

* Last class we were looking at basic considerations in Quantisation. Let us now look at applying Quantisation to LLMs / ML Models. ①

* Two paradigms ① PTQ (Post Training Quantisation.)
② QAT (Quantisation aware training.)

PTQ: Take a model and quantise it without training. / f +

QAT: Mix Quantisation during training.

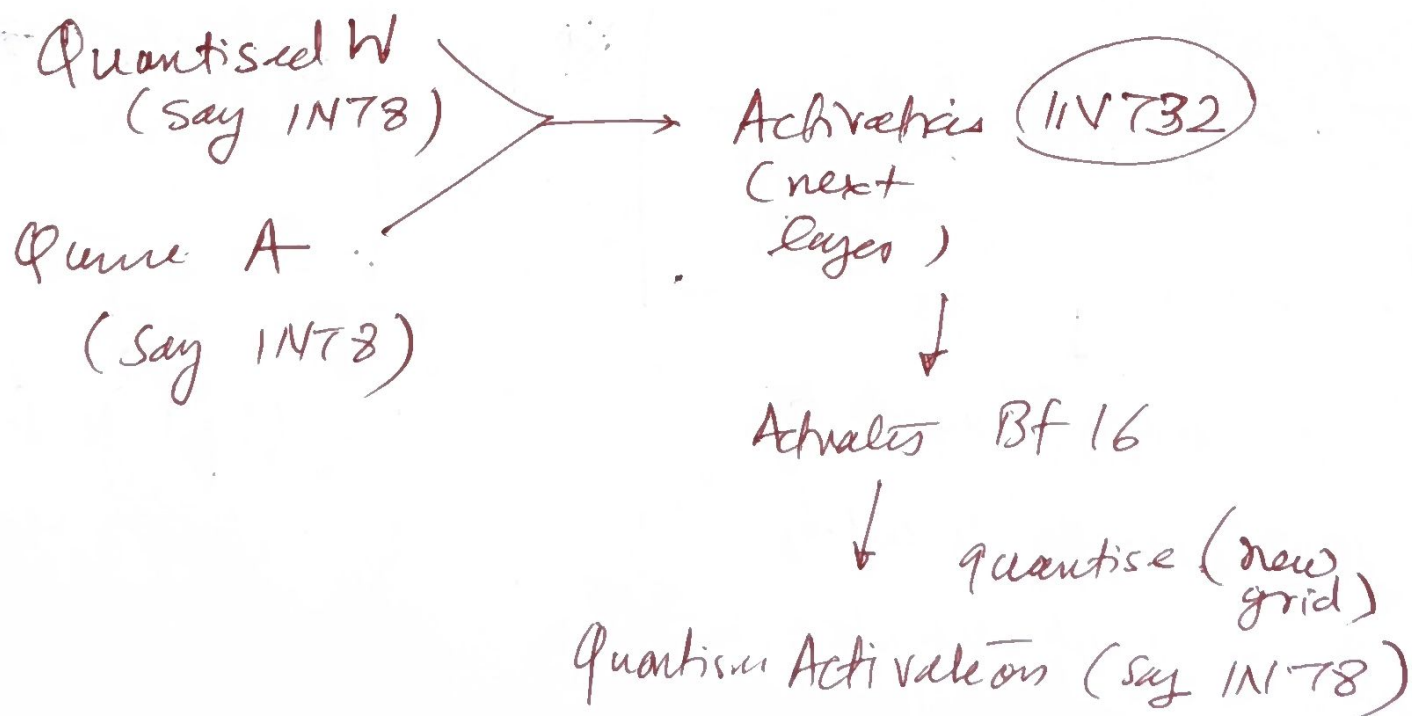
* When you quantise

① Weights only quantisation ($W_8, W_4, \dots$)
② W & A quantisation ($W_8A_8, W_4A_4, \dots$)
✓ Weights only: Activations are not quantised.

Workflow:

Quantised Weight $\rightarrow$ dequantise BF16 $\rightarrow$ use in downstream.

WA:



Methods of Quantisation

① QPTQ (Quantisation Counter part of SparseIT)

Recall: Optimal Brain Surgeon

X = calibration data.

W = weight

$L(X, W)$: loss on the calibration.

$$\min_{\hat{W} \text{ lies on quantisation grid.}} L(X, \hat{W})$$

① NP-Hard

⇒ ② Greedy Algo is suitable.

