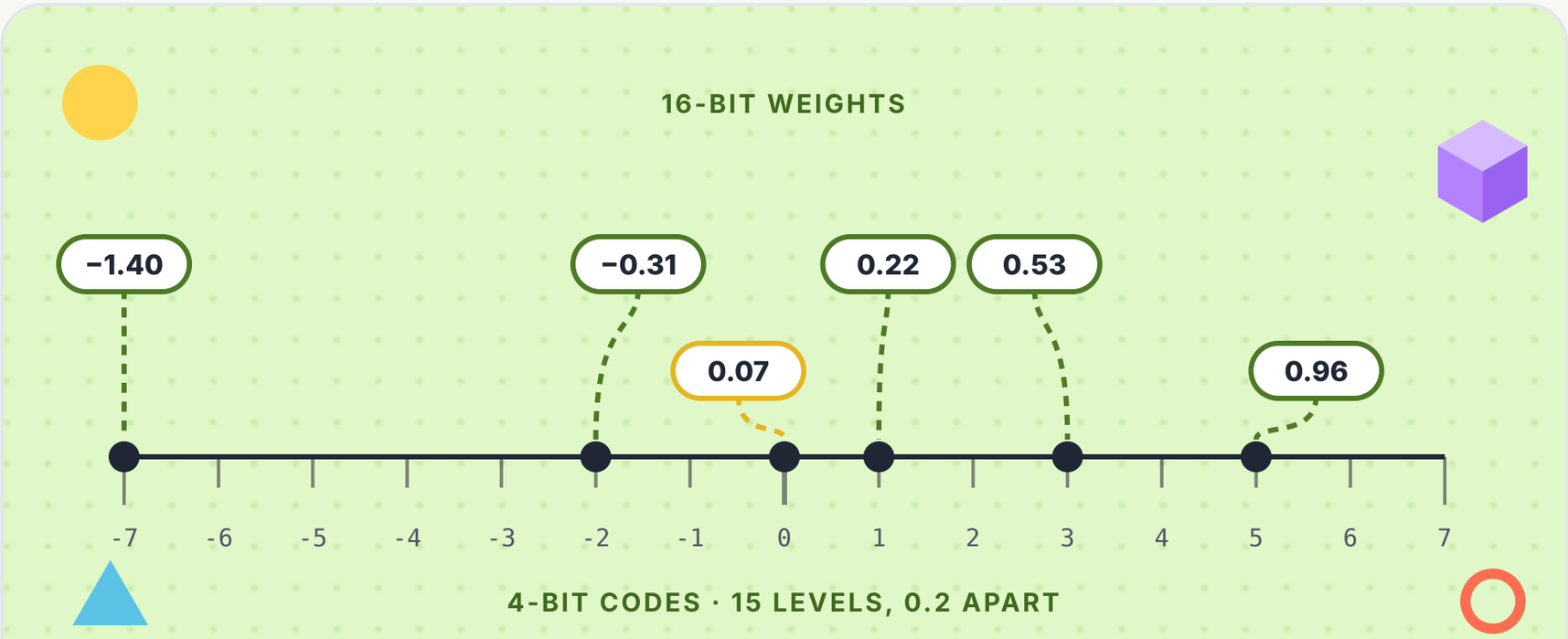


AI TUTORIAL · DAY 006 · Model compression

Quantisation, by hand

Store six weights in 4 bits, measure the error, then add one outlier.

7 steps · ~4 min · Only needs ÷ and round

Start lesson

Mohammad Ashfaq Ur
AI Tutorials · Daily AI Notes

THE BIG PICTURE

Quantisation is three moves

$$s = \max|w| \div q_{max} \qquad q = \text{round}(w \div s) \qquad \hat{w} = s \cdot q$$

General form r = S(q − Z) (Jacob et al. 2018). Symmetric absmax sets Z = 0, as in LLM.int8().

1

Scale

Find the largest weight and map it to the largest integer. That one ratio is the step size s.

`1.40 ÷ 7 = 0.2`

2

Round

Divide every weight by s and round to the nearest integer. Only the integers get stored.

`0.53 ÷ 0.2 = 2.65 → 3`

3

Dequantise

At run time, multiply back by s. The gap between ŵ and w is the rounding error.

