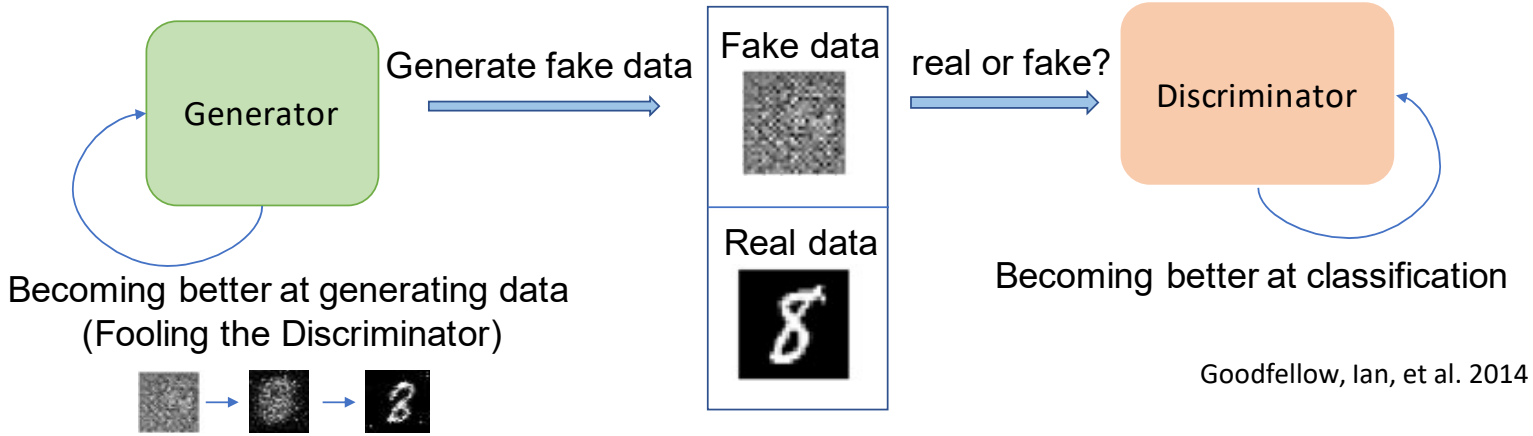
Posit™ Arithmetic for the Training and Deployment of Generative Adversarial Networks

Nhut-Minh Ho∗, Duy-Thanh Nguyen†, Himeshi De Silva∗, John L. Gustafson∗, Weng-Fai Wong∗, Ik Joon Chang†

*National University of Singapore, Singapore
†KyungHee University, Korea

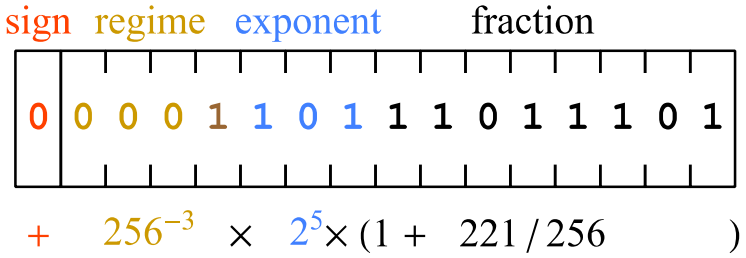
GANs and their application



Goodfellow, Ian, et al. 2014

- Generating data, style transfer, restore old photos, deepfake video...
- Training involves multiple neural networks

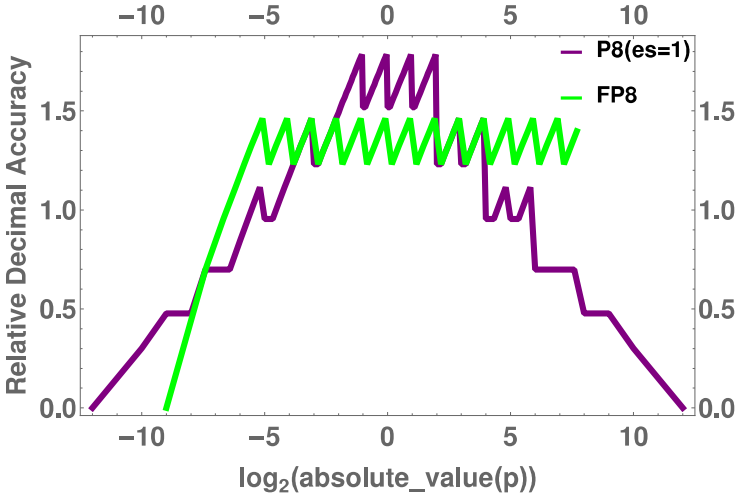
Posit and Floating point formats



- The encoding allows different field sizes
- More values are represented near 1.0

