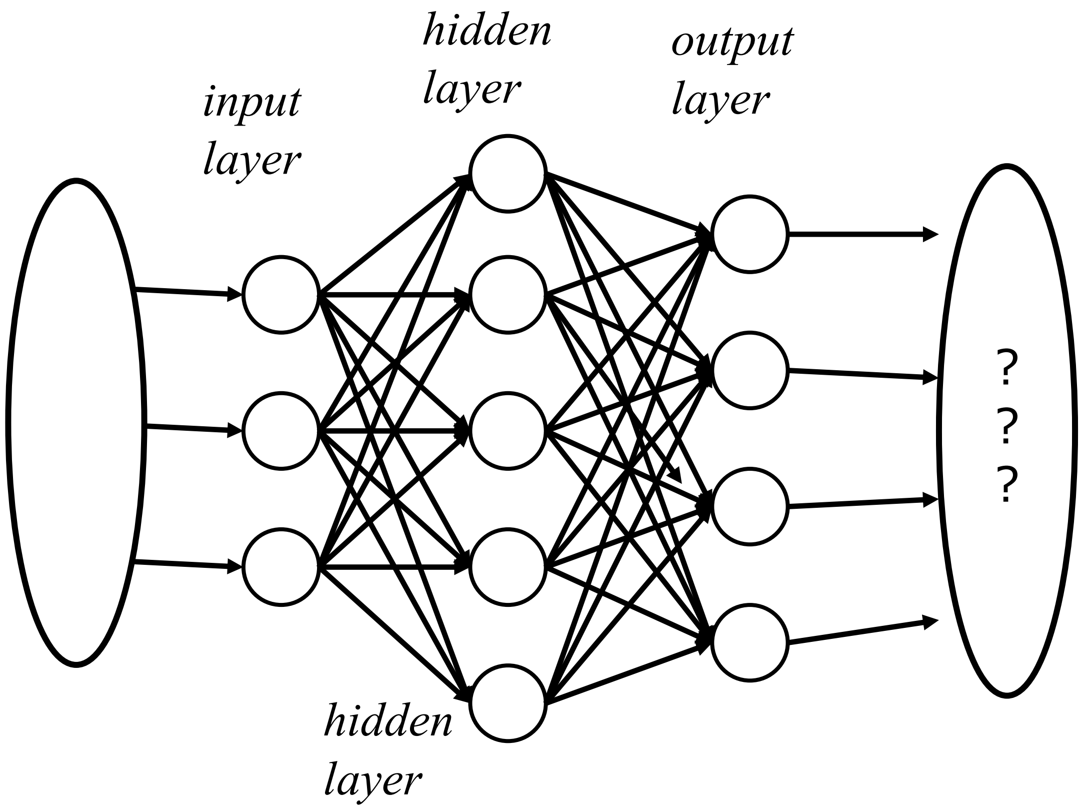
Shallow

What is **Neural Network?**

A algorithm inspired by mechanism of neurons

A neuron outputs
- small value for small input
- large value for large input





Convolutional Filters for Images

Multiply and add surrounding pixel values

Original image

Examples of filters

blur

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

edge detect

0	1	0
1	-4	1
0	1	0

emboss

-2	-1	0
-1	1	1
0	1	2

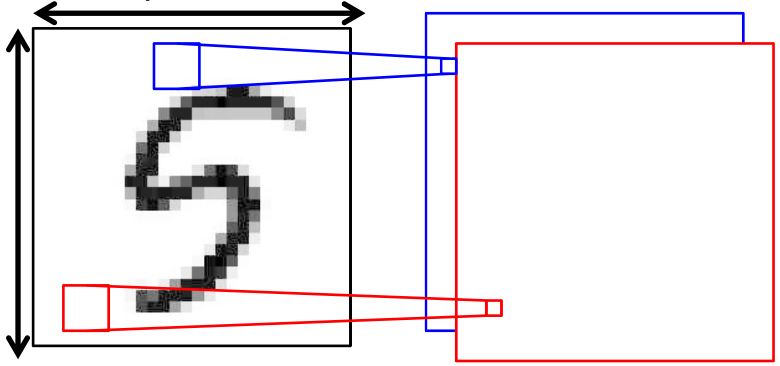
Examples are from the manual of GIMP

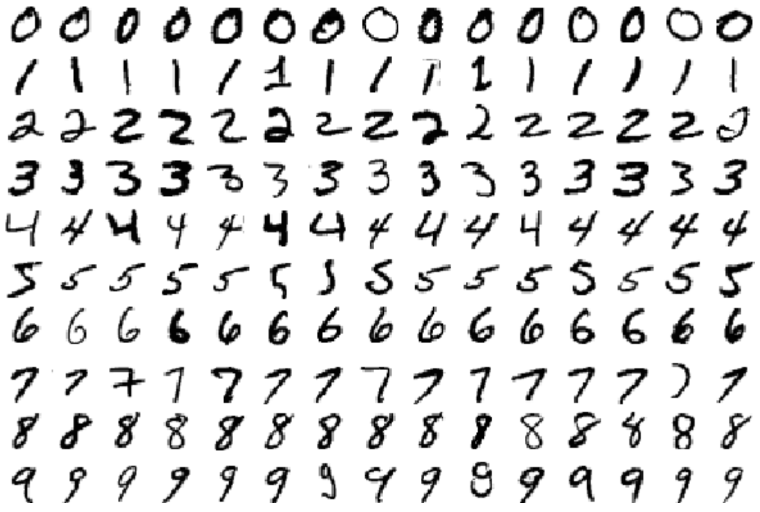
8.2. Convolution Matrix http://docs.gimp.org/en/plug-in-convmatrix.html

