



## Keynote

- int.
- Operations per second  $\rightarrow$  OPS  
floating point oper's / second  $\rightarrow$  fLOPs
- Benchmark  $\rightarrow$  how many ops can it do.

## Applications

### ① Day-to-Day life

- a) Weather forecasting and early warnings
- b) Improving healthcare

$\hookrightarrow$  Bioinformatics

Drug discovery, Drug mol. simul's  
by HPC.

### ② Science

- invert. e<sup>-</sup> structure
- Air quality monitoring
- Computational Fluid Dynamics.

### ③ Nano-science

### ④ N-Cap rating

## Basics of HPC

## Parallel Computing

Principle :- Large problems,  $\rightarrow$  divide into smaller problems  
 $\downarrow$  solve concurrently.

Parallel Processing - Simultaneous processing. Each small task is assigned to a single processing unit.

Distributed Computing - Using multiple, isolated processing elements.

- Mostly used in numeric simulations, modelling and data processing.

Architecture of a supercomputer

Cluster - coll. of compute nodes

Cluster → connected by switch fabric

↓  
high bandwidth, low length

Compute nodes → microprocessor + hardware - monitor - mouse etc.

User gets IP of login  
nodes

Login nodes

Parallel File System (beowulf clusters)

TCP/IP protocols

Primary interconnect

Network

Tape lib./  
backup store

Switch fabric

Compute nodes

Storage  
accelerator

Parallel  
file  
system

- Real {
- PGAS
  - Intelligent nw card
  - TCP/IP protocols
  - wakewulf clusters
  - DMA
  - Parham nets
  - RDMA
  - SSD
  - DgX3
  - SSM
  - GPU
  - GPAPUS
  - Scheduling
  - Boot Service / login

SSD vs Hard disk

→ speed is faster but cost / bit > than hard disk.

→ compute nodes that are acc-

## Process Scheduling

- Multiprogramming to ensure complete CPU Utiliz".
- Some processes running at all times.
- Time sharing → switching CPU among processes so. users can interact with all programs.
- e.g. Downloading files, writing to files, listening to music all happens at the same time acc. to us, but they are switched so freq. so as to maintain continuum.

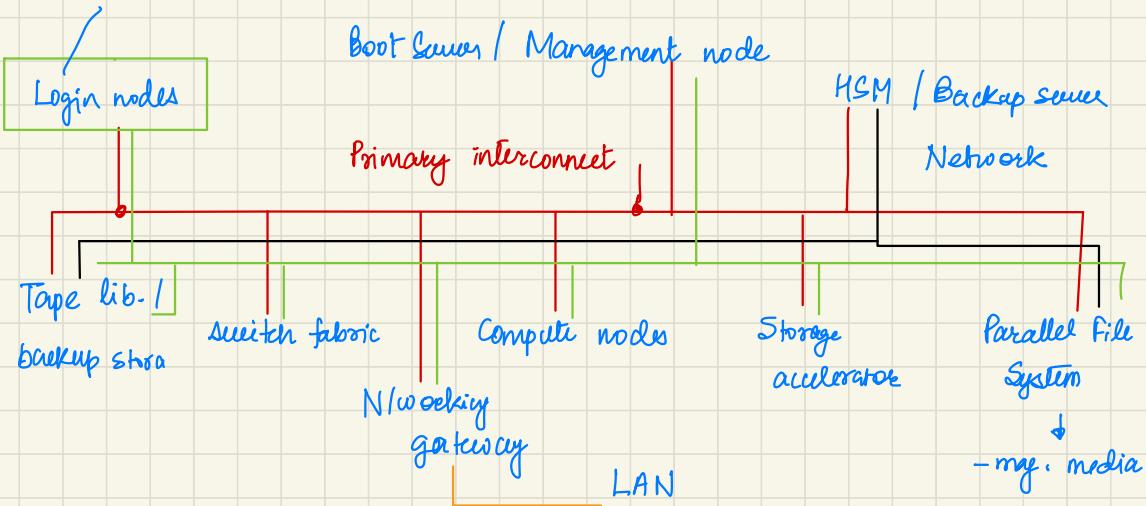
+

### Process scheduler

- Select av. proc and assign CPU core to execute
- Select priority among processes and selects which instructions can be executed at one time
- Rescheduling diff. processes.
- Scheduling queues

CPU's vs. GPUs

- CPUs are powerful, but have limited parallel capability (32 or 64 cores)
- GPUs have 4000 cores → limited memory / core but can do work independant and fast.



- Boot Server → allow these compute nodes to boot up in cfg that
  - one patch applies to all nodes

### Software Subsystems

- ①
  - Every nodes has OS (Cent OS 7.x) in Param Servers.  
↳
- ②
  - Middleware → Cluster Monitoring → monitor clusters
  - Provisioning →
  - file system
  - Res. manage

FISystem { NFS maintains home area  
Lustre pl<sup>al</sup> file system → Scratch

- Job Submitted in queues . Priority queues  
↳ maintained by SLURM → managed

- Shell Scripting

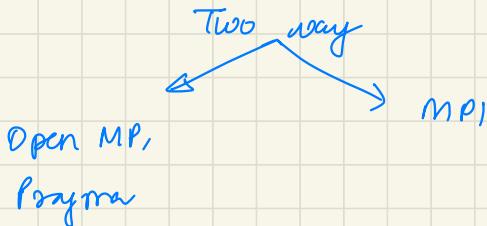
(3)

### Programming tools



- Perf. mon.
- Dev. tools
- App. libz

- Pragmas
- MPI → mech. by which we use a lib.



- hypervisor in Cloud
- Debugging in Dedicated Mode
- SIMD - concept as a part of Istm. Set ,

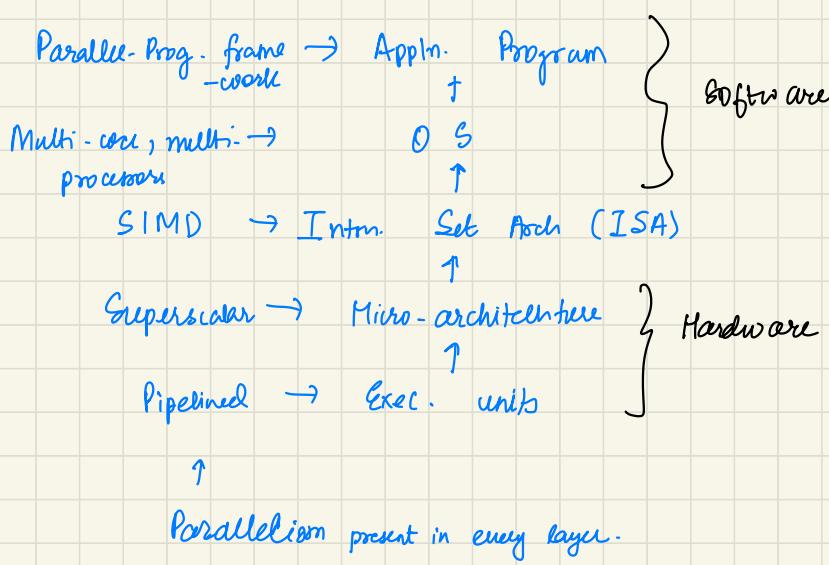
Rack Mountable Server

- One

Parallelism in Arch.

↳ Abstr. layers

black box → hardware      ↳ top software



- multithreading

### Superscalar

- Dir. → 2 clock cycles for exec.

2 diff. ALUs.

### SIMD

- AVX → Intel and AMD intro in x86.
- Intel introduced AVX-512

512 bit reg. partitioned into mult. 32 bit data

- but single instr. operated on all 32 bits

$$\begin{aligned}
 & a(31:0) + b(31:0) \\
 & a(63:32) + b(63:32) \\
 & \vdots \\
 & \vdots
 \end{aligned}
 \quad \left. \right\} \text{Same instrn. but simal}$$

- Increase processor throughput

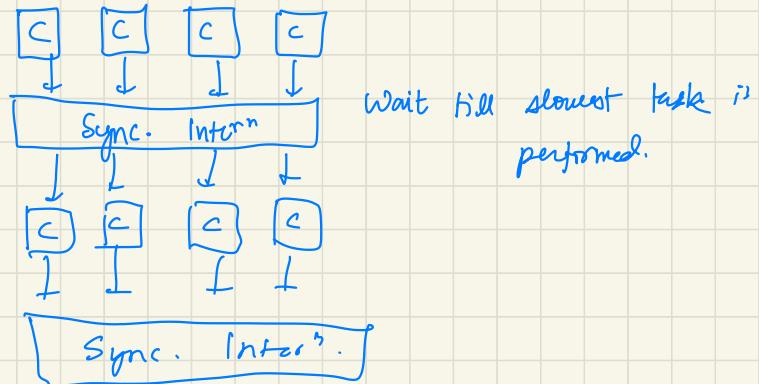
- Streaming SIMD Execution (SSE)

Multi-core Multi-processor OS

- Develop parallel-aware programs

Parallelization paradigms

- Communication among processes is largely unnecessary
- Phase II<sup>nd</sup> method



- helpful in checkpointing
- say Processor fails  $\rightarrow$  int. memory  $\rightarrow$  auto recovery.  
 $\rightarrow$  for long running progs.