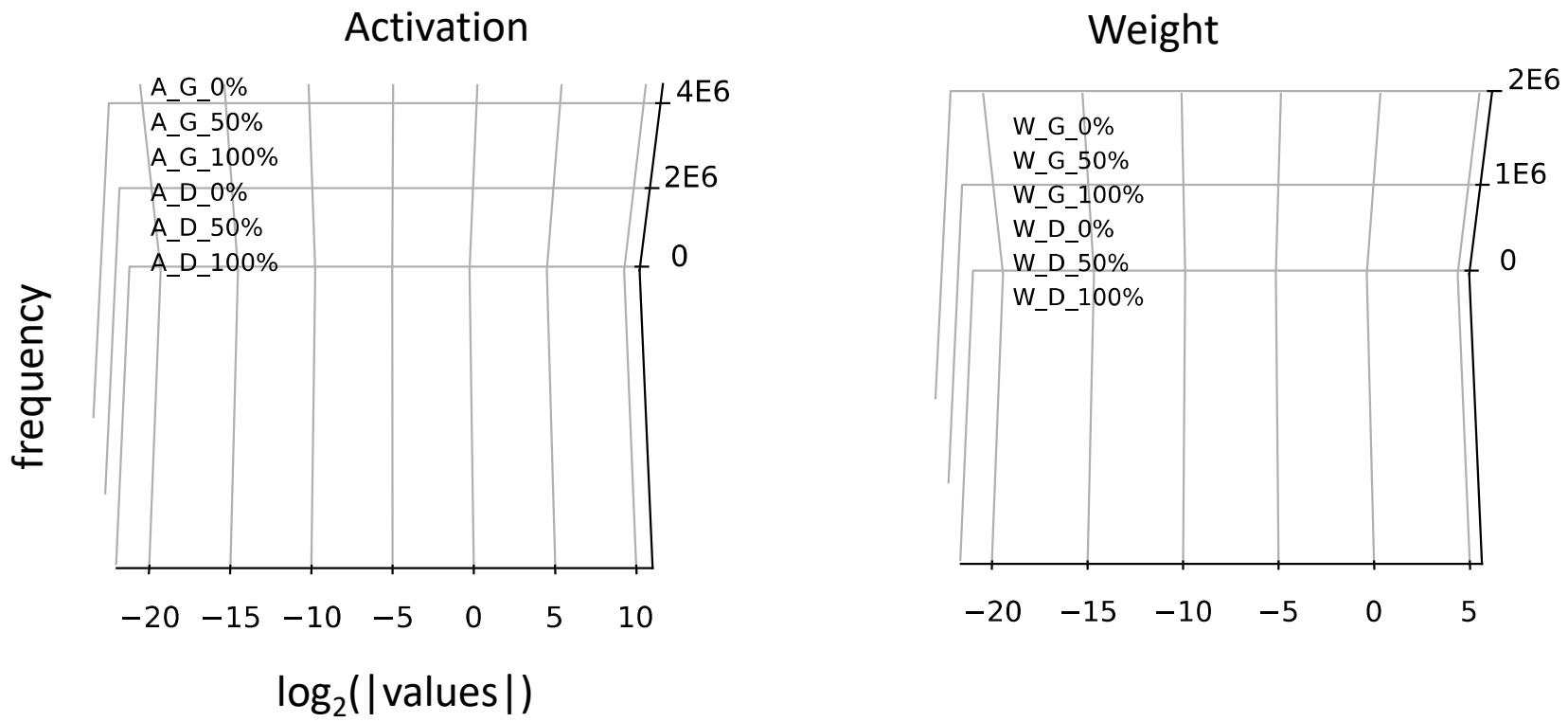
P(8,1):

