Connect with me = https://www.linkedin.com/in/miraj-deep-bhandari-624bb0263/

Quantization in llms:

Quantization is the process of converting the data from higher memory format to lower memory format.

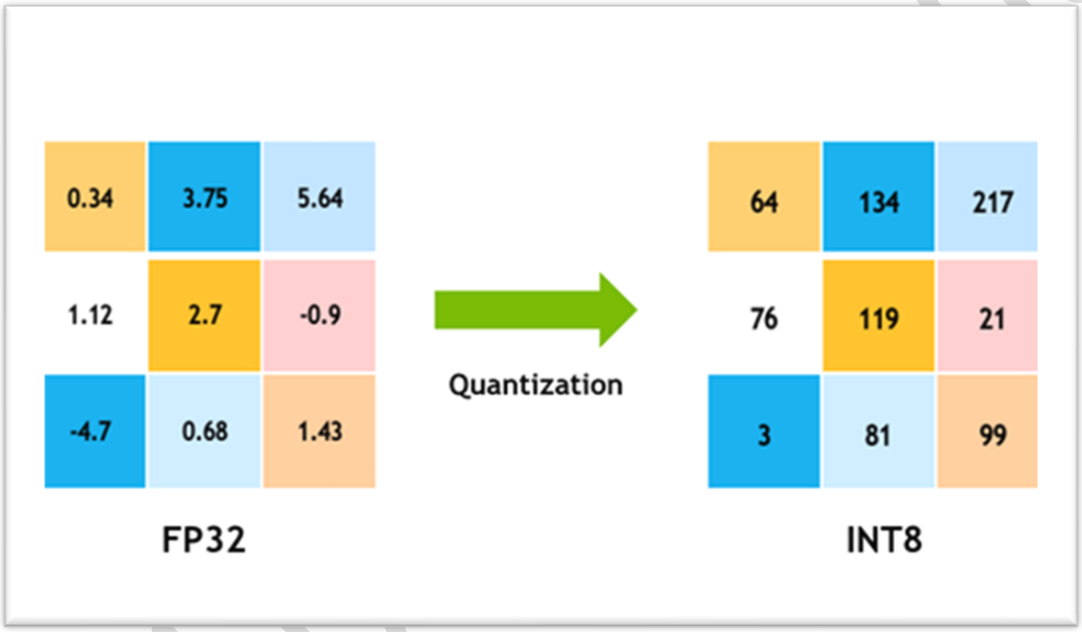


Figure 1: Quantization of Numbers from FP32 into INT8

Connect with me = https://www.linkedin.com/in/miraj-deep-bhandari-624bb0263/

Why do we need Quantization ?

We need quantization to store high-parameter models in our RAM. Without the concept of quantization, it's impossible to store higher GB models in our computer RAM or GPU RAM.

By default, the weights and biases (parameters) of the LLM are in memory format of 32bit or FP32 datatype. Suppose we want to load a language model with 7 billion parameters into our RAM, let's see how much memory in RAM we require:

Lets Visualize It:

- 1 GB = 8 × 10^9 bits
- 1 bit = 1/8 × 10^9 GB

Each parameter requires 32 bits of memory, so:

- 1 parameter = 32 bits

For 7 billion parameters:

- 7 billion parameters = 7 × 10^9

Memory required for 7 billion parameters in bits:

$$= (7 \times 10^9) \times 32 \text{ bits}$$
$$= 2.24 \times 10^{11} \text{ bits}$$

To convert 2.24 × 10^11 bits into gigabytes (GB):

$$= \frac{2.24 \times 10^{11} \text{ bits}}{8 \times 10^9 \text{ bits/GB}}$$
$$= 28 \text{ GB}$$

We need 28 GB of RAM to store the model for inference which is impossible for normal use case computer devices.