③ full Model is expensive to look at

⇒ ④ Local loss computation (linear layers)

↓ Lagrange's Multiplier

$$S_i = \min_{\hat{W}, \hat{W}_i = \text{Quant}(W_i)} L(X, \hat{W})$$

↓
Saliency of i

$$S_i = \frac{1}{2} \cdot \frac{\delta W_i^2}{(H^{-1})_{ii}}$$

↓
Saliency used to select "greedily" which element can be removed

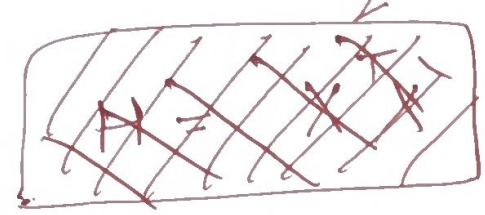
$$\delta W = \hat{W} - W$$

$$\delta W = \frac{\delta W_i H_{eq}^T}{[H^{-1}]_{ii}}$$

↓
Update for rest of the weights.

→ $d(X, \hat{W}) = \frac{1}{2} \| X \hat{W} - X W \|_2^2$

$n \times d$ $d \times d'$ true weight.



$d(X, \hat{W}) = \frac{1}{2} \sum_j \frac{1}{2} \| X \hat{w}_j - X w_j \|_2^2$

↳ broken down into row-wise terms

$H = \underbrace{X^T X}_{d \times d \text{ matrix.}}$

(each row)

$H^{-1} \times \begin{matrix} o(d^3) \\ \times \end{matrix}$
→ $O(d^3 \times d)$ Naive
→ $O(d^3)$ [Exclusion, identity]

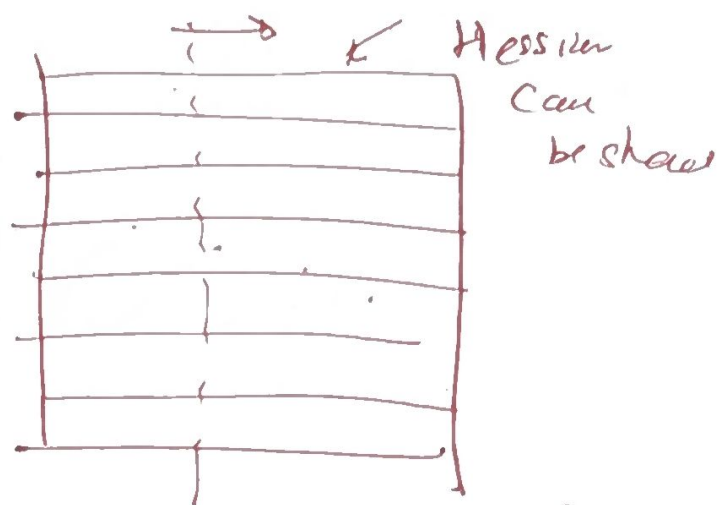
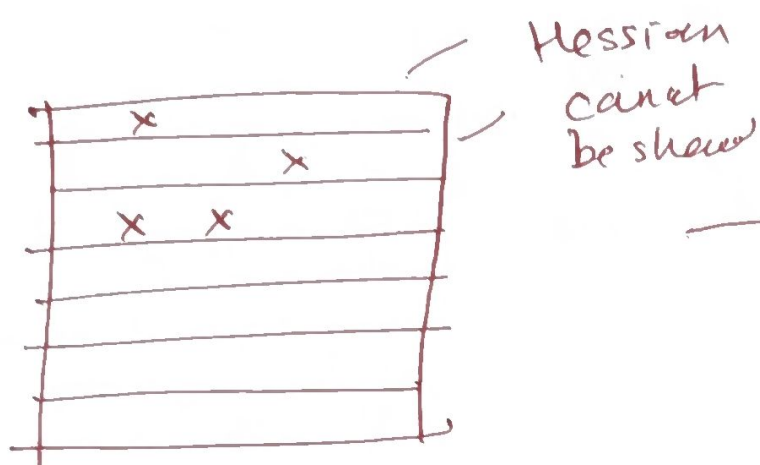
Algo :

- (1) Select based on salary
- (2) Remove and update
- (3) Repeat.

→ $O(d^4)$ for all rows.