Divide & conquer

- Divide workloads and then merge

Pipeline data to increase throughput

Process Farm Method - only 1 time subdivision

# Domain decomposition -

## Desirable Attributes

- Concurrency
- Scalable
- Data locality
- Modularity

Writing parallel codes

\$ run hostname

hostname

\$info

\$ run -- node >= 2

Analyse the time complexity of parallel algorithms.

### • Master method

- div into  $a$  subprob. of size  $n/2$ .
- $f(n)$  → time to div and merge  $n/2$ .

$$T(n) = aT(n/2) + f(n)$$

- Compare  $n^{\log_b a}$  with  $f(n)$

Suppose  $n^{\log_b a} \gg f(n)$

$$T(n) = \Theta(n^{\log_b a})$$

e.g. Matrix mult. by block mat. mult.

$$(3) n^{\log_b a} \ll f(n)$$

$$T(n) = \Theta(n^2)$$

$$C_{ij} = \sum a_{ij} \times b_{ij}$$

→ sync. → synchronize points

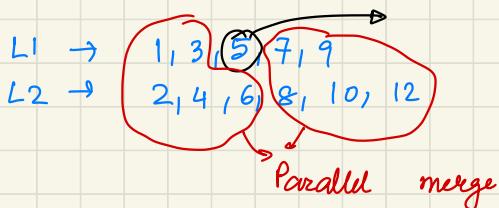
$$\begin{aligned} \text{Span of matrix multiplic.} &\rightarrow M_{\infty}(n) = M_{\infty}(n/2) + \Theta(1) \\ &= M_{\infty}(n/2) + \Theta(\lg n) \\ &= \Theta(\lg^2 n) \end{aligned}$$

• Estimate amt. of parallelism present in an algorithm

- Time for single thread exec.  $\rightarrow O(n^3)$  = Work
- Time for multi-threaded exec.  $\rightarrow O(\log^2 n)$  = Span

$$\text{Amount of parallelism} = \frac{O(\log^2 n)}{O(n^3)} = \frac{\text{Work}}{\text{Span}}$$

Parallel Merge of Merge Sort



• Span of P\_Merge

Work

$$\text{Span } T_{\infty}(n) = T_{\infty}(n/2) + \Theta(\log^2 n)$$

• Click code

• Recursive construction

## Ethernet

- Widely used LAN technology
- Operates in data-link layer and PHY layer
- Supports data b/w of 10, 100, 1000, 10,000, 40,000 and 100,000 Mbps (100 Gbps)

$$T_{\infty}(n) = 5 T_{\infty}(n/3) + O(1)$$



$$n^{\log_b a} = n^{\log_3 5}$$

$$\Theta(n^{\log_3 5})$$

$$\frac{T_i(n)}{T_{\infty}(n)} = \text{Parallel}^* = \frac{O(n^2)}{O(n^{\log_3 5})} = O(n^{2 - \log_3 5})$$

- logical core, physical core
- 2 threads / more in one core.
- I Cache, D Cache
- Instr. Cache, Data Cache.
- L1, L2 Cache local to chip
- L3 Cache common to all core

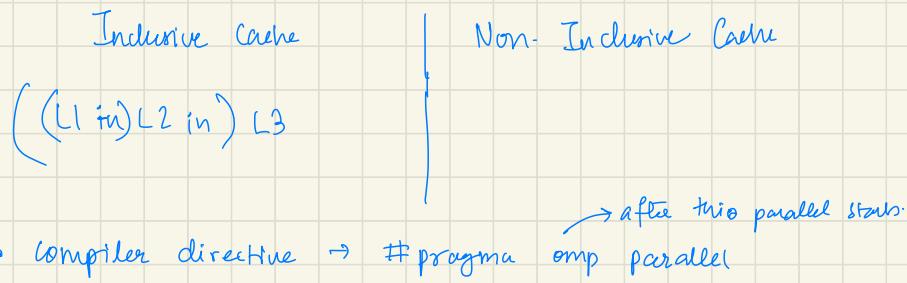
Write progs using shared mem. concepts  $\rightarrow$  Open Mp  
Conv. by MMU

Processor - logical address  $\xrightarrow{\text{Conv. by MMU}}$  physical address  
↳ generate  $\uparrow$

then proc. checks whether data is in L1 Cache

- data stored in main mem - in page mem. (4KB on Sir's system)
- Blocks (64) gen. in mem.
- Inf. needed  $\rightarrow$  entire block sent to cache.

- o DMA is responsible for transfer data from secondary mem. to main mem.



- o Compiler directive  $\rightarrow$  `#pragma omp parallel`

```
omp_get_max_threads();  
omp_set_num_threads(4);
```

master thread  $\rightarrow$  only 1

`#pragma omp barrier`  $\rightarrow$  sync. point

all threads running should complete exec.

- Find area of a square by subdividing into various sub-squares.  
 $\rightarrow$  false sharing

### Parallel Computer Memory Architecture

Shared memory (UMA)

Non-Uniform Shared Mem. (NUMA)

Distributed memory

Hybrid mem  
NUMA + Distr.  
mem.

## MPI

- MPI employs processes to do parallel tasks. Any process contains of a separate address space.
- A process may have multiple threads.

1. Prefix sum problem - Bitonic prefix solution
2. Merge sort problem
3. Matrix Addn.
4. Matrix Multiplication.

## ① Matrix addition.



## Physics Assisted Digital Twins for Complex Systems

- Explicit large time stepper
- Tracking slow manifold
- PDE solving.

$$\frac{\partial^2 \rho}{\partial z_i^2} = \frac{2}{\partial x_i} \left( u_j \frac{\partial u_i}{\partial x_i} \right)$$

Small scale flow  $\rightarrow$  non-linearities  
 $\hookrightarrow$  Reynold's number

- Parallelize non-linear flow.
- Cost of DNS  $\sim Re^3$ .  
 $\hookrightarrow$  compute
- Pipe  $10^3 - Re$  Compute power - teraflop  
Automobile  $- 10^{12} \times 100$
- Reynolds - Assisted Navier Stokes (int. particles)

- Large eddy Simul". (LES)
- Large scale energy systems  $\rightarrow$  complex systems.
- Real physical systems 3% eff  $\rightarrow$  100 b  $\text{CO}_2$  savings

### Semiconductor manufacturing

- Plasma chem., multiscale process.
- comp. Simulns to model real bodies

### Cardiovascular diagnosis protocol

- Cheaper more reliable devices,
- pulmonary artery, lipid deposition of plaque.
- ECG and TMT  $\rightarrow$  but very little info.
- CT-scan  $\rightarrow$  structural info. not func. info.

$\downarrow$                        $\downarrow$   
 how it                      measurements  
 looks                      eg pressure

- Angiogram  $\rightarrow$  invasive but risky

$\downarrow$

Invasive FFR  $\rightarrow$  measure pressure ratio before and after block.

Take CT scan images and try to model pressures in diff. parts of heart.

### Computational Physics

- Numerical methods
  - math. schemes to solve problems
  - Int. or diff.

- Root finding
- Optimiz.
- Linear algebra - ideal value finding
- DE - Navier Stokes,
- Numerical Simulns.
  - Perform sim. inside the computer
  - math. models / eqns.
  - Modelling and mimicking
- Nucleation problem
  - don't perturbate it  $\rightarrow$  supercooled state, metastable state.  
 $\rightarrow$  finite prob.
- microstates given a state  $(n, V, T) \rightarrow$  ensemble theory
- fix accessible microstates
- average all  $\rightarrow$  ensemble average given a microstate.
- Monte-Carlo simulation.  $\rightarrow$  generate eqib. configurations by drawing out from a distro.  
 use molecular dynamics
- $(N, V, E)$  ensemble to define a state.