Many types of operations are possible by adjusting filters' weights and size

Neural Networks can calculate Convolutional Filters

CNN: Convolutional Neural Network





Famous benchmark

MNIST handwritten digit database
http://yann.lecun.com/exdb/mnist/

vertical line filter (3x3)

-1	2	-1
-1	2	-1
-1	2	-1

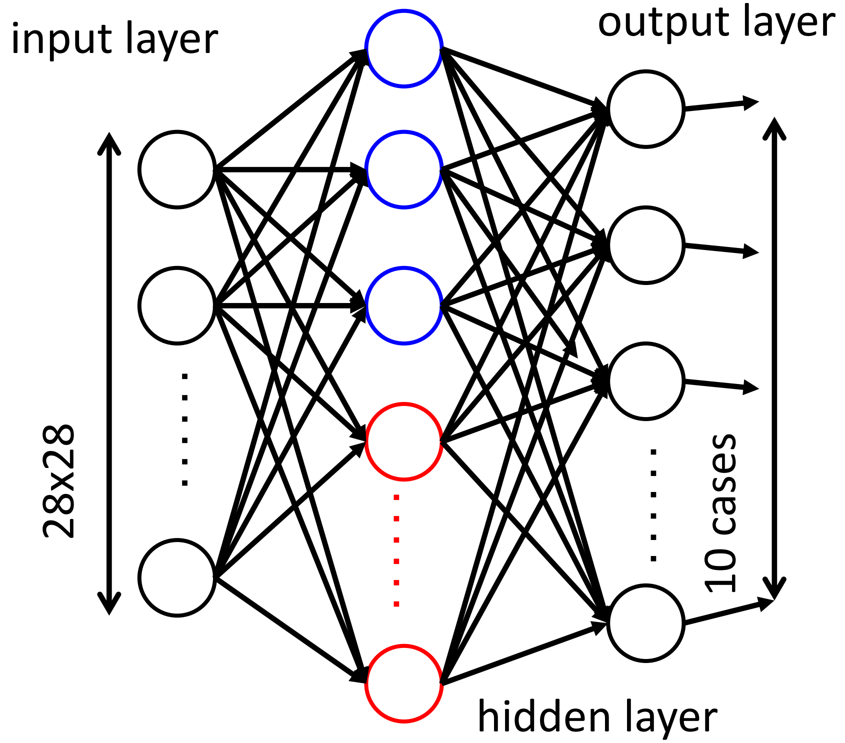
horiz. line filter (5x5)

0	0	0	0	0
-1	-1	-1	-1	-1
2	2	2	2	2
-1	-1	-1	-1	-1
0	0	0	0	0

corner filter (3x3)

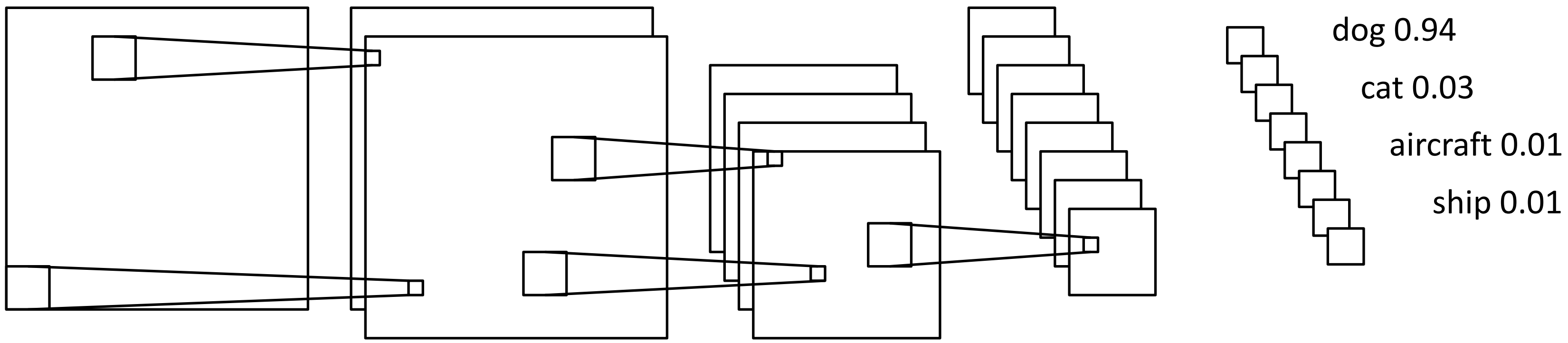
1	1	1
-1	-1	1
-1	-1	1

Three layer CNN can recognize numbers if filters are adjusted.