Connect with me = https://www.linkedin.com/in/miraj-deep-bhandari-624bb0263/

Some Concepts and Terminologies in Quantization and Memory Structures :

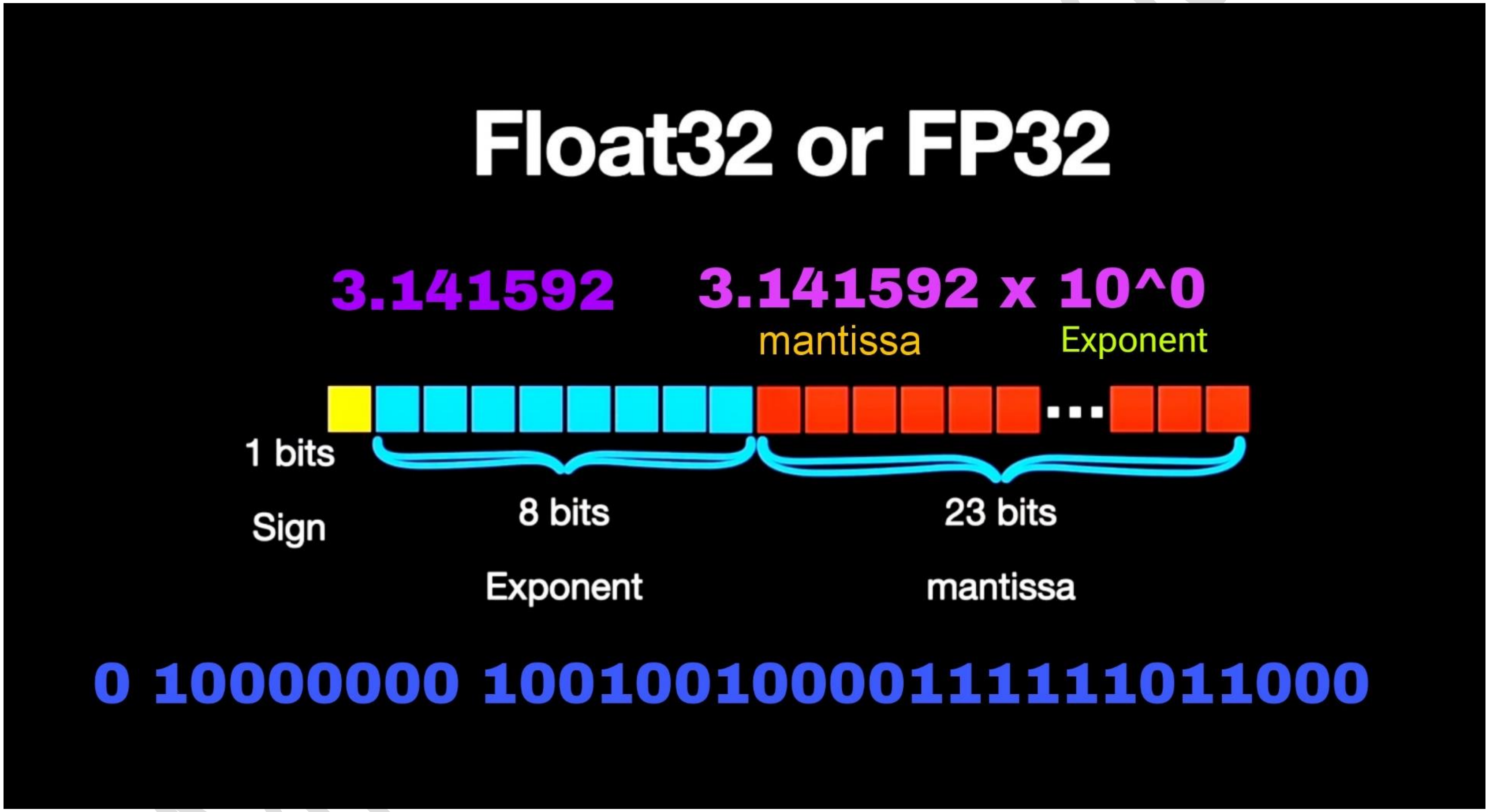
Types of Memory Formats:

- **FP32 :** (Floating Point 32), also known as Single/Full Precision, uses 32 bits of memory.
- **FP16 :** (Floating Point 16), also known as Half Precision, uses 16 bits of memory.
- **BF16 :** (Brain Floating Point 16), also known as Half Precision, similar to FP16, also utilizes 16 bits of memory but is optimized for specific computational tasks, particularly in artificial intelligence and machine learning applications.
- **Int8 :** (Integer 8) uses 8 bits of memory for numbers. It is useful for applications where memory efficiency and a relatively small range of values are important, such as in neural networks quantization and certain signal processing tasks.
- **FP4 :** (Floating Point 4) uses 4 bits of memory to store floating numbers.
- **NF4 :** (Normalized Float 4) normalizes floating-point numbers and uses 4 bits of memory to store them. NF4 performs better than FP4 experimentally.

Connect with me = https://www.linkedin.com/in/miraj-deep-bhandari-624bb0263/

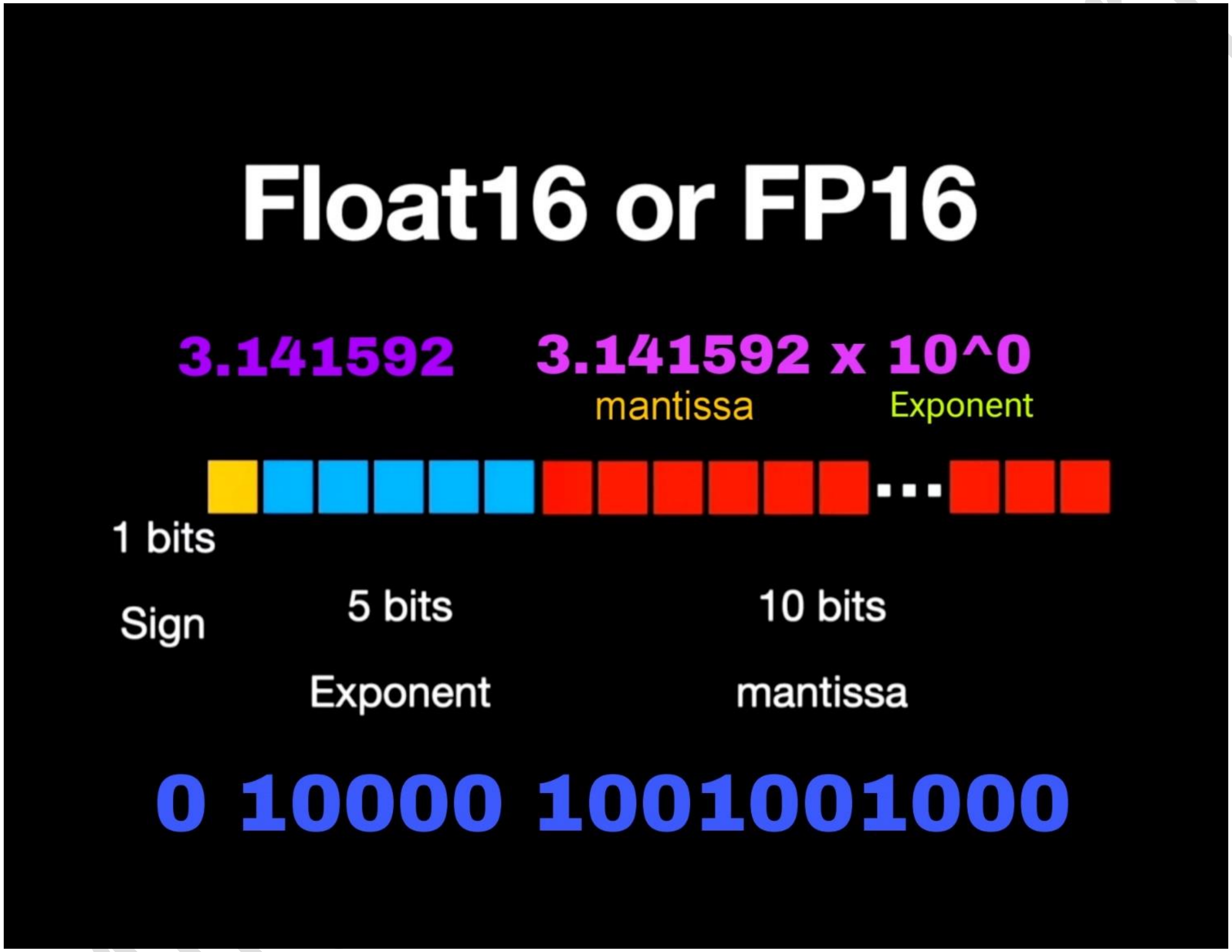
Lets see how does this memory Format Store data :