Inter. Potential

- Lennard Jones potn. - short range potn.
- Soft repulsive, Short attr.: scale for distance

$$U(r_{ij}) = 4\epsilon \left[ \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left( \frac{\sigma_{ij}}{r_{ij}} \right)^6 \right]$$

↑ scale for energy

- Fix bdry points to declare  $\Theta$ , due to asymp.  
↓
- Chopping  $\rightarrow$  non diff. forces  $\rightarrow$  non-realistic  $\rightarrow$  use quadratic.

Implem<sup>n</sup>.

- Initializ<sup>n</sup>.

for a solid { → pos<sup>n</sup>. - say simple cubic  
- define a box length, gen. periodic bdry.

- for a liquid → uniform
  - generate a random no. for a position.
  - non-crystalline systems
- But if high den. particles come closer  $\rightarrow$  force becomes repulsive  $\rightarrow$  start with FCI (high. pckg. frac.) and increase vel. till it melts.
- Just ensure no two part. overlap each other or  $< 0$ .

Velocity initializn.

- follows Maxwell Boltzmann distribn.

$$f(v) = \left( \frac{m}{2\pi k_B T} \right)^{\frac{1}{2}} e^{\left( -\frac{mv^2}{2k_B T} \right)}$$

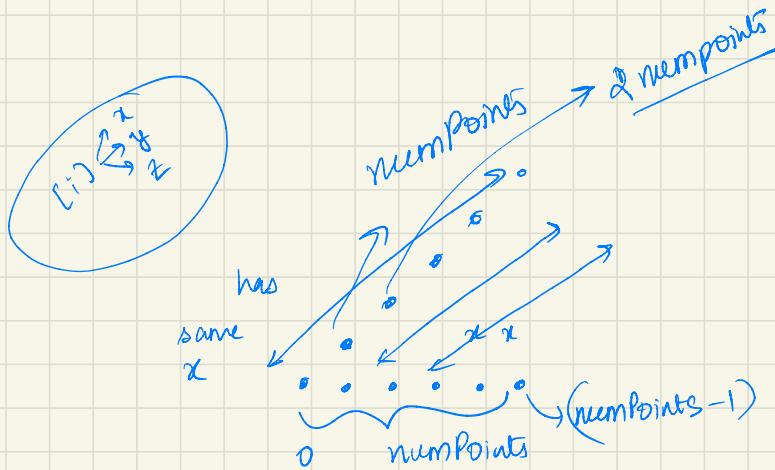
<sup>↑</sup>  
Gaussian distribn.

Variance  $k_B T$

- Generally periodical boundaries, due to

Radial distribution and Co-ordin. number

No. of particles	Temperature	Rho	Time
4.0	1.0	0.1	0.17
4.0	16.0	0.5	0.20
16.0	1.5	0.5	2.14
32.0	1.5	0.5	7.13
64.0	1.5	0.5	17.21
128.0			35.04
256.0			82.82
512.0			212.02
1024.0			618.76



→

Real systems in experiment  $\sim$  Avogadro no.

→ capture inherent coriol. length

Unit of energy -  $E$  (depth of profn.  
length =  $\sigma$  (dia) of part.)  
mass =  $m$  (mass of part.)

Velocity - Verlet  
algo

in simulation.

$$x(t + \Delta t) = x(t) + v(t) \Delta t$$

RK4 method

$$\rightarrow + \frac{f(t)}{2m} \Delta t^2$$

2nd order RK

Kind of Taylor series.

$$x(t+h) = x(t) + k_2 h$$

2nd order

$$k_1 = f_1(x, t)$$

Range- Futta

$$k_2 = f\left(x\left(t + \frac{h}{2}\right), t + \frac{h}{2}\right)$$

→ Energy not conserved

RK4 does not

{

$$r(t + \Delta t) = r(t) + v(t) \Delta t + \frac{f(t)}{2m} \Delta t^2$$

Use here

$$v(t + \Delta t) = v(t) + \frac{f(t + \Delta t) + f(t)}{2m} \Delta t$$

Verlet  
algo.

# Profiling and Optimization

- Algorithmic Analysis  $\rightarrow$  which part takes most time.  
Optimizn.  $\rightarrow$  reduce that time.
- Disable the I/O, e.g. outputting thermodynamic values.
- Disable the optimizn. flags
  - Fix the time  $\rightarrow$  max. timesteps and timestep
  - Vary the parameters - no. of particles, density cores.
- time -p
  - $\rightarrow$  sys time, user time, clock time

## Profiling tools

- Serial code opt.

```
gcc -pg -no-pie -fno-bulletin program.c
```

- Run the code  $\rightarrow$  out  $\rightarrow$  gmon.out
- gprof gmon.out

## Optimizn. $\rightarrow$ Nearest neigh. list

- Finding neighbour list, how often update neigh. list.
- Compute neighbour list  $n = 2.5 + \sigma$   
 $\uparrow$   
skin.
- If the dist. travelled  $|\mathbf{r}(t) - \mathbf{r}(t+\Delta t)| = \Delta r$
- if  $\max_{\forall i} (r_i) > \frac{\text{skin dist.}}{2}$ , update the neighbour list.

# Fast Parallel Algorithms for Short-Range Molecular Dynamics

- Large-scale Atomic / Molecular Massively Parallel Simulator (LA MMPS)

Atom decompos. algo - N/P atoms - each processor

- Commn. overhead  $\rightarrow$  problematic  
 $\hookrightarrow (\log_2 P) \rightarrow$  optimal

$P \rightarrow$  cores.

- Neighbour list is shared to all the processor.
- update shared only with all the cores.
- Force decomposition
- Spatial decomposition

## Advanced Inputs

- Radial Correlation  $\rightarrow$  prob. at what distance of finding another particle

## Assignments

- 1. Implement using neighbourlist.

## 10 Matrix Multiplication

- a. Create 2 Matrix of size  $(n \times n)$ .
- b. Create a pointer to the 3rd matrix.
- c. Scatter the inf. to sub processes from the root (0).
- d. Gather the inf. at the root and concatenate the result.

Simp. 1D matrix  $(1 \times n)$  and  $(n \times 1)$



$n$  processes  $\rightarrow$  buff of size 2

a

b

Returns  $a^T b$

Accelerator architectures and Computation offloading

$$T_{\text{Exe}} = \text{Instr. / program} \times \text{CPI} \times \text{Tcycle}$$

CPI :- cycles per instr.

ISA

Instr. / program :- choose some intelligent arch. / compiler

CPI :- about 0.5 for superscalar arch. + pipelining

Amdahl's law

Speedup  $\rightarrow$  bottlenecked by serial portion of code.

f :- fraction parallelised

$(1-f)$  :- serial code

$$\text{Speedup} = \frac{1}{(1-f) + (f/s)}$$

$$S \text{- speedup} = \frac{\text{new time}}{\text{old time}}$$

$$\max_{s \rightarrow \infty} (\text{Speedup}) = \frac{1}{1-f} \rightarrow \text{serial code}$$

if  $f=2$ ; then even on inf. resources,

$$\max_{s \rightarrow \infty} (\text{Speedup}) = 2$$

→ diminishing return on increasing resources.

$$\text{Efficiency} = \frac{\text{Speedup}(S(n)) = T(1)/T(n)}{n}$$

$$\text{Eff. } (E(n)) = \frac{T(1)}{n T(n)}$$

Gustafson's law

$\Pi$ : - fon. of parallel part.

$S \rightarrow$  serial code, execn. time

$n$ : - no. of processors

$\Pi n$ : - total

$$S_S(n) = (S + \Pi n) / (S + \Pi)$$

- scaling up the problem size as you increase the parallelization resources.
- lineariz. does not scale up.

Specialized Hardware Resources - Accelerators

- Higher efficiency to tailor to a specific purpose.

## Why Domain Specific Architectures

- Ineff. of high level languages
  - ↳ C + parallel loops + memory optimizn.  
+ SIMD instrns.
- optimizn. towards specific applicns.

Domain Specific Architectures (DSA)

Domain Specific Languages (DSL)

Best combin.  $\rightarrow$  (DSA + DSL + Automatn.)