I made these filters up in my head

DCNN: Deep Convolutional Neural Network



Complex shape can be recognized with multiple layers of simple filters
(e.g. edge recognition followed by line detection)

An example is the "cat neuron" found in DCNN for image recognition (by google)
https://googleblog.blogspot.jp/2012/06/using-large-scale-brain-simulations-for.html

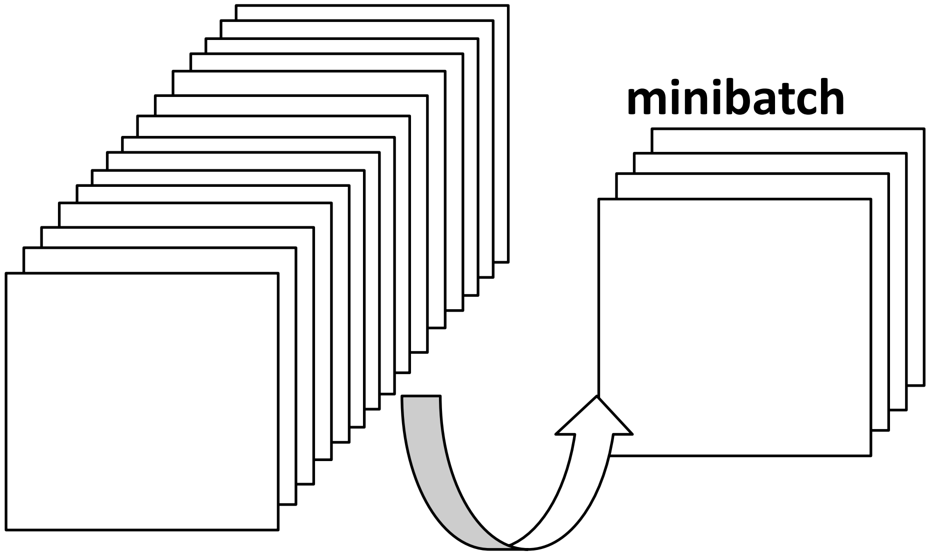
ResNet-152 has 152 layers [K. He et al. 2015]
(ResNet-50 has 50 layers)

"Cat neuron"

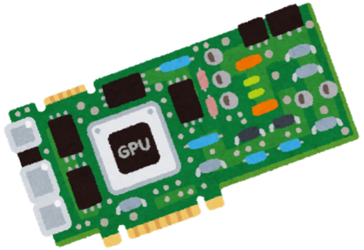
Training of Neural Network

training data

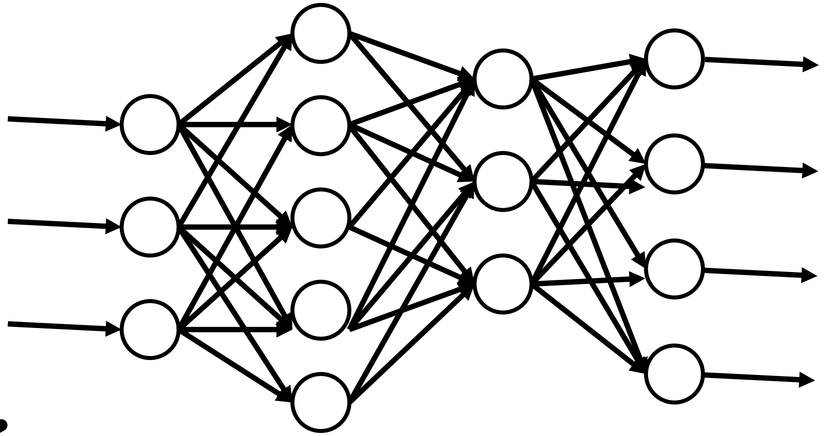
minibatch



shuffle



forward (inference, prediction)



compare with training data

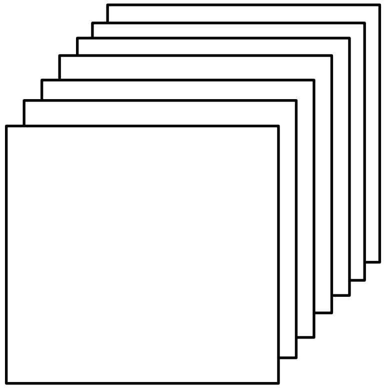
calculate **error** E

backward (backpropagation)