FP32 (Floating Point 32) :



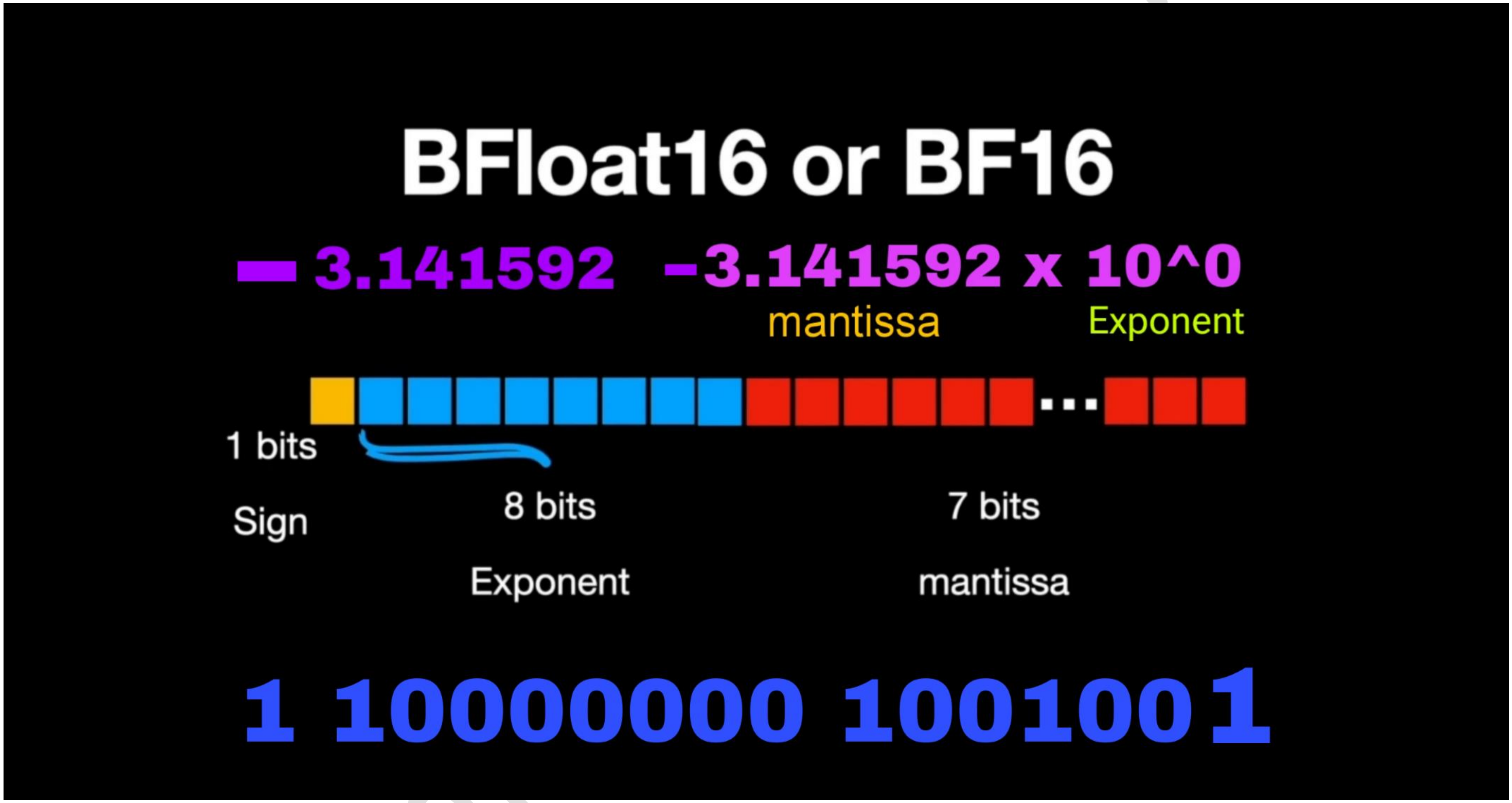
Connect with me = https://www.linkedin.com/in/miraj-deep-bhandari-624bb0263/

