Compression Models are getting larger

Alexnet 2012 → 8 layers 1.4 GFlop
~16% Error.

2015 → 152 22.6 GFlop] **16x**
3.5% Error

The increase in size → increased accuracy
→ need for **acceleration**

- run a network faster (performance, runtime, inf/sec)
- run a network more efficiently (energy, monetary cost)
inf/J inf/$

Objectives of acceleration

For inference (previous lecture) → cascades
→ replication
(just focus on the forward pass)

Training: → fast forward passes
→ fast back-prop. (gradients)
→ update.

Today's lecture: How do we reduce cost for all three operations?

1) Using reduced precision arithmetic
→ increase the amount of data we can fit into a given processing unit and hence optimize util.

2) Compression → pruning. (reduce the ops we are performing)

3) Better algorithm. (fastest alg possible)
Low-rank aprox. of fundamental ops in DNNs.

Reduced Precision

→ We get reduced storage

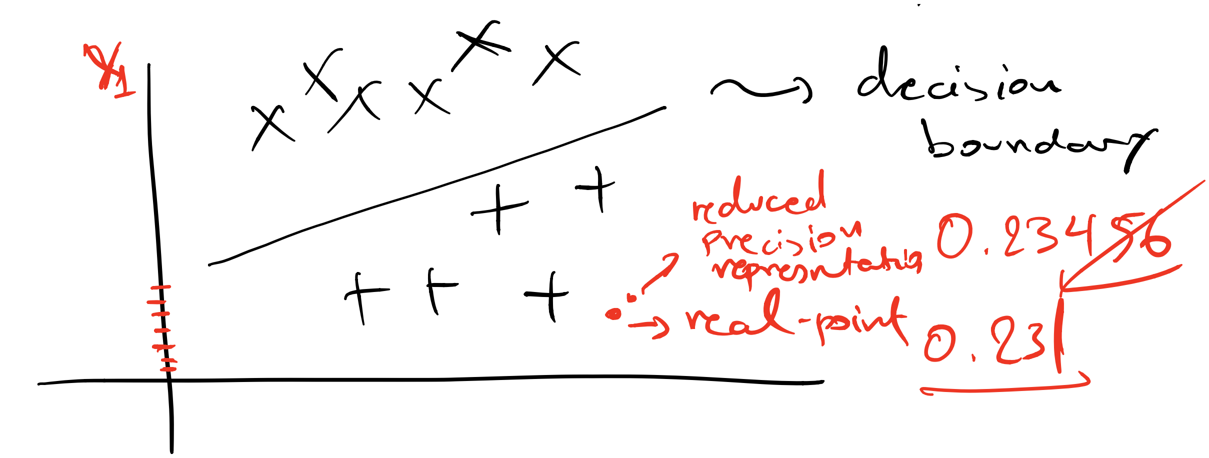
→ We get reduced energy (because we need less complex hardware "fewer transistors" when we use reduced precision.

→ Improved performane (speed)

→ Sometimes: has little effect on the accuracy.

Examples $\max(0, x)$, $x < 0$

→ precision is less important



DNN:

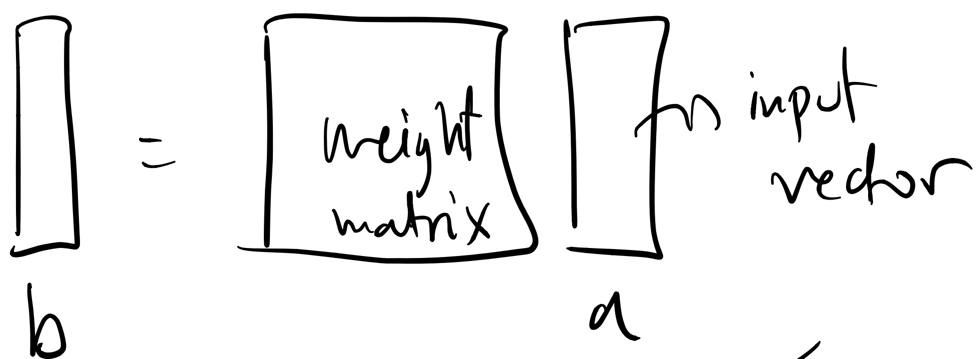
relu, or other non-lin.

$$b_i = f\left(\sum_j w_{ij}\, a_i\right)$$

output

params

input variables



→ Multiplication + Addition.