Calculate gradient $\frac{\delta E}{\delta W}$ for each connection

update weights

validation data



Repeat until validation error stops improving

Learning Curve Example and Learning Rate

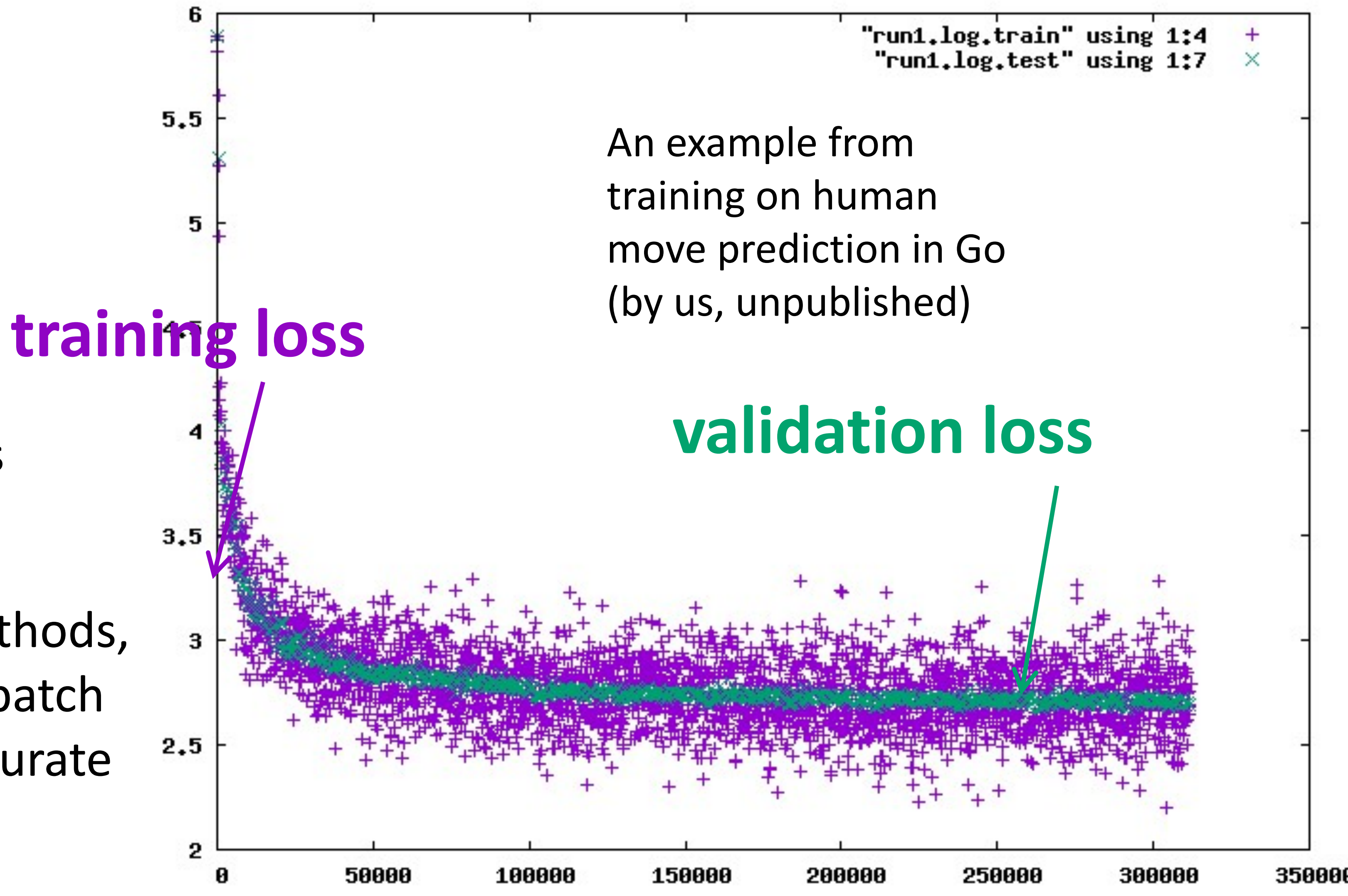
Minibatch size is small.
Typically 32 – 256.

So, losses are very noisy,
small “learning rate” is used.

$$update = lr \cdot grad$$

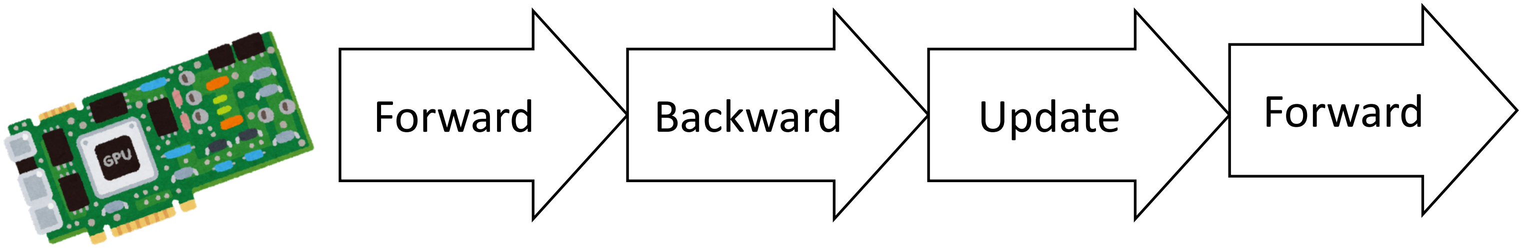
Initial learning rate is 0.01 or less
and gradually decreased.

