Range-extending

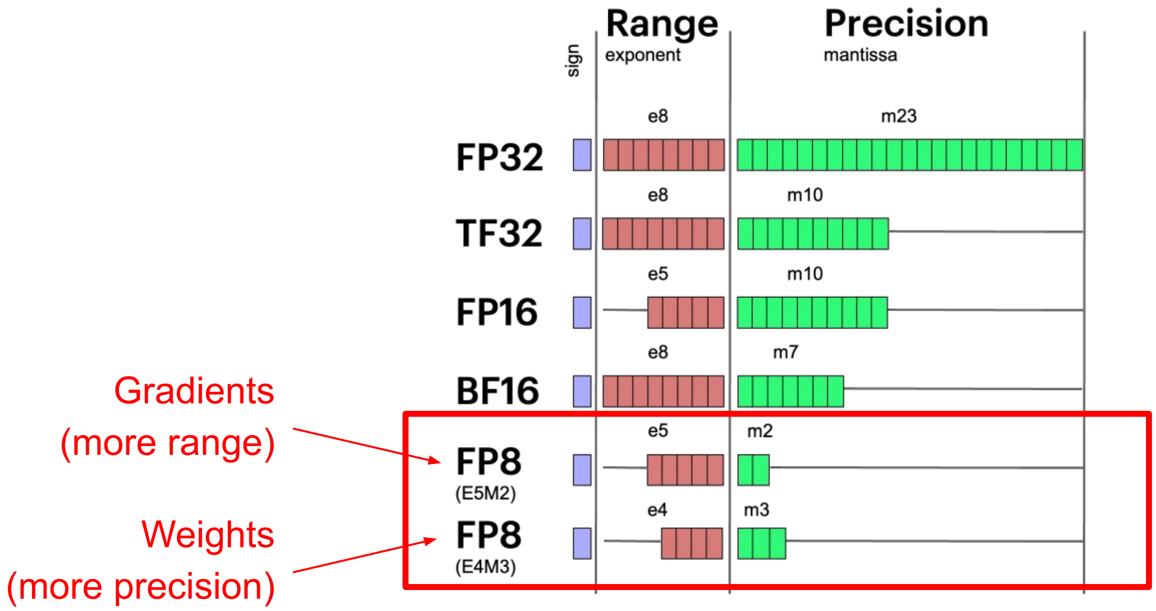
SUPERnormals

Range Extension with Supernormals

for Mixed-Precision 8-bit DNN Training

Shing Wai Pun, *UBC*	Bozhang Bao, *UBC*	Silviu-Ioan Filip, *INRIA*	Guy Lemieux, *UBC*
John V. Kim, *UofT*	Nazar Misyats, *ENS Rennes*	Nirvik Pande, *Austin*	Victor Ravain, *Centrale Nantes*
Robert Sherrick, *TTU*			

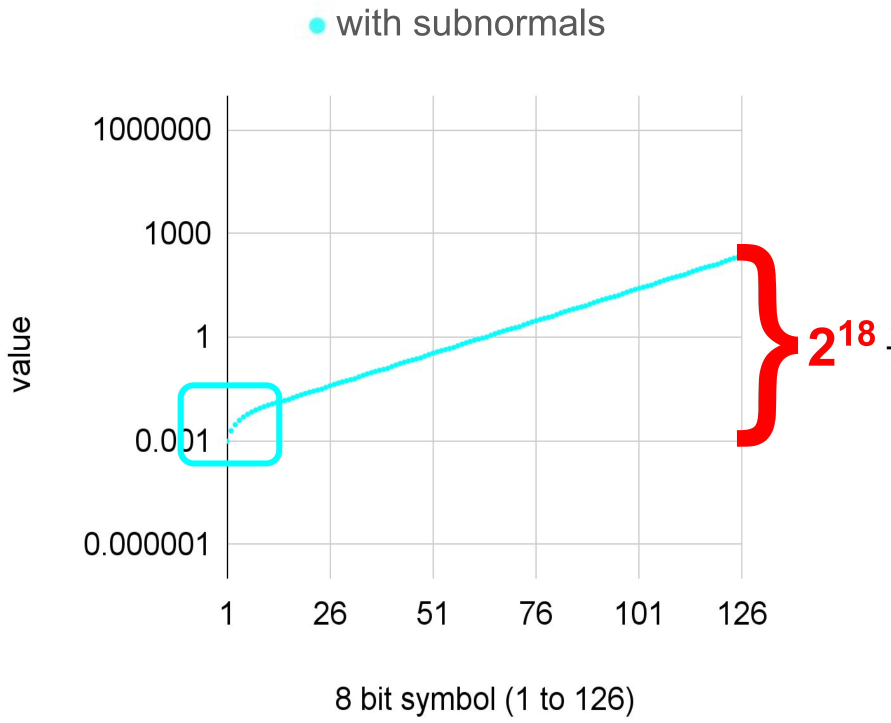
Floating-point Formats & Encodings



8-bit Formats

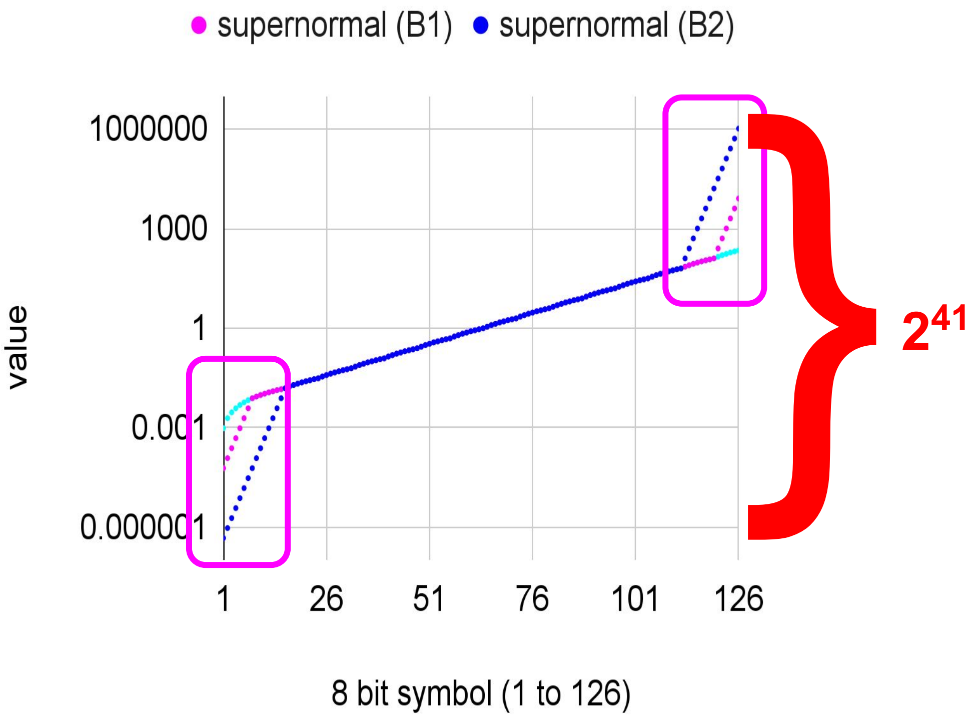
- Limited Range
- Limited Precision
- Challenging for Training
- Better if we need only one (E5M2 or E4M3, not both)

Regular Floats



Range Extension with Supernormals

(always normalized!)



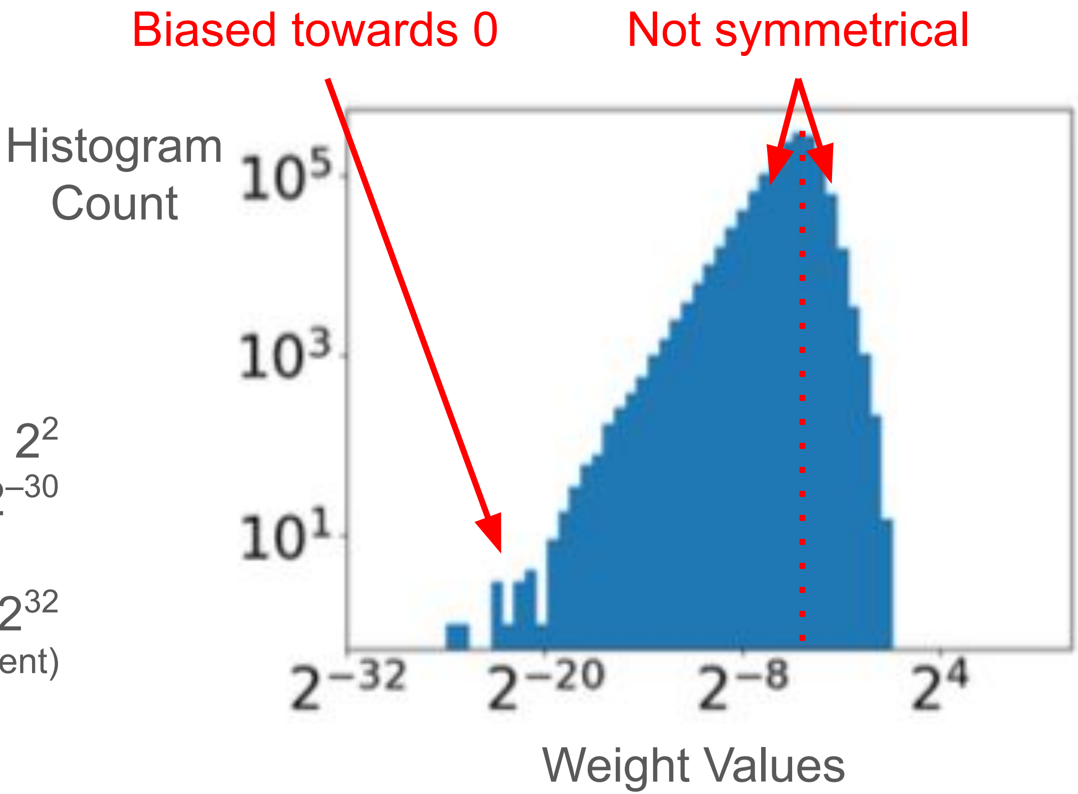
Motivation: Why Range Extension with Supernormals?

- In 8-bit DNN training… dynamic range is important
  - Tiny values
    - Backpropagation of loss/error (differences)
  - Large values
    - Strong (meaningful) results

- Very tiny values close to 0
  - Subnormals (not normalized) provides limited range improvement
  - Subnormals require special case logic (hardware area)

- Very large values close to ∞
  - Immediate overflow to ∞ (which is sticky) ⇒ no gradual overflow

Dynamic Range in DNNs (FP32 training, ResNet-20 CNN)



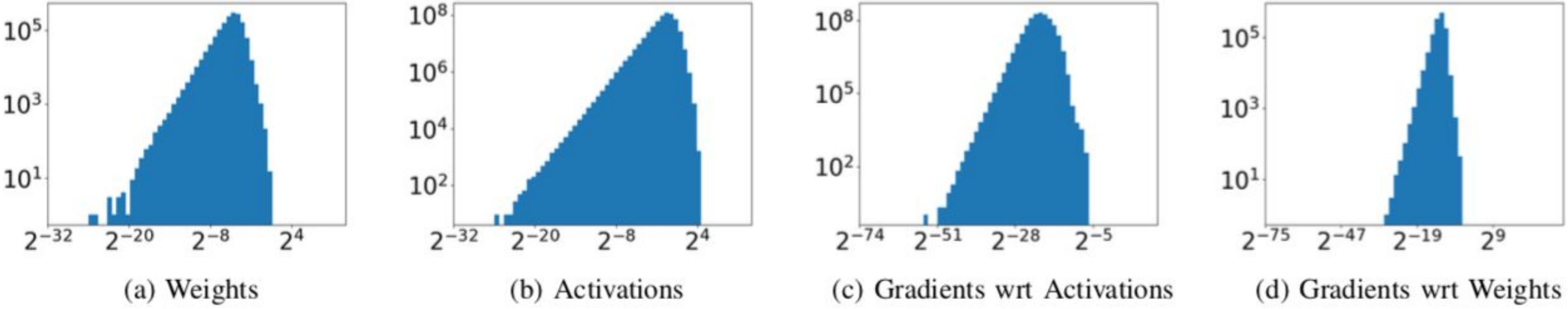
Largest values: 2^2
Smallest values: 2^{-30}

Dynamic range: 2^{32}
(matches 5-bit exponent)

Dynamic Range in DNNs (FP32 training, ResNet-20 CNN)

Still biased towards 0

Still not symmetrical



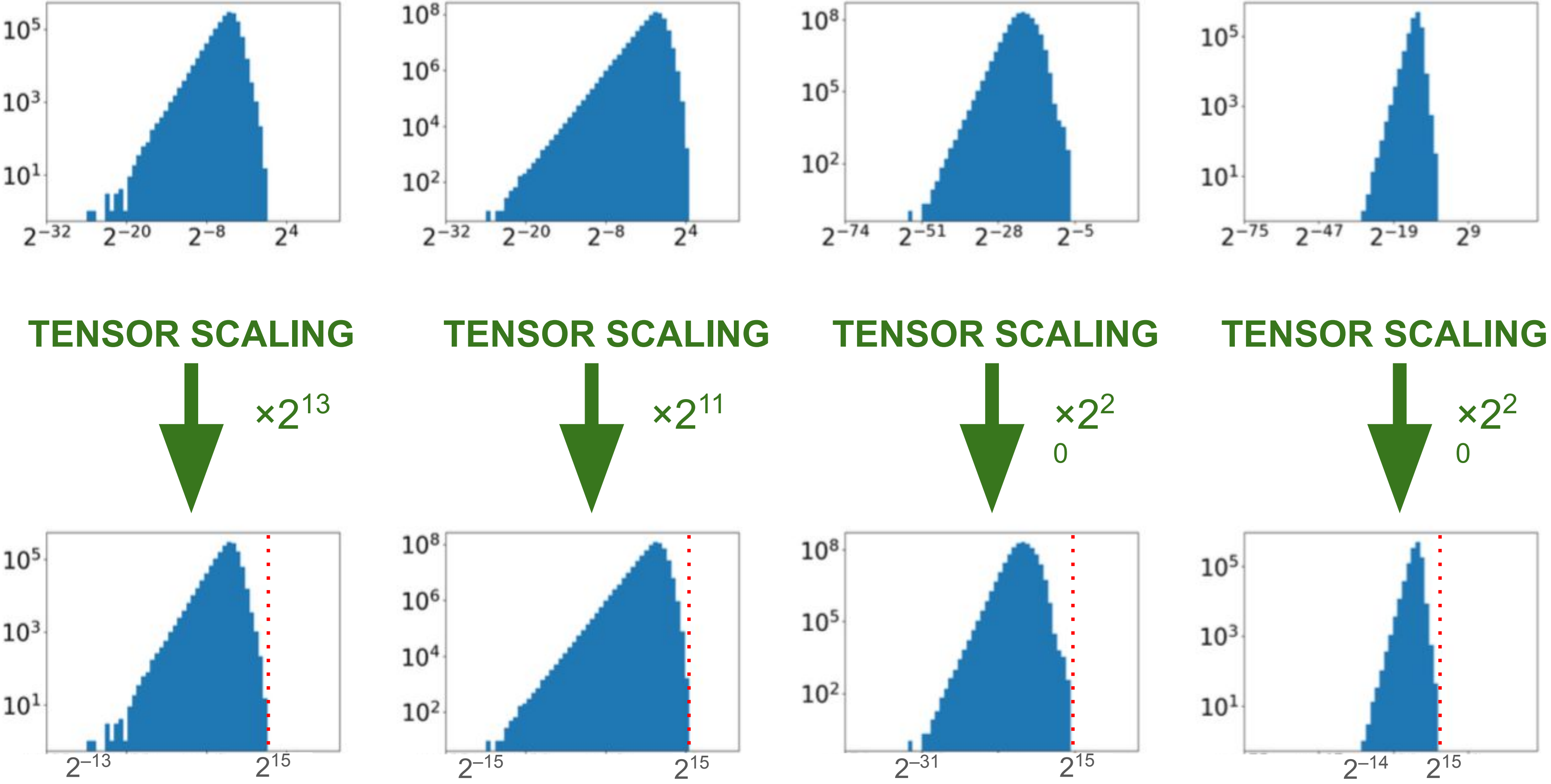
(from previous slide)