$b = f(Wa)$ forward

$W \downarrow$ Matrix, $a \downarrow$ vector

$W_{ij} = W_{ij} + \lambda \underbrace{a_i g_j}$ updates

$\lambda \downarrow$ learning rate

↪ Multiplication
Addition

Add, Mult units Range

FP32 (full precision)	$10^{-38} .. 10^{38}$	→	$6\,10^{-6}\%$
FP16 ↪ S E M	$6\times10^{-5} .. 6\times10^{4}$	–	0.05%
INT32	2×10^{9}		0.5
INT16	6×10^{4}		0,5
			0,5
INT8 ←	0–127		

Goal of low-prec arithmetic

Optimize the tradeoff between prediction accuracy of the network and

"vs." cost = f(energy, runtime, $)

Add in FP32 → 0.9 pJ $4k\ \mu m^2$

FP16 → 0.4 pJ → $1k\ \mu m^2$

Mult, Add → operations

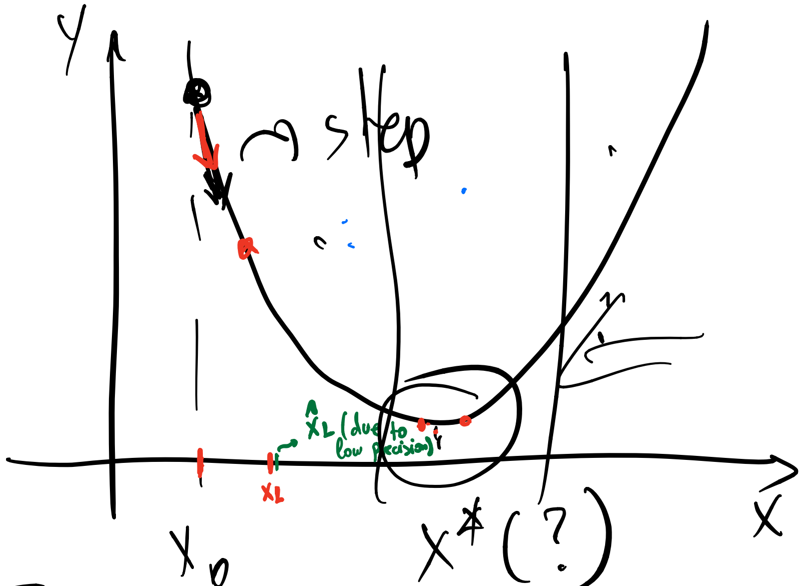
Parameters : { weights, inputs/outputs → raw data to the intermediate results; gradient, learning rate.

1) either use the same precision
2) mixed precision

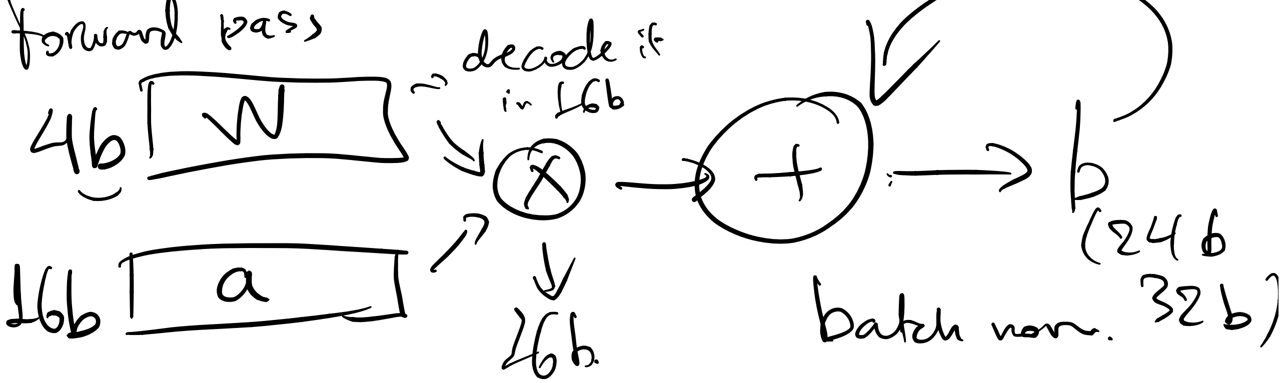
design.

Mixed precision says:
use different arithmetic for
your inputs/outputs/gradients/
weight etc.