FP16 (Floating Point 16) :



BF16 (Brain Floating Point 16) :

Connect with me = https://www.linkedin.com/in/miraj-deep-bhandari-624bb0263/



Int8 (Integer 8) :

Connect with me = https://www.linkedin.com/in/miraj-deep-bhandari-624bb0263/

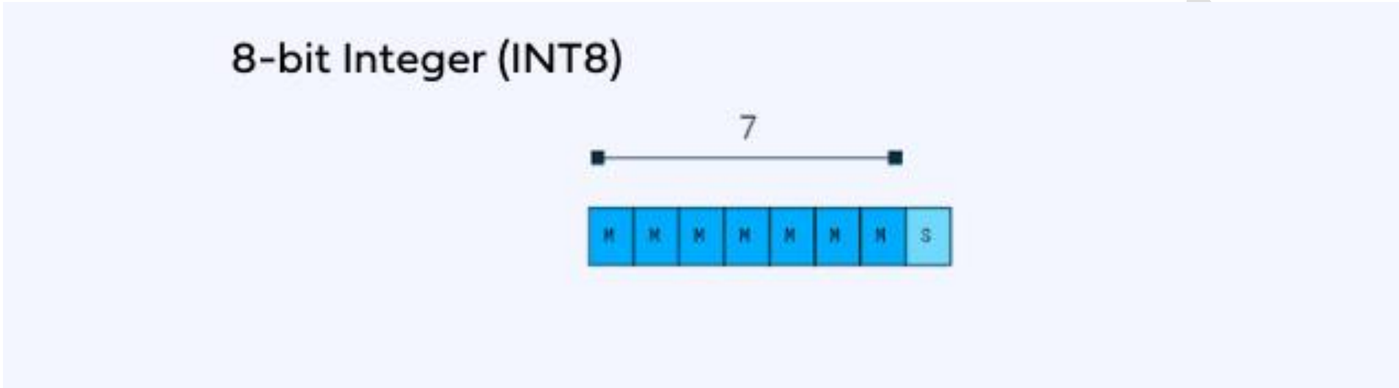
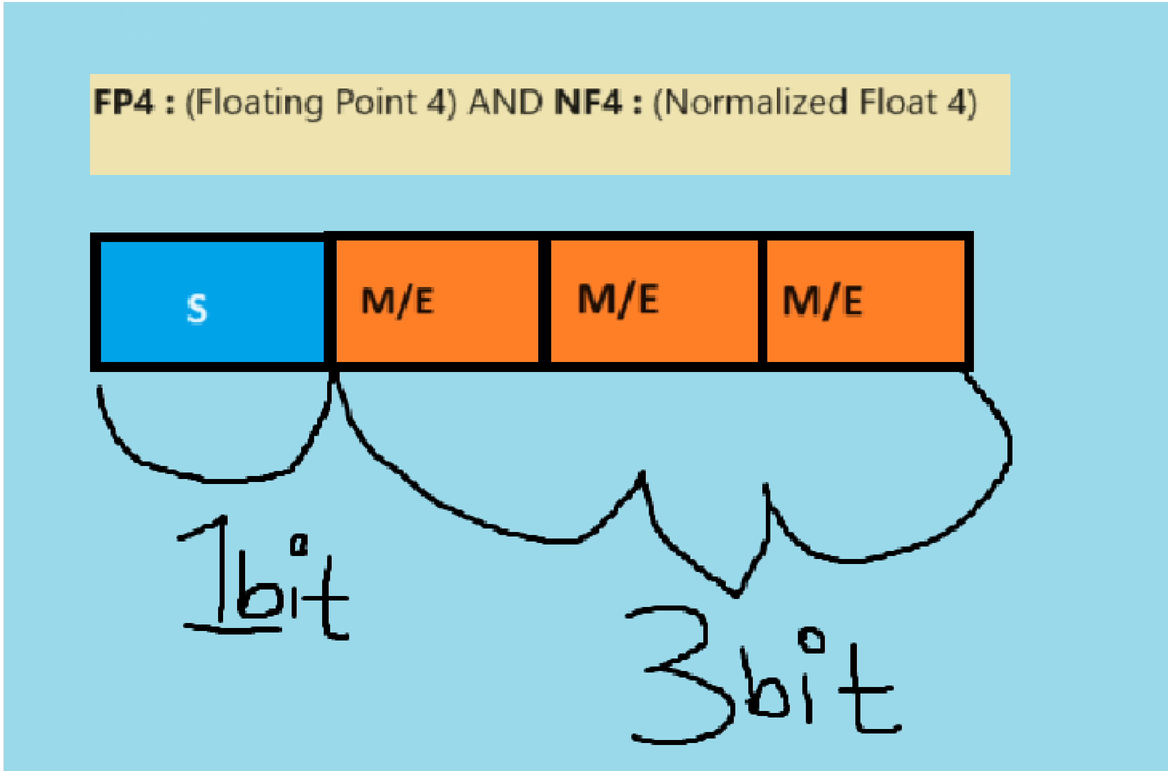