Overall range ~ 2^{64} (2^{-54} to 2^{10})

Training (sometimes) applies a (tensor) scaling factor

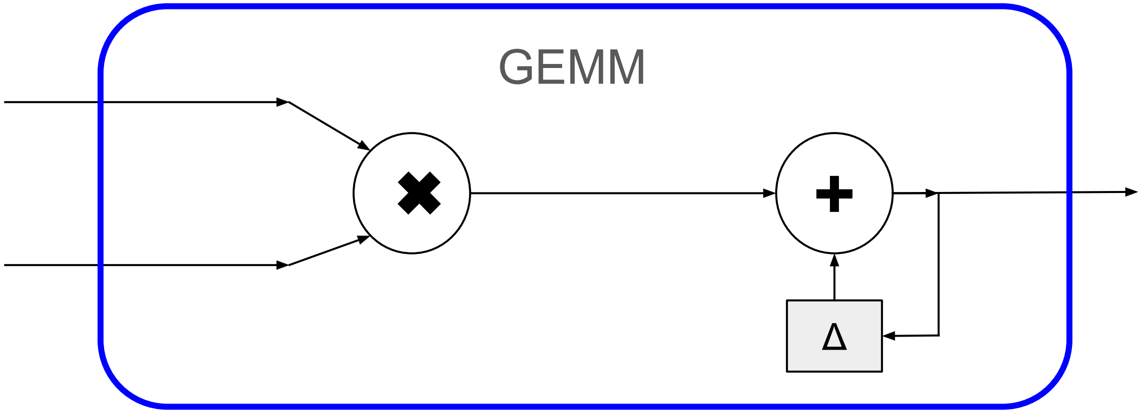
- Scale max value (on right) ⇒ MAX_FLOAT

After scaling, 2^{32} range is usually sufficient


TENSOR SCALING
×2^13
TENSOR SCALING
×2^11
TENSOR SCALING
×2^2
0
TENSOR SCALING
×2^2
0
2^−32 2^−20 2^−8 2^4
2^−32 2^−20 2^−8 2^4
2^−74 2^−51 2^−28 2^−5
2^−75 2^−47 2^−19 2^9
10^1 10^3 10^5
10^2 10^4 10^6 10^8
10^2 10^5 10^8
2^−13 2^15
2^−15 2^15
2^−31 2^15
2^−14 2^15

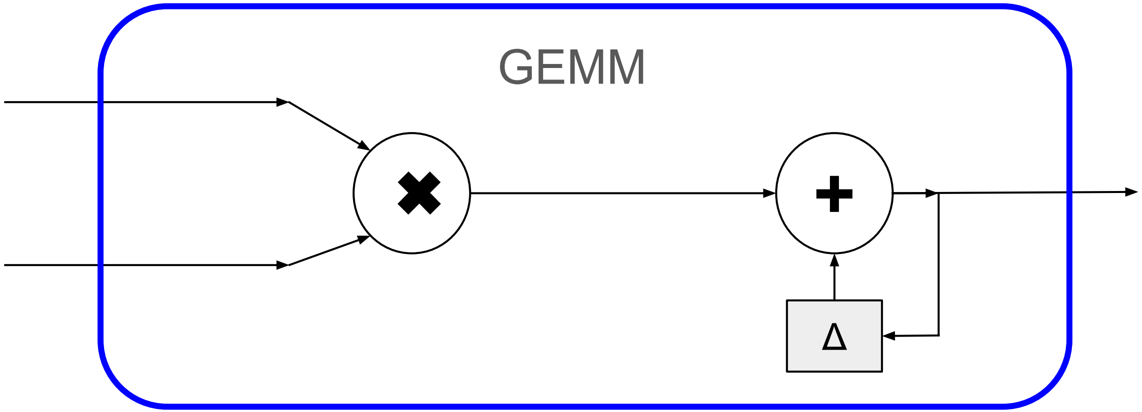
Mixed-Precision Arithmetic in DNNs

- Matrix multiply (GEMM) is the most compute-intensive kernel



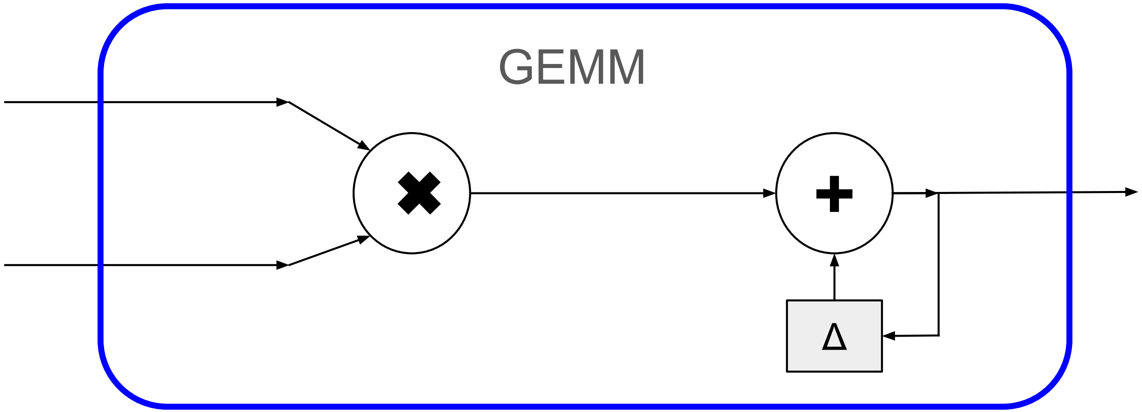
Mixed-Precision Arithmetic in DNNs

- Matrix multiply (GEMM) is the most compute-intensive kernel
  - Inference and Training uses GEMM in forward pass



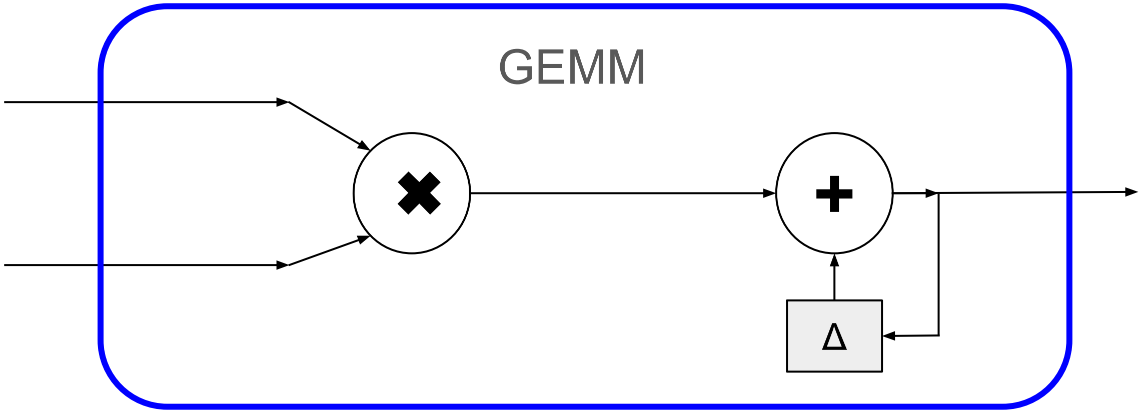
Mixed-Precision Arithmetic in DNNs

- Matrix multiply (GEMM) is the most compute-intensive kernel
  - Inference and Training uses GEMM in forward pass
  - Training uses 2 x GEMM in backwards pass



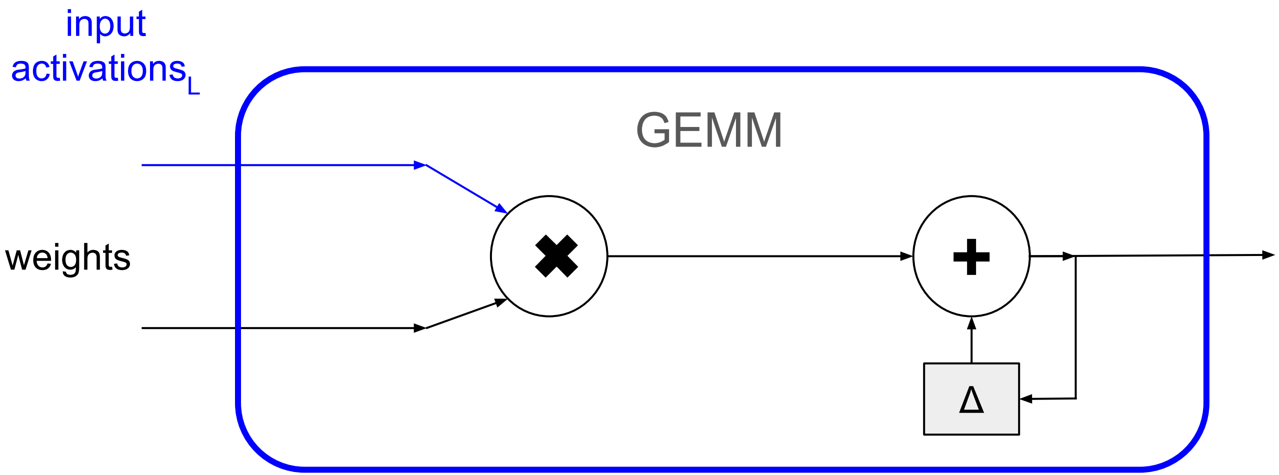
Mixed-Precision Arithmetic in DNNs

- Matrix multiply (GEMM) is the most compute-intensive kernel
  - Inference and Training uses GEMM in forward pass
  - Training uses 2 x GEMM in backwards pass
- GEMM uses mixed/low-precision (M/LP) MAC, remainder is FP32



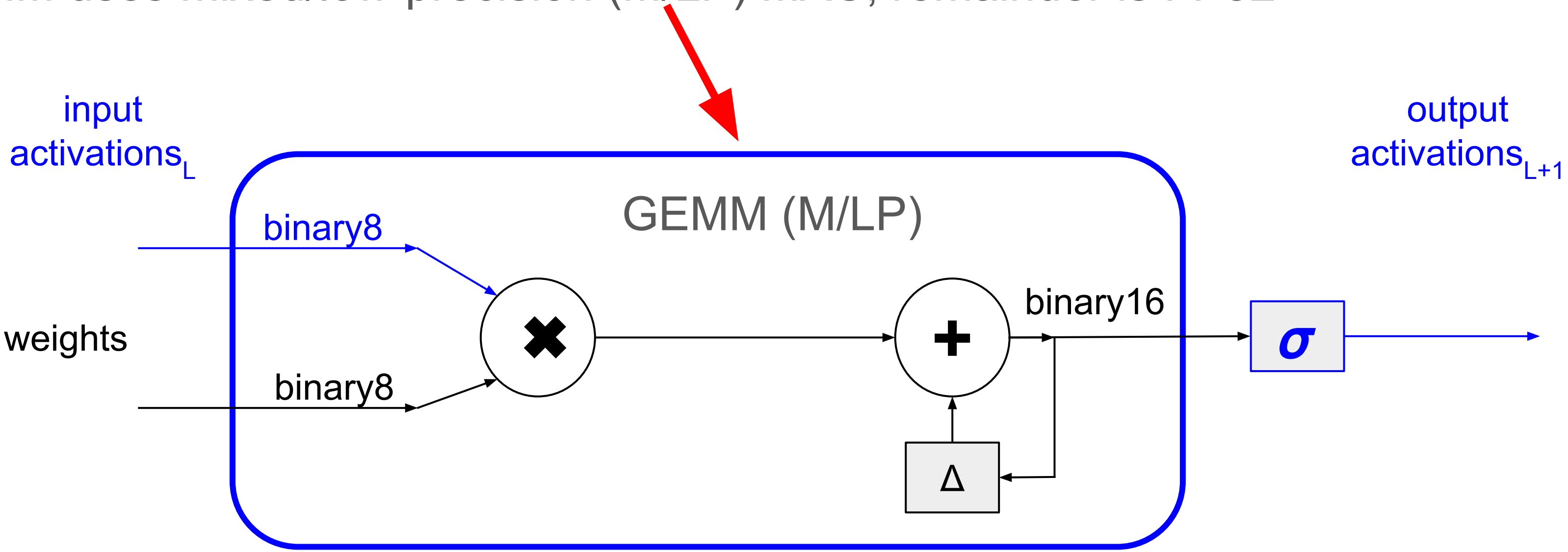
Mixed-Precision Arithmetic in DNNs

- Matrix multiply (GEMM) is the most compute-intensive kernel
  - Inference and Training uses GEMM in forward pass
  - Training uses 2 x GEMM in backwards pass
- GEMM uses mixed/low-precision (M/LP) MAC, remainder is FP32



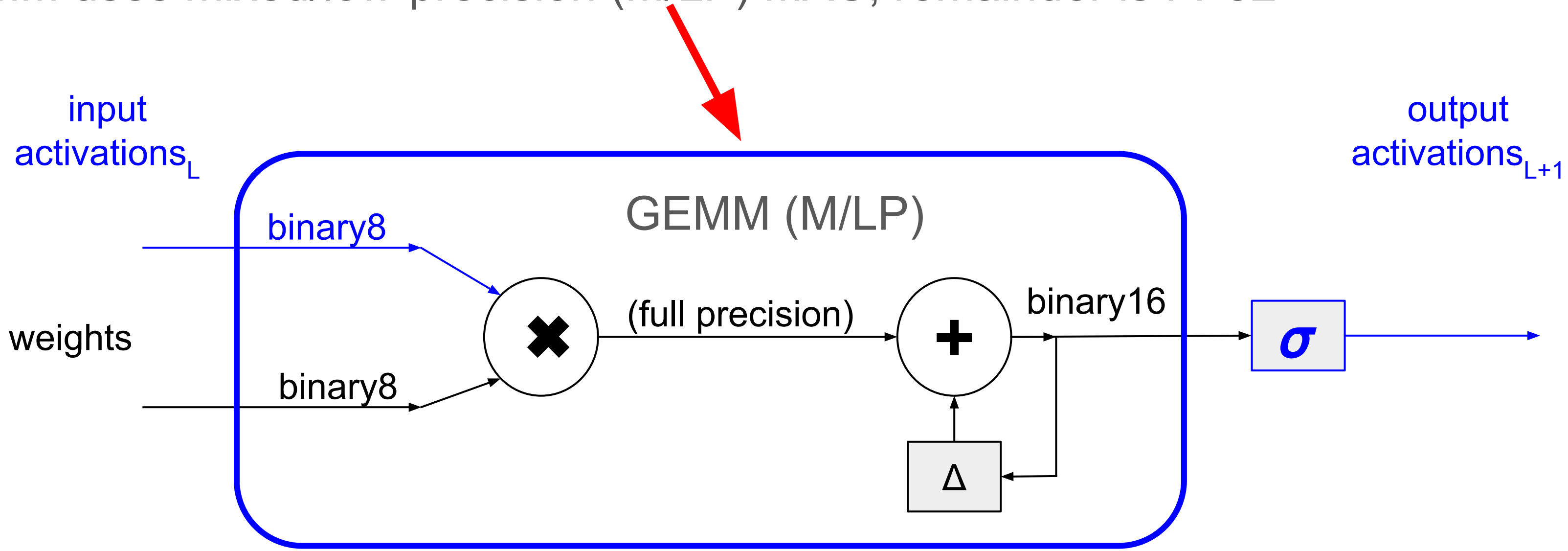
Mixed-Precision Arithmetic in DNNs

- Matrix multiply (GEMM) is the most compute-intensive kernel
  - Inference and Training uses GEMM in forward pass
  - Training uses 2 x GEMM in backwards pass
- GEMM uses mixed/low-precision (M/LP) MAC, remainder is FP32