## How to design domain Specific Accelerators

- Eff. memory design
- Parallelism
- Redn of overhead
- Specialized Ops. - own data type

### Parallelism

- make few global memory
- make few PE - cross data movement.
- 

### Eff. mem. system

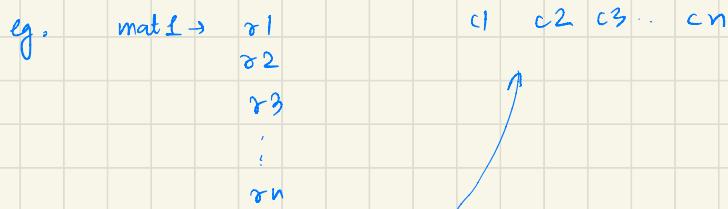
- Specialized mem. system.
- reduce point-to-point commn.
- Arithmetic oper. is free, Memory is expensive.

Add  $\rightarrow$  3 memory op.  $\rightarrow$  high mem. access.

- Storing key data structures on local cache memory
- do data compression eg. storing in sparse matrix.
- gen. DL workloads  $\rightarrow$  matrix  $\uparrow$

Sparse matrix  $\rightarrow$  reduce some of the elements to 0.

- data reuse - Once stored in local memory from global memory, reuse it as much.



store column matrix as row matrix  $\rightarrow$  contiguous allocation  
and reuse it.

CGRAs  $\rightarrow$  point-to-point comm. b/w diff. processors

### Reduction of overhead

- sp. hardware
- out-of-order execution reduces 99 % overhead.
- sign. code to CPU and parallel code to sp. hardware like CGRAs.
- pragma call to compiler
- only comm. to the CPU is whether done or not.

- CPU  $\rightarrow$  hosts, gen. purpose computing
- minimize memory footprint.
- data comp., sparsity
- lots of fast, on-chip memories.

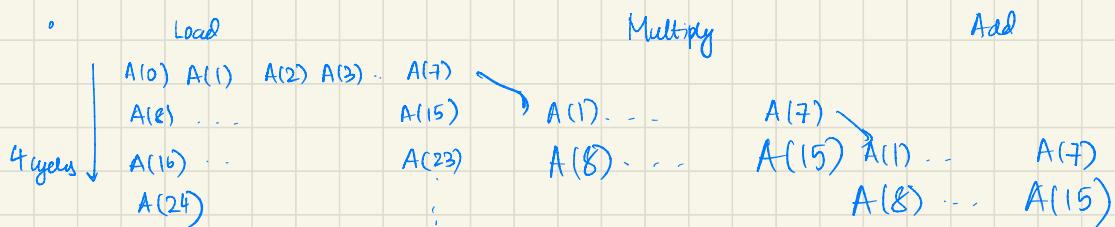
- Part of execution pipeline → tightly coupled
- loosely coupled.

Single instruction, multiple data

{ 
 

- Array processors → different cases
- Vector processors → specialized hardware for performing its own op?

- Memory banks → to indep. access data.
- done wise execution manner.
- Vector-Instruction Parallelism



VLIW arch. and ELI-512.

Instruction stream

◦ LD VR A(3:0)

- disadvantages in Array / Vector processors → data dependencies.
- VLIW → Compiler sees the code and creates the instrn.

Automatic Code Vectorizn.

- Scalar Sign. instrn. (each step is indep.)
- Compiler sees this and does code vectorizn.

vector  $\leftarrow$  load A(1) load A(2) load A(3) ...
   
 instrn.  $\leftarrow$  load B(1) load B(2) load B(3)

- Vector masks -

## SIMD

- acts on multiple data at once
- short arithmetic ops
- vector instrns.
- must modify ALU to eliminate carry values.

## SIMD in modern accelerators

- Cerebras WSEngine
  - neural layers mapped to diff. layers
  - MIMD machine  $\rightarrow$  each tile in a die in a wafer
  - distributed memory, not shared
  - 4-way SIMD for 16
  - 48-KB local SRAM
  - GPUs are SIMD machines
- VLIW
- thread  $\rightarrow$  own threads
- stalls in pipeline

## SPMP

- $A[i] + B[i] = C[i]$
- SIMD extn. (either vector or array)
- Thread for each item.
- Diff. b<sup>w</sup> SIMD machine or MIMD machine

- vector code
- thread
- 

GPU is a SIMD (SIMT) Machine

- SIMT  $\rightarrow$  prog. using threads
- thread  $\rightarrow$  executes same code but diff. data

- Multiple threads → warp (wavefront) grouping threads
- SIMD ops": formed by hardware → warp
- Warp → set of threads perform same instrn.
- SIMD - each instrn.
- SIMT-
  - ↓
  - MIMD processing
- Fine Grained MThr. of warps
- Group scalar thread into warps
- branch div.
- SIMD utiliz.
- Dynamic Warp Formation.

### An Example GPU

- Tensor cores (for deep learning)
- PCI Express for connecting to GPU
- FP16 or FP 32 cores
- Tensor Core Architecture
- 4x4 matrix mult. → 1 cycle

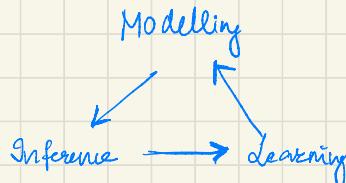
NVIDIA A100

- 108 cores, tensor cores, 64 func. units per core
- New floating point data type (TF32)
- Tensor cores have sparse mat. supp.
- Red. mem-comp by calculn. in that core itself.

- NVIDIA insight CLI profiler
  - what benefits
  - exp. varying block sizes of dim<sup>n</sup>'s
- CUDA programming.

## Artificial Intelligence

- AI →
- Represent<sup>n</sup>. that will play / support models targeted to thinking, percept<sup>n</sup>, defn.
  - All about making models →
  - constraints →
- Also AI →
- model with constraints + represent<sup>n</sup>. that support.
  - Algorithm
- model → constraints → represent<sup>n</sup>. → algorithms → soln.
- Problems → solutions diff. ways.



Modelling - simplific<sup>n</sup>. , precise math. expression.

Inference - asking qns., and get the soln.

Learning - (data + model with param.)

parameters are learnt from the data.

Data  $\rightarrow$  model

+

ask qns. on data that hasn't been trained.

• leap of faith on the model. - generalizn. of machine learning

Reflex based intelligence

e.g. linear classifiers, deep neural nets  
→ fully feed-forward.

State based models e.g.

- search based problems
- Markov decision based prob. (Blackjack)
- Adversarial games

Variable based models

e.g. Constraint satisfn. problems  
Bayesian networks

Optimizn. - discrete optim.

$$\rightarrow \min_{p \in \text{parameters}} \text{Distance}(p)$$

Algorithmic app. - dynamic programming

iif a cont. optim?  $\rightarrow \min_{w \in \mathbb{R}^d} \text{Train\_error}$

Algorithmic opt. - gradient descent.

a) Compute edit distance

Gen. principle

"cat" "dog"

↑ is complete present

iif Substr. present

iif Instrn. needed before / above

- W<sub>i</sub> Insertion needed in between.
- V<sub>i</sub> Deletion of some characters
- V<sub>i,j</sub> Deletion of some charact. + insert of some char.

ab                    cd  
 4                    10 → ①  
 =                    =  
 a) extract common character  
 b) delete from both ends  
 insertn. + deln. (subs.) also  
 = + 9 + 3            ↗ least cost.

**General principle** - Reduce the problem.

-

- dynamic programming
  - reduce the problem size
  - match the problem

dynamic program

## 9 Introduction to Deep learning

→ McCulloch Pitts neuron

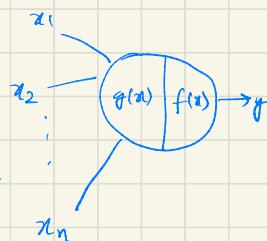
$$g(x) = \sum_{i=0}^n x_i$$

boolean  
Values

$$fog(x) = \begin{cases} 1 & g(x) \geq 0 \\ 0 & g(x) < 0 \end{cases}$$

$f(g(x)) = \text{output} = y$

↳ boolean



→ feedforward - not feedback loop provided.

## Perceptrons

- generalized version of MP neuron
-

$$g(x) = \sum_{i=1}^n (w_i x_i)$$

→ linear eqn. in n dimension  
line) ↑

- weights to provide some preference
- inputs can be integer values as well.
- Perceptron Learning Algorithm

$$\text{if } x \in P \text{ & } w \cdot x < 0 \quad w = w + x$$

$$\text{if } x \in N \text{ & } w \cdot x > 0 \quad w = w - x$$

draw a straight line boundary  
b/w P and N.