For non-DL machine learning methods,
higher LR can be used for larger batch
because the gradient is more accurate
but …

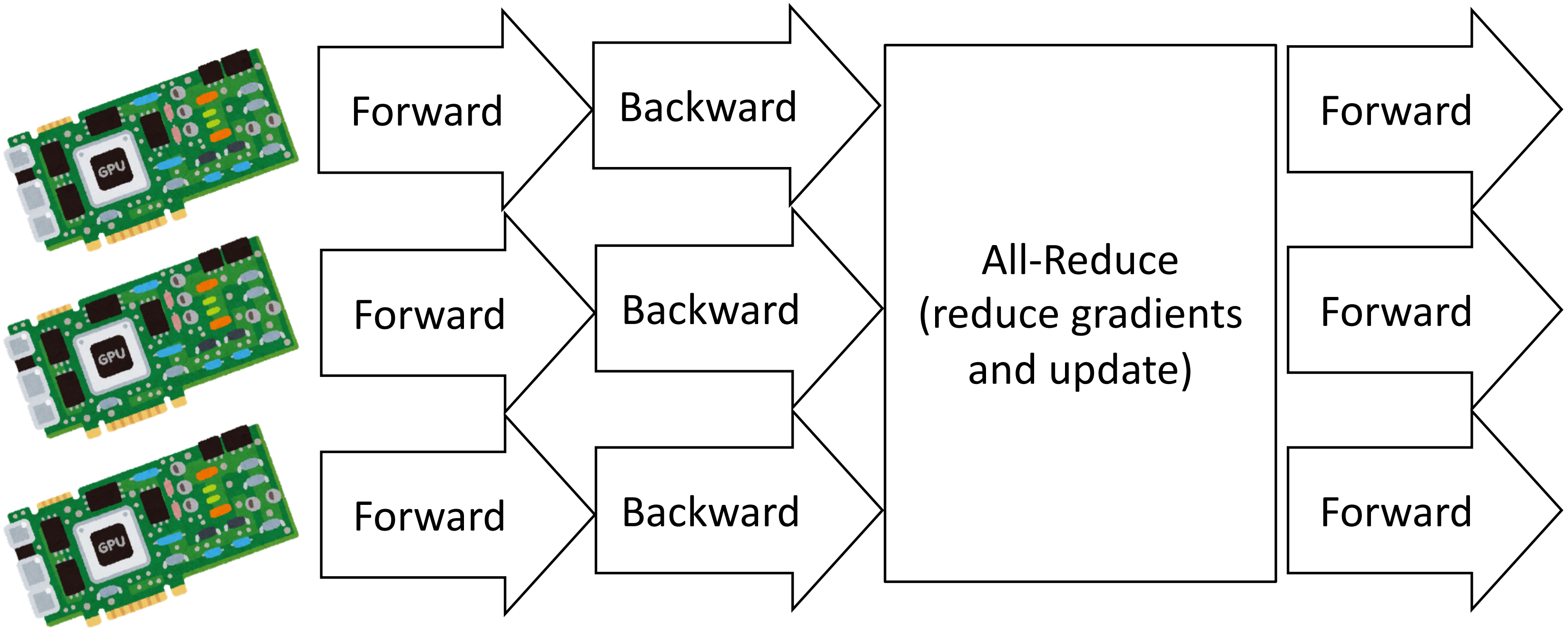


Training: Single GPU, Multi-GPU

Single GPU



Multi GPUs



Note: It is a “synchronous” approach. “asynchronous” approach is omitted because it‘s simply worse.

Large Batch Problem

[Hoffer+ 2017] arXiv:1705.08741

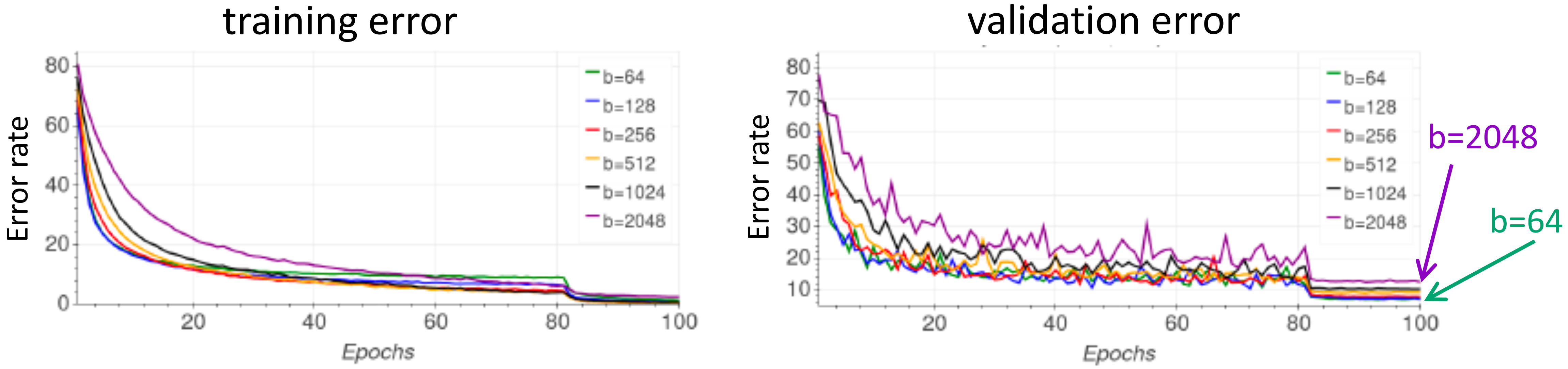
Train longer, generalize better: closing the generalization gap in large batch training of neural networks

Around 100 GPU is the limit of the simple approach. Why?