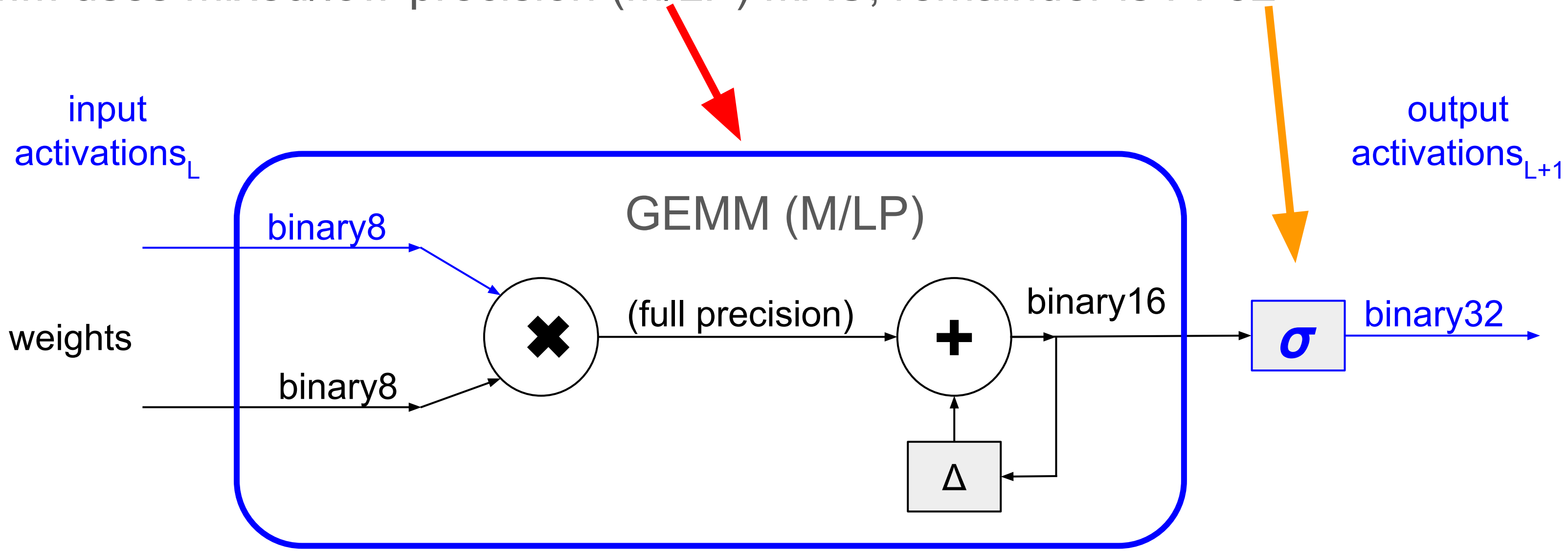
Mixed-Precision Arithmetic in DNNs

- Matrix multiply (GEMM) is the most compute-intensive kernel
  - Inference and Training uses GEMM in forward pass
  - Training uses 2 x GEMM in backwards pass
- GEMM uses mixed/low-precision (M/LP) MAC, remainder is FP32



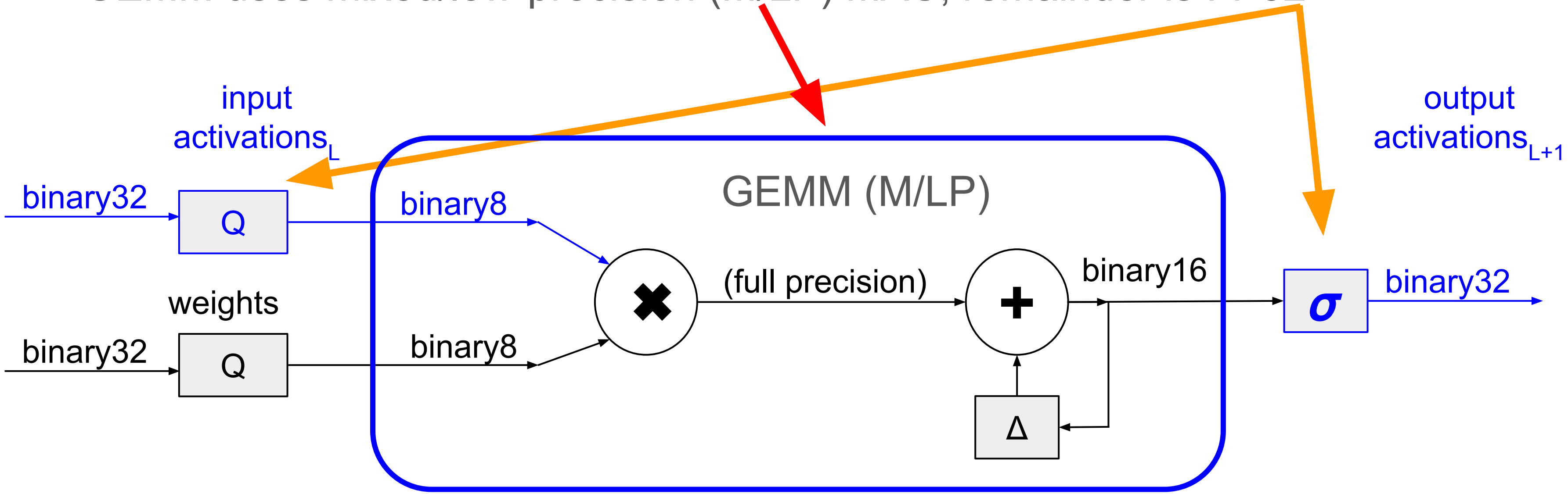
Mixed-Precision Arithmetic in DNNs

- Matrix multiply (GEMM) is the most compute-intensive kernel
  - Inference and Training uses GEMM in forward pass
  - Training uses 2 x GEMM in backwards pass
- GEMM uses mixed/low-precision (M/LP) MAC, remainder is FP32



Mixed-Precision Arithmetic in DNNs

- Matrix multiply (GEMM) is the most compute-intensive kernel
  - Inference and Training uses GEMM in forward pass
  - Training uses 2 x GEMM in backwards pass
- GEMM uses mixed/low-precision (M/LP) MAC, remainder is FP32



Two 8-bit encodings we might consider:

IEEE P3109 Working Group, Arithmetic Formats for Machine Learning
or
Open Compute Project, OCP 8-bit Floating-Point Specification (OFP8)

This paper modifies IEEE P3109 Encodings
for Supernormals

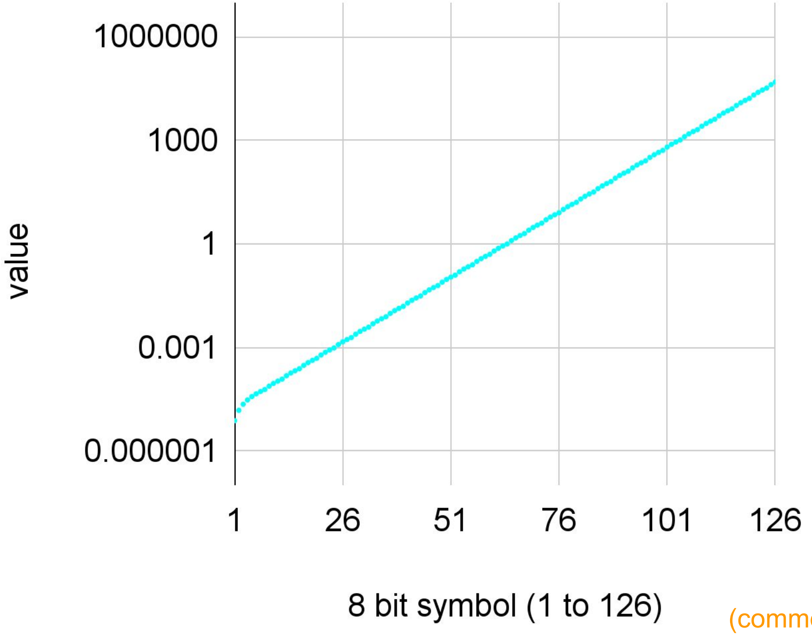
IEEE P3109 Encodings

binary8p3: s_eeeee_mm

normals: $2^{\text{eeeee}-\text{p3bias}} \times 1.\text{mm}$

IEEE P3109 Encodings

binary8p3: s_eeeee_mm
normals: $2^{\text{eeeee}-\text{p3bias}} \times 1.\text{mm}$



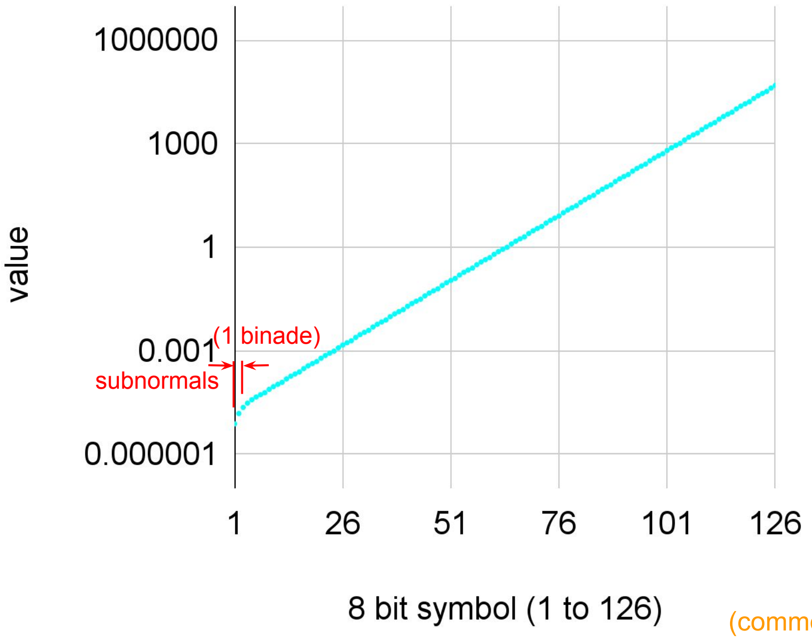
(common symbols 0~0, 127~∞)

IEEE P3109 Encodings

binary8p3: s_eeeee_mm

normals: $2^{eeeee-p3bias} \times 1.mm$

subnormals: $2^{E-p3bias+1} \times 0.mm$

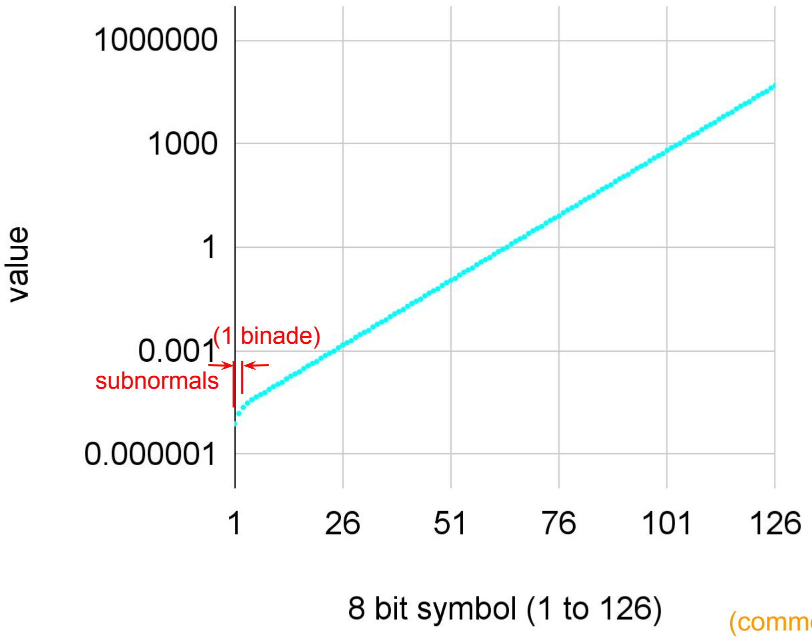


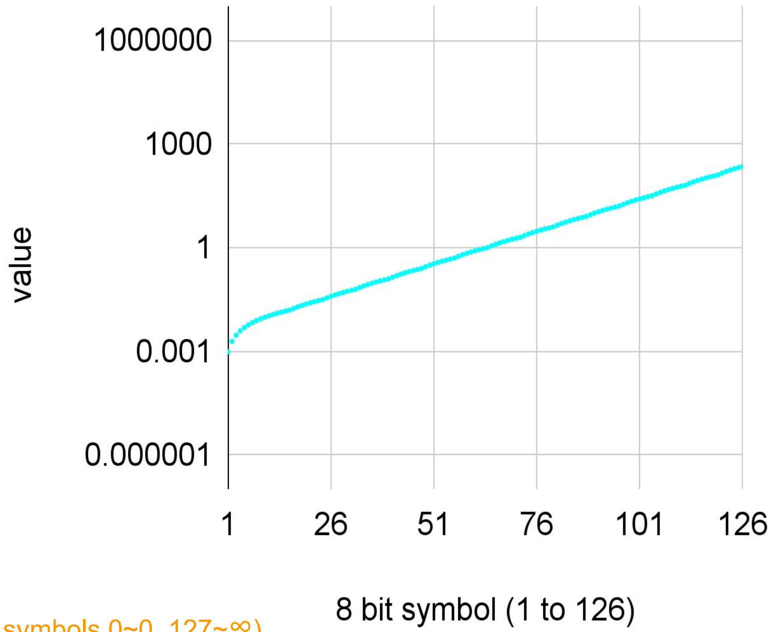
(common symbols 0~0, 127~∞)