Figure 2: Here m is for mantissa and s is for sign

FP4 (Floating Point 4) and NF4 (Normalized Float 4):

Connect with me = https://www.linkedin.com/in/miraj-deep-bhandari-624bb0263/



In the FP4 and NF4 memory formats, one bit is reserved for the sign, and 3 bits are used by the mantissa or exponent. It's not fixed; sometimes, 2 bits are taken by the mantissa, and the remaining one bit is used by the exponent. There is no fixed format, memory try itself best combinations of different mantissa/exponent combinations.

Connect with me = https://www.linkedin.com/in/miraj-deep-bhandari-624bb0263/

Now Lets see how Quantization Occurs Mathematically:

How to perform quantization:

1) Symmetric Quantization:

In this quantization the min value of our set is perfectly align with the min value of quantization set.

let's perform quantization:

$[0.0 - 1000.0] \rightarrow$ uint8

The range of number in uint8 is $[0 - 255]$

So,

$[0.0 - 1000.0] \rightarrow [0 - 255]$

FP32 uint8

x

0.0 1000.0

q

0 255

Now,

let's calculate scale factor first,

$$\text{Scale} = \frac{x_{max} - x_{min}}{q_{max} - q_{min}} = \frac{1000.0 - 0.0}{255 - 0} = 3.92$$