Larger batch results in greater validation error! (long known phenomenon [Lecun+ 1998])

Synchronous parallel training makes the batch size greater (N-fold for N GPUs)

The paper partly solved it, but not enough for larger scale parallelization.



[Hoffer+ 2017] Figure 1

Two years ago, June 2017, the record was 4.4hour using 128GPUs

Training ResNet-50 for ImageNet benchmark

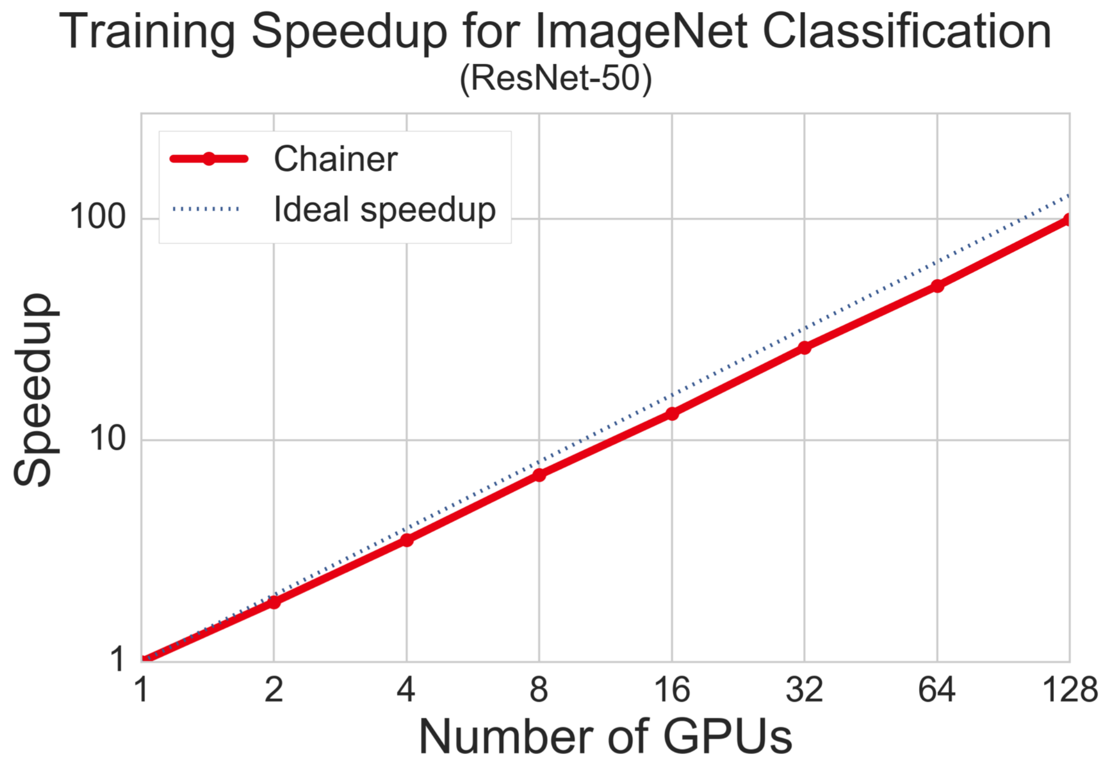
[Akiba 2017]
© Preferred Networks, inc.

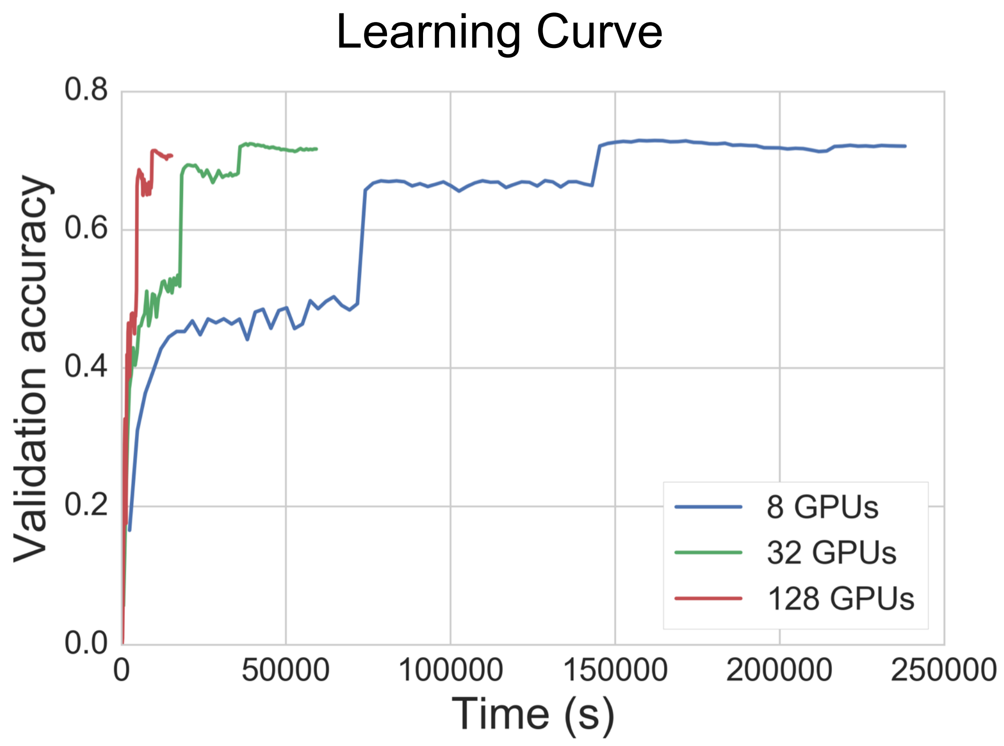
Chainer (a DL framework)