IEEE P3109 Encodings

binary8p3: s_eeeee_mm
normals: $2^{eeeee-p3bias} \times 1.mm$
subnormals: $2^{E-p3bias+1} \times 0.mm$

binary8p4: s_eeee_mmm
normals: $2^{eeee-p4bias} \times 1.mmm$



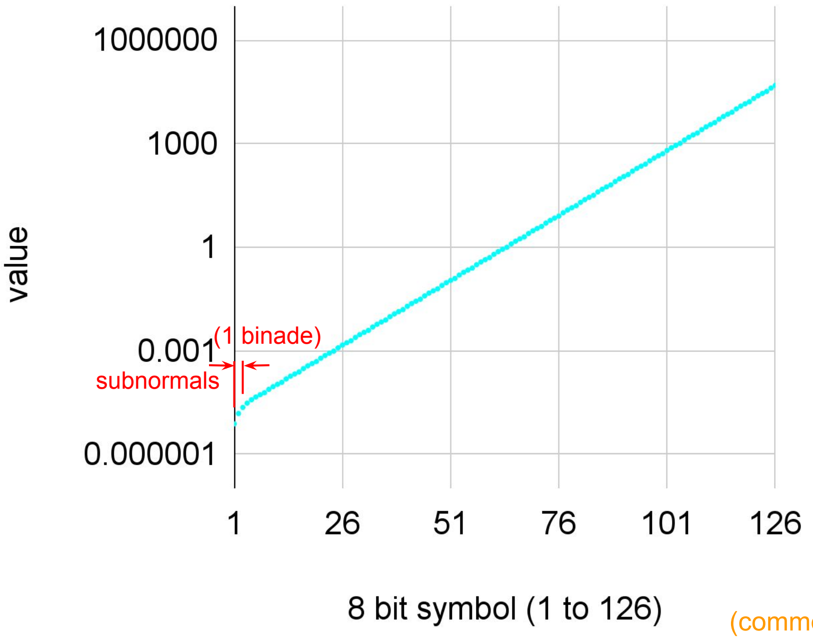


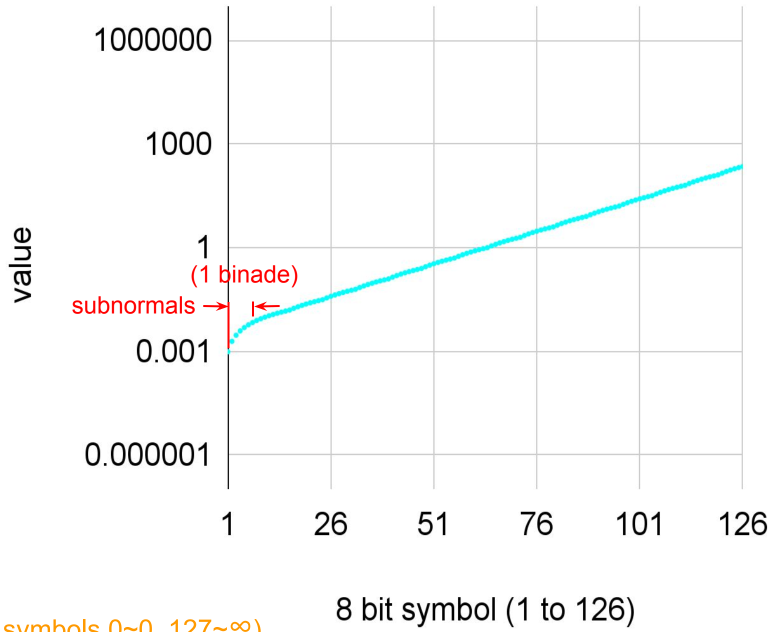
(common symbols 0~0, 127~∞)

IEEE P3109 Encodings

binary8p3: s_eeeee_mm
normals: $2^{eeeee-p3bias} \times 1.mm$
subnormals: $2^{E-p3bias+1} \times 0.mm$

binary8p4: s_eeee_mmm
normals: $2^{eeee-p4bias} \times 1.mmm$
subnormals: $2^{E-p4bias+1} \times 0.mmm$





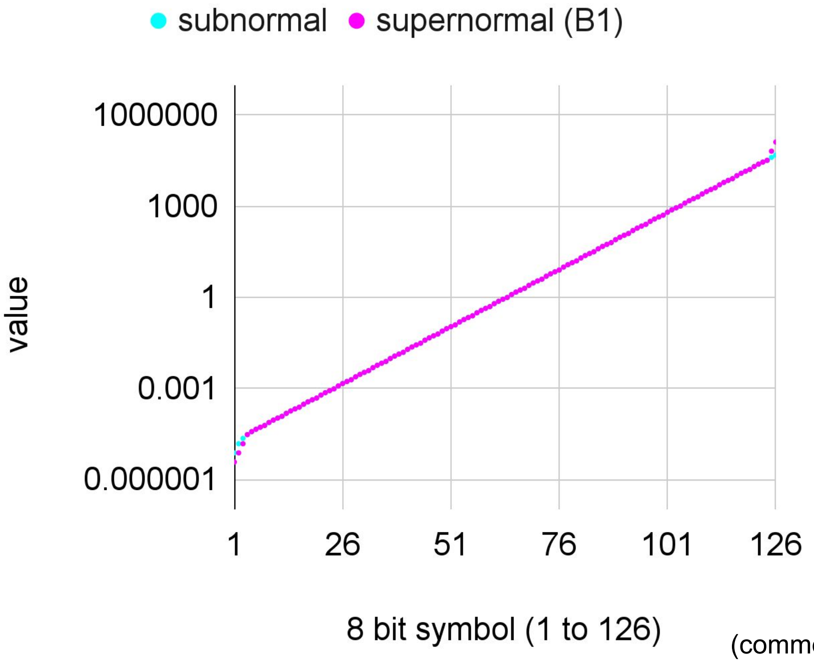
(common symbols 0~0, 127~∞)

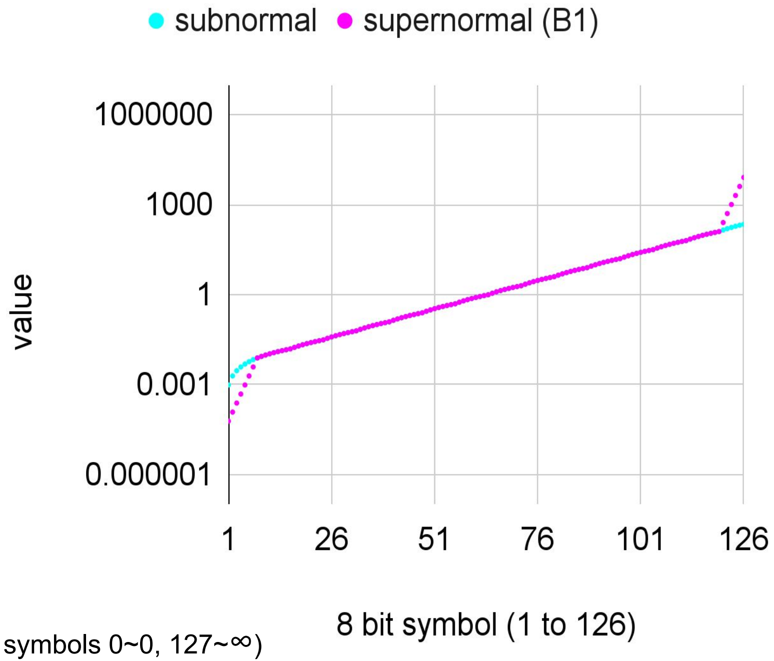
IEEE P3109 Encodings + Supernormals

binary8p3: s_eeeee_mm
normals: $2^{\text{eeeee–p3bias}} \times 1.\text{mm}$

binary8p4: s_eeee_mmm
normals: $2^{\text{eeee–p4bias}} \times 1.\text{mmm}$

Supernormals: $2^{E+M\text{–}bias} \times 1.0$





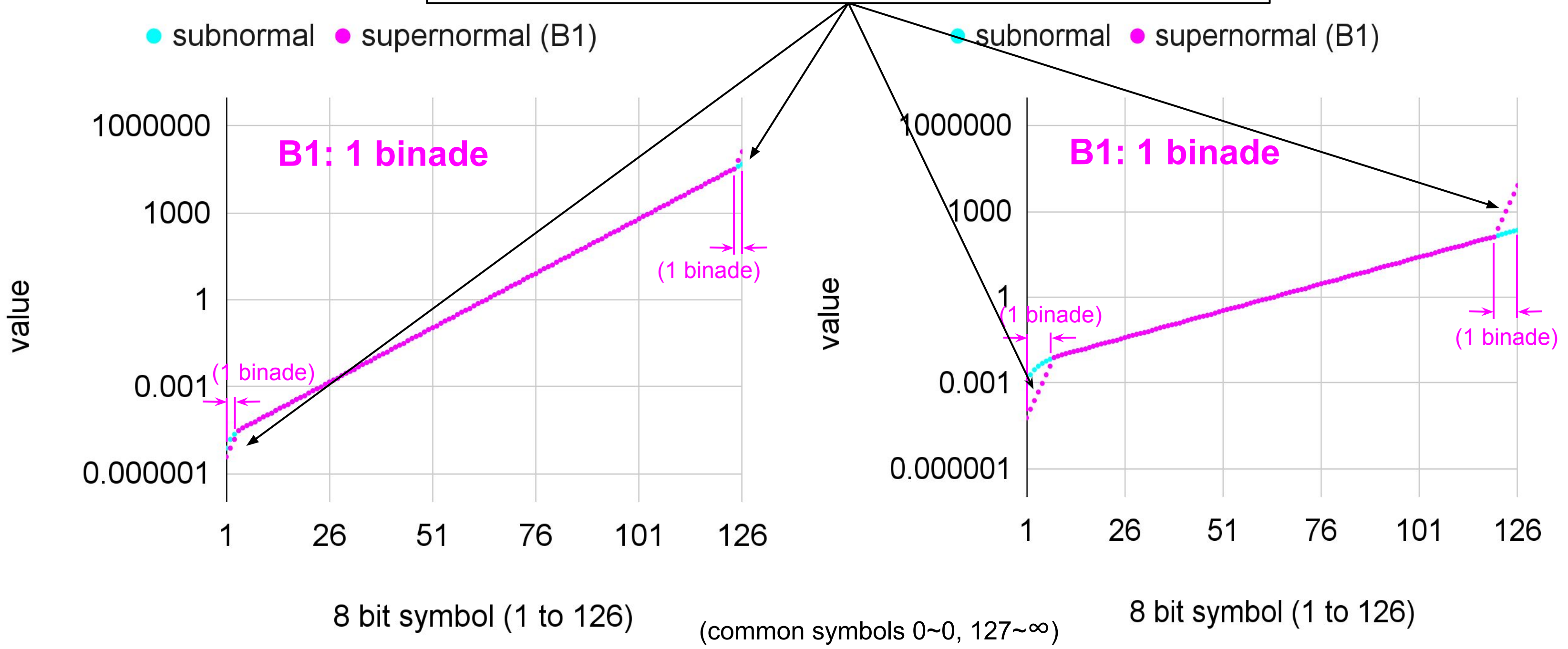
(common symbols 0~0, 127~∞)

IEEE P3109 Encodings + Supernormals

binary8p3: s_eeeee_mm
normals: $2^{eeeee-p3bias} \times 1.mm$

binary8p4: s_eeee_mmm
normals: $2^{eeee-p4bias} \times 1.mmm$

Supernormals: $\mathbf{2^{E+M-bias} \times 1.0}$



IEEE P3109 Encodings + Supernormals

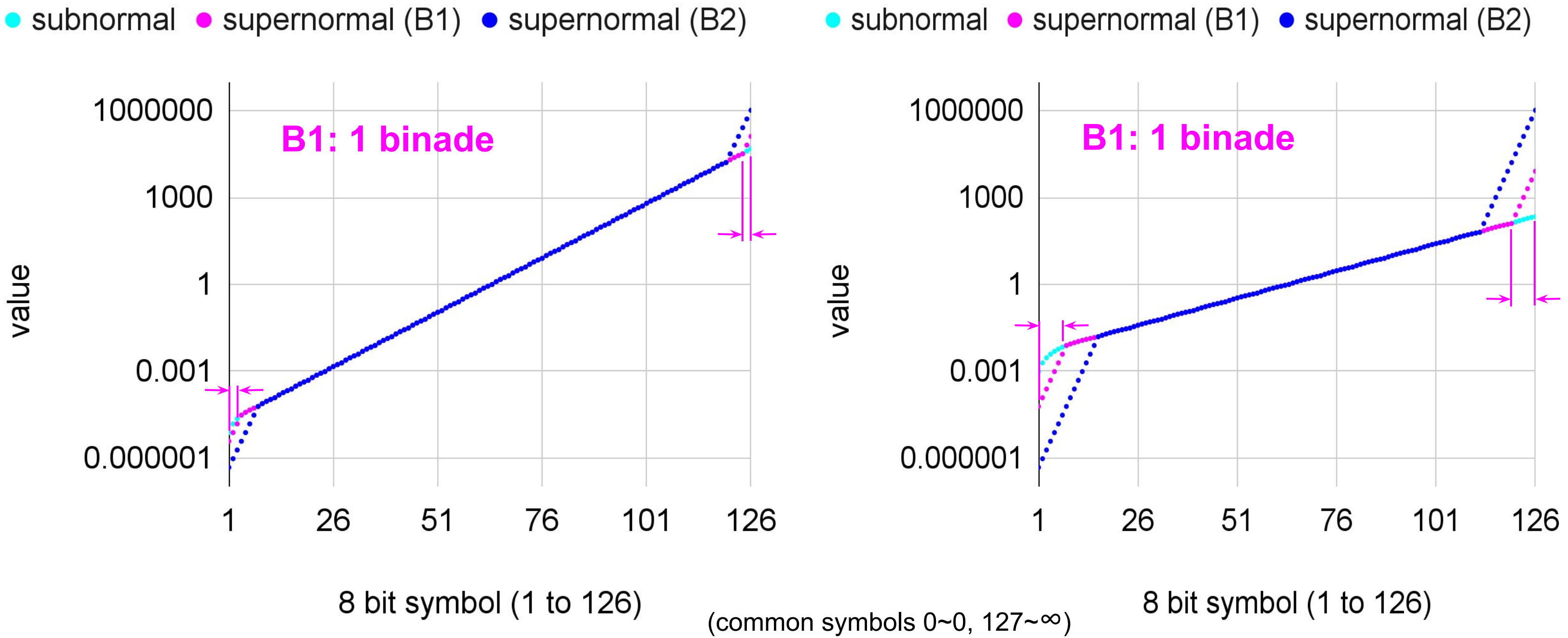
binary8p3: s_eeeee_mm

normals: $2^{eeeee-p3bias} \times 1.mm$

binary8p4: s_eeee_mmm

normals: $2^{eeee-p4bias} \times 1.mmm$

Supernormals: $2^{E+M-bias} \times 1.0$



(common symbols 0~0, 127~∞)