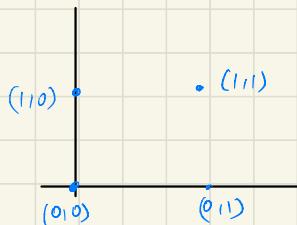
XOR conundrum

$$\rightarrow w_0 < 0$$

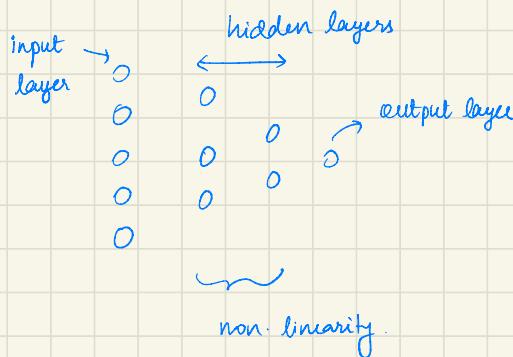
$$w_1 > -w_0$$

$$w_2 > -w_0$$

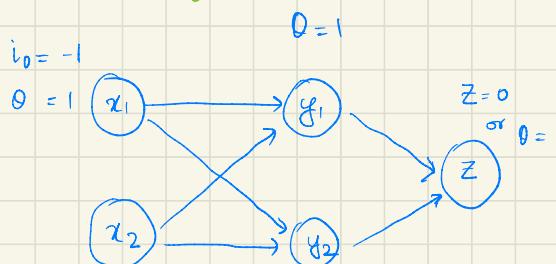
$$w_1 + w_2 < -w_0$$



## Multi-Layer Perceptrons



## Multi-layer Perceptrons



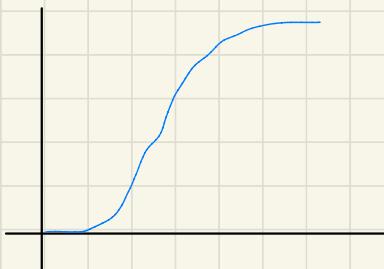
Theorem - Any boolean fn. of n ips can be exactly represented with 1 input layers with n neurons + 1 hidden layer ( $2^n$  neu.)

+ 1 output layer (1 neuron).

Need for activM. functions

Sigmoid neurons  $\rightarrow$  smoother transn.

$$f(x) = \frac{1}{1+e^{-x_i}}$$



- Output is no longer a binary values but
- eg. **ReLU** :-  $y = \max(0, z)$   
**Leaky ReLU** :-  $y = \max(\alpha z, z)$   
**Exp. linear decay (ELU)**  $y = \max(\alpha(e^z - 1), z)$   $\alpha > 0$

Tanh

all these funs need to be differentiable

## Feedforward Neural Networks

- multi-layer perceptron (multiple layer perceptron).
- $f^*$  fun. approx. poss.
- neurons are formed by a directed acyclic graph.

## Gradient Descent

- Loss MSE =

$$= \frac{1}{M} \sum_{i=1}^M (f^*(x) - f(x_i; \theta))^2$$

mean-square error.

- $\min(f(x) = x^2)$
- if neg. slope go ahead,  $x + \frac{df}{dx}$
- if pos. slope go behind,  $x - (-1)\frac{df}{dx}$

## Neg. gradient

Why?  $L(\theta) \rightarrow$  loss gradient

- Random vector, calculate
- $o_i^{next} = o_i^{curr} - \eta \left( \frac{\delta L}{\delta o_i^{curr}} \right)$   
learning rate

$L$ : (loss function)

- $\eta$  :- hyperparameter here
- Compute gradient of loss function

$n_l$ : layers in the network,  $l = 1, 2, 3, 4, \dots, n$ .

$$z \quad a = f(z)$$



$f$  :- activ. funtn.



## Backward pass

→ Trying to compute

$$\frac{\partial L}{\partial w_i} = \left( \frac{\partial L}{\partial a_i}, \frac{\partial a_i}{\partial z_i}, \frac{\partial z_i}{\partial w_i} \right)$$

$\nwarrow$        $\curvearrowleft$        $\curvearrowright$

$\nabla_{w^{(l)}} L$        $a^{l+1}$        $f(z) = (a^l)^T$

$$\frac{\partial L}{\partial w} =$$

use backpropag<sup>n</sup>. to compute  $\nabla_{\theta^l} L(\theta; x, y)$

$$w^l \quad b^l$$

Update the

$$w^l = w^l - \eta \left[ \frac{1}{M} \Delta W^l \right]$$

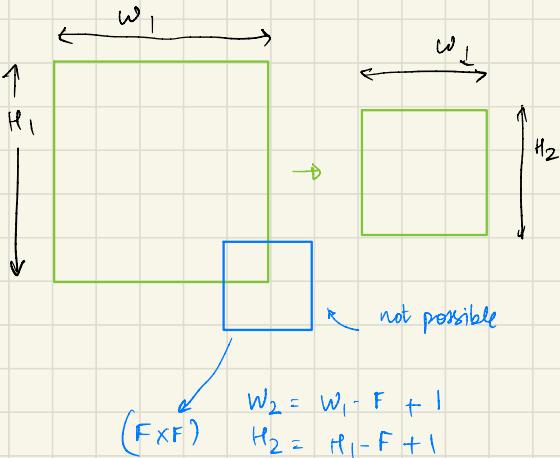
Consider a simple FFNN (or MLP)

pytorch, tensorflow, Keras

## Convolutional Neural Networks

- $Y(i, j) = \sum_{u=-K}^K \sum_{v=-K}^K w(u, v) \times (i-u, j-v)$

- for  $K$  filters  $\rightarrow K$  channels in output feature map.
- Width( $w$ ), Height ( $H$ ) and Depth ( $D$ ) = same as no. of channels of ip.)  
output =  $w_2 \times h_2 \times D_2$
- Basically size reduction



- If we want  $\text{I/p size} = \text{o/p size} \rightarrow \text{padding}$

New formula

$$W_2 = W_1 - F + 2P + 1 \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{Padd. at both ends}$$

$$H_2 = H_1 - F + 2P + 1$$

Striding  $\rightarrow$  skipping some pixels

With stride  $W_2 = \frac{W_1 - F + 2P}{S} + 1$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

$$W_1 = 227$$

$$F = 11$$

$$H_1 = 227$$

$$q_b = n$$

$$D_1 = 3$$

$$W_2 = 55$$

$$D_2 = 96$$

$$H_2 = 55$$

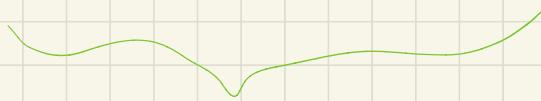
Why Convn?

Traditional ML  
based Vision



- Edge-Detector
  - SIFT-HOG
  - LBP
  - Classification  $\rightarrow$  SVM, Perceptron classifier
- $\left. \begin{array}{l} \\ \\ \\ \end{array} \right\} \text{Filter} \rightarrow \text{static feature extrn.}$

- Instead of handcrafted filters, learn meaningful kernels/filters in
- filters for feature extn.
- Add kernels as parameters, and using back-propg.



## Convolutional neural nets

- Challenges of applying FNNs to MNIST.
  - 2% degrad.
  - Ignores spatial structure, pixels that are spatially separate are treated the same way as other pixels.
  - 1MP  $\rightarrow$  60 million parameters in 1<sup>st</sup> layer.



- local receptive fields
  - $\rightarrow$  Convolution captures local spatial retn.
  - $\rightarrow$  translational invariance of filter  $\rightarrow$  weight sharing.

Pooling layer  $\rightarrow$  condenses info. more

## Local receptive fields

- $\rightarrow$  in CNNs, conn's are much sparser in this case in comp-to fully connected NN. (FNNs).

Is sparse connectivity a really good thing?  
Not really, losing any inf.