NCCL (Nvidia Collective Comm. Library)

CUDA Aware MPI (uses GPUDirect)

mpi4py (MPI for python)





4 GPUs x 32 nodes = 128 GPUs
NVIDIA GeForce Titan X (Maxwell)

ChainerMN: https://github.com/chainer/chainermn
Performance of Distributed Deep Learning using ChainerMN
https://chainer.org/general/2017/02/08/Performance-of-Distributed-Deep-Learning-Using-ChainerMN.html

ResNet-50 ImageNet Training Records

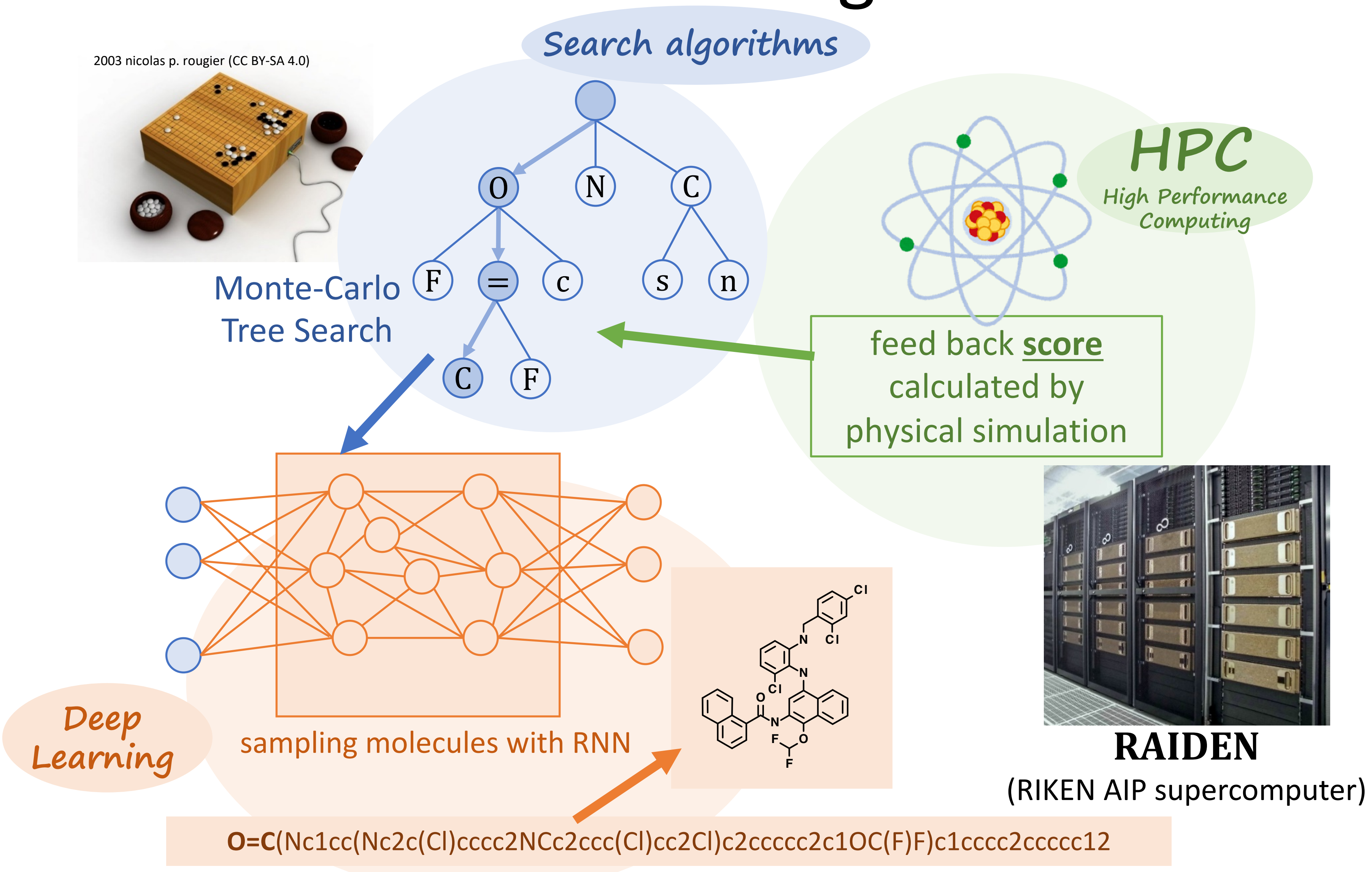
Date	Authors	Main Organization	GPU/TPU	Batch size	Time	Accuracy
Jun. 2017	Goyal et al.	Facebook	P100 x 256	8,192	1 hr.	76.3%
Nov. 2017	Akiba et al.	Preferred Networks	P100 x 1024	32,768	15 min.	74.9%
Jul. 2018	Jia et al.	Tencent Inc.	P40 x 2048	65,536	6.6 min.	75.8%
Nov. 2018	Mikami et al.	Sony	V100 x 2176	34K → 68K	3.8 min.	75.03%
Nov. 2018	Ying et al.	Google	1024chip TPUv3	32,678	2.2 min.	76.3%
Mar. 2019	Yamazaki et al.	Fujitsu	V100 x 2048	81,920	74.7 s.	75.08%
	Osawa et al.	Titech, NVIDIA, RIKEN	----	131,073	----	75.0%