IEEE P3109 Encodings + Supernormals

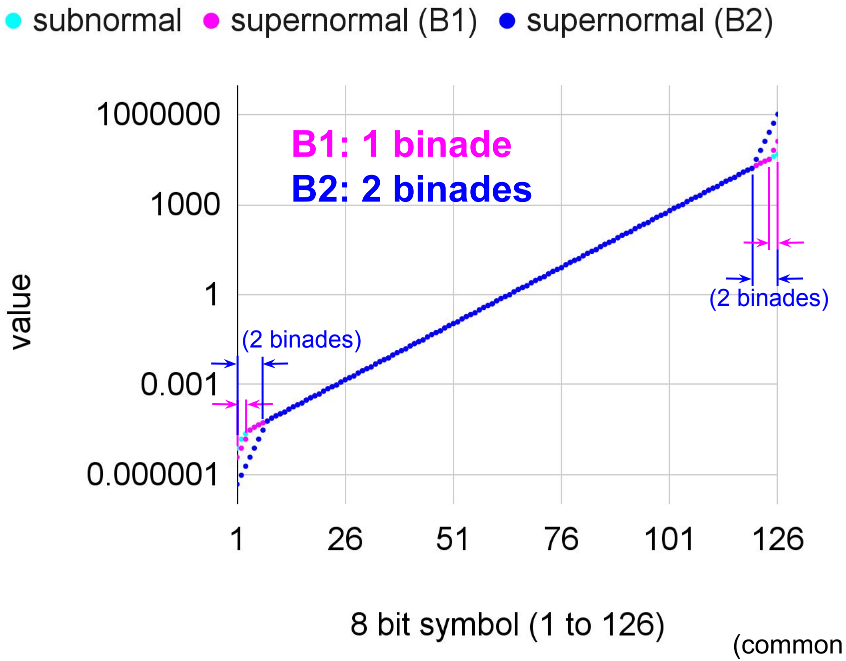
binary8p3: s_eeeee_mm

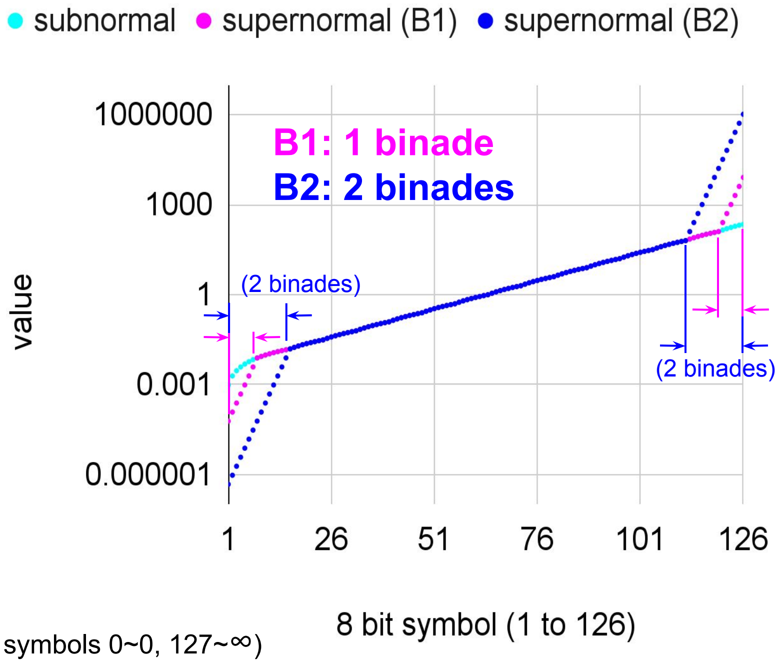
normals: $2^{eeeee-p3bias} \times 1.mm$

binary8p4: s_eeee_mmm

normals: $2^{eeee-p4bias} \times 1.mmm$

Supernormals: $2^{E+M-bias} \times 1.0$





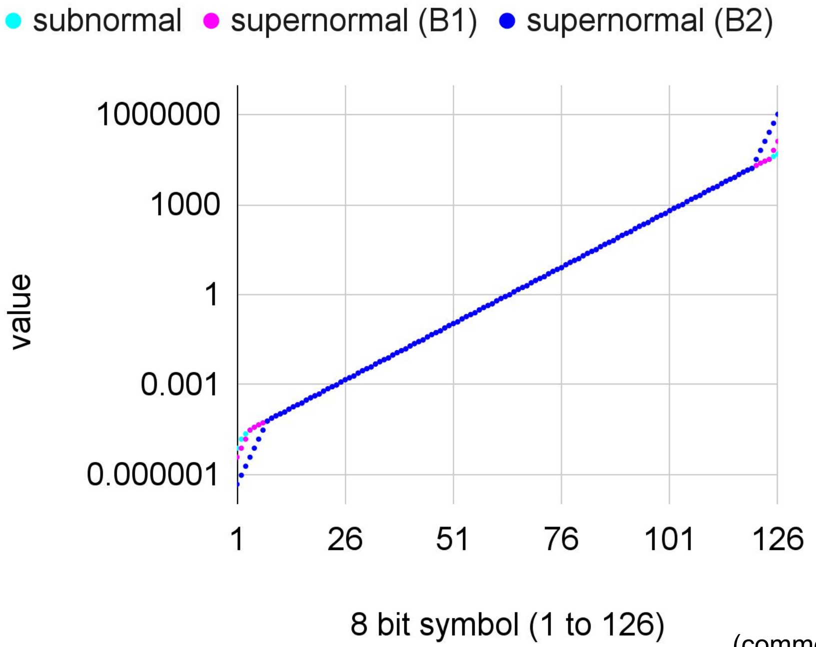
(common symbols 0~0, 127~∞)

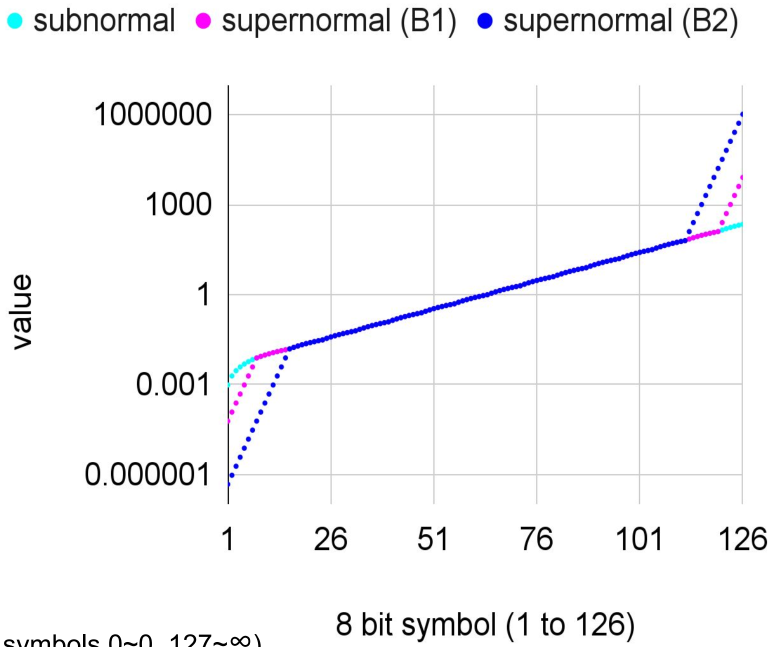
IEEE P3109 Encodings + Supernormals

binary8p3: s_eeeee_mm
normals: $2^{eeeee-p3bias} \times 1.mm$

binary8p4: s_eeee_mmm
normals: $2^{eeee-p4bias} \times 1.mmm$

Supernormals: $2^{E+M-bias} \times 1.0$





(common symbols 0~0, 127~∞)

IEEE P3109 Encodings + Supernormals

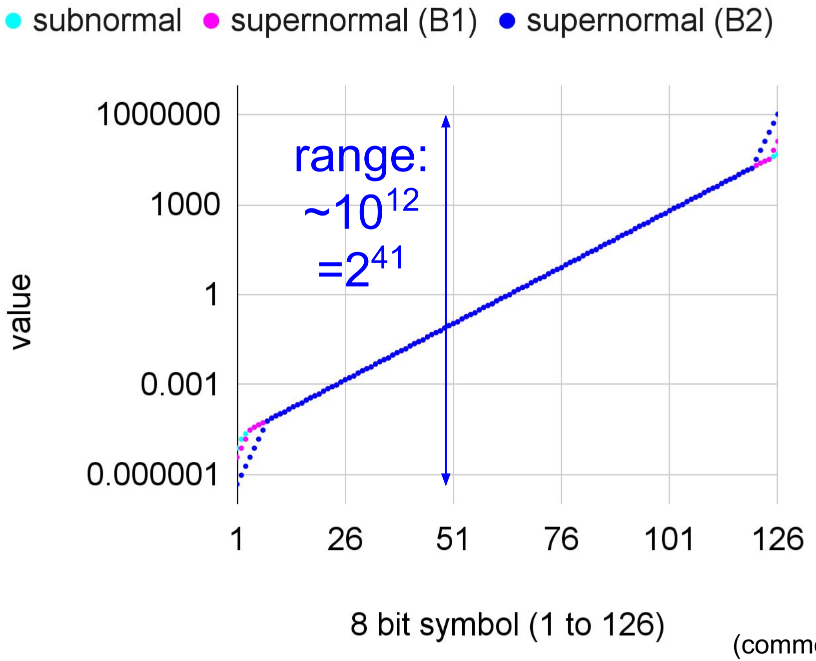
binary8p3: s_eeeee_mm

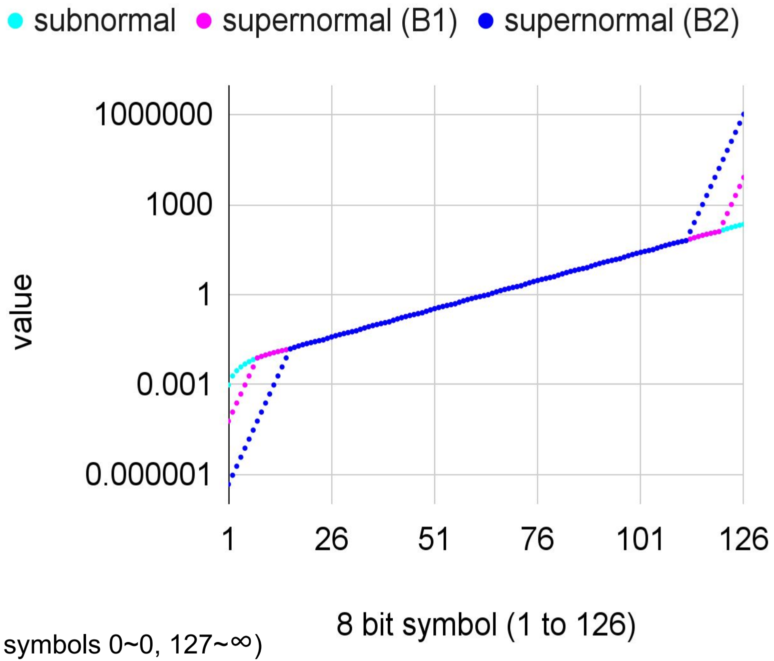
normals: $2^{eeeee-p3bias} \times 1.mm$

binary8p4: s_eeee_mmm

normals: $2^{eeee-p4bias} \times 1.mmm$

Supernormals: $2^{E+M-bias} \times 1.0$





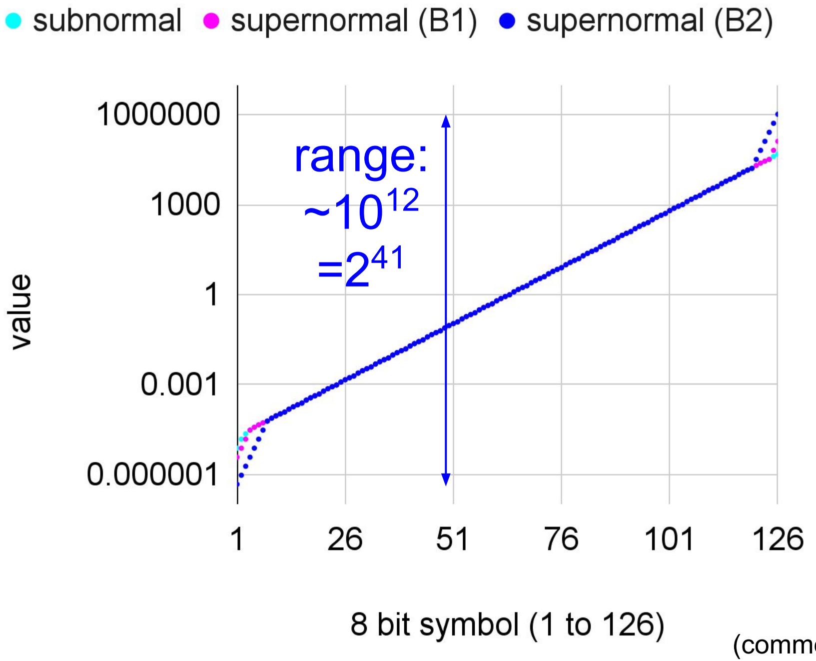
(common symbols 0~0, 127~∞)

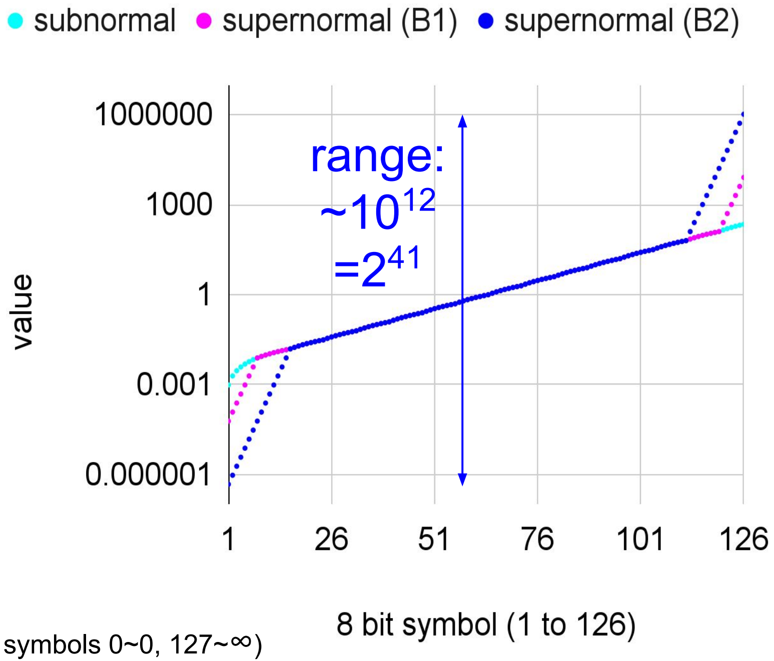
IEEE P3109 Encodings + Supernormals

binary8p3: s_eeeee_mm
normals: $2^{eeeee-p3bias} \times 1.mm$

binary8p4: s_eeee_mmm
normals: $2^{eeee-p4bias} \times 1.mmm$

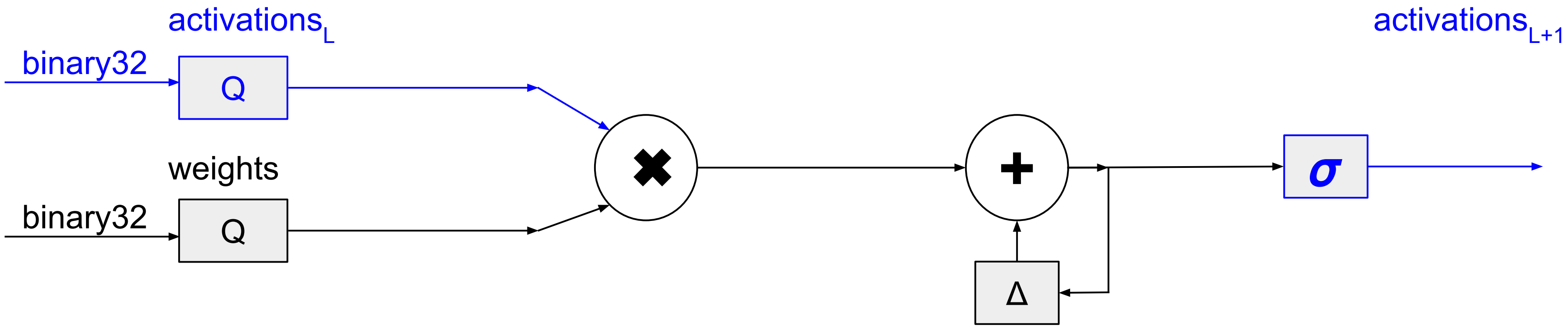
Supernormals: $2^{E+M-bias} \times 1.0$



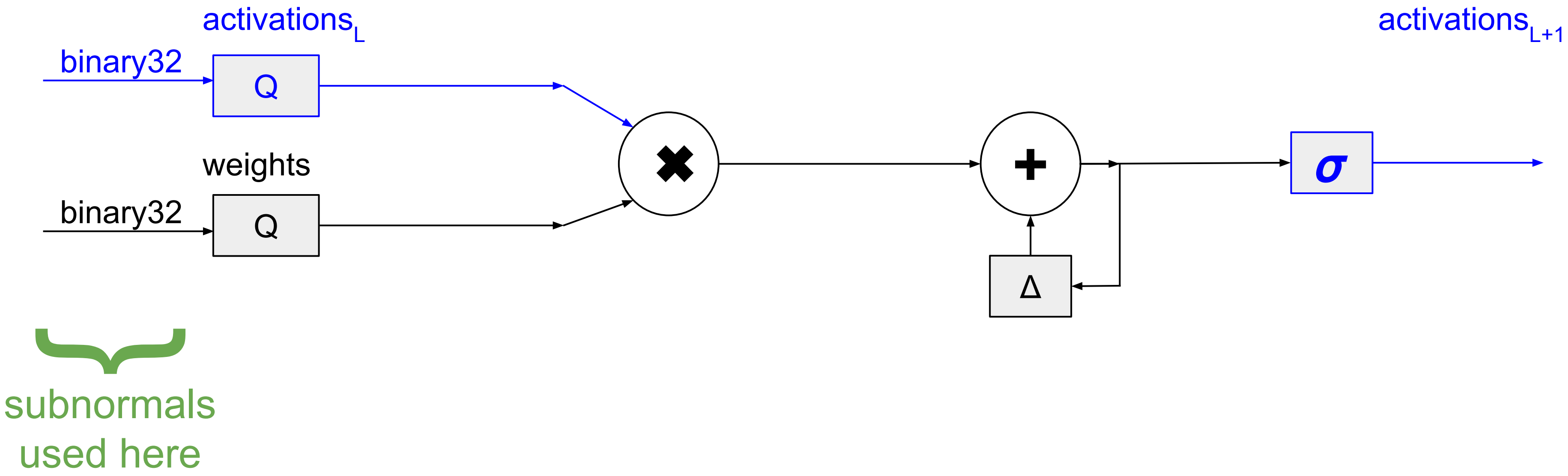


(common symbols 0~0, 127~∞)