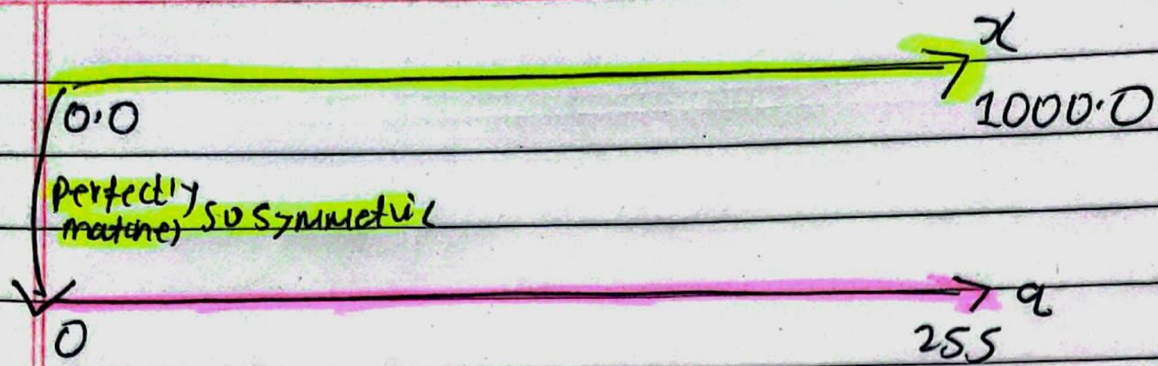
let's check it's symmetric or not

$$\Rightarrow \text{round}\left(\frac{\text{min no of ourset}}{\text{scale factor}}\right) + (\text{no of req no to get min value of quantization set})$$

$$\Rightarrow \text{round}\left(\frac{0.0}{3.92}\right) + (\text{num})$$

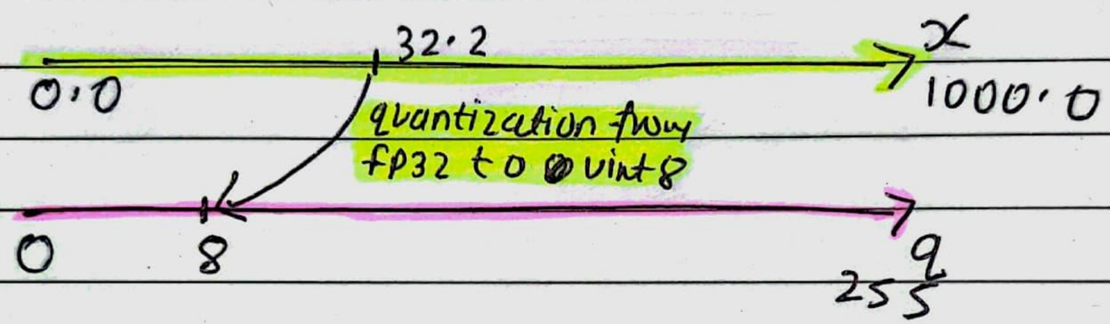
$= 0 + \boxed{0}$ - 0 is added since we already got (q) min value.

= It is symmetric



Now, let convert fp32 (32.2) to uint8

$$\Rightarrow \text{round}\left(\frac{32.2}{3.92}\right) = 8$$



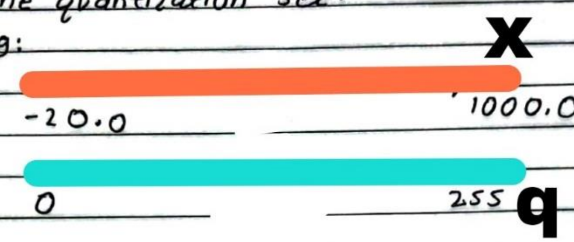
Connect with me = https://www.linkedin.com/in/miraj-deep-bhandari-624bb0263/

Date ______
Page ______

(2) Asymmetric uint8 quantization:

In this quantization the min value of our set is not perfectly align with the minvalue of the quantization set.