Key techniques: warm start, efficient gradient distribution, hyperparameter tuning, and 2nd order optimization

[Goyal et al.] https://arxiv.org/abs/1706.02677
[Akiba et al.] https://arxiv.org/abs/1711.04325
[Jia et al.] https://arxiv.org/abs/1807.11205
[Mikami et al.] https://arxiv.org/abs/1811.05233
[Ying et al.] https://arxiv.org/abs/1811.06992
[Yamazaki et al.] https://arxiv.org/abs/1903.12650
[Osawa et al.] https://arxiv.org/abs/1811.12019

Introducing one of our recent work

discover new molecules using Search + DL + HPC



References

X. Yang, J. Zhang, K. Yoshizoe, K. Terayama, K. Tsuda. "ChemTS: an efficient python library for de novo molecular generation". Science and Technology of Advanced Materials (STAM), 2017 Dec 31;18(1):972-6.

M. Sumita, X. Yang, S. Ishihara, R.Tamura, and K. Tsuda. "Hunting for Organic Molecules with Artificial Intelligence: Molecules Optimized for Desired Excitation Energies", ACS Cent Sci. 2018 Sep 26;4(9):1126-1133.

So, who am I?

Parallel computing lab at graduate school	Search Algorithms
Digital wireless communication (at **FUJITSU**)	Game AI algorithms
Biometric security (finger vein recognition)	Parallel Search

Computer Go
book
(in Japanese)

I am now working for RIKEN AIP
(Center for Advanced Intelligence Project)

Wanted!
People with HPC background
and interested in AI

Our supercomputer RAIDEN
ranked 4th in Green500. (June 2017)
(I am in charge of the selection,
procurement, and maintenance)

雷電 RAIDEN

References

[Silver, Huang et al. 2016] *Nature*, vol. 529, no. 7585, pp. 484–489, 2016.
[Coulom 2006] “Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search.” CG2006.
[Yoshizoe, Terada, Tsuda 2018] Bioinformatics. 2018. doi: 10.1093/bioinformatics/bty219
[Ishii, Yoshizoe, Suzumura 2014] “Scalable Parallel Numerical CSP Solver”, CP2014.
[Izumi, Ishii, Yoshizoe 2018] “An Extended GLB Library for Optimization Problems”, HPC Asia 2018 poster.
[Yoshizoe, et al. 2011] “Scalable Distributed Monte-Carlo Tree Search”, SoCS 2011.
[Saraswat et al. 2011] “Lifeline-based global load balancing”, PPoPP’11.
[Mattern 1990] “Asynchronous Distributed Termination - Parallel and Symmetric Solutions with Echo Algorithms,” Algorithmica, Vol. 5, No.1-4, pp. 325-340, 1990.
[Pasquier, Bastide, Taouil, Lakhal 1999] “Discovering Frequent Closed Itemsets for Association Rules,” ICDT 1999.
[Avis, Fukuda 1996] “Reverse search for enumeration,” Disc. Appl. Mathematics, Vol. 65, 1-3, pp 21-46, 1996.
[Uno, Kiyomi, Arimura 2004] “LCM ver.2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets,” FIMI2004.
[Terada et al. 2014] “Statistical significance of combinatorial regulations,” PNAS Vol.110, No. 32, 2013.
[Minato, Uno, Tsuda, Terada, Sese 2014] “A Fast Method of Statistical Assessment for Combinatorial Hypotheses Based on Frequent Itemset Enumeration,” ECML-PKDD 2014.
[Romein et al. 1999] “Transposition Table Driven Work Scheduling in Distributed Search,” AAAI-1999.
[Kishimoto et al. 2012] “Evaluation of a simple, scalable, parallel best-first search strategy,” Artificial Intelligence, 2013.
[He et al. 2015] ResNet https://arxiv.org/abs/1512.03385
[Hoffer et al. 2017] https://arxiv.org/abs/1705.08741