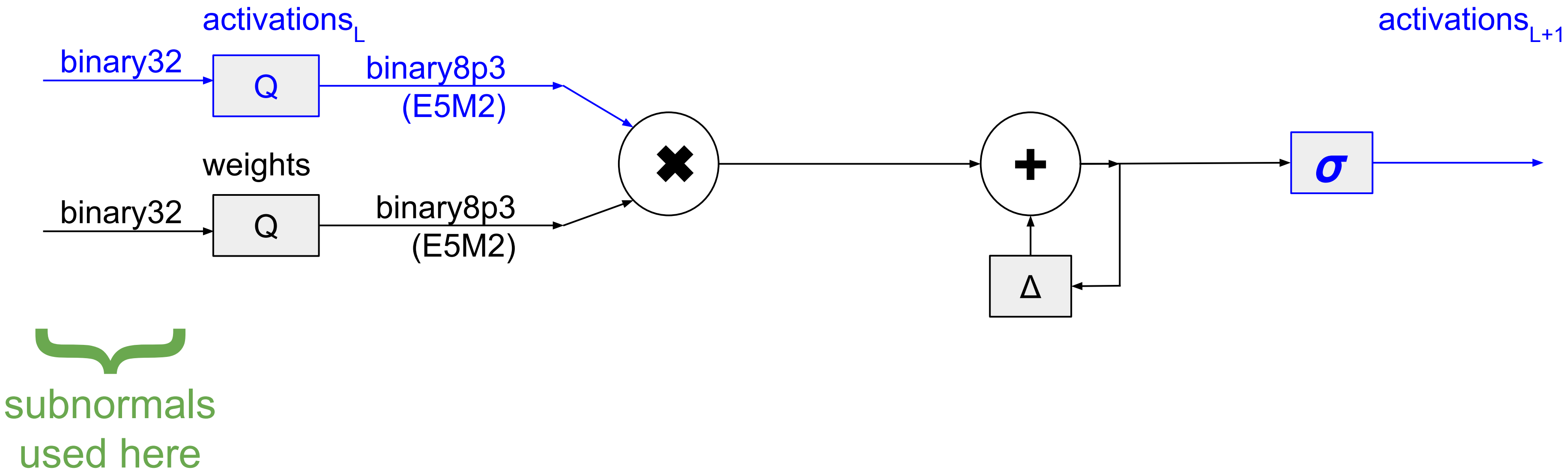
Quantization + Fused MAC + Activation (*σ*)



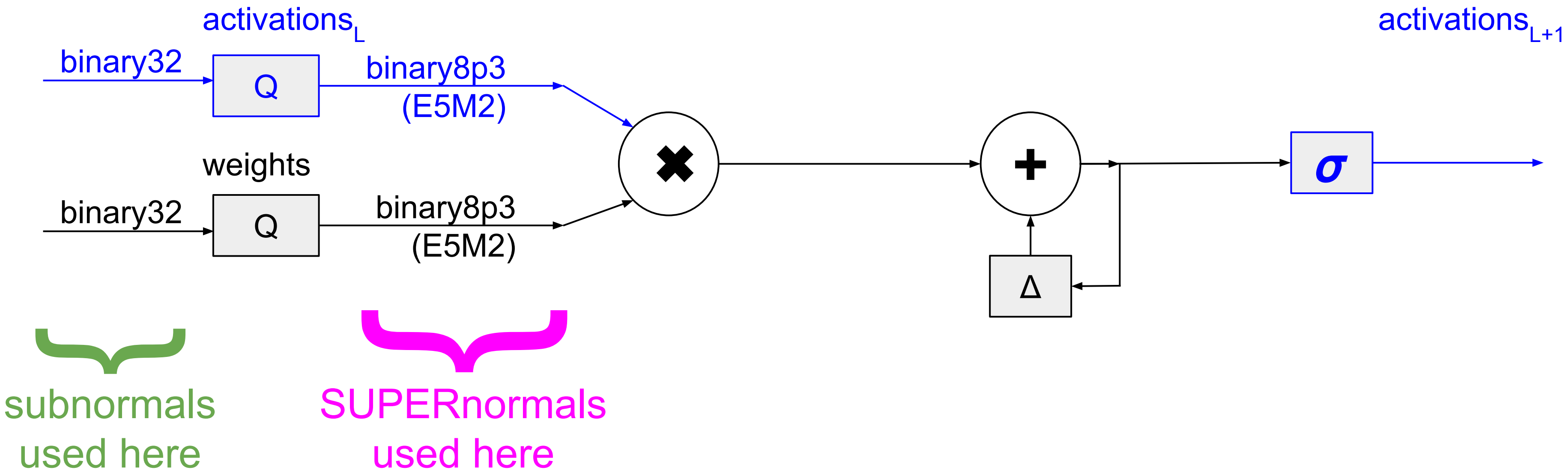
Quantization + Fused MAC + Activation (*σ*)



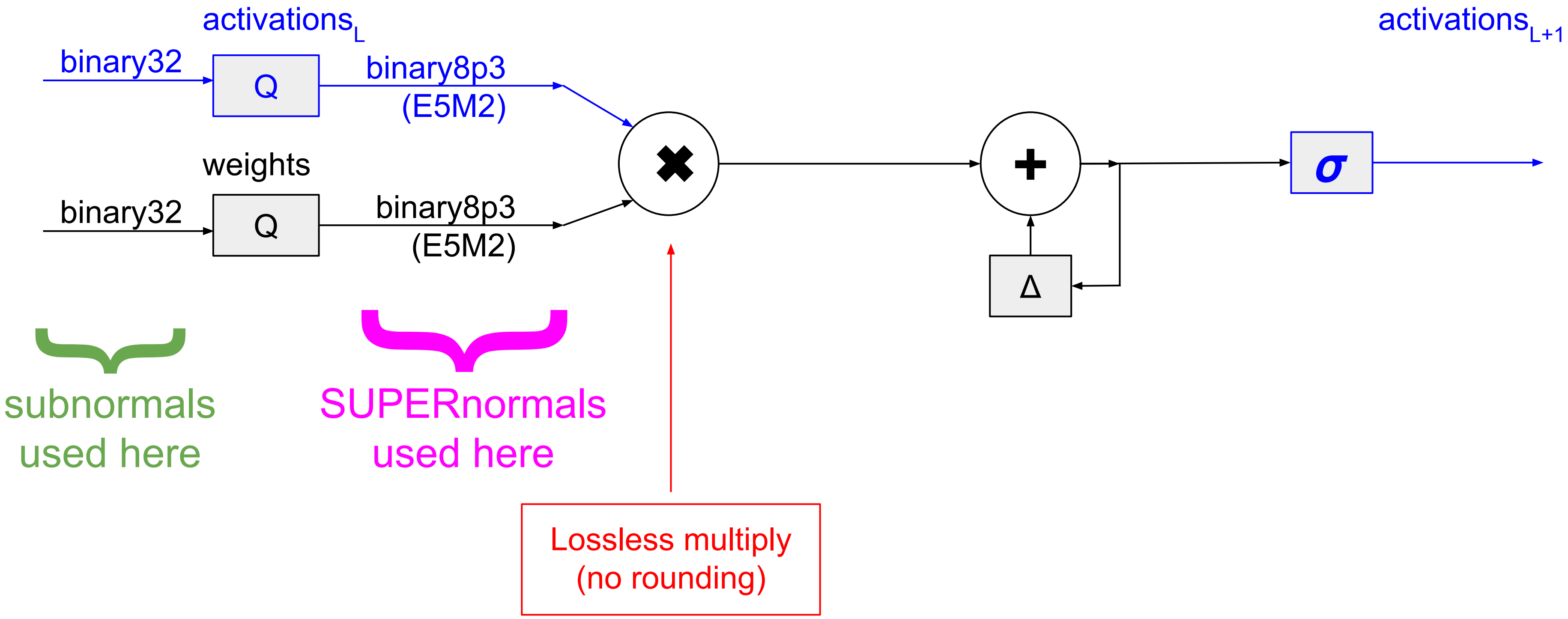
Quantization + Fused MAC + Activation (𝝈)



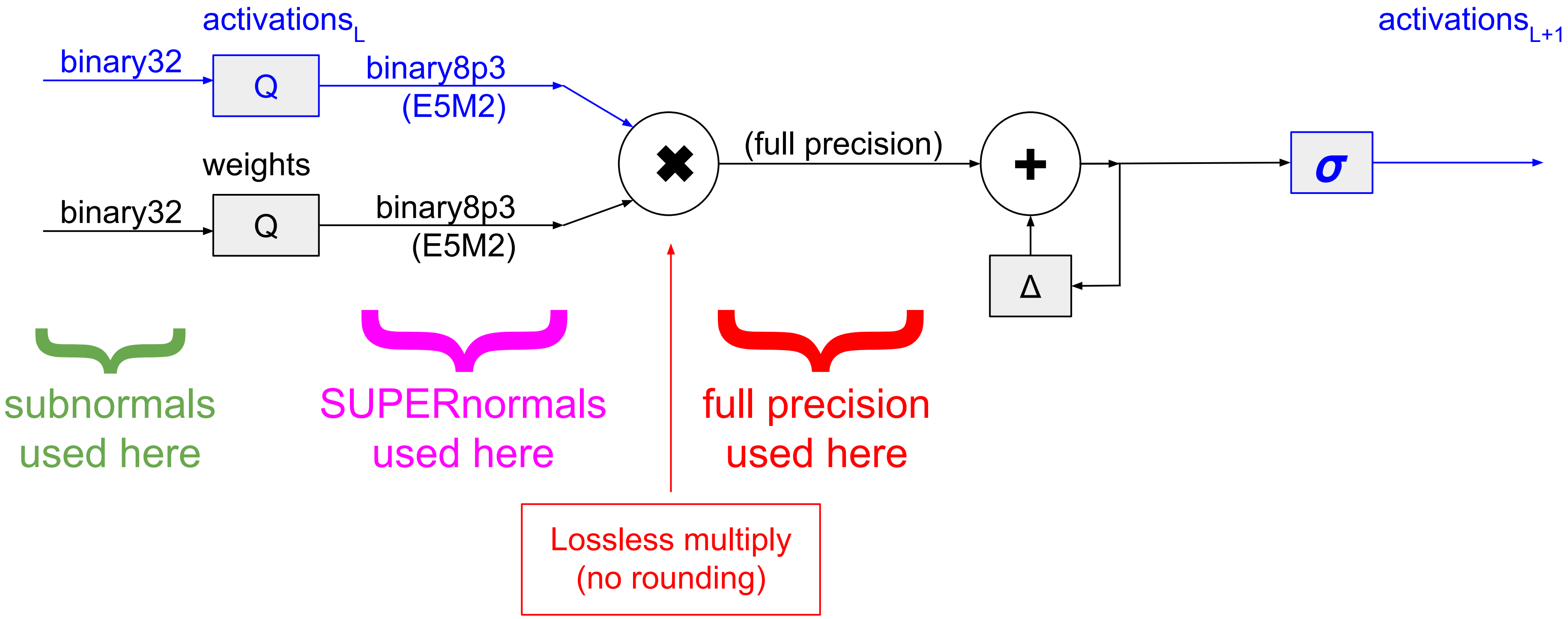
Quantization + Fused MAC + Activation (σ)



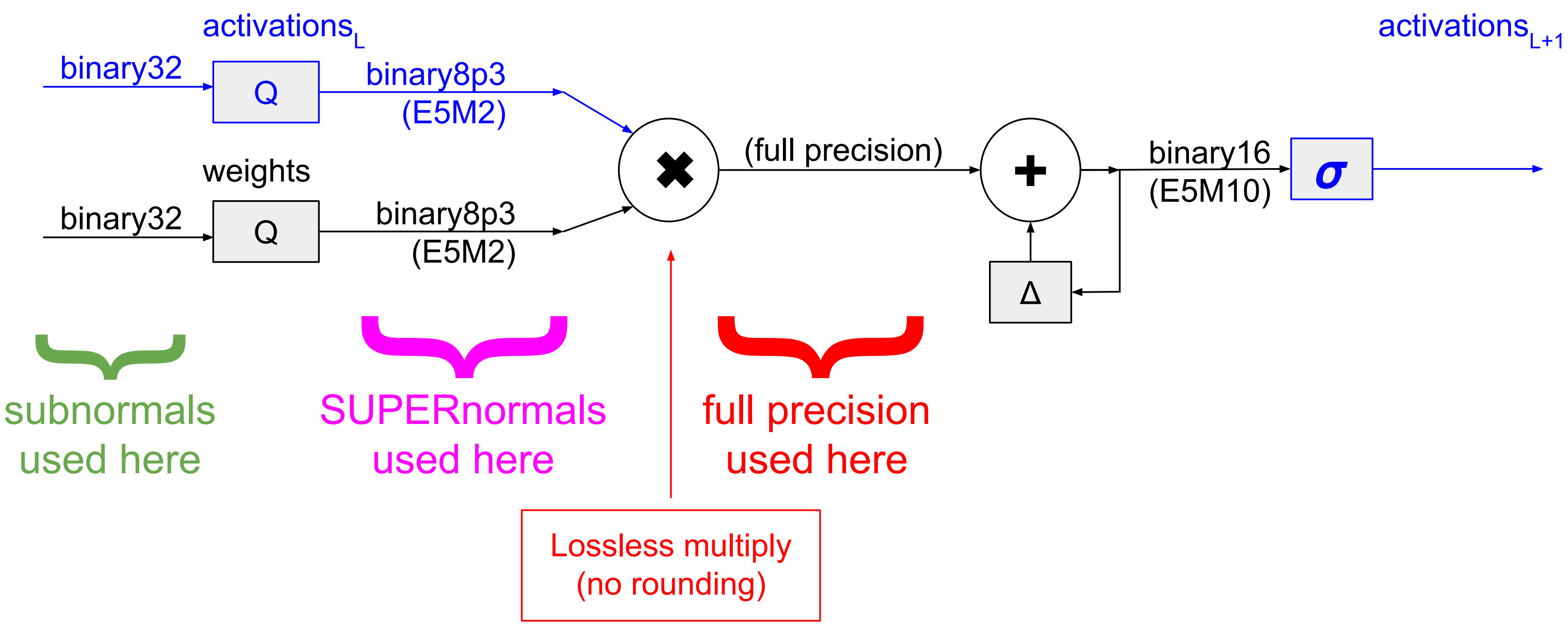
Quantization + Fused MAC + Activation (σ)