- 8 bit total
- 1 bit exponent

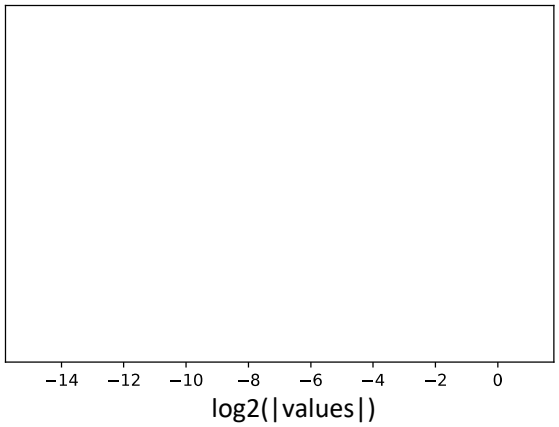


- Normal floating point: fixed exponent and fraction size
- The number of representable values at each bin in log2(value) histogram are almost the same

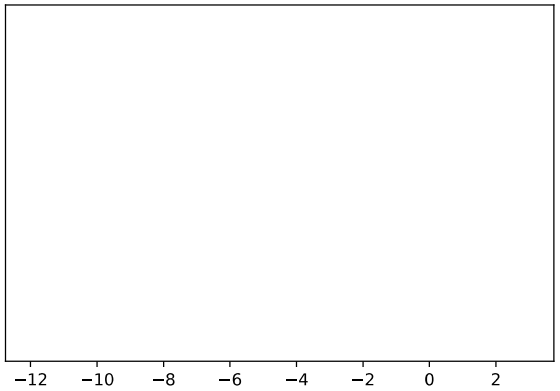
Histogram of weights and activations



Tuning the Posit format by scaling



Scale 2^5

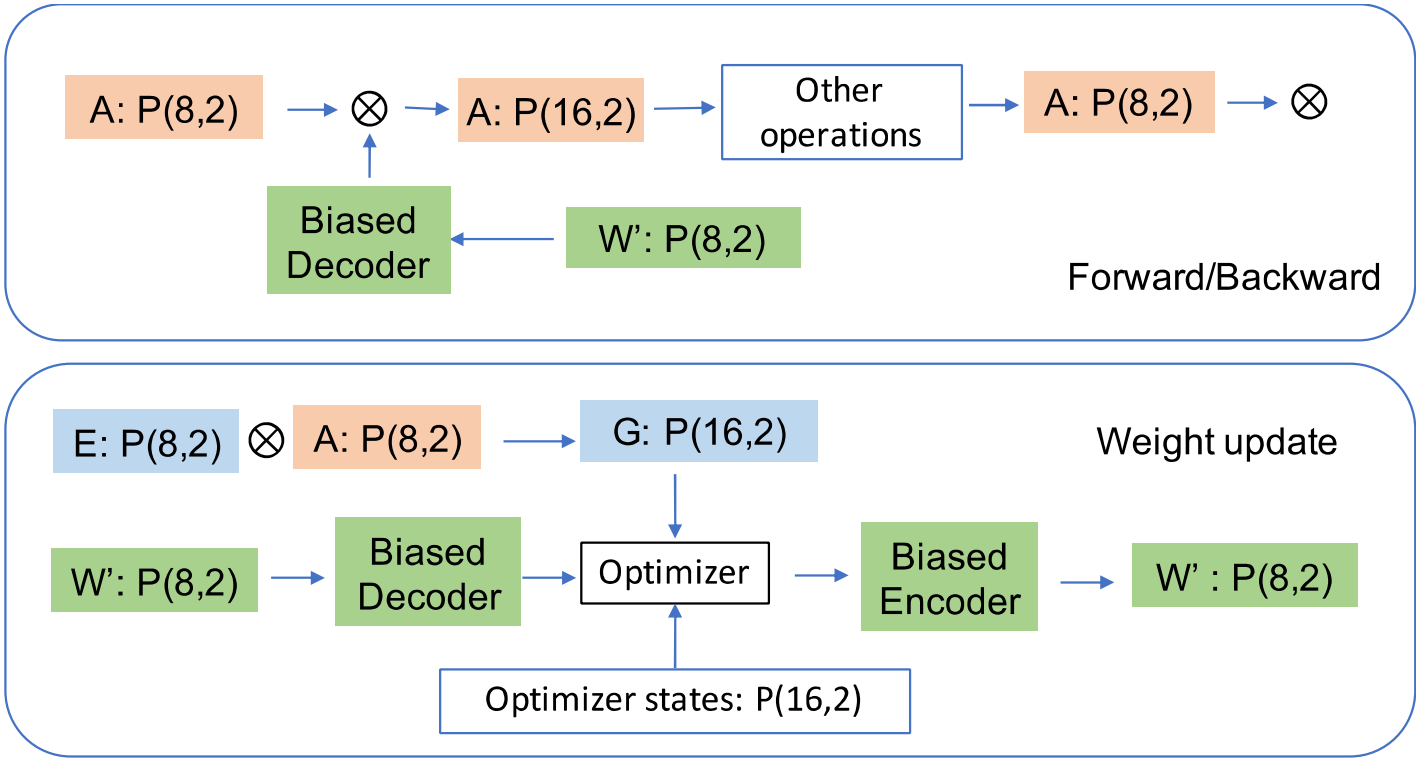


Posit : Sign | Regime | Exponent | Frac

Decode : $(-1)^s$ x $useed^k$ x 2^e x 1.frac