Sparse CPT → share H^{-1} across the rows

↳ Arbitrary order OBS instead of greedy



$$O(d^4) \rightarrow O(d^3)$$

(Wk) AWQ: Activation aware Quantisation

GPTQ $\rightarrow$ Compensation algorithm. But the "grid" or quantisation itself is fixed.

AWQ $\rightarrow$ Adjustments within the granularity

Observation: "Some Activation channels have outliers / high magnitudes"

And protecting these weights is more important than others (Sensitivity wise)

A bit "Heuristic" view $\leftarrow$ Saliency as a measure of importance

If we use the OBD relaxation.

$$S_i = \delta w_i^2 (H_{ii})^2 \rightarrow \|X_{i\cdot}\|_2^2$$

i^{th} dimension

Magnitude of Activator does determine Saliency.

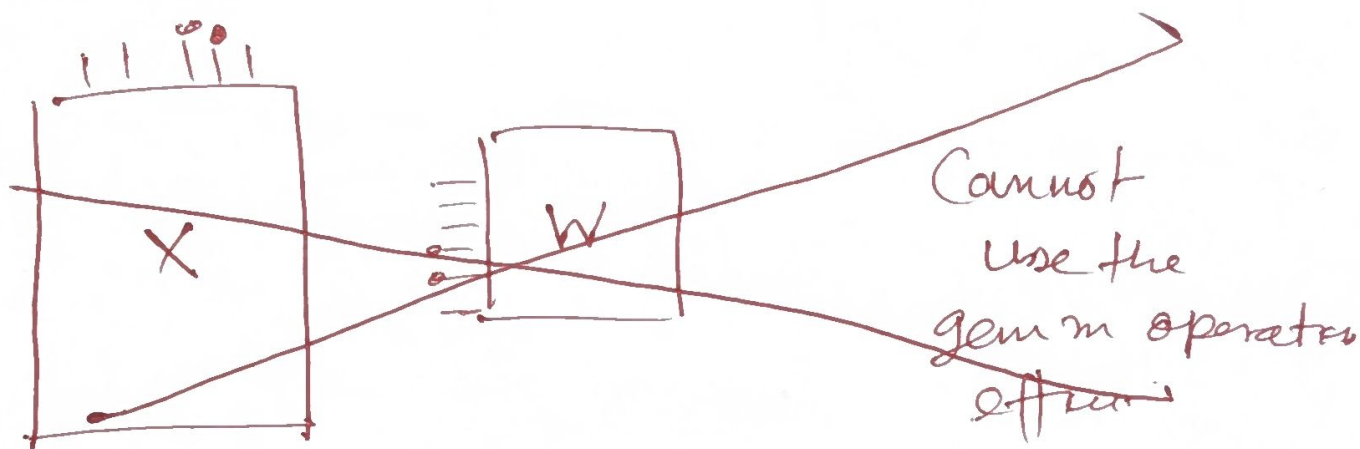
If $\|X_i\|_2^2$ is

you identify channels for which magnitude of X_i is large then, can how can we ensure that those weights get ~~better~~ smaller even

(A) to keep them in high precision.

↳ Mixed Precision.

↳ Difficult to implement efficiently



~~Channelwise scaling~~

(B) Use the same quantisation precision but reduce errors in quantised channels.

Idea →

$$X[i,:] * \frac{1}{S}$$

$$W[i,:] * S$$

look at a single element
say of x

w .

~~$\frac{[x]}{S} / x$~~

$$S > 1$$

x/w
original

$$x \cdot \omega$$

$$x \cdot \left[\frac{\omega}{\Delta} \right] \cdot \Delta$$

$$\frac{x}{s} \cdot \left[\frac{\omega \cdot s}{\Delta'} \right] \cdot \Delta'$$

Original

Quantised

If $s \cdot \omega$ affects scale.