`3 × 0.2 = 0.60`

s **scale**: one float shared by the whole block

q **code**: the small integer that gets stored

Z **zero point**: shift for lopsided ranges; 0 here

Continue

STEP 1 · SCALE

Let the biggest weight set the step

SIX 16-BIT WEIGHTS FROM ONE ROW

`0.53` `−1.40` `0.07` `0.96` `−0.31` `0.22`

Largest magnitude: **`|−1.40| = 1.40`**. It will land exactly on the top code.

FORMAT	BITS	CODES USED	MAX|W| ÷ Q_{max}	SCALE S
int4	4	−7 … 7	1.40 ÷ 7	**0.2**
int8	8	−127 … 127	1.40 ÷ 127	**0.011**

WORKED: INT4

`s = 1.40 ÷ 7 = 0.2`

`grid: −1.4, −1.2, … , 1.2, 1.4`

WHY 7, NOT 8?

4 bits give −8 ... 7. Using ±7 keeps the grid symmetric around zero, the same idea as ±127 in int8.

Toy setup: one block of six hand-picked weights · symmetric absmax · round to nearest.

Continue

STEP 2 · ROUND

Snap every weight to the grid

WEIGHT W	W ÷ 0.2	CODE Q	STORED AS
0.53	2.65	**3**	0011
−1.40	−7.00	**−7**	1001
0.07	0.35	**0**	0000
0.96	4.80	**5**	0101
−0.31	−1.55	**−2**	1110
0.22	1.10	**1**	0001

4×

SMALLER THAN FP16

16 bits per weight become 4, plus one shared scale per block

WATCH THIS ONE

0.07 ÷ 0.2 = 0.35 → 0

Any weight with |w| below s ÷ 2 = 0.1 becomes 0.

Continue

Mohammad Ashfaq Ur
AI Tutorials · Daily AI Notes

STEP 3 · DEQUANTISE

Multiply back and measure the damage

W	Ŵ = 0.2 · Q	ERROR
0.53	0.60	**+0.07**
−1.40	−1.40	**0.00**
0.07	0.00	**−0.07**
0.96	1.00	**+0.04**
−0.31	−0.40	**−0.09**
0.22	0.20	**−0.02**



Bar scale: ±0.1 = half a step, the most rounding can ever cost.

INT4 · S = 0.2

0.048 mean |error|

worst case 0.09, never above s ÷ 2 = 0.1

SAME SIX WEIGHTS IN INT8

0.0012 mean |error|

s = 0.011, about 39× less error for 2× the bits

Continue

Mohammad Ashfaq Ur
AI Tutorials · Daily AI Notes

TWIST · ADD AN OUTLIER

One big weight makes the grid 5× coarser

NEW WEIGHT ● 7.00 max|w| = 7.00 → s = 7.00 ÷ 7 = 1.0

BEFORE · S = 0.2

● 0.53→3 ● −1.40→−7 ● 0.07→0 ● 0.96→5 ● −0.31→−2 ● 0.22→1

AFTER · S = 1.0

● 0.53→1 ● −1.40→−1 ● 0.07→0 ● 0.96→1 ● −0.31→0 ● 0.22→0 ● 7.00→7

Mean error on the six original weights jumps from **0.048** to **0.252**.

QUICK CHECK

With the outlier in the block, how many of the six original weights get stored as 0?

A None

B 1

C 3

D 5

Check

Answer on the next slide →

Mohammad Ashfaq Ur
AI Tutorials · Daily AI Notes

Answer: C, three of them

0.07, −0.31 and 0.22 all sit within 0.5 of zero once s = 1.0, so they round to **0**.

TWO COMMON FIXES

Keep outliers in 16-bit. LLM.int8() runs outlier feature dimensions (magnitude ≥ 6) in 16-bit and the rest in int8.

Use small blocks. One scale per block, so an outlier only coarsens its own block.

TAKEAWAY

The scale is set by the largest value in the block. One outlier decides how precisely everything else gets stored.

CHECK IT IN CODE

```
import torch
w = torch.tensor([.53,-1.4,.07,.96,-.31,.22,7.])
s = w.abs().max() / 7
(w / s).round()
# tensor([ 1., -1., 0., 1., -0., 0., 7.])
```

WHERE THIS SHOWS UP · DAY 005

Gupta et al.: after 4-bit quantisation of the same heads, the share of damage reaching the output rose from **0.55** (Pythia-410M) to **0.91** (12B).

REFERENCES

arxiv.org/abs/1712.05877
arxiv.org/abs/2208.07339
arxiv.org/abs/2611.07891



Mohammad Ashfaq Ur
Follow for daily AI notes and tutorials
ashfaqfardin.dev