## Weight sharing

$\rightarrow$

## Pooling layer

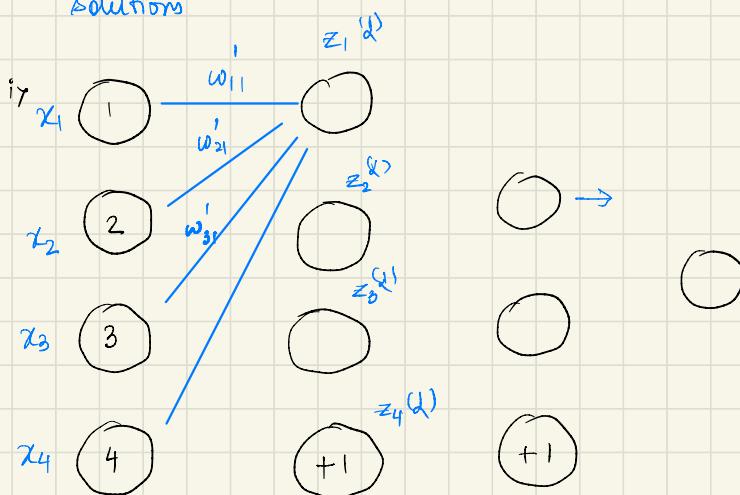
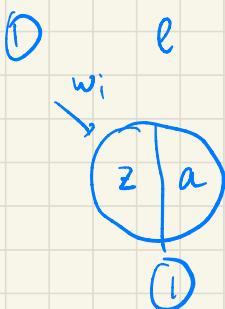
- down sampling
- or average pooling
- diff b/w max pool and convolutn. ???

} generally all filter conv. is  
done in freq. domain.

## Transpose Convolution

- Allows for deconvolution or learnable upscaling
- 3D convol<sup>n</sup>, 1x1 Pointwise Convol<sup>n</sup>, Grouped Convol<sup>n</sup>.

### Solutions



① solve XOR problem for 2 hidden layer

② solve for the weights for all the weights in the  
above problem.

① st.

$$A_{kl} = z_k^{(2)} = \sum_{i=1}^4 \sum_{j=1}^3 w_{ij}^{(1)} x_i + b_i$$

$$\text{so, } z^{(2)} = W^{(1)T} X + B_1 \rightarrow \text{Matrix eqn. of 1st layer}$$

$$\frac{\partial z^{(2)}}{\partial w_{ij}} = \frac{\partial w^{(1)T}}{\partial w_{ij}} X + \frac{\partial X}{\partial w_{ij}} w^{(1)T} \xrightarrow{=} 0$$

$$= \frac{\partial}{\partial w_{ij}} (w^T X) + \frac{\partial B_1}{\partial w_{ij}} \xrightarrow{=} 0$$

$$\frac{\partial z^{(2)}}{\partial w_{ij}} = \begin{bmatrix} 0 \\ 0 \\ x_i \leftarrow j^{\text{th posn.}} \\ 0 \end{bmatrix} = x_i \text{ in } j^{\text{th posn.}}$$

lets say,  $X_{ij} =$

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix}_{4 \times 4}^T \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}_{4 \times 1} \xrightarrow{=} z^1 \text{ or } a^1$$

$\uparrow$   
 $w^{(1)}$

$$a^2 = f(z^2)$$

$$\frac{\partial a^2}{\partial w_{ij}^{(1)}} = \frac{\partial f^2(z^2)}{\partial w_{ij}}$$

$$= \frac{\partial f^2(z^{(2)})}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_{ij}}$$

$$= \frac{\partial f^2(z^{(2)})}{\partial z^{(2)}} X_{ij} = f'(z^{(2)}) X_{ij}$$

$$z^{(3)} = w^{(2)} a^{(2)} + b^{(2)}$$

$$z^4 w_{ij}^2 \frac{\partial z^{(3)}}{\partial w_{ij}^{(1)}} = \frac{\partial w^{(2)}}{\partial w_{ij}^{(1)}} \cdot a^{(2)} + 0 + w^{(2)} \frac{\partial a^{(2)}}{\partial w_{ij}^{(1)}} \xrightarrow{\text{any one of them } = 0}$$

$$= \frac{\partial z^{(4)}}{\partial w_{ij}^{(1)}} = a^4 + w^4 \frac{\partial a^4}{\partial w_{ij}^{(1)}}$$

$$\frac{\partial a^y}{\partial w_{ij}x} = \frac{\partial f(z)}{\partial z} \times \left( \frac{\partial w}{\partial w_{ij}x} a^{y-1} + w^{y-1} \frac{\partial a}{\partial w_{ij}x}^{y-1} \right)$$

$$= h(\theta; x)$$

$$a^y = f(w^y a^y + b^y)$$

$$L(\theta) = \frac{1}{M} \sum_{i=1}^M L(\theta; x^i, y^i)$$

$$= \frac{1}{2M} \sum_{i=1}^M \|h_\theta(x^i) - y^i\|^2$$

$$L(\theta) = \frac{1}{2M} \sum_{i=1}^M \|a^{f_e}(x^i) - y_i\|^2$$

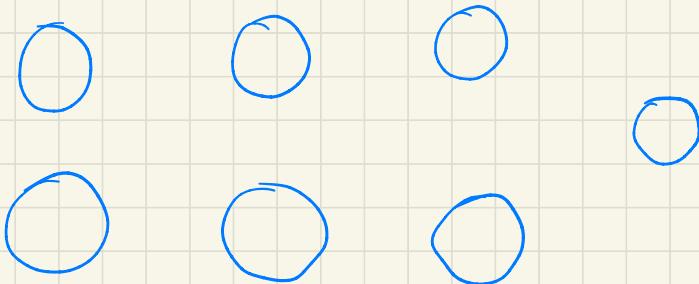
$$L(\theta) = \frac{1}{M} \sum_{i=1}^M \|a^{f_e}(x) - y_i\|^2$$

$$L(\theta) = \sqrt{\frac{1}{M} \sum_{i=1}^M (a^{f_e}(x^i) - y_i)^2}$$

$$\frac{\partial L(\theta)}{\partial w_{ij}x} = \frac{1}{2} \left( \frac{1}{M} \sum_{i=1}^M (a^{f_e}(x^i) - y_i)^2 \right)^{-\frac{1}{2}} \times \left( \frac{1}{M} \sum_{i=1}^M \frac{\partial a^{f_e}(x^i)}{\partial w_{ij}x} \right)$$

$$\text{and } \frac{\partial a^{f_e}(x)}{\partial w_{ij}x} = \frac{\partial f^{f_e}(z)}{\partial z} \left( \frac{\partial}{\partial w_{ij}x} w^{(f_e-1)} a^{(f_e-1)} + w^{(f_e-1)} \frac{\partial}{\partial w_{ij}x} (a^{(f_e-1)}) \right)$$

## XOR Problem



Q i) Can we take any threshold values?

→ Yes

$$\Theta = 1$$

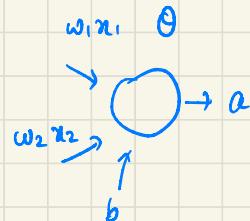
Q ii) How do we calculate weights and biases?

Two ways → diff. and find their local minima.

Else use existing solutions.

Way → go from behind  $\rightarrow \text{XOR} = AB' + A'B$

OR  $\Rightarrow$  two process

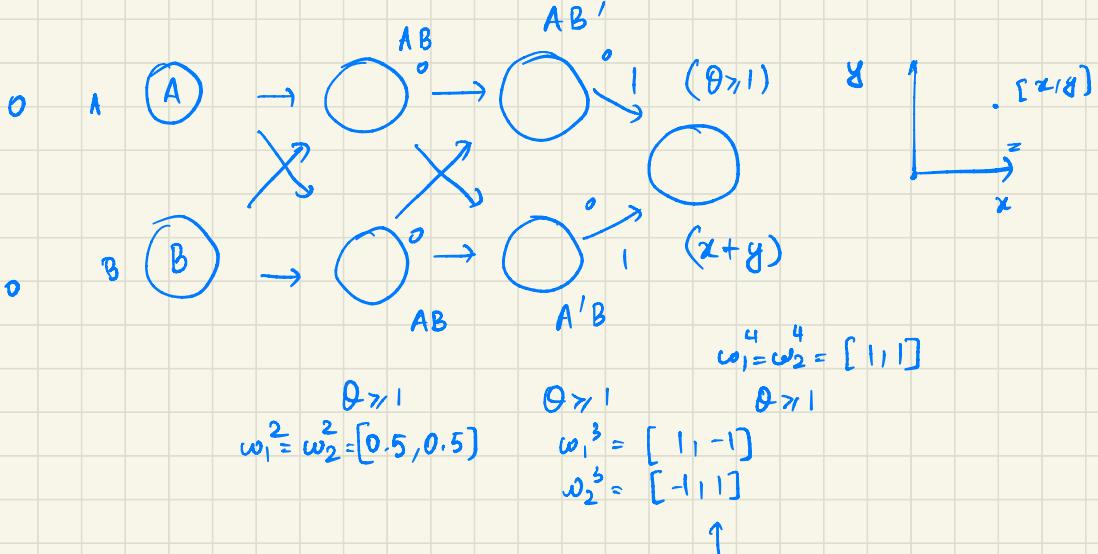


$$a = (w_1x_1 + w_2x_2)$$

$$w_1 = w_2 = 1$$

$$= x_1 + x_2 = (0, 1, 1, 2)$$

if  $a > 1 \rightarrow \text{logical or}$



Neighbour list calculation

$$g_s = g_{\text{skin}} = g_c + \Delta R$$

$\underbrace{\Delta R}_{\text{list of all particles within } (r_s)}$

next 100 it,

## Pytorch Tutorial

- Compute intensive tasks  $\rightarrow$  matrices  $\rightarrow$  data structure  $\rightarrow$  numpy.