- Multiply by 2^N => Exponent e = e+N
- 1 add operation to scale the format

Training System with 8-bit Posit

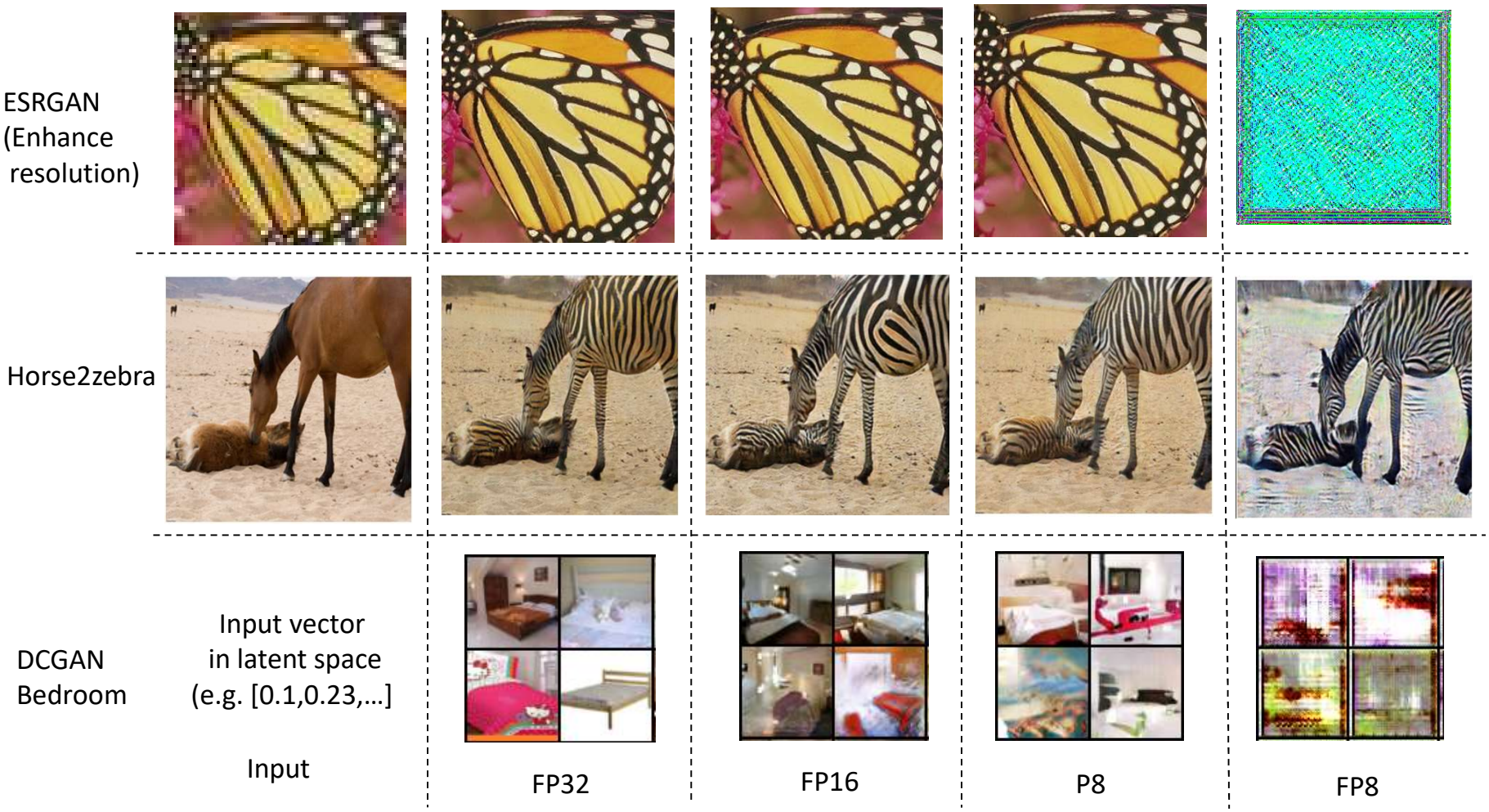


Training Results

	FP32	FP16	P8+	P8	FP8
CGAN-Mnist	67.31	64.58	70	75.41	423.28
DCGAN-Lsun	43.12	49	44.82	49.54	318.07
Horse2zebra	77.2	64.3	61.1	70.1	75.63
Summer2winter	77.6	78.52	77.12	75.67	84.01
Zebra2horse	134.2	134.5	138.36	148.49	138.13
Winter2summer	75	73.46	72.79	74.75	77.16
ESRGAN (PSNR)	24.67	24.69	24.74	24.53	4.62