If $s \cdot \omega$ does not affect scale.

$$x \cdot (\omega \neq \Delta) \quad \underline{\Delta' = s \Delta}$$

$$\Delta' = \Delta$$

Same as

$$x \omega \pm x \frac{\Delta}{2}$$

$$x \omega \pm \frac{x \Delta}{2s}$$

$$x \cdot \omega$$

$$x \cdot \left[\omega \pm \frac{\Delta}{2} \right]$$

$$\frac{x}{s} \cdot \left[\omega s \pm \frac{\Delta'}{2} \right]$$

$$x \omega \pm \frac{x \Delta}{2}$$

$$x \omega \pm \frac{x \Delta'}{2s}$$

General form : $n \times d$ and $d \times d'$

$$X \omega$$

$$\rightarrow X \cdot \text{diag}(1/s) \cdot \text{diag}(s) \cdot W$$

$$s^* = \arg \min d(s)$$

$$L(s) = \left\| \cancel{W \cdot \text{diag}} \left(X \cdot \text{diag}(s) \right)^{-1} \left(\text{diag}(s) \cdot \omega \right) \right\|_2$$

$$L(s) = \cancel{\Phi}.$$

$$\| X \omega - X \cdot \text{diag}(\vec{s})^{-1} \cdot \Phi(\text{diag}(\vec{s}) \cdot \omega) \|_2^2$$

AWQ

$$\vec{s} = S_X^\alpha$$

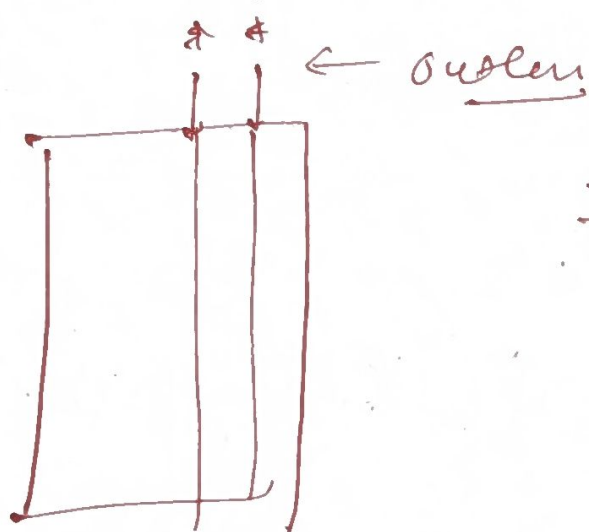
$$S_X[\vec{x}] = \underbrace{[X[:, i]]}_{\text{Average Aug.}} \quad (4)$$

Minimise one α

Smooth Quant

W_k, A_k

→ Activation has outliers and this actually leads to quantiser being hard.



Ideally: → channel with static granularity is inefficient

Why inefficient → Since this is the inner dimension of Matmul - cannot multiply efficient user existence.

→ ~~easy to modify~~

Smooth Quant → transfer hardness of Activation Quantisation to Weight Quantisation.