Q] Why Pytorch?

$\rightarrow$  Pytorch  $\rightarrow$  easy to use API

$\rightarrow$  easy to debug

$\rightarrow$  integr. with GPU is well defined. as compared to Tensorflow.

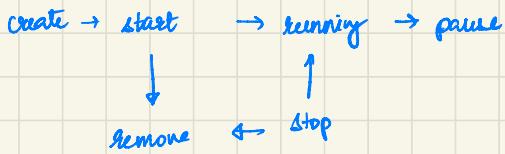
- Computational graphs are dynamic  $\rightarrow$  Pytorch  
Tensorflow and legacy frameworks  $\rightarrow$  static.

# Fundamentals

- Tensors -
  - just like numpy arrays but can run on a GPU
- Capsule networks in Autoencoders

## Dockers

- Dependency problem
- Hypervisor
- Docker pull, build, run. → APIs to Docker host.
- Docker container
  - ↓



- docker client contacted docker daemon
- daemon created a container from the image running the exec.

## Computer Vision

- Sampling and Quantiz?
- Image :- Grid : pixel (P) value on the grid.
- Grayscale :- [0, 255]

## Image Transformations

- $\hat{I}(x,y) = I(x,y) + 20.$
- ↑  
img. matrix  
→ point transform

- $\hat{I}(x,y) = I(-x,y)$

## Image processing ops

- Point op.
- Focal ops - ( $p \times p$ ) weigh dep.
- Global ops - all the values of ip. det. the op.

Pop → Img. enhancing

$$\hat{I}(x,y) =$$

contrast stretching       $x$        $0$

$$\hat{I}(m_{\min}) = \left( I(m_{\min}) - \min_{x,y} I(x,y) \right) / \text{int.}$$

$\uparrow$   
min. pixel

$*$

$$\left( I_{\max} - I_{\min} \right) / \left( \max_{x,y} I(x,y) - \min_{x,y} I(x,y) \right) + I_{\min}$$

$$\hat{I}(m_{\min}) =$$

$$150 \rightarrow (100, 200) \\ \rightarrow (0, 255)$$

) stretch

$$(150 - 100) \times \frac{(255)}{100} + 100$$

$$= 50 \times \frac{255}{100} + 100$$

$$= 100 + 127.5 \\ = (227.5)$$

Histogram equalization →

Importance of point opern. → single pixel info.

Given a camera and a still scene, how do you reduce noise?

- Take many pictures, average them.
- Otherwise need local opern. → since not always possible to take multiple pictures
- local neighbourhood → take average
- Image transform<sup>n</sup>. in Fourier domain → global ops.

Ultimately - Modify pixels based on some fn. of local neighbourhood of each pixel.

fn. → called img. linear

linear comb<sup>n</sup>. → one such filter.

Gross correl<sup>n</sup>:  $g(i,j) = \sum_{u=-k}^k \sum_{v=-k}^k H(u,v) I(i+u, j+v) \frac{1}{(2k+1)^2}$

$$g = H \otimes I$$

Kernel size of  $(k+1) \times (k+1)$

1. box filter in case of  $H(u,v) = (1, 1, \dots, 1)$  all

2. Gaussian Avg. filter → smoothes out

3. Edge filter → horizontal

Correl<sup>n</sup> → Convolut<sup>n</sup>.

→ linear shift-invariant operators → distributive law

convolution  $\rightarrow$  shift on the object  $\rightarrow$  no diff.  
better  $\therefore$

### Prop of conv.

- $a * b = b * a$
- $a * (b * c) = (a * b) * c \Rightarrow ((a * b) * b) * b = a * (b * b * b)$

### Separability -

- Singular Value Decomposn.

Padding  $\rightarrow$  depends on boundary strategies.

### Derivatives and Anteage -

Cross Entropy Loss -  $\frac{1}{c} \sum y_i \log \hat{y}_i$

Softmax activ<sup>n</sup>. -  $\frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$

### AlexNet

- Conv1  $\rightarrow$  Max Pool  $\rightarrow$  Norm1  $\rightarrow$  Krizhevsky
- ZF Net
- VGG  $\rightarrow$  Simonyan and Zisserman.

### Why use smaller filters?

$\rightarrow$  less no of computation.

$\rightarrow$  same effective receptive fields as  $7 \times 7 \rightarrow$  (stack of  $3 \times 3$  filters)

## GoogLeNet

- Deeper net, computational efficiency.
- Inception module
- Stacked up incept<sup>n</sup> module.
- 22M parameters  $\rightarrow$  due to eff. incept<sup>n</sup> module.
- Parallel filters ops using multiple filter sizes.

### 1x1 convol<sup>n</sup>.

- Feature fusion throughout the channel.
- 1x1 conv. + pooling then apply filter.
- 

$\rightarrow$  (Szegedy)

$\rightarrow$  Effect on the no. of layers in convolutional net  $\rightarrow$  ResNet.

### Vanishing gradient problem:-

- Deeper  $\rightarrow$  the multiplic<sup>n</sup>. goes  $\rightarrow$  gradient vanishing.
- Soln.  $\rightarrow$  creating residual network.

### Better weight init.

- Glorot / he initialization.
- Domain knowledge
- Locally trained using supervised learning.

### Regulariz. methods:-

- a) L2 - weight decay, L1 - weight decay
- b) Gradient descent.

low level features  $\rightarrow$  dataset agnostic.

ImageNet  $\rightarrow$  1000 classes.

## Transfer learning

- Pretrained models  $\rightarrow$  low level training.
- Using knowledge learned over a certain task, train them specifically for other tasks.
- Pretrained models  $\rightarrow$  reduce training time  $\rightarrow$  frozen layers  $\rightarrow$  high level layers only learn, low level layers are frozen.
- Choice depends on dataset size

Which model to select

Input $\rightarrow$	Source task general layers	target task Special layer
---------------------	-------------------------------	------------------------------

$\rightarrow$  An int. layer

Code understanding  $\rightarrow$  finetuning

- Created `UnNormalize class.`

a) `Init`  
    i) mean  
    ii) std dev.

b) `call`  
    i) self  
    ii) tensor ( $C, H, W$ ) to be normalized.

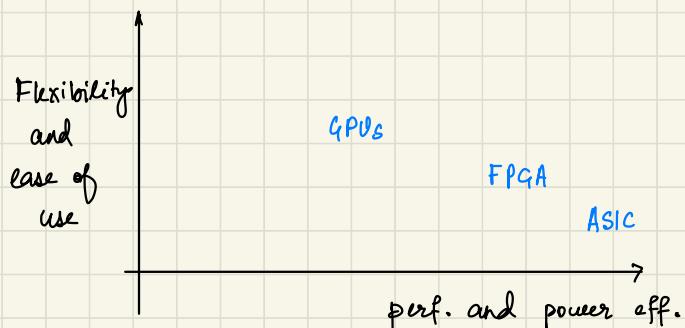
$(t - \Delta + m) \rightarrow \text{normalize}$

c) `dataLoader.mnist` (batch size, num workers = 0, valid<sup>m</sup>:  $\text{fr}^n \rightarrow \text{None}$ ,  
default  $\leftarrow$  (train . transforms = None  
test transforms = None))

if train transforms is None

train.transforms  $\rightarrow$  toTensor()

- computational complex fns. on hardware for more power efficiency and performance



- IC that can be programmed for different algorithms
- Architecture is inherently parallel.
- LUTs  $\rightarrow$  Look up tables.
  - Comb. logic blocks
  - Memories  $\rightarrow$  block memories
  - DSP.
- Implement CNN arch. on FPGAs
- DNN or CNN inference  $\rightarrow$  feed forward computation on FPGAs.
- Convolutional ops  $\rightarrow$  main ops.
- On chip memory not sufficient  $\rightarrow$  off chip needed  $\rightarrow$  latency issues.
- Meters  $\rightarrow$  quality of implement.