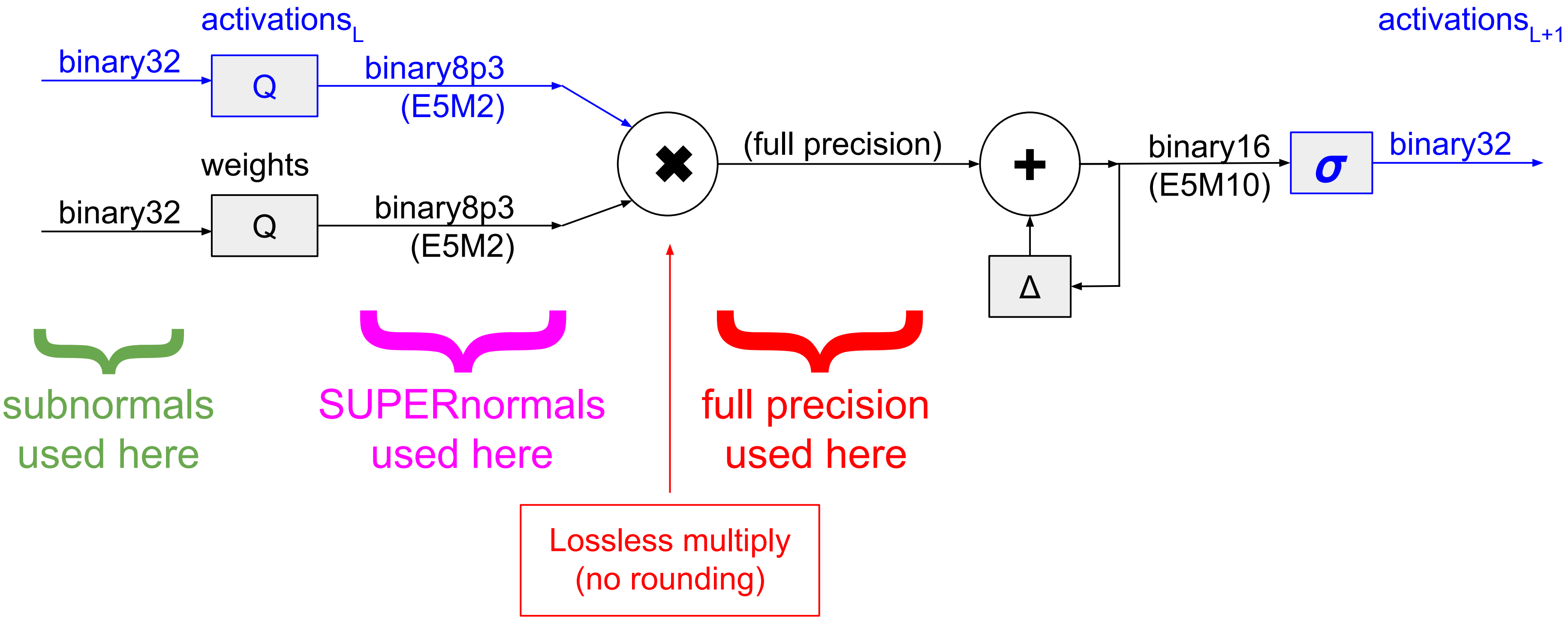
Quantization + Fused MAC + Activation (σ)



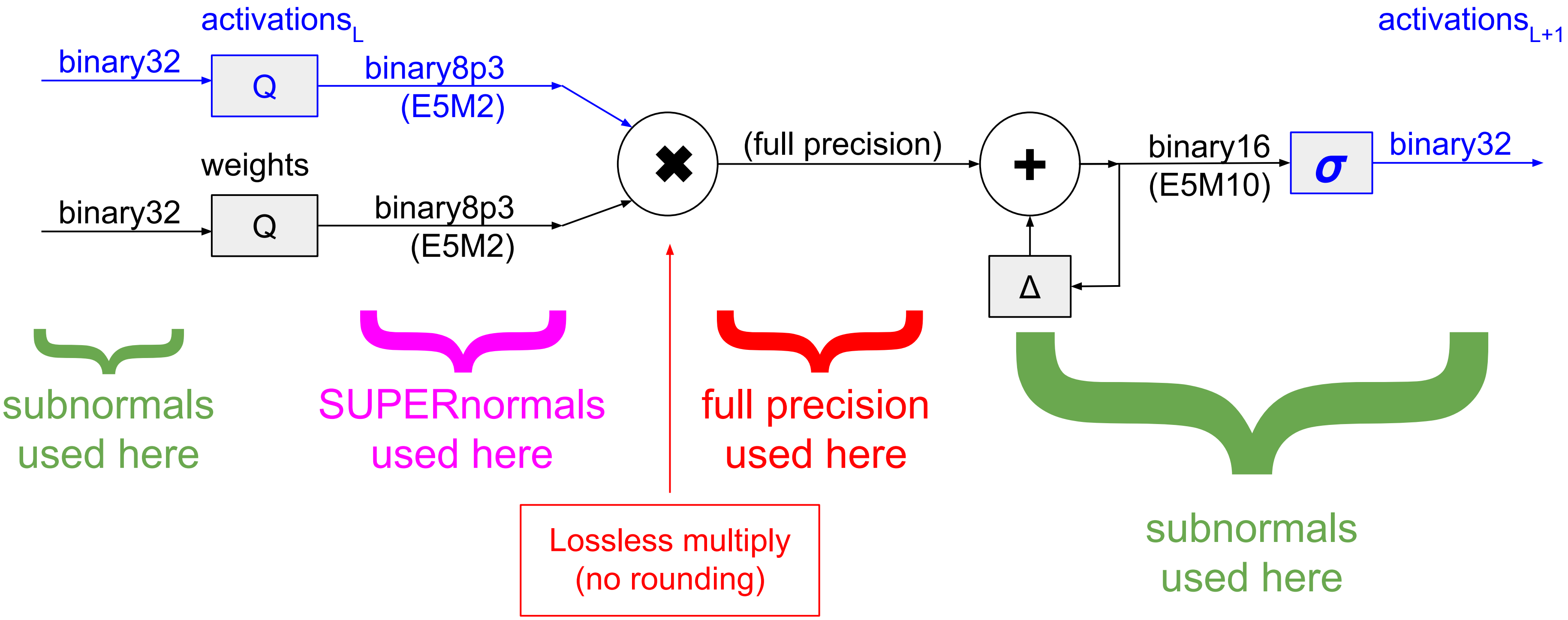
Quantization + Fused MAC + Activation (σ)



Quantization + Fused MAC + Activation (**σ**)

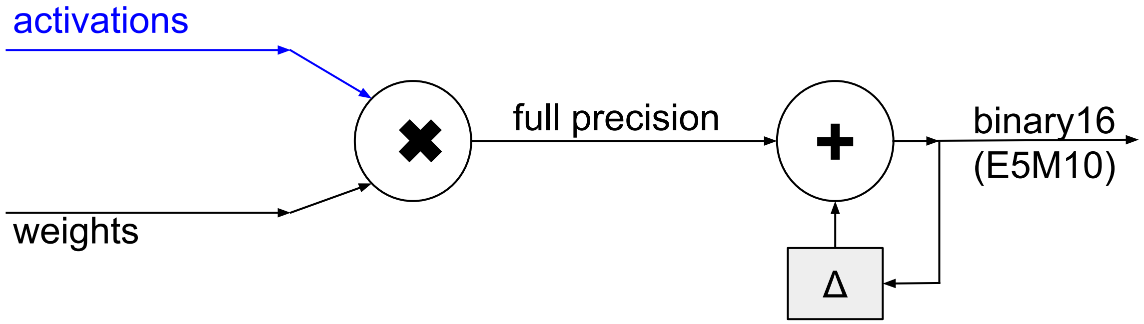


Quantization + Fused MAC + Activation (σ)



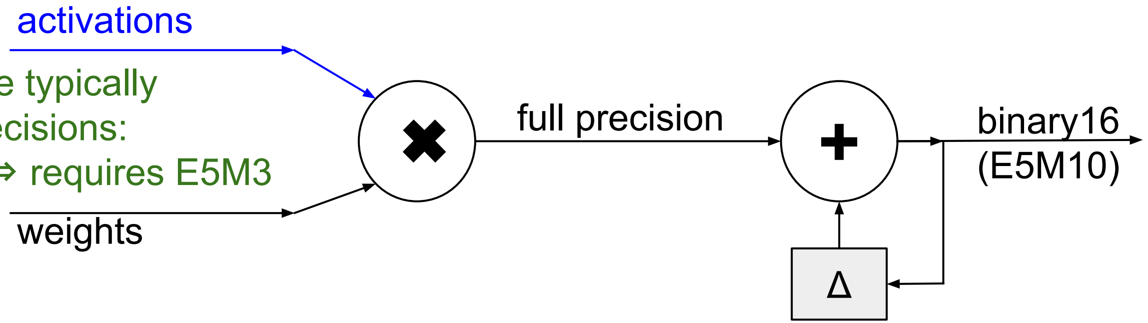
Results

Area Results: Fused MAC (Altera FPGA ALMs)

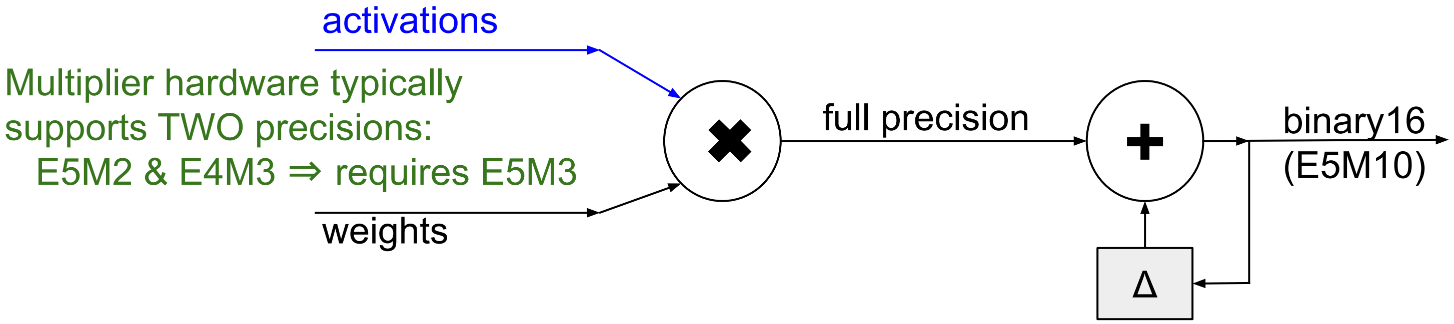


Area Results: Fused MAC (Altera FPGA ALMs)

Multiplier hardware typically supports TWO precisions:
E5M2 & E4M3 ⇒ requires E5M3

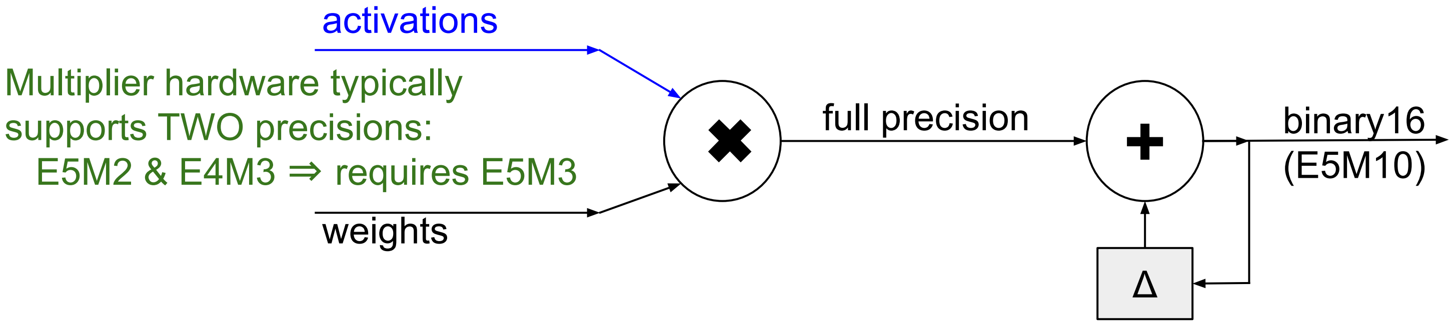


Area Results: Fused MAC (Altera FPGA ALMs)



Wgt. + Act. Inputs	Mult. Area	Adder Area	Fused MAC Area
E5M3 (E5M2 & E4M3)	34.0	188.0	222.5

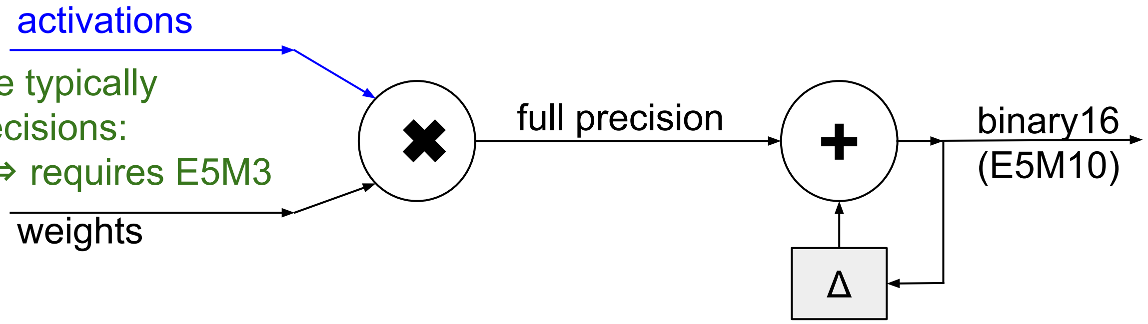
Area Results: Fused MAC (Altera FPGA ALMs)



Wgt. + Act. Inputs	Mult. Area	Adder Area	Fused MAC Area
E5M3 (E5M2 & E4M3)	34.0	188.0	222.5
E5M2	27.0	182.5	210.0

Area Results: Fused MAC (Altera FPGA ALMs)

Multiplier hardware typically supports TWO precisions:
E5M2 & E4M3 ⇒ requires E5M3



Wgt. + Act. Inputs	Mult. Area	Adder Area	Fused MAC Area
E5M3 (E5M2 & E4M3)	34.0	188.0	222.5
E5M2	27.0	182.5	210.0
E5M2B1 (supernormals)	24.5	185.0	210.0

Methodology: MPTorch for mixed-precision training

- **MPTorch** – PyTorch extension for low/mixed precision training
  - Operator-level choice for FWD and BWD compute flows
    - Floating-point, fixed-point, block-floating point
    - Parameterized precision and format
    - Round to nearest, stochastic rounding
  - CPU and GPU targets
    - Mix of C++, CUDA C, and Python
    - Accelerated by GPU tensor cores
    - Some format choices run very slowly (due to software emulation) ⇒ FPGA target soon