$$\begin{bmatrix} X \end{bmatrix} \begin{bmatrix} W \end{bmatrix}$$

$$XW \Rightarrow X \cdot \text{diag}(s)^{-1} \text{diag}(s) W$$

$$\bar{s}_j = \frac{\max(|X_j|)^\alpha}{\max(|W_j|)^{1-\alpha}}$$

α = migration strength

$$\alpha^* = \arg \min_{\alpha} L(\alpha)$$

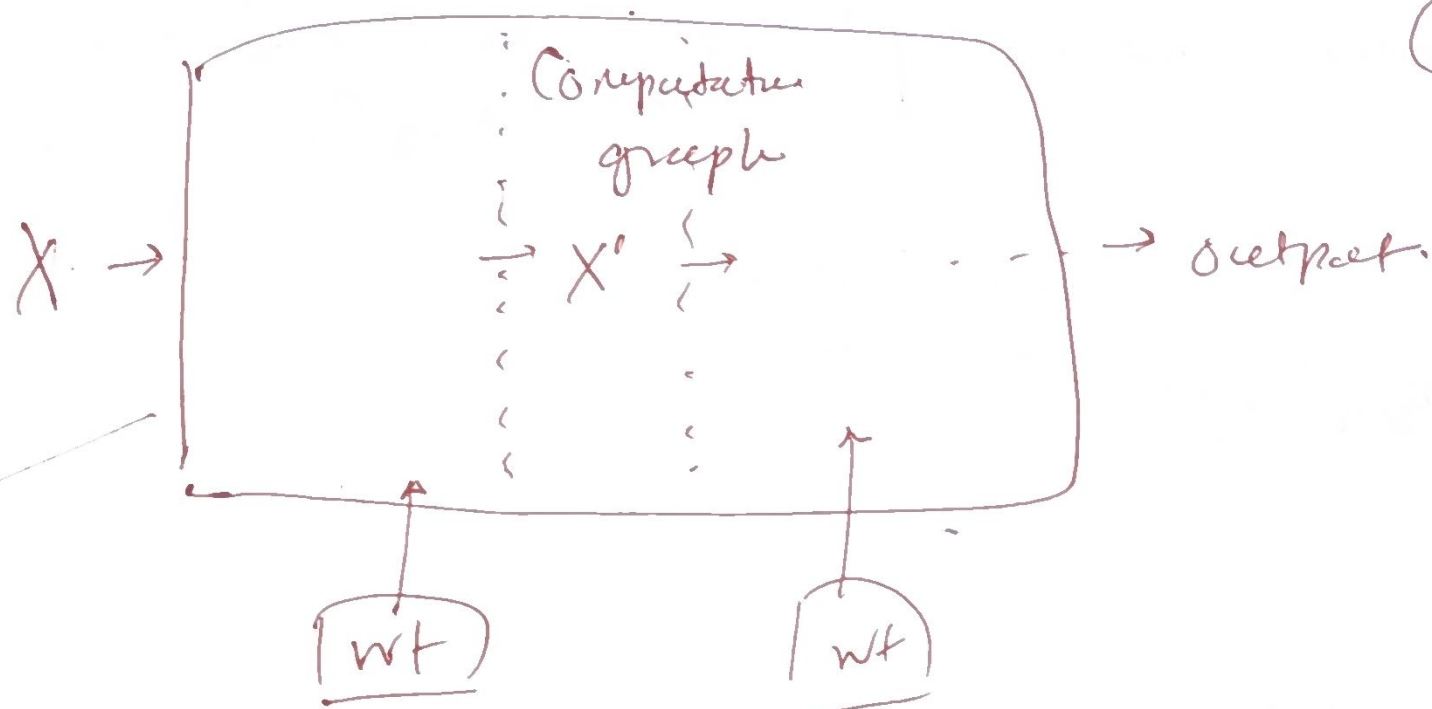
$$= \arg \min \left[\|XW - \Phi(X \cdot \text{diag}(s)^{-1}) \Phi(\text{diag}(s) \cdot W)\|^2 \right]$$

QAT:

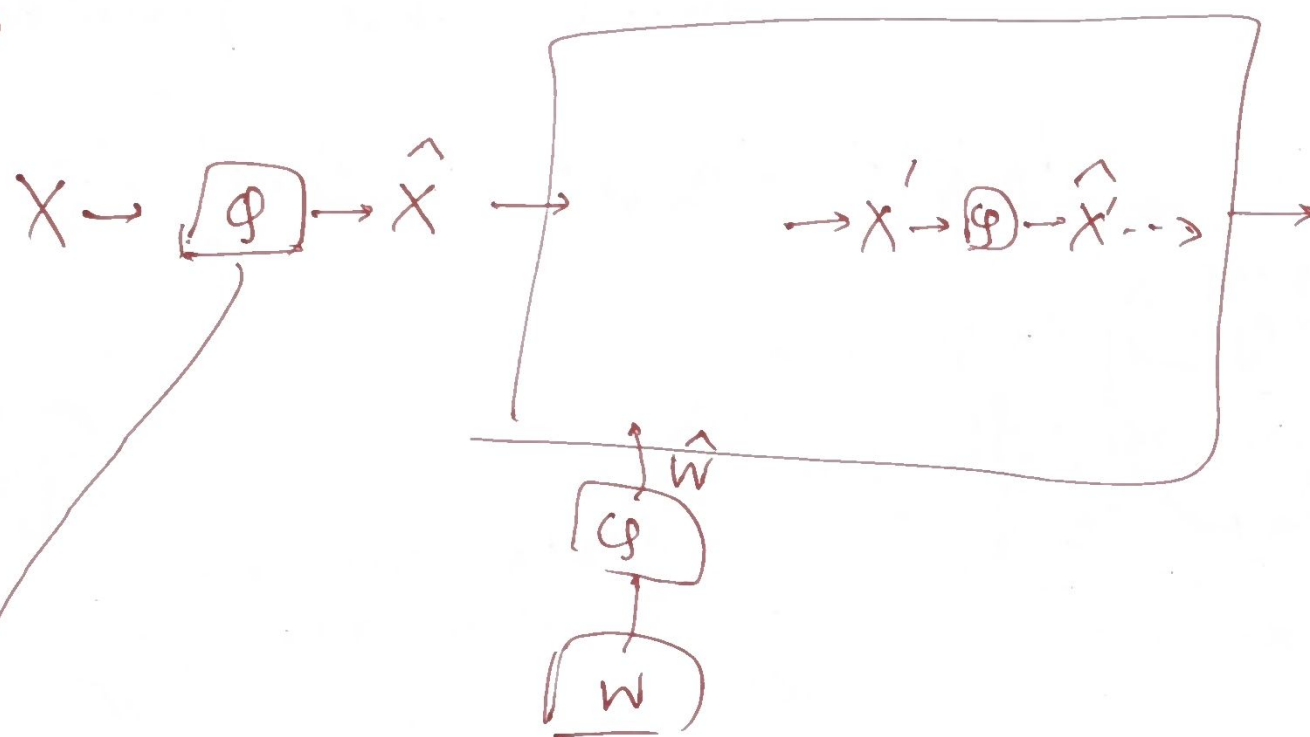
PTQ - can affect accuracy adversely

↳ What if we can train the model in quantized aware manner.

5



Add Fake Quantisation Models just after weights or activations.



$$\Phi(\hat{X}) = \Phi^{-1}(\Phi(X)) \quad [\text{traceful}]$$

$\downarrow$ $\downarrow$
 dequantise quantise

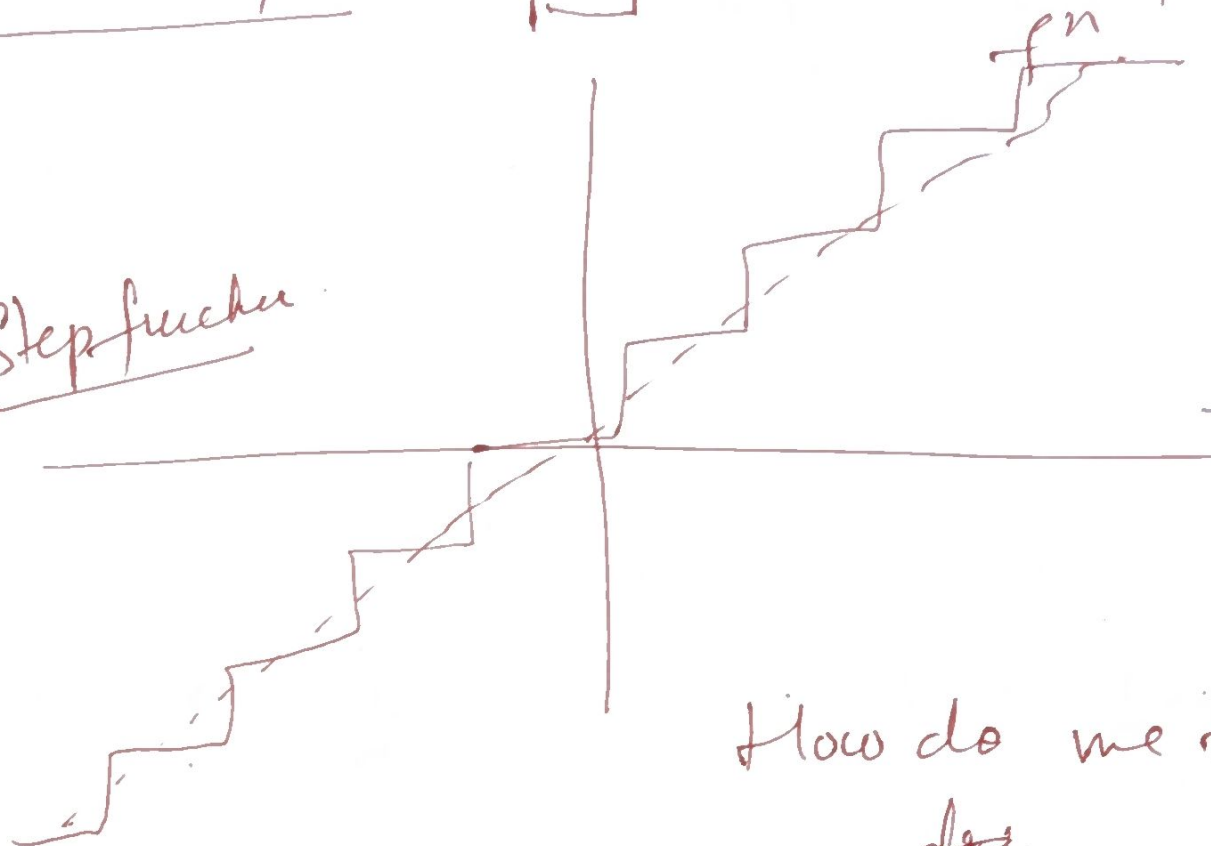
Forward pass $\rightarrow$ happens with quantised values.