Normal GAN : FID Scores lower better
ESRGAN (PSNR- dB): higher better

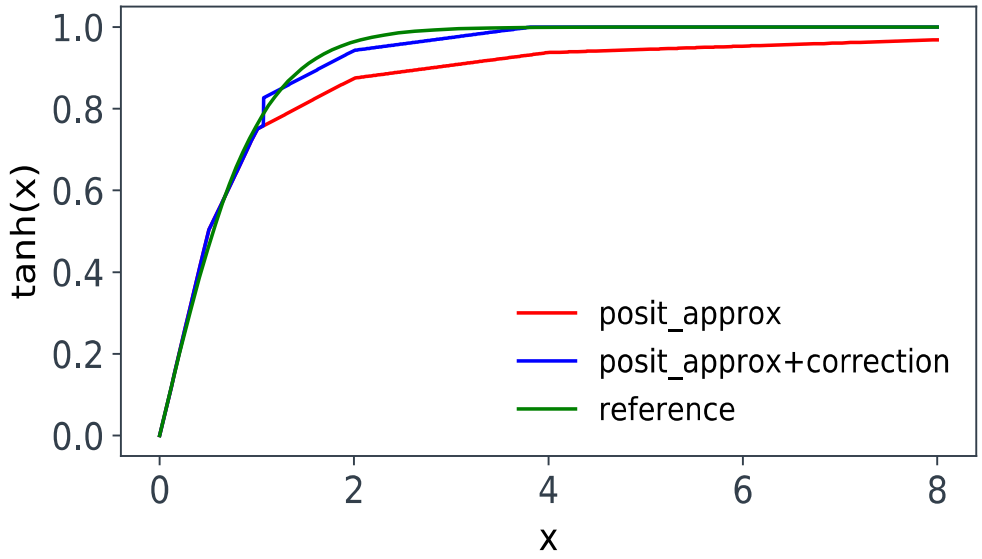
FP16: Nvidia mixed precision training APEX (FP16 and FP32)
P8+: Our method with loss scaling + exponent bias (Fig.5)
P8 : P(8,2) without loss scaling & w/o exponent bias
FP8: IBM's hybrid training float8 (4 and 5 exponents) mixed with FP(1,6,9)


ESRGAN
(Enhance
resolution)
Horse2zebra
DCGAN
Bedroom
Input vector
in latent space
(e.g. [0.1,0.23,…]
Input
FP32
FP16
P8
FP8