Training Results

Configurations				ResNet-20 (CNN)			
Name	**Weight/ Activat.**	**Gradient**	**Accumulator**	**Dyn. Loss Scaling**	**Valid. Accur.**		
1	binary32	binary32	binary32	not req.	91.72%		

Training Results

Configurations				ResNet-20 (CNN)		Llama2 (LLM)	
Name	**Weight/ Activat.**	**Gradient**	**Accumulator**	**Dyn. Loss Scaling**	**Valid. Accur.**	**Scaling (Loss/Tensor)**	**Valid. Loss**
1	binary32	binary32	binary32	not req.	91.72%	no / no	1.14736

Training Results

Configurations				ResNet-20 (CNN)		Llama2 (LLM)	
Name	**Weight/ Activat.**	**Gradient**	**Accumulator**	**Dyn. Loss Scaling**	**Valid. Accur.**	**Scaling (Loss/Tensor)**	**Valid. Loss**
1	binary32	binary32	binary32	not req.	91.72%	no / no	1.14736
2	**E4M3**	**E5M2**	**binary32**	**required**	**91.85%**	**req. / req.**	**1.14833**

Training Results

Configurations				ResNet-20 (CNN)		Llama2 (LLM)	
Name	**Weight/ Activat.**	**Gradient**	**Accumulator**	**Dyn. Loss Scaling**	**Valid. Accur.**	**Scaling (Loss/Tensor)**	**Valid. Loss**
1	binary32	binary32	binary32	not req.	91.72%	no / no	1.14736
2	**E4M3**	**E5M2**	**binary32**	**required**	**91.85%**	**req. / req.**	**1.14833**
3	E5M2	E5M2	binary16	required	91.31%	req. / req.	1.16494

* **nosub** means subnormal encodings are treated as normalized encodings (not flush-to-zero)

Training Results

Configurations				ResNet-20 (CNN)		Llama2 (LLM)	
Name	**Weight/ Activat.**	**Gradient**	**Accumulator**	**Dyn. Loss Scaling**	**Valid. Accur.**	**Scaling (Loss/Tensor)**	**Valid. Loss**
1	binary32	binary32	binary32	not req.	91.72%	no / no	1.14736
2	**E4M3**	**E5M2**	**binary32**	**required**	**91.85%**	**req. / req.**	**1.14833**
3	E5M2	E5M2	binary16	required	91.31%	req. / req.	1.16494
4	E5M2 (nosub *)	E5M2 (nosub *)	binary16	required	91.34%	req. / req.	**diverges**

Training Results

* **nosub** means subnormal encodings are treated as normalized encodings (not flush-to-zero)

Configurations				ResNet-20 (CNN)		Llama2 (LLM)	
Name	**Weight/ Activat.**	**Gradient**	**Accumulator**	**Dyn. Loss Scaling**	**Valid. Accur.**	**Scaling (Loss/Tensor)**	**Valid. Loss**
1	binary32	binary32	binary32	not req.	91.72%	no / no	1.14736
2	**E4M3**	**E5M2**	**binary32**	**required**	**91.85%**	**req. / req.**	**1.14833**
3	E5M2	E5M2	binary16	required	91.31%	req. / req.	1.16494
4	E5M2 (nosub *)	E5M2 (nosub *)	binary16	required	91.34%	req. / req.	**diverges**
5	**E5M2B1**	**E5M2B1**	binary16	required	91.61%	req. / req.	1.16629

Training Results

* **nosub** means subnormal encodings are treated as normalized encodings (not flush-to-zero)

Configurations				ResNet-20 (CNN)		Llama2 (LLM)	
Name	**Weight/ Activat.**	**Gradient**	**Accumulator**	**Dyn. Loss Scaling**	**Valid. Accur.**	**Scaling (Loss/Tensor)**	**Valid. Loss**
1	binary32	binary32	binary32	not req.	91.72%	no / no	1.14736
2	**E4M3**	**E5M2**	**binary32**	**required**	**91.85%**	**req. / req.**	**1.14833**
3	E5M2	E5M2	binary16	required	91.31%	req. / req.	1.16494
4	E5M2 (nosub *)	E5M2 (nosub *)	binary16	required	91.34%	req. / req.	**diverges**
5	**E5M2B1**	**E5M2B1**	binary16	required	91.61%	req. / req.	1.16629
6	**E5M2B4**	**E5M2B4**	binary16	**not required**	91.46%	-	-

Training Results

* **nosub** means subnormal encodings are treated as normalized encodings (not flush-to-zero)

Configurations				ResNet-20 (CNN)		Llama2 (LLM)	
Name	**Weight/ Activat.**	**Gradient**	**Accumulator**	**Dyn. Loss Scaling**	**Valid. Accur.**	**Scaling (Loss/Tensor)**	**Valid. Loss**
1	binary32	binary32	binary32	not req.	91.72%	no / no	1.14736
2	**E4M3**	**E5M2**	**binary32**	**required**	**91.85%**	**req. / req.**	**1.14833**
3	E5M2	E5M2	binary16	required	91.31%	req. / req.	1.16494
4	E5M2 (nosub *)	E5M2 (nosub *)	binary16	required	91.34%	req. / req.	**diverges**
5	**E5M2B1**	**E5M2B1**	binary16	required	91.61%	req. / req.	1.16629
6	**E5M2B4**	**E5M2B4**	binary16	**not required**	91.46%	-	-

1. extra range helpful

Training Results

* **nosub** means subnormal encodings are treated as normalized encodings (not flush-to-zero)

Configurations				ResNet-20 (CNN)		Llama2 (LLM)	
Name	**Weight/ Activat.**	**Gradient**	**Accumulator**	**Dyn. Loss Scaling**	**Valid. Accur.**	**Scaling (Loss/Tensor)**	**Valid. Loss**
1	binary32	binary32	binary32	not req.	91.72%	no / no	1.14736
2	**E4M3**	**E5M2**	**binary32**	**required**	**91.85%**	**req. / req.**	**1.14833**
3	E5M2	E5M2	binary16	required	91.31%	req. / req.	1.16494
4	E5M2 (nosub *)	E5M2 (nosub *)	binary16	required	91.34%	req. / req.	**diverges**
5	**E5M2B1**	**E5M2B1**	binary16	required	91.61%	req. / req.	1.16629
6	**E5M2B4**	**E5M2B4**	binary16	**not required**	91.46%	-	-

1. extra range helpful

2. extra range needed

Training Results

* **nosub** means subnormal encodings are treated as normalized encodings (not flush-to-zero)

Configurations				ResNet-20 (CNN)		Llama2 (LLM)	
Name	**Weight/ Activat.**	**Gradient**	**Accumulator**	**Dyn. Loss Scaling**	**Valid. Accur.**	**Scaling (Loss/Tensor)**	**Valid. Loss**
1	binary32	binary32	binary32	not req.	91.72%	no / no	1.14736
2	**E4M3**	**E5M2**	**binary32**	**required**	**91.85%**	**req. / req.**	**1.14833**
3			binary16	required	91.31%	req. / req.	1.16494
4			binary16	required	91.34%	req. / req.	**diverges**
5	**E5M2B1**	**E5M2B1**	binary16	required	91.61%	req. / req.	1.16629
6	**E5M2B4**	**E5M2B4**	binary16	**not required**	91.46%	-	-

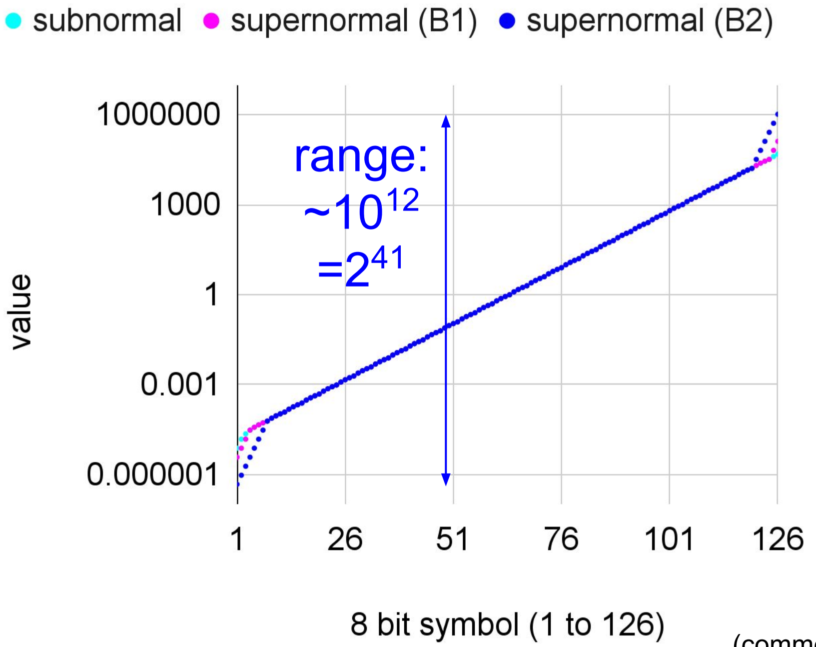
Q: can we use only 1 precision w/ supernormals?

1. extra range helpful

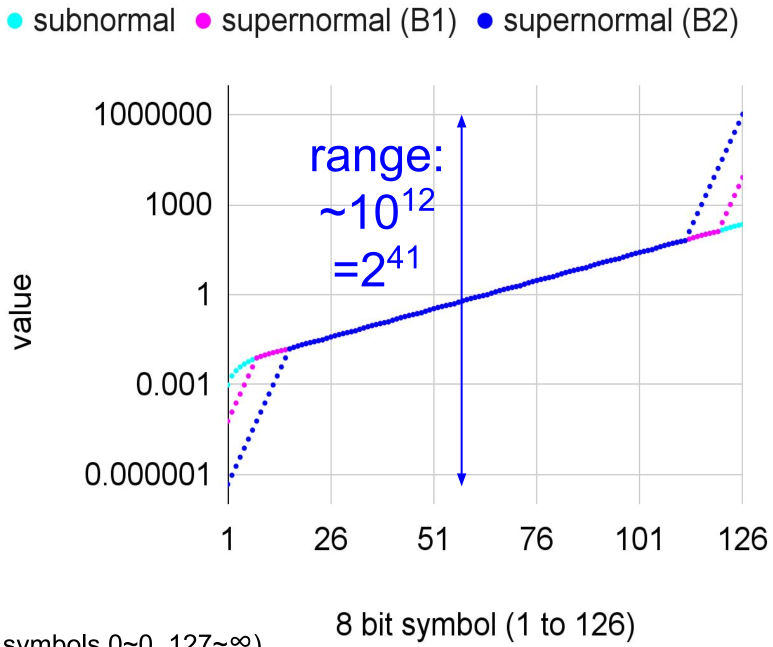
2. extra range needed

Q: can we use only 1 precision w/ supernormals?

P3 (E5M2) – range for gradients



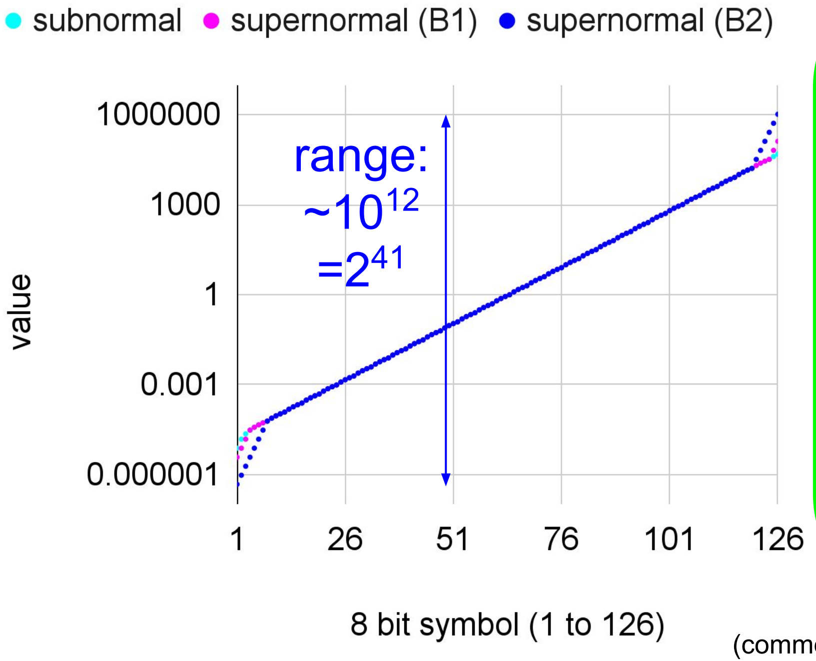
P4 (E4M3) – precision for weights



(common symbols 0~0, 127~∞)

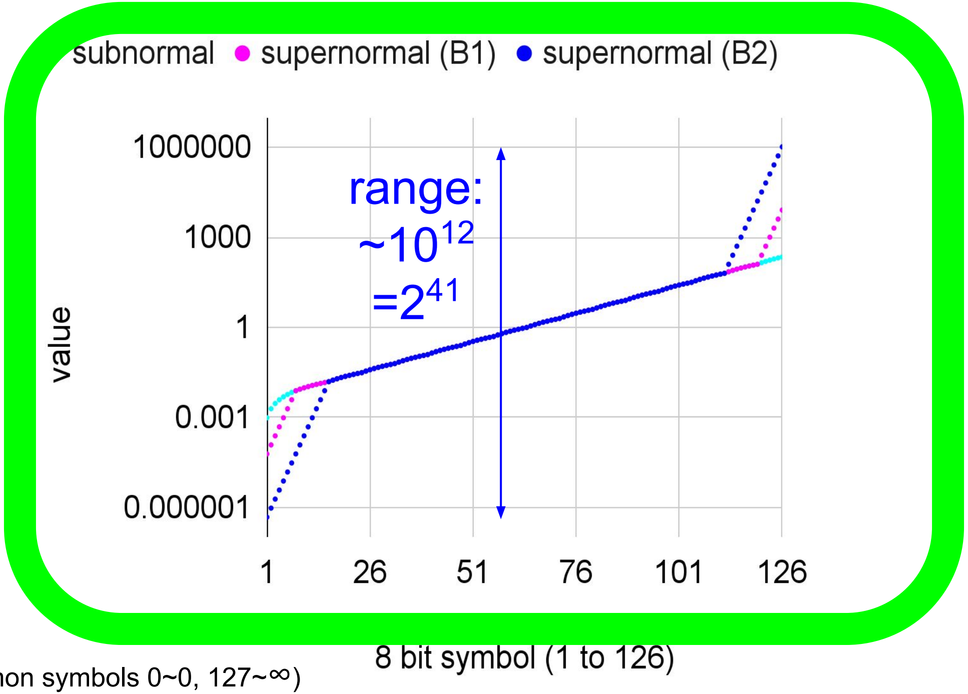
Q: can we use only 1 precision w/ supernormals?

P3 (E5M2) – range for gradients



P4 (E4M3) – precision for weights

supernormals needed to match P3's range



(common symbols 0~0, 127~∞)

Conclusions and Future Work

- Encouraging results when using SUPERnormals in 8-bit DNN training
  - Improved dynamic range extension over subnormals
    - Using extra binades could avoid dependence on tensor/loss scaling
    - Might be able to use only E4M3B2 instead of two precisions E5M2+E4M3
  - More area-efficient than subnormals
    - Better range extension too

- Considerations and Limitations
  - This work is preliminary
  - Current training relies on tensor core acceleration (binary16 accumulators only)
    - Future work with bfloat16/binary32 to test gradual overflow (much slower training)
  - Upper and lower ranges currently extended by same number of binades
    - Could specify different # binades at lower and upper ends of range

Range-extending

SUPERnormals