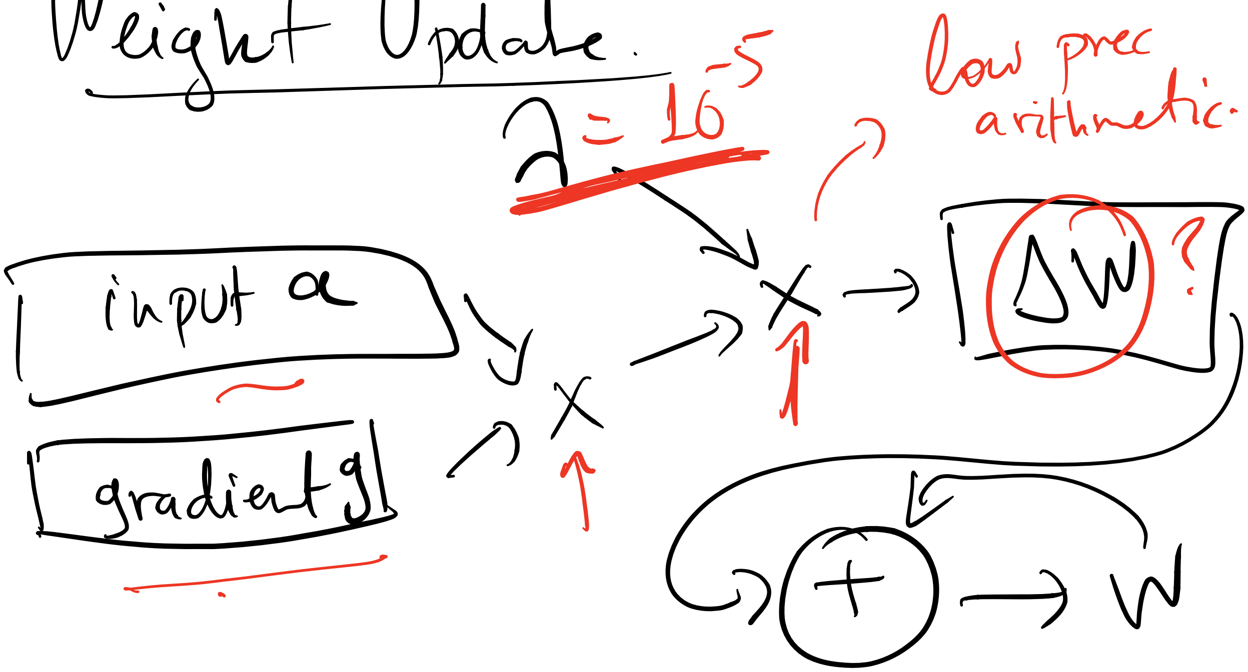
$$X_1 = X_0 + a\underline{g}$$



Forward pass



Weight Update

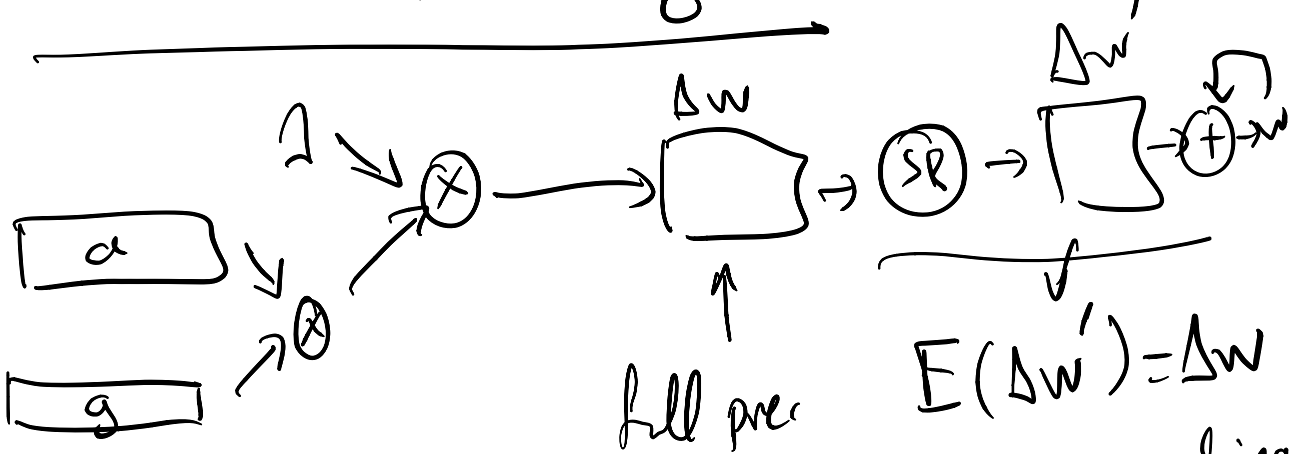


What is the effect of low precision here (assuming (quantization / rounding))

10^{-5} ← very low learning rate.

To avoid $\Delta W = 0$ due to low-prec.

Stochastic Rounding (SR)



$E(\Delta w') = \Delta w$

Reg. rounding

I want to add. 0.3 to 0, 100 times.