Tanh Approximation for Deployment

Sigmoid(x) = (x⊕0x8000)>>2 *

PositTanh(x) = 2·Sigmoid(2x)−1



$$R = PositTanh(x)$$

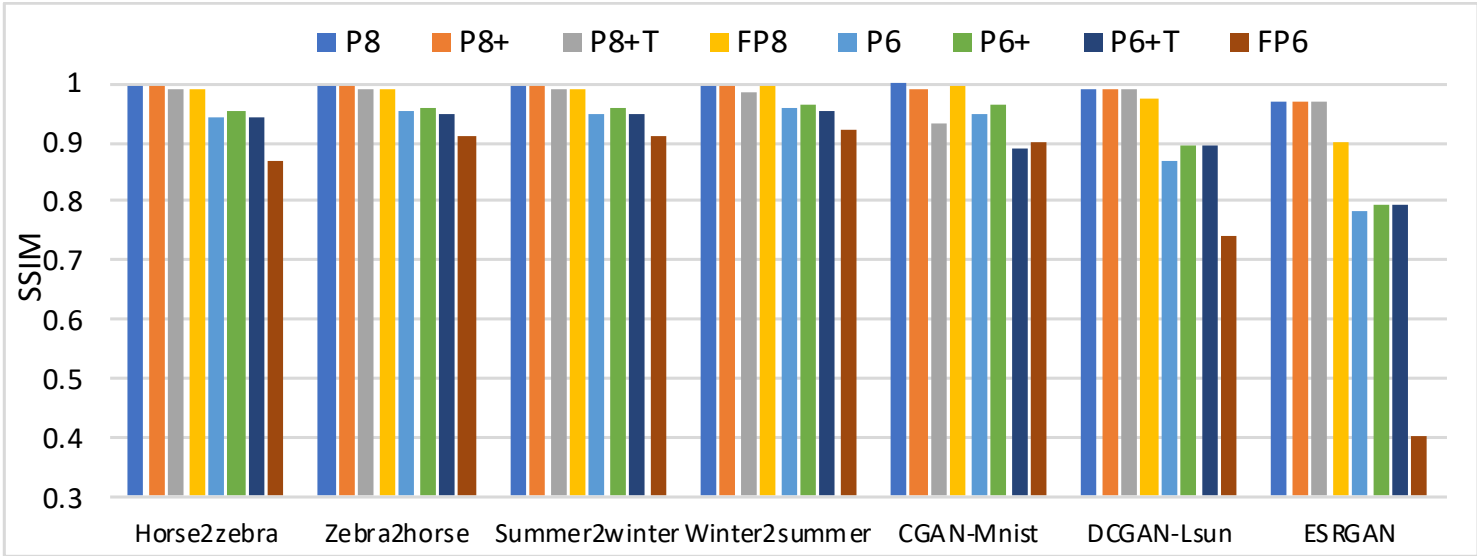
if$(R > \theta)$:
return $\min(R + \delta, 1)$

if$(R < -\theta)$:
return $\max(R - \delta, -1)$

	Max |error|	Average |error|	Norm. Energy
Tanh FP16	-	-	1.0
P_approx	0.098	0.056	0.066
P_approx + correction	0.0379	0.0086	0.087

* X in P(16,0) format

6-bit For Deployment Phase



P8 = P(8,1) for dot product + P(16,1) other layers
P6 = P(6,1) for dot product + P(16,1) other layers
FP8 = IBM hybrid FP8 for dot product + FP16 (1,6,9) other layers
FP6 = best FP6 configuration (searching for all possible variations of {exponent, mantissa}) for dot product + FP16 other layers
P8+ = P8 + exponent bias (weight scaling)
P8+T = P8 + exponent bias (weight scaling) + tanh approximation

Conclusion

- Posit could become the standard for low bitwidth training of GAN in the near future.
- 8-bit formats for Training, 6-bit formats for Deployment.
- Simple & effective bit-wise tricks for complex functions (sigmoid, tanh)
- Posit is now available in a Pytorch extension, can use on your own models (DNN, NLP transformer, GAN, …). Tutorial and Demo available at *https://github.com/minhhn2910/QPyTorch*

Thank you

Q & A