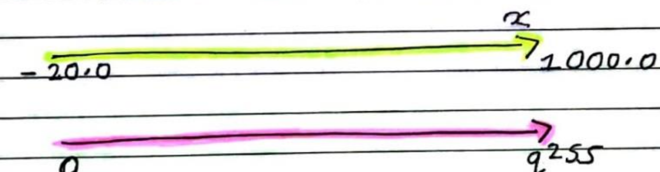
eg:



let's perform quantization:

$[-20.0 — 1000.0] \longrightarrow [0 - 255]$

FP32   uInt 8



Now,

$$\text{Scale} = \frac{x_{max} - x_{min}}{q_{max} - q_{min}} = \frac{1000.0 - (-20.0)}{255 + 20.0} = 4.0$$

lets check its is symmetric or not

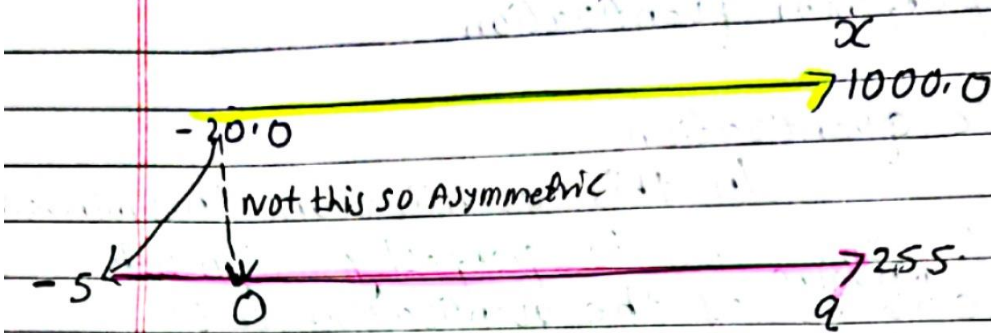
$\Rightarrow \text{round}\left(\frac{\text{min no of our set}}{\text{scale factor}}\right) \pm$ (no that is need to get minvalue of quantization set)

$\Rightarrow \text{round}\left(\frac{-20.0}{4.0}\right) \pm (\text{num})$

$\Rightarrow$ -5 + 5 (adding 5 to get min value of quantization set $\therefore$ zero point = 5)

Asymmetric

Connect with me = https://www.linkedin.com/in/miraj-deep-bhandari-624bb0263/



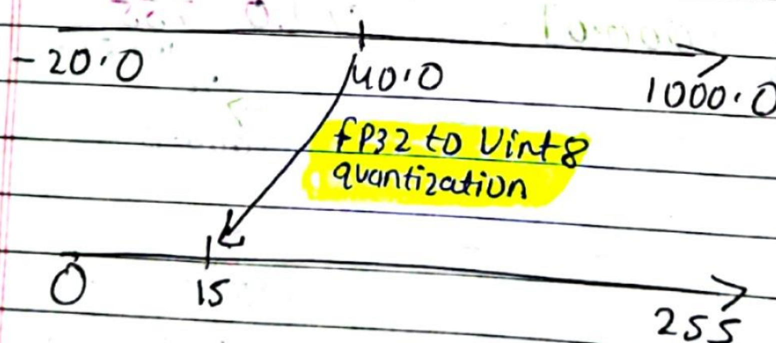
Now, let convert 40.0 (FP32) to uint8

$$\Rightarrow \text{round}\left(\frac{\text{num}}{\text{scalefactor}}\right) + \text{zeropoint}$$

$$\Rightarrow \text{round}\left(\frac{40.0}{4.0}\right) + 5$$

$$\Rightarrow 15$$

In figure



Connect with me = https://www.linkedin.com/in/miraj-deep-bhandari-624bb0263/

Different ranges numbers that comes under different memory format

Memory Format	Range of Values
uint8	0 to 255
int8	-128 to 127
uint16	0 to 65,535
int16	-32,768 to 32,767
uint32	0 to 4,294,967,295
int32	-2,147,483,648 to 2,147,483,647
uint64	0 to 18,446,744,073,709,551,615
int64	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float16	Approximately ±6.1E-05 to ±6.55E+04 (3 significant digits)
float32	Approximately ±1.4E-45 to ±3.4E+38 (7 significant digits)
float64	Approximately ±5E-324 to ±1.8E+308 (15 significant digits)