$0.3 \to 0$

$\downarrow$

$\underline{\underline{0}} \neq 30$

$$\text{Round}(x) = \begin{cases} \lfloor x \rfloor, & \text{w. prob. } 1 - (x - \lfloor x \rfloor) \\ \lfloor x \rfloor + 1, & \text{w. } \quad x - \lfloor x \rfloor \end{cases}$$

$$\text{Round}(0.3) = \begin{cases} 0, & 70\% \\ 1, & 30\% \end{cases}$$

$\underline{\underline{E[\text{Sum}] = 30}}$

Chris De Sa ISCA '17 discusses efficient stochastic rounding schemes

Summary

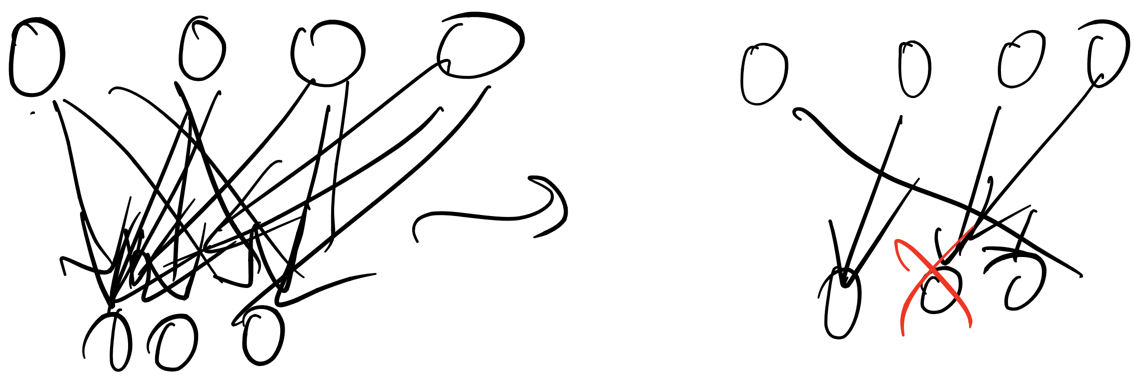
Reduced prec. → save memory space bandwith

→ Important points

High prec → $\boxed{\text{Batch norm}}$ → centr.

Stochastic Rounding during upd.

2. Pruning



after training.

Train w. full Connectivity

↓

Prune Connections

↓

Retrain

Lot.

After pruning (whatever params are left)

→ Quantize. Quantize.

orginal network →

Full connecti

↓

Prune

↓

Retrain

Pruning

same case →

?

reduce the size of my network.

Want to group weights w. similar values into one weight.

2.09, 2.12, 1.92

1.87

assigns these params into a cluster k → 2.00

int8

FP16

4 x FP16 vs. 4x int8 + FP16

This requires changing how one performs updates and learning.

W Matrix = W Matrix + λ [a] [g]

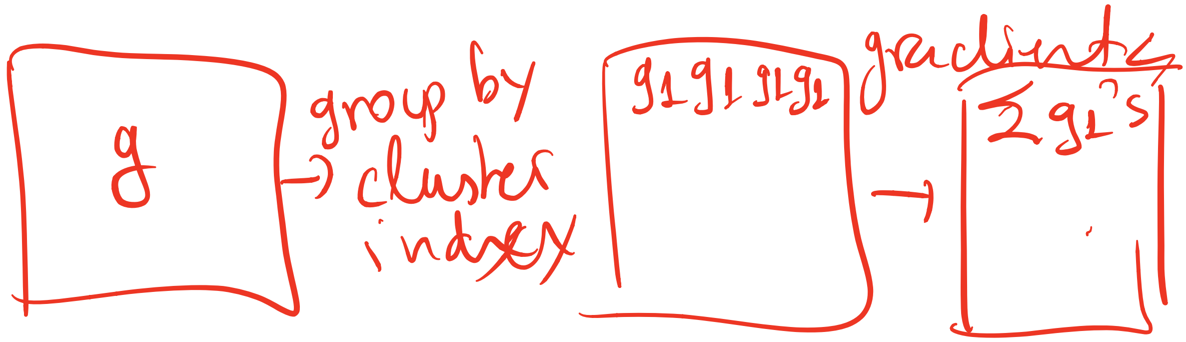
cluster index   a   centroids.

W Matrix = int8 / |W| [ ] → k FP16

cluster.

Now I only update the centroids and the indexes

They don't care about



Some W's are changing frequently

What does it mean tha W is changing frequently?

from an opt perspective. (we have not converged)

freq. weights → less bits

in-freq weights → increased bits

(Huffman Coding)

original Model → Pruning → Quantize → Huffman Coding

10x – 30x 50x

SqueezeNet (great paper)