- a) Accuracy
- b) Throughput (Gflops)
- c) latency
- d) Energy and Power
- e) Bandwidth → Memory
- f) Cost

Reduce size of operands for storage / compute

- a) Bit width redn.
- b) Reduce number of operands → by coding.
- c) Non-linear

Floating point → fixed point (integer)

- Stay on chip ( $1/100 \times$  power)

Main obj → accelerate convolution operations

↓  
do feature extrn<sup>t</sup>.

+ fully connected layers → classifi?

- Winograd minimal filtering
- Arch. → i) Direct convolution (suited for large convoln.)  
ii) FFT convoln. (input size = kernel size)  
iii) Winograd filtering (suited for small kernel size).
- Gemm accelerator
- Blocking Gemm accelerator
  - Better DRAM bw.

Winograd algorithm → Gemm → Blocking technique

- Select tiling  $\rightarrow$  transform<sup>n</sup>. ]  $\rightarrow$  block multiplications.  
filter  $\rightarrow$  transform<sup>n</sup>.

First layer  $\rightarrow$  kernel size  $\uparrow$   $\rightarrow$  direct convoln.

Next  $\rightarrow$  "  $\downarrow$   $\rightarrow$  transform<sup>n</sup>. + direct convoln.

- Based on memory available, and computational time tile sizes are chosen for each layer.

## Accelerating Sparse Neural Networks

- Sparse matrix in some format.
- Develop an arch.

Better algorithms or  
Better runtimes or  
Better hardware.

## RISC-V

- open source ISAs.
- Also 2 diff. processor design.
- 47 instrns. predefined + scalability.
- 

## Processors:-

i) Rocket core - 64 and 32 bit versions

- 64 - processor
- Main core + 64 core.
- in - order

ii) BOOM - out of order machine

## Chipyard

- Open source SoC platform
-

chipyard !!!  
↓  
doimma

Gemmni, SHA3, Custom Accelerators ,



eg. Sparse matrix multipliCn.

eg. MOSCON → sparse openV.

FPGAs using HLS

- HLS tools → C, C++, Python.
- Convolutn. as a matrix multipliCn.
- Xilinx Vivado HLS
- 

## Object detection

- Template matching ; Sliding window → correlation.
- But epic fail → many objects
- Gaussian pyramid construction
- Skippig some pixels
- filter → shrink

→ Different aspect ratio?

Window, aspect ratio,

- extract features → convolutn., HOG, SIFT
- classify features → fully connected, SVMs
- Run classifier at all scales  
→ several windows have extracted the same object.
- Post processing → multiple detectn., Non-maxima suppr.  
↓  
overlap ← intersectn. over union.
- Remove boxes with high overlap.

- $\text{IoU} \geq 0.5 \rightarrow \text{discard}$
- 

$\text{IOU}$  scores  $\rightarrow$  determine model's perf.

$\text{IOU}$  scores  $\rightarrow$  True positives :- detected + right pos.

False positive  $\rightarrow$  wrong class  
 $\text{IoU} < 0.5$   
 No overlap.

False negatives  $\rightarrow$  no predictn.

True negative  $\rightarrow$

precision  $\rightarrow$  determine relevant objects.  
 $\cap$

$$= \frac{\text{true positives}}{\text{true pos.} + \text{false pos.}}$$

recall  $\rightarrow$  over all the

$$= \frac{\text{true positives}}{\text{true pos.} + \text{false neg.}}$$



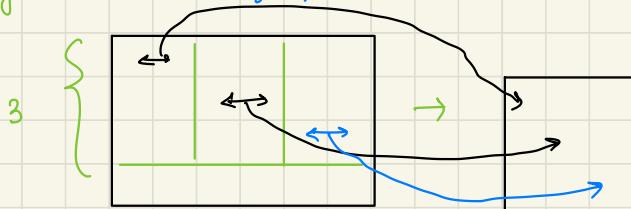
ground truth

Object detection

- Classification, regression, multiple object detn.
- variable size outputs
-

## R-CNN

- Region proposals  $\rightarrow$  down sample by same ratio.
- ROI pooling
- Mismatch b/w proj. and pixels.  
 $\Rightarrow$  Take round off  $\rightarrow$  region proposal  
 $\downarrow$   
losing inf.
- $4 \times 6$  ROI  $\rightarrow 3 \times 3$  ROI  
ROI pooling to compatible to fully conn. layers .  
etc...
- ROI pooling - make  $3 \times 3$  grid in  $4 \times 6$



- Pooling  $\rightarrow$  inf. loss  $\rightarrow$  object detn.  $\rightarrow$  okay  
 $\hookrightarrow$  not a convolution opn.

## Multitask loss

- multiple tasks  $\rightarrow$  loss  $\rightarrow$   $u \rightarrow$  true class name  
 $v \rightarrow$  ground truth boun box

$$t^u \rightarrow (t_x^u, t_y^u, t_z^u)$$

$$P$$

$$L(u, v, t, P) = L_{cls}(p|u) + \lambda[u \neq v] L_{loc}(t^u, v)$$

$$Loc(t^u/v) = \sum \text{smooth}$$

## Fast R-CNN loss

o

### YOLO

- o entire image at once in a CNN-
- o
  - div into cells
  - anchor boxes → predict B bb within cells
  - return bb above conf-threshold

$$\text{predict} = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_n \\ b_w \\ c_1 \\ c_2 \end{bmatrix} \quad \begin{array}{l} \xrightarrow{\hspace{1cm}} \text{prob. of bb w/ object} \\ \downarrow \\ \text{coord. of center of bb} \\ \downarrow \\ \text{size of cell width/height} \\ \downarrow \\ \text{prob. of obj. containing} \\ \text{obj 1 or obj 2} \end{array}$$

but if 2BBs in one cell then

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_n \\ b_w \\ p_c \\ b_x \\ b_y \\ b_n \\ b_w \\ c_1 \\ c_2 \end{bmatrix}$$

$$\text{Training loss} = \sum \text{weights} \times \text{loss}$$

## Semantic Segmentation

- Segment different things as diff. colours
- Mask-RCNN, UNets,

### UNets

- Skip connections, contracting path
- Input Img. tile  $\rightarrow$  pixel classific<sup>n</sup> problem  
↙
- each pixel present at the output  $\rightarrow$  downsample + upsample.
- global context  $\rightarrow$
- output segmentation map.
- proposed to segment biomedical images
- penalize edge pixels  $\rightarrow$  maximise contib<sup>n</sup> in loss fcn.
- merge obj. det<sup>n</sup> with semantic segmentation.
- RCNN + semantic segm.  
↳ Masked RCNN.
- 

### Mask-predictor

Mask RCN  $\rightarrow$  RoIAlign ROI pooling

- lose inf.  $\rightarrow$  lose semantic segm.  $\rightarrow$  no boundary off in proj.  
 $\rightarrow$  create values if you don't have values  
↑  
binary interp<sup>n</sup>.  
↗ +  
(RoIAlign)

→

## Tensorboard callback

- Weight histograms → probability dist. → measure perf. of gradient descent.
- Profiling → measure perf., timing, memory, loss of the models various parameters w.r.t another parameter.

## Tensorboard parameters

- write\_images -
- steps\_per\_second - log train steps per batch/epoch
- update\_freq -
- profile\_batch -
- embeddings\_freq -
- embedding\_metadata -
- histogram\_freq - compute activ. and weight histogram

↳ Training phase in tensorboard module

## Profiles using tensorboard

- Hyperparameters → very imp. → which are indep. but influence the model.
- Diff. hyperparameters → dropouts, optimizer → for each plot the graphs.  
→ visualiz".
- Character plot

XLA :- Linear Algebra Compiler

- Fast Optimizing compiler
- HLO IR →
  - ~50 kernels (~1500 kernels in TF)
  - dataset + shape + [layout]
  - Fusion → analysis and optm. phase

$$\begin{aligned} \mathbf{x}[\text{idx}] &= \mathbf{a} * \mathbf{x}[\text{idx}]; \\ " &= \mathbf{x}[\text{idx}] + \mathbf{y}[\text{idx}]; \end{aligned} \quad \left. \right\} 5 \text{ mem. calls.}$$

↓ Fusion

$$\mathbf{x}(\text{idx}) =$$

- HLO Before / After Optimiz. - XLA Graph optimizer.

- XLA HLO



- Optimized code
- AMP lib, MLIR compiler  
+ XLA comp. + Pytorch native lib