Backward pass $\rightarrow$ Uses full precision wts and updates.

Backward pass

Φ is a non-differentiable

Step function



How do we ~~take~~
~~do~~

propagate the gradients?

Answer: Straight through Estimators...

$$y = \Phi(x)$$

$$\left(\frac{\partial \hat{y}}{\partial x} \right) = 1 \quad \leftarrow$$

Now we can
train Noomelley

Hardware supported formats

6

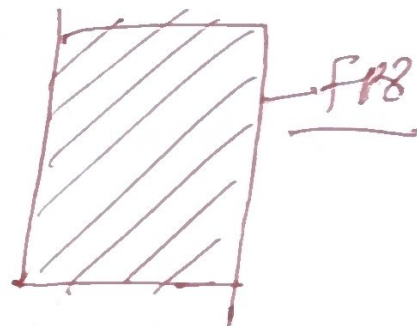
Bf16, FP16, FP32 ← full precision / half precision

① FP8

Two formats

~~E4~~ M3

E5 M2



Additional Scales

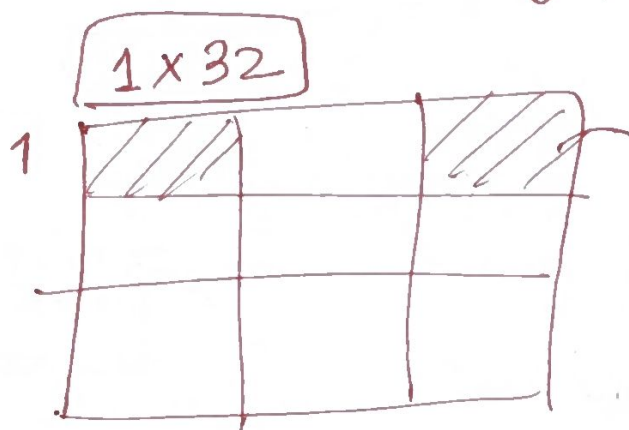
at tensor level FP32

~~FP8~~ FP32

②

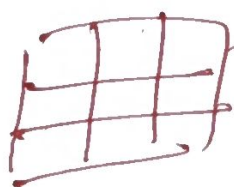
MXFP8

(Microscaling format - Smaller granularity)

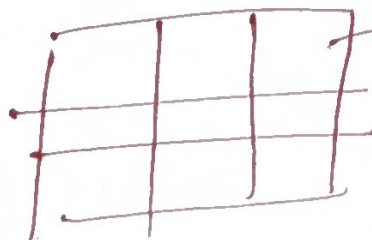


32 value groups
FP8

Scales
per
32 value
groups.

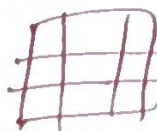


E8M0



1x16

FP4 E2M1



FP8 scalars
E4M3

③ NVFP4

~~Using low prec~~

Hardware Supported meaning?

Hardware
primitives such as

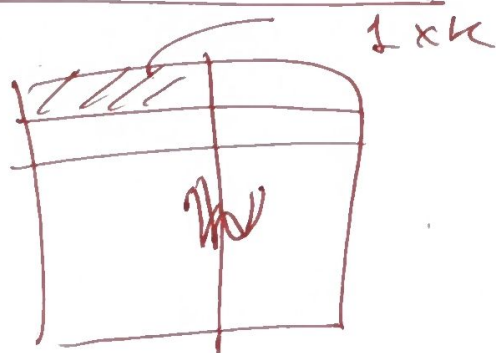
~~func1(tens1~~

→ MM (tensor 1, tensor 2, scale 1, scale 2)

exist Hardware recognises that tensor 1 comes
with scale 1

~~No overheads of~~

1D tiling + tiling issues



→ issue?
↓

$$y = Wx$$

in backward pass we need W^T !!

So 1D tiles are not usable in backward
pass

Solⁿ:

2D tiles NUFPT has two
versions

2D tiles are used while training